

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

**«Дальневосточный федеральный университет»**

(ДВФУ)

**институт математики и компьютерных технологий**

**Департамент информационных и компьютерных систем**

**ОТЧЕТ**

по лабораторной работе №1

по дисциплине «Алгоритмы и структуры данных»

по теме «Внутренние сортировки»

**направление подготовки**

**09.03.03 Прикладная информатика**

**Прикладная информатика в экономике**

|  |  |
| --- | --- |
| Выполнил студент гр. Б9121 / 4 | |
|  | Туровец В. Ю. |
|  | |
| Проверил ст. преподаватель | |
|  | Елсукова Е. А. |
|  | |
| не зачтено/зачтено | |

г. Владивосток

2022г.

Оглавление

[Условие задачи 3](#_Toc125459412)

[Функциональное описание 4](#_Toc125459413)

[Входные данные 4](#_Toc125459414)

[Структура программы 5](#_Toc125459415)

[Интерфейс программы 8](#_Toc125459416)

[Тестовые примеры 8](#_Toc125459417)

[Полученные результаты 11](#_Toc125459418)

[Приложение 12](#_Toc125459419)

Условие задачи

Освоение основных методов упорядочивания данных, расположенных в оперативной памяти.

1. Составить программу, состоящую из следующих пунктов:
2. Сортировка простыми вставками;
3. Сортировка простым обменом;
4. Сортировка простым выбором;
5. Сортировка простым обменом – Расческа;
6. Быстрая сортировка.
7. Характеристики сортировок;
8. Выход.

ВХОДНЫЕ ДАННЫЕ: чтение данных осуществляется из файла.

Выбор типа данных:

1. массив записей; (количество 20) структура:   
   Название альбома, Исполнитель, Год выпуска (поле сортировки).
2. массив чисел (количество 500, 1000, 5000).

ВЫХОДНЫЕ ДАННЫЕ: В пп. 1–5 результат выводится на экран.

В п. 6 результат записывается в файл и считывается из него на экран в виде таблицы 1.

Таблица1 – Пример результирующей таблицы

|  |  |  |  |
| --- | --- | --- | --- |
| **количество сравнений** | **количество перестановок** | **время выполнения** | **вид сортировки** |

Функциональное описание

Входные данные

Для массива чисел. Файл со случайными числовыми значениями.

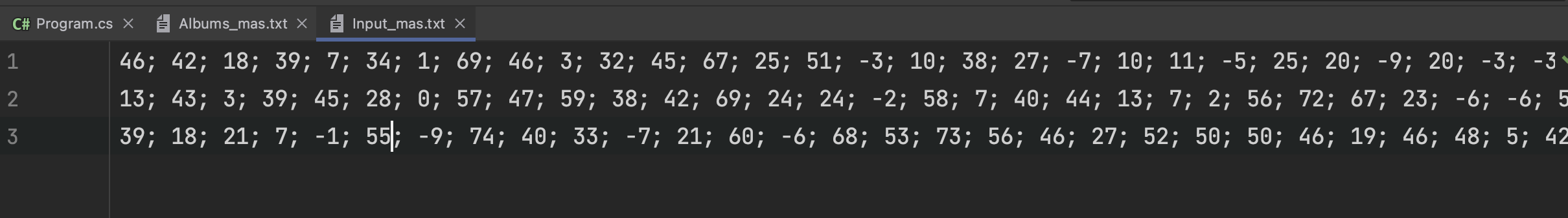


Рисунок 1 – Файл массива чисел

private static int[] Texter(int ask)*//делает готовый массив чисел на основе текста из файла*{  
 const string **path** = "/Users/vlad/Desktop/ConsoleApp1/ConsoleApp1/Input\_mas.txt";  
 string readText = File.ReadLines(**path**).ElementAtOrDefault(ask - 1);  
 *//Console.WriteLine(readText);* readText = readText.Replace("; ", " "); *//замена* var Text = readText.Split(' ').Select(int.Parse).ToArray();  
 return Text;  
}

Приложение 1 – Метод создания массива чисел

Для массива записей.



