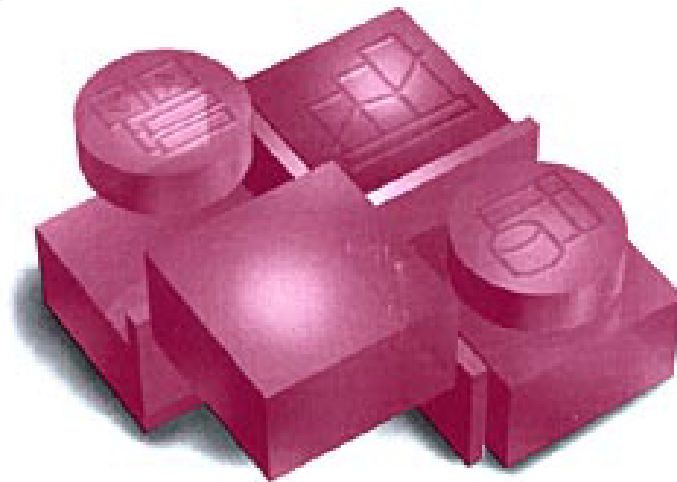


Standard Grade Computing Studies

Pupil Notes Book 3



Microsoft
Visual Basic 6.0
Professional Edition

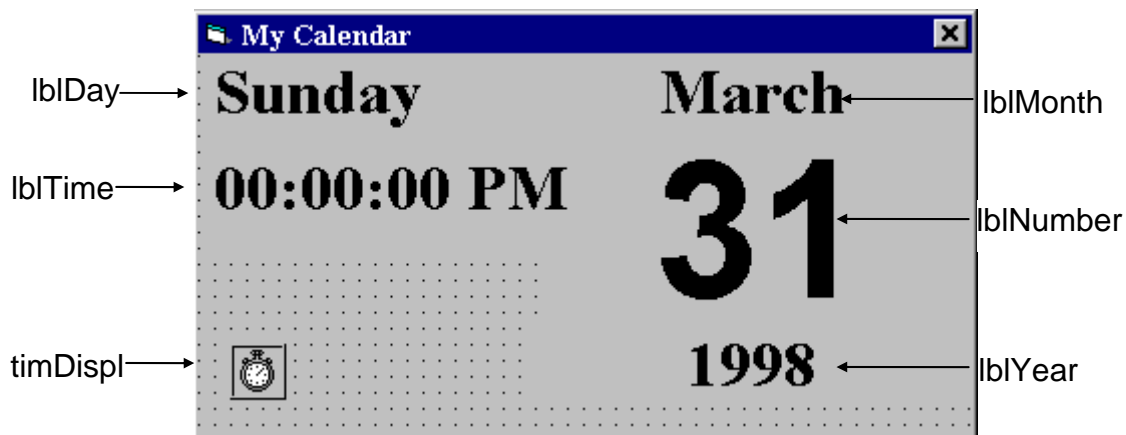
Visual Basic 6

Do

Calendar Program

We are going to design a program that displays the current month, day, and year. Also, display the current time, updating it every second. Make the window look something like a calendar page. Play with object properties to make it pretty.

1. Open Visual Basic from the Start Menu.
2. Create a form as shown below:



3. Set the properties as shown below:

Object	Property	Value
Form	Name	frmCalendar
	Caption	My Calendar
	BorderStyle	Fixed Single
Timer	Name	timDisplay
	Interval	1000
Label 1	Name	lblDay
	Caption	Sunday
	FontName	Times New Roman
	FontBold	True
	FontSize	24
Label 2	Name	lblTime:

	Caption	00:00:00 PM
	FontName	Times New Roman
	FontBold	True
	FontSize	24
Label 3	Name	IblYear:
	Alignment	Center
	Caption	1998
	FontName	Times New Roman
	FontBold	True
	FontSize	24
Label 4	Name	IblNumber:
	Alignment	Center
	Caption	31
	FontName	Arial
	FontBold	True
	FontSize	72
Label 5	Name	IblMonth:
	Alignment	Center
	Caption	March
	FontName	Times New Roman
	FontBold	True
	FontSize	24

4. Double click on the Timer icon to open the code window:

5. Above Private Sub type the General Declaration:

Option Explicit

6. Between Sub and End Sub type the following code:

Dim Today As Variant

Today = Now

lblDay.Caption = Format(Today, "dddd")

lblMonth.Caption = Format(Today, "mmmm")

lblYear.Caption = Format(Today, "yyyy")

lblnumber.Caption = Format(Today, "d")

lblTime.Caption = Format(Today, "h:mm:ss ampm")

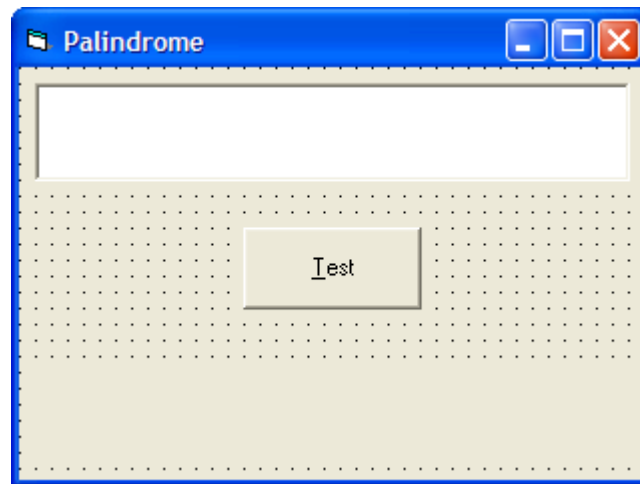
7. Adjust the colour properties of each object to personalise your calendar program's HCI.
8. Save your form as frmCalendar and project as prjctCalendar.
9. Run your program to test that it works.

Do

Palindrome Tester

Too bad I hid a boot. If you read this phrase backwards you will get the same phrase, these are known as palindromes. Other examples include words such as dad, radar, rotor, pip and toot. We are going to create a program that will prompt the user to enter a word or a phrase and test to see if it is a palindrome. The program will display a suitable message to inform the user if their word or phrase is indeed a palindrome.

1. Create a form with a text box, command button and a label similar to the one shown below.



2. Set the properties of each object as shown below.

Object	Property	Value
Form	Name	frmCalendar
	Caption	Palindrome
Text 1	Name	txtInput
	Text	Blank
Label 1	Name	lblAnswer
	Caption	Blank
Command 1	Name	cmdTest
	Caption	&Test

3. Double-click on the command button cmdTest and add the following code between Sub and End Sub.

```
Private Sub cmdTest_Click()
Dim word As String, Pal As String
word = txtInput.Text
Pal = StrReverse(word)
If Not (Pal = word) Then
lblAnswer.Caption = "Sorry that's not a palindrome"
Else
lblAnswer.Caption = "Yes that's a palindrome. Well Done!"
End If
End Sub
```

Remember to add suitable internal comments to you program.

