МГТУ им. Баумана

Факультет «Информатика и системы управления»

Кафедра ИУ5«Системы обработки информации и управления»

Дисциплина «Базовые компоненты интернет технологий»

Отчёт по домашнему заданию

Выполнил:

Попов М.А.

ИУ5-35Б

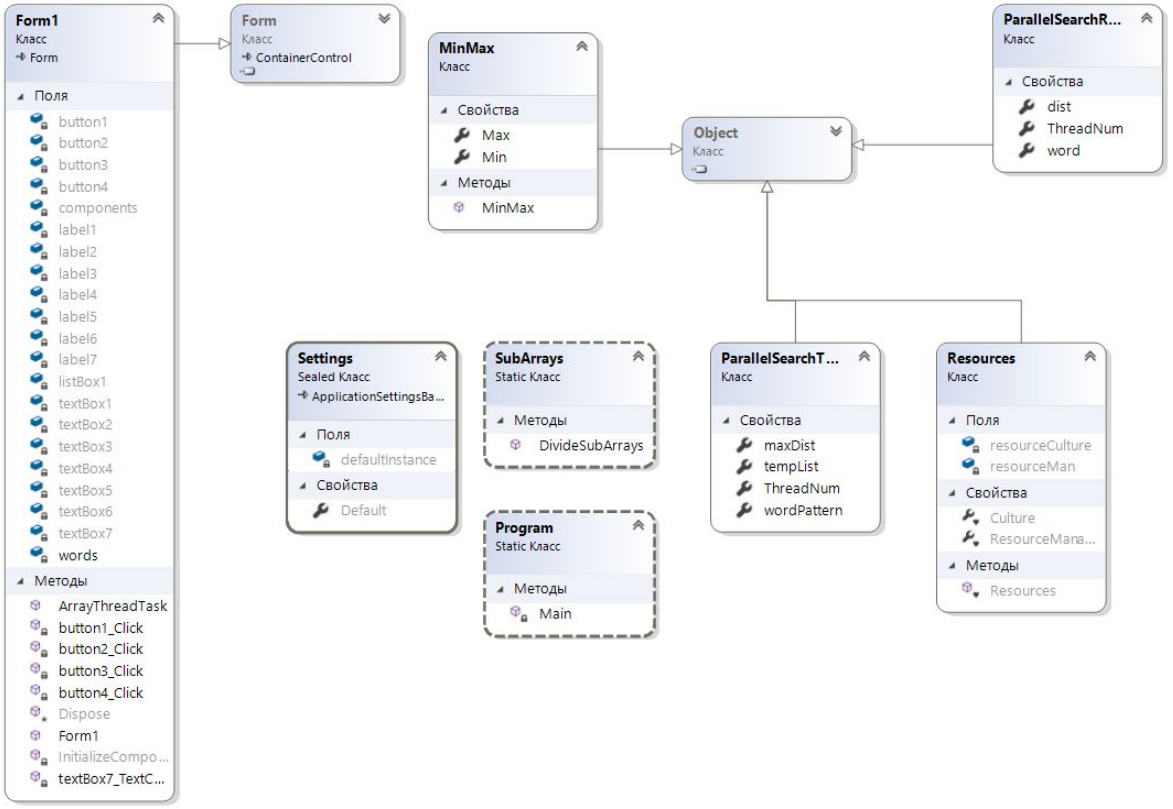
Москва, 2019г

Задание:

Разработать программу, реализующую многопоточный поиск в файле.

1. Программа должна быть разработана в виде приложения Windows Forms на языке C#. По желанию вместо Windows Forms возможно использование WPF;
2. В качестве основы используется макет, разработанный в лабораторных работах №4 и №5;
3. Реализуйте функцию поиска с использованием расстояния Левенштейна в многопоточном варианте. Количество потоков для запуска функции поиска вводится на форме в поле ввода (TextBox).
4. Реализуйте функцию записи результатов поиска в файл отчета. Файл отчета создается в формате .txt или .html

Диаграмма классов:



Текст программы:

Form1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;
using System.Diagnostics;
```

