# Software Design and Development

## *8 Arrays*

*Hip hip array*

**Terry Marris July 2009**

We have completed our study of program constructs:

- sequences - one line after another:

```
x = 2
y = 3
t = x
```

- selections - If ... Then ... Else ... End If:

```
If x < 2 Then
    txtOut.Text = "x is smaller than 2"
Else If x > 2 Then
    txtOut.Text = "x is larger than 2"
Else
    txtOut.Text = "x is the same as 2"
End If
```

- iterations - While ... End While:

```
count = 0
While count < max
    string = string  + count.ToString() + ", "
    count = count + 1
End While
```

- function calls:

```
Function add(ByVal x As Integer, ByVal y As Integer) As Integer
   return x + y
End Function

   ...

  result = add(2, 3)
```

Now we start looking at data structures. The only data structure we shall examine is the one-dimensional array.

## 8.1 Arrays

If we want to store a collection of names we can use an array.

arrayOfNames

| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Tom | Dee | Harry | Ann | May |

The array is named *arrayOfNames*.

It has five *elements*, numbered from 0 up to 4 inclusive. Each element stores a single value. The values stored in an array must all be of the same type e.g. all Strings, or all Integers.

The numbers 0, 1, 2, 3 and 4 are *indices*. They identify each element of the array.

arrayOfNames(0) = Tom
arrayOfNames(1) = Dee
arrayOfrNames(2) = Harry
arrayOfNames(3) = Ann
arrayOfNames(4) = May

Array indices always start off at 0.

The number of elements in an array is known as its size or *length*. In our example that value is 5. The length of an array is always 1 more than the highest index.

## 8.2 Creating an Array

One way to fill an array with names is to write:

```
Dim arrayOfNames() As String = {"tom", "dee", "harry", "ann", "may"}
```

We know that *arrayOfNames* is an array because it has a pair of brackets () after its name. The length of this array is five because we (and VB) can count five names that are copied to the array. Each name is a String because each is surrounded by quotation marks. Each name is separated from the next by a comma and the whole list is enclosed within curly braces, { and }.

## 8.3 Printing an Array

We look at each element in the array and print it.

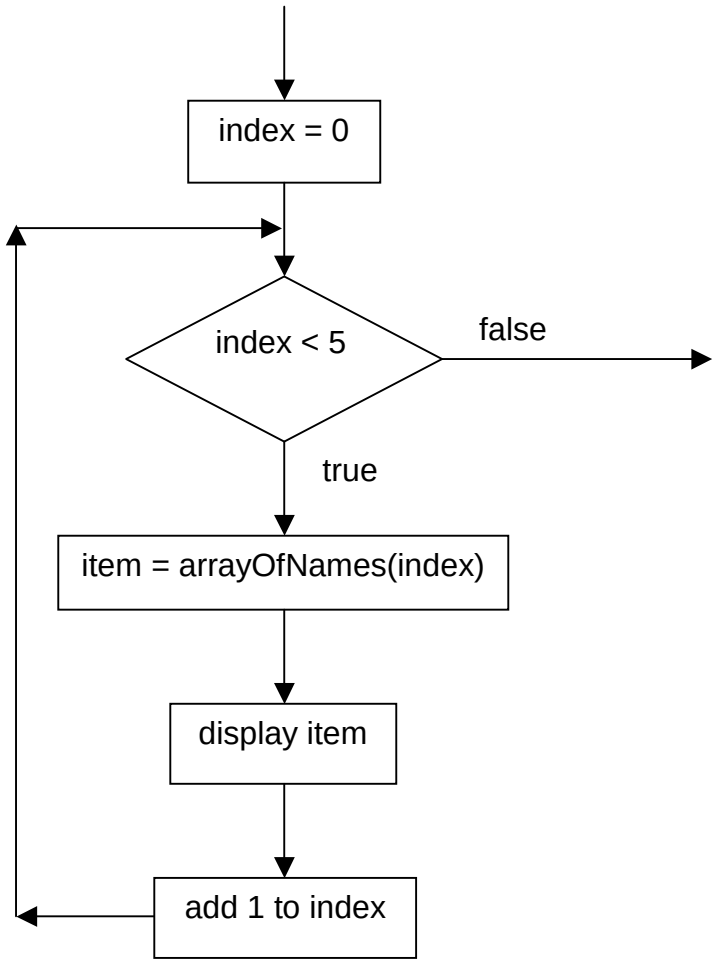
*Design - Structured English*

```
index = 0
while index < 5
        item = arrayOfNames(index)
        display item
        index = index + 1
end while
```

*index* goes from 0 up to 5. It starts with 0. Every time round the loop we add 1 to *index*.

