Selection Controls

- VB .NET provides selection controls for users to make choices. These are controls as follows
  - CheckBox control
  - RadioButton control
  - ListBox control
  - ComboBox control
  - Horizontal and Vertical Scroll Bar control.
- All of these controls, except for the ComboBox control, limit the user to selecting from a predefined set of choices.

The CheckBox Control

- The CheckBox control provides a user with a Yes-or-No, or alternatively On-or-Off, option.
- Check boxes are also known as toggle selections.
- A check box initially checked or unchecked is set at design time using the check box’s Checked property.
- This property can be altered by the user at run time.
- The choice made by the user in one check box on a form has no effect on, nor depends on, the choice made in another check box on the form.
The RadioButton Control

- The **RadioButton** control provides a user with a set of one or more choices, only one of which can be selected.
- Radio buttons always operate as a group, where selecting one radio button immediately deselects and clears all the other buttons in the group.
- All radio buttons placed directly on a form become part of a single group.
- To create separate groups, the radio buttons must be placed in a **GroupBox** or **Panel** control.

The GroupBox Control

- Group boxes are used for providing clearly defined visual areas on a form or grouping controls together.
- They are especially important when there is a need to create two or more groups of radio buttons on the same form.
- Since a group box can be used to group a set of other controls, a group box is called a container control.

The ListBox Control

- The **ListBox** control is well suited for presenting a small list of items, either to be used for selection by the user or just as output.
- A **ListBox** appears as a rectangle that displays rows of text.
- A **ListBox** also contains a vertical scroll bar if the number of rows is too large to fit within the rectangle.
- Each line of output in a list box is referred to as an **Item**.
- The set of **Items** can be specified at design time by changing the **Items** property.
- The set of **Items** can also be changed at run time with procedural code.

Control Structures

- Flow of programming execution control is *sequential* unless a “control structure” is used to change that
- there are 2 general types of control structures:
  - **Selection** (also called branching)
  - **Repetition** (also called looping)
VB .NET control structures

- Selection
  - If-Then
  - If-Then-Else
  - If-Then-ElseIf
  - Select-Case
- Repetition
  - Do While loop
  - Do Until loop
  - For-Next loop

Relational and Logical Expressions

- Control Structures use relational and logical (Boolean) expressions
- Relational expressions use relational operators: < <= > >= = <>
- Boolean expressions use logical operators: Not And Or Xor

Boolean Data Type

- Result value from either a relational expression or Boolean expression is either True or False
- In VB .NET, the data type Boolean is a built-in type consisting of just 2 Boolean values, the constants True and False
- we can declare (Boolean) variables of type Boolean to hold Boolean values True or False

Boolean Expressions

A Boolean expression is an expression consists of
- Relational expressions with relational operators
- a Boolean variable or constant
- Boolean expressions with Boolean operators.
Relational Expressions

- Relational expressions are expressions used to compare quantities.
- A simple relational expression consists of a relational operator that compares two operands.
- The relational operators in Visual Basic are as follows:
  - `<` (less than)
  - `>` (greater than)
  - `<=(less than or equal to)`
  - `>=` (greater than or equal to)
  - `=` (equal to)
  - `<>` (not equal to)
- These relational operators can be used with all Visual Basic data types.
- Relational expressions are evaluated to yield a Boolean value, either True or False.

Relational expressions with relational operators

- Relational expression is in the form

  \[ \text{Operand1} \text{ Relational Operator } \text{Operand2} \]

- Examples:

  - Score1 > Score2
  - SideA + SideB = SideC
  - Length <> 0
  - CoefB*CoefB - 4*CoefA*CoefC > 0.0
  - Count <= 10
  - Response = 'Y'

Example

```vba
Dim Number1, Number2 As Integer
Number1 = 4
Number2 = 6

<table>
<thead>
<tr>
<th>EXPRESSION</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number1 &lt; Number2</td>
<td>True</td>
</tr>
<tr>
<td>Number1 + 2 &lt; Number2</td>
<td>False</td>
</tr>
<tr>
<td>Number1 &lt;= Number2</td>
<td>True</td>
</tr>
<tr>
<td>Number1 &lt;&gt; Number2</td>
<td>True</td>
</tr>
<tr>
<td>Number1 + 3 &gt;= Number2</td>
<td>True</td>
</tr>
<tr>
<td>Number2 = Number1+2</td>
<td>False</td>
</tr>
<tr>
<td>Number2 = Number1</td>
<td>True</td>
</tr>
</tbody>
</table>
```

