Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчет по лабораторной работе №4 «Вычисление расстояния Левенштейна»

Выполнил:
студент группы ИУ5-33
Сергеев МЮ
Подпись и дата:
29.12.20

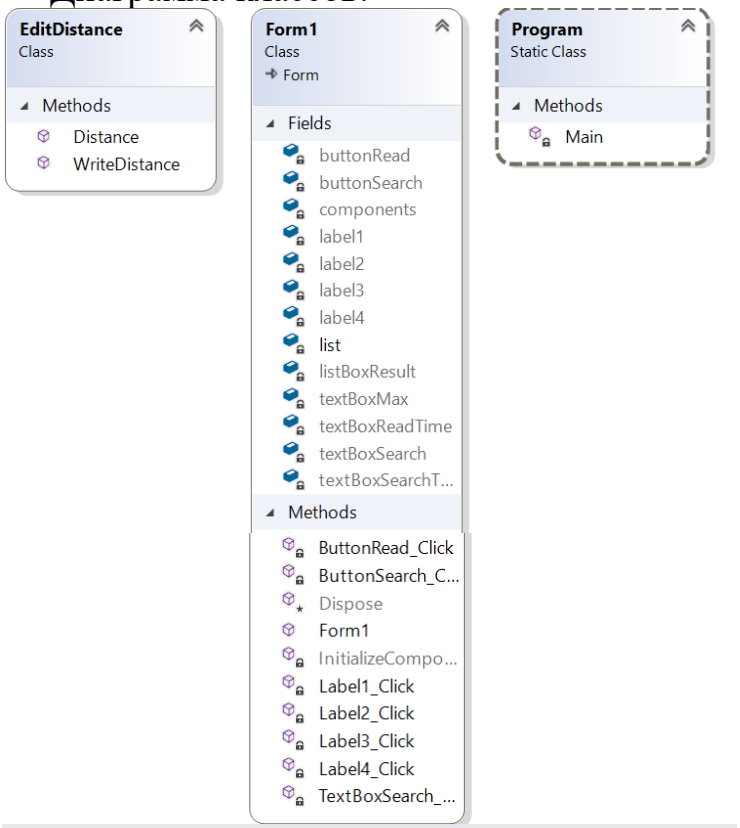
Проверил:
Подпись и дата:

Москва, 2020 г.

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 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.Windows.Forms;
```

```
namespace Lab5
{
   static class Program
   {
      /// <summary>
      /// Главная точка входа для приложения.
      /// </summary>
      [STAThread]
      static void Main()
      {
         Application.EnableVisualStyles();
         Application.SetCompatibleTextRenderingDefault(false);
         Application.Run(new Form1());
      }
   }
}
```

Form1.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Diagnostics;
using System.Windows.Forms;

namespace Lab5
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }
      List<string> list = new List<string>();  // Лист для записи слов в из файла
      private void ButtonRead_Click(object sender, EventArgs e)
      {
         OpenFileDialog fd = new OpenFileDialog();
         fd.Filter = "текстовые файлы|*.txt";
         if (fd.ShowDialog() == DialogResult.OK)
         {
            Stopwatch t = new Stopwatch();  // Таймер
            t.Start();
            string text = File.ReadAllText(fd.FileName);  // Чтение файла в строку
            string[] textWords = text.Split(new char[] { ' ', ',', '.', '"', ':', '[', ']', '!', '?', '/', '\t', '\n' },
StringSplitOptions.RemoveEmptyEntries);  //Разделительные символы чтения файла
            foreach (string strTemp in textWords)  // Удаление пробелов в начале и конце
строки
```

```
            {
                string str = strTemp.Trim();
                if (!list.Contains(str))  // Добавление строки в список, если строка не содержится
в списке
                {
                    list.Add(str);
                }
            }
            t.Stop();
            textBoxReadTime.Text = t.Elapsed.ToString();
        }
        else
        {
            MessageBox.Show("Необходимо выбрать файл");
        }
    }
    private void ButtonSearch_Click(object sender, EventArgs e)
    {
        string word = textBoxSearch.Text.Trim();  // прочли слово
        string strmax = textBoxMax.Text.Trim();  //  прочли максимум
        int max = Convert.ToInt32(strmax);
        if (!string.IsNullOrWhiteSpace(word) && !string.IsNullOrWhiteSpace(strmax) &&
list.Count > 0)  // Проверка пустого слова и не открытого файла
        {
            //string wordUpper = word.ToUpper();  // Поиск в верхнем регистре
            List<string> tempList = new List<string>();  // Временные результаты поиска
            Stopwatch t = new Stopwatch();
            t.Start();
            foreach (string str in list)
            {
                if (str.Contains(word))
                {
                    if (EditDistance.Distance(str, word) <= max)  // вычисление расстояния
                    {
                        tempList.Add(str);
                    }
                }
            }
            t.Stop();
            // Заполнение списка
            textBoxSearchTime.Text = t.Elapsed.ToString();
            listBoxResult.BeginUpdate();
            listBoxResult.Items.Clear();
            foreach (string str in tempList)  // Вывод результатов поиска
            {
                listBoxResult.Items.Add(str);
```

