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

Факультет «Информатика и управление»

Кафедра ИУ5.

Курс «Базовые компоненты интернет-технологий»

Отчет по лабораторной работе №5

Выполнил:

студент группы ИУ5-31Б

Федоров Иван

Подпись и дата:

Проверил:

преподаватель каф. ИУ5

Гапанюк Юрий Евгеньевич

Подпись и дата:

г. Москва, 2020 г.

Постановка задачи

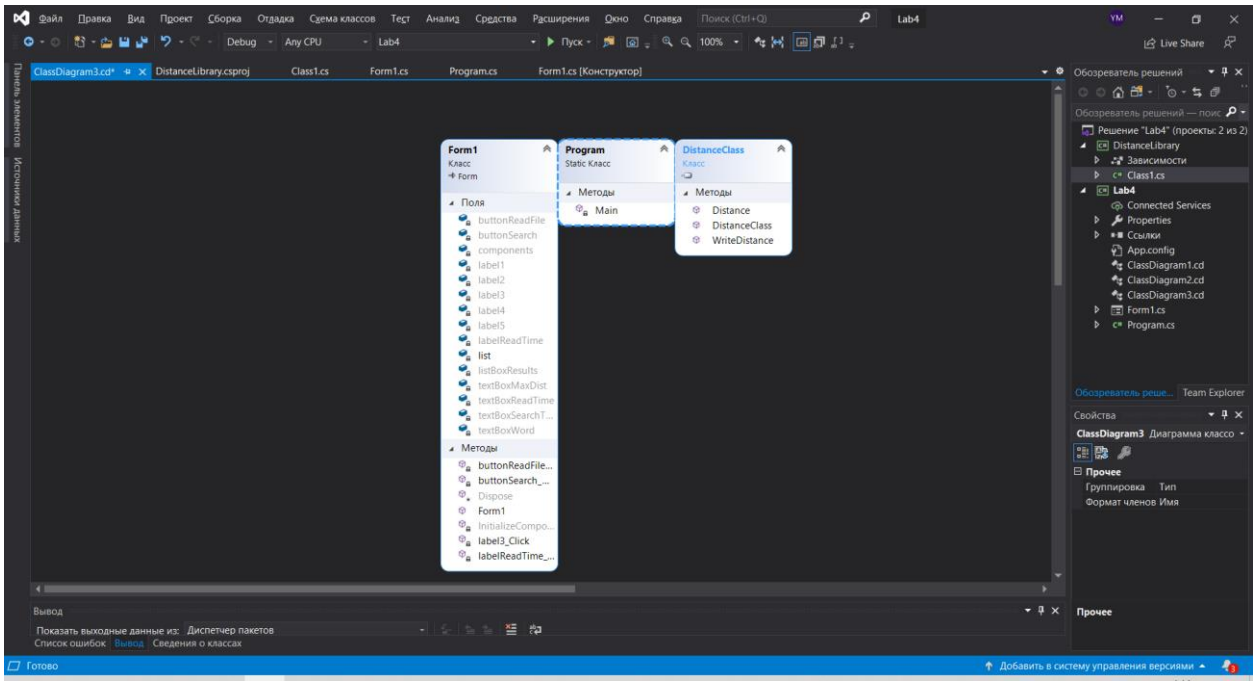
Разработать программу, реализующую вычисление расстояния Левенштейна с использованием алгоритма Вагнера-Фишера.

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.

Контрольные вопросы:

1. Что такое расстояние Левенштейна?
2. Приведите пример вычисления расстояния Левенштейна.
3. Что такое расстояние Дамерау-Левенштейна?
4. Приведите пример вычисления расстояния Дамерау-Левенштейна.
5. В чем состоит поправка Дамерау?
6. Для чего используется транспозиция в поправке Дамерау?
7. Объясните алгоритм Вагнера-Фишера вычисления расстояния Дамерау-Левенштейна (на основе матрицы).
8. Как инициализируются начальные значения в матрице при вычислении расстояния Левенштейна? Почему?
9. Как задать различные веса для операций удаления, добавления и замены?
10. Как осуществить интеграцию разработанного метода в приложение Windows Forms.

