2.6.1 Decision Structure - If

Visual Basic Procedures can test condition and then, depending on the results of the test, perform different operations. The decision structures that VB supports include,

- If..Then
- If...Then...Else
- Select Case

If...Then

We can use either single line syntax or multiple line block syntax.

```
If <condition> Then <statements>
If <condition> Then
        <statements>
End If
```

If...Then...Else

```
If <condition1> Then
        [statement block 1]
[ElseIf <condition2> Then
        [statement block 2]...
[Else
        [Statement block n]]
End If
```

Example :

```
Dim marks as integer
Marks = val(text1.text)
If marks > 90 then
```

```
        Text2.text = "Excellent"
    ElseIf marks > 80 then
        Text2.text = "very good"
    ElseIf marks > 70 then
        Text2.text = "Good"
    Else
        Text2.text = "Average"
    End If
```

2.6.2 Decision Structure – Select Case

Visual Basic provides the Select Case structure as an alternative to IF..Then...ElseIf for selectively executing one block of statements from among multiple blocks of statements. A select case statement provides capability similar to the If...Then...Else but it makes code more efficient and readable.

```
Select Case Text expression
        [ Case expression list 1
                [ statement block 1 ]]
        [ Case expression list 2
                [ statement block 2 ]]
        ...
        ...
        ...
        [ Case Else
                [ statement block n ]]
End Select
```

Each expression list is a list of one or more values. If there is more than one value in a single list, the values are separated by commas. Each statement block contains zero or more statements. If more than one case matches the text expression only the statement block associated with the first matching case will execute. Visual Basic executes statements in the Case Else clause (which is optional), if none of the values in the expression list matches the text expression.

Example :

```
Dim marks as integer
Marks = val(text1.text)
Select case marks
        Case Is  > 90
                Text2.text = "Excellent"
        Case Is  > 80
                Text2.text = "very good"
        Case Is > 70 then
                Text2.text = "Good"
        Case Else
                Text2.text = "Average"
End Select
```

2.6.3 Loop Structure – Do While...Loop

Loop structures allow us to execute one or more lines of code repetitively. The loop structures that VB supports include,

- Do...Loop
- For...Next

Do Loop is used to execute a block of statements an indefinite number of times.

There are several variations of the Do Loop statement. But each evaluates a numeric condition to determine whether to continue execution. As with If..Then, the condition must be a value or expression that evaluates to False (zero) or to True (nonzero).

In the following DO Loop, the statements execute as long as condition is True.

```
DO While <condition>
    statements
Loop
```

When VB executes this DO Loop, it first tests the condition. If condition is false (zero), it skips past all statements. If it is true (nonzero), VB executes the statements and then goes back to the DO While statement and tests the condition again. The statements are not executed at all if the condition is initially false.

Example :

```
Dim k as integer
k=5
Do while k < 10          ' condition is true as value of k is less than 10
    Text1.text = k
    k = k + 1
Loop
```

2.6.4 Loop Structure – Do...Loop While

Another variation of the DO ... Loop statement executes the statements first and then tests condition after each execution. This variation guarantees at least one execution of statements.

```
DO
    statements
Loop While <condition>
```

Example :

```
Dim k as integer
K=10
Do
 Text1.text = k
 K=k+1
Loop While k < 10
```

The loop is executed once even though the condition is false.

Two other variations are analogous to the previous two, except that they loop as long as the condition is false rather than true.

```
DO until <condition>
        statements          loop zero or more times.
Loop
DO
        statements          loop atleast once.
Loop Until <condition>
```

2.6.5 Loop Structure – For...Next

This loop structure is used to execute the statements a specific number of times. For loop uses a counter variable that increases or decreases in value during each repetition of the loop. The syntax is :

```
For [counter] = Start To      End [step increment]
        statements
Next [counter]
```

The arguments Counter, Start and End are all integers.

Note: *The increment argument can be either positive or negative. If increment is positive, Start must be less than or equal to End otherwise the statements in the loop will not execute. If increment is negative, Start must be greater than or equal to End for the body of the loop to execute. If the STEP is not set, then increment defaults to one.*

Example:

```
Dim k as integer
For k = 1 to 10
    Text1.text = k
Next k
```

2.6.5 Loop Structure – For Each...Next

This loop structure is used to repeat a group of statements for each element in an array or collection.

Array : An array is a set of sequentially indexed elements having the same type of data. Each element of an array has a unique identifying index number also referred to as a subscript. If you make a change to one element of an array it does not affect the other elements.

Collection :A collection is an object that contains a set of related objects. An object's position in the collection can change whenever a change occurs in the collection; therefore, the position of any specific object in the collection may vary.

Example: To count the number of textboxes that are present on a form. TextBox belongs to the Control collection.

```
Dim Ctrl as control
Dim Ctr as integer
```

```
For EACH ctrl in Form1
        If typeOf ctrl is Textbox then
            Ctr = ctr + 1
        End if
Next
Msgbox ("Total number of textboxes " & ctr)
```

Note : Some examples, *If Typeof ctrl is commandButton*

If Typeof ctrl is Checkbox

2.7 EXIT STATEMENT

Exiting the Control Structure : The Exit statement allows us to exit directly fr a FOR loop, DO Loop, Sub procedure, or Function procedure. The syntax for t Exit statement is simple.

Exit For can appear as many times as needed inside a FOR Loop, and Exit D can appear as many times as needed inside a DO Loop.

Exiting a Sub or Function Procedure : The syntax of Exit Sub and Ex Function are similar to that of Exit For and Exit Do. Exit Sub can appear as man times as needed, anywhere within the body of a Sub Procedure. Exit functio can appear as many times as needed anywhere within the body of a Functio Procedure.

2.8 CONTROL ARRAY

A control array is an array of controls. A control array of 5 text boxes will have the same name. Each checkbox in the array will be uniquely identified by an index value starting from 0.