B100/I101
Problem Solving Using Computers

VB .NET
Conditional Statements & Boolean Expressions

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

Adapted from Drs. Adaikkalavan, Hakimzadeh & Zhang
Remember this figure?

**Conditional** statements are implemented in the Arithmetic Logic Unit of the computer.
The If statement

\[
\text{If } \text{condition(s)} \text{ Then}\n\text{statement(s)}\n\text{End If}
\]

A condition is expressed as follows that evaluates as TRUE or FALSE:

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

\[
\text{count} \not= 10
\]

The Relational Operators (also called comparison operators) are:

\[
= \quad \text{equal} \\
<> \quad \text{not equal} \\
< \quad \text{less than} \\
> \quad \text{greater than} \\
\leq \quad \text{less than or equal} \\
\geq \quad \text{greater than or equal}
\]
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:

```c
If (temperature <= 32) Then
    Console.WriteLine("ABS ENABLED")
End If
```
The If –Then-Else statement

If condition(s) Then
    The_true_part
Else
    The_false_part
End If

Example:

Dim age As Double
age = CDbl(Console.ReadLine())
If (age >= 21) Then
    Console.WriteLine("can enjoy adult drink..")
Else
    Console.WriteLine("can enjoy soft drink..")
End If
Console.WriteLine("bye")
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
```
What is Boolean Variable?

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

Example:

```
Dim done As Boolean
done = False
```
Boolean Variables

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

```vbnet
Dim finishedHomework As Boolean
finishedHomework = False

If (finishedHomework) then
    Console.WriteLine("Take a break..")
Else
    Console.WriteLine("U have to work hard")
End If
```
Boolean Expressions

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

- A Boolean expression has the following syntax:
  
  `<operand>  relational-operator  <operand>`

- `<operand>` can be variables, constants or even other expressions:
  
  `<variable>  relational-operator  <variable>`
  
  or
  
  `<variable>  relational-operator  <constant>`

- `<relational-operator>`s are:
  
  `= , < , > , <= , >= , <>`

- `<constant>`s may be:
  
  constants (i.e., 1, 5.2, “hello”, 'x', TAX_RATE, PI)
Logical Operators

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

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

- There are **three Logical operators**:
  - *And*
  - *Or*
  - *Not*

- **Complex Boolean Expressions:**
  Is formed by combining simple Boolean expressions with the *And*, *Or*, *Not* operators.

- **Examples:**
  
  
  
  (Salary < MIN_SAL) **Or** (numDependents > 5)
  
  (hoursWorked > 40) **And** (hoursWorked <= 60)
  
  (hoursWorked > 40) **And** (**Not** SALARY_EMP)
Evaluate the following Boolean Expressions: Result is True or False?

Assume:

Dim T, X, Y, Z As Double
Dim FLAG As Boolean

X = 3.0
Y = 4.0
Z = 2.0
T = 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></td>
</tr>
<tr>
<td>(X + Y / Z) &lt;= 3.5</td>
<td></td>
</tr>
<tr>
<td>(Z &gt; X) Or (Z &gt; Y)</td>
<td></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>
<tr>
<td>X * 2 / 3 &gt; 2.0 Or Not Flag</td>
<td></td>
</tr>
</tbody>
</table>
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
```

- **testExpression**: Required. Must evaluate to one of the elementary data types (Boolean, Byte, Char, Date, Double, Decimal, Integer, Long, Object, SByte, Short, Single, String,UInteger, ULong, and UShort).
Summary of Selections Statements

- **If ... Then** selection statement
  - **Single-selection** statement – it selects or ignore a single action (or a sequence of actions)

- **If ... Then...Else** selection statement
  - **Double-selection** statement – it selects between two different actions (or sequences of actions)

- **Select...Case** selection statement
  - **Multiple-selection** statement -- performs one of many possible actions (or sequences of actions)
  - So do the nested IF’s and ElseIf statements
Precedence (for the operators we’ve learned so far)

- ^  Highest precedence (done first)
- *, /, 
- \  
- Mod  
- +, -, string concatenation (+)  
- string concatenation (&)  
- All comparison operators (=, <>, <, <=, >, >=)  
- Not  
- And  
- Or  
- = (assignment)  Lowest precedence (done last)

Operator Precedence in VB: