Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий
институт
Кафедра «Информатика»
кафедра

ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ

Задание 5 - Алгоритмы обработки последовательностей
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

Научится реализовывать алгоритмы внешней сортировки.

1.2 Задача работы

Реализовать в программе один из алгоритмов (по выбору студента) из указанных ниже:

- простого слияния,
- естественного слияния,
- многофазного слияния,
- много путевого слияния.

Важно! При сортировке последовательностей должны использоваться файлы, но не массивы.

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 Описание реализованного алгоритма

Был реализован алгоритм простого слияния.

3 Описание программы (листинги кода)

Листинг 1 – Alg_05/Alg_05.Core/Splitter.cs

```
using System.IO;

namespace Alg_05.Core
{
    public class Splitter
    {
        public Splitter(BinaryReader input, BinaryWriter outputA, BinaryWriter
outputB, int sectionLength)
        {
            Input = input;
            OutputA = outputA;
            OutputB = outputB;
            SectionLength = sectionLength;
        }

        private BinaryReader Input { get; }
        private BinaryWriter OutputA { get; }
        private BinaryWriter OutputB { get; }

        private int SectionLength { get; }

        public void Split()
        {
            Input.BaseStream.Seek(0, SeekOrigin.Begin);
            OutputA.BaseStream.Seek(0, SeekOrigin.Begin);
            OutputB.BaseStream.Seek(0, SeekOrigin.Begin);

            var i = 0;
            while (Input.BaseStream.Position / 4 != Input.BaseStream.Length / 4)
            {
                var a = Input.ReadInt32();
                if (i < SectionLength)
                {
                    OutputA.Write(a);
                }
                else
                {
                    OutputB.Write(a);
                }

                i++;
                i %= SectionLength * 2;
            }
        }
    }
}
```

Листинг 2 – Alg_05/Alg_05.Core/Merger.cs

```
using System.IO;
```

```
namespace Alg_05.Core
{
    public class Merger
    {
        public Merger(BinaryReader inputA, BinaryReader inputB, BinaryWriter
output, int sectionLength)
        {
            InputA = inputA;
            InputB = inputB;
            Output = output;
            SectionLength = sectionLength;
        }

        private BinaryReader InputA { get; }
        private BinaryReader InputB { get; }
        private BinaryWriter Output { get; }
        private int SectionLength { get; }

        public void Merge()
        {
            InputA.BaseStream.Seek(0, SeekOrigin.Begin);
            InputB.BaseStream.Seek(0, SeekOrigin.Begin);
            Output.BaseStream.Seek(0, SeekOrigin.Begin);

            var a = 0;
            var b = 0;

            var mergedAllA = 0;
            var mergedAllB = 0;
            var countA = InputA.BaseStream.Length / 4;
            var countB = InputB.BaseStream.Length / 4;

            var j = 0;
            var readedA = 0;
            var readedB = 0;
            var mergedA = 0;
            var mergedB = 0;
            while (mergedAllA != countA || mergedAllB != countB)
            {
                if (j == 0)
                {
                    readedA = 0;
                    readedB = 0;
                    mergedA = 0;
                    mergedB = 0;
                }

                j++;
                j %= SectionLength * 2;

                if (mergedA == SectionLength || mergedAllA == countA)
                {
                    if (readedB == mergedB)
                    {
                        b = InputB.ReadInt32();
                        readedB++;
                    }

                    Output.Write(b);
                    mergedB++;
                    mergedAllB++;
                    continue;
                }
```

```
                if (mergedB == SectionLength || mergedAllB == countB)
                {
                    if (readedA == mergedA)
                    {
                        a = InputA.ReadInt32();
                        readedA++;
                    }

                    Output.Write(a);
                    mergedA++;
                    mergedAllA++;
                    continue;
                }

                if (readedA == 0 && readedB == 0)
                {
                    a = InputA.ReadInt32();
                    readedA++;
                    b = InputB.ReadInt32();
                    readedB++;
                }
                else
                {
                    if (mergedA == mergedB && readedA > readedB || readedB ==
mergedB)
                    {
                        b = InputB.ReadInt32();
                        readedB++;
                    }
                    else
                    {
                        a = InputA.ReadInt32();
                        readedA++;
                    }
                }


                if (a < b)
                {
                    Output.Write(a);
                    mergedA++;
                    mergedAllA++;
                }
                else
                {
                    Output.Write(b);
                    mergedB++;
                    mergedAllB++;
                }
            }
        }
    }
}
```