```
using LLab5;

namespace Homework
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        List<string> words = new List<string>();
        private void button1_Click(object sender, EventArgs e)
        {
            OpenFileDialog openFileDialog = new OpenFileDialog();

            openFileDialog.InitialDirectory = "D:\\Lab_C\\lab04";
            openFileDialog.Filter = "txt files (*.txt)|*.txt";
            openFileDialog.FilterIndex = 1;
            openFileDialog.RestoreDirectory = true;

            if (openFileDialog.ShowDialog() == DialogResult.OK)
            {
                Stopwatch stopWatch = new Stopwatch();
                stopWatch.Start();

                string path = openFileDialog.FileName;
                StreamReader sr = new StreamReader(path);

                string text = sr.ReadToEnd();

                char[] pattern = { ' ', ',', '.', ';', ':', '(', ')', '\n' };
                string[] words_str = text.Split(pattern);

                for (int j = 0; j < words_str.Length; j++)
```

```
            {
                if ((String.Compare(words_str[j], "\0") != 0) &&
(String.Compare(words_str[j], "\n") != 0) &&
(String.Compare(words_str[j], "\r") != 0))
                {
                    if (!words.Contains(words_str[j]))
                    {
                        words.Add(words_str[j]);
                    }
                }

            }
            stopWatch.Stop();

            this.textBox1.Text = stopWatch.Elapsed.ToString();

        }
    }

    private void button2_Click(object sender, EventArgs e)
    {
        string right_word = textBox2.Text;

        if (!string.IsNullOrWhiteSpace(right_word))
        {
            Stopwatch stopWatch = new Stopwatch();
            stopWatch.Start();
            foreach (string word in words)
            {
                if (word.Contains(right_word))
                {
                    listBox1.BeginUpdate();
                    listBox1.Items.Add(word);
                    listBox1.EndUpdate();
                }
```

```
            }
            stopWatch.Stop();

            this.textBox3.Text = stopWatch.Elapsed.ToString();
        }
        else
        {
            MessageBox.Show("Необходимо выбрать файл и ввести
слово для поиска");
        }
    }

    private void textBox7_TextChanged(object sender, EventArgs e)
    {

    }

    private void button3_Click(object sender, EventArgs e)
    {
        string right_word = textBox2.Text;

        if (!string.IsNullOrWhiteSpace(right_word))
        {
            int max_dist;
            if (!int.TryParse(this.textBox4.Text, out max_dist))
            {
                MessageBox.Show("Необходимо указать максимальное
расстояние");
                return;
            }
            if (max_dist < 1 || max_dist > 5)
            {
                MessageBox.Show("Максимальное расстояние должно
быть в диапазоне от 1 до 5");
                return;
```

```
            }
            int thread_count;
            if (!int.TryParse(this.textBox5.Text, out thread_count))
            {
                MessageBox.Show("Необходимо указать количество
потоков");
                return;
            }

            Stopwatch t = new Stopwatch();

            t.Start();
            List<ParallelSearchResult> Result = new
List<ParallelSearchResult>();
            List<MinMax> arrayDivList = SubArrays.DivideSubArrays(0,
words.Count, thread_count);
            int count = arrayDivList.Count;

            Task<List<ParallelSearchResult>>[] tasks = new
Task<List<ParallelSearchResult>>[count];

            for (int i = 0; i < count; i++)
            {
                List<string> tempTaskList =
words.GetRange(arrayDivList[i].Min,
                    arrayDivList[i].Max - arrayDivList[i].Min);

                tasks[i] = new Task<List<ParallelSearchResult>>(
                    ArrayThreadTask,
                     new ParallelSearchThreadParam()
                     {
                        tempList = tempTaskList,
                        maxDist = max_dist,
                        ThreadNum = i,
                        wordPattern = right_word
```

