POLITEKNIK MALAYSIA
MERLIMAU

VISUAL BASIC 6.0

DESKTOP APPLICATION FOR GEOMATIC

NOOR FAIZAH ZOHARDIN
AZRINA ZOLKIFLI

VISUAL BASIC 6.0

DESKTOP APPLICATION FOR GEOMATIC

Writer
Noor Faizah Binti Zohardin
Azrina Binti Zolkifli

Published in 2022

Perpustakaan Negara Malaysia Cataloguing-in-Publication Data

Noor Faizah Zohardin
VISUAL BASIC 6.0 : DESKTOP APPLICATION FOR GEOMATIC /
NOOR FAIZAH ZOHARDIN , AZRINA ZOLKIFLI.
Mode of access: Internet
eISBN 978-967-2762-17-1
1.Visual basic (computer program language).
2. Geomatics.
3.Goverment publications—Malaysia.
4. Electronic books.
I. Azrina Zolkifli. II. Title
005.133

Published by:
Politeknik Merlimau, Melaka
KB1031 Pej Pos Merlimau,
77300 Merlimau Melaka

PREFACE

VISUAL BASIC PROGRAMMING provides students with knowledge of the programming concepts using the Visual Basics programming language. The course emphasizes on design the programme which includes examining code, looping statement and also creates and documents naming standards.

This book is written specifically to satisfy the syllabus requirements for subject DCG 50252 Visual Basic Programming. This book contains all required topics for Diploma Geomatic.

This book contains 5 chapters that have been planned and arranged carefully based on Polytechnic Malaysia syllabus. All concepts for each topic are accompanied by detail explanations, followed by examples and complete solutions.

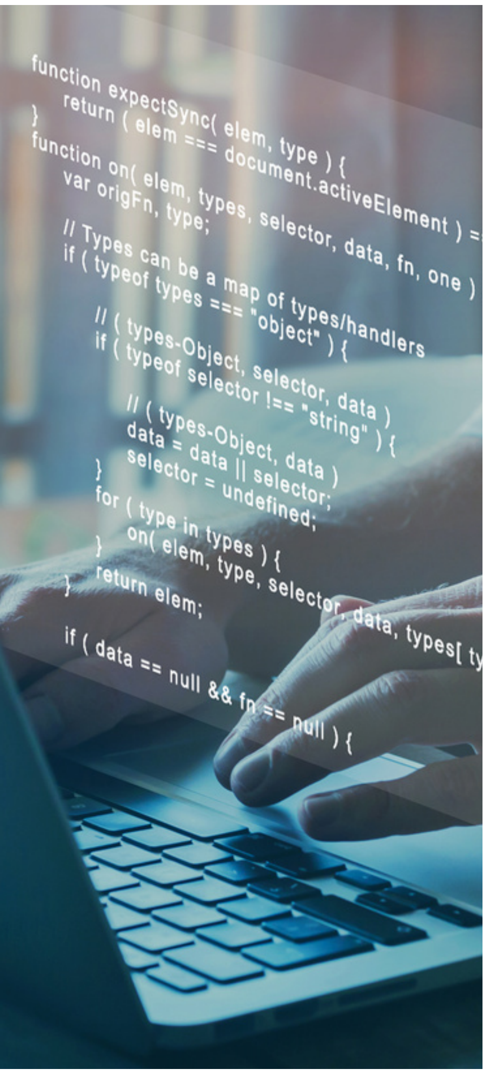
TABLE OF CONTENTS

Chapter 1: INTRODUCTION TO VISUAL BASIC PROGRAMMING

This topic describes button and its functions as well as the development areas in Visual Basic programs.

Chapter 2: VARIABLES AND CONSTANTS

This topic explains about data type, declaration, operator and looping style in Visual Basic program.



```
function expectSync( elem, type ) {
    return ( elem === document.activeElement ) =
}
function on( elem, types, selector, data, fn, one )
    var origFn, type;
    // Types can be a map of types/handlers
    if ( typeof types === "object" ) {
        // ( types-Object, selector, data )
        if ( typeof selector !== "string" ) {
            // ( types-Object, data )
            data = data || selector;
            selector = undefined;
        }
        for ( type in types ) {
            on( elem, type, selector, data, types[ ty
        }
        return elem;
    }
    if ( data == null && fn == null ) {
```

Chapter 3: SIMPLE INTERFACE

This topic demonstrates the steps to develop simple interface using combo box, list box, check box, option button, label, timer, Msg box and progress bar.

Chapter 4: TRAVERSE APPLICATION

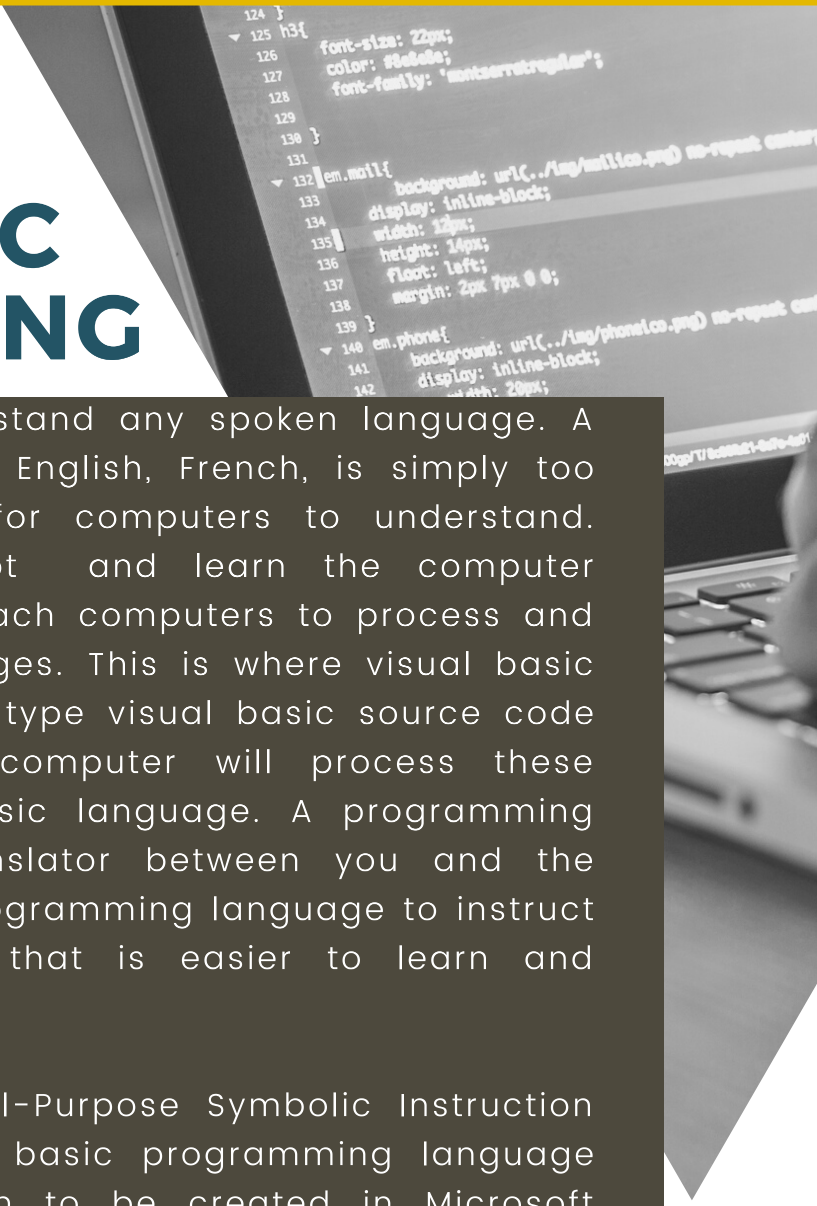
This topic focuses on the development of simple application calculation of latit and depart, calculation latit and depart using Looping method , calculation of linear misclosure. adjusted latit depart using Bowditch method, calculation of coordinate for each station, calculation of area and plot traverse.

Chapter 5: MISSING LINE

This topic demonstrates the steps in developing an application calculation of missing line, calculating bearing and distance from latit and depart and calculating bearing and distance from two know coordinate

CHAPTER 1

INTRODUCTION TO VISUAL BASIC PROGRAMMING



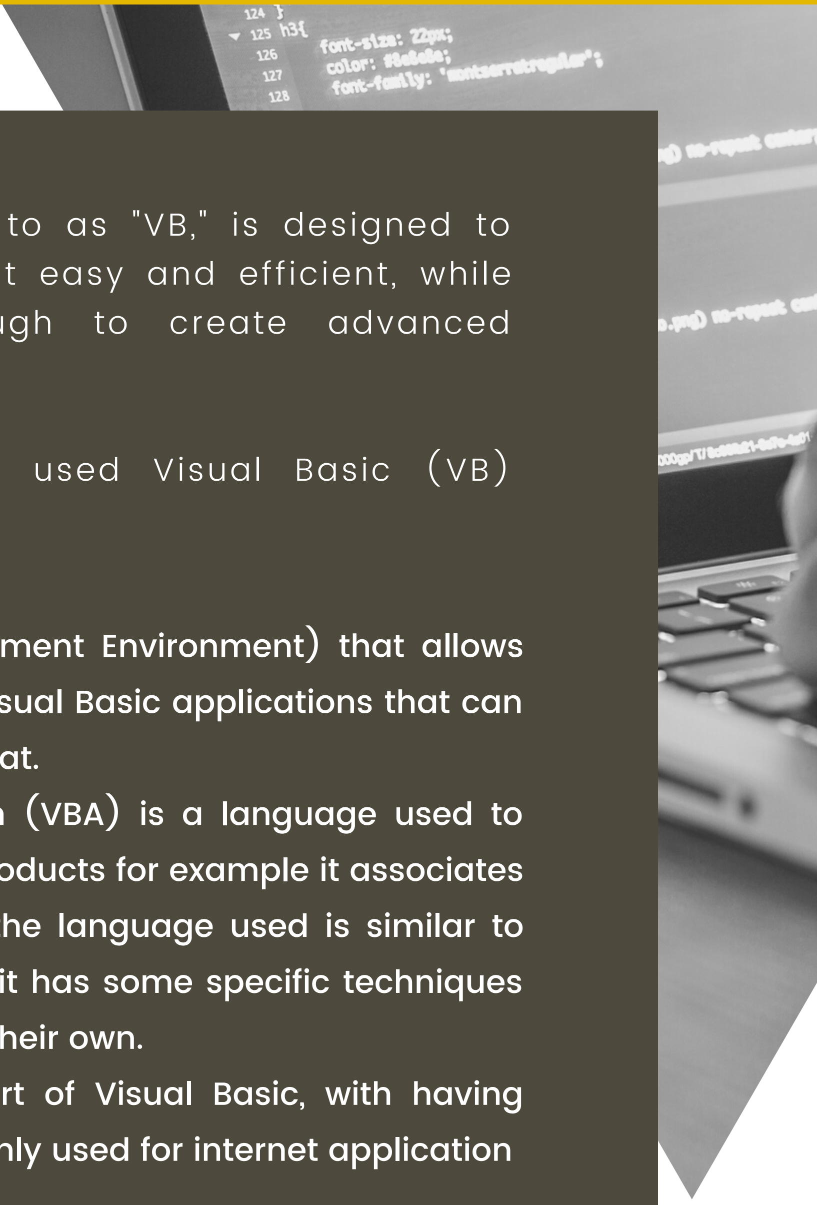
```
}
h3{
    font-size: 22px;
    color: #8e8e8e;
    font-family: 'montserratregular';

}

em.mail{
        background: url(../img/mailico.png) no-repeat center
    display: inline-block;
    width: 12px;
    height: 14px;
    float: left;
    margin: 2px 7px 0 0;
}
em.phone{
        background: url(../img/phoneico.png) no-repeat cen
        display: inline-block;
        width: 20px;
```

VISUAL BASIC PROGRAMMING

- A Computer do not understand any spoken language. A spoken language such as English, French, is simply too general and ambiguous for computers to understand. Therefore, we must adapt and learn the computer language that will help teach computers to process and understand human languages. This is where visual basic comes into it - when you type visual basic source code into the computer, the computer will process these statements into Visual Basic language. A programming language acts as a translator between you and the computer. You can use programming language to instruct the computer in a way that is easier to learn and understand.

- Visual Basic (Beginners All-Purpose Symbolic Instruction Code) is developed from basic programming language that allows an application to be created in Microsoft Windows. It uses an easy-to-understand GUI (Graphic User Interface) method. Visual Basic is one of the simplest and most interesting programming languages because it is user-friendly and looks like a Window environment. Microsoft Windows uses a Graphical User Interface (GUI).

- Visual Basic is a programming language and development environment created by Microsoft. It is an extension of the basic programming language that combines BASIC functions and commands with visual controls. Visual Basic provides a graphical user interface GUI that allows the developer to drag and drop objects into the program as well as manually write program code.



- Visual Basic, also referred to as "VB," is designed to make software development easy and efficient, while still being powerful enough to create advanced programs.
- Here are some commonly used Visual Basic (VB) terms;
  - VB IDE (Integrated Development Environment) that allows you to create standalone Visual Basic applications that can be used via the EXE file format.
  - Visual Basic for Application (VBA) is a language used to integrate Microsoft Office products for example it associates Excel and Word. Although the language used is similar to the standard VB language, it has some specific techniques that need to be learned on their own.
  - VBScript is a language part of Visual Basic, with having limited instructions, commonly used for internet application

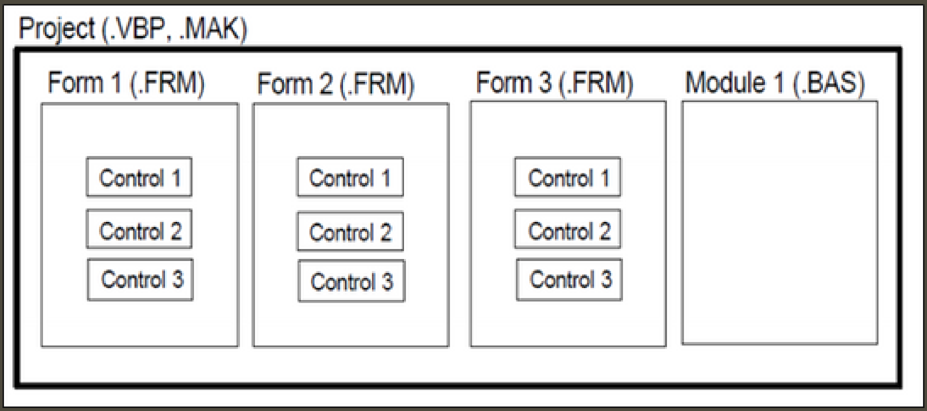
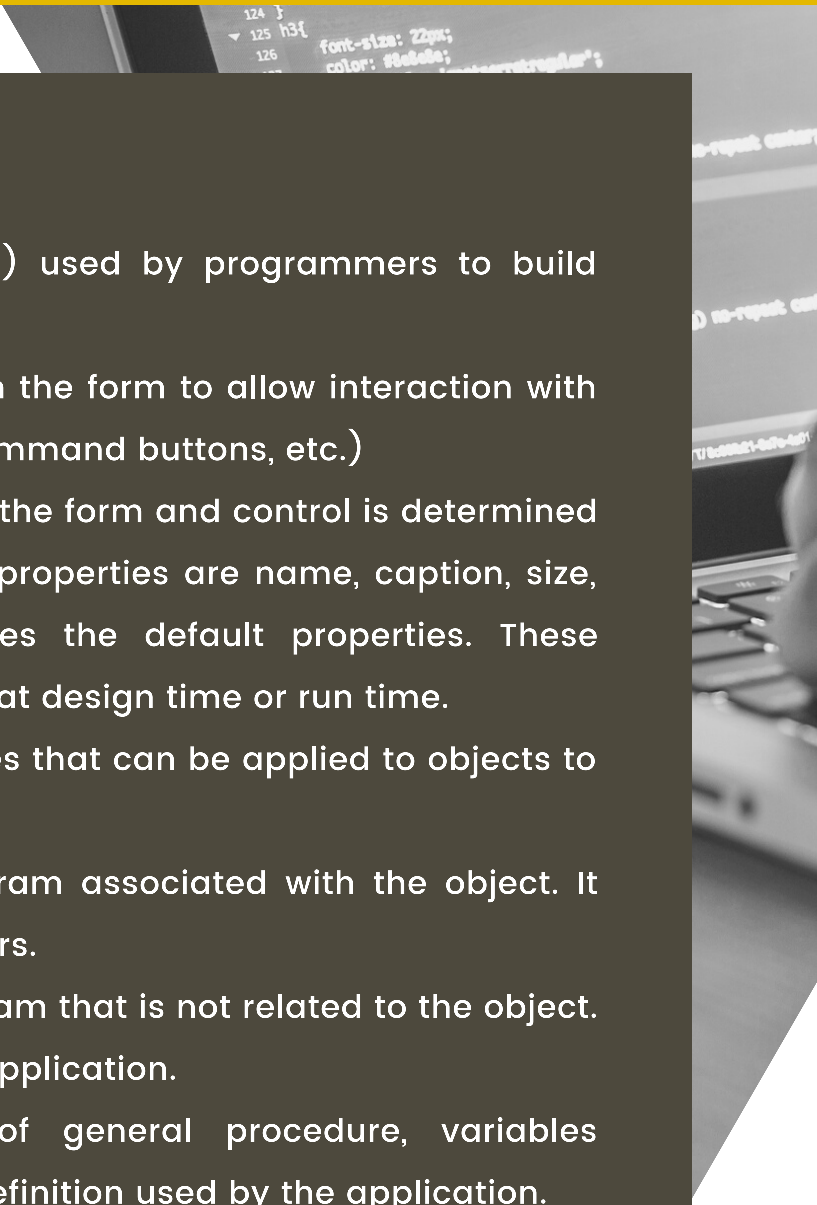


Figure 1: Visual Basic Application Structure



Application (Project) consists of:

- **Forms**: Windows (Windows) used by programmers to build interfaces (interfaces).
- **Controls**: Objects placed on the form to allow interaction with users (text boxes, labels, command buttons, etc.)
- **Properties:** Each feature on the form and control is determined by properties. Examples of properties are name, caption, size, color. Visual Basic specifies the default properties. These properties can be changed at design time or run time.
- **Methods**: Built-in procedures that can be applied to objects to perform certain actions.
- **Event Procedure**: The program associated with the object. It will run when an event occurs.
- **General Procedure**: A program that is not related to the object. It must be charged by the application.
- **Module:** A combination of general procedure, variables declaration and constant definition used by the application.

There are 3 basic steps to Building a Visual Basic Application:

- Draw the user interface.
- Determine the properties (properties) of the object / control.
- Determine / write code on objects / controls.

Visual Basic operates in 3 modes:

- Design Mode: used when designing applications.
- Run Mode: used to run / run applications.
- Break mode: the application stops temporarily and the debugger can be used.

THE DEVELOPMENT ENVIRONMENT

- Learning the ins and outs of the Development Environment before you learn visual basic is similar to learning for a test where you can tell what the functions are all the functions belong and what their purpose is. First, we will start with labeling the development environment.

The view has 3 tabs:

- New: To Create a new project
- Existing: To find an existing project
- Recent: To open a frequently used project

1. Run the Visual Basic 6.0 program. The following display will appear:

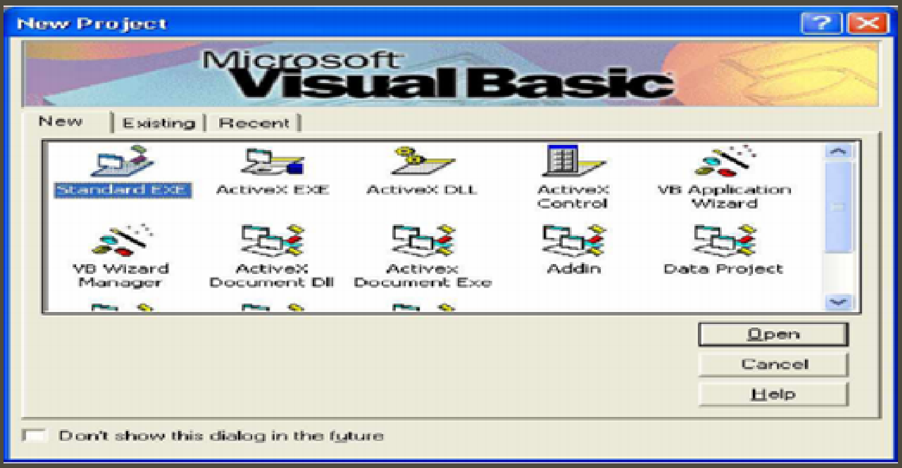


Figure 2: View for New Project

2. To create a new project, select the New tab and select 'Standard EXE' and click Open. The following display will appear.

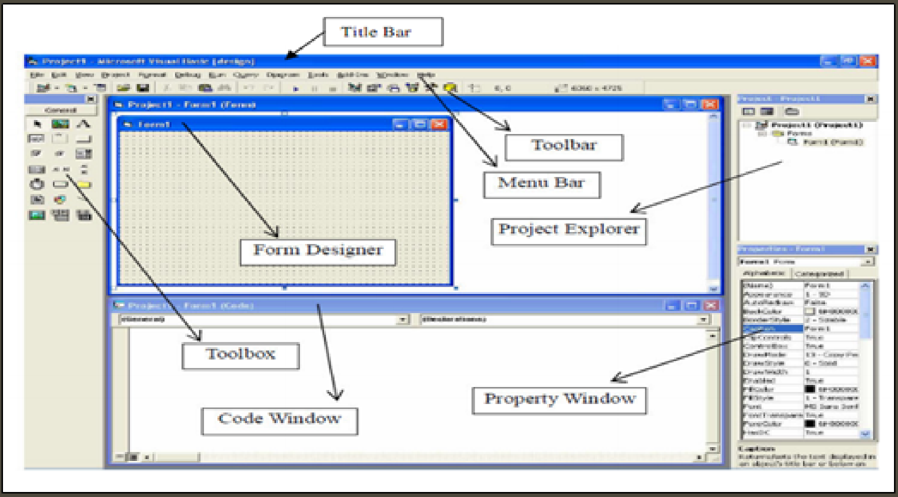


Figure 3: Displays (windows) on the VB interface

3. The Main Window contains the title bar, menu bar and toolbar.

- Title Bar: displays the project name, Visual Basic operation mode and the form in operation.
- Menu Bar: has a drop-down menu from the VB interface.
- Toolbar: has an icon that displays several menu shortcuts.

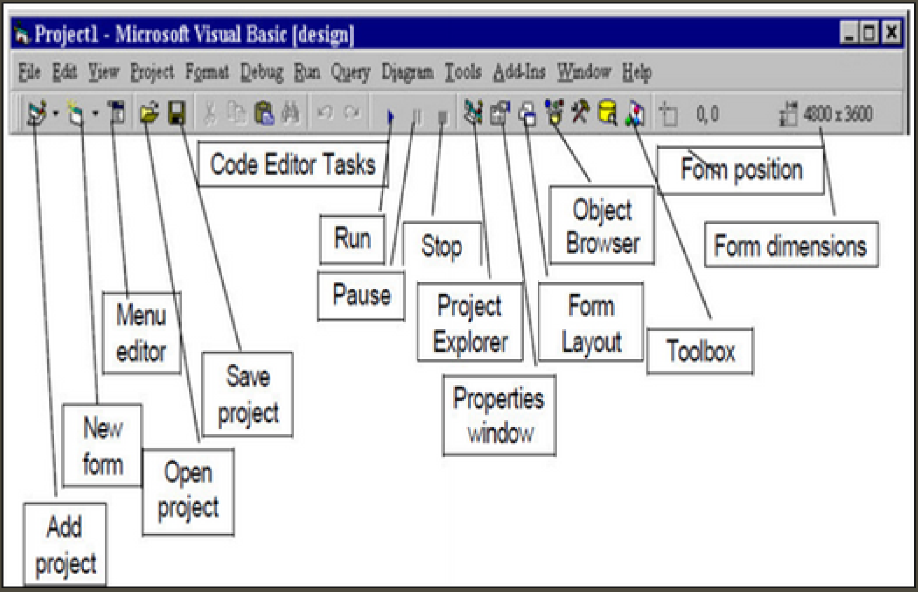


Figure 4: Tools on the VB interface

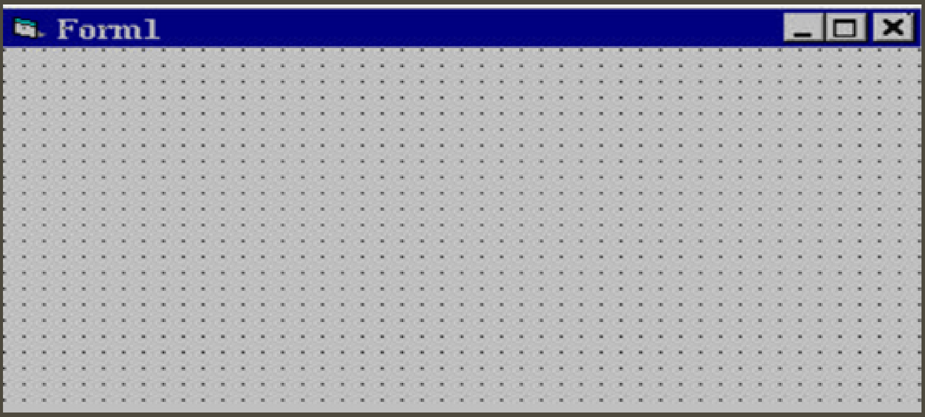


Figure 5: Form (windows) on the VB interface

• **Form Window** - The main view in building VB applications. This is where a design is made.

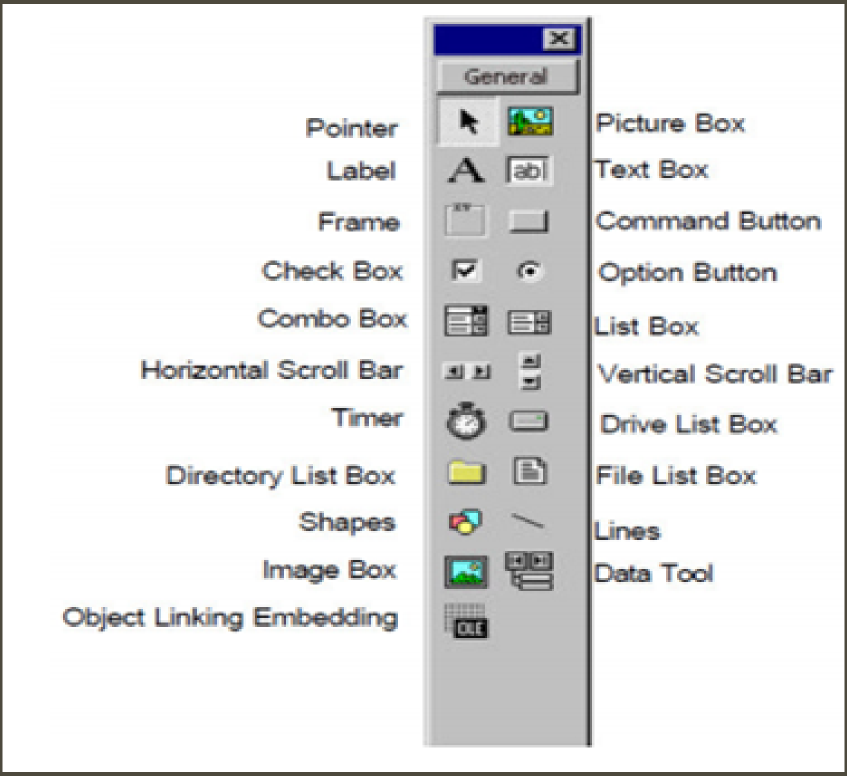
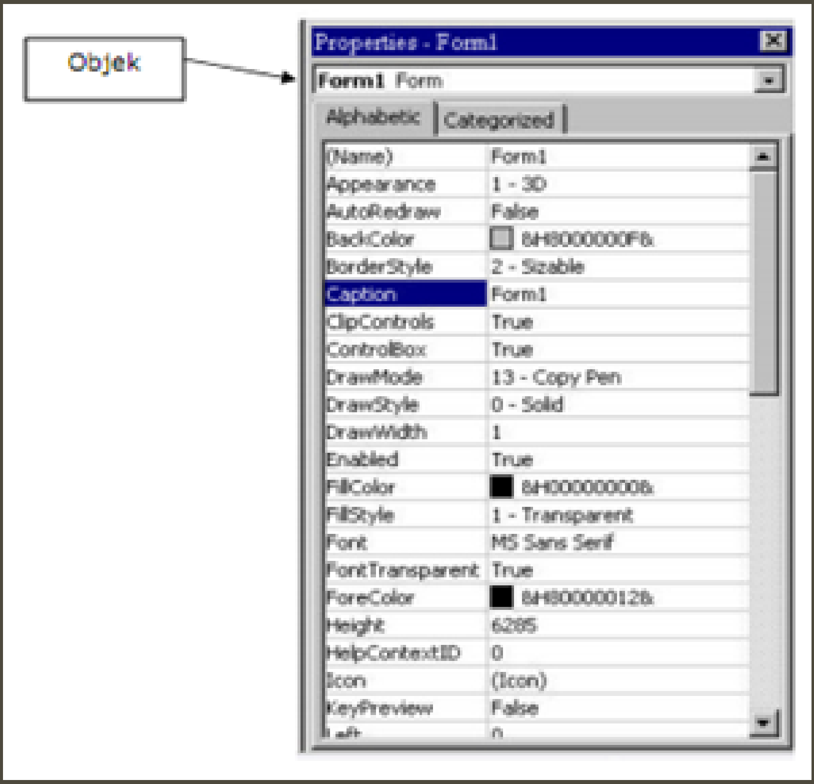


Figure 6: Toolbox - contains objects / controls that will be used in the application



(Name)	Form1
Appearance	1 - 3D
AutoRedraw	False
BackColor	&H8000000F&
BorderStyle	2 - Sizable
Caption	Form1
ClipControls	True
ControlBox	True
DrawMode	13 - Copy Pen
DrawStyle	0 - Solid
DrawWidth	1
Enabled	True
FillColor	&H00000000&
FillStyle	1 - Transparent
Font	MS Sans Serif
FontTransparent	True
ForeColor	&H80000012&
Height	6285
HelpContextID	0
Icon	(Icon)
KeyPreview	False

Figure 7: Properties (windows) on the VB interface

Properties Window - used to set the initial value for the object.

- The drop-down box at the top of the display lists all the objects in the form.
- Two types of display order are available: -According to Alphabetic and -According to Category

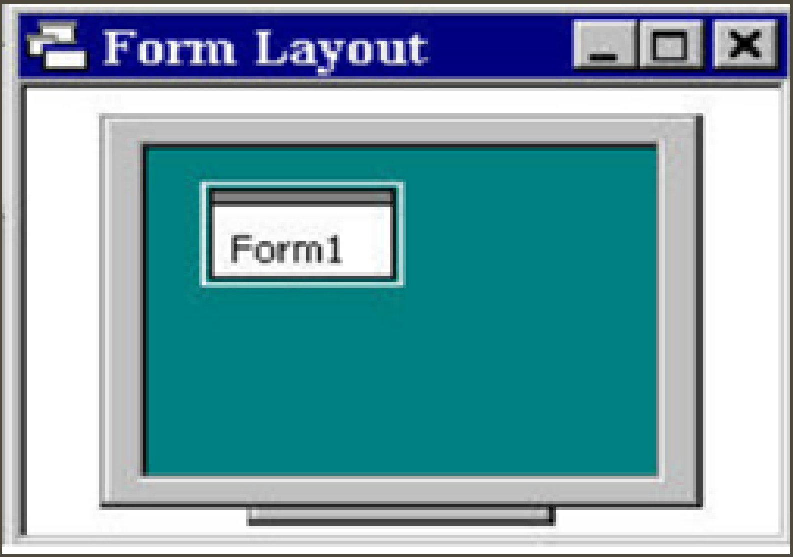
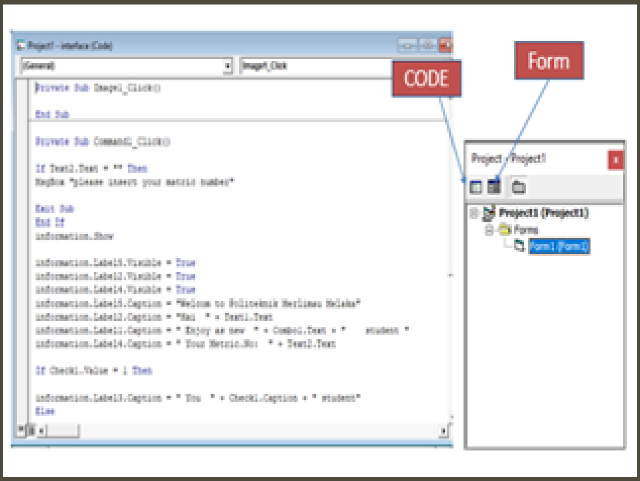


Figure 8: Form Layout (windows) on the VB interface



```
Private Sub Image1_Click()

End Sub

Private Sub Command1_Click()

If Text2.Text = "" Then
MsgBox "please insert your matric number"

Exit Sub
End If
information.Show

information.Label15.Visible = True
information.Label12.Visible = True
information.Label14.Visible = True
information.Label15.Caption = "Welcom to Politeknik Merlimau Melaka"
information.Label12.Caption = "Hai  " + Text1.Text
information.Label11.Caption = " Enjoy as new  " + Combo1.Text + "    student "
information.Label14.Caption = " Your Metric.No:  " + Text2.Text

If Check1.Value = 1 Then

information.Label13.Caption = " You  " + Check1.Caption + " student"
Else
```

Form 9: Project (windows) on the VB interface

•**Project Window** - displays a list of all forms and modules used for the application being built. You can choose to display either the Form View or the Code (program) from the Project.

