**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:

Program:

using z1.Model;

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

myList.Add(1);

myList.Add(2);

myList.Add(3);

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

{

Console.WriteLine(myList[i]);

}

Console.WriteLine($"Size: {myList.Length}");

Console.WriteLine();

Console.WriteLine("Challenge a static method: ");

WriteArray(MyList<int>.GetArray(myList));

void WriteArray<T>(T[] arr)

{

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

{

Console.WriteLine(arr[i]);

}

}

namespace z1.Model

{

internal class MyList<T>

{

private T[] arr;

public T this[int index]

{

get

{

if (arr != null)

{

if (0 <= index && index < arr.Length)

{

return arr[index];

}

}

return default(T);

}

}

public int Length

{

get

{

if (arr != null)

{

return arr.Length;

}

return 0;

}

}

public MyList(T[] arr)

{

this.arr = arr;

}

public T[] GetArray()

{

if (arr != null)

{

return arr;

}

return default(T[]);

}

public void Add(T obj)

{

if (arr != null)

{

Array.Resize(ref arr, arr.Length + 1);

}

else

{

arr = new T[1];

}

arr[arr.Length - 1] = obj;

}

}

}

Таблица 15.1 – Входные и выходные данные программы задание №1

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | 1  2  3  Size: 3 |

Анализ результатов:

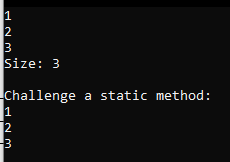


Рисунок 15.1 – Результат выполнения программы задание №1

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.

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:

Program:

using z2.Model;

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

myDictionary.Add(0, "Obj 1");

myDictionary.Add(1, "Obj 2");

myDictionary.Add(2, "Obj 3");

myDictionary.Add(1, "Obj 4");

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

{

Console.WriteLine(myDictionary[i]);

}

Console.WriteLine($"Size myDictionary {myDictionary.Length}");

MyDictionary:

namespace z2.Model

{

internal class MyDictionary<T, V>

{

private T[] keys;

private V[] values;

public V this[int index]

{

get

{

if (values != null)

{

if (0 <= index && index < keys.Length)

{

return values[index];

}

}

return default(V);

}

}

public int Length

{

get

{

if (keys != null)

{

return keys.Length;

}

return 0;

}

}

public void Add(T key, V value)

{

if (keys != null)

{

if (Array.IndexOf(keys,key) != -1)

{

return;

}

Array.Resize(ref keys, keys.Length + 1);

Array.Resize(ref values, values.Length + 1);

}

else

{

keys = new T[1];

values = new V[1];

}

keys[keys.Length - 1] = key;

values[values.Length - 1] = value;

}

}

}

Таблица 15.2 – Входные и выходные данные программы задание №2

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Obj 1  Obj 2  Obj 3  Size myDictionary 3 |

Анализ результатов:

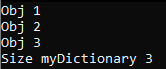


Рисунок 15.2 – Результат выполнения программы задание №2