4. Save the form as frmPalindrome and project as prjctPalindrome.
5. Test your program to see if it works.

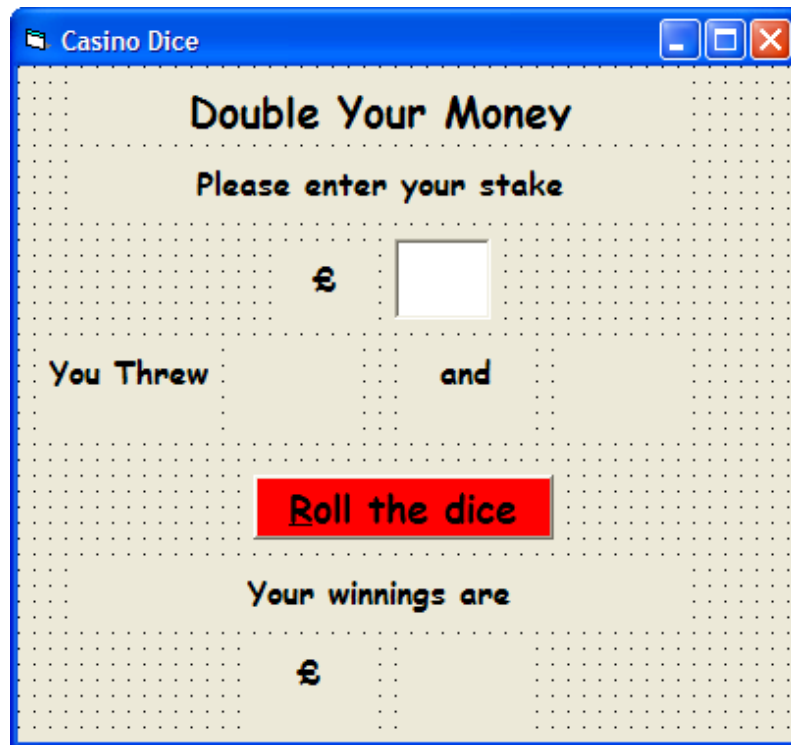
Do

Dice Game

We are going to create a program that will be used to simulate the casino dice game. The user will enter a stake and the program will "throw the dice". If the user hits a double then their stake is doubled.

1. Create a user interface with:
 - 1 form (Name it frmDice)
 - 7 labels to document the form. (Name them lblHeading1 to 7)
 - 1 textbox. (name it txtStake)
 - 2 labels to show the throws. (name them lblDie1 and lblDie2)
 - 1 label to show the winnings. (name it lblWinnings)
 - 1 command button. (name it cmdThrow)
2. Set the properties of each object including the form to individualise your program. (Colours, fonts etc)

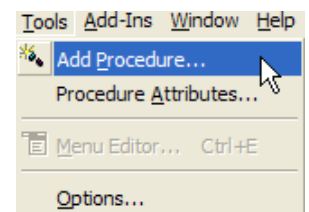
Your form design may be similar to the one shown below.



We are going to create a procedure to roll the dice. A procedure is code which is hidden in the main program but is called for use within the program. Although our program will call this procedure only once, procedures are most commonly used when the same piece of program (or procedure) is used more than once within the program.

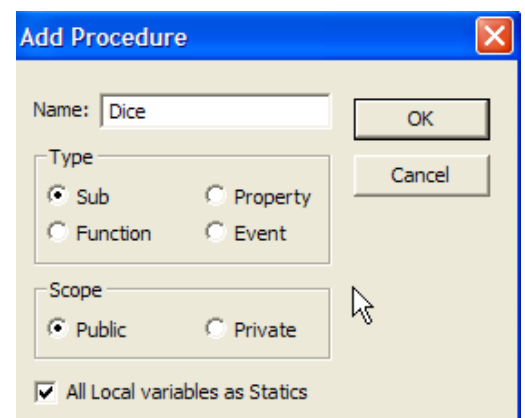
3. Open the code window for the command button cmdThrow.

4. From the Tools menu choose Add Procedure.



5. Complete the Add Procedure window as shown opposite.

6. Click OK.



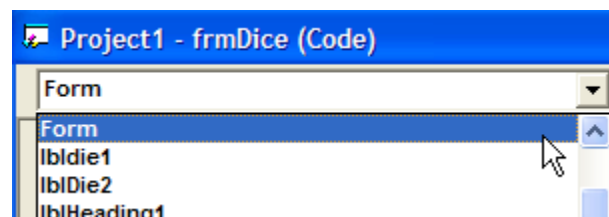
7. Add the following code exactly as shown below.

```
Public Static Sub Dice()  
Dim Throws As Integer  
Dim Doubles As Integer  
Dim Winnings As Integer  
Dim Stake As Integer  
Dim w1 As Integer  
Throws = Throws + 1  
    If Throws > 5 Then  
        If Doubles = 0 Then  
            MsgBox "Oh well, you lose this game. Have another go!"  
            Winnings = 0  
            Throws = 0  
            Exit Sub  
        End If  
        MsgBox "Game Over"  
        Stake = 0  
        Throws = 0  
        Winnings = 0  
        txtStake.SetFocus  
        Exit Sub  
    End If  
    lblDie1 = Int(6 * Rnd) + 1  
    lblDie2 = Int(6 * Rnd) + 1  
    If lblDie1.Caption = lblDie2.Caption Then  
        Doubles = Doubles + 1  
        Stake = Val(txtStake.Text)  
        Winnings = Val(Stake) * 2  
        w1 = Winnings + Stake  
  
        lblWinnings = w1  
    End If  
End Sub
```

8. Add the procedure name Dice to the command button section of the code window.

```
Private Sub cmdThrow_Click()  
    Dice  
End Sub
```

9. Choose form from the drop down arrow on the code window.



10. Add Randomize to the form load window as shown below.

```
Private Sub Form_Load()  
    Randomize  
End Sub
```

11. Save the form as frmDice and the project as prjctDice.

12. Run your program to see if it works.

Read

Defensive Programming

When you use any software on a computer, you can get (very) annoyed when the program crashes. A program which does not crash easily is called a **ROBUST** program.

When you write a program you should try to make it **ROBUST** and there are 2 ways in Visual Basic to do this, - **Input Validation** and **Error Trapping**. When you use these techniques in your programs you are said to be using **DEFENSIVE PROGRAMMING TECHNIQUES**.

The two defensive programming techniques used in Visual Basic are:

1. **Input Validation** - Using a While...Wend loop which stops the program from progressing until valid input has been entered.
2. **Error Trapping** - Using 'On Error GoTo' to send the program to an error handling routine which provides an error message and allows the program to proceed or halt if necessary.

Read

Input Validation

Imagine you are writing a program which asks a user for their age. It might make your program give wrong answers if the user makes a mistake, such as entering their age as 150 instead of 15. Input Validation checks the user input and doesn't let the program move on until a sensible input is entered. You, the programmer set the range of 'sensible' inputs in your code.

Without Input Validation part of your program might look like this:

```
Age = InputBox("Please enter your age")
```

With Input Validation your code would look like this:

```
Age = InputBox("Please enter your age")
```

```
While Age < 0 or Age > 120
```

```
    Age = InputBox("Sorry that is out of range. Please re-  
enter your age")
```

```
Wend
```

In the above code, you ask for the input as usual, then you use a new type of loop called a **While ... Wend** loop which checks the value of the first input to see if it is less than 0 or if it is greater than 120. If the value is not valid, the program will then ask for the age to be re-entered, and this will continue until a valid input is made.

