

Visual Programming

Unit-1

Introduction

Visual Basic:

Visual basic is a RAD (rapid application development) tools that allows programmers to create windows application in very little time. It is the most popular programming language.

Visual Basic is a third-generation [event-driven programming language](#) and [integrated development environment](#) (IDE) from [Microsoft](#) for its [Component Object Model](#) (COM) programming model first released in 1991 and declared legacy in 2008. Microsoft intended Visual Basic to be relatively easy to learn and use.^{[1][2]} Visual Basic was derived from [BASIC](#), a user-friendly programming language designed for beginners, and it enables the [rapid application development \(RAD\)](#) of [graphical user interface \(GUI\)](#) applications, access to [databases](#) using [Data Access Objects](#), [Remote Data Objects](#), or [ActiveX Data Objects](#), and creation of [ActiveX](#) controls and objects.

An integrated development environment (IDE) is a programming environment that has been packaged as an application program, typically consisting of a code editor, a [compiler](#), a debugger, and a graphical user interface ([GUI](#)) builder. The IDE may be a standalone application or may be included as part of one or more existing and compatible applications.

The [BASIC](#) programming language, for example, can be used within Microsoft Office applications, which makes it possible to write a WordBasic program within the Microsoft Word application. IDEs provide a user-friendly framework for many modern programming languages, such as [Visual Basic](#), [Java](#)

Component of IDE

Menu Bar:

This Menu Bar displays the commands that are required to build an application. The main menu items have sub menu items that can be chosen when needed. The toolbars in the menu bar provide quick access to the commonly used commands and a button in the toolbar is clicked once to carry out the action represented by it.

Tool Bar:

The Tool Bar contains all the tools such as Open, Save, Copy, Cut, Start(to run a program) and so on.

Toolbox:

The Toolbox contains a set of controls that are used to place on a Form at design time thereby creating the user interface area. Additional controls can be included in the toolbox by using the Components menu item on the Project menu.

Project Explorer:

Docked on the right side of the screen, just under the toolbar, is the Project Explorer window. The Project Explorer as shown in figure serves as a quick reference to the various elements of a project namely *form*, *classes* and *modules*. All of the object that make up the application are packed in a project. A simple project will typically contain one form, which is a window that is designed as part of a program's interface. It is possible to develop any number of forms for use in a program, although a program may consist of a single form. In addition to forms, the Project Explorer window also lists code modules and classes.

The Code Window:

You need the Code Window to write code that will specify the behavior of the forms and the objects.

Remember that the Form is an object.

Properties Window:

The Properties Window is docked under the Project Explorer window. The Properties Window exposes the various characteristics of selected objects. Each and every form in an application is considered an object. Now, each object in Visual Basic has characteristics such as color and size. Other characteristics affect not just the appearance of the object but the way it behaves too. All these characteristics of an object are called its properties. Thus, a form has properties and any controls placed on it will have properties too. All of these properties are displayed in the Properties Window.

Object Browser:

The Object Browser allows us to browse through the various properties, events and methods that are made available to us. It is accessed by selecting Object Browser from the View menu or pressing the key F2.

Event-Driven Programming

Event-driven programming is a programming paradigm in which the flow of the program is determined by events such as user actions (mouse clicks, key presses), sensor outputs, or messages from other programs/threads. Event-driven programming is the dominant paradigm used in graphical user interfaces and other applications (e.g. JavaScript web applications) that are centered on performing certain actions in response to user input.

In an event-driven application, there is generally a main loop that listens for events, and then triggers a callback function when one of those events is detected. In embedded systems the same may be achieved using hardware interrupts instead of a constantly running main loop. Event-driven programs can be written in any programming language, although the task is easier in languages that provide high-level abstractions, such as closures.

User interface:

An interface is a set of commands or menus through which a user communicates with a program. A command-driven interface is one in which you enter commands.

A menu-driven interface is one in which you select command choices from various menus displayed on the screen.

The user interface is one of the most important parts of any program because it determines how easily you can make the program do what you want. A powerful program with a poorly designed user interface has little value. Graphical user interfaces (GUIs) that use windows, icons, and pop-up menus have become standard on personal computers.

Common controls in vb/elements of user interface

Label:

The label control is used to display text. It is also used to label other controls. The end user cannot edit the label text.

TextBox:

The TextBox control contains characters. End-users can edit the characters contained in the TextBox.

CommandButton :

The CommandButton control is simply a button that we see in our daily-use software. When the end-user clicks the CommandButton, the program behaves according to the code assigned in the CommandButton.

Option Button:

This control enables the end-user to select one among several options. Only one option button among others in a group can be on at the same time. You can name an option using the Caption property.

CheckBox:

The CheckBox control is used to make a yes/no or true-false selection. You can check more than one CheckBox at the same time that let you make multiple choices. You can label this control using the Caption property.

VscrollBar & HscrollBar:

VscrollBar and HscrollBar controls let you create Vertical scroll bar and Horizontal scroll bar respectively.

Frame:

The Frame control is used as a container of other controls. This is also used to group different controls especially in Option Button controls when you wish to select more than one option. The Caption property associated with it is useful to label the frame.

PictureBox & Image:

These controls are used to display images (e.g company logo). The supported picture formats are BMP, DIB (bitmap), ICO (icon), CUR (cursor), WMF (metafile), EMF (enhanced metafile), GIF and JPEG. But PNG format is not supported.

ListBox & ComboBox:

The ListBox control contains a number of items. The user can select one or more items from the list.

The comboBox control has the feature of ListBox and TextBox. This control does not support multiple selections.

DriveListBox, DirListBox & FileListBox:

These controls are often used together to perform file related tasks like opening or selecting files that are stored in the secondary memory.

Timer:

The Timer control is not visible on the form when you run the program. It is used to execute lines of code repeatedly at specific intervals.

Shape & Line:

These controls do not raise events. Shape and Line are used to draw line, rectangle, circle etc on the form.

The Data Control:

The Data control is used for database programming.

OLE(Object Linking & Embedding):

You can connect other programs to your application that you have developed.

Apart from them, there are many other controls provided by the Visual Basic language which will be discussed in the appropriate chapters. You can add external ActiveX controls that will enhance the interface and functionality of your program.

Events:

- An event is a message sent by an object announcing that something is happened.
- An event is a way for an object to provide notification when something of interest happens for e.g. the buttons click event the text boxes text changed event.
- An event is some action that is placed during execution like mouse click, keyboard button press etc.

Few common events:

Mouse event:

The events triggered by mouse action are the most common event in programming with V.B.

- **Click, double click:**

This click event takes place when user clicks the left mouse button, the double click events takes place when user double clicks the left mouse button.

- **Mouse down and mouse up:**

Mouse down event takes place when the mouse button is pressed and the mouse up event takes place when it is released.

- **Mouse move:**

This events takes place continuously as the mouse is moved over a control.

keyboard events:

Keyboard events are generated by key strokes; usually we must program the keyboard event for the controls that can accept text. In addition, we must provide code for keyboard events of control that can be manipulated with both the mouse and keyboard, because many users prefer to work with the keyboard most of the time.

- **key down, key up:**

The key down is triggered when a key is pressed, and the key up events is triggered when a key is released.

- **key pressed events:**

The key press event is used frequently to write keyboards handlers for textboxes, because this events takes place before the character pressed is displayed in the text box.

Change:

The change events are triggered by various controls when their content is changed.

Few common properties:

Name: This properties set the name of control through which we can access the controls properties and methods.

Appearances: These properties can be zero for flat look, and 1 for 3D look.

Back color:These properties set the background color on which text is displayed.

Font:This property sets the face, attributes and size of font use for text, on the control.

Caption:This property sets the text that is displayed on many controls that do not accept input.

Text:This property sets the text that is displays on the controls that accept the user input.

Enable:By default this properties value is true, which means that the control gets the fuscous, set it false to disable to control.

