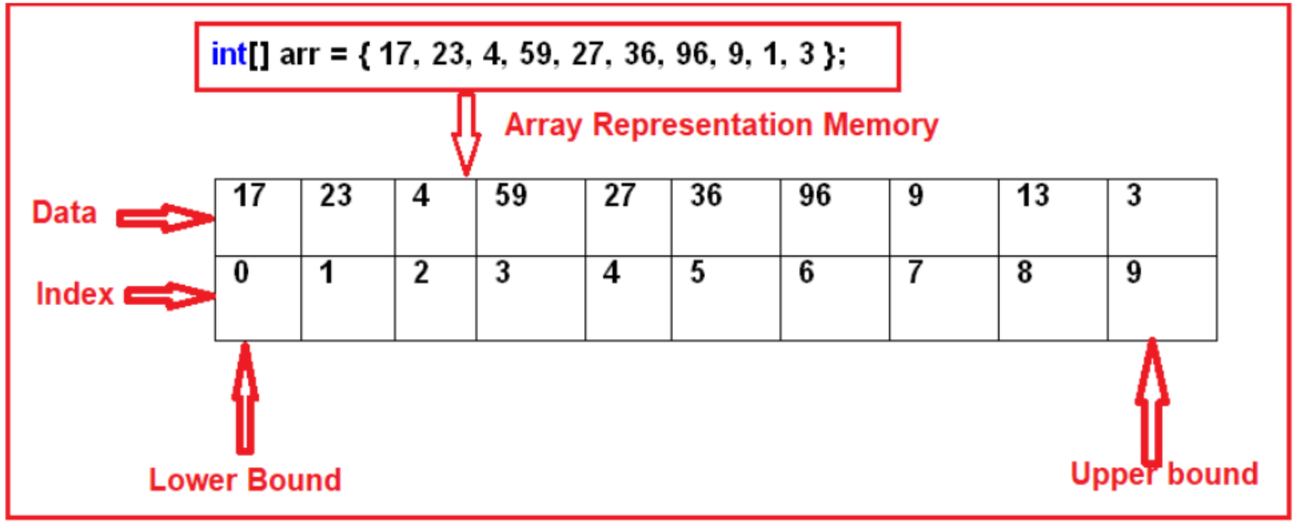
Collections in C#

What is an Array in C#?

In simple words, we can define an array as a collection of similar types of values that are stored in sequential order i.e. they are stored in a contiguous memory location.

Memory Representation of Arrays in C#:



```
int[] arr = { 17, 23, 4, 59, 27, 36, 96, 9, 1, 3 };
```

Data	17	23	4	59	27	36	96	9	13	3
Index	0	1	2	3	4	5	6	7	8	9

1. **Sort(<array>):** Sorting the array elements
2. **Reverse (<array>):** Reversing the array elements
3. **Copy (src, dest, n):** Copying some of the elements or all elements from the old array to the new array
4. **GetLength(int):** A 32-bit integer that represents the number of elements in the specified dimension.
5. **Length:** It Returns the total number of elements in all the dimensions of the Array; zero if there are no elements in the array.

Disadvantages of using Arrays in C#:

1. The array size is fixed. So, we should know in advance how many elements are going to be stored in the array. Once the array is created, then we can never increase the size of an array. If you want then we can do it manually by creating a new array and copying the old array elements into the new array.
2. As the array size is fixed, if we allocate more memory than the requirement then the extra memory will be wasted. On the other hand, if we allocate less memory than the requirement, then it will create the problem.
3. We can never insert an element into the middle of an array. It is also not possible to delete or remove elements from the middle of an array.

What is a Collection in C#?

The **Collections in C#** are a set of predefined classes that are present in the **System.Collections** namespace that provides greater capabilities than the traditional arrays. The collections in C# are reusable, more powerful, more efficient and most importantly they have been designed and tested to ensure quality and performance.

1. Size can be increased dynamically.
2. We can insert an element into the middle of a collection.
3. It also provides the facility to remove or delete elements from the middle of a collection.

Auto-Resizing of collections:

The capacity of a collection increases dynamically i.e. when we keep adding new elements, then the size of the collection keeps increasing automatically. Every collection class has three constructors and the behavior of collections will be as following when created using a different constructor.

1. **Default Constructor:** It Initializes a new instance of the collection class that is empty and has the default initial capacity as zero which becomes four after adding the first element and whenever needed the current capacity becomes double.
2. **Collection (int capacity):** This constructor initializes a new instance of the collection class that is empty and has the specified initial capacity, here also when the requirement comes current capacity doubles.
3. **Collection (Collection):** It Initializes a new instance of the collection class that contains elements copied from the specified collection and that has the same initial capacity as the number of elements copied, here also when the requirement comes current capacity doubles.

What is ArrayList in C#?

The **ArrayList in C#** is a collection class that works like an array but provides the facilities such as dynamic resizing, adding, and deleting elements from the middle of a collection. It implements the System.Collections.IList interface using an array whose size is dynamically increased as required.

Methods and Properties of ArrayList Collection class in C#:

The following are the methods and properties provided by the ArrayList collection class in C#.

1.Add(object value): This method is used to add an object to the end of the collection.

2.Remove(object obj): This method is used to remove the first occurrence of a specific object from the collection.

3.RemoveAt(int index): This method takes the index position of the elements and removes that element from the collection.

4.Insert(int index, Object value): This method is used to inserts an element into the collection at the specified index.

5.Capacity: This property gives you the capacity of the collection means how many elements you can insert into the collection.

```
ArrayList al = new ArrayList();
Console.WriteLine("Initial Capacity: " + al.Capacity);
al.Add(10);

// Inserting an element on index
al.Insert(2, false);

// Delete an element on index
al.Remove(true);
```

Generic Collections in C#:

The **Generic Collections in C#** are strongly typed. The strongly typed nature allows these collection classes to store only one type of value into it. This not only eliminates the type mismatch at runtime but also we will get better performance as they don't require boxing and unboxing while they store value type data.

1. **Stack<T>,**
2. **Queue<T>,**
3. **LinkedList<T>,**
4. **SortedList<T>,**
5. **List<T>,**
6. **Dictionary<TKey, Tvalue>**

```
List<Customer> listCustomer = new List<Customer> ();
```