How to Be a Programmer Programming Basics

Mohan Palleti

How to Be a Programmer

Programming Basics

Mohan Palleti

ISBN-10: 1490915753 ISBN-13: 978-1490915753 FOREWORD

The goal of this book is to be a resource for instructors educating students or individuals that want to learn computer programming. Computer programming is a fun and rewarding skill that teaches critical and logical thinking. Programming can be taught to any person with basic arithmetic and logical skills. With the basic programming skills learned from this book, students will be able to advance to a higher level of programming on their own.

.

CONTENTS

v

ACKNOWLEDGMENTS

I dedicate this book to my endearing wife, Rajani and family for their constant support. And also! Can’t forget my dog, Bud.

.

i

1 INTRODUCTION

Are you among one of those who do not know the ABC’s of programming, and yet at one point in your life, you felt a great desire to learn software programming, but did not know where to begin?

Do not get disheartened! You are not alone! Most people think you need to get a degree in

Computer science engineering to be a programmer, and that programming requires an extensive mathematics background.

The truth, however, is very different. Any type of formal education would be helpful, but as a newbie, you do not have to know computer programming or advanced mathematics. You can learn the basics from this book and then migrate to other programming languages by whichever avenue you choose.

John Holt is a friend of mine who I met on the subway in

New York City many years ago when I used to live there. The morning we met, I was in the subway going to work carrying a heavy programming language manual under my arms.

This is the hallmark of a programmer. He/she doesn't read best sellers or magazines in his/her free time. Instead a programmer reads programming language books during his/her free time. His mind is always in a loop. He can never leave his script behind at the office; always thinking through endless mazes; how can I make the application work? Am I missing something?

I am warning you this is how you might end up in life. But most of all I am warning you to keep an eye on these people. They could bump into you if you are not watching.

Coming back to John, "h e was looking at me and my book and was hoping to catch my attention to start a conversation. He finally managed to get my attention when he said "man, that is a big fatbook you got there" he said. I smiled and said, "yes I use it to do my arm-presses." We shared a laugh.

John continued to say, "You know, if I could do something like that in my office, it would make me a hero at my work place. As a matter of fact, there is this guy in my office who knows so much about these computer things that everyone knows him at work. He is cool and handsome too."

That made me think- if I can write a simple book that would help everyone new to programming, be able to program without having to download hundreds of different software plug-ins and upgrades, I would be able to help many a John Holt become workplace heroes and come into the mainstream of programming.

What do you need to learn this course? A desktop or a laptop computer, a copy of Microsoft Excel, one good finger to poke at the keyboard, and time on your hands!

Do you need to know advanced mathematics? No you do not! You just need the basics! The most important skill necessary to be a programmer is to understand logic. If you know that, then you are good to go!

2 BASIC HOUSEKEEPING RULES

Understanding Data Types

Integer numbers

Any number that is a whole number (means a number without a decimal part to it) is called an integer. For example 9,23, 54, 328 etc.. These are all integers.

Real or ***floating point*** numbers

Any number that has a decimal part to it is called a ‘Real’ or ‘floating point’ number. For example: 2.34, 42.389, 197823.34578, etc. are all real or floating point numbers.

Variables
A variable is a string of characters you choose to store a value (a numerical number or characters).

Let us say you name a variable called ‘myFirstNumber ,’ you can then assign a number to it, for example 300.7567. Let us say you create a second variable called ‘mySecondNumber’ and you assign the value 200 to it.

We now want to add the two variables and store the result to anothervariable called ‘my Result’ in the next step.

Addition:
How does the programmer write this in code? Here are 4 lines of code that should do it.

1. myFirstNumber = 300.7567
2. mySecondNumber = 200
3. myResult = myFirstNumber + mySecondNumber
4. Print myResult

Happening at the computer level?

1. The program looks at the first line. Then looks at the left of the ‘=’ sign and creates a blackbox with the label ‘myFirstNumber’. Then it looks at the right of the ‘=’ sign; picks the numerical value of 300.767; stores it inside the box with the label ‘myFirstNumber’. In technical terms we say that a value of 300.767 is assigned to the variable on the left side. In our case we named it as ‘myFirstNumber’

2. The program now looks at the second line and similarly creates a variable ‘mySecondNumber’ and assigns a value of 200 to it.

3. The program now reads the third line of code and creates a variable called ‘myResult’ adds the values stored in the variables ‘myFirstNumber’ and ‘mySecondNumber’ and stores the result in the variable ‘myResult’

4. Line 3 of code prints the value of ‘myResult’ which is 500.767
That was easy,wasn’t it?

Before you create a variable, you should define its **properties**. How large are your numbers in the data set? Accordingly we will allocate memory size for the variables.

Numbers with decimal fractional parts are called Floating points and should be stored as either **Single** or **Double**.

The **Single** data type requires 4 bytes of memory and can store negative values between -3.402823 x 1038 and 1.401298 x 10-45 and positive values between 1.401298 x 10-45 and 3.402823 x 1038.

The **Double** data type requires 8 bytes of memory and can store negative values between -1.79769313486232 x 10308 and -4.94065645841247 x 10-324 and positive values between 4.94065645841247 x 10-324 and
1.79769313486232 x 10308.

In our example we have 2 numbers. The highest number is 300.767, which is a real number and can be stored as type ‘Single’. The second number is 200. We can either save it as a Byte or as an integer.

As for the variable ’myresult,’ it is up to us to decide how the results are going to be stored. If you only want the integer part and get rid of the decimal fractions, then create the variable ‘myresult, ’ as an Integer. If you want the result to store the complete number with the decimals, then you should store the variable as ‘Single.’ Now we will rewrite the program with some very slight changes.

```
1. Dim myFirstNumber as single
2. Dim mySecondNumber as int
3. Dim myResult as int
4. myFirstNumber = 300.7567
5. mySecondNumber = 200
6. myResult = myFirstNumber + mySecondNumber
7. Print myResult
```

The word Dim before a variable name indicates that you are declaring the dimension type of the variable.

Line 1 indicates that you have created a variable called ‘myFirstNumber’ and you have declared it as type ’Single.’

Remember the Single data type requires 4 bytes of memory and can store negative values between -3.402823 x 1038 and -1.401298 x 10-45 and positive values between 1.401298 x 10-45 and 3.402823 x 1038.

For now we will store the value 300.7567 in this variable (as per code in line 4)
The second line of code states that you have created a variable called ‘mySecondNumber’ as int.

‘ **int**’ is short for integer and that is how the VBA compiler understands it. Integer Data Type can hold values between
-2147483648 through 2147483647. A **Byte** can store values between 0 and 255.

Now you can store values in the range -32,768 and 32,767 in this variable . We will store the value 200 in it (as per code in line 5)

Because myResult has dimension type ‘int,’ the result will print only the integer value. In our case ‘500’
Question: If myResult was declared as type ‘Single’ then what would have been my result?
Answer: 500.7567
Subtraction

The symbol for Subtraction is ‘ -’. This is the same as the symbol we learned in school. Look at the Example code below:

```
1. Dim myFirstNumber as single
2. Dim mySecondNumber as int
3. Dim myResult as int
```

4. myFirstNumber = 300
5. mySecondNumber = 200
6. myResult = myFirstNumber - mySecondNumber
7. Print myResult

Question: What is the value stored in myResult? **Answer:** 100
Multiplication:

The symbol for multiplication in programming is '*****'. Instead of the multiplication symbol 'X' that we learned in school, we should use '*****' between the two operands.
1. Dim myFirstNumber as single
2. Dim mySecondNumber as int
3. Dim myResult as int
4. myFirstNumber = 300
5. mySecondNumber = 200
6. myResult = myFirstNumber * mySecondNumber 7. Print myResult

Question: What is the value stored in myResult?
Answer: 60000
Division:

The symbol for division is ' /'. Instead of the division symbol '÷' that we learnedin school, we should use '/' between the two operands or variables.

The '/' is called the operator and the numbers on either side of it are called the operands. In our code example below, we do not use the numbers (300 and 20) directly for the division. Instead we will store them in two different variables (myFirstNumber and mySecondNumber).

1. Dim myFirstNumber as single
2. Dim mySecondNumber as int
3. Dim myResult as int
4. myFirstNumber = 300
5. mySecondNumber = 20
6. myResult = myFirstNumber / mySecondNumber 7. Print myResult

Question: What is the value stored in myResult? **Answer:** 15
So far we have learned how to play with numbers, to store and manipulate them.
What if we wanted to store some texts in the variables instead of numbers?

For example we want to design a Graphical User Interface (GUI), where a user enters his first name and last name in two different text boxes. You want to grab them and construct a personalized greeting. That is easy!

First let us learn how to store texts in VBA. A text is stored in a variable as a 'string.'

1. Dim Fname, Lname, myGreeting as string
2.Fname = 'John'
3.Lname = 'Holt'
4. myGreeting= " Hello Friendly Neighbor " & Fname & " " & Lname
Question: What is stored in the variable 'myGreeting?'

Answer: Hello Friendly Neighbor John Holt
I went ahead and wrote the code. Don't panic - I am going to explain every line.

The first line created three variables **Fname**, **Lname** and **myGreeting** all followed by' ,' (we can choose any name for our variables). They are all of **string** type**.** A string type variable can store about 65535 or so characters. Most people never use it to store more than 100 characters.

The second line of code put a string of characters that read '**John**' in the variable **Fname.**
Line 3 assigns the character string'**Holt**' to the variable **Lname**
Line 4 takes a bunch ofstring's and joins them using an '&' operator.

Let us see what strings we are joining. Some of the strings in line 4 have to be retrieved from the variables where they are stored, while others like "**Hello Friendly Neighbor**" and **" " are** not stored in any variables.

As a result **myGreeting** now stores the joined strings**"Hello Friendly Neighbor John Holt**"

Concatenate
Joining multiple strings into one string.

For example "**Mr.**" & " **John**" & " **Holt**" when joined together become"**Mr. John Holt.**" Notice that I had deliberately provided a single space before and after John in the quotes to make it readable. If the space was not provided it would have read "**MrJohnHolt.**"

String manipulation

There are several functions available in VBA to manipulate strings. We will pick a few of them and describe the formats of how they are to be used and then we will use them in an example.

Mid function
(*Carves out a new subset of string from a bigger string***)** Usage: MID(text, start_position, number_of_characters)

1. Dim Aword, NewWord as string
2.Aword = "The night wolf is on the prowl"
3. NewWord = Mid (Aword,11,4)

Question: What does the variable NewWord contain? **Answer**: 'wolf'
In line 1we created two variables 'Aword' and 'NewWord' as type strings.

In line 2 we stored a line of character strings, "The night wolf is on the prowl" in the variable '**Aword**' Aword now has 30 characters stored in it. Character #1 is 'T' Character #2 is 'h' and so on.

In line3 a new set of string is carved out from the contents stored in the variable '**Aword**' The new string reads 4 characters starting from the 11^{th} character or position as shown above in highlights. A blank space is also counted as a character.

The four charactershere represent the word 'wolf'.

LEN Function

(*Counts the total number of characters in a string***)** Usage: LEN (Text)

4. Dim vlength as int
5. vlength = Len (Aword)
Question: What does **wlength** contain? **Answer**: 30

6. The function picks up the string from the variable inside the parenthesis; calculates the length of the entire string; transfers the value to the variable on the left side of the '=' sign. In our case Aword has the string "**The night wolf is on the prowl**" stored or assigned to it. The entire length of this string is 30 (including the spaces between the quotes).

Right Function
(*Carves out the last few characters of a given string the length of which is defined by the user***)**
Last 3 characters
Usage: RIGHT (Text, [number of characters]) 7. Dim word1 as string
8. word1 = Right (Aword,3)
Question: What does the variable word1 contain? **Answer**:'owl'
The function picks up the last 3 characters of the string
"**The night wolf is on the prowl**" and assigns it to the variable'word1'.

Left function
Usage: Left (Text, [number of characters])

(*Carves out the first few characters of a given string the length of which is defined by the user***)**
9. Dim word2 as string
10. Word2 = left(Aword,9)

Question : What does the variable Word2 contain? **Answer**: 'The night'

The function picks up the first 9 characters of the string " **The night wolf is on the prowl**" and assigns it to the variable 'word2'.
3 MS EXCEL

Let us do an exercise to make my points clear. String manipulations using Excel:

1. Open up an Excel page. You see an empty table with columns denoted by A, B, C …. and row numbers on the left starting with 1.

2. Write the text "The night wolf is on the prowl" in the cell A1 without the quotes. A cell is always identified by its Column ID and Row number. Cell A1 Means the cell at Column A and Row 1.

3. Expand the cell so that we can see the entire text . This is done by putting the cursor between the Header cells A & B and double clicking it as in Fig 1.

4. Click on cell D1 and enter this formula on the function bar = mid(A1,10,5) as in Fig 1.
5.See how it extracts the word 'wolf' from the text stored in the A1 cell and puts it into the

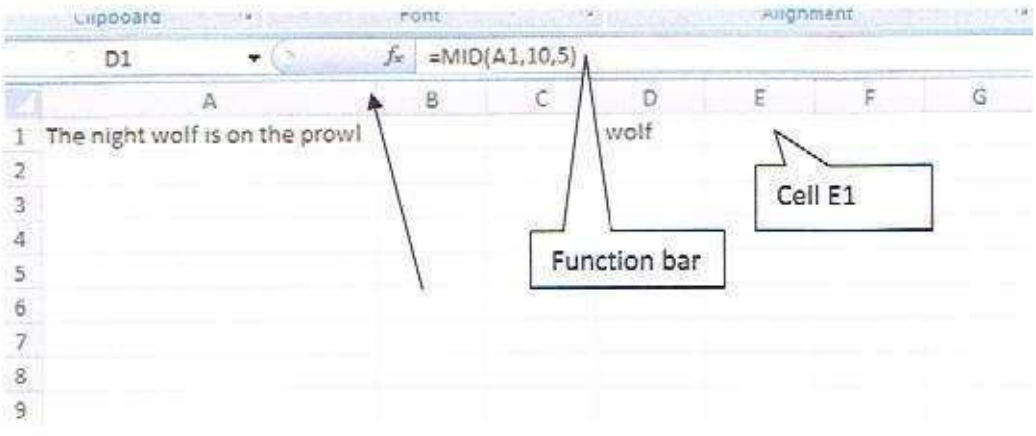


Fig 1

cell D1 as in (Fig 1).

