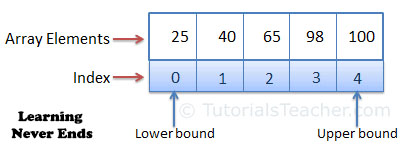
**Arrays In C#**

* An array is a list of items or collection of elements which is of same or similar data type (homogenous elements).
* An array is a collection of elements of a single data type stored in adjacent memory locations.
* An array is a collection of related values placed in contiguous memory locations and these values are referenced using a common array name.
* An array simplifies the task of maintaining these values
* An array always stores values of a single data type.
* Each value is referred to as an **element**.
* These elements are accessed using **subscripts** or **index numbers** that determine the position of the element in the array list.
* C# supports **zero-based index** values in an array.
* This means that the first array element has an index number zero while the last element has an index number n-1, where n stands for the total number of elements in the array.
* This arrangement of storing values helps in efficient storage of data, easy sorting of data, and easy tracking of the data length.



**Declaring Arrays**

Arrays are reference type variables whose creation involves two steps:

**Declaration**

* An array declaration specifies the **type of data** that it can hold and an **identifier**.
* This identifier is basically an **array name** and is used with a **subscript / index** to retrieve or set the data value at that location.

**Memory allocation:**

* Declaring an array does not allocate memory to the array.

**Following is the syntax for declaring an array:**

data\_type[] arrayName;

**In Above syntax:**

* **data\_type:** Specifies the data type of the array elements (for example, int and char).
* **arrayName**: Specifies the name of the array.

**Initializing Arrays**

**An array can be:**

* Created using the new keyword and then initialized.
* Initialized at the time of declaration itself, in which case the new keyword is not used.

Creating and initializing an array with the new keyword involves specifying the size of an array.

The number of elements stored in an array depends upon the specified size.

The new keyword allocates memory to the array and values can then be assigned to the array.

If the elements are not explicitly assigned, default values are stored in the array.

**The following table lists the default values for some of the widely used data types:**

| **DATA TYPES** | **DEFAULT VALUES** |
| --- | --- |
| int | 0 |
| float | 0.0 |
| double | 0.0 |
| char | '\0' |
| string | Null |

﻿

**The following syntax is used to create an array:**

type[] arrayName = new type[size-value];

**In Above Syntax,**

* size-value: Specifies the number of elements in the array. You can specify a variable of type int that stores the size of the array instead of directly specifying a value.

**The following code creates an integer array which can have a maximum of five elements in it:**

public int[] number = new int[5];

number[0] = 11;  
number[1] = 22;  
number[2] = 33;  
number[3] = 44;  
number[4] = 55;

**The following syntax is used to create and initialize an array without using the new keyword:**

type[] arrayIdentifier = {val1, val2, val3, ..., valN};

**In Above Syntax,**

* **val1**: It is the value of the first element.
* **valN**: It is the value of the nth element.

**Example:**

public string[] studNames = new string{“Allan”, “Wilson”, “James”, “Arnold”};

**Using The Foreach Loop For Arrays**

**The foreach loop:**

* In C# is an extension of the for loop.
* Is used to perform specific actions on large data collections and can even be used on arrays.
* Reads every element in the specified array.
* Allows you to execute a block of code for each element in the array.
* Is particularly useful for reference types, such as strings.

**Source Code - Example 1**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ArraysDemo

{

class Program

{

static void Main(string[] args)

{

//string[] myArray = { "Adil", "Zubia", "Sameed", "Ammad" };

//Console.WriteLine(myArray.Length);

//for (int i = 0; i < myArray.Length; i++)

//{

// Console.WriteLine(myArray[i]);

//}

//foreach (string name in myArray)

//{

// Console.WriteLine(name);

//}

//Console.WriteLine("Foreach Loop Ends..");

//int[] myArray = new int[] { 10,20,30,40 };

//int[] myArray = new int[4];

//myArray[0] = 10;

//myArray[1] = 20;

//myArray[2] = 30;

//myArray[3] = 40;

//Console.WriteLine(myArray[0]);

//Console.WriteLine(myArray[1]);

//Console.WriteLine(myArray[2]);

//Console.WriteLine(myArray[3]);

Console.ReadLine();

}

}

}

﻿

**Source Code - Example 2**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Arrays\_Demo

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("How many names you want to store ??");

int size = int.Parse(Console.ReadLine()); // 5

string[] names = new string[size];

for (int i = 0; i < size; i++)

{

Console.WriteLine("Enter Data: ");

string name = Console.ReadLine();

names[i] = name;

}

Console.WriteLine("----------------");

foreach (string name in names)

{

Console.WriteLine(name);

}

//string[] names = new string[] {"Adil","Ali","Anas","Amir" };

//string[] names = {"Adil", "Zubia", "Anum", "Farah","Abdul Rehman","Sameed","Ammad" };

//foreach(string i in names)

//{

// Console.WriteLine(i);

//}

//Console.WriteLine("Foreach loop terminates..");

//for(int i = 0; i < names.Length; i++)

//{

// Console.WriteLine(names[i]);

//}

// 2 types of arrays

// single dimensional array

// multi dimensional array

//string[] names = new string[3];

//names[0] = "Abdul Rehman";

//names[1] = "Zubia";

//names[2] = "Saad";

//Console.WriteLine(names[0]);

//Console.WriteLine(names[1]);

//Console.WriteLine(names[2]);

// 4 types of loops

// for loop

// while loop

// do while loop

// foreach loop - arrays

//int[] nums = new int[3];

//nums[0] = 11;

//nums[1] = 22;

//nums[2] = 33;

//nums[3] = 44;

//Console.WriteLine(nums[2]);

//comment - ctrl + k, ctrl + c

//uncomment - ctrl + k, ctrl + u

Console.ReadLine();

}

}

}