When *index* = 0 we display the item in element number 0.
When *index* = 1 we display the item in element number 1
When *index* = 2 we display the item in element number 2
When *index* = 3 we display the item in element number 3
When *index* = 4 we display the item in element number 4
When *index* = 5 we stop looping.

*Design - Program Flowchart*



*Implementation - VB*

We add each item in the array onto the end of a string, and then show that string in a text box.

```
Dim str As String = ""
Dim i As Integer = 0
While i < arrayOfNames.Length
    str = str + arrayOfNames(i) + vbCrLf
    i = i + 1
End While
txtNames.Text = str
```

We start with an empty string.

Our Integer variable *i* is used to look at each element in the array in turn. It starts off at zero because we want to look at the first item in the array. Programmers often use i to stand for index.

Then we loop for as long as our variable *i* is strictly less than the length of the array.

We take the *i*'th item in the array and add it on to the end of our string.

When *i* is 0 we add the contents of the array in location 0.
When *i* is one we add the contents of the array in location 1.
When *i* is 2 we add the contents of the array in location 2.
...

After adding each item we add the newline character, vbCrLf. (vb carriage return line feed).

When *i* passes the end of the array the loop ends. We copy our string containing all the names to a text box for us to see.

*User Interface*

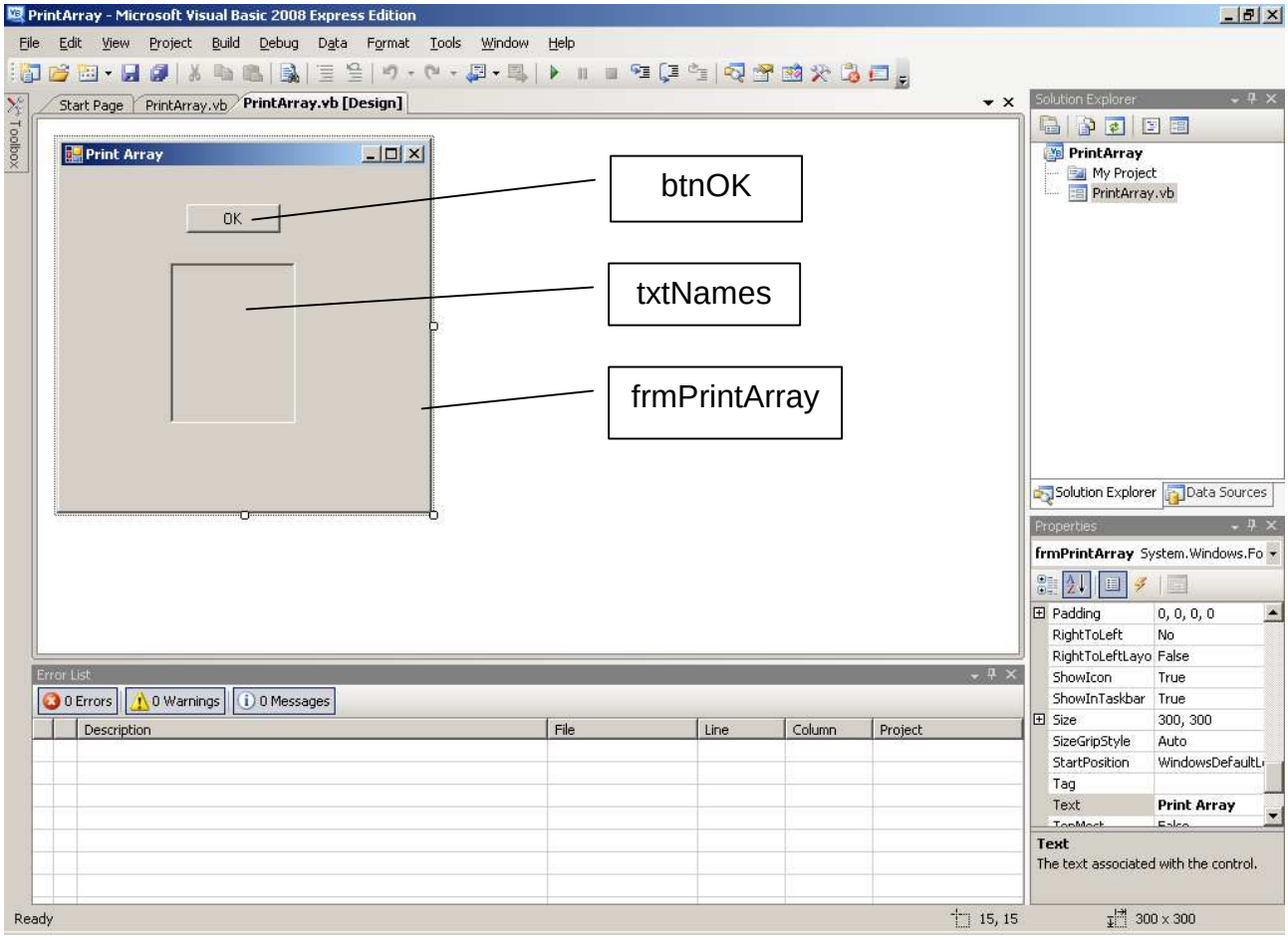
1. Start a new project and name it *PrintArray*