Листинг 3 – Alg_05/Alg_05.Core/MergeSort.cs

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

```
namespace Alg_05.Core
{
    public class MergeSort
    {
        public MergeSort(BinaryReader input, IEnumerable<BinaryReader>
intermediateInputs,
            IEnumerable<BinaryWriter> intermediateOutputs, BinaryWriter output)
        {
            Input = input;
            IntermediateInputs = intermediateInputs.Append(input).ToList();
            IntermediateOutputs = intermediateOutputs.Append(output).ToList();
        }

        private BinaryReader Input { get; }
        private IList<BinaryReader> IntermediateInputs { get; }
        private IList<BinaryWriter> IntermediateOutputs { get; }

        public void Sort()
        {
            var sectionLength = 1;
            var count = Input.BaseStream.Length / 4;
            var i = 0;
            var prevMerged = Input;
            while (sectionLength < count)
            {
                var splitter = new Splitter(prevMerged, IntermediateOutputs[i],
IntermediateOutputs[i + 1],
                    sectionLength);
                splitter.Split();

                var merger = new Merger(IntermediateInputs[i],
IntermediateInputs[i + 1], IntermediateOutputs[i + 2],
                    sectionLength);
                merger.Merge();

                prevMerged = IntermediateInputs[i + 2];
                i += 3;
                sectionLength *= 2;
            }
        }
    }
}
```

Листинг 4 – Alg_05/Alg_05.Core/MergeSortFileHelper.cs

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

namespace Alg_05.Core
{
    public class MergeSortFileHelper : IDisposable
    {
        public MergeSortFileHelper(string pathInput, string
pathPrefixIntermediate, string suffixIntermediate,
            string pathOutput)
        {
```

```
            InputFile = new FileStream(pathInput, FileMode.Open);
            var input = new BinaryReader(InputFile);
            OutputFile = new FileStream(pathOutput, FileMode.Create);
            var output = new BinaryWriter(OutputFile);

            var count = input.BaseStream.Length / 4;

            Files = new List<FileStream>();
            var sectionLength = 1;
            while (sectionLength < count)
            {
                Files.Add(new
FileStream($"{pathPrefixIntermediate}_{sectionLength}_SplittedA{suffixIntermedia
te}",
                    FileMode.Create));
                Files.Add(new
FileStream($"{pathPrefixIntermediate}_{sectionLength}_SplittedB{suffixIntermedia
te}",
                    FileMode.Create));
                if (sectionLength * 2 < count)
                {
                    Files.Add(new
FileStream($"{pathPrefixIntermediate}_{sectionLength *
2}_Merged{suffixIntermediate}",
                        FileMode.Create));
                }

                sectionLength *= 2;
            }

            var br = Files.Select(s => new BinaryReader(s));
            var bw = Files.Select(s => new BinaryWriter(s));