The method (or ALGORITHM) for this is:

Use INPUTBOX to ask for data

WHILE input not valid

Use INPUTBOX to re-enter

WEND

(WEND stands for While END.)

Read

Using Input Validation is easy once you practice it a few times and it is something you should always use when you input data into your programs.

Error Trapping

The second method of Defensive programming is called error trapping. When you run one of your programs, the user can sometimes get an error message when something goes wrong in the program, causing the program to stop.

Error trapping is where you add code into your program to tell Visual Basic what to do when an unexpected error happens. (The proper name for this type of error is a *run-time error*.)

Examples of run-time errors are when you click on the *Cancel* button of an InputBox, or if your program tries to open a file that is not there, or if your program tries to divide by zero. (Computers get confused when you ask them to divide by zero!)

Here is the code from a simple program which has one button on a form. The button is called cmdEnterDetails.

```
Private Sub cmdEnterDetails_Click()  
    Dim names As String  
    Dim age As Integer  
    names = InputBox("please enter your name")  
    age = InputBox("Please enter your age")  
    MsgBox "Your name is " & names & " and you are " &  
    age & " years old"  
End Sub
```

If you run this code and click on the cancel button at each of the InputBox lines, you will find that on the second one, when you are asked for your age, you will cause a *run-time error* to happen.

Read

(The reason it doesn't happen the first time is a bit technical - when you click the cancel button, a 'space' character is returned, in place of the normal input. A space is a string character, so when asking for a string variable, (names), this is OK. When asking for an integer, (age), pressing the cancel button returns a space character, as before, but this is a string, not an integer, so you get a 'type mismatch' error.)

In defensive programming, you, the programmer must try to predict any run-time errors such as this, and make your program more robust by adding the following code.

New code with error trapping

```
Private Sub cmdEnterDetails_Click()  
    On Error GoTo errorhandler  
  
    Dim names As String  
    Dim age As Integer  
    names = InputBox("please enter your name")  
    age = InputBox("Please enter your age")  
    MsgBox "Your name is " & names & " and you are " & age & "  
    years old"  
  
    Exit Sub  
    errorhandler: ' this is the line label  
    MsgBox "Oops, an error has occurred, press OK to continue"  
    Resume Next ' this tells the program to go on with the next line  
                    after the error  
  
End Sub
```

Read

Notes:

- ❖ **On Error GoTo errorhandler** should be the first line of the procedure.
- ❖ **Exit Sub** - this statement must be in at the end of the normal code, and just before the line label. This makes the procedure end before doing the error routine if there are no errors.
- ❖ **errorhandler:** the line label must match the name in the 'On Error GoTo' line. It must also have a colon (:) after it to mark it as a line label.
- ❖ **MsgBox** - put the message which you want to appear when an error occurs.
- ❖ **Resume Next** - tells the program to go on with the next line of code after the one which has caused the error.
- ❖ You can use any name in place of errorhandler if you want.

Do

Guess my lucky number

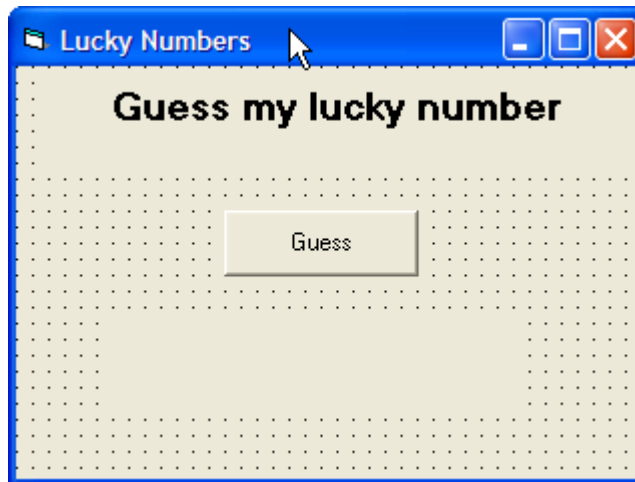
We are going to create a program that will prompt a user to guess the lucky number. The user's guess must be between 1 and 10, if not the user will be prompted to enter another guess higher or lower than their last guess. The program will then display a suitable message telling them the lucky number and how many valid attempts they had.

Program Design

Problem (In my own words)

I have to design a program that will ask the user to guess the lucky number. The program will check this number with the lucky number and if wrong it will inform the user that is either too high or too low. If the user guesses the lucky number the program will display the number and tell them how many guesses they had. The program will also validate the user's input to ensure it is between 1 and 10. If out with the range the user will be informed.

Interface Design



Property Grid

Object	Property	Value
Form	Name	frmLuckynos
	Caption	Lucky Numbers
Command Button	Name	cmdGuess
	Caption	Guess
Label	Name	lblHeading
	Caption	Guess my lucky number
Label	Name	lblTries
	Caption	(Blank)

Algorithms for Each Button (Procedure)

Start

1. Set Counter
2. Get user guess
3. Check user input and validate
4. Repeat until user guesses lucky number

5. Display lucky number
6. Display number of attempts

Refinements

- 1.1 set counter to 1

- 2.1 Display input box prompting user to enter their guess
- 2.2 Assign user guess to variable **guess**

- 3.1 Repeat until user input is between 1 and 10

- 4.1 Repeat until variable **guess** = 7
- 4.2 If variable **guess** is not 7 then display suitable message
- 4.3 Prompt user to enter new guess and assign to variable **guess**

- 5.1 If variable **guess** = 7 display suitable message

- 6.1 Display value of variable **Counter** in label **Tries**

Evaluation

I found this program fairly easy as I have completed programs that had loops and if statements previously. I had some difficulty with the validation of the user input but all I had to do was to take the skills I had learned in earlier exercises and adapt them to suit the program requirements. If I had to improve my program I would create a nicer user interface with graphics or coloured text and prompt the user if they didn't enter a number.

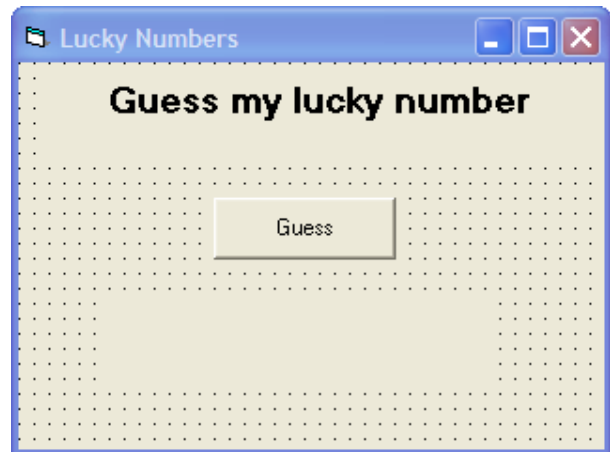
You now, are going to create the program above from design through to evaluation using the sheets provided.

1. Copy the problem statement, design grid and properties table shown above.
2. Create the HCI, change the object properties and add the code.

Show your teacher your design sheets.