Visible:Set these properties to false to make a control in visible, by default this properties value is true.

Few common methods:

Objects have methods too, which are the action they can carry out, generally methods are the actions, of an objects. The form object for example, knows how to clear it self, and we can invoke the “CLR” to clear a form.

Clear methods:

Clear methods tells the control to discard its contents, if the object is a “list box” the clear methods removes its entire item from control.

Move:

This method let us move and resize the controls at ones run times.

Multiple Document Interface (MDI)

The Multiple Document Interface (MDI) was designed to simplify the exchange of information among documents, all under the same roof. With the main application, you can maintain multiple open windows, but not multiple copies of the application. Data exchange is easier when you can view and compare many documents simultaneously. You almost certainly use Windows applications that can open multiple documents at the same time and allow the user to switch among them with a mouse-click.

Multiple Word is a typical example, although most people use it in single document mode. Each document is displayed in its own window, and all document windows have the same behavior. The main Form, or MDI Form, isn't duplicated, but it acts as a container for all the windows, and it is called the parent window. The windows in which the individual documents are displayed are called Child windows.

An MDI application must have at least two Form, the parent Form and one or more child Forms. Each of these Forms has certain properties. There can be many child forms contained within the parent Form, but there can be only one parent Form.

The parent Form may not contain any controls. While the parent Form is open in design mode, the icons on the ToolBox are not displayed, but you can't place any controls on the Form. The parent Form can, and usually has its own menu.

To create an MDI application, follow these steps:

- Start a new project and then choose Project >>> Add MDI Form to add the parent Form.
- Set the Form's caption to MDI Window
- Choose Project >>> Add Form to add a SDI Form.
- Make this Form as child of MDI Form by setting the MDI Child property of the SDI Form to True. Set the caption property to MDI Child window.

Single Document Interface (SDI)

SDI:

SDI applications allow only one open document frame window at a time. It's made up of one or more independent windows, which appears separately on the windows desktop. An example of this would be a simple text document(Notepad).

Advantages of SDI

An SDI interface works very well with multiple monitors and multiple virtual desktops. It also allows users to switch between multiple open documents using the native Windows taskbar and task manager, rather than through special code that you would need to write into your application

Unit-2

The Language Basic

Data Types: Data type is a type of data that hold in a variable. By default Visual Basic variables are of variant data types. The variant data type can store numeric, date/time or string data. When a variable is declared, a data type is supplied for it that determines the kind of data they can store. The fundamental data types in Visual Basic including variant are integer, long, single, double, string, currency, byte and boolean. Visual Basic supports a vast array of data types. Each data type has limits to the kind of information and the minimum and maximum values it can hold.

1. Numeric

Byte: Store integer values in the range of 0 – 255

Integer:Store integer values in the range of (-32,768) - (+ 32,767)

Long:Store integer values in the range of (- 2,147,483,468) - (+ 2,147,483,468)

Single:Store floating point value in the range of (-3.4x10⁻³⁸) - (+ 3.4x10³⁸)

Double:Store large floating value which exceeding the single data type value

Currency:store monetary values. It supports 4 digits to the right of decimal point and 15 digits to the left

Syntax:

Dim MusicTracks As Integer

2. String

Use to store alphanumeric values. A variable length string can store approximately 4 billion characters

```
Dim CountryName As String
```

3. Date

Use to store date and time values. A variable declared as date type can store both date and time values and it can store date values 01/01/0100 up to 12/31/9999

4. Boolean

Boolean data types hold either a true or false value. These are not stored as numeric values and cannot be used as such. Values are internally stored as -1 (True) and 0 (False) and any non-zero value is considered as true.

```
Dim IsMarried As Boolean
```

5. Variant

Stores any type of data and is the default Visual Basic data type. In Visual Basic if we declare a variable without any data type by default the data type is assigned as default.

Variable:

In the V.B. variables are used for storing values temporally. A variable has a name and its data type. Variables can be declared in two ways:

- **Explicit variable declaration:**

To declare a variable the “dim” statement followed by the variables name and type is used as follows:

Dim names as string

Dim roll as integer

Where, roll and names are variables and integer and strings are data type.

We also can declare multiple variables as:

Dim roll as integer, names as string

When V.B. finds a “dim” statement it creates one or more new variables as specified in the statement.

– **Implicit variable declaration:**

In this type when V.B. meets an undeclared variable it creates a new variable on the spot and uses it, then the new variable's type is variant the generic data type that can accommodate all other data types. Using a new variable encode is equivalent to declaring it without type, V.B. adjust its type according to the value we assign to it, for example:

```
Dim var1, var2 And then assign,
```

```
Var1="welcome to vb6"
```

```
Var2= 29.6
```

Var1 is string type and var2 is numeric one. We can verify this with the `type name ()` function, which returns a variable type as shown below:

```
Debug. print "variable var1 is" & type name(var1)
```

```
Output: variable var1 is string
```

```
Debug. print "variable var2 is" & type name (var2)
```

```
Output: variable var2 is double.
```

Variable Scope:

Every variable in V.B has a scope. The scope of the variables is the section of application that can see and manipulate the variable. If a variable declared within a procedure only the code in specific procedure has access to that variable. This variable does not exist for the rest of application, when variables scope is limited to a procedure it is called local.

For syntax:

```
Private sub command1_click ()
```

```
Dim j as integer
```

```
-----
```

```
-----
```

```
-----
```

```
End sub
```

In some situations the entire application must access a certain variable in this case the variable must be declared as public. Public variable has a global scope, to declare a public variable use “public” statement in place of “dim” statement, moreover public variable may not appear inside procedures. They must be declared as form variable or in a module.

For example:

```
Public A as integer.
```

Variable Declaration

Depending on where the variables are declared and how they are declared, there are many ways to declare a variable in visual basic. When you declare a variable, memory space for the variable is reserved. This is called memory allocation. Different amount of memory space is reserved for different data types.

You can declare a variable with the 'dim' keyword.

Syntax:

Dim variable As [Type]

Example:

```
Private Sub cmdSum_Click()
```

```
    Dim m As Integer
```

```
    Dim n As Integer
```

```
    Dim sum As Integer
```

```
    m = 10    'm is a variable, 10 is a constant
```

```
    n = 30
```

```
    sum = m + n
```

```
    Print "The sum is " & sum
```

```
End Sub
```

Output: The sum is 40

Constants:

Some variables do not change value during program execution are called constants, the statement $\text{area} = \text{PI} * \text{Radius} * \text{radius}$, is much easier to understand than equivalent.

$\text{Area} = 3.14156 * \text{radius} * \text{radius}$.

Example:

```
Private Sub cmdCalculate_Click()  
    Const pi = 3.1415926 'or Const pi As Double = 3.1415926  
    Dim area As Double, r As Double  
    r = 2  
    area = pi * r * r  
    Print area  
End Sub
```

Output: 12.5663704

Expressions and Mathematical operators

- An *operator* is a code element that performs an operation on one or more code elements that hold values. Value elements include variables, constants, literals, properties, returns from **Function** and **Operator** procedures, and expressions.
- An *expression* is a series of value elements combined with operators, which yields a new value. The operators act on the value elements by performing calculations, comparisons, or other operations.

Types of Operators

Visual Basic provides the following types of operators:

- [Arithmetic Operators](#) perform familiar calculations on numeric values, including shifting their bit patterns.
- [Comparison Operators](#) compare two expressions and return a **Boolean** value representing the result of the comparison.
- [Concatenation Operators](#) join multiple strings into a single string.
- [Logical and Bitwise Operators in Visual Basic](#) combine **Boolean** or numeric values and return a result of the same data type as the values.

