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 №5

“ Разработка программы, реализующую вычисление расстояния Левенштейна с использованием алгоритма Вагнера-Фишера”

Выполнил:
студент группы ИУ5-33Б
Ларионова Амина
Подпись и дата:
16.12.20

Проверил:
Подпись и дата:

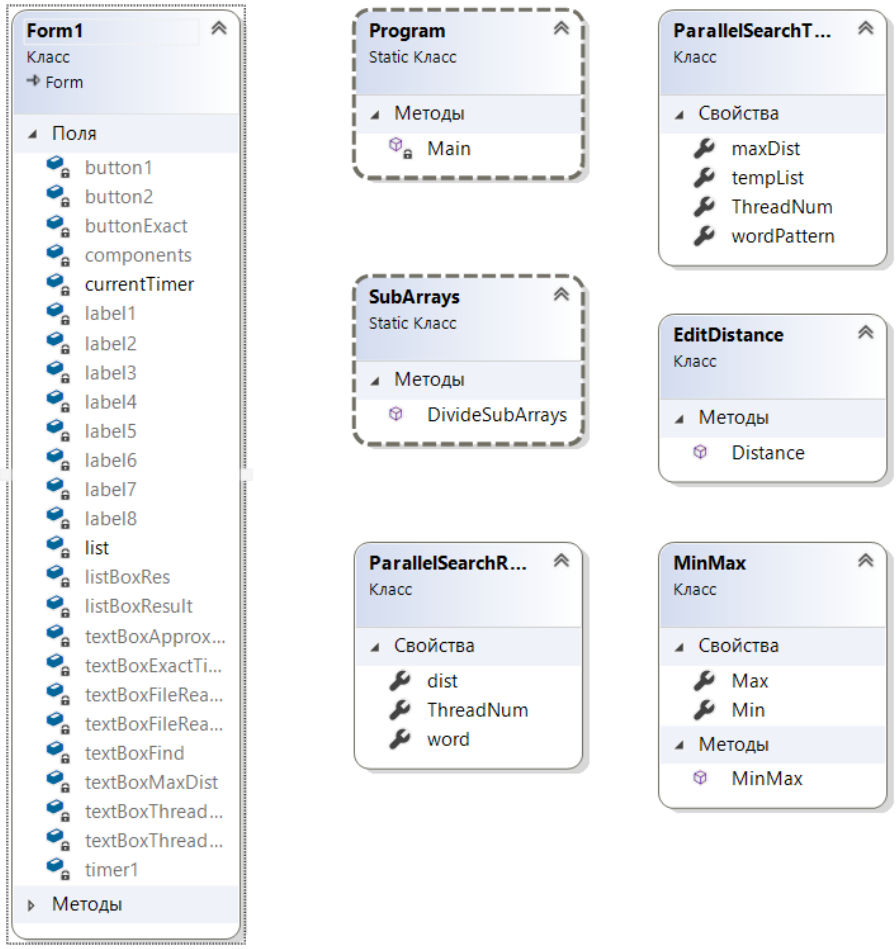
Москва, 2020 г.

1)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.

2) Диаграмма классов



3) Текст программы

Program.cs

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

namespace Лабораторная___4
{
    static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

MinMax.cs

```
namespace Лабораторная___4
{
    public class MinMax
    {
        public int Min { get; set; }
        public int Max { get; set; }
        public MinMax(int pmin, int pmax)
        {
            this.Min = pmin;
            this.Max = pmax;
        }
    }

}
```

ParellesSearchResult.cs

```
namespace Лабораторная___4
{
    public class ParallelSearchResult
    {
        /// <summary>
        /// Найденное слово
        /// </summary>
        public string word { get; set; }
        /// <summary>
        /// Расстояние
        /// </summary>
        public int dist { get; set; }
        /// <summary>
        /// Номер потока
        /// </summary>
        public int ThreadNum { get; set; }
    }
}
```

ParellelSearchThreadParam.cs

```
using System.Collections.Generic;

namespace Лабораторная___4
{
    class ParallelSearchThreadParam
    {
        /// <summary>
        /// Массив для поиска
        /// </summary>
        public List<string> tempList { get; set; }
        /// <summary>
        /// Слово для поиска
        /// </summary>
        public string wordPattern { get; set; }
        /// <summary>
        /// Максимальное расстояние для нечеткого поиска
         /// </summary>
        public int maxDist { get; set; }
        /// <summary>
        /// Номер потока
        /// </summary>
        public int ThreadNum { get; set; }
    }


}
```