Do

3. Create your form with a command button and two labels. As shown.
4. Change the properties of each object including the form as per your design.



5. Double-click on the **cmdGuess** button and add the following code.

```

Counter = 1
guess = InputBox("Please enter your guess")
While guess < 0 Or guess > 10
    guess = InputBox("Sorry that is out of range. Please re-enter
        your guess")
Wend
Do While guess <> 7
If guess > 7 Then
MsgBox "try lower"
Counter = Counter + 1
guess = InputBox("Sorry that is too high. Please re-enter your
    guess")
ElseIf guess < 7 Then
MsgBox "try higher"
guess = InputBox("Sorry that is too low. Please re-enter your
    guess")
Counter = Counter + 1
End If
Loop
'guess = 7 Then
MsgBox "well done my lucky number is 7"
lbltries.Caption = "You needed " & Counter & " goes"
    
```

Note: Make sure the label names used in the code match the properties of **your** labels.

7. Save your project and form with a suitable name.
8. Run your program to see if it works.
9. Demonstrate your working program to your teacher.

Copy the Guess Number assessment from the PrepWork folder into your own folder.

PRACTICAL ABILITIES TASK

GUESS NUMBER

Your teacher will outline how you should do this task and give you opportunities for discussion. Read and follow all instructions carefully, use hardware and software properly and write your report neatly using the headings given below.

A program is required to prompt the user to guess a randomly-chosen whole number between 1 and 20. The input should be validated. If the guess is incorrect, the user should be told if the target number is bigger or smaller. This process should continue until the target number is guessed correctly. The user should then be told how many valid guesses were made.

An example of output is shown below. The output from your program may look different but must meet the specification.

```
I am thinking of a whole number between 1 and 20.

What is the number? 83
Enter a whole number between 1 and 20, please. 0
Enter a whole number between 1 and 20, please. 4.9
Enter a whole number between 1 and 20, please. 11
  My number is smaller than your guess.
What is the number? 3
  My number is bigger than your guess.
What is the number? 8
  My number is smaller than your guess.
What is the number? 5
  My number is bigger than your guess.
What is the number? 7

Correct. I was thinking of 7.

The number of valid guesses was 5.
```

1 DESCRIBE METHOD - Analysis

3 marks

Show that you understand what is required by describing how you will do this task.

2 LIST STEPS - Design *6 marks*

Show the main steps and refinements.

3 ENTER PROGRAM - Implementation *5 marks*

Enter the program listing. Include internal commentary. Correct any mistakes that you make. Save the listing.

4 TEST PROGRAM - Testing *6 marks*

Describe in detail how you tested fully that your program - including input validation - worked as specified.

5 GET PRINTOUTS - Implementation *2 marks*

Get **two** printouts with your name in the footer. One should show your listing. The other printout should show a run using one of your sets of test data. It should demonstrate input validation and show accurate output.

6 WRITE TECHNICAL GUIDE - Documentation *3 marks*

Write a short technical guide that describes how to install the programming language that you used and the system requirements.

7 CHANGE PROGRAM - Maintenance *3 marks*

Describe how you would change the program to make it count the total number of guesses and not just the valid ones.

8 SUGGEST IMPROVEMENT - Evaluation *2 marks*

Suggest **one** way in which your program could be better apart from the maintenance given above.

Read

The Select Case Statement

The If/Then statements that you have used previously allowed your programs to make decisions. However, the If/Then statement is only one way a computer can make decisions. Using a Select Case statement can give the same result.

The Select Case statement compared to the If/Then statement.

The If/Then Statement

```
Private Sub Cmdprocess_Click()  
Dim name As String  
name = Txtname.Text  
name = UCase(name)  
If name = "BART" Then  
    MsgBox "Hi,Bart, my man"  
ElseIf name = "LISA" Then  
    MsgBox "Hi, Lisa"  
ElseIf name = "HOMER" Then  
    MsgBox "Hi, Homer"  
Else  
    MsgBox "Hi, stranger"  
End If  
End Sub
```

Read

The Select Case statement

```
Private Sub cmdProcess_Click()  
Dim name as String  
name = txtName.Text  
name = UCase (name)  
Select Case name  
    Case "Bart"  
        MsgBox "Hi Bart"  
    Case "Lisa"  
        MsgBox "Hi Lisa"  
    Case "Homer"  
        MsgBox "Hi Homer"  
    Case Else  
        MsgBox "Hi Stanger"  
End Select  
End Sub
```

The procedure using the Case Statement is about the same size as the one using the If/Then statement. However, it is easier to understand what is going on because the decision-making is more concise.

An explanation of the code.

The Select Case statement begins with

Select Case name

This tells Visual Basic that that you are starting a Select Case statement

The name part of the line tells Visual Basic that the computer will make a decision based on the contents of the variable name.

Read

Case "BART"

This is just a quick way of saying

If name = "BART" Then

This Case Statement works with strings, but Case Statements can also handle other data types, such as integers. The Case Statement used in the Bart program checks a variable for a specific value. However, Case Statements can also check for a range of values. For example -

A program that will allow the user to enter a mark. The program then outputs a message that tells the user what grade that mark will get.

A mark over 80 gets a grade 1

A mark between 60 - 80 gets a grade 2

A mark between 50 and 60 gets a grade 3

A mark under 50 fails.

Here is the structure of the Case statement

```
mark = InputBox("Please enter mark")
Select Case mark
Case Is >= 80
MsgBox ("You have scored a grade 1")
Case Is >= 60
MsgBox ("You have scored a grade 2")
Case Is >= 50
MsgBox ("You have scored a grade 3")
Case Else
MsgBox ("Sorry you havee failed the exam")
End Select
```

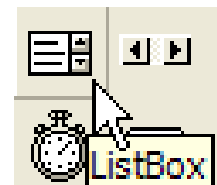
Do

International Welcome Program

We are going to design a program that will be used at an international convention for doctors. The program will allow the doctors from four countries to select their country from a list and receive instructions about where to go.

1. Open Visual Basic from the Start menu.
2. Create a form similar to the one below with 4 labels, 1 command button and 1 list box.

List boxes are used to display information as a table from which the user can select one or more from the list. To create a List box you will need to select the ListBox icon.



3. Set the properties of each object as shown in the properties grid.

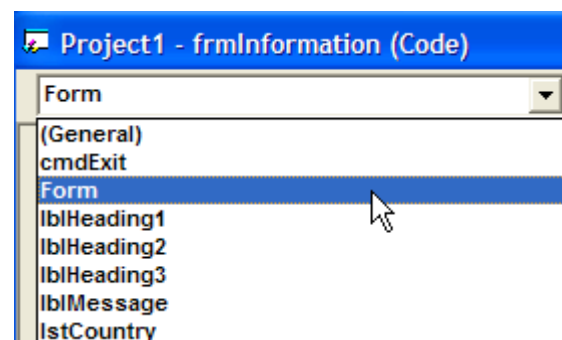
Object	Property	Value
Form	Name	frmWelcome
	Caption	Doctor's Information
Command 1	Name	cmdExit
	Caption	Exit
Label 1	Name	lblHeading1
	Caption	International Welcome Program
Label 2	Name	lblHeading2
	Caption	Choose a country
Label 3	Name	lblHeading3
	Caption	Blank
Label 4	Name	lblMessage
	Caption	Blank
List 1	Name	lstCountry

- Double-click on the cmdExit button and add the code to end the program.

```
Private Sub cmdExit_Click()
End
End Sub
```

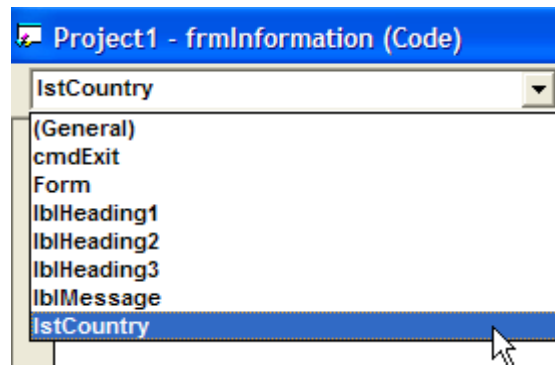
- From the drop down menu in the code window choose Form.

