Unit-II

Basic Programming Fundamentals:

Variables:

To declare variables to store values either at design time or at run time. Beside this variable are also used to store values returned by function, calculated result, intermediate data.

Now to remember while declaring a variable,

1. A variable must begin with an alphabet.
2. It may contain any mixture of alphabet and numbers.
3. The variable can contain more than 255 characters.
4. The variable should be unique within the same procedure.
5. It cannot contain space. It can use underscore to separate the words of the variable for example First_name.

In Visual Basic can declare variables in two ways i.e. explicitly and implicitly. When to declare the variables explicitly have to use the DIM statement to declare each variables that are going to use in the program. Again, can declare variables implicitly also without mentioning the DIM statement.

Variables are used to store information to be referenced and manipulated in a computer program. They also provide a way of labeling data with a descriptive name, so our programs can be understood more clearly by the reader and ourselves. It is helpful to think of variables as containers that hold information.

Declaring a Variable

A variable is declared with the Dim statement, followed by the name for the variable:

Syntax,

Dim variable name [As Type]

Where, variable name is the name of the variable and type is the datatype of the variable. Variable declared with the Dim statement within a procedure exist only as long as the procedure is being executed.

There are many ways of declaring variables in Visual Basic depends on where the variables are declared and how they are declared, we can determine how they can be used by our application. The different ways of declaring variable in visual basic are as below.

1. Implicit declaration
2. Explicit declaration

1.Implicit Declaration

Declaring a variable tells Visual Basic to reserve space in memory. it is not a must that a variable should be declared before using it.; automatically whenever Visual Basic encounters a new variable, it assigns the default variable type and value. This is called implicit declaration. In implicit declaration, there is no need to declare a variable before using it. The new variables type is variant, the generic datatype that can accommodate all other datatypes.

2. Explicit Declaration

To declare a variable, use the Dim statement followed by the variable name and type as follows

Dim meter As Integer

```
Dim Greeting As String
```

Scope of Variables

A variable is scoped to a procedure level or module level variable depending on how it is declared. The scope of a variable, procedure or object determines which parts of the code in our application are aware of the variable's existence. A variable is declared in general declaration section of a form, and hence is available to all the procedures. Local variables are recognized only in the procedure in which they are declared. They can be declared with Dim and Static keywords. If we want a variable to be available to all of the procedures with the same module or to all the procedures in an application, a variable is declared with broader scope.

Scope defines the visibility of a variable. There are three types of scope for variable in Visual Basic

1) Public / Global Scope

Global variable are available anywhere in your program, any line of code in any procedure can read or write the value of variable. To create a global variable declare it in the declaration section of a standard module using the global or **Public** keyword. When declare a variable as Public, it can be accessed from anywhere of the application. The syntax for declaring a Public variable given below,

Public<variable name>As<datatype>

Example : Public Salary As Integer

2) Private/ Module Scope

Module level variable are available to any code within the module where they are declared. Module level variables allow you to share data between procedures without exposing that data to every procedure in the application. To create a module level variable, declare it in the declaration section of a module using either the Dim or **Private** keyword. When declare a variable as Private, it can be accessed only within that procedure or module, where it is declared. The syntax for declaring a Private variable given below

Private <variable name>As <datatype>

Example: Private Name As String *20

3) Local Scope

Local variable are only available to the procedure in which they are created. Local variable are the most restricted in scope. Not even other procedure in the same module may read or modify local variables. You create local variables by declaring them with Dim or Static keyword within the body of a sub function.

Declaration statement

1) Dim

Declares the variable and allocates the storage. The variable declared at the module level is available to all the procedure within the module. At the procedure level the variable available only within the procedure.

```
Dim x as Integer
    Private sub command1_Click ()
        x = 100
        Msgbox x
    End sub
    Private sub command2_click ()
        x = x + 200
        Msgbox x
    End sub
```

2) Static

Used at the procedure level at declared the variable and to allocate the storage space. the variable declared with the static statement return their values as long as the code is running.

Variable by declaring it as Static with procedure has stopped executing, Static variables retain their value.

Static<variable name>As<datatype>

Example: Static Total As

```
    Private sub command1_Click()
        Dim y as Integer
        y = y + 100
        Msgbox y
    End sub
    Private sub command2_click( )
        Static y as Integer
        y = y + 200
        Msgbox y

        End sub
```

Operators:

An operator may be defined as a symbol, which performs the specific operations. The data on which operator operate is called as the operant.

The type of operators on basis of operations they are,

1. Arithmetic operator
2. Comparison operator
3. Logical operator
4. Miscellaneous operator

1)Arithmetic operator

This operator is created for perform addition, subtraction, multiplication, division, exponential, modulus, string connectors.

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
^	Exponential
mod	Modulus
&	String connect

2) Comparison operator

Comparison operator are use to compare data. For ex, <, >, =, >=, <=, <>

Operator	Description
=	Equality and assignment
>	Use to find out greater number
<	Use to find out lowest number
>=	Find out value is greater or equal to
<=	Find out value is less or equal to
<>	Find out not equal to value

3) Logical operator

Logical operator is use to connect more than one condition. Generally, And, Or, Not. Logical operators are use.

And

If all given condition are true then output will be true if any condition goes in false then output will be false.

C1	C2	Output
True	False	False
False	True	False
True	True	True
False	False	False

Or

In Or operator, if any one condition is true. Output will be goes false when all condition are false.

C1	C2	Output
True	False	True
False	True	True
True	True	True
False	False	False

Not

Not operator are use to get reverse result.

C1	Output
True	False
False	True

4. Miscellaneous operator

„$“ any single character
„#“ any single digit

„*“ zero, one or more than one character
[set] any character in set
[!set] any character except the character in the set

Precedence of Operator

When several operations occur in an expression, each part is evaluated and resolved in a predetermined order called operator precedence. Visual Basic 6 has well defined rules for specifying the order in which the operators in an expression are evaluated when the expression has several operators. Precedence rules can be changed if there is parenthesis in an expression because parenthesis have highest precedence. Eg. A = 6 + 5 * 8

Can be solved as
A = (6+5) * 8 = 88
Or
A = 6 + (5*8) = 46

Visual Basic calculate such expression according to the priority given to the operators that is called operator precedence.
The order of the precedence for all the operators,

1. Exponentiation (^)
2. Negation (-)
3. Multiplication and division (*,/)
4. Integer division (\)

5. Modulus operator (mod)
6. Addition / concatenation and subtraction (+,-)
7. String concatenation (&)
8. All comparison operation (=,<>,<,<=,>,>=) have equal precedence.

Control Structure

In Visual Basic application the control flows from left to right and from top to bottom of the code. There are two types of Control Structures

1. Decision Structures
2. Loop Structures

Decision Structure:

An application needs a built-in capability to contest conditions and take a different course of action depending on the outcome of the test. Visual Basic provides three decision structures.

1. If ----- Then
2. If ----- Then -----Else
3. Select case

1. If ---- then

The If--- Then structure tests the condition specified and, if it's true, executes the statements that follow. The if structure can have a single line or multiple line syntax, to execute one statement conditionally use the single lien syntax and to execute the multiple line statement separate them with colon.

Syntax, for single Line: - If condition Then
statement

Syntax, for multiple Line: - If condition Then Statements
End If

The condition is usually a comparison, but it can be any expression that evaluates to a numeric value also, Visual Basic evaluates this value as true or false. A zero numeric value is false, and any non-zero numeric value is considered True.

If condition is true, Visual Basic executes all the statements following the Then keyword we can use either a single line or multiple line syntax to execute just one statement conditionally.

2. If-then –Else

The variation of the If—Then statement is the If—Then --- Else statement which executes one block of statements if the condition is true and another if the condition is false.

The syntax,

If Condition Then

Statement block 1

Else

Statement block 2

End if

Visual Basic evaluates the condition. If it's true, it execute the first block of statements and then jumps to the statement following the End if statements. If the condition is false. Visual Basic ignores the first block of statements and executes the block following the else keyword.

Another variation of the If -- Then --- Else statement is If ---- Then ---- Else if statement uses several conditions with the Else if Keyword.

```
If condition 1 Then
Statement block 1
Elseif condition 2 Then
Statement block 2
ElseIf condition 3 Then
Statement block 3
Else
Statement block 4
End If
```

You can have any number of else clauses. The condition are evaluated from the top and if one of them is True. The corresponding block of statement is executed. The Else clause will be executed if none of the previous expression are True.

Ex: - Find the given no is even or odd

```
Private sub command1_Click ()
Dim i As Integer

i = Text1.text
If i mod 2 = 0 Then

Msgbox (Number is even)
Else
Msgbox (Number is Odd)

End if
End sub
```

Ex: - Find the greater number from the two number

```
Private sub command1_Click ()
Dim a As Integer, b As Integer
a = Inputbox (Enter value for a‖)
b = Inputbox (Enter value for b‖)
If a > b Then
Msgbox (a is greater &a)
Else
Msgbox (b is greater &b)
End if
End sub
```

Ex: - Find the grade of student from the following information

Marks	Grade
Less than 50	Fail
Greater than 50 and less than 75	Pass
Greater than 75 and less than 90	Very good
Greater than 90 and less equal 100	Excellent

Select Case:

In the previous section we have learned how to use if--- else if control structure to control the program flow. Now we will learn another way to control the program flow that is the select case control structure. In some situation we need to select option among multiple. In such situation we can use if – else statement. but it may increase the complexity of the program and the programmer may get confused.

Because programmer need to write lot many nested if statement which will be complicated to read and modify the code. In such situation it is better to use multi way decision making statement i.e. select case statement, select case is more convenient to use than the if --- else --- End if

Syntax,

```
Select case expression
Case value 1
        Statement 1
Case value 2
        Statement 2

Case value 3
        Statement 3
Case Else
        Statement
End select
```

Where,

1) The select case is the Visual Basic keyword. The expression after the select case statement is called select condition when the program starts executing the value of expression is compared with the case value statement one by one. It compare with first case statements and then moves to next till the match found. If the match is found the program execute the code written in statement. The value of expression can be string or number. But if the expression is string the value of case statement must be string or if it is number the value of case statements is also number i.e. data type of expression and case value must be same. You can write as many case statements as you with. There is no limit on the number of case statement.

2) Statement is the valid Visual Basic statement. If any of the case is match with the value of select expression program execute statement of that particular case value.

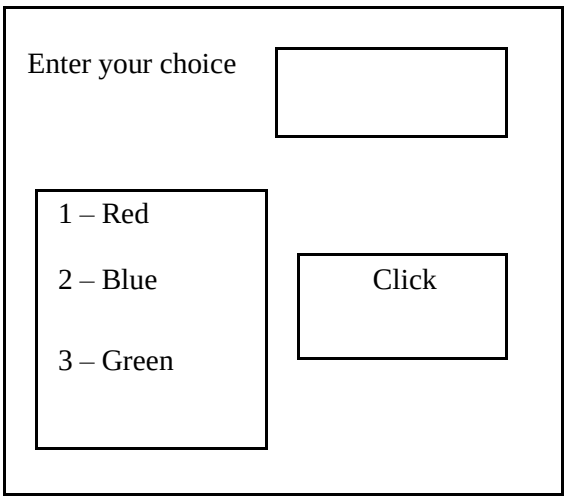
3) You can write multiple values with case. All the values should be separated with commas. Eg. case 10,20,30.