2. Name the vb file PrintArray.vb

3. Set
    a. Form
        i. Name = *frmPrintArray*
        ii. Text = *Print Array*

    b. Button
        i. Name = btnOK
        ii. Text = OK

    c. TextBox
        i. Name = txtNames
        ii. Multiline = True
        iii. Read Only = True

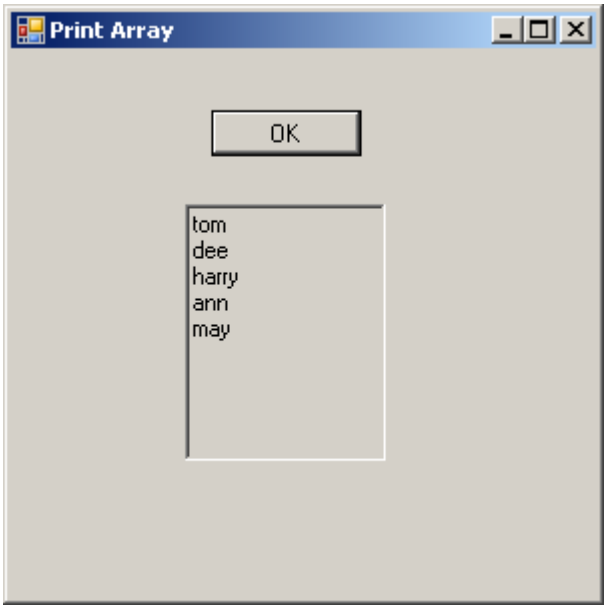


*Programming Code*

4. Rapid double click on the OK button. The entire program code is shown below. You do not repeat the *Public Class* ... heading nor the *Private Sub btnAdd_Click* ... heading. Nor do you repeat *End Sub* or *End Class*. They should already be there and you should not delete them.

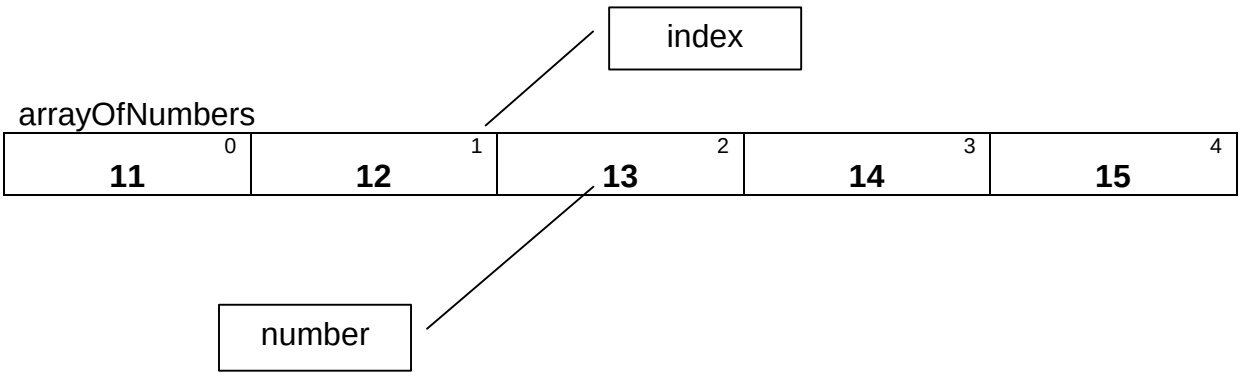
```
Public Class frmPrintArray
' Terry Marris 27 July 2009
' Prints an array of names

Function getArrayOfNames() As String()
     ' creates an array of 5 names
     Dim arrayOfNames() As String = {"tom","dee","harry","ann","may"}
     Return arrayOfNames
End Function

Function printArray(ByVal arrayOfNames As Array) As String
     ' prints an array of names in a text box
     Dim str As String = ""
     Dim i As Integer = 0
     While i < arrayOfNames.Length
          str = str + arrayOfNames(i) + vbCrLf
          i = i + 1
     End While
     txtNames.Text = str
     Return "OK"
End Function

Private Sub btnOK_Click(ByVal sender As System.Object, ByVal e As
                              System.EventArgs) Handles btnOK.Click
     ' displays an array of names
     Dim arrayOfNames() As String = getArrayOfNames()
     PrintArray(arrayOfNames)
End Sub
End Class
```

