VISUAL PROGRAMMING

DEC-2020

PART-C

20. Explain about the usage of menus in VB with examples.

Menus are one of the most important parts of a graphical user interface (GUI). In Visual Basic (VB), menus provide a structured, user-friendly way for users to access commands, operations, and features in an application. They make software easier to use, more organized, and more professional.

This answer covers definition, purpose, types, menu creation, properties, events, examples, advantages, and best practices—suitable for full 25 marks.

✅ 1. Meaning of Menus in Visual Basic

A menu is a set of commands listed in a drop-down list, which appears when the user clicks on a menu title such as *File*, *Edit*, *View*, or *Help*.
Menus guide the user through different actions in a clear and organized way.

In VB, menus are usually created using the Menu Editor, and they appear at the top of a form.

✅ 2. Purpose of Menus in VB

Menus are used in VB applications for the following purposes:

✅ (a) Provide easy navigation

Users access features without typing commands.

✅ (b) Group related functions

For example, *File* menu contains Open, Save, Exit.

✅ (c) Save screen space

Instead of placing many buttons on the form, menus hide options until the user needs them.

✅ (d) Improve usability

Menus follow standard structure, making applications intuitive.

✅ (e) Make programs professional

Menus are essential for business, office, and commercial applications.

✅ 3. Types of Menus in VB

✅ (1) Main Menu

Appears at the top of the form (e.g., File, Edit, Help).

✅ (2) Submenu

Appears under a main menu (e.g., File → Open).

✅ (3) Popup Menu / Context Menu

Appears when the user right-clicks.
Used for quick access to commands such as Cut, Copy, Paste.

Example:

PopupMenu mnuEdit

✅ (4) Separator Line

A horizontal line used to group related items.
Caption is given as "-".

✅ 4. Creating Menus Using Menu Editor

✅ Step 1: Open Menu Editor

Go to Tools → Menu Editor

Shortcut key: Ctrl + E

✅ Step 2: Enter Caption

Caption is the text shown to the user.
Example: &File (Alt + F will open it)

✅ Step 3: Enter Name

Internal name used in coding.
Example: mnuOpen, mnuExit

✅ Step 4: Set Hierarchy

Use Indent → to create submenu

Use Outdent → to move it to main menu level

✅ Step 5: Add Shortcut Keys

VB provides shortcuts like:

Ctrl + N

Ctrl + S

Ctrl + P

✅ Step 6: Create Separators

Enter "-" in Caption.

✅ Step 7: Set Properties

Checked – shows a tick mark

Enabled – enables or disables item

Visible – hides or shows the menu

✅ Step 8: Write Event Code

Double-click a menu in the form to write the event procedure.

✅ 5. Menu Properties

✅ (a) Name

Used in code.

✅ (b) Caption

Displayed text.

✅ (c) Index

Used for creating menu arrays.

✅ (d) Shortcut

Keyboard shortcut.

✅ (e) Visible / Enabled

Control availability of commands.

✅ (f) Checked

Used for options like "Word Wrap" in Notepad.

✅ 6. Menu Events

Menus trigger events when clicked.

✅ Click Event

Executed when a menu item is selected.

Example:

```
Private Sub mnuExit_Click()
    Unload Me
End Sub
```

✅ Popup Events

Used for right-click menus.

✅ 7. Examples of Menus in VB

✅ Example 1: File Menu

Menu Structure

File

New

Open

Save

-

Exit

Code:

```
Private Sub mnuNew_Click()
    MsgBox "New File Created"
End Sub

Private Sub mnuOpen_Click()
    MsgBox "Open File Dialog"
End Sub

Private Sub mnuSave_Click()
    MsgBox "File Saved"
End Sub

Private Sub mnuExit_Click()
    Unload Me
End Sub
```

✅ Example 2: Edit Menu with Popup Menu

Popup Menu Creation:

Edit

Cut

Copy

Paste

Show Popup Menu:

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Button = vbRightButton Then
        PopupMenu mnuEdit
    End If
End Sub
```

✅ Example 3: Toggle Menu (Checked Property)

Used for ON/OFF settings like Word Wrap.

```
Private Sub mnuWordWrap_Click()
```

```
    mnuWordWrap.Checked = Not mnuWordWrap.Checked
End Sub
```

✅ 8. Advantages of Using Menus in VB

✅ 1. Clean and organized user interface

Menus prevent cluttering of buttons.

✅ 2. Easy to use

Users are familiar with menu structures.

✅ 3. Supports shortcut keys

Improves speed and accessibility.

✅ 4. Uses less screen space

Ideal for complex applications.

✅ 5. Professional and standard look

Makes VB applications look like real software.

✅ 6. Easy to edit and maintain

Menu Editor simplifies changes.

✅ 7. Supports event-driven programming

Each menu item triggers actions via events.

✅ 8. Dynamic menus possible

Menus can be shown or hidden based on conditions.

✅ 9. Real-World Applications of Menus

Menus are used in:

Text editors

Billing software

Accounting applications

Library management systems

Inventory control systems

Notepad, WordPad

Database front-end applications

Almost every GUI application uses menus.

✅ 10. Best Practices for Menu Design

Group similar commands logically

Keep menu names short and clear

Use shortcut keys for commonly used actions

Use separators to organize long menus

Disable unnecessary options using Enabled = False

Use Checked option for toggle settings

Avoid overcrowding menu bar with too many items

✅ Conclusion

Menus are an essential part of Visual Basic applications, providing structured navigation, better user experience, and professional interface design. Using the Menu Editor, programmers can easily create main menus, submenu items, and popup menus. With properties like Caption, Name, Shortcut, and events like Click, menus become powerful tools for handling user commands. They help build clear, efficient, and user-friendly applications.

21. Explain the following :

(a) Control arrays

(b) DOEVENT and sub menu.

✅ (a) CONTROL ARRAYS

A control array in Visual Basic is a group of controls that share the same name and same set of event procedures, but differ by their Index property.
Control arrays allow programmers to create multiple similar controls efficiently and handle them with one block of code.

✅ 1. Meaning of Control Array

A control array consists of multiple instances of a control (such as TextBox, CommandButton, Label, etc.) that share:

Same name

Same type

Common event procedures

Each control is uniquely identified by an Index number.

✅ 2. Purpose of Control Arrays

Reduce code duplication

Allow dynamic addition and removal of controls

Useful for creating multiple similar controls like list rows, input fields, options, etc.

Simplify event handling

✅ 3. Creating a Control Array

You can create a control array in two ways:

✅ Method 1: At Design Time

Place a control (e.g., TextBox1) on the form.

Copy and paste it. VB asks:
"Do you want to create a control array?"

Click Yes.

The controls will now have indices:

TextBox1(0)

TextBox1(1)

✅ Method 2: At Runtime

Use the Load statement:

Load Text1(1)

Text1(1).Visible = True

✅ 4. Example of Control Array

Example: Clicking any button in a control array

```
Private Sub Command1_Click(Index As Integer)
    MsgBox "You clicked button number: " & Index
End Sub
```

This single event works for all buttons in the array.

✅ 5. Advantages of Control Arrays

Saves coding effort

Helps manage multiple similar controls

Allows dynamic UI creation

Efficient memory usage

Clean and organized program structure

✅ (b) DOEVENTS AND SUBMENU

✅ (b1) DOEVENTS

DoEvents is a statement in VB used to yield control temporarily to the operating system so that VB can process other events while a program is running.

✅ 1. Purpose of DoEvents

Allows the GUI to remain responsive during long operations

Prevents the application from freezing

Allows VB to process keystrokes, mouse clicks, and screen updates

✅ 2. Example of DoEvents

Without DoEvents

A long loop may freeze the program:

```
For i = 1 To 50000
    lblCount.Caption = i
Next
```

With DoEvents

```
For i = 1 To 50000
    lblCount.Caption = i
    DoEvents    'Keeps the program responsive
Next
```

✅ 3. Uses of DoEvents

Long calculations

Updating progress bars

Reading files

Animations

Avoiding “Program Not Responding” messages

✅ 4. Disadvantages of DoEvents

Can slow down performance

May cause unexpected user actions if misused

✅ (b2) SUBMENU

A submenu in Visual Basic is a menu item that appears under another menu item.
It helps in grouping related commands inside a main menu.

✅ 1. Purpose of Submenus

Organize commands hierarchically

Keep the main menu clean

Group related options (e.g., File → Open, Save, Close)

✅ 2. Creating a Submenu

Submenus are created using the Menu Editor by using the Indent button.

Example Structure:

File

  Open

  Save

Exit

✅ 3. Example of Submenu Event

```
Private Sub mnuOpen_Click()
    MsgBox "Open selected"
End Sub

Private Sub mnuExit_Click()
    Unload Me
End Sub
```

✅ 4. Uses of Submenus

Creating multi-level menus

Organizing complex applications

Providing step-by-step user options

Grouping similar features (e.g., Edit → Cut, Copy, Paste)

✅ ✅ FINAL SUMMARY (For 25 Marks)

(a) Control Arrays

Group of controls with same name and type

Identified using Index

Created at design time or runtime

Useful for reducing code and creating dynamic controls

Example and advantages included

(b) DoEvents

Allows VB to process other events during long operations

Keeps UI responsive

Used in loops, animations, file operations

Example included

(c) Submenu

A child menu under a main menu item

Created using Menu Editor with indent

Helps organize and structure menu commands

Simple coding example included

22. How to work with graphics in visual basis?

Visual Basic provides built-in tools and methods to draw shapes, display images, and perform graphical operations on forms and picture boxes. Working with graphics is useful for building drawing programs, charts, games, simulations, and visual user interfaces.

This answer covers methods, properties, events, examples, objects, and step-by-step usage—ideal for full marks.

✅ 1. Introduction to Graphics in Visual Basic

Visual Basic supports graphics using:

Form object

PictureBox control

Graphics methods (Line, Circle, PSet, PaintPicture)

Image controls

API functions (advanced)

VB uses a coordinate system:

Origin (0, 0) at top-left corner

X increases rightwards

Y increases downwards

✅ 2. Graphics Containers in VB

✅ (a) Form

The form itself can be used as a canvas for drawing shapes, lines, and displaying images.

✅ (b) PictureBox

Most commonly used for graphics:

Supports persistent drawing

Allows image loading

Supports AutoRedraw

✅ (c) Image Control

Used mainly for displaying images, not drawing.

✅ 3. Important Graphics Properties

✅ (1) AutoRedraw

If set to True, drawings remain on the form even after window refresh.

Prevents graphics from disappearing.

✅ (2) BackColor

Sets background color of a graphic area.

✅ (3) ScaleMode

Defines coordinate system:

Twips (default)

Pixels

Inches

User-Defined

✅ (4) Font, ForeColor

Used for drawing text.

✅ 4. Basic Graphics Methods in VB

Visual Basic provides several drawing commands:

✅ (a) PSet

Used to draw a single pixel (point).

Syntax:

PSet (x, y), vbRed

✅ (b) Line

Draws straight lines or filled boxes.

Syntax:

Line (x1, y1)-(x2, y2), vbBlue

To draw a filled box:

Line (x1, y1)-(x2, y2), vbGreen, BF

✅ (c) Circle

Used to draw circles, arcs, and ellipses.

Syntax:

Circle (x, y), radius, vbYellow

✅ (d) PaintPicture

Used to draw images or copy part of an image.

Syntax:

Picture1.PaintPicture Picture2.Picture, 0, 0

✅ (e) Print Method

Prints text on a Form or PictureBox.

Picture1.Print "Hello Graphics"

✅ 5. Using Images in VB

✅ (a) Loading Images

You can load images like BMP, JPG, GIF into PictureBox or Image control.

