Ch4 : Decisions

The If statement
Suppose a program is required to display multiple-choice questions with one correct answer out of three possible choices. For example, one of the questions could be:

A BYTE is the name given to
- (a) Four bits
- (b) Eight bits
- (c) Sixteen bits

Your answer is:___

The program is also required to display the message

Correct - well done!

if the answer is correct, and display a message such as

Sorry, the correct answer is (b)

if the answer provided is incorrect.

The program must therefore be able to take two alternative courses of action depending on the answer supplied. An IF …..THEN statement is one possible way of achieving this requirement.

The appropriate form of the IF statement is illustrated in Example 4-1 which shows the VB code required to display the question above and provide the response appropriate to the letter ' a' , ' b' or ' c' , typed in.

```
Sub Main()
    'This program demonstrates the use of IF to make a decision.


    Const ROGUEVALUE = 0

    'Variables
    Dim Answer As Char

    Console.Clear()
    Console.WriteLine("Enter the letter corresponding to the correct answer")
    Console.WriteLine("for the following question.")
    Console.WriteLine()
    Console.WriteLine("A BYTE is the given name to")
    Console.WriteLine("(a) Four bits")
    Console.WriteLine("(b) Eight bits")
    Console.WriteLine("(a) Sixteen bits")
    Console.WriteLine()
    Console.Write("Enter your answer :")
    Answer = Console.ReadLine()

    If Answer = "b" Then
         Console.WriteLine("Correct -well done")
    Else
         Console.WriteLine("Sorry. The correct answer was (b)")
    End If

    Console.ReadKey()

End Sub
```

The IF statement extending over lines 17 to 21 shows how the program can take one of two possible courses of action depending on the value of a variable. We saw in the last section concerning the use of the **while** statement that logical expressions are either true or false. This is also the case with the logical expression **Answer = ' b'** in the IF statement on line 18. If the letter stored in the character variable Answer is the letter `b', then the logical expression Answer =' b' will be true, otherwise it will be false. If it is true, the statement following the word then is executed (that is, line 17), otherwise the statement after else is executed (that is, line 20).

The general form of the if statement is

```
if   {logical expression}    then
          {statement 1}
else
          {statement 2}
endif
```

Note that {statement 1} is the instruction that is performed if {logical expression) is true;
{statement 2} is performed if {logical expression) is false.

Note also that either {statement 1} or { statement 2), or both of them, can be a block of instructions.

```
if Answer = 'b' then
        Console.Writeline("Correct - well done!")
else
        Console.Writeline("Sorry, the correct answer is (b)")
        Console.Writeline("There are eight bits in a byte'")
end if
```

Sometimes it is necessary to choose between more than just two courses of action in a program. For example, Example 4.2 shows a program which converts a percentage mark to a pass, merit, distinction or fail grade. The program repeatedly accepts marks and converts them to grades until the mark entered is the rogue value -1 (or any negative integer value) signifying the end of the mark inputs. The rules that are used to determine the grade are as follows:

For a distinction the mark must be over 80.
For a merit the mark must be greater than or equal to 60 and less than 80.
For a pass the mark must be greater than or equal to 40 and less than 60.
Below 40 is a fail.

```
1     Sub Main()
2         'This program demonstrates the use of IF…THEN…Else

3         Const DIST = 80
4         Const MERIT = 60
5         Const PASS = 40
6         'Variables
7         Dim Mark As Integer

8         Console.Clear()
9         Console.Write("Please enter the first mark :")
10        Mark = Console.ReadLine()

11        While Mark >= 0
12             If Mark >= DIST Then
13                 Console.WriteLine("Distinction")
14             ElseIf Mark >= MERIT And Mark < DIST Then
15                 Console.WriteLine("Merit")
16             ElseIf Mark >= PASS And Mark < MERIT Then
17                 Console.WriteLine("Pass")
18             Else
19                 Console.WriteLine("Fail")
20             End If

21             Console.Write("Please enter the first mark :")
22             Mark = Console.ReadLine()
23
24        End While

25        Console.ReadKey()

26     End Sub
```

The IF statement between lines 12 and 20 reflects this logic exactly. It is possible to chain IF satements in this way to cope with quite complex lines of reasoning. Added flexibility is provided by the use of the logical **AND** operator used for the logical expressions on lines 14 and 16.

The **AND** operator requires that both of the logical expressions either side of it are true for the complete logical expression to be true. If either or both are false, then the whole expression is false.

A programmer needs to have a good understanding of logical operators – they play a major role in all software development.

Here is a typical output from the program:

```
Please enter the first mark(-1 to end) :46
Pass
Please enter the next mark(-1 to end) :68
Merit
Please enter the next mark(-1 to end) :32
Fail
Please enter the next mark(-1 to end) :83
Distinction
Please enter the next mark(-1 to end) :  -1
```

Logical operators

Logical operators allow you to combine logical expressions. There are three logical operators in VB
and **or** **not**.

An example of the use of the **and** operator was provided in Example 4.2. The **and** and the **or** operators are always placed between two logical expressions, and they each combine these logical expressions to produce a value of true or false.

