**15 ОБОБЩЕННИЯ**

Задание 1. Создайте класс MyList&lt;T&gt;. Реализуйте возможность использования его экземпляра аналогично экземпляру класса List&lt;T&gt;. 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.

Листининг программы:

using System;

namespace task1

{

class MyList<T>

{

protected int index = 0;

protected T[] Values = new T[0];

public void Add(T value)

{

Array.Resize(ref Values, index+1);

Values[index] = value;

index++;

}

public T this[int index]

{

get => Values[index];

}

public int Length

{

get => Values.Length ;

}

}

class Program

{

static void Main(string[] args)

{

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

list.Add(32994);

list.Add(200002);

list.Add(101);

Console.WriteLine("Элемент с индексом 1: " + list[1]);

Console.WriteLine("Длина: " + list.Length);

}

}

}

Таблица 15.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| [32994, 200002, 101] | 200002, 3 |

Анализ результатов:



Рисунок 15.1 – Результат работы программы

Источник – собственная разработка

Задание 2. Создайте класс MyDictionary &lt;TKey, TValue&gt;. Реализуйте возможность использования его экземпляра аналогично экземпляру класса Dictionary.

Минимально требуемый интерфейс взаимодействия с экземпляром, должен включать метод добавления пар элементов, индексатор для получения значения элемента по указанному индексу и свойство только для чтения для получения общего количества пар элементов.

Листининг программы:

using System;

namespace task2

{

class MyDictionary<TKey,TValue>

{

protected int index = 0;

protected TKey[] Keys = new TKey[0];

protected TValue[] Values = new TValue[0];

public void Add(TKey key, TValue value)

{

if (Array.IndexOf(Keys, key) == -1)

{

Array.Resize(ref Keys, index + 1);

Keys[index] = key;

Array.Resize(ref Values, index + 1);

Values[index] = value;

index++;

}

else { Console.WriteLine("Значение с таким ключем уже существуюе"); }

}

public TValue this[TKey index]

{

get => Values[Array.IndexOf(Keys,index)];

}

public int Length

{

get => Values.Length;

}

}

class Program

{

static void Main(string[] args)

{

MyDictionary<string,int> list = new MyDictionary<string, int>();

list.Add("lol",32994);

list.Add("popit",200002);

list.Add("shazam",101);

list.Add("popit", 5345353);

Console.WriteLine("Элемент с ключем popit: " + list["popit"]);

Console.WriteLine("Количество пар: " + list.Length);

}

}

}

Таблица 15.2 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| [32994, 200002, 101] | Значение с таким ключом уже существует, 200002, 3 |

Анализ результатов:



Рисунок 15.2 – Результат работы программы

Источник – собственная разработка

Задание 3. Создайте расширяющий метод: public static T[] GetArray&lt;T&gt;(this MyList&lt;T&gt; list) Примените расширяющий метод к экземпляру типа MyList&lt;T&gt;, разработанному в задании 2 для данного урока. Выведите на экран значения элементов массива, который вернул расширяющий метод GetArray().

Листининг программы:

using System;

namespace task3

{

public class MyList<T>

{

public int index = 0;

public T[] Values = new T[0];

public void Add(T value)

{

Array.Resize(ref Values, index + 1);

Values[index] = value;

index++;

}

public T this[int index]

{

get => Values[index];

}

public int Length

{

get => Values.Length;

}

}

public static class MyListExtension

{

public static T[] GetArray<T>(this MyList<T> list)

{

for (int i = 0; i < list.Values.Length; i++)

{

Console.WriteLine(list.Values[i]);

}

return list.Values;

}

}

class Program

{

static void Main(string[] args)

{

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

list.Add(32994);

list.Add(200002);

list.Add(101);

Console.WriteLine("Элемент с индексом 1: " + list[1]);

Console.WriteLine("Длина: " + list.Length);

list.GetArray();

}

}

}

Таблица 15.3 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| [32994, 200002, 101] | 200002, 3, [32994, 200002, 101] |

Анализ результатов:

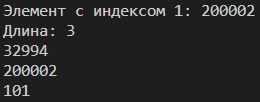


Рисунок 15.3 – Результат работы программы

Источник – собственная разработка