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

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

Курс «Парадигмы и конструкции языков программирования»
Отчет по лабораторной работе №5

Выполнил:
Студент группы ИУ5-31Б
Куртинец Роман

Проверил:
Гапанюк Ю. Е.

2025 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.

Листинг программы:

LevenshteinDistance.cs

```
using System;

namespace MyClassLibrary;

public static class LevenshteinDistance
{
    public static int Calculate(string ftWord, string secWord)
    {
        int ftLen = ftWord.Length;
        int secLen = secWord.Length;

        var matrix = new int[ftLen + 1, secLen + 1];

        for (int i = 0; i <= ftLen; ++i) matrix[i, 0] = i;

        for (int i = 0; i <= secLen; ++i) matrix[0, i] = i;

        for (int i = 1; i <= ftLen; ++i)
        {
            for (int j = 1; j <= secLen; ++j)
            {
                int cost = ftWord[i - 1] == secWord[j - 1] ? 0 : 1;

                int deletion = matrix[i - 1, j] + 1;
                int insertion = matrix[i, j - 1] + 1;
                int substitution = matrix[i - 1, j - 1] + cost;
```

```
                matrix[i, j] = Math.Min(Math.Min(deletion, insertion),
substitution);

                if (i > 1 && j > 1 && (ftWord[i - 1] == secWord[j - 2]) &&
(ftWord[i - 2] == secWord[j - 1]))
                {
                    matrix[i, j] = Math.Min(matrix[i, j], matrix[i - 2, j - 2] +
1);
                }
            }
        }

        return matrix[ftLen, secLen];
    }

    public static void Main()
    {
        System.Console.WriteLine(Calculate("слово", "строка"));
        System.Console.WriteLine(Calculate("кот", "кит"));
        System.Console.WriteLine(Calculate("", "привет"));

        System.Console.WriteLine(Calculate("кот", "кто"));
        System.Console.WriteLine(Calculate("строка", "соркта"));
        System.Console.WriteLine(Calculate("абв", "бав"));
        System.Console.WriteLine(Calculate("hello", "helo"));
    }
}
```