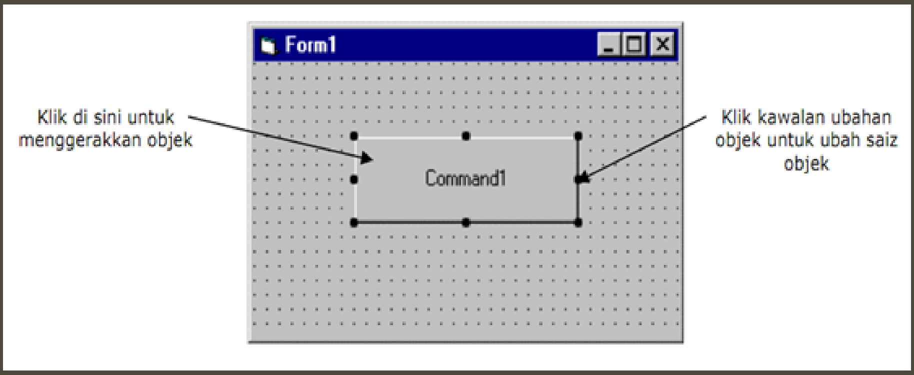
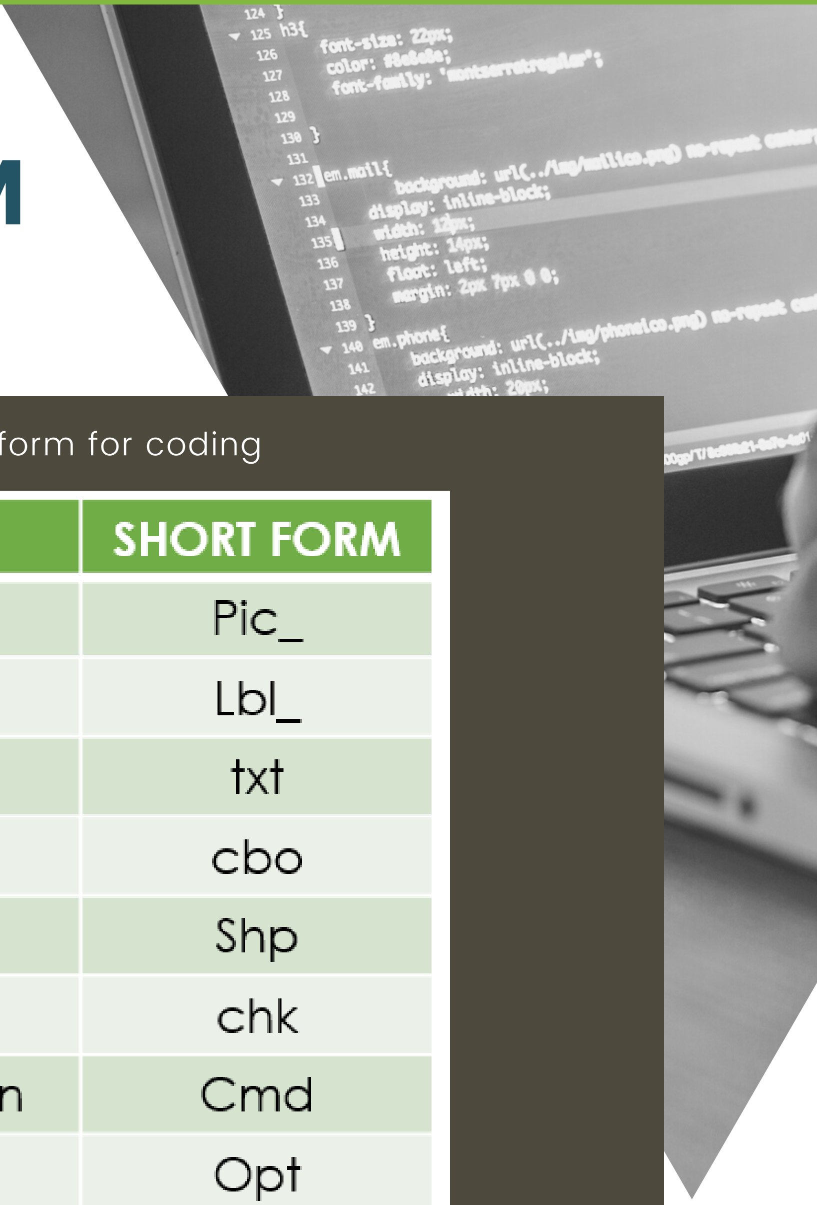

Form1
Klik di sini untuk menggerakkan objek
Command1
Klik kawalan ubahan objek untuk ubah saiz objek

Figure 10: Placing objects/controls on forms



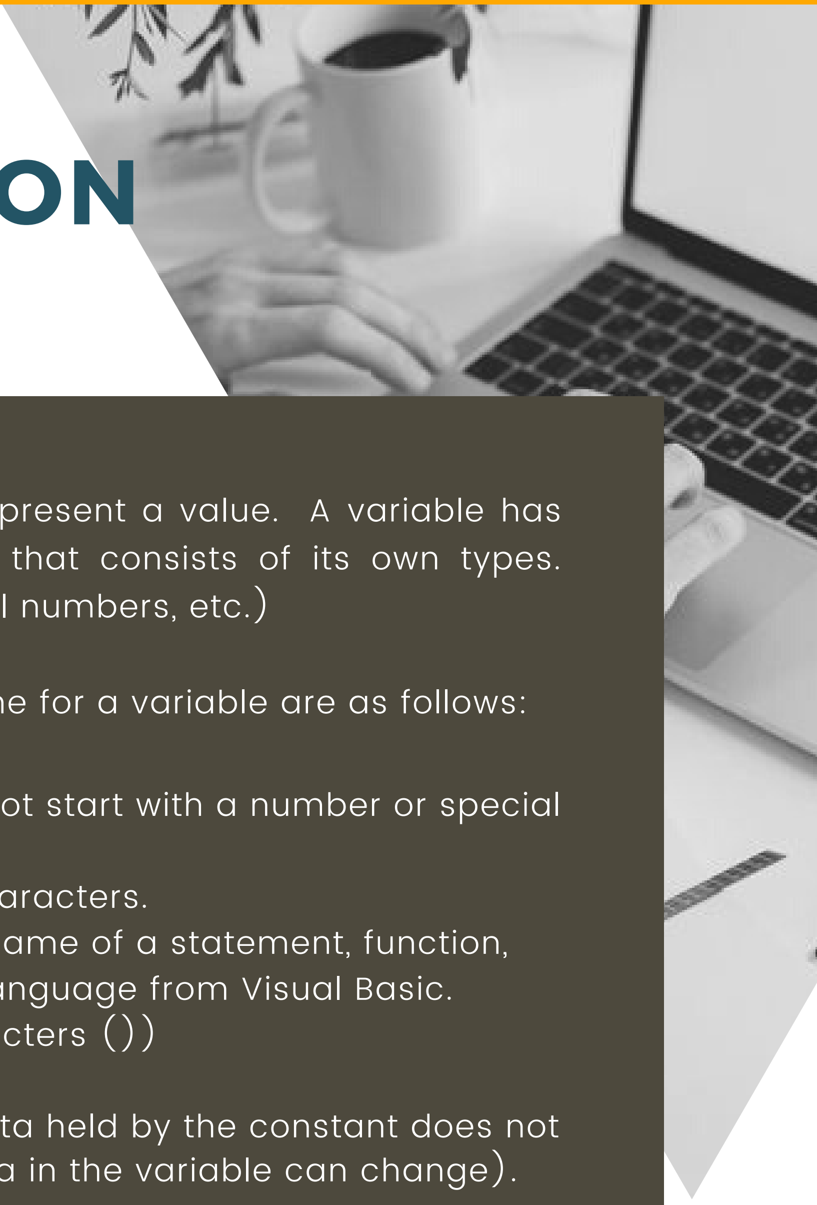
SHORT FORM FOR CODING

Table 1: Short form for coding

OBJECT	SHORT FORM
Picture box	Pic_
label	Lbl_
Text Box	txt
Combo box	cbo
shape	Shp
Check box	chk
Command button	Cmd
Option button	Opt
data	dat
Directory list box	Dir
Driver list box	Drv
File list box	Fil
frame	frm
List box	lst
Vertical scroll bar	vsb
timer	tmr

CHAPTER 2

VARIABLES AND CONSTANTS



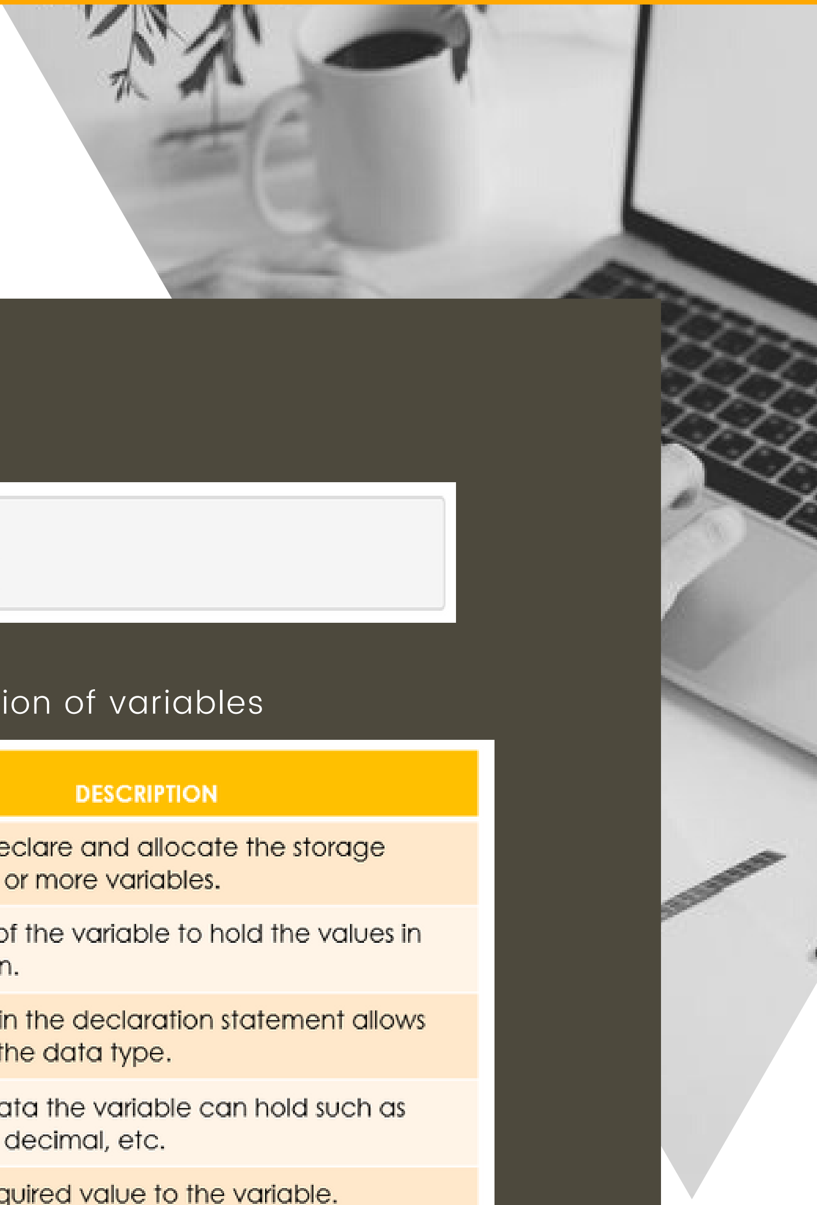
INTRODUCTION

- A variable is a name used to represent a value. A variable has its own name and stores data that consists of its own types. (Numbers, letters, words, decimal numbers, etc.)
- The conditions for forming a name for a variable are as follows:
  - Must start with a letter (cannot start with a number or special character)
  - Cannot be more than 255 characters.
  - Cannot be the same as the name of a statement, function, method, object or part of a language from Visual Basic.
  - No spaces and special characters ())

Constant: the value of the data held by the constant does not change (the value of the data in the variable can change).

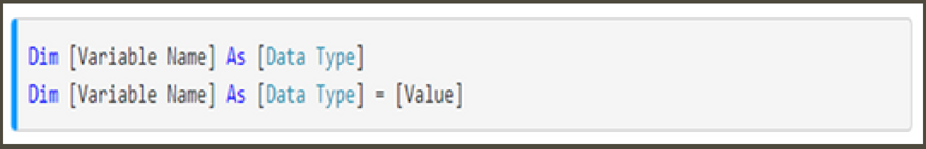
Example:

```
Const A = 10
```



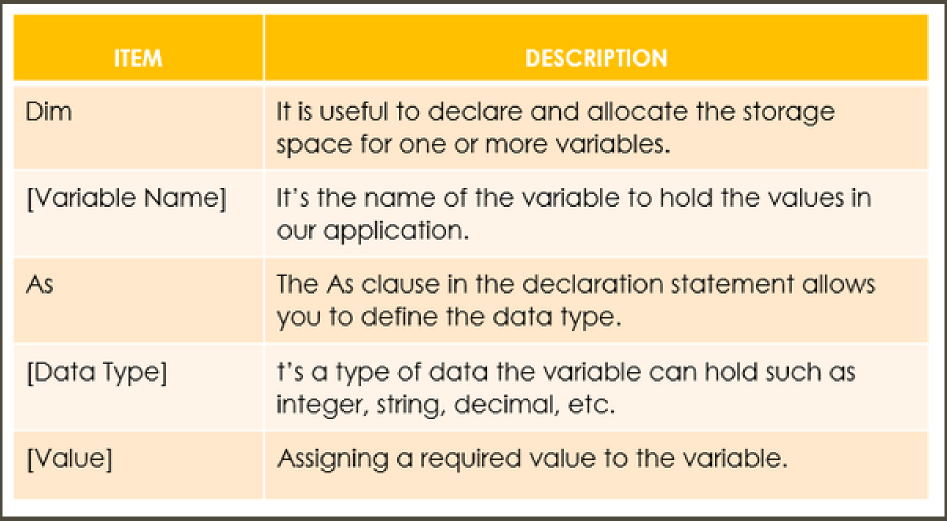
DATA TYPE

• Defining Visual Basic Data Types



```
Dim [Variable Name] As [Data Type]
Dim [Variable Name] As [Data Type] = [Value]
```

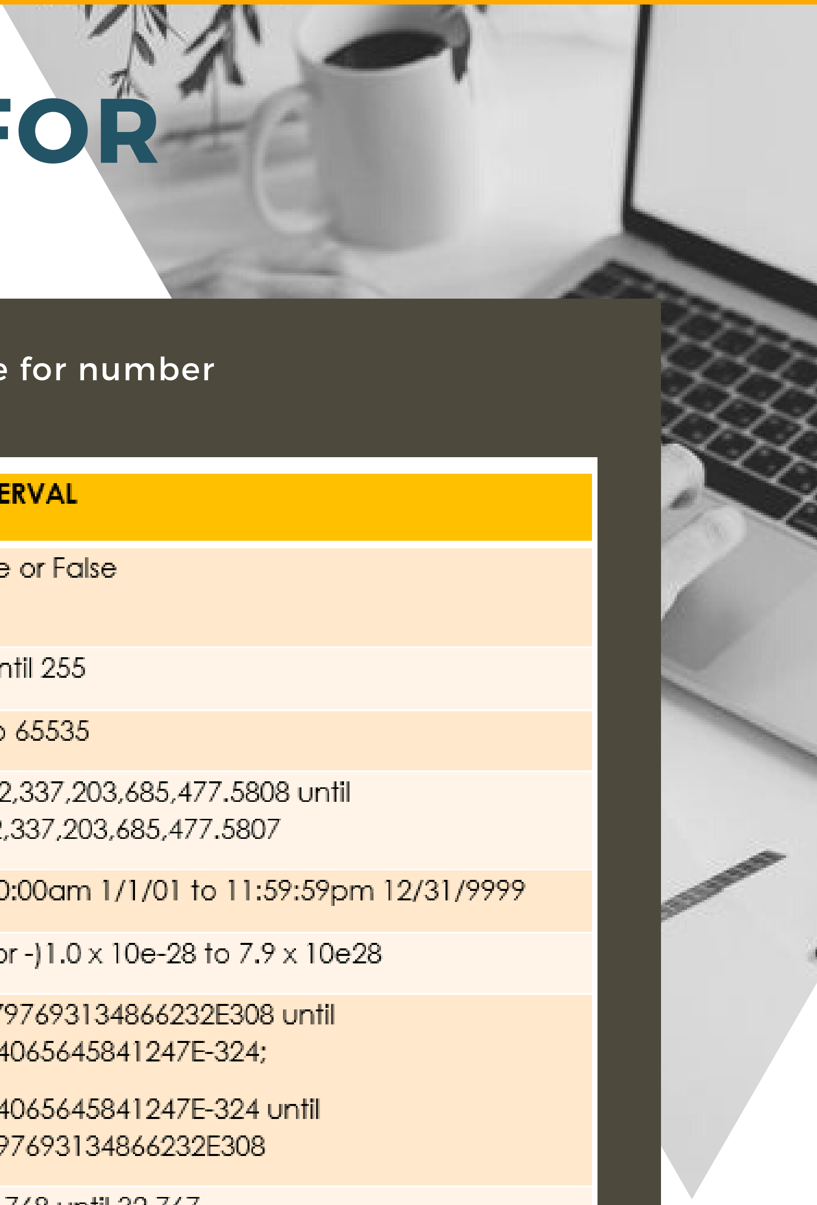
Table 2: Definition of variables



ITEM	DESCRIPTION
Dim	It is useful to declare and allocate the storage space for one or more variables.
[Variable Name]	It's the name of the variable to hold the values in our application.
As	The As clause in the declaration statement allows you to define the data type.
[Data Type]	t's a type of data the variable can hold such as integer, string, decimal, etc.
[Value]	Assigning a required value to the variable.

Data type refers to the data stored by a variable. It can be divided into 3 :

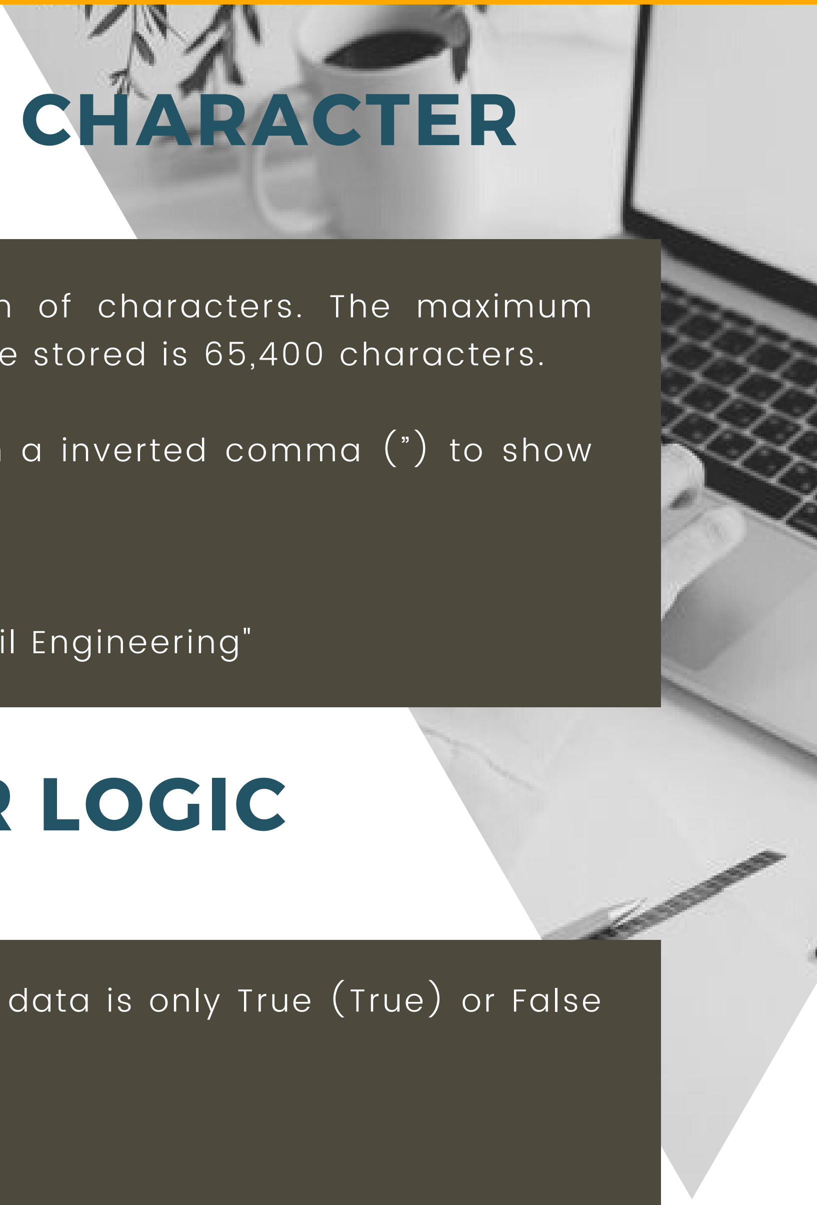
a) Data type for numbers
b) Data type for character (String)
c) Data Type for Logic (Boolean)



DATA TYPE FOR NUMBERS

Table 3: Data type for number

DATA TYPE	STORAGE SIZE	INTERVAL
Boolean	It depends on the Platform	True or False
Byte	1 byte	0 until 255
Char	2 bytes	0 to 65535
Currency	8 byte	-922,337,203,685,477.5808 until 922,337,203,685,477.5807
Date	8 bytes	0:00:00am 1/1/01 to 11:59:59pm 12/31/9999
Decimal	16 bytes	(+ or -)1.0 x 10e-28 to 7.9 x 10e28
Double	8 byte	-1.797693134866232E308 until 4,94065645841247E-324; 4,94065645841247E-324 until 1.797693134866232E308
Integer	2 byte	-32,768 until 32,767
Long	4 byte	-2,147,483,648 until 2,147,483,647
Single	4 byte	-3,402823E38 until -1,401298E-45; 1,401298e-45 until 3,402823E38
String	Depends on Platform	0 to approximately 2 billion Unicode characters
UInteger	4 bytes	0 to 4,294,967,295
ULong	8 bytes	0 to 18,446,744,073,709,551,615 (1.8...E+19 †)
UShort	2 bytes	0 to 65,535
User-Defined	Depends on Platform	Each member of the structure has a range determined by its data type and independent of the ranges of the other members



DATA TYPE FOR CHARACTER (STRING)

Used to store data in the form of characters. The maximum number of characters that can be stored is 65,400 characters.

This type of data is written with a inverted comma (") to show where it begins and ends.

Example:

```
Dim Name As String
Name = "Department of Civil Engineering"
```

DATA TYPE FOR LOGIC (BOOLEAN)

Used to test logic. Value for this data is only True (True) or False (False).

Example:

```
Dim New As Boolean
New = True
```

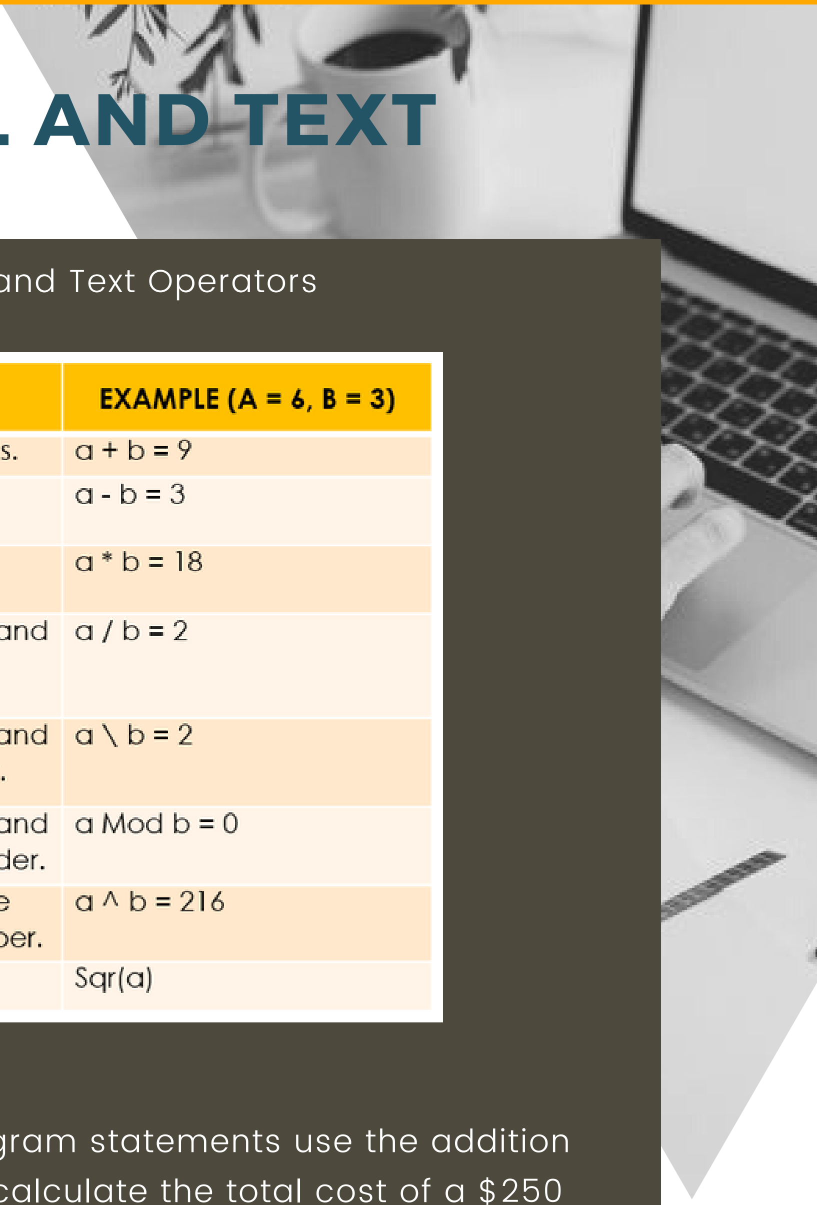
DATA TYPE FOR LOGIC (BOOLEAN)

The declaration of variables can be made as follows:

```
Public <variable name> As <Data Type>
Or
Dim <variable name> As <Data Type>
```

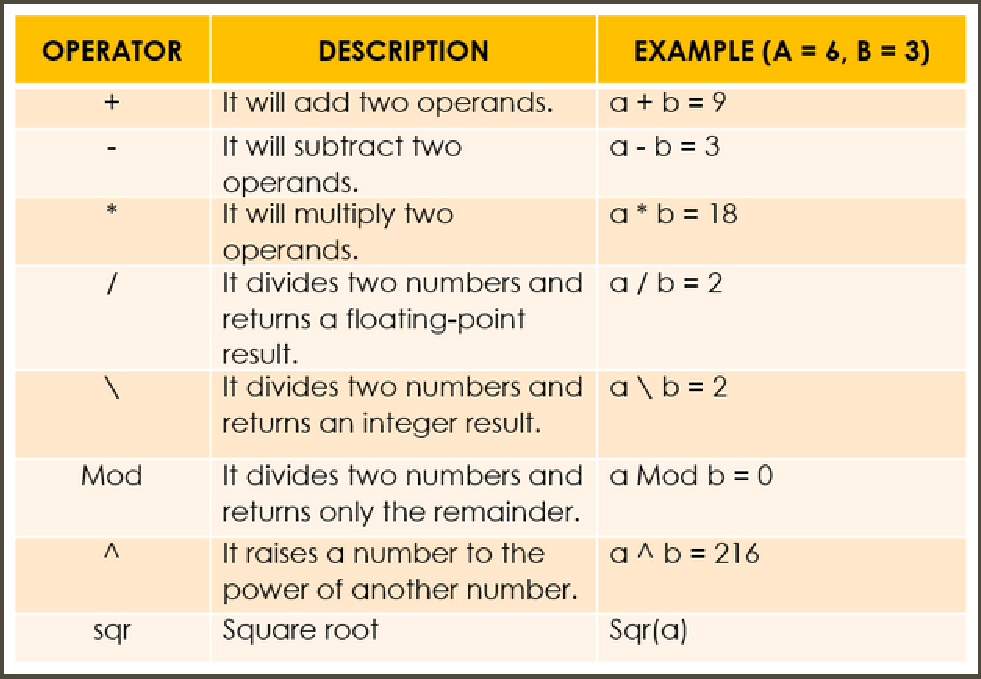
Example:

```
Public Number1 As Integer
Dim Name As String
```



MATHEMATICAL AND TEXT OPERATORS

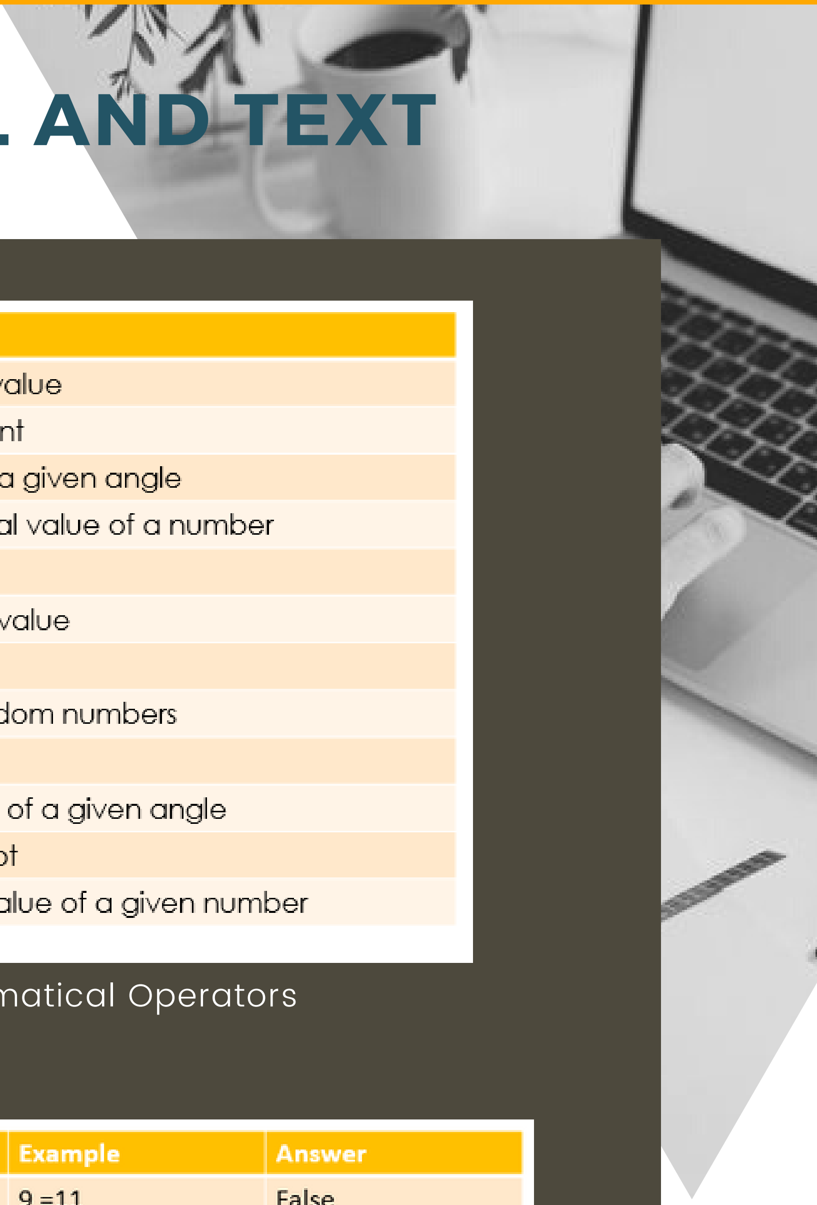
Table 4: Mathematical and Text Operators



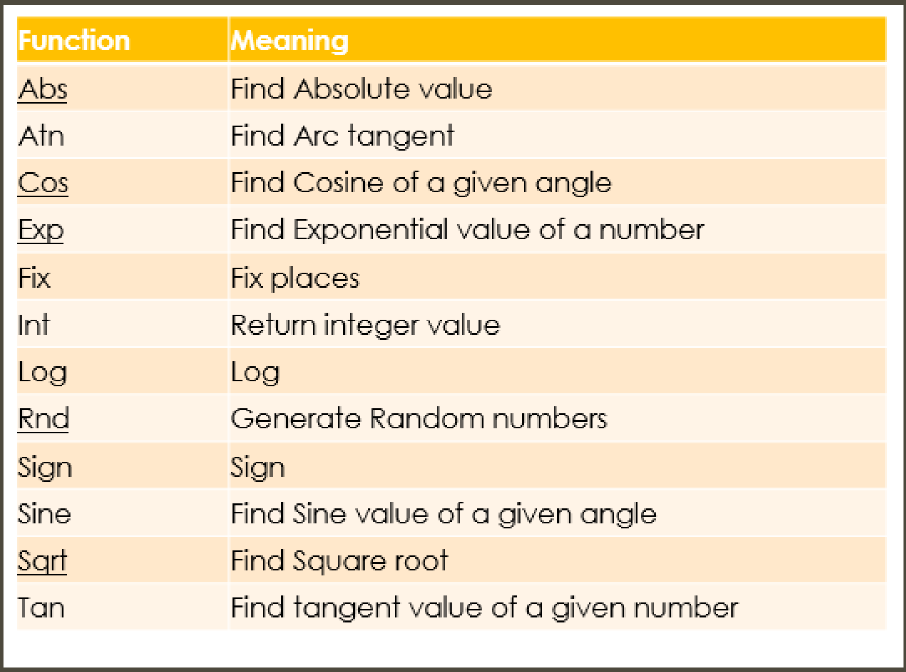
OPERATOR	DESCRIPTION	EXAMPLE (A = 6, B = 3)
+	It will add two operands.	a + b = 9
-	It will subtract two operands.	a - b = 3
*	It will multiply two operands.	a * b = 18
/	It divides two numbers and returns a floating-point result.	a / b = 2
\	It divides two numbers and returns an integer result.	a \ b = 2
Mod	It divides two numbers and returns only the remainder.	a Mod b = 0
^	It raises a number to the power of another number.	a ^ b = 216
sqr	Square root	Sqr(a)

• For example, the following program statements use the addition and multiplication operators to calculate the total cost of a $250 bicycle including 8.1% sales tax:

```
Dim BicycleCost, TotalPrice
Const SalesTaxRate = 0.081
BicycleCost = 250
TotalPrice = BicycleCost * SalesTaxRate + BicycleCost
```

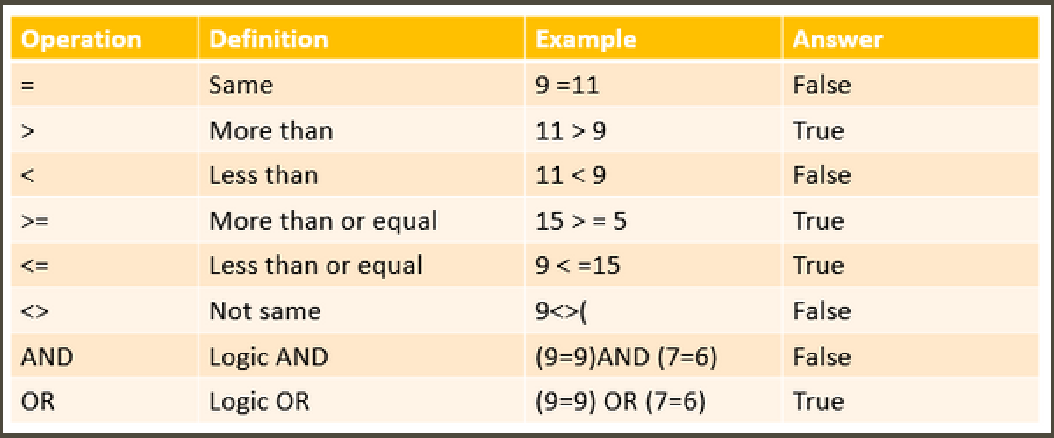


MATHEMATICAL AND TEXT OPERATORS



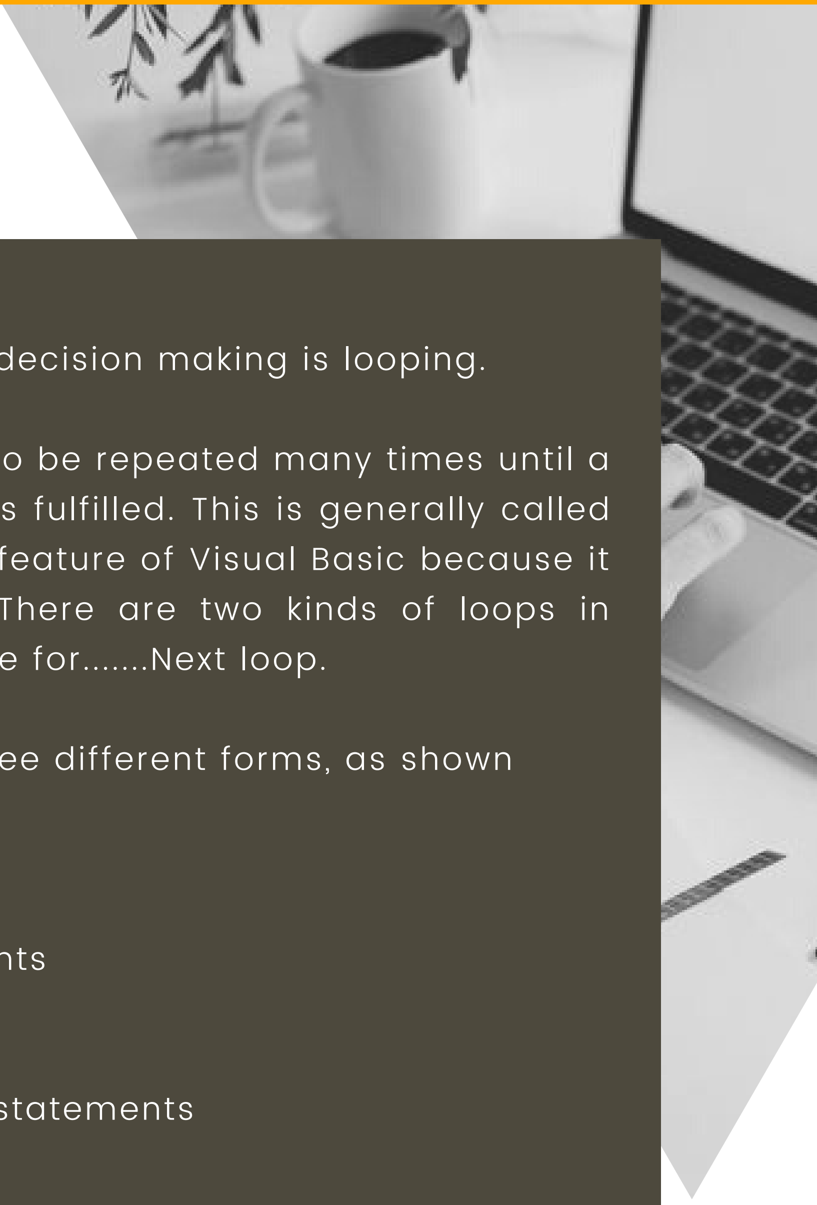
Function	Meaning
Abs	Find Absolute value
Atn	Find Arc tangent
Cos	Find Cosine of a given angle
Exp	Find Exponential value of a number
Fix	Fix places
Int	Return integer value
Log	Log
Rnd	Generate Random numbers
Sign	Sign
Sine	Find Sine value of a given angle
Sqrt	Find Square root
Tan	Find tangent value of a given number

Figure 11: Mathematical Operators



Operation	Definition	Example	Answer
=	Same	9 =11	False
>	More than	11 > 9	True
<	Less than	11 < 9	False
>=	More than or equal	15 > = 5	True
<=	Less than or equal	9 < =15	True
<>	Not same	9<>(	False
AND	Logic AND	(9=9)AND (7=6)	False
OR	Logic OR	(9=9) OR (7=6)	True

Figure 12: Logical Operators



LOOPING

Another procedure that involves decision making is looping.

