

Programming with Visual Studio Higher (v. 2013)



Contents/Requirements Checklist

Multiple selection: using ifs & case	
While Loops	
Using arrays	
Filling arrays	
Displaying array contents	
Types of variables: string, integer, single, double, boolean	
Use of subroutines	
Concatenation and substrings	
Local and global variables	
Scope	
Passing parameters between sub routines call by reference/value,	
Using random function	
Standard Algorithms	
• max	
• min	
• linear search	
• count occurrence	

You are going to start your Higher programming by revising some of the programming you covered in Standard Grade: notably multiple selection, the use of arrays and real and integer variables and the use of While loops to validate data.

If you need further revision of your Standard Grade programming: work your way through the General and Credit level notes!
Time to get started!

Higher Example 1

Multiple Outcome Selection / Case

The First multiple outcome selection is the:

If.then ..elseif..elseif..else end if

You can use **If.then ..elseif..elseif..else end if** to allow your program to select between a range of options.

1. Create a **New Project** in Visual Studio.
2. Name the Project **Higher Example 1**.
3. Create a Form with one **Button** in the centre.

Label the button **Start**.

Double click the button to enter the code.

4. **Type the code** on the right for that action Button click.
5. **Test** your program by running it and then entering the different names. This will then display the multiple comment outcomes.
6. Use the **Save All** option to save the program as:

Higher Example 1

```
Dim name As String
Dim comment As String

name = InputBox("Enter a name")

If name = "Jack" Then
    comment = "Hi Jack"
ElseIf name = "Mack" Then
    comment = "Hi Mack"
ElseIf name = "Ed" Then
    comment = "Hi ED "
Else
    comment = "Who are you? "
End If

MsgBox(comment)
```

Higher Example 2

Multiple Outcome Selection / Case

The second multiple outcome selection is the:

Case...Case Is...Select

You can use **Case...Case Is...Select** to allow your program to select between a range of options.

1. Create a **New Project** in Visual Studio.
2. Name the Project **Higher Example 2**.
3. Create a Form with one **Button** in the centre.

Label the button **Start**.

Double click the button to enter the code.

4. **Type the code** on the right for that action Button click.
5. **Test** your program by running it and then entering the different names. This will then display the multiple comment outcomes.
6. Use the **Save All** option to save the program as:

Higher Example 2

```
Dim name As String  
  
name = InputBox("Enter a name")  
  
Select Case name  
  
    Case Is = "Jack"  
        MsgBox("Hi Jack")  
    Case Is = "Mack"  
        MsgBox("Hi Mack")  
    Case Else  
        MsgBox("Who are you?")  
  
End Select
```

Higher Task 1

Write a program to process an exam mark using the following table:

Mark out of 100	Award
80-100	Grade 1
70-79	Grade 2
60-69	Grade 3
50-59	Grade 4
<50	No Award

You Must:

1. Use either *If.then ..elseif..elseif..else end if* or **Case...Case Is...Select**
2. Use a Listbox to display the names, marks and award.
3. Write the **Pseudocode** before you begin and show it to your teacher.
4. Test it then save it as **Higher Task 1**.

Higher Example 3

Do While ... Loop

When you use a **Do While Loop** the condition is at the start of a loop.

In the next example you will **Validate Higher Task 1**.

1. Create a **New Project** in Visual Studio.
2. Name the Project **Higher Example 3**.
3. Create the Form design shown on the right (add a listbox and button).
4. Double click the Button to enter the code below.



```
Dim name As String
Dim mark As Integer
Dim award As String
Dim counter As Integer

For counter = 1 To 5
    name = InputBox("Pupil's name")
    mark = InputBox("Pupil's mark")

    Do While mark < 0 Or mark > 100
        mark = InputBox("Mark out of range. Please enter a mark between 0 and 100.")
    Loop

        If mark > 79 And mark < 101 Then
            award = "A"
        ElseIf mark > 69 And mark < 80 Then
            award = "B"
        ElseIf mark > 59 And mark < 70 Then
            award = "C"
        ElseIf mark > 49 And mark < 60 Then
            award = "D"
        ElseIf mark > -1 And mark < 50 Then
            award = "No Award"
        End If

    name_mark_award.Items.Add(name & " " & mark & " " & award)

Next
```

Use the **Save All** option to save the program as:

Higher Example 3

5. **Test** your program by running it and then entering the following test data into a Word document (Save it as Higher Example 1 Test Data) or your jotter:

Test Data	Expected Outcome	Actual Outcome
Normal data		
30	Accepted – “No Award”	
55	Accepted – “D”	
65	Accepted – “C”	
75	Accepted – “B”	
95	Accepted – “A”	
Extreme		
100	Accepted – “A”	
101	Out of Range Message	
0	Accepted – “No Award”	
-1	Out of Range Message	
Exceptional		
-876	Out of Range Message	
567	Out of Range Message	
345.67	Out of Range Message	

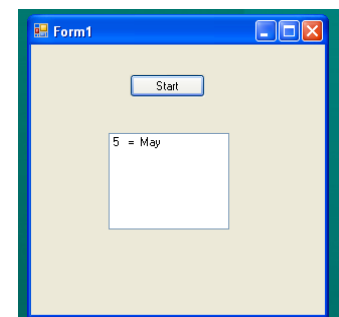
Higher Task 2

Write a program that will take in a number and output that specific month of the year. The program will also check that a month being entered as a number is greater than 0 and less than 13 (validation).

For example: The user inputs 5. The program outputs May.

You Must:

1. Use either *If.then ..elseif..elseif..else end if* or **Case...Case Is...Select**
2. Use a Listbox to display the number and month.
3. Write the **Pseudocode** before you begin and show it to your teacher.
4. Test it using similar test data table as the example.
5. Save it as **Higher Task 2**.



Using Arrays

An **array** is a list of numbers or words. Each array has a name.

Eg 1: This array is called **Name**.

INDEX	0	1	2	3
Name	Fred	Ted	Ed	Ned

Each array has an **index**. The index is used to count down (or up) the array.

Note: visual basic starts counting from 0.

So **Name(0)** is Fred, **Name(1)** is Ted, **Name(2)** is Ed and **Name(3)** is Ned.

Setting up and filling an array

The first thing you have to do is set up your array using the DIM instruction:

e.g. DIM Name(4) as string.
 size type

This sets up an array called **Name** with **5** spaces (remember Name(0), Name(1), Name(2), Name(3) and Name(4) making 5 in total) which can hold **text (string)**.

You have to use a loop with index to fill the array.

```
For index = 0 To 4  
    name(index) = InputBox("enter a name ")  
Next index
```

Don't forget to set up your index

DIM index as integer

Now to do an example!

Higher Example 4

In this example you will learn how to use an **Array**, as well as setting and filling the array.

Pseudocode:

Copy the Pseudocode below into a **new word document**.

