Chapter Five: Visual Basic Statement

5.1 Visual basic statement

1-End statement

The syntax is

End

Example:-

```
Private sub command1-click()

  End
End sub
```

If an End statement is executed, the program terminates.

2-Visual Basic Looping

Looping is done with the Do/Loop format. Loops are used for operations are to be repeated some number of times. The loop repeats until some specified condition at the beginning or end of the loop is met.

2-1 Do While/Loop

Example:-

```
Counter=1

Do While Counter <= 1000

Debug.Print Counter
Counter = Counter + 1

Loop
```

This loop repeats as long as (While) the variable Counter is less than or equal to 1000. Note a Do While/Loop structure will not execute even once if the While condition is violated (False) the first time through. Also note the Debug.Print statement. What this does is print the value Counter in the Visual Basic Debug window. We'll learn more about this window later in the course.

2-2 Do Until/Loop

Example:-

Counter = 1

Do Until Counter > 1000

Counter = Counter + 1

Loop

This loop repeats Until the Counter variable exceeds 1000. Note a Do Until/Loop structure will not be entered if the Until condition is already True on the first encounter.

2-3 Do/Loop While

Example:-

Sum = 1

Do

Sum = Sum + 3

Loop While Sum <= 50

This loop repeats While the Variable Sum is less than or equal to 50. Note, since the While check is at the end of the loop, a Do/Loop While structure is always executed at least once.

2-4 Do/Loop Until

Example:-

Sum = 1

Do

Sum = Sum + 3

Loop Until Sum > 50

This loop repeats Until Sum is greater than 50. And, like the previous example, a Do/Loop Until structure always executes at least once.

2-5 For....Next Loop

The format is:

For counter=startNumber to endNumber (Step increment)

One or more VB statements

Next

Example:-

(a) **For** counter=1 to 10

display.Text=counter

Next

(b) **For** counter=1 to 1000 step 10

counter=counter+1

Next

(c) **For** counter=1000 to 5 step -5

counter=counter-10

Next

Note:-

1. Make sure you can always get out of a loop! Infinite loops are never nice. If you get into one, try **Ctrl+Break.**
2. The statement **Exit Do** will get you out of a loop and transfer program control to the statement following the Loop statement.

3- Visual Basic Branching - If Statements

Branching statements are used to cause certain actions within a program if a certain condition is met.

3.1 The If/Then statement

The format of If/then:

If comparison Test **Then**

One or more Visual Basic statements

End If

If a comparison test is true, the body of the If statement executes

3.2 If/Then/Else/End If blocks:

If Balance - Check < 0 **Then**

Print "You are overdrawn"

Print "Authorities have been notified"

Else

Balance = Balance - Check

End If

Here, the same two lines are printed if you are overdrawn (Balance - Check < 0), but, if you are not overdrawn (Else), your new Balance is computed.

3·3 Or, we can add the ElseIf statement:

If Balance - Check < 0 Then

Print "You are overdrawn"

Print "Authorities have been notified"

ElseIf Balance - Check = 0 Then

Print "Whew! You barely made it"

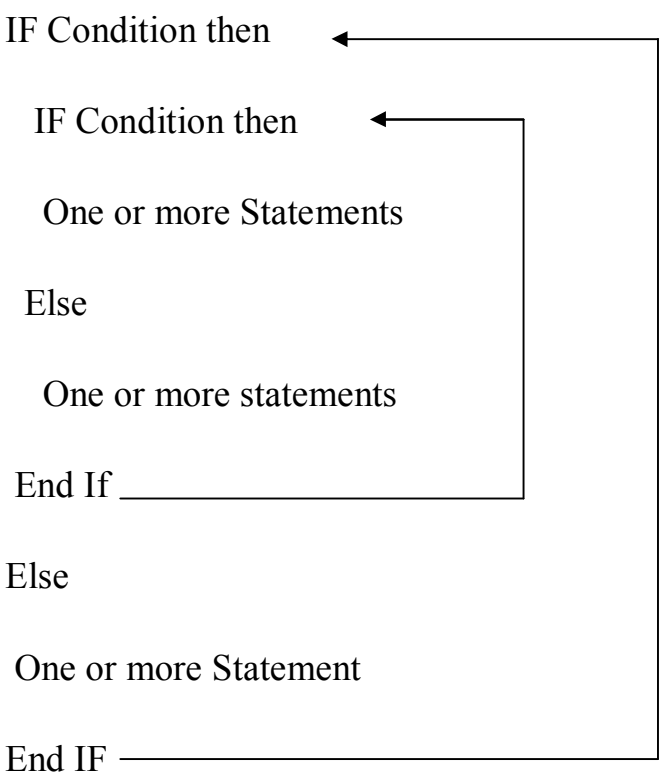
Balance = 0

Else

Balance = Balance - Check

End If

3.4 Nested if



Comparison Operators

All the comparison operators produce true or false results. In other words, the comparison is either true or the comparison is false. The mathematical operators produce numeric values, whereas the comparison operators produce true or false values. See table 1

Operator	Meaning
=	Equal to
>	More than
<	Less Than
>=	More than and equal
<=	Less than and equal
<>	Not Equal to

Logical Operators

In addition to comparison operators, there are a few logical operators which offer added power to the VB programs. Logical operators let you combine two or more comparison tests into a single compound comparison. There are shown in Table.2.

Operator	Meaning
And	Both sides must be true
or	One side or other must be true
Not	Negates truth

3.5 Using If.....Then.....Else Statements with Operators

Operator	Usage	Description
And	If (A > B) And (C < D)	Produces True if both sides of the And are true. Therefore, A must be greater than B and C must be less than D. Otherwise, the expression produces a false result.
Or	If (A > B) Or (C < D)	Produces True if either side of the Or is true Therefore, A must be greater than B or C must be less than D. If both sides of the Or are false, the entire expression produces a false result.
Not	If Not(strAns = "Yes")	Produces the opposite true or false result. Therefore, if strAns holds "Yes", the Not turns the true result to false

.

