# **C# - List<T>**

**List<T>** - це набір строго типізованих об'єктів, до яких можна отримати доступ за індексом і які мають методи для сортування, пошуку та модифікації списку.

**Характеристики List<T>:**

* **List<T>** еквівалент **ArrayList**, який реалізує **IList<T>**.
* Належить до простору імен **System.Collections.Generic.**
* **List<T>** може містити елементи заданого типу. Він забезпечує перевірку типу під час компіляції і не виконує блокування-розблокування, оскільки є узагальненим.
* Елементи можна додавати за допомогою методів **Add(), AddRange()** або синтаксису ініціалізатора колекції.
* Доступ до елементів можна отримати, передавши індекс, наприклад, **myList[0]**. Індекси починаються з нуля.
* **List<T>** працює швидше і менш схильний до помилок, ніж **ArrayList.**

**Створення List<T> (списку)**

List<T> є узагальненою колекцією, тому вам потрібно вказати параметр type для типу даних, які він може зберігати. У наступному прикладі показано, як створити список і додати елементи.

public class Program

{

public static void Main()

{

// adding elements using add() method

var primeNumbers = new List<int>();

primeNumbers.Add(1);

primeNumbers.Add(3);

primeNumbers.Add(5);

primeNumbers.Add(7);

Console.WriteLine("No of elelemts: "+ primeNumbers.Count);

var cities = new List<string>();

cities.Add("New York");

cities.Add("London");

cities.Add("Mumbai");

cities.Add("Chicago");

cities.Add(null); // null is allowed

Console.WriteLine("No of elelemts: " + cities.Count);

// adding elements using collection initializer syntax

var bigCities = new List<string>() {"New York", "London", "Mumbai", "Chicago"};

Console.WriteLine("No of elelemts: " + bigCities.Count);

}

}

У вищенаведеному прикладі **List<int> primeNumbers = new List<int>();** створює список типу **int**. Аналогічно, **cities** і **bigCities** є списками рядкового типу. Потім ви можете додавати елементи до списку за допомогою методу **Add()** або синтаксису ініціалізатора колекції.

Ви також можете додавати елементи користувацьких класів за допомогою синтаксису колекції-ініціалізатора. Нижче наведено додавання об'єктів класу **Student** до списку **List<Student>**.

var students = new List<Student>() {

new Student(){ Id = 1, Name="Bill"},

new Student(){ Id = 2, Name="Steve"},

new Student(){ Id = 3, Name="Ram"},

new Student(){ Id = 4, Name="Abdul"}

};

Console.WriteLine("No of elelemts: " + students.Count);

}

}

class Student{

public int Id { get; set; }

public string Name { get; set; }

}

**Додавання масиву в список(Adding an Array in a List)**

Метод **AddRange()** використовується для додавання усіх елементів масиву або іншої колекції до списку **List**.

Сигнатура **AddRange(): void AddRange(IEnumerable<T> collection)**

**Example: Add Arrays in List**

public class Program

{

public static void Main()

{

string[] cities = new string[3]{ "Mumbai", "London", "New York" };

var popularCities = new List<string>();

// adding an array in a List

popularCities.AddRange(cities);

var favouriteCities = new List<string>();

// adding a List

favouriteCities.AddRange(popularCities);

Console.WriteLine("Popular cities is: " + popularCities.Count);

Console.WriteLine("Favourite cities is: " + favouriteCities.Count);

}

}

**Доступ до списку**

Доступ до списку можна отримати за допомогою індексу, циклу **for/foreach** та за допомогою **LINQ**-запитів. Індекси списку починаються з нуля. Передайте індекс у квадратних дужках, щоб отримати доступ до окремих елементів списку, так само як і до масиву. Використовуйте цикл **foreach** або **for** для ітерації колекції **List<T>**.

using System;

using System.Collections.Generic;

public class Program

{

public static void Main()

{

List<int> intList = new List<int>() { 10, 20, 30, 40, 50 };

intList.ForEach(el => Console.WriteLine(el));

foreach (var el in intList)

Console.WriteLine(el);

for(int i =0; i < intList.Count; i++)

Console.WriteLine(intList[i]);

}

}

**Доступ до списку з допомогою LINQ**

**List<T>** реалізує інтерфейс **IEnumerable**. Отже, ми можемо запитувати список, використовуючи синтаксис запиту або методу **LINQ**, як показано нижче.

using System;

using System.Linq;

using System.Collections.Generic;

public class Program

{

public static void Main()

{

var students = new List<Student>() {

new Student(){ Id = 1, Name="Bill" },

new Student(){ Id = 2, Name="Steve" },

new Student(){ Id = 3, Name="Ram" },

new Student(){ Id = 4, Name="Abdul" },

new Student(){ Id = 5, Name="Bill" }

};

//get all students whose name is Bill

var studNames = from s in students

where s.Name == "Bill"

select s;

foreach(var student in studNames)

Console.WriteLine(student.Id + ", " +student.Name);

}

}

public class Student

{

public int Id { get; set; }

public string Name { get; set; }

}

**Вставка елементів у список**

Метод **Insert()** вставляє елемент у колекцію **List<T>** за заданим індексом.