SubArrays1.cs

```
using System;
using System.Collections.Generic;

namespace Лабораторная___4
{
    public static class SubArrays
    {
        /// <summary>
        /// Деление массива на последовательности
        /// </summary>
        /// <param name="beginIndex">Начальный индекс массива</param>
        /// <param name="endIndex">Конечный индекс массива</param>
        /// <param name="subArraysCount">Требуемое количество подмассивов</param>
        /// <returns>Список пар с индексами подмассивов</returns>
        public static List<MinMax> DivideSubArrays(int beginIndex, int endIndex, int
subArraysCount)
        {
            //Результирующий список пар с индексами подмассивов
            List<MinMax> result = new List<MinMax>();
            //Если число элементов в массиве слишком мало для деления
            //то возвращается массив целиком
            if ((endIndex - beginIndex) <= subArraysCount)
            {
                result.Add(new MinMax(0, (endIndex - beginIndex)));
            }
            else
            {
                //Размер подмассива
                int delta = (endIndex - beginIndex) / subArraysCount;
                //Начало отсчета
                int currentBegin = beginIndex;
                //Пока размер подмассива укладывается в оставшуюся
                //последовательность
                while ((endIndex - currentBegin) >= 2 * delta)
                {
                    //Формируем подмассив на основе начала
                    //последовательности
```

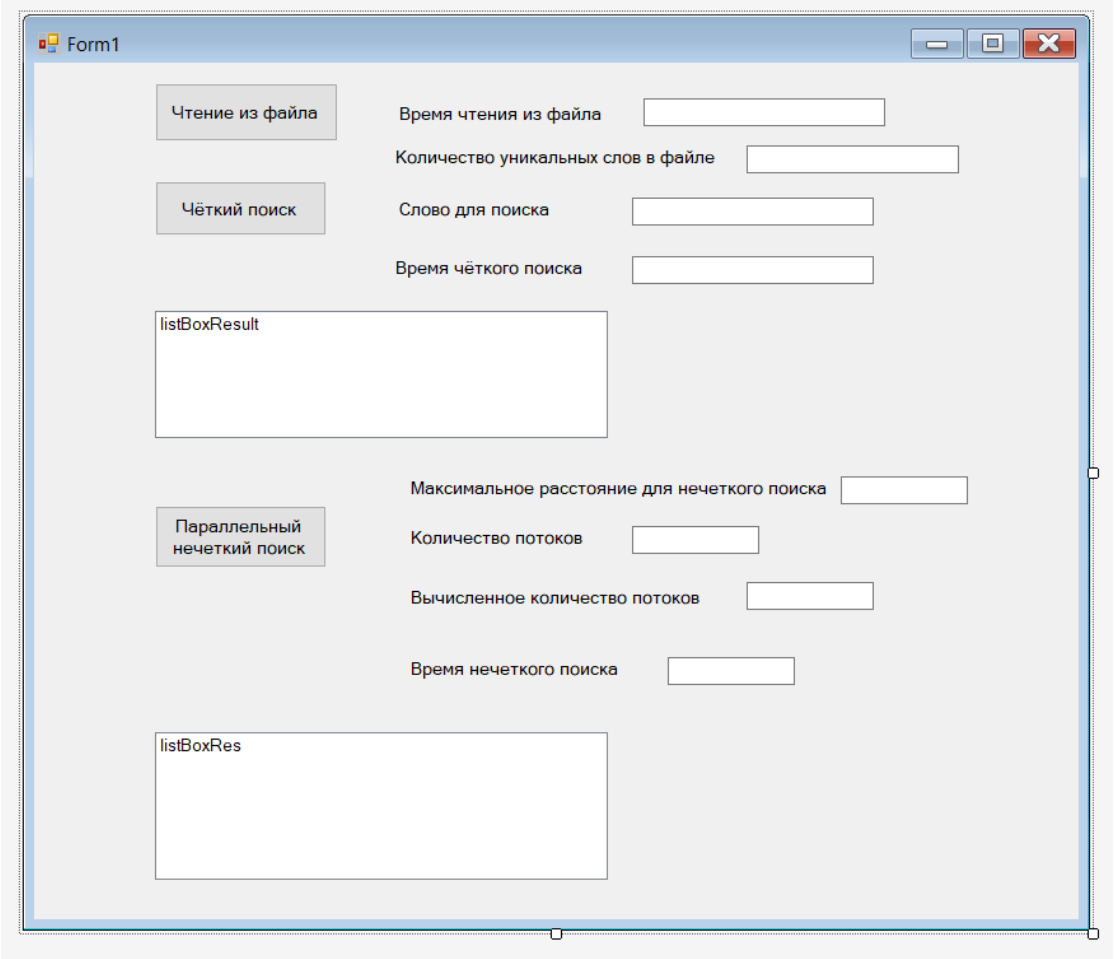
```
                    result.Add(new MinMax(currentBegin, currentBegin + delta));
                    //Сдвигаем начало последовательности
                    //вперед на размер подмассива
                    currentBegin += delta;
                }
                //Оставшийся фрагмент массива
                result.Add(new MinMax(currentBegin, endIndex));
            }
            //Возврат списка результатов
            return result;
        }
    }
}
```