6. Click on cell E1 and enter this formula on the function bar = len(A1)
7. See how the total length of the string in Cell A1 is calculated and provided at cell E1 as in Fig2.

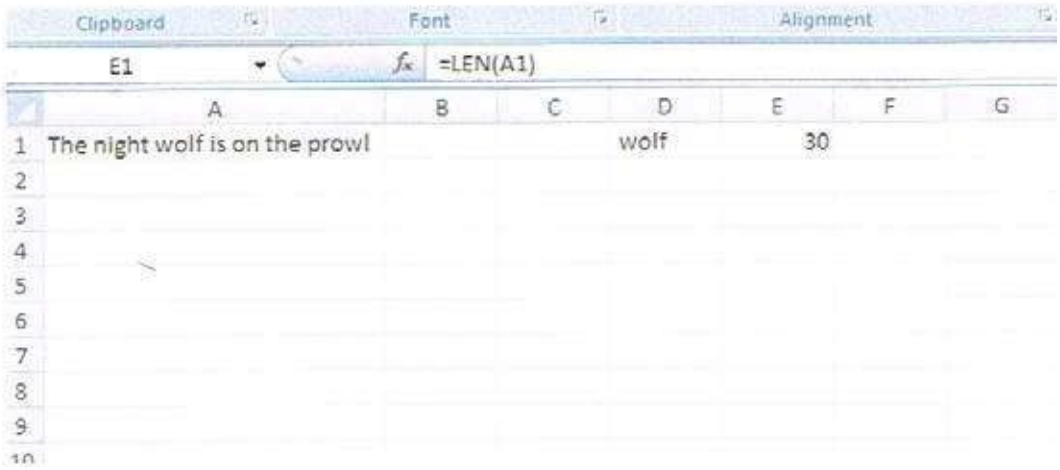


Fig 2

8. Click on cell F1 and enter this formula on the function bar = Right(A1,3)
9. Click on cell F1 and enter this formula on the function bar = Right(A1,3)
10. See how it picked up the last three characters on the extreme right and wrote it in Cell F1 as in Fig 3.

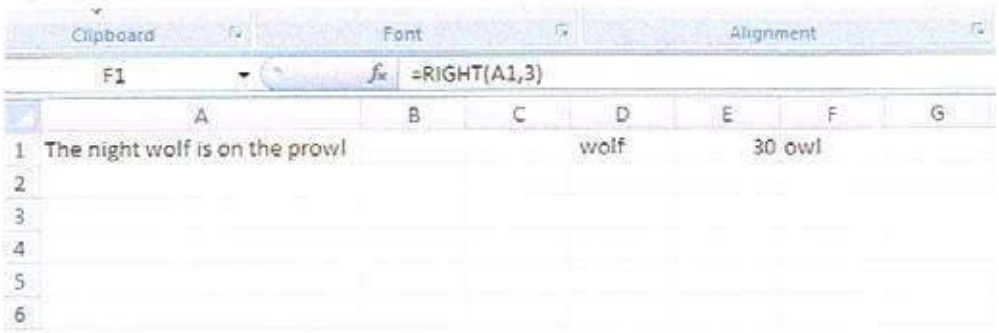


Fig 3

11. Click on cell G1 and enter this formula on the function bar“ = Left(A1,9)”
12. See how it picked up the first 9 characters from the extreme left and wrote it in Cell G1 as in Fig 3a.

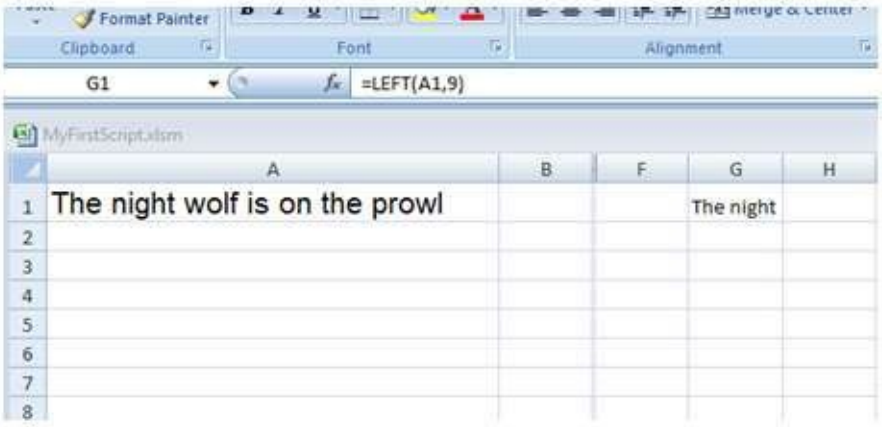


Fig 3a

Numerical manipulations using Excel:
Problem 1

D'Macy and D'JCP are two large chains that sell the same perfume ***Este-La-Noir*** in two different size perfume bottles. Our goal is to see who has a better price.

D'Macy sells the perfume in an 8oz bot tle for $115 D'JCP sells the same perfume in a 5oz bottle for $70 Create the Excel file to fit the above data.

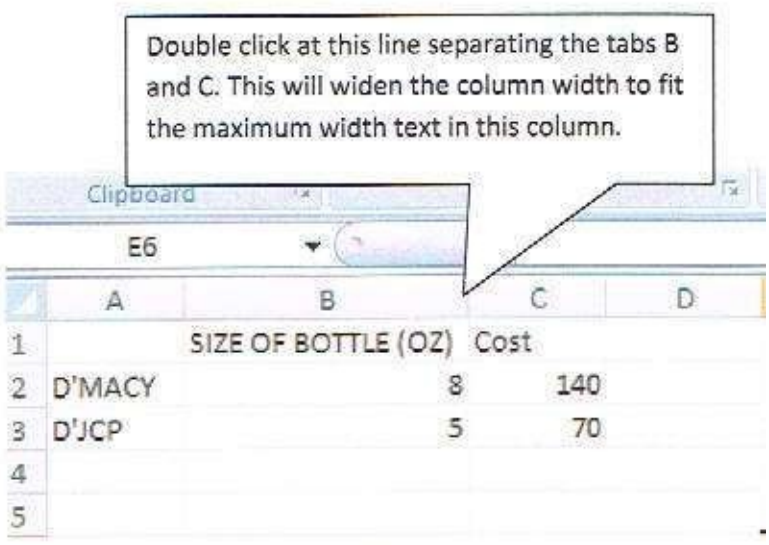


Fig 3.1

We need to find the cost per oz at each of these chains.

Let us create a heading also called a field name in Cell D1; we will call it Cost per oz. See Fig 3.2

The cost per oz at DMacy's can be found by dividing the cost of the bottle by its containing bottle size, which is 140 divided by 8. This is stored in Cells C2 and B2 respectively.

Click on Cell D2 and write the equation '=C2/B2'as shown in Fig 3.2 and press enter on

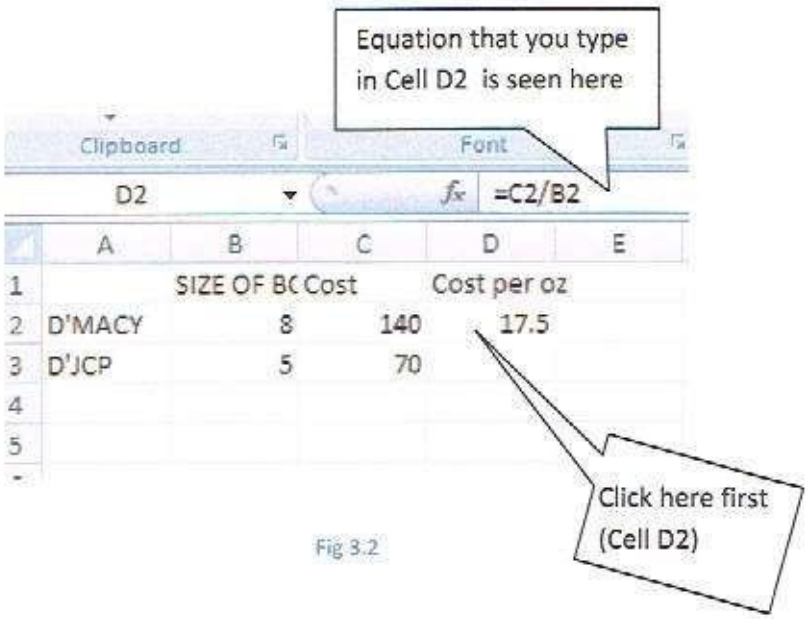


Fig 3.2

the keyboard.

Similarly write the equation (without the quotes of course) '= c3/b3' in cell D3 and press enter on the keyboard. See Fig 3.3

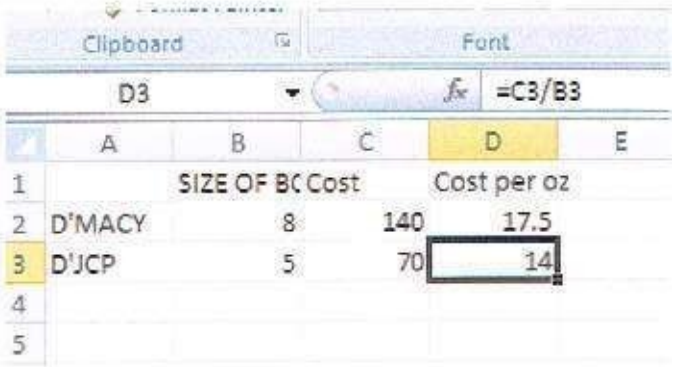


Fig 3.3

The other option is to copy the equation in Cell D2 and paste in Cell D3. This can be done by clicking the right button on your mouse and selecting the options to copy or paste. You may also click the bottom right corner of Cell D2 and drag it below to cell D3. The cell intelligently copies the equation but replaces with the respective cell ID's. See Fig 3.4

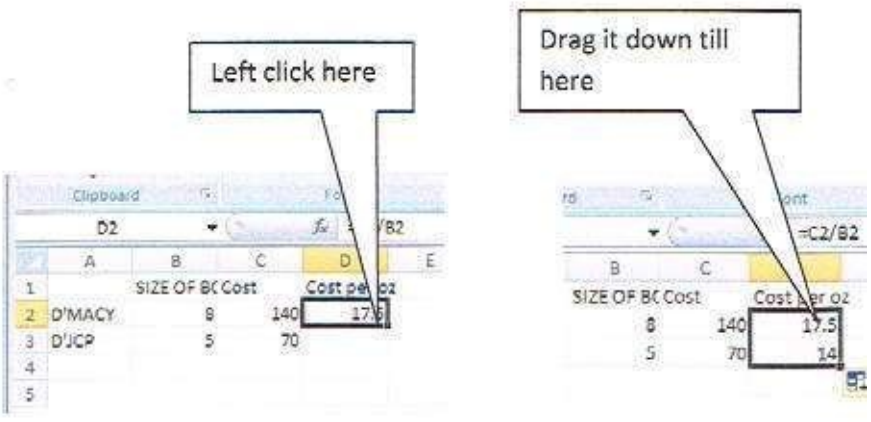


Fig 3.4

Question: By what percentage is D'JCP less expensive than D'Macy?

Answer: 20%. See Fig 3.5 for explanation.

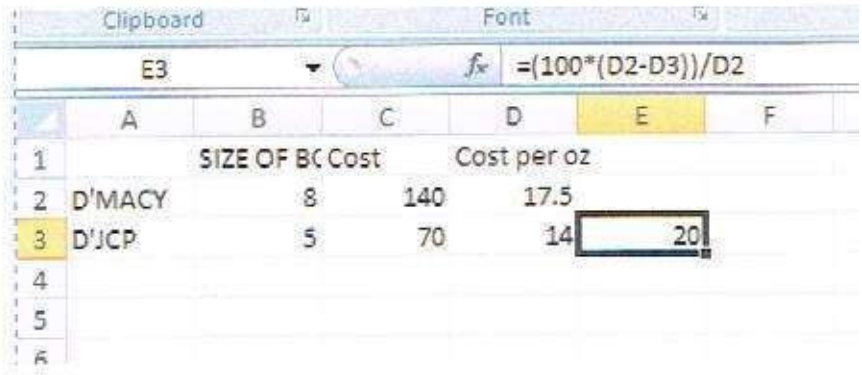


Fig 3.5

Explanation: Notice the formula
in cell E3 '=(100*(D2-D3))/D2'

First find how cheap is D'JCP? By subtracting the price per oz at D'Macy and D'JCP (D2-D3). The parenthesis ensures that you subtract them first.

D3))
Now you divide the result by the cost at D'Macy (D2)
and store the result at cell E3

Problem 2

Below are the first week 's average day temperature readings at Ockracoke Island in North Carolina for the month of May. We have Professor Duncan McDonald visiting the Island from Scotland. He wants to know the temperature in degrees Centigrade so that he can decide if he needs to pack any warm clothing. In his country they measure temperature in Centigrade.

The temperature readings we have for the days May 1-7 (shown below) are in degrees Fahrenheit.

