Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Рязанский государственный радиотехнический университет

имени В.Ф.Уткина»

Кафедра «Электронные вычислительные машины»

Отчет по лабораторной работе №5

на тему

«Коллекции»

по дисциплине  
**«Визуальное программирование»**

Выполнили:

Студенты группы №140

Бригада 5

Сафаров Д. А.

Тимохин Е. С.

Проверили:  
ст. преп. Хизриева Н.И.

ст. преп. Бастрычкин А.С.

**Цель работы**

Изучить работу с коллекциями в языке C#.

**Задание**

Разработать класс DynamicArray<T>. Класс должен содержать:

1) Приватное поле для хранения элементов динамического массива T[ ];

2) Свойство Count – количество элементов в массиве;

3) Свойство Capacity – емкость массива;

4) Конструктор без параметров. Значение Capacity задается по умолчанию (например, 20);

5) Конструктор, принимающий значение Capacity;

6) Метод Add(T element) для вставки элемента в конец массива;

7) Метод Add(IEnumerable<T> elements) для добавления коллекции в конец массива;

8) Метод Insert(T element, int position) для вставки элемента по указанной позиции;

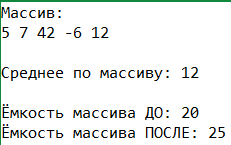
9) Метод RemoveAt(int position) для удаления элемента по указанной позиции;

10) Метод IncreaseCapacity(int n) для увеличения емкости массива на n элементов;

11) Метод IEnumerator<T> GetEnumerator() (реализовать интерфейс IEnumerable). Требуется самостоятельно реализовать интерфейс IEnumerator<T> (свойство Current, методы MoveNext и Reset) в отдельном классе.

**Практическая часть**

Код программы представлен в приложении А. Результат её работы представлен на рисунке 1.

Рисунок 1 – Результаты работы программы

Приложение А. Текст программы

**DinamicArray.cs**

using System.Collections;

namespace DinamicArrayNS

{

public class DinamicArray<T> : IEnumerable<T>

{

///\* СВОЙСТВА \*///

// Элементы массива

private T[] \_elements;

// Количество элементов

public int Count { get; private set; }

// Ёмкость массива

public uint Capacity { get; private set; }

// Шаг увеличения ёмкости

private uint \_capacityStep;

public uint CapacityStep

{

get

{

return \_capacityStep;

}

set

{

if (value <= 0) throw new Exception("Capacity step must be positive

number!");

\_capacityStep = value;

}

}

///\* КОНСТРУКТОРЫ \*///

public DinamicArray(uint capacity = 20)

{

this.Capacity = capacity;

\_elements = new T[Capacity];

}

///\* МЕТОДЫ \*///

// Вставка элемент в конец массива

public void Add(T element)

{

if (Count == Capacity)

IncreaseCapacity(\_capacityStep);

\_elements[Count++] = element;

}

// Добавление коллекции в конец массива

public void Add(IEnumerable<T> elements)

{

if (Count + elements.Count() >= Capacity)

IncreaseCapacity(Convert.ToUInt32(elements.Count()));

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

\_elements[Count+i] = elements.ElementAt(i);

Count += elements.Count();

}

// Вставка элемента на указанную позицию

public void Insert(T element, int position)

{

if (position < 0 || position >= Count) throw new Exception("Invalid insert

position!");

if(Count == Capacity)

IncreaseCapacity(\_capacityStep);

for (int i = Count; i > position; i--)

\_elements[i] = \_elements[i-1];

\_elements[position] = element;

Count++;

}

// Удаление элемента с указанной позиции

public void RemoveAt(int position)

{

if (position < 0 || position >= Count) throw new Exception("Invalid remove

position!");

for (int i = position; i < Count; i++)

\_elements[i] = \_elements[i+1];

Count--;

}

// Увеличение ёмкости на n элементов

public void IncreaseCapacity(uint n)

{

if (n == 0) return;

Capacity += n;

T[] newElements = new T[Capacity];

for (uint i = 0; i < Count; i++)

newElements[i] = \_elements[i];

\_elements = newElements;

}

public IEnumerator<T> GetEnumerator()

{

return new DAEnumerator<T>(\_elements, Count);

}

IEnumerator IEnumerable.GetEnumerator() => this.GetEnumerator();

}

}

**DAEnumerator.cs**

using System.Collections;

namespace DinamicArrayNS

{

class DAEnumerator<T> : IEnumerator<T>

{

private T[] \_elements; // Элементы массива

private int count; // Доступное количество элементов

private int position = -1;

public DAEnumerator(T[] elements, int count)

{

this.\_elements = elements;

this.count = count;

}

// Свойство получения нынешнего элемента

public T Current

{

get

{

if (position == -1 || position >= count) throw new

Exception("Invalid enumeration position!");

return \_elements[position];

}

}

object IEnumerator.Current => this.Current;

public bool MoveNext()

{

return (++position < count);

}

public void Reset()

{

position = -1;

}

public void Dispose()

{

}

}

}

**Program.cs**

using DinamicArrayNS;

DinamicArray<int> dinamicArray = [5,7];

dinamicArray.Add(42);

dinamicArray.Add([0, 12]);

dinamicArray.Insert(-6, 3);

dinamicArray.RemoveAt(4);

Console.WriteLine("Массив:");

foreach (var item in dinamicArray)

{

Console.Write($"{item.ToString()} ");

}

Console.WriteLine($"\n\nСреднее по массиву: {dinamicArray.Average()}\n");

Console.WriteLine($"Ёмкость массива ДО: {dinamicArray.Capacity}");

dinamicArray.IncreaseCapacity(5);

Console.WriteLine($"Ёмкость массива ПОСЛЕ: {dinamicArray.Capacity}");