Visual Basic allows a procedure to be repeated many times until a condition or a set of conditions is fulfilled. This is generally called looping. Looping is a very useful feature of Visual Basic because it makes repetitive works easier. There are two kinds of loops in Visual Basic, the Do...Loop and the for.......Next loop.

The Do Loop statements have three different forms, as shown below:

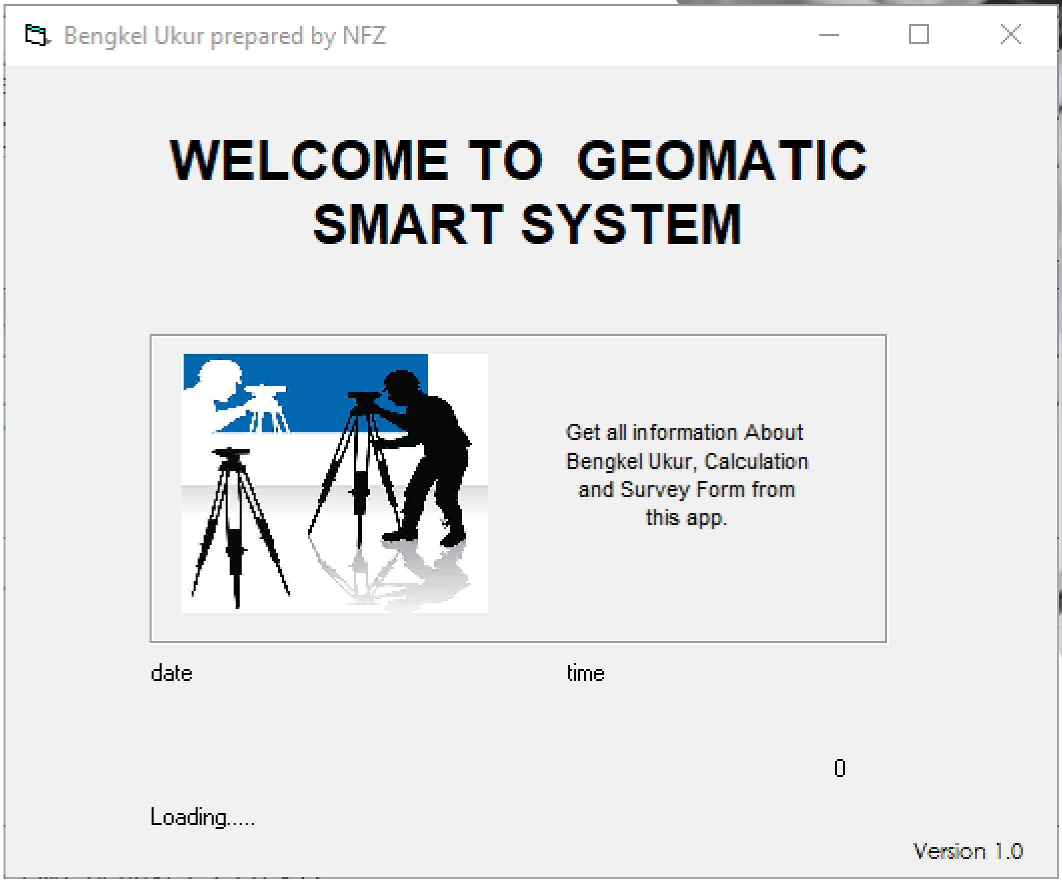
```
Do While condition
Block of one or more VB statements
Loop
b)  Do
          Block of one or more VB statements
     Loop While condition

c)   Do Until condition
           Block of one or more VB statements
      Loop

d)   Do
          Block of one or more VB statements
      Loop Until condition
```

CHAPTER 3

SIMPLE INTERFACE

- Set date and time using time
- Progress bar
- Msg Box
- Combo box
- List box
- Check box
- Option button
- image

Simple Interface


Bengkel Ukur prepared by NFZ

WELCOME TO GEOMATIC SMART SYSTEM

Get all information About Bengkel Ukur, Calculation and Survey Form from this app.

date

time

0

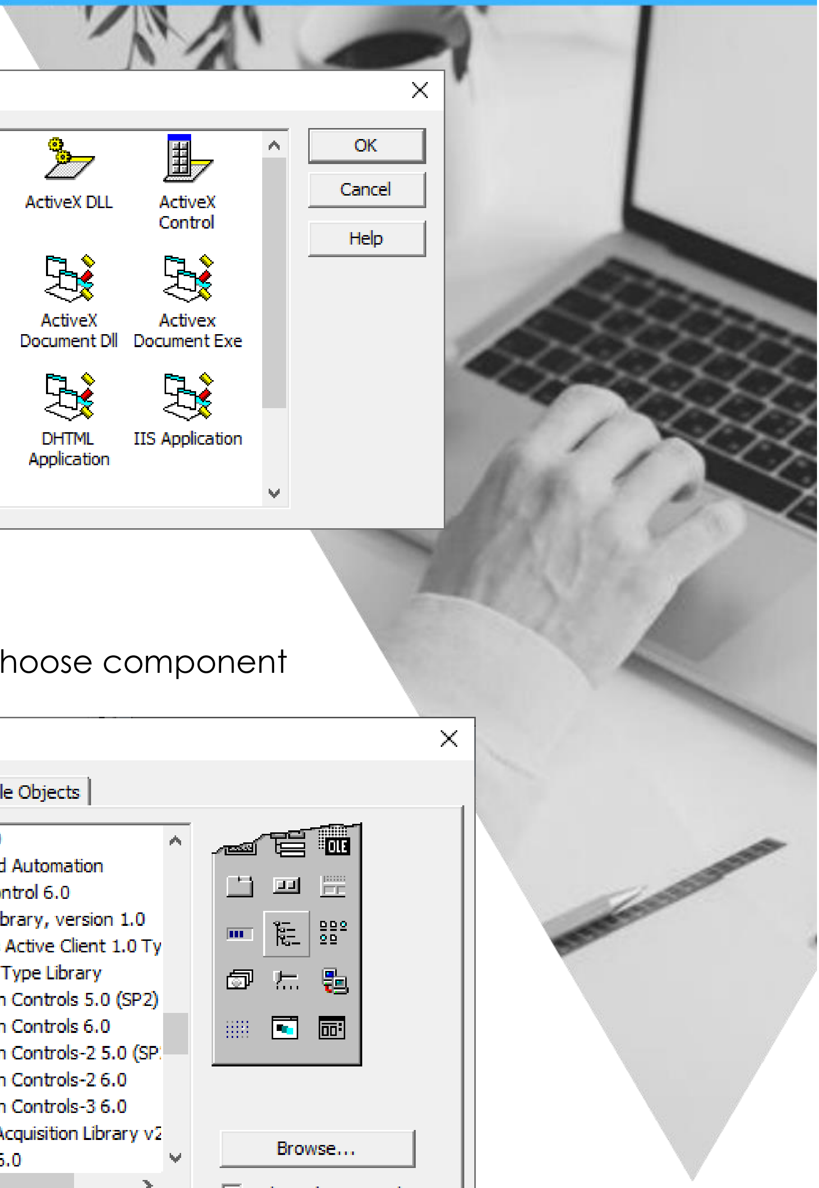
Loading.....

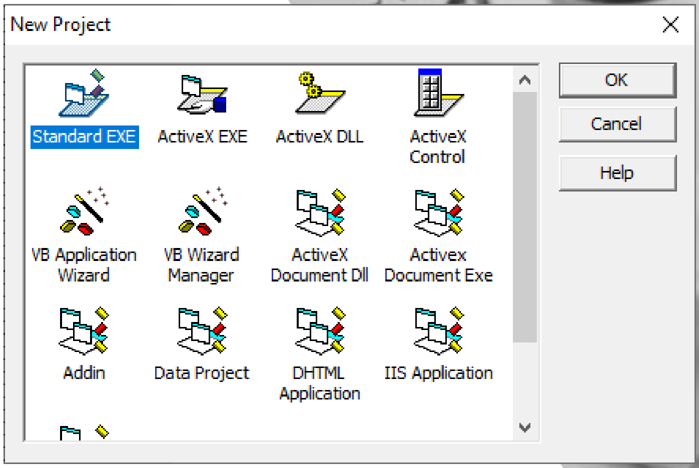
Version 1.0

Figure 13: Simple interface

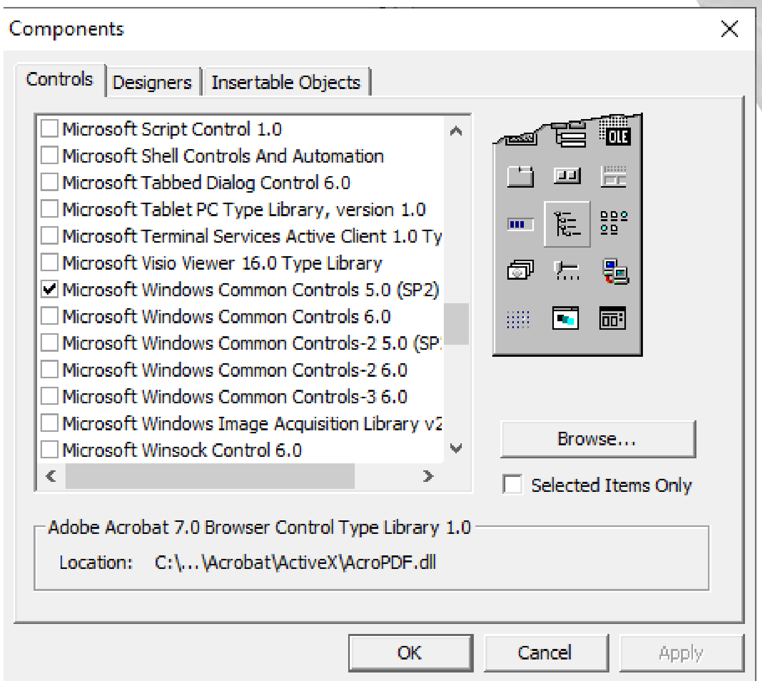
Procedure

1. Start a new project.
2. Choose Standard EXE from the New Project dialog box.





3. Right click at toolbox and choose component

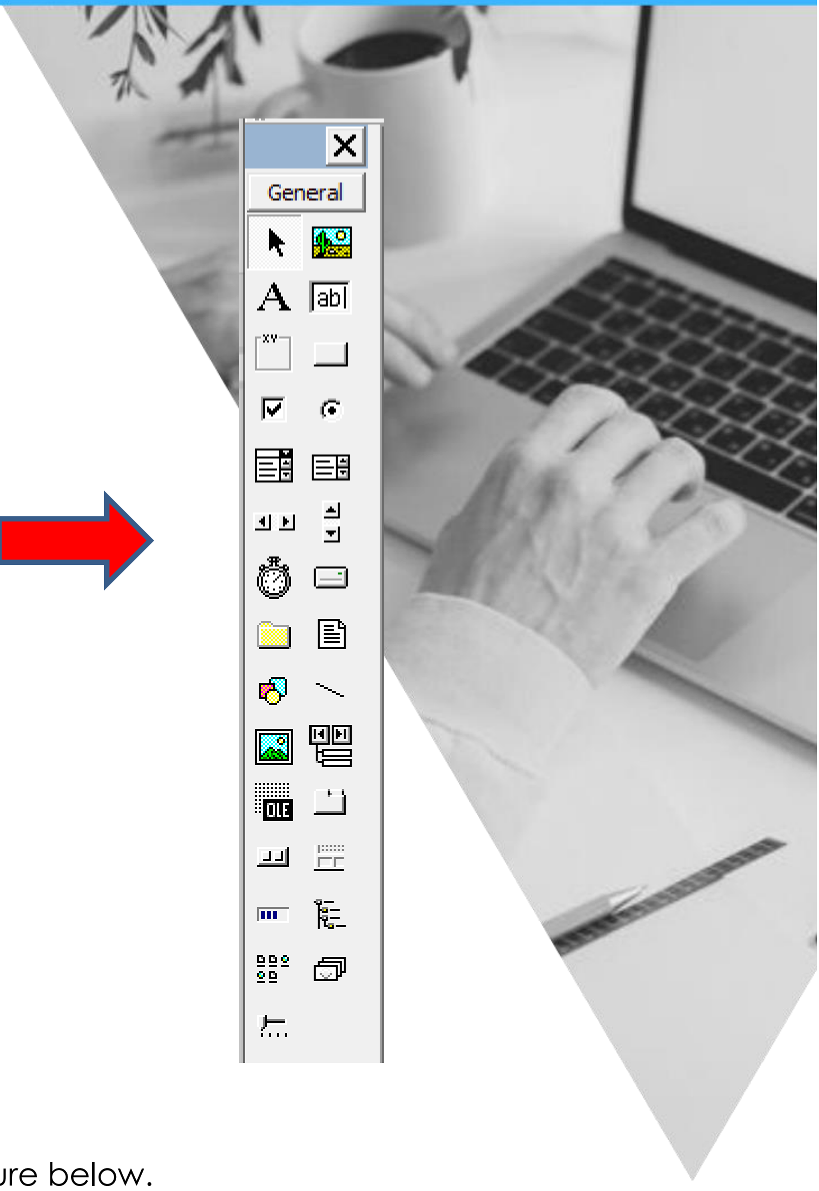


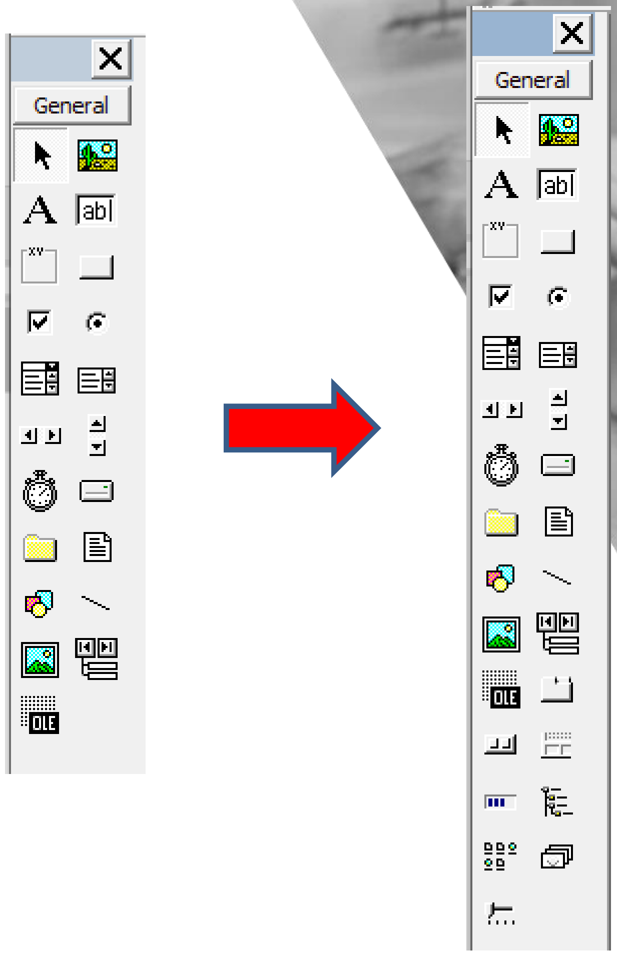
- [ ] Microsoft Script Control 1.0
- [ ] Microsoft Shell Controls And Automation
- [ ] Microsoft Tabbed Dialog Control 6.0
- [ ] Microsoft Tablet PC Type Library, version 1.0
- [ ] Microsoft Terminal Services Active Client 1.0 Ty
- [ ] Microsoft Visio Viewer 16.0 Type Library
- [x] Microsoft Windows Common Controls 5.0 (SP2)
- [ ] Microsoft Windows Common Controls 6.0
- [ ] Microsoft Windows Common Controls-2 5.0 (SP
- [ ] Microsoft Windows Common Controls-2 6.0
- [ ] Microsoft Windows Common Controls-3 6.0
- [ ] Microsoft Windows Image Acquisition Library v2
- [ ] Microsoft Winsock Control 6.0

- [ ] Selected Items Only

Adobe Acrobat 7.0 Browser Control Type Library 1.0
Location: C:\...\Acrobat\ActiveX\AcroPDF.dll

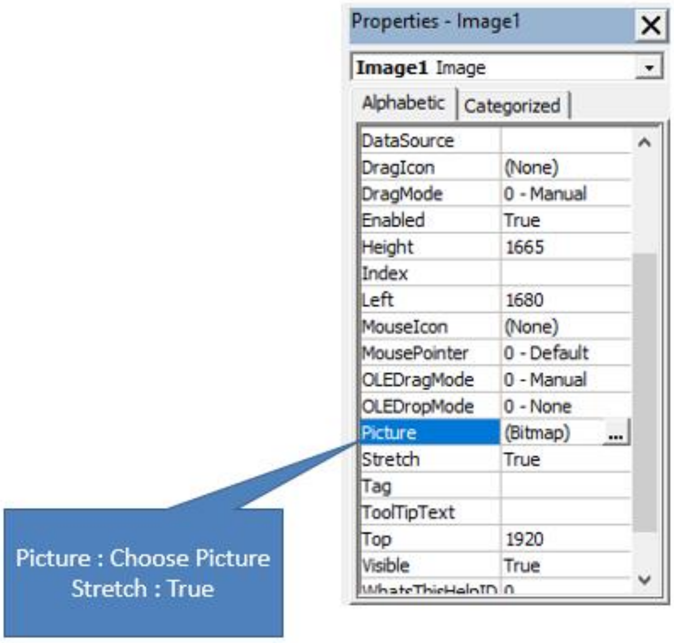
4. Select Microsoft Window Common Control 5.0





5. Insert all object base on figure below.
6. Rename object name at properties

7. At image properties, set stretch True.



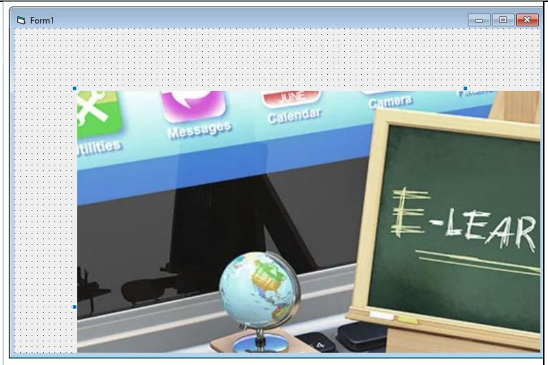
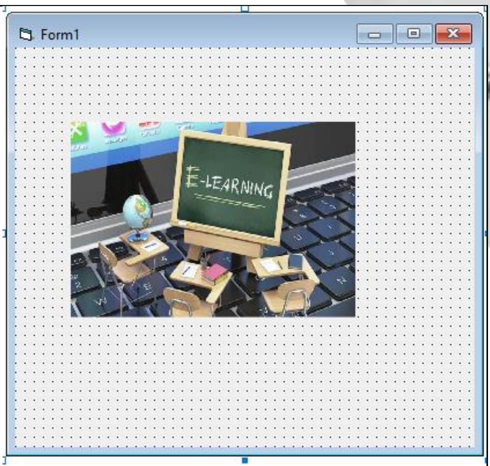
Properties - Image1

Image1 Image

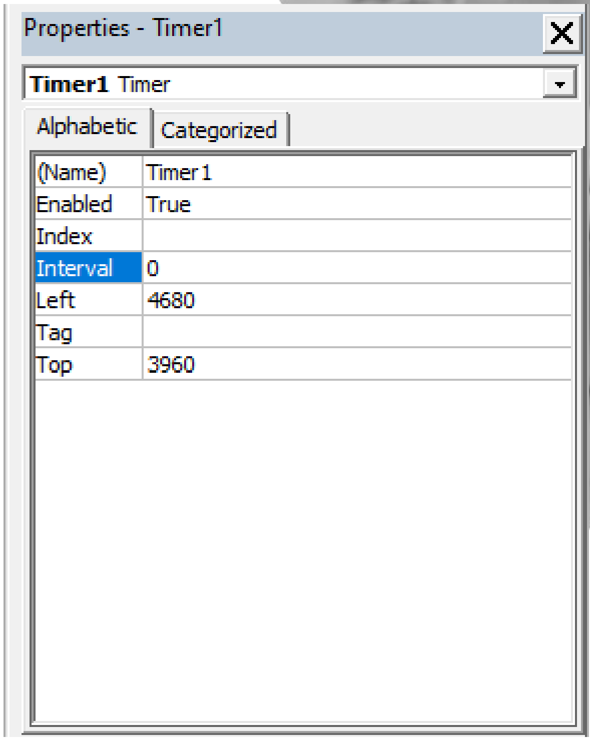
Alphabetic | Categorized

```
DataSource
DragIcon        (None)
DragMode        0 - Manual
Enabled         True
Height          1665
Index
Left            1680
MouseIcon       (None)
MousePointer    0 - Default
OLEDragMode     0 - Manual
OLEDropMode     0 - None
Picture         (Bitmap)
Stretch         True
Tag
ToolTipText
Top             1920
Visible         True
WhatsThisHelpID 0
```

Picture : Choose Picture
Stretch : True

	
Stretch: False Image size: Based on size of picture	Stretch: True Image size: Based on image box size

8. At timer properties, insert interval between 200 to 250.



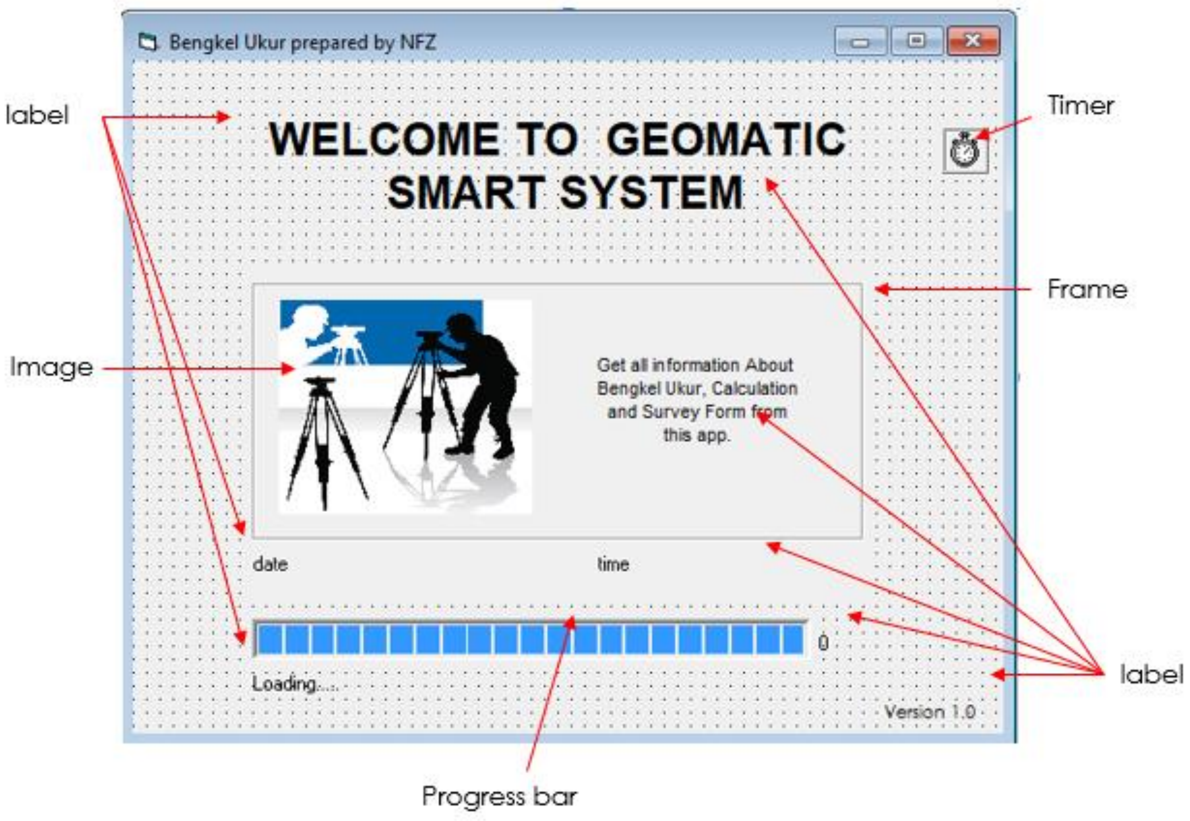
Properties - Timer1

Timer1 Timer

Alphabetic | Categorized

(Name)	Timer1
Enabled	True
Index	
Interval	0
Left	4680
Tag	
Top	3960

9. Interface has been developed.


Bengkel Ukur prepared by NFZ
label
WELCOME TO GEOMATIC SMART SYSTEM
Timer
Frame
Image
Get all information About Bengkel Ukur, Calculation and Survey Form from this app.
date
time
0
Loading....
Version 1.0
label
Progress bar

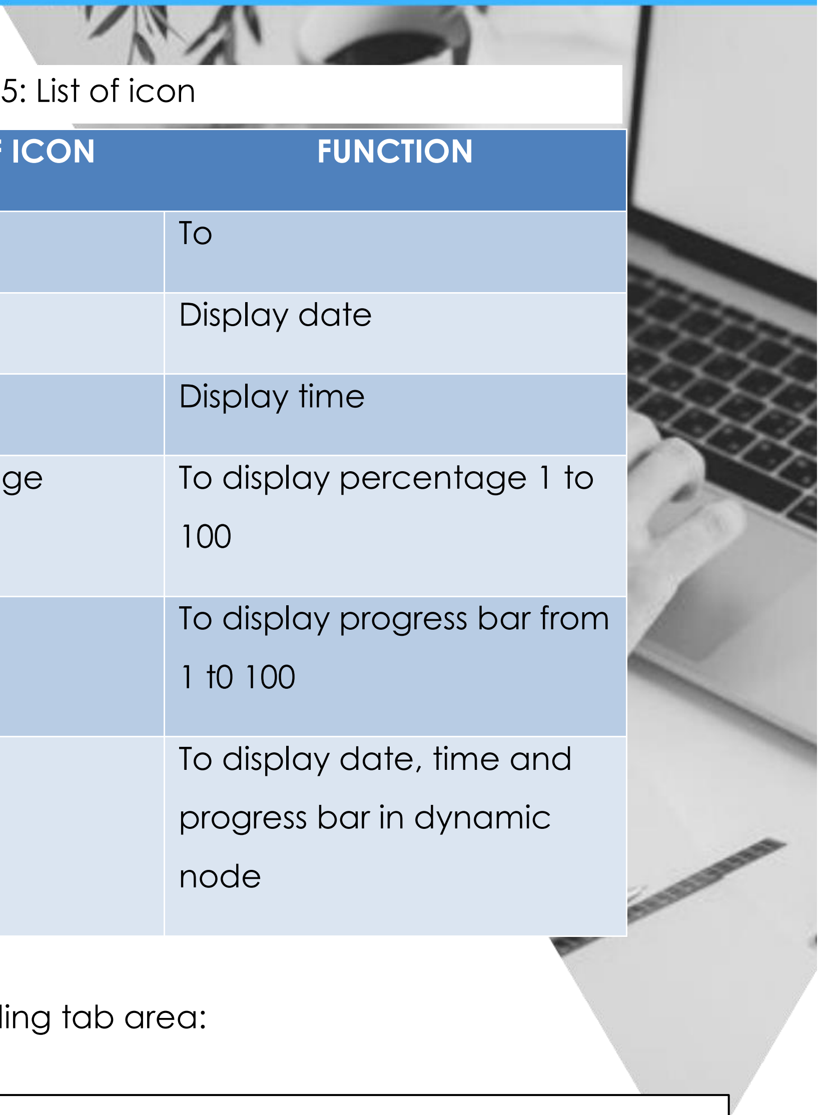
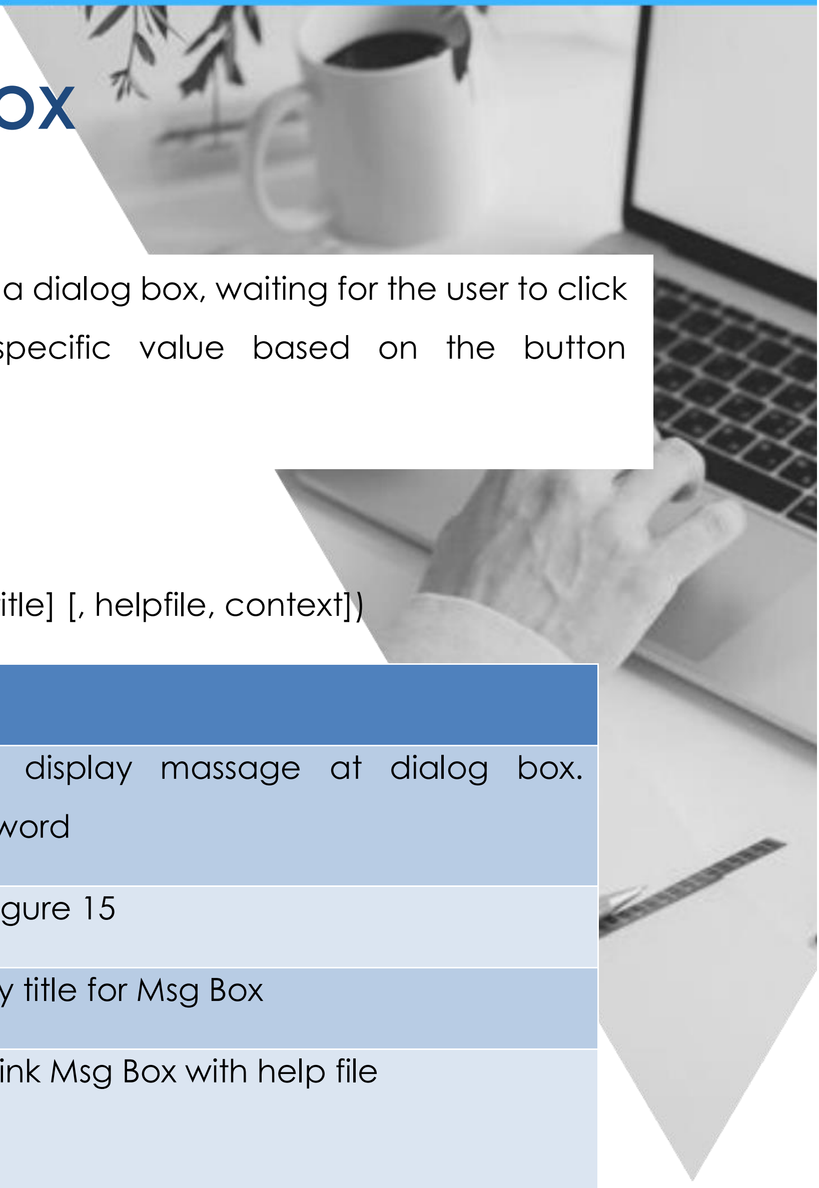


Table 5: List of icon

COMPONENT	NAME OF ICON	FUNCTION
Timer	timer	To
Label	lbl_date	Display date
	lbl_time	Display time
	lbl_percentage	To display percentage 1 to 100
Progress bar	Progressbar1	To display progress bar from 1 t0 100
Timer	Timer1	To display date, time and progress bar in dynamic node

10. Write coding below in the coding tab area:

```
Private Sub Timer1_Timer()

ProgressBar1.Value = ProgressBar1.Value + 5          ' tambah progress bar start from 0 + 5
lbl_loading.Caption = "Loading....."                 ' display loading
lbl_percentage.Caption = ProgressBar1.Value & "%"    ' display percentage

lbl_date.Caption = " Date  : " & Date                ' display date
lbl_time.Caption = Time                              ' display time


If ProgressBar1.Value = ProgressBar1.Max Then        ' when progress bar = 100..stoptimer
Timer1.Enabled = False

frm_interface.Hide                                   'close interface
frm_login.Show                                       'open login

End If


End Sub
```



LOGIN AND MSGBOX

- MsgBox displays a message in a dialog box, waiting for the user to click a button, returning to a specific value based on the button pressed/clicked.

- Message Box usage syntax:

  MsgBox (prompt [, buttons] [, title] [, helpfile, context])

Part	Explaination
Prompt	Must insert. Its display massage at dialog box. Maximum 1024 word
Buttons	Optional, refer figure 15
Tittle	Optional. Display title for Msg Box
Helpfile And Context	Optional. Use if link Msg Box with help file

Figure 14: Definition of Message box usage syntax

BUTTON ARGUMENT

- **First value:** the type of button to display
- **Second value:** icon type
- **Third value:** select which button is the default
- **Fourth value:** button functionality

- **First value:** The type of button to display

Constant	Value	Description
vbOKOnly	0	Display **OK** button only.
vbOKCancel	1	Display **OK** and **Cancel** buttons.
vbAbortRetryIgnore	2	Display **Abort**, **Retry**, and **Ignore** buttons.
vbYesNoCancel	3	Display **Yes**, **No**, and **Cancel** buttons.
vbYesNo	4	Display **Yes** and **No** buttons.
vbRetryCancel	5	Display **Retry** and **Cancel** buttons.

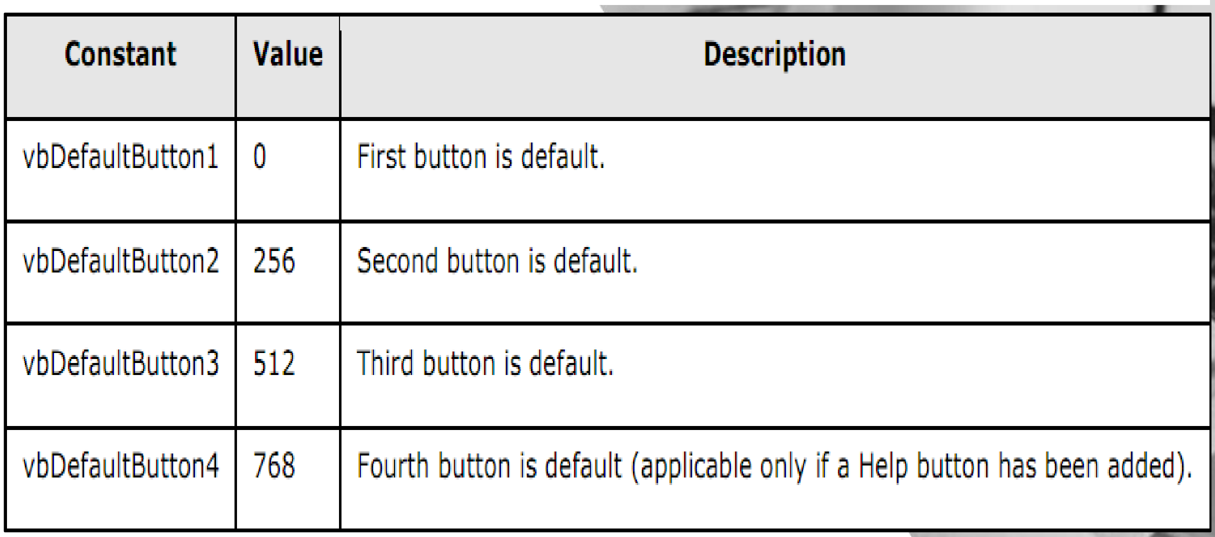
Figure 15: First value of button argument

- **Second value:** icon type

Constant	Value	Description	Icon
vbCritical	16	Display **Critical Message** icon.	❌
vbQuestion	32	Display **Warning Query** (question mark) icon.	❓
vbExclamation	48	Display **Warning Message** icon.	⚠
vbInformation	64	Display **Information Message** icon.	ℹ

Figure 16: Second value of button argument

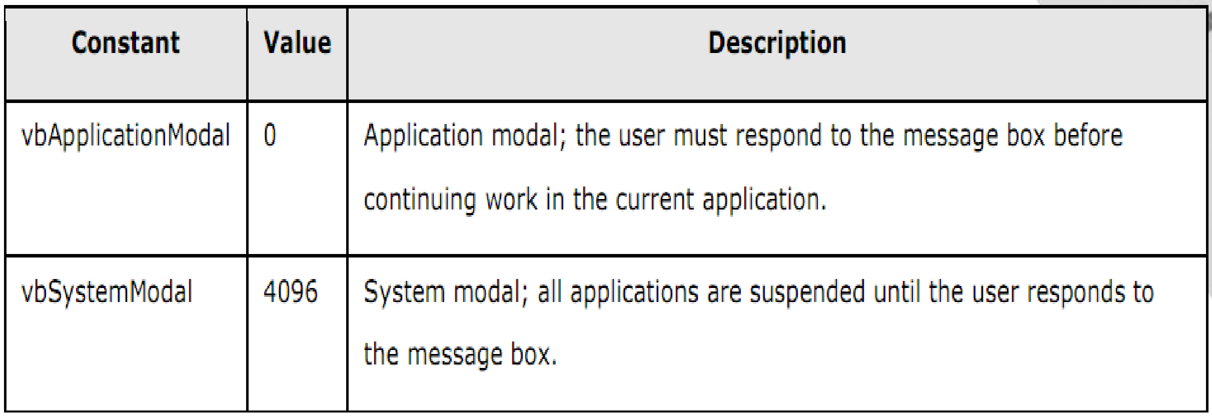
- **Third value:** choose which button is the default



Constant	Value	Description
vbDefaultButton1	0	First button is default.
vbDefaultButton2	256	Second button is default.
vbDefaultButton3	512	Third button is default.
vbDefaultButton4	768	Fourth button is default (applicable only if a Help button has been added).