```
Picture1.Picture = LoadPicture("C:\image.jpg")
```

✅ (b) Stretching Images

```
Image1.Stretch = True
```

✅ (c) Clearing Graphics

```
Picture1.Cls
```

✅ 6. Working With Events for Graphics

✅ (a) MouseMove Event

Used for drawing while dragging the mouse.

Example: Freehand drawing

```
Private Sub Picture1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Button = 1 Then
        PSet (X, Y)
    End If
End Sub
```

✅ (b) Form_Paint Event

Redraws graphics when the form refreshes.

✅ 7. Example Programs

✅ Example 1: Drawing a line

```
Private Sub Command1_Click()
    Picture1.Line (0, 0)-(2000, 2000), vbRed
End Sub
```

✅ Example 2: Drawing a circle

```
Private Sub Command2_Click()
    Picture1.Circle (1500, 1500), 1000, vbBlue
End Sub
```

✅ Example 3: Displaying and stretching an image

```
Image1.Stretch = True
Image1.Picture = LoadPicture("C:\flower.jpg")
```

✅ Example 4: Clear drawing

Picture1.Cls

✅ 8. Advanced Graphics Features

✅ (a) Animation

Using timers to move shapes.

✅ (b) Double Buffering

Reducing flicker by drawing off-screen.

✅ (c) Using API functions

For advanced features like transparency, rotation, and filtering.

✅ (d) Drawing on memory DC

For games and simulations.

✅ 9. Differences Between PictureBox and Image

Feature	PictureBox	Image Control
Used for drawing	✅ Yes	❌ No
Has AutoRedraw	✅ Yes	❌ No
Faster for simple display	❌ Slower	✅ Faster
Supports graphics methods	✅ Yes	❌ Limited

✅ 10. Applications of Graphics in VB

Drawing programs

CAD tools

Simple games

Charts and graphs

Animations

Image editors

Signature capture

Simulation programs

✅ Conclusion

Working with graphics in Visual Basic involves using PictureBox or Form as a drawing surface and applying methods like PSet, Line, Circle, and PaintPicture. VB supports both simple and advanced graphical operations. Through properties like AutoRedraw, ForeColor, and ScaleMode, and by handling mouse events, VB provides a powerful and flexible way to develop graphical applications, animations, and interactive drawing tools.

23. Explain about monitoring mouse activity with examples.

Monitoring mouse activity is an important feature in Visual Basic (VB). Almost every graphical application depends on mouse events for interacting with controls, forms, and objects. Visual Basic provides several mouse events, mouse-related properties, and built-in constants to track every movement and action of the mouse.

This answer includes definitions, events, properties, and multiple examples—suitable for full marks.

✅ 1. Introduction to Mouse Activity in VB

Mouse activity refers to actions performed by the user using the mouse, such as:

Moving the mouse

Clicking buttons

Double-clicking

Pressing and holding a button

Releasing a button

Dragging the mouse

VB allows programmers to detect these actions using mouse events for:

Forms

PictureBoxes

CommandButtons

Labels

Any VB control

✅ 2. Mouse Events in Visual Basic

VB provides several mouse-related events to monitor activity:

✅ (1) MouseDown Event

Occurs when the user presses any mouse button.

Syntax:

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

Uses:

Detect which button is pressed

Start drag-and-draw operations

Selecting objects

✅ (2) MouseUp Event

Occurs when the user releases a mouse button.

Uses:

Completing drag-drop operations

Ending drawing of shapes

✅ (3) MouseMove Event

Occurs whenever the mouse pointer moves over a control.

Uses:

Displaying coordinates

Drawing lines while dragging

Creating painting applications

✅ (4) Click Event

Occurs when a mouse button is pressed and released.

✅ (5) DblClick Event

Occurs when the user double-clicks a control.

Uses:

Opening files

Selecting items

Editing content

✅ 3. Mouse Event Parameters

Most mouse events include:

✅ (a) Button

Indicates which mouse button was pressed:

vbLeftButton = 1

vbRightButton = 2

vbMiddleButton = 4

✅ (b) Shift

Indicates whether special keys are pressed:

1 = Shift key

2 = Ctrl key

4 = Alt key

✅ (c) X, Y

Coordinates of the mouse pointer inside the control.

✅ 4. Examples of Monitoring Mouse Activity

✅ Example 1: Display Mouse Coordinates

```
Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    Label1.Caption = "X = " & X & "  Y = " & Y
End Sub
```

✅ Use: Shows live mouse position.

✅ Example 2: Detect Mouse Click Type

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

    If Button = vbLeftButton Then
        MsgBox "Left button clicked!"
    ElseIf Button = vbRightButton Then
        MsgBox "Right button clicked!"
    End If

End Sub
```

✅ Use: Differentiates left and right clicks.

✅ Example 3: Simple Drawing Using MouseMove

```
Private Sub Picture1_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Button = vbLeftButton Then
        Picture1.PSet (X, Y), vbRed
    End If
End Sub
```

✅ Use: Basic freehand drawing (paint program).

✅ Example 4: Using Double-Click Event

```
Private Sub List1_DblClick()
    MsgBox "You double-clicked: " & List1.Text
End Sub
```

✅ Use: Common in file browsers (open file on double-click).

✅ Example 5: Drag the Form with Mouse

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    MouseDownX = X
```

```
    MouseDownY = Y
End Sub

Private Sub Form_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Button = vbLeftButton Then
        Me.Move Me.Left + (X - MouseDownX), Me.Top + (Y - MouseDownY)
    End If
End Sub
```

✅ Use: Move form without a title bar.

✅ 5. Advanced Mouse Monitoring

✅ Dragging and Dropping

MouseDown → Start
MouseMove → Drag
MouseUp → Drop

✅ Drawing shapes

Use MouseDown for starting point and MouseUp for ending point.

✅ Custom cursors

Use MousePointer property.

6. Applications of Mouse Activity Monitoring

Drawing applications (Paint)

Games (catching objects, movement control)

Image editors

Drag-and-drop interfaces

Graphic simulations

Interactive forms

Data selection tools

✅ 7. Advantages of Monitoring Mouse Activity

Enhances user interaction

Makes applications intuitive

Allows interactive drawing

Enables drag-and-drop

Useful for games and animations

Supports advanced UI features

✅ Conclusion

Monitoring mouse activity is essential in Visual Basic for developing interactive and user-friendly applications. By using events like MouseDown, MouseUp, MouseMove, Click, and DblClick, VB can detect and respond to mouse actions. These events, combined with parameters like Button and coordinates, allow developers to create drawings, drag-and-drop operations, custom interfaces, and dynamic graphical applications.

24. Write a note on :

(a) File handling

(b) DLL servers.

✅ (a) FILE HANDLING (Detailed Notes)

File handling in Visual Basic refers to the process of creating, storing, reading, writing, and managing files on the disk. It allows a program to save data permanently, even after the program is closed. VB supports different types of files and functions for handling them.

✅ 1. Purpose of File Handling

Store data permanently

Read/write user information

Create text files, data files, log files

Transfer data between applications

Allow backup and retrieval of data

✅ 2. Types of Files in VB

✅ (1) Sequential Files

Data stored line by line

Best for text files

Accessed in order (top to bottom)

✅ (2) Random Access Files

Access any record directly using position

Useful for large databases

Data stored in fixed-length records

✅ (3) Binary Files

Store data in binary format

Suitable for images, audio, or special data

✅ 3. Important File Handling Commands

✅ (a) Open Statement

Used to open a file in different modes.

Syntax:

```
Open "C:\data.txt" For Input As #1
```

Modes:

Input – Read only

Output – Write (overwrites existing data)

Append – Add new data at the end

Random – Random access mode

Binary – Binary file operations

✅ (b) Close Statement

Closes an open file.

```
Close #1
```

✅ (c) Print #

Writes text into a file.

```
Print #1, "Hello World"
```

✅ (d) Write #

Similar to Print but adds commas and quotes for structured data.

✅ (e) Input #

Reads data from a file.

```
Input #1, Name, Age
```

✅ (f) EOF (End of File)

Used to detect the end of a file while reading.

```
While Not EOF(1)
    Input #1, line
Wend
```

✅ 4. Example: Writing to a File

```
Open "C:\test.txt" For Output As #1
Print #1, "Welcome to VB File Handling"
Close #1
```

✅ 5. Example: Reading from a File

```
Open "C:\test.txt" For Input As #1
While Not EOF(1)
    Line Input #1, text
    Print text
Wend
Close #1
```

✅ 6. Advantages of File Handling

Permanent storage

Easy transfer of data

Efficient handling of large data

Supports different file types

✅ (b) DLL SERVERS (Detailed Notes)

DLL stands for Dynamic Link Library.
A DLL Server is a file containing reusable code or functions that can be shared by multiple applications at runtime.

DLLs improve memory usage, reduce code duplication, and allow modular programming.

✅ 1. Meaning of DLL Server

A DLL is a module that contains:

Functions

Procedures

Classes

Resources (icons, images, messages)

These can be loaded dynamically by programs when needed.

Example file: user32.dll, kernel32.dll

✅ 2. Purpose of DLL Servers

To share common code among many applications

To reduce compilation time

To minimize memory usage

To update or fix functionality without rebuilding the entire program

To build modular and expandable applications

✅ 3. Types of DLLs

✅ (1) In-Process DLL (.dll)

Runs inside the memory space of the calling application

Fast execution

✅ (2) Out-of-Process DLL (EXE server)

Runs as a separate process

Used in COM and ActiveX EXE servers

✅ (3) ActiveX DLL

Specific for VB6

Used for object-oriented programming

Contains classes and reusable components

✅ 4. Advantages of DLL Servers

Code reuse: Same DLL can be used by many programs

Memory saving: Only one DLL loaded for multiple applications

Easy updates: Updating the DLL updates all applications

Encapsulation: Hides internal code from application

Modular programming: Split application into parts

✅ 5. Example of Using a DLL in VB

Using Windows API DLL

```
Declare Function GetTickCount Lib "kernel32" () As Long
Private Sub Command1_Click()
    MsgBox "Milliseconds since system start: " & GetTickCount()
End Sub
```

✅ Here, kernel32.dll is used as a DLL server.

✅ 6. Creating an ActiveX DLL in VB

Steps:

Choose ActiveX DLL project type

Create classes (Public)

Compile the DLL

Reference DLL from VB project

✅ 7. Uses of DLL Servers

Handling system-level operations

Database operations

Common business functions

Encryption, security, and validation

Interfacing with Windows API

Building enterprise applications

✅ Conclusion

File handling in VB allows reading, writing, and managing data files using commands like Open, Close, Input, Print, and EOF. It ensures permanent data storage and supports text, random, and binary files.

DLL servers provide reusable code libraries shared by many applications. They support modularity, efficient memory use, and code reuse. DLLs play a crucial role in Windows programming and VB applications by supplying common functions and system-level operations.

DECEMBER 2021

PART-C

1. Explain the properties and methods of Text Boxes.

A TextBox is used to accept or display text in a VB application. It is one of the most commonly used controls.

A. Important Properties of TextBox

Text

Stores the actual text inside the TextBox.

Example: Text1.Text = "Hello"

Name

Identifier for the control to use in code.

MaxLength

Limits the number of characters the user can type.

Example: Text1.MaxLength = 10

Alignment

Sets text alignment: Left, Center, Right.

PasswordChar

Masks characters (useful for passwords).

Example: `Text1.PasswordChar = "*"``

Locked

If True, text cannot be edited.

MultiLine

Allows more than one line of text.

ScrollBars

Adds vertical or horizontal scrollbars in multiline mode.

Enabled

If False, user cannot interact with the box.

BackColor / ForeColor

Background and text color.

B. Methods of TextBox

SetFocus()

Moves the cursor to the TextBox.

SelStart / SelLength

Used to control text selection within the box.

Refresh()

Refreshes the control display.

C. Events

Change

Triggered when text changes.

GotFocus / LostFocus

Fires when the TextBox gets or loses focus.

D. Example Program

```
Private Sub Command1_Click()
    Text1.Text = UCase(Text1.Text)
End Sub
```

---

21. Explain Determinate and Indeterminate Loops with examples.

Loops help perform repeated actions.

A. Determinate Loops

A determinate loop runs a fixed number of times, known before execution.

Types

For…Next Loop

```
For i = 1 To 10
    Print i
Next i
```

2. For Each…Next Loop
Used for collections or arrays.

