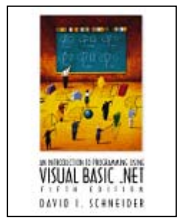
Chapter 5: Decisions

- Relational & Logical Operators (5.1)
- If Blocks (5.2)
- Select Case Blocks (5.3)



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

Relational and Logical Operators (5.1)

- *Condition* is an expression involving relational or logical operators
- Result of the condition is Boolean – that is, True or False

Relational and Logical Operators (5.1) (cont.)

- Relational Operators in VB.Net

  < less than
  <=less than or equal to
  > greater than
  >=greater than or equal to
  = equal to
  <>not equal to

Relational and Logical Operators (5.1) (cont.)

- Example

When a = 3, b = 4

(a + b) < 2 * a

3 + 4 = 7 2 * 3 = 6

Relational and Logical Operators (5.1) (cont.)

- Another Example

a = 4 b = 3 c = "hello" d = "bye"

(c.Length – b) = (a / 2)

5 – 3 = 2

4 / 2 = 2

Relational and Logical Operators (5.1) (cont.)

- Relational Operator Notes
  - Relational operators are binary – they require an operand on both sides of the operator
  - Result of a relational expression will always be Boolean
  - They are evaluated from left to right with no order of operations

Relational and Logical Operators (5.1) (cont.)

- Logical Operators
  - Used for joining Boolean expressions
  - *Not* – makes a False condition True and vice versa
  - *And* – will yield a True if and only if both expressions are True
  - *Or* – will yield a True if one or the other or both expressions are True

Relational and Logical Operators (5.1) (cont.)

- Example

To test if n falls between 2 and 5:

(2 < n) And (n < 5)

A complete relational expression must be on either side of the logical operators And and Or.

Relational and Logical Operators (5.1) (cont.)

- Syntax error

The following is NOT a valid way to test if n falls between 2 and 5:

(2 < n < 5)

Relational and Logical Operators (5.1) (cont.)

- Example 5.3

n = 4, *answ* = "Y" Are the following conditions true or false?

Not (n < 6)
(answ = "Y") Or (answ = "y")
(answ = "Y") And (answ = "y")
Not (answ = "y")

Relational and Logical Operators (5.1) (cont.)

- Order of Operations
  - The order of operations for evaluating Boolean expressions is:
    1. Arithmetic operators
    2. Relational operators
    3. Logical operators

Relational and Logical Operators (5.1) (cont.)

- Arithmetic Order of Operations
  1. Parenthesis
  2. Exponentiation
  3. Division and multiplication
  4. Addition and subtraction

Relational and Logical Operators (5.1) (cont.)

- Relational Order of Operations
  - They all have the same precedence

Relational and Logical Operators (5.1) (cont.)

- Logical Order of Operations

1. Not
2. And
3. Or

Relational and Logical Operators (5.1) (cont.)

- Common Error in Boolean Expressions
  - A common error is to replace the condition *Not (2 < 3)* by the condition (2 > 3)
  - The correct replacement is *(2 >= 3)*
  - Because >= is the opposite of <, just as <= is the opposite of >

If Block (5.2)

- The program will take a course of action based on whether a condition is true.

```
If condition Then
  action1
Else
  action2
End If
```

Will be executed if condition is true

Will be executed if condition is false

If Block (5.2) (cont.)

- Another example of If block

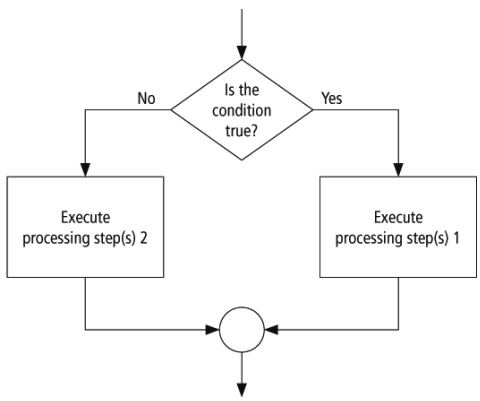
```
If condition Then
   action1
End If
Statement2
Statement3
```

Regardless of whether the condition in the If statement is True or False, these statements will be executed

If Block (5.2) (cont.)

- Pseudocode and flowchart for an If block

```
If condition is true Then
  Processing step(s) 1
Else
  Processing step(s) 2
End If
```



If Block (5.2) (cont.)

