Московский государственный технический университет
им. Н.Э.Баумана

Факультет ИУ

Кафедра ИУ5

Лабораторные работы №4,5 по курсу C#

Группа: *ИУ5-32*

Миронов С.В.

Лабораторные работы № 4,5

Лаб 4: 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 (Windows Presentation Foundation).

2. Добавить кнопку, реализующую функцию чтения текстового 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.
10 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).

Лаб 5: Разработать программу, реализующую вычисление расстояния Левенштейна с использованием алгоритма Вагнера-Фишера.

1. Программа должна быть разработана в виде библиотеки классов на языке C#.

2. Использовать самый простой вариант алгоритма без оптимизации.

3. Дополнительно возможно реализовать вычисление расстояния Дамерау-Левенштейна (с учетом перестановок соседних символов).

4. Модифицировать предыдущую лабораторную работу, вместо поиска подстроки используется вычисление расстояния Левенштейна.

5. Предусмотреть отдельное поле ввода для максимального расстояния. Если расстояние Левенштейна между двумя строками больше максимального, то строки считаются несовпадающими и не выводятся в список результатов.

Код программы:

Статическая библиотека:

```
namespace ab5library
{
    public class EditDistanse
    {

        public static int Distance(string str1Param, string str2Param)
        {
            if ((str1Param == null) || (str2Param == null)) return -1;
            int str1Len = str1Param.Length;
            int str2Len = str2Param.Length;
            if ((str1Len == 0) && (str2Len == 0)) return 0;
            if (str1Len == 0) return str2Len;
            if (str2Len == 0) return str1Len;
            string str1 = str1Param.ToUpper();
            string str2 = str2Param.ToUpper();
            int[,] matrix = new int[str1Len + 1, str2Len + 1];
            for (int i = 0; i <= str1Len; i++) matrix[i, 0] = i;
            for (int j = 0; j <= str2Len; j++) matrix[0, j] = j;
            for (int i = 1; i <= str1Len; i++)
            {
                for (int j = 1; j <= str2Len; j++)
                {
                    int symbEqual = ((str1.Substring(i - 1, 1) == str2.Substring(j - 1, 1)) ? 0 : 1);
                    int ins = matrix[i, j - 1] + 1; //Добавление
                    int del = matrix[i - 1, j] + 1; //Удаление
                    int subst = matrix[i - 1, j - 1] + symbEqual;
                    matrix[i, j] = Math.Min(Math.Min(ins, del), subst);
                    if ((i > 1) && (j > 1) &&
                    (str1.Substring(i - 1, 1) == str2.Substring(j - 2, 1)) &&
                    (str1.Substring(i - 2, 1) == str2.Substring(j - 1, 1)))
                    {
                        matrix[i, j] = Math.Min(matrix[i, j], matrix[i - 2, j - 2] + symbEqual);
                    }
                }
            }
            return matrix[str1Len, str2Len];
        }
    }
}
```

Приложение Windows Forms:

```
namespace lab4.csh
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }


        private void buttonLoadFile_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[] separators = new char[] { ' ', '.', ',', '!', '?', '/', '\t', '\n' };
                string[] textArray = text.Split(separators);
                foreach (string strTemp in textArray)
                {
                    string str = strTemp.Trim();
                    if (!listBoxResult.Items.Contains(str)) listBoxResult.Items.Add(str);
                }
                t.Stop();
                this.textBoxFileReadTime.Text = t.Elapsed.ToString();
                this.textBoxFileReadCount.Text = listBoxResult.Items.Count.ToString();
            }
            else
            {
                MessageBox.Show("Необходимо выбрать файл");
            }
        }
```

