

Lab 8: Arrays

In this section, you are going to learn all about the power of arrays, and how easy they can make your programming life. First, you need to know what an array is.

1. Arrays and the Index Number

So one variable was holding one piece of information. An array is a variable that can hold more than one piece of information at a time. If you had an array variable called MyNumbers, you could hold more than one number at a time. You set them up like this:

```
Dim MyNumbers(4) As Integer
```

```
MyNumbers(0) = 1  
MyNumbers(1) = 2  
MyNumbers(2) = 3  
MyNumbers(3) = 4  
MyNumbers(4) = 5
```

When you set up an array with the Dim word, you put the name of your array variable, and tell Visual Basic how many items you want to store in the array. But you need to use parentheses around your figure. You then assign your data to a position in the array. In the example above we've set up an Integer array with 5 items in it. We've then said put number 1 into array position 0, put number 2 into array position 1, put number 3 into array position 2, and so on.

You might be thinking that the array was set to the number 4 - MyNumbers(4) - but always remember that an array starts counting at zero, and the first position in your array will be zero.

So that's what an array is - a variable that can hold more than one piece of data at a time - but how do they work? A programming example might help to clear things up.

- Start a new VB project.
- Add a Button to your Form.
- Set the Text property of the Button to "Integer Array"
- Put the following code behind your button click event:

```
Dim MyNumbers(4) As Integer
```

```
MyNumbers(0) = 10  
MyNumbers(1) = 20  
MyNumbers(2) = 30  
MyNumbers(3) = 40  
MyNumbers(4) = 50
```

```
MsgBox("First Number is: " & MyNumbers(0))  
MsgBox("Second Number is: " & MyNumbers(1))  
MsgBox("Third Number is: " & MyNumbers(2))  
MsgBox("Fourth Number is: " & MyNumbers(3))  
MsgBox("Fifth Number is: " & MyNumbers(4))
```

Test out the program when you are finished. The numbers 10 to 50 should have been displayed in your message boxes. In the code, we first set up an Integer array with 5 items in it.

```
Dim MyNumbers(4) As Integer
```

We then assigned values to each position in the array.

```
MyNumbers(0) = 10
```

To get at the values in the array, and display them in messages boxes, we just used the array name, followed by the position in the array.

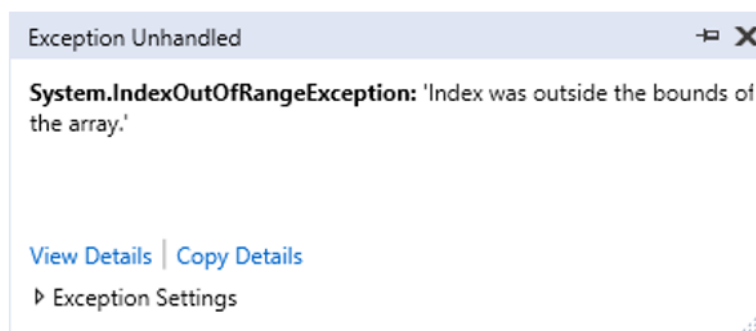
```
MsgBox("First Number is: " & MyNumbers(0))
```

So we've said, "Display whatever number is in array position 0, then display whatever number is in array position 1 ... " and so on.

Add another messages box statement on a line below the others. Put this:

```
MsgBox("Sixth Number is: " & MyNumbers(5))
```

Run your program again. When you clicked your button, you probably got an error box popping up, telling you that the "Index was outside the bounds of the array." This is the error message you may have received:



The Index that the error message is talking about is the figure in parentheses in your array. We set up this array

```
Dim MyNumbers(4) As Integer
```

And the highest Index number is therefore 4. But we tried to display something at index number 5:

```
MyNumbers(5)
```

Visual Basic said "Wait a minute, your array hasn't got a position number 5!" So it stopped the program and gave you an error message. Delete this line from your code:

```
MsgBox("Sixth Number is: " & MyNumbers(5))
```

So the way to get at information held in an array is through its Index number. A very handy way to get at the information in your array is by accessing its Index number in a For Loop and displays the results in a List Box.