Рисунок 2 – Файл массива записей

*// -название альбома;  
// -исполнитель;  
// -год выхода. (поле сортировки)*private static (List<string>, List<string>, List<int>) Texter()*//делает готовые массивы дат, исполнителей и названий на основе текста из файла*{  
 var listOfAlbums = new List<string>();  
 var listOfNames = new List<string>();  
 var listOfyears = new List<int>();  
  
 const string **path** = "/Users/vlad/Desktop/ConsoleApp1/ConsoleApp1/Albums\_mas.txt";  
 for (int i = 0; File.ReadLines(**path**).ElementAtOrDefault(i) != null; i++)  
 {  
 string? readText = File.ReadLines(**path**).ElementAtOrDefault(i);  
 readText = readText.Replace("; ", ";"); *//замена* string[] text = (readText.Split(';').ToArray());  
 int j = 0;  
 foreach (var part in text)  
 {  
 j++;  
 *//Console.WriteLine(j);* if (j == 1){listOfAlbums.Add(part);}  
 if (j == 2){listOfNames.Add(part);}  
 if (j == 3){int.TryParse(part, out var year);listOfyears.Add(year);}  
   
 }  
 }  
   
 return (listOfAlbums, listOfNames, listOfyears);  
}

Приложение 2 – Инициализация массива записей

var listOfAlbums = new List<string>(); *//Названия всех альбомов*var listOfNames = new List<string>(); *//Имена всех исполнителей*var listOfyears = new List<int>(); *//Года выпуска альбомов*(listOfAlbums, listOfNames, listOfyears) = Texter();  
  
listOfyears = QuickSort(listOfyears, 0, listOfyears.Count - 1);  
  
List<int> QuickSort(List<int> array, int minIndex, int maxIndex)  
{  
 if (minIndex >= maxIndex)  
 {  
 return array;  
 }  
  
 int pivotIndex = GetPivotIndex(array, minIndex, maxIndex);  
  
 QuickSort(array, minIndex, pivotIndex - 1);  
  
 QuickSort(array, pivotIndex + 1, maxIndex);  
  
 return array;  
}  
  
int GetPivotIndex(List<int> array, int minIndex, int maxIndex)  
{  
 int pivot = minIndex - 1;  
  
 for (int i = minIndex; i <= maxIndex; i++)  
 {  
 if (array[i] < array[maxIndex])  
 {  
 pivot++;  
 (listOfyears[pivot], listOfyears[i]) = (listOfyears[i], listOfyears[pivot]); *//Замена местами годов* (listOfNames[pivot], listOfNames[i]) = (listOfNames[i], listOfNames[pivot]); *//Замена местами названий альбовом* (listOfAlbums[pivot], listOfAlbums[i]) = (listOfAlbums[i], listOfAlbums[pivot]); *//Замена местами имён исполнителей* }  
 }  
  
 pivot++;  
 *//меняем значения местами* (listOfyears[pivot], listOfyears[maxIndex]) = (listOfyears[maxIndex], listOfyears[pivot]); *//Замена местами годов* (listOfNames[pivot], listOfNames[maxIndex]) = (listOfNames[maxIndex], listOfNames[pivot]); *//Замена местами названий альбовом* (listOfAlbums[pivot], listOfAlbums[maxIndex]) = (listOfAlbums[maxIndex], listOfAlbums[pivot]); *//Замена местами имён исполнителей* return pivot;  
}

Приложение 3 – Сортировка миссива записей

Структура программы

Программа реализована на c#, все сортировки выполнены в виде функций:

1. Сортировка простыми вставками

internal static void EasyInserts(int[] array)

1. Сортировка простым обменом

internal static void Bubble(int[] array)

1. Сортировка просты выбором

internal static void SelectionSort(int[] array)

1. Сортировка вставками с бинарным поиском – Сортировка бинарными вставками

internal static void BinInsertSort(int[] array)

1. Быстрая сортировка

internal static void QuickSort(int[] array)

public static void Swap(ref int e1, ref int e2) {(e1, e2) = (e2, e1);}

Приложение 4 – замена элементов местами

for (int i = 0; i < array.Length; i++)  
{  
 var key = array[i];  
 var lo = 0;  
 var hi = i - 1;  
   
 while (lo < hi)  
 {  
   
 int mid = lo + (hi - lo) / 2;  
 if (key < array[i]) { hi = mid; }  
 else { lo = mid + 1; }  
 }  
   
 for (int j = i; i < j && j < lo + 1; j--)  
 {  
 comparisons++; array[j] = array[j - 1]; }  
  
 array[lo] = key;  
}

Приложение 5 – Реализация сортировки бинарными вставками(массив чисел).

