ARRAY

I. Single-Dimension Arrays

This is a consecutive group of memory locations (called elements) that are referenced by the same name and have the same data type (list of values).To refer to a particular location or element in the array, you need to specify the array name and the array element position number. Arrays are reference variables

- Arrays are based on the **System.Array** object – a type of **collection** object. Arrays can store almost any type of data such as integer, string, decimal, etc.
- Each **element** is numbered beginning with the number **zero**. Example, **0**, **1**, **2**, **3**, etc.
- The number referring to an array element is placed inside parentheses and is called a **subscript** or **index**. This is the general format for an array reference.

```
ArrayName(ElementNumber)
```

Example

The following is a reference to the fourth element in an array named AmountDecimal

```
AmountDecimal(3)
```

Characteristics of Arrays

- An array stores multiple data items
  - The elements of an array must have the same data type – called the base type
  - An array can store a list of primary data types or other data types e.g. an array of integers or text boxes
- An array has one or more dimensions
  - A one-dimensional array can be thought of as a list
  - A two-dimensional array can be thought of as a grid or table (or a "rectangular array")
  - A Visual Basic array can have between 1 and 60 dimensions

Declaring an Array

Arrays are created in memory by declaring them:

- Generally you declare local arrays with the keyword **Dim** and module-level arrays with the keyword **Private**.
- The number inside parentheses that specifies the number of elements should be an **integer**.
- Numeric array elements default to a value of zero; string array elements default to a value of the empty string.

Syntax

[Public | Friend | Private | Dim] ***arrayName*** **([*upperbound*]) As** ***dataType*** **=** ***initExpr***

- The access modifier (**Public**, **Private**, **Friend**) defines the array's visibility
- The identifier (name) is defined by *arrayName*
- The optional *upperbound* contains the value of the largest subscript (UpperSubscript)
  - Omit the *upperbound* to declare a dynamic uninitialized array
- *dataType* defines the data type of the array
- *initExpr* is used to assign initial values to the array's elements

The following are several different correct syntax variations for declaring arrays.

```
Dim ArrayName(UpperSubscript) As DataType
Dim ArrayName() As DataType = {ListOfValues}
Dim ArrayName As DataType() = {ListOfValues}
```

Examples:

```
'A String array with 26 elements numbered 0 through 25
Dim NameString(25) As String
'A decimal array of 11 elements numbered 0 through 10
```

```
Private BalanceDecimal(10) As Decimal
```

When storing values immediately to the array, you **cannot** specify the number of elements within the parentheses—VB will determine the number of elements automatically.

```
'A private string array (module level) with 6 elements numbered 0 through 5
Private SchoolString() As String = {"Arts and Sciences", "Business", _
    "Nursing", "Engineering", "Education", "Pharmacy"}
```

Element Number	Array Elements
0	*Arts and Sciences*
1	*Business*
2	*Nursing*
3	*Engineering*
4	*Education*
5	*Pharmacy*

Array Class (Members)

- The **Length** property gets the number of elements in the array.
- The **Rank** property gets the number of dimensions. A one-dimensional array has a **Rank** of 1
- The **GetLength** method gets the number of array elements in a particular dimension
- The **GetUpperBound** and **GetLowerBound** methods get the largest and smallest subscript for a dimension. The array dimension is passed as an argument
- The **Sort** method sorts an array or part of an array
- The **Max** method gets the maximum element in an array
- The **Min** method gets the minimum element in an array
- The **Sum** method gets the sum of values pf the elements in an array
- The **Average** method gets the average value of elements in an array

For Each..Next Loops

Arrays are often processed in loops. The power of arrays comes with using a variable as the subscript.