Add a List Box to your form. Make it fairly wide, and just leave it on the default Name of ListBox1. Then change your code to the following:

```
Dim MyNumbers(4) As Integer
```

```
Dim i As Integer
```

```
MyNumbers(0) = 10
```

```
MyNumbers(1) = 20
```

```
MyNumbers(2) = 30
```

```
MyNumbers(3) = 40
```

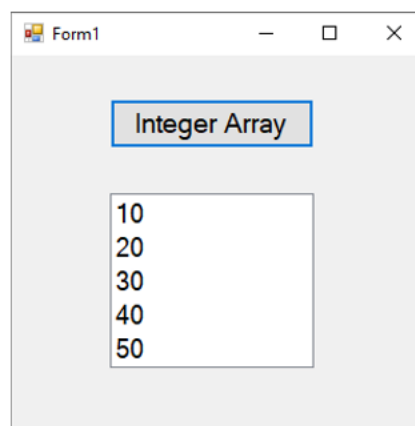
```
MyNumbers(4) = 50
```

```
For i = 0 To 4
```

```
    ListBox1.Items.Add(MyNumbers(i))
```

```
Next
```

Run your program, and click your button. Your form should look something like this one:



The first time round the loop, the variable called *i* will hold the number 0. Visual Basic will test to see if our end condition is met. The end condition was "Loop until the variable *i* holds the number 4". The variable *i* only holds the number 0, so Visual Basic drops down to the next line:

```
ListBox1.Items.Add(MyNumbers(i))
```

And what is inside the variable *i*? The number 0. So what's really getting adding to the List Box is this:

```
MyNumbers(0)
```

In other words, "Add to the List Box whatever is inside the array at position number 0". The next time round the loop, the variable *i* will hold the number 1. So this gets executed

```
ListBox1.Items.Add(MyNumbers(1))
```

And the loop continues round and around, adding whatever is inside our array until the end condition for the loop is met.

Arrays and Strings of Text

Arrays can hold other types of data, too. They can hold Strings of text.

- Put another Button on your Form
- Set the Text property to "String Array"
- Put the following code in your button click event:

```
Dim MyText(4) As String
```

```
Dim i As Integer
```

```
MyText(0) = "This"
```

```
MyText(1) = "is"
```

```
MyText(2) = "a"
```

```
MyText(3) = "String"
```

```
MyText(4) = "Array"
```

```
For i = 0 To 4
```

```
    ListBox1.Items.Add(MyText(i))
```

```
Next
```

When you have finished, run the program and click your new button. The text you put into the 5 array positions should display in the List Box.

2. Assigning Values to an Array

There are a number of ways you can put data into each position of an array. The code we just wrote for the two buttons had known values stored into each position. But you don't have to know what the values are. You can assign values straight from a Textbox into the position of your array. Like this:

```
MyNumbers(0) = Val(Textbox1.Text)
MyNumbers(1) = Val(Textbox2.Text)
```

With that code, whatever you typed into the Textboxes on your Form would be stored into the positions of your array. The same would be true of a String Array:

```
MyNumbers(0) = Textbox1.Text
MyNumbers(1) = Textbox2.Text
```

But do we have to keep typing out a value for each and every position of our array. What if we had an array with a hundred items in it, MyText(99)? Would we have to type out text for all one hundred positions of the array?

Well, obviously not. You can use code to assign values to your array. Here is an example where we don't type out values for all positions of an array. It's the times table again. This time we'll use an array. And we'll write a line of code to assign values to each position of the array.

- First, add another Button to your form.
- Set the Text Property to "Times TableArray"
- Add a Textbox to your Form
- Set the Text Property to a blank string (in other words, delete Textbox1 from the Text property)
- Add a Label near the Textbox
- Set the Text property of the Label to "Which Times Table do you want?"
- Insert a Listbox to your form
- Now double click your new button to get at the code window. Add the following code:

```
Dim numbers(10) As Integer
Dim times As Integer
Dim StoreAnswer As Integer
Dim i As Integer
```

```
ListBox1.Items.Clear()
```

```
times = Val(TextBox1.Text)
```

```
For i = 1 To 10
```

```
    StoreAnswer = i * times
```

```
numbers(i) = StoreAnswer  
ListBox1.Items.Add(times & " times " & i & " = " & numbers(i))
```

Next

Run the program. Enter a number in your new text box, and then click the Times Table Array button. The times table for the number should have been printed.