Разработка интерфейса класса



Листинг программы

```
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 DistanceLibrary;

namespace Lab4
{

    public partial class Form1 : Form
    {

        public Form1()
        {
            InitializeComponent();
        }

        List<string> list = new List<string>();
        private void buttonReadFile_Click(object sender, EventArgs e)
        {
            //выбираем файл через стандартное диалоговое окно
            OpenFileDialog txt = new OpenFileDialog();
            //Фильтр на TXT файлы
            txt.Filter = "Только текстовые файлы|*.txt";
            //если пользователь не выбрал файл
```

```
    if (txt.ShowDialog() != DialogResult.OK)
    {
        MessageBox.Show("Необходимо выбрать файл");
        return;
    }
    //разделители слов
    char[] separators = { '?', '.', ',', '!', '*','/', ' ', '\t', '\n'};
    Stopwatch timer = new Stopwatch();
    timer.Start();
    //считывание текста из файла
    string text = File.ReadAllText(txt.FileName);
    //удаление ненужных символов в начале и конце текста
    text = text.Trim(separators);
    //разделение строки на слова
    string[] textArray = text.Split(separators);
    //Запись слов в список
    foreach (string word in textArray)
    {
        if (!word.Contains("-"))
        //проверка на неповторяющиеся слова
            if (!list.Contains(word))
                list.Add(word);
    }
    timer.Stop();
    //запись даных из таймера
    this.textBoxReadTime.Text = timer.Elapsed.ToString();
    //this.listBoxFile.BeginUpdate();
    //this.listBoxFile.Items.Clear();
    //foreach (string word in list)
    //{
    //    this.listBoxFile.Items.Add(word);
    //}
    //this.listBoxFile.EndUpdate();

}

private void buttonSearch_Click(object sender, EventArgs e)
{
    //запись слова и его очистка
    string word = this.textBoxWord.Text.Trim();
    //Подтягиваем буквы вверх, чтобы не было проблем с регистром
    word = word.ToUpper();
    string MaxDiststr = this.textBoxMaxDist.Text.Trim();
    //запуск листбокса
    this.listBoxResults.BeginUpdate();
    this.listBoxResults.Items.Clear();
    Stopwatch timer = new Stopwatch();
    if (!string.IsNullOrWhiteSpace(MaxDiststr))
    {
        int MaxDist = int.Parse(MaxDiststr);
        if (!string.IsNullOrWhiteSpace(word) && list.Count > 0)
        {
            //Проверка на случай отсутствия совпадений
            bool NoMatches = true;
            //запуск таймера
            timer.Start();

            foreach (string w in list)
            {
                //идёт проверка слов в верхнем регистре
                if (DistanceClass.Distance(w.ToUpper(), word) <= MaxDist)
                {
                    this.listBoxResults.Items.Add(w);
                    NoMatches = false;
                }
```

```
                    }
                    //остановка таймера
                    timer.Stop();
                    //Если совпадений всё же не нашлось
                    if (NoMatches) this.listBoxResults.Items.Add("Нет сопадений");
                }
                else
                {
                    MessageBox.Show("Необходимо отрыть файл и выбрать слово для поиска");
                }
            }
            else
            {
                MessageBox.Show("Необходимо ввести максимальное расстояние");
            }
            this.listBoxResults.EndUpdate();
            //запись даных из таймера
            this.textBoxSearchTime.Text = timer.Elapsed.ToString();
        }

        private void label3_Click(object sender, EventArgs e)
        {

        }

        private void labelReadTime_Click(object sender, EventArgs e)
        {

        }
    }
}
```

```
using System;

namespace DistanceLibrary
{
    public class DistanceClass
    {
        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) return str2Len;
            if (str2Len == 0) return str1Len;

            string str1 = str1Param.ToUpper();
            string str2 = str2Param.ToUpper();

            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;
            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; //Замена
                    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];
        }
        public static void WriteDistance(string str1Param, string str2Param)
        {
            int d = Distance(str1Param, str2Param);
            Console.WriteLine("'" + str1Param + "','" + str2Param + "' -> " +
d.ToString());
        }
    }
}
```

Анализ результатов