**15 «СОБЫТИЯ».**

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;

class MyList<T>

{

private T[] items;

private int count;

public MyList()

{

items = new T[0];

count = 0;

}

public void Add(T item)

{

Array.Resize(ref items, count + 1);

items[count] = item;

count++;

}

public T this[int index]

{

get

{

if (index < 0 || index >= count)

{

throw new IndexOutOfRangeException();

}

return items[index];

}

}

public int Count

{

get { return count; }

}

}

class Program

{

static void Main(string[] args)

{

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

myList.Add(10);

myList.Add(20);

myList.Add(30);

Console.WriteLine("Elements in MyList:");

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

{

Console.WriteLine($"Element at index {i}: {myList[i]}");

}

}

}

Таблица 1.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Elements in MyList:  Element at index 0: 10  Element at index 1: 20  Element at index 2: 30 |

Анализ результатов:

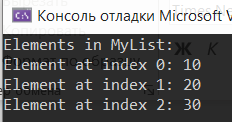


Рисунок 1.1 – Результат работы программы

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:

using System;

using System.Collections.Generic;

class MyDictionary<TKey, TValue>

{

private List<KeyValuePair<TKey, TValue>> items;

public MyDictionary()

{

items = new List<KeyValuePair<TKey, TValue>>();

}

public void Add(TKey key, TValue value)

{

items.Add(new KeyValuePair<TKey, TValue>(key, value));

}

public TValue this[TKey key]

{

get

{

foreach (var pair in items)

{

if (pair.Key.Equals(key))

{

return pair.Value;

}

}

throw new KeyNotFoundException();

}

}

public int Count

{

get { return items.Count; }

}

}

class Program

{

static void Main(string[] args)

{

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

myDictionary.Add("one", 1);

myDictionary.Add("two", 2);

myDictionary.Add("three", 3);

Console.WriteLine("Elements in MyDictionary:");

Console.WriteLine("Key: one, Value: " + myDictionary["one"]);

Console.WriteLine("Key: two, Value: " + myDictionary["two"]);

Console.WriteLine("Key: three, Value: " + myDictionary["three"]);

}

}

Таблица 1.3 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Elements in MyDictionary:  Key: one, Value: 1  Key: two, Value: 2  Key: three, Value: 3 |

Анализ результатов:

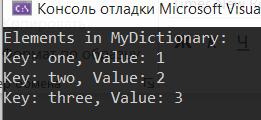


Рисунок 1.2 – Результат работы программы

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 System;

class MyList<T>

{

private T[] items;

private int count;

public MyList()

{

items = new T[0];

count = 0;

}

public void Add(T item)

{

Array.Resize(ref items, count + 1);

items[count] = item;

count++;

}

public T this[int index]

{

get

{

if (index < 0 || index >= count)

{

throw new IndexOutOfRangeException();

}

return items[index];

}

}

public int Count

{

get { return count; }

}

}

static class MyListExtensions

{

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

{

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

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

{

array[i] = list[i];

}

return array;

}

}

class Program

{

static void Main(string[] args)

{

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

myList.Add(10);

myList.Add(20);

myList.Add(30);

int[] array = myList.GetArray();

Console.WriteLine("Values in the array returned by GetArray():");

foreach (var item in array)

{

Console.WriteLine(item);

}

}

}

Таблица 1.3 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Values in the array returned by GetArray():  10  20  30 |

Анализ результатов:

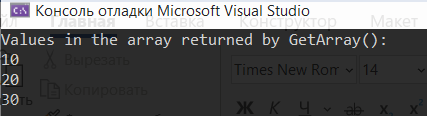


Рисунок 1.3 – Результат работы программы