- A **For..Next** loop can process an entire array with a few lines of code as shown below, but the code requires you to manipulate the array index.

```
For RowInteger = 0 To NumberOfSchoolsInteger
    Debug.WriteLine(SchoolString(RowInteger))
Next RowInteger
```

- Another type of count-controlled loop is the **For Each…Next** loop – the advantage of this coding construct is that you don't have to manipulate the loop index. The following looping construct is equivalent to the one shown above.

```
For Each SchoolNameString As String In SchoolString
    Debug.WriteLine(SchoolNameString )
Next SchoolNameString
```

When using For **Each..Next** loops, the **index data type** must match the array data type, i.e., for a string array the index should be a string index, for an integer array the index should be an integer, etc.

Example 1

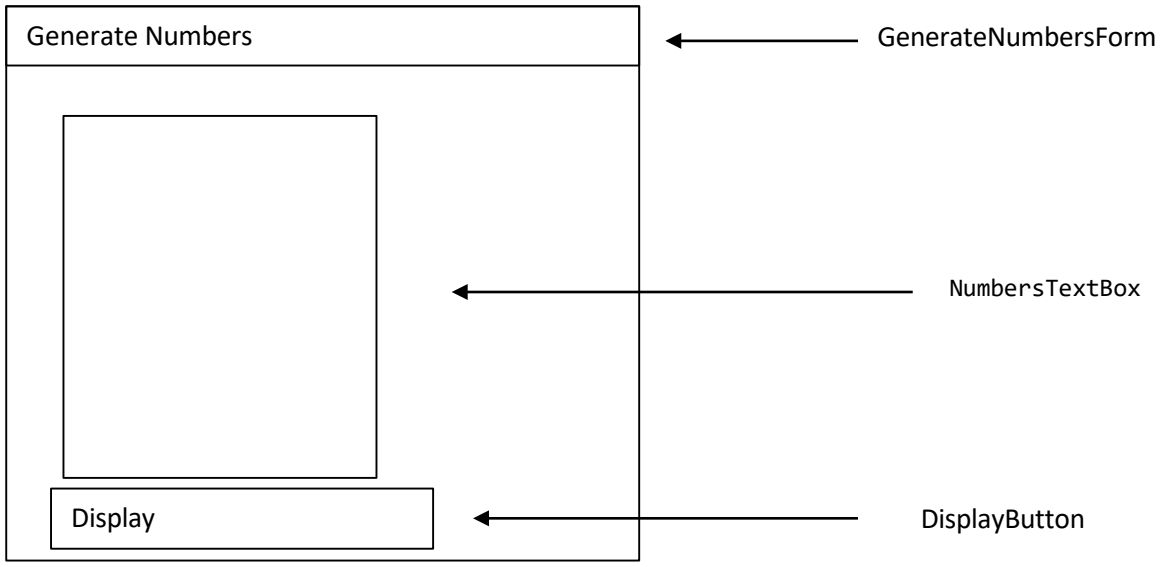
Write a Visual Basic program that initializes the elements of a six-element array to the integers 4, 5, 6, 3, 2, 1 and then displays the array elements in an output window sorted in ascending order. Your program should also display the minimum, maximum, sum and average of the numbers using message box dialogs.

```
Public Class ArraysForm
```

```
    Private Sub ArraysButton_Click(sender As Object, e As EventArgs) Handles
    ArraysButton.Click
        Dim Numbers() As Integer = {4, 5, 6, 3, 2, 1}
        Array.Sort(Numbers)
        For Each Num As Integer In Numbers
            Debug.WriteLine(Num)
        Next Num
        MsgBox("Minimum number is : " & Numbers.Min())
        MsgBox("Maximum number is : " & Numbers.Max())
        MsgBox("Sum is : " & Numbers.Sum())
        MsgBox("Average is : " & Numbers.Average())
    End Sub
End Class
```

Example 2

Write a Visual Basic program that initializes the elements of a ten-element array to even integer numbers from 2 to 20 and then displays the indices and array elements contents using an appropriate format in a text box control after a click of the button control. Design your GUI.




```
Public class GenerateNumbersForm
    Private Sub DisplayButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles DisplayButton.Click
        Dim numbers(0 To 9), i As Integer
        For i = 0 To numbers.GetUpperBound(0)
            numbers(i) = 2 + 2 * i
        Next i
        NumbersTextBox.AppendText("Index " & ControlChars.Tab & "Numbers" & ControlChars.NewLine)
        For i = 0 To numbers.GetUpperBound(0)
            NumbersTextBox.AppendText(i & ControlChars.Tab & numbers(i).ToString & _
           ControlChars.NewLine)
        Next i
    End Sub
End Class
```

II. Multidimensional Arrays

A multidimensional array has more than one dimension – we will work with arrays that have both rows and columns to produce a **two-dimensional** table.