- Example 1

```
Private Sub btnFindLarger_Click(...) _
    Handles btnFindLarger.Click
  Dim num1, num2, largerNum As Double
  num1 = CDbl(txtFirstNum.Text)
  num2 = CDbl(txtSecondNum.Text)
  If num1 > num2 Then
    largerNum = num1
  Else
    largerNum = num2
  End If
  txtResult.Text = "The larger number is “ &
  largerNum
End Sub
```

If Block (5.2) (cont.)

- Example 2

```
If costs = revenue Then
  txtResult.Text = "Break even"
Else
  If costs < revenue Then
    profit = revenue - costs
    txtResult.Text = "Profit is " & _
                     FormatCurrency(profit)
  Else
    loss = costs - revenue
    txtResult.Text = "Loss is " & _
                     FormatCurrency(loss)
  End If
End If
```

If Block (5.2) (cont.)

- Example 3

```
Private Sub btnEvaluate_Click(...) _
    Handles btnEvaluate.Click
  Dim answer As Double
  answer = CDbl(txtAnswer.Text)
  If (answer >= 0.5) And (answer <= 1) Then
    txtSolution.Text = "Good, "
  Else
    txtSolution.Text = "No, "
  End If
  txtSolution.Text &= "it holds about 3/4 of" _
                    & " a gallon."
End Sub
```

If Block (5.2) (cont.)

- Example 4

```
Private Sub btnDisplay_Click(...) Handles btnDisplay.Click
  Dim message As String
  message = "Skittles is an old form of bowling in " _
   & "which a wooden disk is used to knock down nine" _
   & " pins arranged in a square."
  If txtAnswer.Text.ToUpper = "N" Then
    MsgBox(message, , "")
  End If
  txtQuote.Text = "Life ain't all beer and skittles."
End Sub
```

If Block (5.2) (cont.)

- ElseIf clause

```
If condition1 Then
  action1
ElseIf condition2 Then
  action2
ElseIf condition3 Then
  action3
Else
  action4
End If
```

If Block (5.2) (cont.)

- Example 5

```
Private Sub btnFindLarger_Click(...) _
    Handles btnFindLarger.Click
  Dim num1, num2 As Double
  num1 = CDbl(txtFirstNum.Text)
  num2 = CDbl(txtSecondNum.Text)
  If (num1 > num2) Then
    txtResult.Text = "Larger number is " & num1
  ElseIf (num2 > num1) Then
    txtResult.Text = "Larger number is " & num2
  Else
    txtResult.Text = "The two are equal."
  End If
End Sub
```

If Block (5.2) (cont.)

- Example 6

```
Function CalculateFICA(ByVal ytdEarnings As Double, _
                       ByVal curEarnings As Double) _
                                         As Double
  Dim socialSecurityBenTax, medicareTax As Double
  If (ytdEarnings + curEarnings) <= 84900 Then
    socialSecurityBenTax = 0.062 * curEarnings
  ElseIf ytdEarnings < 84900 Then
    socialSecurityBenTax = 0.062 * (84900 - ytdEarnings)
  End If
  medicareTax = 0.0145 * curEarnings
  Return socialSecurityBenTax + medicareTax
End Function
```

If Block (5.2) (cont.)

- Simplified Nested If Statement

```
If cond1 Then
  If cond2 Then
    action
  End If
End If
```

```
If cond1 And cond2 Then
    action
  End If
```

Nested If

Less Confusing

If Block (5.2) (cont.)

- Comments
  - When one If block is contained inside another If block, the structure is referred to as nested If blocks.
  - Care should be taken to make If blocks easy to understand.

If Block (5.2) (cont.)

- More Comments
  - Some programs call for selecting among many possibilities. Although such tasks can be accomplished with complicated nested If blocks, the Select Case block (discussed in the next section) is often a better alternative.

Select Case blocks (5.3)

- A decision-making structure that simplifies choosing among several actions.
- Avoids complex nested If constructs.
- If blocks make decisions based on the truth value of a condition; Select Case choices are determined by the value of an expression called a selector.

Select Case blocks (5.3) (cont.)

- Select Case Terminology
  - Each of the possible actions is preceded by a clause of the form

    `Case valueList`

  - where valueList itemizes the values of the selector for which the action should be taken.

Select Case blocks (5.3) (cont.)