**5.** Click the OK button and the contents of the array are shown.




## 8.4 Summing an Array

We look at each item in an array in turn, and add each item to a sum.




The sum is 11 + 12 + 13 + 14 + 15 = 65

*Design - Structured English*

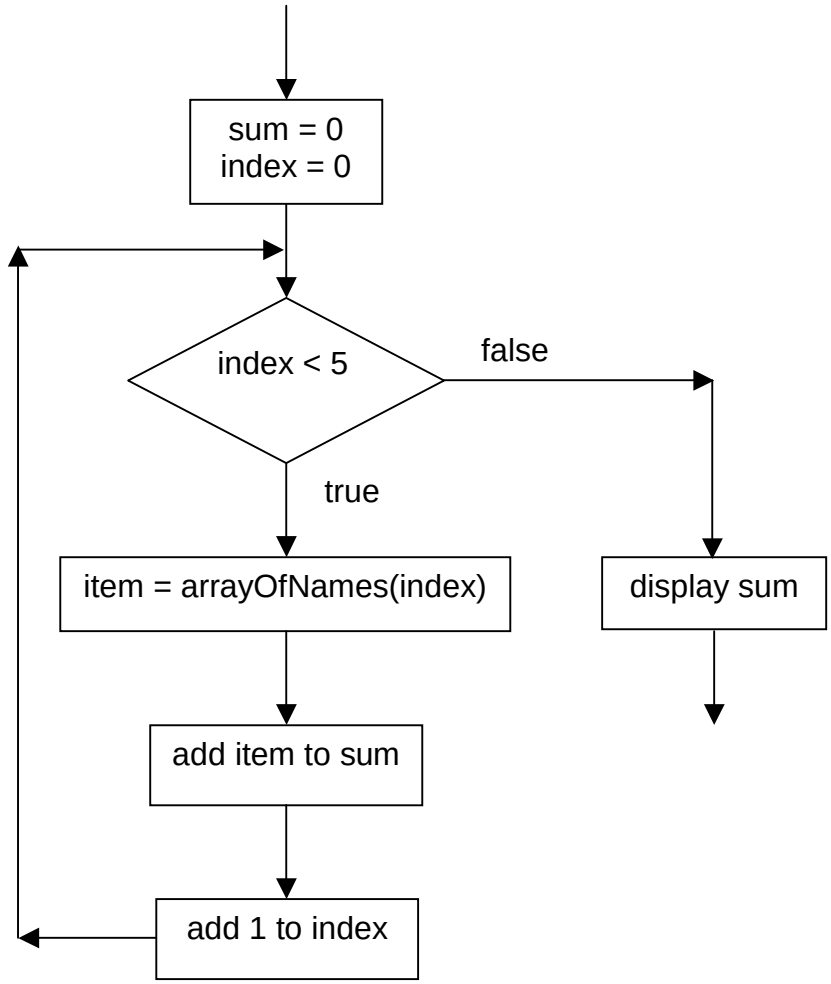
We look at each element in the array and add it to a sum.

```
sum = 0
index = 0
while index < 5
        item = arrayOfNames(index)
        sum = sum + item
        index = index + 1
end while
```

*index* goes from 0 up to 5. It starts with 0. Every time round the loop we add 1 to *index*.

When *index* = 0 we add the item in element number 0 to sum.
When *index* = 1 we add the item in element number 1 to sum
When *index* = 2 we add the item in element number 2 to sum
When *index* = 3 we add the item in element number 3 to sum
When *index* = 4 we add the item in element number 4 to sum
When *index* = 5 we stop looping.