Type conversion: In some situation we need to convert variable from one type into another the visual basic functions that performs data type conversions are:

Function	Converts its argument to
Cbool	Boolean
Cbyte	Byte
Ccur	currency
Cdate	date
Cdbl	double
Cstr	string
Cvar	variant

To convert the variable initialized as dim a as integer,

To a double type we use the function as

B = cdblo(A)

For e.g.

Dim A as integer, B as integer

A = 23

B = 7

Debug .print A/B -- -- -- -- o/p à 3.287514

But ,

Debug .print cdbl(A/B) -- -- -- -- o/p à 3.28751435789368

array

An array is a consecutive group of memory locations that all have the same name and the same type. To refer to a particular location or element in the array, we specify the array name and the array element position number.

The Individual elements of an array are identified using an index. Arrays have upper and lower bounds and the elements have to lie within those bounds.

Declaring arrays

The programmer specifies the array type and the number of elements required by the array so that the compiler may reserve the appropriate amount of memory. Arrays may be declared as Public (in a code module), module or local. Module arrays are declared in the general declarations using keyword Dim or Private. Local arrays are declared in a procedure using Dim or Static. Array must be declared explicitly with keyword "As".

There are two types of arrays in Visual Basic namely:

- **Fixed-size array** : The size of array always remains the same-size doesn't change during the program execution.
- **Dynamic array** : The size of the array can be changed at the run time- size changes during the program execution.

Syntax:

Declaring a fixed-array

```
Dim numbers(5) As Integer
```

```
Dim numbers (1 To 6 ) As Integer
```

Multidimensional Arrays

Arrays can have multiple dimensions. A common use of multidimensional arrays is to represent tables of values consisting of information arranged in rows and columns. To identify a particular table element, we must specify two indexes: The first (by convention) identifies the element's row and the second (by convention) identifies the element's column.

Tables or arrays that require two indexes to identify a particular element are called two dimensional arrays. Note that multidimensional arrays can have more than two dimensions. Visual Basic supports at least 60 array dimensions, but most people will need to use more than two or three dimensional-arrays.

Syntax:

```
Dim AvgMarks ( 50, 50)
```

Collections:

Arrays are convenient for storing related data, but accessing a individual data can be a problem, if you don't know index of the array you have to scan each element until you find the desired one. V.B. provides an alternative known as "collection".

Similar to an array a collection stores related data items. The advantage of collection over an array is that the collection let us access its item via key.

For ex. Temperature("Butwal")

To use a collection:

To use collection we must declare as follows : **Dim temperature as new collection**

The keyword new tells V.B. to create a new collection and name it temperature.

The collection object provides 3 methods and 1 property.

Add method: add items to the collections.

Remove method: removes items from the collections.

Item method: returns the no of items in the collection.

Count property: returns the no of items in the collection.

Adding a collection

The *Add* method adds new items to the collection and it has following syntax:

collection. Add value, key, before, after

For ex. Temperature.Add 30, "Palpa"

Procedure:

In visual basic the application is made up of a small self contain segments which are known as procedure. Procedures are useful for implementing repeated task such as frequently used calculations. The procedures are “divide and conquer” approach which permits the V.B. language and even the longest applications are written by breaking into small, well defined task. Each task is performed by a separate procedure that is written and tested separately from the other the two types of procedures are:

- **Subroutine:**

A subroutine is a block of statements that carries out a well defined task. The block of statement is placed with a pair of sub/end sub statement and can be involved by its name. The following subroutine displays the current date in a message box and can be called by its name show date.

For example:

```
Sub show date()
```

```
Msg box date()
```

```
End sub
```

- **Functions:** A function is similar to a subroutine but a function returns a value or result. Subroutine performs a task and does not report anything to the calling program, function commonly carry out calculation and report the result. The statements that make up a function are placed in a pair of function/end function statements, moreover a function reports a result it must have a type.

For example:

Function next day() as date

Next day = date() +1

End function

This function next day () returns the tomorrow's date by adding one day to the current date.

Event Handler:

An event handler is a short segment of code that is executed each time an external or internal condition triggers the event. When the user clicks the controls, the controls click event handler is executed. This handler is nothing more than the subroutine which performs all the actions we want when the control is click.

Every application contains a no of event handler which contain code to react to user action. Event handler needs not return any result and they are implemented as subroutine.

Control flow statement:

An application needed a built in capability to test condition and take a different course of action depending on the outcome of the test. V.B. provides three decision control flow, structure respectively, they are as:

- **If ...then statements:** This statement test the condition specified and if it is true executes the statement that follows.

For syntax:

If condition then statement.

With multiple statements:

If.....then statement a, statement b.....

Statement 2.

For example:

If (a=b) then msg box “a &b are equal”

- **If Thenelse statement:** a variation of if.... Then statement is if...then.... Else statement, which executes one block of statement if the condition is true and another if the condition is false. The syntax of if...then....else statement is as follows:

if condition then(true block) statement block1

Else statement block2

End if

For example:

If (a<b) then

Msg box “a is smaller”

Else

Msg box “b is smaller”

End if

- Select case: Runs one of several groups of statements, depending on the value of an expression.

Select case expression

Case value 1

Statement block_1

Case value 2

Statement block_2

Case else

Statement block

End select

For example:

Select case weekday(date)

Case 1

Day name = " Sunday"

Message = "have a nice day"

Case 2

Case 7

day name = " Saturday"

message = "have a nice weekend"

case else

message = "have a good day always"

end select

Loop structure:

Loop statement allows us to execute one or more lines of code repetitively V.B. supports following loop statements.

- **Do..... loop statement:** The do.... Loop executes a block of statements for as long as a condition is true. To execute a block of statement, while a condition is true use the following syntax.

Do while condition

Statement block

--- --

--- --

Loop

To execute a block at statement until the condition is true, we use following syntax.

Do until condition

Statement block

--- --

--- --

Loop

For example:

K = 0

Do while k<100

K = k+1

Label1.caption = k

Loop

For.....Next :Repeats a group of statements a specified number of times.

The syntax for..... Next loop statement is:

For counter = start to end [step increment]

Statements

Next [counter]

The keyword in square brackets are optional, the argument counter start, end and increment are all numeric.

For example:

For I = 0 to 100

Debug .print i

I = I +1

Next I

While..... wend loop statement:

This loop executes a block of statement while a condition is true.
The while.....wend loop has the following syntax:

```
While condition  
Statement block  
Wend
```

For example:

```
Number = 0  
While number =>0  
Total =total+number  
Number = input box ("enter another value")  
Debug .print number  
Wend
```

Recursive programming:

Recursion program is used for implementing algorithm or mathematical definition or functions that are described recursively that is in terms of them selves. A recursive definition is implemented by a procedure that calls itself and hence it is called recursive procedure for example let us consider the following procedure and it can be taken as syntax for recursive programming:

Syntax for recursive procedure:

Function do something (n as integer) as integer

{ other statement-----}

Value = value-1

If value = 0 then exit function

New value = do something (value)

{more statement-----}

End function

A program to calculate factorial of a number

Codes:

```
Private sub cmdfact_click()  
Function factorial (n as integer) as integer  
N = val (txt1.text)  
If n = 0 then  
Factorial = 1  
Else  
Factorial = n * factorial(n-1)  
End if  
End function  
Txt2.text = factorial  
End sub
```

Arguments:

Subroutine and argument are not entirely isolated from the rest of application, most procedure accept argument from the calling programs. An argument is a value we pass to the procedure and on which the procedure usually acts. This is how subroutine and function communicate with the rest of application.

Unlike the argument less click event, the key pressed subroutine provides an integer argument which is the ASCII code of the character pressed. The definition of key press subroutine is as follows.

For example:

```
Private sub command1_click(key ASCII as integer)
```

```
End sub
```

Here key ASCII is an argument that conveys information about the key passed.

For e.g. to find the greatest between two values

```
Function min(A as single, B as single) as variant
```

```
Min = IIF(A<B, A, B)
```

```
End function
```

Argument pass by value

If the calling code element underlying the argument is a nonmodifiable element, declare the corresponding parameter `ByVal`. No code can change the value of a nonmodifiable element.