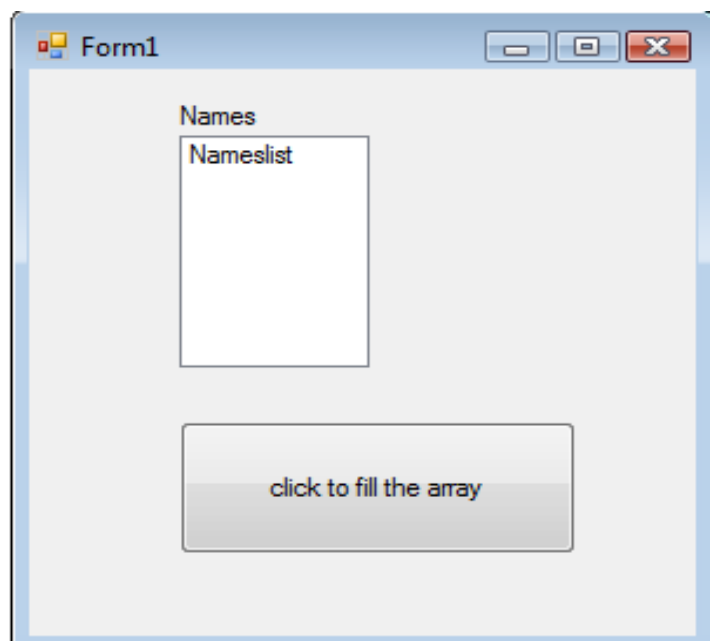
Save the Pseudocode as Higher Example 4.

1. Initialise
 2. loop to fill array
 3. loop to display array contents
-
- 1.1 Set up array called **name** with 5 spaces
 - 1.2 set up index as an integer
-
- 2.1 loop 5 times
 - 2.2 input name into array using index
-
- 3.1 loop 5 times
 - 3.2 display names in array using index

Create the program:

1. Open a new project.
Call it Higher Example 4.
2. Add a button to start filling the array.
3. Add a **listbox** to display the names in your array . Call it **Nameslist**.
4. Add a label above your listbox. the text of the label should read '**Names**'.

The form design should look like the image on the right.



Information:

Before you add your code: remember: Visual Basic starts numbering from 0.

Code the Button:

Now add the following code to your button by double clicking on the button.

```
Dim name(4) As String
Dim index As Integer

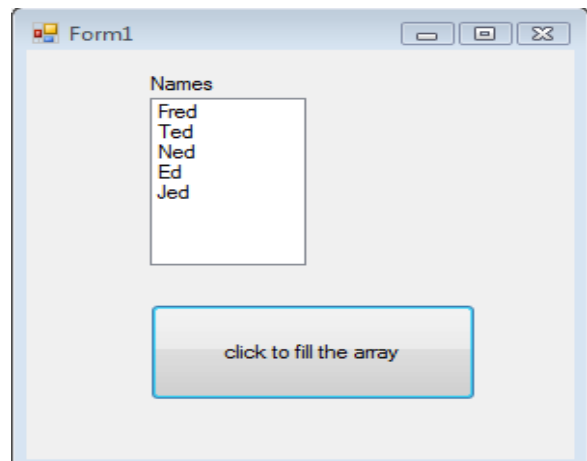
For index = 0 To 4
    name(index) = InputBox("Enter a name")
Next index

For index = 0 To 4
    nameslist.Items.Add(name(index))
Next index
```

Test your program:

Test your program by entering 5 names.

The image on the right shows a list of 5 names that were entered into the array and then displayed successfully.



Save your Work:

Save your project as **Higher Example 4**.

Higher Example 5

Using one loop to fill the array and another loop to display the names means that the names don't appear in the listbox until you have entered all the names.

Repeat your last program (Higher Example 4) but change it so that the names appear in as you type them in. Change the coding to:

```
Dim index As Integer
Dim name(4) As String

For index = 0 To 4
    Name(index) = InputBox("Enter a name")
    nameslist.Items.Add(Name(index))
Next index
```

Information:

*This version only uses 1 loop and so is more **efficient**.*

Test your program and save it as **Higher Example 5**.

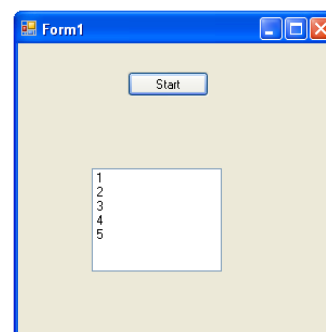
Higher Task 3

Write a program that inputs 5 whole numbers into an array and displays them in a listbox.

For example: The user inputs 1, 2, 3, 4 and 5. The program outputs 1, 2, 3, 4 and 5 in the listbox.

You Must:

1. Use an Array.
2. Use a Listbox to display the numbers.
3. Write the **Pseudocode** before you begin and show it to your teacher.
4. Test it.
5. Save it as **Higher Task 3**.



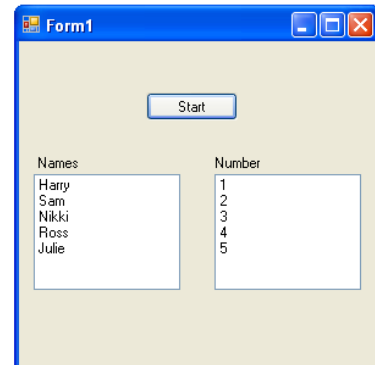
Higher Task 4

Write a program that inputs 5 names into array an array and 5 whole numbers into an array and displays them, each in their own listbox.

For example: The user inputs Harry 1, Sam 2, Nikki 3, Ross 4 and Julie 5. The program outputs Harry 1, Sam 2, Nikki 3, Ross 4 and Julie 5 their separate listboxes.

You Must:

1. Use an Array.
2. Use separate Listboxes to display the names and numbers.
3. Write the Pseudocode before you begin and show it to your teacher.
4. Test it.
5. Save it as Higher Task 4.



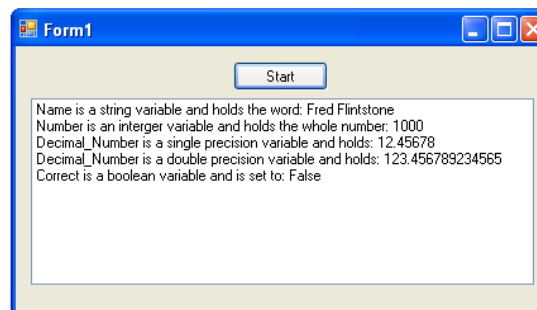
Higher Example 6

Types of Variable

1. Create a new project with one button and a listbox on the form.
2. Enter the following code to remind you of the different types of variable you can use.

```
Dim name As String = "Fred Flintstone" 'creates a string variable
Dim number As Integer = 1000 ' creates a whole number variable
Dim decimal_number1 As Single = 12.45678 ' creates a single precision floating point number
Dim decimal_number2 As Double = 123.456789234565 ' creates a double precision floating point number
Dim correct As Boolean = False ' creates a boolean variable set to false
```

```
ListBox1.Items.Add("Name is a string variable and holds the word: " & name)
ListBox1.Items.Add("Number is an interger variable and holds the whole number: " & number)
ListBox1.Items.Add("Decimal_Number is a single precision variable and holds: " & decimal_number1)
ListBox1.Items.Add("Decimal_Number is a double precision variable and holds: " & decimal_number2)
ListBox1.Items.Add("Correct is a boolean variable and is set to: " & correct)
```



3. Save it as **Higher Example 6**

Information

Boolean variables are new to you.