- Example 1

```
Private Sub btnEvaluate_Click(...) _
    Handles btnEvaluate.Click
  Dim position As Integer 'selector
  position = CInt(txtPosition.Text)
  Select Case position
    Case 1
      txtOutcome.Text = "Win"
    Case 2
      txtOutcome.Text = "Place"
    Case 3
      txtOutcome.Text = "Show"
    Case 4, 5
      txtOutcome.Text = "You almost placed in the
  money."
    Case Else
      txtOutcome.Text = "Out of the money."
  End Select
End Sub
```

Selector

Value Lists

Select Case blocks (5.3) (cont.)

- Example 2

```
Private Sub btnDescribe_Click(...)
    Handles btnDescribe.Click
  Dim position As Integer
  position = CInt(txtPosition.Text)
  Select Case position
    Case 1 To 3
      txtOutcome.Text = "In the money."
    Case Is >= 4
      txtOutcome.Text = "Not in the money."
  End Select
End Sub
```

Select Case blocks (5.3) (cont.)

- Select Case Syntax

The general form of the Select Case block is

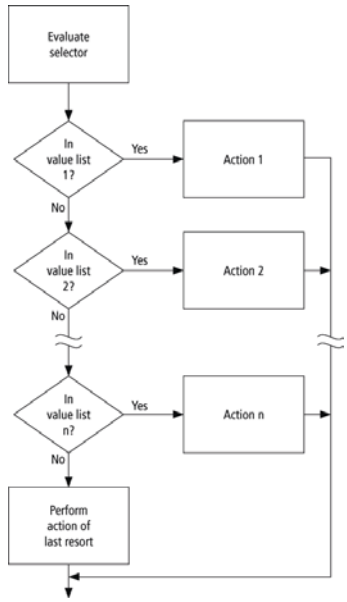
```
Select Case selector
  Case valueList1
    action1
  Case valueList2
    action2
  Case Else
    action of last resort
End Select
```

Select Case blocks (5.3) (cont.)

- Rules for Select Case
  - Case Else (and its action) is optional
  - Each value list contains one or more of the following types of items separated by commas:
    1. a literal;
    2. a variable;
    3. an expression;
    4. an inequality sign preceded by Is and followed by a literal, variable, or expression;
    5. a range expressed in the form a To b, where *a* and *b* are literals, variables, or expressions.

Select Case blocks (5.3) (cont.)

- Flowchart for Select Case



Select Case blocks (5.3) (cont.)

- Example 3

```
Dim x As Integer = 2, y As Integer = 3
Dim num As Integer
Select Case num
  Case y - x, x
    txtPhrase.Text = "Buckle my shoe."
  Case Is <= 4
    txtPhrase.Text = "Shut the door."
  Case x + y To x * y
    txtPhrase.Text = "Pick up sticks."
  Case 7, 8
    txtPhrase.Text = "Lay them straight."
  Case Else
     txtPhrase.Text = "Start all over again."
End Select
```

Select Case blocks (5.3) (cont.)

- Example 4

```
Dim firstName As String
firstName = txtName.Text.ToUpper
Select Case firstName
  Case "THOMAS"
    txtReply.Text = "Correct."
  Case "WOODROW"
    txtReply.Text = "Sorry, his full name" _
     & "  was Thomas Woodrow Wilson."
  Case "PRESIDENT"
    txtReply.Text = "Are you for real?"
  Case Else
    txtReply.Text = "Nice try."
End Select
```

Select Case blocks (5.3) (cont.)

- Example 7

```
Private Sub btnNumber_Click(…) Handles btnNumber.Click
    'Determine the number of days in a season

     Dim season As String
    season = txtSeason.Text
    txtNumdays.Text = season & " has " & Numdays(season) & "
  days."
End Sub
```

Function call

Select Case blocks (5.3) (cont.)

- Called Function in example 7

```
Function NumDays(ByVal season As String) _
                                 As Integer
  Select Case season.ToUpper
    Case "WINTER"
      Return 87
    Case "SPRING"
      Return 92
    Case "SUMMER", "AUTUMN", "FALL"
      Return 93
  End Select
End Function
```

Select Case blocks (5.3) (cont.)

- Comments
  - In a Case clause of the form *Case b To c*, the value of *b* should be less than or equal to the value of *c*.
  - The word Is should precede an inequality sign in a value list.
  - If the word Is is accidentally omitted where required, the editor will automatically insert it when checking the line.

Select Case blocks (5.3) (cont.)

- Data type comments
  - The items in the value list must evaluate to a literal of the same data type as the selector.
  - For instance, if the selector evaluated to a string value, as in

```
Dim firstName As String
firstName = txtBox.Text
Select Case firstName
```

then the clause

```
Case firstName.Length
```

would be meaningless.