static int[] QuickSort(int[] array)  
{

array = QuickSort(array, 0, array.Length - 1);   
 *//Для простоты метод получает на вход границы участка массива от минимального до максимального* int[] QuickSort(int[] quickSortingArray, int leftIndex, int rightIndex)   
 {  
 if (leftIndex >= rightIndex){return quickSortingArray;} *//Если левая граница правее правой - возвращаем масив* int pivotIndex = GetPivotIndex(quickSortingArray, leftIndex, rightIndex);*//Стержневой индекс   
  
 //Делим масив на две часты (слева и справа от стержня)  
 //Соритруем слева от стержня* QuickSort(quickSortingArray, leftIndex, pivotIndex - 1);  
 *//Соритруем справа от стержня* QuickSort(quickSortingArray, pivotIndex + 1, rightIndex);  
  
 return quickSortingArray;  
 }  
  
 int GetPivotIndex(int[] pivotingArray, int leftIndex, int rightIndex)*//Стержневой индекс* {  
 int pivot = leftIndex - 1;  
  
 for (int i = leftIndex; i <= rightIndex; i++)  
 {  
 if (pivotingArray[i] < pivotingArray[rightIndex])  
 {  
 pivot++;  
 Swap(ref pivotingArray[pivot], ref pivotingArray[i]);   
 }  
 }  
  
 pivot++;  
 Swap(ref pivotingArray[pivot], ref pivotingArray[rightIndex]);  
  
 return pivot;  
 }  
  
 return array;  
}

Приложение 6 –Реализация Быстрой сортировки (массив чисел)

1. Характеристики

Все характеристики выводятся сразу после сортировок (см. рисунок 3).

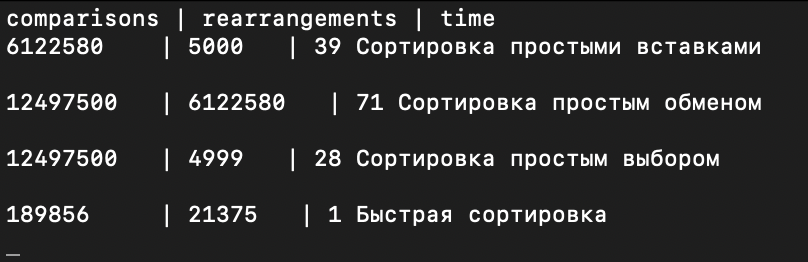


Рисунок 3 –Характеристики после сортировки 5000 элементов

Интерфейс программы

Интерфейс состоит из двух уровней на первом можно выбрать тип (см. рисунок 4), а на втором размер массива (см. рисунок 5).

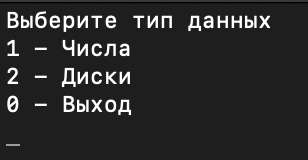


Рисунок 4 – Первый уровень интерфейса

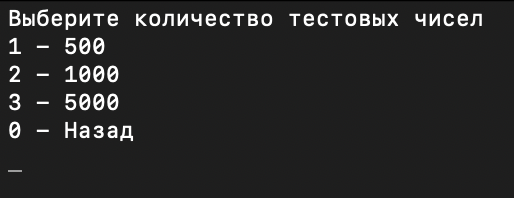


Рисунок 5 – Второй уровень интерфейса

Тестовые примеры

Приведены результаты функционала приложения.