```
For Each x In List1.Items
    Print x
Next
```

Features

Beginning and ending limits are defined.

Loop counter changes automatically.

Easy to control.

B. Indeterminate Loops

Indeterminate loops run until a condition becomes true or false. The number of repetitions is unknown.

Types

Do While…Loop

```
Do While x < 100
    x = x + 1
Loop
```

Do Until…Loop

```
Do Until Total > 500
  Total = Total + 20
Loop
```

While…Wend

```
While a < 5
    a = a + 1
Wend
```

Features

Runs based on a condition.

Useful for searching, reading files, waiting for user action.

---

22. Narrate the Sub Main and DoEvents.

A. Sub Main

Sub Main is the starting point of a VB program when selected in project properties.

It allows programs to run without opening a form first.

Helpful for initialization tasks.

Used in console-style or utility programs.

Example

```
Sub Main()
   MsgBox "Program Started"
   Load Form1
   Form1.Show
End Sub
```

B. DoEvents

DoEvents allows VB to process pending events while a long loop is running.

Without DoEvents, the application may freeze.

It gives temporary control back to Windows.

Example

```
For i = 1 To 50000
   DoEvents
Next
```

Uses

Screen refresh

Allowing UI actions like clicking

Long calculations

Animation or timers

Together, Sub Main and DoEvents help in managing program flow and responsiveness.

---

23. Describe the usage of Control Array using example program.

A control array is a group of controls that share the same name and type, but have different Index values.

Uses of Control Arrays

Reduces repeated code.

Useful for controls created at runtime.

Saves memory and makes form lighter.

Easy to handle similar actions (like 10 buttons doing similar jobs).

Properties

All controls have the same name.

Each has an Index property.

Accessed using:
Text1(i).Text

Events

All controls in a control array share one event procedure.
Example:

```
Private Sub Command1_Click(Index As Integer)
   MsgBox "Button: " & Index
End Sub
```

Example Program

Task: Display the button number when clicked.

```
Private Sub Command1_Click(Index As Integer)
   MsgBox "You clicked Button No: " & Index
End Sub
```

If the form contains Command1(0), Command1(1), Command1(2), all clicks are handled by this one event.

Advantages

Cleaner code

Easy looping

Dynamic control creation

Easy maintenance

Useful for calculators, numeric keypads, quiz apps

---

24. Describe the sorting and searching methods.

A. Sorting Methods

Sorting is the process of arranging data in order (ascending or descending).

1. Bubble Sort

Repeatedly compares adjacent elements and swaps them.
Example:

```
For i = 0 To n-1
   For j = 0 To n-i-2
      If A(j) > A(j+1) Then
         Temp = A(j)
         A(j) = A(j+1)
```

```
            A(j+1) = Temp
        End If
    Next j
Next i
```

2. Selection Sort

Selects minimum element and places it at correct position.

3. Insertion Sort

Builds the final sorted list one element at a time.

Applications

Student mark lists

Employee salary lists

Inventory sorting

B. Searching Methods

Searching is used to find an item in a list.

1. Linear Search

Checks each element one by one.

Simple but slow for large lists.

Example:

```
Found = False
For i = 0 To n-1
    If A(i) = item Then
        Found = True
        Exit For
    End If
Next
```

2. Binary Search

Faster than linear search.

Works only on sorted lists.

Divides the list into halves repeatedly.

Example:

```
Low = 0
High = n - 1
```

```
Do While Low <= High
  Mid = (Low + High) \ 2
  If A(Mid) = item Then
     Found = True: Exit Do
  ElseIf A(Mid) > item Then
     High = Mid - 1
  Else
     Low = Mid + 1
  End If
Loop
```

Why Sorting and Searching are Important

Improve data retrieval speed.

Essential in databases, files, and arrays.

Used in billing systems, student records, inventory management, etc.

Dec 2022

PART-C

20. Explain in detail the Data Types available in Visual Basic.

Introduction:

In Visual Basic (VB), data types define the kind of data that a variable can hold, such as numbers, text, date, or logical values.

Each data type requires a specific amount of memory space and determines how operations are performed on that data.

Choosing the correct data type helps in efficient memory usage, faster program execution, and error-free computation.

Variables are declared in VB using the Dim statement along with the data type.

Example:

```
Dim Name As String
Dim Age As Integer
Dim Salary As Double
```

1. Numeric Data Types:

Numeric data types are used to store numbers — both whole and decimal values.

Data Type	Storage Size	Range	Description	Example
Byte	1 byte	0 to 255	Stores small positive integers.	Dim b As Byte = 200
Integer	2 bytes	-32,768 to 32,767	Stores whole numbers.	Dim x As Integer = 125
Long	4 bytes	-2,147,483,648 to 2,147,483,647	Stores large whole numbers.	Dim num As Long = 300000
Single	4 bytes	-3.4E–38 to 3.4E+38	Stores floating-point (decimal) numbers.	Dim a As Single = 12.56
Double	8 bytes	-1.8E–308 to 1.8E+308	Stores double-precision floating-point numbers.	Dim rate As Double = 2500.657
Currency	8 bytes	±922,337,203,685,477.5808	Stores monetary values with 4 decimal places.	Dim salary As Currency = 12500.75
Decimal	14 bytes	±7.9... × 10^28	Used for very large and precise numbers.	Dim val As Decimal = 123456.789

→ Use numeric types for mathematical and financial calculations.

2. String Data Type:

Used to store text or character sequences (letters, digits, symbols).

Type	Description	Example
Fixed-length String	Holds a fixed number of characters.	Dim str As String * 10
Variable-length String	Can store varying length of text.	Dim name As String = "Meena"

Example:

```
Dim StudentName As String
StudentName = "Divya Ayyappan"
```

→ String data types are useful for names, addresses, and text-based inputs.

3. Boolean Data Type:

Used for storing logical values: True or False.

Data Type	Storage Size	Description	Example
Boolean	2 bytes	Stores True or False values.	Dim isPass As Boolean = True

→ Used in conditions, comparisons, and decision-making statements.

Example:

```
If isPass = True Then
  MsgBox "Congratulations!"
End If
```

4. Date Data Type:

Stores date and time values in VB.

Data Type Storage Size Range Example

Date 8 bytes 1 Jan 100 – 31 Dec 9999 Dim today As Date = #11/13/2025#

Example:

```
Dim today As Date
today = Now
MsgBox "Today's Date and Time: " & today
```

→ Useful for programs involving scheduling, billing, or reports.

5. Object Data Type:

Used to hold references to objects, such as forms, controls, or external components.

Data Type Description Example

Object Can refer to any VB object. Dim obj As Object = Form1

Example:

```
Dim obj As Object
Set obj = Text1
obj.Text = "Hello"
```

→ Commonly used in object-oriented programming and automation.

6. Variant Data Type:

The most flexible data type that can hold any type of data—numbers, strings, or dates.

Data Type Description Example

Variant Can store any type of value. Dim v As Variant

Example:

```
Dim x As Variant
x = "Hello"
x = 123
x = #11/13/2025#
```

Note: Though flexible, Variant uses more memory and reduces execution speed. It is suitable when data type is unknown during coding.

7. User-Defined Data Types (UDT):

VB allows creation of custom data types using the Type keyword.

This helps group multiple variables of different data types under one structure.

Example:

```
Type Student
    RollNo As Integer
    Name As String
    Marks As Single
End Type
```

To use it:

```
Dim S As Student
S.RollNo = 101
S.Name = "Divya"
S.Marks = 89.5
```

→ Useful in applications like student records, employee details, or product management.

8. Summary Table;

Category	Data Types	Example
Numeric	Byte, Integer, Long, Single, Double, Currency, Decimal	Dim total As Double
Text	String	Dim name As String
Logical	Boolean	Dim valid As Boolean
Date/Time	Date	Dim billDate As Date
Object-based	Object	Dim obj As Object
General	Variant	Dim data As Variant
User-defined	Type ... End Type	Type Student ... End Type

9. Importance of Data Types:

Efficient Memory Use: Each data type uses memory according to its need.

Error Reduction: Prevents invalid operations like adding text to numbers.

Faster Execution: Numeric types perform faster calculations.

Data Accuracy: Correct data type ensures valid results.

Program Readability: Declaring data types makes code easier to understand.

Conclusion:

Data types are the foundation of programming in Visual Basic.

They specify what kind of values a variable can hold and how those values are processed.

VB offers a wide range of data types — Numeric, String, Boolean, Date, Object, Variant, and User-defined — to handle all kinds of data efficiently.

Using proper data types ensures accuracy, efficiency, and reliability of the program.

21. Discuss on the Concept of Sub Procedures in Visual Basic

Introduction:

In Visual Basic (VB), a Sub Procedure (short for Subroutine) is a block of code that performs a specific task but does not return a value.

It helps to break a large program into smaller, manageable, and reusable parts.

Each Sub Procedure can be called whenever needed, improving the structure, readability, and efficiency of the program.

Definition:

A Sub Procedure is a group of statements enclosed within Sub and End Sub keywords.

Syntax:

```
Sub ProcedureName ( [arguments] )
    'Statements to execute
End Sub
```

Example:

```
Sub DisplayMessage()
    MsgBox "Welcome to Visual Basic Programming!"
End Sub
```

Explanation:

Sub — used to start the procedure

DisplayMessage — the name of the procedure

End Sub — marks the end of the sub procedure

Features of Sub Procedures:

1. Do not return a value to the calling code.

2. Can be called multiple times from different places in the program.

3. May or may not take arguments (parameters).

4. Improve code reusability and organization.

5. Execute independently when called.

Types of Sub Procedures:

1. Standard (General) Sub Procedures:

These are defined in the General Declaration section of a form or module.

They can be called from any part of the program.

Example:

```
Sub GreetUser()
    MsgBox "Hello User!"
End Sub
```

To call it:

```
Call GreetUser()
```

2. Event-driven Sub Procedures:

These are automatically executed when a specific event occurs, such as a button click, key press, or form load.

Example:

```
Private Sub Command1_Click()
    MsgBox "Button clicked!"
End Sub
```

Explanation:

This procedure executes when the user clicks on the button Command1.

3. Parameterized Sub Procedures:

These Sub Procedures accept parameters (arguments), allowing you to pass data values.

Syntax:

