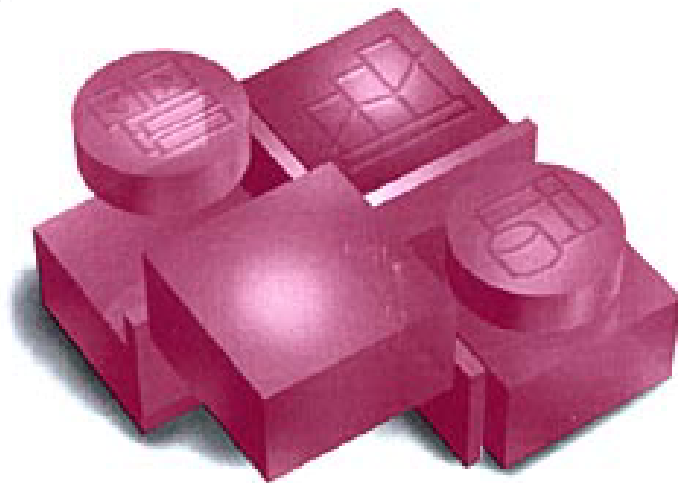


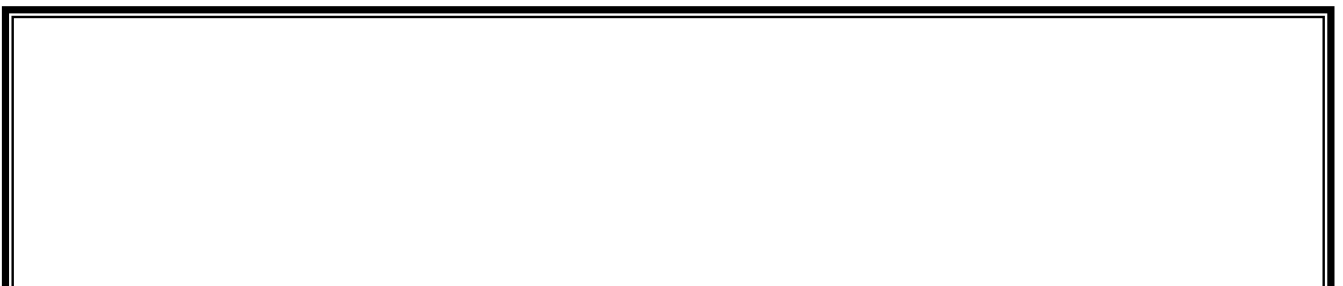
Standard Grade Computing Studies

Pupil Notes Book 1



Microsoft
Visual Basic 6.0
Professional Edition

Visual Basic



Read

Introduction

You will be using a "programming environment" called Visual Basic to plan and write computer programs.

What is a Program?

A program is a set of instructions which cause a computer to perform a useful task, such as playing a game, writing a letter, drawing a graphic, sending e-mail or managing files on a network. In fact, everything you ever do on a computer works by running a program.

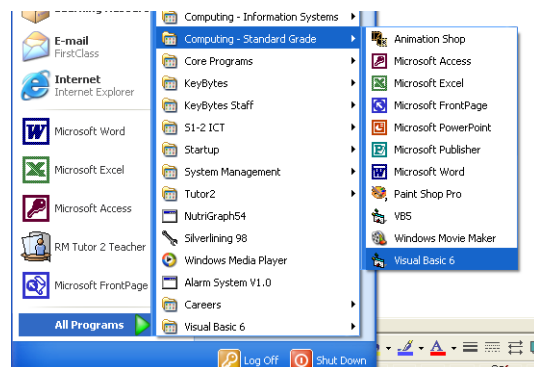
A program can be quite small—something designed to add two numbers together, or it can be a large application, such as Microsoft Word.

A Visual Basic program is a Windows-based application which you create in the '*Visual Basic Development Environment*'.

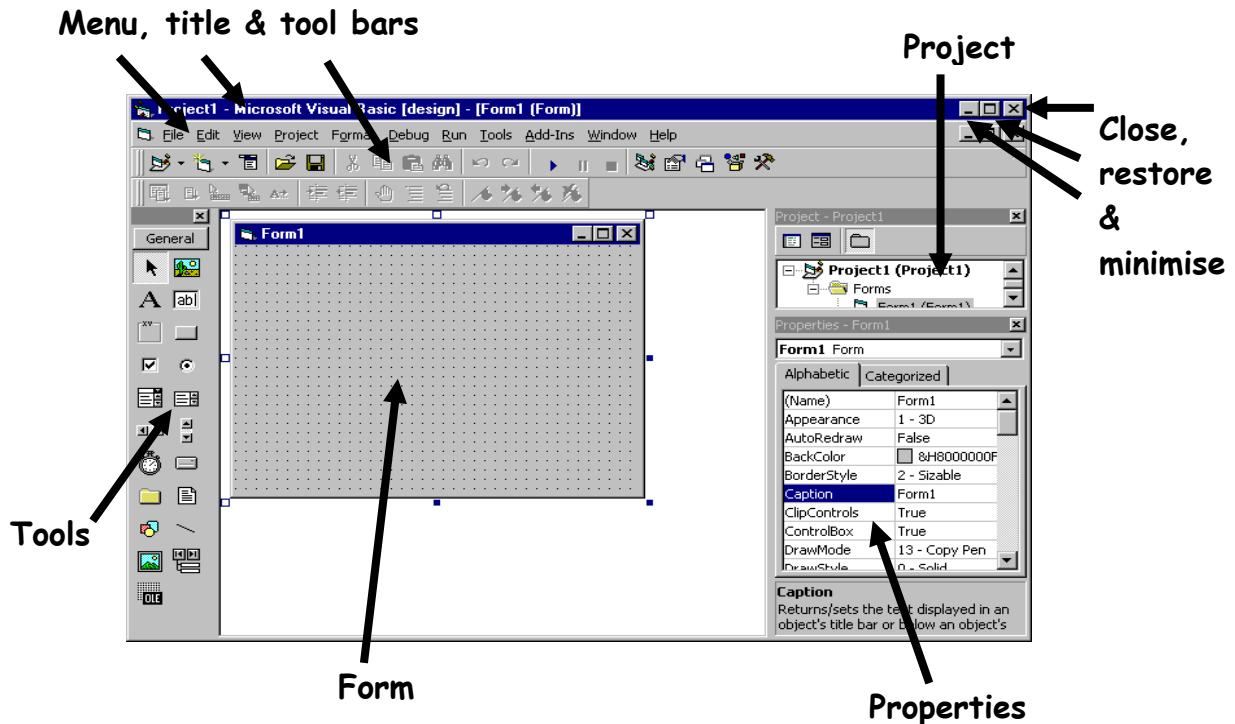
Do

Starting Visual Basic

1. Click on the **Start Button**, go to **All Programs, Computing - Standard Grade** and choose **Visual Basic 6**.



The following window should appear.