Рисунок 6 – Результат Быстрой сортировки

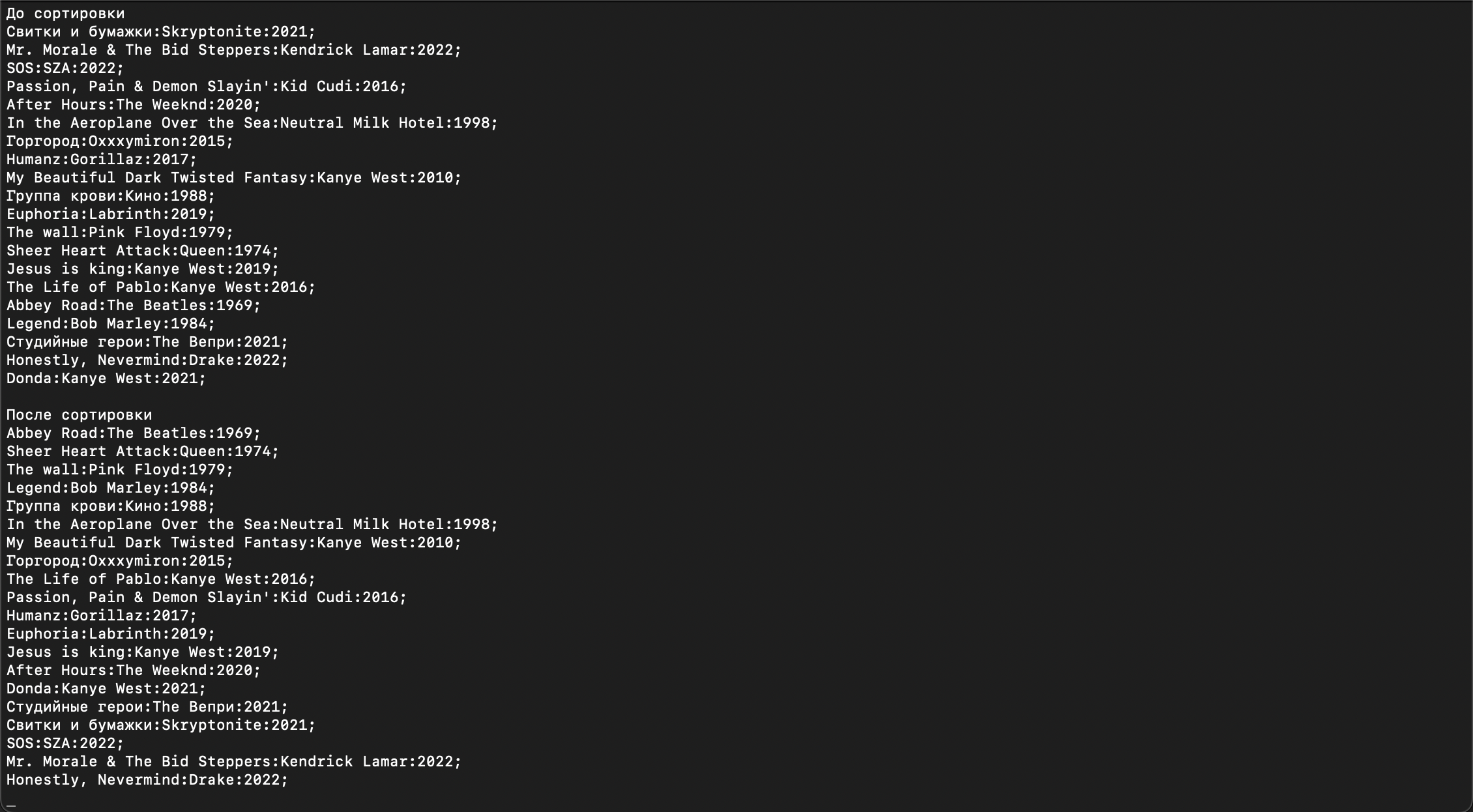


Рисунок 7 – Результат сортировки записей быстрой сортировкой

Полученные результаты

Получены характеристики разных видов сортировок с разными видами данных. Сравнив характеристики можно оценить каждую сортировку.

Быстрая сортировка оправдывает своё называние, являясь самой быстрой.

Сортировка простыми вставками вторая по скорости.

Сортировка бинарными вставками почти не получает выигрыша во времени (поскольку поиск места вставки отнимает не так много времени как этап вставки в найденную позицию) а в почти отсортированном массиве и вовсе проиграет сортировке простыми вставками, он важен скорее как часть других алгоритмов сортировки (сортировки библиотекаря или пасьянсная).

Сортировка простым обменом является самой медленной из-за очень большого количества сравнений и перестановок. В сортировке простым выбором при одном и том же количестве сравнений с сортировкой простым обменом количество перестановок в тысячу раз меньше, что уменьшает время на сравнение.

Сортировка простым выбором имеет такое же количество сравнений, как и сортировка простым обменом, но из-за того, что количество перестановок в тысячу раз меньше, значительно уменьшает время, потраченное на сравнения. Гномья сортировка является модификацией сортировки простым выбором, а именно в ней оптимизировано количество сравнений, что делает ее быстрее чем сортировку простым обменом.