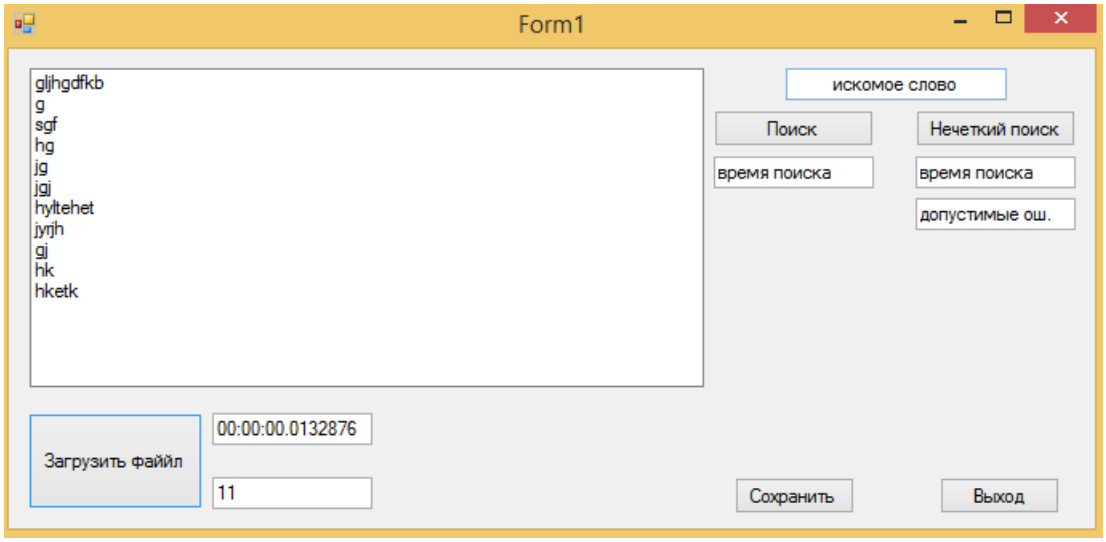
```
private void buttonExact_Click(object sender, EventArgs e)
{
    string word = this.textBoxFind.Text.Trim();
    if (!string.IsNullOrWhiteSpace(word) && listBoxResult.Items.Count > 0)
    {
        string wordUpper = word.ToUpper();
        List<string> tempList = new List<string>();
        Stopwatch t = new Stopwatch();
        t.Start();
        foreach (string str in listBoxResult.Items)
        {
            if (str.ToUpper().Contains(wordUpper))
            {
                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 buttonExit_Click(object sender, EventArgs e)
{
    this.Close();
}
```

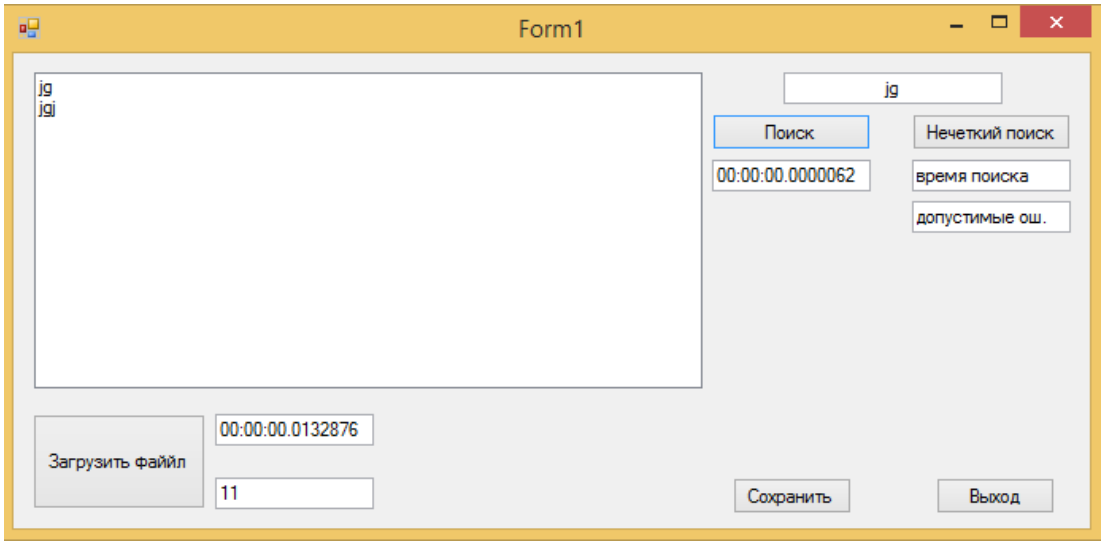
```
        private void buttonSaveReport_Click(object sender, EventArgs e)
        {
            string TempReportFileName = "Report_" + DateTime.Now.ToString("dd_MM_yyyy_hhmmss");
            //Диалог сохранения файла отчета
            SaveFileDialog fd = new SaveFileDialog();
            fd.FileName = TempReportFileName;
            fd.DefaultExt = ".html";
            fd.Filter = "HTML Reports|*.html";
            if (fd.ShowDialog() == DialogResult.OK)
            {
                string ReportFileName = fd.FileName;
                //Формирование отчета
                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.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>");
```

