

Initializing Arrays

This lesson covers initializing static, dynamic and implicitly typed arrays.

WE'LL COVER THE FOLLOWING ^

- Initializing Static Arrays
 - Example Explanation
- Initializing Dynamic Arrays
- Implicitly Typed Arrays

Initializing Static Arrays

An array can be declared and filled with the default value using square bracket (`[]`) initialization syntax. For example, creating and initializing an array of 5 integers:

```
using System;

public class MainClass {

    public static void Main(String [] args)
    {
        int[] arr = { 24, 2, 13, 47, 45 };
        foreach(var item in arr)
        {
            Console.WriteLine(item.ToString());
        }
    }
}
```



Example Explanation

In the above example, in line 7 we have an *array* `arr`. We **don't** need to specify its size - the compiler will automatically know that it's an array of size

5, depending on the number of values stored.

We use *indexing* to refer to the *location* of the values in an *array*. For example:

- The value **24** is present at the index **0** of the *array*. This is written as `arr[0] = 24`. Here, by using `arr[0]` we are referring to the location of **24**.
- Similarly, the *index* of the value **2** in the *array* will be **1**. It'll be written `arr[1] = 2`.
- The *index* of the value **13** is **2** in the *array*. So `arr[2] = 13`.

Note: Indices in C# are zero-based. The indices of the array above will be **0-9** i.e. Arrays have **0** as the *first* index not **1**.

The `new int[]` portion can be omitted when declaring an array variable. This is not a *self-contained expression*, so using it as part of a different call does not work:

```
using System;

public class MainClass {

    public static void Main(String [] args)
    {
        int[] arr1;
        arr1 = { 24, 2, 13, 47, 45 }; // Won't compile
    }
}
```

Initializing Dynamic Arrays

We discussed earlier that the system starts counting the element index from **0**. Moreover, accesses to elements of arrays are done in **constant time**. That means:

Accessing to the first element of the array has the same cost (in time) of accessing the second element, the third element and so on.

This is how you'll initialize a dynamic array:

```
using System;

public class MainClass {

    public static void Main(String [] args)
    {
        int[] arr = new int[3] {7,9,4};
        Console.WriteLine(arr[0]); //outputs 7
        Console.WriteLine(arr[1]); //outputs 9
    }
}
```



An array can also be created and initialized with custom values using collection initialization syntax, without specifying the array size:

```
using System;

public class MainClass {

    public static void Main(String [] args)
    {
        int[] arr = new int[] { 24, 2, 13, 47, 45 };
        Console.WriteLine(arr[0]); //outputs 24
        Console.WriteLine(arr[1]); //outputs 2
    }
}
```



Implicitly Typed Arrays

Alternatively, in combination with the **var** keyword, the specific type may be omitted so that the type of the array is inferred:

```
using System;

public class MainClass {

    public static void Main(String [] args)
    {
        // same as int[]
        var arr = new [] { 1, 2, 3 };
        foreach(var item in arr)
        {

```

```
        Console.WriteLine(item.ToString());
    }

    // same as string[]
    var arr1 = new [] { "one", "two", "three" };
    foreach(var item in arr1)
    {
        Console.WriteLine(item.ToString());
    }

    // same as double[]
    var arr2 = new [] { 1.0, 2.0, 3.0 };
    foreach(var item in arr2)
    {
        Console.WriteLine(item.ToString());
    }
}
}
```



So far you've now got the idea of how we initialize arrays - now let's move over accessing array values!