Код программы

*// Внутренние сортировки  
// Туровец В.Ю.  
// 2 янв 2023*using System.Diagnostics;  
  
namespace Internal\_sorts  
{  
 class Disk  
 {   
 *// -название альбома;  
 // -исполнитель;  
 // -год выхода. (поле сортировки)* private static (List<string>, List<string>, List<int>) Texter()*//делает готовые массивы дат, исполнителей и названий на основе текста из файла* {  
 var listOfAlbums = new List<string>();  
 var listOfNames = new List<string>();  
 var listOfyears = new List<int>();  
  
 const string **path** = "/Users/vlad/Desktop/ConsoleApp1/ConsoleApp1/Albums\_mas.txt";  
 for (int i = 0; File.ReadLines(**path**).ElementAtOrDefault(i) != null; i++)  
 {  
 string? readText = File.ReadLines(**path**).ElementAtOrDefault(i);  
 readText = readText.Replace("; ", ";"); *//замена* string[] text = (readText.Split(';').ToArray());  
 int j = 0;  
 foreach (var part in text)  
 {  
 j++;  
 *//Console.WriteLine(j);* if (j == 1){listOfAlbums.Add(part);}  
 if (j == 2){listOfNames.Add(part);}  
 if (j == 3){int.TryParse(part, out var year);listOfyears.Add(year);}  
   
 }  
 }  
   
 return (listOfAlbums, listOfNames, listOfyears);  
 }  
  
 public static void DiskSort()  
 {  
 var listOfAlbums = new List<string>(); *//Названия всех альбомов* var listOfNames = new List<string>(); *//Имена всех исполнителей* var listOfyears = new List<int>(); *//Года выпуска альбомов* (listOfAlbums, listOfNames, listOfyears) = Texter();  
  
 Console.WriteLine("До сортировки");  
 for (int i = 0; i < listOfNames.Count; i++)  
 {  
 Console.Write($"{listOfAlbums[i]}:{listOfNames[i]}:{listOfyears[i]}; \n");  
 }  
   
 listOfyears = QuickSort(listOfyears, 0, listOfyears.Count - 1);  
  
 List<int> QuickSort(List<int> array, int minIndex, int maxIndex)  
 {  
 if (minIndex >= maxIndex)  
 {  
 return array;  
 }  
  
 int pivotIndex = GetPivotIndex(array, minIndex, maxIndex);  
  
 QuickSort(array, minIndex, pivotIndex - 1);  
  
 QuickSort(array, pivotIndex + 1, maxIndex);  
  
 return array;  
 }  
  
 int GetPivotIndex(List<int> array, int minIndex, int maxIndex)  
 {  
 int pivot = minIndex - 1;  
  
 for (int i = minIndex; i <= maxIndex; i++)  
 {  
 if (array[i] < array[maxIndex])  
 {  
 pivot++;  
 (listOfyears[pivot], listOfyears[i]) = (listOfyears[i], listOfyears[pivot]); *//Замена местами годов* (listOfNames[pivot], listOfNames[i]) = (listOfNames[i], listOfNames[pivot]); *//Замена местами названий альбовом* (listOfAlbums[pivot], listOfAlbums[i]) = (listOfAlbums[i], listOfAlbums[pivot]); *//Замена местами имён исполнителей* }  
 }  
  
 pivot++;  
 *//меняем значения местами* (listOfyears[pivot], listOfyears[maxIndex]) = (listOfyears[maxIndex], listOfyears[pivot]); *//Замена местами годов* (listOfNames[pivot], listOfNames[maxIndex]) = (listOfNames[maxIndex], listOfNames[pivot]); *//Замена местами названий альбовом* (listOfAlbums[pivot], listOfAlbums[maxIndex]) = (listOfAlbums[maxIndex], listOfAlbums[pivot]); *//Замена местами имён исполнителей* return pivot;  
 }  
   
 Console.WriteLine("\nПосле сортировки");  
   
 for (int i = 0; i < listOfNames.Count; i++)  
 {  
 Console.Write($"{listOfAlbums[i]}:{listOfNames[i]}:{listOfyears[i]}; \n");  
 }  
 }  
   
  
 }  
   
 class Program  
 {  
 private const int Minimal = -10; *//минимальный возможный элемент* private const int Maximum = 75; *//максимальный возможный элемент* public static void Swap(ref int e1, ref int e2) {(e1, e2) = (e2, e1);} *//обмен элементов  
   