62
74
72
68
71
70
82

The equation for conversion between Fahrenheit (F) and Centigrade (C) is given by C = (F-32) x (5÷9)
First we copy the data to an Excel spreadsheet Convert the above formula for computing. The formula

now is C= (F-32)*(5/9)
Write the above equation in Cell C3.
The equation is tobe written as '= (B3-32) * (5/9)'

because Cell B3 contains the Fahrenheit (F) temperature for the 1st of May. See Fig 4.1

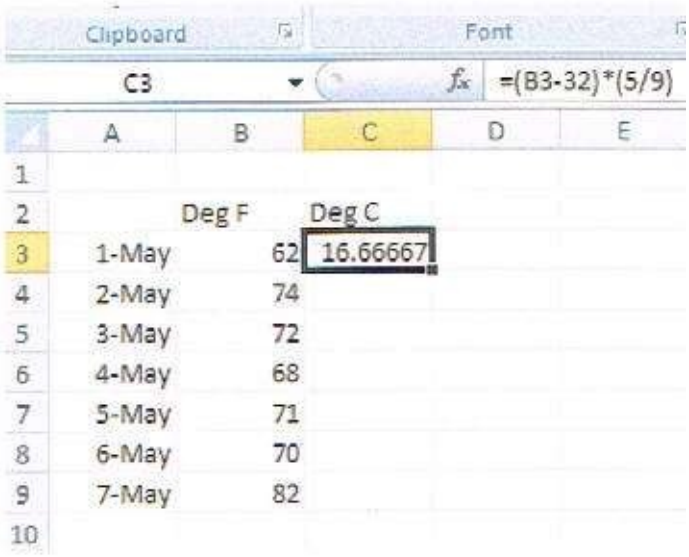


Fig 4.1

Now copy the equation in Cell C3 to the rest of the cells below (to Cell C9) as shown in Fig 4.2

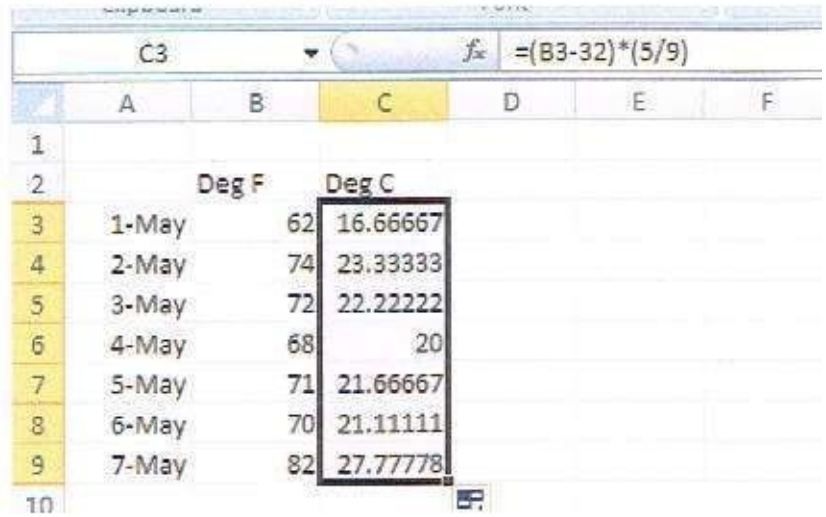


Fig 4.2

Click on the cells C3 thru C9 and format to show the results in 2 decimal places by clicking the right buttonand selecting ‘Format Cells’ as shown in Fig 4.3

Next select the ‘Number’ option and select the decimal places to ‘2’ as shown in Fig 4.4 and click ‘OK’ button.

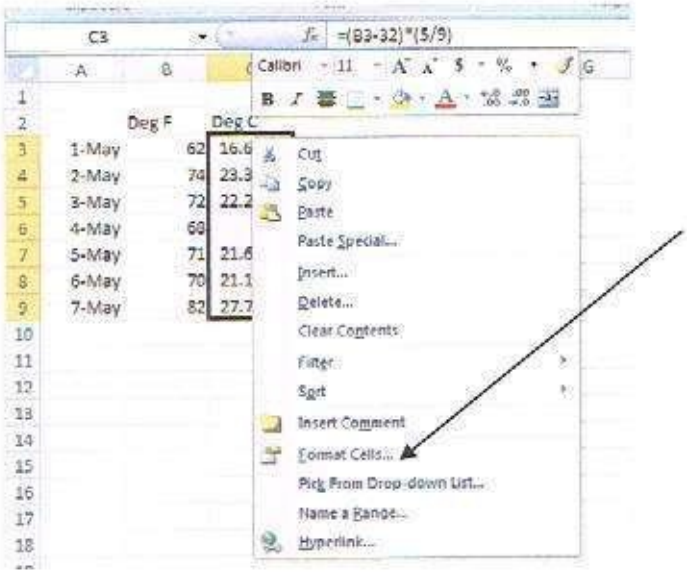


Fig 4.3

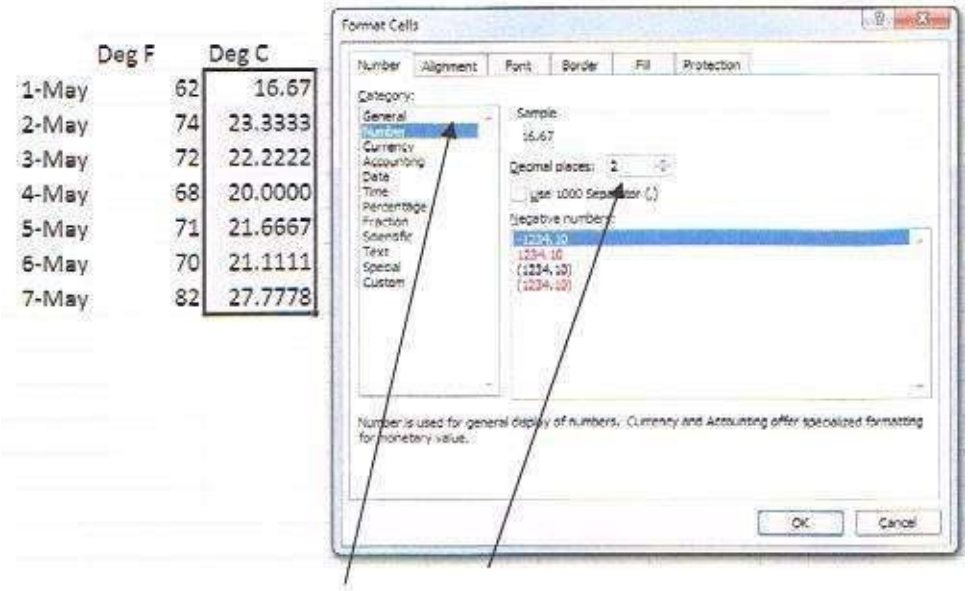


Fig 4.4

The results are now shown in two decimal places (see Fig 4.5).

	A	B	C	D
1				
2		Deg F	Deg C	
3	1-May	62	16.67	
4	2-May	74	23.33	
5	3-May	72	22.22	
6	4-May	68	20.00	
7	5-May	71	21.67	
8	6-May	70	21.11	
9	7-May	82	27.78	
10				

Fig 4.5

Initializing

With the knowledge that you have amassed in the previous chapters, it should be fairly easy for you to follow the rest of the chapters on how to be a programmer without having any programming software with you.

You have been using MS Excel spreadsheets, so you do have something that we can use.

MS Excel has a limited version of Visual Basic called VBA that is available but hiding in the background for us to use. We are going to leverage that to learn how to make our GUI's and applications.

There are a lot of things you can do with Visual Basic if you have a copy of the software with you. You can design your application and then make it into an executable file and share it around. On the other hand with VBA you can still do some rudimentary programming, but will not be able to share it as an executable file. Instead you will send a copy of your Excel file that contains your program.

Developer Tool
Open up an Excel spreadsheet and see if you can spot the Developer tool on the Top-Bar. If you do not see this bar then we will have to switch it on. Click the Office Button on the top left corner of

Microsoft Excel and select Excel Options (Fig 5.1). Under Popular tab, check the "Show developer tab in
the Ribbon"and select the Developer check box
(Fig 5.2).

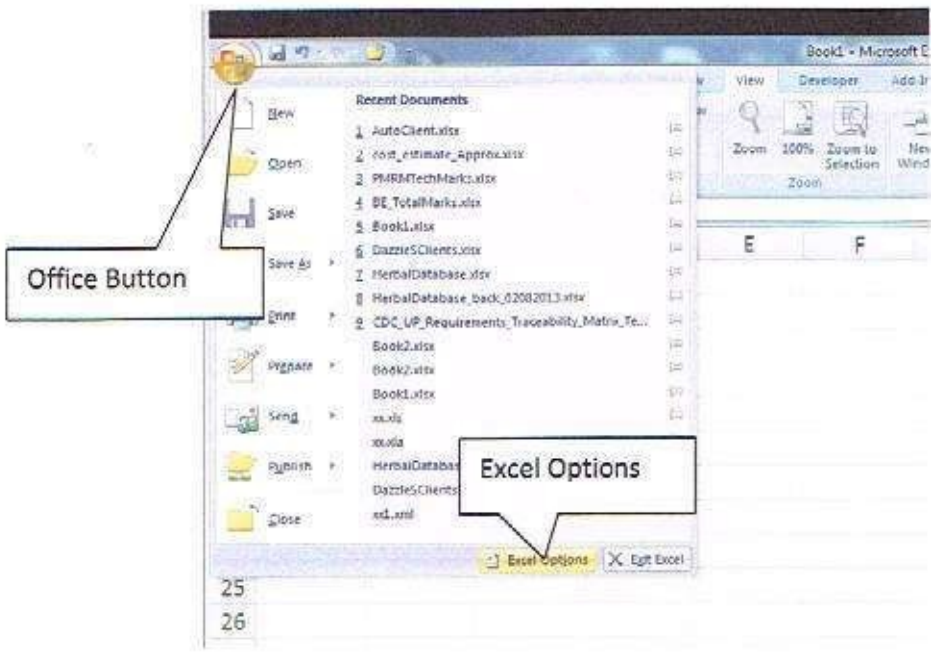


Fig 5.1

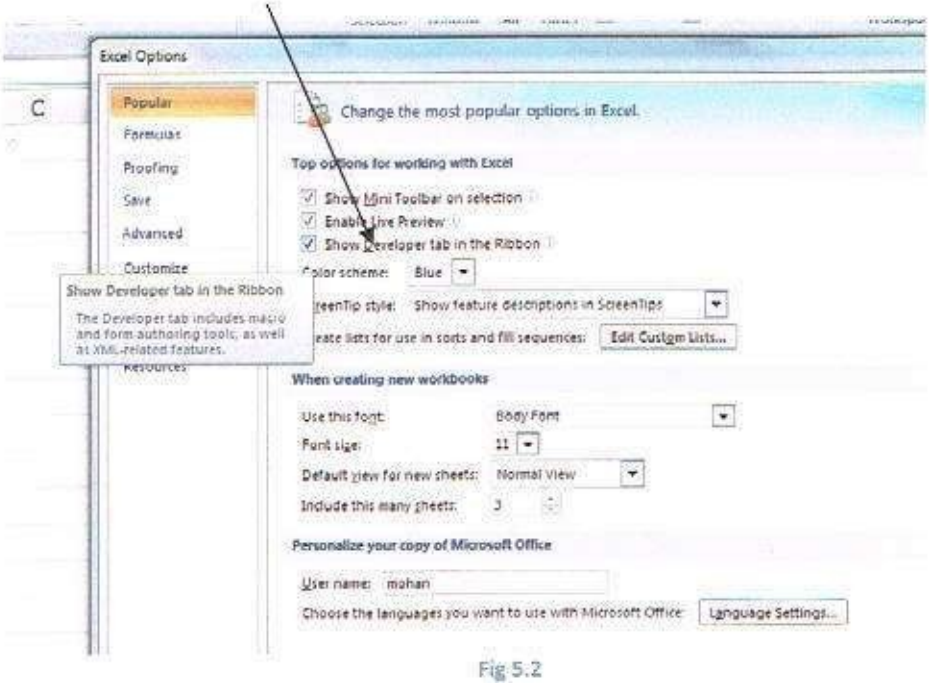


Fig 5.2

1. Now you will see the Developer Tab on the Top-bar as in Fig 5.3

Fig 5.3

2. Click on the Developer Tab; click on the Visual Basic Tab as in Fig 5.4

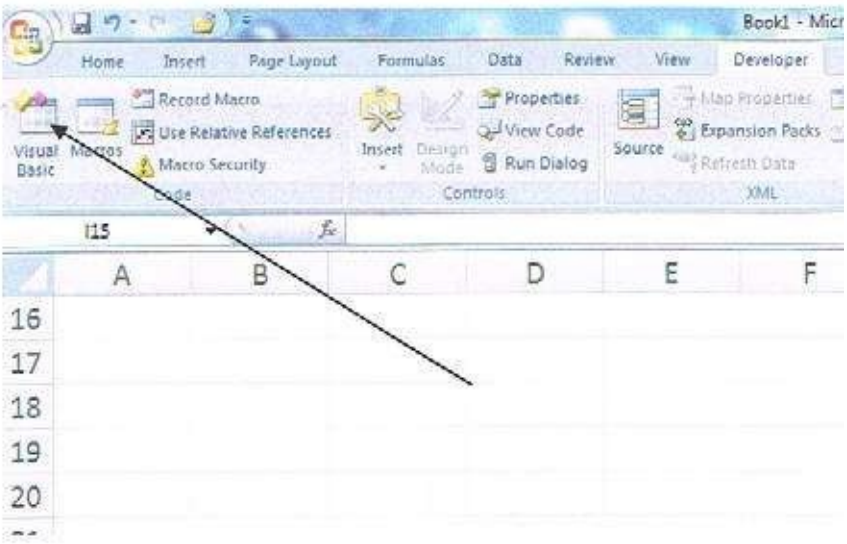


Fig 5.4

3. Now you get a Blank page (refer to Fig 5.5).

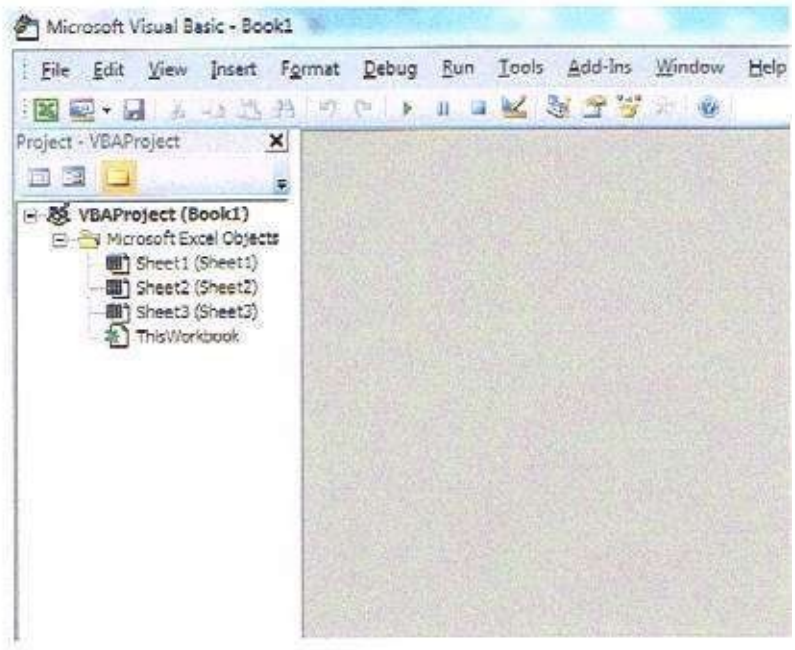


Fig 5.5

Congratulations!
Now you are ready to go into a new world of programming. **Creating your First Graphical User Interface Form** Let us take up a problem and try to code a solution!