After you have read the names for each of the areas on screen, try rearranging the windows to organise the screen into a view where you can see everything clearly.

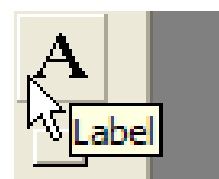
Program 1 - 'Visual Basic is Fun'

In this task you will make a program which displays the words "Visual Basic is Fun" on the screen, when you click a button.

2. Click on the **Command** button from the **tools** box.

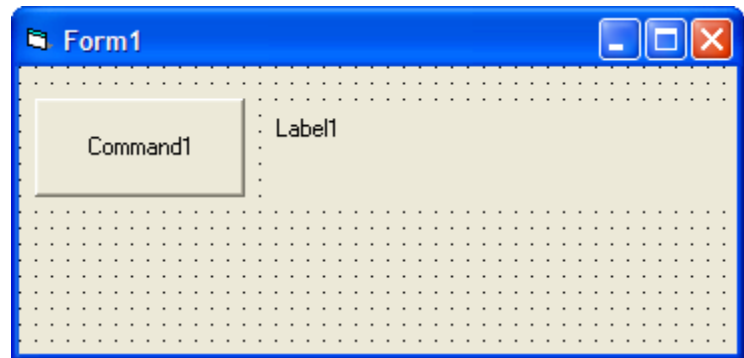


3. With the **Command** button selected click and drag a button on the form as shown below.



4. Click on the **Label** button from the **tools** box.

5. With the **Label** button selected click and drag a label on the form as shown below.



Read

Objects and Properties

You must understand that Visual Basic programs are made up of **OBJECTS**. For example, in the first program there are 3 objects:

- A Form
- A Command Button
- A Label

You add all kinds of objects onto a form to make up the **Human Computer Interface (HCI)** of the program.

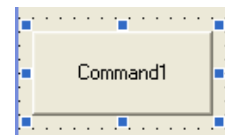
Every **OBJECT** has a number of **PROPERTIES**. For example, every object has a **NAME** property, some objects have a **CAPTION** property and many objects have a **COLOR** property. (Yes, because Visual Basic is made in America, some of the spellings are different. Don't worry, you'll get used to it!)

Properties have **VALUES**, and the value of properties can be changed when you are creating the HCI, (at design time), or they can be changed when the program is running, (at run time).

If you can understand that Visual Basic programs are made up of **objects** and that these objects have a number of **properties**, and the **values** of properties can be changed, then you will find programming with Visual Basic is easy!

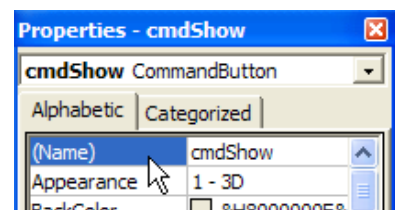
Visual Basic is called "**Event Driven**" programming. This is because you can write programming code and attach it to an event. When that event happens, the code which you have written for that event is run.

6. Click on the **command** button to select it (with handles).



7. Go to the **properties** box and change the name to **cmdShow**.

8. With the **command** button selected go to Caption and change it to **Show**.



9. Change the properties of form and label to the values shown below. Remember to select each object to before changing properties.

Object	Property	Value
Form	Name	frmFirst
	Caption	My First Program
Label	Name	lblDisplay
	Caption	Blank
	BorderStyle	1 - Fixed Single

You have now changed some properties of 3 objects at *design time*.

We are going to add some code to the *cmdShow* command button, so that when the user clicks the button (clicking is an event) our code will be run.

10. Double-click on *cmdShow*. You see the code window as shown below:

```

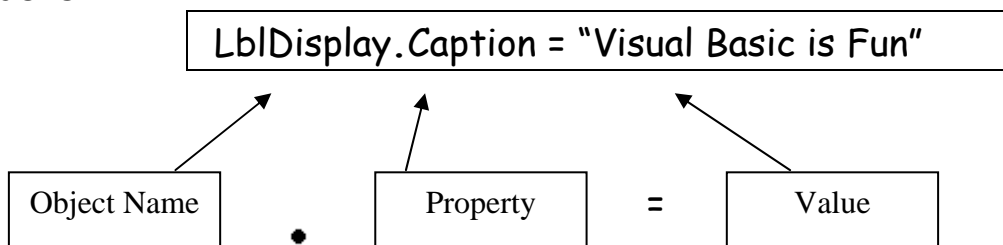
Project1 - Form1 (Code)
cmdshow Click
Rem Your Name
Rem Program Name
Rem Date
Private Sub cmdshow_Click()
lbldisplay.Caption = "Visual Basic is fun" 'This will display a message
End Sub
    
```

11. Add the code *lblDisplay.Caption = "Visual Basic is Fun"* between Sub and End Sub.

Read

'Private Sub' and 'End Sub' mark the start and end of a *subroutine*, (also called a *procedure*). You must type your code between these two lines.

Our code is:



Note the position of the full stop (or 'dot')

What this means is... That we are telling the program to change the **Caption Property** of the **lblDisplay Object** to be the **Value** "Visual Basic is Fun".

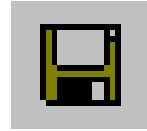
All code is written like this and it follows the pattern:

ObjectName.Property = New Value

Saving and Running Your Program

You should save your program before running it.

12. To Save, click on the **Save** button on the **toolbar**.



Unlike other programs such as Word, you will have to save more than one file in Visual Basic.

First, you have to save any forms which are part of the project and then you have to save your Visual Basic Project.

In this case, you will be asked for a name for the form,

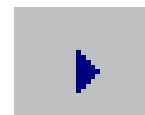
13. Save the form in the Visual Basic folder in your directory as ***First***.

Next you are asked for a name for the project.

14. Save the project in the Visual Basic folder in your directory as ***First***.

You will now have a project called ***First***, which is made up of one form which is also called ***First***. (This can be useful for keeping items together)

15. Now run your program by clicking on the **Start** button on the Visual Basic **toolbar**.



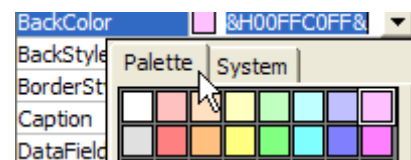
Improving Your Program

When your program runs, it places the words "Visual Basic is Fun" in the Label. If it doesn't check the code to make sure you have entered it properly.

16. Make sure your program is stopped by pressing the **stop** button from the **toolbar**.



17. Change the **Alignment** property of the lblDisplay to "Center", and the **Font** property of the lblDisplay to Comic Sans MS. You should also change the **BackColor** and **ForeColor** properties of the lblDisplay to new colours from the palette.