Logical Operators

- The logical operators in Visual Basic are `And`, `Or`, `Xor`, and `Not` operators.
- These can be used to create more complex conditions than simple relational expressions.
- The `And`, `Or`, and `Xor` operators are applied between two expressions.
- The `And` operator evaluates to True only if both individual expressions are True by themselves.
- The `Or` operator evaluates to True if either one or both of the two expressions is True.
- The `Xor` operator evaluates to True only when one of the two expressions is True.
- The `Not` operator is applied to a single expression. It changes an expression to its opposite state.
- Like arithmetic operators, the order in which relational and logical operators are evaluated is determined by their precedence.
- Relational operators have a higher precedence than logical operators.
### Boolean variable or constant

```vbnet
Dim Done, Flag, Result, Test As Boolean
Dim Score1, Score2, Score3 As Double
Dim SideA, SideB, SideC As Integer
Done = True
Flag = False
Test = Score1 > Score2
Result = SideA + SideB = SideC
Result = Done And Flag
```

### And Operator

<table>
<thead>
<tr>
<th>Value of X</th>
<th>Value of Y</th>
<th>Value of X And Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
</tbody>
</table>

### Or Operator

<table>
<thead>
<tr>
<th>Value of X</th>
<th>Value of Y</th>
<th>Value of X Or Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
</tbody>
</table>

### Xor Operator

<table>
<thead>
<tr>
<th>Value of X</th>
<th>Value of Y</th>
<th>Value of X Xor Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
</tbody>
</table>
### Not Operator

<table>
<thead>
<tr>
<th>Value of X</th>
<th>Value of Not X</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
</tr>
</tbody>
</table>

### Example

**Dim age As Integer**
**Dim isSenior, hasFever As Boolean**
**Dim temperature As Double**

- age = 20
- temperature = 102.0
- isSenior = (age >= 55)
- hasFever = (temperature > 98.6)

<table>
<thead>
<tr>
<th>EXPRESSION</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>isSenior</td>
<td>False</td>
</tr>
<tr>
<td>hasFever</td>
<td>True</td>
</tr>
<tr>
<td>isSenior Or hasFever</td>
<td>True</td>
</tr>
<tr>
<td>Not isSenior</td>
<td>True</td>
</tr>
<tr>
<td>Not hasFever</td>
<td>False</td>
</tr>
</tbody>
</table>

### Example

**Dim age, height As Integer**

- age = 25
- height = 70

<table>
<thead>
<tr>
<th>EXPRESSION</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not (age &lt; 10)</td>
<td>True</td>
</tr>
<tr>
<td>Not (height &gt; 60)</td>
<td>False</td>
</tr>
</tbody>
</table>
Example

```vbnet
Dim age, weight As Integer
age = 25
weight = 195
```

<table>
<thead>
<tr>
<th>EXPRESSION</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>(weight &gt; 180) And (age &gt;= 50)</td>
<td>False</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
</tr>
</tbody>
</table>

Precedence of operator

```
<table>
<thead>
<tr>
<th>Operator</th>
<th>Not</th>
<th>*</th>
<th>/</th>
<th>+</th>
<th>-</th>
<th>\</th>
<th>*</th>
<th>/</th>
</tr>
</thead>
<tbody>
<tr>
<td>And</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Or</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Xor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Xnor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>=</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;=</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&gt;=</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Example

```vbnet
Dim myState As String
Dim yourState As String
myState = "Texas"
yourState = "Maryland"
```

<table>
<thead>
<tr>
<th>EXPRESSION</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>myState = yourState</td>
<td>False</td>
</tr>
<tr>
<td>myState &gt; yourState</td>
<td>True</td>
</tr>
<tr>
<td>myState = &quot;Texas&quot;</td>
<td>True</td>
</tr>
<tr>
<td>myState &lt; &quot;texas&quot;</td>
<td>True</td>
</tr>
</tbody>
</table>

Write an expression for each

- taxRate is over 25% and income is less than $20000
  `(taxRate > 0.25) And (income < 20000)`
- temperature is less than or equal to 75 or humidity is less than 70%
  `(temperature <= 75) Or (humidity < .70)`
- age is over 21 and age is less than 60
  `(age > 21) And (age < 60)`
- age is 21 or 22
  `(age = 21) Or (age = 22)`
WARNING about Expressions in VB .NET

- “Boolean expression” means an expression whose value is True or False
- an expression is any valid combination of operators and operands
- each expression has a value
- this can lead to UNEXPECTED RESULTS
- construct your expressions CAREFULLY
- use of parentheses is encouraged
- otherwise, use precedence chart to determine order

Selection statements

are used to choose an action depending on the current situation in your program as it is running

Control Structure

- A selection statement is a control structure used to (alter the sequential flow of control) choose an action depending on the current situation in your program as it is running.
- If–Then–Else statement

If–Then–Else Structure

- The If–Then–Else structure directs a procedure to perform a series of one or more instructions, based on the result of a comparison.
- The use of the Else statement is optional.
- The version of the If statement in which the Else statement is not included is called a one-way If structure.
- It has the following general form:
  
  If (condition) Then
  statement(s)
  End If

- If only one statement is to be executed and the condition in the If statement is true and there is no Else statement, a single-line If–Then statement without the End If may be used.
- This version of the If statement has the following general form:
  
  If (condition) Then statement
Syntax of **If-Then-Else**

If (condition) Then
  statement(s)
Else
  statement(s)
End If

The use of the Else statement is optional. The Else part can be dropped if there is no second choice.

**Example**

If (Weight < 300) Then
  DisplayLabel.Text = "You are okay."
Else
  DisplayLabel.Text = "You are overweight."
End If

**Use of blocks recommended**

If (Expression) Then
  “if clause”
Else
  “else clause”
End If

**Selection Statement If-Then-Else**

