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

Факультет «Информатика и управление»

Кафедра ИУ5.

Курс «Базовые компоненты интернет-технологий»

Отчет по ЛР4

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 работу с файлами.

1.

Программа должна быть разработана в виде приложения Windows Forms на языке C#. По желанию вместо Windows Forms возможно использование WPF.

2.

Добавить кнопку, реализующую функцию чтения файла в список слов List<string>.

3.

Для выбора имени файла используется класс OpenFileDialog, который открывает диалоговое окно с выбором файла. Ограничить выбор только файлами с расширением «.txt».

4.

Для чтения из файла рекомендуется использовать статический метод ReadAllText() класса File (пространство имен System.IO). Содержимое файла считывается методом ReadAllText() в виде одной строки, далее делится на слова с использованием метода Split() класса string. Слова сохраняются в список List<string>.

5.

При сохранении слов в список List<string> дубликаты слов не записываются. Для проверки наличия слова в списке используется метод Contains().

6.

Вычислить время загрузки и сохранения в список с использованием класса Stopwatch (пространство имен System.Diagnostics). Вычисленное время вывести на форму в поле ввода (TextBox) или надпись (Label).

7.

Добавить на форму поле ввода для поиска слова и кнопку поиска. При нажатии на кнопку поиска осуществлять поиск введенного слова в списке. Слово считается найденным, если оно входит в элемент списка как подстрока (метод Contains() класса string).

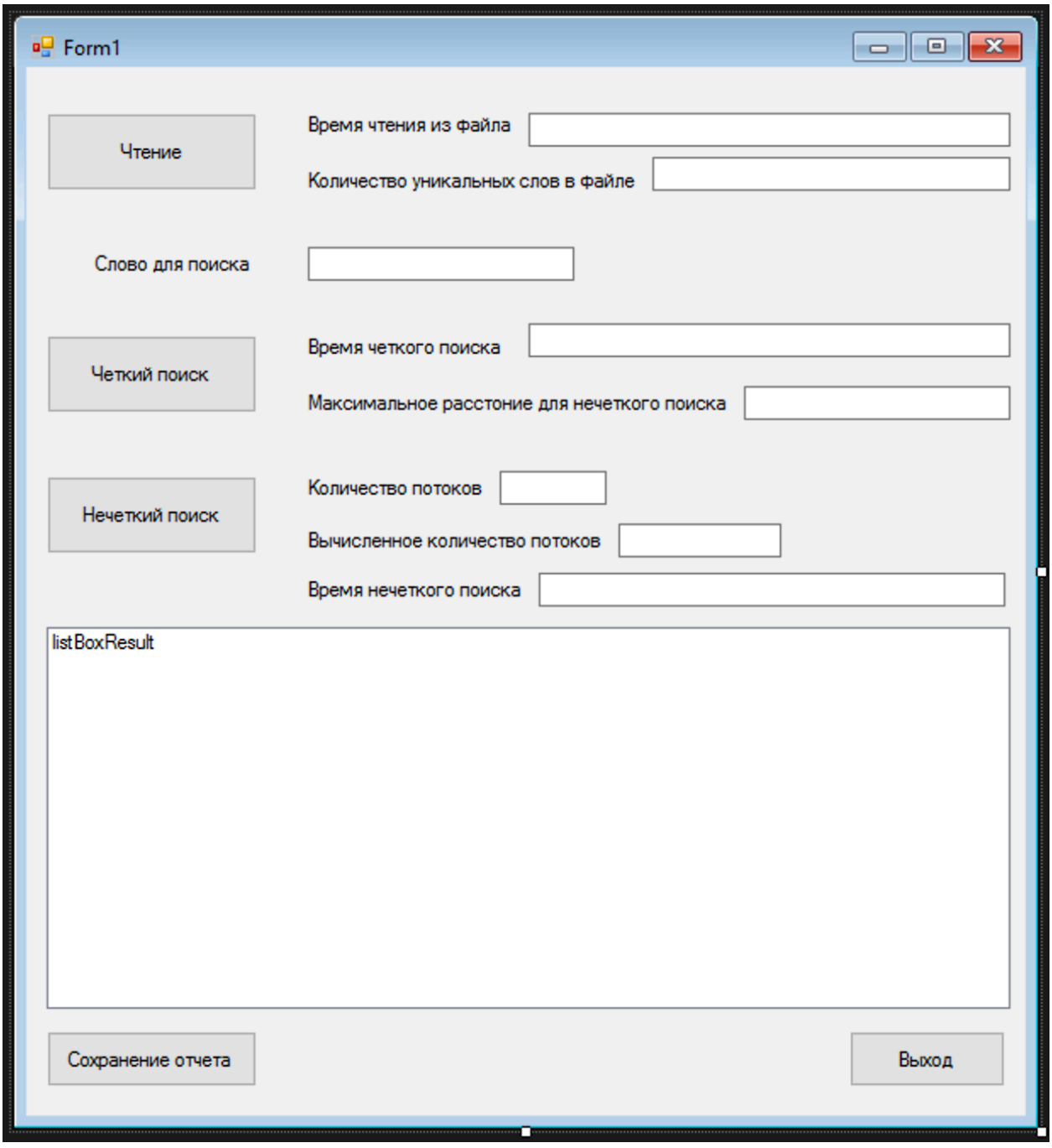
8.