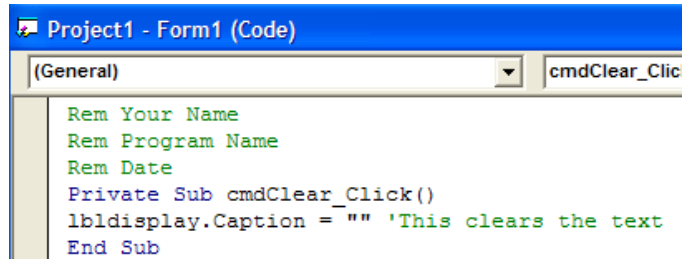
18. Now run your program again, and say "Yes" if you are asked to save any changes to the form and the project.

Further Improvements

19. Add 2 more command buttons which will **Clear** the text from the label and will allow us to **Exit** from the program. (refer to steps 2 & 3 if necessary)
20. Change the value of the properties of both buttons as shown in the table below.

Object	Property	Value
Command Button	Name	cmdClear
	Caption	Clear
Command Button	Name	cmdExit
	Caption	Exit

21. Double-click on the Clear button to open the code window for the cmdClear button and add the code `lblDisplay.Caption = ""` 'This clears the text between Sub and End Sub.



```
Project1 - Form1 (Code)
(General) cmdClear_Click
Rem Your Name
Rem Program Name
Rem Date
Private Sub cmdClear_Click()
lbldisplay.Caption = "" 'This clears the text
End Sub
```

22. Open a code window for the CmdExit button and add the code `End 'This ends the program` between Sub and End Sub.

```
Private Sub cmdExit_Click()
End 'This ends the program
End Sub
```

23. Save and Run your program again and test that all 3 buttons work.

You may have noticed that some of the code you added appears in green. This is an internal comment that is added to help future understanding of the program. All your programs should have three REM lines at the top of the program for your program name, program author (you) and program date. You should always add internal comments to your program where appropriate.

Read

Naming Objects

In Visual Basic objects should be named in a particular way. For example, you may have noticed that all command buttons used so far have started with 'cmd'. This is short for Command.

Your objects should have meaningful names. That is, they should be named in a way which tells you what the button is used for.

Good examples of command button names are:

cmdExit - To exit from your program.

cmdClear - To clear something on screen.

Other objects are named with these letters:

Forms	start with	frm
Picture boxes	start with	pic
Labels	start with	lbl
Text boxes	start with	txt
Check boxes	start with	chk
Option buttons	start with	opt
Combo boxes	start with	cbo
List boxes	start with	lst
Image boxes	start with	img

Read

Program Design

When writing programs in Visual Basic you need to learn how to plan out your form on paper before you start, showing the names of each object on the form. This is a part of your *program design*. It is very important to carefully plan what you want to happen in your program, so that you have fewer problems when writing your code.

Imagine your parents had left you at home on your own for the weekend and you decided to hold a big party when they were away! To make the party go well you would have to plan things in advance. You might make a list of things to do like this:

1. Organise music
2. Prepare Food
3. Get in some Irn Bru
4. Invite guests

These are the main steps in organising a party, but if you were really organised you could break down each of your main steps into more detail like this:

1. Organise Music
 - 1.1. Check CD Player is working
 - 1.2. Buy latest dance CD
 - 1.3. Look out best CDs
 - 1.4. Borrow CDs from friends
 2. Prepare Food
 - 2.1. Make list of food
 - 2.2. Go to shops
 - 2.3. Take pizza out of freezer
- Etc.

The idea is that if you plan things carefully before hand, things will go more smoothly. It is the same with programming. We should try to plan out our programs before we start, so that we will have fewer problems when we try to write the Visual Basic code.

Visual Basic Program Design (the following are the 4 main steps in designing a program)

1. **Problem Description** - Rewrite the problem you have been given in your own words.
2. **Screen Design** - Draw out a plan for your form on paper, showing each object and its name and complete the properties grid.
3. **Refinement** - For each command button, write out a numbered list of steps, in English, which you want to happen when you click on that button.

When you have completed these 3 stages you should then start to create your Visual Basic program on the computer by:

- **Creating the HCI**
- **Changing object properties**
- **Adding code**

4. **Evaluation** - Discuss any problems that you had and how you overcame them. Suggest improvements that could be made to make your program better. (This should be completed after implementation)

If you have done all of this carefully, you should have given yourself a greater chance of writing a program which works.

Example of program Design (Don't worry this is just an example)

Here is the program description you are given by your teacher:

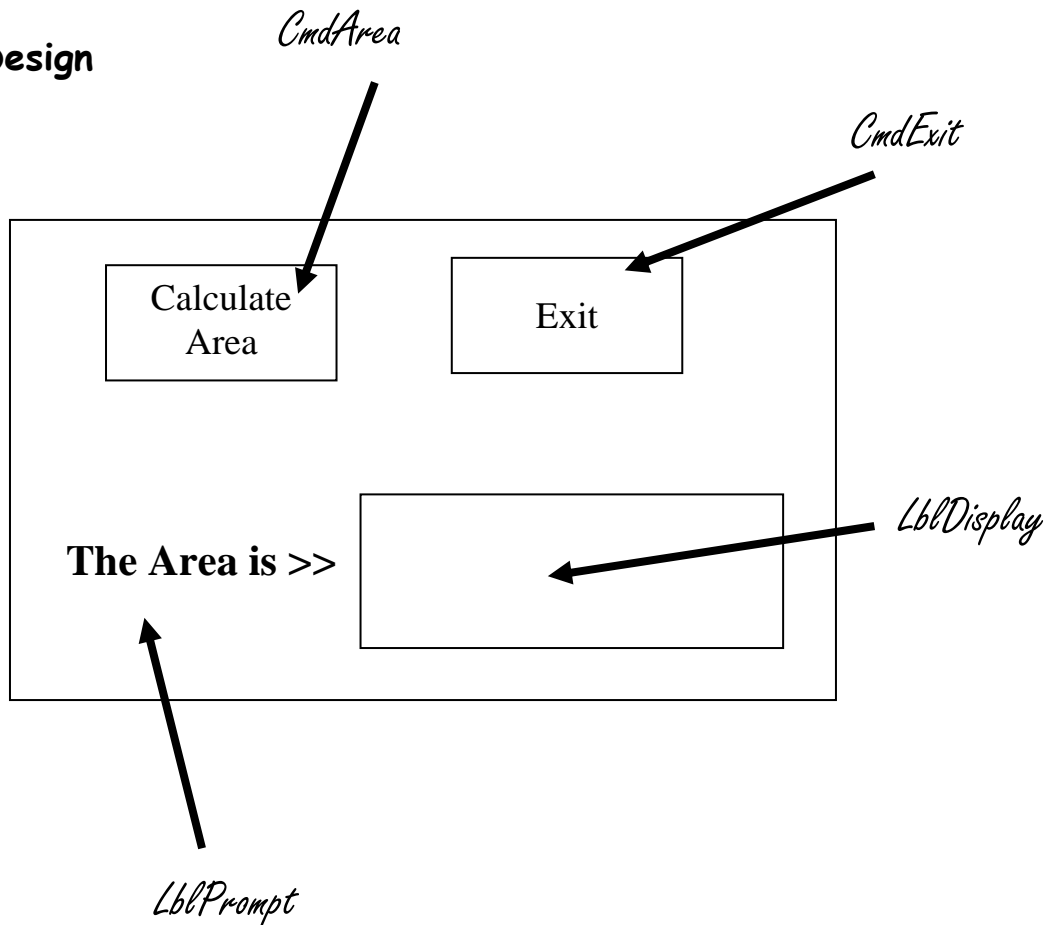
You have to design and write a program in Visual Basic which will calculate the area of a rectangle. Your program should allow the user to enter the length and the breadth, then calculate the area. Your program should also allow the user to exit from the program.

