Список List<T>

Класс List<T> из пространства имен `System.Collections.Generic` представляет простейший список однотипных объектов.

Среди его методов можно выделить следующие:

- **void Add(T item)**: добавление нового элемента в список
- **void AddRange(ICollection collection)**: добавление в список коллекции или массива
- **int BinarySearch(T item)**: бинарный поиск элемента в списке. Если элемент найден, то метод возвращает индекс этого элемента в коллекции. При этом список должен быть отсортирован.
- **int IndexOf(T item)**: возвращает индекс первого вхождения элемента в списке
- **void Insert(int index, T item)**: вставляет элемент item в списке на позицию index
- **bool Remove(T item)**: удаляет элемент item из списка, и если удаление прошло успешно, то возвращает true
- **void RemoveAt(int index)**: удаление элемента по указанному индексу index
- **void Sort()**: сортировка списка

Посмотрим реализацию списка на примере:

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

namespace Collections
{
    class Program
    {
        static void Main(string[] args)
        {
            List<int> numbers = new List<int>() { 1, 2, 3, 45 };
            numbers.Add(6); // добавление элемента

            numbers.AddRange(new int[] { 7, 8, 9 });

            numbers.Insert(0, 666); // вставляем на первое место в списке число 666

            numbers.RemoveAt(1); //  удаляем второй элемент

            foreach (int i  in numbers)
            {
                Console.WriteLine(i);
            }

            List<Person> people = new List<Person>(3);
            people.Add(new Person() { Name = "Том" });
            people.Add(new Person() { Name = "Билл" });

            foreach (Person p  in people)
            {
                Console.WriteLine(p.Name);
            }

            Console.ReadLine();
        }
    }

    class Person
```

```
33          {
34              public string Name { get; set; }
35          }
    }
```

Здесь у нас создаются два списка: один для объектов типа `int`, а другой - для объектов `Person`. В первом случае мы выполняем начальную инициализацию списка: `List<int> numbers = new List<int>() { 1, 2, 3, 45 };`

Во втором случае мы используем другой конструктор, в который передаем начальную емкость списка: `List<Person> people = new List<Person>(3);`. Указание начальной емкости списка (capacity) позволяет в будущем увеличить производительность и уменьшить издержки на выделение памяти при добавлении элементов. Также начальную емкость можно установить с помощью свойства `Capacity`, которое имеется у класса List.