Learning VB.Net

Tutorial 09 – Arrays

Hello everyone… welcome to vb.net tutorials. These are going to be very basic tutorials about using the language to create simple applications, hope you enjoy it. If you have any notes about it, please send them to **notes@mka-soft.com** I will be happy to answer them. Finally if you find these tutorials are useful, it would be nice from you to send a small donation via PayPal to **donation@mka-soft.com**.

Tutorial posted on 2009-December-17.

Arrays

In many cases you need to perform processing on huge amount of data. For example you may want to find an average of 7000 values in a file, or reading unknown number of people names into your application and recalling them back. In such case it is inconvenient to define all these variables separately. Instead you define an array which holds all the records.

Think of an array as a big variable that you can get and store values in it by specifying position. For example, below is an array of integer, let us call it **A**:

Element 0 in A	33
Element 1 in A	123
Element 2 in A	3
Element 3 in A	-5
Element 4 in A	19
Element 5 in A	77
Element 6 in A	1212
Element 7 in A	0

The cells in green are just shown here to show you the position of the variable, while the cells in purple show you the content of each variable. As you can see here this array can store 8 variables (starting from 0 and ending at 7). The first location is always 0, some people might find it confusing. You can ignore the first location and avoid any confusion, but you need to know where does the indexing of the array start.

In VB.Net to create an array like the above, you write the following:

```
Dim A(7) As Integer
```

This statement generates an array that can store integers. The array has 8 elements (starting with index 0, and ending with 7). When the statement above is executed, you get the following result in memory:

Element 0 in A	0
Element 1 in A	0
Element 2 in A	0
Element 3 in A	0
Element 4 in A	0
Element 5 in A	0
Element 6 in A	0
Element 7 in A	0

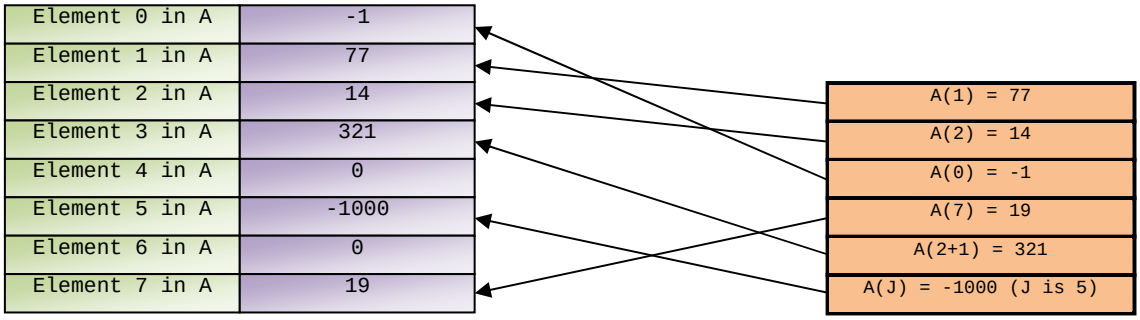
To fill the array, you use something like below:

```
' fill element in locations 0, 1, 2, and 7 (no need to fill in order)
        A(1) = 77
        A(2) = 14
        A(0) = -1
        A(7) = 19
```

```
' fill element in location 3 (can use expression to find the location)
        A(2 + 1) = 321

' fill element in location 5 (can use variable to find the location)
        Dim J As Integer
        J = 5
        A(J) = -1000
```

In order to understand how this works, check out the graph below:



Now, let us fill the array with numbers from 0 to 7:

```
        ' define array first
        Dim MyNumbers(7) As Double

        ' define a counter
        Dim I As Integer

        ' fill the array
        For I = 0 To 7
            MyNumbers(I) = I
        Next
```

If we want to fill the values from the keyboard (let us say, we want the user to enter 10 names):

```
        ' define the array
        Dim My10Names(9) As String

        ' define a counter
        Dim C As Integer

        ' fill the array
        For C = 0 To 9
            My10Names(C) = InputBox("Enter the number number " & C)
        Next
```

To get the values out of the array, you use the same format used above. For example the following code shows the content of an array in a list box:

```
        ' display the result
        ListBox1.Items.Clear()
        For I = 0 To 7
            ListBox1.Items.Add(My10Names(I))
```

```
        Next
```

To test what we have learned, you can find a simple application on the web site that shows you how to use arrays, by filling them, finding max and min values, and finding average. You need to add a DataGridView, and 4 CommandButtons to your form. The code is:

```
Public Class Form1

    ' we define two arrays one to store names, another to store marks
    Dim Names(9) As String
    Dim Marks(9) As Integer

    ' we define a variable to store how many element of the array we used
    Dim StCount As Integer = 0

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

        ' read the name and mark and put them in the next empty slot
        Names(StCount) = InputBox("Enter the name of student")
        Marks(StCount) = InputBox("Enter the mark")

        ' the new name and mark should be displayed on the data grid
        DataGridView1.Rows.Add(Names(StCount), Marks(StCount))

        ' move the counter to the next empty slot
        StCount = StCount + 1
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button2.Click
        ' find the maximum mark
        Dim I As Integer            ' used for counting
        Dim MaxPos As Integer       ' used to remember the index of maximum mark

        MaxPos = 0                  ' assume first mark is the maximum

        For I = 1 To StCount - 1                ' loop over all other slots
            If Marks(I) > Marks(MaxPos) Then    ' is there an element with a mark greater
than the current maximum?
                MaxPos = I                      ' we found a new max, update our maximum
            End If
        Next

        MsgBox("student " & Names(MaxPos) & " has the maximum mark")
    End Sub

    Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button3.Click
        ' find minimum
        ' it is identical to the previous, except for the condition
        Dim I As Integer
        Dim MinPos As Integer

        MinPos = 0
        For I = 1 To StCount - 1
            If Marks(I) < Marks(MinPos) Then
                MinPos = I
            End If
        Next

        MsgBox("student " & Names(MinPos) & " has the minimum mark")

    End Sub
```

```
    Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button4.Click
        Dim I As Integer            ' I is counter
        Dim AVG As Double           ' Used to store the sum and finding the average
        AVG = 0                     ' The avg is zero
        For I = 0 To StCount - 1    ' Loop over all elements in the array
            AVG += Marks(I)         ' Add each element to the some of the previous ones
        Next
        AVG = AVG / StCount         ' divide the total by number of elements to get the
average
        MsgBox("the average is:" & AVG)
    End Sub

End Class
```

Next tutorial we will start working with collections, and how they are simpler than arrays. If you have any notes or questions send them to notes@mka-soft.com.

Thank you.

mkaatr.