Conditional Statements

If . . . Then Statements

If . . . Then Statements allow you to test for a certain condition to be true. **If** that condition is true, the program code will do whatever comes after the command **Then**. The conditional operators are:

=	**equal to**
>	**greater than**
<	**less than**
>=	**greater than or equal to**
<=	**less than or equal to**
<>	**not equal to**

There are several ways to write **If . . . Then** statements.

Some Examples

Single-line Form

```
If Mark > 49 Then lblPass.visible = True

If Mark  >=75 AND Mark <101 Then lblHonors.visible = True

If Mark < 0 OR Mark > 100 Then lblMark.Text = "This is not a valid mark."
```

AND requires **both** conditions to be true.

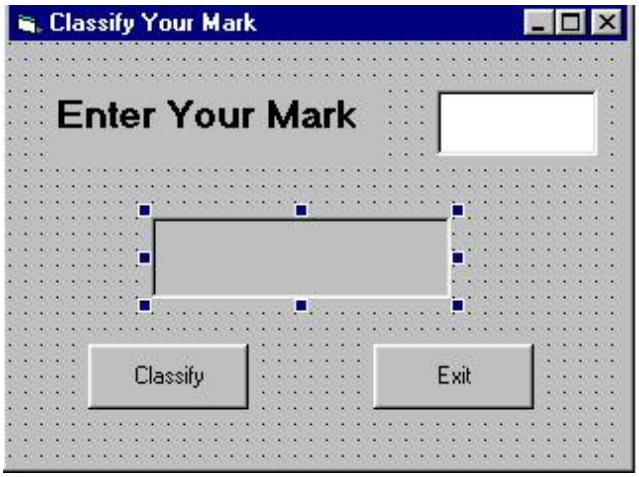
OR requires **one** or **the other** of the conditions to be true.

Block Form—This is the preferred format for Visual Basic

```
If Mark> 49 Then
        lblPass.visible=True
End If

If Mark>74 Then
        lblHonours.visible = True
ElseIf Mark < 50 Then
        lblFail.visible = True
Else
        lblPass.visible = True
End If
```

An Example Program using Conditional Statements



Step 1—Design a form similar to the one above:

You should have 2 labels (for the words written and to display the answer in the middle), 1 textbox (to enter the mark), and 2 command buttons.

Step 2—Set the Properties as follows:

Form	Name Text	frmMark Classify Your Mark
Label1	Text Font	Enter Your Mark Bold, 14
TextBox	Name Text	txtMark (delete text)
Label2	Name Text Autosize Borderstyle Font	lblType (delete text) True 1—Fixed Single Bold, 24
Command1	Name Text	btnClassify Classify
Command2	Name Text	btnExit Exit

Step 3—Write the Code

```
Private Sub cmdClassify_Click()
   Dim Mark As Integer
   Mark = Val(txtMark.Text)
   If Mark >= 75 Then
      lblType.ForeColor = color.blue
      lblType.Text = "Honours"
   ElseIf Mark > 49 And Mark < 75 Then
      lblType.ForeColor = color.green
      lblType.Text = "Pass"
   Else
      lblType.ForeColor = color.red
      lblType.Text = "Fail"
   End If
   txtMark.Text = ""
   txtMark.Focus
End Sub

Private Sub cmdExit_Click()
   Me.Close
End Sub
```

Run Your Project—Try entering marks that fit each of the classifications.

Add a Clear Button—Add a Clear command button and set the properties and write the code similar to the last project.

Save Your Project—Create a folder called **Marks** within your folder. Save the form and project files in this folder.

Explanation of Code

- The first two lines declare the variable *Mark* and obtain its value from the textbox.
- Pay particular attention to the If…Then section of code. If the first condition is met, the 2 lines of code that come after the **Then** are carried out.
- If the mark is greater than 49 and also less than 75, the 2 lines of code after that statement are carried out.
- If any number is entered that does not meet either of those conditions, then the code under **Else** is carried out.
- An **End If** statement is required to state that the If…Then section of code is finished.
- *txtMark.Text =””* clears the textbox.

