# DECISION STRUCTURES:
## THE *IF* AND *SELECT CASE* STATEMENT

When writing programs, you often have to make a decision about which line of code to execute next. If you don't make a decision, all the code in a procedure will simply execute from left to right and top to bottom. Many times you need to execute a different set of code based on some criteria input by the user. Visual Basic helps you perform this type of conditional logic by supplying you with **If** statements and **Select Case** statements.


Just as you might say to yourself, "If she doesn't call me, I am not going to speak to her again," a computer program uses **if** to decide what actions to take.

## THE *If...Then* STATEMENT

The **If...Then** statement is a decision structure that executes code when a condition is TRUE. For example, **guess = 10** is the condition in the following **If...Then** statement:

```
If guess = 10 Then
   lblMessage.Text = "You guessed the correct number!"
End If
```

If the value of **guess** is 10, the assignment statement changing the Text property of the label is executed. If the value of the **guess** is not equal to 10 (i.e. it is greater or less than 10), then the assignment statement is not executed and program flow continues to the **End If**, which is required to end the **If...Then** statement.

The **If...Then** statement, thus, takes the following form:

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

## THE *If...Then...Else* STATEMENT

The second version of the **If** statement includes an **Else** clause which is executed when the **If** condition returns FALSE. The **If...Then...Else** statement takes the following form:

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

If the condition returns TRUE, the statements after **Then** are executed. If the condition returns FALSE, the statements after **Else** are executed, skipping the statements after **Then**.

**EXAMPLE:**

```
If guess = 10 Then
   lblMessage.Text = "You guessed the correct number!"
Else
   lblMessage.Text = "Incorrect!  Try again!"
End If
```

## THE *If...Then...ElseIf* STATEMENT

The **If...Then...ElseIf** statement is used to decide among three, four or more actions and takes the following form:

```
If condition1 Then
   statement(s)
ElseIf condition2 Then
   statements(s)
Else
   statements(s)
End If
```

This decision structure is extremely useful if a number of conditions need to be tested. You can add as many **ElseIf** statements as necessary. The statements after **Else** are executed if none of the previous conditions tested return TRUE.

**EXAMPLE:**

```
If guess = 10 Then
   lblMessage.Text = "You guessed the correct number!"
ElseIf guess < 10 Then
   lblMessage.Text = "Your guess is too low!  Try again!"
ElseIf guess > 10 Then
   lblMessage.Text = "Your guess is too high!  Try again!"
Else
   lblMessage.Text = "Invalid guess!"
End If
```

## RELATIONAL OPERATORS

Logical expressions are constructed with **relational operators**. A relational operator determines whether a specific relationship exists between two values. For example, the equal sign (=) is a relational operator that compares the value of the expression to the left, with the value of the expression to the right. It always returns a TRUE or FALSE value.

The following table outlines the relational operators used in Visual Basic:

| RELATIONAL OPERATOR | DESCRIPTION |
|---|---|
| > | Greater than |
| < | Less than |
| = | Equals |
| <> | Not equal |
| >= | Greater than or equal to |
| <= | Less than or equal to |

## LOGICAL OPERATORS

**Logical operators** combine two or more relational expressions into one, or reverse the logic of an expression. The following table outlines the logical operators used in Visual Basic:

| LOGICAL OPERATOR | DESCRIPTION | EXAMPLE |
|---|---|---|
| And | Combines two or more expressions into one; each expression must be true for the overall expression to be true. | `If (age >= 18) And (age <= 65) Then Label1.Text = "You're hired!" End If` |
| Or | Combines two or more expressions into one; only one expression needs to be true for the overall expression to be true. | `If (age < 18) Or (age > 65) Then Label1.Text = "You're NOT hired!" End If` |
| Xor | Combines two or more expressions into one; only one expression (not more than one) must be true for the overall expression to be true. | `If (tot > 100) Xor (avg > 20) Then Label1.Text = "Try again!" End If` |
| Not | Reverses the logical value of an expression (i.e. makes a true expression false and a false expression true. | `If Not (course = "TIK2O1") Then Label1.Text = "Not in my class!" End If` |

## THE Select...Case STATEMENT

The **Select...Case** statement is a decision structure that uses the value of an expression to determine which block of code to execute. Unlike the **If...Then...ElseIf** statement which performs a series of tests and executes a set of statements when one of the tests returns true, the **Select...Case** statement test the value of an expression once and then uses that value to determine which set of statements to execute.

The **Select...Case** statement takes the following form:

```
Select Case expression
   Case condition1
      statement(s)
   Case condition2
      statement(s)
   Case Else
      statement(s)
End Select
```

The **expression** may be any numeric or string expression whose value you wish to test. There can be multiple **Case** clauses, and the **Case Else** clause is optional. The **condition** type must match the **expression** type and can be a single value, a list separated by commas or a range separated by the keyword **To**. The **End Select** statement is required to end the **Select...Case** statement.

**EXAMPLE:**

The following **Select...Case** statement uses the value of a score to determine the message to display:

```
Dim score As Integer

Select Case score
   Case 0
      MessageBox.Show("You got zero!")
   Case 1 To 4
      MessageBox.Show("You failed!")
   Case 5 To 10
      MessageBox.Show("You passed!")
   Case Else
      MessageBox.Show("Invalid score!")
End Select
```

## THE *Select...Case Is* STATEMENT

The **Select...Case Is** statement compares the result of an expression to a range of values when a relational operator is part of the value. For example, the following statement uses ranges to determine the message to display:

```
Dim score As Integer

Select Case score
   Case 0
      MessageBox.Show("You got zero!")
   Case 1 To 4
      MessageBox.Show("You failed!")
   Case 5, 6
      MessageBox.Show("You passed!")
   Case 7
      MessageBox.Show("Good work!")
   Case Is >= 8
```

```
      MessageBox.Show(“Great work!”)
   Case Else
      MessageBox.Show(“Invalid score!”)
End Select
```