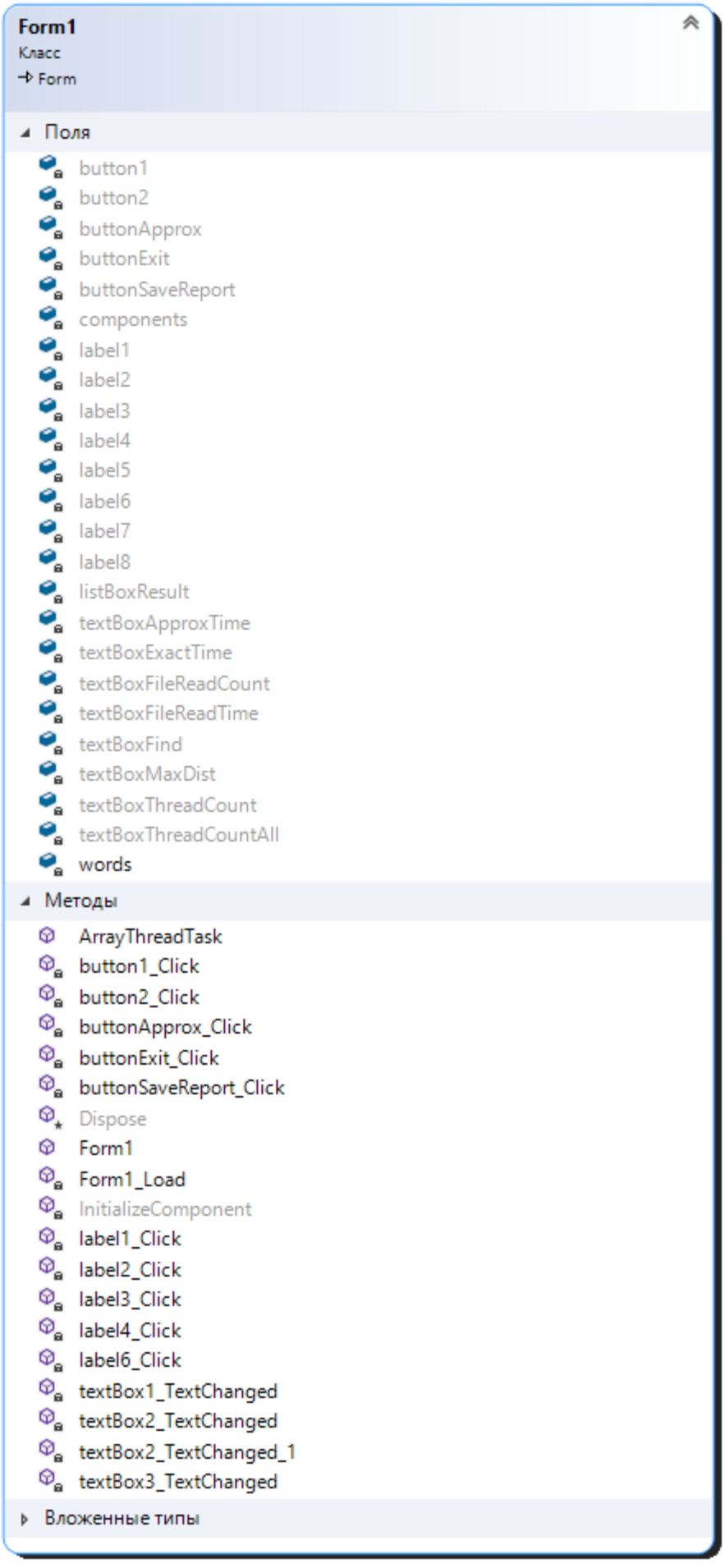
Добавить на форму список (ListBox). Найденные слова выводить в список с использованием метода «название_списка.Items.Add()». Вызовы метода «название_списка.Items.Add()» должны находится между вызовами методов «название_списка.BeginUpdate()» и «название_списка. EndUpdate()».