In the earlier chapter we opened up a MS Excel spreadsheet and converted a set of numbers in Fahrenheit to degrees Centigrade.

In this exercise we will create a GUI (Graphical User Interface) where a user can enter a random two digit number and convert it into centigrade as shown in Fig 6.1

a. When the user initiates the application, an input box should ask the user to enter their name.
b. Next the application should prompt the user to enter a number (temperature) in Fahrenheit.
c. The application should calculate the equivalent Fahrenheit value and show it in the

second box; this should repeat as many times as the user inputs new numbers in the first Box.
d. If the user clicks on the ‘Exit’ button then the application should say thank you to the user and exit the program.

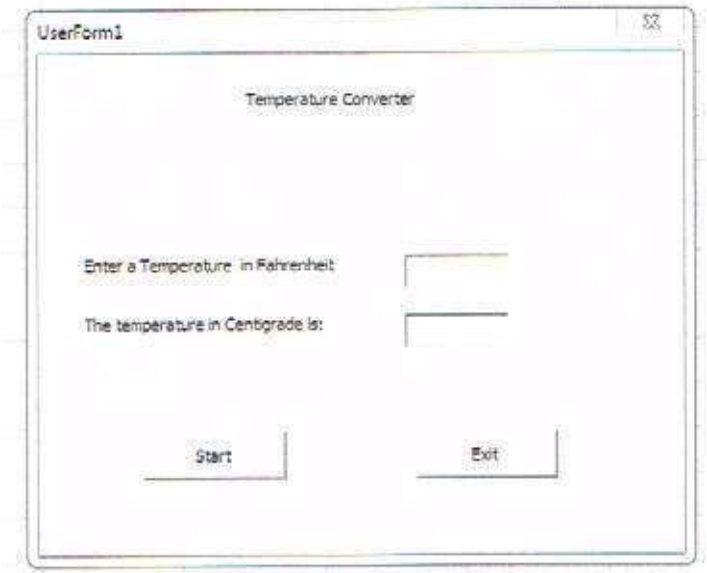


Fig 6.1

1. Click on Tab ‘Insert’ and select ‘UserForm.’ This will insert a User-Form as shown in Fig 6.2

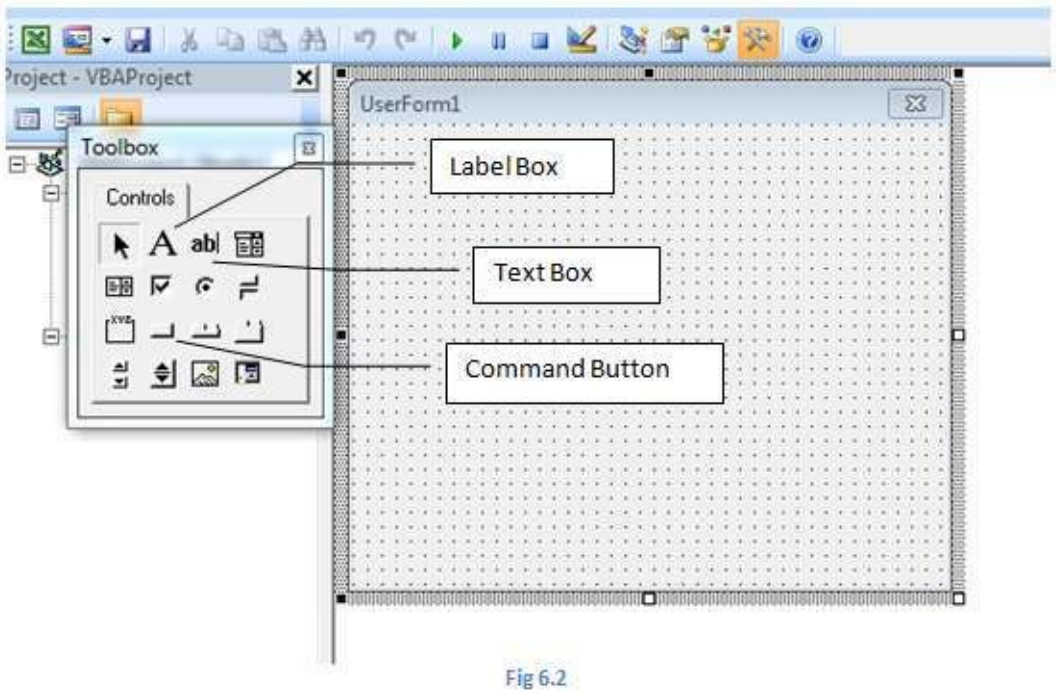


Fig 6.2

2. You can add any of the objects from the toolbox to the form and change its properties to change its look and feel using the Properties windows like Name, Caption, Font, Color etc. as in Fig 6.3. At this time, we will leave the properties as it is. In case the **Toolbox** is not visible, you may click on **View** and then select **Toolbox** option.

3. Click on the **TextBox** button in the Excel Controls **Toolbox** as in Fig 6.2 and click anywhere on the form to place a copy of the **TextBox** button on the Form.

4. Let us name it **TxtBox1** in the properties box by changing the **Name** property to **TxtBox1** as shown in Fig 6.3. You could have given any name to it. For now we named it as TxtBox1. This will be the box where a user will input a 2 digit number.

5. Similarly draw another Textbox below it and name it **TxtBox2** in the Name property. This box should display the calculated equivalent Centigrade value of the user entered value in **TxtBox1**.

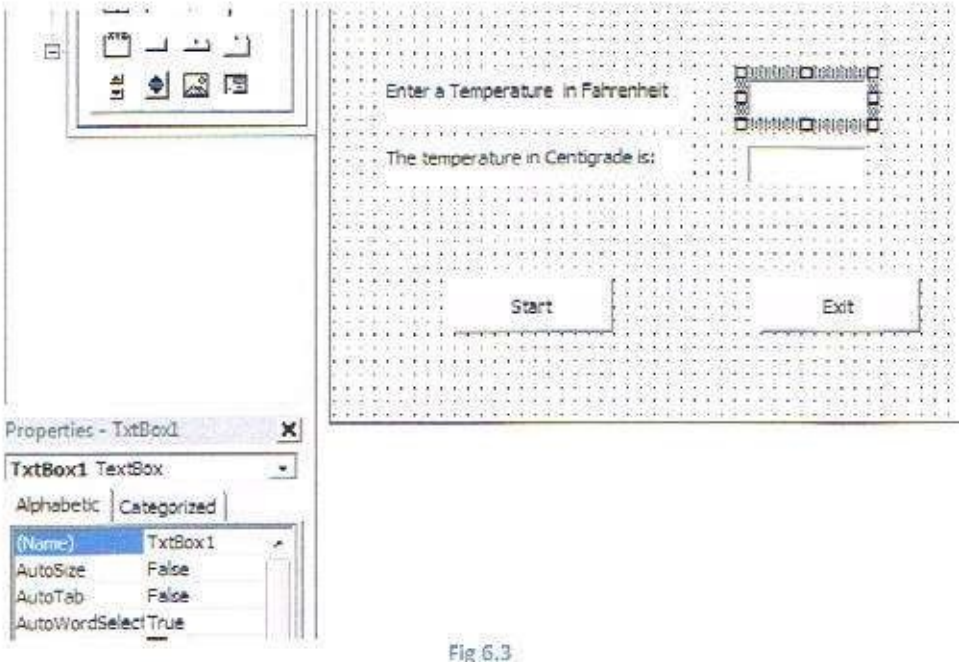


Fig 6.3

Put 3 Label boxes on the Form and change their texts to read **'Enter a Temperature in Fahrenheit,'** '**The Temperature in Centigrade is:**' and '**Temperature Converter** ' and place it around on the forms in the right locations as in Fig 6.4. Label boxes help describe functionalities to the user. By default the first label reads as LabeI1 You must change the texts by changing the text in the **Captions** property.

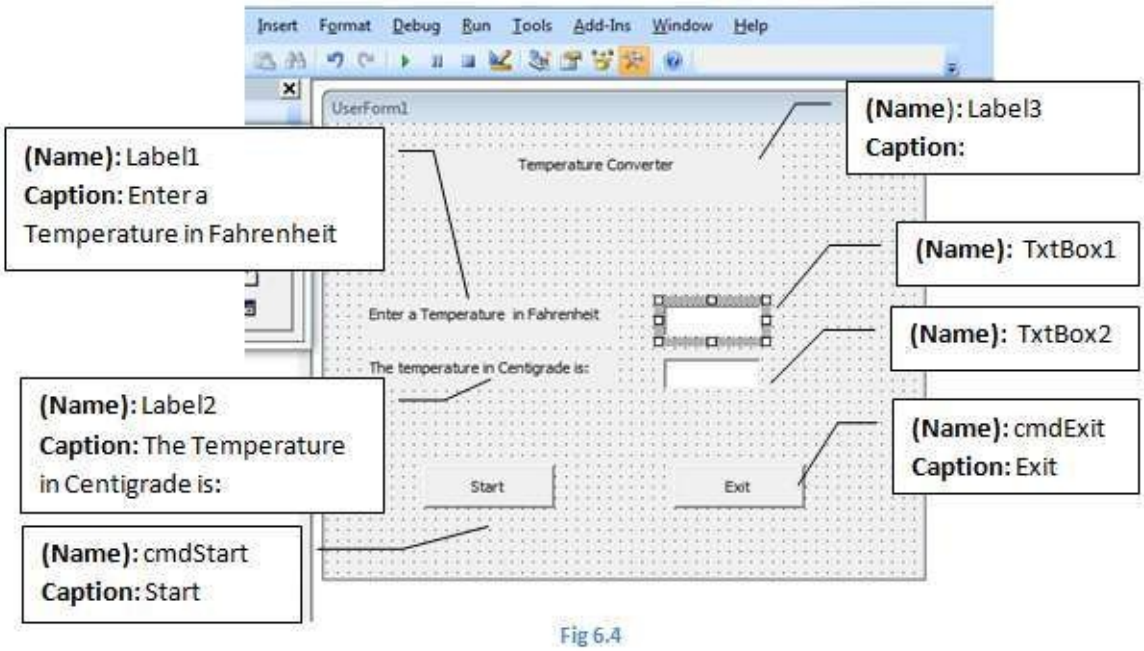


Fig 6.4

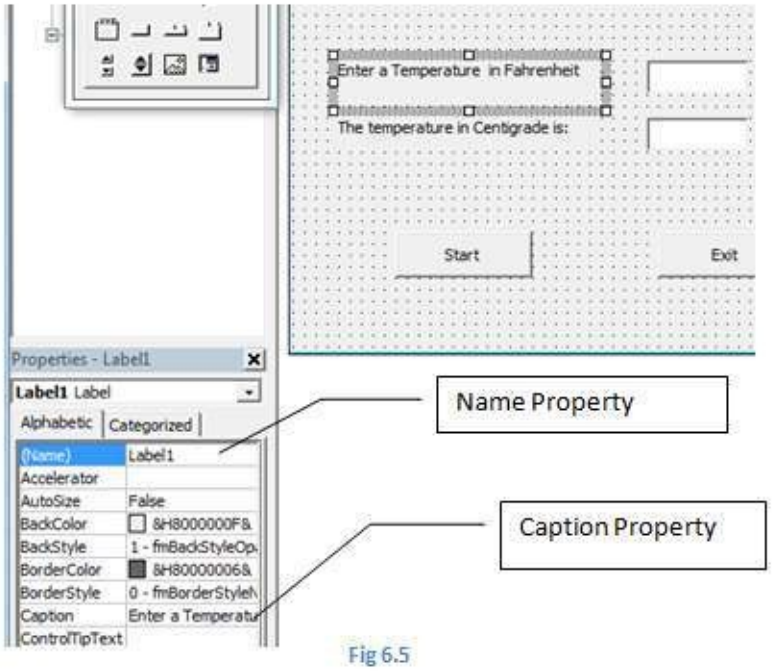


Fig 6.5

As in Fig 6.4 add two command buttons on the form

and name it **cmdStart** and **cmdExit** respectively by changing the default names in the **Name** Property tags. See Fig 6.5.

The button named **cmdStart** should read the word **Start**. This can be done by typing the text **Start** in the **Caption** Property. Similarly change the second button named **cmdExit** to read the text **Exit** on the button.

When the application is initiated, the application should ask the user to enter his/her name and store the name in a variable for later use.

Every time the user clicks on the button **cmdStart**, the application should read the data entered in **txtBox1** and output the converted result to **TxtBox2**.

When user clicks on the button **cmdExit** we want it to give a message thanking the user for using the application and then allowing the user to exit.

Before continuing, this file should be saved as a **MacroEnabled workbook**. It is up to you where you choose to save this file at.

Once I save the file, I would like to see the location where my file is stored . When we open up Explorer, we see only the file name but for the program to access it, we have to know the complete file name which includes the folder location as well.

Right click on the file name and choose properties as in Fig 7.0, and construct the complete file address.