            Sort = new MergeSort(input, br, bw, output);
        }

        public FileStream InputFile { get; }

        public FileStream OutputFile { get; }

        public IList<FileStream> Files { get; }

        public MergeSort Sort { get; }

        public void Dispose()
        {
            InputFile.Dispose();
            OutputFile.Dispose();
            foreach (var i in Files)
            {
                i.Dispose();
            }
        }
    }
}
```

Листинг 5 – Alg_05/Alg_05.Core.Tests/MergeSortTests.cs

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

using NUnit.Framework;

namespace Alg_05.Core.Tests
{
    public class MergeSortTests
    {
        [Test]
        public void MergeSortTest1()
        {
            var a = new[] {54, 32, 12, 30, 16, 24, 92};

            var r1 = new byte[a.Length * 4];
            using (var g = new BinaryWriter(new MemoryStream(r1)))
            {
                foreach (var i in a)
                {
                    g.Write(i);
                }
            }

            var oa1 = new byte[16];
            var ob1 = new byte[12];
            var r2 = new byte[28];
            var oa2 = new byte[16];
            var ob2 = new byte[12];
            var r4 = new byte[28];
            var oa4 = new byte[16];
            var ob4 = new byte[12];
            var r8 = new byte[28];

            var interm = new[] {oa1, ob1, r2, oa2, ob2, r4, oa4, ob4};

            var br = interm.Select(s => new BinaryReader(new MemoryStream(s)));
            var bw = interm.Select(s => new BinaryWriter(new MemoryStream(s)));

            var sort = new MergeSort(new BinaryReader(new MemoryStream(r1)), br,
bw,
                new BinaryWriter(new MemoryStream(r8)));

            sort.Sort();

            var b = new List<int>();
            using (var g = new BinaryReader(new MemoryStream(r8)))
            {
                b.AddRange(Enumerable.Range(0, 7).Select(i => g.ReadInt32()));
            }

            Assert.That(b, Is.EquivalentTo(a.OrderBy(y => y)));
        }
    }
}
```

Листинг 6 – Alg_05/Alg_05.Core.Tests/MergeSortFileHelperTests.cs

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

using NUnit.Framework;

namespace Alg_05.Core.Tests
{
    public class MergeSortFileHelperTests
    {
        [Test]
        public void FileMergeSortTest1()
        {
            var a = new[] {54, 32, 12, 30, 16, 24, 92};
            var pi = "test1.bin";
            var pip = "test1_intr";
            var pis = ".bin";
            var po = "test1_result.bin";

            using (var g = new BinaryWriter(new FileStream(pi,
FileMode.Create)))
            {
                foreach (var i in a)
                {
                    g.Write(i);
                }
            }

            using (var msh = new MergeSortFileHelper(pi, pip, pis, po))
            {
                var sort = msh.Sort;
                sort.Sort();
            }

            var b = new List<int>();
            using (var g = new BinaryReader(new FileStream(po, FileMode.Open)))
            {
                b.AddRange(Enumerable.Range(0, a.Length).Select(i =>
g.ReadInt32()));
            }

            Assert.That(b, Is.EquivalentTo(new[] {12, 16, 24, 30, 32, 54, 92}));
        }
    }
}
```

Листинг 7 – Alg_05/Alg_05.Console/Program.cs

```
using System;
using System.IO;
using System.Text;

using Alg_05.Core;

namespace Alg_05.Console
{
```

```
    internal static class Program
    {
        private static void OutputBinaryFile(FileStream s)
        {
            System.Console.Write($"{Path.GetFileName(s.Name)}: ");
            var br = new BinaryReader(s);
            br.BaseStream.Seek(0, SeekOrigin.Begin);
            while (br.BaseStream.Position != br.BaseStream.Length)
            {
                System.Console.Write(br.ReadInt32());
                System.Console.Write(" ");
            }

            System.Console.WriteLine();
        }

        private static void Main(string[] args)
        {
            System.Console.InputEncoding = Encoding.UTF8;
            System.Console.OutputEncoding = Encoding.UTF8;

            while (true)
            {
                try
                {
                    System.Console.WriteLine("Ввести данные в файл или
произвести сортировку (1/2)?");
                    var ans = Int32.Parse(System.Console.ReadLine());
                    System.Console.WriteLine("Введите название файла:");
                    var path = System.Console.ReadLine();
                    switch (ans)
                    {
                        case 1:
                        {
                            using var bw = new BinaryWriter(new FileStream(path,
FileMode.Create));
                            System.Console.WriteLine("Введите элементы через
Enter:");
                            while (true)
                            {
                                var l = System.Console.ReadLine();
                                if (l == "")
                                {
                                    break;
                                }