4) You can also give the range of values, while giving range the start and end value must be connected with To‘ keyword. Eg. case 10 to 100

5) You can also write logical expression with case. Use _Is ‘keyword with case and then specify the logical expression. Eg. case Is <> 100

6) The case else part is optional. If match could not found among available case values then case else is executed.

7) Select case is easy to use for menu entries.



Goal: - Use of select case

```
Dim c As Integer
Private sub cmdclick_Click() c =
val(Text1.Text) Select case c
Case 1
Form1.Backcolor = VBRed
Case 2
Form1.Backcolor = VBBlue
Case 3
        Form1.Backcolor = VBGreen Case Else
Msgbox (Wrong Selection)
End Select
End sub
```

Goal : - WAP to calculate given operation Addition, multiplication, division, subtraction

```
Private sub Command1_Click ()
        Dim a As Integer, b As Integer
        Dim c as String
        a = Inputbox (Enter first value)
        b = Inputbox (Enter second value)
        c = Inputbox (Enter operator)
    select case c
        Case — +
                Msgbox a + b
        Case ---  -
                Msgbox a - b
        Case — *
                Msgbox a * b
        Case — /
                Msgbox a / b
        Case else
                Msgbox (can‘t understands)
    End select
End sub
```

Loop structure

Loop statement allow you to execute one or more lines of code repetitively, may task consist of trivial operations that must be repeated over and over again and looping structure are an important part of any programming language. Visual Basic support the following loop statement.

1. Do ---- while loop
2. For ---- next
3. While ---- Wend

1) Do ---- loop

The Do ---- loop execute a block of statement for as a condition is true Visual Basic evaluate an expression, and if it is true. The statement are executed if the expression is False. The program continues and the statement following the loop is executed.

There are two variation of Do --- loop statement and both use the same basic model. A loop can be executed either while the condition is True or Until the condition becomes True. These two variations use the keyword _While 'and Until'to specify how long the statement are executed. to execute a block of statement while condition is True.

Syntax,

```
Do While <condition>
        Statement block
Loop
```

Goal:- Print the 1 to 10 numbers on form

```
Private sub command 1_Click ( )
    Dim i As Integer
    i = 1
    Do While i < 10
        Print i
        i = i  + 1
    Loop
End sub
```

To execute a block of statement until the condition becomes True use the following

Syntax,

```
Do Until condition
        Statement block
Loop
```

When Visual Basic execute the previous loops it first evaluates condition. If condition is false. The Do --- While or Do --- Until is skipped the statement are not even executed once. when the loop statement is reached Visual Basic evaluated the expression again and repeat the statement block of the Do --- While loop if the expression is True or Repeated the statement of the Do – Until loop if the expression is False.

The Do --- loop can execute any number of times as long as condition is True. If the condition is initially False the statement may never execute.

Goal: - Print the 1 to 10 numbers on form

```
Private sub command 1_Click ()
    Dim i As Integer
    i = 1
    Do Until i = 10
        Print i
        i = i + 1
    Loop
End sub
```

Another variation of Do – loop execute the statement first and evaluated the condition after each execution. The statement in this type of loop execute at least once, since the condition is examined at the end of the loop.

This Do – loop has the following syntax,

Syntax,

Do	Do
Statement	Statement
Loop While condition	Loop Until condition

2) For ---- Next

The For --- Next loop is one of the oldest loop structures in programming language. Unlike the Do - - loop. The For --- Next loop requires that you know how many times the statement in the loop will execute. The For ---- Next loop uses a variable it's called loop counter that increases or decreases in value during each repetition of the loop. The For ---- Next loop has the following syntax,

Syntax,

```
For counter = start To End [step Increase / Decrease]
    Statement
Next counter
```

The keywords in the square bracket are optional. The argument counter, start, end and increase or decrease are all numeric. The loop is executed as many times as required for the counter to reach the end value.

In executing a For --- Next loop Visual Basic complete the following steps,

1) Set counter equal to start
2) Test to see if counter is greater than end. If so, it exits the loop. If increment is negative. Visual Basic test to see if counter is less than end. If it is it exits the loop.
3) Execute the statement in the block
4) Increment counter by amount specified with the increment argument. If the increment argument isn't specified counter is incremented by 1
5) Repeat the statement.

Eg.

```
For i = 0 To 10
    Print i
Next i
```

The single most important thing to keep in mind when working with For --- Next loop is that the loop counter is set at the beginning of the loop. Changing the value of the end variable in the loop body won't have any effect for example

The following loop will be executed 10 times, not 100

```
times endvalue = 10
For i = 0 To endvalue

    endvalue = 100
    print i;
next i
```

following is an example of an endless or infinite loop

```
For i =0 To 10
```

```
        Print i
        i = i  - 1
    Next
```

This loop never ends because the loop counter. In effect is never increased (if you try this press control + break to interrupt the endless loop)

The increment argument can be either positive or negative. If start is greater than end. The value of increment must be negative. If not, the loop body won‘t be executed not even once.

3) While ---- Wend

The while --- wend loop executes a block of statement while a condition True. The While --- Wend loop has the following syntax,

While condition
    Statement
Wend

If condition is true, all the statement are executed and when Wend statement is reached, control is returned to the while statement which evaluated condition again if condition is still True, the process is repeated. If condition is False, the program resumes with the statement following the wend statement.

The following While --- Wend loop prompts the user for numeric data, the user can type a negative value to indicate that all value is entered.

```
Number = 0
While Number > = 0
    Total = Total + Number
    Number = Inputbox (Please Enter value)
Wend
```

You assign the value 0 to the Number variable before the loop starts because this value can‘t affect the Total.

4) With --- End with

Execute a series of statements making repeated reference to a single object or structure, with -- end with allows you to perform a series of statements on a specified object without prequalifying the name of the object. If the qualification path to the object is long, using with --- End with can improve your performance. A with block also reduce repetitive typing of the qualification path and the risk of mistyping one of its elements.

Syntax,

With object
    [statement]
End with

This concept can be clearly under stood with the following example,

```
Text1.font. size = 16                    with Text1
Text1.font. italic = True                    .font. size = 16
Text1.height = 235                           .font. italic = True
Text1.Text = —Welcome‖                       . Height = 235

. Text = —Welcome‖
End with
```

Arrays:

Array is of group of elements of some datatype referred by a same name. each element of array identifies by subscript or index number value. In Visual Basic there are two type of array.

1. Fixed array
2. Dynamic array or Run time array.

When we need to store name of 10 student in this case declaring 10 different variables could not be very practical. In such a situation you can make use of array. Array can be declared using Dim statement followed by name and the subscript value in a parenthesis.

Syntax,

Dim Variable name (subscript) As datatype

Eg. Dim x(5) As Integer

When declared array need to specify the lower and upper bound for the subscript.

Upper Bound and Lower Bound (UBound, LBound)

Lbound and UBound are build in function which return the lower and upper bounds of an array with the help of these functions. We can sort the arrays.

Eg. Dim intarray (20) as Integer

Print Ubound(intarray)

Print Lbound(intarray)

In the above example the function Ubound and Lbound will return 20 and 0 respectively. The behavior of array subscript can be changed by using the option base statement. This statement, in the declaration section of a module, causes all array in that module to begin with the index number specified along with the statement, instead of 0. By default the option base setting for a module is 0. For ex, option base 1 statement changes the index of an array to start with 1.

Visual Basic allows us to change the lower and upper bound at time of declaration.

For eg,

Dim intarraycount (15 to 35) as integer

The lower bound in the above case is 15 and the upper bound is 35.

1) Fixed Array

A fixed size array which always remain the same size it can not be change at run time. The lower limit is specified consider to be zero (0) the upper bound cannot exceed the range of the long data type. This array declaration can be appearing in the declaration section of a module.

Dim counter (1 to 15) As Integer

Dim sum(25) As Double

The first declaration Index number of counter range from 1 to 15 and the element number of 15. In second declaration subscript is 25 it means it can hold 26 elements from 0 to 25.

2) Dynamic Array / Run time array

The dynamic array can be used if it is not decide how many elements the array is going to have. Dynamic array give the ability of changing the size of the array at runtime. You can declare the array as dynamic by giving it can empty dimension place.

Syntax,

Dim array_name ( ) as Datatype

Eg.

Dim a ( ) As Integer

You can also allocate the actual number of element with Redim statement can appear only a procedure. Redim array_name(subscript) As Integer

Unlike the Dim and Static statement Redim is an executed by statement it may the application carry out an action at run time.

Preserve Keyword

Some time we want to change the size of array without losing the data into array you can do this by using Redim with preserve keyword

Eg. as enlarge an array by element without losing the value of the existing element used the Ubound function to refer to the upper bound.

Redim preserve Array name(size)

Multidimensional Array

Multidimensional arrays allow us to keep track of related information in an array. for example, to keep track of telephone numbers and registered users, we can use a two-dimensional array to store the values. With Visual Basic we can declare arrays of multiple dimensions, for example, the following statement declares a two dimensional 10 by 10 array with in a procedure

Dim dblarray (9,9) As Integeror

Dim dblarray (1 to 10, 1 to 10) As Integer

Control Arrays:

In Visual Basic, a control array is a group of related controls in a Visual Basic form that share the same event handlers. Control arrays are always single-dimensional arrays, and controls can be added or deleted from control arrays at runtime. One application of control arrays is to hold menu items, as the shared event handler can be used for code common to all of the menu items in the control array.

Control arrays are a convenient way to handle groups of controls that perform a similar function. All of the events available to the single control are still available to the array of controls, the only difference being an argument indicating the index of the selected array element is passed to the event.

A control array is a group of controls that share the same name and type, they also share the same event procedures. To add controls based on the result of a run time condition Control array is used. A control array has at least one element and can grow to as many elements as our system resources and memory permit. Its size also depends on how much memory and windows resources each control requires. The maximum index we can use in a control array is 32767. Element of the same control array have their own property settings. Common uses for control arrays include menu controls and option button groupings.

All the elements in a Control Array share the code, which have to write in the event procedure of the first element of the array. Except three properties (Visible, Index and TabIndex) members of

Control Array share all the properties of the lowest element of the array. A Control Array has a minimum of one element and you can add unlimited number of elements.

All the elements of the array are identified with its unique index number.

Control Array in three different methods. They are

1. Copy an instance of the control and paste it on the Form.
2. Assign similar names for multiple controls.
3. Set non-null Index number for a control.

Control Array using the first method.

