**15 Обобщения**

Задание №1. Используя Visual Studio, создайте проект по шаблону Console Application.

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.

Листинг программы:

try

{

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

list.Add(1);

list.Add(2);

Console.WriteLine($"Первый элемент списка: {list[0]}");

Console.WriteLine($"Общее количество элементов: {list.Capacity}");

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

class MyList<T>

{

List<T> list = new List<T>();

public T this[int i]

{

get { return list[i]; }

}

public int Capacity { get { return list.Count; } }

public void Add(T value)

{

list.Add(value);

}

}

Таблица 15.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Первый элемент списка: 1  Общее количество элементов: 2 |

Анализ результатов представлен на рисунке 15.1.



Рисунок 15.1 – Результат работы программы

Источник: собственная разработка

Задание №2. Используя Visual Studio, создайте проект по шаблону Console Application.

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.

Листинг программы:

try

{

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

dict.Add(0, "1");

dict.Add(1, "0");

Console.WriteLine($"Первый элемент словаря: {dict[0]}");

Console.WriteLine($"Общее количество элементов: {dict.Capacity}");

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

class MyDictionary<TKey, TValue>

{

Dictionary<TKey, TValue> dict = new Dictionary<TKey, TValue>();

public TValue this[TKey key]

{

get { return dict[key]; }

}

public int Capacity { get { return dict.Count; } }

public void Add(TKey key, TValue value)

{

dict.Add(key, value);

}

}

Таблица 15.2 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Первый элемент словаря: 1  Общее количество элементов: 2 |

Анализ результатов представлен на рисунке 15.2.



Рисунок 15.2 – 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:

try

{

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

list.Add(1);

list.Add(2);

int[] arr = list.GetArray();

Console.WriteLine($"Первый элемент списка: {list[0]}");

Console.WriteLine($"Общее количество элементов: {list.Capacity}");

foreach (int i in arr)

{

Console.Write(i + " ");

}

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

public class MyList<T>

{

public List<T> list = new List<T>();

public T this[int i]

{

get { return list[i]; }

}

public int Capacity { get { return list.Count; } }

public void Add(T value)

{

list.Add(value);

}

}

public static class GetArr

{

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

{

return list.list.ToArray();

}

}

Таблица 15.3 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Первый элемент списка: 1  Общее количество элементов: 2  1 2 |

Анализ результатов представлен на рисунке 15.3.



Рисунок 15.3 – Результат работы программы

Источник: собственная разработка