B100/I101
Problem Solving Using Computers

Loop/Repetition Statements
in VB.NET

Dr. David R. Surma
Department of Computer and Information Sciences

Adapted from Drs. Adaikkalavan, Hakimzadeh and Zhang
LOOPING (repetition)

WHILE (Condition)
Statement(s)
END WHILE

Condition

Statement(s)

False

True

...
LOOPING - examples

- while (gas tank is not full)
  - keep adding gas

- While the mixture is too liquid:
  - Add flour and mix.

- While the mixture is too solid:
  - Add milk and mix.
The Condition

- Boolean expression
  - Relational Operators
  - Boolean Operators
While Loop Example

Dim count As Integer

count = 4

While (count > 1)
    Console.WriteLine(count)
    count = count - 1
End While

Console.WriteLine(“Done.”)
**While Loop Example: counter-controlled**

```
Dim count As Integer

count = 4

While (count > 1)
    Console.WriteLine(count)
    count = count - 1
End While

Console.WriteLine("Done.")
```

**OUTPUT**

```
count

6

Done.
```
**While Loop Example: counter-controlled**

```
Dim count As Integer

count = 4

While (count > 1)
    Console.WriteLine(count)
    count = count - 1
End While

Console.WriteLine("Done.")
```

**OUTPUT**

```
count
4
```

```
OUTPUT
```

```
```
**While Loop Example: counter-controlled**

```
Dim count As Integer

count = 4

While (count > 1)        TRUE
    Console.WriteLine(count)
    count = count - 1
End While

Console.WriteLine(“Done.”)
```

**OUTPUT**

```
4
```

```
**While Loop Example: counter-controlled**

```vbnet
Dim count As Integer

count = 4
While (count > 1)
    Console.WriteLine(count)
    count = count - 1
End While

Console.WriteLine(“Done.”)
```

**OUTPUT**

```
4
```

**count**

```
4
```
While Loop Example: counter-controlled

Dim count As Integer

count = 4

While (count > 1)
    Console.WriteLine(count)
    count = count - 1
End While

Console.WriteLine(“Done.”)
While Loop Example: counter-controlled

Dim count As Integer

count = 4

While (count > 1) True
    Console.WriteLine(count)
    count = count - 1
End While

Console.WriteLine(“Done.”)

OUTPUT

count

3

4
While Loop Example: counter-controlled

```vbnet
Dim count As Integer

count = 4

While (count > 1)
    Console.WriteLine(count)
    count = count - 1
End While

Console.WriteLine(“Done.”)
```

**OUTPUT**

```
4
3
```

**count**

```
3
```
**While Loop Example: counter-controlled**

```vbnet
Dim count As Integer

count = 4

While (count > 1)
    Console.WriteLine(count)
    count = count - 1
End While

Console.WriteLine(“Done.”)
```

OUTPUT:

```
4
3
2
```

**Count**

- 2

- 4

- 3
**While Loop Example: counter-controlled**

```vbnet
Dim count As Integer

count = 4

While (count > 1) ' TRUE
    Console.WriteLine(count)
    count = count - 1
End While

Console.WriteLine(“Done.”)
```

**OUTPUT**

4
3
2
**While Loop Example: counter-controlled**

```
Dim count As Integer

count = 4

While (count > 1)
    Console.WriteLine(count)
    count = count - 1
End While

Console.WriteLine("Done.")
```

**OUTPUT**

```
2
4
3
2
```
**While Loop Example: counter-controlled**

```vbnet
Dim count As Integer

count = 4

While (count > 1)
    Console.WriteLine(count)
    count = count - 1
End While

Console.WriteLine(“Done.”)
```

**OUTPUT**

```
1
2
3
4
```

---

**count**

1
While Loop Example: counter-controlled

```vbnet
Dim count As Integer
count = 4
While (count > 1)  FALSE
    Console.WriteLine(count)
    count = count - 1
End While
Console.WriteLine(“Done.”)
```

OUTPUT