```
                });
            tasks[i].Start();
        }
        Task.WaitAll(tasks);
        t.Stop();
        for (int i = 0; i < count; i++)
        {
            Result.AddRange(tasks[i].Result);
        }
        t.Stop();

        this.textBox7.Text = t.Elapsed.ToString();

        this.textBox6.Text = count.ToString();
        this.listBox1.BeginUpdate();
        this.listBox1.Items.Clear();

        foreach (var x in Result)
        {
            string temp = x.word + "(расстояние=" + x.dist.ToString() +
" поток="
                + x.ThreadNum.ToString() + ")";
            this.listBox1.Items.Add(temp);
        }
        this.listBox1.EndUpdate();
    }
    else
    {
        MessageBox.Show("Необходимо выбрать файл и ввести
слово для поиска");
    }
}

    public static List<ParallelSearchResult> ArrayThreadTask(object
paramObj)
```

```
        {
            DamLev L1 = new DamLev();
            ParallelSearchThreadParam param =
(ParallelSearchThreadParam)paramObj;

            string wordUpper = param.wordPattern.Trim().ToUpper();

            List<ParallelSearchResult> Result = new
List<ParallelSearchResult>();

            foreach (string str in param.tempList)
            {
                int dist = L1.Distance(str.ToUpper(), wordUpper);

                if (dist <= param.maxDist)
                {
                    ParallelSearchResult temp = new ParallelSearchResult()
                    {
                        word = str,
                        dist = dist,
                        ThreadNum = param.ThreadNum
                    };
                    Result.Add(temp);
                }

            }
            return Result;
        }

        private void button4_Click(object sender, EventArgs e)
        {
            string TempReportFileName = "Report_" +
DateTime.Now.ToString("dd_MM_yyyy_hhmmss");
            SaveFileDialog fd = new SaveFileDialog();
            fd.FileName = TempReportFileName;
```

```
            fd.DefaultExt = ".html";
            fd.Filter = "HTML Reports|*.html";

            if (fd.ShowDialog() == DialogResult.OK)
            {
                string ReportFileName = fd.FileName;
                StringBuilder b = new StringBuilder();

                b.AppendLine("<html>");

                b.AppendLine("<head>");
                b.AppendLine("<meta http-equiv='Content-Type'
content='text/html; charset=UTF-8'/>");
                b.AppendLine("<title>" + "Отчет: " + ReportFileName +
"</title>");
                b.AppendLine("</head>");

                b.AppendLine("<body>");

                b.AppendLine("<h1>" + "Отчет: " + ReportFileName +
"</h1>");
                b.AppendLine("<table border='1'>");

                b.AppendLine("<tr>");
                b.AppendLine("<td>Время чтения из файла</td>");
                b.AppendLine("<td>" + this.textBox1.Text + "</td>");
                b.AppendLine("</tr>");

                b.AppendLine("<tr>");
                b.AppendLine("<td>Слово для поиска</td>");
                b.AppendLine("<td>" + this.textBox2.Text + "</td>");
                b.AppendLine("</tr>");

                b.AppendLine("<tr>");
```

```
            b.AppendLine("<td>Максимальное расстояние для
нечеткого поиска</td>");
            b.AppendLine("<td>" + this.textBox4.Text + "</td>");
            b.AppendLine("</tr>");

            b.AppendLine("<tr>");
            b.AppendLine("<td>Время четкого поиска</td>");
            b.AppendLine("<td>" + this.textBox3.Text + "</td>");
            b.AppendLine("</tr>");

            b.AppendLine("<tr>");
            b.AppendLine("<td>Время нечеткого поиска</td>");
            b.AppendLine("<td>" + this.textBox7.Text + "</td>");
            b.AppendLine("</tr>");

            b.AppendLine("<tr valign='top'>");
            b.AppendLine("<td>Результаты поиска</td>");
            b.AppendLine("<td>");
            b.AppendLine("<ul>");

            foreach (var x in this.listBox1.Items)
            {
                b.AppendLine("<li>" + x.ToString() + "</li>");
            }

            b.AppendLine("</ul>");
            b.AppendLine("</td>");
            b.AppendLine("</tr>");

            b.AppendLine("</table>");

            b.AppendLine("</body>");
            b.AppendLine("</html>");

            File.AppendAllText(ReportFileName, b.ToString());
```

```
            MessageBox.Show("Отчет сформирован. Файл: " +
ReportFileName);
          }
       }
   }
}
```

MinMax.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Homework
{
   public class MinMax
    {
        public int Min { get; set; }
        public int Max { get; set; }

        public MinMax(int pmin, int pmax)
        {
            this.Min = pmin; this.Max = pmax;
        }
    }
}
```