Form1.cs

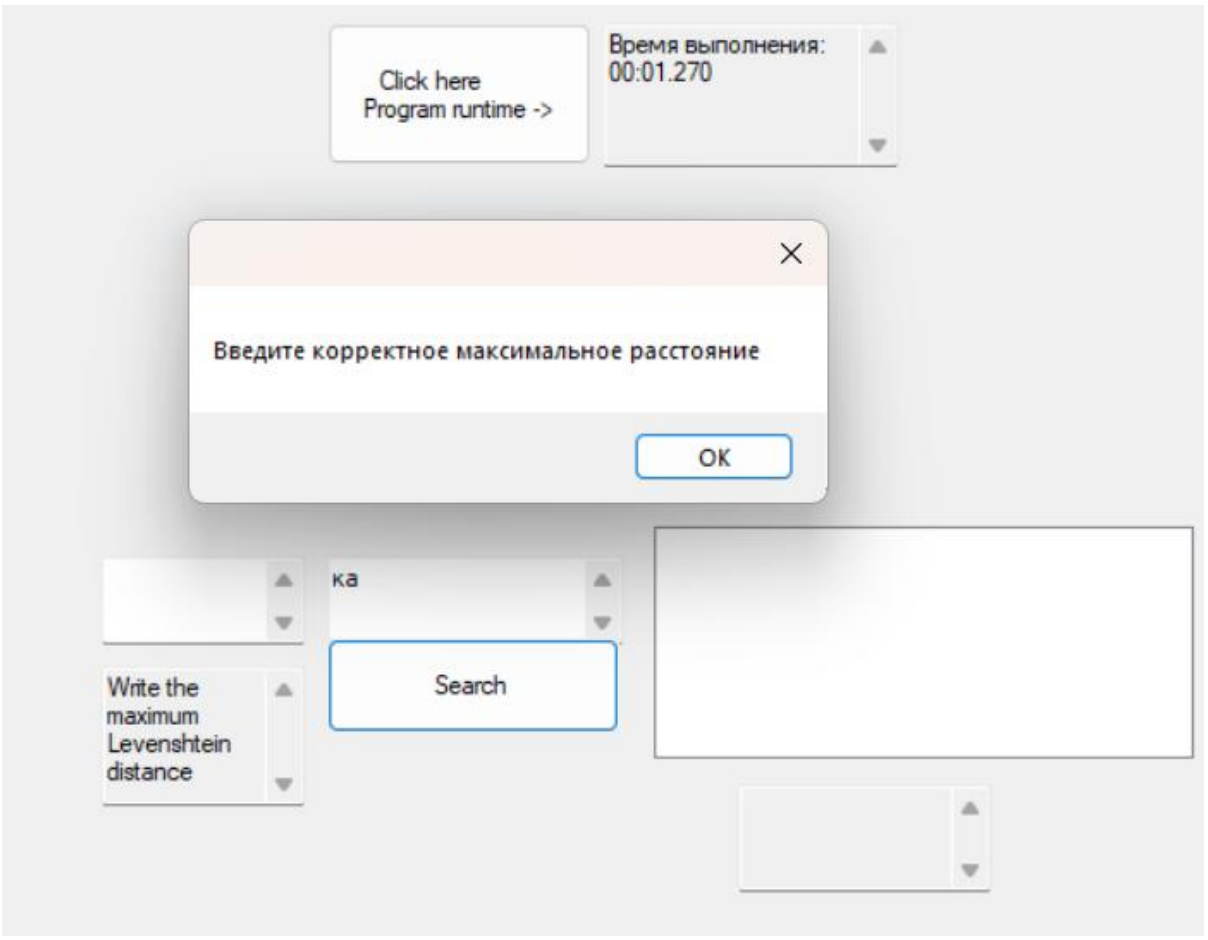
```
using System;
using System.Collections;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;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

namespace Lab_4
{
    public partial class Form1 : Form
    {

        private List<string> words = new List<string>();
        public Form1()
        {
            InitializeComponent();
        }
```

