**15 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:

try

{

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

list.Add(5);

list.Add(9);

list.Add(18);

Console.WriteLine($"First list element: {list[0]}");

Console.WriteLine($"Total amount of elements: {list.TotalElements}");

}

catch (Exception e)

{

Console.WriteLine($"An error occured! {e.Message}");

}

class MyList<T>

{

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

public T this[int i]

{

get { return list[i]; }

}

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

public void Add(T a)

{

list.Add(a); }

}

Таблица 15.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 5, 9, 18 | 5, 3 |

Источник: собственная разработка

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 <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:

try

{

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

dict.Add(0, "10");

dict.Add(1, "15");

dict.Add(2, "20");

Console.WriteLine($"First dictionary element: {dict[0]}");

Console.WriteLine($"Total amount of elements: {dict.TotalAmount}");

}

catch (Exception e)

{

Console.WriteLine($"An error occured! {e.Message}");

}

class MyDictionary<TKey, TValue>

{

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

public TValue this[TKey key]

{

get { return dict[key]; }

}

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

public void Add(TKey key, TValue a)

{

dict.Add(key, a);

}

}

Таблица 15.2 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 10, 15, 20 | 10, 3 |

Источник: собственная разработка

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(5);

list.Add(9);

list.Add(18);

int[] array = list.GetArr();

Console.WriteLine($"First dictionary element: {list[0]}");

Console.WriteLine($"Total amount of elements: {list.TotalAmount}");

foreach (int i in array)

{

Console.WriteLine(i);

}

}

catch (Exception e)

{

Console.WriteLine($"An error occured! {e.Message}");

}

public class MyList<T>

{

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

public T this[int i]

{

get { return list[i]; }

}

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

public void Add(T a)

{

list.Add(a);

}

}

public static class GetArray

{

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

{

return list.list.ToArray();

}

}

Таблица 15.3 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 5, 9, 18 | 5, 3, 5, 9, 18 |

Источник: собственная разработка

Анализ результатов представлен на рисунке 15.3.

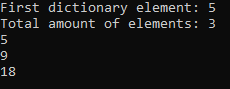


Рисунок 15.3 – Результат работы программы

Источник: собственная разработка