```
Sub ProcedureName(ByVal arg1 As DataType, ByVal arg2 As DataType)
    'Statements
End Sub
```

Example:

```
Sub AddNumbers(ByVal a As Integer, ByVal b As Integer)
    Dim sum As Integer
    sum = a + b
    MsgBox "Sum = " & sum
End Sub
```

To call:

```
Call AddNumbers(10, 20)
```

4. Nested Sub Procedures:

One Sub Procedure can call another Sub Procedure inside it.

Example:

```
Sub MainProcedure()
    MsgBox "Main starts"
```

```
    Call SubProcedure()
    MsgBox "Main ends"
End Sub
Sub SubProcedure()
    MsgBox "Inside Sub Procedure"
End Sub
```

Output Sequence:

1. “Main starts”
2. “Inside Sub Procedure”
3. “Main ends”

Passing Arguments in Sub Procedures

Arguments can be passed in two ways:

Method	Keyword	Description	Effect:
ByVal	Pass by Value	A copy of the variable is passed. Changes inside the Sub do not affect the original variable.	Safe and preferred method
ByRef	Pass by Reference	The actual variable is passed. Changes inside the Sub affect the original value.	Used when modification is needed

Example:

```
Sub ChangeValue(ByRef x As Integer)
    x = x + 10
End Sub
Dim num As Integer
num = 20
Call ChangeValue(num)
MsgBox num   'Output: 30
```

Advantages of Sub Procedures:

1. Code Reusability: Once defined, it can be called multiple times.
2. Modularity: Divides complex programs into smaller, logical parts.
3. Easy Debugging: Errors are easier to find and correct.
4. Improves Readability: Code becomes organized and structured.
5. Encapsulation: Keeps related logic together in one block.
6. Maintenance: Updating one procedure updates all its usages.

Difference Between Sub Procedure and Function

Feature	Sub Procedure	Function Procedure
Return Value	Does not return a value	Returns a value
Declaration	Sub ... End Sub	Function ... End Function
Usage	Called independently	Used in expressions
Example	Call Display()	result = Add(5, 10)

Practical Example

Example 1: Without Parameters

```
Sub Welcome()
    MsgBox "Welcome to VB!"
End Sub
```

Calling:

```
Call Welcome()
```

Example 2: With Parameters

```
Sub ShowDetails(ByVal name As String, ByVal age As Integer)
    MsgBox "Name: " & name & " Age: " & age
End Sub
```

Calling:

```
Call ShowDetails("Divya", 20)
```

Example 3: Using Event

```
Private Sub btnSubmit_Click()
    Call DisplayMessage()
End Sub
Sub DisplayMessage()
    MsgBox "Form Submitted Successfully!"
End Sub
```

Importance of Sub Procedures in VB:

Promote structured programming.

Avoid code duplication.

Enable collaborative development (different programmers handle different subs).

Simplify testing and modification.

Provide clarity and modularity in projects.

Conclusion:

A Sub Procedure in Visual Basic is an essential programming construct that allows developers to organize code into smaller, reusable units.

It helps in writing clean, efficient, and maintainable programs.

Although it doesn't return a value like a function, it plays a crucial role in event handling, modular design, and user interaction.

22. Describe the Methods Used to Sort Records in Visual Basic.

Answer:

Introduction:

In Visual Basic (VB), sorting means arranging a set of records or data items in a specific order, such as ascending (A–Z or 0–9) or descending (Z–A or 9–0) order.

Sorting is an essential process in programming because it helps in:

Quick searching of records

Easy reporting and comparison

Better organization of data

Definition:

> Sorting is the process of arranging data in a meaningful order based on one or more fields.

For example, arranging student records in order of:

Name (alphabetical order)

Roll Number (numerical order)

Marks (highest to lowest)

Types of Sorting Methods:

There are several methods to sort records in VB. The most common ones are:

1. Bubble Sort

2. Selection Sort

3. Insertion Sort

4. Using Built-in VB Sorting Functions

5. Sorting Records from a Database

Explaination:

1. Bubble Sort Method:

Concept:

The Bubble Sort method compares adjacent elements and swaps them if they are in the wrong order. This process is repeated until all records are arranged properly.

Algorithm Steps:

1. Compare the first and second elements.

2. If the first is greater, swap them.

3. Continue comparing next pairs until the end of the list.

4. Repeat the process for all elements.

Example in VB:

```
Dim arr(4) As Integer
Dim i As Integer, j As Integer, temp As Integer

arr(0) = 45
arr(1) = 20
arr(2) = 30
arr(3) = 10
arr(4) = 50
```

```
For i = 0 To 3
    For j = 0 To 3 - i
        If arr(j) > arr(j + 1) Then
            temp = arr(j)
            arr(j) = arr(j + 1)
            arr(j + 1) = temp
        End If
    Next j
Next i

For i = 0 To 4
    Print arr(i)
Next i
```

Output:

10

20

30

45

50

✅ Advantage: Simple and easy to understand.

❌ Disadvantage: Slow for large data sets.

2. Selection Sort Method:

Concept:

In this method, the smallest element is selected and placed at the beginning. The process repeats for the remaining unsorted elements.

Algorithm Steps:

1. Find the smallest record in the unsorted list.
2. Swap it with the first element.
3. Repeat the process for the next positions.

Example in VB:

```
Dim arr(4) As Integer
Dim i As Integer, j As Integer, temp As Integer, min As Integer

arr(0) = 35
arr(1) = 10
arr(2) = 50
arr(3) = 25
arr(4) = 5

For i = 0 To 3
   min = i
   For j = i + 1 To 4
      If arr(j) < arr(min) Then
         min = j
      End If
   Next j
   temp = arr(i)
   arr(i) = arr(min)
   arr(min) = temp
Next i

For i = 0 To 4
   Print arr(i)
```

Next i

Output:

5

10

25

35

50

✅ Advantage: Simple and efficient for small data.

❌ Disadvantage: Requires more comparisons.

3. Insertion Sort Method:

Concept:

Insertion Sort inserts each element into its proper position within a sorted part of the array.

Algorithm Steps:

1. Start from the second element.
2. Compare it with previous elements.
3. Insert it in its correct position.
4. Repeat until all elements are sorted.

Example in VB:

```
Dim arr(4) As Integer
Dim i As Integer, j As Integer, key As Integer

arr(0) = 40
arr(1) = 20
```

```
arr(2) = 30
arr(3) = 10
arr(4) = 50

For i = 1 To 4
    key = arr(i)
    j = i - 1
    While j >= 0 And arr(j) > key
        arr(j + 1) = arr(j)
        j = j - 1
    Wend
    arr(j + 1) = key
Next i

For i = 0 To 4
    Print arr(i)
Next i
```

Output:

```
10
20
30
40
50
```

✅ Advantage: Efficient for small or partially sorted data.

❌ Disadvantage: Not suitable for very large data sets.

4. Using Built-in Sorting Functions in VB:

Visual Basic provides built-in sorting capabilities, especially when using collections or arrays.

Example 1: Using Array.Sort

```
Dim names() As String = {"Divya", "Arun", "Kumar", "Bala"}
Array.Sort(names)
For Each n In names
    Print n
Next
```

Output:

```
Arun
Bala
Divya
Kumar
```

Example 2: Using ListBox.Sorted = True

If you are displaying records in a ListBox, you can automatically sort them.

```
List1.Sorted = True
```

✅ Advantage: Easy and fast.

❌ Disadvantage: Limited customization.

5. Sorting Records in Database (Using SQL ORDER BY Clause):

When data is stored in a database (like MS Access), sorting can be done using SQL queries.

Example:

```
Dim rs As Recordset
Set rs = db.OpenRecordset("SELECT * FROM Student ORDER BY Name ASC")
```

Explanation:

ORDER BY Name ASC → sorts names in ascending order.

You can use DESC for descending order.

✅ Advantage: Very fast and efficient for large datasets.

❌ Disadvantage: Requires database connection.

Difference Between Ascending and Descending Sort:

Type	Order	Example (Numbers)	Example (Names)
Ascending	Small to Large	10, 20, 30, 40	A, B, C, D
Descending	Large to Small	40, 30, 20, 10	D, C, B, A.

Advantages of Sorting

1. Helps in quick searching of records.
2. Makes reports and lists more organized.
3. Allows easy comparison between records.
4. Improves data readability.
5. Useful in data analysis and presentations.

Applications of Sorting in VB:

Sorting student marks or names.

Arranging customer records alphabetically.

Sorting product prices from low to high.

Organizing employee data by ID or salary.

Conclusion:

Sorting is an essential operation in any data-handling application.

In Visual Basic, records can be sorted using programming techniques (like Bubble, Selection, or Insertion sort) or built-in methods (like Array.Sort or SQL ORDER BY).

Proper sorting makes data systematic, meaningful, and easier to process.

23. Describe the Working Principle of Common Dialog Box in Visual Basic.

Introduction:

In Visual Basic (VB), a Common Dialog Box is a predefined standard dialog box provided by Windows that allows users to perform common tasks easily, such as:

Opening files

Saving files

Choosing colors

Selecting fonts

Printing documents

It helps programmers avoid designing these dialogs from scratch — saving time and ensuring a consistent Windows look.

Definition:

A Common Dialog Box is a control in Visual Basic that provides standard Windows dialog boxes for file handling, printing, color selection, and font selection.

It is implemented using the CommonDialog Control available in the VB toolbox.

The control is invisible at runtime but can display standard dialog boxes when called in code.

Common Dialog Control :

Control Name: CommonDialog

File Name: COMDLG32.OCX

To use it:

1. Go to Project → Components.

2. Check Microsoft Common Dialog Control 6.0 (SP6).

3. Click OK.

4. A small icon (CommonDialog) appears in the toolbox.

5. Place it on the form (it will appear as CommonDialog1).

Working Principle:

The CommonDialog control acts as an interface between the VB program and the Windows operating system's built-in dialog boxes.

When a method such as ShowOpen or ShowSave is executed:

VB calls the Windows API (Application Programming Interface).

The corresponding standard dialog box is displayed.

User selects or enters information (like a filename, color, or font).

The control stores the result in its properties (like FileName, Color, FontName).

The program can then use these values for further processing.

Common Dialog Box Methods:

Method	Purpose / Function:
ShowOpen	Displays the Open File dialog box.
ShowSave	Displays the Save File dialog box.
ShowColor	Displays the Color Selection dialog box.
ShowFont	Displays the Font Selection dialog box.
ShowPrinter	Displays the Printer Setup dialog box.
ShowHelp	Displays the Help dialog box.

Important Properties of CommonDialog Control:

Property	Description	Example
FileName	Returns the selected file's name (path).	CommonDialog1.FileName
Filter	Specifies which file types to display.	`"Text Files (*.txt)
Color	Stores the color selected by the user.	Form1.BackColor = CommonDialog1.Color
FontName	Stores the selected font name.	"Arial", "Times New Roman"
FontSize	Stores the selected font size.	12, 14, 16
Flags	Controls the behavior of the dialog box.	cdlOFNFileMustExist, cdlOFNHideReadOnly
CancelError	Generates an error if the user cancels the dialog.	True or False.

1. Open File Dialog Box:

Used to allow the user to select a file from the system.

Example:

```
CommonDialog1.Filter = "Text Files (*.txt)|*.txt|All Files (*.*)|*.*"
CommonDialog1.ShowOpen
Text1.Text = CommonDialog1.FileName
```

Working:

Displays the Open File dialog box.

When the user selects a file, its path appears in the textbox.

2. Save File Dialog Box:

Used to save a file with a given name.

Example:

```
CommonDialog1.Filter = "Text Files (*.txt)|*.txt"
CommonDialog1.ShowSave
MsgBox "File Saved as: " & CommonDialog1.FileName
```

Working:

Displays the Save As dialog box.

The selected filename and path are stored in the FileName property.

3. Color Dialog Box:

Used to select a color.

Example:

```
CommonDialog1.ShowColor
Form1.BackColor = CommonDialog1.Color
```

Working:

Displays the Color Selection dialog box.

The selected color is applied to the form background.

4. Font Dialog Box:

Used to select font style, size, and color.

Example:

CommonDialog1.ShowFont

Text1.FontName = CommonDialog1.FontName

Text1.FontSize = CommonDialog1.FontSize

Text1.ForeColor = CommonDialog1.Color

Working:

Displays the Font Selection dialog box.

Applies the chosen font settings to the text box.

5. Printer Setup Dialog Box:

Used to select printer and print options.

Example:

CommonDialog1.ShowPrinter

Working:

Displays the Printer Setup dialog box.

User can choose printer, number of copies, and page range.

6. Help Dialog Box:

Used to display help files.

Example:

CommonDialog1.HelpFile = "helpfile.hlp"

CommonDialog1.ShowHelp.

Advantages of Using Common Dialog Box:

1. Provides standard Windows interface for users.
2. Reduces programming effort — no need to design dialogs manually.
3. Ensures consistency and user-friendliness.
4. Supports multiple tasks (file, color, font, print, help).
5. Compatible with Windows applications.
6. Saves development time and improves reliability.

Events Used in Common Dialog Box:

Event Description:

CancelError Occurs if the user cancels the dialog.

Error Triggered when an unexpected error occurs.

Example:

```
On Error GoTo CancelHandler
CommonDialog1.CancelError = True
CommonDialog1.ShowOpen
Exit Sub

CancelHandler:
MsgBox "User Cancelled the Dialog".
```

Applications:

Opening and saving text or image files.

Choosing background or font colors in applications.

Selecting printers before printing reports.

Providing user-friendly font customization.

Loading help files in applications.

Conclusion:

The Common Dialog Box is a powerful control in Visual Basic that allows developers to integrate standard Windows dialogs for common tasks like opening, saving, printing, color, and font selection.

It works by calling Windows APIs and returning user-selected information through its properties.

This enhances user interaction, ensures consistency, and saves development time.

24.Explain:

A. File List Box :

Answer:

Definition:

A File List Box is a control in Visual Basic used to display a list of files in a specified directory. It allows the user to select one file at a time.

Key Points:

1. Purpose: To let users see and select files from a folder.

2. Properties:

Path → Folder whose files are displayed.

Pattern → File type filter (e.g., *.txt, *.doc).

List → Array of filenames.

ListIndex → Index of the selected file.

3. Methods:

No special methods; selection is handled via ListIndex.

4. Example:

FileListBox1.Path = "C:\Documents"

FileListBox1.Pattern = "*.txt"

MsgBox "Selected File: " & FileListBox1.List(FileListBox1.ListIndex)

5. Advantages:

Easy file selection.

Filters files by type.

Simple integration into forms.

Diagram idea (optional in exam): Show a box listing files like file1.txt, file2.txt... with Path displayed above.

B. DLL Servers:

Answer:

Definition:

A DLL Server (Dynamic Link Library Server) is a file containing functions, procedures, or objects that can be used by one or more applications. In VB, DLLs allow code reusability and modular programming.

Key Points:

1. Purpose: To share code between multiple applications without rewriting.

2. Types of DLLs in VB:

Standard DLL: Provides functions/procedures.

ActiveX DLL: Provides objects and methods that can be used by other VB programs or COM applications.

3. Usage in VB:

Declare external functions using Declare statement.

Reference ActiveX DLL in project and create objects.

4. Example of Standard DLL Declaration:

```
Declare Function MessageBox Lib "user32" Alias "MessageBoxA" _
   (ByVal hwnd As Long, ByVal lpText As String, _
   ByVal lpCaption As String, ByVal wType As Long) As Long
```

5. Advantages:

Saves memory.

Promotes code reuse.

Easy maintenance and modular programming.

Diagram idea: Show VB project calling a DLL server with arrows pointing to functions or objects inside the DLL.

DEC-2023

PART-C

20. Explain about Image controls and text boxes with examples.

1. Introduction

In Visual Programming (such as Visual Basic), controls are the basic building blocks of a graphical user interface (GUI).
Among the most commonly used controls are Image Controls and Text Boxes, which help in displaying graphics and accepting user input respectively.

2. Image Control

Definition:

The Image Control is used to display pictures, icons, or graphics such as .bmp, .jpg, .gif, .png files on a Visual Basic form.

It is mainly used for decorative purposes or to display logos, photos, or background images in applications.

Properties of Image Control:

Property	Description
Picture	Used to set or change the image file to be displayed.
Stretch	If set to True, the image resizes to fit the control's size.
BorderStyle	Determines whether the image has a border (FixedSingle or None).
Visible	Shows or hides the image at runtime.
Height / Width	Used to set the size of the image area.

Methods:

Method	Description
LoadPicture	Used to load an image file at runtime.
Cls	Clears the image displayed in the control.

Example of Image Control:

```
Private Sub Form_Load()
   Image1.Picture = LoadPicture("C:\Users\Public\Pictures\Sample.jpg")
   Image1.Stretch = True
End Sub
```

Explanation:

When the form loads, an image named *Sample.jpg* is displayed inside Image1.

The Stretch property makes the image fit perfectly within the control size.

Uses of Image Control:

Displaying company logos.

Showing product pictures in an inventory system.

Displaying backgrounds or icons.

Creating photo galleries in an application.

3. Text Box Control

Definition:

A Text Box Control is used to accept text input from the user or display text output to the user. It allows users to enter, edit, or view information such as names, numbers, or messages.

Properties of Text Box:

Property	Description
Text	Contains the text entered or displayed in the box.
MaxLength	Sets the maximum number of characters allowed.
MultiLine	Allows multiple lines of text if set to True.
PasswordChar	Masks the characters (useful for passwords).
Enabled / Locked	Determines if the user can modify the text.
Alignment	Aligns text to left, center, or right.

Methods:

Method	Description
SetFocus	Moves the cursor to the text box.
SelStart / SelLength	Used to select a portion of text.
Text = ""	Clears the content of the text box.

Example of Text Box Control:

```
Private Sub Command1_Click()
   Dim name As String
   name = Text1.Text
   MsgBox "Welcome, " & name & "!"
End Sub
```

Explanation:

The user types their name into Text1.

When the button is clicked, a message box greets the user using their input.

Uses of Text Box:

Accepting user inputs (like name, address, age).

Displaying results or messages.

Collecting data for forms.

Password fields using the PasswordChar property.

4. Differences Between Image Control and Text Box

Feature	Image Control	Text Box
Purpose	Displays pictures or graphics.	Accepts or displays text input.
Input Type	Image files (.bmp, .jpg, etc.)	Text or numeric values.
User Interaction	Not editable by user.	Editable by user.
Common Use	Display logos, photos, icons.	Input forms, data entry, messages.

21. Explain about various tools in VB with examples.

1. Introduction

Visual Basic (VB) is a powerful visual programming language that allows developers to create Graphical User Interface (GUI) applications easily.
VB provides a Toolbox that contains various tools (controls) used to design forms and create interactive applications.

These tools help the programmer add, modify, and control objects like buttons, text boxes, labels, images, etc., on the form.

2. Various Tools in VB

The most commonly used tools in Visual Basic are listed below:

(1) Label Control

Purpose:
Used to display text or messages on the form that the user cannot modify.

Important Properties:

Caption – Sets the text to be displayed.

Alignment – Sets text alignment (left, center, right).

Font – Changes the style and size of the text.

Example:

Label1.Caption = "Welcome to Visual Basic"

Displays the message “Welcome to Visual Basic” on the form.

(2) Text Box Control

Purpose:
Used to accept input or display output in the form of text.

Important Properties:

Text – Holds the text value.

MaxLength – Limits character input.

PasswordChar – Hides typed characters for passwords.

Example:

MsgBox "Your Name is: " & Text1.Text

Displays the text entered by the user.

(3) Command Button

Purpose:
Used to perform an action or execute code when clicked.

Important Properties:

Caption – Displays text on the button.

Enabled – Enables or disables the button.

Example:

```
Private Sub Command1_Click()
    MsgBox "Button Clicked!"
End Sub
```

(4) Picture Box

Purpose:
Used to display graphics, images, or drawings on the form.
Unlike Image Control, it can be used for drawing shapes at runtime.

Example:

Picture1.Picture = LoadPicture("C:\Logo.jpg")

(5) Image Control

Purpose:
Displays pictures like .bmp, .jpg, .png, etc., but uses less memory than PictureBox.

Example:

```
Image1.Picture = LoadPicture("C:\Images\Photo.jpg")
```

(6) Check Box

Purpose:
Used to select or deselect one or more independent options.

Example:

```
If Check1.Value = 1 Then
   MsgBox "You have selected the option!"
End If
```

(7) Option Button (Radio Button)

Purpose:
Used when the user has to select only one option from multiple choices.

Example:

```
If Option1.Value = True Then
   MsgBox "Male Selected"
ElseIf Option2.Value = True Then
   MsgBox "Female Selected"
End If
```

(8) Combo Box

Purpose:
Allows the user to choose an item from a drop-down list or enter a new value.

Example:

```
Combo1.AddItem "Apple"
Combo1.AddItem "Banana"
Combo1.AddItem "Mango"
```

(9) List Box

Purpose:
Displays a list of items from which the user can select one or more.

Example:

```
List1.AddItem "C++"
List1.AddItem "Java"
List1.AddItem "Python"
```

(10) Timer Control

Purpose:
Used to perform actions at regular time intervals automatically.

Properties:

Interval – Time between two events in milliseconds.

Example:

```
Private Sub Timer1_Timer()
    Label1.Caption = Time$
End Sub
```

Displays the current time every second.

(11) Frame Control

Purpose:
Used to group related controls together, such as radio buttons or checkboxes.

Example:
A Frame named *Frame1* can contain *Option1* and *Option2* for Gender selection.

(12) Shape Control

Purpose:
Used to draw geometric shapes like circles, rectangles, or lines on the form.

Example:

```
Shape1.Shape = 1    'Rectangle
```

(13) Line Control

Purpose:
Used to draw straight lines on the form or between controls.

Example:

```
Line1.X1 = 100
Line1.X2 = 500
```

(14) Data Control

Purpose:
Used to connect VB forms to databases like MS Access for data handling.

Example:

```
Data1.DatabaseName = "C:\Student.mdb"
Data1.RecordSource = "StudentTable"
```

3. Example: Simple Login Form using Tools

Tools Used: Labels, Text Boxes, Command Buttons