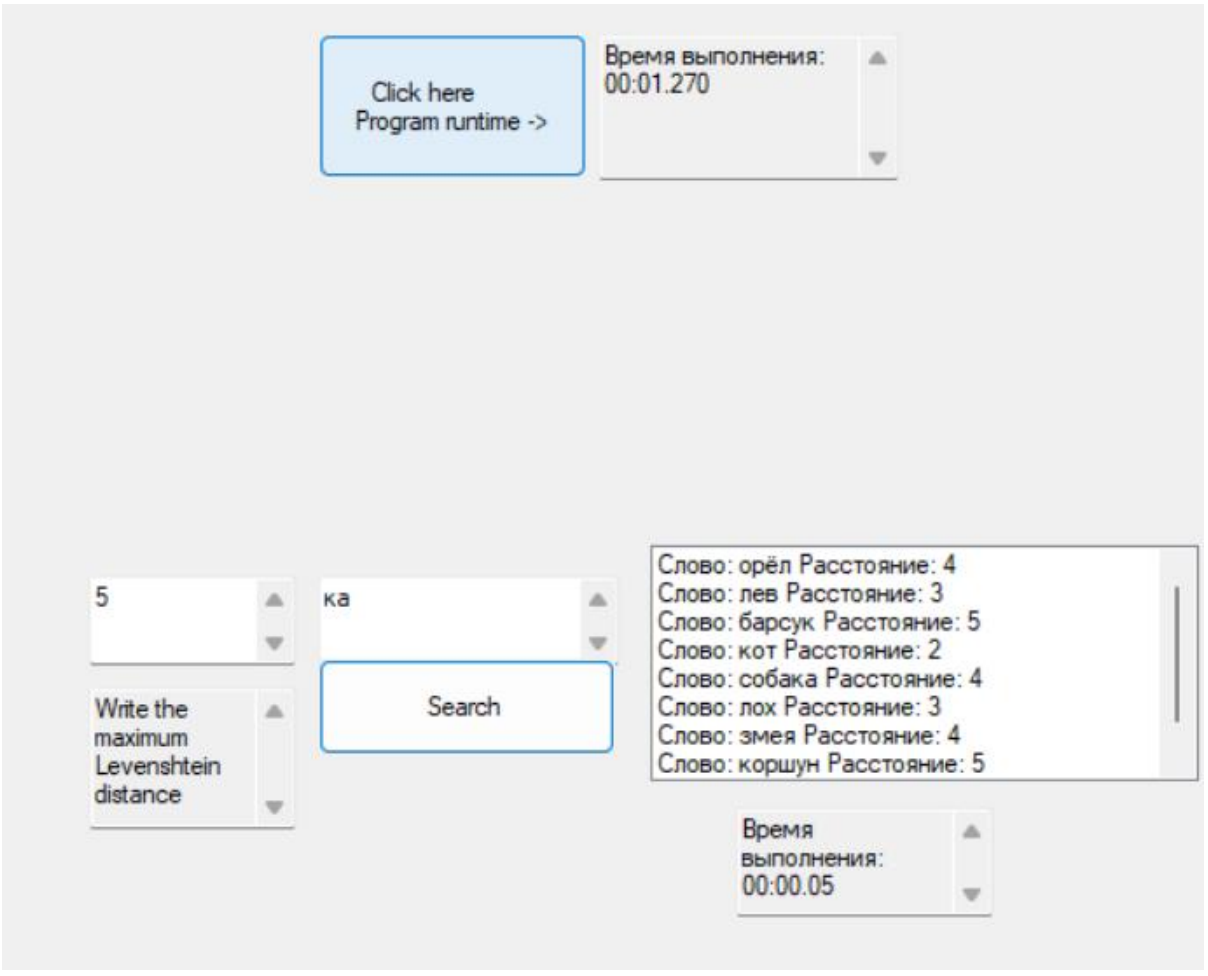
```
        private void button1_Click(object sender, EventArgs e)
        {
            OpenFileDialog openFileDialog = new OpenFileDialog();
            openFileDialog.Title = "Open Text File";
            openFileDialog.Filter = "TXT files|*.txt";
            openFileDialog.InitialDirectory = @"C:\Users\romak\OneDrive\Рабочий
стол\Lab_5\Lab_5";

            if (openFileDialog.ShowDialog() == DialogResult.OK)
            {
                string filePath = openFileDialog.FileName;
                try
                {
                    Stopwatch stopWatch = new Stopwatch();
                    stopWatch.Start();

                    string content = File.ReadAllText(filePath, Encoding.Default);
                    string[] wordArray = content.Split(new char[] { ' ', '\t', '\r', '\n'
}, StringSplitOptions.RemoveEmptyEntries);

                    foreach (string word in wordArray)
                    {
                        if (!words.Contains(word))
                        {
                            words.Add(word);
                        }
                    }
                    MessageBox.Show(string.Join(" ", words), "Список слов");

                    stopWatch.Stop();
                    TimeSpan ts = stopWatch.Elapsed;

                    string elapsedTime = String.Format("{0:00}:{1:00}.{2:00}",
                        ts.Minutes, ts.Seconds, ts.Milliseconds);
                    textBox1.Text = $"Время выполнения: {elapsedTime}";
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Ошибка чтения файла: " + ex.Message);
                }
            }
        }

        private void textBox1_TextChanged(object sender, EventArgs e)
        {
        }

        private void textBox2_TextChanged(object sender, EventArgs e)
        {
        }

        private List<(string word, int distance)> LevenshteinDistance(string ftWord, int
maxDistance)
        {
            var results = new List<(string word, int distance)>();


            int ftLen = ftWord.Length;

            foreach (var secWord in words)
            {

                int secLen = secWord.Length;

                var matrix = new int[ftLen + 1, secLen + 1];

                for (int i = 0; i <= ftLen; ++i) matrix[i, 0] = i;
```

