

**Московский государственный технический  
университет им. Н.Э. Баумана.**

**Факультет «Информатика и управление»**

**Кафедра ИУ5.**

**Курс «Базовые компоненты интернет-технологий»**

**Отчет по домашнему заданию**

Выполнил:

студент группы ИУ5-31Б

Вардумян Арсен

Подпись и дата:

Проверил:

преподаватель каф. ИУ5

Гапанюк Юрий Евгеньевич

Подпись и дата:

г. Москва, 2020 г.

## **Постановка задачи**

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). В качестве примера используйте проект «Parallel» из примера «Введение в C#».

4.

Реализуйте функцию записи результатов поиска в файл отчета. Файл отчета создается в формате .txt или .html. В качестве примера используйте проект «WindowsFormsFiles» (обработчик события кнопки «Сохранение отчета») из примера «Введение в C#».

## Разработка интерфейса класса

The image shows a Windows application window titled "Form1". The interface is designed for file search operations and includes the following elements:

- Buttons:**
  - "Чтение" (Read) - located on the left side.
  - "Четкий поиск" (Exact search) - located on the left side.
  - "Нечеткий поиск" (Fuzzy search) - located on the left side.
  - "Сохранение отчета" (Save report) - located at the bottom left.
  - "Выход" (Exit) - located at the bottom right.
- Input Fields:**
  - "Время чтения из файла" (Reading time from file) - text box.
  - "Количество уникальных слов в файле" (Number of unique words in file) - text box.
  - "Слово для поиска" (Search word) - text box.
  - "Время четкого поиска" (Exact search time) - text box.
  - "Максимальное расстояние для нечеткого поиска" (Maximum distance for fuzzy search) - text box.
  - "Количество потоков" (Number of threads) - text box.
  - "Вычисленное количество потоков" (Calculated number of threads) - text box.
  - "Время нечеткого поиска" (Fuzzy search time) - text box.
- Output Area:**
  - A large text area labeled "listBoxResult" for displaying search results.

## Form1

Класс

→ Form

### Поля

- button1
- button2
- buttonApprox
- buttonExit
- buttonSaveReport
- components
- label1
- label2
- label3
- label4
- label5
- label6
- label7
- label8
- listBoxResult
- textBoxApproxTime
- textBoxExactTime
- textBoxFileReadCount
- textBoxFileReadTime
- textBoxFind
- textBoxMaxDist
- textBoxThreadCount
- textBoxThreadCountAll
- words

### Методы

- ArrayThreadTask
- button1\_Click
- button2\_Click
- buttonApprox\_Click
- buttonExit\_Click
- buttonSaveReport\_Click
- Dispose
- Form1
- Form1\_Load
- InitializeComponent
- label1\_Click
- label2\_Click
- label3\_Click
- label4\_Click
- label6\_Click
- textBox1\_TextChanged
- textBox2\_TextChanged
- textBox2\_TextChanged\_1
- textBox3\_TextChanged

### Вложенные типы

## Листинг программы

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

namespace Lab_4
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

            List<string> words = new List<string>();
            private void button1_Click(object sender, EventArgs e)
            {
                OpenFileDialog fd = new OpenFileDialog();
                fd.Filter = "текстовые файлы|*.txt";

                if (fd.ShowDialog() == DialogResult.OK)
                {
                    Stopwatch t = new Stopwatch();
                    t.Start();

                    string text = File.ReadAllText(fd.FileName);
                    char[] sep = new char[] { ' ', '.', ',', '!', '?', '/', '\\t', '\\n' };

                    string[] textArray = text.Split(sep);

                    foreach (string strTemp in textArray)
                    {
                        string str = strTemp.Trim();
                        if (!words.Contains(str)) words.Add(str);
                    }

                    t.Stop();
                    this.textBoxFileReadTime.Text = t.Elapsed.ToString();
                    this.textBoxFileReadCount.Text = words.Count.ToString();
                }
                else
                {
                    MessageBox.Show("Необходимо выбрать файл");
                }
            }

            private void textBox1_TextChanged(object sender, EventArgs e)
            {
            }
        }
    }
}
```

