Chapter 6 – Repetition

6.1 Do Loops
6.2 For...Next Loops
6.3 List Boxes and Loops
6.1 Do Loops

- Pretest Form of a Do Loop
- Posttest Form of a Do Loop
Do Loops

• 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.
Pretest Do Loop

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.
Pseudocode and Flow Chart

Do While condition is true
Process step(s)
Loop
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
Example: Repeat Request as Long as Response in Incorrect

Dim passWord As String = ""
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.
Example 3: Sentinel-Controlled Loop

```vba
Dim num As Double = 0
Dim prompt As String = "Enter a nonnegative number. Enter -1 " & 
"to terminate entering numbers."
num = CDb1(InputBox(prompt))
Do While num <> -1 '-1 is sentinel value
. 
. 
    num = CDb1(InputBox(prompt))
Loop
```
Posttest Do Loop

Do

\textit{statement(s)}

Loop Until \textit{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 Until statement are executed.
Example: Repeat Request Until Proper Response

Dim passWord As String = ""
Do
    passWord = InputBox("What is the password?")
    passWord = passWord.ToUpper
Loop Until passWord = "SHAZAM"
Pseudocode and Flowchart

Do
statement(s)
Loop Until condition is true

Is the condition true?

No

Yes

Execute statements that follow the loop

Execute statements within the loop
Example 5: Form

txtAmount

txtWhen
Private Sub btnCalculate_Click(...) Handles _
    btnCalculate.Click

Dim balance As Double, numYears As Integer
balance = CDBl(txtAmount.Text)
Do While balance < 1000000
    balance += 0.06 * balance
    numYears += 1
Loop

    txtWhen.Text = "In " & numYears &
        " years you will have a million dollars."

End Sub
Example 5: Output

Amount deposited: 100000

Calculate Years to Become a Millionaire

In 40 years you will have a million dollars.
Comments

• Be careful to avoid infinite loops – loops that never end.
• Visual Basic allows for the use of either the While keyword or the Until keyword at the top or the bottom of a loop.
6.2 For…Next Loops

- General Form of a For…Next Loop
- Step Keyword
- Nested For…Next Loops
- Local Type Inference
For...Next Loops

- Used when we know how many times we want the loop to execute
- A counter controlled loop
For…Next Loop Syntax

- **Initial value**
- **Counter variable**
- **For** 
  - **i As numDataType = m To n**
  - **statement(s)**
  - **Next**
- **Terminating value**
- **Body**
Sample

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

The loop counter variable, i, is

- initialized to 1
- tested against the stop value, 5
- incremented by 1 at the Next statement
Similar Do While Loop

Dim i As Integer = 1
Do While i <= 5
    lstTable.Items.Add(i & " " & i ^ 2)
    i += 1
Loop
Example 1: Output

![Population Chart]

- **2012**: 300,000
- **2013**: 309,000
- **2014**: 318,270
- **2015**: 327,818
- **2016**: 337,653
Example 1: Code

```vbnet
Dim pop As Double = 300000
For yr As Integer = 2012 To 2016
    lstTable.Items.Add(yr & "     " & pop.ToString("N0")
    pop += 0.03 * pop
Next
```
Step Keyword

- Normally after each pass the value of the counter variable increases by 1.
- If `Step s` is appended to the For statement, the value of `s` will be added to the counter variable after each pass.
- If the value of `s` is a negative number, the value of the counter variable will decrease after each pass.
Example with Negative Step Value

For j As Integer = 10 To 1 Step -1
    lstBox.Items.Add(j)
Next
lstBox.Items.Add("Blastoff")
Example: Nested For...Next Loops

For i As Integer = 65 To 70
    For j As Integer = 1 To 25
        lstBox.Items.Add(Chr(i) & j)
    Next
Next

Output:  A1
         A2
         A3
         :
For and Next Pairs

- For and Next statements must be paired.
- If one is missing, the automatic syntax checker will complain with a wavy blue underline and a message such as "A ‘For’ must be paired with a ‘Next’."
Start, Stop, and Step values

• Consider a loop beginning with

  \[
  \text{For } i \text{ As Integer } = m \text{ To } n \text{ Step } s
  \]

• 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.
Altering the Counter Variable

- The value of the counter 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.
Non-Integer Step Values

• Can lead to rounding errors with the result that the loop is not executed the intended number of times.

• We will only use Integers for all values in the header.
6.3 List Boxes and Loops

• Some Properties, Methods, and Events of List Boxes
• List Boxes Populated with Strings
• List Boxes Populated with Numbers
• Searching an Ordered List
List Box Properties

- The total number of items in a list box is
  \[ \text{lstBox.Items.Count} \]

- **Note:** Each item in lstBox is identified by an index number from 0 to lstBox.Items.Count – 1

- The index number of the currently highlighted item is given by:
  \[ \text{lstBox.SelectedIndex} \]
More List Box Properties

- `lstBox.Items()` is the list of items in the list box.
- The value of the item with an index of \( n \) is:
  \[
  \text{lstBox.Items}(n)
  \]
- The data type of the elements in the `lstBox.Items()` array is Object. To display the first element of `lstBox.Items` in a text box:
  \[
  \text{txtBox.Text} = \text{CStr(lstBox.Items}(0))
  \]
Currently Highlighted Item in a List Box

The value of the currently highlighted item as a string can be obtained as

```c
lstBox.Text
```
The Sorted Property

- Items can be placed into the list at design time or run time
- The **Sorted** property causes items in the list to be ordered automatically as strings
- Caution: Numbers in a sorted list box will not necessarily be in increasing numerical order
Example 1: Form

IstStates is filled at design time with the names of the U.S. states in the order they joined the union.
Example 1: Code

Private Sub btnDisplay_Click(...) Handles _
    btnDisplay.Click
    'Display last ten states to join the union
    Dim n As Integer = lstStates.Items.Count
    For i As Integer = (n - 1) To (n - 10) Step -1
        lstLastTen.Items.Add(lstStates.Items(i))
    Next
End Sub
Example 1: Output
Example: Form

IstAges is filled at design time with the ages of the U.S. presidents when taking office.
Private Sub btnCalculate_Click(...) Handles _
btnCalculate.Click

'Calculate average age of presidents when assuming office
Dim n As Integer = lstAges.Items.Count
Dim sum As Double = 0
For i As Integer = 0 To (n - 1)
    sum += CDbl(lstAges.Items(i))
Next
txtAvgAge.Text = (sum / n).ToString("N")
End Sub
Example: Output

![Image of a software interface showing a list of ages and a button to calculate the average age, with the result 54.84]
Searching an Ordered List of Strings

- A search often can be terminated earlier if the list is sorted.
- This is particularly the case when the sought-after item is not in the list. The search can be terminated when an item is reached that follows the sought-after item alphabetically.
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**.
- Used in a loop to provides information to be used after loop terminates. Or, allows for the early termination of the loop.
Searching an Unsorted List

- A **flag** is used to indicate whether or not the sought-after item has been found.
- The flag variable is initially set to False and then set to True if and when the item is found.
More About Flags

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

\[
\text{If } \text{flagVar} = \text{True } \text{Then}
\]

and

\[
\text{If } \text{flagVar} = \text{False } \text{Then}
\]

can be replaced by

\[
\text{If } \text{flagVar } \text{Then}
\]

and

\[
\text{If Not } \text{flagVar } \text{Then}
\]
More About 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
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.