For a two-dimensional (rectangular) array, information is stored in rows and columns.

– Requires two indexes:

- The first, by convention, represents the row

- The second represents the column.
- Cannot use the same index for both.

An example of an application that uses a two-dimensional table is the use of tax tables for computing personal income taxes to be paid.

TaxTableDecimal

Income	Single	Married	Head of Household
	0	1	2	3
0	8,000	554	329	308
1	14,000	966	875	852
2	21,000	1,850	1,411	1,389
3	30,000	3,084	2,424	2,407

(more rows would be in a full table representing all income ranges)

This declaration statement allocates memory for **TaxTableDecimal** as a module-level array assuming there are 150 different income level brackets:

```
Private TaxTableDecimal(149,3) As Decimal
```

- The **rows** represent the income levels.
- The **columns** represent tax classifications such as single, married, and head of household.
- When referencing the **TaxTableDecimal**, the first subscript represents the **row**. The second subscript represents the **column**. for example, to display the value of row 20, column 2 to a label, the code is:

```
TaxAmountTextBox.Text = TaxTableDecimal(19,1).ToString("C")
```

One limitation of the two-dimensional table is that you can only store one type of data in the array – you can work around this to a certain extent by storing data as **string** that would otherwise be **numeric** and using Convert methods to convert data from string to Decimal or Integer or another numeric data type to support processing.

Example 1:

This figure shows a table named StatesString that stores the names of states of the United States as string data. States are classified in columns according to their land mass size as **Large**, **Middle**, and **Small** sized.

StatesString

	0	1	2
0	Alaska	Indiana	R. Island
1	Texas	Maryland	Delaware
2	California	Idaho	Vermont
3	Florida	Iowa	Hawaii
	Large	Middle	Small

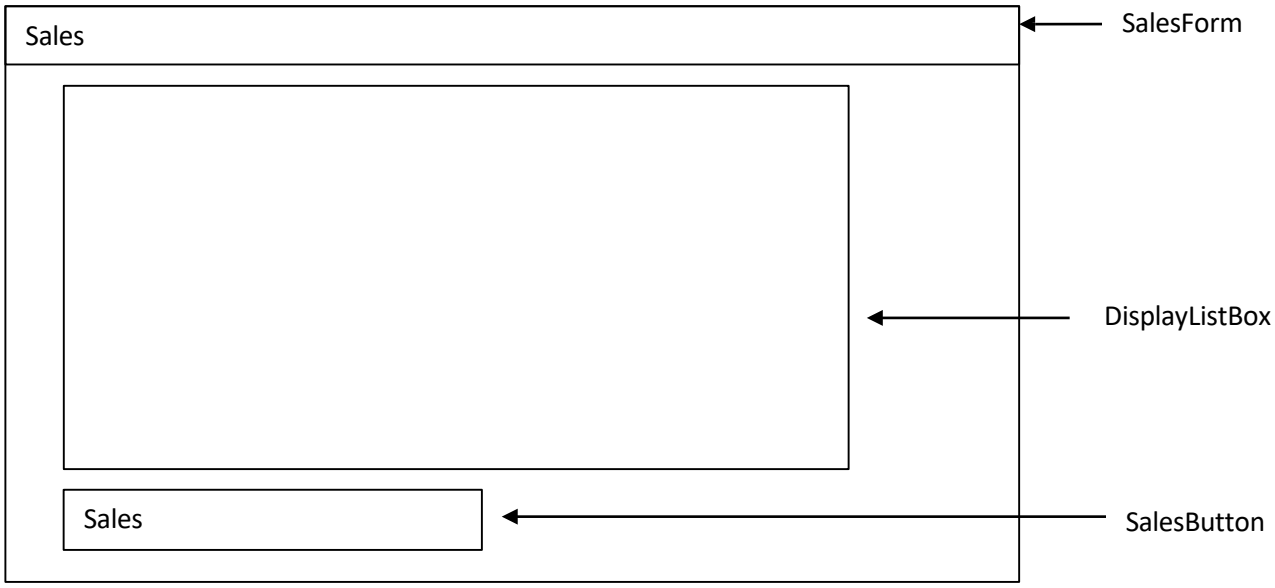
This sub procedure illustrates printing the **StatesString** table to the immediate output window

```
Private Sub StatesButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles statesButton.Click
    Dim StatesString(,) As String = { _
        {"Alaska", "Indiana", "R. Island"}, _
        {"Texas", "Maryland", "Delaware"}, _
        {"California", "Idaho", "Vermont"}, _
        {"Florida", "Iowa", "Hawaii"}}
    Dim RowInteger As Integer = 0
    Dim ColumnInteger As Integer = 0
    'Print to the Immediate Window as an output location
      For RowInteger = 0 To StatesString.GetUpperBound(0)
            For ColumnInteger = 0 To StatesString.GetUpperBound(1)
```

```
                    Debug.Write(StatesString(RowInteger,
                          ColumnInteger) & ControlChars.Tab)
                Next
                Debug.WriteLine("")
            Next
        End Sub
```

