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

Факультет «Информатика и управление»

Кафедра «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчет по лабораторной работе №5

Выполнил:
студент группы ИУ5-32
Кудрявцев Сергей
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.С.
Подпись и дата:

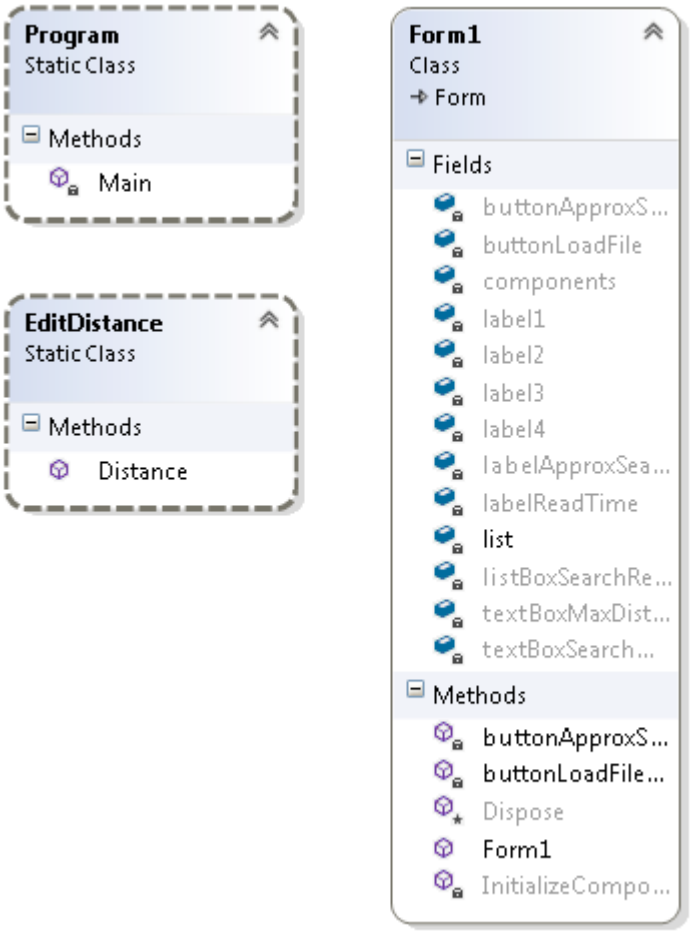
г. Москва, 2018 г.

**Описание задания:**

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.

**Диаграмма классов:**



## Текст программы:

**Program.cs:**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Lab5
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

**EditDistance.cs:**

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

namespace Lab5
{
    public static class EditDistance
    {
        public static int Distance(string str1Param, string str2Param)
        {
            if ((str1Param == null) || (str2Param == null)) return -1;

            int str1Len = str1Param.Length;
            int str2Len = str2Param.Length;

            //Если хотя бы одна строка пустая,
            //возвращается длина другой строки
            if ((str1Len == 0) && (str2Len == 0)) return 0;
            if (str1Len == 0) return str2Len;
            if (str2Len == 0) return str1Len;

            //Приведение строк к верхнему регистру
            string str1 = str1Param.ToUpper();
            string str2 = str2Param.ToUpper();

            //Объявление матрицы
            int[,] matrix = new int[str1Len + 1, str2Len + 1];

            //Инициализация нулевой строки и нулевого столбца матрицы
            for (int i = 0; i <= str1Len; i++) matrix[i, 0] = i;
            for (int j = 0; j <= str2Len; j++) matrix[0, j] = j;

            //Вычисление расстояния Дамерау-Левенштейна
            for (int i = 1; i <= str1Len; i++)
            {
                for (int j = 1; j <= str2Len; j++)
                {
                    //Эквивалентность символов, переменная symbEqual
```

```
                    //соответствует m(s1[i],s2[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; //Замена
                                                                  //Элемент матрицы вычисляется
                                                                  //как минимальный из трех случаев
                    matrix[i, j] = Math.Min(Math.Min(ins, del), subst);