1. Start with a new standard EXE project.
2. In the Form add a Command button. You can create Control Array with any other controls according to your choice.
3. Select the Command button and click on copy. After copying, click on Paste from the Standard Toolbar.
4. A message gets displayed asking you whether you want to create a Control Array of Command1.
5. Click on Yes to create the Control Array. Another Command button gets added in the Form and both controls have the same name Command1.
6. All the properties of this control are identical with that of the copied control except the index property. The original control has the index property 0 and the new control has the index property 1.

Control array can be declared at the following time.

1. Design time
2. Run time

1) Creating a control array at design time

a. Assign the same name to more than one control.
b. Copying an existing control and then paste it on to the form.
c. Set control's index property to a value that is not null.

a) To add a control array element by changing its name do this

1. Draw a control we want to be in the control array.
2. Select one of the controls and change its name. setting to name setting for the first element in the array.
3. When we type an existing name for a control in the array, v displays a dialog box asking us to confirm that we want to create a control array. choose yes to confirm the action.

b) To add a control array element by copying an existing control do this.

a) Draw a control in the control array.
b) While the control has the focus, choose copy from the Edit menu.
c) From the edit menu, choose paste.

Visual Basic displays a dialog box as asking us to confirm that we want to create a control array. choose yes to confirm the action.

This control is assigned an index value of 1. The first control we drew has a value 0. The index value of each new array element corresponds to the order in which the element was added to the control array. when controls are added this way, most of the visual properties, such as height, width, and color are copied from the first control in the control array to the new controls.

2) Creating control array at Runtime

We can add and remove controls in a control array at runtime using the Load and Unload statements. However, the control to be added must be an element of an existing control array. we must

have created a control at design time with the index property set in most cases to 0 then, at run time use the following syntax,

Load object (index %)

Unload object (index %)

Object:- Name of the control to add to or delete from the control array.

Index %: - The control's index value in the array.

When we load a new element of a control array, most of the property setting are copied from the lowest existing element in the array. The visible, index and tab index property setting are not automatically copied to new elements of a control array, so to make the newly added control visible, we must set its visible property to True.

Working with Procedure, Function

Procedure in Visual Basic

Procedure:

A procedure is a group of statement which can be executed from the other part of program or can be associated with the certain event. Procedure are useful for contains repeated or shared task procedure allow you to break your program into different logical unit which debug more easily than a program without procedure we can call the procedure for other programs. Usually when little or no modification. Procedure will not be executed till it is called.

The scope of procedure may be private or public. Private indicate that the procedure is accessible only to the procedure in modules where it is declare. Public indicate that the sub procedure accessible in all modules.

Procedures are building blocks that consists a set of logically related statements and execution of these statements performs an action. There are three types of procedures provided by Visual Basics 6.0 They are,

- **Event**
- **Property**
- **General**

Event:

This procedure is mainly defined for forms and controls and it gets created automatically either by system action or by a user's action.

For example, you need to write a procedure in the click event of a command button. So by double clicking on the control, you can invoke the procedure and that procedure will execute when the user will click on that button.

Property:

Property procedure mainly used to create properties for a class and can set its value from any application that creates an instance of this class. Property procedure is a procedure mentioned inside the

class module and whenever the client retrieves or sets the value of a property it is called. It can be defined for Form modules and for Standard modules also.

Again, there are three types of property procedures. They are

Get: This is used for retrieving the property value.

Set: This is used for setting reference for the object-type property of the class.

Let: This is used for assigning new values.

General: General procedures are of two types. They are

Function: - Function begins with Function and ends with End Function. It returns the value to the calling procedure. While creating, if don't specify the data type, variant data type
is assigned to the returned value.

The syntax is,

[Public | Private] [Static] Function<name>

[(arglist)][As type]

[statements]

[name=expression]

End Function

Sub-procedures:

Sub Procedure executes the specified set of statements together but it does not return value. It is enclosed with Sub and End Sub Statements. The syntax is,

[Public|Private][Static] Sub <name>[(arglist)]

[statements]

End Sub

Methods for passing arguments to a procedure

1. To create application must know the methods used for passing arguments to a procedure.
2. It can pass arguments by value or by reference to a procedure. It passing an argument by value then have to use Byval keyword in the procedure declaration.
3. Moreover, when are passing an argument by value, pass a copy of the variable, so the changes of variable are reflected only within the called procedure.

Syntax:

[Public |Private][Static][Function name]

(Byval arg1, Byval arg2,…) As<datatype>

When are passing an argument then have to use the ByRef keyword. Moreover, by using ByRef keyword, can access the contents of the variable stored in its original memory address location. If the procedure changes value of the variable, the changed value is available when the procedure is called.

The syntax is,

[Public | Private][Static] [Function_name]

(ByRef arg1, ByRef arg2…) As<datatype>

To simplify the specification of arguments when are calling a procedure, two more arguments are used. They are Named arguments and Optional arguments.

Named arguments: - In general when are passing arguments to a procedure specify the arguments in the same order, as have done while procedure declaration. While doing this might do any mistake and to prevent that should use the named arguments so that can specify arguments in any order.

Optional arguments: Specify the argument optional, by using the Optional keyword so that can omit the passing of argument to the procedure while calling it.

A variable can be declared in the declaration section rather than with in procedure. This makes the variable available to all the procedure in the module. The two types of procedure are,

1. Subroutine, 2. Function

1) Subroutine

A subroutine is a block of statement that carries out a well-defined task. The block of statement is place with a pair of sub----- End sub statement and can be invoke by name. A subroutine normally performs a more complicated task. All the event procedures in Visual Basic are coded as subroutine. Eg.

```
sub showdate ( )
    Msgbox date ( )
  End sub
Private sub command1_click ( )
    Showdate
End sub
```

Subroutine and Event Handler

An event handler is a short segment of code that is executed each time an external condition is triggers the event. Every application contains a number of event handlers which contain to react to user action. Event handler need not return any result and they are implemented as subroutine. They are automatically activated by Visual Basic in response to external events.

Sub-procedures: Sub-procedure executes the specified set of statements together but it does not return a value. It is enclosed with Sub and End statements.

The syntax is,

[Public| Private][static] sub<name>[(arglist)]

[statements]

End Sub

2) Function

A function is similar to a subroutine but a function return a result. Subroutine perform a task and don‘t report anything to the calling program. Functions commonly carry out calculations and report the result. The statement that make up a function are place in pair of **function ----end function** statement more over a function reports a result it must have a type.

To create function in Visual Basic performs following procedure.

Tools

Add procedure

Name

Type (function)

Scope (public / private)

Syntax,

```
Public Function Function_name () as Return Type
    Statement (s)
End Function
```

Calling Procedure

To call procedure you use the call statement and supply its name and arguments in parenthesis as follows.

```
Call Procedure name (var1, var2, --, var n)
```

Calling Subroutine

You can omit the call statement and call the subroutine by name. The argument are supplied without the parenthesis as follows,

```
Subroutine Name var1, var2,---, var n
```

The number of arguments you supply to the subroutine and their types must match those in the procedure declaration.

Calling Function

Function are called by name and a list of arguments follows the name in parentheses as follows, Variable = Function Name (var1, var2, --- , var n)

Methods for passing arguments to a Procedure

To create application must know what are the methods used for passing arguments to procedure.

Arguments

Most procedures accept arguments from the calling program. Recall that an argument is a value you pass to the procedure and on which the procedure usually acts. This is how subroutine and function communicate with the rest of the application.

```
Dim a As Integer, b As Integer
Private Sub Command1_Click()
  a = Val (Text1.Text)
  b = Val (Text2.Text)
  Call add (a, b)
  subtract a, b
  Text3.Text = mult(a, b)
End Sub
Public Sub add (n As Integer, m As Integer)
  Dim p As Integer
```

```
  p = n + m
  MsgBox "answer addition =" & p
End Sub
Sub subtract (p As Integer, q As Integer)
  Dim r As Integer
  r = p - q
  MsgBox "answer subtract =" & r
End Sub
Public Function mult(s As Integer, t As Integer) As Integer
  mult = s * t
End Function
```

Argument passing mechanism

One of the most important procedure issues is the mechanism used to pass argument. the most of examples use the default mechanism, passing arguments by reference. The other mechanism is passing by value. Although most programmers use the default mechanism.

1) Passing Arguments by reference

Passing arguments by reference gives procedure access to the actual variables. The calling procedure passes the address of the variable in the memory so that the procedure can change its value permanently. In earlier version of BASIC and Visual Basic this was the only argument passing mechanism.

```
Function add (num1 as integer, num as Integer) As Integer
  Add = num1 + num2
      Num1 = 0
      Num2 = 0
End function
Private Sub Command1_click ( )
      Dim a as integer
      Dim b as integer
      a = 10
      b = 2
      sum = Add(a,b)
      print a
      print b
      print sum
End sub
```

The code displays the following result,

```
0
0
12
```

The changes made to the functions arguments take effect even after the function has ended. The value of variable a and b has change permanently.

2) Passing Arguments by value

When you pass an arguments by value the procedure see only a copy of the argument. Even if the procedure changes it the changes aren‘t permanent. The benefit of passing arguments by value is that the argument values are isolated from the procedure and only the program in which they are declared can change their values. Passing arguments by value requires a bit of extra typing, since this isn‘t the default arguments passing mechanism.

```
Function add (num1 as integer, num as Integer) As Integer
  Add = num1 + num2
      Num1 = 0
      Num2 = 0
End function
Private Sub Command1_click ( )
      Dim a as integer
      Dim b as integer
      a = 10
      b = 2
      sum = Add(a,b)
      print a
      print b
      print sum
End sub
```

The code displays the following result,

10

2

12

The function has changed the values of the arguments but these changes remain in effect only in the function.

The variable a and b in the command1_click ( ) subroutine haven‘t been affected.

The differences between passing an argument by value and by reference

By Value	By Reference
A separate copy of that variable is passed as an argument	A reference of a variable is passed to the procedure
The variable value in the calling and the called procedure is stored at the separate location	The variable value in the calling and the called procedure is stored at the same location

Any changes made in the called procedure are not updated in the calling procedure	Any changes made in the called procedure are updated in the calling procedure
Consumption of memory space is more	Consumption of space is less.

Optional Key word

You can specify arguments to a procedure as optional by placing the optional keyword. In the arguments lists. If you can specify an option argument all subsequent arguments the argument list must be optional and declared with optional keyword.

Dim a As Integer, b As Integer

Private Sub Command1_Click ( )

a = Inputbox (Enter any no:-—)

b = Inputbox (Enter any no:-—)

Call add()

End sub

Public sub add(n1 As Integer, optional n2 As Integer)

Dim n3 As Integer

n3 = n1 + n2

msgbox n3

End sub

We can give default value to optional variable as per our requirements.

Eg, Public sub add (n1 As Integer, optional n2 As Integer=50)

Functions

Visual Basic provides different types of functions. They are as follows

- User-interaction Functions
- String Functions
- Math Functions
- Date Functions
- Conversion Functions

User-Interaction Functions:

In Visual Basic can interact with the user with InputBox and with MsgBox functions. These functions displays dialog boxes through can display some information and can collect the input also.