If the underlying element is modifiable, but you do not want the procedure to be able to change its value, declare the parameter **`ByVal`**. Only the calling code can change the value of a modifiable element passed by value.

Argument pass by reference

If the procedure has a genuine need to change the underlying element in the calling code, declare the corresponding parameter **ByRef** .

If the correct execution of the code depends on the procedure changing the underlying element in the calling code, declare the parameter **ByRef**. If you pass it by value, or if the calling code overrides the **ByRef** passing mechanism by enclosing the argument in parentheses, the procedure call might produce unexpected results.

Unit-3

The Forms and Basic controls

Forms:

In VB the form is the container for all the controls that make up the user interface. Forms have a built in functionality that is always available without any programming, for example moving form, resizing with mouse, keyboard or through control menu.

Appearance of form:

The main characteristics of a form are the title bar on which the forms caption is displayed on the left end of the title bar is the control menu icon. Clicking this icon opens the control menu, max size, min size, and close buttons are found on the right side of the form.

We can customize the appearance of the form with the following form properties:

- **Min/max buttons:**

These two properties are true by default set it to false to hide the corresponding buttons on the title bar.

- **Control menu:**

This property is also true by default, set it to false to hide the icon and dissolve the control menu.

- **Border style:**

This property determines the style of forms border and the appearance of form.

Loading, hiding and showing form:

The 3 possible status of a form are:

- **Not loaded:**

The form lives on a disk file and doesn't take up any resources.

- **Loaded but not shown:**

The form is loaded into memory, takes up the required resources and is ready to be displayed.

- **Loaded and shown:**

The form is shown and the user can interact with it.

Loading and unloading forms:

In VB load and unload statements are used to load and unload forms. The load statement has the following syntax:

Load_name

And the unload statement has the syntax is:

Unload_name

Where, form_name = variable name of the form to be loaded or unloaded.

Showing form:

To show a form, we use the show method, if the form is loaded but not visible the show method brings the specified form on top on your desktop. If the form is not loaded, the show methods load it and then display it. The show method has the following syntax:

Form_name.show mode

Where, form_name = variable of form name.

Mode = optional argument which determines whether the form will be modal or modeless

It can have one of the following values:

0 = modal (default)

1 = modeless (e.g. input box()).

Hiding form:

If the application uses many forms, we may want to hide some of them to make room on desktop for others. To hide a form use the forms hide method whose syntax is given below:

`Form_name.hide`

To hide a form from within its own code, use the statement,

`Me.hide`

Designing menu:

Menus are one of the most common and characteristics elements of the windows user interface. Menus are the most popular means of organizing a large number of options.

- **The menu editor:**

Menus can be attached only too forms and we design with menu editor. To know how the menu editor works, start a new standard exe project and when form1 appears in the design window, choose tools,-> menu editor -> to open the menu editor as shown below:

Menu Editor



Caption:

OK

Name:

Cancel

Index:

Shortcut:

(None)



HelpContextID:

0

NegotiatePosition:

0 - None



☐ Checked

☒ Enabled

☒ Visible

☐ WindowList



Next

Insert

Delete

<- = remove submenu

-> = add submenu

↑ = move the selected menu up

↓ = move the selected menu down

Next = move to the next menu

Insert = insert a new menu

Delete = delete the selected menu

Each menu command has two properties:

- Caption:

This is starting that appears on the applications menu bar.

- Name:

This is the name of menu command.

Caption is what the user sees on the form and the name is the means of accessing the control form within the code. To add a command to the forms menu bar, enter a caption and a name for each command.

2. Creating menus:

Let us create the menu structure as shown below where menu contains two commands file and edit, when user clicks on either one, the sub menus are displayed.



- ⇒ Open a new project and a form.
- ⇒ Choose tool
- ⇒ Menu editor and type the menu items as shown below.

File	Name
File:	<u>Mnu file</u>
Open	<u>Mnu open</u>
Save	<u>Mnu save</u>
Exit	<u>Mnu exit</u>
Edit:	<u>Mnu edit</u>
Copy	<u>Mnu copy</u>
Cut	<u>Mnu cut</u>
Paste	<u>Mnu paste</u>

- ⇒ Run the application by pressing F5 and now you can run the menu you created.

Developing short cut key and access key: Opening menus and selecting commands with the mouse sometimes becomes inconvenient. To simplify menu access, VB supports access key and shortcut keys:

- **Access key:** It allows the user to open a menu by pressing the alt key and a letter key. To open the edit menu in all windows, application for e.g. we can press ALT+E. E (letter key) is the edit menus access key.

Access keys are designed by the designer of application and they are marked with an underline character. The underline under the character E in the edit menu denotes that E is the menus access key and the key stroke ALT+E opens the edit command.




To assign an access key to menu command, insert the & (ampersand symbol) in front of the character you want to use as an access key in the menu caption.

- **Shortcut keys:** Shortcut keys are similar to access keys but instead of opening a menu, they run a command when pressed. Assigns that shortcut key to frequently used menu commands, so the user can reach them with a single stroke. Shortcut keys are combination of the control key and a function or character key.

To assign a shortcut key to a menu command, drop down the shortcut list in the menu editor and select a key stroke.

The text box control:

The textbox is the primary mechanism for displaying and entering text and is one of the most common elements of the windows user interface. The text box control is a small text editor that provides all the basic text editing facilities like inserting and selecting text, scrolling the text. The text box is an extremely versatile data entry tool that can be used for entering a single line text, such as a no or password, or for entering simple text, files as shown in figure below.

		  	
Name	<input type="text" value="Admin"/>	Last name	<input type="text" value="govinda"/>
Password	<input type="password" value="*****"/>	First name	<input type="text" value="Raj"/>
		Roll no	<input type="text" value="42"/>
		Add	<input type="text" value="Butwal"/>

Basic properties of text box: The basic properties of the textbox which determines the appearance and the functionality of text box control are as following bellows:

- **Multi line:** This property determines whether the textbox will hold a single line or multiple lines of text by default the control holds a single line of text, to change this set the multi line property to true.
- **Scroll box:** This property controls the attachment of scroll bar to the text box control, if the text exceeds the controls dimension. Single line text boxes can have a horizontal scroll bar so that the user can view any part of a long line of text multi line text boxes can have a horizontal or a vertical or both.
- **Max length:** This property, determines the no of characters the text box control will accept. Its default value is zero, which means the text may be of any length, upto the controls capacity limit (upto 64 kb or 255 characters for single line textbox).The restrict the no of character the user can type save the value of this property accordingly.
- **Text:** The most important property of the text box control is text property which holds the controls text. This property is also assigning some initial text to the control. A run time, use this property to extract the text entered by the user or to replace the existing text by assigning a new value to the text property.
- **Password char:** It is available at design time and it turns the characters typed into any character you specify. It you set this value to an asterisk * for e.g. the user sees and asterisk in place of every character typed.
- **Seltext:** This property returns the selected text. If you want to manipulate the currently selected text from within your code uses seltext property.

List box and combo box controls:

- The list box and combo box control present list of choices for the user to select.
- The list box occupies a user specified amount of space on the form and is populated with a list of items, the user can select one or more with the mouse buttons.
- The combo box control also contains multiple items but occupies less space on the screen.
- The combo box control is an expandable list box control.
- The real advantage to the combo box control is that the user can enter new information in the combo box, rather than being forced to select only the items listed.

list box control methods:

Add item: it add an item to the list its syntax is:

List1.additem item, index (optional)

For example:

List1.additem "ram"

The item is string to be added to the list and index is no or order.

Remove item:

It removes an item from list. Syntax is:

list1.removeitem index.

The index parameter is the order of the item to be removed but at this time it is not optional. For example: list1.removeitem 0 (remove the top item from list).