```
4
3
2
```
**While Loop Example: counter-controlled**

Dim count As Integer

count = 4

While (count > 1)
    Console.WriteLine(count)
    count = count - 1
End While

Console.WriteLine(“Done.”)

**OUTPUT**

1

4 3 2

Done.
## Compound Assignment Operators

<table>
<thead>
<tr>
<th>Compound assignment operator</th>
<th>Sample expression</th>
<th>Explanation</th>
<th>Assigns</th>
</tr>
</thead>
<tbody>
<tr>
<td>+=</td>
<td>c += 7</td>
<td>c = c + 7</td>
<td>11 to c</td>
</tr>
<tr>
<td>-=</td>
<td>c -= 3</td>
<td>c = c - 3</td>
<td>1 to c</td>
</tr>
<tr>
<td>*=</td>
<td>c *= 4</td>
<td>c = c * 4</td>
<td>16 to c</td>
</tr>
<tr>
<td>/=</td>
<td>c /= 2</td>
<td>c = c / 2</td>
<td>2 to c</td>
</tr>
<tr>
<td>=</td>
<td>c = 3</td>
<td>c = c \ 3</td>
<td>1 to c</td>
</tr>
<tr>
<td>^=</td>
<td>c ^= 2</td>
<td>c = c ^ 2</td>
<td>16 to c</td>
</tr>
<tr>
<td>&amp;=</td>
<td>d &amp;= &quot;Hello&quot;</td>
<td>d = d &amp; &quot;Hello&quot;</td>
<td>&quot;Hello&quot; to d</td>
</tr>
</tbody>
</table>

- Assume: \(c = 4\), \(d = "He"\)

- The \(=, +, -\), \(*, /, \backslash, ^, \\&\) operators are always applied last in an expression.

- When a compound assignment is evaluated, the appropriate operator is applied to the lvalue’s original value and the value to the operator’s right, then the resulting value is assigned to the lvalue on the left.
Two Types of *while* Loops

**Counter-controlled loops**

repeat a specified number of times (as our previous example!)

**Event-controlled loops**

some condition within the loop body changes and this causes the repeating to stop
Event-controlled **While Loops**

- **Sentinel-controlled**
  keep processing data until a special value which is not a possible data value is entered to indicate that processing should stop

- **Flag-controlled**
  keep processing data until the value of a flag changes in the loop body

- **others**
Sentinel-controlled Loops

- Sentinel-controlled
  keep processing data until a special value which is not a possible data value is entered to indicate that processing should stop

- Example:
  - Keeping reading integer numbers from keyboard until reading 0. Find and output the largest one.
Sentinel-controlled Loop Example:

- Problem example: Find largest
- Problem Definition
  - The user is asked to enter a few positive numbers
  - If the user enters a zero, the program should stop
  - After the program stops, it should display the largest numbers entered by the user
Review: Find Largest

- **Design**
  - Get a number from the user
  - Set the Largest to the above number
  - Start a loop which only stops when the user enters a zero
  - Inside the loop, check to see if the number if larger than the largest. If so, replace the largest with the new number
  - Inside the loop, get a new number from the user
  - After the loop, output the Largest
Sentinel-controlled Loop Example: Find Largest

Start

Input number

largest = number

While number <> 0

If number > largest

largest = number

Input number

Output largest

End
**Sentinel-controlled Loop Example: Find Largest**

```vbnet
Dim number, largest As Integer

Console.WriteLine("Enter a series of integer numbers (0 to stop)")
number = CInt(Console.ReadLine())
largest = number

While (number <> 0)
    If (number > largest) Then
        largest = number
    End If
    number = CInt(Console.ReadLine())
End While

Console.WriteLine("The largest was " & largest)
```
Flag-controlled Loops

**Flag-controlled**

keep processing data until the value of a flag changes in the loop body

**Example:**

Read blood pressures until a dangerously high BP (200 or more) is read. Then output the number of good readings.
Flag-controlled Loop Example:

Dim countGoodReadings As Integer = 0
Dim thisBP As Integer
Dim isSafe As Boolean = True

While (isSafe)
    Console.Write("Please enter a BP value: ")
    thisBP = CInt(Console.ReadLine())
    If (thisBP >= 200) Then
        isSafe = False
    Else
        countGoodReadings += 1
    End If
End While

Console.WriteLine("The number of good readings is " & countGoodReadings)
The semantics of the DO loop is:

‘Do this work \textit{while} a condition is true’

Or

‘Do this work \textit{until} a condition becomes true’
## Do Loop

### Syntax:

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Do While** condition  
statements  
Loop | 'condition is checked at the beginning |
| **Do Until** condition  
statements  
Loop | 'condition is checked at the beginning |
| **Do**  
statements  
**Loop While** condition | 'condition is checked at the end |
| **Do**  
statements  
**Loop Until** condition | 'condition is checked at the end |
Do While… Loop

Do While (Condition)
Statement(s)
Loop

Do While… Loop is equivalent to While… End While Loop
' Do While
Dim counter As Integer = 0

Console.WriteLine("Let's try counting up")

Do While (counter < 3)
    Console.WriteLine("Counter = {0}", counter)
    counter += 1
Loop
Exercises

- Identify and correct the error(s):
  - The following code should total the values from 1 to 50.

```vbnet
Dim value as Integer = 50
Dim sum as Integer = 0

Do While value >= 0
    sum += value
Loop
```
Exercises

- Identify and correct the error(s):
  - The following code should display the squares of 1 to 10 in resultListBox

```vba
Dim number as Integer = 1

Do While number < 10
    resultListBox.Items.Add (number ^ 2)
End While
```
Exercises

- Identify and correct the error(s):
  - The following code should display the integers from 888 to 1000 in resultListBox.