InputBox Function:

InputBox display a dialog containing a textbox and two buttons OK and Cancel. user inputs the text in the textbox and click on OK to close the dialog. Remember the text provided by the user in the stored as string variable . study the syntax of this function.

InputBox(Prompt[,title][,default][,xpos][,ypos][helpfile, context])

The Prompt, title and default parameters contain string expressions. For example, you want to collect from the user the password .so you have to specify,

Pass = InputBox ("Enter your password "," Password")

The input provided by the user is collected in Pass variable.

MsgBox Function –

MsgBox Function is used to display message boxes. Message boxes are used in the applications to provide different types of message like warning message, confirmation message, error message etc. or just for displaying information. Moreover it is also used to collect input from the user and according to that action is performed depending upon the button the user had clicked. For example a message box contain three buttons Yes, No and Cancel. It is asking the user whether to save the application or not. The syntax of MsgBox function is ,

MsgBox (Prompt [, buttons][, title] [, helpfile, context])

See the code of an example, which displays the message box in the click event of a button and according to the selection of the button the work is performed .

Visual Basic Built-in Function

Many built in functions are offered by Visual Basic that fall under various categories. These functions are procedure that return a value. The functions fall into the following basic categories.

1. Data and Time function
2. Text strings function

1) Date and Time function

Date and time are internally stored as numbers in Visual Basic. The whole number portion of number represents the date between 1 January, 1000 and December 31, 9999 inclusive. The decimal portion represents the time between 00:00:00 and 23:59:59 hours inclusive. The system's current date and time can be retrieved using the Now and Date and Time function in Visual Basic.

Function	Syntax	Description
Date	Date ()	Returns the current system date.
Now	Now ()	Returns the current date and time according our computer 's system date and time
Day	Day ()	Returns a whole number between 1 and 31, inclusive representing the day of the month
Month	Month(date)	Returns a whole number between 1 and 12 inclusive representing the month of the year

Minute	Minute (time)	Returns a whole number between 0 and 59 inclusive representing the minute of the hour.
Hour	Hour(Time)	Returns a whole number between 0 and 23, inclusive representing the hours of the day
Date Add	Date Add (interval, number, date)	Add the number in the given interval in the given date Date Add (—m‖,1, —10-Jan-2012‖) Output: - 10-Feb-2012
Date Diff	DateDiff (interval, date1, date2)	Returns a variant specifying then specifying then number of time intervals between two specified dates .

2) Text Strings Functions

Visual Basic string functions can manipulate strings in applications. The following table gives a brief description about the string functions in Visual Basic

1.String – This function returns string value that contains character string of specified length .

The syntax is,

String(<Number>,<Character>)

For eg.

a = string(5,"H")

The output of the function will be HHHHH

Remember the return string is built with the first character of the string expression.

For eg.

A string(5, "Hello")

This time also the output will be HHHHH.

2.Ltrim: This function is used to remove the leading spaces. It returns the string after removing the leading space.

The syntax is, Ltrim("String")

Eg. A= "Hellow"

T = Ltrim(A)

In the output you will find the leading space are removed.

3. Rtrim: This function is used to remove the leading spaces it returns the string after removing the leading space.

Syntax, Rtrim("String")

Eg. A= "Hello"

T= Rtrim(A)

In the output you will find that the trailing spaces are removing.

4. Trim : This function does the both job of removing the leading and trailing space and then return the string.

Syntax, Trim("String")
Eg. A= "Hello"
T=Trim(A)
In the output you will find that the leading and trailing spaces are removed.

5) Strcomp: This function compares between the string and return an integer that indicates the result .the return values of StrComp function are,

When String1 is less than String 2 then – 1 is returned.

When both the string are equal then 0 is returned.

When String1 is greater that String2 then 1 is returned.

Incase any of these string is null the value returned is also null.

Eg.

A = "HELLO"

B = "hello"

C = Strcomp(A,B,1)

StrComp function will return 0.

6)InStr: This function is used to return the position of the first occurrence of one string within another.

Syntax, InStr([Start,] string1,string2 [,compare])

Eg. S="Hello" H="I"

C=InStr(1,S,H, vbBinaryCompare)

In output the function will return 3.

7)Mid: This function is used to returns specified number of characters from the specified position of the string Syntax, MID ("String", Start, Length)

Eg.MID

(—ABCD,2,3) o/p :-

BCD

8)Lcase: This function is used to returns a string that has been convert into lower case.

Syntax, Lcase("String")

Eg.Lcase(—HelloWorld1234‖)

o/p :- helloworld1234

9)Ucase: This function is used to returns a string that has been convert into uppercase. Syntax, Ucase("String")

Eg.Ucase(—HelloWorld1‖)

o/p :-

HELLOWORLD1234

10)Asc: This function is used to return one ASCII number for the first Letter in the String. Syntax, Asc("String")

Eg. Asc(—A‖) O/P:- 65

11) Chr: This function is used to return the character associated with the specified ASCII number Syntax, Chr(charcode)

Eg.Chr(97) O/P :- a

12)Left: This function is used to return a specified number of from the left side of the string. Syntax, Left("String", Length)

Eg.Left(ABCD,2) O/P :- AB

13)Right: This function is used to return specified number of characters from the right side of the string Syntax, Right ("String", Length)

Eg. Right (ABCD,2) o/p :- CD

14)Len: This function is used to find out a length of entered string.

Syntax, Len("String")

Eg. Len(Hello)

o/p :- 5

Math functions

Visual basic provides for the programmer's math function also.

1.ABS: This function returns absolute value, which is nether negative nor positive. The data type returned will be of similar type that was passed. for ex. if the data type passed is integer, it returns integer only. The syntax is

ABS(number)

For example, absolute value of 0 is 0.

A=ABS(0)

Consider another example

B=ABS(-10)

Absolute value of -10 will be 10.

EXP: In this function the value of e, which is approximately 2.718282, raised to the power of a number and that is passed as an argument. The number value that you are passing should not exceed beyond 709.782712893. otherwise, error occurs. The syntax is,

Exp(number)

For example E=Exp(3) The output is 20.085553692318

Val: This function returns numeric value, which represents the number specified in a string .

The syntax is

Val (Expression)

C=Val ("90.76")

The function returns in output 90.76.

Log-

The function returns logarithm i.e. logarithm to the base e, of the specified number. The function returns the output as Double data type .

The syntax is,

Log(number)

L=Log (10) The output is 2.302585092994

Sqr:

This function returns the square root of the specified number. the function returns the output as Double data type. The syntax is **Sqr(number).**

For example

L=Sqr(4) The output 2 is returned.

Fix:

This function is used to strip the fractional part of the number that is passed and returns the remaining integer part. but when negative numbers are passed it returns the first negative integer that might be greater or equal to the number that was passed to it.

The syntax is

F=Fix(14.54)

F=Fix(-14.54)

Output of first example is 14 and second one will return -14.

Int:

Int function also does the same work like Fix function but when negative numbers are passed it returns the first negative number that might be less or equal to the number that was passed to it.

The syntax is:

Int(number)

F=Int(14.54)

F=Int(-14.54)

Output of first example is 14 and second one will return -15.

Sign:

This function determines the sign of the specified number. It returns integer value and there are three possibilities. when the number is greater than 0, it returns 1. When the number is 0, a 0 is returned. Again, if the number is less than 0, -1 is returned. The syntax is

Sign(number)

S=Sign(4) It is greater than 0,so the output of the function is 1.

Trigonometric Functions:

Beside these functions you will find different types of trigonometric functions that you can use in your applications according to your requirements mainly they are used while drawing, in creation of graph etc. They are

1.Atn(number)

This function returns arctangent of an angle.

For example. A=Atn(90)

This function will return 1.559685672897

2.Cos(number)- Returns cosine of an angle

C=Cos(90)

The function will return -0.448073616

3.Sin(number) returns sine of an angle.

S=Sin(90)

The function will return 0.893996663600558

4.Tan(number)

Returns tangent of an angle

T=Tan(90)

The function will return -1.99520041220824

Conversion Function

Now we will about the conversion functions. These functions are used to convert values from one data type to another data type.

The conversion functions in visual basics are mentioned below.

1)Cint :

This function is used to convert the specified expression to integer data type. The syntax is

CInt(Expression)

C=Cint(567.00987)

The function returns in output is 567

2)CLng

This function is used to convert the specified expression to Long data type. The syntax is,

CLng(Expression)

C=CLng(67.88)

The function returns in output 67.

3)CSng: This function is used to convert the specified expression to single data type.

CSng(Expression)

C=CSng(678.988454324)

The function returns in output 678.8885

4)CSng- this function is used to convert the specified expression to Single data type. The syntax is

CSng(Expression)

C=CSng(678.988454324)

The function returns in output 678.8885

5)CVar: This function is used to convert the specified expression to variant data type. The syntax is,

Cvar(Expression)

C=Cvar(700 &"000")

The function returns in output 700000

6) CStr-This function is used to convert the specified expression to String data type. The syntax is,

CStr(Expression)

C=CStr(45.98)

The function returns in output 45.98

7) Str-This function returns string representation number. The syntax is

C=Str(Expression)

C=Str(45.98)

The function returns in output 45.98.

8)Val- This function returns numeric value, which represents the number specified in a string .

The syntax is

Val(expression)

C=Val ("90.76")

The function returns in output 90.76

9) Asc: This function returns numeric value which represents the ASCII character code corresponding to the first letter of the specified string.

The syntax is

Asc(String)

C=Asc("B")

The function returns in output 66.

10) Chr-This function returns the string that represents the character that is associated with the mentioned character code.

The syntax is

Chr(charCode)

C=Chr(66)

The function returns in output B.

Modules :

Module is Visual Basic file that contains code. Code written in Visual Basic is stored in modules. Every application in Visual Basic can be viewed as a collection of modules. Modules are the building blocks of a project. A module is a collection of procedures, function and event handlers.

As the application gets complex and more sophisticated, we can add more forms. There might be a common code, which we want to execute in several forms. We do not want to duplicate the code in both forms, so we create a separate module containing a procedure that implements the common code. This separate module should be standard module. There are three kinds of modules in Visual Basic.

A module is a set of declarations that are followed by procedures. Modular code presents the executable set of instructions in a structured format. Modules are used to make the code more understandable.

They are

- Form modules
- Class modules
- Standard modules

1)Form modules

Form modules are the foundation of most Visual Basic application. They have the file name **.frm** extension. They provide the user interface to an application. They contain procedure that handle events, general procedures, and form level declarations of variables, constants, types and external procedures. The declarations in the form module are private by default. The constants, variables and procedures of a form are not available to the code outside the form, unless the declaration are public.