Your program design might be as follows:

Program Description

I have to write a program which will ask for the length, then ask for the breadth, then work out the area of a rectangle. My program must also have an exit button.

Screen Design



Refinement

Calculate Area Button (cmdArea)

1. Use InputBox to ask for length
2. Use InputBox to ask for breadth
3. Calculate area
4. Display area on a label (lblDisplay)

Exit Button (cmdExit)

1. Display Message Box asking if user is sure they want to exit
2. If answer is yes,
3. exit from program

Notes:

A refinement should be done for every object which has some code associated with it.

You should list the main steps in English. The steps should be numbered.

When you have written out your refinements you should find it easy to write out your Visual Basic code on the computer. The more effort you put into the refinement and the more detail you give, the easier it will be to write your code.

Once your program is working you should see a close similarity between your refinements and your finished code.

Evaluation

My program works quite well. It allows a user to enter the length and breadth of a rectangle when prompted and calculates the area. I had some problems with the code but I used the booklet and found my problem. I could improve my program by adding graphics make it look better or I could get it to display the area to two decimal places.

Do

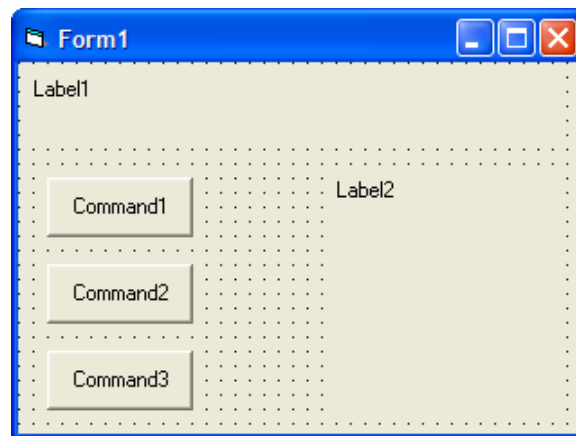
Program 2 - Doctors on Duty

Here is the design for a program that you are going to implement.

Problem (In my own words)

The local surgery requires a program to inform patients which doctors are on call. I have been asked to design a user interface for the surgery. This interface will be controlled with three buttons, one to display the names of the doctors on duty, one to remove the display and another to end the program.

Interface Design



Property Grid

Object	Property	Setting
Form	Caption	Doctors on Duty
	Name	frmDoctors
Command1	Caption	Doctors
	Name	cmdDoctors
Command2	Caption	Clear
	Name	cmdClear
Command3	Caption	End
	Name	cmdEnd
Label1	Caption	Glenbrae Surgery
	Name	lblHeading
Label2	Caption	none
	Name	lblDisplay

Algorithm

1. Set properties
2. Display the names of the doctors
3. Clear the names
4. End program

Refinements

- 2.1 Set lblDisplay caption to display Doctors on Duty and three doctors names.
- 3.1 Set lblDisplay caption to be clear.

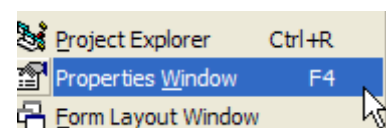
Evaluation

This problem was quite difficult. I was a bit stuck with the way the design was to be done, but I looked at the online help and found out how to do it.

If I had to make improvements to this interface, I would add some colour and place some instructions on the form to invite people to press the buttons.

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. Copy the Visual basic folder in the shared document area to your directory.
3. Open the form frmdoctors from your directory.
4. From the **View** menu choose **Properties Window** to see the properties table.
5. Change the properties for the all the objects as listed in the properties table above. Remember to select each object first.



6. Add the code to each of the command buttons as listed below by double-clicking on each button and placing the cursor between Sub and End Sub.

cmdEnd

End 'This ends the program

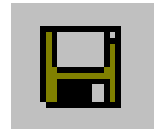
cmdDoctors

lblDisplay.caption = "Doctors on duty" & Chr(13) + Chr(13) & "Dr B Onez" & Chr(13) & "Dr S Poch" & Chr(13) & "Dr I Sore" 'this displays the names of the Doctors on duty

cmdClear

lblDisplay.caption = "" 'This clears the display

7. Save project by choosing the **Save** button from the **toolbar**.



8. Now run your program by clicking on the **Start** button on the Visual Basic **toolbar**.

If your program doesn't work go back and check that you have entered all the code as shown above.

9. Complete this project by evaluating the program as shown above.

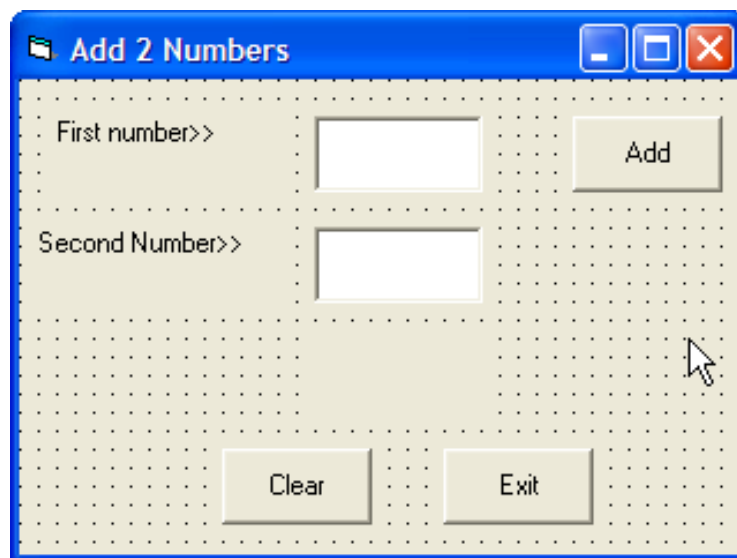
Remember to add program name, author and date to the program using REM.