ParallelSearchResult.cs

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

```
using System.Text;
using System.Threading.Tasks;

namespace Homework
{
   public class ParallelSearchResult
    {
        public string word { get; set; }

        public int dist { get; set; }

        public int ThreadNum { get; set; }
    }
}
```

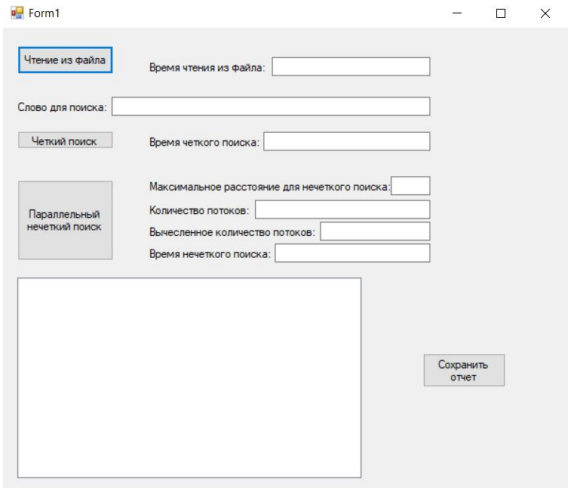
ParallelSearchThreadParam.cs

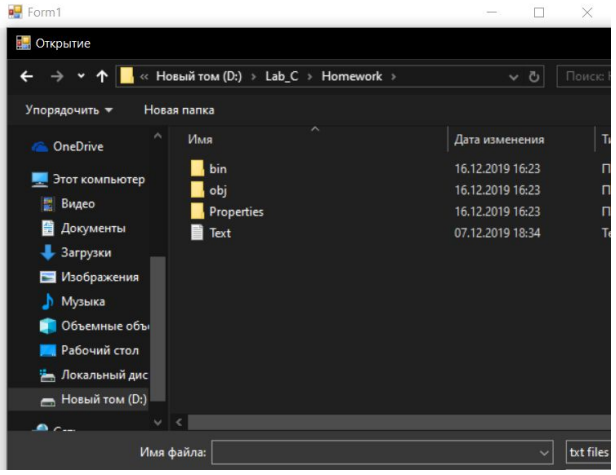
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Homework
{
    class ParallelSearchThreadParam
    {
        public List<string> tempList { get; set; }
        public string wordPattern { get; set; }
        public int maxDist { get; set; }
        public int ThreadNum { get; set; }
    }
}
```

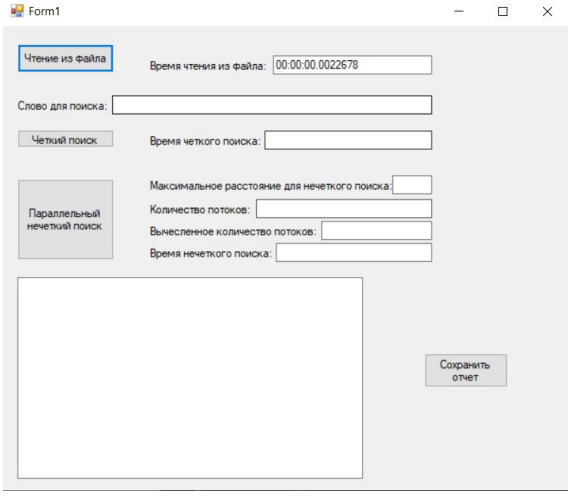
SubArrays.cs

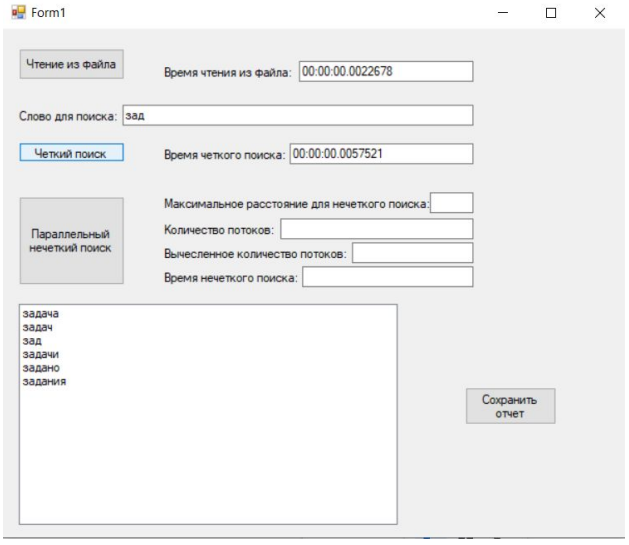
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Homework
{
   public static class SubArrays
    {
        public static List<MinMax> DivideSubArrays(int beginIndex, int
endIndex, int subArraysCount)
        {
            List<MinMax> result = new List<MinMax>();
            if ((endIndex - beginIndex) <= subArraysCount)
            {
                result.Add(new MinMax(0, (endIndex - beginIndex)));
            }
            else
            {
                int delta = (endIndex - beginIndex) / subArraysCount;
                int currentBegin = beginIndex;
                while ((endIndex - currentBegin) >= 2 * delta)
                {
                    result.Add(new MinMax(currentBegin, currentBegin +
delta));
                    currentBegin += delta;
                }
                result.Add(new MinMax(currentBegin, endIndex));
            }
            return result;
        }
    }
}
```