```
Private Sub Command1_Click()
   If Text1.Text = "admin" And Text2.Text = "1234" Then
      MsgBox "Login Successful"
   Else
      MsgBox "Invalid Username or Password"
   End If
End Sub
```

Explanation:

Text1 – Username box

Text2 – Password box (with PasswordChar property)

Command1 – Login button

==========

22. Explain about functions and procedures in VB with examples.

1. Introduction

In Visual Basic (VB), the main building blocks of a program are procedures and functions.
They are used to organize and reuse code efficiently.

Both are groups of statements written to perform specific tasks.

The major difference is that functions return a value, while procedures do not.

2. What is a Procedure?

Definition:

A procedure in VB is a block of code written to perform a specific task but does not return a value to the calling program.

Procedures make the program modular, readable, and easy to debug.

Types of Procedures in VB:

Sub Procedure (Subroutine)

Event Procedure

(a) Sub Procedure

A Sub Procedure is a block of code that performs a task but does not return any value.

Syntax:

```
Sub ProcedureName(Arguments)
    'Statements
End Sub
```

Example:

```
Sub DisplayMessage()
    MsgBox "Welcome to Visual Basic!"
End Sub
```

Explanation:

The above code defines a sub-procedure named DisplayMessage.

When called, it shows a message box.

Calling the Procedure:

```
Call DisplayMessage
```

(b) Event Procedure

Event procedures are automatically executed when an event occurs, such as clicking a button or loading a form.

Example:

```
Private Sub Command1_Click()
    MsgBox "Button Clicked!"
End Sub
```

Explanation:
This procedure executes automatically when the user clicks the Command1 button.

3. What is a Function?

Definition:

A Function in VB is a block of code that performs a specific task and returns a value to the calling code.

Functions are useful for calculations, data processing, and returning results.

Syntax:

```
Function FunctionName(Arguments) As DataType
    'Statements
    FunctionName = value
End Function
```

Example 1: Function to Add Two Numbers

```
Function AddNumbers(a As Integer, b As Integer) As Integer
    AddNumbers = a + b
End Function
```

Calling the Function:

```
Private Sub Command1_Click()
    Dim result As Integer
    result = AddNumbers(10, 20)
    MsgBox "The sum is: " & result
End Sub
```

Explanation:

The function AddNumbers receives two numbers as input.

It returns their sum to the calling procedure.

Example 2: Function to Calculate Square

```
Function Square(num As Integer) As Integer
    Square = num * num
End Function
```

Calling the Function:

```
MsgBox "Square of 5 is " & Square(5)
```

4.Differences Between Function and Procedure

Feature	Function	Procedure (Sub)

Purpose	Performs task and returns a value.	Performs task but does not return a value.
Return Type	Must specify data type (e.g., Integer, String).	No return type.
Use in Expressions	Can be used directly in expressions.	Cannot be used in expressions.
Syntax	Uses Function and End Function.	Uses Sub and End Sub.
		DisplayMessage()
Example	x = Square(5)	

5. Advantages of Using Functions and Procedures

Code Reusability: Write once, use multiple times.

Modularity: Breaks large programs into manageable sections.

Easy Debugging: Errors can be found and corrected easily.

Improved Readability: Makes code structured and organized.

Efficient Maintenance: Changing one function automatically affects all calls to it.

6. Example Program Using Function and Procedure

```
Private Sub Command1_Click()
    Dim n1 As Integer, n2 As Integer, sum As Integer
    n1 = Val(Text1.Text)
    n2 = Val(Text2.Text)

    sum = AddNumbers(n1, n2)
    DisplayResult sum
End Sub

Function AddNumbers(a As Integer, b As Integer) As Integer
    AddNumbers = a + b
End Function

Sub DisplayResult(result As Integer)
    MsgBox "The sum of the numbers is: " & result
End Sub
```

Explanation:

AddNumbers → Function that returns the sum.

DisplayResult → Procedure that shows the result using a message box.

Command1_Click → Event procedure that calls both.

23. Discuss about Grid controls and Multiple forms with examples

1. Introduction

In Visual Basic (VB), user interfaces are designed using various controls.
Two important features that enhance VB applications are:

Grid Controls – used to display and manage data in a tabular (row-column) form.

Multiple Forms – used to create applications with more than one window or screen.

Both are essential for developing interactive and data-driven applications.

2. Grid Controls

Definition:

A Grid Control is a powerful tool used to display, edit, and manage tabular data (rows and columns) in a VB form.

It allows the programmer to display database records or large amounts of structured data in a user-friendly way.

Types of Grid Controls in VB:

DataGrid Control

MSFlexGrid Control

DBGrid Control

(a) DataGrid Control

Purpose:
Displays and allows editing of records from a database table.

Important Properties:

Property	Description
DataSource	Links the grid to a data control (like ADODC or Data Control).
AllowUpdate	Allows editing of data if set to True.
Columns	Used to set the number and headings of columns.

Example: Displaying Student Records

Private Sub Form_Load()

```
    Adodc1.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Student.mdb"
    Adodc1.RecordSource = "Select * from Student"
    Set DataGrid1.DataSource = Adodc1
End Sub
```

Explanation:

The Adodc1 control connects to the Access database *Student.mdb*.

The DataGrid1 displays all records from the “Student” table.

(b) MSFlexGrid Control

Purpose:
Used to display data in a read-only grid format.
Unlike DataGrid, the user cannot directly edit the data.

Properties:

Property	Description
Rows	Sets the number of rows.
Cols	Sets the number of columns.
TextMatrix(row, col)	Sets or gets the value of a specific cell.

Example: Displaying Numbers in Grid

```
Private Sub Form_Load()
    MSFlexGrid1.Rows = 5
    MSFlexGrid1.Cols = 3
    For i = 1 To 4
        For j = 1 To 2
            MSFlexGrid1.TextMatrix(i, j) = "Cell(" & i & "," & j & ")"
        Next j
    Next i
End Sub
```

Explanation:
This code displays a 4×2 table where each cell is filled with its position text (e.g., *Cell(1,1)*).

Advantages of Grid Controls:

Easy to display large sets of data in a table.

Simplifies database connectivity and record management.

Allows sorting, editing, and updating data.

Gives a professional look to business applications.

3. Multiple Forms

Definition:

In Visual Basic, an application can have more than one form.

Each form acts as a separate window or screen in the program.

For example:

Form1 → Login Screen

Form2 → Main Menu

Form3 → Report Page

Creating Multiple Forms:

Choose Project → Add Form from the menu.

A new blank form (e.g., Form2) will be added to the project.

You can design and code it like the main form.

Displaying Multiple Forms:

(a) Showing a New Form

Form2.Show

(b) Hiding a Form

Form1.Hide

(c) Unloading a Form

Unload Form2

Example: Login Form and Main Form

Form1 – Login Form

```
Private Sub Command1_Click()
    If Text1.Text = "admin" And Text2.Text = "1234" Then
        Form2.Show
        Me.Hide
```

```
    Else
        MsgBox "Invalid Username or Password"
    End If
End Sub
```

Form2 – Main Form

```
Private Sub Command2_Click()
    MsgBox "Welcome to Main Application!"
End Sub
```

Explanation:

The user enters credentials in Form1.

If correct, Form1 hides and Form2 (main screen) appears.

This demonstrates navigation between multiple forms.

Advantages of Using Multiple Forms:

Allows separation of application functions (Login, Menu, Report).

Improves user interface design and organization.

Makes large applications modular and easy to maintain.

Enables reuse of forms in other projects.

4.Differences Between Grid Control and Multiple Forms

Feature	Grid Control	Multiple Forms
Purpose	Display data in table format.	Manage different screens/windows.
Main Use	Database handling.	Navigation and interface design.
User Interaction	Data viewing and editing.	Screen switching.
Example	DataGrid displaying student records.	Login form leading to main form.

24. Explain the following

(a) File System controls

(b) Built in function in VB.

1. Introduction

In Visual Basic (VB), File System Controls are used to work with files, folders, and drives on a computer.
These controls allow the user to browse directories, select files, and perform file-related operations like open, save, or delete.

They help in creating file-handling applications, such as file explorers or data management systems.

2. Types of File System Controls in VB

Control	Description
DriveListBox	Displays the list of all available drives (C:, D:, E:, etc.).
DirListBox	Displays the list of directories (folders) in a selected drive.
FileListBox	Displays the list of files in a selected folder.

3. Description of Each Control

(i) DriveListBox Control

Used to display a list of all drives (e.g., hard disk, CD-ROM, USB).

The user can select a drive, which automatically updates the directory list.

Example:

```
Private Sub Drive1_Change()
   Dir1.Path = Drive1.Drive
End Sub
```

Explanation:
When the user changes the drive, the folder list (Dir1) automatically updates.

(ii) DirListBox Control

Displays the folders available in a selected drive.

The user can navigate through different directories.

Example:

```
Private Sub Dir1_Change()
   File1.Path = Dir1.Path
End Sub
```

Explanation:
When the user selects a folder, the file list updates to show files from that folder.

(iii) FileListBox Control

Displays the list of files in a selected directory.

Can be used to select, open, or process files.

Example:

```
Private Sub File1_Click()
    MsgBox "You selected: " & File1.FileName
End Sub
```

4. Combined Example of File System Controls

```
Private Sub Drive1_Change()
    Dir1.Path = Drive1.Drive
End Sub

Private Sub Dir1_Change()
    File1.Path = Dir1.Path
End Sub

Private Sub File1_Click()
    MsgBox "You selected file: " & File1.FileName
End Sub
```

Explanation:

DriveListBox (Drive1) shows all drives.

DirListBox (Dir1) displays folders from the selected drive.

FileListBox (File1) shows files from the selected folder.

Together, they create a mini file explorer interface.

5. Uses of File System Controls

Browsing drives, folders, and files.

Creating file open/save dialog systems.

Managing and processing user-selected files.

File organization and backup tools.

(b) Built-in Functions in VB

1. Introduction

Built-in functions are predefined functions provided by Visual Basic to perform common tasks quickly and easily.
They help reduce the amount of code and improve program efficiency.

Functions can handle mathematical, string, date/time, and conversion operations.

2. Categories of Built-in Functions

Category	Purpose
Mathematical Functions	Perform arithmetic or numeric calculations.
String Functions	Manipulate and handle text data.
Date and Time Functions	Work with system date and time.
Conversion Functions	Convert data types.
Miscellaneous Functions	Perform general-purpose tasks.

3. Common Built-in Functions with Examples

(i) Mathematical Functions

Function	Description	Example
Abs(x)	Returns the absolute value.	Abs(-5) → 5
Sqr(x)	Returns the square root.	Sqr(16) → 4
Int(x)	Returns the integer portion of a number.	Int(7.9) → 7
Rnd	Returns a random number between 0 and 1.	Rnd → 0.4523

(ii) String Functions

Function	Description	Example
Len(str)	Returns the length of a string.	Len("VB") → 2
LCase(str)	Converts to lowercase.	LCase("HELLO") → "hello"
UCase(str)	Converts to uppercase.	UCase("vb") → "VB"
Left(str, n)	Returns leftmost n characters.	Left("Visual", 3) → "Vis"
Right(str, n)	Returns rightmost n characters.	Right("Basic", 2) → "ic"
Mid(str, start, length)	Extracts part of a string.	Mid("Visual", 2, 3) → "isu"

(iii) Date and Time Functions

Function	Description	Example
Date	Returns current date.	Date → "12-11-2025"
Time	Returns current time.	Time → "2:45:10 PM"

Now	Returns current date and time.	Now → "12-11-2025 2:45 PM"
Day(Date)	Returns day of the date.	Day(Date) → 12

(iv) Conversion Functions

Function	Description	Example
Val(str)	Converts string to number.	Val("123") → 123
Str(number)	Converts number to string.	Str(45) → "45"
CInt(value)	Converts to integer.	CInt(9.6) → 10
CSng(value)	Converts to single precision.	CSng(10) → 10.0

Example Program Using Built-in Functions