Example 2

A company distributes thirty different items around Nairobi through its twenty salesmen. Using arrays write a program to input a salesman name and the corresponding sales made by each of the salesman for each of the item. The program should output each of the salesman's name, sales and the total sales for each salesman in the company as well as the grand total in a list box control. All input should be done by use of input box dialogs. Sketch your GUI




```
Public Class SalesForm
    Private Sub SalesButton_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles SalesButton.Click
     Dim Sales(2, 3) As String
      Dim Total As Integer, GrandTotal As Integer
      Dim DisplaySales As String
      Dim I As Integer, J As Integer
      'Input sales values
      For I = Sales.GetLowerBound(0) To Sales.GetUpperBound(0)
          Sales(I, 0) = InputBox("Enter the salesman's name")
          For J = Sales.GetLowerBound(1) + 1 To Sales.GetUpperBound(1)
              Sales(I, J) = InputBox("Enter the sales for item:" & J)
          Next J
      Next I
      'Compute and display sales details
      For I = Sales.GetLowerBound(0) To Sales.GetUpperBound(0)
          SalesListBox.Items.Add(Sales(I, 0))
          Total = 0
          DisplaySales = String.Empty
          For J = Sales.GetLowerBound(1) + 1 To Sales.GetUpperBound(1)
              DisplaySales &= Sales(I, J) & ControlChars.Tab
              Total = Total + Convert.ToInt32 (Sales(I, J))
          Next J
          SalesListBox.Items.Add(DisplaySales)
          SalesListBox.Items.Add("Total Sales = " & Total.ToString)
          GrandTotal = GrandTotal + Total
      Next I
      SalesListBox.Items.Add("The Grand Total is:" & GrandTotal.ToString)
  End Sub
End Class
```

Exercises

1. A meteorologist friend of yours has to take three readings of air humidity a day (morning, midday and evening), Monday to Friday. Write a program that allows the user to enter the day of the week and the readings. It should then calculate and display the day and the day's average for each of the days of the week as well as the week's average using a suitable format in a list box control. Sketch your GUI

2. Write a Visual Basic program that reads the student name and marks scored in five subjects by ten students in a class and stores them in to two arrays. Your program should then compute the average mark for each student, and then it displays the student name, marks and the average mark for each student in a list box control in an appropriate format. Design an appropriate user interface