- *txtMark. Focus* sets the focus to the textbox. In other words, the cursor will be blinking in that textbox ready for you to enter another mark.
- An If…Then section of code could contain many *ElseIf* statements, but the last one should always be just *Else* as that is the last possibility left so the condition does not have to be stated.
- *ElseIf Mark > 49* ***And*** *Mark < 75 Then*
  requires that both conditions be true in order for the code to be carried out because of the ***And***.
- A statement such as *If Number >100* ***Or*** *Number < 0 Then* would test for one condition or the other to be met; thus, this line of code would check to see if a number is not within the normal Mark range of 0 to 100.

Additional Ways to Control Focus

Once a value is entered in a textbox, a user should be able to move on to the next event in any of three ways. These include:

- Pressing the Tab key
- Clicking on the next box
- Pressing the Enter key

To allow users to press the tab key, you need to change a property in the Properties window. You will change this property for *txtAge*, cmdClassify, and cmdClear as follows:

- Select **txtAge**. Set the TabIndex property to 0.
- Select **btnClassify**. Set the TabIndex property to 1.
- Select **btnClear**. Set the TabIndex property to 2.

To allow users to hit the enter key after typing a number in the textbox**,** you need to enter this code in the **Keypress** event of **txtAge**.

```
If KeyAscii = 13 Then
        btnClassify. Focus
End If
```

13 is the ASCII code for the Enter key. Therefore, if this is hit, focus shifts to btnClassify.

PROGRAMMING PROBLEMS

1. Create a program that will enable you to enter any number. The computer will then display TRUE in a label if the number is between –30 and +30 inclusive or FALSE if it is not. Hint: You will need to treat the words "TRUE" and "FALSE" as **strings**. String values must be enclosed in quotes.

2. **The Cereal Program**.
   This program will tell you whether you have to buy cereal if you are getting low. Use 2 textboxes, 1 for the number of bowls eaten, and 1 for the number of boxes bought. You can eat 10 bowls for every 1 box. Assume you will eat the same numbers of bowls next week as this week.

   You should have 3 command buttons: 1 for Check Supply, 1 for Clear, and 1 for Quit. At any time, if the user clicks Check Supply, a message will appear either "BUY MORE" or "YOU HAVE ENOUGH".

3. **The Movie Theatre Problem**
   Create a program that uses a textbox to enter a person's age. If the person is 12 or older, they pay the adult fee at the movie theatre, $6. If the person is younger than 12, the fee is $4. Display the label **Adult Fee $6** or **Child Fee $4** based on the value entered.

   Use the Visible property to turn the labels on and off. use a button with the caption **Test** to test the age. Include a **Clear** button to clear the textbox and turn off the labels.

Divisibility

Visual Basic provides an easy way to tell whether one number is evenly divisible by another, and if not to tell what the remainder is. The **Mod** operator divides one number by another and returns just the remainder of that division as its result.

Eg.
5 Mod 3 is 2
21 Mod 7 is 0
25 Mod 2 is 1
24 Mod 5 is 4

When the **Mod** operator returns 0, the first number is evenly divisible by the second.

Eg.

```
Remainder = Number Mod 10
If Reminder = 0 Then
        lblAnswer.Text = "The number is evenly divisible by 10."
Else
        lblAnswer.Text = " The remainder is " Remainder
End If
```

4. Create a program to allow you to enter a number. The computer will then display whether that number is EVEN or ODD. Hint: What is it about a number that makes it even?

5. Create a payroll application that calculates gross weekly wages (before taxes) given the hours worked and the hourly rate. Then modify the program so that you can choose by way of option buttons whether the employee is exempt from taxes or not. If not exempt, deduct 18% for taxes. If exempt, deduct no taxes.

6. The Printing Place has different printing prices based on the number of copies to be printed:

0 – 499 copies	$0.30 per copy
500 – 749 copies	$0.28 per copy
750 – 999 copies	$0.27 per copy
1000 copies or more	$0.25 per copy

Create a Printing Prices application that asks the user for the number of copies and then calculates the total price.