They are used to check the outcome of conditions that a program sets.

Double is new to you. Using *Single* sets variables which store decimal numbers with to an accuracy of up to 7 digits.

Double sets variables which store decimal numbers with an accuracy of up to 16 digits and is generally used in scientific calculations.

Higher Example 7

Boolean Variable

1. Enter this coding in a new Project called **Higher Example 7** which:

- checks an answer to a question
- sets a boolean variable to TRUE if it is correct.
- uses the boolean variable to determine a score and an output statement.

```
Dim question As String = "What is the capital of Sweden?"
Dim answer As String = "Stockholm"
Dim score As Integer = 0
Dim correct As Boolean = False
```

```
MsgBox(question)
```

```
If answer = InputBox(" Enter your answer please") Then
    correct = True
End If
```

```
If correct = True Then
    score = score + 10
    MsgBox(" Correct, Your score is " & score)
End If
```

```
If correct <> True Then
    MsgBox(" Wrong answer, your score is " & score)
End If
```

2. Test the program. Your program should look like the screen shots below.



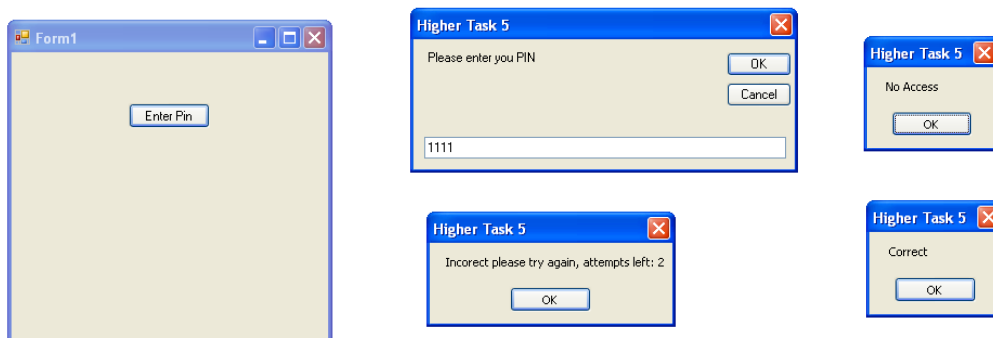
3. Save it as **Higher Example 7**.

Higher Task 5

Write a short program which

- Asks for a PIN number a maximum of 3 times before giving a 'No access ' message.
- uses a Boolean variable to determine the output message

For example: The user inputs the incorrect code “1111” 3 times. The program outputs “Incorrect, Try Again” twice before finally the “No Access” message. If the user inputs the code “1234” the message “Correct” will be displayed.



You Must:

1. Use Boolean.
2. Write the Pseudocode before you begin and show it to your teacher.
3. Test it and Save it as **Higher Task 5**.

Higher Example 8

Concatenation

Concatenation is the process of joining together parts of string variables. It is often used to produce codes and IDs.

1. Create a new project named **Higher Example 8**.

2. Enter the following code:

```
Dim firstname, secondname As String
Dim initial1, initial2, code1, code2 As String

firstname = InputBox("Enter a first name: ")
secondname = InputBox("Enter a second name: ")

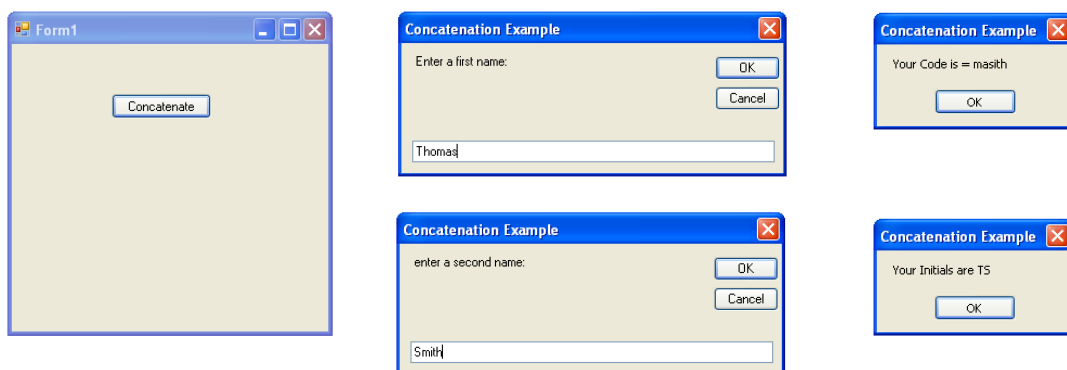
'The next 2 lines gets the initial letters of each name
initial1 = Mid$(firstname, 1, 1) 'takes the first letter of the string
initial2 = Mid$(secondname, 1, 1) 'takes the first letter of the string

'The next line combines the first letters to get initials
MsgBox("Your initials are " & initial1 & " " & initial2)

'The next line works by selecting the last 3 letters of each name entered.
'It works out length of string, subtracts 2 to start at 3 from the end.
code1 = Mid$(firstname, (Len(firstname) - 2), Len(firstname))
'It works out length of string, subtracts 2 to start at 3 from the end.
code2 = Mid$(secondname, (Len(secondname) - 2), Len(secondname))

'The next line combines the 2 parts of the code
MsgBox("Your Code is = " & code1 & " " & code2)
```

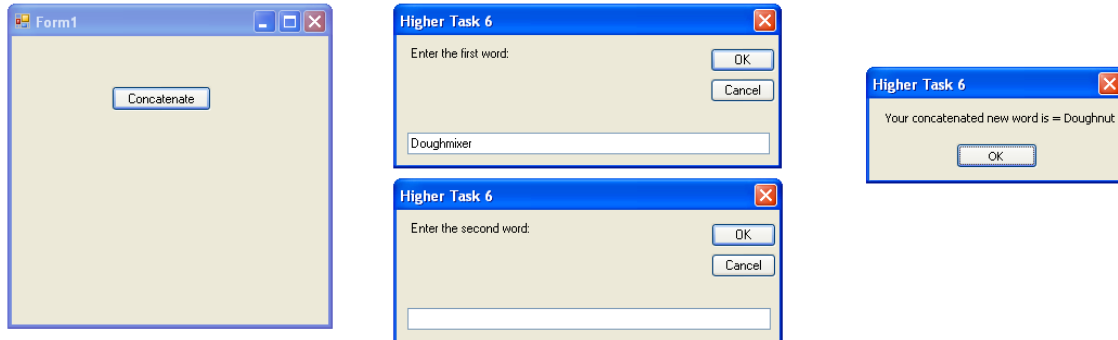
3. Test it.



