# Lists in C#

In C#, a List<T> is a generic collection that provides dynamic array functionality. It is part of the System.Collections.Generic namespace and allows you to store a collection of objects of any specific type. The List<T> class provides methods to perform operations like adding, removing, searching, and sorting items.

**Key Features of List<T>:**

- **Dynamic Size**: Unlike arrays, which have a fixed size, a List<T> can dynamically grow and shrink as you add or remove elements.
- **Type Safety**: Being generic, List<T> ensures that all elements are of the same type, which provides type safety and avoids runtime errors.
- **Indexing**: You can access elements using an index, similar to arrays.
- **Rich API**: The List<T> class provides many useful methods to manipulate the collection.

**Example: Basic Operations with List<T>**

```
using System;
using System.Collections.Generic;

class Program
{
  static void Main()  {
    // Creating a list of integers
    List<int> numbers = new List<int>();

    // Adding elements to the list
    numbers.Add(10);
    numbers.Add(20);
    numbers.Add(30);

    // Adding multiple elements at once
    numbers.AddRange(new int[] { 40, 50, 60 });

    // Accessing elements by index
    Console.WriteLine("Element at index 0: " + numbers[0]); // Output: 10
    Console.WriteLine("Element at index 2: " + numbers[2]); // Output: 30
```

```
    // Iterating over the list
    Console.WriteLine("All elements:");
    foreach (int number in numbers)
    {
        Console.WriteLine(number);
    }

    // Removing an element by value
    numbers.Remove(20); // Removes the first occurrence of 20

    // Removing an element by index
    numbers.RemoveAt(0); // Removes the element at index 0 (which was 10)

    // Finding an element
    int foundNumber = numbers.Find(x => x > 30);
    Console.WriteLine("First number greater than 30: " + foundNumber); // Output: 40

    // Checking if the list contains an element
    bool contains50 = numbers.Contains(50);
    Console.WriteLine("List contains 50: " + contains50); // Output: True

    // Sorting the list
    numbers.Sort();
    Console.WriteLine("Sorted list:");
    foreach (int number in numbers)
    {
        Console.WriteLine(number);
    }

    // Getting the number of elements in the list
    Console.WriteLine("Count of elements: " + numbers.Count); // Output: 4
} }
```

**Explanation:**

1. **Creating a List**:

List<int> numbers = new List<int>();

This creates an empty list of integers.

2. **Adding Elements**:

numbers.Add(10);

numbers.AddRange(new int[] { 40, 50, 60 });

The Add method adds a single element, while AddRange allows adding multiple elements at once.

3. **Accessing Elements**:

Console.WriteLine("Element at index 0: " + numbers[0]);

Elements can be accessed using an index, starting from 0.

4. **Iterating Over a List**:

foreach (int number in numbers)

{ Console.WriteLine(number); }

You can iterate over the elements using a foreach loop.

5. **Removing Elements**:

numbers.Remove(20);

numbers.RemoveAt(0);

Remove deletes the first occurrence of a specified element, while RemoveAt deletes the element at a specified index.

6. **Finding an Element**:

int foundNumber = numbers.Find(x => x > 30);

Find searches for the first element that matches a specified condition using a lambda expression.

7. **Checking for an Element**:

bool contains50 = numbers.Contains(50);

Contains checks if a specific element is present in the list.

8. **Sorting the List**:

numbers.Sort();

Sort arranges the elements in ascending order.

9. **Counting Elements**:

Console.WriteLine("Count of elements: " + numbers.Count);

Count returns the number of elements in the list.