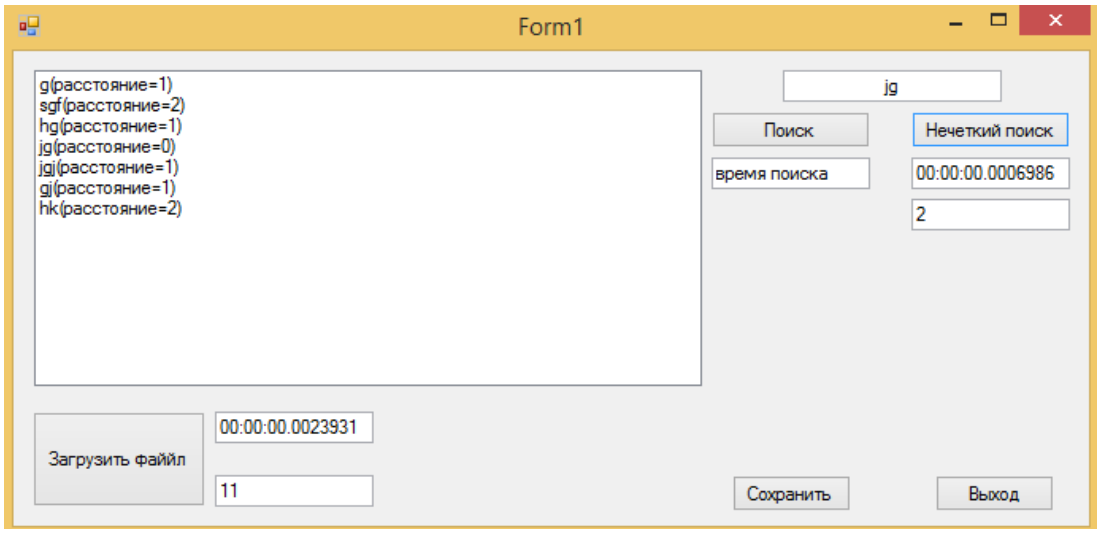
```
                b.AppendLine("<td>Количество уникальных слов в файле</td>");
                b.AppendLine("<td>" + this.textBoxFileReadCount.Text + "</td>");
                b.AppendLine("</tr>");
                b.AppendLine("<tr>");
                b.AppendLine("<td>Слово для поиска</td>");
                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>");
                //Сохранение файла
                File.AppendAllText(ReportFileName, b.ToString());
                MessageBox.Show("Отчет сформирован. Файл: " + ReportFileName);
            }

        }
```

```
        private void buttonApprox_Click(object sender, EventArgs e)
        {
            string word = this.textBoxFind.Text.Trim();
            if (!string.IsNullOrWhiteSpace(word) && listBoxResult.Items.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<Tuple<string, int>> tempList = new List<Tuple<string, int>>();
                Stopwatch t = new Stopwatch();
                t.Start();
                foreach (string str in listBoxResult.Items)
                {
                    //Вычисление расстояния Дамерау-Левенштейна
                    int dist = abSlibrary.EditDistanse.Distance(str.ToUpper(), wordUpper);
                    //Если расстояние меньше порогового, то слово добавляется в результат
                    if (dist <= maxDist)
                    {
                        tempList.Add(new Tuple<string, int>(str, dist));
                    }
                }

                t.Stop();
                this.textBoxApproxTime.Text = t.Elapsed.ToString();
                this.listBoxResult.BeginUpdate();
                this.listBoxResult.Items.Clear();
                foreach (var x in tempList)
                {
                    string temp = x.Item1 + "(расстояние=" + x.Item2.ToString() + ")";
                    this.listBoxResult.Items.Add(temp);
                }
                this.listBoxResult.EndUpdate();
            }
            else
            {
                MessageBox.Show("Необходимо выбрать файл и ввести слово для поиска");
            }
        }

    }
```

Результат выполнения программы:


Form1
gljhgdfkb
g
sgf
hg
jg
jgj
hyltehet
jyrjh
gj
hk
hketk
искомое слово
Поиск
Нечеткий поиск
время поиска
время поиска
допустимые ош.
Загрузить файл
00:00:00.0132876
11
Сохранить
Выход


Form1
jg
jgj
jg
Поиск
Нечеткий поиск
00:00:00.0000062
время поиска
допустимые ош.
Загрузить файл
00:00:00.0132876
11
Сохранить
Выход


Form1
g(расстояние=1)
sgf(расстояние=2)
hg(расстояние=1)
jg(расстояние=0)
jgj(расстояние=1)
gj(расстояние=1)
hk(расстояние=2)
jg
Поиск
Нечеткий поиск
время поиска
00:00:00.0006986
2
Загрузить файл
00:00:00.0023931
11
Сохранить
Выход