TutorialKart

C# List – List Operations with Examples

C# List

C# List is a collection of elements that can be used to store and fetch dynamically.

C# List preserves the index of elements in it.

C# List can have duplicate elements.

In C#, you can access List in System.Collections.Generic.

Initialize C# List

You can declare a C# List and add elements to it.

Example 1 – Initialize List

In this example, we will define a list and then add elements to it, thus initializing the list with some elements.

Program.cs

```
using System.Collections.Generic;

namespace CSharpExamples {

    class Program {
        static void Main(string[] args) {
            var names = new List<string>();
            names.Add("Mack");
            names.Add("Daisy");
            names.Add("Ward");
        }
    }
}
```

Example 2 – Initialize List with Values

We can also define a C# list with initial values. In the following example, we will initialize the list during its

declaration itself.

Program.cs

```
using System.Collections.Generic;

namespace CSharpExamples {

    class Program {
        static void Main(string[] args) {
            var names = new List<string>() {"Mack", "Daisy", "Ward"};
        }
    }
}
```

Print C# List

You can use foreach to print elements in a List.

In this example, we will initialize a list, and print the elements of this list using foreach function.

Program.cs

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

namespace CSharpExamples {

    class Program {
        static void Main(string[] args) {
            var names = new List<string>() {"Mack", "Daisy", "Ward"};
            foreach( var name in names) {
                Console.WriteLine(name);
            }
        }
    }
}
```

Output

```
PS D:\workspace\csharp\HelloWorld> dotnet run
Mack
Daisy
Ward
```

Modify C# List

You can use index like in an array to modify the elements of a list.

In the following example, we will initialize a list with some elements, and then modify the element at index 1 with a new value.

Program.cs

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

namespace CSharpExamples {

    class Program {
        static void Main(string[] args) {
            var names = new List<string>() {"Mack", "Daisy", "Ward"};
            names[1] = "Fitz";
            foreach( var name in names) {
                Console.WriteLine(name);
            }
        }
    }
}
```

Output

```
PS D:\workspace\csharp\HelloWorld> dotnet run
Mack
Fitz
Ward
```

Remove an element from C# List using index

You can use `List.RemoveAt(int index)` to remove an element from List using index.

In the following example, we have list of elements, and we will delete the element present at index 1 using RemoveAt() method.

Program.cs

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

namespace CSharpExamples {

    class Program {
        static void Main(string[] args) {
            var names = new List<string>() {"Mack", "Daisy", "Ward"};
            names.RemoveAt(1);
            foreach( var name in names) {
                Console.WriteLine(name);
            }
        }
    }
}
```

Output

```
PS D:\workspace\csharp\HelloWorld> dotnet run
Mack
Ward
```

Remove an element from C# List using value

You can use `List.Remove(object)` to remove the first occurrence of element from List.

In the following example, we will remove the element from List based on value using Remove() method.

Program.cs

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

namespace CSharpExamples {

    class Program {
        static void Main(string[] args) {
            var names = new List<string>() {"Mack", "Daisy", "Ward"};
            names.Remove("Daisy");
            foreach( var name in names) {
                Console.WriteLine(name);
            }
        }
    }
}
```

Output

```
PS D:\workspace\csharp\HelloWorld> dotnet run
Mack
Ward
```

C# List Length

The property List.Count gives the number of elements in the List.

In the following list, we have a list of strings, and we will count the number of elements in this list using List.Count property.

Program.cs

```
using System;
```

```
using System.Collections.Generic;

namespace CSharpExamples {

    class Program {
        static void Main(string[] args) {
            var names = new List<string>() {"Mack", "Daisy", "Ward"};
            int listLength = names.Count;
            Console.WriteLine("Length of the list is : "+listLength);
        }
    }
}
```

Output

```
PS D:\workspace\csharp\HelloWorld> dotnet run
Length of the list is : 3
```

Conclusion

In this C# Tutorial, we learned about C# List, some of the useful methods and properties with well detailed examples.

C# Tutorial

- C# Tutorial
- C# Basic Example
- C# Comments
- C# Variables
- C# Constants
- C# if, if-else
- C# switch
- C# while loop
- C# for loop
- C# foreach
- C# break
- C# continue
- C# struct
- C# enum

- C# String
- C# Array
- C# Command Line Arguments

C# Console Operations

- C# Write to Console
- C# Read from Console

C# Object Oriented Programming Concepts

- C# Class & Constructors
- C# Encapsulation
- C# Polymorphism
- C# Method Overloading
- C# Interfaces

C# String Operations

- C# String Length
- C# Substring

C# File Operations

- C# Read Text File
- C# Write to File
- C# Delete File
- C# Copy File

C# Exception Handling

- C# try-catch
- C# finally
- C# throw
- C# Custom Exception
- C# SystemException
- C# DivideByZeroException
- C# NullReferenceException
- C# InvalidCastException

- C# IOException
- C# FieldAccessException

C# Collections

- **C# List**
- C# SortedList
- C# HashSet
- C# SortedSet
- C# Stack
- C# Queue
- C# LinkedList
- C# Dicionary
- C# SortedDictionary

C# Errors [Solved]

- C# Error: Class does not contain a constructor that takes arguments