Figure 17: Default button

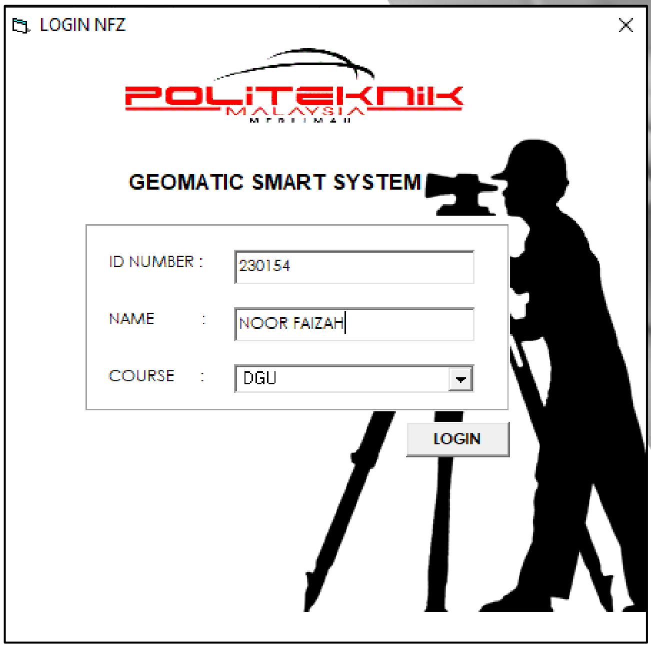
-**Fourth value:** button functionality



Constant	Value	Description
vbApplicationModal	0	Application modal; the user must respond to the message box before continuing work in the current application.
vbSystemModal	4096	System modal; all applications are suspended until the user responds to the message box.

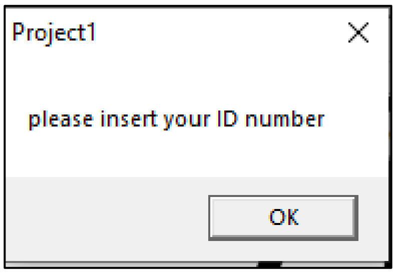
Figure 18: Button functionality

Example:



ID NUMBER : 230154
NAME : NOOR FAIZAH
COURSE : DGU

Figure 19: login interface



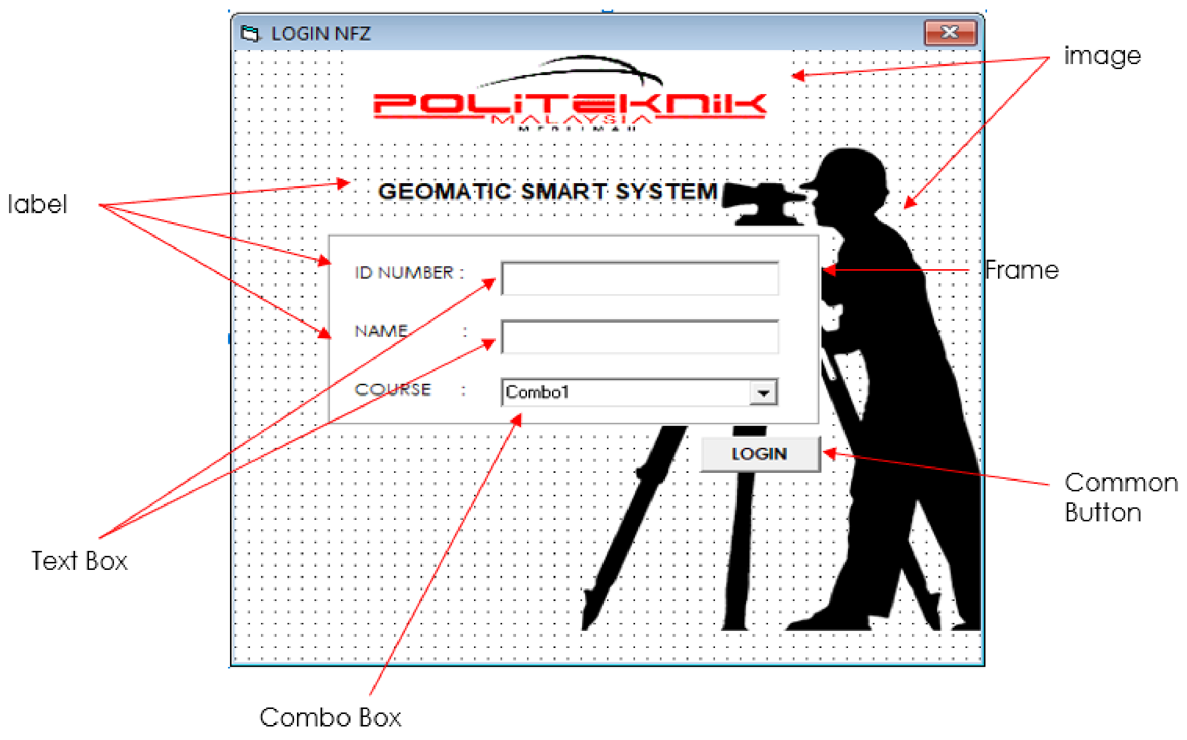
please insert your ID number

Figure 20: Msg box pop up massage

Procedure

1. Continue from previous project.
2. Right click at project, choose add and form
3. Insert all object base on figure below.
4. Write coding for

   i. keypress
   ii. Msg Box

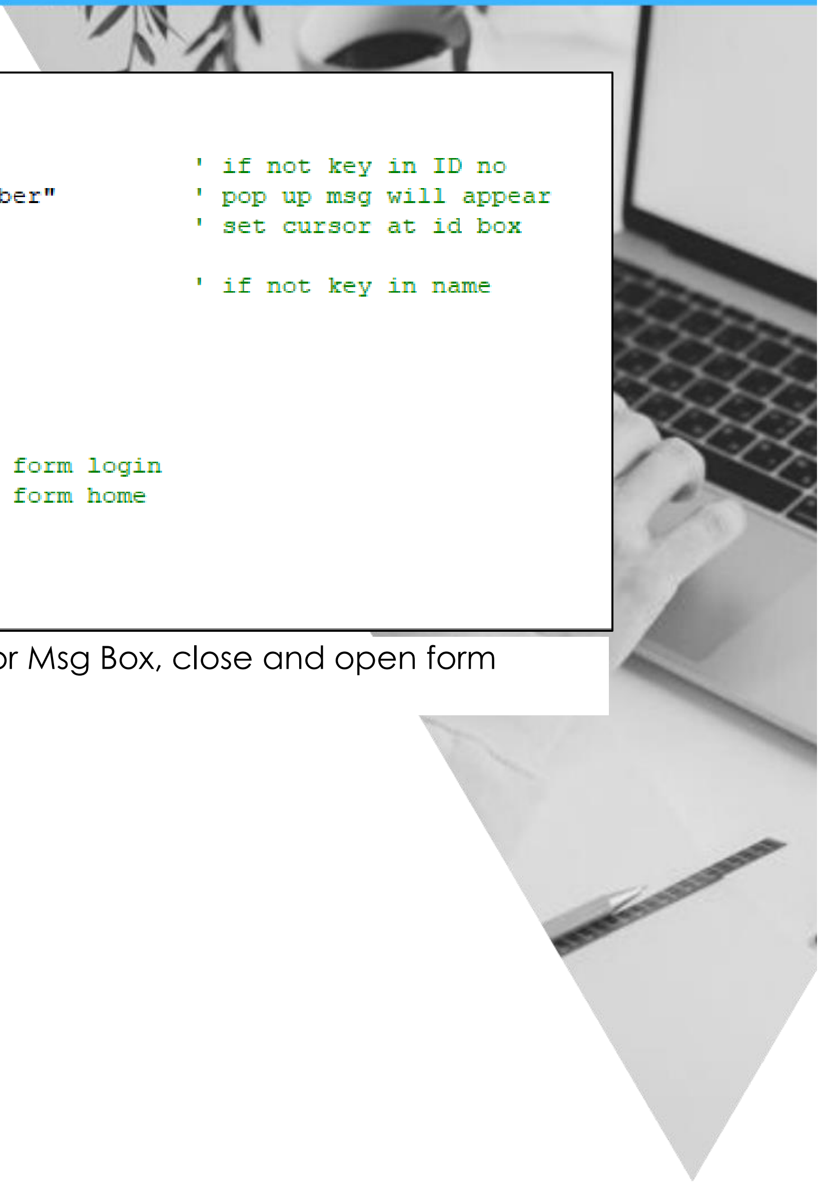


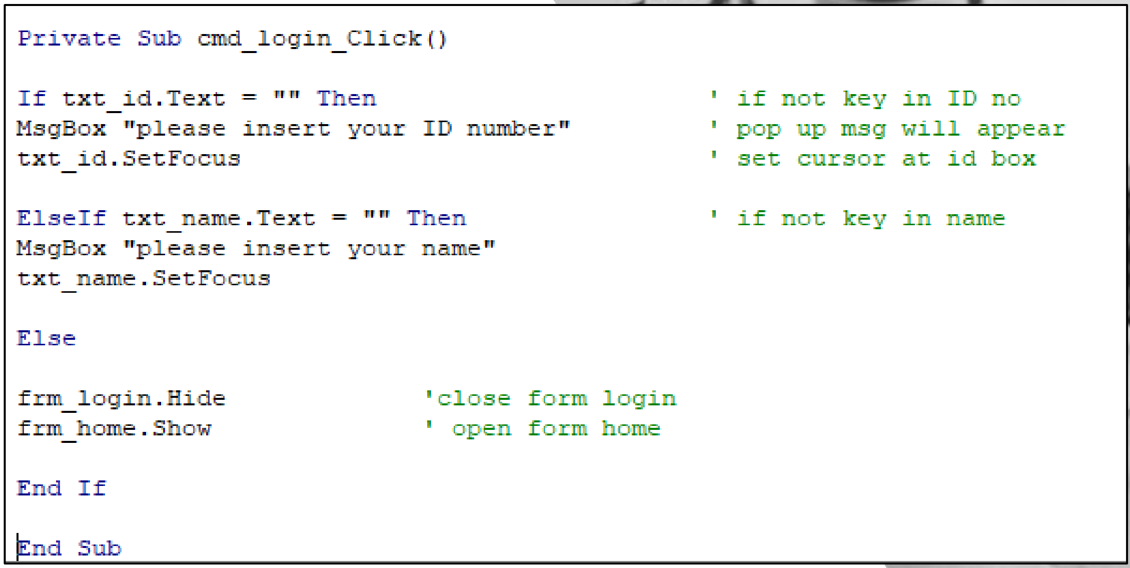
```
Private Sub txt_id_KeyPress(KeyAscii As Integer)

If KeyAscii = 13 Then                    ' when press enter
txt_name.SetFocus                        ' cursor automatic set at name box
End If


End Sub
```

Figure 21: coding for key press

```
Private Sub Form_Load()

'list data at combo box

com_course.AddItem " DGU "
com_course.AddItem " DKA "
com_course.AddItem " DSB "

End Sub
```

Figure 22: coding for combo box at Form load

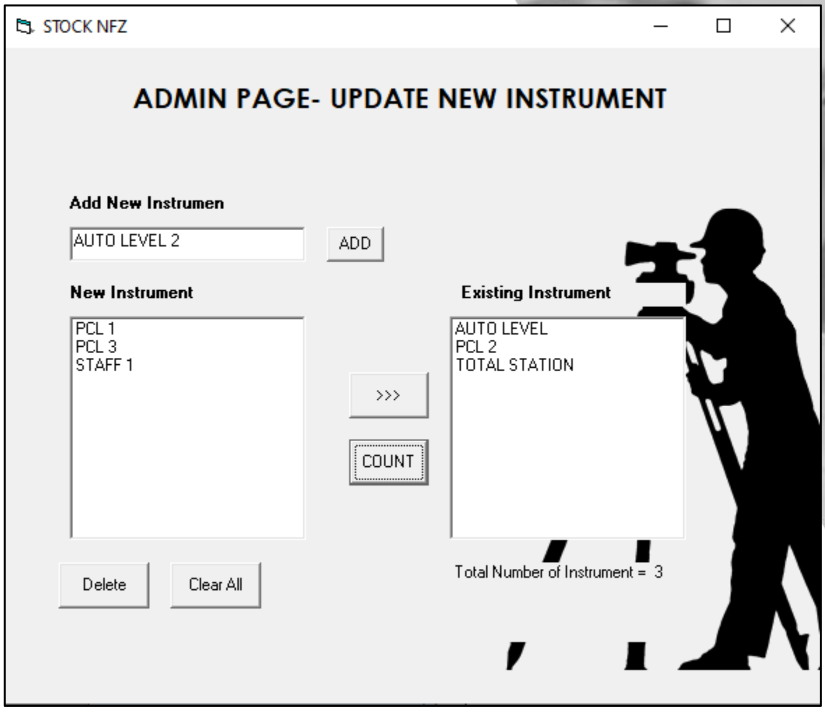




```
Private Sub cmd_login_Click()

If txt_id.Text = "" Then                     ' if not key in ID no
MsgBox "please insert your ID number"        ' pop up msg will appear
txt_id.SetFocus                              ' set cursor at id box

ElseIf txt_name.Text = "" Then               ' if not key in name
MsgBox "please insert your name"
txt_name.SetFocus

Else

frm_login.Hide                'close form login
frm_home.Show                 ' open form home

End If

End Sub
```

Figure 23: coding for Msg Box, close and open form

LIST BOX



AUTO LEVEL 2

PCL 1
PCL 3
STAFF 1

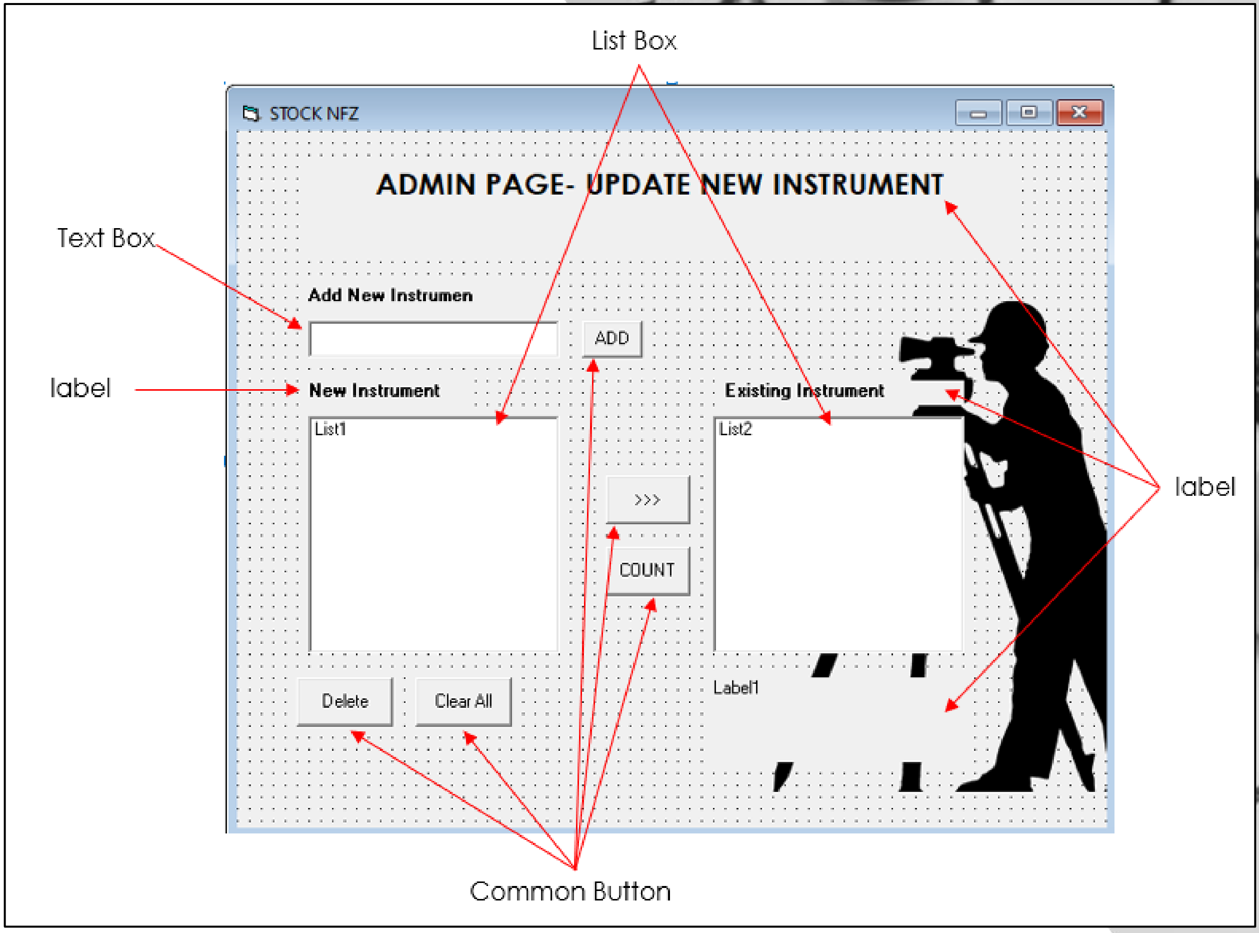
AUTO LEVEL
PCL 2
TOTAL STATION

Total Number of Instrument = 3

Procedure

1. Continue from previous project.
2. Add list box
3. Insert all object base on figure below.
4. Write coding for

   i. Add data
   ii. Delete data
   iii. Clear All data
   iv. Transfer data
   v. Count number of data



```
Private Sub cmd_add_Click()

List1.AddItem txt_newdata.Text              ' item will appear at list box
txt_newdata.Text = ""                       ' clear box add item
txt_newdata.SetFocus                        ' set cursor

End Sub
```

Figure 24: coding for add item to List box

```
Private Sub txt_newdata_KeyPress(KeyAscii As Integer)

If KeyAscii = 13 Then                        ' when press enter
cmd_add.SetFocus                             'cursor automatic set at button add
End If

End Sub
```

Figure 25: set mouse cursor to ADD button after key in the new data

```
Private Sub cmd_delete_Click()

List1.RemoveItem List1.ListIndex           ' delete item on list box

End Sub
```

```
Private Sub cmd_clear_Click()

List1.Clear                                ' clear all data on list box

End Sub
```

Figure 26: coding for delete and clear all item

```
Private Sub cmd_count_Click()

' count number of item appear on list box

lbl_count.Caption = " Total Number of Instrument =  " & List2.ListCount

End Sub
```

Figure 27: coding for count item on list box

```
Private Sub cmd_transfer_Click()

'remove selected item from list1 and transfer data to list 2

List2.AddItem List1.Text
List1.RemoveItem List1.ListIndex


End Sub
```

Figure 28: coding to transfer data from list box 1 to another list box

CHECK BOX

- The Check Box works like a toggle switch - click once for ON and click again for OFF.

- Compared to Option Button, Check Box does not depend between each other in a frame or form. If a check box is clicked or ON, the other check boxes will not be OFF (not affected directly).

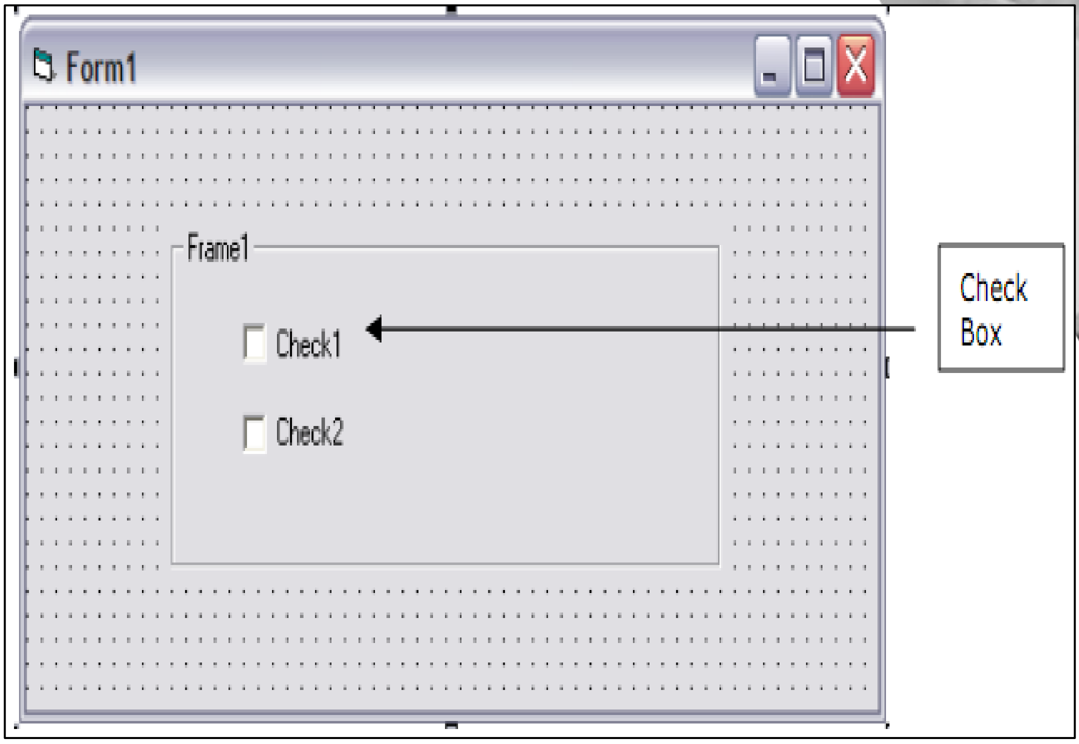
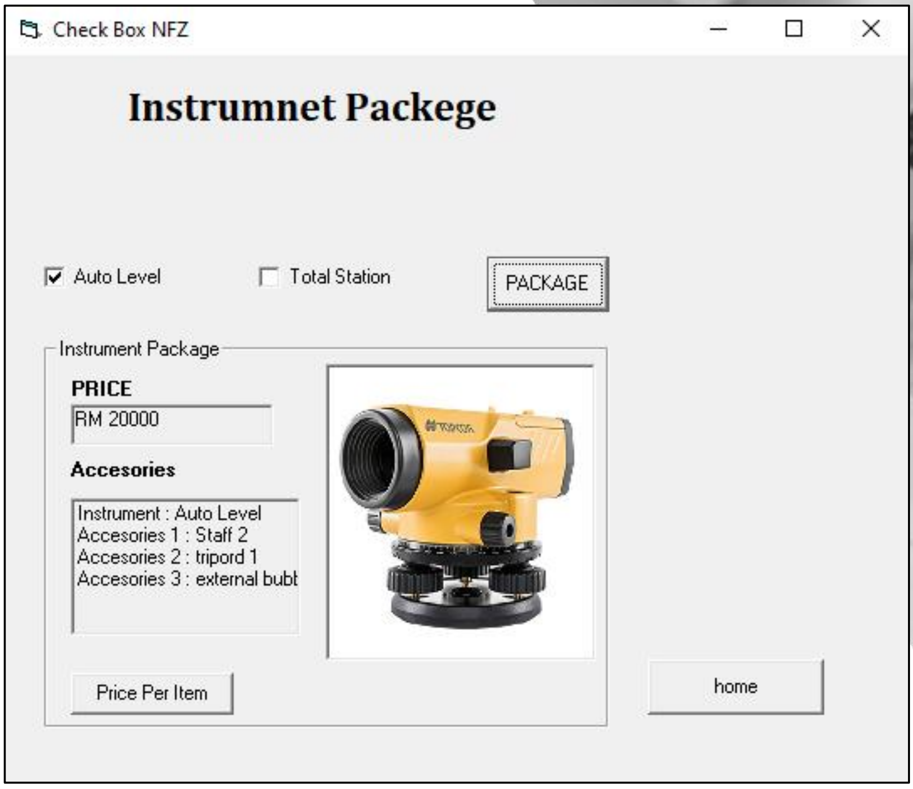


Figure 29: Check Box Button in Windows

Example:



Check Box NFZ

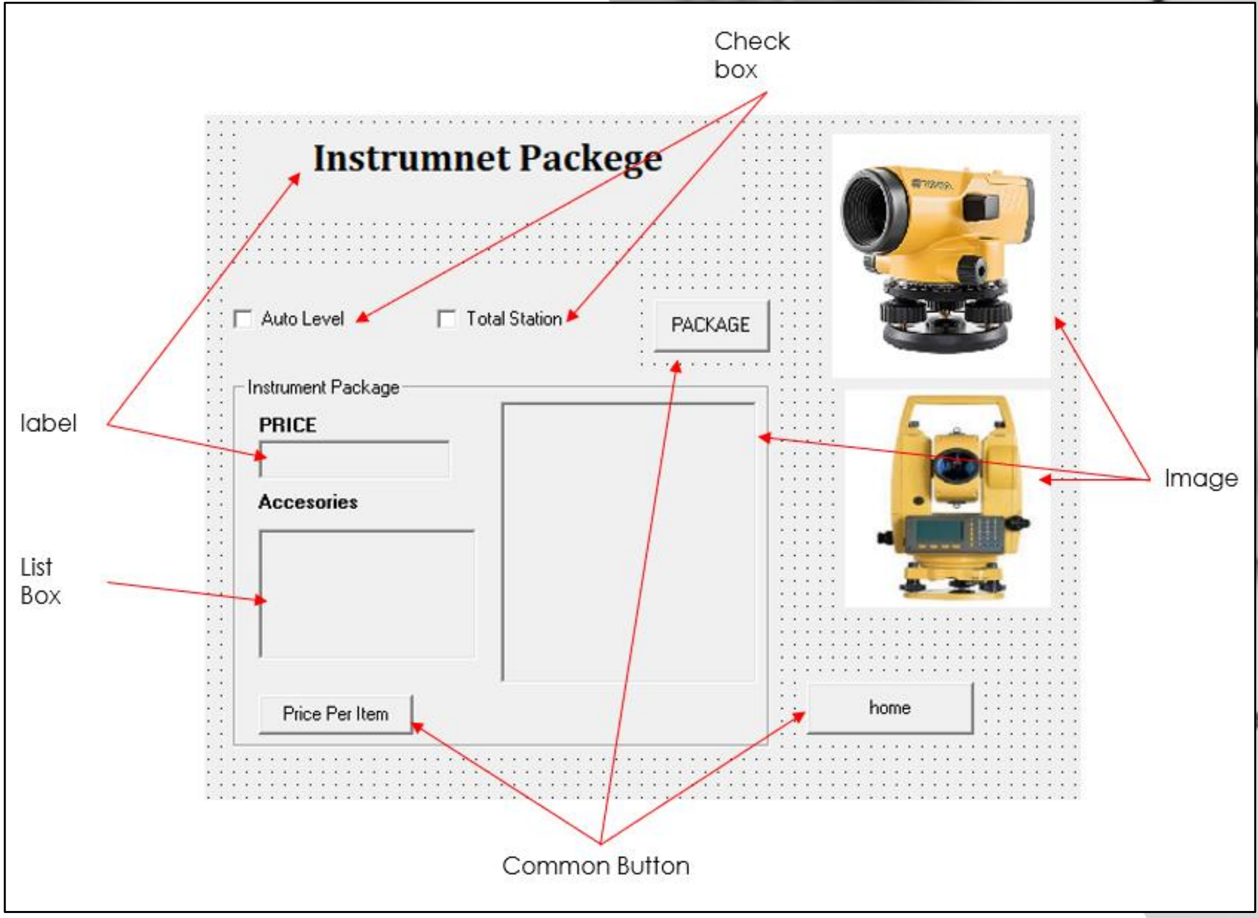
Instrumnet Packege

RM 20000

Instrument : Auto Level
Accesories 1 : Staff 2
Accesories 2 : tripord 1
Accesories 3 : external bubt

Procedure

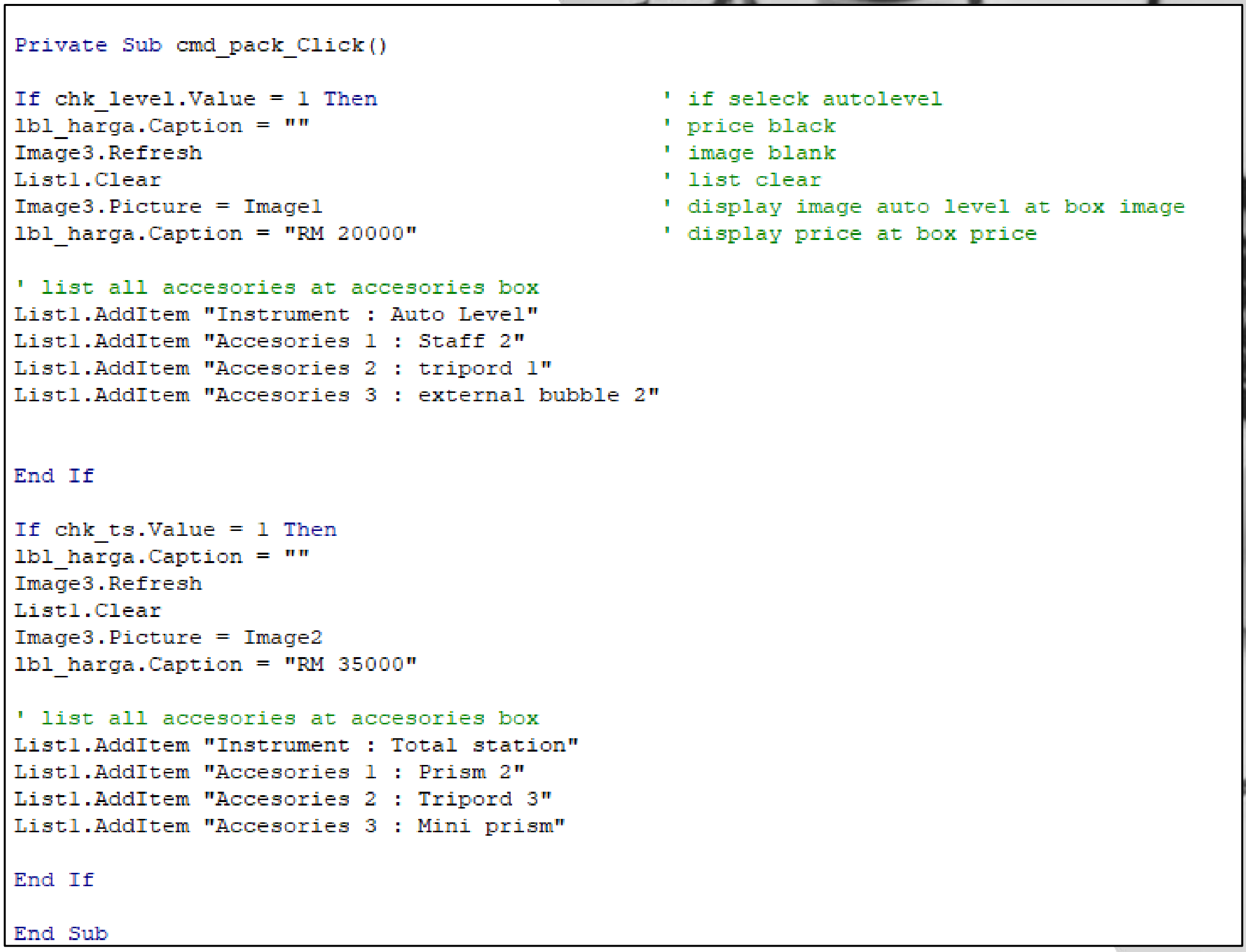
1. Continue from previous project.
2. Add Check box
3. Insert all object base on figure below.
4. Write coding for
   i. List item on list box and display image when choose check box
   ii. Change check box to option button
   iii. Select item from list box and display price for selected item



```
Private Sub chk_ts_Click()

If chk_ts.Value = 1 Then   ' if select total station automatic auto level uncheck
chk_level.Value = 0

End If

End Sub
```

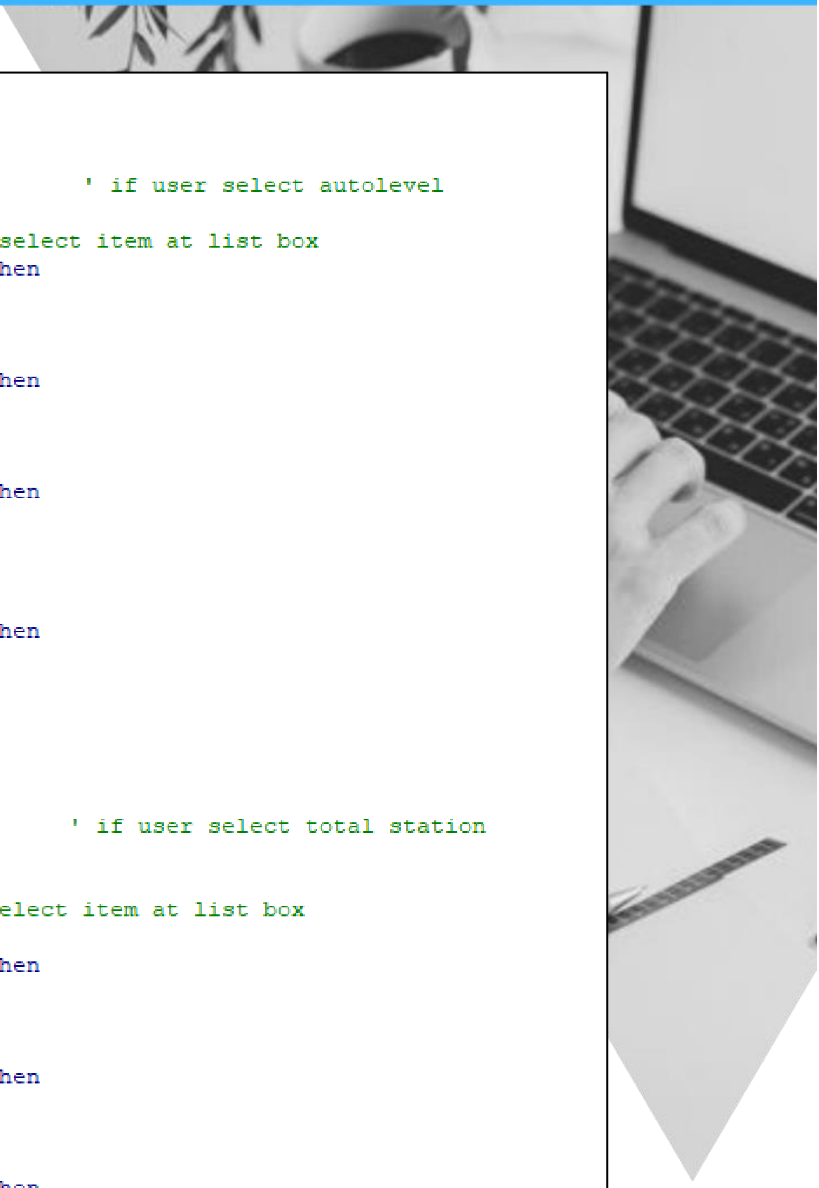
```
Private Sub chk_level_Click()

If chk_level.Value = 1 Then   ' if select auto level automatic total station uncheck
chk_ts.Value = 0

End If

End Sub
```