9.

Вычислить время поиска с использованием класса Stopwatch. Вычисленное время вывести на форму в поле ввода (TextBox) или надпись (Label).

Разработка интерфейса класса




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.IsNullOrWhiteSpace(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.IsNullOrWhiteSpace(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
        {
```

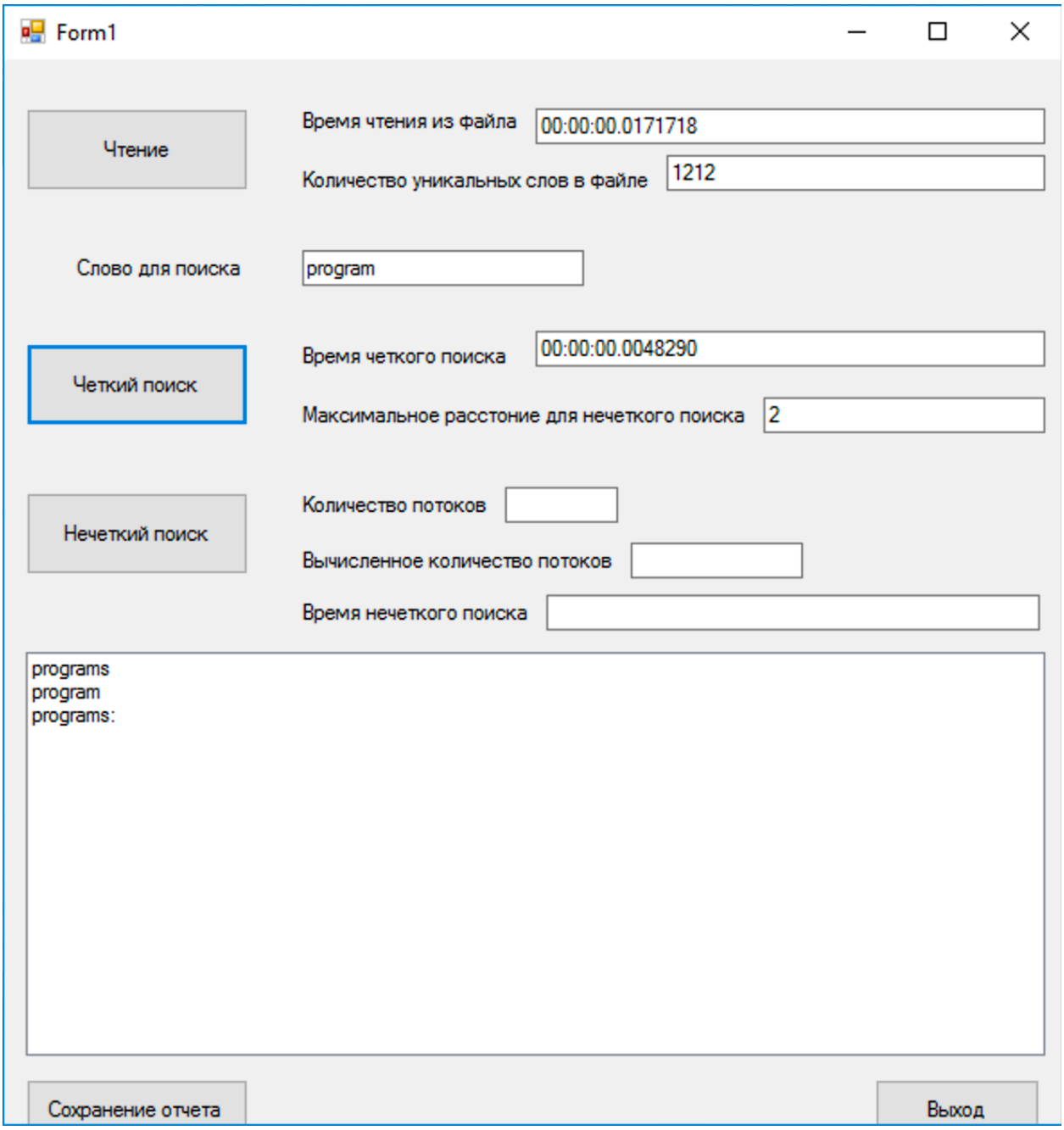
```
            public string word { get; set; }
            public int dist { get; set; }
            public int ThreadNum { get; set; }
        }