2) Class modules

Class modules are the foundation of object-oriented programming in Visual Basic. They have file name. cls extension. We can write code in class modules to create new objects. These new objects can include our own customized properties and methods. Actually, forms are just class modules that can have controls placed on them and can display form windows. Class modules do not have visual components as compared to form module. We need to write the code for a class in a class module. We can then create objects of this class in other module

3) Standard modules

Standard modules are containers for procedures and declarations commonly accessed by other modules within the application. Standard modules have the file name **.bas** extension. They can contain global or module level declarations of variables, constant, types, external procedure, and global procedures. The code that we write in a standard module is not necessarily tied to a particular application.

	Class modules		Standard modules
1.	Public variables in class modules, can only be accessed, if we have an object variable containing.	1.	Variable declared public in a standard module are visible from anywhere in our project.
2.	a reference to a particular instance of a class Class modules are an implementation of object-oriented programming.	2.	Standard modules are an implementation of a procedural language.
3.	All the object have their own copies of member data.	3.	Only one copy of global variable exists.

Menus: -

It have handled different menus while working with Windows applications. For example it have worked with File Menu, Edit Menu , View Menu,Help Menu etc. in various Window based applications.

Using the menus the user can access and execute the menu commands, which are grouped logically. For example from the File menu the user can open a file , save a file, print a file etc.

Creating Menus:

In Visual Basic can create menus using the menu Editor. It can be invoked by

1.Clicking on the Tools menu and then click on Menu Editor.

2.Click on Menu Editor icon from the Standard Toolbar.

These points given below, which describes the task details that can be performed using the menu Editor.

1. It can create menu bar, which will contain menus with various menu items. It can also set shortcut and index for the menu item from the Menu editor.
2. Moreover it is used to set the different properties like visible, enabled, checked etc. of the menu items.
3. Menu Editor is used to create cascading menus.
4. Used for creating popup.

To create an application with menus :

1. start with a new project. Invoke the menu Editor from the Standard Toolbar.

2.To specify the caption of the menu in the caption textbox.

3.To set an access key have to put an ampersand before that character. For example going to create a File menu. So the caption will be File.

4.To access the File menu using the keyboard have to set an access key. Lets create F as the access key. So by pressing the ALT+F key will activate the File menu. So set the caption as &File.

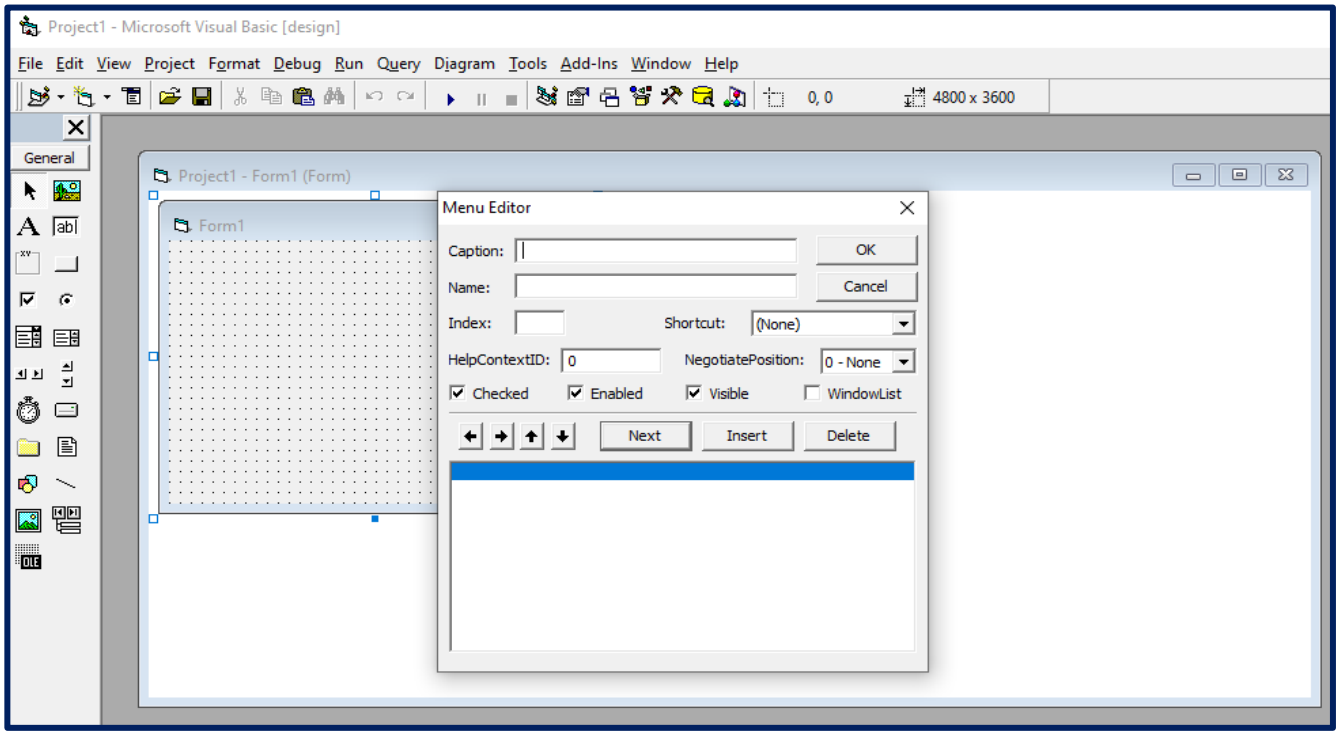
5.Then have to provide a menu item name in the Name textbox to identify the menu item. It can refer to the menu item by searching the item name while writing the code.

6.It have to provide the menu item number in the index textbox, which identify the menu item in the menu control array.

7.To provide a shortcut for the menu item, to select them from the Shortcut list. Check the different properties as per requirement. For example to provide a check mark in the left side of the menu item check the Checked checkbox.

8.Then to add the menu item in the menu item list box use the Next button. Use the Insert button to insert the menu item or submenu item above the currently selected menu item in the menu listbox.

9.Use the arrow buttons to move the menu items in the menu listbox.



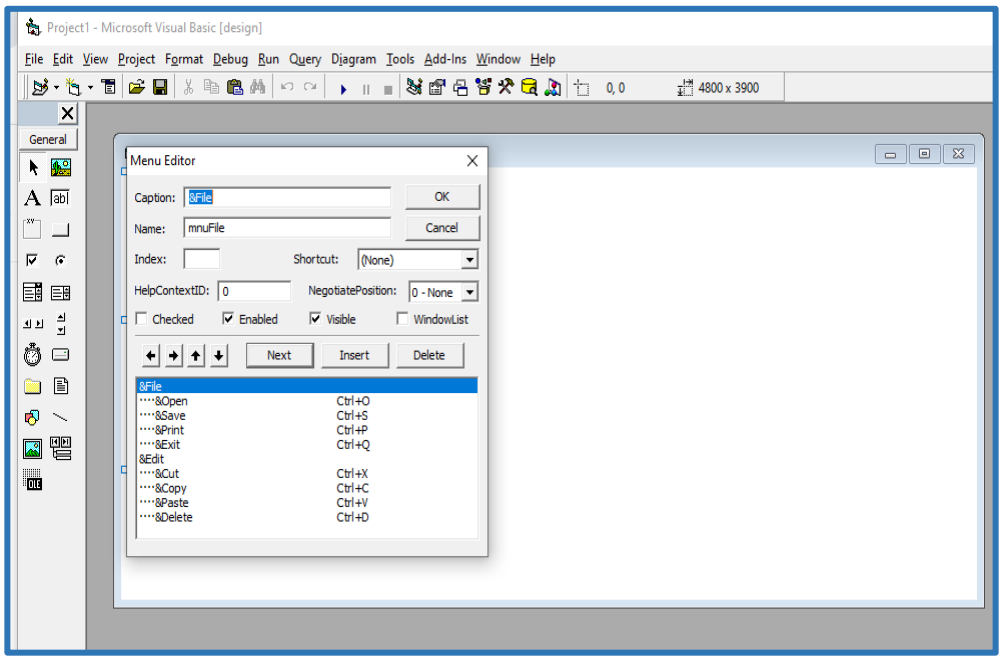
Adding menu items

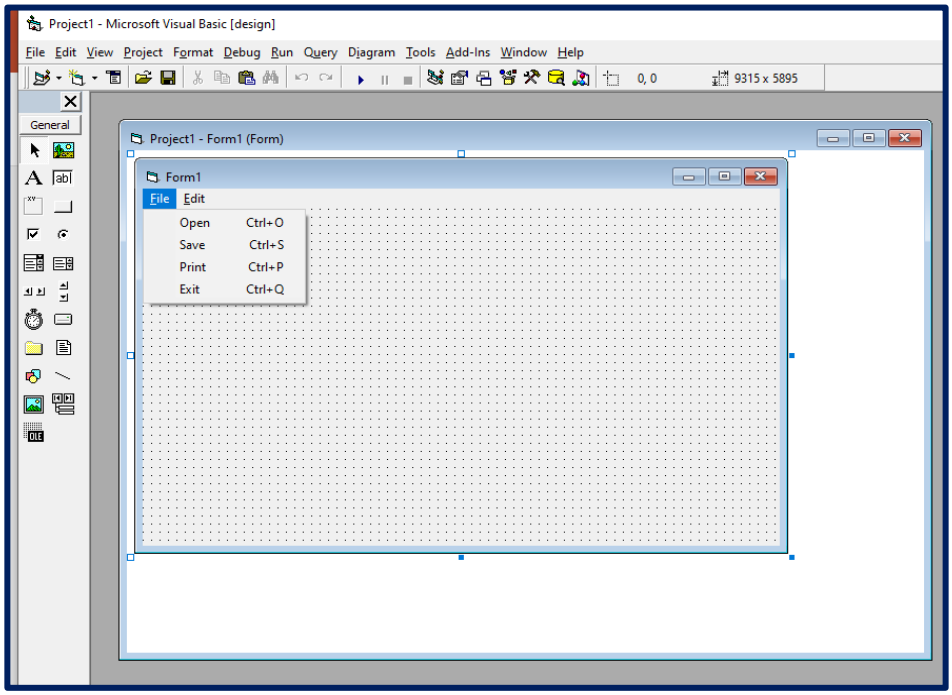
After creating a menu, need to add menu items under that menu. Let's add Open as menu item of File menu. Add the Caption as &Open. Set the Name as mnuFileOpen.