Example:-

```
If (sngSales > 5000.00) Then
  If (intUnitsSold > 10000) Then
    sngBonus = 50.00
End If
```

Here is the same code rewritten as a single If. It is easier to read and to change later if you need to update the program:

```
If (sngSales > 5000.00) And (intUnitsSold > 10000) Then
  sngBonus = 50.00
End If
```

How can you rewrite this If to pay the bonus if the salesperson sells either more than $5,000 in sales or if the salesperson sells more than 10,000 units? Here is the code:

```
If (sngSales > 5000.00) Or (intUnitsSold > 10000) Then
  sngBonus = 50.00
End If
```

Example:-

```
If A=5 Then
If B = 7 Then
Text1.text = "odd no"
End if
Else
text1.text = "the no . is not 5 "
End if
```

3.6 Select Case - Another Way to Branch

If is great for data comparisons in cases where one or two comparison tests must be made. When you must test against more than two conditions, however, If becomes difficult to maintain.

Visual Basic supports a statement, called Select Case that handles such multiple-choice conditions better than If-Else. Here is the format of the standard Select Case statement:

```
Select Case Expression
  Case value
    One or more Visual Basic statements
  Case value
    One or more Visual Basic statements
  Case value
    One or more Visual Basic statements
  Case Else
    One or more Visual Basic statements
End Select
```

Example:-

Consider the If statement shown in Listing 7.3. Although the logic of the If statement is simple, the coding is extremely difficult to follow.

```
If (intAge = 5) Then
  lblTitle.Caption = "Kindergarten"
Else
  If (intAge = 6) Then
    lblTitle.Caption = "1st Grade"
  Else
```

```
    If (intAge = 7) Then
      lblTitle.Caption = "2nd Grade"
    Else
      If (intAge = 8) Then
        lblTitle.Caption = "3rd Grade"
      Else
        If (intAge = 9) Then
          lblTitle.Caption = "4th Grade"
        Else
          If (intAge = 10) Then
            lblTitle.Caption = "5th Grade"
          Else
            If (intAge = 11) Then
              lblTitle.Caption = "6th Grade"
            Else
              lblTitle.Caption = "Advanced"
            End If
          End If
        End If
      End If
    End If
  End If
End If
```

The corresponding code with Select Case would be:

```
Select Case intAge
   Case 5:  lblTitle.Caption = "Kindergarten"
   Case 6:  lblTitle.Caption = "1st Grade"
   Case 7:  lblTitle.Caption = "2nd Grade"
   Case 8:  lblTitle.Caption = "3rd Grade"
   Case 9:  lblTitle.Caption = "4th Grade"
   Case 10: lblTitle.Caption = "5th Grade"
   Case 11: lblTitle.Caption = "6th Grade"
   Case Else:  lblTitle.Caption = "Advanced"
End Select
```

The two additional formats differ only slightly from the standard Select Case Here is the first additional format:

```
Select Case Expression
   Case Is Relation:
     One or more Visual Basic statements
   Case Is Relation:
     One or more Visual Basic statements
    Case Is Relation:
     One or more Visual Basic statements]
    Case Else:
     One or more Visual Basic statements]
End Select
```

Relation can be whatever comparison test you want to perform against Expression at the top of the Select Case. The standard Select Case statement, discussed in the previous section, compared the Expression value against an exact Case match. When you use the comparison Is Select Case option, each Case can be matched on a comparison test.

Here is the format of the second extra Select Case format:

Select Case Expression
  Case expr1 **To** expr2:
    One or more Visual Basic statements
  Case expr1 **To** expr2:
    One or more Visual Basic statements
  Case expr1 **To** expr2:
    One or more Visual Basic statements]
  Case Else:
    One or more Visual Basic statements]
End

The Case lines require a range, such as 4 To 6. The To Select Case option enables you to **Select** match against a range instead of a relation or an exact match

Example:-

Write the program to enter the age of student and then assigns a student's grade and school name to the label on the form depend on the his/her age .The code checks make sure that the student is not too young to be going to school. Attach this code to the command buttons and use select case to determine the age of the students.

1- Draw the user interface that consists of two labels and one command.
2- Set the properties of these objects.

Label1	
Name	lblTitle
Appearance	0-flat
Label2	
Name	lblSchool
Appearance	0-flat
Command1	
Name	ok
Caption	start
Attach the code to the ok –click	

```
Private sub ok –click ()
 Dim intage as integer
  Intage=inputbox(“enter the age of student” )
Select Case intAge
   ` Check for too young...
   Case Is <5:  lblTitle.caption = "Too young"
   ` Five-year olds are next assigned
   Case 5: lblTitle.caption= "Kindergarten"
   ` Six to eleven...
   Case 6 To 11:  lblTitle.caption= "Elementary"
                  lblSchool.caption= "Lincoln"
   ` Twelve to fifteen...
   Case 12 To 15: lblTitle.caption = "Intermediate"
                  lblSchool.caption= "Washington"
   ` Sixteen to eighteen
   Case 16 To 18: lblTitle.caption = "High School"
                  lblSchool.caption = "Betsy Ross"
   ` Everyone else must go to college
   Case Else: lblTitle.caption = "College"
```

```
            lblSchool.caption = "University"
    End Select
End sub
```

If the age is less than 5, the title label becomes Too young, and the school name remains blank. If the age is exactly 5 (intAge is obviously an integer value), the title gets Kindergarten, and the school name still remains blank. Only if the child is 5 or older do both the title and school name get initialized.