*Design - Program Flowchart*




x

*Implementation - VB*

```
Dim intSum As Integer = 0
Dim i As Integer = 0
While i < arrayOfNumbers.Length
    intSum = intSum + arrayOfNumbers(i)
    i = i + 1
End While
txtSum.Text = intSum.ToString()
```

We start with *intSum* = 0.

Our Integer variable *i* is used to look at each element in the array in turn. It starts off at zero because we want to look at the first item in the array. Programmers often use i to stand for index.

Then we loop for as long as our variable *i* is strictly less than the length of the array.

We take the *i*'th item in the array and add it to *intSum*.

When *i* is 0 we add the contents of the array in location 0.
When *i* is one we add the contents of the array in location 1.
When *i* is 2 we add the contents of the array in location 2.
...

When *i* passes the end of the array the loop ends. We copy our final value of *intSum* to a text box for us to see.

*User Interface*

1. Start a new project and name it *SumArray*

2. Set File Name = *SumArray.vb*

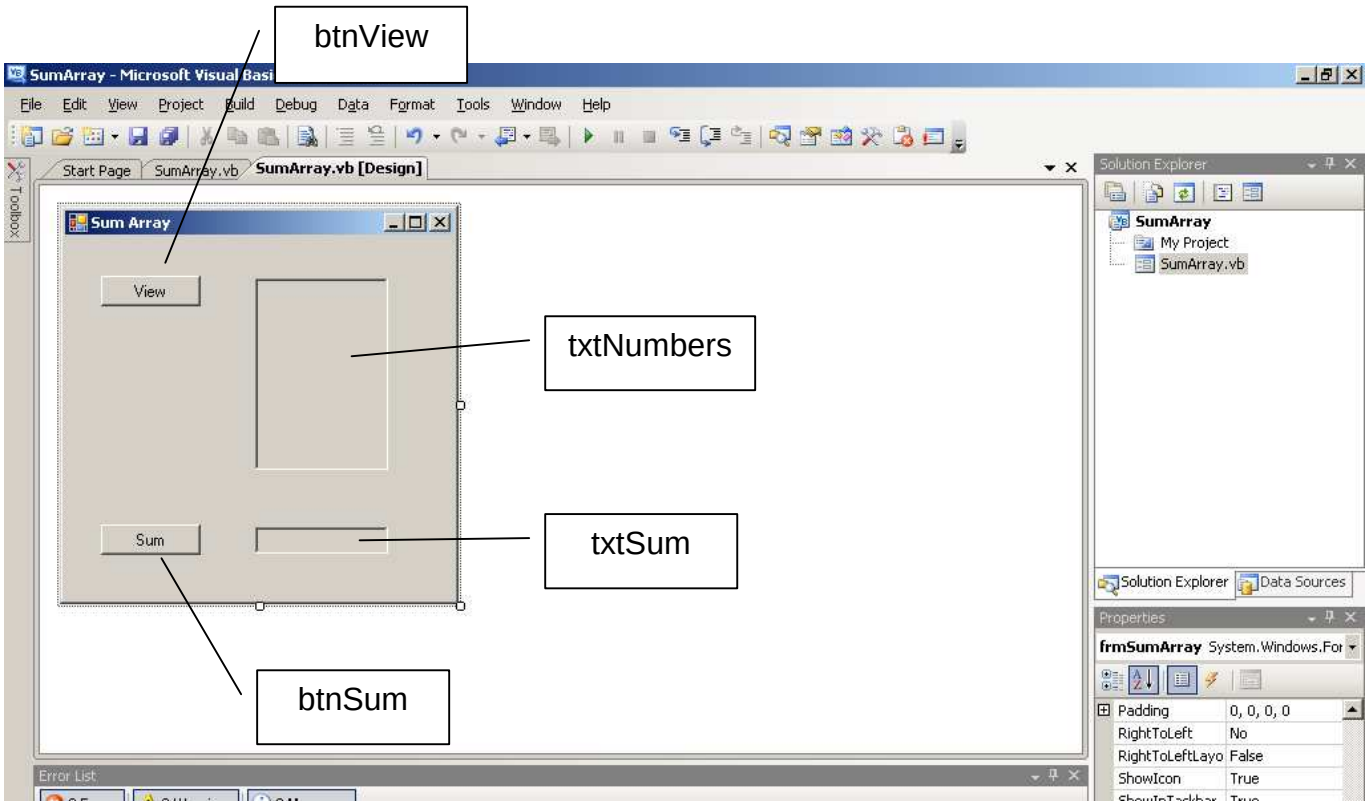
3. Set
    a. Form
        i. Name = *frmSumArray*
        ii. Text = *Sum Array*

    b. Button
        i. Name = *btnShow*
        ii. Text = *Show*

    c. Text Box
        i. Name = *txtNumbers*
        ii. Multiline = *True*
        iii. ReadOnly = *True*

