Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ

Задание 3 - Алгоритмы сортировки массивов

тема

|  |  |  |
| --- | --- | --- |
| Преподаватель |  | Р.Ю. Царев |
|  | подпись, дата | инициалы, фамилия |
| Студент КИ18-17/1б 031831229 |  | В.А. Прекель |
| номер группы, зачетной книжки | подпись, дата | инициалы, фамилия |

Красноярск 2020

# 1 Цель работы с постановкой задачи

## 1.1 Цель работы

Алгоритмы сортировки массивов.

## 1.2 Задача работы

Реализовать в программе два алгоритма (по выбору студента) из указанных ниже:

а) сортировка с помощью прямого включения,

б) сортировка с помощью прямого выбора,

в) сортировка с помощью прямого обмена

Сравнить эффективность реализованных алгоритмов по числу перестановок и количеству сравнений.

Требования к выполнению лабораторной работы:

1. Строгое соответствие программы и результатов ее работы с полученным заданием.

2. Самостоятельные тестирование и отладка программы.

3. Устойчивость работы программы при любых воздействиях, задаваемых пользователем через интерфейс программы.

4. Предоставление демонстрационного примера и исходного текста программы для защиты.

5. Предоставление отчета по практическому заданию, содержащего описание реализованного алгоритма, программы, результатов работы программы (отчет необходимо загрузить на сайт курса).

# 2 2 Описание реализованного алгоритма

Реализован алгоритм сортировки с помощью прямого включения, алгоритм сортировки с помощью прямого выбора. Подсчитывается число сравнений и число присваиваний. Реализовано на языке C# и поддерживает различные сравнимые типы данных, а также выбор в каком порядке сортировать. Написаны юнит-тесты для различных типов данных используя фреймворк NUnit.

# 3 Описание программы (листинги кода)

Листинг 1 – Alg\_03/Alg\_03.Core/AbstractSort.cs (абстрактный класс для сортировок)

using System;  
using System.Collections.Generic;  
  
namespace Alg\_03.Core  
{  
 public abstract class AbstractSort<T>  
 where T : IComparable  
 {  
 public enum SortOrder  
 {  
 Ascending = 1,  
 Descending = -1  
 }  
  
 public int AssignmentCount { get; protected set; }  
 public int CompareCount { get; private set; }  
  
 public IList<T> List { get; private set; } = new List<T>();  
  
 public SortOrder Order { get; set; } = SortOrder.Ascending;  
  
 protected int Compare(T a, T b)  
 {  
 CompareCount++;  
 return (int) Order \* a.CompareTo(b);  
 }  
  
 public virtual void Sort(IList<T> list)  
 {  
 List = list;  
 AssignmentCount = 0;  
 CompareCount = 0;  
 }  
 }  
}

Листинг 2 – Alg\_03/Alg\_03.Core/InclusionSort.cs (сортировка с помощью прямого включения)

using System;  
using System.Collections.Generic;  
  
namespace Alg\_03.Core  
{  
 public class InclusionSort<T> : AbstractSort<T>  
 where T : IComparable  
 {  
 public override void Sort(IList<T> list)  
 {  
 base.Sort(list);  
 for (var i = 1; i < List.Count; i++)  
 {  
 var value = List[i];  
 var j = i;  
 while (j > 0 && Compare(List[j - 1], value) > 0)  
 {  
 AssignmentCount++;  
 List[j] = List[j - 1];  
 j--;  
 }  
  
 AssignmentCount++;  
 List[j] = value;  
 }  
 }  
 }  
}

Листинг 3 – Alg\_03/Alg\_03.Core/SelectionSort.cs (сортировка с помощью прямого выбора)

using System;  
using System.Collections.Generic;  
  
namespace Alg\_03.Core  
{  
 public class SelectionSort<T> : AbstractSort<T>  
 where T : IComparable  
 {  
 public override void Sort(IList<T> list)  
 {  
 base.Sort(list);  
 for (var i = 0; i < List.Count - 1; i++)  
 {  
 var min = i;  
  
 for (var j = i + 1; j < List.Count; j++)  
 {  
 if (Compare(List[j], List[min]) < 0)  
 {  
 min = j;  
 }  
 }  
  
 AssignmentCount += 2;  
 var temp = List[i];  
 List[i] = List[min];  
 List[min] = temp;  
 }  
 }  
 }  
}

Листинг 4 – Alg\_03/Alg\_03.Core.Tests/TheoryGenericSortTests.cs (юнит-тесты)

using System;  
using System.Collections.Generic;  
using System.Linq;  
  
using NUnit.Framework;  
  
namespace Alg\_03.Core.Tests  
{  
 internal class Comparable : IComparable  
 {  
 private static readonly Random Random = new Random();  
 private int Value { get; } = Random.Next();  
  
 public int CompareTo(object? obj) => obj == null ? 1 : Value.CompareTo(((Comparable) obj).Value);  
 }  
  