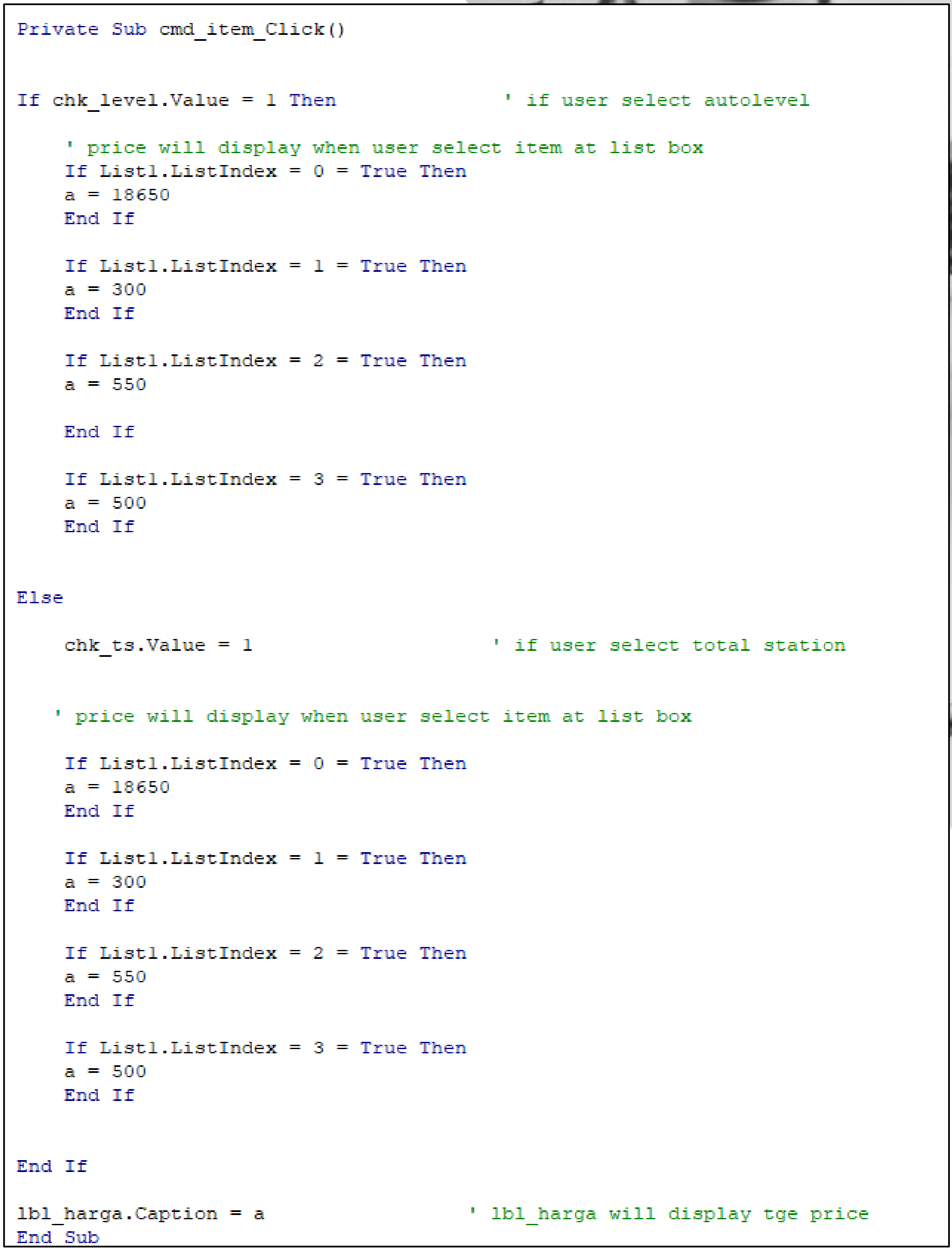
Figure 30: coding for convert check box to option box



```
Private Sub cmd_pack_Click()

If chk_level.Value = 1 Then                          ' if seleck autolevel
lbl_harga.Caption = ""                               ' price black
Image3.Refresh                                       ' image blank
List1.Clear                                          ' list clear
Image3.Picture = Image1                              ' display image auto level at box image
lbl_harga.Caption = "RM 20000"                       ' display price at box price

' list all accesories at accesories box
List1.AddItem "Instrument : Auto Level"
List1.AddItem "Accesories 1 : Staff 2"
List1.AddItem "Accesories 2 : tripord 1"
List1.AddItem "Accesories 3 : external bubble 2"


End If

If chk_ts.Value = 1 Then
lbl_harga.Caption = ""
Image3.Refresh
List1.Clear
Image3.Picture = Image2
lbl_harga.Caption = "RM 35000"

' list all accesories at accesories box
List1.AddItem "Instrument : Total station"
List1.AddItem "Accesories 1 : Prism 2"
List1.AddItem "Accesories 2 : Tripord 3"
List1.AddItem "Accesories 3 : Mini prism"

End If

End Sub
```

Figure 31: coding check box for display image and list item on list box





```
Private Sub cmd_item_Click()


If chk_level.Value = 1 Then              ' if user select autolevel

    ' price will display when user select item at list box
    If List1.ListIndex = 0 = True Then
    a = 18650
    End If

    If List1.ListIndex = 1 = True Then
    a = 300
    End If

    If List1.ListIndex = 2 = True Then
    a = 550

    End If

    If List1.ListIndex = 3 = True Then
    a = 500
    End If


Else

    chk_ts.Value = 1                    ' if user select total station


   ' price will display when user select item at list box

    If List1.ListIndex = 0 = True Then
    a = 18650
    End If

    If List1.ListIndex = 1 = True Then
    a = 300
    End If

    If List1.ListIndex = 2 = True Then
    a = 550
    End If

    If List1.ListIndex = 3 = True Then
    a = 500
    End If


End If

lbl_harga.Caption = a                 ' lbl_harga will display tge price
End Sub
```

Figure 32: coding for select item from list box and display price for selected item

OPTION BUTTON AND CHECK BUTTON

- Option Button is commonly used in more than one application. When one button is clicked (ON), the other button will be OFF. (Not 'clicked').

- It is placed in a group (container) like a frame. If the frame not used, the form itself will act as parentheses.

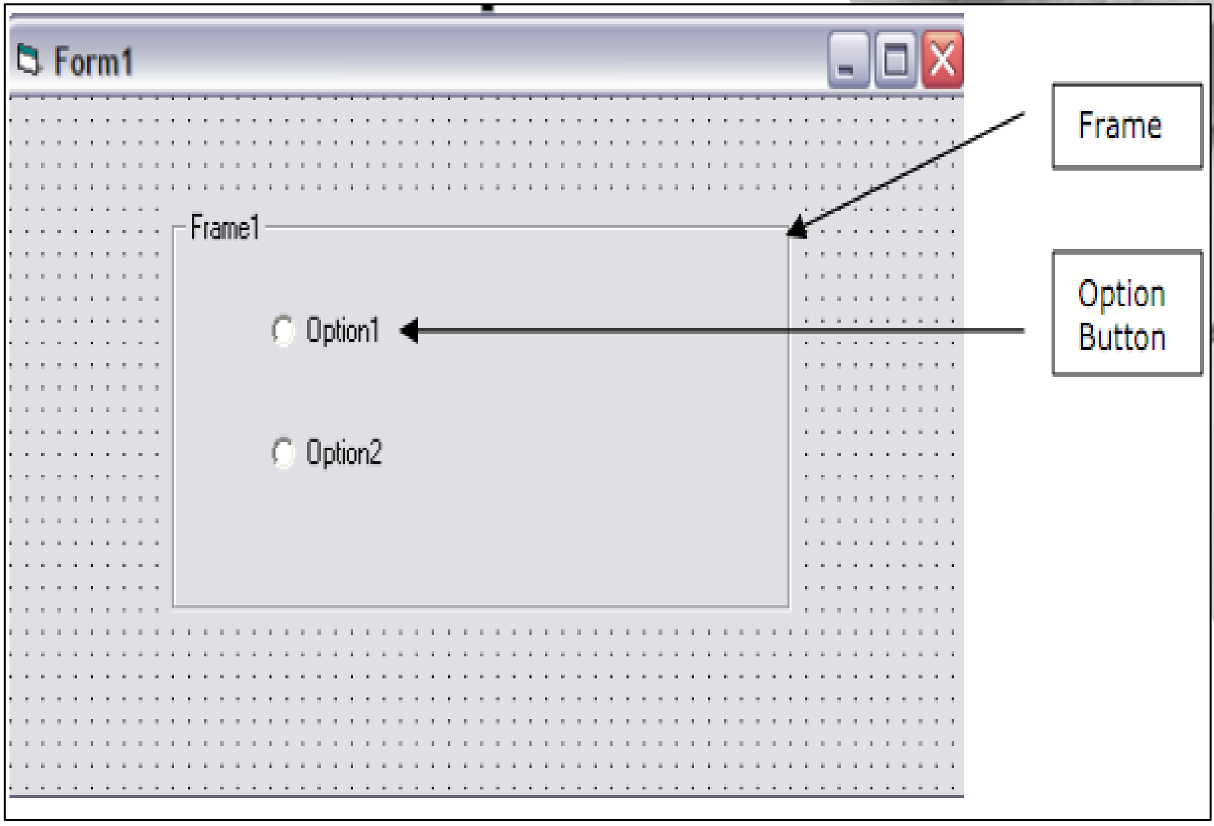
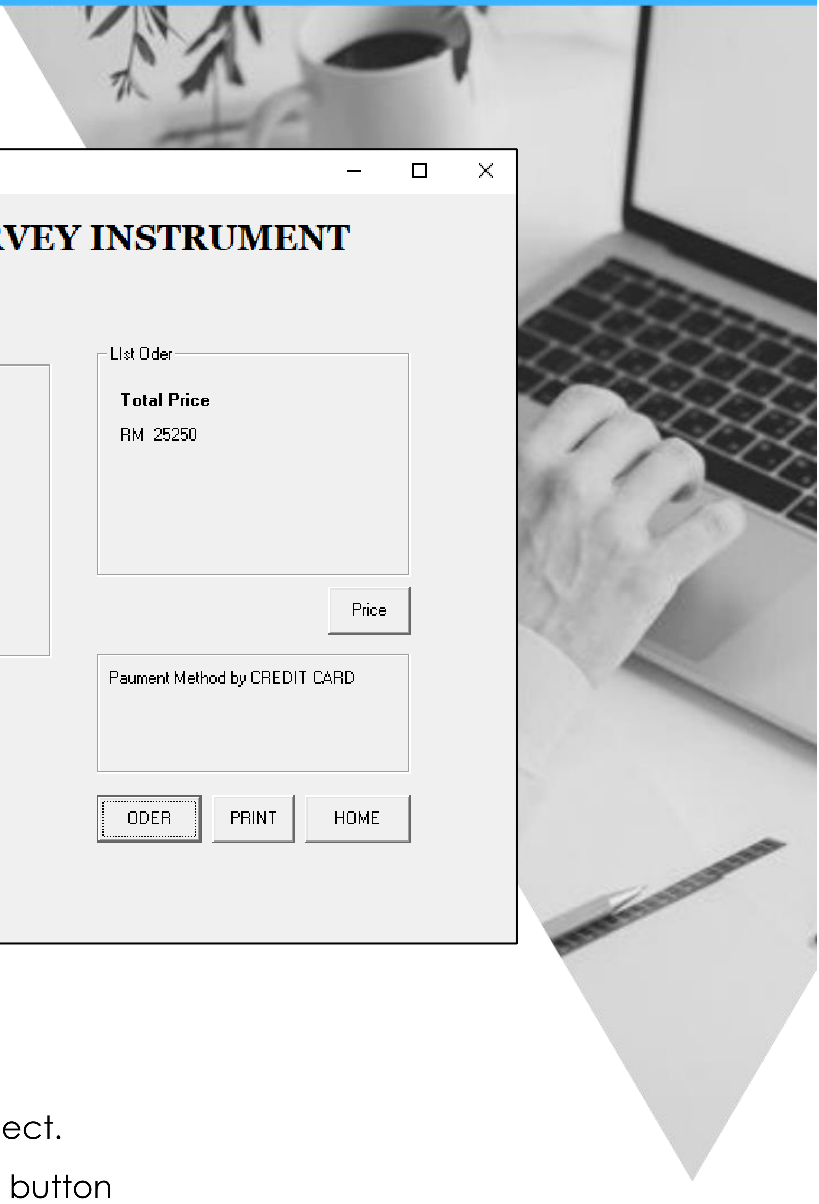
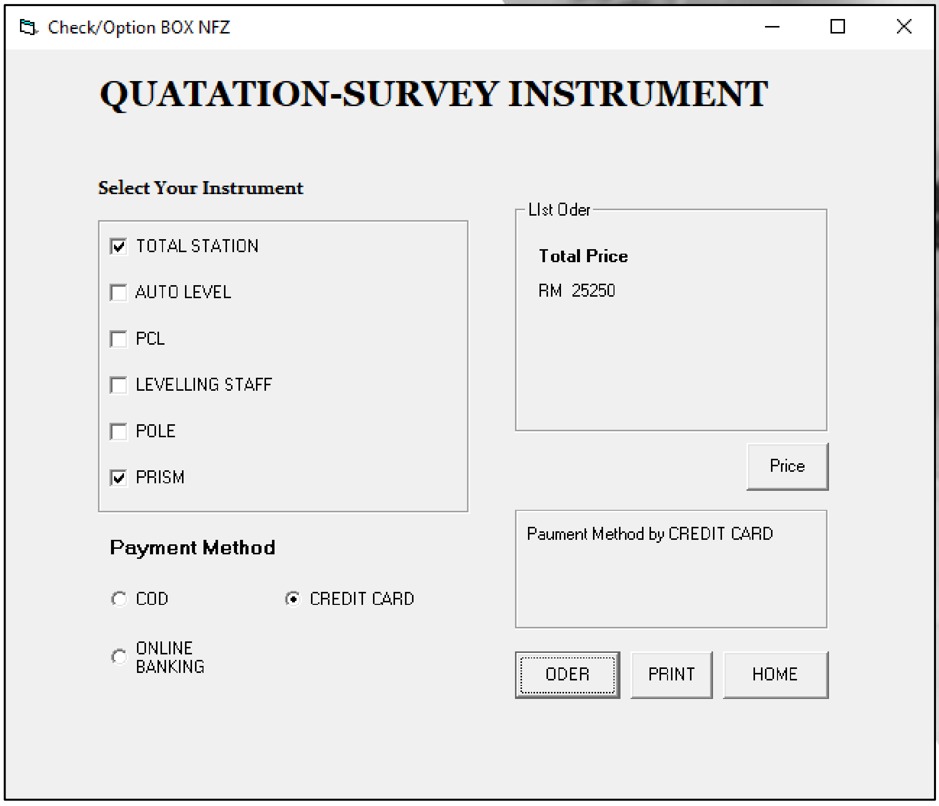


Figure 33: Display Option Button in Windows



Example:



Check/Option BOX NFZ

QUATATION-SURVEY INSTRUMENT

Select Your Instrument

- [✓] TOTAL STATION
- [ ] AUTO LEVEL
- [ ] PCL
- [ ] LEVELLING STAFF
- [ ] POLE
- [✓] PRISM

LIst Oder

Total Price

RM 25250

Payment Method

COD [✓ CREDIT CARD] ONLINE BANKING

Paument Method by CREDIT CARD

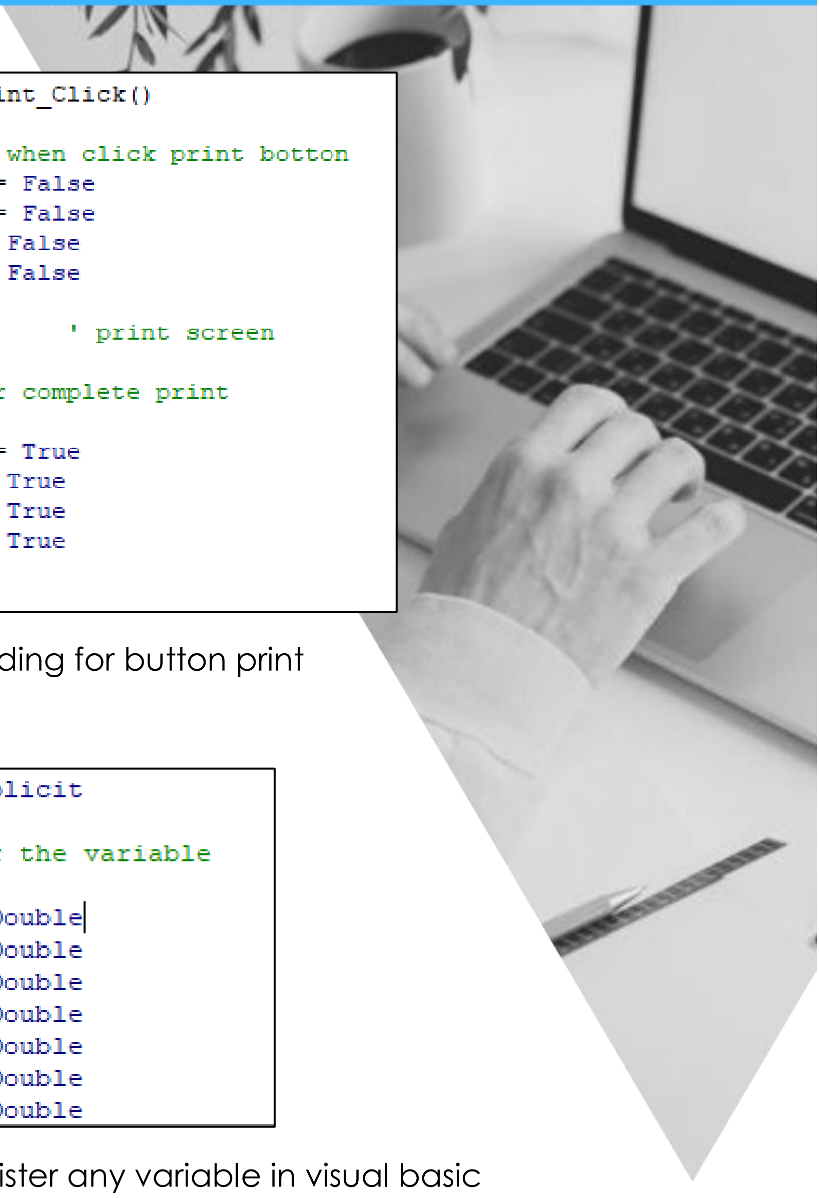
Procedure

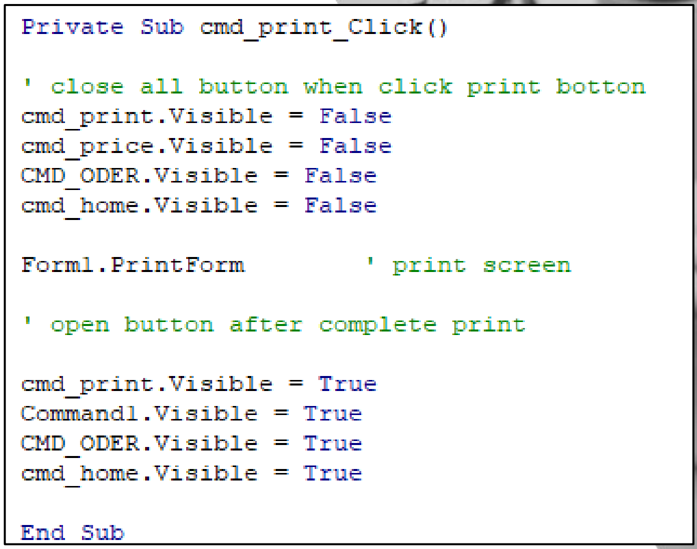
1. Continue from previous project.
2. Add Check box and option button
3. Insert all object base on figure below.
4. Write coding for

   i. Calculate price if select more than one check box
   ii. Identify selected option button
   iii. Print form
   iv. Button back to home

```
Private Sub cmd_home_Click()

frm_quotation.Hide                    'close form quotation
frm_home.Show                         ' open form home

End Sub
```

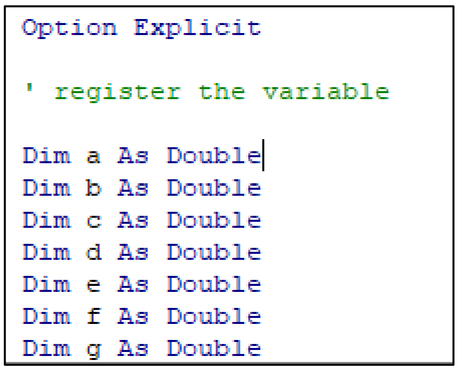
Figure 34: coding for button home





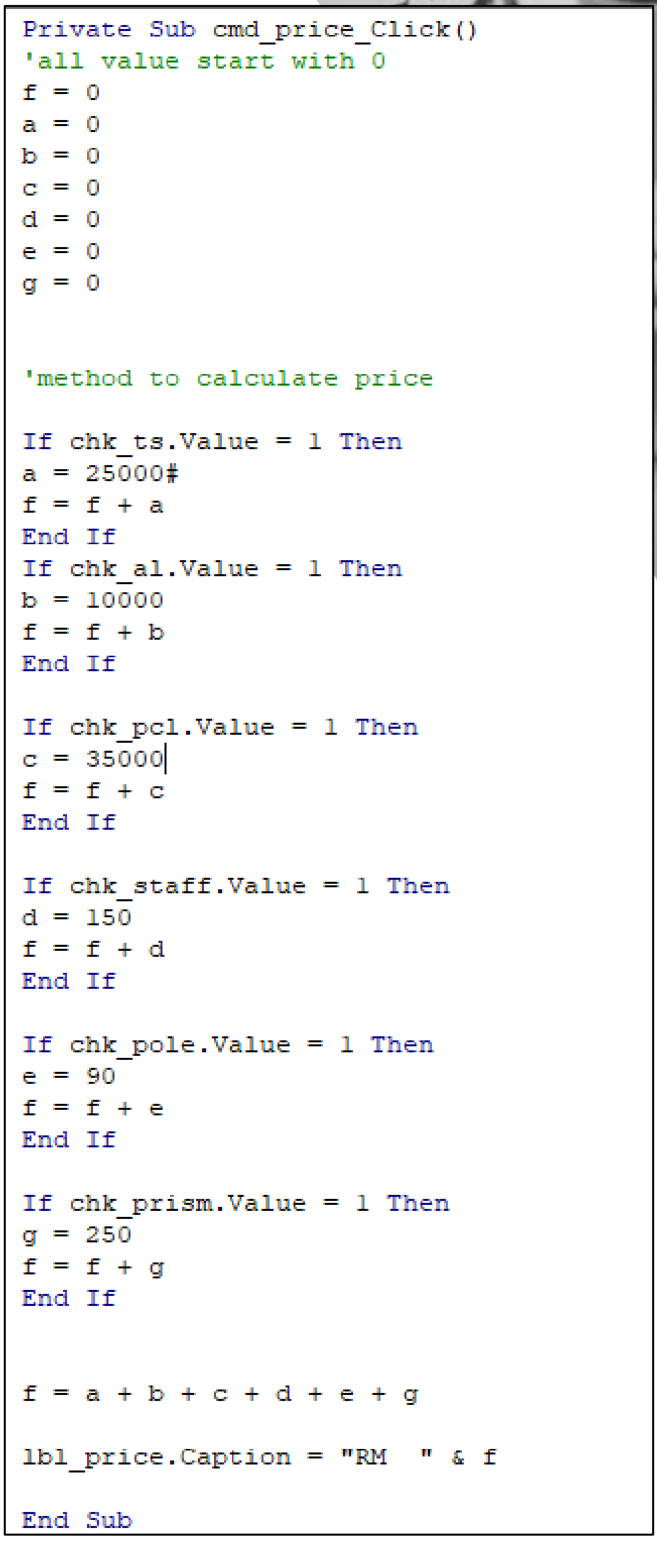
```
Private Sub cmd_print_Click()

' close all button when click print botton
cmd_print.Visible = False
cmd_price.Visible = False
CMD_ODER.Visible = False
cmd_home.Visible = False

Form1.PrintForm        ' print screen

' open button after complete print

cmd_print.Visible = True
Command1.Visible = True
CMD_ODER.Visible = True
cmd_home.Visible = True

End Sub
```

Figure 35: coding for button print



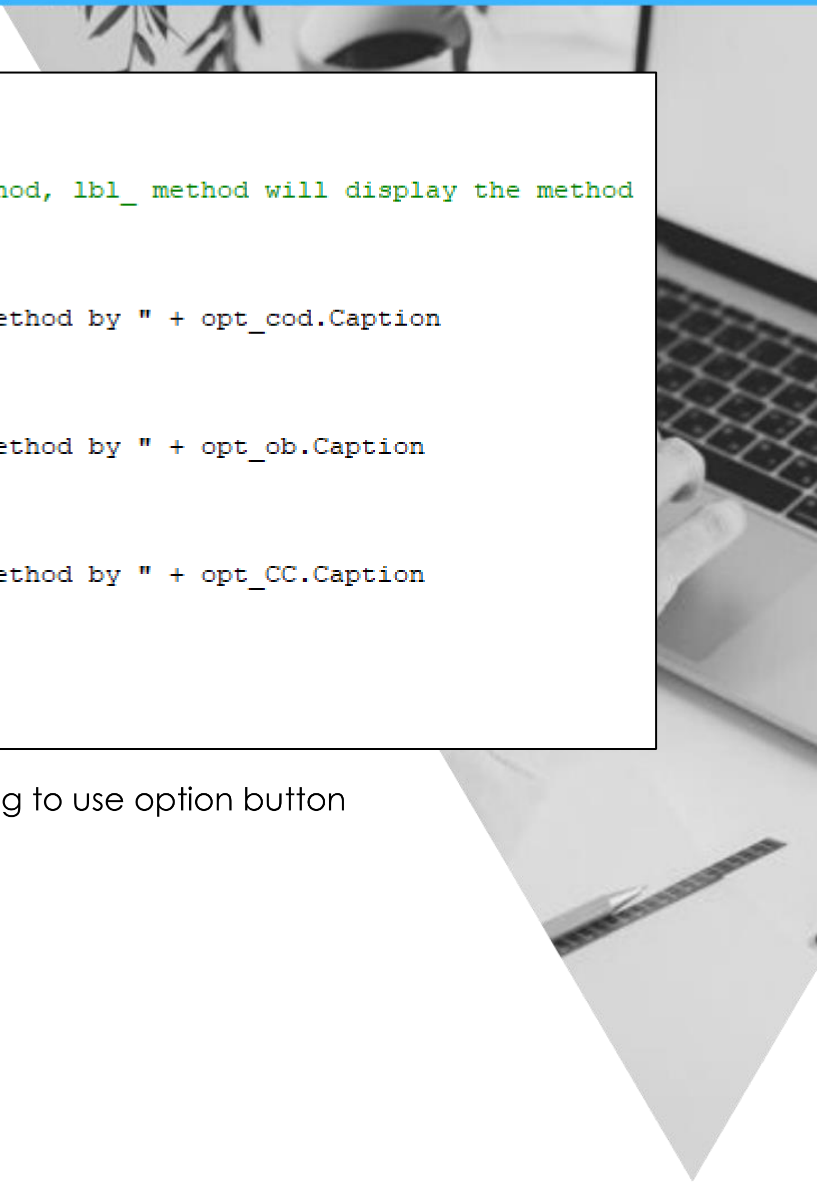
```
Option Explicit

' register the variable

Dim a As Double
Dim b As Double
Dim c As Double
Dim d As Double
Dim e As Double
Dim f As Double
Dim g As Double
```

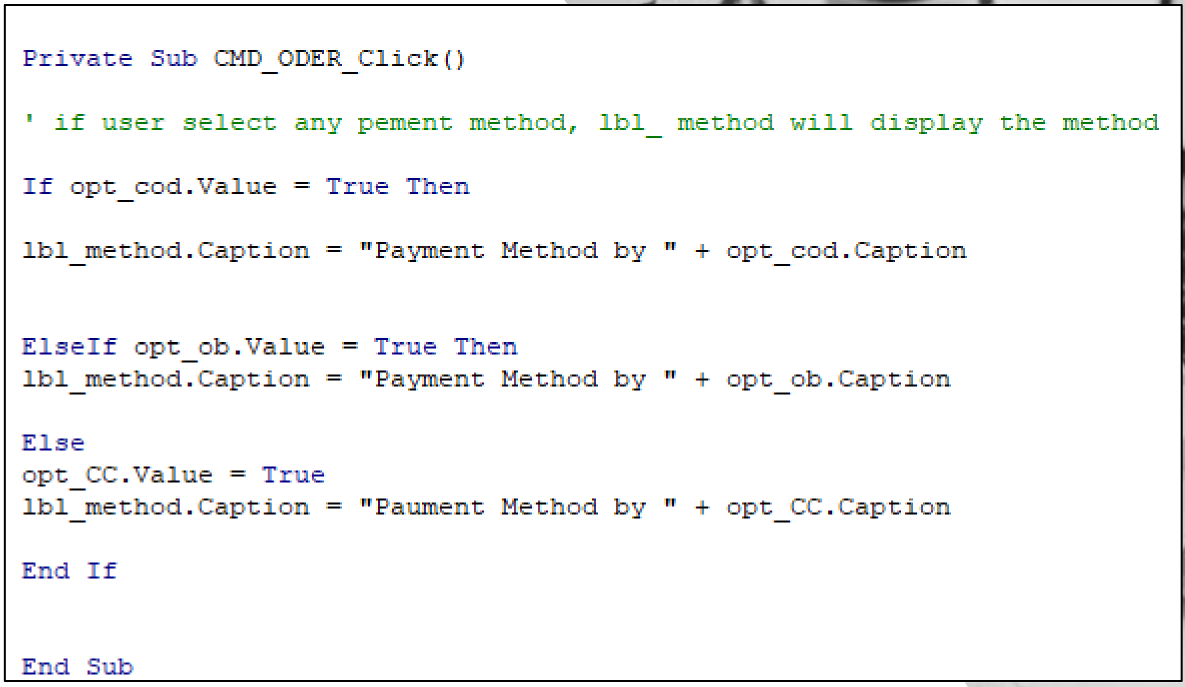
Figure 36: coding for register any variable in visual basic



```
Private Sub cmd_price_Click()
'all value start with 0
f = 0
a = 0
b = 0
c = 0
d = 0
e = 0
g = 0


'method to calculate price

If chk_ts.Value = 1 Then
a = 25000#
f = f + a
End If
If chk_al.Value = 1 Then
b = 10000
f = f + b
End If

If chk_pcl.Value = 1 Then
c = 35000
f = f + c
End If

If chk_staff.Value = 1 Then
d = 150
f = f + d
End If

If chk_pole.Value = 1 Then
e = 90
f = f + e
End If

If chk_prism.Value = 1 Then
g = 250
f = f + g
End If


f = a + b + c + d + e + g

lbl_price.Caption = "RM  " & f

End Sub
```

Figure 37: coding for calculate price





```
Private Sub CMD_ODER_Click()

' if user select any pement method, lbl_ method will display the method

If opt_cod.Value = True Then

lbl_method.Caption = "Payment Method by " + opt_cod.Caption


ElseIf opt_ob.Value = True Then
lbl_method.Caption = "Payment Method by " + opt_ob.Caption

Else
opt_CC.Value = True
lbl_method.Caption = "Paument Method by " + opt_CC.Caption

End If


End Sub
```

Figure 38: coding to use option button

CHAPTER 4

TRAVERSE APPLICATION

- Adjusted latit depart using Bowditch method
- Calculate coordinate for each station
- Calculate area
- Plot traverse

Calculate latit and depart for 1 station

Design part

1. Design form base on your creativity
2. Use text box for degree, minute, second and distance
3. Use label to display latit, depart and linear misclosure
4. Design all text box and label in one frame.
5. Refer figure and table for more explanation.