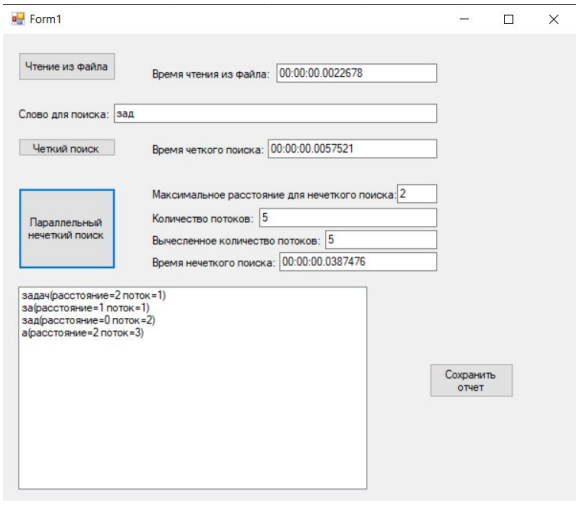
Экранные формы с примерами выполнения программы:

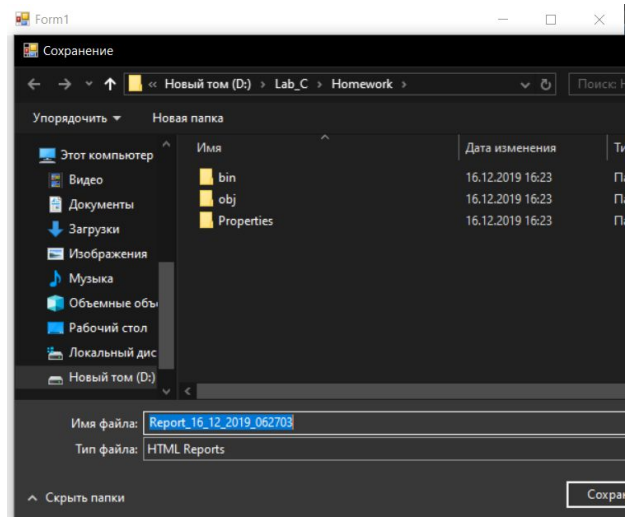


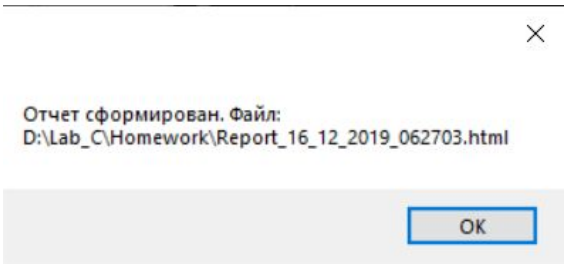






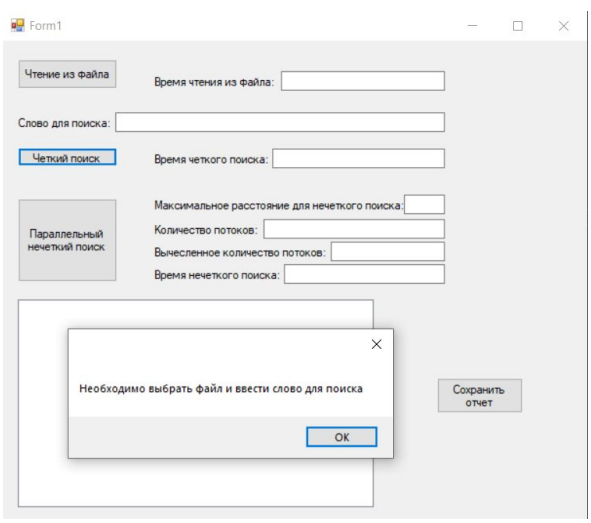


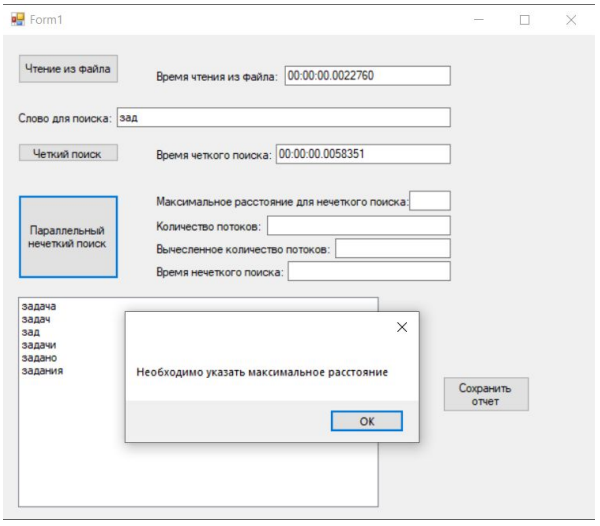


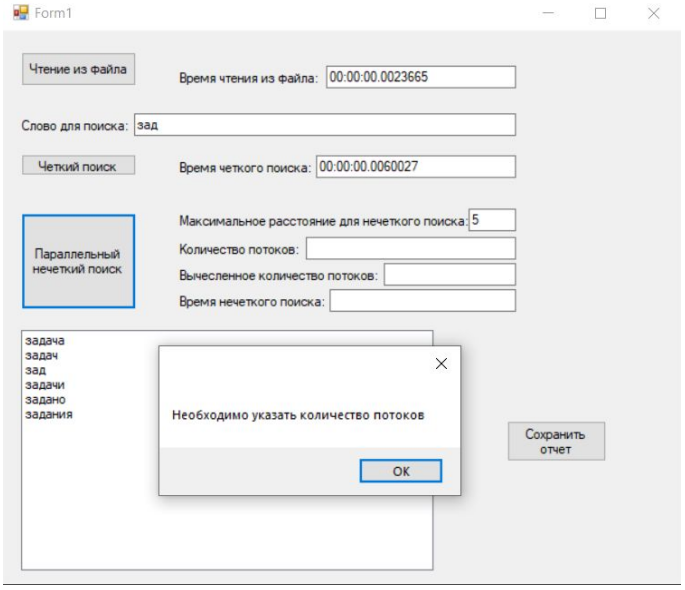


Отчет: D:\Lab_C\Homework\Report_16_12_2019_062703.html

Время чтения из файла	00:00:00.0022678
Слово для поиска	зад
Максимальное расстояние для нечеткого поиска	2
Время четкого поиска	00:00:00.0057521
Время нечеткого поиска	00:00:00.0387476
Результаты поиска	• задач(расстояние=2 поток=1) • за(расстояние=1 поток=1) • зад(расстояние=0 поток=2) • а(расстояние=2 поток=3)







Ссылка на репозиторий исходных кодов
GitHub: https://github.com/4Marvin2/Lab_C.git