МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждения высшего образования
«Дагестанский государственный технический университет»
Факультет компьютерных технологий, вычислительной техники и энергетики.
Кафедра «ПОВТиАС»

ОТЧЕТ

к лабораторной работе №3
на тему: «Массивы и динамические массивы»
По дисциплине: «**Типы структур данных**»

Выполнил студент 2-го курса
Гр. «У-033»: Магомедов А.С..
Проверил: ст.преп. Тетакаев У.Р.

Махачкала 2021

Условие задачи:

Сгенерировать список, состоящий из натуральных чисел от 0 до 9 включительно, количество элементов в списке является количеством символов ФИО студента. С полученным списком получить следующие действия:

1)Отсортировать список от начала до середины по возрастанию, а от середины до конца по убыванию.

2)Отсортировать список тремя различными алгоритмами и сравнить количество операций.

Листинг 1. Код программы:

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace Orange
{
   class Program
   {
      //Объявим коллекцию целых чисел
      public static List<Int32> collection { get; set; }
      //Объявляем переменные для сохранения количества интераций
      public static int interationSort { get; set; }
      public static int interationSortLast { get; set; }
      public static int interationDescending { get; set; }
      static void Main(string[] args)
      {
         //Вводим данные студента
         Console.Write("Введите своё ФИО полностью: ");
         string name = Console.ReadLine();
         Console.Write("Введите от: ");
         int indexAt = int.Parse(Console.ReadLine());
         Console.Write("Введите до: ");
         int indexEnd = int.Parse(Console.ReadLine());

         collection = GenerationList(collection, indexAt, indexEnd, name);
         Console.WriteLine("Количество символов: " + name.Length);
```

```
        foreach (var item in collection)
        {
            Console.WriteLine(item);
        }
        Console.WriteLine("Отсортирован по возрастанию: ");
        foreach (var item in Sort(collection))
        {
            Console.WriteLine(item);
        }
        Console.WriteLine("Отсортирован по возрастанию: ");
        foreach (var item in SortLast(collection))
        {
            Console.WriteLine(item);
        }
        Console.WriteLine("Отсортирован по убыванию: ");
        foreach (var item in Descending(collection))
        {
            Console.WriteLine(item);
        }
        Console.WriteLine("Iteration Sort: " + interationSort);
        Console.WriteLine("Iteration Sort Last: " + interationSortLast);
        Console.WriteLine("Iteration Descending: " + interationDescending);
    }

    // Написал метод сортировки первой половины списка по возрастанию с помощью средств LINQ
    public static List<Int32> Sort(List<Int32> collection)
    {
        // Вычисляем центр
        int center = collection.Count() / 2;
        // Для удобства создаю новый список чисел, чтобы отдельно зафиксировать первую половину
сгенерированного списка
        List<Int32> collectionFirst = new List<Int32>();
        int item = 0;
        // Прохожу циклом, чтобы сохранить первую половину сгенерированного списка и сохранить в новый
отдельный список
        for (int i = 0; i < center; i++)
        {
            item = collection[i];
            collectionFirst.Add(item);
            interationSort += i;
        }
```

```
        //Новый список сортируем средствами LINQ
        collectionFirst.Sort();
        //Возвращаю количество итераций
        return collectionFirst;
    }

    //Написал метод сортировки первой половины списка по возрастанию с помощью сортировки Пузрьком
    public static List<Int32> SortLast(List<Int32> collection)
    {
        //Вычисляем центр
        int center = collection.Count() / 2;
        //Для удобства создаю новый список чисел, чтобы отдельно зафиксировать первую половину
сгенерированного списка
        List<Int32> collectionFirst = new List<Int32>();
        int item = 0;
        //Прохожу циклом, чтобы сохранить первую половину сгенерированного списка и сохранить в новый
отдельный список
        for (int i = 0; i < center; i++)
        {
            item = collection[i];
            collectionFirst.Add(item);
        }
        //Новый список сортируем методом сортировки Пузырька
        for (int i = 0; i < collectionFirst.Count; i++)
        {
            for (int j = i + 1; j < collectionFirst.Count; j++)
            {
                if (collectionFirst[i] > collectionFirst[j])
                {
                    int a = collectionFirst[i];
                    collectionFirst[i] = collectionFirst[j];
                    collectionFirst[j] = a;
                }
                //Сохраняю количество итераций
                interationSortLast += i + j;
            }
        }
        //Возвращаю отсортированный список
        return collectionFirst;
    }

    //Написал метод сортировки второй половины списка по убыванию с помощью сортировки Пузрьком
```

```
        public static List<Int32> Descending(List<Int32> collection)
        {
            //Вычисляем центр
            int center = collection.Count() / 2;
            //Для удобства создаю новый список чисел, чтобы отдельно зафиксировать вторую половину сгенерированного списка
            List<Int32> collectionFirst = new List<Int32>();
            int item = 0;
            //Прохожу циклом, чтобы сохранить вторую половину сгенерированного списка и сохранить в новый отдельный список
            for (int i = center; i < collection.Count(); i++)
            {
                item = collection[i];
                collectionFirst.Add(item);
            }
            //Сортировка позырьком
            for (int i = 0; i < collectionFirst.Count; i++)
            {
                for (int j = i + 1; j < collectionFirst.Count; j++)
                {
                    if (collectionFirst[i] < collectionFirst[j])
                    {
                        int element = collectionFirst[i];
                        collectionFirst[i] = collectionFirst[j];
                        collectionFirst[j] = element;
                    }
                    //Сохраняю количество итераций
                    interationDescending += i + j;
                }

            }
            //Возвращаю отсортированный список
            return collectionFirst;
        }
        //Написал метод генерации случайных чисел
        public static List<Int32> GenerationList(List<Int32> collection, int indexAt, int indexEnd, string name)
        {
            //Инициализирую ранее объявленный список чисел
            collection = new List<Int32>();
            int item = 0;
            //Обращаюсь к классу Random
            Random random = new Random();
```

```
        // Перебираю список и заполняю рандомными числами в указанном пользователем диапазоне чисел
        for (int i = 0; i < name.Length; i++)
        {
            item = random.Next(indexAt, indexEnd);
            collection.Add(item);
        }
        // Возвращаб сгенерированный список
        return collection;
    }
  }
}
```

Ввод данных и генерация листа целых чисел:

```
thevckit@MacBook-Pro-VcKit Orange % dotnet run
Введите своё ФИО полностью: Magomedov Abdulkhakim Salimsoltanovich
Введите от: -100
Введите до: 100
Количество символов: 38
-55
-51
-78
32
-21
50
-95
-12
36
0
67
-42
82
73
-60
30
23
-37
37
-74
26
-3
-12
-33
-26
4
67
73
20
48
25
-8
-40
82
79
29
5
66
```

Вывод:

Отсортирован по возрастанию с помощью средств Linq

```
Отсортирован по возрастанию:
-95
-78
-60
-55
-51
-42
-37
-21
-12
0
23
30
32
36
37
50
67
73
82
```

Вывод:

Отсортирован по возрастанию с помощью сортировки «Пузырьком»

```
Отсортирован по возрастанию:
-95
-78
-60
-55
-51
-42
-37
-21
-12
0
23
30
32
36
37
50
67
73
82
```

Вывод:

Отсортирован по убыванию сортировкой «Пузырьком»

```
Отсортирован по убыванию:
82
79
73
67
66
48
29
26
25
20
5
4
-3
-8
-12
-26
-33
-40
-74
```

Количество операций:

```
Iteration Sort: 171
Iteration Sort Last: 3078
Iteration Descending: 3078
```