- Add the following code between Sub and End sub.



```
Private Sub Form_Load()
    lstCountry.AddItem "England"
    lstCountry.AddItem "Germany"
    lstCountry.AddItem "Spain"
    lstCountry.AddItem "Italy"
End Sub
```

7. From the drop down menu in the code window choose lstcountry.
8. Add the following code between Sub and End sub.



```
Private Sub lstCountry_Click()  
    lblHeading3.Caption = lstCountry.Text  
    Select Case lstCountry.ListIndex  
    Case 0  
        lblMessage.Caption = "Hello, Doctor, please report to  
        reception"  
    Case 1  
        lblMessage.Caption = "Hallo, Doktor, bitte berichten sie zu  
        empfang"  
    Case 2  
        lblMessage.Caption = "Hola, Medico, informe por favor a la  
        recepcion"  
    Case 3  
        lblMessage.Caption = "Ciao, per favore di riferire alla ricezione"  
    End Select  
End Sub
```

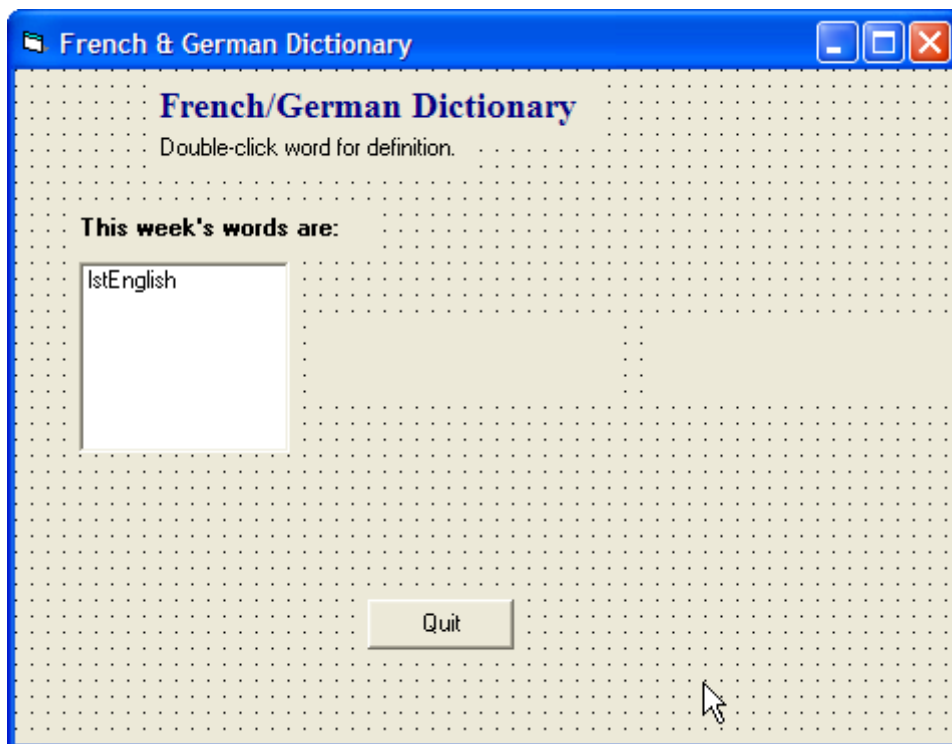
9. Save your form as frmWelcome and the project as prjctWelcome.
10. Run your program to see if it works.

Do

French & German Vocabulary

You have been asked by the Modern Languages teacher to develop a program that will allow the user to choose a word from a list of English words and display the French and German equivalents.

1. Open Visual Basic from the Start menu.
2. Use your previous programming experience to create the above program. (Hint you will need to have 5 labels). See form and possible code below.



Form Code

```
Private Sub Form_Load()  
    lstEnglish.AddItem "Door"  
    lstEnglish.AddItem "Shower"  
    lstEnglish.AddItem "Bath"  
    lstEnglish.AddItem "Book"  
    lstEnglish.AddItem "Cat"  
    lstEnglish.AddItem "Dog"  
    lstEnglish.AddItem "Cake"  
End Sub
```

End Code

```
Private Sub cmdQuit_Click()  
    MsgBox ("Auvoir, Cheuse")  
End  
End Sub
```

List Code

```
Private Sub lstEnglish_Click()  
    Select Case lstEnglish.ListIndex  
        Case 0  
            lblFrench.Caption = "The French word is Porte"  
            lblGerman.Caption = "The German word is Tur"  
        Case 1  
            lblFrench.Caption = "The French word is Douche"  
            lblGerman.Caption = "The German word is Dushe"  
        Case 2  
            lblFrench.Caption = "The French word is Bain"  
            lblGerman.Caption = "The German word is Bad"  
        Case 3  
            lblFrench.Caption = "The French word is Livre"  
            lblGerman.Caption = "The German word is Buch"
```

Case 4

IblFrench.Caption = "The French word is Chat"

IblGerman.Caption = "The German word is Katze"

Case 5

IblFrench.Caption = "The French word is Chien"

IblGerman.Caption = "The German word is Hund"

Case 6

IblFrench.Caption = "The French word is Gateau"

IblGerman.Caption = "The German word is Kuchen"