At the top of the code we set up three normal Integer variables, i, times and StoreAnswer. We also set up an array. (Notice that we set it to 10. This actually gives us 11 positions in the array. But we're only putting something in positions 1 to 10. This is because it is more convenient for us, and for our Loop.)

```
Dim numbers(10) As Integer
```

We need to know what number we are going to be multiplying by, which times table we're working out. We get this number from the Textbox, and can assign it directly to the variable times

```
times = Val(Textbox1.Text)
```

We can then set up a For Loop. Inside the For Loop is where we'll assign values to each position of our array:

```
numbers(i) = StoreAnswer
```

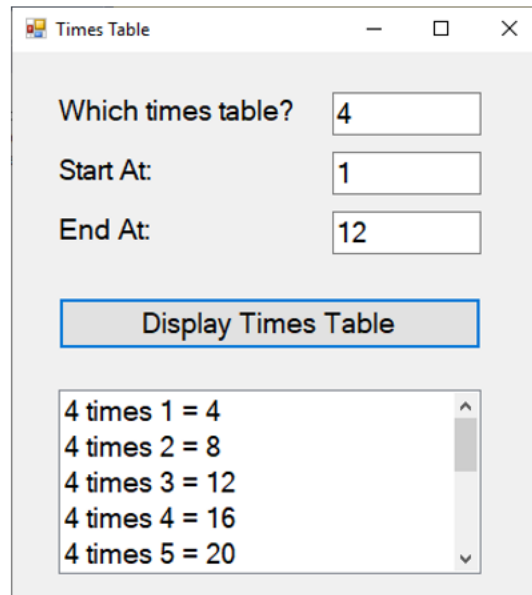
First time around the loop, the variable i will hold a value of 1. So the second position of our array, numbers(1) will be assigned whatever is in the variable StoreAnswer.

The second time around the loop, the variable i will hold a value of 2. So the second position of our array, numbers(2), will again be assigned whatever is in the variable StoreAnswer

We go round and round the loop assigning values to all ten positions of our array. The other two lines of code inside the array just work out the times tables, and Adds the answer to the List Box. Study them, and make sure you understand how they work.

3. Arrays where the Boundaries are not known

Study the following Form:



Times Table

Which times table? 4

Start At: 1

End At: 12

Display Times Table

4 times 1 = 4
4 times 2 = 8
4 times 3 = 12
4 times 4 = 16
4 times 5 = 20

In the above Form, the user is invited to enter values into three Textboxes. The first Textbox is for whatever times table he or she wants. The second Textbox asks for the starting value of the times table. The third Textbox is for the end number of the times table. In other words, 1 times 4, 2 times 4, 3 times 4, right up to 12 times 4.

The point is that the user can enter any values he or she wants. We won't know until they are entered, and the button is clicked. Up until now, we've used an array with a fixed size. Our previous times table program only went up to 10, and it started at 1. We used this to set up our array:

```
Dim numbers(10) As Integer
```

But that array would be no good for the above Form. Our array only held 11 positions. The user definitely wants the 4 times table right up to 12. Is there any way we can set up an array where the number of positions is not known? How can we set up a Non-Fixed size array?

You do it like this. First set up an array with empty brackets

```
Dim numbers( ) As Integer
```

Next, pass the values from the Text Boxes to some variables

```
times = Val(Textbox1.Text)  
startAt = Val(Textbox2.Text)  
endAt = Val(Textbox3.Text)
```

We can then use these values to reset the array. You reset an array by using the ReDim word. You then specify the new values. Like this:

```
ReDim numbers(endAt)
```

Our original array did not have its size set - Dim numbers() As Integer. So we've got the end number from the Textbox. When we reset an array, we can use this new value. Since our user entered the value 12 for the end number, our array is now really this:

```
Dim numbers(12) As Integer
```

We can use the same variables for our For Loop. Then we can go round and round the loop assigning values to our array.

To test this concept out, either start a new project, or amend the one you have displayed. Create three textboxes and labels. And add a new Button. Double click your button to open the code window. Then add the following code (the important line is in red):

```
Dim numbers() As Integer
Dim startAt As Integer
Dim endAt As Integer
Dim times As Integer
Dim StoreAnswer As Integer
Dim i As Integer

times = Val(TextBox1.Text)
startAt = Val(TextBox2.Text)
endAt = Val(TextBox3.Text)

ReDim numbers(endAt)

For i = startAt To endAt

    StoreAnswer = i * times
    numbers(i) = StoreAnswer
    ListBox1.Items.Add(times & " times " & i & " = " & numbers(i))

Next
```

When you're finished, run your program and test it out. Click the button and the times table should appear in the List Box.

And that is how to set up an array when you don't know what size the array is going to be - set up an array with empty brackets. Reset the array with the ReDim word, and then give it some new values.

Exercise

Write a program in VB.NET to handle 10 data values by including the following operations:

- Add up the values and display the total.
- Find the largest value, the smallest value and display them.
- Find the index of the largest value.