Do

Program 4 - Adding Numbers

In this program you have to plan, then create, then test a program which will allow the user to add 2 numbers together, then show the answer.

1. Open the form **Add** from your directory. You should see the following form.



The properties of all the objects have been altered to the standard expected and the code the clear and exit buttons have been added.

Take a few moments to look at the way the properties for each object has been altered and look at the code in the clear button which shows how to clear the text boxes.

Read the next page carefully before adding the code to your Add button.

Read

Variables

The idea (concept) of variables is very important in programming.

A variable is a **temporary storage location** for data in your program. It is called a variable because the data it stores can change (or vary). Your code can use many variables. Variables can store words, numbers, dates, or properties. Variables are useful because they let you assign a short, easy-to-remember name to a piece of data you plan to work with. Variables can hold:

- User information entered at run time.
- The result of a calculation.
- A piece of data you want to display on your form.

If you want to use a variable in a program you have to tell Visual Basic that you want to use it. This is called **DECLARING** a variable, and is done using the special Visual Basic keyword called **DIM**. For example:

Dim number as Integer

Dim name as String

Dim age as Integer

These three statements all declare, (or set up) variables called number, name and age.

When declaring a variable you also have to declare what type of data you are going to use. **Integers** are whole numbers and **Strings** are words or text data. Integers and Strings are only two of the Visual Basic **data types**, but we will only look at these for now.

In the above examples you are declaring that you want to use a variable called number to store whole numbers, a variable called name to store words and a variable called age to store whole numbers.

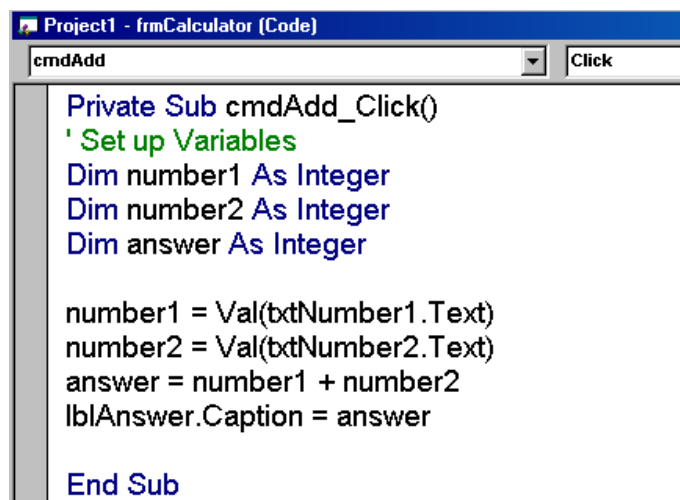
Adding 2 Numbers - Step 3, writing code.

In the code for the add button, we are going to use three variables to store data about our first number, our second number and our answer.

These variables will be called number1, number2 and answer. They will all be used to store whole numbers (Integers), so we can declare them like this:

```
Dim number1 as Integer  
Dim number2 as Integer  
Dim answer as Integer
```

2. Double-click on the **Add** button and type in the following code:



```
Project1 - frmCalculator (Code)  
cmdAdd Click  
Private Sub cmdAdd_Click()  
' Set up Variables  
Dim number1 As Integer  
Dim number2 As Integer  
Dim answer As Integer  
  
number1 = Val(txtNumber1.Text)  
number2 = Val(txtNumber2.Text)  
answer = number1 + number2  
lblAnswer.Caption = answer  
  
End Sub
```

3. Save your program by clicking on the **Save** button on the toolbar.
4. Run your program to make sure it works. If it doesn't check your code with the code above. There is a deliberate error in the program. Use the **debug** button to find where the error is and then check the names in the code against the object names in the **property window**.

Here we use the variable called number1 to store the value of what's in the text box called txtNumber1.text. The same is done with the variable called number2, and the variable called answer is used to store the result of the calculation where number1 is added to number2.

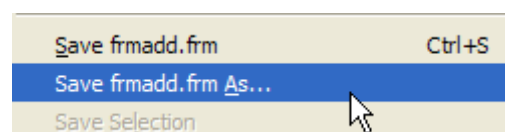
Finally we changed the caption property of lblAnswer to equal the value of the answer variable.

Do

Program 5 - Subtract Two Numbers

When programmers develop programs they often take existing programs or procedures of a similar nature and alter them to suit. We are going to take the program **Add** and alter it to subtract two numbers.

1. Open the form **Add** from your directory. Make sure you can see the **properties** window and **project explorer** window.
2. From the **File** menu choose **save project as** and save your program as *prjcttakeaway*.
3. From the **File** menu choose **save frmadd.frm As** and save your form as *frmtakeaway*.



4. Change the properties as shown below.

Object	Property	Value
Command Button Add	Name	cmdTakeaway
	Caption	Takeaway

Next we have to change the code to subtract rather than add the two numbers together.

5. Open the **Takeaway** button.

6. Change the code as shown below.

```
Private Sub cmdAdd_Click()  
    'Set up Variables  
    Dim Number1 As Integer  
    Dim Number2 As Integer  
    Dim Answer As Integer  
  
    Number1 = Val(txtNumber1.Text)  
    Number2 = Val(txtNumber2.Text)  
    Answer = Number1 - Number2  
    lblAnswer.Caption = Answer  
  
End Sub
```

All that is changed is the operator, from + to -

7. Save your program by clicking on the **Save** button on the toolbar.

8. Run your program to make sure it works. If it doesn't check your code with the code above.

Do

Program 6 - Product of Two Numbers

A shopkeeper has seen a copy of your add program and has asked you to develop a program that will allow him to enter the price of the item and the number of units sold. You have to design a user interface that allows him to enter these two numbers. The program should then multiply the two numbers and display the answer.

Note: Two of the variables (Price and Total Cost will be type Currency.)

Problem (in my own words)

Design Grid

Properties table

Algorithm

Refinements

Evaluation

Complete the above program, begin with the design then implement and once complete evaluate your program.

Copy the Food Mixers assessment from the PrepWork folder into your own folder.

PRACTICAL ABILITIES TASK

FOOD MIXERS

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.

Instead of being paid an hourly rate, some workers are paid for the number of items they make each week. Ann Hay assembles 86 food mixers and is paid £3.50 for each one. You are required to write a program to calculate and display her earnings. The user should be asked to enter the number of mixers and payment for each mixer. You should use meaningful variable names such as *mixers*.