7. Create a Sandwich Order application that allows the user to generate a sandwich order that includes the size of the sandwich (small or large) and the fixings for the sandwich (lettuce, tomato, onion, mustard, mayonnaise, cheese). A small sandwich is $2.50 and a large sandwich is $4.00. Mustard and mayonnaise are free, lettuce and onion are $0.10 each, tomato is $0.25, and cheese if $0.50. Use option buttons for the size and checkboxes for the fixings. Calculate the total price of the sandwich.

8. Speedy Overnight Delivery service does not accept packages heavier than 27 kilograms or larger than 0.1 cubic meters (100,000 cubic centimeters). Create a Package Check application that asks the user to input the weight of a package and its length, width, and height in centimeters, and displays an appropriate message if the package does not meet the requirements (eg. Too large, too heavy, or both).

Simple Menus

To create a menu, first select the Form, then hit the menu Editor button. In the Caption box, enter what you want the menu item to show, inserting **&** before the letter you wish to underline.

eg. **&Calculate**

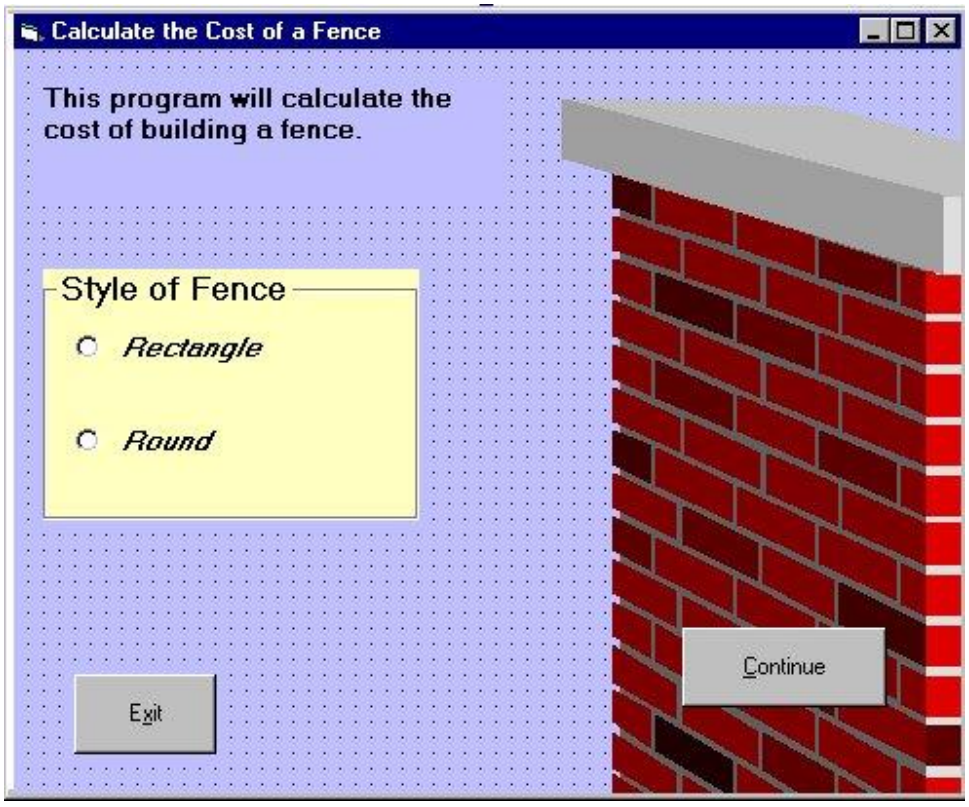
Hitting Alt and the underlined character will activate the menu. You can also do this when captioning command buttons.

Menu items are named with the prefix **mnu**. eg. **mnuCalculate**

After entering the caption and name for the first one, hit next. Continue in the same manner until finished, and then click OK.

The Garden Fencing Problem—Multiple Forms.