4. Save it as **Higher Example 8**.

Higher Task 6

Write a program to concatenate the first 5 letters of the word **doughmixer** and the last 3 letters of the word **coconut**.



You Must:

1. Use **Mid\$** and **Len** as you did in Higher Example 8.
2. Write the Pseudocode before you begin and show it to your teacher.
3. Test it and Save it as **Higher Task 6**.

Using Subroutines, Global and Local Variables & Parameters

Scope

The scope of a variable is the extent to which it can be used throughout a program.

A **global variable** can be used anywhere in a program and can be passed in and out of sub routines.



A **local variable** can only be used inside the subroutine or module in which it is declared.



In Visual Studio there are **3 levels** of scope

1. A variable that is declared inside a subroutine using **DIM**: it can only be used in that subroutine. For example:

```
Public Class Form1

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click

        Dim length As Single ' these variables can only be used in this subroutine
        Dim width As Single
        Dim area As Single

        length = InputBox("length please")
        width = InputBox("width please")

        area = length * width
        TextBox1.Text = (" the area = " & area)

    End Sub

End Class
```

2. A variable that is declared at the top level of a form using **DIM** or **Private**: it can be used by any sub-routine in the form, but only in that form. For example:

```
Public Class Form1
    ' these variables are available to any subroutine within this form, their scope
    ' is global but limited to this form.
    Dim length As Single
    Dim width As Single
    Dim area As Single

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Call get_data(length, width)
        Call calculate(length, width, area)
    End Sub

    Private Sub get_data(ByRef length As Single, ByRef width As Single)
        length = InputBox("length please")
        width = InputBox("width please")
    End Sub

    Private Sub calculate(ByVal length As Single, ByVal width As Single, ByVal
area As Single)
        area = length * width
        TextBox1.Text = (" the area = " & area)
    End Sub

End Class
```

3. A global variable declared using the keyword **PUBLIC**: it can be used throughout a program. For Example:

```
Public Class Form1
    Public volume As Single ' this is a truly global variable available
    'throughout a program
    Dim length As Single ' this is a variable local to this form / module
    Dim width As Single
    Dim area As Single
```

Note: As most of your programs only use 1 form you will not be required to use this type of variable.

Higher Example 9

Using Subroutines, Local and Global variables and Parameters

When you are designing complex programs it is best to divide the program into separate sub-routines each with its own job to do.

Before you begin to program you need to:

1. Start a new project and call it **Higher Example 9**.
2. Set up two buttons one with the text ' **Calculate Area**' and the other button with the text '**Calculate Volume**'.
3. Next add two textboxes, one called **areabox** (to display the area) and the other called **volumebox** (to display the volume).
4. Give each Textbox a suitable label (Area in Square Centimetres and Volume in Cubic Centimetres).

Your form should look like this:

In this example we are going to use different subroutines to

- input the length, breadth and height of a cube
- calculate the area of a rectangle
- calculate the volume of the box

Your program will have 3 subroutines

get_data
calculate_area
calculate_volume

In addition it will use 2 other subroutines called **Button1_click** and **Button2_click** which are attached to the buttons which call the subroutines **get_data**, **calculate_area** and **calculate_volume** into action.

Global variables

Step 1

The first thing is to declare global variables. These are variables that can be called upon by any subroutine attached to this form. They are set out at the start of the program right after the statement **Public Class Form1**.

This is done by Entering this coding:

```
Public Class Form1
    Dim length, width, area, height, volume As Single
```

They are distinct from local variables which are declared inside a subroutine and can only be used by that sub-routine

Parameters

You need to know about parameters at this point. Parameter is the term we use to describe variables that we pass into and out of sub-routines. e.g.

```
'These are the parameters being passed into the sub-routine
get_data.
Call get_data(length, width, height)
```

Step 2

You need to set out the sub routine which is attached to the calculate area button. This calls the first two subroutines into action

This is done by Double clicking on the *calculate area button* and Entering this coding:

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
'note the system sets its own parameters when you use a Call

Call get_data(length, Width, Height) 'These are the parameters being passed.
Call calculate_area(length, Width, area)

End Sub
```

Passing Parameters into a sub-routine

There are 2 types of parameters:

- Those that are passed into a sub-routine to be changed and passed out again: **called passing by reference (ByRef)**
- Those that are going into a sub-routine and will not be passed back out: called **passing by value (ByVal)**

When you set up a sub-routine you need to indicate which parameters are (**ByRef**) and which are (**ByVal**).

Step 3

Enter the code for the sub-routine called get-data. Note that you have to tell it which parameters it needs and whether they are *passed by reference* (**ByRef**) or *passed by value* (**ByVal**).

This is done by Entering this coding:

```
Private Sub get_data(ByRef length As Single, ByRef width As Single, ByRef height
As Single)
    length = InputBox("length please")
    width = InputBox("width please")
    height = InputBox("height please")
End Sub
```

This sub routine reads in the length, width and height of a box and then passes them on to other sub routines **by reference (ByRef)**.

Step 4

Now you need to enter the code for the **calculate_area** sub-routine.

Again, note that you have to tell it which parameters it needs and whether they are *passed by value* (**ByRef**) or *passed by value* (**ByVal**).

This is done by Entering this coding:

```
Private Sub calculate_area(ByVal length As Single, ByVal width As Single, ByVal area As Single)
    area = length * Width
    areabox.Text = (" the area = " & area)
End Sub
```

Step 5

Now enter the coding which is attached to the *calculate_volume button* and which calls up the calculate volume sub-routine.

This is done by Double clicking on the *calculate volume button* and Entering this coding:

```
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
    'note the system sets its own parameters when you use a Call
    Call calculate_volume(length, width, height, volume)
End Sub
```

Step 6

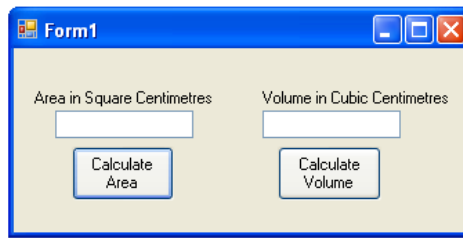
Now you need to enter the code for the **calculate_volume** sub-routine.

Again, note that you have to tell it which parameters it needs and whether they are *passed by value* (**ByRef**) or *passed by value* (**ByVal**).

This is done by Entering this coding:

```
Private Sub calculate_volume(ByVal length As Single, ByVal width As Single, ByVal height As Single, ByVal volume As Single)
    volume = length * width * height
    volumebox.Text = volume
End Sub
```

Your program should now look like this:



The Coding for the program will be:

```
Public Class Form1
Dim length, width, area, height, volume As Single

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
    Call get_data(length, width, height) 'note the system sets its own
    'parameters when you use a Call
    Call calculate_area(length, width, area)
End Sub