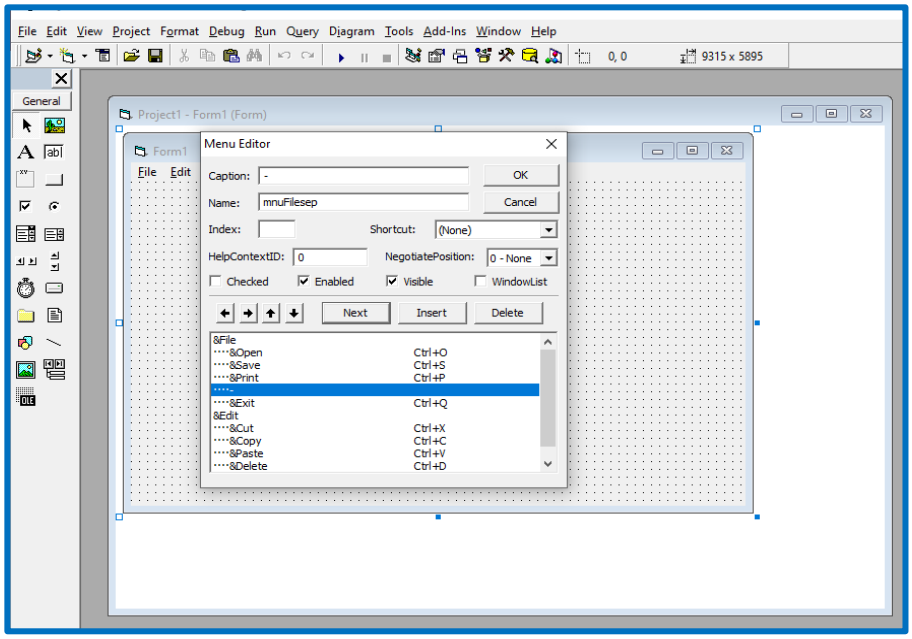
Creating Shortcut for the menu items

1.To set the Shortcut for the Open menu item select it from the menu listbox.

2.Set the Shortcut as Ctrl+O.

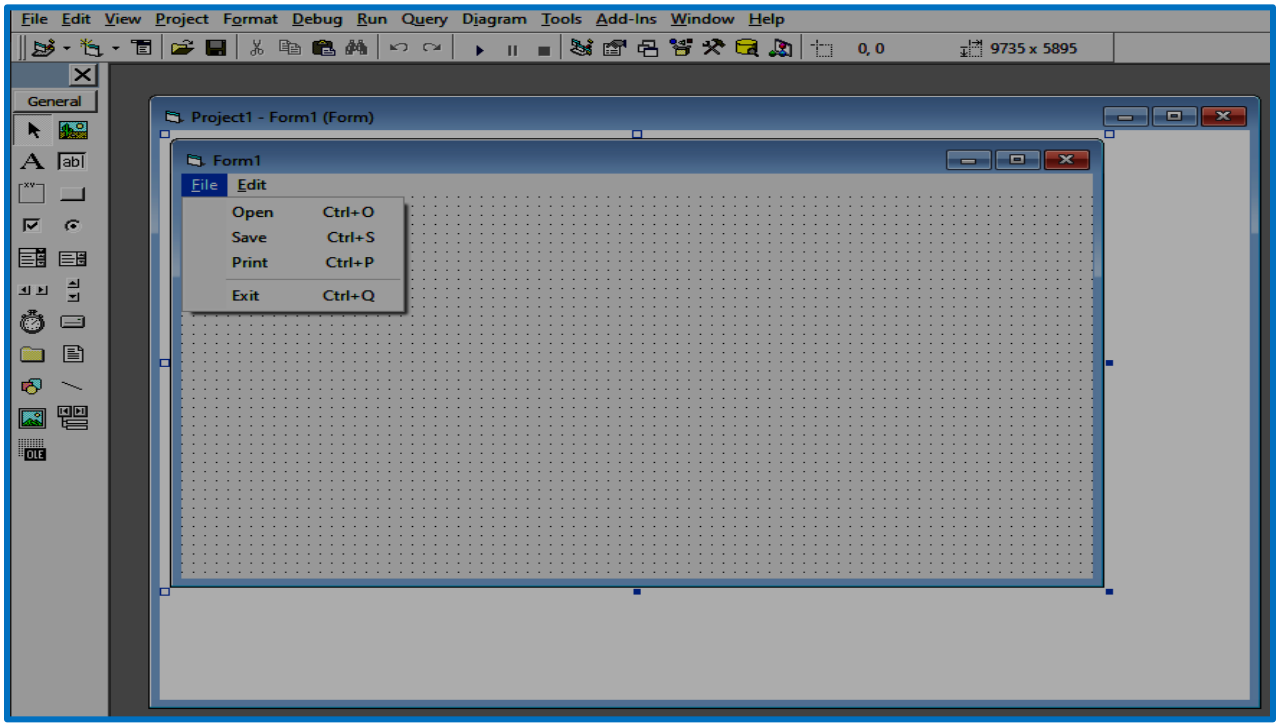



Project1 - Microsoft Visual Basic [design]
File Edit View Project Format Debug Run Query Diagram Tools Add-Ins Window Help
0, 0
9315 x 5895
General
Project1 - Form1 (Form)
Form1
File Edit
Open Ctrl+O
Save Ctrl+S
Print Ctrl+P
Exit Ctrl+Q


File Edit View Project Format Debug Run Query Diagram Tools Add-Ins Window Help
0, 0
9315 x 5895
General
Project1 - Form1 (Form)
Form1
File Edit
Menu Editor
Caption: -
OK
Name: mnuFilesep
Cancel
Index:
Shortcut: (None)
HelpContextID: 0
NegotiatePosition: 0 - None
Checked
Enabled
Visible
WindowList
Next
Insert
Delete
&File
&Open Ctrl+O
&Save Ctrl+S
&Print Ctrl+P
-
&Exit Ctrl+Q
&Edit
&Cut Ctrl+X
&Copy Ctrl+C
&Paste Ctrl+V
&Delete Ctrl+D

Adding Separator bars

1. To group the menu items logically Separator bar is added. Let's add a Separator bar before exit.
2. So, select Exit from the menu list box. Click on Insert.
3. Now set the Caption of the menu item as (-) and set the Name as mnuSep.
4. Click on Ok to close the dialog. Click on the File menu. It will find a separator is added before Exit.



Creating Popup Menu:

Popup is mainly created to have quick access on the menu commands of a particular object. There are two types of Popup they are,

System: This popup menu appears at run time by default. For example, if you right click on the Textbox control at run time a popup menu gets displayed.

Custom: Custom popup menu is the popup menu created by the designer according to the requirement of the application.

Follow the instruction step by step to create the popup menu.

1.Start with a new project. Change the Caption of Form1 as "Popup".

2.Invoke the Menu Editor.

3.Create a menu Picture i.e. set the Caption as Picture and Name as mnuPicture. Deselect the Visible Checkbox.

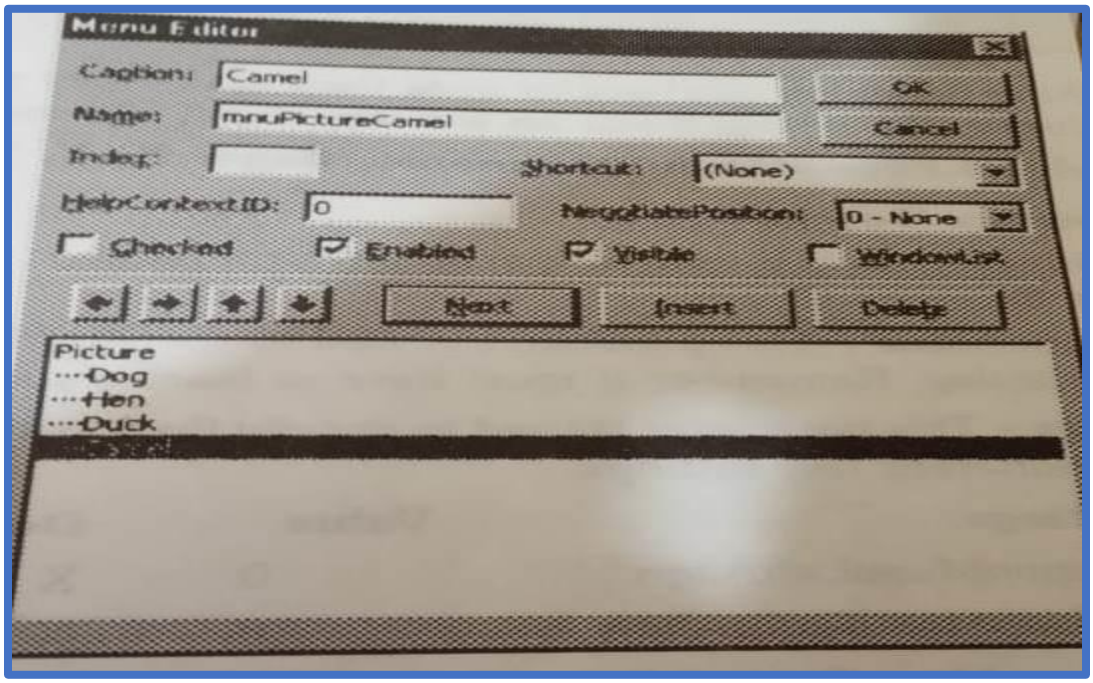
4. So that it won't gets displayed as a part of the menu bar.

5.Now add four menu items Dog, Hen, Duck and Camel under Picture menu. Check your Menu editor.

6.Click on OK to close the Menu Editor dialog.

7.Now add a Label in the Form and Change its Caption as" Working with Popup". The Font size is increased to 18 and the Label is placed in the top central position.

8.Add an image control in the Form. Change the Name property as imgPopup.



After designing have to write the Popup Menu method in code to display the menu. To display the popup menu on a Form object Popup Menu method is used.

Object.PopupMenu menu name, flags, X,Y,boldcommand

Syntax Description:

Object: object is used to specify the form name in which are going to display the popup menu.

menuname:-This parameter is used to specify the popup menu name which are going to display.

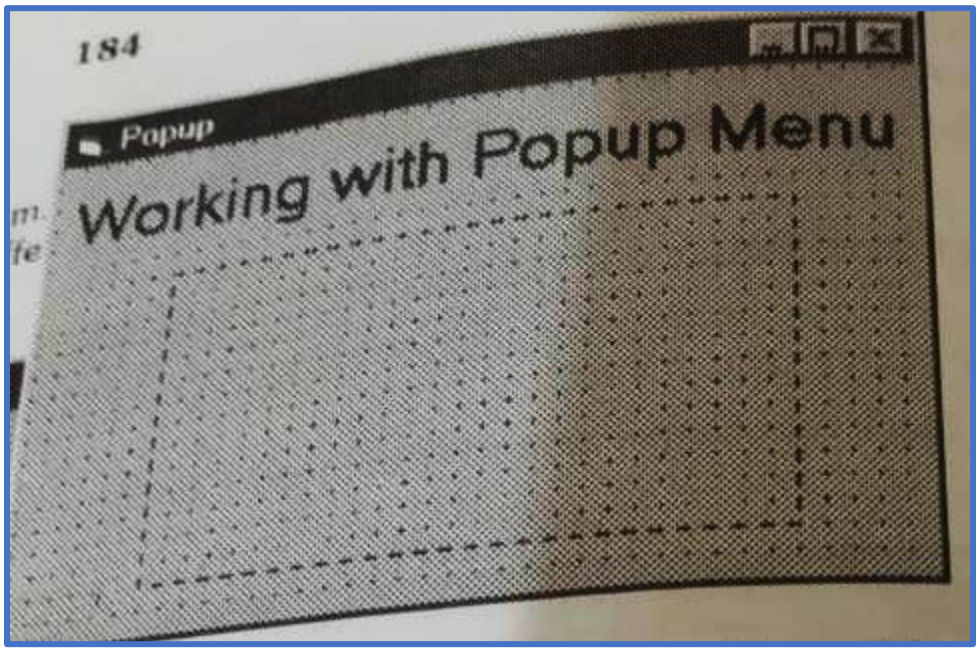
Flags: This parameter is used to specify the location and behavior of the popup.