End Select

End Sub

3. Save your form as frmVocabulary and the project as prjctVocabulary.
4. Run your program to see if it works.

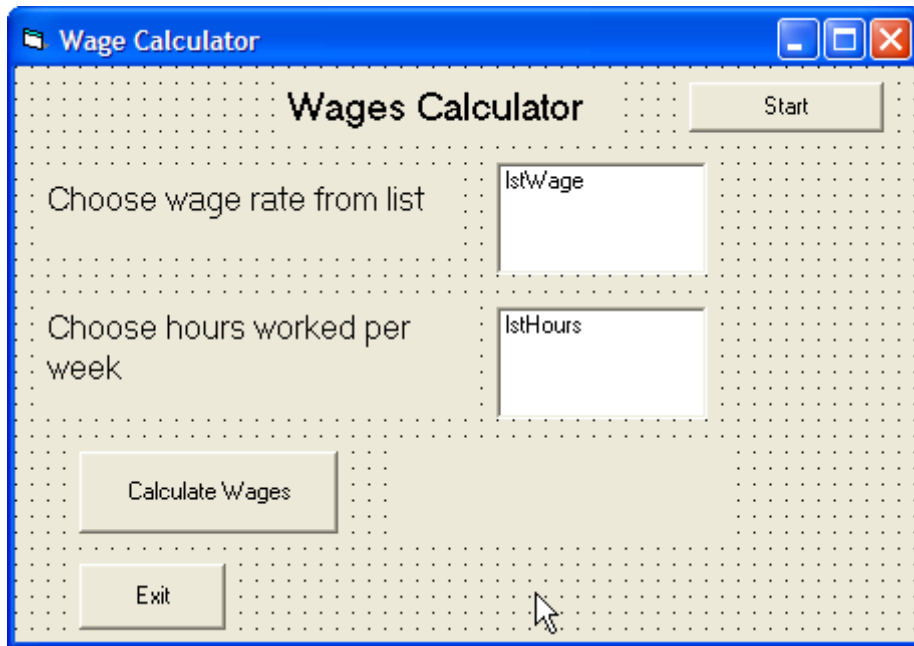
Do

Wages Calculator

Betty would like a program which will help her calculate the wages for different categories of employees i.e. a junior worker receives £5 per hour and may work 40 hours per week. The calculator should have two list boxes, and start, calculate and end buttons. The pay rates and hour selections are shown below:

Wages	Hours
£5	25
£6	30
£7	40

1. Open Visual Basic from the Start menu.
2. Create a form similar to the one below with 4 labels, 3 command buttons and 2 list boxes.



3. Set the properties of each object as shown in the properties grid.

Object	Property	Value
Form	Name	frmWage
	Caption	Wage Calculator
Command 1	Name	cmdExit
	Caption	Exit
Command 2	Name	cmdStart
	Caption	Start
Command 3	Name	cmdCalc
	Caption	Calculate Wages
Label 1	Name	lblHeading1
	Caption	Wages Calculator
Label 2	Name	lblHeading2
	Caption	Choose wage rate from list
Label 3	Name	lblHeading3
	Caption	Choose hours worked per week
Label 4	Name	lblWages
	Caption	Blank

List 1	Name	IstWage
	Enabled	False
	Visible	False
List 2	Name	IstHours
	Enabled	False
	Visible	False

4. Add the following code under each object.

Program Code:

General

Option Explicit

Dim totalwages As Integer

Calculate Command

```
Private Sub cmdCalc_Click()
```

```
totalwages = Val(IstWage.Text) * Val(IstHours.Text)
```

```
lblWages.Caption = "£" & totalwages
```

```
End Sub
```

```
Private Sub cmdExit_Click()
```

```
End
```

```
End Sub
```

Start Command

```
Private Sub cmdStart_Click()
```

```
IstWage.Enabled = True
```

```
IstHours.Enabled = True
```

```
IstWage.Visible = True
```

```
IstHours.Visible = True
```

```
End Sub
```

Form

```
Private Sub Form_Load()  
lstWage.AddItem "5.00"  
lstWage.AddItem "6.00"  
lstWage.AddItem "7.00"  
lstHours.AddItem "25"  
lstHours.AddItem "40"  
lstHours.AddItem "50"  
End Sub
```

5. Save your form as frmWages and the project as prjctWages.
6. Run your program to see if it works.

Copy the Language Course assessment from the PrepWork folder into your own folder.

PRACTICAL ABILITIES TASK LANGUAGE COURSE

Your teacher will outline how you should do this task and give you opportunities for discussion. Read and follow all instructions carefully, use hardware and software properly and write your report neatly using the headings given below.

Students taking a language course must pass examinations in French and German or French and Spanish to pass. A pass is half marks or more. A program is required to take in the three marks which are validated as being whole numbers between 0 and 30. The output indicates if the student has passed or failed and gives the percentage mark to the nearest whole number.

Examples of output are shown right. The outputs from your program may look different but must meet the specification.

```
LANGUAGE COURSE

Enter French mark. 21
Enter German mark. 18
Enter Spanish mark. 13

You have passed.
Your percentage mark was 58.
```

```
LANGUAGE COURSE

Enter French mark. 14
Enter German mark. 20
Enter Spanish mark. 19

You have failed.
Your percentage mark was 59.
```

```
LANGUAGE COURSE

Enter French mark. 74
Enter a whole number between 0 and 30. -9
Enter a whole number between 0 and 30. 25
Enter German mark. 28
Enter Spanish mark. 15

You have passed.
Your percentage mark was 76.
```

1 DESCRIBE METHOD - Analysis *3 marks*

Show that you understand what is required by describing how you will do this task.

2 LIST STEPS - Design *6 marks*

Show the main steps and refinements.

3 ENTER PROGRAM - Implementation *5 marks*

Enter the program listing. Include internal commentary. Correct any mistakes that you make. Save the listing.

4 TEST PROGRAM - Testing *6 marks*

Describe in detail how you tested fully that your program - including input validation - worked as specified.

5 GET PRINTOUTS - Implementation *2 marks*

Get **two** printouts with your name in the footer. One should show your listing. The other printout should show a run using one of your sets of test data. It should demonstrate input validation and show accurate output.

6 CHANGE PROGRAM - Maintenance *2 marks*

Describe how you would change the program to make a pass in Italian an **additional** requirement to pass the course.

7 SUGGEST IMPROVEMENT - Evaluation *2 marks*

Suggest **one** way in which your program could be better apart from the maintenance given above.

Read

Arrays

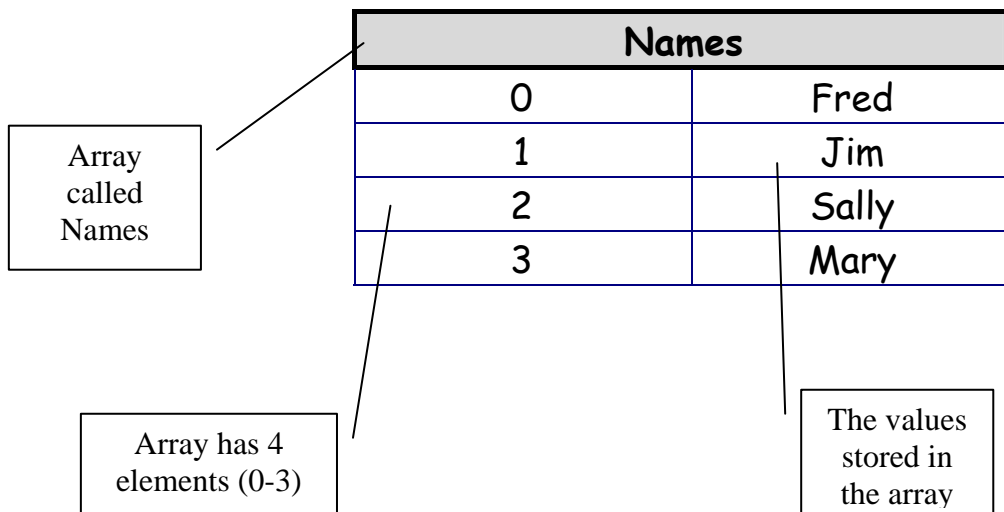
An array is much easier to use than it is to describe. If you can use arrays in your program, you are becoming a *real* programmer.