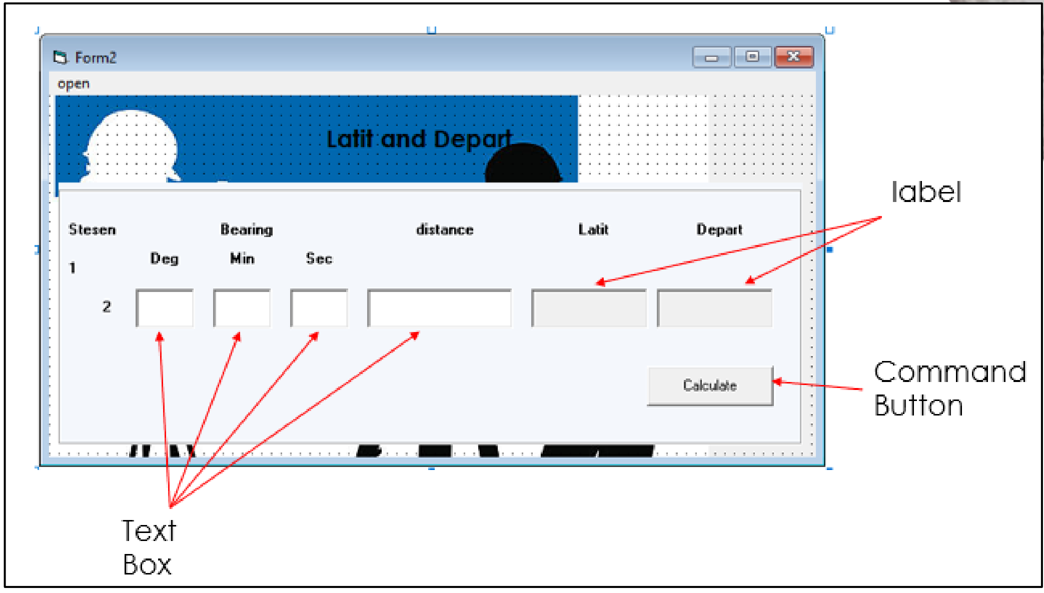


Figure 39: Form design for calculate single latit and depart

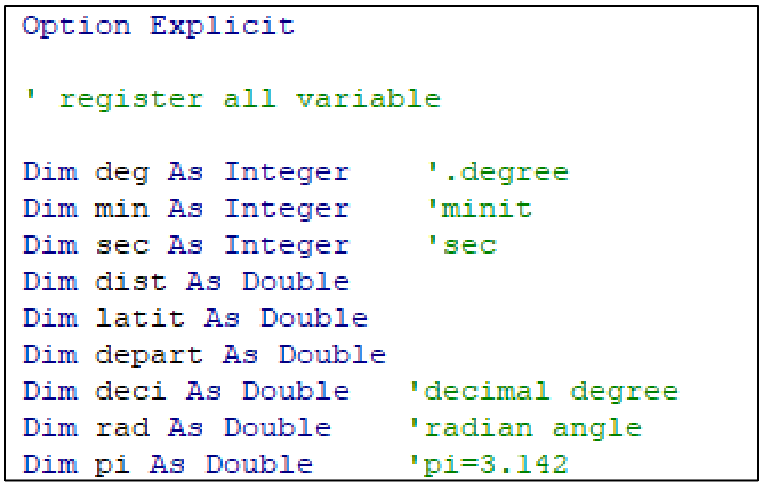
Table 6: Components of button

COMPONENT	NAME OF ICON	FUNCTION
Text box	txt_deg	Key in degree value
	txt_min	Key in minute value
	txt_sec	Key in second value
	txt_dist	Key in distance value
Label	lbl_latit	Display latit
	lbl_depart	Display depart
Command button	cmd_calculate	Button to calculate latit and depart

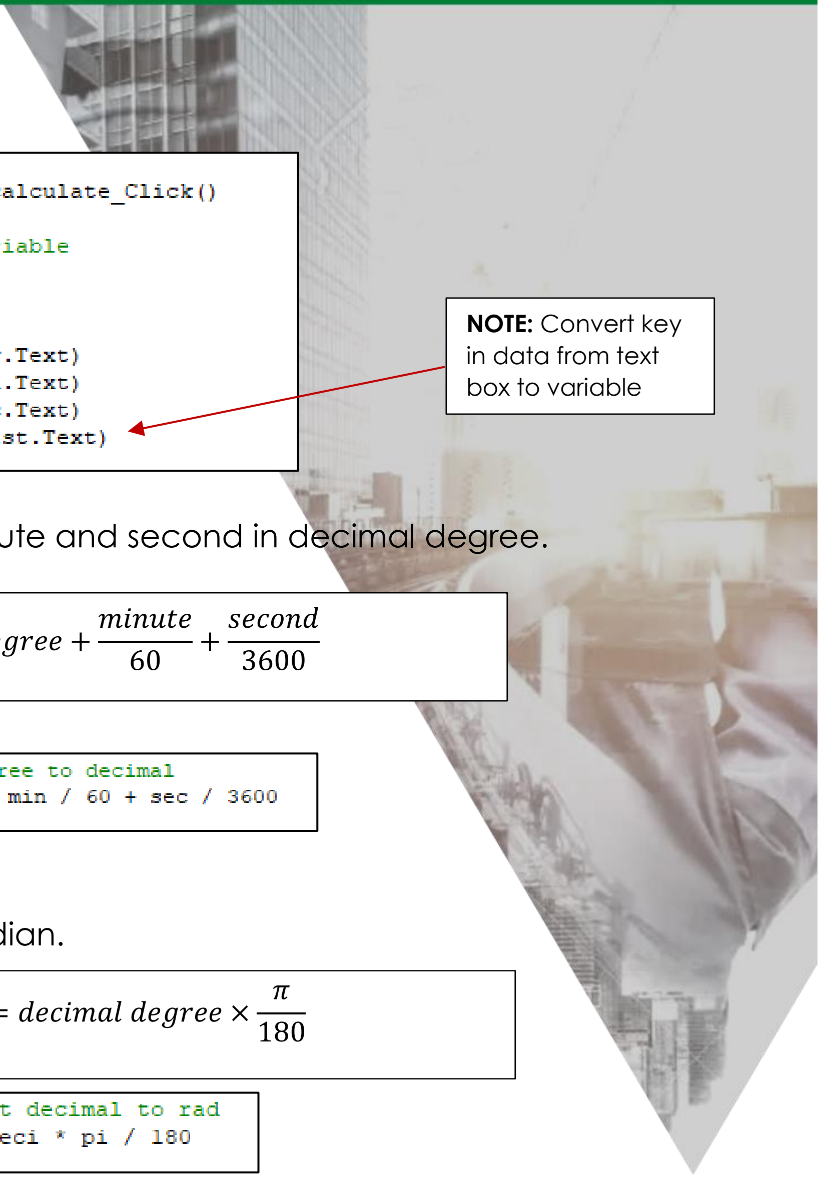
Write coding for calculate latit and depart

Write all coding at cmd_calculate button

1. Register all the variables to be used in the latit and depart calculation. Follow next step to identify the suitable variables.



```
Option Explicit

' register all variable

Dim deg As Integer     '.degree
Dim min As Integer     'minit
Dim sec As Integer     'sec
Dim dist As Double
Dim latit As Double
Dim depart As Double
Dim deci As Double    'decimal degree
Dim rad As Double     'radian angle
Dim pi As Double      'pi=3.142
```



2. Verify each variable

```
Private Sub cmd_calculate_Click()

' verify each variable

pi = 3.141592654

deg = Val(txt_deg.Text)
min = Val(txt_min.Text)
sec = Val(txt_sec.Text)
dist = Val(txt_dist.Text)
```

NOTE: Convert key in data from text box to variable

3. Convert all angle, Degree, minute and second in decimal degree.

$$decimal\ degree = degree + \frac{minute}{60} + \frac{second}{3600}$$

```
'convert degree to decimal
deci = deg + min / 60 + sec / 3600
```

4. Convert decimal degree to radian.

$$degree\ in\ radian = decimal\ degree \times \frac{\pi}{180}$$

```
'convert decimal to rad
rad = deci * pi / 180
```

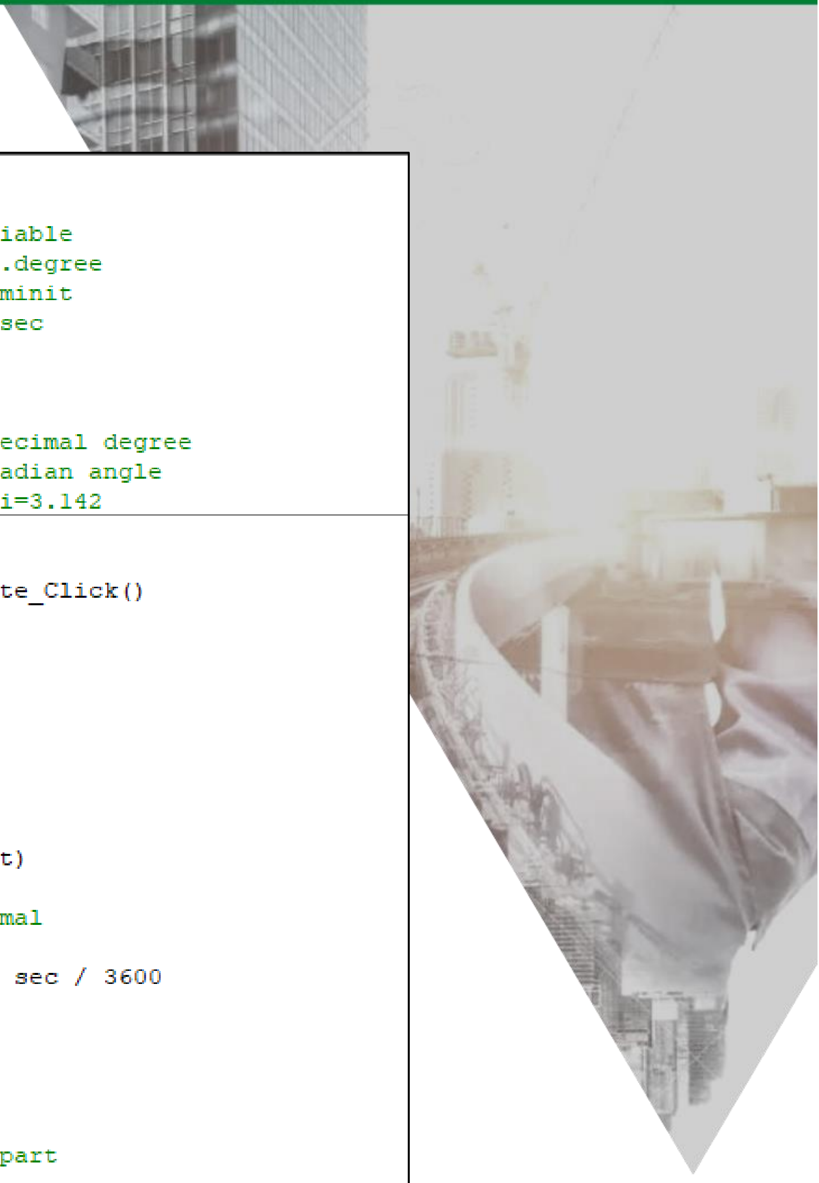
5. Identify the formula to calculate latit and depart

$$latit = distance \times \cos\theta$$

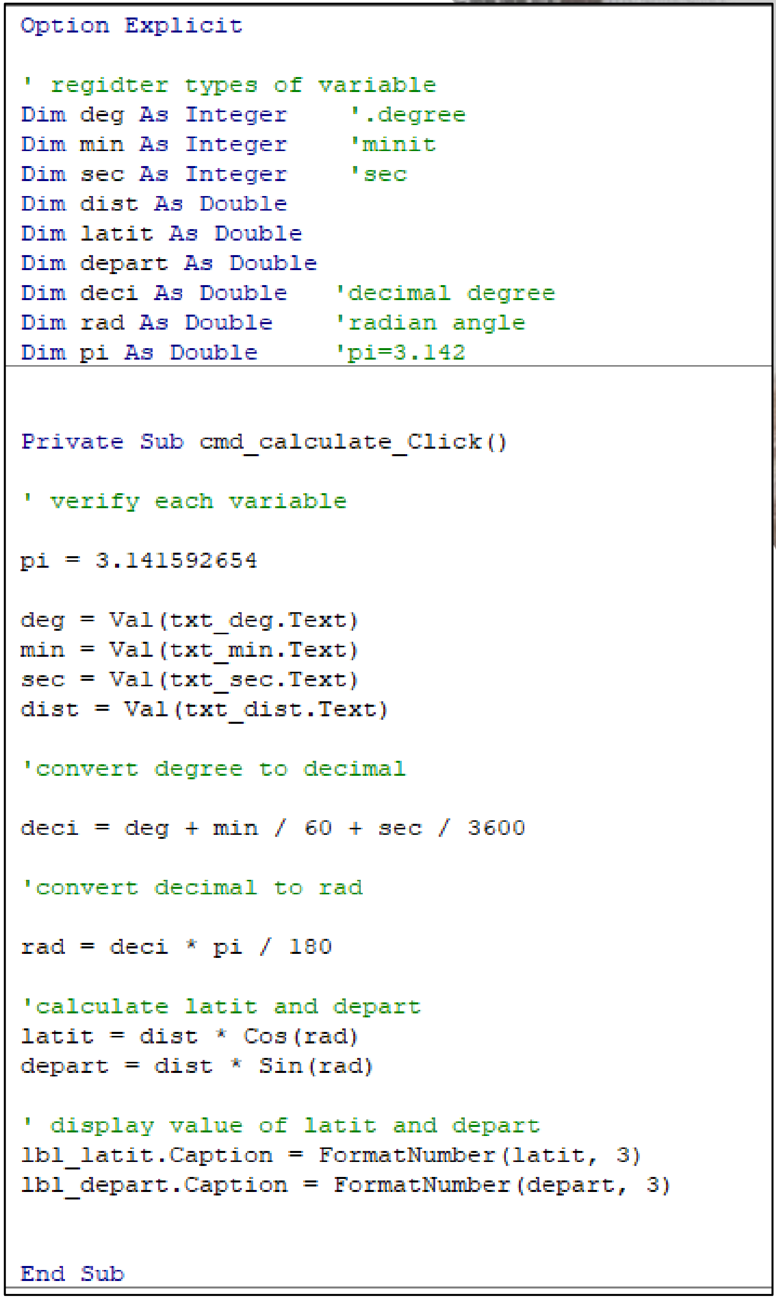
$$depart = distance \times \sin\theta$$

```
'calculate latit and depart
latit = dist * Cos(rad)
depart = dist * Sin(rad)

' display value of latit and depart
lbl_latit.Caption = FormatNumber(latit, 3)
lbl_depart.Caption = FormatNumber(depart, 3)


End Sub
```

NOTE: coding to display result in 3 decimal places



6. Refer figure for full coding



```
Option Explicit

' regidter types of variable
Dim deg As Integer    '.degree
Dim min As Integer    'minit
Dim sec As Integer    'sec
Dim dist As Double
Dim latit As Double
Dim depart As Double
Dim deci As Double   'decimal degree
Dim rad As Double    'radian angle
Dim pi As Double     'pi=3.142


Private Sub cmd_calculate_Click()

' verify each variable

pi = 3.141592654

deg = Val(txt_deg.Text)
min = Val(txt_min.Text)
sec = Val(txt_sec.Text)
dist = Val(txt_dist.Text)

'convert degree to decimal

deci = deg + min / 60 + sec / 3600

'convert decimal to rad

rad = deci * pi / 180

'calculate latit and depart
latit = dist * Cos(rad)
depart = dist * Sin(rad)

' display value of latit and depart
lbl_latit.Caption = FormatNumber(latit, 3)
lbl_depart.Caption = FormatNumber(depart, 3)


End Sub
```

Figure 40: Full coding to create application for latit depart

- **Create simple application for traverse calculation**

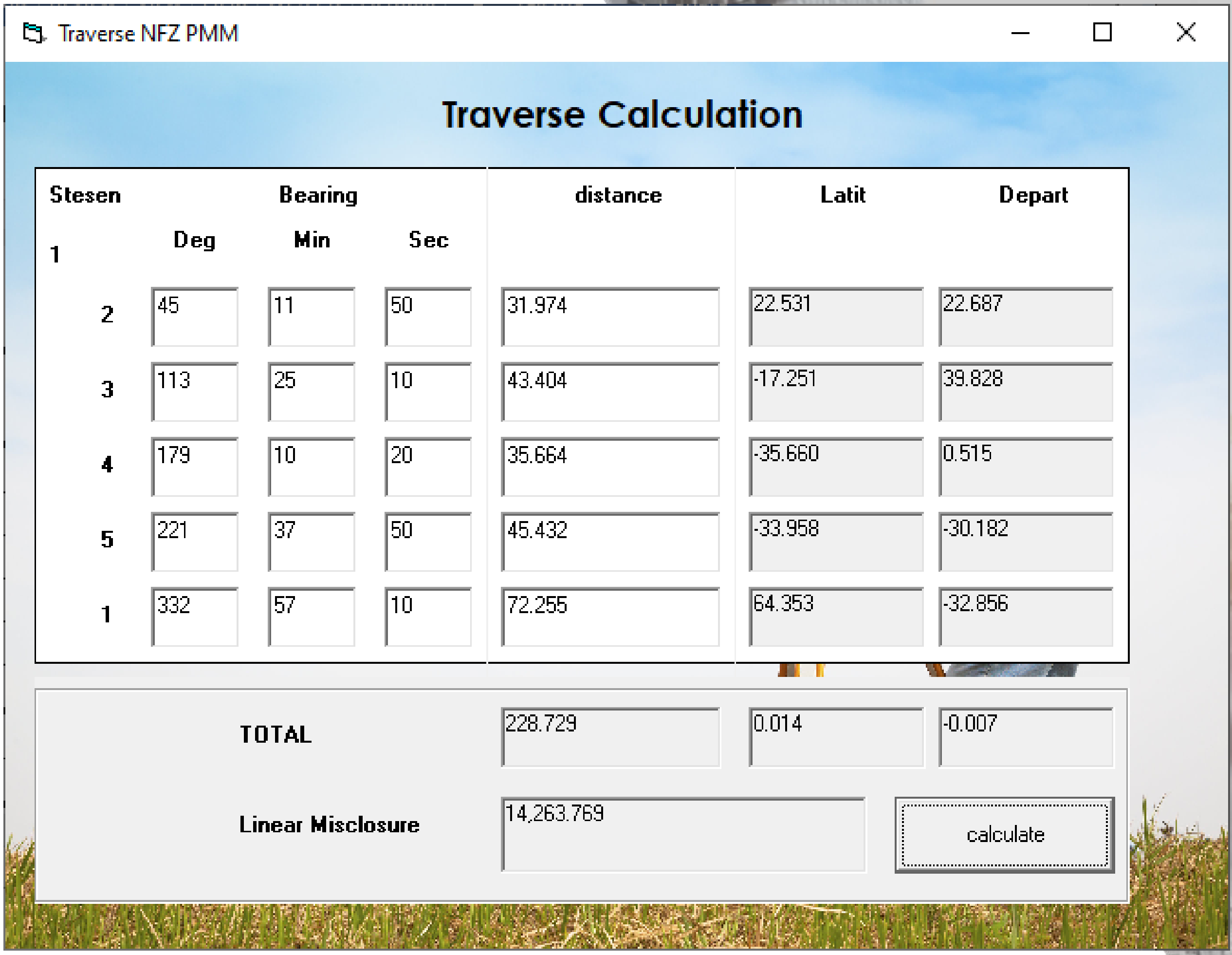
Table 7: Common table for manual traverse calculation

STN	BEARING	DISTANCE	LATIT		DEPART	
			N	S	E	W
1						
2	45º 11' 50"	31.974	22.531		22.687	
3	113º 25' 10"	43.404		17.251	39.828	
4	179º 10' 20"	35.664		35.660	0.515	
5	221º 37' 50"	45.432		33.958		30.182
1	332º 57' 10"	72.255	64.353			32.856
	TOTAL	228.729	86.884	86.869	63.030	63.038
	DIFFERENT		0.014		-0.007	

$$Linear\ Misclosure = 1 : \frac{\sqrt{(\Delta latit)^2 + (\Delta depart)^2}}{\sum distance}$$

$$= 1 : \frac{\sqrt{(0.014)^2 + (0.007)^2}}{228.729}$$

$$= 1 : \ 14\,263.769$$



Stesen	Bearing Deg	Min	Sec	distance	Latit	Depart
1						
2	45	11	50	31.974	22.531	22.687
3	113	25	10	43.404	-17.251	39.828
4	179	10	20	35.664	-35.660	0.515
5	221	37	50	45.432	-33.958	-30.182
1	332	57	10	72.255	64.353	-32.856
TOTAL				228.729	0.014	-0.007

Linear Misclosure: 14,263.769

Figure 41: Application for Traverse calculation

OBJECTIVE :

Create desktop application for calculate

i. Calculate latitude and depart
ii. Calculate linear misclosure

TIPS:

Textbox = keyin data

Label = display data

Design part

1. Design form base on your creativity
2. Use text box for degree, minute, second and distance
3. Separate text box for degree, minute and second
4. Use label to display latit, depart and linear misclosure
5. Design all text box and label in one frame.
6. Refer figure 2 for more explanation.

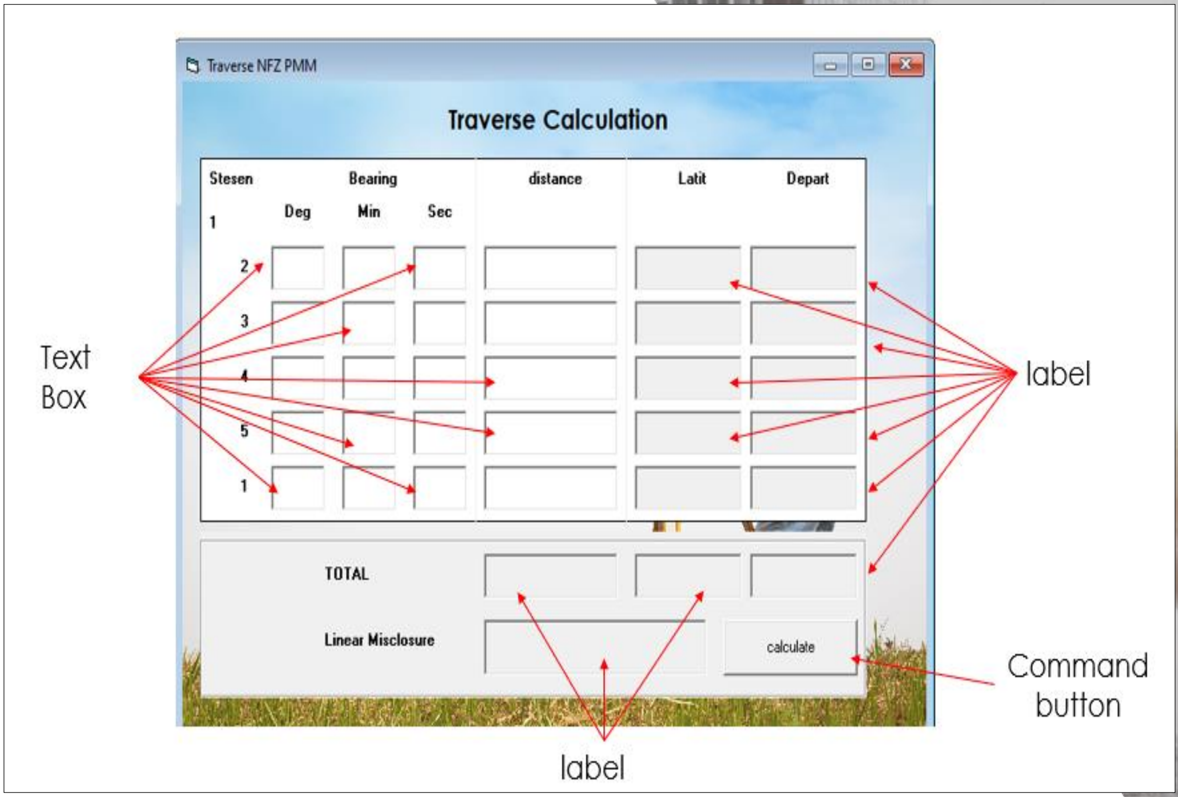


Figure 42: Form design for traverse calculation

Table 8: description for each icon

COMPONENT	NAME OF ICON	FUNCTION
Text box	txt_deg1	Key in degree value for station 2
	txt_deg2	Key in degree value for station 3
	txt_min1	Key in minute value for station 2
	txt_sec1	Key in second value for station 2
	txt_dist1	Key in distance value for station 2
Label	lbl_latit1	Display latit for station 2
	lbl_depart1	Display depart for station 2
	lbl_totallatit	Display total latit
	lbl_totaldepart	Display total depart
	lbl_totaldist	Display total distance
	Lbl_lm	Display result for linear misclosure
Command button	cmd_calculate	Button to calculate all calculation

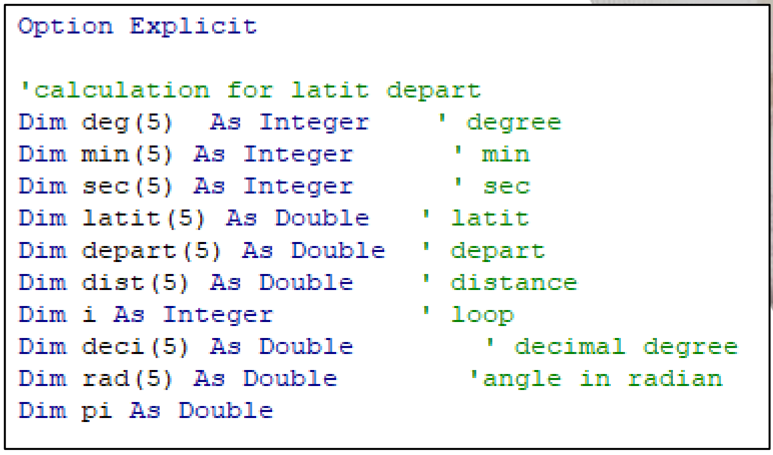
NOTE: rename all icon and use continuous number for each icon name.

For example: txt_min1, txt_min2, txt_min3, txt_min4

- **WRITE CODING FOR CALCULATE LATIT AND DEPART**

Write all coding at cmd_calculate button

1. For this application use looping method to simplify the coding.
2. Register all the variables to be used in this application. Follow next step to identify the suitable variables.



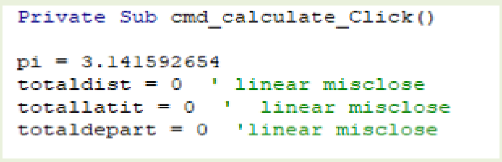
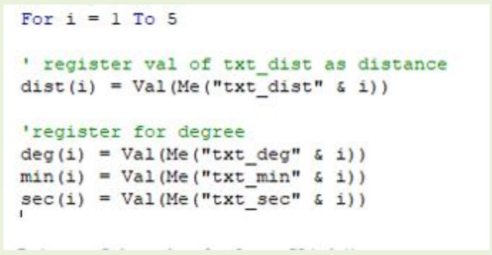
```
Option Explicit

'calculation for latit depart
Dim deg(5)  As Integer     ' degree
Dim min(5) As Integer       ' min
Dim sec(5) As Integer       ' sec
Dim latit(5) As Double     ' latit
Dim depart(5) As Double    ' depart
Dim dist(5) As Double      ' distance
Dim i As Integer           ' loop
Dim deci(5) As Double          ' decimal degree
Dim rad(5) As Double          'angle in radian
Dim pi As Double
```

-number in bracket after variable use for looping method. For example deg(5) is variable for deg1, deg2, deg3, deg4 and deg5

- for this programming variable " i " will use as looping variable

- use loop, if you want to repeat same formula/coding for next station.

CODING	EXPLANATION


`Private Sub cmd_calculate_Click()`
 	
`pi = 3.141592654`
`totaldist = 0  ' linear misclose`
`totallatit = 0  '  linear misclose`
`totaldepart = 0  'linear misclose`
 	

`For i = 1 To 5`
 	
`' register val of txt_dist as distance`
`dist(i) = Val(Me("txt_dist" & i))`
 	
`'register for degree`
`deg(i) = Val(Me("txt_deg" & i))`
`min(i) = Val(Me("txt_min" & i))`
`sec(i) = Val(Me("txt_sec" & i))`
`'`
	• Register value of π
	• Set value for total distance,totallatit and total depart equal to 0
	• Total latit = $\Delta\, latit$
	• Total depat = $\Delta\, depart$
	
	Note : set 0 is process to refresh and clear old data.

Coding without loop	Coding using loop

Deg1=val(txt_deg1)
Deg2=val(txt_deg2)
Deg3=val(txt_deg3)
Deg4=val(txt_deg4)
	For i = 1 To 4
	Deg(i)=val(Me("txt_deg"&i))
	Next i

If you have 10 data, you will repeat the coding until Deg10 =val(txt_deg10)	If you using loop, just chage 1. Dim deg(10) as double 2. For i = 1 To 10

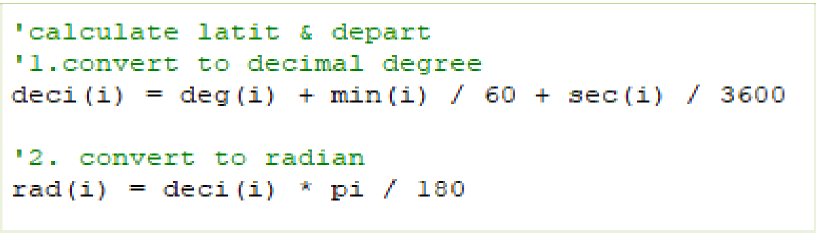
- For i= 1 to 5 is coding to looping.
  System will repeat the coding until i = 5

Convert key in data from text box to variable

-in loop process, all number will replace with (i)

Note:

If you have more than 1 station, you will repeat same coding for all station. Function of Loop is to reduce coding in your programming.



```
'calculate latit & depart
'1.convert to decimal degree
deci(i) = deg(i) + min(i) / 60 + sec(i) / 3600

'2. convert to radian
rad(i) = deci(i) * pi / 180
```

Convert degree, min and second to radian

`'3. latit & depart`
`latit(i) = dist(i) * Cos(rad(i))`
`depart(i) = dist(i) * Sin(rad(i))`
 	
`' write latit at lbl_latit`
`With Me("lbl_latit" & i)`
`.Caption = FormatNumber(latit(i), 3)`
`End With`
 	
`' write depart at lbl_depart`
`With Me("lbl_depart" & i)`
`.Caption = FormatNumber(depart(i), 3)`
`End With`
	Calculate latit and depart
	
	Display latit and depart without loop
	`' display value of latit and depart`
	`lbl_latit.Caption = FormatNumber(latit, 3)`
	`lbl_depart.Caption = FormatNumber(depart, 3)`
	
	Display latit using loop
	`With Me("lbl_latit" & i)`
	`.Caption = FormatNumber(latit(i), 3)`
	`End With`

`' calculate total diatance`
`totaldist = totaldist + dist(i)`
 	
`' calculate total latit`
`totallatit = totallatit + latit(i)`
 	
`'calculate total depart`
`totaldepart = totaldepart + depart(i)`
 	
`Next i`
	Total distance = total all traverse distance
	
	Total latit = $\sum N\ latit\ - \sum S\ latit$
	
	Total depart = $\sum E\ depart\ - \sum W\ depart$
	
	Next i = lopping process, all coding will repeat until I equal to number of station

`'display total distance, total latit & total depart`
`lbl_totaldist.Caption = totaldist`
`lbl_totallatit.Caption = FormatNumber(totallatit, 3)`
`lbl_totaldepart.Caption = FormatNumber(totaldepart, 3)`
	Coding use to display calculate value in 3 decimal places

`' calculate linear misclosure`
`linear = 1 / ((Sqr((totallatit ^ 2) + (totaldepart ^ 2))) / totaldist)`
 	
`'display linear misclosure`
`lbl_lm = FormatNumber(linear, 3)`
 	
`End Sub`
	Linear =1: $\frac{\sqrt{(\Delta latit)^2+(\Delta depart)^2}}{\sum distance}$
	
	Display linear misclosure in 3 decimal places.

BOWDITCH METHOD

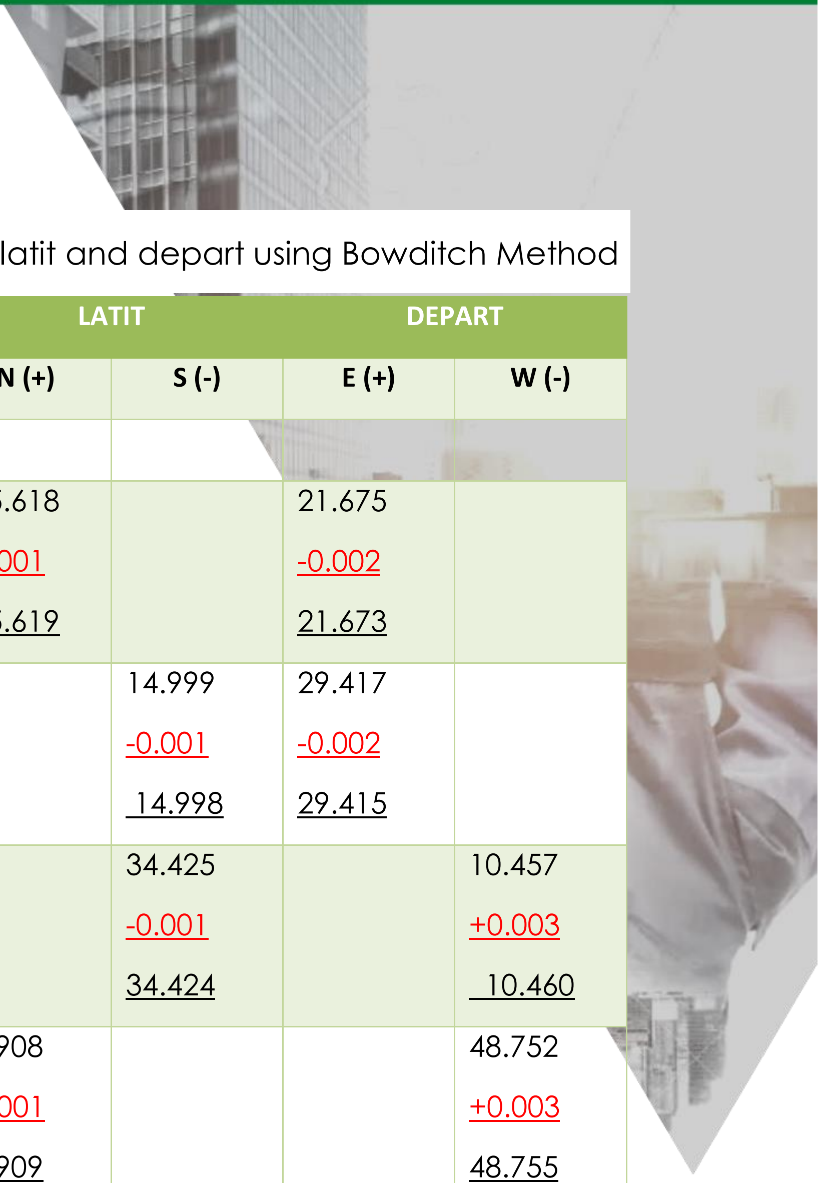


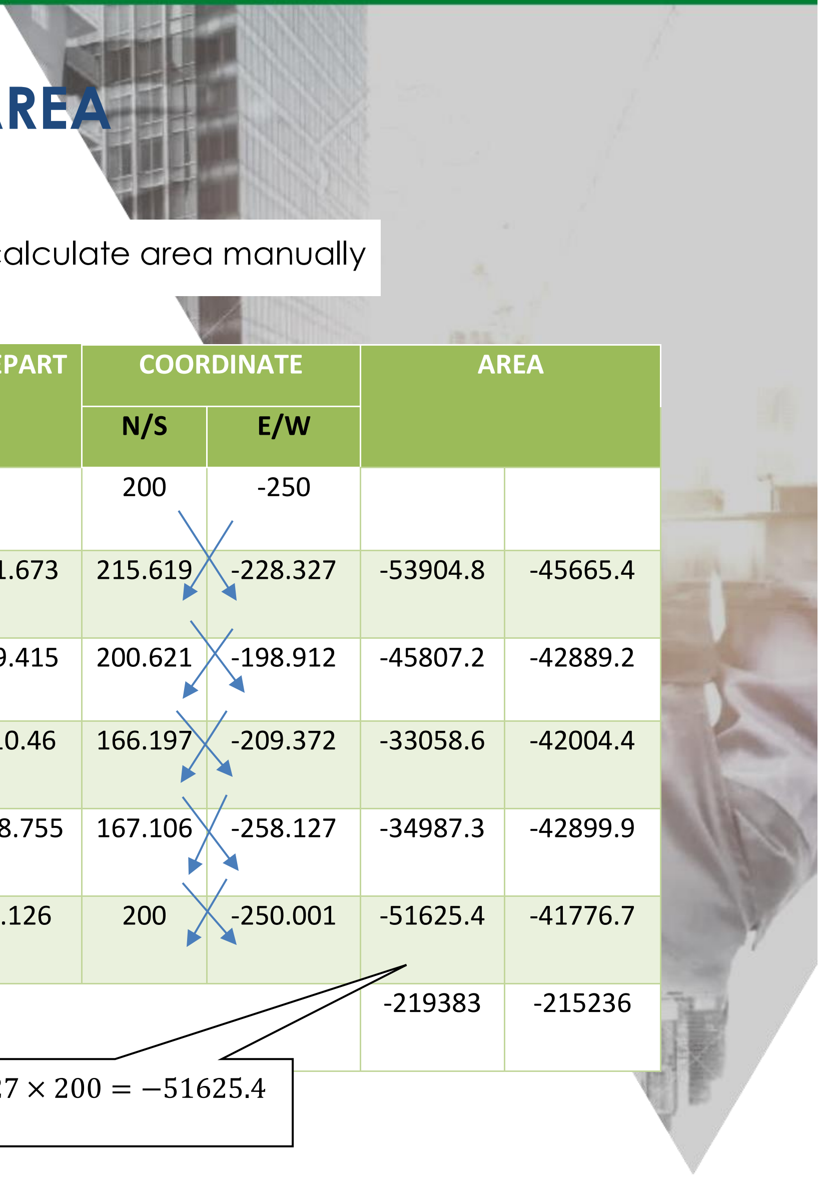
Table 9: table for calculate adjusted latit and depart using Bowditch Method

STN	BEARING	DISTANCE	LATIT		DEPART	
			N (+)	S (-)	E (+)	W (-)
1						
2	54° 13' 30"	26.716	15.618 +0.001 15.619		21.675 -0.002 21.673	
3	117° 01' 00"	33.020		14.999 -0.001 14.998	29.417 -0.002 29.415	
4	196° 53' 50"	35.978		34.425 -0.001 34.424		10.457 +0.003 10.460
5	271° 04' 00"	48.760	0.908 +0.001 0.909			48.752 +0.003 48.755
1	13° 52' 50"	33.882	32.893 +0.001 32.894		8.128 -0.002 8.126	
	Total	178.356	49.419	49.424	59.220	59.209
	Different		0.005		0.011	