3 Forms	frmTitle	(shown below)
	frmRect	(on next page)
	frmRound	(not shown—make similar to frmRect)

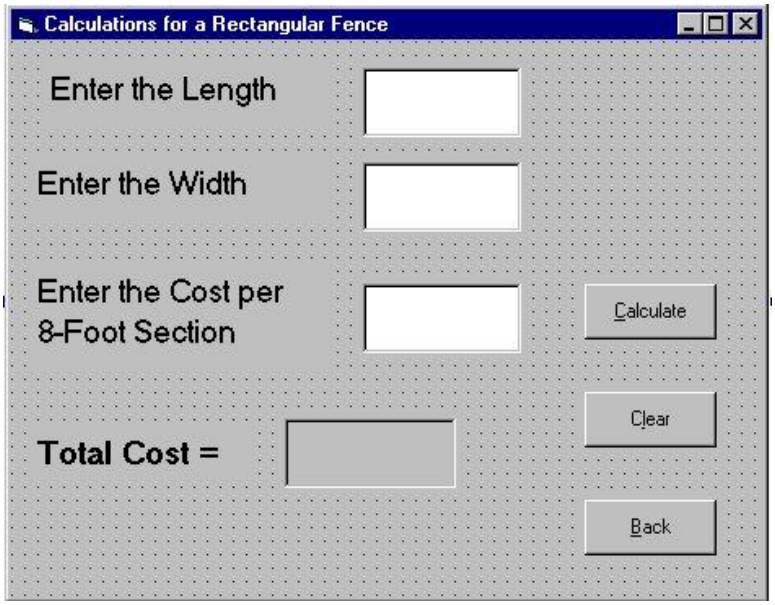


Set the **Name** property of the form to **frmTitle**.

The picture is placed in an imagebox. To do this, set the Picture property of the imagebox. You should find graphics files on the internet.

The Option buttons must be placed in a Group Box. First draw a group box and Set the Text property to **Style of Fence**. Then click and drag 2 option buttons inside the Group Box.
Note: You cannot double click to put an object inside a frame. Name the first one **radRect** with the Text of **Rectangle**. Name the second one **radRound** with the Text of **Round**.

Create two new forms using the add form button. Name one **frmRound** and the other **frmRect**. You navigate between with forms with the **show** and **hide** methods as shown in the code for the command buttons. For example, the Continue button should take you to either frmRect or frmRound depending on which option button you have checked.
Me.Hide()
frmRound.show() frmRect.show()



FrmRect - 1

```
Private Sub cmdBack_Click()
   frmRect.Hide
   frmTitle.Show
End Sub

Private Sub cmdCalculate_Click()
Dim Length As Single, Width As Single, Cost As Single
       Dim Answer As Single
       Length = Val(txtLength)
       Width = Val(txtWidth)
       Cost = Val(txtCost)
       Answer = (Length * 2) + (Width * 2) * Cost / 8
       lblAnswer = Answer
       cmdClear.SetFocus
End Sub

Private Sub cmdClear_Click()
   txtLength = ""
   txtWidth = ""
   txtCost = ""
   lblAnswer = ""
   txtLength.SetFocus
   End Sub

Private Sub txtCost_KeyPress(KeyAscii As Integer)
   If KeyAscii = 13 Then
      cmdCalculate.SetFocus
```

```
   End If

End Sub

Private Sub txtLength_KeyPress(KeyAscii As Integer)
   If KeyAscii = 13 Then
      txtWidth.SetFocus
   End If
End Sub

Private Sub txtWidth_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
      txtCost.SetFocus
   End If
End Sub
```

FrmTitle – 1

```
Private Sub cmdContinue_Click()
   If optRound.Value = True Then
      frmTitle.Hide
      frmRound.Show
   Else
      frmTitle.Hide
      frmRect.Show
   End If

End Sub

Private Sub cmdExit_Click()
   End
End Sub
```

You are to design a form for **frmRound** and write the appropriate code. Think about what values the user would have to enter in order for the program to be able to calculate the answer.

More Visual Basic Tips

Comments (also known as Remarks) – are notes that explain the purpose of statements or sections of code. Although remarks are part of an application's code, they are ignored by the compiler. They are intended for the programmer or others who might read the application's code.

