Московский государственный технический университет им. Н.Э. Баумана
Факультет «Информатика и системы управления»

Кафедра ИУ5. Группа 31.

Отчет по лабораторной работе 5

Выполнила:
студентка группы ИУ5-31
Качанюк Татьяна
Подпись и дата:

Проверил:
Гапанюк Ю.Е.
Подпись и дата:

г. Москва, 2017 г.

1. **Задание**

   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 ДамерауЛевенштейна (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.

2. **Текст программы**

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.IO;
using System.Diagnostics;

namespace LAB5
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }

      List<string> list = new List<string>(); // Список слов
```

```
    private void button1_Click(object sender, EventArgs e)
    {
        OpenFileDialog fd = new OpenFileDialog();
        fd.Filter = "текстовые файлы|*.txt";
        if (fd.ShowDialog() == DialogResult.OK)
        {
            //Чтение файла в виде строки
            string text = File.ReadAllText(fd.FileName);
            //Разделительные символы для чтения из файла
            char[] separators = new char[] { ' ', '.', ',', '!', '?', '/', '\t', '\n' };
            string[] textArray = text.Split(separators);
            foreach (string strTemp in textArray)
            {
                //Удаление пробелов в начале и конце строки
                string str = strTemp.Trim();
                //Добавление строки в список, если строка не содержится в списке
                if (!list.Contains(str)) list.Add(str);
            }

        }
        else
        {
            MessageBox.Show("Необходимо выбрать файл");
        }

    }

    private void button2_Click(object sender, EventArgs e)
    {
        //Слово для поиска
        string word = this.textBox1.Text.Trim();
        //Если слово для поиска не пусто
        if (!string.IsNullOrWhiteSpace(word) && list.Count > 0)
        {

            int Max = Convert.ToInt32(textBox2.Text);
            if (Max < 1 || Max > 6) MessageBox.Show("Значение не выходит в
границы 1 < x < 6");
            //Слово для поиска в верхнем регистре
            string wordUpper = word.ToUpper();

            //Временные результаты поиска
            List<Tuple<string, int>> tempList = new List<Tuple<string, int>>();
```

```
            foreach (string str in list)
            {
                // Левинштейн
                int distance = Levinstain(str.ToUpper(), wordUpper);
                if(distance <= Max) tempList.Add(new Tuple<string, int>(str,
distance));

            }

            this.listBox1.BeginUpdate();
            //Очистка списка
            this.listBox1.Items.Clear();

            //Вывод результатов поиска
            foreach (var var in tempList)
            {
               string tempStr = var.Item1 + "(" + var.Item2.ToString() + ")";
               this.listBox1.Items.Add(tempStr);
            }
            this.listBox1.EndUpdate();
         }
         else
         {
            MessageBox.Show("Необходимо выбрать файл и ввести слово для
поиска");
         }

      }

      private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
      {

      }

      private void textBox1_TextChanged(object sender, EventArgs e)
      {

      }

      public static int Levinstain(string str1Param, string str2Param)
      {

         if ((str1Param == null) || (str2Param == null)) return -1;
         int str1Len = str1Param.Length;
         int str2Len = str2Param.Length;
```

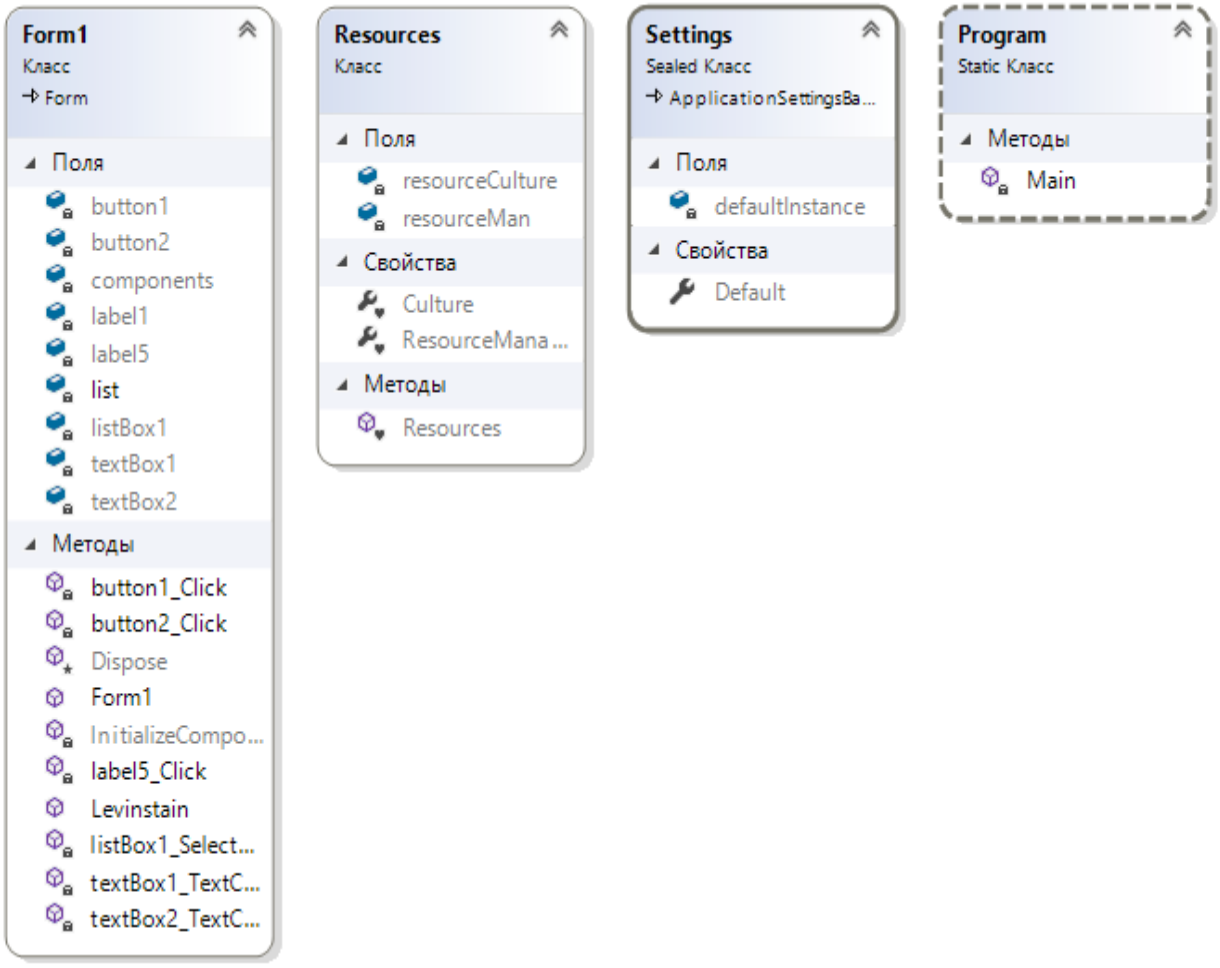
```
        //Если хотя бы одна строка пустая, возвращается длина другой строки
        if ((str1Len == 0) && (str2Len == 0)) return 0;
        if (str1Len == 0) return str2Len;
        if (str2Len == 0) return str1Len;
        //Приведение строк к верхнему регистру
        string str1 = str1Param.ToUpper();
        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];
    }

    private void textBox2_TextChanged(object sender, EventArgs e)
```

```
        {

        }

        private void label5_Click(object sender, EventArgs e)
        {

        }
    }
}
```

Program.cs

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

namespace LAB4
{
    static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

3. Диаграмма классов



4. Результаты выполнения программы