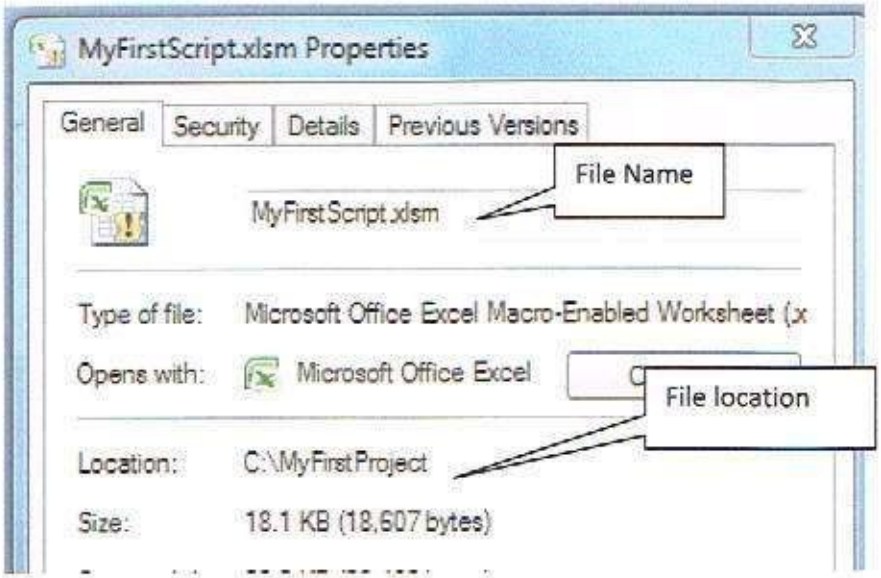


Fig 7.0

In my case the file name is **myFirstScript.xlsm** and it is located at **C:\MyFirstProject**. The complete file name is therefore **C:\MyFirstProject\MyFirstScript.xlsm.**

Note: Your complete file name is the File location followed by a \ and then your file name.
Write down your complete file name if it is different

If you click on the run button as in Fig 7.1 the application is initiated and the User-Form is ready to go as in Fig 7.2. But the application does nothing because the controls are yet to be programmed. Click on the **X** to exit the program (Fig 7.2).

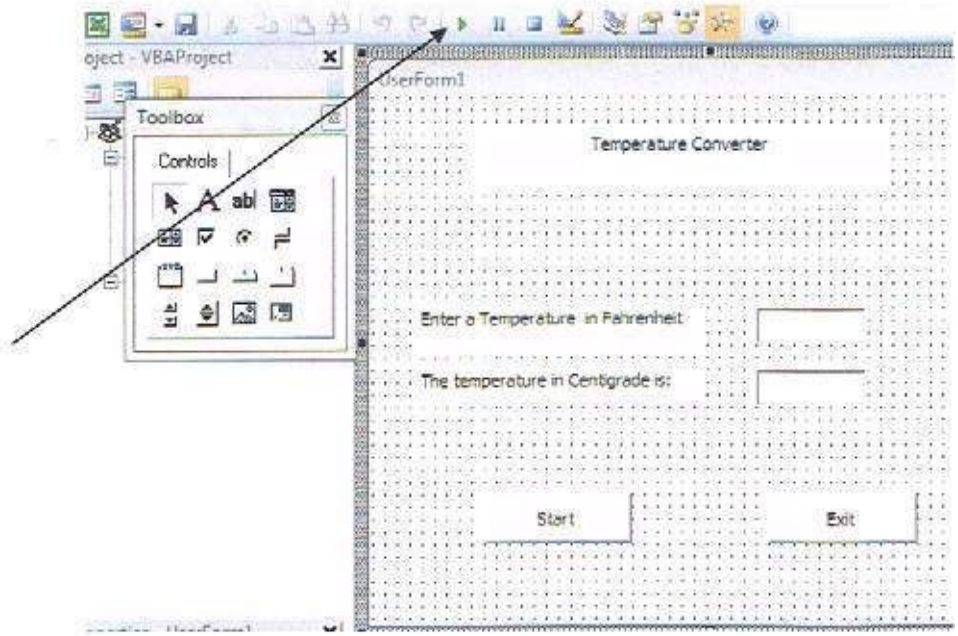


Fig 7.1

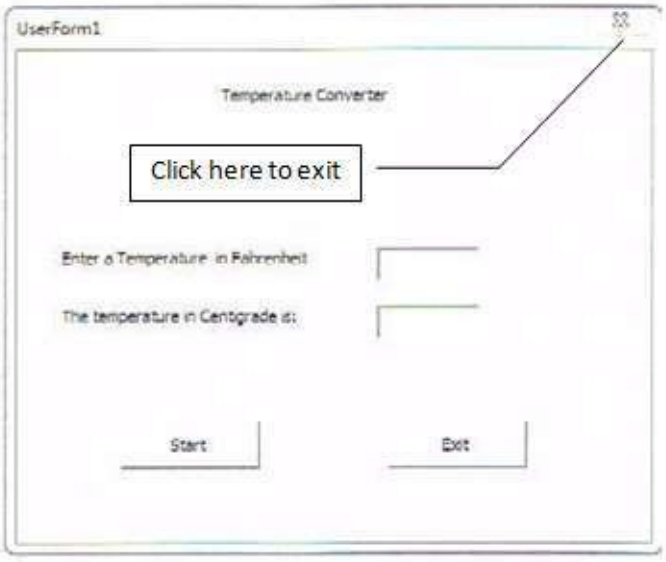


Fig 7.2

Let us begin programming the controls now.

Initialize the Form Double click anywhere on the form. You will see the code below:
Private Sub UserForm_Initialize()
End Sub
What it means: It creates an empty Sub Procedure between the two key words Private Sub and End Sub.
A Sub Procedure is a task that will be activated every time an event occurs.
This Sub Procedure belongs to the 'UserForm', i.e. the form you just created.

When the application is initialized the form will be triggered (event). You choose when the code needs to be executed. This can be chosen by selecting any one of the options on the top right of the page (see Fig 7.3).

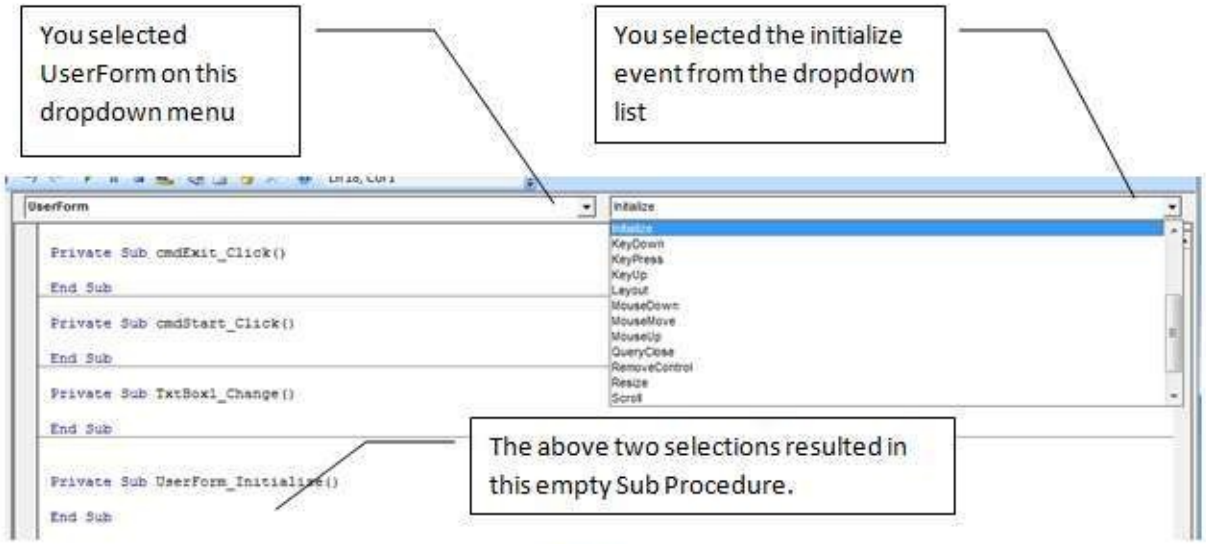


Fig 7.3

Let us say, you want to change this to a different event. You want to fire a routine task when user clicks on the form. Then from the dropdown event selection list, you should select **Click**. This will give you a new empty subroutine called **Sub UserForm_Click()**

and end with **End Sub**.

For now we will continue with the default sub procedure.

The word **Private** at the beginning of the Sub indicates that any variable you declare here is visible only to this sub procedure.

Click on **Userform1** under the Folder Forms in Project Explorer as shown in Fig 7.4 to go back to your form.

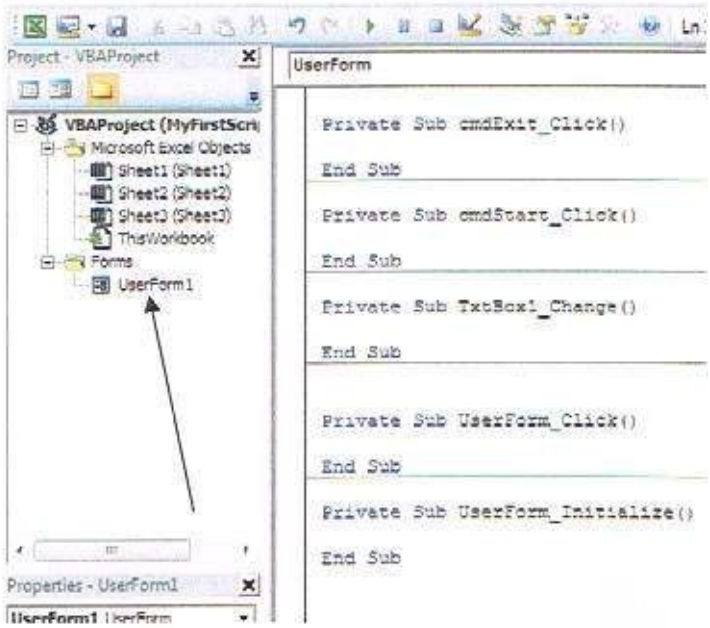


Fig 7.4

Test: Double click on the first text box called **TxtBox1**. You will notice that it will take you to an empty subroutine for TxtBox1_Change () where the default event intelligently picked by the application is **Change**. This means, the routine task you want the application to do when a user inputs a new number, thus implying a change event.

I would like the program to pick the user input number from **txtbox1** and feed it to a temperature conversion equation and throw the result to the second text box called **txtBox2**

Begin Scripting to make it happen Initialize the form

When the application starts for the first time, the **UserForm** gets initialized. We double click on the form to go to the code page.

You will see an empty sub procedure called **Private Sub** UserForm_Click() that ends with an **End Sub.**

You got this because you double clicked on the form and therefore it created a UserForm_Click event. Leave this section alone.

On the top of the of the form click the dropdown list window and select **UserForm**. This dropdown list contains a list of all the objects on the form and the form itself.

On the right hand top corner of the code window contains a dropdown list of all the events that can possibly take place. Select **Initialize**.

Now you get a empty wrapper code **UserForm_Initialize () ()** that ends with an **End Sub.**

Fill in the missing code from below **Code sample 1**

```
Private Sub UserForm_Initialize()
'Dim strName As String
strname = InputBox(Prompt="You name please.", _
Title:="Name Form ", Default:="Your Name here") End Sub
```

Code sample 1

1. This pops out an **InputBox** and prompts the user to enter his/her name as shown in Fig 8.1 when the program runs. Close this by clicking on the **X** on the top right of the form.

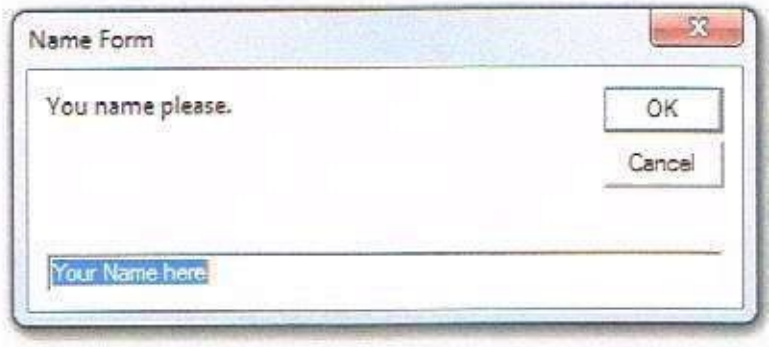


Fig 8.1

2. When the user enters his/her name on the form above it is stored in a variable called **strname** I gave it variable type **string**. I then realized that if this variable is declared in this private subroutine then other buttons outside will not be able to pick up the stored value in this variable. In our case we want the application to remember the user's name in a variable and hand it over to the exit button when clicked. Then the **Exit** button was supposed to say **Thank you** to the user.

3. On second thoughts I commented the dim statement by adding a**'** to the beginning of the statement. The statement looks green now. The application process ignores all comments. I have copied the statement to the top of the page in the general section. Any variable declared here can be called by any sub routine.

Coding the General section

Add the **Dim strName As String** at the top of the page in the general section as in Fig 8.1.1

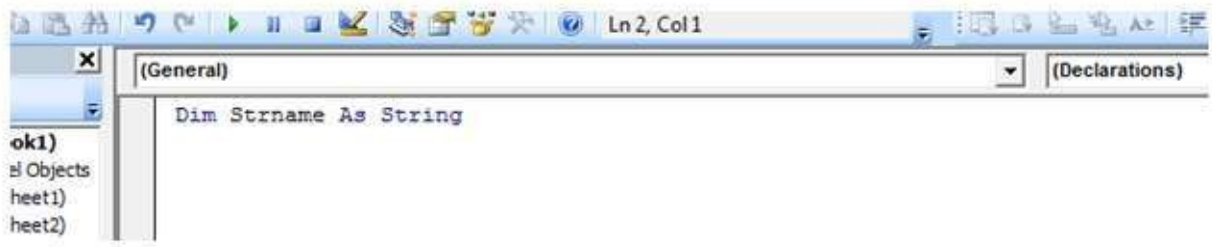


Fig 8.1.1

Coding

cmdStart button

Double click on the **cmdStart** button to go to the empty subroutine. Write the codes lines 2 thru 6 in Code sample 2

```
Private Sub cmdStart_Click()
Dim fNumber As Integer
txtBox2.Text = ” “
fNumber = Val(TxtBox1.Text)
txtBox2.Text = Str(fNumber - 32) * (5 / 9) TxtBox1.Text = ” “
End Sub
```

Code sample 2

1. First we declared a variable **fNumber** as an integer using the Dim statement (Dim stands for Dimension)
2. A string is any number of characters between quotes. If you put a blank space between the quotes then your string contains just an empty space.
3. We want to clear **txtBox2** by writing a blank space in the box. You can either go to the text properties and set it manually by entering a blank space as in Fig 8.2

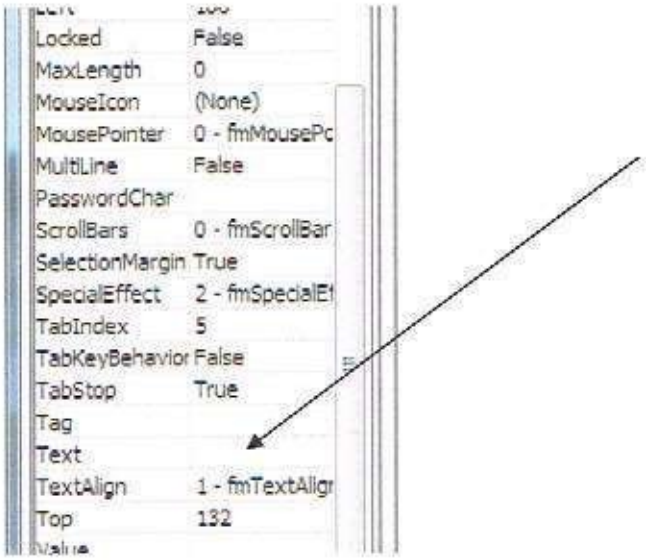


Fig 8.2

Or set the textBox2 text parameter automatically by code as in line 2 **txtBox2.Text = ” “**

4. When a user enters a temperature value in **txtBox1**, the value is stored as a string value by default in a variable called **TxtBox1.text**.