Linear misclosure =1: $\frac{\sqrt{(0.005)^2+(0.011)^2}}{178.356}$

= 1 : 14 760

$$\boxed{\frac{33.882}{175.356} \times 0.011 = 0.002}$$



COORDINATE AND AREA

Table 10 : Table for calculate area manually

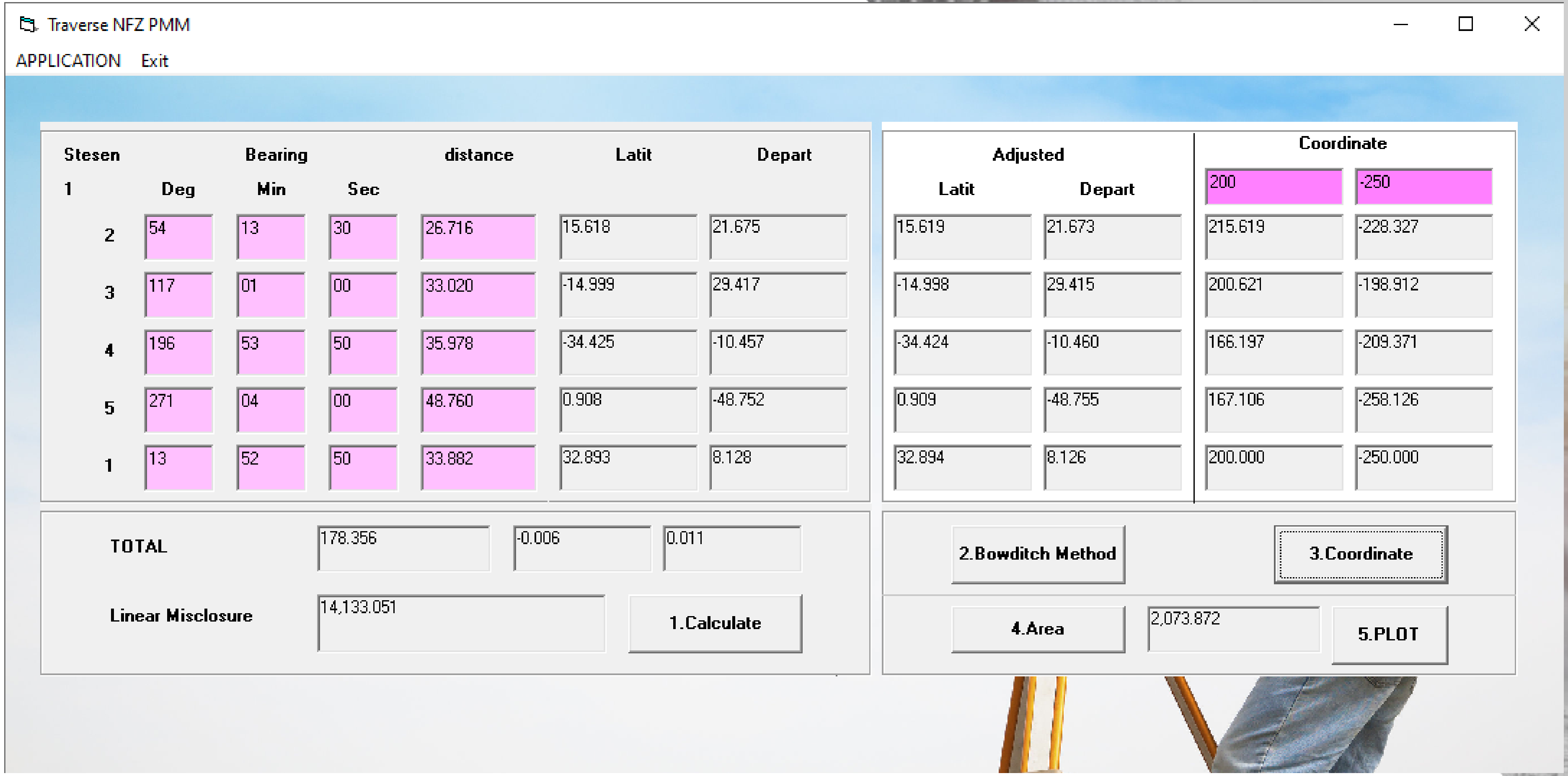
STN	BEARING	DISTANCE	LATIT	DEPART	COORDINATE		AREA	
					N/S	E/W		
1					200	-250		
2	54° 13’ 30”	26.716	15.619	21.673	215.619	-228.327	-53904.8	-45665.4
3	117° 01’ 00”	33.020	-14.998	29.415	200.621	-198.912	-45807.2	-42889.2
4	196° 53’ 50”	35.978	-34.424	-10.46	166.197	-209.372	-33058.6	-42004.4
5	271° 04’ 00”	48.760	0.909	-48.755	167.106	-258.127	-34987.3	-42899.9
1	13° 52’ 50”	33.882	32.894	8.126	200	-250.001	-51625.4	-41776.7
							-219383	-215236

$-258.127 \times 200 = -51625.4$

Area = $\dfrac{-219383-(-215236)}{2}$

$= 2073.8$

TRAVERSE APPLICATION



Stesen	Bearing Deg	Bearing Min	Bearing Sec	distance	Latit	Depart	Adjusted Latit	Adjusted Depart	Coordinate	Coordinate
1									200	-250
2	54	13	30	26.716	15.618	21.675	15.619	21.673	215.619	-228.327
3	117	01	00	33.020	-14.999	29.417	-14.998	29.415	200.621	-198.912
4	196	53	50	35.978	-34.425	-10.457	-34.424	-10.460	166.197	-209.371
5	271	04	00	48.760	0.908	-48.752	0.909	-48.755	167.106	-258.126
1	13	52	50	33.882	32.893	8.128	32.894	8.126	200.000	-250.000

TOTAL: 178.356 | -0.006 | 0.011

Linear Misclosure: 14,133.051

4.Area: 2,073.872

Figure 43 : Complete desktop application for traverse calculation

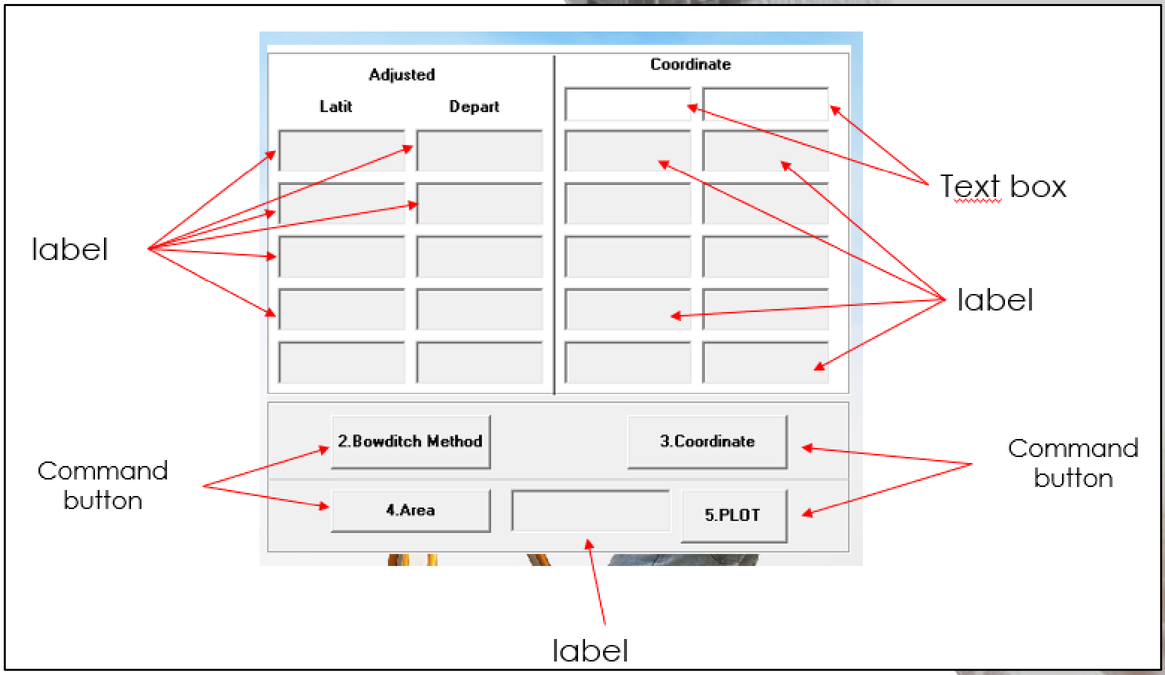


Figure 44: Form design for traverse calculation

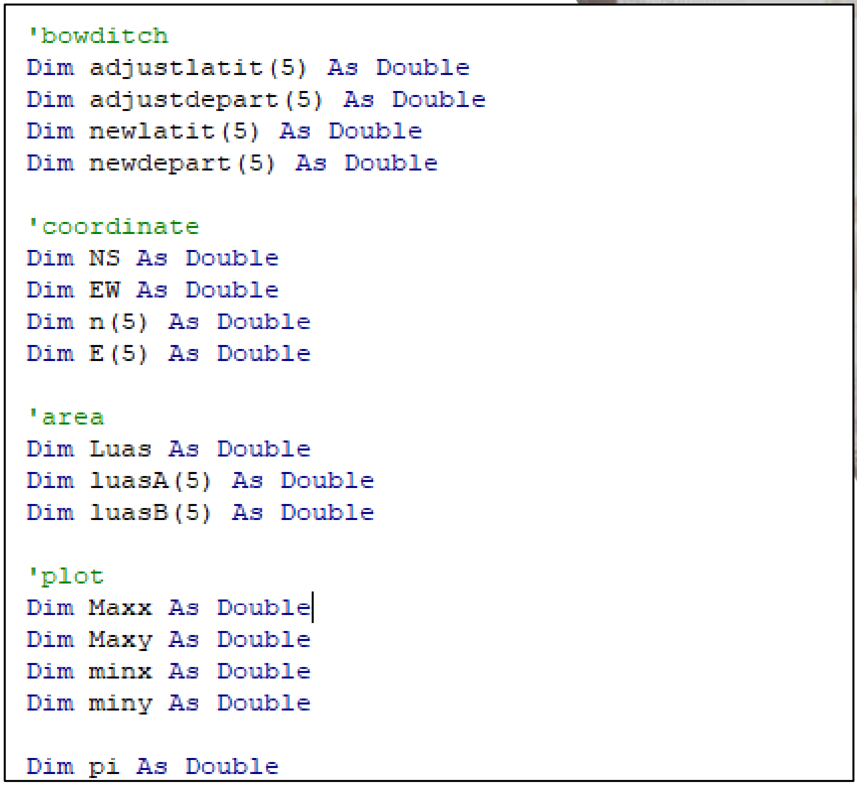
Table 11: description for each icon

Component	Name of icon	function
Text box	txt_coorN	Key in know coordinate N/S for 1st station
	txt_coorE	Key in know coordinate E/W for 1st station
Label	lbl_adjlatit1	To display new latit = correction + latit
	lbl_adjdepart2	To display new depart = correction + depart
	lbl_n1	To display coordinate N/S
	lbl_e1	To display coordinate E/W
	Lbl_area	To display traverse area
Command button	cmd_bowditch	Button to calculate adjusted latit and depart
	Cmd_coordinate	Button to calculate coordinate
	Cmd_area	Button to calculate cordinate

NOTE: rename all icon and use continuous number for each icon name.

For example: lbl_adjlatit1, lbl_adjlatit2, lbl_adjlatit3

CODING FOR CALCULATE BOWDITCH, COORDINATE, AREA AND PLOT TRAVERSE



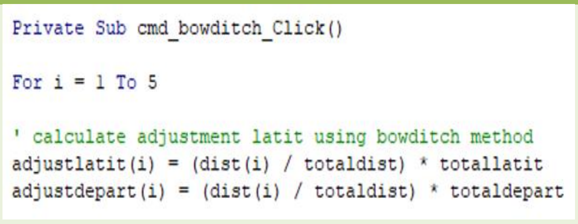
```
'bowditch
Dim adjustlatit(5) As Double
Dim adjustdepart(5) As Double
Dim newlatit(5) As Double
Dim newdepart(5) As Double

'coordinate
Dim NS As Double
Dim EW As Double
Dim n(5) As Double
Dim E(5) As Double

'area
Dim Luas As Double
Dim luasA(5) As Double
Dim luasB(5) As Double

'plot
Dim Maxx As Double
Dim Maxy As Double
Dim minx As Double
Dim miny As Double

Dim pi As Double
```

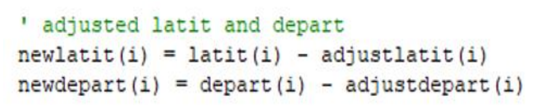
Figure 8: Register all variable

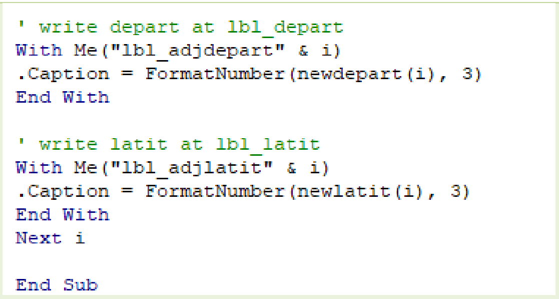
BOWDITCH METHOD

Table 12: coding for calculate adjusted latit and depart using Bowditch method

Coding	Explanation

 `Private Sub cmd_bowditch_Click()`
 	
`For i = 1 To 5`
 	
`' calculate adjustment latit using bowditch method`
`adjustlatit(i) = (dist(i) / totaldist) * totallatit`
`adjustdepart(i) = (dist(i) / totaldist) * totaldepart`
	• Write all coding in button cmd_bowditch
	• Use looping method to simplify the coding.
	• Adjustlatit / adjustdepart = correction value for each station using bowditch method.
	• $Adjustlatit1 = \frac{distance\ for\ stn\ 1}{\sum distance} \times \Delta latit$
	• $Adjusdepart1 = \frac{distance\ for\ stn\ 1}{\sum distance} \times \Delta depart$

 `' adjusted latit and depart`
`newlatit(i) = latit(i) - adjustlatit(i)`
`newdepart(i) = depart(i) - adjustdepart(i)`
	• newlatit1 = latit1 – correction for latit
	• newdepart1 = depart -correction for depart

 `' write depart at lbl_depart`
`With Me("lbl_adjdepart" & i)`
`.Caption = FormatNumber(newdepart(i), 3)`
`End With`
 	
`' write latit at lbl_latit`
`With Me("lbl_adjlatit" & i)`
`.Caption = FormatNumber(newlatit(i), 3)`
`End With`
`Next i`
 	
`End Sub`
	• display adjusted latit and depat and set the answer in 3 decimal places.
	• Use With and End With to display the answer.
	• Next i = loop the same coding until station 5.

COORDINATE CALCULATION

Table 13: Coding for calculate coordinate

CODING	Explanation

`Private Sub cmd_coordinate_Click()`
 	
`' reference coordinate`
`NS = Val(txt_coorN.Text)`
`EW = Val(txt_coorE.Text)`
	• Write all coding at cmd_coordinate
	
	• process to convert any value at text box to register variable.
	
	• Register know coordinate

`'coordunate station 2`
`n(1) = NS + newlatit(1)`
`E(1) = EW + newdepart(1)`
	• Formula to calculate coordinate for station 2
	• Coordinate station 2= know coordinate + adjusted latit

`' coordinate for next station`
`For i = 2 To 5`
`n(i) = n(i - 1) + newlatit(i)`
`E(i) = E(i - 1) + newdepart(i)`
 	
`Next i`
	• Using loop to calculate coordinate for next station.
	• Coordinate station 2=coordinate station 1+ adjusted latit

`' dispaly coordinate value`
 	
`For i = 1 To 5`
 	
`With Me("lbl_n" & i)`
`.Caption = FormatNumber(n(i), 3)`
 	
`End With`
 	
`With Me("lbl_e" & i)`
`.Caption = FormatNumber(E(i), 3)`
 	
`End With`
 	
`Next i`
 	
`End Sub`
	• Use loop to display all coordinate in 3 decimal places

AREA CALCULATION

Table14: Coding for calculate area

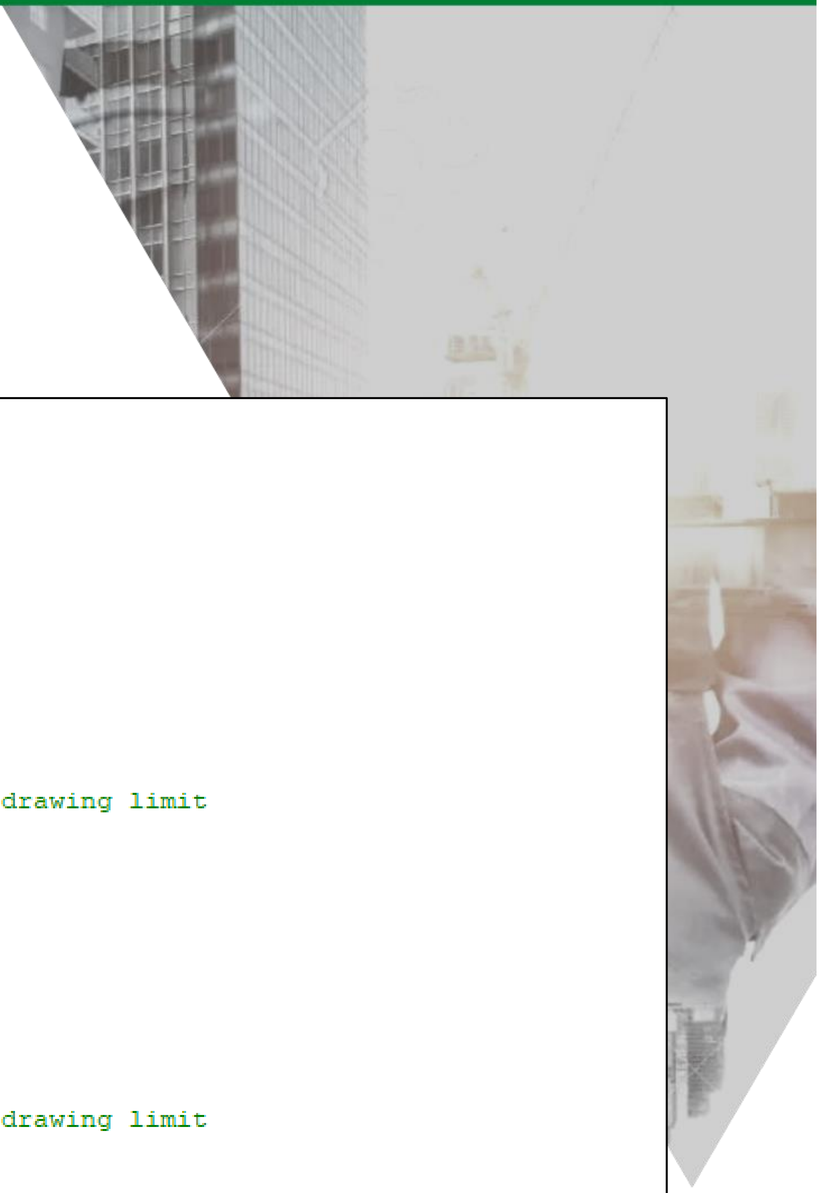
Coding	Explanation

`Private Sub cmd_area_Click()`
 	
`'calculate area`
`luasA(1) = EW * n(1)`
`luasB(1) = NS * E(1)`
	• Write all coding in cmd_area
	• Calculate area using coordinate method

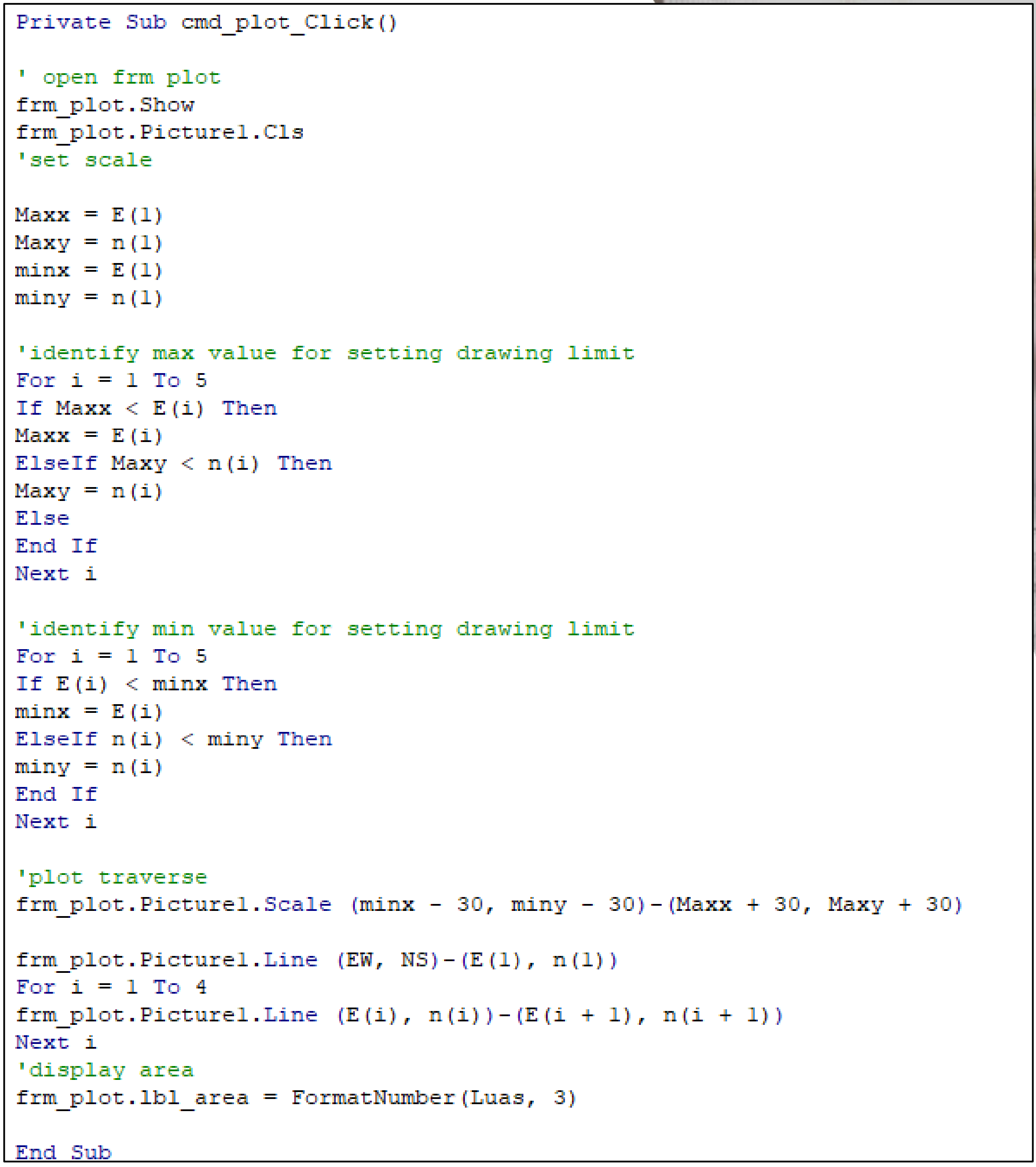
`For i = 2 To 5`
 	
`luasA(i) = luasA(i - 1) + (E(i - 1) * n(i))`
`luasB(i) = luasB(i - 1) + (n(i - 1) * E(i))`
`Next i`
	• Calculate area using coordinate method

`For i = 1 To 5`
`Luas = (luasA(i) - luasB(i)) / 2`
`Next i`
 	
`If Luas < 0 Then`
`Luas = Luas * -1`
`End If`
	• Calculate area for traverse

`'dispaly area value`
`lbl_area.Caption = FormatNumber(Luas, 3)`
 	
`End Sub`
	Display area at lbl_area with 3 decimal places



PLOT TRAVERSE



```
Private Sub cmd_plot_Click()

' open frm plot
frm_plot.Show
frm_plot.Picture1.Cls
'set scale

Maxx = E(1)
Maxy = n(1)
minx = E(1)
miny = n(1)

'identify max value for setting drawing limit
For i = 1 To 5
If Maxx < E(i) Then
Maxx = E(i)
ElseIf Maxy < n(i) Then
Maxy = n(i)
Else
End If
Next i

'identify min value for setting drawing limit
For i = 1 To 5
If E(i) < minx Then
minx = E(i)
ElseIf n(i) < miny Then
miny = n(i)
End If
Next i

'plot traverse
frm_plot.Picture1.Scale (minx - 30, miny - 30)-(Maxx + 30, Maxy + 30)

frm_plot.Picture1.Line (EW, NS)-(E(1), n(1))
For i = 1 To 4
frm_plot.Picture1.Line (E(i), n(i))-(E(i + 1), n(i + 1))
Next i
'display area
frm_plot.lbl_area = FormatNumber(Luas, 3)

End Sub
```

Figure 12: Coding for plot traverse

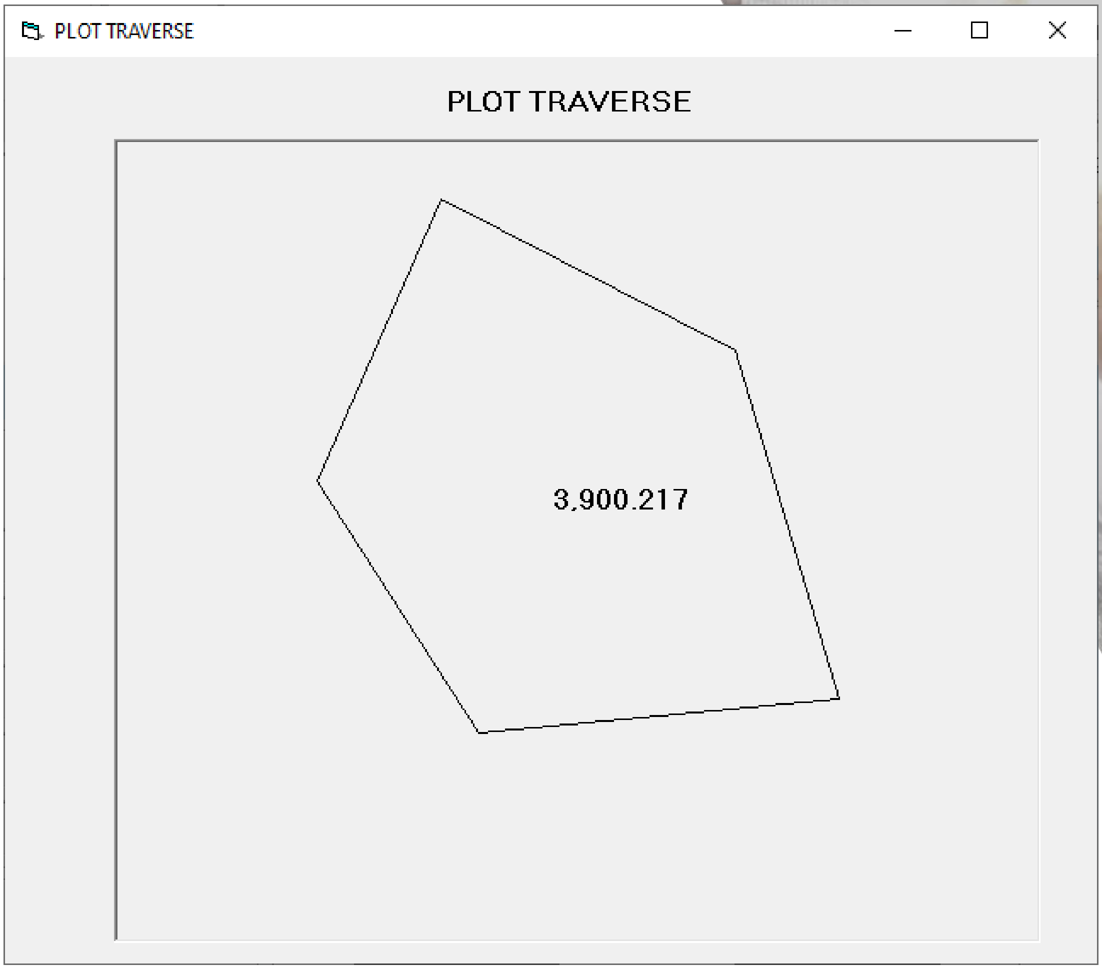
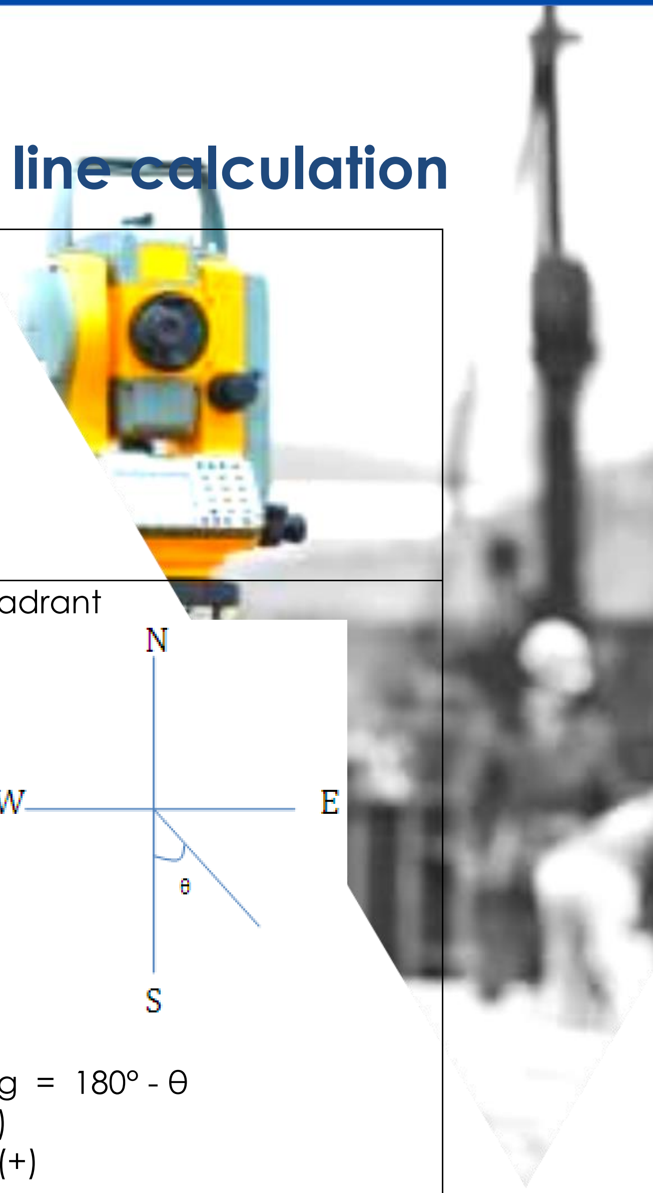


Figure 45: App for complete traverse calculation

CHAPTER 5

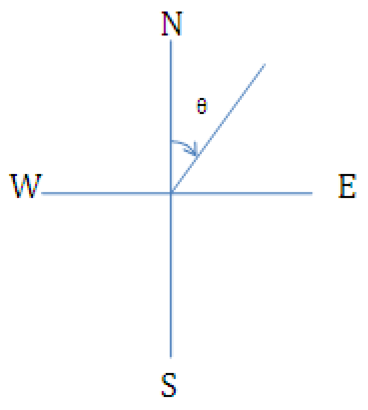
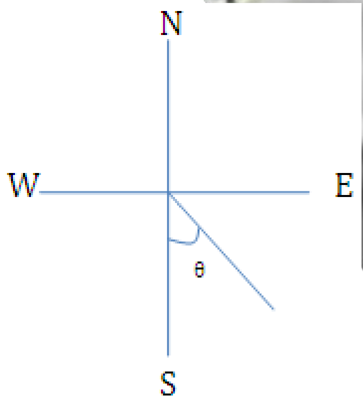
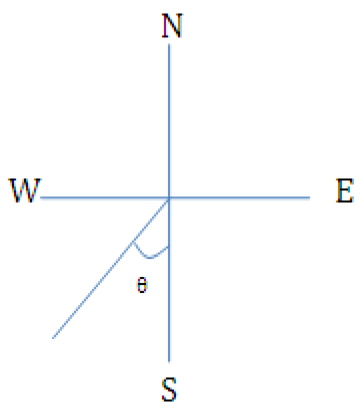
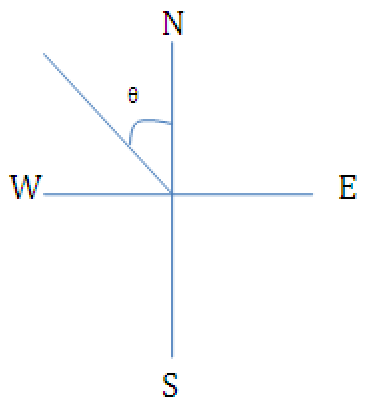
MISSING LINE

- Calculate bearing and distance from latit and depart value
- Calculate bearing and distance from2 know coordinate
- Calculate missing Line in traverse

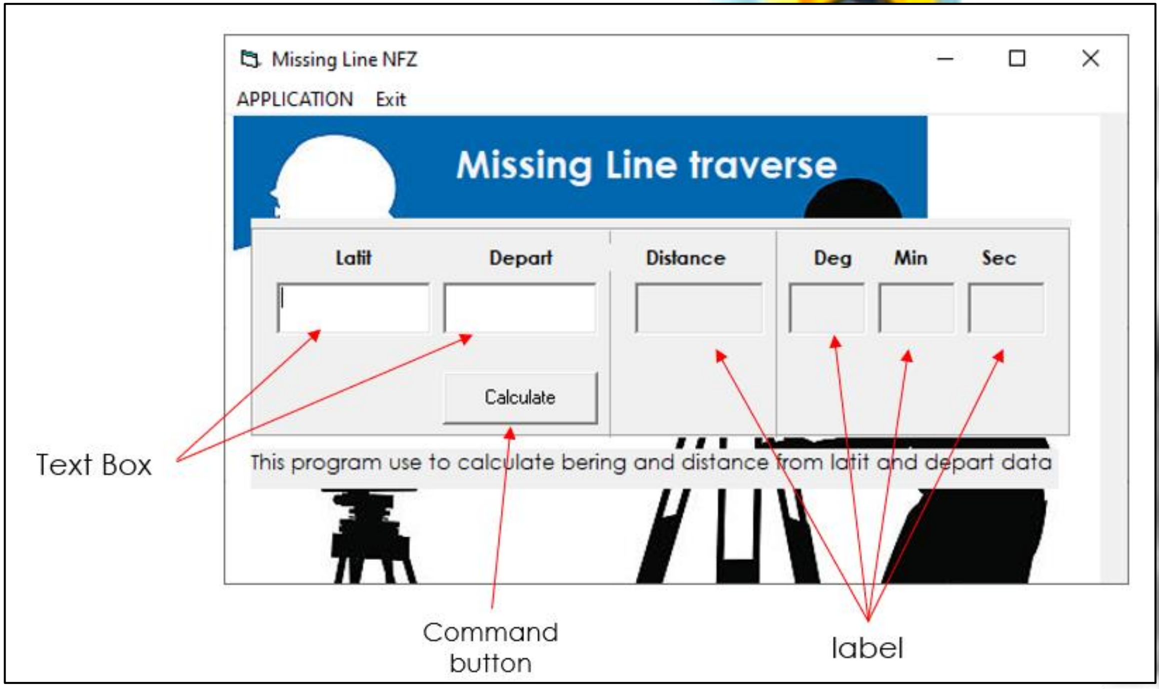


Basic Formula in missing line calculation

1. **Latitude = distance x cos θ**
2. **Departure = distance x sin θ**
3. **Distance =** $\sqrt{(latit)^2 + (depart)^2}$
4. **Angle =** $\tan^{-1}\dfrac{depart}{latit}$

1st quadrant  Bearing = value of θ Latit (+) Depart(+)	2nd quadrant  Bearing = 180° - θ Latit (-) Deprt (+)
3rd quadrant  Bearing = 180° + θ Latit (+) Depart (+)	4th quadrant  Bearing = 360° - θ Latit (+) Depart(-)