Private Sub get_data(ByRef length As Single, ByRef width As Single, ByRef
height As Single)
    length = InputBox("Length please")
    width = InputBox("Width please")
    height = InputBox("Height please")
End Sub

Private Sub calculate_area(ByRef length As Single, ByRef width As Single,
ByRef area As Single)
    area = length * width
    areabox.Text = ("The Area = " & area)
End Sub

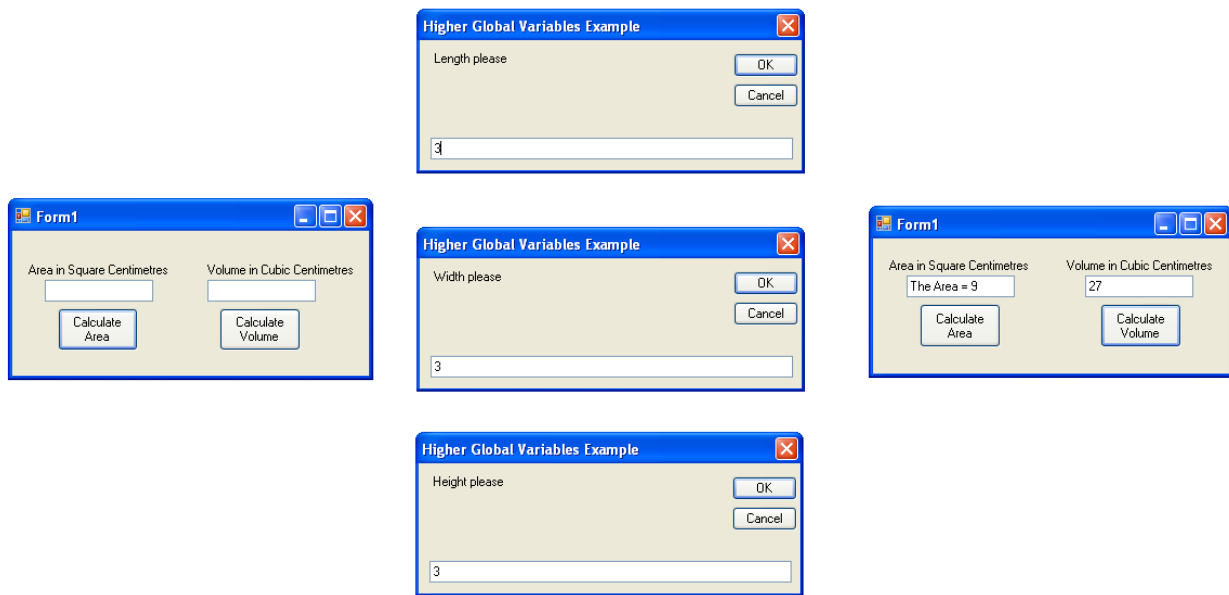
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
    Call calculate_volume(length, width, height, volume)
End Sub

Private Sub calculate_volume(ByRef length As Single, ByRef width As Single,
ByRef height As Single, ByRef volume As Single)
    volume = length * width * height
    volumebox.Text = volume
End Sub

End Class
```

Now test the program.

You should see the following results:

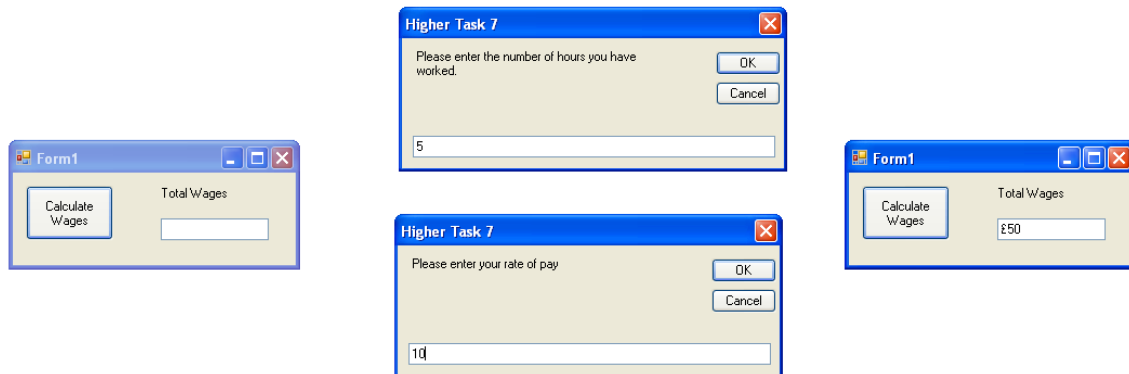


Save it as Higher Example 9.

Higher Task 7

Write a program which uses sub-routines and parameters to

- read in the *hours worked*, the *rate of pay*.
- calculate the total wages
- display the total wages



You Must:

1. Use **Global Variables, sub-routines and parameters** as you did in Higher Example 9.
2. Write the Pseudocode before you begin and show it to your teacher.
3. Test it and Save it as **Higher Task 7**.

Higher Example 10

Passing Arrays as Parameters

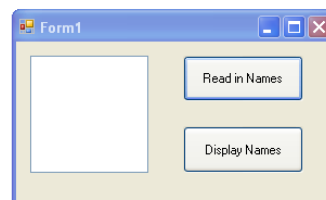
When you start to use the standard algorithms you will need to pass arrays into and out of sub-routines as parameters.

This example:

- Calls up two sub-routines using two buttons.
- The 1st subroutine uses an array to read in and store 5 names.
- It then passes that array to a 2nd subroutine as a parameter by reference.
- The 2nd sub-routine then displays the names in a listbox.

1. Create a new project named **Higher Example 10**.

2. Set up a form with two buttons like this:



3. Enter this code:

```
Public Class Form1

    Dim index As Integer
    Dim names(4) As String

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Call getname(names)
    End Sub

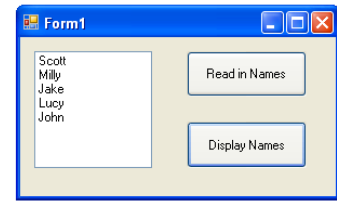
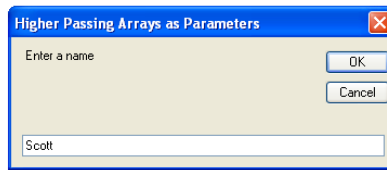
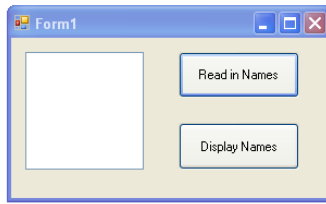
    Private Sub getname(ByRef names() As String)
        For index = 0 To 4
            names(index) = InputBox("Enter a name")
        Next
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
        Call display(names)
    End Sub

    Private Sub display(ByRef names() As String)
        For index = 0 To 4
            ListBox1.Items.Add(names(index))
        Next
    End Sub