Settings	Value	Description
vbPopupMenuLeftAlign	0	X specifies the left side position
vbPopupMenuRightAlign	8	X specifies the right-side position
vbPopupMenuCentreAlign	4	X specifies the centered position
vbPopupMenuLeftButton	0	Menu items responds on the left mouse button click only.
vbPopupMenuRightButton	2	Menu items responds on the left or Right mouse button click.

X- It is used to specify the x-coordinate where the popup menu will be displayed. So according to the specified value the popup menu gets displayed.

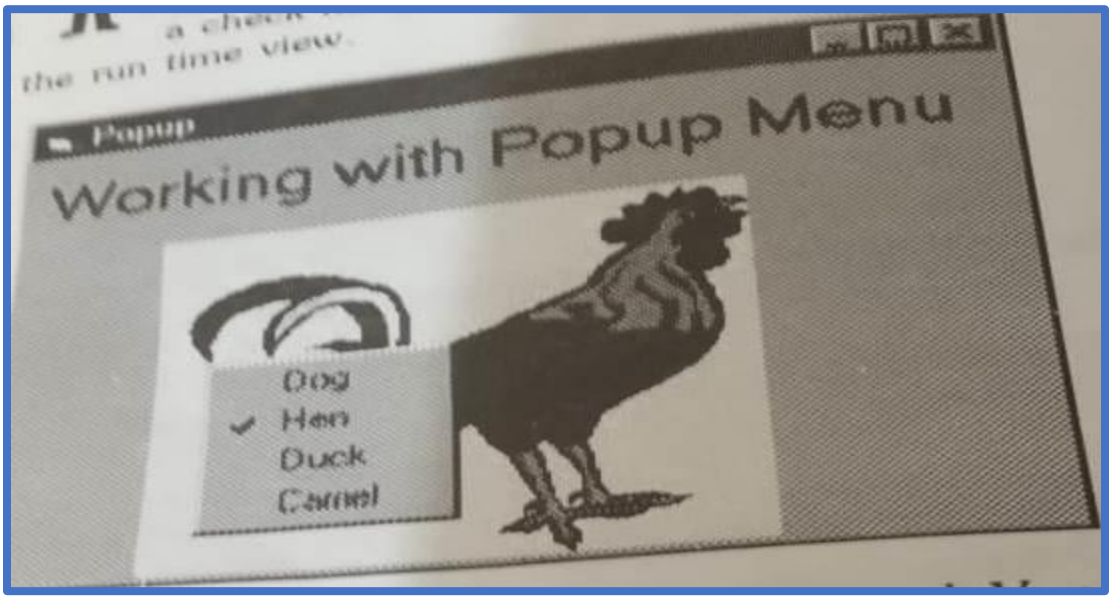
Y- It is used to specify the y coordinate where the popup menu will be displayed. So according to the specified value the popup menu gets displayed.

Boldname: This parameter is used to specify the name of the menu control in the popup menu to display its caption bold face. This is default menu item.




```
Private sub-Form_Load ()
imgPopup. Picture=LoadPicture (“c:\my documents\my pictures\dog.bmp”)
End Sub
Private Sub imgPopup_MouseUp (Button As Integer, Shift As Integer, X As Single, Y As Single)
If button =vbRightButton Then
PopupMenu mnuPicture
End If
End Sub
Picture Sub mnuPictureCamel_Click()
imgPopup. Picture=LoadPicture (“c:\my documents\my pictures\camel.bmp”)
mnuPictureCamel.Checked=True
mnuPictureDog.Checked=False
mnuPictureHen.Checked=False
mnuPictureDuck.Checked=False
End Sub
```

```
Picture Sub mnuPictureDog_Click()
imgPopup.Picture=LoadPicture ("c:\my documents\my pictures\dog.bmp")
mnuPictureCamel.Checked=False
mnuPictureDog.Checked=True
mnuPictureHen.Checked=False
mnuPictureDuck.Checked=False
End Sub
```



Working With Custom Controls:

1.Image list control:

- An ImageList Control contains a collection of images that can be used by other Windows Common Control
- It does not appears on the form at run time.

It serves as a container for icon that are accessed by other control such as ListView, TrueView, TabStrip and ToolBar controls.

To add Image list control in the tool box:

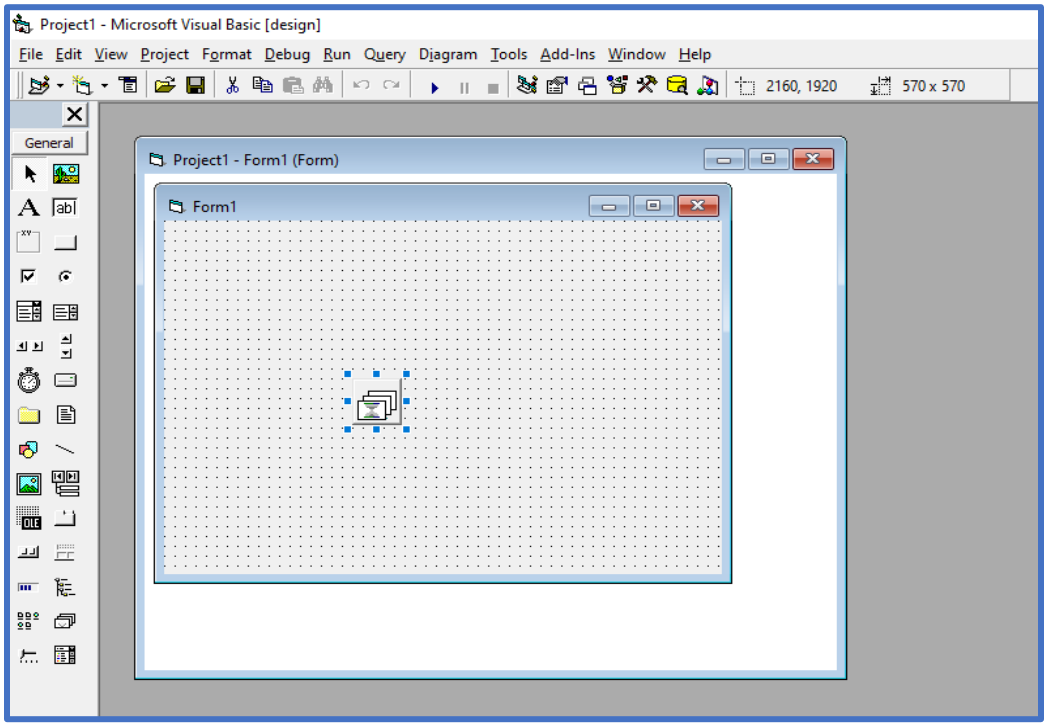
1. Click on "Start" . Then select "All Programs, Microsoft Visual Studio 6.0 and Microsoft Visual Basic 6.0." .

2. Select "Standard EXE" from the list in the New Project dialog box and click on "Open." Click on "Project" on the menu bar, then select "Components" from the drop-down menu.

3. Scroll down the list in the box until Microsoft Windows Common Controls 6.0 (SP4) is visible. Click on the checkbox to select the component and then click on "OK." All components show in the Toolbox.

4. To add Image List control in the form:

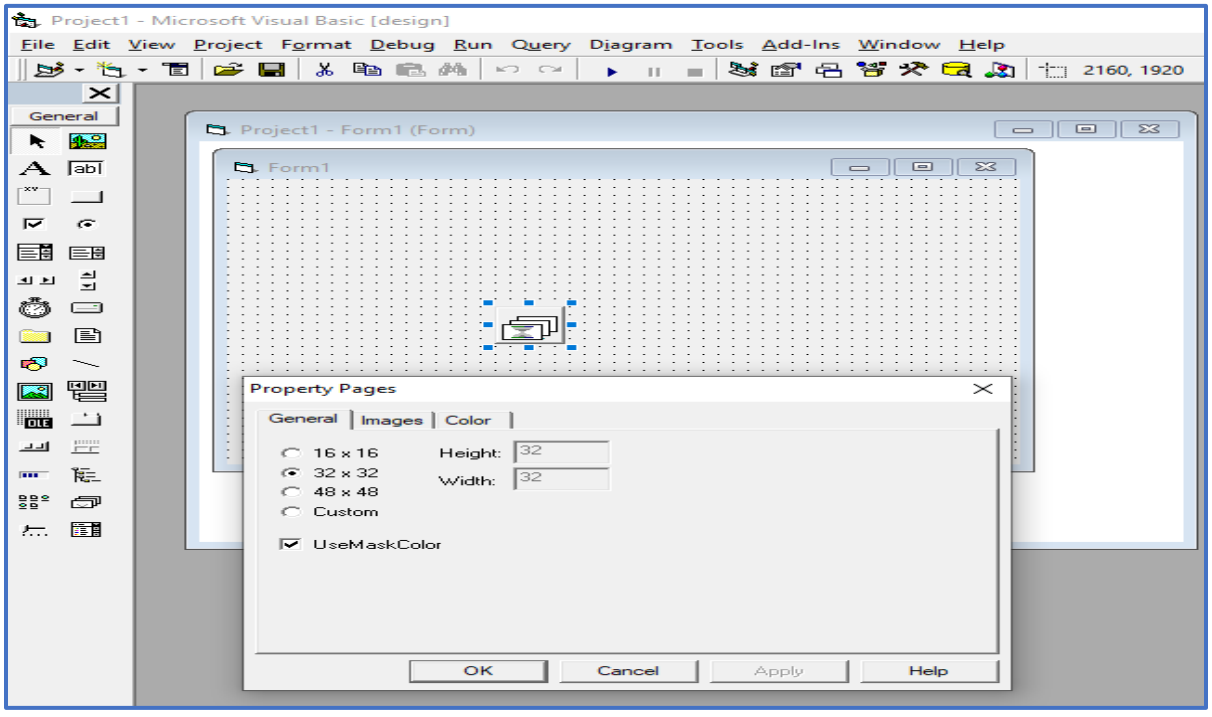
5. Choose the “ListView1” control from the list of controls in the toolbox. Place the control on the form in Visual Basic.