Clear:

It removes the entire item form list. For e.g. list1.clear

List count:

This is the no of item in the list. For syntax: list1.list count.

List ():

This is an array that holds the lists item.

The element list (0) holds the first elements of the list, list (1) holds the second item and so on upto list. (Listcount-1).

The combo box controls style property:

Value	Description
1. 0	(Default) dropdown combo. This control is made up of a drop down list and a text box.
2. 1	(simple combo box) include a text box and a list that doesn't dropdown. The user can select from the list or type in the text box.
3. 2	Dropdown list. This style is a drop down list from which the user can select one of its items but can't enter a one.

The scroll bar:

It is a long stripe with an indicator that lets the user select a value between two ends of controls. The basic properties of scroll bar control are:

- Min property: the controls minimum value.
- Max property: the controls maximum value
- Value: the controls current value specified by indicators position.

Scroll bars events:

It has mainly two events:

- **Change:**
It occurs every time the user changes the indicators position and release the mouse button.
- **Scroll:** It occurs continuously while the indicators are moving.

The slider control:

- it is similar to the scroll bar control but it lacks the granularity of the scroll bar control.
- It does not appear on tool box by default.
- To use this control in project, right click on the tool box or select components in project menu, on the component dialogue box, check Microsoft windows common controls to your toolbox including slider control.
- As with scroll bar small change and large change properties are available. Small change is the smallest increment by which the slider value can change. To change the sliders value by large the user can click either side of indicator.
- It has the similar property and event as with those of scroll bar control.

The common dialog control: The common dialog controls are used most frequently and are common to every application to prompt the user for file name, font name, size, color, which are to be used in our application but are very difficult to design for own purpose (us). So nearly all window application, use some standard dialog boxes for common operation such as selecting a font or opening a file.

The common dialog control provides its services to the application but, doesn't appear on the form at run time. It provides the following types of built-in windows dialogue boxes.

Save:

The file save common dialog box lets user select or specify a file name in which **the current document will be saved.**

Open:

The file open common dialog box lets user select a file to open.

Color:

The color dialog box lets user select color or specifies custom colors.

Font:

It lets user select a type face and style to be applied to the current text selection.

Print:

It lets user select and setup a printer.

Help:

The help common dialog box displays help topic.

The file control: Three of controls on tool box let us access the computers file system, they are drive list box, dirlist box and file list box control.

Drive list box:

It displays the name of drives within and connected to our PC.

The basic property of this control is drive property which sets the drive to be initially selected in the control or returns the users selection.

Dir list box:

It displays the folder of current drive.

The basic property of these controls is path property, which is the name of folder whose sub folder is displayed in the control.

File list:

It displays the files of current folder.

The basic property of this control is also called path and each path name of the folder whose files are displayed.

The pattern property of the control let us specify which files will be displayed with a file matching string such as.

“*.txt” or “abc.txt”

It has mainly two events change and click event.

The tree view and list view control:

The tree view control implements a data structure known as tree. A tree is the most appropriate structure for storing hierarchical information.

For e.g. in the PC the window explorer, a utility for examining and navigating the hard disk structure.

The left pane, where folders are displayed is a tree view control.

Image list control:

- It is a simple control that stores a number of images used by other controls at run time.
- To use the image list control in our project, add windows common controls component.

The rich text box:

- It is the code of a full blown word processor.
- It provides all the functionality of a text box.
- It gives the capability to mix different font, size and attributes.
- By using this we can even place images in our text on a rich text box.
- The fundamental properties of rich text are text RTF.
- The text RTF property returns the text along with any formatting information

ActiveX controls

ActiveX controls are COM components or objects you can insert into a Web page or other application to reuse packaged functionality someone else has programmed. You can use ActiveX controls developed for Visual Basic 6.0 and earlier versions to add features to the **Toolbox** of Visual Studio.

To add ActiveX controls to the toolbox

- On the **Tools** menu, click **Choose Toolbox Items**.
- The **Choose Toolbox** dialog box appears.
- Click the **COM Components** tab.
- Select the check box next to the ActiveX control you want to use, and then click **OK**.

Unit-4

Drawing

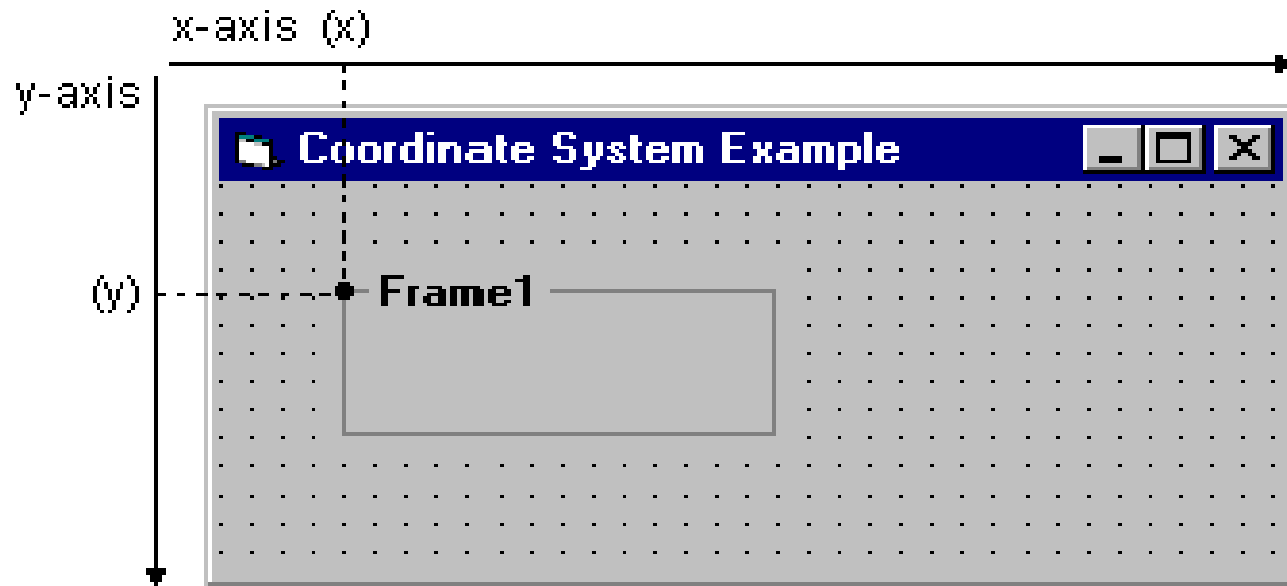
coordinate system

The coordinate system is a two-dimensional grid that defines locations on the screen, in a form, or other container (such as a picture box or Printer object). You define locations on this grid using coordinates in the form:

(x, y)

The value of x is the location of the point along the x -axis, with the default location of 0 at the extreme left. The value of y is the location of the point along the y -axis, with the default location of 0 at the extreme top. This coordinate system is illustrated in Figure 12.3.

Figure : The coordinate system of a form



The following rules apply to the Visual Basic coordinate system:

- When you move or resize a control, you use the coordinate system of the control's container. If you draw the object directly on the form, the form is the container. If you draw the control inside a frame or picture box, the frame or the control is the container.
- All graphics and Print methods use the coordinate system of the container. For example, statements that draw inside a picture box use the coordinate system of that control.
- Statements that resize or move a form always express the form's position and size in twips. When you create code to resize or move a form, you should first check the Height and Width properties of the Screen object to make sure the form will fit on the screen.
- The upper-left corner of the screen is always (0, 0). The default coordinate system for any container starts with the (0, 0) coordinate in the upper-left corner of the container.