 [TestFixture(typeof(InclusionSort<int>), typeof(int))]  
 [TestFixture(typeof(SelectionSort<int>), typeof(int))]  
 [TestFixture(typeof(InclusionSort<double>), typeof(double))]  
 [TestFixture(typeof(SelectionSort<double>), typeof(double))]  
 [TestFixture(typeof(InclusionSort<string>), typeof(string))]  
 [TestFixture(typeof(SelectionSort<string>), typeof(string))]  
 [TestFixture(typeof(InclusionSort<DateTime>), typeof(DateTime))]  
 [TestFixture(typeof(SelectionSort<DateTime>), typeof(DateTime))]  
 [TestFixture(typeof(InclusionSort<Guid>), typeof(Guid))]  
 [TestFixture(typeof(SelectionSort<Guid>), typeof(Guid))]  
 [TestFixture(typeof(InclusionSort<Comparable>), typeof(Comparable))]  
 [TestFixture(typeof(SelectionSort<Comparable>), typeof(Comparable))]  
 public class TheoryGenericSortTests<TSort, T>  
 where TSort : AbstractSort<T>, new()  
 where T : IComparable  
 {  
 private TSort Sort { get; } = new TSort();  
  
 [Datapoint]  
 private List<double> \_arrayDouble1 = new List<double>(new[] {1.2, 3.4, 1.2, 3.4});  
  
 [Datapoint]  
 private List<double> \_arrayDouble2 = new List<double>(new[] {5.6, 7.8, 1.2, 3.4});  
  
 [Datapoint]  
 private List<int> \_arrayInt = new List<int>(new[] {0, 1, 5, 3});  
  
 [Datapoint]  
 private List<int> \_arrayInt1 = new List<int>(new[] {1, 3, 4, 34, 5, 6, 2, 33, 2});  
  
 [Datapoint]  
 private List<string> \_arrayString1 =  
 new List<string>(new[] {"gj", "hjhk", "ukft", "re", "aaa", "zzz", "hj", "fthf", "abcde"});  
  
 [Datapoint]  
 private List<string> \_arrayString2 =  
 new List<string>(new[] {"z", "x", "c"});  
  
 [Datapoint]  
 private List<DateTime> \_arrayDateTime1 =  
 new List<DateTime>(new[] {DateTime.Now, DateTime.Today, DateTime.MaxValue});  
  
 [Datapoint]  
 private List<DateTime> \_arrayDateTime2 =  
 new List<DateTime>(new[]  
 {  
 new DateTime(123, DateTimeKind.Utc),  
 new DateTime(214324, DateTimeKind.Utc),  
 new DateTime(325235235235, DateTimeKind.Utc),  
 new DateTime(433344, DateTimeKind.Utc),  
 new DateTime(0, DateTimeKind.Utc)  
 });  
  
 [Datapoint]  
 private List<DateTime> \_arrayDateTime3 =  
 new List<DateTime>(new[]  
 {  
 DateTime.Now,  
 new DateTime(2020, 05, 14),  
 new DateTime(2025, 05, 14)  
 });  
  
 [Datapoint]  
 private List<Guid> \_listGuid1 = new List<Guid>(new[]  
 {  
 Guid.NewGuid(), Guid.NewGuid(), Guid.NewGuid(), Guid.NewGuid(), Guid.NewGuid()  
 });  
  
 [Datapoint]  
 private List<Comparable> \_listComparable1 =  
 new List<Comparable>(Enumerable.Range(0, 100)  
 .Select(p => new Comparable())  
 );  
  
 [Theory]  
 public void ListSortTest(List<T> list)  
 {  
 Sort.Sort(list);  
 Assert.That(list.OrderByDescending(p => p).SequenceEqual(list), Is.False);  
 Assert.That(list.OrderBy(p => p).SequenceEqual(list), Is.True);  
 }  
  
 [Theory]  
 public void ListSortTestDescending(List<T> list)  
 {  
 Sort.Order = AbstractSort<T>.SortOrder.Descending;  
 Sort.Sort(list);  
 Assert.That(list.OrderByDescending(p => p).SequenceEqual(list), Is.True);  
 Assert.That(list.OrderBy(p => p).SequenceEqual(list), Is.False);  
 }  
 }  
}

Остальной код доступен в репозитории по адресу <https://github.com/prekel/AlgorithmsAndDataStructures/tree/master/Alg_03>.

# 4 Результаты работы программы

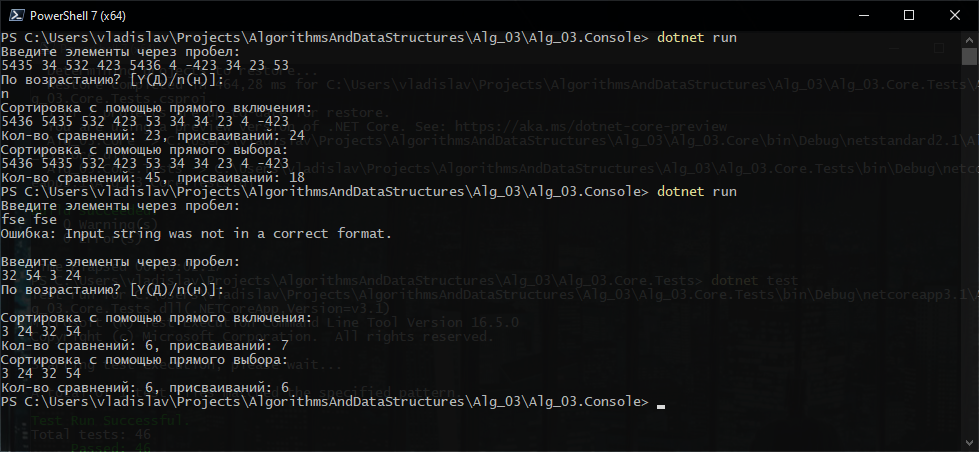


Рисунок 1 – Запуск программы



Рисунок 2 – Запуск тестов