```
If (expression) Then
  if clause
Else
  else clause
End If
```
Selection Statement

If-Then-Else

- True: Weight < 300
  - DisplayLabel.Text = "You are okay."
- False: Weight ≥ 300
  - DisplayLabel.Text = "You are overweight."

Example

- Assign value .25 to discountRate and assign value 10.00 to shipCost if purchase is over 100.00
- Otherwise, assign value .15 to discountRate and assign value 5.00 to shipCost
- Either way, calculate totalBill

Code for Example

```vbnet
If (Purchase > 100.00) Then
    DiscountRate = 0.25
    ShipCost = 10.00
Else
    DiscountRate = 0.15
    ShipCost = 5.00
End If
TotalBill = Purchase * (1.0 - DiscountRate) + ShipCost
```
If-Then Syntax

If (Boolean expression) Then
Statement(s)
End If

• NOTE: Statement can be a single statement, a null statement, or a block.

Example

Number = InputTextBox.Text
If (Number < 0) Then
  DisplayLabel.Text = "The number that you enter is negative."
End If

If-Then or If-Then-Else

• If TaxCode is 'T', increase price by adding TaxRate times Price to it.
  If (TaxCode = "T") Then
    Price = Price + TaxRate * Price
  End If

• If code has value 1, retrieve values from IncomeTextBox income and TaxRateTextBox, and calculate and display TaxDue as their product.
  If (code = 1) Then
    TaxDueTextBox.Text = Val(IncomeTextBox.Text) * Val(TaxRateTextBox.Text)
  End If

• If A is strictly between 0 and 5, set B equal to 1/A, otherwise set B equal to A.
  If ((A > 0) And (A < 5)) Then
    B = 1/A
  Else
    B = A
  End If
A Numerical Accuracy Problem

- Tests for equality of floating point and double precision values and variables using the relational operator, =, should be avoided.
- Many decimal numbers cannot be represented exactly in binary using a finite number of bits.
- As a result, tests of exact equality of these numbers can fail.

Nested If Statements

- In a nested If statement, one or more If-Then-Else statements are included within another If-Then-Else statement.
- The If-Then-ElseIf structure is a modification to the If-Then-Else structure. It has the following syntax:

```
If condition-1 Then
  statement(s)
ElseIf condition-2 Then
  statement(s)
ElseIf condition-3 Then
  statement(s)
ElseIf condition-4 Then
  statement(s)
Else
  statement(s)
End If
```

- Each Expression is evaluated in sequence, until some Expression is found that is true.
- Only the specific Statement following that particular true Expression is executed.
- If no Expression is true, the Statement following the final Else is executed.
- Actually, the final Else and final Statement are optional. If omitted, and no Expression is true, then no Statement is executed.

Nested Selections

```
If (creditsEarned >= 90) Then
  DisplayLabel.Text = "SENIOR STATUS"
ElseIf (creditsEarned >= 60) Then
  DisplayLabel.Text = "JUNIOR STATUS"
ElseIf (creditsEarned >= 30) Then
  DisplayLabel.Text = "SOPHOMORE STATUS"
Else
  DisplayLabel.Text = "FRESHMAN STATUS"
End If
```

Example

```
If (Average >= 90) Then
  DisplayLabel.Text = "Your grade is an A."
ElseIf (Average >= 80) Then
  DisplayLabel.Text = "Your grade is an B."
ElseIf (Average >= 70) Then
  DisplayLabel.Text = "Your grade is an C."
ElseIf (Average >= 60) Then
  DisplayLabel.Text = "Your grade is an D."
Else
  DisplayLabel.Text = "Your grade is an F."
End If
```
Writing Nested if Statements

• Display one word to describe the value of number as “Positive”, “Negative”, or “Zero”

```vbnet
If (Number > 0) Then
    DisplayLabel.Text = "Positive"
ElseIf (Number < 0) Then
    DisplayLabel.Text = "Negative"
Else
    DisplayLabel.Text = "Zero"
EndIf
```

Writing Nested if Statements

Your city classifies a pollution index
• less than 35 as “Pleasant”,
• 35 through 60 as “Unpleasant”,
• and above 60 as “Health Hazard.”
• Display the correct description of the pollution index value.