An array is like a table, where you can use one variable name (like *names*, or like *ages*) to hold several values at the same time.

Until now, you have been using variables to hold a single value at a time. Arrays let you hold more than one value with a single variable name.

Examples:

Dim Names(4) as String



Read

Dim ages(6) as Integer

Ages	
0	12
1	13
2	12
3	14
4	13
5	16

This array has 6 elements, numbered from 0 to 5, which can store 6 integer numbers.

Setting Up An Array

You set up, or declare, an array in much the same way as you set up a normal variable.

The only difference is that you put a number in brackets to set up the number of *elements* you will have in your array.

Examples:

Dim money(12) as Integer

Dim people(99) as String

These examples will set up arrays to hold 12 Integer numbers, and one to hold 99 Strings.

Using An Array

Here is some code from a button in Visual Basic which will set up an array to hold 5 names, then loop 5 times, asking the user to enter names, which will then be stored in the array.

Read

```
Dim names(5) as String ' Set up an array with 5 elements (0-4)
Dim loops as Integer   ' Set up loop variable
For loops = 0 to 4     'Note the starting point of 0
Names(loops) = Inputbox("Please enter the name of person " & loops + 1)
Next
```

Notes:

The first time the loop is done, the value of *loops* will be 0.

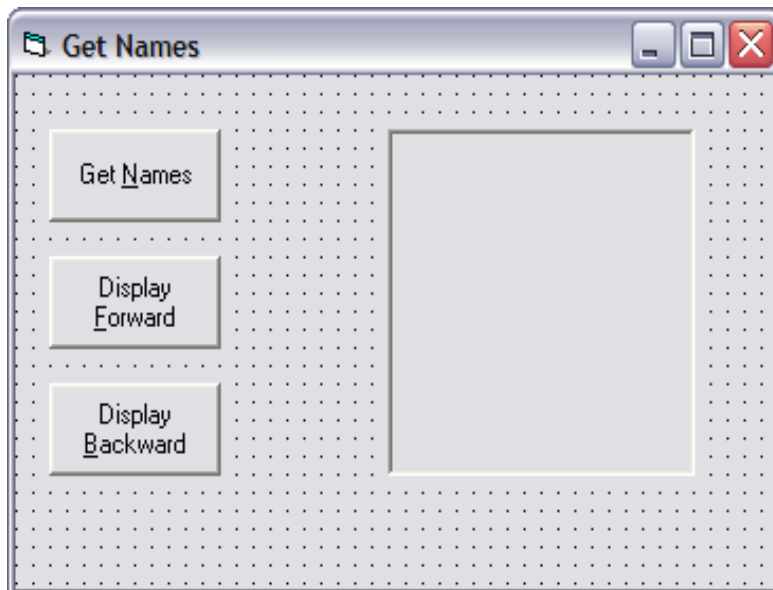
The first question asked will be - **Please enter the name of person 1**

Do

Get Names Array

We are going to create a program that will prompt the user to enter 6 friend's names. The program will store these names in an array. The user will then have the option to display the name in the order they where entered or in reverse order.

1. Open Visual Basic from the Start menu.
2. Create a form similar to the one below with 3 command buttons and 1 picture box.



3. Set the properties of each object as shown in the properties grid.

Object	Property	Value
Form	Name	frmNames
	Caption	Get Names
Command 1	Name	cmdGetnames
	Caption	Get &Names
Command 2	Name	cmdForward
	Caption	Display &Forward
Command 3	Name	cmdBackward
	Caption	Display &Backward
Picture Box 1	Name	picDisplay

4. Add the following code under each object.

Program Code:

General

Option Explicit

Dim names(5) As String

Get Names Command

```
Private Sub cmdGetNames_Click()  
Dim loops As Integer  
For loops = 0 To 4  
    names(loops) = InputBox("Please enter the name of person " &  
        loops + 1)  
Next  
End Sub
```

Display Forward Command

```
Private Sub cmdForward_Click()  
Dim loops As Integer  
picDisplay.Cls ' Clears the picture box  
For loops = 0 To 4  
    picDisplay.Print names(loops)  
Next  
End Sub
```

Display Backward Command

```
Private Sub cmdBackward_Click()  
Dim loops As Integer  
picDisplay.Cls ' Clears the picture box  
For loops = 4 to 0 Step -1  
    picDisplay.Print names(loops)  
Next  
End Sub
```

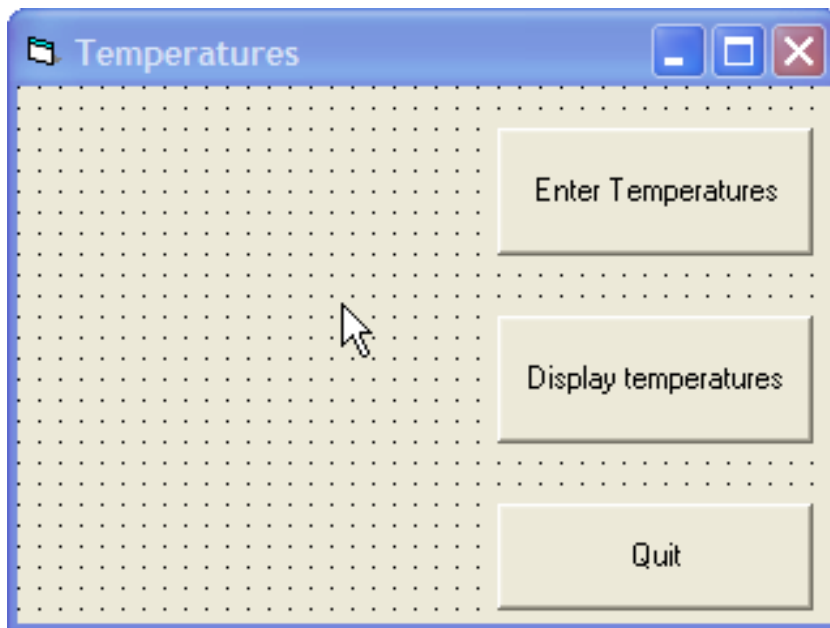
5. Save your form as frmGetnames and the project as prjctGetnames.
6. Run your program to see if it works.

Do

Weekly Temperature Recorder

We are going to create a program that will prompt the user to enter the daily temperature readings for a week. The program will store the daily temperature in an array. The user will then be able to display each daily temperature and average temperature for the week.

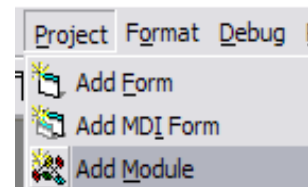
1. Open Visual Basic from the Start menu.
2. Create a form similar to the one below with 3 command buttons



- Set the properties of each object as shown in the properties grid.

Object	Property	Value
Form	Name	frmTemperatures
	Caption	Temperatures
Command 1	Name	cmdEnter
	Caption	Enter Temperatures
Command 2	Name	cmdDisplay
	Caption	Display Temperatures
Command 3	Name	cmdQuit
	Caption	Quit

- From the Project menu choose Add Module.