EditDistance1.cs

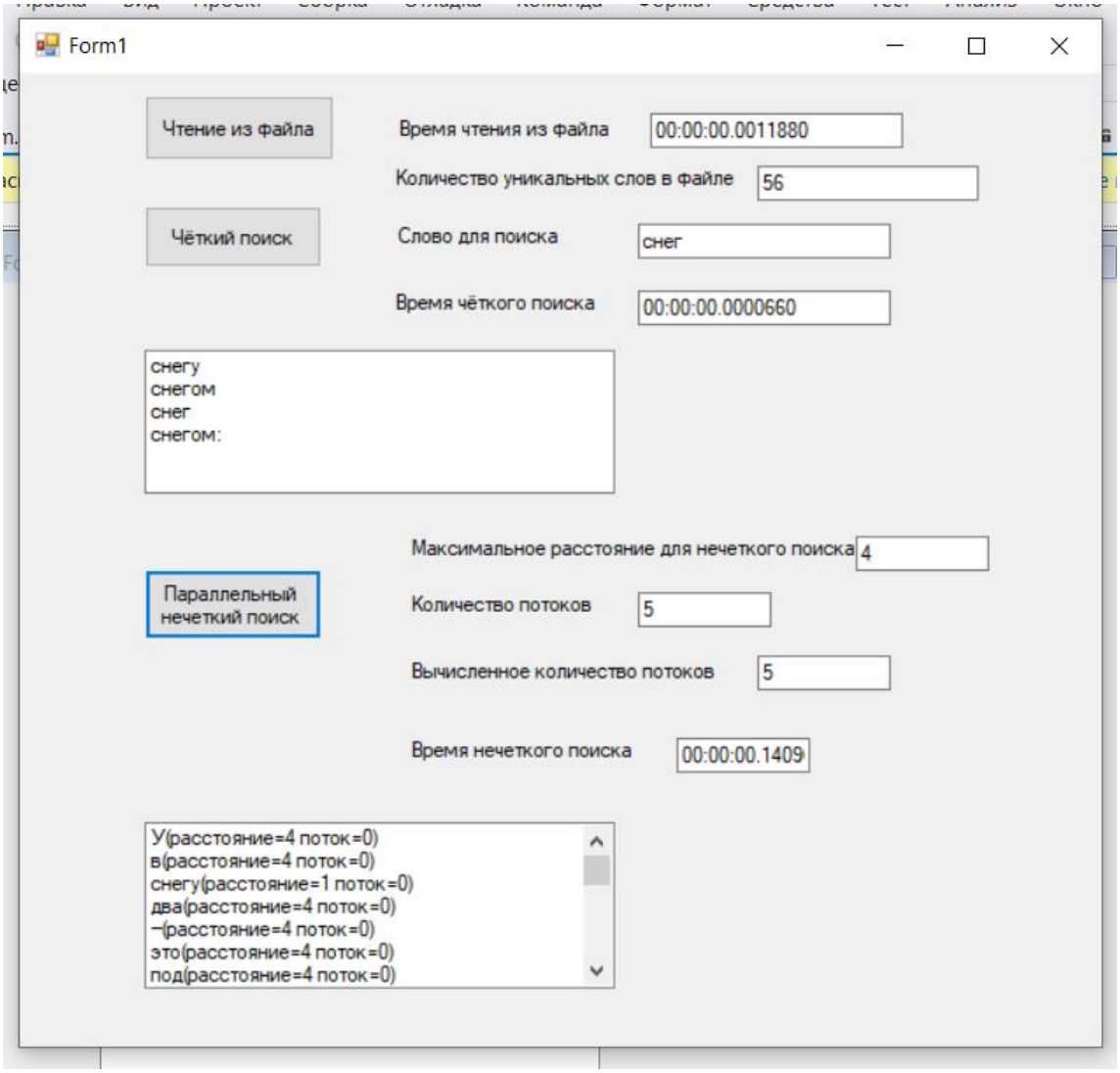
```
using System;

namespace Лабораторная___4
{
    internal class EditDistance
    {

        /// 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];
        }


    }
}
```

Forms1.cs



4) Экранные формы с примерами выполнения программы


Form1
Чтение из файла
Время чтения из файла
00:00:00.0011880
Количество уникальных слов в файле
56
Чёткий поиск
Слово для поиска
снег
Время чёткого поиска
00:00:00.0000660
снегу
снегом
снег
снегом:
Максимальное расстояние для нечеткого поиска
4
Параллельный нечеткий поиск
Количество потоков
5
Вычисленное количество потоков
5
Время нечеткого поиска
00:00:00.1409
У(расстояние=4 поток=0)
в(расстояние=4 поток=0)
снегу(расстояние=1 поток=0)
два(расстояние=4 поток=0)
—(расстояние=4 поток=0)
это(расстояние=4 поток=0)
под(расстояние=4 поток=0)