```

private void label1_Click(object sender, EventArgs e)
{

}

private void label2_Click(object sender, EventArgs e)
{

}

private void label3_Click(object sender, EventArgs e)
{

}

private void button2_Click(object sender, EventArgs e)
{
    string word = this.textBoxFind.Text.Trim();

    if (!string.IsNullOrEmpty(word) && words.Count > 0)
    {
        int maxDist;
        if (!int.TryParse(this.textBoxMaxDist.Text.Trim(), out maxDist))
        {
            MessageBox.Show("Необходимо указать максимальное расстояние");
            return;
        }

        if (maxDist < 1 || maxDist > 5)
        {
            MessageBox.Show("Максимальное расстояние должно быть в
диапазоне от 1 до 5");
            return;
        }

        string wordUpper = word.ToUpper();

        List<string> tempList = new List<string>();
        Stopwatch t = new Stopwatch();
        t.Start();

        foreach (string str in words)
        {
            if (MyLib5.Levenshtein.LevenshteinDistance(str.ToUpper(),
wordUpper) <= maxDist)
            {
                tempList.Add(str);
            }
        }
        t.Stop();

        this.textBoxExactTime.Text = t.Elapsed.ToString();
        this.listBoxResult.BeginUpdate();
        this.listBoxResult.Items.Clear();

        foreach (string str in tempList)
        {
            this.listBoxResult.Items.Add(str);
        }
        this.listBoxResult.EndUpdate();
    }
}

```

```

        }
        else
        {
            MessageBox.Show("Необходимо выбрать файл и ввести слово для
поиска");
        }
    }

    private void label4_Click(object sender, EventArgs e)
    {

    }

    private void textBox2_TextChanged(object sender, EventArgs e)
    {

    }

    private void textBox2_TextChanged_1(object sender, EventArgs e)
    {

    }

    private void label6_Click(object sender, EventArgs e)
    {

    }

    private void textBox3_TextChanged(object sender, EventArgs e)
    {

    }

    private void buttonApprox_Click(object sender, EventArgs e)
    {
        string word = this.textBoxFind.Text.Trim();

        if (!string.IsNullOrEmpty(word) && words.Count > 0)
        {
            int maxDist;
            if (!int.TryParse(this.textBoxMaxDist.Text.Trim(), out maxDist))
            {
                MessageBox.Show("Необходимо указать максимальное расстояние");
                return;
            }

            if (maxDist < 1 || maxDist > 5)
            {
                MessageBox.Show("Максимальное расстояние должно быть в
диапазоне от 1 до 5");
                return;
            }

            int ThreadCount;
            if (!int.TryParse(this.textBoxThreadCount.Text.Trim(), out
ThreadCount))
            {
                MessageBox.Show("Необходимо указать количество потоков");
                return;
            }

```

```

        Stopwatch timer = new Stopwatch();
        timer.Start();

        var Result = new List<ParallelSearchResult>();

        var arrayDivList = SubArrays.DivideSubArrays(0, words.Count,
ThreadCount);
        int count = arrayDivList.Count;

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

        for (int i = 0; i < count; i++)
        {
            var tempTaskList = words.GetRange(arrayDivList[i].Min,
arrayDivList[i].Max - arrayDivList[i].Min);
            tasks[i] = new
Task<List<ParallelSearchResult>>(ArrayThreadTask,
            new ParallelSearchThreadParam()
            {
                tempList = tempTaskList,
                maxDist = maxDist,
                ThreadNum = i,
                wordPattern = word
            });
            tasks[i].Start();
        }