                    //Дополнение Дамерау по перестановке соседних символов
                    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);
                    }
                }
            }
            //Возвращается нижний правый элемент матрицы
            return matrix[str1Len, str2Len];
        }
    }
}
```

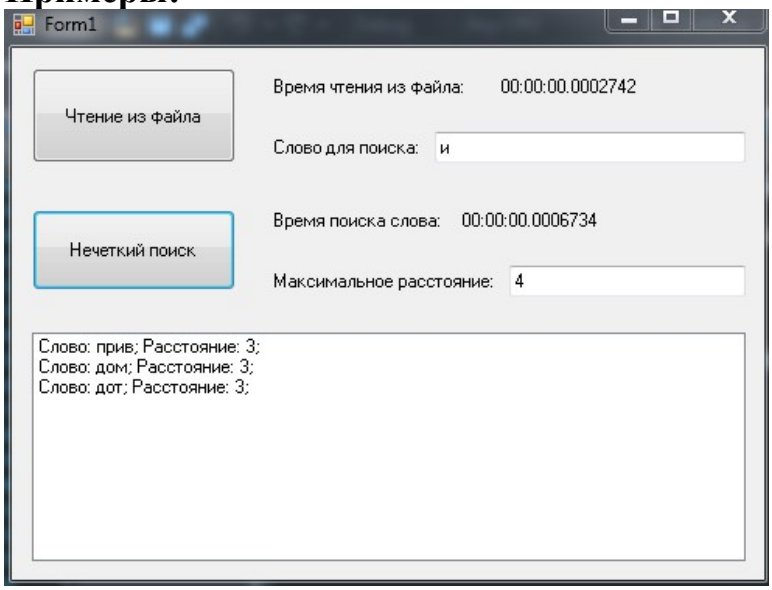
**Form1.cs:**

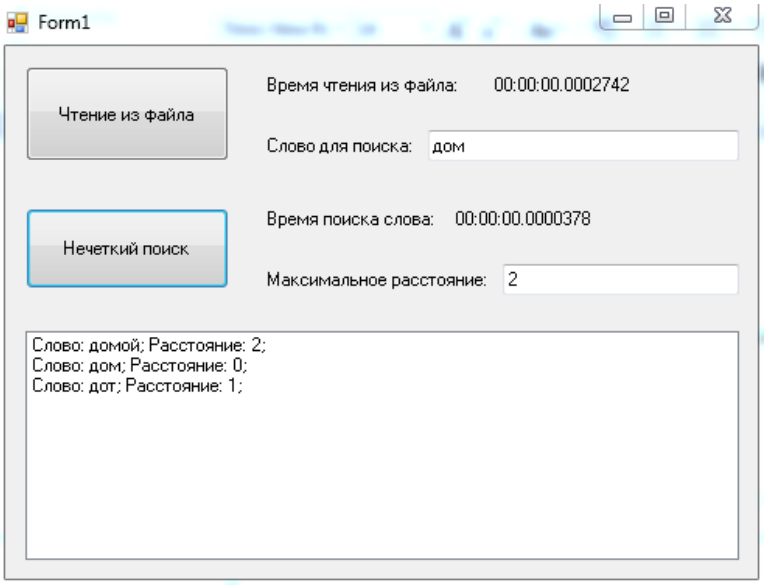
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Diagnostics;
using System.IO;

namespace Lab5
{
    public partial class Form1 : Form
    {
        List<string> list = new List<string>();

        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)
            {
                //хммммммм
                list.Clear();

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

                //У меня на Windows очень странно читаются файлы. Нормально работает только с
юникодом
                string text = File.ReadAllText(fd.FileName, Encoding.Unicode);
```

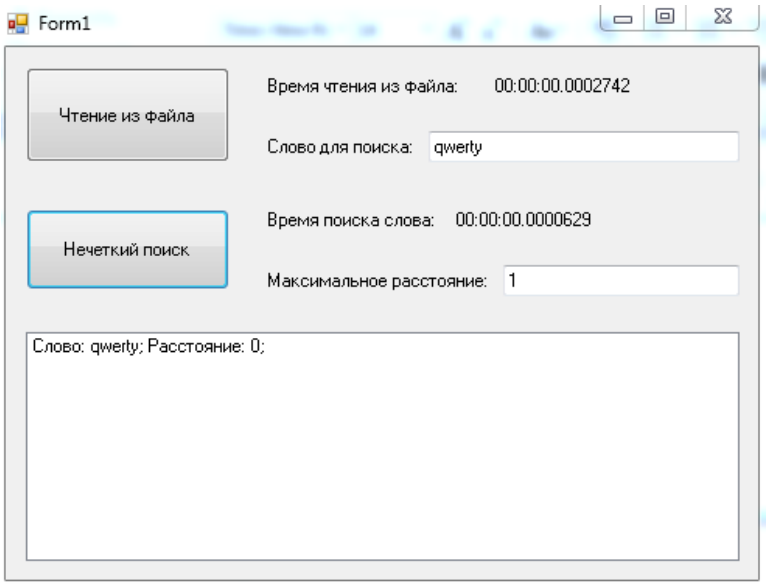
```
            char[] separators = new char[] { ' ', '.', ',', '!', '/', '\t', '\n' };

            string[] textArray = text.Split(separators);
            foreach (string strTemp in textArray)
            {
                string str = strTemp.Trim();

                if (!list.Contains(str))
                    list.Add(str);
            }

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

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

        int maxDist;
        if (!string.IsNullOrWhiteSpace(word) && list.Count > 0)
        {
            if (!int.TryParse(this.textBoxMaxDistance.Text, 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 list)
            {
                int tempDist;
                if((tempDist = EditDistance.Distance(str.ToUpper(), wordUpper)) <= maxDist)
                {
                    tempList.Add("Слово: "+ str + "; Расстояние: " + tempDist.ToString()
+";");

                }
            }

            t.Stop();
            this.labelApproxSearchTime.Text = t.Elapsed.ToString();

            this.listBoxSearchResult.BeginUpdate();

            this.listBoxSearchResult.Items.Clear();

            foreach (string str in tempList)
            {
                this.listBoxSearchResult.Items.Add(str);
            }
```

```
                this.listBoxSearchResult.EndUpdate();
            }
            else
            {
                MessageBox.Show("Необходимо выбрать файл или ввести слово для поиска");
            }
        }
    }
}
```

**Примеры:**




Form1
Чтение из файла
Время чтения из файла: 00:00:00.0002742
Слово для поиска: дом
Нечеткий поиск
Время поиска слова: 00:00:00.0000378
Максимальное расстояние: 2
Слово: домой; Расстояние: 2;
Слово: дом; Расстояние: 0;
Слово: дот; Расстояние: 1;


Form1
Чтение из файла
Время чтения из файла: 00:00:00.0002742
Слово для поиска: qwerty
Нечеткий поиск
Время поиска слова: 00:00:00.0000629
Максимальное расстояние: 1
Слово: qwerty; Расстояние: 0;