1. **Обобщения**

Задание 1. Используя Visual Studio, создайте проект по шаблону Console Application.

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.Collections;

namespace Task\_1

{

class Program

{

public static void Main(string[] args)

{

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

for (int i = 0; i < 6; i++)

{

Console.Write("Введите значение в массив: ");

list.Add(int.Parse(Console.ReadLine()));

}

Console.WriteLine("Введите индекс значения: ");

int index = int.Parse(Console.ReadLine());

Console.WriteLine($"Значение под индексом ({index}): {list[index]}");

Console.WriteLine($"Кол-во элементов: {list.Count}");

}

}

class MyList<T>

{

private ArrayList myList;

public T this[int index]

{

get { return (T)myList[index]; }

set { myList[index] = value; }

}

public MyList()

{

this.myList = new ArrayList();

}

public MyList(int count)

{

this.myList = new ArrayList(count);

}

public void Add(T item)

{

myList.Add(item);

}

public int Count

{

get { return myList.Count; }

}

}

}

Таблица 1.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 5 45 67 3 45 3 5 | 3 6 |

Анализ результатов:

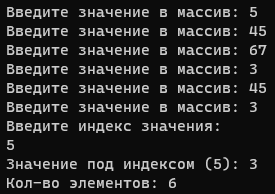


Рисунок 1.1 – Результат работы программы

Источник: собственная разработка

Задание 2. Используя Visual Studio, создайте проект по шаблону Console Application.

Создайте класс MyDictionary <TKey, TValue>. 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.

Листинг программы:

namespace Task\_2

{

class Program

{

public static void Main(string[] args)

{

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

for (int i = 0; i < 6; i++)

{

Console.Write("Введите номер зачетки: ");

int zach = int.Parse(Console.ReadLine());

Console.Write("Введите имя учащегося: ");

string name = Console.ReadLine();

diction.Add(zach, name);

}

Console.WriteLine("Введите номер зачетки: ");

int index = int.Parse(Console.ReadLine());

Console.WriteLine($"Зачеткой под номером ({index}) владеет: {diction[index]}");

Console.WriteLine($"Кол-во элементов в словаре: {diction.Counter}");

}

}

class MyDictionary<TKey, TVal>

{

private int counter = 0;

private TKey[] keysArray = null;

private TVal[] valsArray = null;

public int Counter

{

get { return this.counter; }

}

public void Add(TKey key, TVal val)

{

this.counter++;

Array.Resize(ref keysArray, counter);

keysArray[counter - 1] = key;

Array.Resize(ref valsArray, counter);

valsArray[counter - 1] = val;

}

public TVal this[TKey key]

{

get

{

int ind = 0;

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

{

if (key.Equals(keysArray[i]))

ind = i;

}

return valsArray[ind];

}

}

}

}

Таблица 2.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 2345 Степан 234 Толик 222 Андрей 9876 Слава 3456 Никита 435 Сева 234 | Толик 6 |

Анализ результатов:

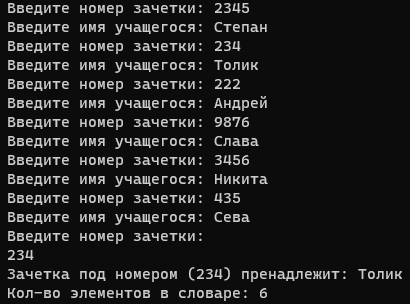


Рисунок 2.1 – Результат работы программы

Источник: собственная разработка

Задание 3. Используя Visual Studio, создайте проект по шаблону Console Application.

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

Листинг программы:

using Task\_1;

namespace Task\_3

{

class Program

{

public static void Main(string[] args)

{

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

for (int i = 0; i < 6; i++)

{

Console.Write("Введите значение в массив: ");

list.Add(int.Parse(Console.ReadLine()));

}

Console.WriteLine("Введите индекс значения: ");

int index = int.Parse(Console.ReadLine());

Console.WriteLine($"Значение под индексом ({index}): {list[index]}");

Console.WriteLine($"Кол-во элементов: {list.Count}");

Console.WriteLine("Список значений");

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

{

Console.Write(list.GetArray()[i] + " ");

}

}

}

static class MyListExtension

{

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

{

T[] arr = new T[list.Count];

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

arr[i] = list[i];

return arr;

}

}

}

Таблица 3.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 2 5 34 78 5 4 2 | 34 6 2 5 34 78 5 4 |

Анализ результатов:

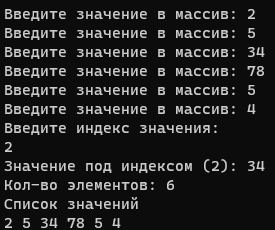


Рисунок 3.1 – Результат работы программы

Источник: собственная разработка