5. The Val function usage is given by **Val(ByVal Expression As String)**
Any number previously stored as a string in a variable can be converted to its real numerical value by putting the variable in parenthesis with the **Val** function on the left of the parenthesis. When a user inputs a number via the input box the value is stored as a word or a string. You cannot do anything with a number that is still in a word form. You have to use special functions like the Val function to convert it into a numerical value. To make it simple if you type 2+3= in Microsoft word it won’t give you a value because the numbers are read as words. Whereas if you type 2+3= on a calculator it gives you the value 5 because it takes the numbers that you type and reads them as a numerical value.

6. fNumber = Val(TxtBox1.Text). In this line of code the user input number stored as a string by default in the variable **txtBox1.Text** is converted to a numerical integer value and stored in the variable **fNumber**. Remember that we have declared **fNumber** as

Integer. It can only store an Integer value.

7. **txtBox2.Text = Str((fNumber - 32) * (5 / 9))**
In this code, All operations within parenthesis are performed first. So the integer number stored in the variable fNumber is subtracted by 32; multiplying the result with the result from dividing 5 by 9; the result which is obviously another integer number is now converted to a string value using the **Str** function; This string value now sets the text parameter of **txtBox2** and displays the value in the text-box. Text boxes can only display strings. Therefore if you have a numerical number which you want to display in a text box you will need to convert it to a string type.

The **Str** function usage is given by **Str(ByVal Number As Object)**

Any variable that has a number integer or real stored inside it can be converted to a string if you put it within a parenthesis with the **Str** function to the left side of the parenthesis.

Confusion about a number as a String or Integer? If two numbers 4 & 5 are stored in variables Var1 and Var2 as integers then Var1 + Var2 will add 4 + 5 and give you a value 9

If on the other hand the same two numbers 4 & 5 are stored in variables Var1 and Var2 as strings then both these variables cannot be added or multiplied because it is a text.

Coding the cmdExit button Double click on the cmdExit button Fill the missing code from below

```
Private Sub cmdExit_Click()
MsgBox strname & ” ,Thank you for using my application!” End
End Sub
```

Code Sample 3

1. The first line tells that if a user clicks on the **cmdExit** button, then execute the rest of the code.
2. Line2 uses the message-box function to pop out a message. The message will write the value stored in the variable **strname** and joins the string **Thank you for using my application**.

For example if the user had signed in as **Mr. Duncan McDonald** then at the end of the application the system will flash a message stating **Mr. Duncan McDonald, Thank you for using my application!**

3. Line 3 signifies that the application needs to be terminated.

Now run the program and test the results!
In case you have a problem, look to see that the following codes are correctly entered Dim strname As String

```
Private Sub cmdExit_Click()
MsgBox strname & “, Thank you for using my application!” End
End Sub

Private Sub cmdStart_Click()
Dim fNumber As Integer
txtBox2.Text = ” “
```

```
fNumber = Val(TxtBox1.Text)
txtBox2.Text = Str(fNumber - 32) * (5 / 9) TxtBox1.Text = ” “
End Sub

Private Sub UserForm_Initialize()
‘Dim strName As String
strname = InputBox(Prompt:=”Your name please.”, _ Title:=“Name Form “, Default:=“Your Name here”) End Sub
```

Code Sample 4

Now is a good time to run this application to see what it looks like in practice. Congratulations!
You have made your first GUI.

But you are not finished yet. I have to teach a few more things that will strengthen your arsenal and make you a real programmer. But from here I am going to be briefer in my descriptions. Straight to the point, if I may!
5 LOGICAL PROGRAMMING

Do loop

As the name suggest, it is a loop and you are asking the application to repeat doing something till something happens. We will deal with this when we do the next project.

If Statement

In **Code Sample 5** given below you can insert a **Conditional if** to allow a user to enter data no more than 5 times. After the 5th time a message should pop up to say **Thank you for using my application! You may restart again!** See Code sample 5. This was done using an‘If’ to continue repeat something until a specific criterion is met. Click anywhere on the form and delete all the previous code. Copy the code (lines 1-27) from below and paste it where the previous code was located

```
1 Dim strname As String
2 Dim recordNo As Integer

3 Private Sub cmdExit_Click()
4 MsgBox strname & “, Thank you for using my application!”
5 End
6 End Sub
7 Private Sub cmdStart_Click()
8 Dim fNumber As Integer
9 recordNo = recordNo + 1
10 txtBox2.Text = ” “
11 fNumber = Val(TxtBox1.Text)
12 If recordNo < 5 Then
13 txtBox2.Text = Str(fNumber - 32) * (5 / 9) 14 TxtBox1.Text = ” “
15 Else
16 MsgBox _
17 “Thank you for using my application”
18 MsgBox ” You may restart again!”
```

```
19 End If
20 End Sub
21 Private Sub UserForm_Initialize()
22 recordNo = 0
23 strname = InputBox(Prompt:=“Enter Your name.” _ 24 , Title:=“Name Form “,
Default:=“Your Name here”)
25 End Sub
```

Code Sample 5

Codes explained

1. Understand that line numbers have been added for readability.
2. If you copy the above code to use in Visual Basic, make sure you do not add the line numbers.
3. In Line number 2, we have introduced a Variable **recordNo** of type Integer
4. In line 12, we used an **If** Statement that ends its utility in line 19.
5. Let us Google the usage of an **If** statement (the correct format for using an **If** statement).

```
If condition [ Then ]
[ statements ]
[ ElseIf elseifcondition [ Then ]
[ elseifstatements ] ]
[ Else
[ elsestatements ] ]
End If
-or
If condition Then [ statements ] [ Else [ elsestatements ] ]
```

Source: http://msdn.microsoft.com/en us/library/752y8abs(v=vs.80).aspx

6. From the list of options available, we choose the **If Then** condition followed by an **Else** and an **End If** statement.

Follow the code:

When the user starts the application, by default whatever you have scripted in **Sub UserForm_Initialize()** and the dimensions of variable types in the General section gets initialized. In our case, two string variables declared in the general section **strname** of type **String** and **recordNo** of type **integer** are created. At this time the variables do not contain any values.

7. UserForm initialize() subroutine stores a value ‘0’ in the variable **recordNo**. And the variable **strname** stores whatever name the user types when prompted to enter his name in line 23 and 24. Note Line 23 and 24 are actually one single line of code. The continuation symbol‘_’ at the end of line 23 informs the computer program that the next line is a continuation ***** of this line.

8. Next the form gets initialized and prompts the user to enter a value in the first box and waits till the user clicks the **start** button.

Line 23 and 24 can be written in one single line without the continuation symbol if space permit as

strname = InputBox(Prompt:="Enter Your name." , Title:="Name Form ", Default:="Your Name here")

9. When the **start** button is clicked, it kick-starts the cmdStart_Click() sub procedure (lines 7 – 20)

10. Note that line 11 has a variable recordNo which was '0' when we first started the program. After every click event, recordNo will be incremented by **1** .

11. Line 10 blanks the text in **txtbox2** with a space (the character between the quote is a single space). This action clears the text box.

12. **txtBox2.Text** refers to the text stored in the text box called **txtBox2**. By default anything typed here is stored as a text (also called a string in computer language).

13. In line 11 the text stored in **txtbox1** is converted to a numerical value using the **Val** function. Note that if the user had entered 'fifty' you will get an error. You have to type thenumber '50' in its numeric form instead.

14. Line 12 checks for a condition. If the variable **recordNo** has a value less the 5, than do the next statement, i.e. convert the value in **fNumber** to a Centigrade temperature value and display it in the second text box (line 13); next blank out the first textbox by a space so that the user can enter a new value (line 14).

15. Each time the user enters a new number, variable **recordNo** gets incremented by one number and then as long as it satisfies the **if** condition, it calculates the equivalent Centigrade values using the equation on the right side of the "=" sign and assigns it to the variable on the left side. This refers to the text to be displayed in **txtBox2** on the form.

16. The user is able to enter new values four times (less than 5). On the ^{5th} time variable **recordNo** becomes 5 and therefore no longer satisfies the condition in line 12. Now the program jumps to line 15 which is the else statement**.**

17. Lines 16-17 pops out a message box stating **Thank you for using my application!** After you click okay you get another message box from line 18 that says **You may restart again!**

18. This whole sequence will repeat again unless you click on the **Exit** button to kick start the codes in lines 3-6.

19. Line 4 pops out a one line message stating the user's name (the value stored in the variable **strname**) followed by a ',' and the message **Thank you for using my application!**

Congratulations you have learned how to use a logical 'If' statement!

6 MANIPULATING DATA USING VBA (VISUAL BASIC FOR APPLICATION)

Let us do a final project that will give you more insight on the benefits of programming.

Project:

Bob Fenton works at the front desk of a Chiropractor. The office has a text file in their desktop computer that contains information about the company's clients.

When a customer calls in , b ased on a customer's caller ID, Bob wants to be able to know who is calling and find their corresponding File ID using a GUI.

The best way to do that would be to take the telephone number from the caller ID and search the database in their desktop for a match. If a match is found, he could pull up the rest of the client information like the File ID, First Name and Last Name and show it in the form, as shown in Fig 9.0.

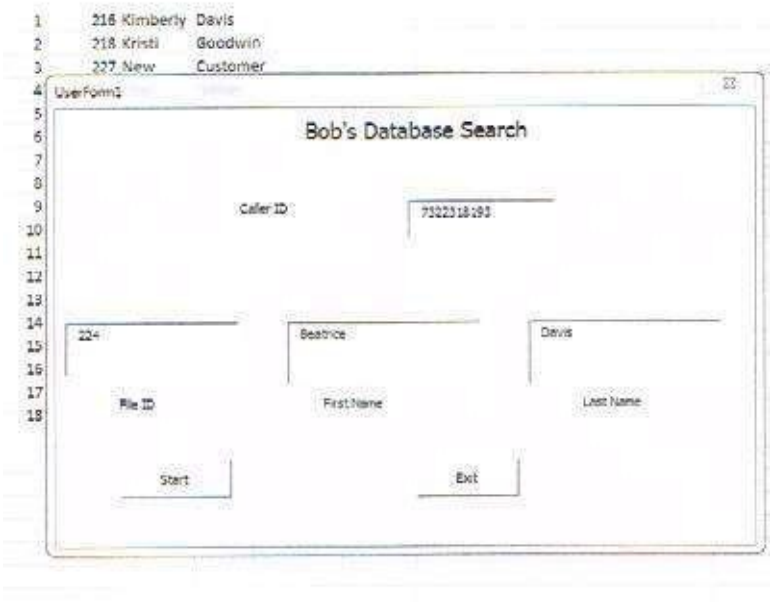


Fig 9.0

Can he do this? Yes he can! And let us help him achieve it! Let us take a small chunk of his database file and see what it looks like (the numbers have been changed for security) His database seems to have a fixed format. There are 4 items per line and each item is separated by a comma. The values are
File-ID, First Name, Last Name and Telephone Number 215, Mary, Barber, (919) 382-4156 216, Kimberly, Davis, (910) 563-7181 217, Jessica, Lowry, (919) 342-7181 218, Kristi, Goodwin, (919) 782-3146 219, Tavia, Bullock, (917) 781-9132 220, Tanneisha, Bryant, (212) 879-2135 221, Nakai, Abercrombie, (910) 782-8193 222, Chesare, Boeger, (919) 782-6172 223, Michelle, Bullock, (616) 987-2137 224, Beatrice, Davis, (732) 231-8193 225, Cerese, Barlow, (917) 781-5162 226, Candace, Jackson, (919) 675-2118 227, Geneva, Tally, (919) 235-8193 228, Paige, Cooper, (245) 781-9117 229, Staci, Newsome, (919) 765-2136 230, Maria, Black, (910) 326-8179

Let us copy & paste this in notepad and save it as a text file* to my **C** drive at: **C:\MyFirstProject\ClientData** and call it **cClient.txt** as in Fig 9.1 (create a similar location on your hard disk).

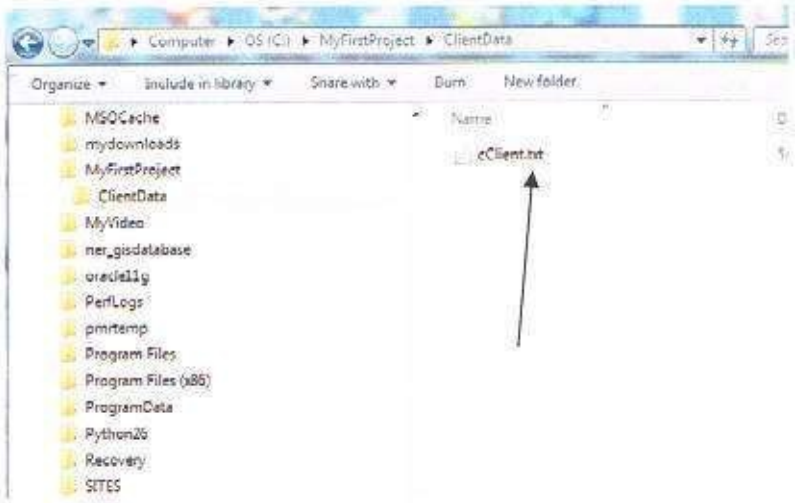


Fig 9.1

When you right-click on the file name in Explorer and select properties, it gives you the location of the file in your computer as in Fig 9.2.

