Защищено:
Гапанюк Ю.Е.

"__"_____**2023** г.

Демонстрация:
Казакова В.В.

"__"_____**2023** г.

Отчет по лабораторной работе № 5 по курсу Парадигмы и конструкции языков программирования

Тема работы: " Вычисление расстояния Левенштейна с использованием алгоритма Вагнера-Фишера"

3
(количество листов)

ИСПОЛНИТЕЛЬ:

студент группы ИУ5Ц-51Б _____
(подпись)

Казакова В.В. "__"_____**2023** г.

Москва, МГТУ - **2023**

1. Описание задания

Разработать программу на языке программирования C#, 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. Текст программы

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

namespace LevenshteinDistance
{
    public class LevenshteinDistanceCalculator
    {
        public int CalculateDistance(string a, string b, int maxDistance)
        {
            if (maxDistance < 0)
            {
                throw new ArgumentException("MaxDistance should be non-
negative.");
            }

            int[,] matrix = new int[a.Length + 1, b.Length + 1];

            for (int i = 0; i <= a.Length; i++)
            {
                matrix[i, 0] = i;
            }

            for (int j = 0; j <= b.Length; j++)
            {
                matrix[0, j] = j;
            }

            for (int i = 1; i <= a.Length; i++)
            {
                for (int j = 1; j <= b.Length; j++)
```

```
                {
                    int cost = (a[i - 1] == b[j - 1]) ? 0 : 1;
                    matrix[i, j] = Math.Min(
                        Math.Min(matrix[i - 1, j] + 1, matrix[i, j - 1] + 1),
                        matrix[i - 1, j - 1] + cost);

                    if (matrix[i, j] > maxDistance)
                    {
                        return maxDistance;
                    }
                }
            }

            return matrix[a.Length, b.Length];
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            LevenshteinDistanceCalculator calculator = new
LevenshteinDistanceCalculator();
            string a = "Saturday";
            string b = "Sunday";
            int maxDistance = 10;

            int distance = calculator.CalculateDistance(a, b, maxDistance);
            Console.WriteLine($"The Levenshtein distance between \"{a}\" and
\"{b}\" is {distance}.");
            Console.ReadLine();
        }
    }
}
```

3. Экранные формы