End Class
```

4. Now Test your program.

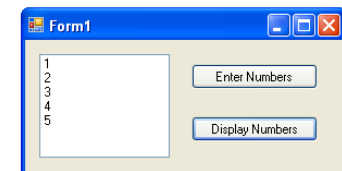
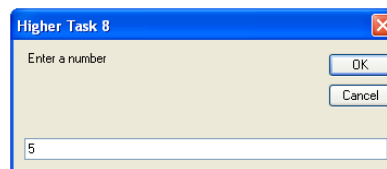
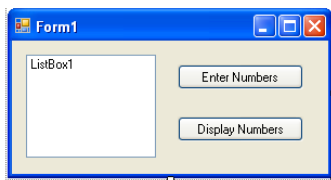


5. Save it as **Higher Example 10**.

Higher Task 8

Write a program that uses:

- One button with a subroutine and parameters to store 5 numbers in an array.
- One button to pass the array through another subroutine to display the numbers.



You Must:

1. Fill and then Pass an array in to a 2nd subroutine as a parameter by reference as you did in **Higher Example 10**.
2. Write the Pseudocode before you begin and show it to your teacher.
3. Test it and Save it as **Higher Task 8**.

Higher Example 11

Standard Algorithms

Standard Algorithms are used to carry out a range of common tasks in programs.

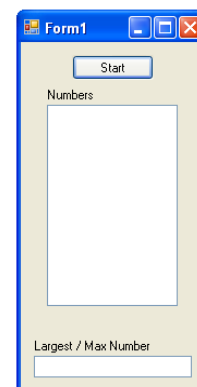
The algorithms you will learn to use are:

- **MAX**: used to find the largest number in a list
- **MIN**: used to find the smallest number in a list
- **Linear Search**: used to find specific items in a list
- **Count Occurrence**: used to find how many times an item appears in a list

MAX Standard Algorithm

1. initialise
 - 1.1 dimension array to 30
 - 1.2 dimension max as integer
2. fill array
 - 2.1 loop index 0 to size of list
 - 2.2 fill array with random numbers
 - 2.3 display array
3. find max
 - 3.1 set max to first item in the array
 - 3.2 loop index 1 to size of list
 - 3.3 if list index > max
 - 3.4 set max to list index
4. display max
 - 4.1 display result

1. Create a new project named **Higher Example 11**.
2. Create the form layout on the right.



3. Now enter this code:

```
Public Class Form1

    Dim index As Integer
    Dim max As Integer
    Dim numbers(30) As Integer

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        Handles Button1.Click
            Call fillarray(numbers, index)
            Call find_max(numbers, max, index)
            Call result(max)
        End Sub

    Private Sub fillarray(ByRef numbers() As Integer, ByVal index As Integer)

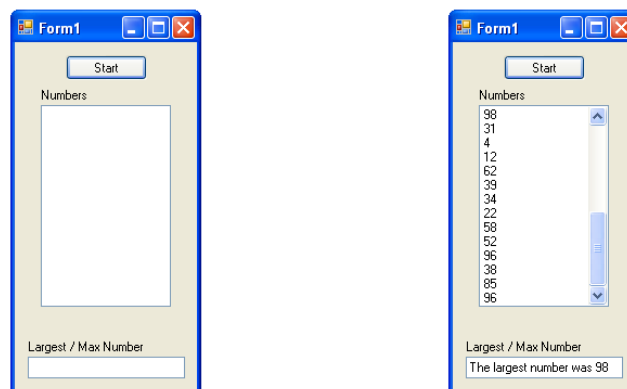
        For index = 0 To 29
            Randomize()
            numbers(index) = Rnd() * 100
            resultbox.Items.Add(numbers(index))
        Next
    End Sub

    Private Sub find_max(ByRef numbers() As Integer, ByRef max As Integer, ByRef index As Integer)
        For index = 0 To 29
            If numbers(index) > max Then
                max = numbers(index)
            End If
        Next
    End Sub

    Private Sub result(ByRef max As Integer)
        TextBox1.Text = ("The largest number was " & max)
    End Sub

End Class
```

4. Test the program



5. Save it as **Higher Example 11**.

Higher Task 9

Write a program:

- To enter and store 10 scores out of 100
- Find and display the top score

Form1

Enter Scores

Top Score

Higher Task 9

Enter score

OK

Cancel

100

Form1

Enter Scores

10
20
30
40
50
60
70
80
90
100

Top Score

Top Score = 100

You Must:

1. Use **MAX** to find the largest number in a list. See **Higher Example 11**.
2. Write the Pseudocode before you begin and show it to your teacher.
3. Test it and Save it as **Higher Task 9**.

Higher Example 12

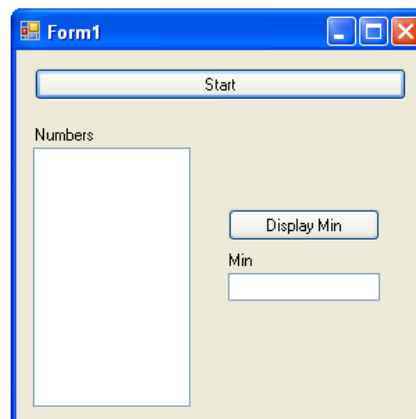
MIN

MIN Standard Algorithm: used to find the smallest number in a list

1. initialise
 - 1.1 dimension array to 30
 - 1.2 dimension min as integer
 - 1.3 dimension index as integer
2. fillarray
 - 2.1 loop index 0 to size of list
 - 2.2 fill array with random numbers
 - 2.3 display array
3. find max
 - 3.1 set min to first item in array
 - 3.2 loop index 0 to size of list
 - 3.3 if list index < min
 - 3.4 set min to list index
4. display max
 - 4.1 display result

1. Create a new project named **Higher Example 12**.

2. Create a form similar to this:



The screenshot shows a Windows form titled "Form1" with a standard Windows XP-style title bar (blue with minimize, maximize, and close buttons). The form has a light beige background. At the top, there is a horizontal button labeled "Start". Below this, on the left side, is a label "Numbers" above a large, empty rectangular text area. On the right side, there is a button labeled "Display Min". Below this button is a label "Min" followed by a small, empty rectangular text box.

3. Now enter this code:

```
Public Class Form1

    Dim index As Integer
    Dim min As Integer
    Dim numbers(30) As Integer

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Call fillarray(numbers)
        Call find_min(numbers, min)
    End Sub

