**15 Обобщения**

**Задание №1:** Создайте класс MyList<T>. Реализуйте возможность использования его экземпляра аналогично экземпляру класса List<T>. Минимально требуемый интерфейс взаимодействия с экземпляром, должен включать метод добавления элемента, индексатор для получения значения элемента по указанному индексу и свойство только для чтения для получения общего количества элементов.

Листинг программы:

using System;

namespace MyListImplementation

{

class Program

{

static void Main(string[] args)

{

MyList<int> myList = new MyList<int>();

myList.Add(10);

myList.Add(20);

myList.Add(30);

Console.WriteLine($"Количество элементов: {myList.Count}");

Console.WriteLine($"Элемент с индексом 1: {myList[1]}");

try

{

Console.WriteLine(myList[5]);

}

catch (IndexOutOfRangeException ex)

{

Console.WriteLine(ex.Message);

}

}

}

public class MyList<T>

{

private T[] \_items;

private int \_size;

private const int DefaultCapacity = 4;

public MyList()

{

\_items = new T[DefaultCapacity];

\_size = 0;

}

public int Count => \_size;

public T this[int index]

{

get

{

if (index < 0 || index >= \_size)

{

throw new IndexOutOfRangeException("Индекс выходит за пределы списка");

}

return \_items[index];

}

}

public void Add(T item)

{

if (\_size == \_items.Length)

{

EnsureCapacity();

}

\_items[\_size] = item;

\_size++;

}

private void EnsureCapacity()

{

int newCapacity = \_items.Length == 0 ? DefaultCapacity : \_items.Length \* 2;

T[] newItems = new T[newCapacity];

Array.Copy(\_items, newItems, \_size);

\_items = newItems;

}

}

}

Анализ результатов:

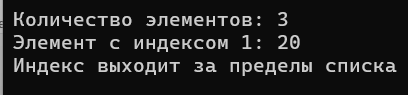


Рисунок 15.1 – Результат работы программы