Сигнатура методу **Insert():void Insert(int index, T item);**

using System;

using System.Collections.Generic;

public class Program

{

public static void Main()

{

var numbers = new List<int>(){ 10, 20, 30, 40 };

numbers.Insert(1, 11);// inserts 11 at 1st index: after 10.

foreach (var num in numbers)

Console.WriteLine(num);

}

}

**Видалення елементів зі списку**

Використовуйте метод **Remove()** для видалення першого входження заданого елементу в колекції **List<T>**. Метод **RemoveAt()** використовується для видалення елементу з заданого індексу. Якщо за вказаним індексом елементу немає, то буде згенеровано виключення **ArgumentOutOfRangeException.**

Сигнатура методу **Remove(): bool Remove(T item)**

Сигнатура **RemoveAt(): void RemoveAt(int index)**

using System;

using System.Collections.Generic;

public class Program

{

public static void Main()

{

var numbers = new List<int>(){ 10, 20, 30, 40, 10 };

numbers.Remove(10); // removes 10 elements from a list

numbers.RemoveAt(2); //removes the 3rd element (index starts from 0)

//numbers.RemoveAt(10); //removes the 3rd element (index starts from 0)

foreach (var num in numbers)

Console.WriteLine(num);

}

}

**Перевірка елементів у списку**

Використовуйте метод **Contains()**, щоб визначити, чи є елемент у списку **List<T>** чи ні.

using System;

using System.Collections.Generic;

public class Program

{

public static void Main()

{

var numbers = new List<int>(){ 10, 20, 30, 40 };

Console.WriteLine(numbers.Contains(10));

Console.WriteLine(numbers.Contains(11));

Console.WriteLine(numbers.Contains(20));

}

}

**List<T> Властивості та методи класу**

**(List<T> Class Properties and Methods)**

У наступній таблиці перераховано важливі властивості та методи класу List<T>:

| **Method** | **Usage** |
| --- | --- |
| **Add** | **Adds an element at the end of a List<T>.**  *Додає елемент в кінець списку List<T>.* |
| **AddRange** | **Adds elements of the specified collection at the end of a List<T>.**  Додає елементи заданої колекції в кінець списку List<T> |
| **BinarySearch** | **Search the element and returns an index of the element.**  *Шукає елемент і повертає його індекс.* |
| **Clear** | **Removes all the elements from a List<T>.**  *Видаляє всі елементи зі списку List<T>.* |
| **Contains** | **Checks whether the specified element exists or not in a List<T>.**  *Перевіряє, чи існує заданий елемент у списку List<T>.* |
| **Find** | **Finds the first element based on the specified predicate function.**  *Знаходить перший елемент на основі заданої предикатної функції.* |
| **Foreach** | **Iterates through a List<T>.**  *Ітерація по списку List<T>.* |
| **Insert** | **Inserts an element at the specified index in a List<T>.**  *Вставляє елемент за вказаним індексом у List<T>.* |
| **InsertRange** | **Inserts elements of another collection at the specified index.**  *Вставляє елементи іншої колекції за вказаним індексом.* |
| **Remove** | **Removes the first occurrence of the specified element.**  *Видаляє перше входження вказаного елемента.* |
| **RemoveAt** | **Removes the element at the specified index.**  *Видаляє елемент за вказаним індексом.* |
| **RemoveRange** | **Removes all the elements that match the supplied predicate function.**  *Видаляє всі елементи, які відповідають заданій функції-предикату.* |
| **Sort** | **Sorts all the elements.**  *Сортує всі елементи.* |
| **TrimExcess** | **Sets the capacity to the actual number of elements.**  *Встановлює ємність на фактичну кількість елементів.* |
| **TrueForAll** | **Determines whether every element in the List<T> matches the conditions defined by the specified predicate.**  *Визначає, чи кожен елемент у List<T> відповідає умовам, визначеним заданим предикатом.* |

| **Property** | **Usage** |
| --- | --- |
| **Items** | **Gets or sets the element at the specified index**  *Отримує або встановлює елемент за заданим індексом* |
| **Count** | **Returns the total number of elements exists in the List<T>**  *Повертає загальну кількість елементів, що існують в списку List<T>* |