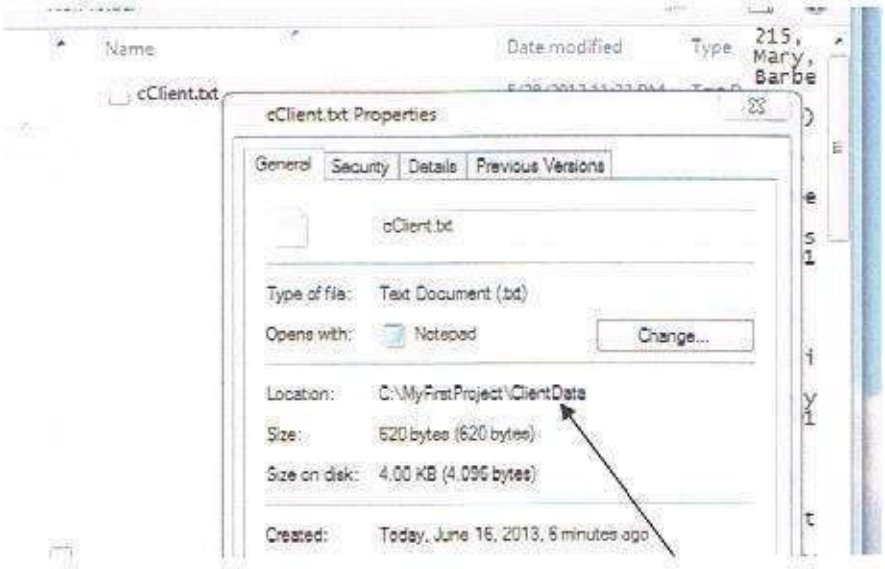
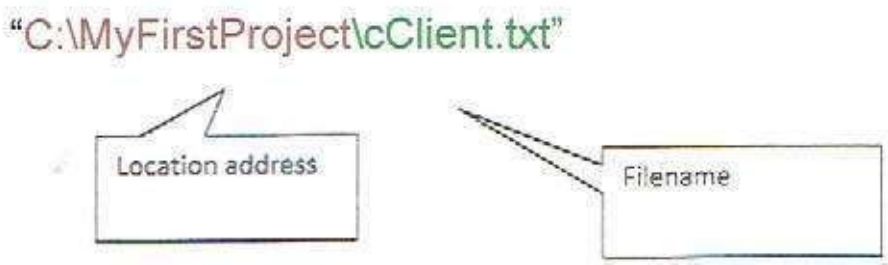


Fig 9.2

Your complete file name is the location address followed by the file name. Location address is also called the **Path**



cClient.txt is your customer database file in a text format.

Let us create a form with a text box where you can type in a client's telephone number; open up the customer database file; search line by line for a matching telephone number and display the corresponding First name, Last name and Client file number. Also store the same information in the excel file to list out the people who called. If a match is not

found just give a message saying **New Customer!**

1. If Excel is open, close it.
2. Open a new Excel file; save the file as
C:\MyFirstProject \ clientListReader.xlsm as in fig 9.3a.
a. Click on the Microsoft Office Button and select the file **Save as** option.
b. Browse to your **C:\MyFirstProject** folder. c. Type your file name **clientListReader** in the **File name** box.
d. In the **Save as Type** box choose **Excel Macro Enabled Workbook (*.xlsm)** from the drop down list.

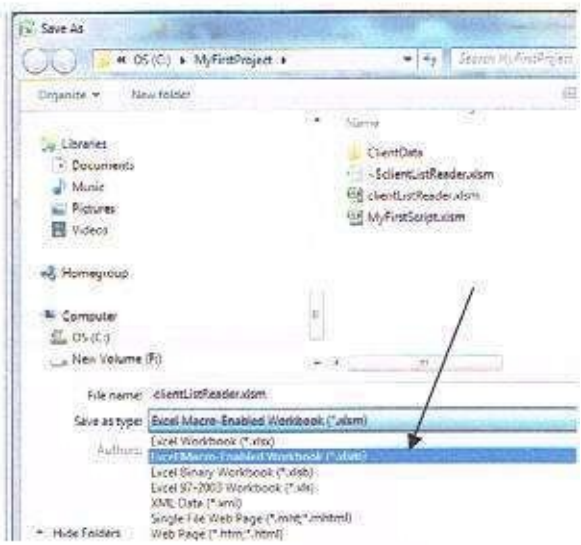


Fig 9.3a

3. Click on the **Developer** tab; click on Visual Basic; now you are in Microsoft Visual Basic page. As in Fig 9.3b

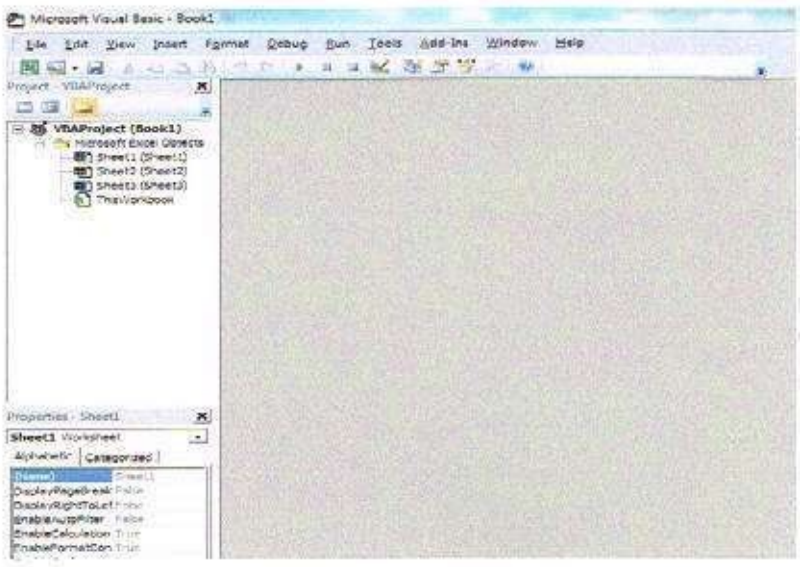


Fig 9.3b

4. Click Insert/UserForm as in Fig 9.4a. This will insert a new form. Resize it by selecting the handlebars on the edge of the form and dragging it to an appropriate size as in Fig 9.4b

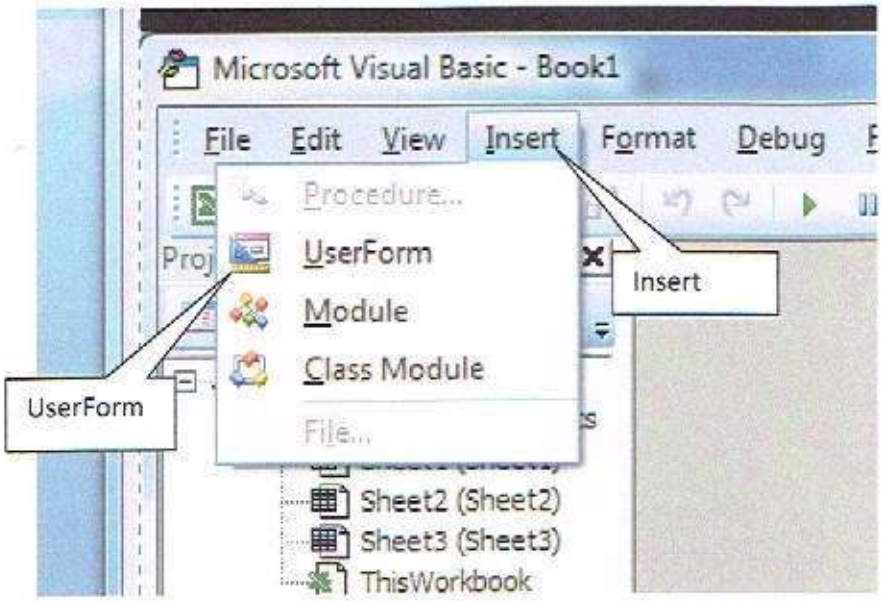


Fig 9.4a

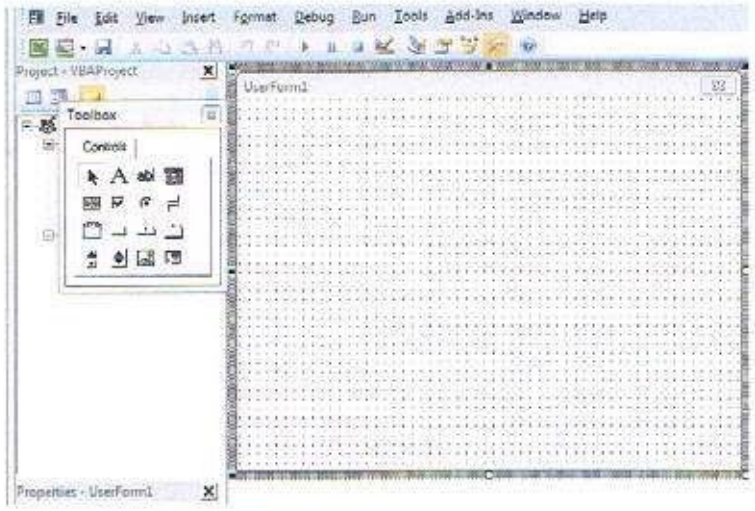


Fig 9.4 b

5. We need 4 Textboxes. We will call them **txtPhoneN**, **txtFileN**, **txtFname**, **txtLname**; we also need 4 label boxes with appropriate texts to describe the Textboxes. Let us name them **lblPhoneN**, **lblFileID**, **lblFname**, **lblLastN**, (note: Text box, Label names, command button names etc. do not have spaces in their names) with Captions set to Caller ID, File Number, First Name and Last Name respectively; we need 2 Command Buttons. One to start the search and match processes and the other to exit the program. Let us call them **cmdStart** and **cmdExit**. Now set their captions to **Start** and **Exit** respectively. See Fig 9.5

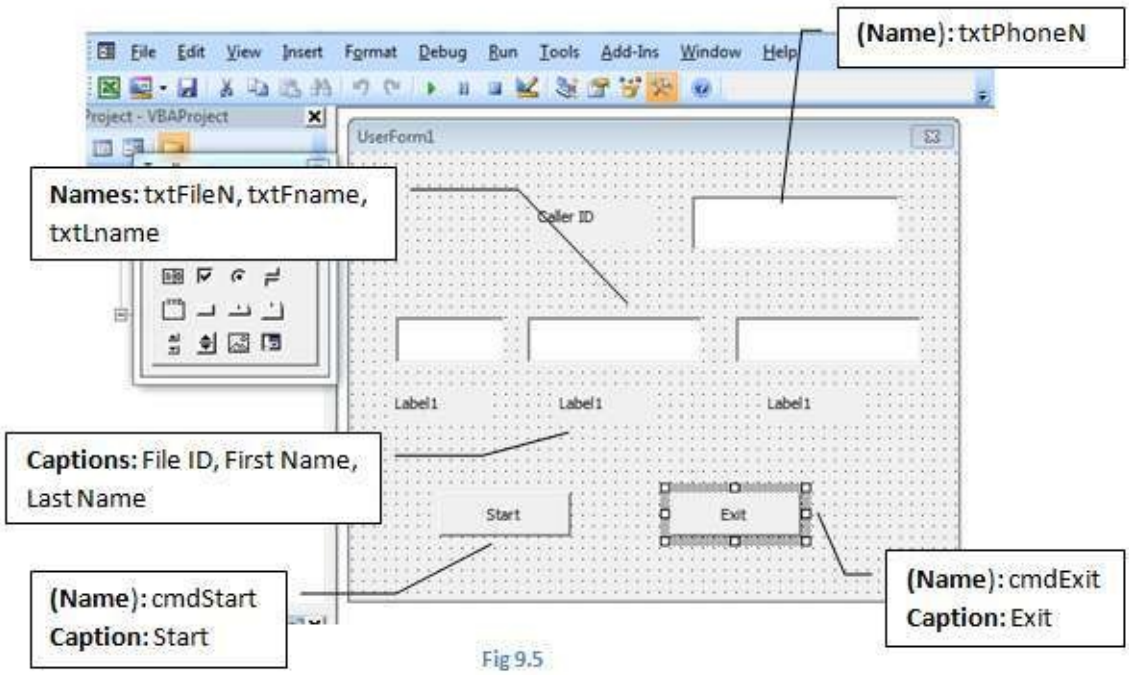


Fig 9.5

6. Double click on the form to create an empty subroutine for the Userform_Click() ; go back to the form by clicking on **Userform1** under **Forms** in Project Explorer. See Fig 9.6

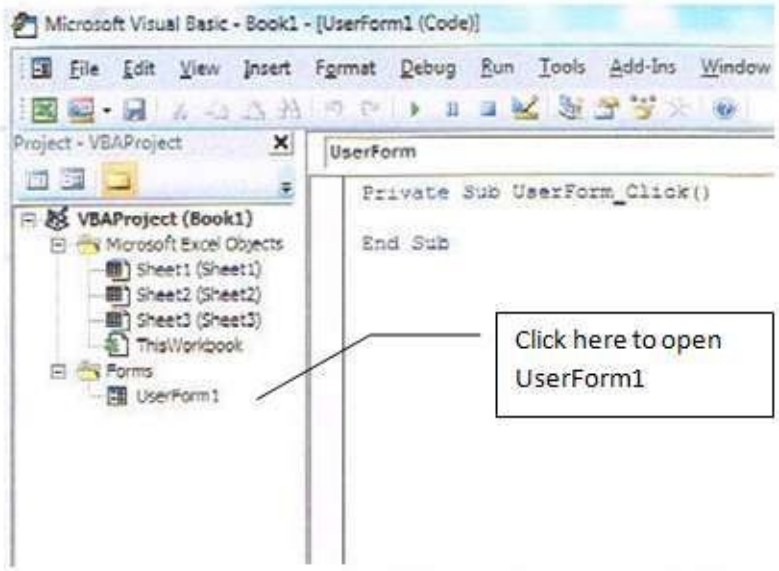


Fig 9.6

7. Similarly when you double click in each of the text boxes or command buttons, empty subroutines with their respective event actions are created. 8. But we will delete them all and copy and paste Code Sample 6.
9. Select all the code as in Fig 9.7 and delete it.