 //Конвертер из строки в массив* private static int[] Texter(int ask)*//делает готовый массив чисел на основе текста из файла* {  
 const string **path** = "/Users/vlad/Desktop/ConsoleApp1/ConsoleApp1/Input\_mas.txt";  
 string readText = File.ReadLines(**path**).ElementAtOrDefault(ask - 1);  
 *//Console.WriteLine(readText);* readText = readText.Replace("; ", " "); *//замена* var Text = readText.Split(' ').Select(int.Parse).ToArray();  
 return Text;  
 }  
  
 *//Сортировка простыми вставками* static int[] EasyInserts(int[] array)  
 {  
 var watch = System.Diagnostics.Stopwatch.StartNew();  
 int comparisons = 0; *//кол-во сравнений* int rearrangements = 0; *//кол-во перестановок* for (var i = 0; i < array.Length; i++) *//Проходим по массиву слева направо и обрабатываем по очереди каждый элемент.* {  
 var key = array[i];  
 var j = i;  
 rearrangements++;  
 *//Слева от элемента наращиваем отсортированную часть массива,  
 //справа по мере процесса потихоньку испаряется неотсортированная.* while ((j >= 1) && (array[j - 1] > key))  
 {  
 {  
 *//В отсортированной части массива ищется точка вставки для очередного элемента.* comparisons++;  
 Swap(ref array[j - 1], ref array[j]);  
 j--;  
 }  
 }  
  
 array[j] = key;  
 }  
   
 watch.Stop();  
 var elapsedMs = watch.ElapsedMilliseconds;  
 Console.WriteLine($"{comparisons} | {rearrangements} | {elapsedMs} Сортировка простыми вставками");  
 return array;  
 }  
  
 *//Сортировка простым обменом* static int[] Bubble(int[] array)  
 {  
 var watch = System.Diagnostics.Stopwatch.StartNew();  
 int comparisons = 0; *//кол-во сравнений* int rearrangements = 0; *//кол-во перестановок* var len = array.Length;  
 for (var i = 1; i < len; i++)  
 {  
 for (var j = 0; j < len - i; j++)  
 {  
 comparisons++;  
 if (array[j] > array[j + 1])  
 {  
 Swap(ref array[j], ref array[j + 1]);  
 rearrangements++;  
 }  
 }  
 }  
  
 watch.Stop();  
 var elapsedMs = watch.ElapsedMilliseconds;  
 Console.WriteLine($"\n{comparisons} | {rearrangements} | {elapsedMs} Сортировка простым обменом");  
 return array;  
 }  
   
 *//Сортировка простым выбором* static int[] SelectionSort(int[] array)  
 {  
 var watch = System.Diagnostics.Stopwatch.StartNew();  
 int comparisons = 0; *//кол-во сравнений* int rearrangements = 0; *//кол-во перестановок* int min, temp;  
  
 for (int i = 0; i < array.Length - 1; i++)  
 {  
 min = i; *//устанавливаем начальное значение минимального индекса  
  
 //находим минимальный индекс элемента* for (int j = i + 1; j < array.Length; j++)  
 {  
 comparisons++;  
 if (array[j] < array[min])  
 {  
 min = j;  
 }  
 }  
 rearrangements++;  
 *//меняем значения местами* Swap(ref array[min], ref array[i]);  
 }  
  
 watch.Stop();  
 var elapsedMs = watch.ElapsedMilliseconds;  
 Console.WriteLine($"\n{comparisons} | {rearrangements} | {elapsedMs} Сортировка простым выбором");  
 return array;  
 }  
   
 *//Сортировка бинарными вставками* static int[] BinInsertSort(int[] array)  
 {  
 var watch = System.Diagnostics.Stopwatch.StartNew();  
 int comparisons = 0; *//кол-во сравнений* int rearrangements = 0; *//кол-во перестановок* for (int i = 0; i < array.Length; i++)  
 {  
 var key = array[i];  
 var lo = 0;  
 var hi = i - 1;  
   
 while (lo < hi)  
 {  
 int mid = lo + (hi - lo) / 2;  
 if (key < array[i]) { hi = mid; }  
 else { lo = mid + 1; }  
 }  
   
 for (int j = i; i < j && j < lo + 1; j--) { array[j] = array[j - 1]; }  
  
 array[lo] = key;  
 }  
  
 watch.Stop();  
 var elapsedMs = watch.ElapsedMilliseconds;  
 Console.WriteLine($"\n{comparisons} | {rearrangements} | {elapsedMs} Сортировка бинарными вставками");  
 return array;  
 }  
   