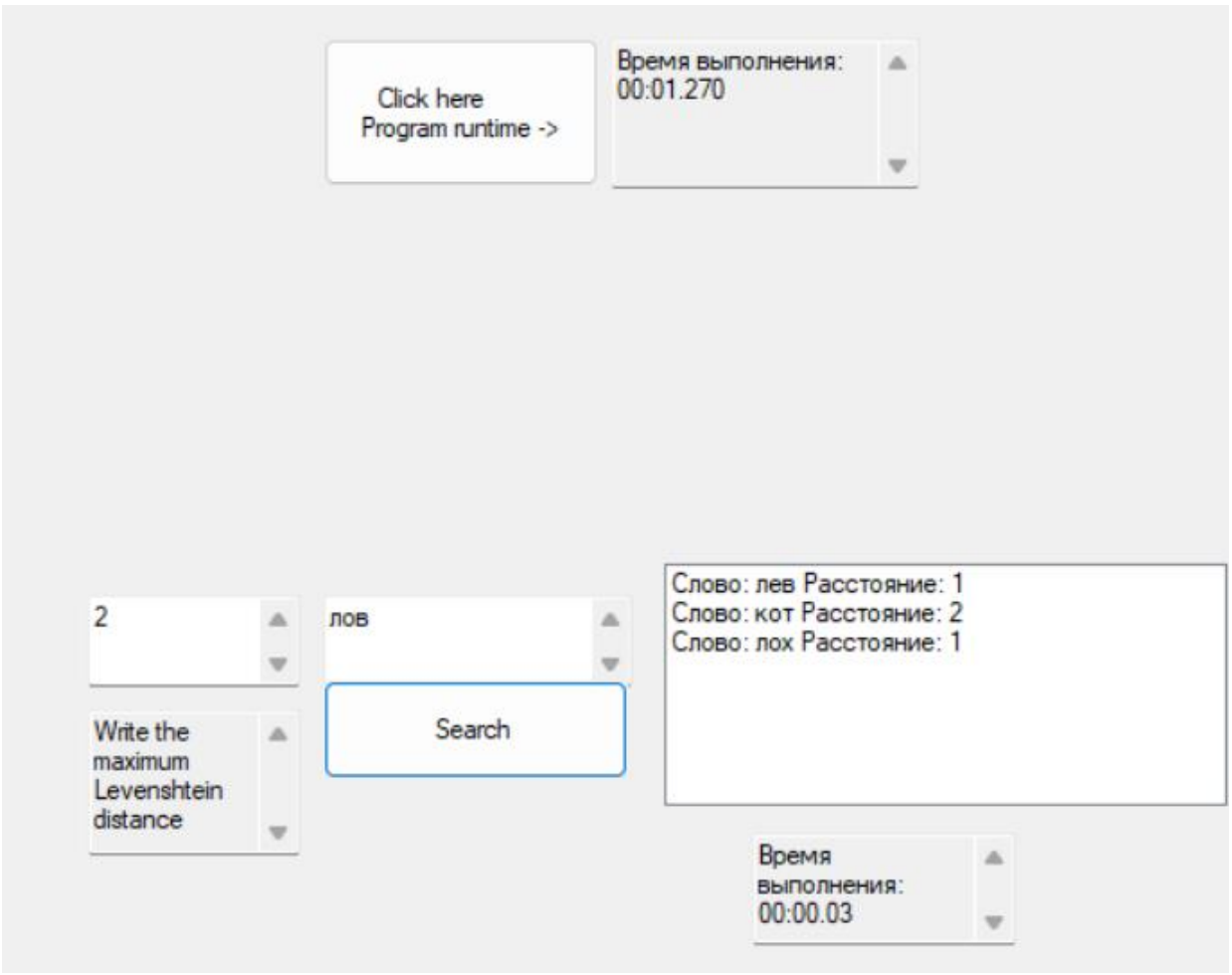
```
                for (int i = 0; i <= secLen; ++i) matrix[0, i] = i;

                for (int i = 1; i <= ftLen; ++i)
                {
                    for (int j = 1; j <= secLen; ++j)
                    {
                        int cost = ftWord[i - 1] == secWord[j - 1] ? 0 : 1;

                        int deletion = matrix[i - 1, j] + 1;
                        int insertion = matrix[i, j - 1] + 1;
                        int substitution = matrix[i - 1, j - 1] + cost;

                        matrix[i, j] = Math.Min(Math.Min(deletion, insertion),
substitution);

                        if (i > 1 && j > 1 && (ftWord[i - 1] == secWord[j - 2]) &&
(ftWord[i - 2] == secWord[j - 1]))
                        {
                            matrix[i, j] = Math.Min(matrix[i, j], matrix[i - 2, j - 2] +
1);
                        }
                    }
                }

                if (maxDistance >= matrix[ftLen, secLen])
                {
                    results.Add((secWord, matrix[ftLen, secLen]));
                }
            }
            return results;
        }

        private void button2_Click(object sender, EventArgs e)
        {
            if (words.Count == 0)
            {
                MessageBox.Show("Сначала загрузите файл с помощью первой кнопки!");
                return;
            }


            string ftWord = textBox2.Text.Trim();
            int maxDistance;
            bool parse = Int32.TryParse(textBox4.Text.Trim(), out maxDistance);
            if (string.IsNullOrEmpty(ftWord))
            {
                MessageBox.Show("Введите слово для поиска!");
                return;
            }
            if (!parse || maxDistance <= 0)
            {
                MessageBox.Show("Введите корректное максимальное расстояние");
            }

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

            listBox1.BeginUpdate();
            try
            {

                listBox1.Items.Clear();
                var results = LevenshteinDistance(ftWord, maxDistance);
                if (results.Count == 0)