The Table below shows the rules that are applied by VB to determine whether a compound logical expression is true or false. This type of table is usually called a *truth table.*

Expression 1	Expression 2	Expression1 **AND** Expression2	Expression1 **OR** Expression2
False	False	False	False
False	True	False	True
True	False	False	True
True	True	True	True

Referring back to Example 4.2 in the previous section, on line 14, where the compound logical expression (Mark >= MERIT) and (Mark < DIST) is used to determine whether the mark is equivalent to a merit grade. In the expression, (Mark >= MERIT) is an example of (Expr 1) and (Mark < DIST) is an example of (Expr 2) shown in the table above.

The table below shows how the **and** operator combines these two logical expressions for a number of cases.

Mark	Mark >=MERIT	Mark < DIST	(Mark >=MERIT) AND (Mark < DIST)	WriteLine("MERIT")
45	False	True	False	No
86	True	False	False	No
67	True	True	True	Yes

Thus, **both** logical expressions must be true for the complete expression to be true; with the **or** operator, however, only one of the expressions needs to be true for the complete expression to be true.

As an example, consider example 4.3 below(on the next page)

The program reads some text and counts how many vowels it contains. The program uses a FOR loop to test each letter in turn in the text against each possible vowel. If the current letter is a vowel, that is `"a", "e", "i", "o" or "u", a count is incremented.

The text is held in a string variable called Letters. Each letter in Letters is accessed by specifying its position within the text.

For example, if the text entered was the string `hello there', then Letters(1) is the letter "h", Letters (2) is the letter "e", Letters (3) is the letter "l", and so on.
The for loop control variable, c, starts at 1 and goes up in steps of 1 to the length of the string (11 for the string `hello there').
The length of the string is determined by the predefined VB string method **.length**, on line *11,* which returns the number of characters in its string.

```
    Main()
        'This program demonstrates the use of an if statement to make decisions.

        'Variables
        Dim VowelCount As Integer
        Dim Letters As String
        Dim LengthOfText As Integer
        Dim c As Integer

        'set variables
        VowelCount = 0

        Console.Clear()
        Console.WriteLine("Type texy followed by <enter>")
        Letters = Console.ReadLine()
        LengthOfText = Letters.Length

        For c = 0 To LengthOfText - 1
            If Letters(c) = "a" Or _
                Letters(c) = "e" Or _
                Letters(c) = "i" Or _
                Letters(c) = "o" Or _
                Letters(c) = "u" Then

                VowelCount = VowelCount + 1
            End If
        Next

        Console.WriteLine("There are {0} vowels in the text typed.", VowelCount)

        Console.ReadKey()
    End Sub
```

Here is the output from the program when the string, “the cat sat on the mat” is typed in:

```
Type text followed by ENTER:
the cat sat on the mat

There are 6 vowels in the text typed
```

Note that the program will only work with lower-case text. The reason is that lower-case letters “a”, “b”, “c”, etc are represented in a computer using a different set of codes from the equivalent upper-case letters `A', `B', 'C', etc.

The third logical operator is the **not** operator which simply reverses the logical value of a logical expression. Thus, the logical expression not (x > 3) is true only when x is less than or equal to 3. Similarly, the logical expression not (Balance <= 0) is true only when Balance has a value that is greater than zero. The truth table shown in Table 4.3 defines the operation of the not logical operator.

Exercises

1) Write a program to accept the marks from 3 tests, calculate the average mark and then awards a grade according to the following criteria:

85 and above	Grade A
70 to 85	Grade B
55 to 70	Grade C
45 to 55	Grade D
35 to 45	Grade E
below 35	Fail

2) The following program is a simple guessing game. It includes some new elements which are described below.

a) Enter the code and run the program.
b) Write a Code Analysis for the program and make sure you understand how it works.

```
Dim RandomGenerator As New System.Random
```

System.Random

System = include various prewritten functions/ subroutines & methods.
Random =psuedo random number generator.

```
RandNum = RandomGenerator.Next(1, 10)
```

RandomGenerator.Next(1, 10)

Generates a random number between 1 (lowest) & 10 (highest).
This value is placed into the variable RandNum.

```
Console.SetCursorPosition(x, y)
```

Moves the cursor to a specific screen position.
Assume the screen has 80 positions **across** the X-axis and 50 positions **down** the y-axis such that the top left corner of the screen is position (0, 0) and the bottom right corner is position (24, 79). Therefore gotoxy(40, 25) will position the cursor 40 positions across and 12 down the screen (approximately the centre of the screen).

```
    Sub Main()
         'This program plays a simple number guessing game.

         Dim RandNum As Integer
         Dim RandomGenerator As New System.Random
         Dim UserGuess As Integer

         Console.Clear()
         RandNum = RandomGenerator.Next(1, 10)
         'Console.WriteLine("{0}", RandNum)

         Console.SetCursorPosition(20, 12)
         Console.Write("Enter Your Guess  :")
         UserGuess = Console.ReadLine()
         While UserGuess <> RandNum
             If UserGuess <> RandNum Then
                 Console.SetCursorPosition(20, 14)
                 Console.WriteLine("{0} is wrong. try again.", UserGuess)
             End If
             Console.SetCursorPosition(20, 12)
             Console.Write("Enter Your Guess  :")
             UserGuess = Console.ReadLine()

         End While
         Console.SetCursorPosition(20, 14)
         Console.WriteLine("{0} is Correct! well done", UserGuess)

         Console.ReadKey()

     End Sub
```

c) Modify the guessing game program above.

i) to give clues to the user such as “too big” or “too small” as appropriate.
ii) to allow only 5 guesses before telling the user the answer

d) Write a Code Analysis for the modified program.