```
Private Sub Command1_Click()
   Dim name As String, length As Integer
   name = Text1.Text
   length = Len(name)
   MsgBox "Your name in uppercase: " & UCase(name)
   MsgBox "Length of name: " & length
End Sub
```

Explanation:

The user enters a name in a textbox.

The program displays the uppercase version and length of the name using built-in functions.

5. Advantages of Built-in Functions

Simplifies complex operations.

Saves programming time.

Reduces code size and errors.

Improves performance and reliability.

Makes code more readable and maintainable.

May 2021

PART-C

20. Explain about Image Controls and Text Boxes.

Image Controls

The Image Control in Visual Basic is used to display images or pictures such as .bmp, .jpg, .gif, .ico, etc. It allows developers to create a more visually appealing interface.

Properties of Image Control

Property	Description
Picture	Specifies the image to display.
Stretch	If set to True, stretches the image to fit the control.
BorderStyle	Sets the border appearance (None or Fixed Single).
Visible	Controls whether the image is visible.
Enabled	Enables or disables the image control.

Methods

LoadPicture(filename): Used to load an image dynamically during runtime.

Image1.Picture = LoadPicture("C:\photo.jpg")

Events

Click: Executes when the user clicks the image.
Example:

```
Private Sub Image1_Click()
    MsgBox "You clicked the image!"
End Sub
```

Uses of Image Control

Displaying logos or photos.

Setting background images for forms.

Creating animation effects by changing images dynamically.

Text Box Control

The Text Box Control allows users to input or display text during program execution. It is one of the most commonly used controls in VB.

Properties

Property	Description
Text	Contains the text displayed.
MaxLength	Specifies the maximum number of characters.
MultiLine	Allows multiple lines of text when set to True.
PasswordChar	Masks the characters for password entry.
Alignment	Aligns text (Left, Right, Center).

Events

Change: Triggered when text changes.

GotFocus / LostFocus: Triggered when control gains or loses focus.

Example:

```
Private Sub Text1_Change()
   Label1.Caption = Text1.Text
End Sub
```

Uses of Text Boxes

Accepting user input like name, email, or age.

Displaying computed results or data from files.

Creating forms and input fields.

Comparison:

Feature	Image Control	Text Box
Purpose	Display pictures	Accept/display text
Input Type	Graphical	Textual
User Interaction	Click or view	Type or edit
Example	Image1.Picture	Text1.Text

Conclusion

Both Image Controls and Text Boxes are essential for designing graphical and interactive VB interfaces. Image controls enhance visual appeal, while text boxes manage data input and output, making them fundamental in user interface design.

---

21. Explain the following:

(a) File System Controls

File System Controls allow users to navigate drives, folders, and files in a VB application. They are used for file management and selection tasks.

Common File System Controls

DriveListBox: Displays available drives.

Drive1.Drive = "C:\"

DirListBox: Displays folders in a selected drive.

FileListBox: Displays files in a selected folder.

Example:

```
Private Sub Drive1_Change()
```

```
    Dir1.Path = Drive1.Drive
End Sub


Private Sub Dir1_Change()
    File1.Path = Dir1.Path
End Sub
```

When the user changes the drive, the directory and file list automatically update.

Uses:

File browsing in applications.

File open/save operations.

Managing directories dynamically.

(b) DLL Server

DLL (Dynamic Link Library) is a file containing reusable code and procedures that can be used by multiple applications.

Features of DLL:

Contains functions, classes, or resources.

Does not execute independently; it is called by other programs.

Promotes code reuse and reduces redundancy.

Used in VB through the Declare or References options.

Example of Using DLL:

```
Declare Function GetTickCount Lib "kernel32" () As Long
MsgBox GetTickCount()
```

Here, GetTickCount is a function from the Windows DLL kernel32.dll.

Advantages:

Saves memory and storage space.

Easier maintenance and version control.

Enables modular programming.

Allows integration with Windows API functions.

Conclusion:

File system controls simplify file management for users, while DLL servers promote code reuse and extend the capabilities of Visual Basic programs through shared libraries.

---

22. Explain about Functions and Procedures in VB with Example.

1. Introduction

In Visual Basic, functions and procedures are reusable blocks of code that perform specific tasks. They improve program modularity and reduce repetition.

2. Types

Sub Procedures

Perform tasks but do not return values.

Declared using Sub and End Sub.

Example:

```
Sub ShowMessage()
    MsgBox "Hello, Welcome to VB!"
End Sub
```

Calling:

```
Call ShowMessage()
```

Function Procedures

Perform tasks and return a value.

Declared using Function and End Function.

Example:

```
Function Add(x As Integer, y As Integer) As Integer
    Add = x + y
End Function
```

Calling:

```
result = Add(10, 20)
Print result
```

Event Procedures

Triggered automatically when an event occurs (e.g., clicking a button).
Example:

```
Private Sub Command1_Click()
    MsgBox "Button was clicked!"
End Sub
```

3. Advantages of Using Functions and Procedures

Reusability: Can be called from different parts of the program.

Readability: Makes code easier to understand.

Maintenance: Easier to fix errors and update logic.

Modularity: Divides complex programs into smaller tasks.

4. Difference Between Function and Procedure

Feature	Function	Procedure (Sub)
Returns Value	Yes	No
Usage	Used in expressions	Called independently
Declaration	Function ... End Function	Sub ... End Sub
Example	Function Add(a, b)	Sub Display()

Conclusion

Functions and procedures are vital for structured programming in VB. They make programs modular, reusable, and efficient, leading to better design and maintainability.

---

23. Discuss about OLE Drag and Drop.

Definition

OLE (Object Linking and Embedding) Drag and Drop in Visual Basic allows users to drag data or objects (like text, images, or files) from one control and drop them into another control.

1. Concept

OLE Drag Source: The control from which data is dragged.

OLE Drop Target: The control where the data is dropped.

Example: Dragging text from a TextBox and dropping it in another TextBox.

2. Enabling OLE Drag and Drop

Each control has two important properties:

OLEDragMode – Defines if the control can start a drag operation.

Values: 0 - Manual, 1 - Automatic

OLEDropMode – Defines if the control can receive dropped data.

Values: 0 - None, 1 - Manual, 2 - Automatic

3. Example

```
Private Sub Text1_OLEStartDrag(Data As DataObject, AllowedEffects As Long)
    Data.SetData Text1.Text
End Sub


Private Sub Text2_OLEDragDrop(Data As DataObject, Effect As Long, Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
    Text2.Text = Data.GetData(vbCFText)
End Sub
```

Here, text from Text1 can be dragged and dropped into Text2.

4. Advantages

Enhances user interaction and ease of use.

Provides a modern interface for users to move data.

Supports OLE objects (images, text, files, etc.).

Reduces the need for copy/paste actions.

5. Applications

Text editors (dragging text).

File managers (moving files).

Image editing applications.

Email or data management systems.

Conclusion

OLE Drag and Drop adds interactive and intuitive functionality to Visual Basic applications, allowing users to manipulate data objects naturally between controls or even different programs.

---

24. Write a short note on:

(a) File System Controls

File System Controls are built-in controls in VB used for navigating and managing drives, directories, and files without writing complex code.

Types:

DriveListBox: Displays available drives.

DirListBox: Displays folders of the selected drive.

FileListBox: Displays files in a selected folder.

Example:

```
Private Sub Drive1_Change()
    Dir1.Path = Drive1.Drive
End Sub


Private Sub Dir1_Change()
    File1.Path = Dir1.Path
End Sub
```

Uses:

To open or save files dynamically.

To build file navigation features in VB applications.

(b) Built-in Functions in VB

Built-in functions are predefined routines in VB that perform common tasks without additional code.

Types of Built-in Functions:

Mathematical Functions:

Abs(x), Sqr(x), Rnd(), Int(x)
Example: x = Sqr(25) → Returns 5

String Functions:

Len(), Left(), Right(), Mid(), UCase(), LCase()
Example: Left("Visual Basic", 6) → Returns “Visual”

Date and Time Functions:

Date(), Time(), Now(), Month(), Year()

Conversion Functions:

CInt(), CSng(), CStr(), Val()

Financial Functions:

FV(), PV(), Rate() used for business calculations.

Importance:

Simplifies coding and saves time.

Reduces the need for user-defined logic.

Enhances performance and reliability.

Conclusion

File System Controls manage files and directories efficiently, while Built-in Functions make programming faster and easier by providing ready-to-use operations for calculations, text manipulation, and data handling.

May-2022

PART-C

20.What is a command button.? Explain the properties and methods associated with a command button.?

A command button, in visual programming and graphical user interfaces (GUIs), is a clickable control element that initiates a specific action or sequence of actions when a user interacts with it, typically by clicking. It is a fundamental component for user input and program control.

Properties Associated with a Command Button:

Properties are attributes that define the appearance, behavior, and state of the command button. Common properties include:

- Name (or ID): A unique identifier for the button, used to reference it in code.
- Caption (or Text): The text displayed on the button, indicating its purpose.
- Enabled: A Boolean property determining if the button can be clicked (True) or is disabled/grayed out (False).
- Visible: A Boolean property determining if the button is displayed (True) or hidden (False).
- Default: A Boolean property indicating if the button is the default action when the Enter key is pressed (only one button can be default per form).
- Cancel: A Boolean property indicating if the button is activated when the Escape key is pressed (only one button can have this set to True per form).
- Appearance (or Style): Defines the visual style, such as 3D or flat.
- Font: Specifies the font type, style, and size of the button's caption.
- BackColor: Sets the background color of the button.
- ForeColor: Sets the color of the text displayed on the button.
- Picture: Allows displaying an image on the button, often used with a graphical style.
- ToolTipText (or ControlTipText): Text that appears when the user hovers the mouse pointer over the button.

Methods Associated with a Command Button:

Methods are actions or functions that can be performed on or by the command button, typically through code. The most significant method is often an event handler:

- Click Event (e.g., OnClick, CommandButton_Click): This is the primary method, or more accurately, an event handler, associated with a command button. When the user clicks the button, this event is triggered, and the code written within this event procedure is executed. This is where the specific action or logic associated with the button is defined.

While there might be other methods depending on the specific visual programming environment (e.g., SetFocus to give the button focus), the Click event is the most crucial for defining the button's functionality.

21.Explain the various determinate and indeterminate loops with examples.?

- A loop is a programming construct that allows a block of code to be executed repeatedly until a specific condition is met.
- Loops are classified as Determinate (fixed number of iterations) and Indeterminate (number of iterations not known in advance).

(A) Determinate Loops

These loops run for a known number of times.

1. For...Next Loop

Used when the number of repetitions is known.

Syntax:

```
For counter = start To end [Step increment]
    'Statements
Next
```

Example:

```
For i = 1 To 5
    Print i
Next i
```

Output:

1

2

3

4

5

2. For Each...Next Loop

Used to iterate through a collection or array.

Example:

```
Dim arr() As Integer = {10, 20, 30}
Dim num As Integer
For Each num In arr
    Print num
Next
```

(B) Indeterminate Loops

Used when the number of repetitions is not known in advance.

1. Do While...Loop

Executes statements as long as the condition is True.

Example:

```
Dim i As Integer
i = 1
Do While i <= 5
    Print i
```

```
    i = i + 1
Loop
```

2. Do Until...Loop

Executes until the condition becomes True.

Example:

```
Dim i As Integer
i = 1
Do Until i > 5
    Print i
    i = i + 1
Loop
```

3. While...Wend Loop

Similar to Do While Loop (older syntax).

Example:

```
Dim i As Integer
i = 1
While i <= 5
    Print i
    i = i + 1
Wend
```

22.Write a VB procedure to arrange the given set of numbers into ascending order.?

This program accepts a list of numbers and arranges them in ascending order using a Bubble Sort technique.

Program:

```
Private Sub cmdSort_Click()
    Dim arr(5) As Integer
    Dim i As Integer, j As Integer, temp As Integer

    'Input numbers
    arr(1) = 45
    arr(2) = 12
    arr(3) = 67
    arr(4) = 23
```

```
    arr(5) = 5

    'Sorting Logic (Bubble Sort)
    For i = 1 To 4
        For j = i + 1 To 5
            If arr(i) > arr(j) Then
                temp = arr(i)
                arr(i) = arr(j)
                arr(j) = temp
            End If
        Next j
    Next i
    'Display sorted numbers
    Print "Ascending Order:"
    For i = 1 To 5
        Print arr(i)
    Next i
End Sub
```

Output:

Ascending Order:

5

12

23

45

67

Explanation:

The program compares each element with the next and swaps them if necessary until all numbers are sorted in ascending order.

23.Explain the various common dialog controls with examples.?

Common Dialog Controls in VB are used to perform common tasks such as opening files, saving files, choosing colors, and printing documents.

These controls belong to the Microsoft Common Dialog Control Library.

Common Dialog Types:

Control Description

Open Dialog Used to open existing files.

Save Dialog Used to save files with a given name.

Color Dialog Allows users to select colors.

Font Dialog Allows users to select font style, size, and type.

Print Dialog Provides a standard print interface.

Example 1: Open Dialog

CommonDialog1.ShowOpen

Text1.Text = CommonDialog1.FileName

Explanation:

Displays the Open File dialog and shows the selected file path in a text box.

Example 2: Save Dialog

CommonDialog1.ShowSave

MsgBox "File will be saved as: " & CommonDialog1.FileName

Example 3: Color Dialog

CommonDialog1.ShowColor

Form1.BackColor = CommonDialog1.Color

Example 4: Font Dialog

CommonDialog1.ShowFont

Text1.FontName = CommonDialog1.FontName

Text1.FontSize = CommonDialog1.FontSize

Example 5: Print Dialog

CommonDialog1.ShowPrinter

Printer.Print "Hello, this is a test print!"

Printer.EndDoc

24.What are the various file system controls available in VB.? Explain.

Visual Basic provides several File System Controls that help users interact with files and folders visually.

Visual Basic (VB) provides a set of specialized controls designed to interact with the file system, allowing users to browse and select files and directories. These are often referred to as File System Controls. The three primary file system controls in VB are:

DriveListBox Control:

- Purpose: This control is a specialized drop-down list that displays all the valid drives available on the user's system (e.g., C:, D:, network drives).

- Explanation: It allows users to select a specific drive, and its Drive property can be used to get or set the currently selected drive. When the selected drive changes, it typically triggers an event that can be used to update other file system controls.

DirListBox Control:

- Purpose: This control displays a hierarchical list of directories (folders) within the currently selected drive or path.
- Explanation: It allows users to navigate through the directory structure. Its Path property determines which directory's subdirectories are displayed. When the selected directory changes, it can trigger an event to update the FileListBox to show the files in that directory.

FileListBox Control:

- Purpose: This control displays a list of files within the currently selected directory.
- Explanation: It allows users to select individual files. Its Path property is typically synchronized with the DirListBox to display files from the chosen directory. The FileName and Pattern properties can be used to filter the displayed files.

How they work together:

These controls are often used in conjunction to create a complete file browsing interface. For example:

- A DriveListBox is used to select a drive.
- The Change event of the DriveListBox updates the Path property of the DirListBox to display directories on the selected drive.
- The Change event of the DirListBox then updates the Path property of the FileListBox to display files in the selected directory.
- Users can then select a file from the FileListBox.

Main File System Controls:

Control Description

DriveListBox Displays available drives (C:, D:, etc.).

DirListBox Displays directories or folders in a selected drive.

FileListBox Displays files in a selected folder.

1. DriveListBox Control

Used to select a disk drive.

Syntax Example:

```
Drive1_Change
    Dir1.Path = Drive1.Drive
End Sub
```

2. DirListBox Control

Displays folders in the selected drive.

Example:

```
Private Sub Dir1_Change()
    File1.Path = Dir1.Path
End Sub
```

3. FileListBox Control

Displays files from the selected directory.

Example:

```
Private Sub File1_Click()
    Text1.Text = File1.FileName
End Sub
```

Working Together Example:

```
Private Sub Drive1_Change()
    Dir1.Path = Drive1.Drive
End Sub

Private Sub Dir1_Change()
    File1.Path = Dir1.Path
End Sub
```

Explanation:

- The user selects a drive in DriveListBox.
- The corresponding directories appear in DirListBox.
- Selecting a folder shows its files in FileListBox.

May-2024

10-marks

20. Explain about various menus in VB with examples.?

Menus in Visual Basic (VB) provide an easy way for users to navigate and interact with an application. They appear on the menu bar, offering various options or commands.

Types of Menus in VB:

Main Menu

The primary menu displayed on the top of a form.

Created using the Menu Editor.

Example:

File → New, Open, Save, Exit

Edit → Cut, Copy, Paste

Submenus

Sub-level menus under a main menu.

Created by indenting items in the Menu Editor.

Example:

File

  └── New

  └── Open

  └── Exit

Popup Menus (Context Menus)

Menus that appear when you right-click a form or control.

Created using the PopupMenu method.

Example:

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
   If Button = vbRightButton Then
      PopupMenu mnuEdit
   End If
End Sub
```

Toolbar Menus

Contain icons or shortcuts for commonly used commands (like Save, Open, Print).

Created using the Toolbar Control.

MDI Form Menus

Used in Multiple Document Interface applications.

The parent form has its own menu, and child forms can merge their menus.

Example:

```
Private Sub mnuExit_Click()
   End
End Sub


Private Sub mnuOpen_Click()
```

```
    MsgBox "Open File Menu Selected"
End Sub
```

Advantages:

Makes applications user-friendly.

Organizes commands logically.

Saves screen space by grouping options.

21. Explain about various Indeterminate Loops with examples.?

Loops in VB repeat a block of code multiple times.
Indeterminate loops are loops that continue until a condition becomes false — the number of repetitions is not known beforehand.

Types of Indeterminate Loops:

While...Wend Loop

Executes a block of code while a condition is True.

Syntax:

```
While condition
    'Statements
Wend
```

Example:

```
Dim i As Integer
i = 1
While i <= 5
    Print i
    i = i + 1
Wend
```

Do While...Loop

Checks the condition before executing the loop.

Runs only if the condition is true.

Example:

```
Dim x As Integer
x = 1
Do While x <= 3
    Print "Value of x is "; x
    x = x + 1
```

Loop

Do Until...Loop

Executes until the condition becomes True.

Example:

```
Dim n As Integer
n = 1
Do Until n > 5
   Print n
   n = n + 1
Loop
```

Do...Loop While

Executes the loop first, then checks the condition.

Example:

```
Dim a As Integer
a = 1
Do
   Print a
   a = a + 1
Loop While a <= 5
```

Do...Loop Until

Executes the block at least once, checks condition after execution.

Example:

```
Dim y As Integer
y = 1
Do
   Print y
   y = y + 1
Loop Until y > 5
```

Advantages:

Used when the number of repetitions is unknown.

Useful for reading files, waiting for user input, or processing until a condition changes.

22. How to work with graphics in Visual Basic? Explain.?

Visual Basic provides powerful tools to draw and manipulate graphics on forms and picture boxes.

1. Graphics Controls

Form or PictureBox can be used to draw shapes.

Common properties:

AutoRedraw – Keeps drawings visible even after the form refreshes.

ScaleMode – Sets the unit of measurement (pixels, twips).

2. Drawing Methods

VB provides built-in methods for drawing:

Method	Description	Example
Line	Draws a line or rectangle	Line (0, 0)-(200, 200), vbRed
Circle	Draws a circle or ellipse	Circle (100, 100), 50, vbBlue
PSet	Plots a single pixel	PSet (50, 50), vbGreen
Cls	Clears drawings from a form or PictureBox	Cls

3. Example Program

```
Private Sub Form_Click()
    Me.Cls
    Line (100, 100)-(200, 200), vbRed
    Circle (150, 150), 50, vbBlue
    PSet (200, 200), vbGreen
End Sub
```

4. Loading Images

You can display images using PictureBox or Image control.

```
Picture1.Picture = LoadPicture("C:\image.bmp")
```

5. Animation

The Timer Control can be used for simple animations by redrawing shapes at different positions.

6. Use of Graphics:

Used in games, visual simulations, charts, and custom UI designs.

23. Discuss about lists and Arrays with necessary examples.?

1. Lists:

A List is a collection of items displayed using controls like ListBox or ComboBox.

ListBox Example:

List1.AddItem "Apple"

List1.AddItem "Banana"

List1.AddItem "Mango"

Properties:

ListCount – Number of items.

ListIndex – Index of selected item.

Text – Text of selected item.

ComboBox Example:

Combo1.AddItem "Red"

Combo1.AddItem "Blue"

Advantages:

Easy to store and display multiple values.

Saves space and simplifies selection.

2. Arrays:

An Array stores multiple values of the same data type under a single variable name.

Types of Arrays:

Single-Dimensional Array:

Dim Marks(4) As Integer

Marks(0) = 90

Marks(1) = 80

Multi-Dimensional Array:

Dim Matrix(2, 2) As Integer

Matrix(0, 0) = 1

Matrix(0, 1) = 2

Loop Example:

Dim i As Integer

Dim num(5) As Integer

For i = 0 To 5

    num(i) = i * 2

```
    Print num(i)
Next i
```

Dynamic Arrays:

Arrays whose size can change at runtime.

```
Dim arr() As Integer
ReDim arr(3)
```

Advantages of Arrays:

Efficient data storage.

Easy to process collections of similar items.

Supports loops for bulk operations.

24. Explain the following:

(a) File Handling.?

File handling in VB allows programs to store, retrieve, and modify data on disk.

File Modes:

Input – For reading files.

Output – For writing new files.

Append – For adding data to existing files.

Random – For accessing data using record numbers.

Example (Sequential File):

```
Open "C:\student.txt" For Output As #1
Write #1, "John", 85
Close #1
```

Reading from File:

```
Open "C:\student.txt" For Input As #1
Input #1, name, mark
Close #1
```

File Functions:

EOF(1) – Checks end of file.

FreeFile – Returns next available file number.

Advantages:

Permanent storage of data.

Easy retrieval and modification.

(b) DLL Servers (Dynamic Link Libraries).?

DLL (Dynamic Link Library) is a file that contains reusable code or functions shared by multiple programs.

Features:

Promotes code reuse.

Saves memory since multiple programs can share a single DLL.

Easier to update functionality without changing the main program.

Example of Using DLL in VB:

```
Declare Function GetTickCount Lib "kernel32" () As Long


Private Sub Command1_Click()
    MsgBox GetTickCount()
End Sub
```

Here, GetTickCount is a Windows API function from kernel32.dll.

Advantages:

Modular programming.

Reduces executable file size.

Simplifies maintenance.