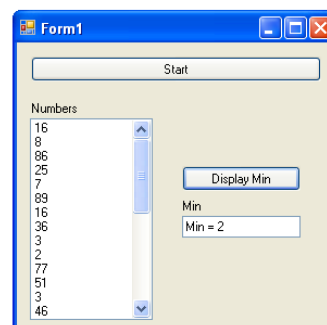
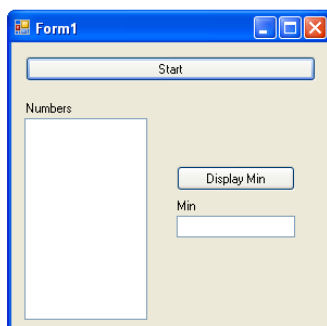
    Private Sub fillarray(ByRef numbers() As Integer)
        For index = 0 To 29
            Randomize()
            numbers(index) = Rnd() * 100
            ListBox1.Items.Add(numbers(index))
        Next
    End Sub

    Private Sub find_min(ByVal numbers() As Integer, ByRef min As Integer)
        min = numbers(0)
        For index = 0 To 29
            If numbers(index) < min Then
                min = numbers(index)
            End If
        Next
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
        TextBox1.Text = ("Min = " & min)
    End Sub

End Class
```

4. Test the program.

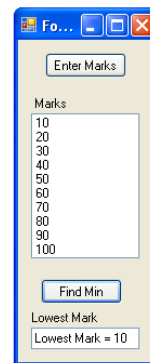
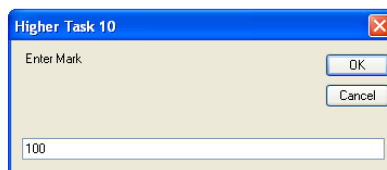


5. Save it as **Higher Example 12**.

Higher Task 10

Write a program:

- To enter and store 10 Marks out of 100
- Find and display the bottom score



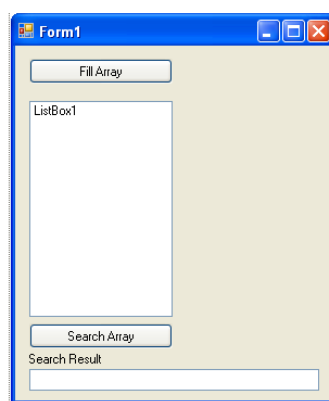
You Must:

1. Use **MIN** to find the smallest number in an array. See **Higher Example 12**.
2. Write the Pseudocode before you begin and show it to your teacher.
3. Test it and Save it as **Higher Task 10**.

Higher Example 13

Linear Search: used to find specific items in a list

1. Create a new Project named **Higher Example 13**.
2. Create a form like this:



3. Enter the following code which uses a target number to search an array.

```
Public Class Form1

    Dim position As Integer
    Dim numbers(29) As Integer
    Dim found As Boolean
    Dim target_number As Integer

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Call fillarray(numbers)
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
        Call search_array(numbers, position, found, target_number)
        Call search_result(position, found, target_number)
    End Sub

    Private Sub fillarray(ByRef numbers() As Integer)
        Dim index As Integer
        For index = 0 To 29
            Randomize()
            numbers(index) = Rnd() * 100
            ListBox1.Items.Add(numbers(index))
        Next
    End Sub

    Private Sub search_array(ByRef numbers() As Integer, ByRef position As
Integer, ByRef found As Boolean, ByRef target_number As Integer)
        Dim index As Integer
        found = False
        target_number = InputBox("Please enter the number you are looking for ")
        For index = 0 To 29
            If numbers(index) = target_number Then
                found = True
                position = index + 1
            End If
        Next
    End Sub

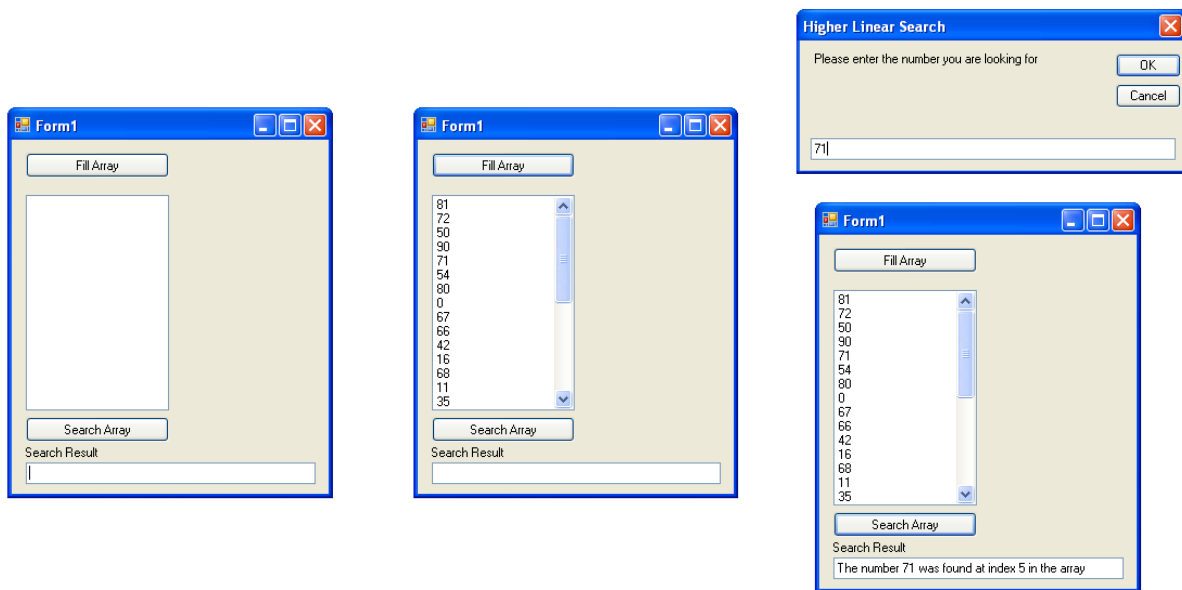
    Private Sub search_result(ByRef position As Integer, ByRef found As Boolean,
ByRef target_number As Integer)
        If found = True Then
            TextBox1.Text = ("The number " & target_number & " was found at
index " & position & " in the array")
        End If

        If found = False Then
            TextBox1.Text = ("The number " & target_number & " was not found in
the array")
        End If