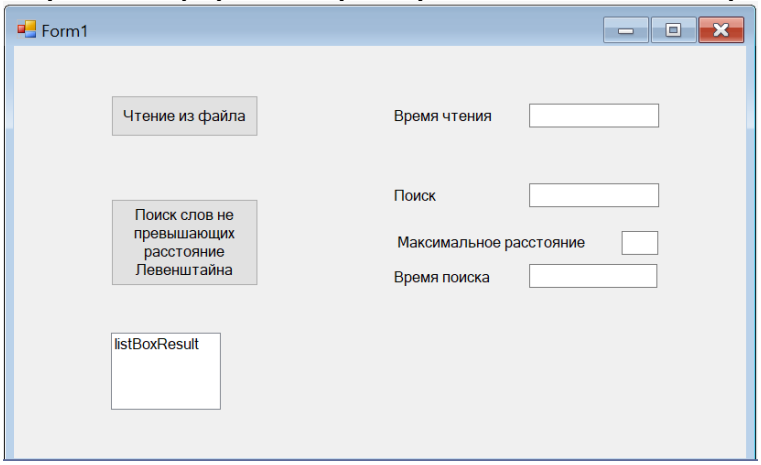
```
            }
            listBoxResult.EndUpdate();
        }
        else
        {
            MessageBox.Show("Необходимо выбрать файл, ввести слово для поиска и
максимальное значение расстояния");
        }
    }
    private void Label1_Click(object sender, EventArgs e){}
    private void Label4_Click(object sender, EventArgs e){}
    private void Label3_Click(object sender, EventArgs e){}
    private void Label2_Click(object sender, EventArgs e){}
    private void TextBoxSearch_TextChanged(object sender, EventArgs e){}
  }
}
```

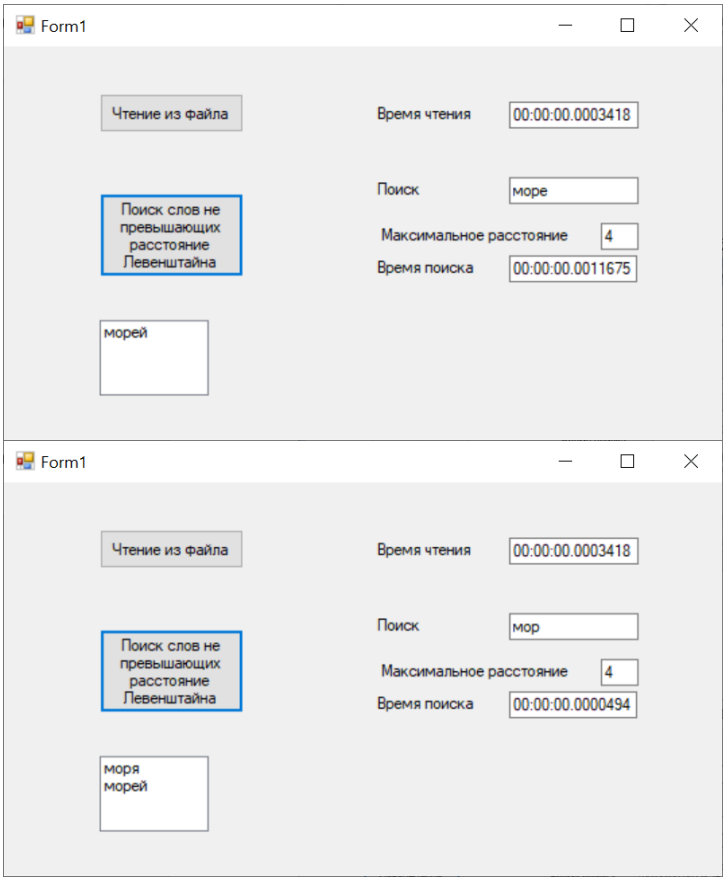
EditDistance.cs

```
using System;
namespace Lab5
{
  public class EditDistance
  {
    //Вычисление расстояния Дамерау-Левенштейна
    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);
                //Добавление
                int ins = matrix[i, j - 1] + 1;
                //Удаление
                int del = matrix[i - 1, j] + 1;
                //Замена
                int subst = matrix[i - 1, j - 1] + symbEqual;
                //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];
    }
    //Вывод расстояния Дамерау-Левенштейна в консоль
    public static void WriteDistance(string str1Param, string str2Param)
    {
        int d = Distance(str1Param, str2Param);
        Console.WriteLine("'" + str1Param + "','" + str2Param + "' -> " + d.ToString());
    }
  }
}
```

Экранные формы с примерами выполнения программы:




Form1
Чтение из файла
Время чтения
00:00:00.0003418
Поиск
море
Поиск слов не превышающих расстояние Левенштайна
Максимальное расстояние
4
Время поиска
00:00:00.0011675
морей
Form1
Чтение из файла
Время чтения
00:00:00.0003418
Поиск
мор
Поиск слов не превышающих расстояние Левенштайна
Максимальное расстояние
4
Время поиска
00:00:00.0000494
моря
морей