        Task.WaitAll(tasks);
        timer.Stop();

        for (int i = 0; i < count; i++)
        {
            Result.AddRange(tasks[i].Result);
        }

        timer.Stop();

        this.textBoxApproxTime.Text = timer.Elapsed.ToString();
        this.textBoxThreadCountAll.Text = count.ToString();

        this.listBoxResult.BeginUpdate();
        this.listBoxResult.Items.Clear();

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

public class ParallelSearchResult
{

```



```

        public string word { get; set; }
        public int dist { get; set; }
        public int ThreadNum { get; set; }
    }

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

    public static List<ParallelSearchResult> ArrayThreadTask(object paramObj)
    {
        ParallelSearchThreadParam param = (ParallelSearchThreadParam)paramObj;

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

        var Result = new List<ParallelSearchResult>();

        foreach (string str in param.tempList)
        {
            int dist = MyLib5.Levenshtein.LevenshteinDistance(str.ToUpper(),
wordUpper);
            if (dist <= param.maxDist)
            {
                ParallelSearchResult temp = new ParallelSearchResult()
                {
                    word = str,
                    dist = dist,
                    ThreadNum = param.ThreadNum
                };
                Result.Add(temp);
            }
        }
        return Result;
    }

    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;
        }
    }

    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;
    }

    private void buttonSaveReport_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|TXT|*.txt";

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

            if (Path.GetExtension(fd.FileName) == ".html")
            {
                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.textBoxFileReadTime.Text +
"</td>");

                b.AppendLine("</tr>");
                b.AppendLine("<tr>");
                b.AppendLine("<td>Количество уникальных слов в файле</td>");
                b.AppendLine("<td>" + this.textBoxFileReadCount.Text +
"</td>");

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

                b.AppendLine("<td>" + this.textBoxMaxDist.Text + "</td>");
            }
        }
    }

```

```

        b.AppendLine("</tr>");
        b.AppendLine("<tr>");
        b.AppendLine("<td>Время четкого поиска</td>");
        b.AppendLine("<td>" + this.textBoxExactTime.Text + "</td>");
        b.AppendLine("</tr>");
        b.AppendLine("<tr>");
        b.AppendLine("<td>Время нечеткого поиска</td>");
        b.AppendLine("<td>" + this.textBoxApproxTime.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.listBoxResult.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);
    }
    else
    {
        b.AppendLine("Отчет: " + ReportFileName);
        b.AppendLine();
        b.AppendLine("Время чтения из файла: " +
this.textBoxFileReadTime.Text);
        b.AppendLine("Количество уникальных слов в файле:" +
this.textBoxFileReadCount.Text);
        b.AppendLine("Слово для поиска:" + this.textBoxFind.Text);
        b.AppendLine("Максимальное расстояние для нечеткого поиска:" +
this.textBoxMaxDist.Text);
        b.AppendLine("Время четкого поиска:" +
this.textBoxExactTime.Text);
        b.AppendLine("Время нечеткого поиска:" +
this.textBoxApproxTime.Text);
        b.AppendLine("Результаты поиска:");

        foreach (var x in this.listBoxResult.Items)
        {
            b.AppendLine(x.ToString() + "\n");
        }

        File.AppendAllText(ReportFileName, b.ToString());
        MessageBox.Show("Отчет сформирован. Файл: " + ReportFileName);
    }
}