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

2 marks

The main steps are *Calculate Wage*, *Get Inputs*, and *Show Wage*.

Write the steps in the correct order below.

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

4 marks

Work out Ann's wage for assembling 86 mixers at £3.50 each. You may use a calculator.

What was your answer? £_____

Run the program to find its answer for 86 mixers at £3.50 each.

What was the program's answer? £_____

Work out Ann's wage for assembling 90 mixers at £4.10 each.
You may use a calculator.

What was your answer? £_____

Run the program to find its answer for 90 mixers at £4.10 each.

What was the program's answer? £_____

Compare your answers with the program's answer and write a comment.

5 GET PRINTOUTS - Implementation

2 marks

Get **two** printouts with your name in the footer. One should show your listing and the other a run using one set of test data given above.

6 SUGGEST IMPROVEMENT - Evaluation

2 marks

Write down **one** way in which your program could be better.

7 JUDGE PERFORMANCE - Evaluation

2 marks

Read this task sheet again. Describe how well you think you have done the task.

Grade 5 - 15 or more

Grade 6 - 14 to 11

Grade 7 - 10 or less

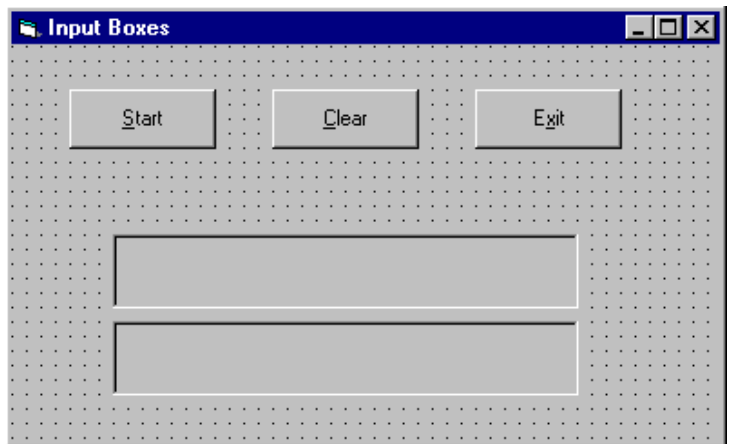
Do

Program 7 - Input Boxes

In previous programs you have used text boxes to allow you to Input information into a program.

This example uses a special function called an *Inputbox* which also lets you enter information while your program is running.

1. Open the form **Input Boxes** from the VB folder.



It has the following properties:

Object	Property	Value
Form	Name	frmInputbox
Command Button	Name	cmdStart
	Caption	&Start
Command Button	Name	cmdClear
	Caption	&Clear
Command Button	Name	cmdExit
	Caption	E&xit
Label	Name	LblMessage1
	Caption	blank
Label	Name	LblMessage2
	Caption	blank

Read

Input Boxes

Inputbox is a special code word in Visual Basic. It is used like this:

```
Variable_Name = Inputbox("Ask for input", "Title for box")
```

Eg.

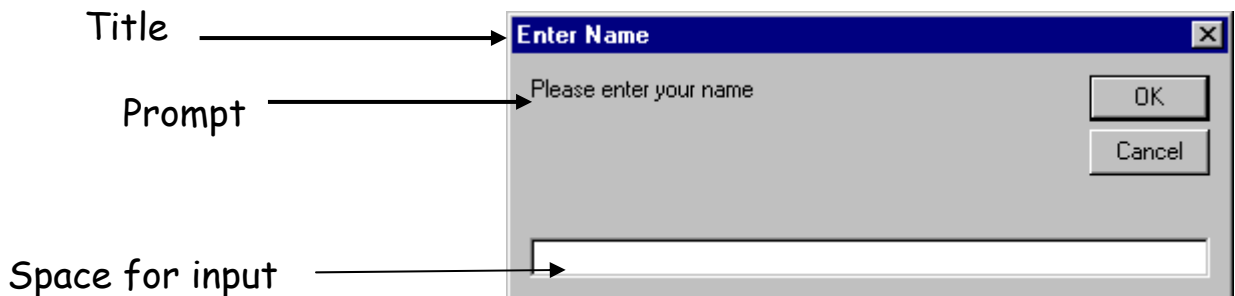
```
Myname = Inputbox("Please enter your name", "Enter Name")
```

Where *Myname* is a variable which we have already declared.

"*Please enter your name*" is a prompt which asks the user to do something.

"*Enter Name*" will be the title of the Inputbox.

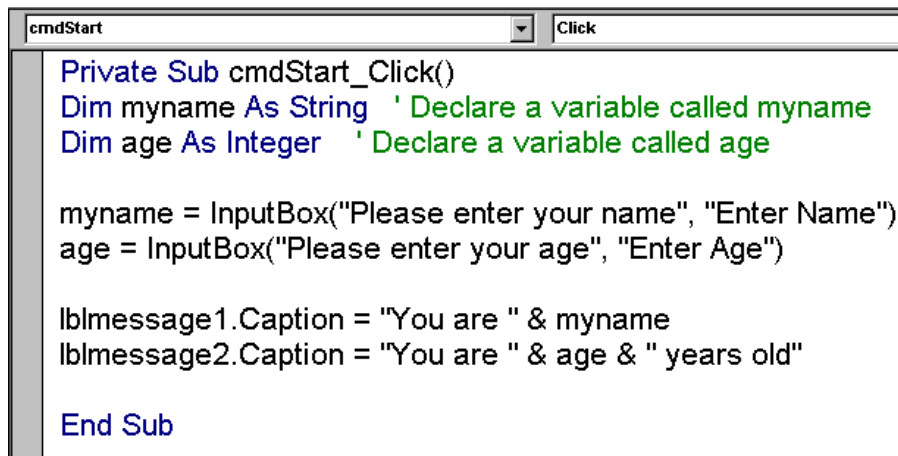
The above example will produce this:



When you run the program, what you type into the Inputbox will become the *value* of the variable at the start of the line.

In this example, if you type ***Susan*** into the Inputbox, the value of the variable called ***myname*** will be ***Susan***.

2. This code can be added to the **Start** button. (Double-click on the button to open the code window)



```
cmdStart Click
Private Sub cmdStart_Click()
Dim myname As String ' Declare a variable called myname
Dim age As Integer ' Declare a variable called age

myname = InputBox("Please enter your name", "Enter Name")
age = InputBox("Please enter your age", "Enter Age")

lblmessage1.Caption = "You are " & myname
lblmessage2.Caption = "You are " & age & " years old"

End Sub
```

3. **Save** and **Run** your program and test all three buttons to see if they work.