                {
                    listBox1.Items.Add("Совпадений нет");
                }
```

```
                else
                {
                    foreach (var r in results)
                    {
                        listBox1.Items.Add($"Слово: {r.word} Расстояние: {r.distance}");
                    }
                }
            }
            finally
            {
                listBox1.EndUpdate();
            }
            stopWatch.Stop();
            TimeSpan ts = stopWatch.Elapsed;

            string elapsedTime = String.Format("{0:00}:{1:00}.{2:00}",
                    ts.Minutes, ts.Seconds, ts.Milliseconds);
            textBox3.Text = $"Время выполнения: {elapsedTime}";
        }

        private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
        }

        private void textBox3_TextChanged(object sender, EventArgs e)
        {
        }

        private void textBox4_TextChanged(object sender, EventArgs e)
        {
        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }

        private void textBox5_TextChanged(object sender, EventArgs e)
        {
        }

        private void textBox4_KeyPress(object sender, KeyPressEventArgs e)
        {
            char number = e.KeyChar;

            if (!Char.IsDigit(number))
            {
                e.Handled = true;
            }
        }
    }
}
```

Результат выполнения:


Click here
Program runtime ->
Время выполнения:
00:01.270
Введите корректное максимальное расстояние
OK
ка
Search
Write the maximum Levenshtein distance


Click here
Program runtime ->
Время выполнения:
00:01.270
5
ка
Search
Write the maximum Levenshtein distance
Слово: орёл Расстояние: 4
Слово: лев Расстояние: 3
Слово: барсук Расстояние: 5
Слово: кот Расстояние: 2
Слово: собака Расстояние: 4
Слово: лох Расстояние: 3
Слово: змея Расстояние: 4
Слово: коршун Расстояние: 5
Время выполнения:
00:00.05


Click here
Program runtime ->
Время выполнения:
00:01.270
2
лов
Слово: лев Расстояние: 1
Слово: кот Расстояние: 2
Слово: лох Расстояние: 1
Write the maximum Levenshtein distance
Search
Время выполнения:
00:00.03