Московский государственный технический университет им. Н.Э. Баумана

Факультет «Радиотехнический»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчет по лабораторной работе №5

Выполнил:
студент группы РТ5-31Б
Борисочкин М.И.
Подпись и дата:

Проверил:
к.т.н., доцент
Гапанюк Ю. Е.
Подпись и дата:

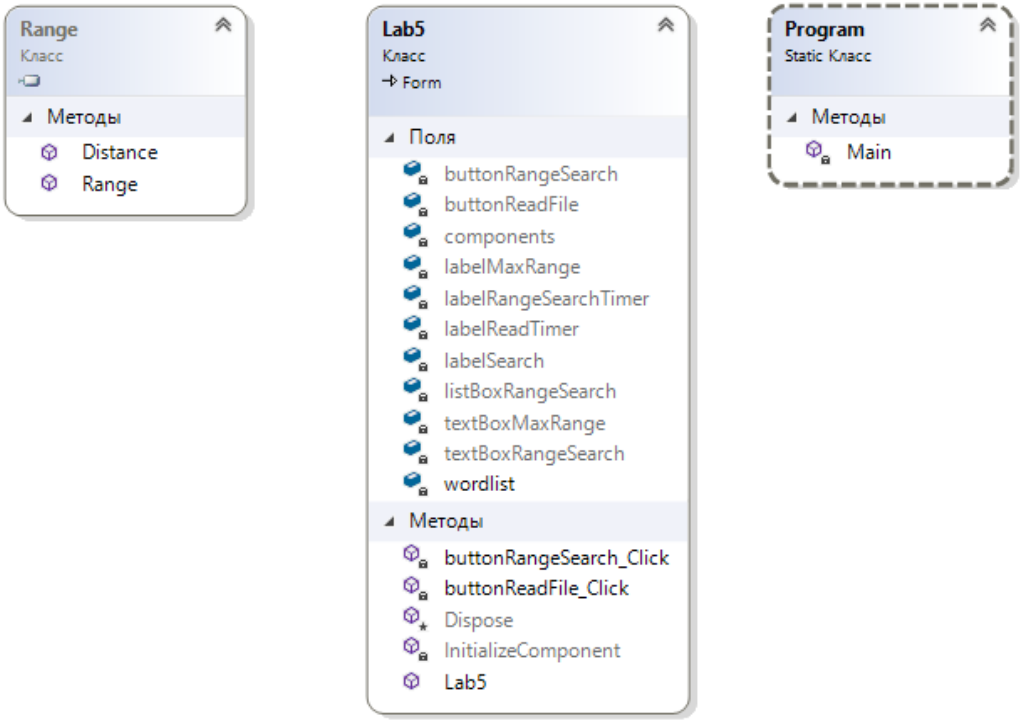
Москва, 2020 г.

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

Range.cs

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

namespace ClassLibraryLab5
{
    public class Range
    {
        /// <summary>
        /// Вычисление расстояния Дамерау-Левенштейна
        /// </summary>
        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];
        }
    }
}
```

Lab5.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;
using ClassLibraryLab5;

namespace Lab5
{
    public partial class Lab5 : Form
    {
        public Lab5()
        {
            InitializeComponent();
        }

        /// <summary>
        /// Список со словами
        /// </summary>
        List<string> wordlist = new List<string>();

        /// <summary>
        /// Чтение файла
        /// </summary>
        private void buttonReadFile_Click(object sender, EventArgs e)
        {
            OpenFileDialog fd = new OpenFileDialog();
            fd.Filter = "Текстовые файлы|*.txt";

            if (fd.ShowDialog() == DialogResult.OK)
            {
                // Запуск таймера
                Stopwatch tmr = new Stopwatch();
                tmr.Start();

                //Чтение файла в виде строки
                string ftext = File.ReadAllText(fd.FileName);

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

                string[] ftetxArray = ftext.Split(separators);
```

```
                foreach (string strTemp in ftetxArray)
                {
                    // Удаление пробелов в начале и в конце строки
                    string str = strTemp.Trim();

                    // Добавление строки в список, если её нет с списке
                    if (!wordlist.Contains(str))
                        wordlist.Add(str);
                }

                // Остановка таймера
                tmr.Stop();
                this.labelReadTimer.Text = "Время чтения: " + tmr.Elapsed.ToString();
            }
            else
                MessageBox.Show("Необходимо выбрать файл", "Предупреждение",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
        }

        private void buttonRangeSearch_Click(object sender, EventArgs e)
        {
            string checkword = this.textBoxRangeSearch.Text.Trim();
            int maxdistance = Convert.ToInt32(this.textBoxMaxRange.Text.Trim());

            if (!string.IsNullOrWhiteSpace(checkword) && wordlist.Count > 0)
            {
                // Слово для поиска в верхнем регистре
                string wordUpper = checkword.ToUpper();

                List<string> tempList = new List<string>();

                // Запуск таймера
                Stopwatch tmr = new Stopwatch();
                tmr.Start();

                foreach (string str in wordlist)
                {
                    int distance = Range.Distance(wordUpper, str.ToUpper());

                    // Проверка максимального расстояния
                    if (distance <= maxdistance)
                        tempList.Add(str);
                }

                // Остановка таймера
                tmr.Stop();
                this.labelRangeSearchTimer.Text = "Время поиска: " +
tmr.Elapsed.ToString();

                this.listBoxRangeSearch.BeginUpdate();

                // Очистка списка
                this.listBoxRangeSearch.Items.Clear();

                //Вывод результатов поиска
                foreach (string str in tempList)
                {
                    this.listBoxRangeSearch.Items.Add(str);
                }
```

```
                this.listBoxRangeSearch.EndUpdate();
            }
        }
    }
}
```

Примеры выполнения программы