Object Oriented Programming with Visual Basic .Net

By:

Dr. Hossein Hakimzadeh
Computer Science and Informatics
IU South Bend

(c) Copyright 2007 to 2015 H. Hakimzadeh
What do we need to learn in order to write computer programs?

- Fundamental programming constructs:
  - Variables,
  - Arithmetic operators,
  - Input and output
  - **Conditionals**, 
  - Loops,
  - Procedures and functions,
  - Arrays,
  - Structures, classes and objects,
  - Files
Conditionals:

- Remember this figure.

- **Conditional** statements are implemented in the Arithmetic Logic Unit of the computer.
Conditionals

Conditionals, are statements that allow the programmer to express logical conditions and make decisions based on the result of that condition.

For example, let us assume that as a programmer you have been asked to develop a program that controls a car’s Anti-lock-break system (ABS), you are told that the ABS system should only be enabled if the temperature is equal or below freezing.

How would you code this requirement into your program?
The If statement

- if \textbf{Condition} then
- \hspace{1em} \textbf{Statement}
- end if

The **Condition** is expressed as follows that evaluates as TRUE or FALSE:

\[
< \text{expression} > \ < \text{relational operator} > \ < \text{expression} >
\]

\[
X \hspace{1em} = \hspace{1em} Y
\]

The **Relational Operators** are:

- \(=\) equal
- \(<>\) not equal
- \(<\) less than
- \(>\) greater than
- \(\leq\) less than or equal
- \(\geq\) greater than or equal
The If statement

...develop a program that controls a car’s Anti-lock-break system (ABS), you are told that the ABS system should only be enabled if the temperature is equal or below freezing.

Example:

- If (temperature <= 32) then
  ABS = “ENABLED”
- End if
The If statement

- write a condition which checks the student's test score and displays "Excellent Work" if their score is greater than or equal to 90.

Example:
- If (score >= 90) then
- Console.WriteLine("Excellent Work!")
- End if
The If –Then-Else statement

- if **Condition** then
- The_true_part
- else
- The_false_part
- end if

**Example:**
- If age >= 21 then
  - Console.WriteLine("drink beer..")
- else
  - Console.WriteLine("drink root beer..")
- End if
Things to watch out for!

What’s wrong with this example?

- If age = 21 then
  - Console.WriteLine("drink beer..")
- else
  - Console.WriteLine("drink root beer..")
- End if

Note what happens if the person’s age is 15. How about when their age is 25?
Things to watch out for!

- Can we fix the problem if we change it to:

  ```csharp
  If  age <> 21 then
  Console.WriteLine("drink root beer..")
  else
  Console.WriteLine("drink beer..")
  End if
  ```

- What happens if the person’s age is 15. How about when their age is 25? Doesn’t seem to help does it? The condition that you choose for your IF statement is very important. You need to desk check it to make sure it works before trying it inside your program.
ElseIF Statement

- Sometimes, the ELSE part of an IF statement immediately starts with another IF statement. In such cases, it is simpler and cleaner if we use the “ElseIF” statement.

- Example:

```csharp
If score >= 90 Then
    Console.WriteLine("A")
ElseIf score >= 80 Then
    Console.WriteLine("B")
ElseIf score >= 70 Then
    Console.WriteLine("C")
ElseIf score >= 60 Then
    Console.WriteLine("D")
Else
    Console.WriteLine("Failing Grade")
End If
```
Compare and Contrast the **ElseIF** vs. a series of **Nested IF’s**

If score >= 90 Then
    Console.WriteLine("A")
ElseIf score >= 80 Then
    Console.WriteLine("B")
ElseIf score >= 70 Then
    Console.WriteLine("C")
ElseIf score >= 60 Then
    Console.WriteLine("D")
Else
    Console.WriteLine("Failing Grade")
End If

If score >= 90 Then
    Console.WriteLine("A")
Else
    If score >= 80 Then
        Console.WriteLine("B")
    Else
        If score >= 70 Then
            Console.WriteLine("C")
        Else
            If score >= 60 Then
                Console.WriteLine("D")
            Else
                Console.WriteLine("Failing Grade")
            End If
        End If
    End If
End If
Select Case Statement

- An alternative to a series of if-then-elseif statements.
- A long sequence of if-then-elseif statements is hard to read, and debug.
- An alternative is to replace the if-then-elseif statements with a “Select Case” statement.