Calculate bearing and distance from latit and depart data

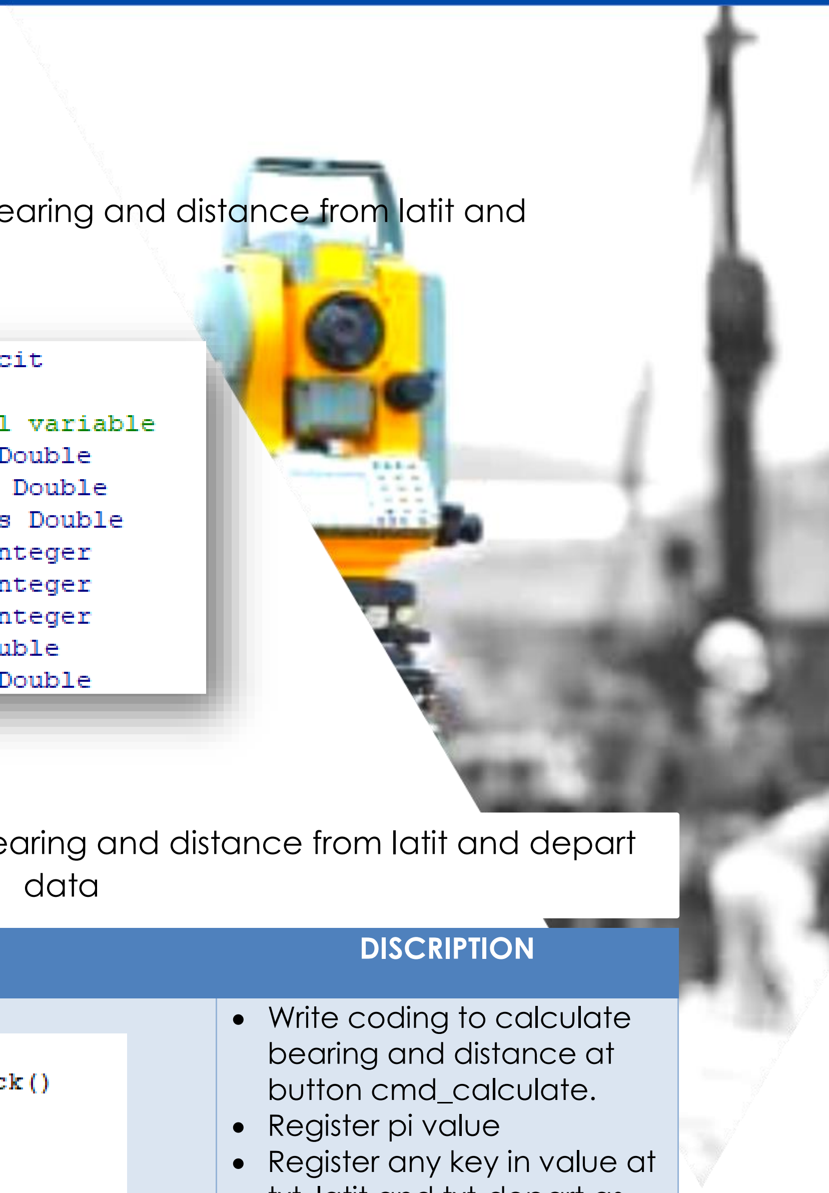


This program use to calculate bering and distance from latit and depart data

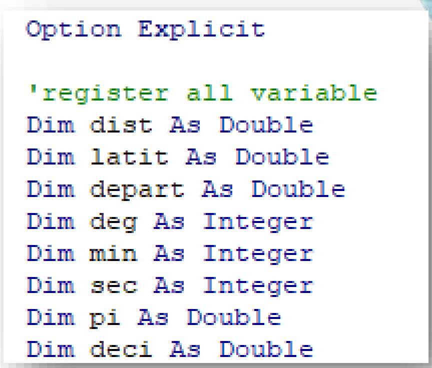
Figure 47:Missing Line application

Table 19: description for each icon

Component	Name of icon	function
Text box	txt_latit	Key in any value for latit
	txt_depart	Key in any value for depart
Label	lbl_dist	Display calculate distance
	lbl_deg	Display bearing (degree)
	lbl_min	Display bearing (minute)
	lbl_sec	Display bearing (second)
Command button	cmd_calculate	Calculate distance and bearing



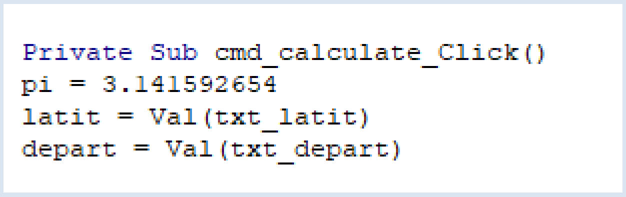
- Set variable to calculate bearing and distance from latit and depart data

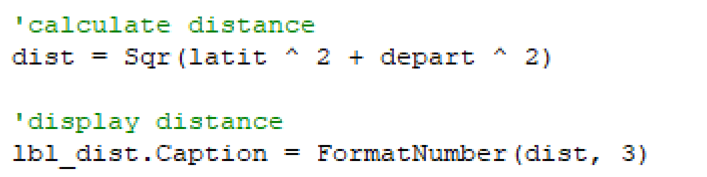


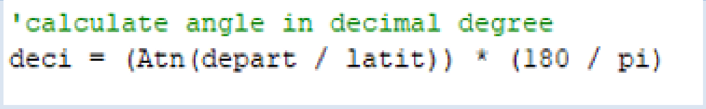
```
Option Explicit

'register all variable
Dim dist As Double
Dim latit As Double
Dim depart As Double
Dim deg As Integer
Dim min As Integer
Dim sec As Integer
Dim pi As Double
Dim deci As Double
```

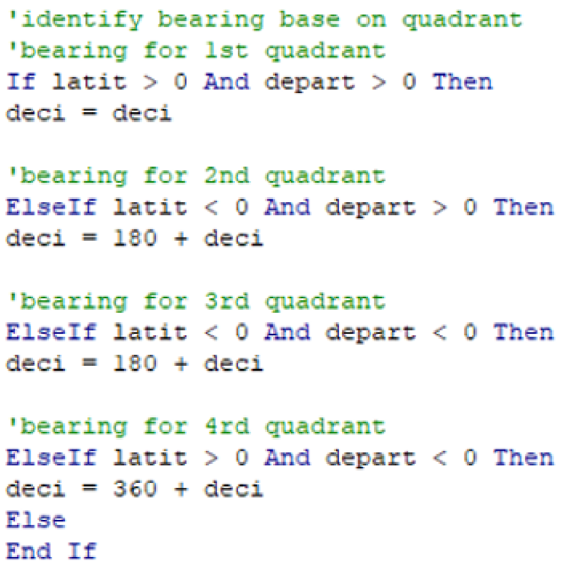
Table 20: Coding for calculate bearing and distance from latit and depart data

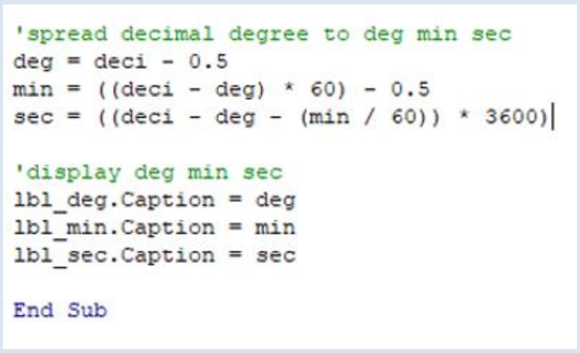
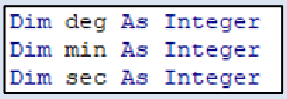
CODING	DISCRIPTION

 `Private Sub cmd_calculate_Click()`
`pi = 3.141592654`
`latit = Val(txt_latit)`
`depart = Val(txt_depart)`
	• Write coding to calculate bearing and distance at button cmd_calculate.
	• Register pi value
	• Register any key in value at txt_latit and txt-depart as latit and depart.

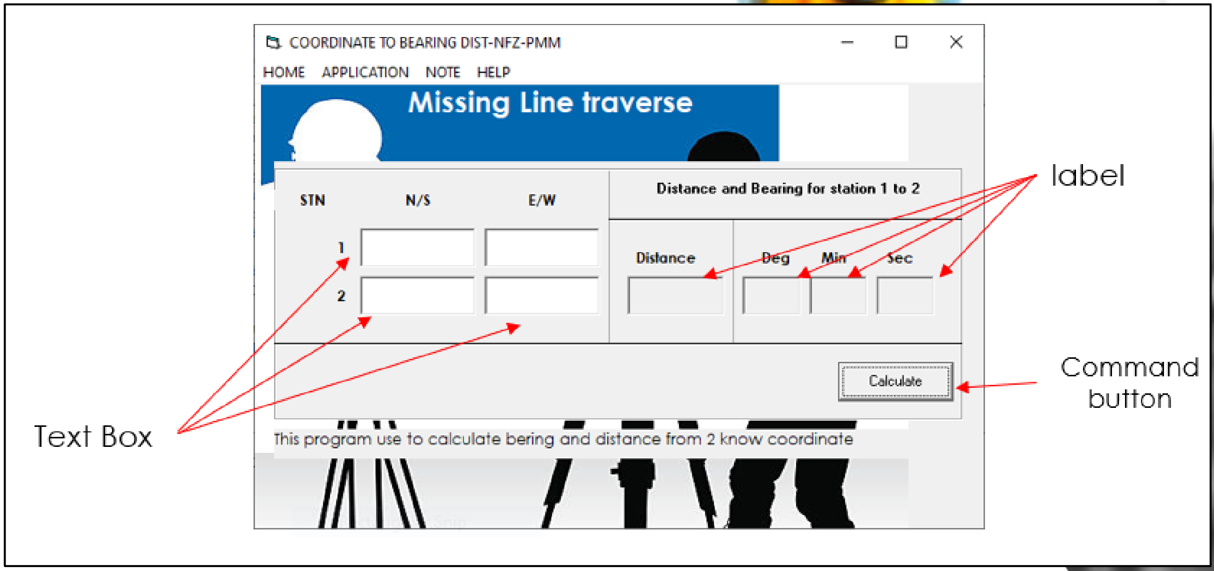
 `'calculate distance`
`dist = Sqr(latit ^ 2 + depart ^ 2)`
 	
`'display distance`
`lbl_dist.Caption = FormatNumber(dist, 3)`
	• Calculate distance using formula
	$dist = \sqrt{(latit)^2 + (depart)^2}$
	• Display distance in 3 decimal places

 `'calculate angle in decimal degree` `deci = (Atn(depart / latit)) * (180 / pi)`	• Calculate angle formula $\theta = \tan^{-1}\frac{depart}{latit}$ • Convert θ in radian to decimal degree $\theta\, rad = \theta \times \frac{180}{\pi}$

 `'identify bearing base on quadrant`
`'bearing for 1st quadrant`
`If latit > 0 And depart > 0 Then`
`deci = deci`
 	
`'bearing for 2nd quadrant`
`ElseIf latit < 0 And depart > 0 Then`
`deci = 180 + deci`
 	
`'bearing for 3rd quadrant`
`ElseIf latit < 0 And depart < 0 Then`
`deci = 180 + deci`
 	
`'bearing for 4rd quadrant`
`ElseIf latit > 0 And depart < 0 Then`
`deci = 360 + deci`
`Else`
`End If`
	• Identify the value of latit and depart to determine the quadrant.
	• lati t > 0 = positif value of latit
	• latit < 0 = negative value of latit

 `'spread decimal degree to deg min sec`
`deg = deci - 0.5`
`min = ((deci - deg) * 60) - 0.5`
`sec = ((deci - deg - (min / 60)) * 3600)`
 	
`'display deg min sec`
`lbl_deg.Caption = deg`
`lbl_min.Caption = min`
`lbl_sec.Caption = sec`
 	
`End Sub`
	• display result in degree, minute and second.
	• Register deg, min and sec as integer.
	 `Dim deg As Integer`
	`Dim min As Integer`
	`Dim sec As Integer`
	• Display all the result at lbl_deg for degree value, lbl_min for minute value, lbl_sec for second value.

Calculate bearing and distance from 2 know coordinate


COORDINATE TO BEARING DIST-NFZ-PMM
HOME APPLICATION NOTE HELP
Missing Line traverse
STN
N/S
E/W
1
2
Distance and Bearing for station 1 to 2
Distance
Deg
Min
Sec
Calculate
label
Command button
Text Box

This program use to calculate bering and distance from 2 know coordinate

Figure48: Form design for traverse calculation

- Set suitable variable to calculate bearing and distance from 2 known coordinate.

```
Option Explicit
'register all variable
Dim dist As Double
Dim latit As Double
Dim depart As Double
Dim deg As Integer
Dim min As Integer
Dim sec As Integer
Dim pi As Double
Dim deci As Double
```

Table21 : Coding for calculate bearing and distance from 2 known coordinate

CODING

```
Private Sub cmd_calculate_Click()

pi = 3.141592654
' calculate latit from N/S coordinate
latit = Val(txt_coorN1) - Val(txt_coorN2)

'calculate depart from E/W coordinate
depart = Val(txt_coorE1) - Val(txt_coorE2)

'calculate distance
dist = Sqr(latit ^ 2 + depart ^ 2)

'display distance
lbl_dist.Caption = FormatNumber(dist, 3)
```

DESCRIPTION

Calculate latit = 1st coordinate North – 2nd coordinate North

Calculate depart = 1st coordinate east – 2nd coordinate east

CODING

```
'calculate angle in decimal degree
deci = (Atn(depart / latit)) * (180 / pi)

'identify bearing base on quadrant
'bearing for 1st quadrant
If latit > 0 And depart > 0 Then
deci = deci

'bearing for 2nd quadrant
ElseIf latit < 0 And depart > 0 Then
deci = 180 + deci

'bearing for 3rd quadrant
ElseIf latit < 0 And depart < 0 Then
deci = 180 + deci

'bearing for 4rd quadrant
ElseIf latit > 0 And depart < 0 Then
deci = 360 + deci
Else
End If


'spread decimal degree to deg min sec
deg = deci - 0.5
min = ((deci - deg) * 60) - 0.5
sec = ((deci - deg - (min / 60)) * 3600)

'display deg min sec
lbl_deg.Caption = deg
lbl_min.Caption = min
lbl_sec.Caption = sec


End Sub
```

DESCRIPTION

Same with previous explanation

Calculate bearing and distance for missing line at traverse

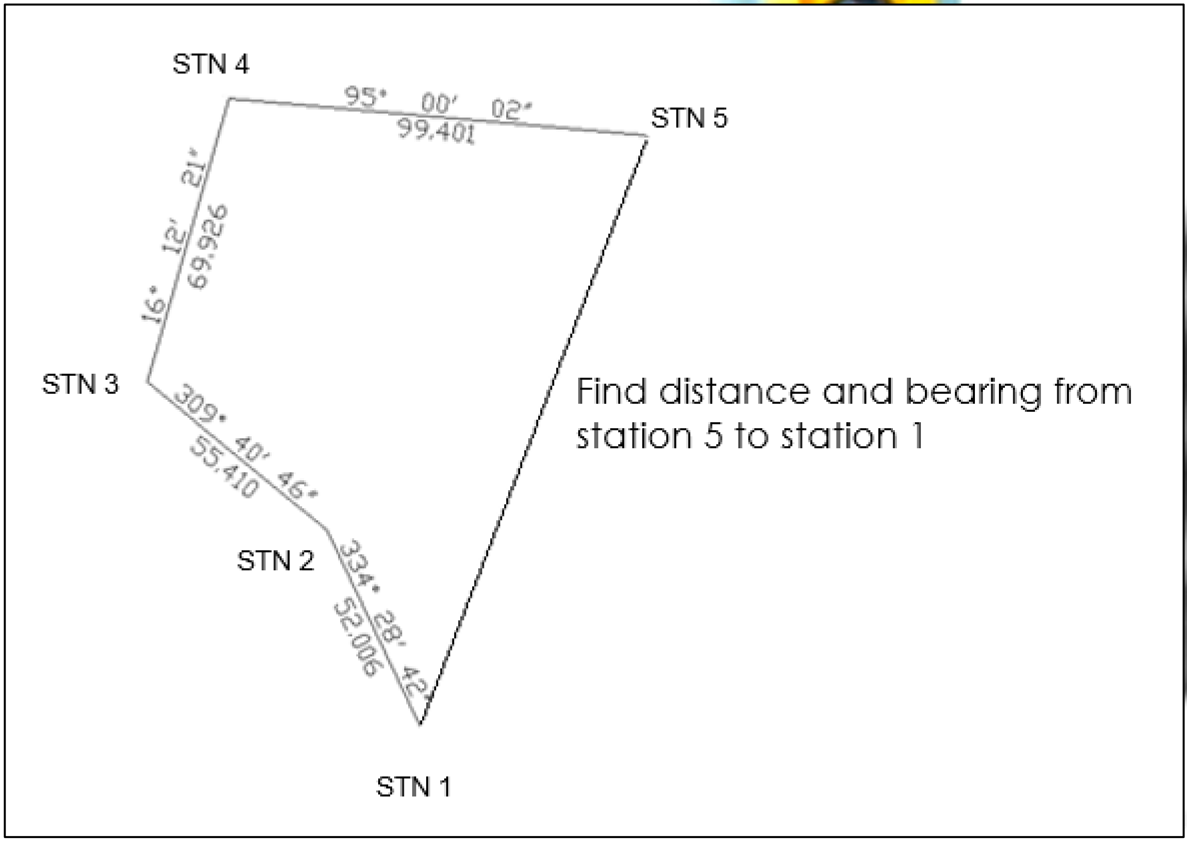


Figure 49: Example situation for missing line at traverse

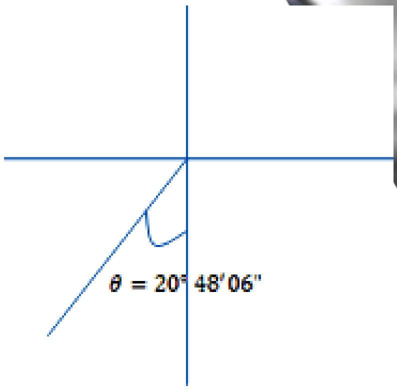
LINE	BEARING	DISTANCE	LATIT	DEPART
1-2	334º28'42"	52.006	46.931	-22.407
2-3	309º40'46"	55.410	35.379	-42.645
3-4	16º12'21"	69.926	67.148	19.516
4-5	95º00'02"	99.401	-8.664	99.023
5-1			**-140.794**	**-53.487**
		check	**0.000**	**0.000**

Distance 5-1 = $\sqrt{140.794^2 + 53.487^2}$
= 150.611

$$\tan\theta = \frac{depart}{latit}$$

$$\theta = \tan^{-1}\frac{53.487}{140.794}$$

$$\theta = 20°48'06"$$



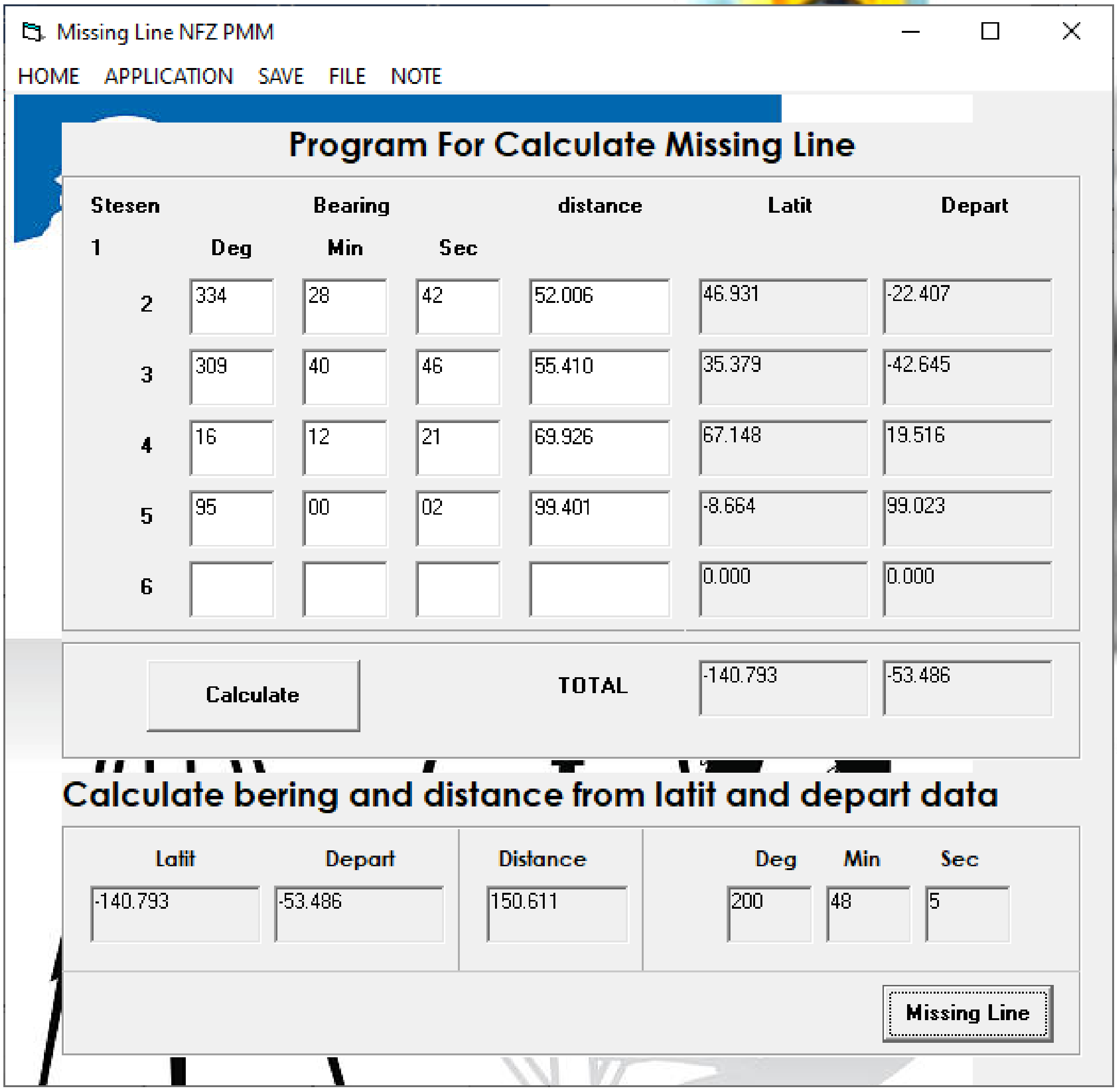
Latit (N/S)	Depart (E/W)
-140.794	**-53.487**

Bearing 5-1 $= 180° + 20°48'06"$
= 200º 48' 06"

Identify the bearing position base on latit and depart data.

LINE	BEARING	DISTANCE	LATIT	DEPART
5-1	200º 48' 06"	150.611	**-140.794**	**-53.487**

APPLICATION FOR CALCULATE MISSING LINE FOR TRVERSE



Program For Calculate Missing Line

Stesen	Bearing Deg	Min	Sec	distance	Latit	Depart
1						
2	334	28	42	52.006	46.931	-22.407
3	309	40	46	55.410	35.379	-42.645
4	16	12	21	69.926	67.148	19.516
5	95	00	02	99.401	-8.664	99.023
6					0.000	0.000
TOTAL					-140.793	-53.486

Calculate bering and distance from latit and depart data

Latit	Depart	Distance	Deg	Min	Sec
-140.793	-53.486	150.611	200	48	5

Figure : Missing line app for traverse

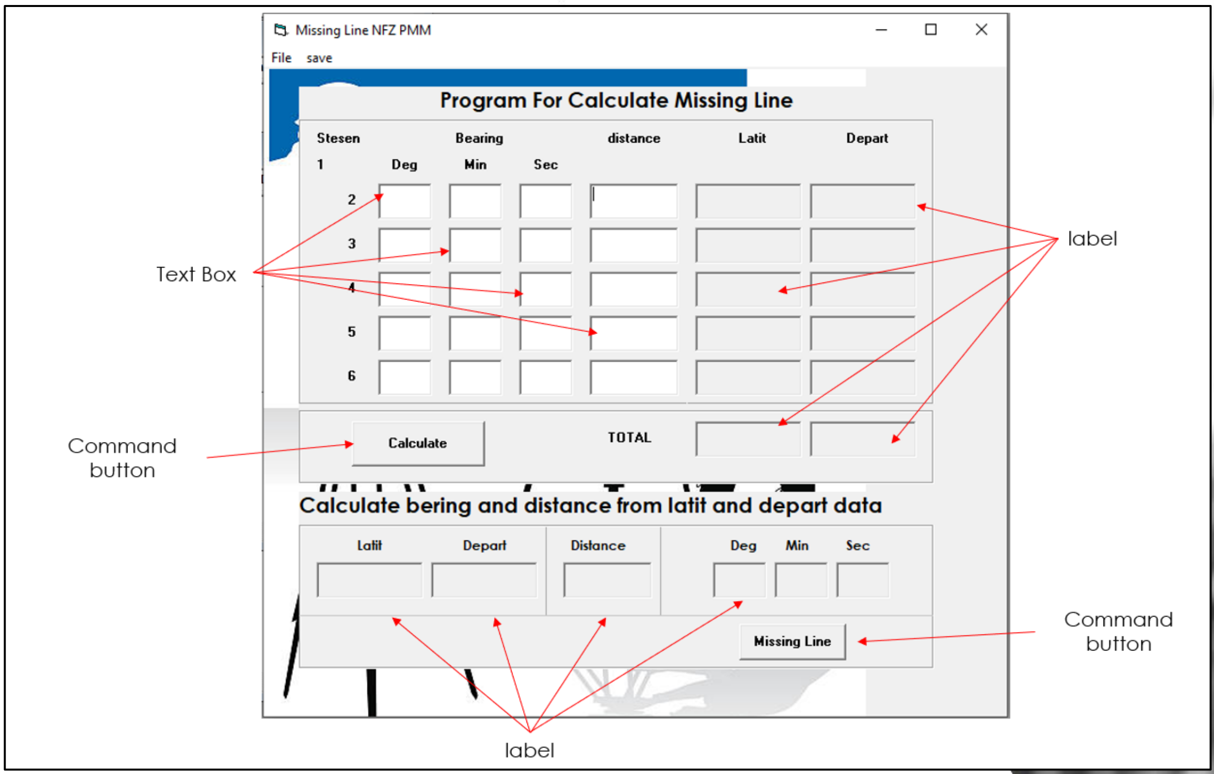
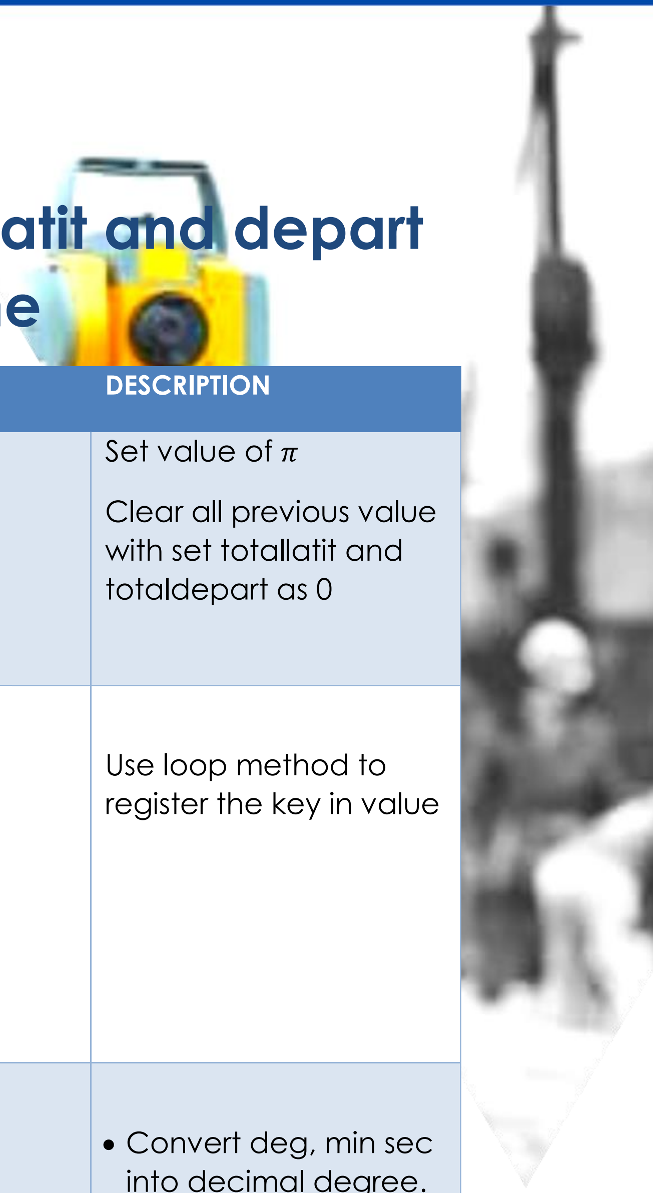


Figure 50: Form design for missing line traverse calculation

1. First step, find out latit and depart for missing line
2. Calculate bearing and distance from latit and depart data

- First set suitable variable for app missing line

```
'calculation for latit depart
Dim deg(6)  As Integer    ' degree
Dim min(6) As Integer      ' min
Dim sec(6) As Integer      ' sec
Dim latit(6) As Double   ' latit
Dim depart(6) As Double  ' depart
Dim dist(6) As Double    ' distance
Dim i As Integer         ' loop
Dim deci(6) As Double        ' decimal degree
Dim rad(6) As Double         'angle in radian

'identify total latit & depart
Dim totallatit As Double ' total latit
Dim totaldepart As Double ' total depart


Dim pi As Double
```



1. Coding for calculate latit and depart for traverse missing line

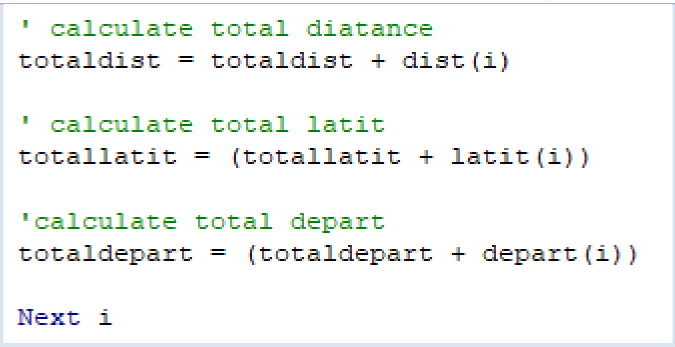
CODING	DESCRIPTION

`Private Sub cmd_calc_Click()`
 	
`pi = 3.141592654`
`'refresh the value of latit and depart`
`totallatit = 0`
`totaldepart = 0`
	Set value of π
	
	Clear all previous value with set totallatit and totaldepart as 0

`' loop i until 5`
`For i = 1 To 5`
 	
`' register val of txt_dist as distance`
`dist(i) = Val(Me("txt_dist" & i))`
 	
`'register for degree`
`deg(i) = Val(Me("txt_deg" & i))`
`min(i) = Val(Me("txt_min" & i))`
`sec(i) = Val(Me("txt_sec" & i))`
	Use loop method to register the key in value

`'calculate latit & depart`
`'1.convert to decimal degree`
`deci(i) = deg(i) + min(i) / 60 + sec(i) / 3600`
 	
`'2. convert to radian`
`rad(i) = deci(i) * pi / 180`
 	
`'3. latit & depart`
`latit(i) = dist(i) * Cos(rad(i))`
`depart(i) = dist(i) * Sin(rad(i))`
	• Convert deg, min sec into decimal degree.
	• Convert decimal degree to radian
	• Calculate latit and depart

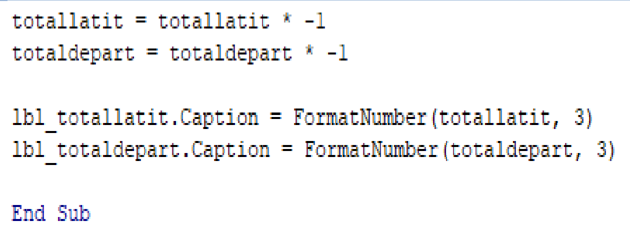
`' write latit at lbl_latit`
`With Me("lbl_latit" & i)`
`.Caption = FormatNumber(latit(i), 3)`
 	
`End With`
 	
`'' write depart at lbl_depart`
`With Me("lbl_depart" & i)`
`.Caption = FormatNumber(depart(i), 3)`
`End With`
	• Display latit and depart with 3 decimal places



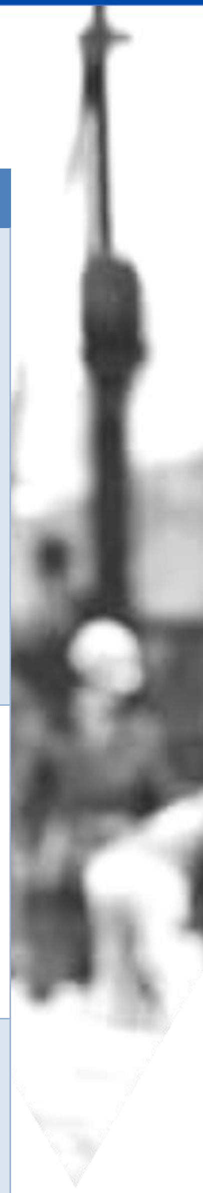
```
' calculate total diatance
totaldist = totaldist + dist(i)

' calculate total latit
totallatit = (totallatit + latit(i))

'calculate total depart
totaldepart = (totaldepart + depart(i))

Next i
```

Totallatit = different of latit (missing latit)

Total depart = different of depart (missing depart)



```
totallatit = totallatit * -1
totaldepart = totaldepart * -1

lbl_totallatit.Caption = FormatNumber(totallatit, 3)
lbl_totaldepart.Caption = FormatNumber(totaldepart, 3)

End Sub
```



CODING FOR CALCULATE MISSING LINE

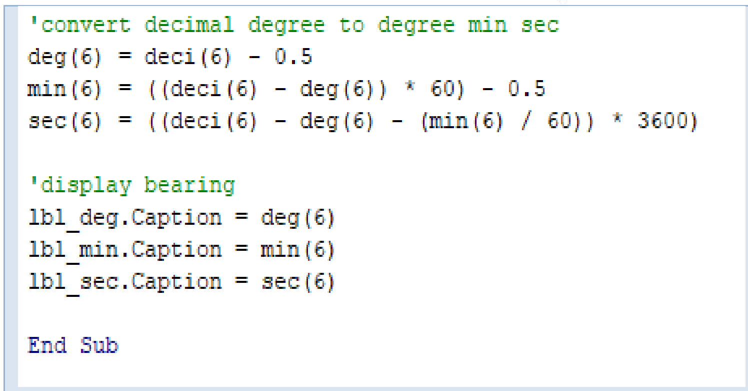
CODING	DESCRIPTION

`Private Sub cmd_missingline_Click()`
 	
`'display latit and depart from pevious calculation`
`lbl_latit6.Caption = FormatNumber(totallatit, 3)`
`lbl_depart6.Caption = FormatNumber(totaldepart, 3)`
 	
`' register total latit as latit for missing line`
`latit(6) = totallatit`
 	
`' register total depart as depart for missing line`
`depart(6) = totaldepart`
	Register latit and depart as latit6 and depart6

`' calculate distance for missing line`
`dist(6) = Sqr(latit(6) ^ 2 + depart(6) ^ 2)`
 	
`'display distance`
`lbl_dist.Caption = FormatNumber(dist(6), 3)`
	Calculate distance and display distance with 3 decimal placeses

`' calculate bering in decimal degree`
`deci(6) = (Atn(depart(6) / latit(6))) * (180 / pi)`
	Calculate bearing
	
	Answer in decimal degree

`' bearing for 1st quadrant`
`If latit(6) > 0 And depart(6) > 0 Then`
`deci(6) = deci(6)`
 	
`'bearing for 2nd quadrant`
`ElseIf latit(6) < 0 And depart(6) > 0 Then`
`deci(6) = 180 + deci(6)`
 	
`' bearing for 3rd quadrant`
`ElseIf latit(6) < 0 And depart(6) < 0 Then`
`deci(6) = 180 + deci(6)`
 	
`' bearing for 4rd quadrant`
`ElseIf latit(6) > 0 And depart(6) < 0 Then`
`deci(6) = 360 + deci(6)`
`Else`
`End If`
	Identify the quarant for bearing
	
	Answer for bearing from calculation = 0 to 90
	
	If latit (+) and depart (+) = 1st quadrant
	
	If latit(-) depart (+) = 2nd quadrant
	
	If latit (-) and depart (-) =3rd quadrant
	
	If latit (+) and depart (-) = 4rd quadrant



```
'convert decimal degree to degree min sec
deg(6) = deci(6) - 0.5
min(6) = ((deci(6) - deg(6)) * 60) - 0.5
sec(6) = ((deci(6) - deg(6) - (min(6) / 60)) * 3600)

'display bearing
lbl_deg.Caption = deg(6)
lbl_min.Caption = min(6)
lbl_sec.Caption = sec(6)

End Sub
```

Convert decimal degree to degree, min sec and display each answer

SMART CALCS
HOME APPLICATION FEEDBACK HELP
SMART CALCS
TRAVERSE
MISSING LINE
CONVERTER
CURVE
Missing Line travers
Offset From Deflection Angle

REFERENCE

REFERENCES

Liew Voon Kiong(2006)Visual Basic ® 6 Made Easy: A Complete Tutorial for Beginner, BookSurge Publishing

Lou Tylee. (1998). Course Note for: Learn Visual Basic 6.0.

Gary Haggard, Wade Hutchison & Christy Shibata. (2013). Introduction: Visual Basic 6.0 First Edition. ISBN 978-87-403-0341-4

David I. Schneider. (1999). Computer Programming Concepts and Visual Basic

VISUAL BASIC PROGRAMMING provides students with knowledge of the programming concepts using the Visual Basics programming language. The course emphasizes on design the programme which includes examining code, looping statement and also creates and documents naming standards.

This book is written specifically to satisfy the syllabus requirements for subject DCG 50252 Visual Basic Programming. This book contains all required topics for Diploma Geomatic.

e ISBN 978-967-2762-17-1

9 7 8 9 6 7 2 7 6 2 1 7 1

VISUAL BASIC 6.0