```vbnet
Dim value as Integer = 888

Do While value < 1000
    value -= 1
Loop
```
Don’t Forget these Necessary Components of a loop:

- A loop condition should be **true** at one point! Otherwise we will *never enter* the loop.

- There should be a way to exit a loop. In other words, the loop condition must become **false** at some point. Otherwise we will *never stop* the loop!
What is the result of the following loops:

```
Dim X As Integer
X = 0

Do While (X > 0)  'a loop that never executes
    Console.WriteLine("hello")
Loop
```
What is the result of the following loops:

```vbnet
Dim X As Integer
X = 10

Do While (X > 0) 'a loop that never stops
    Console.WriteLine("hello")
Loop
```
Do Until... Loop

Do Until (Condition)
Statement(s)
Loop

- Condition
- Statement
- True
- False
' Do Until
Dim counter As Integer

Console.WriteLine("Now lets try counting down.")
counter = 10

Do Until counter = 0
    Console.WriteLine("Counter = {0}", counter)
counter -= 1
Loop
What is the result of the following loops:

Dim X As Integer
X = 0

Do Until (X = 0)  'a loop that never executes
    Console.WriteLine("hello")
Loop
Placing the loop condition at the end:

- **Examples:**

  ```csharp
  counter = 0
  Do
      Console.WriteLine("I will print at least once. ")
  Loop While counter > 5
  Do While counter > 5
      Console.WriteLine("I will never print. ")
  Loop
  ```

- **Note:**
  - In a do-while or do-until loop, if the condition is checked at the **end**, the body of the loop is executed **one or more times**.
Do... Loop While

Do

Statement(s)

Loop While (Condition)

Statement → Condition

Condition → False

False → ...
Placing the loop condition at the end:

- **Examples:**

  ```
  counter = 0
  Do
    Console.WriteLine("I will never stop")
  Loop Until counter > 5
  
  Do Until counter > 5
    Console.WriteLine("I will never stop ")
  Loop
  ```

**Note:**
- In a do-while or do-until loop, if the condition is checked at the end, the body of the loop is executed **one or more times**.
Placing the loop condition at the end:

Examples:

counter = 10
Do
    Console.WriteLine("I will print at least once")
Loop Until counter > 5
Do Until counter > 5
    Console.WriteLine("I will never print ")
Loop

Note:

• In a do-while or do-until loop, if the condition is checked at the end, the body of the loop is executed one or more times.
Do... Loop Until

Do

Statement(s)

Loop Until (Condition)

Statement

Condition

True

False
Summary of Do Loops

- **Do While... Loop**
  - Condition → False -> Statement → True → Condition
  - True in the flowchart implies the loop continues.

- **Do Until... Loop**
  - Condition → True -> Statement → False → Condition
  - False in the flowchart implies the loop continues.

- **Do... Loop While**
  - Statement → False → Condition → True → Statement
  - True in the flowchart implies the loop continues.

- **Do... Loop Until**
  - Statement → True → Condition → False → Statement
  - False in the flowchart implies the loop continues.
Exercises

What is the output of the following?

```vbnet
Dim x as Integer = 10
Do While (x > 0)
    Console.WriteLine (x)
    x = x - 3
Loop
```

Answer:

10
7
4
1

```vbnet
Dim x as Integer = -10
Do While (x > 0)
    Console.WriteLine (x)
    x = x - 3
Loop
```

Answer:
Exercises

What is the output of the following?

```
Dim x as Integer = -10
Do
    Console.WriteLine (x)
    x = x -3
Loop While (x > 0)
```

Answer:
-10

```
Dim x as Integer = -10
Do
    Console.WriteLine (x)
    x = x -3
Loop Until (x > 0)
```

Answer:
-10
-13
-16
-19
-22
-25
For loops:

- For Loop is used when the programmer know exactly how many times a loop should execute.

- Syntax:

```plaintext
For loopVariable = startValue to endValue [step stepSize]
    statements
Next [loopVariable]
```

[ ... ] means the content included is optional
For loops:

Module Module1
    Sub Main()
        Dim count As Integer

        For count = 0 To 10 Step 2
            Console.WriteLine("hello: {0}", count)
        Next count

        Console.ReadLine() ' pause
    End Sub
End Module
For loops:

Module Module1
Sub Main()
    Dim LoopCounter As Integer

    For LoopCounter = 0 To 10 Step 3
        Console.WriteLine("hello: {0}", LoopCounter)
    Next LoopCounter

    Console.ReadLine() ' pause
End Sub
End Module
For loops:

Module Module1
    Sub Main()
        Dim LoopCounter As Integer

        For LoopCounter = 10 To 0 Step -2
            Console.WriteLine("hello: {0}", LoopCounter)
        Next LoopCounter

        Console.ReadLine() ' pause
    End Sub
End Module
Module Module1
Sub Main()

    Dim LoopCounter As Double

    For LoopCounter = 0.5 To 9.5 Step 0.5
        Console.WriteLine("hello: {0}", LoopCounter)
    Next LoopCounter

    Console.ReadLine() ' pause

End Sub
End Module
Module Module1
Sub Main()
    Dim count As Integer

    For count = 0 To 10
        Console.WriteLine("hello: {0}" , count)
    Next count

    Console.ReadLine() ' pause
End Sub
End Module
Module Module1
    Sub Main()
        Dim count As Integer

        For count = 10 To 0
            Console.WriteLine("hello: {0}", count)
        Next count

        Console.ReadLine() ' pause
    End Sub
End Module