 *//Быстрая сортировка* static int[] QuickSort(int[] array)  
 {  
 var watch = System.Diagnostics.Stopwatch.StartNew();  
 int comparisons = 0; *//кол-во сравнений* int rearrangements = 0; *//кол-во перестановок* array = QuickSort(array, 0, array.Length - 1);   
 *//Для простоты метод получает на вход границы участка массива от минимального до максимального* int[] QuickSort(int[] quickSortingArray, int leftIndex, int rightIndex)   
 {  
 if (leftIndex >= rightIndex){return quickSortingArray;} *//Если левая граница правее правой - возвращаем масив* int pivotIndex = GetPivotIndex(quickSortingArray, leftIndex, rightIndex);*//Стержневой индекс   
  
 //Делим масив на две часты (слева и справа от стержня)  
 //Соритруем слева от стержня* QuickSort(quickSortingArray, leftIndex, pivotIndex - 1);  
 *//Соритруем справа от стержня* QuickSort(quickSortingArray, pivotIndex + 1, rightIndex);  
  
 return quickSortingArray;  
 }  
  
 int GetPivotIndex(int[] pivotingArray, int leftIndex, int rightIndex)*//Стержневой индекс* {  
 int pivot = leftIndex - 1;  
  
 for (int i = leftIndex; i <= rightIndex; i++)  
 {  
 comparisons++;  
 if (pivotingArray[i] < pivotingArray[rightIndex])  
 {  
 pivot++;  
 Swap(ref pivotingArray[pivot], ref pivotingArray[i]);  
 rearrangements++;  
 }  
 }  
  
 pivot++;  
 Swap(ref pivotingArray[pivot], ref pivotingArray[rightIndex]);  
  
 return pivot;  
 }  
  
  
 watch.Stop();  
 var elapsedMs = watch.ElapsedMilliseconds;  
 Console.WriteLine($"\n{comparisons} | {rearrangements} | {elapsedMs} Быстрая сортировка");  
 return array;  
 }static void Main()  
 {  
 do  
 {  
 Console.Clear();  
 Console.WriteLine("Выберите тип данных");  
 Console.WriteLine("1 - Числа");  
 Console.WriteLine("2 - Диски");  
 Console.WriteLine("0 - Выход");  
 int.TryParse(Console.ReadLine(), out var how); if (how == 0) {break;} if (how > 2) {continue;}  
  
 switch (how)  
 {  
 case 1:  
 Console.Clear();  
 Console.WriteLine("Выберите количество тестовых чисел");  
 Console.WriteLine("1 - 500");  
 Console.WriteLine("2 - 1000");  
 Console.WriteLine("3 - 5000");  
 Console.WriteLine("0 - Назад");  
 int.TryParse(Console.ReadLine(), out var ask); if (ask == 0) {continue;} if (ask > 3) {continue;}  
  
 int[] numbersArray = Texter(ask);*//Готовый массив чисел на основе текста из файла* for (var i = 0; i < 10; i++){Console.Write($"{numbersArray[i]} ");} *//Вывод неотсортированного массива* Console.Clear();  
 Console.WriteLine($"comparisons | rearrangements | time ");  
 EasyInserts(numbersArray);  
  
 numbersArray = Texter(ask); *//Приведение масива к неотсортированному виду* Bubble(numbersArray);   
   
 numbersArray = Texter(ask); *//Приведение масива к неотсортированному виду* SelectionSort(numbersArray);  
   
 numbersArray = Texter(ask); *//Приведение масива к неотсортированному виду* BinInsertSort(numbersArray);   
   
 numbersArray = Texter(ask); *//Приведение масива к неотсортированному виду* QuickSort(numbersArray);   
 break;  
   
 case 2:  
 Console.Clear();  
 Disk.DiskSort();  
 break;  
 }  
 int.TryParse(Console.ReadLine(), out var waiter);  
 } while (true);  
 }  
 }  
}

Приложение 7 – Код программы