

Declaring and Initializing in One Line

A simple variable declaration looks like this:

```
Dim num1 As Integer
```

Initializing the variable means to simply assign it a value:

```
num1 = 86
```

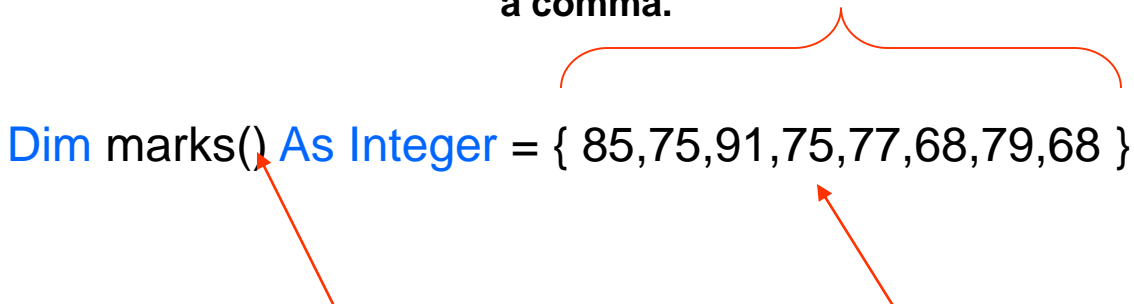
Variables can be declared and initialized in one line:

```
Dim num2 As Integer = 75
```

Similarly, an array can be declared and initialized in one line. This is useful for cases where the values of each element in the array are known ahead of time.

The values for each element in the array are placed within curly braces and separated by a comma.

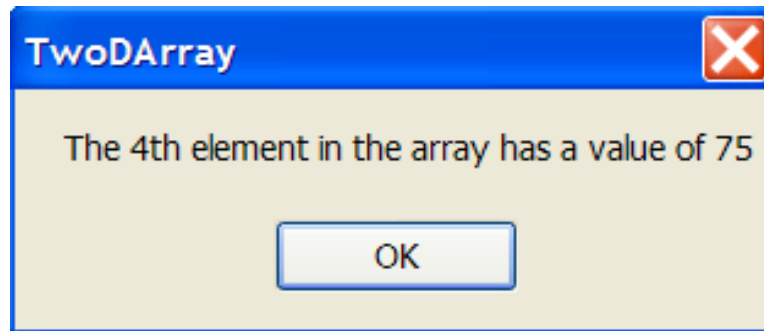
```
Dim marks() As Integer = { 85,75,91,75,77,68,79,68 }
```



No number is required here, the compiler looks at the number of values in the initialization part of the code and automatically 'knows' how many elements go in the array.

An Example

```
Public Class Form1
    Dim marks() As Integer = {85, 75, 91, 75, 77, 68, 79, 68}
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
        MsgBox("The 4th element in the array has a value of " & marks(3))
    End Sub
End Class
```



2 Dimensional Arrays

A single array such as the one we have been dealing with looks like this.

Dim marks(8) As String

85	75	91	75	77	68	79	68
----	----	----	----	----	----	----	----

**In many cases information needs to be displayed as a series of lists.
You can think of this as an array of arrays.**

An Array of Arrays

If the table in the previous slide represents the 8 test marks of a single student, then a series of 5 students' marks could be represented like this.

85	75	91	75	77	68	79	68
82	78	74	68	85	96	89	78
77	87	91	78	56	74	77	75
78	74	65	84	58	77	68	56
74	78	87	74	76	87	85	77

A Simple 2D Array

Lets begin with a scenario involving 4 students and the marks they received on 3 tests.

Student	Test1	Test2	Test3
one	74	76	77
two	54	53	57
three	84	88	80
four	65	64	67

The values being stored in the array are whole numbers so the array will be of type integer. We have the 3 test marks of 4 students so the array will have dimensions of 4 X 3.(rows by columns).

marks(row,column)

Declaring a 4 X 3 Array

4 rows by 3 columns gets declared as follows.

Dim marks(4,3) As Integer

```
Public Class Form1
    Dim marks(4, 3) As Integer
    Private Sub Form1_Load(ByVal sender As System.Object,
        marks(0, 0) = 74
        marks(0, 1) = 76
        marks(0, 2) = 77
        marks(1, 0) = 54
        marks(1, 1) = 53
        marks(1, 2) = 57
        marks(2, 0) = 84
        marks(2, 1) = 88
        marks(2, 2) = 80
        marks(3, 0) = 65
        marks(3, 1) = 64
        marks(3, 2) = 67
    End Sub
End Class
```

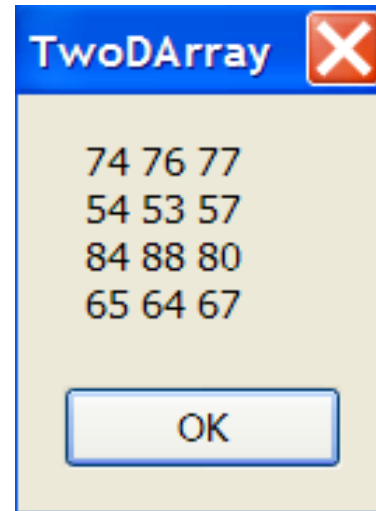
← **Initializing...the hard way!**

Using Nested Loops to Display 2D Arrays

Nested loops and 2D arrays work very well together. The outer loop affects the rows in the array while the inner loop affects the columns.

```
Private Sub btnDisplay_Click(ByVal sender As System.Object
    Dim row As Integer
    Dim column As Integer
    Dim output As String = ""
    Dim wrap As String = Chr(13) & Chr(10)

    For row = 0 To 3
        For column = 0 To 2
            output = output & " " & marks(row, column)
            MsgBox(output)
        Next
        output = output & wrap
        MsgBox(output)
    Next
End Sub
```



An Easier Way to Initialize

Each row has enclosed curly brace.
and is separated from other rows by
comma.

```
Dim marks(,) As Integer = {{74, 76, 77}, {54, 53, 57}, {84, 88, 80}, {65, 64, 67}}
```

Comma indicates a
2D array.

Extra curly brace at beginning and
end of initializers

Using Nest Loops To Enter Values

```
Public Class Form1
    Dim marks(4, 3) As Integer
    Private Sub btnInitialize_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnInitialize.Click
        Dim row As Integer
        Dim column As Integer

        For row = 0 To 3
            For column = 0 To 2
                marks(row, column) = Val(TextBox1.Text)
            Next column
        Next row
    End Sub
End Class
```