                                var elem = Int32.Parse(l);
                                bw.Write(elem);
                            }

                            break;
                        }
                        case 2:
                        {
                            using var msh = new MergeSortFileHelper(path,
                                Path.GetFileNameWithoutExtension(path),
                                Path.GetExtension(path),

$"{Path.GetFileNameWithoutExtension(path)}_Result{Path.GetExtension(path)}");
                            msh.Sort.Sort();

                            OutputBinaryFile(msh.InputFile);
                            foreach (var i in msh.Files)
```

```
                                {
                                    OutputBinaryFile(i);
                                }

                                OutputBinaryFile(msh.OutputFile);

                                break;
                            }
                            default:
                                throw new ApplicationException("Введите 1 или 2!");
                        }

                        break;
                    }
                    catch (Exception e)
                    {
                        System.Console.Error.WriteLine($"Ошибка: {e.Message}\n");
                    }
                }
            }
        }
    }
}
```

4 Результаты работы программы

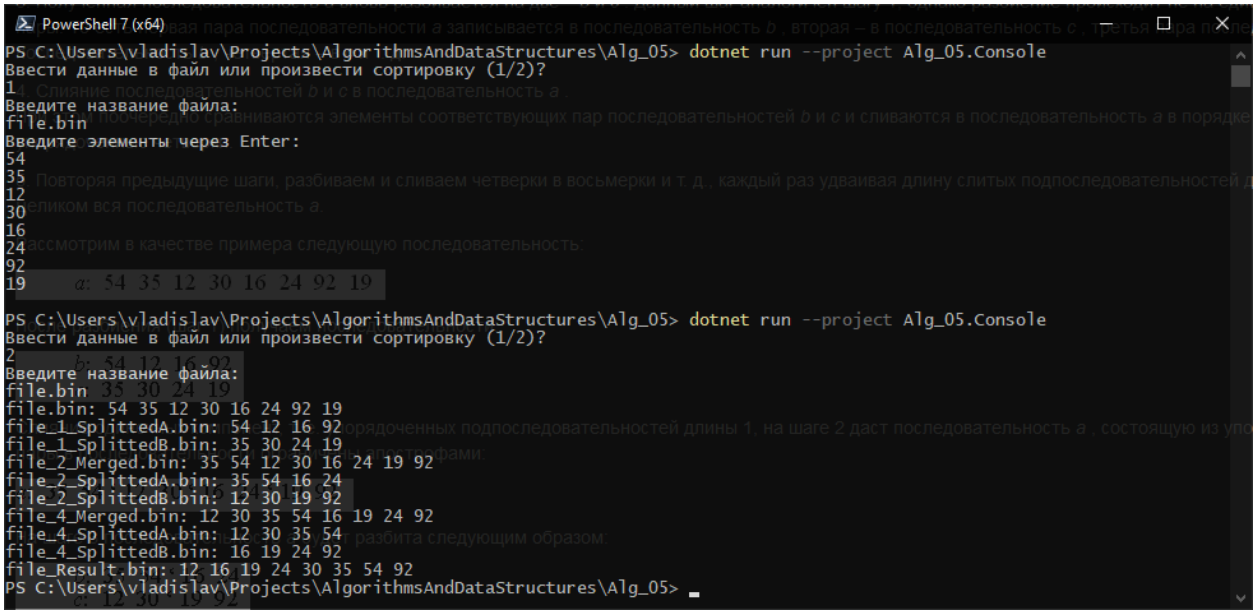

Рисунок 1 – Запуск программы, которая записывает в file.bin пример из буи и запуск сортировки этого файла