```vbnet
If (Index < 35)
    DisplayLabel.Text = "Pleasant"
ElseIf (Index <= 60)
    DisplayLabel.Text = "Unpleasant"
Else
    DisplayLabel.Text = "Health Hazard"
EndIf
```

Input Data Validation

• In defensive programming, code is used to check for improper input data before attempting to process it further.
• The defensive programming technique of checking user input data for erroneous or unreasonable data is called input data validation.
• **If-Then-Else** statements can be used to perform input data validation.

```vbnet
Dim InputAmount, AmountLeft, Quarters, Dimes, Nickels, Pennies As Integer
REM Set the InputAmount to the input amount
InputAmount = InputAmountTextBox.Text
If (InputAmount <= 0) Or (InputAmount >= 100) Then
    DisplayChangeListBox.Items.Add(CStr(InputAmount) + " is invalid.")
Else
    REM Find the Number of Quarters
    Quarters = InputAmount \ 25
    REM Find the amount left from Quarters
    AmountLeft = InputAmount Mod 25
    REM Find the Number of Dimes
    Dimes = AmountLeft \ 10
    REM Find the amount left from Dimes
    AmountLeft = AmountLeft Mod 10
    REM Find the Number of Nickels
    Nickels = AmountLeft \ 5
    REM Find the number of pennies
    Pennies = AmountLeft Mod 5
    REM Display the result
    DisplayChangeListBox.Items.Add("The amount of " + CStr(InputAmount) + " cents has:")
    DisplayChangeListBox.Items.Add("" + CStr(Quarters) + " Quarters")
    DisplayChangeListBox.Items.Add("" + CStr(Dimes) + " Dimes")
    DisplayChangeListBox.Items.Add("" + CStr(Nickels) + " Nickels")
    DisplayChangeListBox.Items.Add("" + CStr(Pennies) + " Pennies")
End If
```
Example

Select Case Structure

The Select-Case structure is an alternative to the If-Then-ElseIf structure. The syntax of the Select-Case construct is as follows:

```
Select Case expression
  Case value_1
    statement(s)
  Case value_2
    statement(s)
  Case value_n
    statement(s)
  Case Else
    statement(s)
End Select
```

Example

If (creditsEarned >= 90) Then
  Select Case creditsEarned
    Case 90 to 99
      DisplayLabel.Text = "SENIOR STATUS"
    Case 61 to 89
      DisplayLabel.Text = "JUNIOR STATUS"
    Case 31 to 59
      DisplayLabel.Text = "SOPHOMORE STATUS"
    Case Else
      DisplayLabel.Text = "FRESHMAN STATUS"
  End Select
ElseIf (creditsEarned >= 60) Then
  Select Case creditsEarned
    Case 90 to 99
      DisplayLabel.Text = "SENIOR STATUS"
    Case 61 to 89
      DisplayLabel.Text = "JUNIOR STATUS"
    Case 31 to 59
      DisplayLabel.Text = "SOPHOMORE STATUS"
    Case Else
      DisplayLabel.Text = "FRESHMAN STATUS"
  End Select
Else
  DisplayLabel.Text = "FRESHMAN STATUS"
End If
Example

If (Average >= 90) Then
  DisplayLabel.Text = "Your grade is an A."
ElseIf (Average >= 80) Then
  DisplayLabel.Text = "Your grade is a B."
ElseIf (Average >= 70) Then
  DisplayLabel.Text = "Your grade is a C."
ElseIf (Average >= 60) Then
  DisplayLabel.Text = "Your grade is a D."
Else
  DisplayLabel.Text = "Your grade is an F."
End If

Select Case Average
  Case Is >= 90
    DisplayLabel.Text = "Your grade is an A."
  Case Is >= 80
    DisplayLabel.Text = "Your grade is a B."
  Case Is >= 70
    DisplayLabel.Text = "Your grade is a C."
  Case Is >= 60
    DisplayLabel.Text = "Your grade is a D."
  Case Else
    DisplayLabel.Text = "Your grade is an F."
End Select