6. To add images to the image list control:

7. Right click on Image control a popup menu is open choose properties option. A Properties Page is open.

8. Click on Images Tab and insert the image as you want.



9. To insert more than one picture then click on Insert Picture Button.

2.Toolbar Control:

Tool bar control is collection of button used to perform particular function. it gives graphical interface to the user. It consists picture or text on button. Picture set by image property and text by caption property. Toolbar control, we can object to toolbar at design time or at run time. toolbar contain button which perform particular function they are generally used. It saves time as respect to menu option. To create toolbar performing following process,

1. Put toolbar control on form from project toolbox.
2. Select toolbar control and right click on it.
3. Select property option from shortcut menu.
4. It displays property dialog box
5. In the general tab of the property window select image list to imagelist1
6. Select button tab in property window
7. Click on insert button
8. In index of button will be set to 0 (zero)
9. Set the key for the button as ―one‖ 1
10. In the image list box write down the index of the image to be displayed on the toolbar.
11. Repeat step 7 to 10 to insert another button on toolbar.

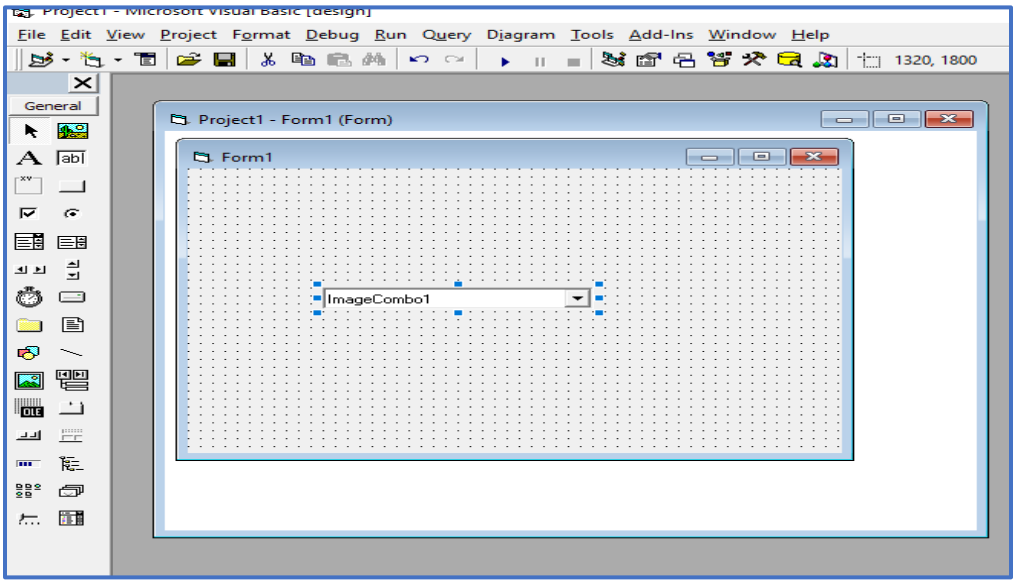
```
If Text1.FontBold = True Then
Text1.FontBold = False
Else
Text1.FontBold = True
End If
If Text1.FontBold = True Then
Button.Value = tbrPressed
Else
Button.Value = tbrUnpressed
End If
End Select
End Sub
```

3.Image combo Control:

Image Combo control can display simultaneously data and picture from the Image List control.

Let's create an application, which will display the pictures of the Image list control in an Image Combo control.

- Start with a new Project.
- From the Components dialog add the Microsoft windows Common Controls 6.0.



- Now add in the Form an image List, Image Combo, Image control and a label control.
- Initially you have to add images in the Imagelist Control. So, open its property page dialog by right clicking on Image List1 and click on properties option from the Context menu.
- From the General tab select the Custom option and set the height and Width both as 20.
- Open the images tab and add images. Here four images are added.
- After inserting the images close the Property Pages dialog by clicking on Ok.
- Select the ImageCombo1.Change the Text property as "Pictures". Now by right clicking on the ImageCombo1,click on properties option to invoke the Property pages dialog.

- From the Image List list select ImageList1.If required you can change the font from the Font tab, color from the Color tab and from the Picture tab can change the properties.
- After setting the options of property pages click on Apply and then on OK to close the dialog.
- Change the caption property of Label1 as “Select the picture from the List", set the font size as 10 and style as bold.
- After finish designing, it’s time to add the code. You have to add the images in the Image Combo control.

```
Private Sub Form_Load()
Set ImageCombo1.ImageList=ImageList1
Dim picture As ComboItem
Set picture = ImageCombo1.ComboItems.Add(1, ,”camel”,1,1,1)
Set picture =ImageCombo1.ComboItems.Add(2, ,”Dog”,2,2,1)
Set picture = ImageCombo1.ComboItems.Add(3, ,”Hen”,3,3,1)
Set picture =ImageCombo1.ComboItems.Add(4, ,”Duck”,4,4,1)
End Sub
Private Sub ImageCombo1_Click()
If ImageCombo1.SelectedItem.Index=1 Then
Image1.picture=LoadPicture (“c:\my documents\my pictures\Camel.bmp”)
End If
If ImageCombo1.SelectedItem. Index=2 Then
Image1.picture=LoadPicture (“c:\my documents\my pictures\Dog.bmp”)
End If
If ImageCombo1.SelectedItem. Index=3 Then
Image1.picture=LoadPicture (“c:\my documents\my pictures\Hen.bmp”)
End If
If ImageCombo1.SelectedItem.Index=4 Then
Image1.picture=LoadPicture (“c:\my documents\my pictures\Duck.bmp”)
End If
End Sub
```

4.List view control

List View Control is used to display the result of a query, displays all the records in a database table and displays an expanded view of a Node in a TrueView control. The list view control display data as list item objects. Each list item object can have an optional icon associated with the label of the object. The control excels at representing subsets of data (such as members of a database) or discrete objects (such as document templates). Following are the possible use of list view control

1. To display the result of a query on a database.
2. To display all the records in a database table.
3. In tandem with a tree view control to give users an expanded view of a tree view control node.

Using the List view control can display items in four different types of views. They are

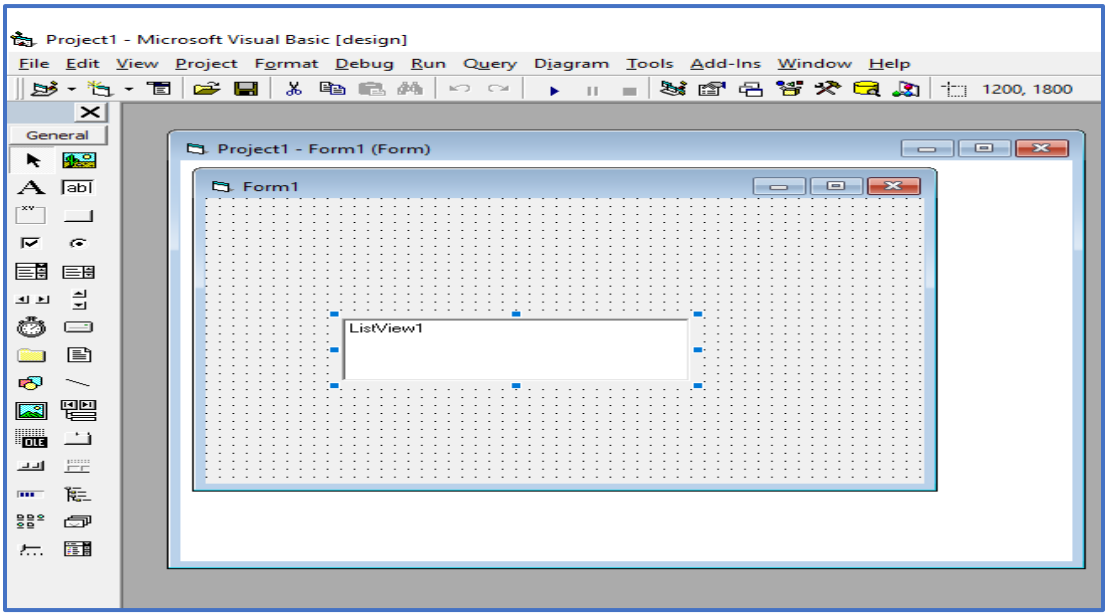
1.Small icons

2.Large icons

3.List

4.Report

It displays data as List Item object and each entry in the List View control is referred as list Item object.



5.Status bar control

A status bar control is used to display the current status of the application under execution. Eg. in Microsoft word the status bar display the current page number in which the user is currently working as well as the total number of pages in the document.

The steps to create the panels in the status bar are as follows

1. Place a status bar on the form

2. Right click on the control
3. Select the option properties from the menu displayed
4. Select the panels tab
5. Click on the insert panel button to create a new panel.
6. Each time a new panel is created, the index of the panel is incremented by 1.

Advantage and uses of status bar control

1. Used to display the current status of the application under execution.
2. Can be used to display the coordinates of the mouse pointer
3. Can be used to display whether the toggle keys like caps lock num lock keys are on or off
4. Can be used to display the number of nodes int eh treeview control
5. Can be used to display the current system time.

```
Private Sub Form_Load() StatusBar1.Panels(2) = Date StatusBar1.Panels(3) = Time
StatusBar1.Panels(4) = "Visual Basic"

End Sub

Private Sub Form_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

StatusBar1.Panels(1).Text = X & "," & Y

End Sub
```

6. Tree view control

The tree view control is used to display the parent child type of relationship among the items. It is used to display the hierarchy of items in the tree form. The individual item in the tree view control is called as node of a tree. The typical tree structure in the windows application is the windows explorer which displays the list of drives which can be expanded to folders and files, folders can be expanded to subfolders and files.

A tree is comprised of cascading branches of ―nodes‖ and each node typically consists of an image and a label. A node can be collapsed or expand. The total number of node is not limited. +ve sign indicate this item has child node, but not currently expanded. If expanded it display –ve sign.

```
Dim frmtag As String
Private Sub Form_Load( )
Tree. Nodes. Add , , "Item", "ITEM"
Tree. Nodes. Add "Item", tvwChild, "Shape", "SHAPE"
Tree. Nodes. Add "Item", tvwChild, "Color", "COLOR"
Tree. Nodes. Add "Shape", tvwChild, "Square", "SQUARE"
Tree.Nodes.Add "Shape", tvwChild, "Circle", "CIRCLE"
Tree.Nodes.Add "Color", tvwChild, "Red", "RED"
Tree.Nodes.Add "Color", tvwChild, "Blue", "BLUE"
Tree.Nodes.Item("Item").Expanded = True
End Sub
Private Sub Tree_NodeClick(ByVal Node As MSComctlLib.Node)
If Node = "SQUARE" Then
frmtag = "s"
Shape1.Visible = True
Shape2.Visible = False
ElseIf Node = "CIRCLE" Then
frmtag = "c"
Shape1.Visible = False
Shape2.Visible = True
End If
 If Node = "RED" Then
If frmtag = "s" Then
Shape1.FillColor = vbRed
ElseIf frmtag = "c" Then
Shape2.FillColor = vbRed
End If
ElseIf Node = "BLUE" Then
```

```
If frmtag = "s" Then
Shape1.FillColor = vbBlue
ElseIf frmtag = "c" Then
Shape2.FillColor = vbBlue
End If
End If
```

End Sub