Using Graphical Controls

Visual Basic provides three controls designed to create graphical effects in an application:

- The image control
- The line control
- The shape control

Advantages of Graphical Controls

- The image, line, and shape controls are very useful for creating graphics at design time. One advantage of graphical controls is that they require fewer system resources than other Visual Basic controls, which improves the performance of your Visual Basic application.
- Another advantage of graphical controls is that you can create graphics with less code than with graphics methods. For example, you can use either the Circle method or the shape control to place a circle on a form. The Circle method requires that you create the circle with code at run time, while you can simply draw the shape control on the form and set the appropriate properties at design time.

Using Graphics Methods

In addition to the graphical controls, Visual Basic provides several methods for creating graphics. The graphics methods, summarized in the following table, apply to forms and picture boxes.

Method	Description
Cls	Clears all graphics and Print output.
PSet	Sets the color of an individual pixel.
Point	Returns the color value of a specified point.
Line	Draws a line, rectangle, or filled-in box.
Circle	Draws a circle, ellipse, or arc.
PaintPicture	Paints graphics at arbitrary locations.

Advantages of Graphics Methods

- The graphics methods work well in situations where using graphical controls require too much work. For example, creating gridlines on a graph would need an array of line controls but only a small amount of code using the Line method. Tracking the position of line controls in an array as the form changes size is more work than simply redrawing lines with the Line method.
- When you want a visual effect to appear briefly on a form, such as a streak of color when you display an About dialog, you can write a couple of lines of code for this temporary effect instead of using another control.
- Graphics methods offer some visual effects that are not available in the graphical controls.

Working with Color:

Visual Basic uses a consistent system for all color properties and graphics methods. A color is represented by a Long integer, and this value has the same meaning in all contexts that specify a color.

To use the RGB function to specify a color

- Assign each of the three primary colors (red, green, and blue) a number from 0 to 255, with 0 denoting the least intensity and 255 the greatest.
- Give these three numbers as input to the RGB function, using the order red-green-blue.
- Assign the result to the color property or color argument.

Using the Picture Object:

The Picture object is similar in some respects to the Printer object — you can't see it, but it's useful nonetheless. You could think of the Picture object as an invisible picture box that you can use as a staging area for images. For example, the following code loads a Picture object with a bitmap and uses that bitmap to set the Picture property of a picture box control:

```
Private Sub Command1_Click()  
    Dim objPic As Picture  
    Set objPic = LoadPicture("Butterfly.gif")  
    Set Picture1.Picture = objPic  
End Sub
```

The Picture object supports bitmaps, GIF images, JPEG images, metafiles, and icons.

PSet Method:

To set a single point in a graphic object (form or picture box) to a particular color, use the **PSet** method. We usually do this to designate a starting point for other graphics methods. The syntax is:

`ObjectName.PSet (x, y), Color`

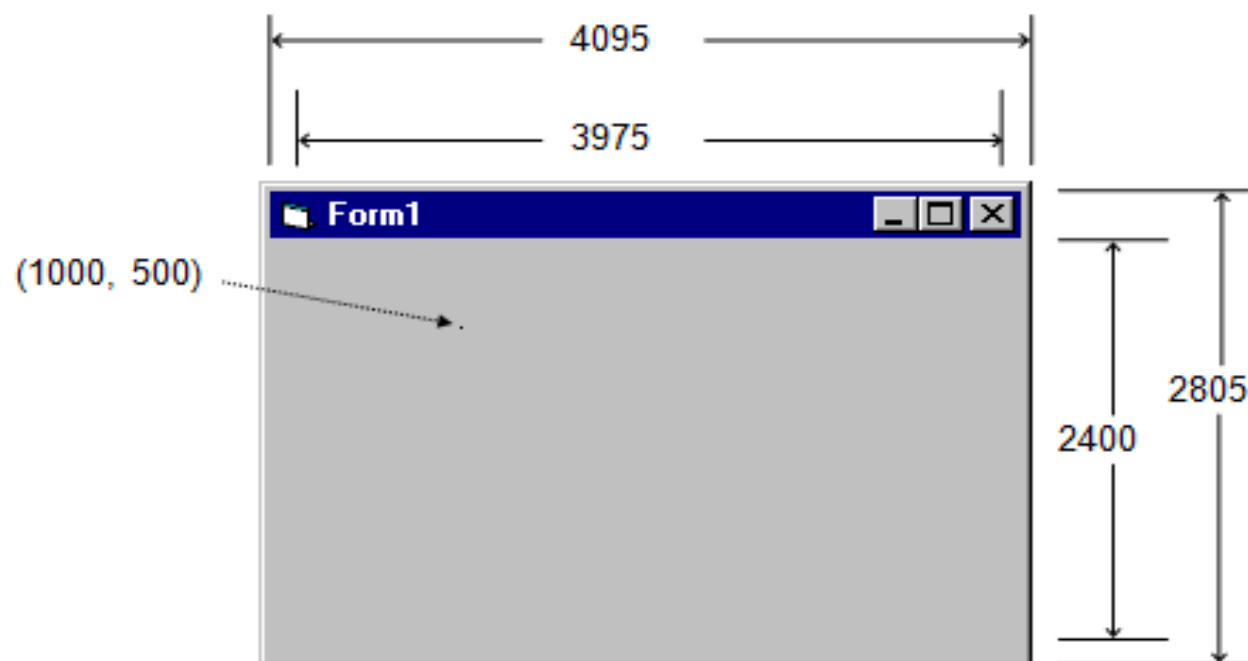
where **ObjectName** is the object name, **(x, y)** is the selected point, and **Color** is the point color (discussed in the next section). If the **ObjectName** is omitted, the current form is assumed to be the object. If **Color** is omitted, the object's **ForeColor** property establishes the color. **PSet** is usually used to initialize some further drawing process.

Pset Method Example:

This form has a ScaleWidth of 3975 (Width 4095) and a ScaleHeight of 2400 (Height 2805). The command:

PSet(1000, 500)

will have the result:



Unit-5

Working with file

File: A file is a collection of information in a organized way on a given subject, stored on a storage medium, usually a disk or CD. There are many types of file such as

- An executable files with “.exe” extension.
- A library files with “.DLL” extension.
- Word document file “.doc” extension and hundred other types.

The information is formed by arranging the data into rows and column.

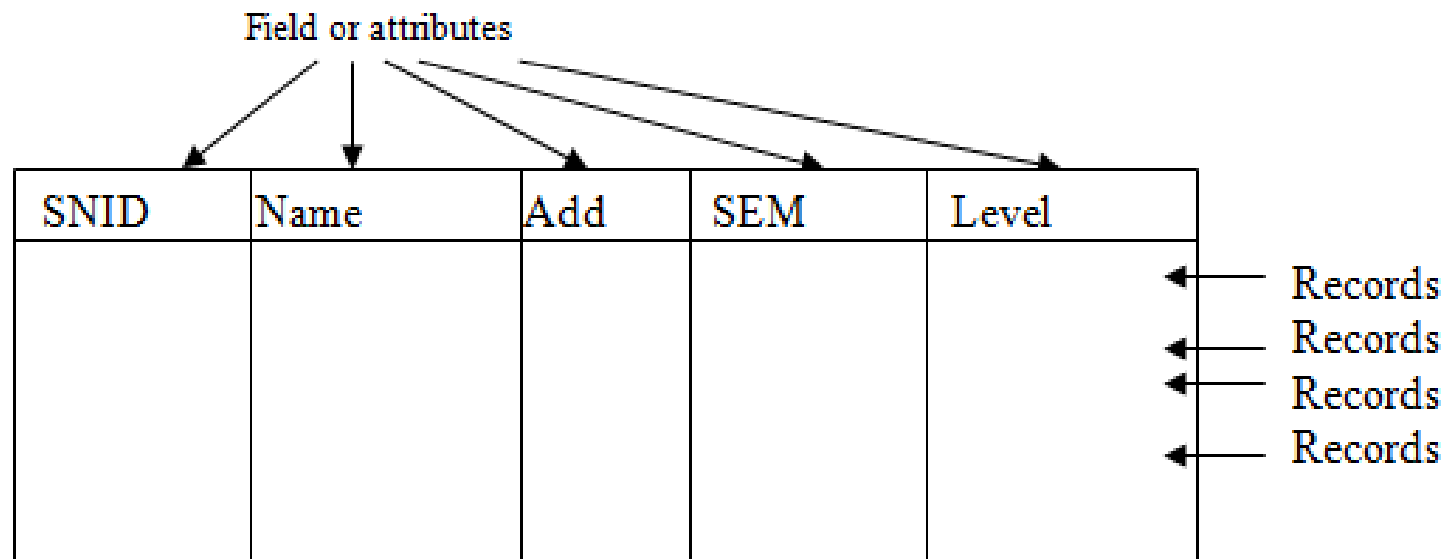
- Each individual rows and called record (types)
- And the collection of similar type of data in a column is called attributes (field)
- Hence we can say that collection of multiple columns related data arranged in a row is called record and collection of record are called data file.