    End Sub

End Class
```

4. Test your program.

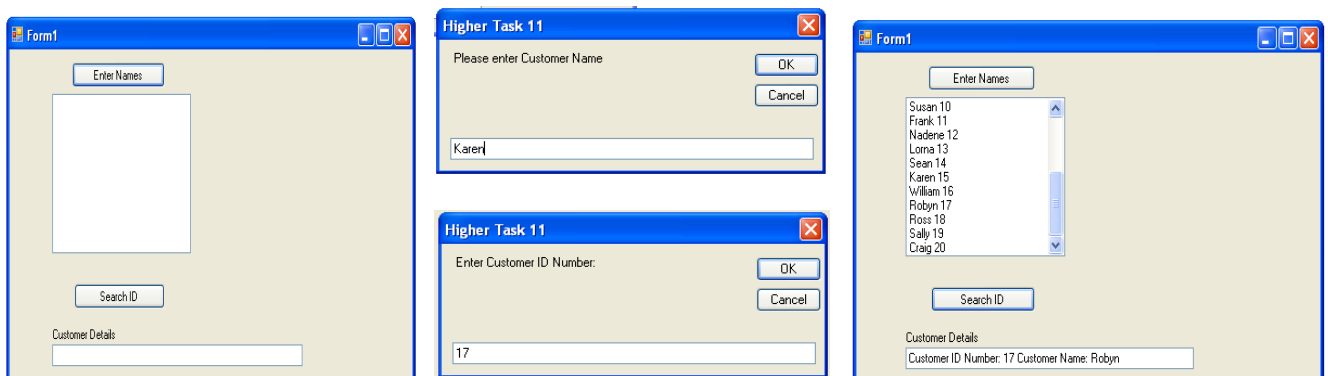


5. Save your program as **Higher Example 13**.

Higher Task 11

Write a program to:

- enter and store the names and ID numbers of 20 customer numbers.
- each customer should be given an id number between 1 & 20.
- allow the user to search for a customer using their ID number
- display the customer's name and ID number



You Must:

1. Use **Linear Search** to find specific items in a list. See **Higher Example 13**.
2. Write the Pseudocode before you begin and show it to your teacher.
3. Test it and Save it as **Higher Task 11**.

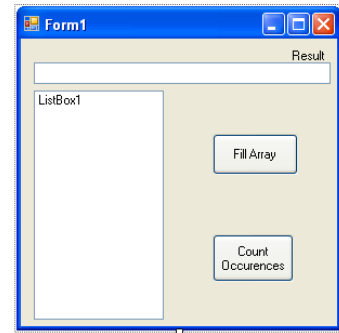
Higher Example 14

Count Occurrence: used to find how many times an item appears in a list.

Algorithm

1. initialise
 2. fillarray
 3. count array
 4. display result
-
- 1.1 Dimension index, target number, counter and numbers array As Integer
 - 1.2 Dimension found as Boolean
-
- 2.1 loop index 0 to size of list
 - 2.2 fill array with random numbers
 - 2.3 display array
-
- 3.1 set found to False
 - 3.2 input target_number
 - 3.3 loop 0 To 29
 - 3.4 If numbers(index) = target_number Then
 - 3.5 set found to True
 - 3.6 add 1 to counter
-
- 4.1 if found = true then state target number found and display counter
 - 4.2 if found = false then state target number not found

1. Create a new project named **Higher Example 14**.
2. Create a form like the form on the right:
3. Enter this code:



```
Public Class Form1

    Dim index As Integer
    Dim numbers(30) As Integer
    Dim found As Boolean
    Dim target_number As Integer
    Dim counter As Integer

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Call fillarray(numbers)
    End Sub

    Private Sub fillarray(ByRef numbers() As Integer)
        For index = 0 To 29
            numbers(index) = Rnd() * 10
            ListBox1.Items.Add(numbers(index))
        Next
    End Sub

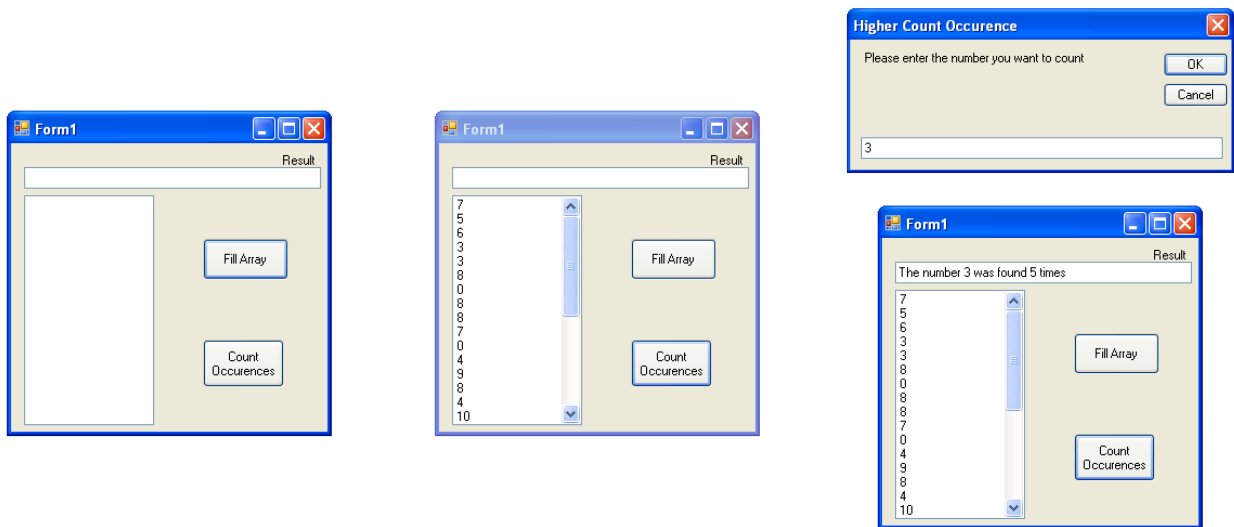
    Private Sub count_array(ByRef numbers() As Integer, ByRef found As Boolean,
ByRef target_number As Integer, ByRef counter As Integer)
        found = False
        target_number = InputBox("Please enter the number you want to count")
        For index = 0 To 29
            If numbers(index) = target_number Then
                found = True
                counter = counter + 1
            End If
        Next
    End Sub

    Private Sub result(ByRef found As Boolean, ByRef target_number As Integer,
ByRef counter As Integer)
        If found = True Then TextBox1.Text = "The number " & target_number & "
was found " & counter & " times"

        If found = False Then TextBox1.Text = "The number " & target_number & "
was not found in the array"
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
        Call count_array(numbers, found, target_number, counter)
        Call result(found, target_number, counter)
    End Sub
End Class
```

4. Test the program

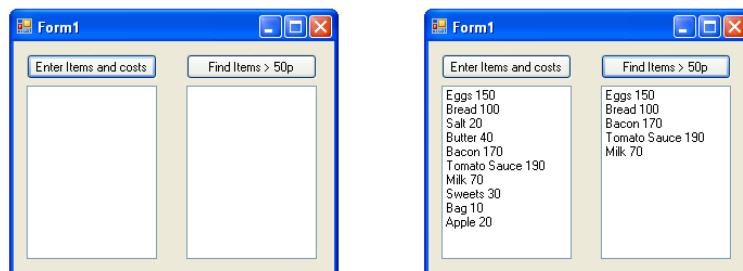


5. Save it as **Higher Example 14**.

Higher Task 12

Write a program to

- enter and store the names and cost of 10 items sold in a low price store that sells everything at a maximum price of 200 pence .
- find and display how many items are priced above 50 pence
- display the names of all items priced above 50 pence



You Must:

1. Use **Count Occurance** to find and count the specific items in a list. See **Higher Example 14**.
2. Write the Pseudocode before you begin and show it to your teacher.
3. Test it and Save it as **Higher Task 12**.



**You have now finished
the Higher Computing
Programming Practical
course.**