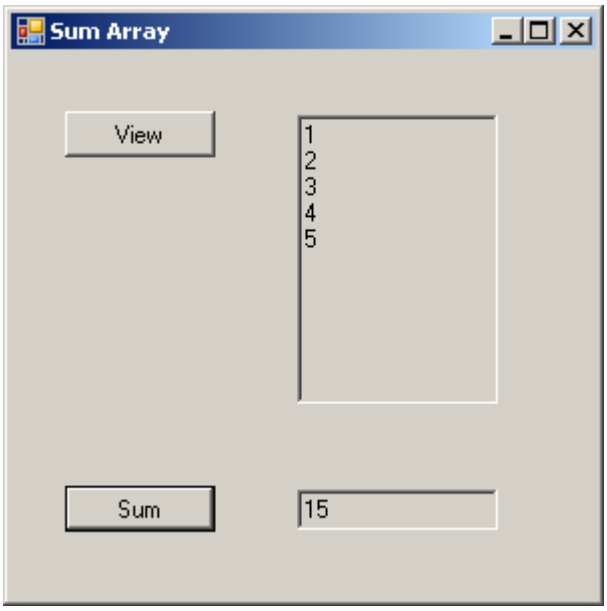
    d. Button
        i. Name = *btnSum*
        ii. Text = *Sum*

    e. Text Box
        i. Name = *txtSum*
        ii. ReadOnly = *True*



*Programming Code*

```
Public Class frmSumArray
    ' Terry Marris  28 July 2009
    ' Sums the contents of an array of integers

    Function getArrayOfNumbers() As Integer()
        ' creates an array of 5 numbers
        Dim arrayOfNumbers() As Integer = {1, 2, 3, 4, 5}
        Return arrayOfNumbers
    End Function

    Function printArray(ByVal arrayOfNumbers As Array) As String
        ' prints an array of numbers in a text box
        Dim str As String = ""
        Dim i As Integer = 0
        While i < arrayOfNumbers.Length()
            str = str + arrayOfNumbers(i).ToString() + vbCrLf
            i = i + 1
        End While
        txtNumbers.Text = str
        Return "OK"
    End Function

    Function getSumOfArray(ByVal arrayOfNumbers As Array) As Integer
        ' sums the contents of an array of numbers
        Dim intSum As Integer = 0
        Dim i As Integer = 0
        While i < arrayOfNumbers.Length
            intSum = intSum + arrayOfNumbers(i)
            i = i + 1
        End While
        Return intSum
    End Function

    Private Sub btnView_Click(ByVal sender As System.Object, ByVal e
                    As System.EventArgs) Handles btnView.Click
        ' displays an array of numbers
        Dim arrayOfNumbers() As Integer = getArrayOfNumbers()
        printArray(arrayOfNumbers)
    End Sub

    Private Sub btnSum_Click(ByVal sender As System.Object, ByVal e As
                    System.EventArgs) Handles btnSum.Click
        ' displays the sum of an array of numbers
        Dim arrayOfNumbers() As Integer = getArrayOfNumbers()
        txtSum.Text = getSumOfArray(arrayOfNumbers).ToString()
    End Sub
End Class
```




Notice that we have used small, simple values for testing - makes it easy for us see whether the results are correct.

## 8.5 Problem

A problem with the examples shown above is that it is the programmer and not the user who decides on the contents of the array. This is a problem to be fixed on the next programming course.

## Exercise 8.2

1. Design, write and test a program that prints the sum of the squares of 7 numbers. The numbers should be stored in an array.

2. Find the average (arithmetic mean) of 6 numbers. The numbers should be stored in an array.

3. Store the exam mark obtained by 10 students in an array. Count the number of students who passed. The pass mark is 50.

4. Count the number of negative and the number of non-negative numbers in a list containing 5 numbers.

## Bibliography

Deitel, Deitel & Yaeger *Simply Visual Basic.NET 2003* Pearson 2004 pp 374

**Next** we look at debugging.