# Visual Web Development

Terry Marris  October 2007

## *6  Selections*

We look at relational operators, logical operators and the If ... Then .... Else construct and some variations.

### 6.1 The Relational or Comparison Operators

We have already met the types Integer (whole numbers) and Double (numbers with a decimal point) in Chapters 4 and 5.  We now look at the relational or comparison operators:

| **Operator** | **Meaning** | **Example** |
|---|---|---|
| < | is less than | 2 < 3 |
| <= | is less than or equal to | 2 <= 3, 3 <= 3 |
| > | is greater than | 3 > 2 |
| >= | is greater than or equal to | 3 >= 2, 3 >= 3 |
| = | is equal to | 3 = 3 |
| <> | is not equal to | 2 <> 3 |

Notice that the **L**ess-than symbol has its point on the **L**eft.  Notice also that where two symbols are used, as in <=, there is no space between them and the = symbol is always last.

We need to take care when using the relational operators with real values since they are not always stored exactly.  For example, 0.19999999 and 0.20000001 may both represent the value 0.2.  One way of determining whether two real numbers are equal is to:

- define a tolerance for two real numbers to be considered equal
- use Maths.Abs() to calculate the absolute difference between the two values. Absolute means the negative sign is ignored, so -0.001 is taken to be +0.001.
- define equal to be true if the absolute difference is less than the tolerance

For example:

```
Dim dblA As Double = 0.19
Dim dblB As Double = 0.21
Dim dblTolerance As Double = 0.1
Dim dblAbsoluteDifference As Double = Math.Abs(dblA - dblB)
Dim boolEqual As Boolean = (dblDifference < dblTolerance)
```

So:

```
dblAbsoluteDifference = 0.21 - 0.19 = 0.02
(dblDifference < dblTolerance) = (0.02 < 0.1) = True
```

Therefore, according to the tolerance we set, 0.19 = 0.21.

## 6.2 The Logical or Boolean Operators

The logical or Boolean operators include:

| **Operator** | **Meaning** | **Example** |
|---|---|---|
| And | both true at the same time | isRich And isGoodLooking |
| Or | either one or the other or both are true | isRich Or isGoodLooking |
| Not | negates (or reverses) the truth value | Not (isPoor) ⇒ isRich |

```
Dim isRich As Boolean
Dim isGoodLooking As Boolean
Dim strStatus As String = "Unknown"
...
If isRich And isGoodLooking  Then
   strStatus = "desirable"
End If
```

Both isRich and isGoodLooking must be true if status is to be "desirable". (isRich And isGoodLooking) is true if both isRich and isGoodLooking are true.

```
If isRich Or isGoodLooking  Then
   strStatus = "acceptable"
End If
```

Either isRich is true or isGoodLooking is true (or possibly both are true) if status is to be "acceptable". (isRich Or isGoodLooking) is true if at least one of IsRich and IsGoodLooking is true.

## 6.3 If ... Then ...

```
Dim strStatus As String = "Fail"
Dim intMark As Integer
...

If intMark  >= 40 Then
    strStatus = "Pass"
End If
```

Initially, strStatus has the value Fail. This value is changed to Pass only if intMark has a value of 40 or more.

The general format is:

```
If <booleanExpression> Then
    statementSequence
End If
```

In our example, the <booleanExpression> is intMark >= 40; its value is either True or False. And there is just one statement in the statementSequence, namely strStatus = "Pass". In theory, you could have as many statements as you like in a statementSequence.

If, Then and End are all Visual Basic words.

## 6.4  If ... Then ... Else ...

```
Dim strStatus As String = "Unknown"
Dim intMark As Integer
...

If intMark >= 40 Then
   strStatus = "Pass"
Else
   strStatus = Fail""
End If
```

strStatus has the value "Pass" only if intMark is 40 or more; if intMark is not 40 or more, strStatus has the value "Fail".

The general format is:

```
If  <booleanExpression> Then
   statementSequence1
Else
   statementSequence2
End If
```

If the <booleanExpression> is True, then statementSequence1 is executed.  But if <booleanExpression> is False, then statementSequence2 is executed.

## 6.5 If ... Then ... ElseIf ...

```
Dim strStatus As String = "Unknown"
Dim intMark As Integer
...

If intMark < 0 Then
   strStatus = "Error"
ElseIf intMark < 40 Then
   strStatus = "Fail"
ElseIf intMark <= 100 Then
   strStatus = "Pass"
Else
   strStatus = "Cheat"
End If
```

Understanding a sequence of If ... ElseIfs  is easy.  Just look down the sequence of Boolean expressions one by one, find the first one that is True, execute the corresponding action, and then skip to the End If.

For example suppose intMark has the value 50:

   intMark < 0?  No
   intMark < 40? No
   intMark <= 100? Yes.  So copy "Pass" into strStatus and finish.

The general format is:

```
If <booleanExpression1> Then
   statementSequence1
ElseIf <booleanExpression2> Then
   statementSequence2
...

Else
   statementSequenceN
End If
```

Notice that:

- there are no spaces in ElseIf
- each ElseIf is written directly in line under the previous If or ElseIf
- the final Else catches none of the above cases.  It is good practice to always include the final else.

## 6.6 Exercise

1. Complete a program that meets the Use Case shown below.

| | |
|---|---|
| **Use Case** | Grade From Mark |
| **Goal** | to display the grade corresponding to an exam mark input |
| **Pre-condition** | an integer representing an exam mark between 0 and 100 inclusive is entered |
| **Post-condition** | the corresponding grade is displayed:<br>fail if the mark is between 0 and 39 inclusive<br>pass if the mark is between 40 and 100 inclusive |
| **Initiating Actor** | the user |

**Main Success Scenario**

**1** system prompts user for exam mark
**2** user enters exam mark
**3** system displays grade
**4** exit success

**Exceptions**

**2a** non-numeric input
  **1** system displays error message
  **2** exit failure
**2b** mark < 0
  **1** system displays error message
  **2** exit failure
**2c** mark > 100
  **1** system displays error message
  **2** exit failure

2. Design, write and test a program that wil input a weight in Kilograms and a height in metres, and output Clinically Obese if weight / height$^2 \geq 30$