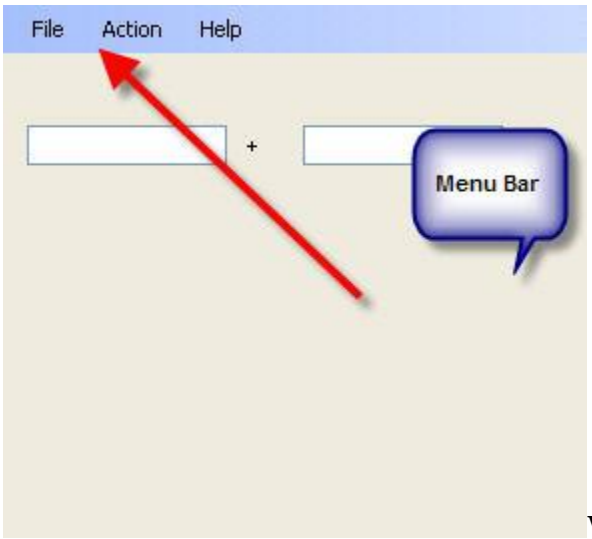
A comment must begin with either an apostrophe (') or the REM keyword. When a program runs, the computer ignores comments. You should always add descriptive comments to your code.

An example of comments:
'Convert the values in the box to number,
'and calculate the gross pay.

All programs should contain the following information as a comment in the code:

- Name
- Date
- Course
- Title
- Description

Creating Menus



When creating a menu we use the menustrip control. Type all the Names and submenu names into the box that say Type Here.

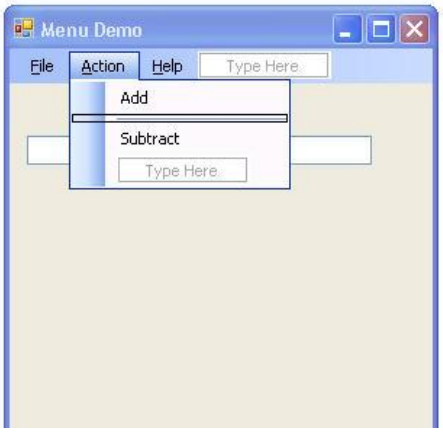
Shortcut Keys

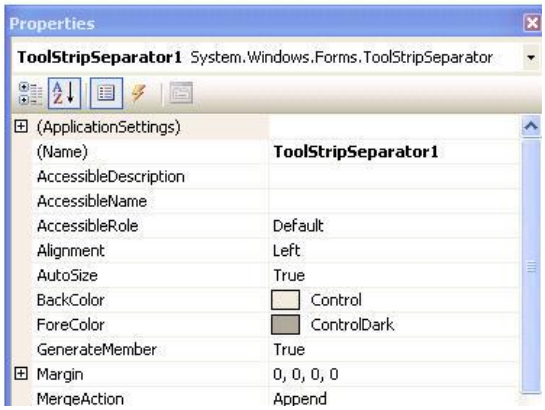
These are the keys you need to press in order for an action to occur. Shortcut keys usually use the alt, ctrl or shift in combination with a letter.

Some of the more common shortcut keys are:

Ctrl + S	**Save**
Ctrl + P	**Print**
Ctrl + C	**Copy**
Ctrl + X	**Cut**
Ctrl + V or Shift + Insert	**Paste**

Separator Bars





A separator bar is a horizontal bar used to separate groups of commands on a menu.

You can insert a separator bar into a menu in either of the following way:

- Right click an existing menu item. On the pop-up menu that appears, select Insert, and then select Separator. A separator bar will be inserted above the menu item.
- Type a hyphen (-) as a menu item's Text property.

Here is the code used in the background of the buttons:

```
Public Class frmMenu

    Private Sub ExitToolStripMenuItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ExitToolStripMenuItem.Click
        Me.Close()

    End Sub

    Private Sub AboutToolStripMenuItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles AboutToolStripMenuItem.Click
        Me.Hide()
        frmAbout.Show()
```

```
    End Sub

    Private Sub AddToolStripMenuItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles AddToolStripMenuItem.Click
        Dim Number1 As Single
        Dim Number2 As Single
        Dim Answer As Single
        Number1 = Val(txtNumber1.Text)
        Number2 = Val(txtNumber2.Text)
        Answer = Number1 + Number2
        lblAnswer.Text = Answer

    End Sub

    Private Sub HelpToolStripMenuItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles HelpToolStripMenuItem.Click

    End Sub
   End Class
```