C# ArrayList

In C#, an `ArrayList` stores elements of multiple data types whose size can be changed dynamically. For example,

```
using System;
using System.Collections;

class Program
{
    public static void Main()
    {
        // create an ArrayList
        ArrayList student = new ArrayList();

        // add elements to ArrayList
        student.Add("Jackson");
        student.Add(5);

        // display every element of myList
        for (int i = 0; i < student.Count; i++)
        {
            Console.WriteLine(student[i]);
        }
    }
}
```

Output

```
Jackson
5
```

Here, `student` is an `ArrayList` that contains elements (`"Jackson"` and **5**) of different data types.

We will learn about `ArrayList` in detail.

Create an ArrayList

To create `ArrayList` in C#, we need to use the `System.Collections` namespace. Here is how we can create an arraylist in C#.

```
// create an arraylist
ArrayList myList = new ArrayList();
```

Here, we have created an arraylist named `myList`.

Basic Operations on ArrayList

In C#, we can perform different operations on arraylists. We will look at some commonly used arraylist operations in this tutorial:

- Add Elements
- Access Elements
- Change Elements
- Remove Elements

Let's see how we can perform these operations in detail!

Add Elements in ArrayList

C# provides a method `Add()` using which we can add elements in `ArrayList`. For example,

```
using System;
using System.Collections;

class Program
{
    public static void Main()
    {
        // create an ArrayList
        ArrayList student = new ArrayList();

        // add elements to ArrayList
        student.Add("Tina");
        student.Add(5);



    }
}
```

In the above example, we have created an `ArrayList` named `student`.
Then we added `"Tina"` and **5** to the `ArrayList` using the `Add()` method.

> **Note:** `ArrayList` stores elements with different data types. However, if you want to store elements of the same data type use `List<T>` class instead.

Access ArrayList Elements

We use indexes to access elements in `ArrayList`. The indexing starts from **0**.
For example,

```
using System;
using System.Collections;

class Program
{
    public static void Main()
    {
        // create an ArrayList
```

```
        ArrayList schoolDetails = new ArrayList();
        schoolDetails.Add("Mary's");
        schoolDetails.Add("France");
        schoolDetails.Add(23);

        // access the first element
        Console.WriteLine("First element: " + schoolDetails[0]);

        // access the second element
        Console.WriteLine("Second element: " + schoolDetails[1]);

    }
}
```

Output

```
First element: Mary's
Second element: France
```

Since the index of the `ArrayList` starts from **0**:

- `schoolDetails[0]` - accesses the first element
- `schoolDetails[1]` - accesses the second element

Iterate ArrayList

In C#, we can also loop through each element of `ArrayList` using a `for` loop. For example,

```
using System;
using System.Collections;

class Program
{
    public static void Main()
    {
        // create an ArrayList containing 3 elements
```

```
        ArrayList myList = new ArrayList();

        myList.Add("Science");
        myList.Add(true);
        myList.Add(5);

        // display every element of myList
        for (int i = 0; i < myList.Count; i++)
        {
            Console.WriteLine(myList[i]);
        }

    }
}
```

Output

```
Science
True
5
```

In the above example, we have looped through `myList` using a `for` loop.

Here, `myList.Count` gives the number of elements in `myList`.

Change ArrayList Elements

We can change the value of elements in `ArrayList` as:

```
using System;
using System.Collections;

class Program
{
    public static void Main()
    {
        // create an ArrayList
        ArrayList myList = new ArrayList();
```

```
        myList.Add("Harry");
        myList.Add("Miller");

        Console.WriteLine("Original Second element: " + myList[1]);

        // change the value of second element
        myList[1] = "Styles";

        Console.WriteLine("Updated second element: " + myList[1]);
    }
}
```

Output

```
Original Second element: Miller
Updated second element: Styles
```

Here, we have changed the value of the second element in `myList`.

Remove ArrayList Elements

C# provides methods like `Remove()`, `RemoveAt()`, `RemoveRange()` to remove elements from `ArrayList`.

We will see an example below using `Remove()` to a remove element:

```
using System;
using System.Collections;

class Program
{
    public static void Main()
    {
        // create an ArrayList
        ArrayList myList = new ArrayList();
        myList.Add("Jack");
```

```
        myList.Add(4);
        myList.Add("Jimmy");

        // remove "Jack" from myList
        myList.Remove("Jack");


        // iterate through myList after removing "Jack"
        for (int i = 0; i < myList.Count; i++)
        {
            Console.WriteLine(myList[i]);
        }
    }
}
```

Output

```
4
Jimmy
```

In the above example, we have removed `"Jack"` from `myList` using the `Remove()` method.

How to check whether an element is present inside `ArrayList`?

C# provides a method `Contains()` using which we can determine whether an element is present inside `ArrayList`. For example,

```
using System;
using System.Collections;

class Program
{
    public static void Main()
    {
        // create an ArrayList
        var myList = new ArrayList() { "Delicate", "Willow", "Style", 3 };

        // check whether myList contains "Willow"
        var result = myList.Contains("Willow");
```

```
        Console.WriteLine(result);
    }
}
```

Output

```
True
```

Here, `myList` contains `"Willow"` so `myList.Contains("Willow")` returns `True`.