- Click Open

- Enter the following code into the window.

Option Base 1

Public Temperatures(7) As Variant

- Close the window

- Add the following code under each object.

Program Code:

General

Dim i, total As Single, prompt, title As String

Enter Temperatures Command

```
Private Sub cmdEnter_Click()  
Cls  
    Prompt = "Enter the high temperature."  
    For i = 1 To 7  
        Title = "Day " & i  
        Temperatures(i) = InputBox(Prompt, Title)  
    Next i  
End Sub
```

Display Temperatures Command

```
Private Sub cmdDisplay_Click()  
Print "High temperatures for the week:"  
Print  
For i = 1 To 7  
    Print "Day "; i, Temperatures(i)  
    Total! = Total + Temperatures(i)  
Next i  
Print  
Print "Average high temperature: "; Total / 7  
  
End Sub
```

Quit Command

```
Private Sub cmdQuit_Click()  
End  
End Sub
```

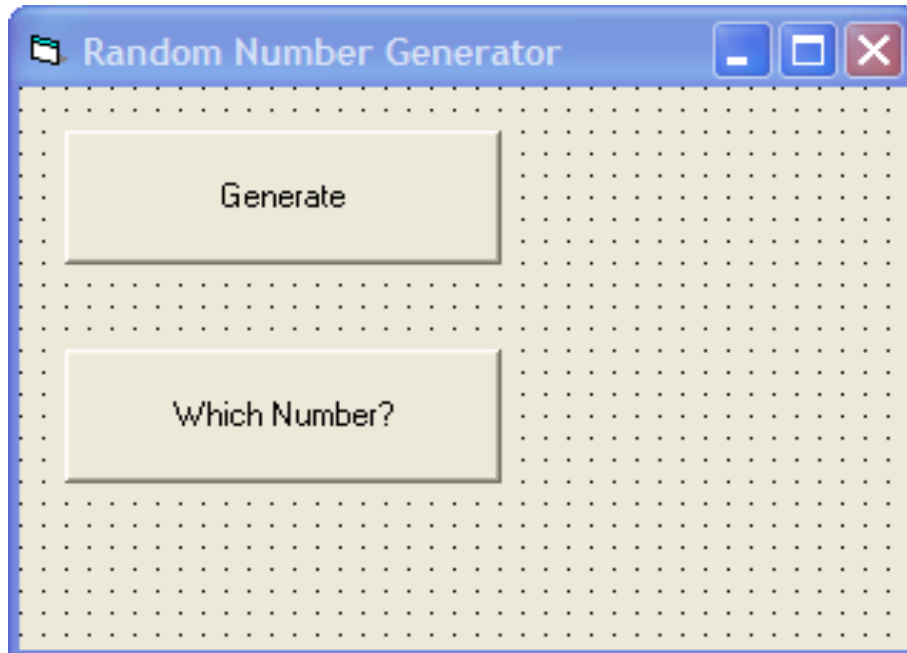
5. Save your form as frmTemperatures, the project as prjctTemperatures and module as mdlTemp.
6. Run your program to see if it works.

Do

Random Number Checker

We are going to create a program that will prompt the user to enter a number between 1 and 50. The program will generate 100 numbers between 1 and 50 and store them in an **array**. The program will then check the users enter against the numbers in the **array** and count how many times their number was generated. The number of occurrences will be shown in a message box.

1. Open Visual Basic from the Start menu.
2. Create a form similar to the one below with 2 command buttons



3. Set the properties of each object as shown in the properties grid.

Object	Property	Value
Form	Name	frmRandomNos
	Caption	Random Number Generator
Command 1	Name	cmdGenerate
	Caption	Generate
Command 2	Name	cmdNumber
	Caption	Which Number?

4. Add the following code under each object.

Program Code:

General

Option Explicit

Dim numbers(100) As Integer

Generate Command

```
Private Sub cmdGenerate_Click()
```

```
Dim Loops As Integer
```

```
For Loops = 1 To 100
```

```
numbers(Loops) = Int((50 * Rnd) + 1)
```

```
Next
```

```
MsgBox "100 Numbers generated", 0, "Generated"
```

```
cmdGenerate.Enabled = False
```

```
End Sub
```

Number Command

```
Private Sub cmdNumber_Click()  
Dim Number_To_Find, Loops, Counter As Integer  
Number_To_Find = InputBox("Which Number Do You Want To  
Find?")  
While Number_To_Find < 1 Or Number_To_Find > 50  
Number_To_Find = InputBox("Sorry out of Range. Which Number  
do you want to find between 1 and 100?")  
Wend  
For Loops = 1 To 100  
If numbers(Loops) = Number_To_Find Then  
Counter = Counter + 1  
End If  
Next  
MsgBox "Number " & Number_To_Find & " Was Found " & Counter  
& " Times "  
End Sub
```

5. Save your form as frmRandomNos, the project as prjctRandomNos.
6. Run your program to see if it works.

Copy the Charity assessment from the PrepWork folder into your own folder.

PRACTICAL ABILITIES TASK

CHARITY LOTTERY

Your teacher will outline how you should do this task and give you opportunities for discussion. Read and follow all instructions carefully, use hardware and software properly and write your report neatly using the headings given below.

A group of twelve office workers decides to have a weekly charity lottery. Each worker is asked to contribute £1 and a draw is made. The winner gets half of the contributions with the other half going to charity. If the person selected has not paid that week, all contributions go to charity. You are required to write a program that selects a winner from a set of workers' names read from data statements and held in an array. The user is asked if the selected person has paid that week. The input should be validated. An appropriate message is given.

Examples of output are shown right. The outputs from your program may look different but must meet the specification.

CHARITY LOTTERY

Isabel Irvine has been selected.
Has Isabel Irvine paid? Y

This week's winner is Isabel Irvine
who gets half of the contributions.
The other half goes to charity.

CHARITY LOTTERY

Clare Carson has been selected.
Has Clare Carson paid? N

Clare Carson did not pay this week
so all contributions go to charity.

CHARITY LOTTERY

Hugh Harper has been selected.
Has Hugh Harper paid? W
Please enter Y for yes or N for no. k
Please enter Y for yes or N for no. y

This week's winner is Hugh Harper
who gets half of the contributions.
The other half goes to charity.

1 DESCRIBE METHOD - Analysis *3 marks*

Show that you understand what is required by describing how you will do this task.

2 LIST STEPS - Design *6 marks*

Show the main steps and refinements.

3 ENTER PROGRAM - Implementation *5 marks*

Enter the program listing. Include internal commentary. Correct any mistakes that you make. Save the listing.

4 TEST PROGRAM - Testing *6 marks*

Describe in detail how you tested fully that your program - including input validation - worked as specified.

5 GET PRINTOUTS - Implementation *2 marks*

Get **two** printouts with your name in the footer. One should show your listing. The other printout should show a run using one of your sets of test data. It should demonstrate input validation and show accurate output.

6 CHANGE PROGRAM - Maintenance *3 marks*

Describe how you would change the program to make it ask if each worker had contributed before the draw was made.

7 SUGGEST IMPROVEMENT - Evaluation *2 marks*

Suggest **one** way in which your program could be better apart from the maintenance given above.