private void Form1_Load(object sender, EventArgs e)
{
}

private void buttonExit_Click(object sender, EventArgs e)

```

```

    {
        this.Close();
    }
}

```

## Анализ результатов

Введенные значения:

Чтение

Время чтения из файла

00:00:00.0142279

Количество уникальных слов в файле

1212

Слово для поиска

golang

Четкий поиск

Время четкого поиска

00:00:00.0027983

Максимальное расстояние для нечеткого поиска

3

Нечеткий поиск

Количество потоков

4

Вычисленное количество потоков

4

Время нечеткого поиска

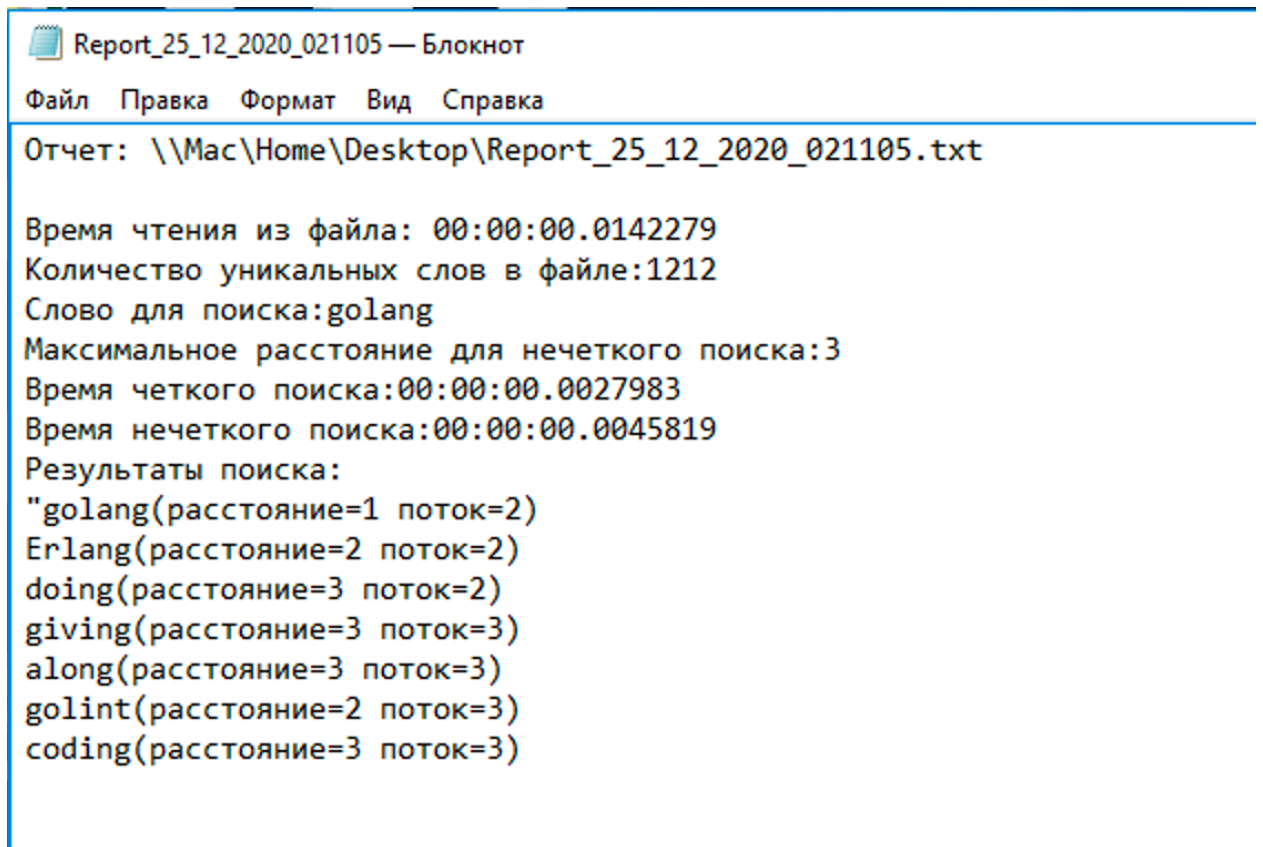
00:00:00.0045819

"\"golang(расстояние=1 поток=2)  
Erlang(расстояние=2 поток=2)  
doing(расстояние=3 поток=2)  
giving(расстояние=3 поток=3)  
along(расстояние=3 поток=3)  
golint(расстояние=2 поток=3)  
coding(расстояние=3 поток=3)

Сохранение отчета

Выход

Отчет в формате “.txt”:



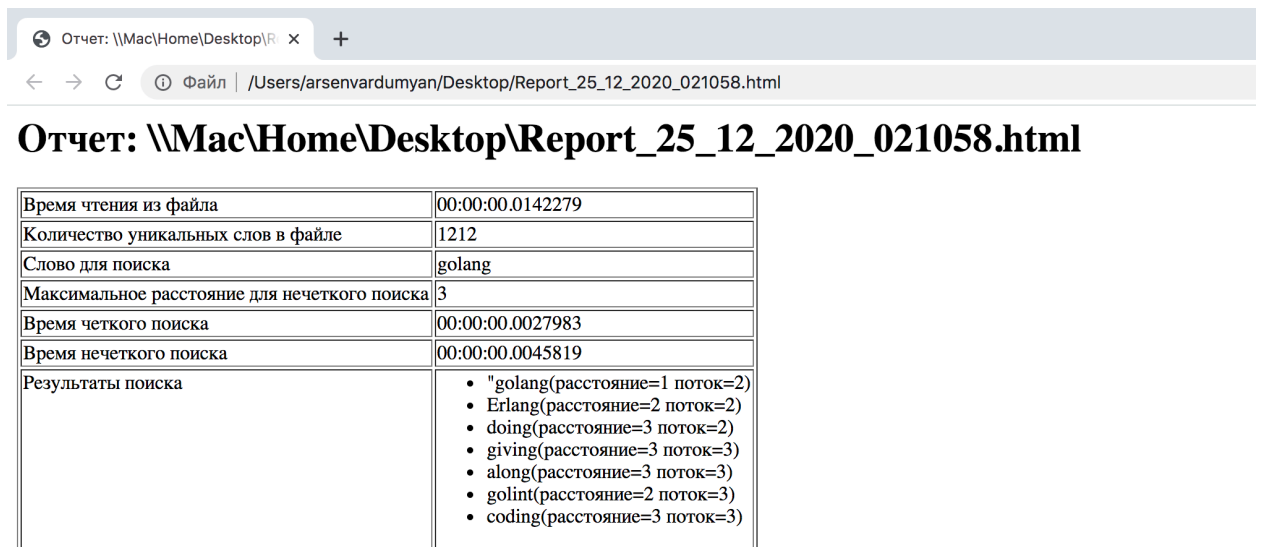
Report\_25\_12\_2020\_021105 — Блокнот

Файл Правка Формат Вид Справка

Отчет: \\Mac\Home\Desktop\Report\_25\_12\_2020\_021105.txt

Время чтения из файла: 00:00:00.0142279  