- **Records:**

It is a logical section of a file that holds a related set of data.

- **Field:**

It is a part of record that defines specific information. In e.g. student info, file above student ID, name, add, SEM, level are fields.



Student info → file name

In VB the user can create and access three types of data files.

Sequential access file:

This is the file where all the information is read written in order from the beginning to the end.

To access a given record you have to read all the records stored before it. This is similar to listening to cassette player or tape.

This method was used in old days when magnetic tapes were used mostly for file or data storing and all the files were organized in this way.

It is still useful when there is a small amount of data to store.

Random access file:

It is the file where all records are accessible individually.

It is like a CD where you can jump to any track.

This is useful when there is a large quantity of data to store and it has to be available quickly.

These methods of storage become popular when hard disk drives were developed.

Binary access file:

This is a special compacted form of the random file.

Data are stored at the byte level and you can read and write individual bytes to the file.

This makes the file access very fast and efficient.

Opening and closing file:

The open commands assign the file to a numbered file handle. The format of the open command is:

Open “file name” [for mode][access restriction][lock type]as # file number.

For example:

Open “myfile.txt” for random read lock read as #1

Here,

“myfile.txt” is the name of file on disc.

For random means access to the records can be random. If access is not specified for random is default value.

Read for restricts access to read only.

Lock read mean that only the person reading the record can have access to it at any given time, it is not shared among users.

As # 1 means file is assigned file handle #1 for all processing in programmed. It will always be referred to as # 1, not its file name.

Note: access restriction and lock type, parameters are used mostly with files in a network environment.

Access mode:

For mode in the open statement indicates how the file will be used. There are access modes areas are as:

- **Input:**
Open for sequential input the file will be read sequentially starting at the beginning.
- **Output:**
Open for the sequential output records will be written sequentially starting at the beginning.
- **Random:**
Open for the random read and write any specified record can be accessed.
- **Append:**
Sequential output to end of existing file.
- **Binary:** Open for binary read/write access is at byte level. Once processing is finishing, we need to close all file that was opened. The format for close statement is:
Close # file number 1 [3file number 2].....

For example:

Close #1, #2, #3.

The above statement closes all open files.

Writing to reading form a sequential file: There are two methods that allow us to write data to sequential files: print and write.

The format of write command is:

Write # file number, output list.

Where,

File number is the no the file was opened.

Output list is one or more variable we want to write to the file.

Write and reading records from a random file:

The command to write records to the random file is put. Its format is:

Put # file number [record number], variable.

Where,

Record number is optional.

The command to give records from a random file is get and its format is:

Get # file number, [record number] variabl

Extending VB with OLE:

OLE stands for “object linking and embedding” lets VB application access in the functionality of other application in the windows environment. The controls of OLE are built in VB editor’s toolbox. They can be placed on form with pointer and click operation. Some object in the window environment, however are not unique to VB and do not come with language. They are supplied by other application but can be used within VB application by using OLE automation.

Understanding OLE terminology

OLE object:

- It is an item that is exposed or made available by an OLE server application.
- An item that is exposed or made available from other application by another application is called OLE object.
- The 3rd party application which exposes the ole object to VB application or VB form is called ole server.
- Where as the VB application or form which uses the ole object exposed by server application are called “container application”
- Container applications are those applications which use the ole object. For e.g. forms in VB which contains the controls.

Object linking and embedding:

Object embedding:

with this technique we can insert an object from one application (the server application) into another application (container application)

The inserted object is a copy of original and can be manipulated and stored separately and a part from the original object. The changes may not effect the original location.

Object linking:

Changes to the object in the server application are reflected automatically in the container application.

Linking does not store the object; it makes a reference to the object exposed by the server application.

Linking objects are not copies, they are originals viewed from within different containers.

An example of OLE:

- Start
- choose object from insert menu to display “insert object dialog box”
- check the create from file option
- If linking has to be done then also check link check box.
- click on browse button
- Select the required file name to embedded or linked.
- Click on ok.

OLE container control:

- The OLE container control allows the user or programmer to insert object from other application into the currently opened application or in your program. We can place only one object in the containers control at a time where as on the same form multiple OLE container control can be placed, each one with its own objects.
- The ole container controls is the door to the various objects available within the OS that VB can't handle on its own, including word document, sounds, bitmaps and excel spread sheet.
- The ole container control can host any of the objects listed in the insert object dialog box and make our VB application act like a container application.
- The good news is that the ole container control hides nearly all the complexity of ole and allows building ole capable applications.
- We can embed object in an ole container control at design time or at runtime.

Unit-6

Working with database

Database :

A database is a collection of information that is organized so that it can easily be accessed managed and update. There are two type of database:

- Conventional Database: Collection of data without use of computer. E.g. Telephone diary, register.
- Computerized database: The collection of data with the help of computer in the form of data file is known as computerized database. The types of computer systems that can run database management system are Centralized PC and Client/Server and distributed.

Elements of Database:

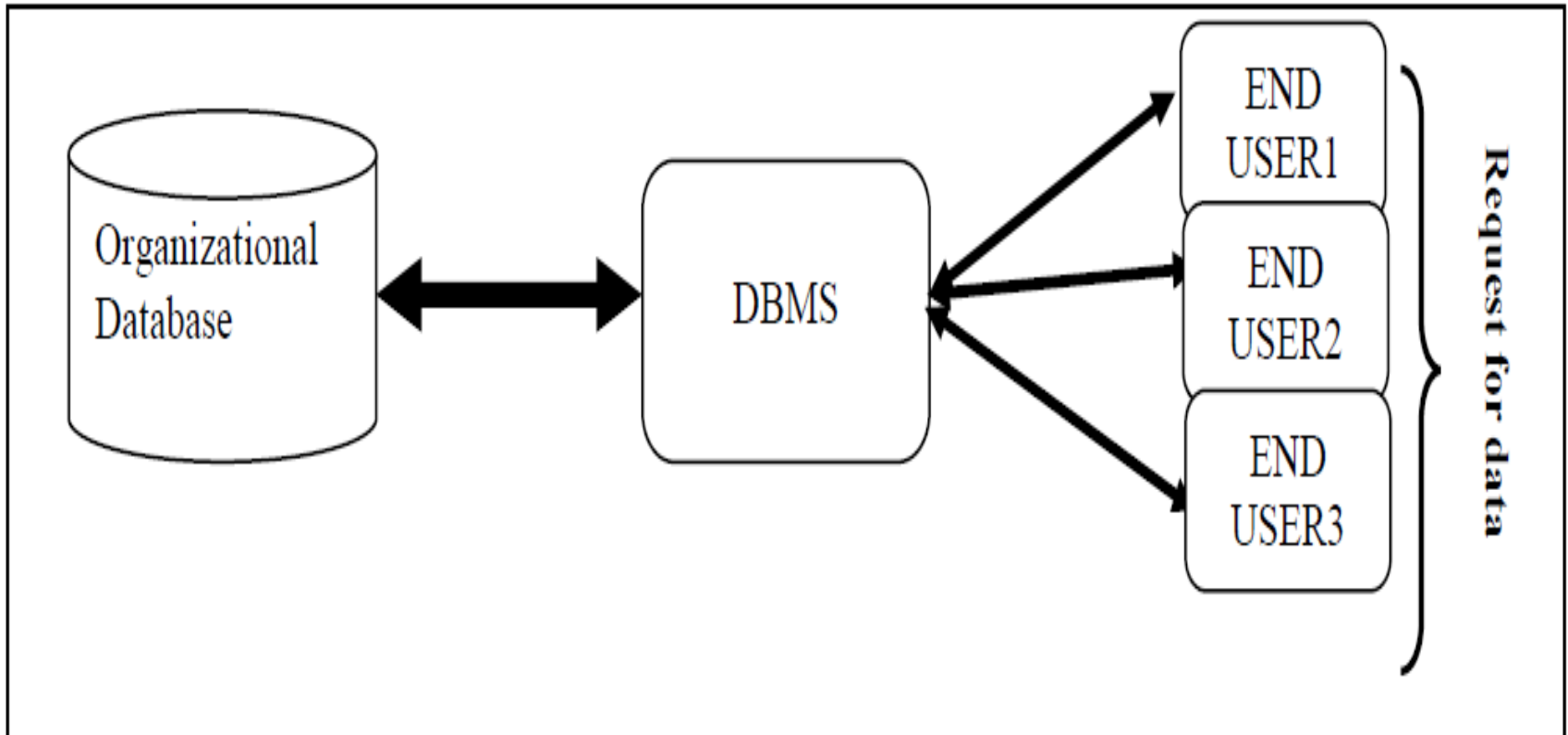
- 1. Field: Smallest unit of Database.**
- 2. Record: A Collection of multiple related fields.**
- 3. Table: A collection of records or group of records with the row and column order.**
- 4. Tuple: A record row in the database.**
- 5. Index: It is a process of organizing data in specific order.**
- 6. Cell: A cell is a inter section of rows and columns.**

DBMS (Database management system):

Database management systems (DBMSs) are specially designed application software that interact with the user, other applications, and the database itself to capture and analyze data. A general-purpose DBMS is a software system designed to allow the definition, creation, querying, update, and administration of databases.

Well-known DBMSs include [MySQL](#), [MariaDB](#), [PostgreSQL](#), SQLite, Microsoft SQL Server, Oracle, SAP HANA, [dBASE](#), FoxPro, IBM DB2, LibreOffice Base, FileMaker Pro, Microsoft Access .

In this figure illustrate that the DBMS stands between the database and the users.



visual data manager:

- The visual data manager is a VB tool for designing database. We can use the visual data manager to create and modify tables to implement security and to experiment with SQL.
- It is an add-ins program available in VB using which developer can create own database to be used in the application interface (API).
- The visual data manager creates a set of database just like ms access having an extension.mdb.
- The visual data manager can be loaded or started by selecting add-ins menu from menu bar and then select visual data manager.

How to create table using visual data manager:

- Start visual data manager by selecting add-ins menu from menu bar and then select visual data manager. Then a visual data screen will appear.
- From file menu select new, then from the list of database type select Microsoft access and version 7.0 MDI or any updated.
- Now type the name of database as “student.mdb”
- Next the database window will appear.
- Now by placing the mouse pointer inside the database window right click and select new table another dialog box will appear for defining table structure as given below:

Table structure

Field list

Table name:

Name:

Type:

Size:

Validation text:

Validation rule:

Default value:

Add field

Remove field

Add index

- Type the name of the table in the table name box.
- Type the field name as stud_id in the name box.
- Type the field type as integer and give the size as 5.
- After defining the first field structure, click on add field button to add next field. Then another dialog box will appear, in that dialog box define the structure of next field. At least click on close button then it will return to table structuring dialog box.
- Similarly by clicking add field button define all the fields.
- To make stu_id as primary key, click on add index button, then select the required field list and type the primary key as PK field name and click ok.
- At last click on build the table button.

Exploring the structure of BIBLIO database:

- One of the two sample database that comes with visual basic is called BIBLIO.
- It is a readymade database supplied with visual basic by Microsoft Company.
- It can be explored and used inside the visual data manager application on of the DBMS supporting built in program under VB.
- The BIBLIO database has defined no of normalized forms of table having defined relationship between the attributes of the table.
- To explore it we can follow the given steps:
 - Start visual data manager.
 - Select open from file menu.
 - Search the BIBLIO.mdl and open it.

The BIBLIO database contains book titles, authors, and publishers and it's made up of four tables.

Validation the data entry: The validation of data entry in the DBMS through VB can be applied at two different levels.

- **At system level validation:** This validation can be implemented during the definition of table structure in the VB of data manager when database is created and in the table structure definition dialog box setting the validation rule, validation value and the validation text (message).
- **At application level:** In the application level the user (developer) can develop a validating event procedure in the application in such a way that whenever any new in such a way that whenever any new data is entered to the table or data base the event procedure (validating event) automatically executed and validation the entered value. If the entered value, satisfies the validation rule defined inside the event procedure then that value is entered to the database and committed (saved), otherwise error message will appear.

Entering data:

The data control is a great tool for browsing tables and editing their contents, but how about entering new information or deleting existing records. These actions require a few lines of code.

To write data entry application, we use the following methods:

- Add new = appends a record to the table.
- Delete = deletes the current record
- Update = writes the current record to database.
- Refresh = reloads the data from the database.

Accessing fields in record set:

- The fields or values of any record can be accessed via the fields object of the record set. The following expression: `recordset.fields`
- The record set variable represents a records set.
- We can access the individual fields through name or through fields ordinal position in the table.
- `Booktitle = data1.recordset.fields(0)`
- `Booktitle = data1.recordset.fields("title")`
- If data1 data control is connected to the titles tables of BIBLIO database, we can access the title fields of the current record with the either of above statements.

Advanced data bound controls:

In VB are other advanced data bound controls which are as below:

- The data bound list control
 - The data bound control box control
 - The data bound grid control.
- Unlike the other bound controls these controls can display fields from multiple rows.
- The data bound list control is similarly to regular list box control can be populated with an entire column of a record set.
- The data bound grid control is similar to the grid control and can display an entire record set.
- Each row of the grid holds a row (records) of the record set and the record sets column correspond to the columns (fields) of record set.
- These two controls list and grid aren't installed by default to use them we must first add them to the toolbox, to do so follow:
 - Right clicks the toolbox and select components to open the components dialog box.
 - Check the Microsoft data bound grid control and the Microsoft data bound list control boxes.
 - Click the close button to close the components dialog box.

Active data objects:

- Visual basic supports several data access tools with the active data object (ADO) being the most recent addition.
- Where as the first VB data access tools (the data access object) allowed programmer to access. ms access database all major database is Microsoft's foundation for a universal technology for accessing all types of data in all environment.
- With ADO, to access any database VB needs 3 objects.
- a connection object, which establishes a connection to the database.
- A command object, which executes commands against database.
- A record set object, which holds the records retrieved from the database on the records to be updated on database.

Creating a data project:

- To see the new VB tools i.e. ADO in action, start a new project and in project type dialog box, select data project.
- Its not a special project type, just basically a standard exe project but VB loads all the database tools into the projects tool box.
- In the project explorer window, VB displays a new form, and two active x designers.
- A designer is a special add INS that simplifies the design of components. Database rely on two basic components:
 - one or more data environment component
 - One or more data report component.

To access database with ADO we need two types of objects:

- One or more connection objects
- One or more command objects.

The data projects tool box also contains a no of active x controls.

The ADODC (ADO data control) which is same as data control.

The data list and data combo controls which are data bound versions of the list box and control box they work with the ADO data control.