Note the captions on the buttons. To do this the property captions were written like this: &Start; &Clear; E&xit.

The & symbol puts the underline under the next letter and allows you to control the command button from the keyboard by holding down the Alt key, and pressing the underlined letter. For example Alt + X will end the program.

4. **Run** your program again and this time instead of clicking on the **Start** button hold down the **Alt** key on the keyboard and press the **S** key.
5. Repeat step 4 and use these keyboard shortcuts to clear the display and exit from the program.

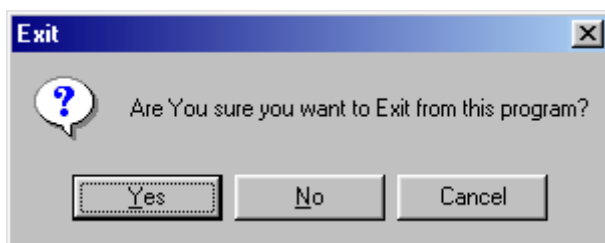
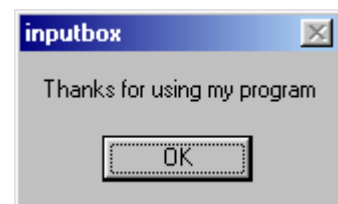
Read

Message Boxes

A message box displays a message in a dialog box, and then waits for the user to click a button before going on with the program.

A simple message box will only have an OK button, but more complicated message boxes can have several buttons and an icon to give the user information.

A simple message box:



A more complex message box

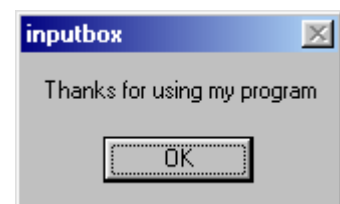
Adding a simple message box is easy. For example when a user clicks on the exit button of your program, you can get a simple message box to appear as follows:

```
Private Sub cmdExit_Click()

MsgBox "Thanks for using my program"
End

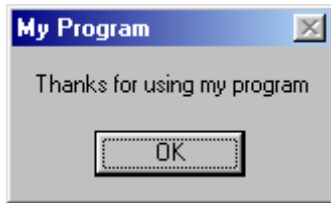
End Sub
```

In this example a message box will appear containing the message "Thanks for using my program" and an OK button is automatically added to the message box.



The next improvement on this is to say:

Msgbox "Thanks for using my program",,"My Program"



This gives your message box a title of "My Program", but note the 2 commas between the message part and the title part.

More Complex Message Boxes.

A more complex message box displays a message in a dialogue box and waits for the user to choose a button. It then returns an integer value indicating which button the user has chosen.

A complex message box can contain a wide range of buttons and a range of icons. To add more buttons and icons you have to do a simple sum.

Number	Buttons Displayed
0	Display OK button only.
1	Display OK and Cancel buttons.
2	Display Abort, Retry, and Ignore buttons.
3	Display Yes, No, and Cancel buttons.
4	Display Yes and No buttons.
5	Display Retry and Cancel buttons.

Number	Icon Displayed
16	Stop Icon
32	Question Mark Icon
48	Exclamation Icon
64	Information Icon

E.g. If you want to have a message box with an OK and Cancel Buttons, and an exclamation Icon, you add together 1 + 48, which is 49.

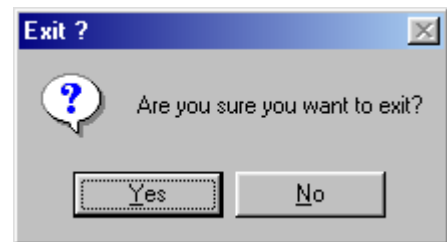
E.g. If you want to have a message box with an Yes, No and Cancel Buttons, and an information Icon, you add together 3 + 64, which is 67.

Many combinations are possible.

Add this number to the new MsgBox function as follows:

MsgBox "Are you sure you want to exit?",36,"Exit ?"

This will give you a message box with a title of "Exit ?", a message of "Are you sure you want to exit?", a question mark icon (number 32) and 2 buttons - Yes and No. (number 4)



This is not the end of the story however! You now need to know which button the user has clicked on. For example if there are Yes and No Buttons, the program might do different things depending on which button was clicked.

To find out what button was clicked, our new line is:

```
response = MsgBox("Are you sure you want to exit?",36,"Exit ?")
```

Which ever button is pressed has a numeric value. That number will be passed to the variable '*response*' in the example above. Here is a list of the values of the buttons:

Number	Meaning
1	OK button selected.
2	Cancel button selected.
3	Abort button selected.
4	Retry button selected.
5	Ignore button selected.
6	Yes button selected.
7	No button selected.

Your new code attached to an exit button might now be :

```
Private Sub cmdExit_Click()
```

```
Dim response As Integer ' Declare a variable called  
response
```

```
response = MsgBox("Are you sure you want to exit?", 36,  
"Exit ?")
```

```
If response = 6 Then 'If the Yes button is clicked  
End  
End If
```

```
End Sub
```

In this example there will be a message box with 2 buttons - Yes and No. If the user clicks on Yes, then 'response' will equal 6. If the user clicks on No, then 'response' will equal 7.

In the second part of the code, If response is 6 then the 'end' code will be run, if it is 7 then nothing will happen.

Message boxes are easy to put into your program and they can be used to give users information or to ask them to make a choice.

Program 8 - Message Boxes

1. Open the form **Input boxes** from your Visual Basic folder.

We are going to add a simple message box to the **Exit** button.

2. Amend the code in the **Exit** button as shown below.

```
Private Sub cmdE&xit_Click()
```

```
Msgbox "Thanks for using Input Boxes",, "Input Boxes"
```

```
End
```

```
EndSub
```

Read

For...Next Loops

You can use a loop to do something over and over again. A *For...Next loop* uses the Visual Basic keywords '*For*' and '*Next*' to repeat a section of your code a fixed number of times. For Example:

- (1) **Dim loops as integer** ' loops is called the loop variable.
- (2) **For loops = 1 to 4** ' This marks the start of the loop.
- (3) **PicDisplay.Print "Hello"** ' This code is repeated 4 times
- (4) **Next** ' This marks the end of the loop.

In the above example, the variable called '**loops**' (NOTE - You can't use a variable called 'loop') is given the value of 1 the first time the '**For**' line is done, then the value of **loops** is increased by 1 every time the **Next** line is done. The program ends when the value of **loops** goes past 4.