Syntax:

```
Select Case testExpression
  [ Case expressionList
    [ statements] ]
  [ Case Else
    [ else-statements] ]
End Select
```
Compare and Contrast

Select-Case vs. an If-Then-Else Statement

```csharp
Select Case value
    Case 1
        Console.WriteLine("The value is 1")
    Case 2, 3
        Console.WriteLine("The value {0} is 2 or 3", value)
    Case 4, 5, 6
        Console.WriteLine("The value {0} is 4, 5 or 6", value)
    Case Else ' Default case
        Console.WriteLine("The value {0} is something other than 1,2,3,4,5 or 6", value)
End Select
```

```csharp
If value = 1 Then
    Console.WriteLine("The value is 1")
ElseIf value = 2 Or value = 3 Then
    Console.WriteLine("The value {0} is 2 or 3", value)
ElseIf value = 4 Or value = 5 Or value = 6 Then
    Console.WriteLine("The value {0} is 4, 5 or 6", value)
Else ' Default case
    Console.WriteLine("The value {0} is something other than 1,2,3,4,5 or 6", value)
End If
```
Boolean...

- Boolean Variables, Expressions, and Operators
What is Boolean Variable?

- A **Boolean** variable is a variable which can only hold one of two values. Either **True** or **False**.

Example:

```plaintext
Dim DONE As Boolean
DONE = FALSE
```
Boolean Variables

- BOOLEAN variables can make your programs more readable. Consider the following code segment:

```vbnet
Dim FINISHED_WORKING As Boolean
FINISHED_WORKING = FALSE
...
If (FINISHED_WORKING) then
    Console.WriteLine("Take a break..")
```
Boolean Expressions

- An expression which evaluates to either TRUE or FALSE.
  - \( X > Y \)
  - \( X = Y \)
  - \( X \leq 48 \)

- A Boolean expression has the following syntax:
  - \(<\text{operand}> \ \text{relational-operator} \ <\text{operand}>\)

- Operands can be variables, constants or other expressions:
  - \(<\text{variable}> \ \text{relational-operator} \ <\text{variable}>\)
  - or
  - \(<\text{variable}> \ \text{relational-operator} \ <\text{constant}>\)

- Relational Operators are:
  - =, <, >, <=, >=, <>

- Constants may be:
  - literal constants (i.e. 1, 5.2, “hello”, ‘x’, TAX_RATE, PI)
Boolean Operators

- **Boolean operators** are used to form more complicated Boolean expressions.

- **Syntax:**
  
  `<Boolean expr> Boolean_operator <Boolean expr>`

- There are **three Boolean operators**:
  - AND
  - OR
  - NOT

- **Complex Boolean Expressions:**

  Is formed by combining simple Boolean expressions with the **AND, OR, NOT** operators.

- **Examples:**

  (Salary < MinSal) **OR** (NumDependents > 5)
  (hours_worked > 40) **AND** (hours_worked <= 60)
  (hours_worked > 40) **AND** (**NOT** SALARY_EMP)
# Truth Table

<table>
<thead>
<tr>
<th>AND</th>
<th>True</th>
<th>False</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OR</th>
<th>True</th>
<th>False</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NOT</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td></td>
</tr>
</tbody>
</table>
# Evaluate the following Boolean Expressions:

**Assume:**

Dim X, Y, Z As Double  
Dim FLAG As Boolean  
X = 3.0  
Y = 4.0  
Z = 2.0  
Flag = false

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>(X &gt; Z) AND (Y &gt; Z)</td>
<td>TRUE</td>
</tr>
<tr>
<td>(X + Y / Z) &lt;= 3.5</td>
<td>TRUE</td>
</tr>
<tr>
<td>(Z &gt; X) OR (Z &gt; Y)</td>
<td>FALSE</td>
</tr>
<tr>
<td>NOT Flag</td>
<td></td>
</tr>
<tr>
<td>(X = 1.0) OR (X = 3.0)</td>
<td></td>
</tr>
<tr>
<td>(Z &lt; X) AND (X &lt; Y)</td>
<td></td>
</tr>
<tr>
<td>(X &lt;= Z) OR (X &gt;= Y)</td>
<td></td>
</tr>
<tr>
<td>(NOT Flag) OR ((Y + Z) &gt;= (X - Z))</td>
<td></td>
</tr>
<tr>
<td>NOT (Flag OR ((Y + Z) &gt;= (X - Z)))</td>
<td></td>
</tr>
</tbody>
</table>
Combining Ideas...

Let us redo our if statement for calculating grades using Boolean expressions and Boolean operators:

- If (test_score >= 90) And (test_score <= 100) Then
  - letter_grade = "A"
  - End If

- If (test_score >= 80) And (test_score < 90) Then
  - letter_grade = "B"
  - End If

- If (test_score >= 70) And (test_score < 80) Then
  - letter_grade = "C"
  - End If

- If (test_score >= 60) And (test_score < 70) Then
  - letter_grade = "D"
  - End If

- If (test_score < 60) Then
  - letter_grade = "F"
  - End If