        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
```

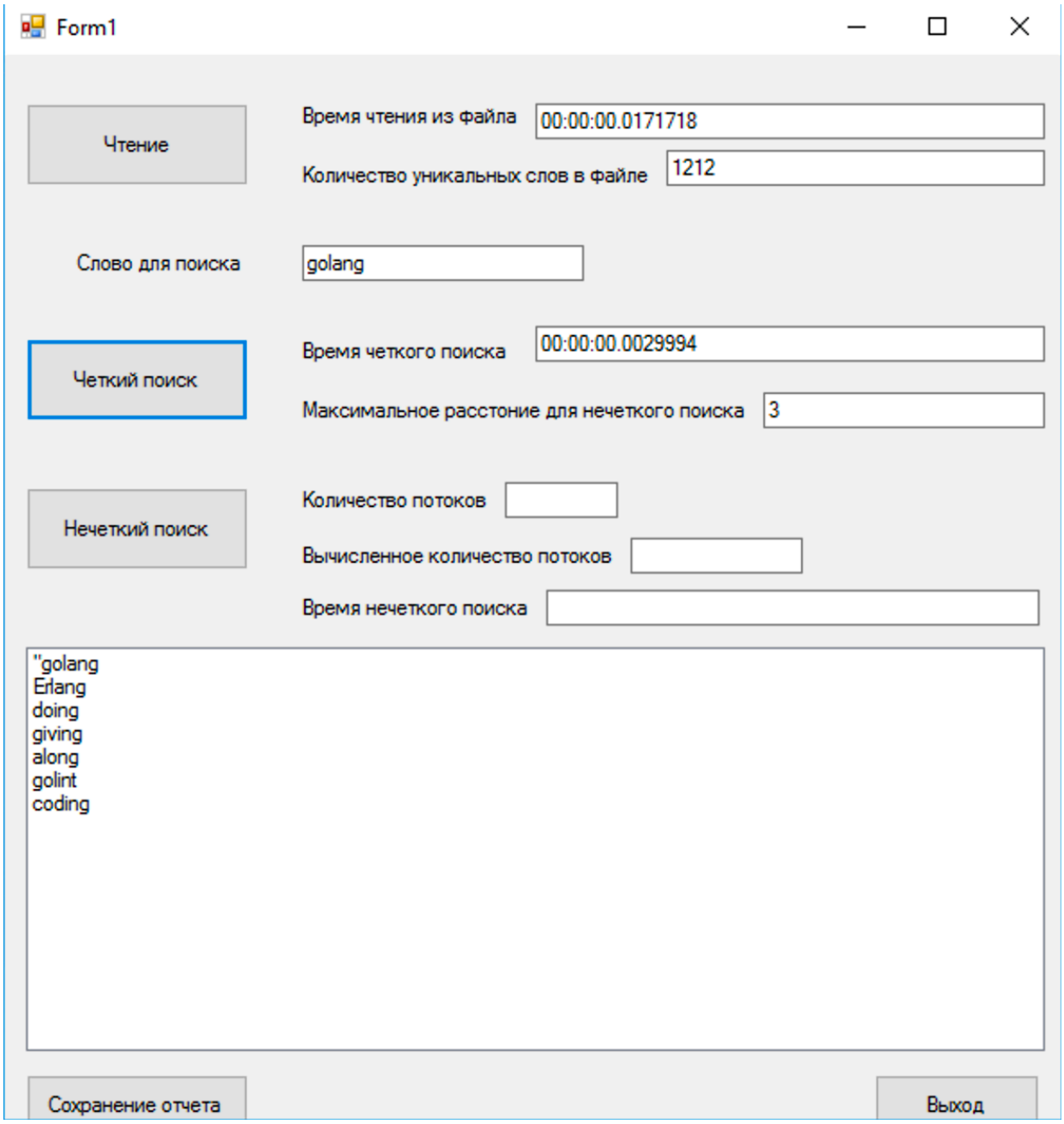
```
                {
                    int delta = (endIndex - beginIndex) / subArraysCount;
                    int currentBegin = beginIndex;

                    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




Form1
Чтение
Время чтения из файла
00:00:00.0171718
Количество уникальных слов в файле
1212
Слово для поиска
golang
Четкий поиск
Время четкого поиска
00:00:00.0029994
Максимальное расстоние для нечеткого поиска
3
Нечеткий поиск
Количество потоков
Вычисленное количество потоков
Время нечеткого поиска
''golang
Erlang
doing
giving
along
golint
coding
Сохранение отчета
Выход