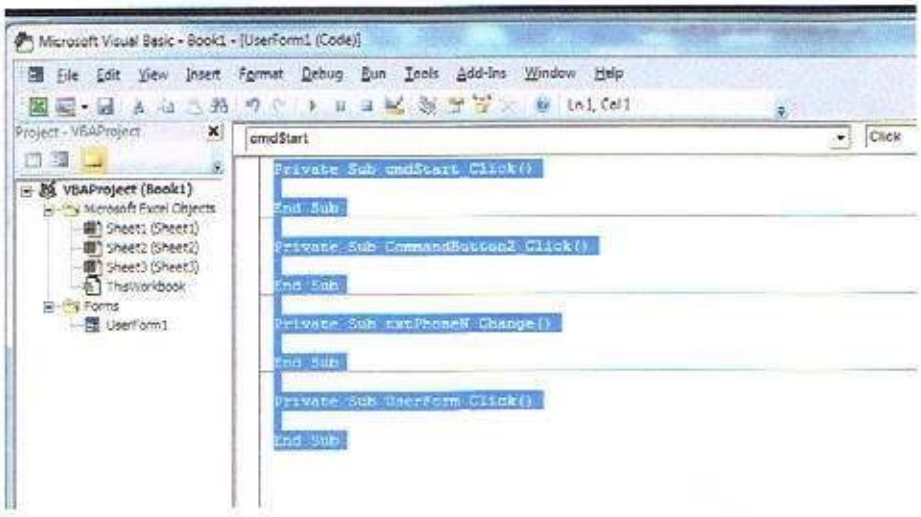


Fig 9.7

10. Copy and paste **Code sample 6** from below on the form Code window.

```
Dim callerID As String
Dim myLastRow As Long
Dim recno As Integer
Dim myLastColumn As Long

Private Sub cmdExit_Click()
End
End Sub

Private Sub cmdStart_Click()
callerID = “(” & Mid(txtPhoneN.Text, 1, 3) & “) ” _
& Mid(txtPhoneN.Text, 4, 3) & “-” & Mid(txtPhoneN.Text, 7, 4)

Call TextMatchSearch
End Sub

Sub TextMatchSearch()
Dim FileID, FirstName, LastName, _
PhoneNumber As String
Dim matchFound, lastrow As Integer
‘Variable matchFound is created to track the status
‘ of a telephone number.search
‘If a match is found we will set matchFound to 1
‘else we set it to ‘0’
matchFound = 0
‘ Open the database file as an input to read it.
‘ Call this connection as ‘#1’
```

```
Open “C:\MyFirstProject\ClientData\cclient.txt” _ For Input As #1

‘Read the above file line by line till the end of the ‘file is
reached
While Not EOF(1)
‘ Each line has a fixed format of 4 items all
‘separated by a comma.
‘ Feed the 4 items to the 4 variables.
Input #1, FileID, FirstName, LastName, _
PhoneNumber
‘ If a match has been found enter this section
‘Lines 41 to 62
If callerID = PhoneNumber Then
matchFound = 1
txtFileN.Text = FileID
txtFname.Text = FirstName
txtLname.Text = LastName
‘Call the “myLastCell” function.
‘This function tells the record number
‘of the last filled cell in the current Excel file
‘after which you can add new customer data.
Call myLastCell
‘The application returns back with current values ‘for ” myLastRow”

‘ Variable recno contains the next empty row
‘number where you can add new data.
recno = myLastRow + 1
Cells(recno, 1).Value = recno
Cells(recno, 2).Value = FileID
Cells(recno, 3).Value = FirstName
Cells(recno, 4).Value = LastName
txtPhoneN.Text = ””
End If
‘Go back to line# 40 and repeat the above steps
‘till you complete reading the end of the file
Wend
‘ If a match has not been found in our records then
‘do this
If matchFound = 0 Then
‘ clear out the four textboxes
txtFileN.Text = ””
txtFname.Text = ””
```

```
txtLname.Text = ""
txtPhoneN.Text = ""
MsgBox "New Customer!"
'Call the "myLastCell" function to find the last
' filled row number in your Excel file

Call myLastCell
'recno consists the value of the next empty row
'in the Excel file.
' write the string "New" "customer" in cells C & D
'respectively.
recno = myLastRow + 1
Cells(recno, 1).Value = recno
Cells(recno, 2).Value = ""
Cells(recno, 3).Value = "New"
Cells(recno, 4).Value = "Customer"
End If
Close (1)
End Sub
Sub myLastCell()
' This is a function that is tasked to open the current Excel
'sheet and find out the last filled row and column.
'When you call this function, it will update 'myLastRow' and
'myLastColumn with the last filled row and column numbers
'of 'your excel 'sheet.


Range("A1").Select
On Error Resume Next
myLastRow = Cells.Find("*", Range("A1") _
, xlFormulas, , xlByRows, xlPrevious) _
.Row
myLastColumn = Cells.Find("*", Range("A1") _ , xlFormulas, , xlByColumns,
xlPrevious).Column  Cells(myLastRow, myLastColumn).Select
End Sub
```

Code Sample 6

If you manually type in the code, remember that all statements that begin with ' and is green in color are comment statements therefore need not be copied.

Codes Explained

1. When you later run the application, our form would look like Fig 10.1 below.

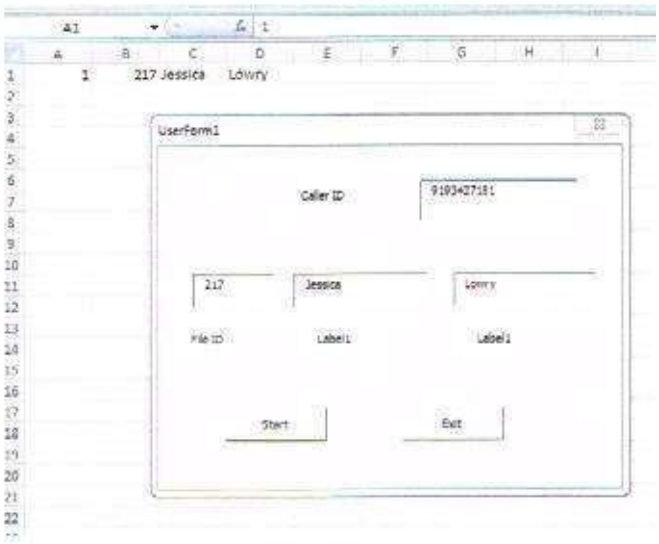


Fig 10.1

The user types in a telephone number and clicks on the start button. The application fires the **Private Sub cmdStart_Click()** sub procedure (lines 10-15.)

2. The user entered number (e.g.: 9193427181) is taken and reconstructed to a telephone number taken and reconstructed to a telephone number 7181) and stores it in variable **callerID**. The 10 digit numbers are split into three parts; parenthesis are inserted to hold the first three digits; a space followed by the next three digits; and a hyphen followed by the next 4 digits.

3. Next it sees the word **call TextmatchSearch**. (see line 14). It looks in the entire program to see if there is a function or procedure by that name. It does find the function between lines 17 and 90.

4. A **Function**/**Procedure** is like a helper whose job is to do a routine job. For example you are in an icecream shop. Each time the front desk takes a specific order. He calls out to the helper who fetches the made-to-order ice-cream and disappears till he is called again.

5. The function **TextmatchSearch** is designed to pick a re-constructed telephone number and compare it with each record in the database to see if it can find a match in the database reading it line by line till the end of the file. If it finds a matching number then it will pick the rest of the client information from the database and populate the GUI, else it will give a message saying **New Customer**. In both cases it will write some client information on the excel file.

6. At lines 18 and 20 it has created a bunch of string and integer variables.

Remember that our database format has 4 items per line and each item is separated by a comma. The values are
File-ID, First Name, Last Name and Telephone Number. We are going to read the database line by line to compare with the values stored in the variable **callerID** so we will need 4 temporary variables. Let us call them **FileID** , **FirstName**, **LastName** and **PhoneNumber** to hold them as we read each line. **matchFound** is a variable that will have a value **1** if a match is found and a **0** if no match is found, you will see as we go further why we need to keep changing this value.

7. At line 28 the program opens our database text file C:\MyFirstProject\ClientData\cclient.txt as an input for reading and calls this connection Channel #1.

8. Remember you need to change this line of code to your complete file name if it is different

9. Lines 33 to 65 creates a loop. A **while** statement always ends with a **wend** statement.

10. Line 33 says that while the file opened and named as channel #1 has not reached the **End of the File** (EOF) condition repeat some steps.

Line 37 states that in the connection named #1, all the 4 items that are read sequentially in each line is temporarily stored in the 4 variables, **FileID**, **FirstName**, **LastName** and **PhoneNumber**. These variables remain unchanged till the program finishes one cycle at line 65 (the **wend** statement); then it reads the next line of our data file; picks up the next set of 4 values; stores them in the above variables and so on…

Before the **wend** statement is reached, however there are a set of tasks to complete. Let us look at them sequentially…

11. Remember that in line 11 we have constructed the telephone number of the client and stored them in variable **callerID**

12. In line 41 we are checking to see if the values stored in the variable **callerID** matches with the values in variable **PhoneNumber** read from the first line of our data file. The 'If' statement starts at line 41 and ends at line 62.
If a match is not found then the program jumps to the **Wend** statement at line 65 and loops back to the beginning of the while statement at line 33 and reads the next line of data. This process repeats till all the lines in our input file is read and the End of File (**EOF**) has reached.

If a match is found then we set variable **matchFound** to 1. This is only for our reference. We could have given it any value! This is in Line 42

Pick up the values in variables FileID, FirstName, LastName and display them on the GUI form in textboxes **txtFileN**, **txtFname** and **txtLname** respectively (lines 43-45).

13. At line 50 the program sees **call myLastCell** and It recognizes that there is a function by that name.
14. Then the program jumps to the procedure at line 91 and runs the procedure till it ends at line 108; this procedure finds the last filled row in sheet1 of the Excel file.

15. This is required because the program needs to know where the program should write the next line of information in the Excel file.

16. Notice that two variables myLastRow and myLastColumn were set to contain the last filled row and the last filled Column numbers. These variables were declared in the general section at the top of the program page. This variable is therefore available to any sub procedures or functions that want to read the current values in these variables. These variables are called **Public** variables.

17. Had these variables been declared in the sub procedure **myLastCell** then it would have been called a **Private** variable, meaning that these variables will not be accessible to other

sub procedures.

18. After the Procedure is completed, the program automatically returns back to the point where it was called from. In our case it goes to the next executable line after the **call myLastCell** statement after line 78

19. At line 83, variable **recno** stores the value stored in myLastRow incremented by 1; This is the next empty row in our Excel file to write the new line of customer information.

In that row we write the values in **recno**, **FileID**, **FirstName**, and **LastName** in Cells 1,2,3 and 4 respectively (Lines 43-45)

20. At line 61 we overwrite the textbox containing the phone number with nothing. In other words we cleared the slate prompting the user to type in a new phone number.

21. If a match was found then the 'matchFound' value is set to **1** therefore lines 68-88 is not executed. The 'If ' statement began at line 68 and ended at line 88 with the **End if** statement.

22. Line 89 closes the file called Channel #1
23. Line 90 ends the sub procedure.
24. The entire process will repeat after you enter a new number and click on the **Start** command button.

25. If a telephone number match was not found in our database then the value **matchFound** is still **0**, then the program satisfies the condition in line 68 and therefore executes statements lines 68-88

It then blanks out all our textboxes (Lines 70-73) by putting nothing between the quotes **""** .
The program pops out a message box stating "New Customer!" in line 74.

At line 78 the script finds the last filled row of the excel file.
At line 83 we increment the value in **myLastRow** by 1 and store it in variable **recno**. This is the row number which is the next empty row in our excel file. Lines 84-87 updates the columns 1,2,3 and 4 with the value in **recno**, a nothing (""), the word **New** and **Customer** respectively.

At line 89 the script closes Channel #1 which was opened to read the input file in Line 28 If a file that has been opened for reading has not been closed by the end of the program, then that file is said to be locked and is not available to any other program or people to read it. This is done by the Close statement followed by the channel number within parenthesis. In our case we opened the file as #1 in line 28, so we had to close the file when we completed reading it by the **Close (1)** statement.

26. If you click on the **Exit** button on the form, the program executes the procedure called **Sub cmdExit_Click()** in lines 6-8. The program sees the **End** statement in line 7 signifying the end of the application. This terminates the application.

Try entering some existing numbers in the database to see if it picks up the respective customer information

For e.g. try 9177819132 or 2128792135. The application should pick up Tavia Bullock or

Tanneisha Bryant’s information. Next try entering your telephone number. It should say **New Customer!**

Congratulations once again! You are now a new member in the programming world.

This is not a complete book that teaches you visual basic in its entirety. It introduces you to programming basics and helps you dispel the fear of programming, preparing you for further self help reading.

7 MORE READING

If you have followed this book completely then you will have sufficient knowledge and background to pick-up from where this book ends.

There are several free online resources and books in the market which you can use for advanced reading . Most important of all, you will be able to do basic manipulations of data residing either in an external text file or in an Excel file.

You have already finished taking your first step in the field of software programming. Suggestions for improving this book is welcome, please email me at author.ProgrammingBasics@gmail.com Thank you for being with me in this journey!

ABOUT THE AUTHOR

Mohan Palleti has a Post Graduate Degree in Computer Science Engineering and a Graduate Degree in Electronics and Communication Engineering. He has extensive experience in the field of Geospatial Science and Research.

Mohan is a coveted motivational speaker for international conferences concerning the advancement of youth in the Informational Technology marketplace.

His style is that of a motivator. He encourages our youth to examine the world of programming to assist them in launching both educational as well as career goals. He also prepares instructors to be able to start an introductory course in software programming.

He is gifted in making complicated subjects like Math, Science and Software Programming look easy and he is able to draw the audience into these fields and dispel their fears.

Mohan Palletti is available for speaking engagements, schedule permitting.