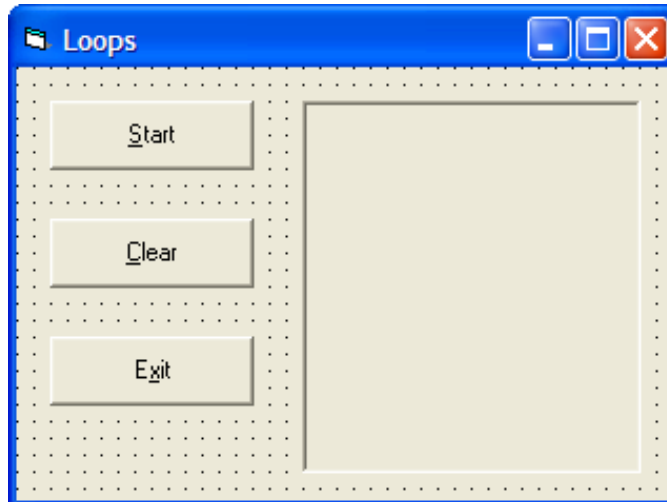
The sequence of the code in the above example is as follows:

Order	Value of loops
The line marked (1) is done first	Loops = 0
Then (2)	Loops = 1
Then (3)	Loops = 1
Then (4)	Loops = 2
Then back to (3)	Loops = 2
Then (4)	Loops = 3
Then back to (3)	Loops = 3
Then (4)	Loops = 4
Then back to (3)	Loops = 4
Then finally (4)	Loops will = 5 when the program stops

Do

Program 9 - Loopy message

1. Open the form **Loops** from the Visual Basic folder. (shown below)



2. Enter the code shown below in the **Start** button.

```
Private Sub cmdStart_Click()  
Dim loops As Integer, band As String  
band = InputBox("Please enter the name of your favourite band", "Favourite Band")  
For loops = 1 To 5  
picDisplay.Print "My favourite pop group is " & band  
Next  
End Sub
```

Picture Boxes

If you put a Picture Box on your form and call it something like PicDisplay, you can print text in it, using code like this:

```
PicDisplay.Print "My Name Michael Cane"
```

To clear a Picture Box, use the code **PicDisplay.Cls**

3. Add a suitable message box to the **Exit** button.
4. **Save** and **Run** your program making sure all the buttons work.

Do

Program 10 - Ten Number Total

Here is the design for a program that you are going to implement.

Problem

Produce a program that will make use of the InputBox function and a FOR . . . NEXT loop to get 10 numbers from the user, add them together and display the total.

Note:

Your program will need two variables declared at the start:

Number: used to store the numbers

Total: used to keep a running total

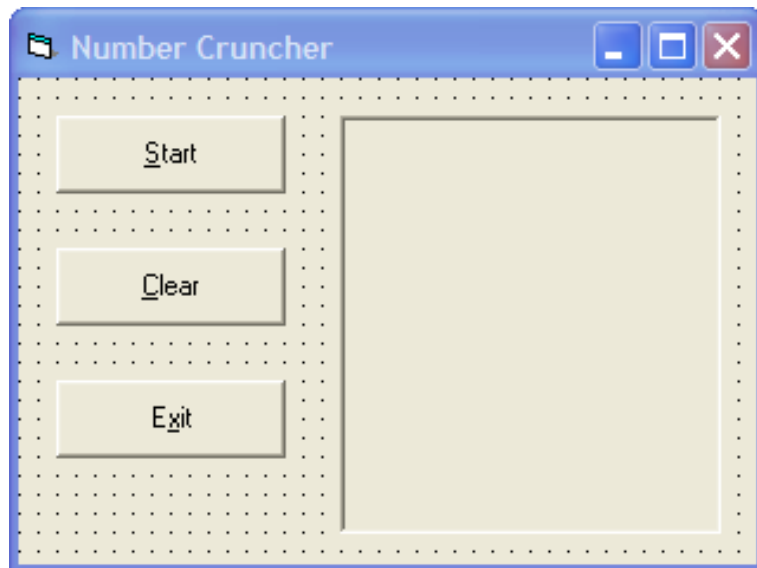
To keep a running total of the numbers entered by the user, your program will need the following line of code such as this:

Total = Total + Number

Problem (In my own words)

I have to design a program that will ask the user to enter 10 numbers. The program will add each number enter and display the new total after each number is entered.

Interface Design



Property Grid

Object	Property	Value
Form	Name	frmNumberCruncher
	Caption	Number Cruncher
Command Button	Name	cmdStart
	Caption	&Start
Command Button	Name	cmdClear
	Caption	&Clear
Command Button	Name	cmdExit
	Caption	E&xit
Picture Box	Name	picDisplay
	Caption	blank

Algorithms for Each Button (Procedure)

Start

1. Set up variables
2. Repeat 10 times
3. Prompt user for number
4. Calculate the running total
5. Display total

Clear

1. Clear display

End

1. End program

Refinements

3.1 Display input box prompting user to enter a number

3.1 Assign user number to variable number

4.1 Value of variable total equals variable total and variable number

5.1 Display message and value of variable total

Evaluation

I found this program easy as I had completed a program that was similar. All I had to do was take this program and change it to prompt the user for 10 numbers using a loop. If I had to improve my program I would allow the user to enter decimal numbers 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.

Read

Improvements to FOR ... NEXT Loops

Changing the number of times the loop is done.

For times = 1 to 10 - This will do a loop 10 times.

If you want to change the number of times the loop is done then you change the number 10 to a different number.

Here's the clever bit - Why not use a variable to change the number of times the loop is done?

Like this:

Dim number, times as Integer

Number = InputBox("Enter the number of times you want to do the loop")

For times = 1 to number

PicDisplay.Print "Visual Basic loops are easy"

Next

Copy the Car Stickers assessment from the PrepWork folder into your own folder.

PRACTICAL ABILITIES TASK

CAR STICKERS

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.

You are required to write a program for a firm that makes stickers for car windows. The user should input the sticker text at the keyboard. The program will display that text four times as shown right. There should be a blank line after each line of text. A FOR ... NEXT loop should be used.

YOU ARE DRIVING TOO CLOSE!

YOU ARE DRIVING TOO CLOSE!

YOU ARE DRIVING TOO CLOSE!

YOU ARE DRIVING TOO CLOSE!

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

2 marks

Write down the main steps in the program

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

3 marks

Describe how you tested that your program worked.

5 GET PRINTOUTS - Implementation

2 marks

Get **two** printouts with your name in the footer. One should show your listing and the other a run.

6 WRITE INSTRUCTIONS - Documentation

3 marks

Write a short explanation on how to run and use your program after it has been loaded.

7 SUGGEST IMPROVEMENT - Evaluation

2 marks

Write down **one** way in which your program could be better.

8 JUDGE PERFORMANCE - Evaluation

2 marks

Read this task sheet again. Describe how well you think you have done the task.

Grade 5 - 16 or more

Grade 6 - 15 to 12

Grade 7 - 11 or less