Количество уникальных слов в файле:1212  
Слово для поиска:golang  
Максимальное расстояние для нечеткого поиска:3  
Время четкого поиска:00:00:00.0027983  
Время нечеткого поиска:00:00:00.0045819  
Результаты поиска:  
"golang(расстояние=1 поток=2)  
Erlang(расстояние=2 поток=2)  
doing(расстояние=3 поток=2)  
giving(расстояние=3 поток=3)  
along(расстояние=3 поток=3)  
golint(расстояние=2 поток=3)  
coding(расстояние=3 поток=3)

Отчет в формате “.html”:



Отчет: \\Mac\Home\Desktop\Report\_25\_12\_2020\_021058.html

Время чтения из файла	00:00:00.0142279
Количество уникальных слов в файле	1212
Слово для поиска	golang
Максимальное расстояние для нечеткого поиска	3
Время четкого поиска	00:00:00.0027983
Время нечеткого поиска	00:00:00.0045819
Результаты поиска	<ul style="list-style-type: none"><li>"golang(расстояние=1 поток=2)</li><li>Erlang(расстояние=2 поток=2)</li><li>doing(расстояние=3 поток=2)</li><li>giving(расстояние=3 поток=3)</li><li>along(расстояние=3 поток=3)</li><li>golint(расстояние=2 поток=3)</li><li>coding(расстояние=3 поток=3)</li></ul>