**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 №6

«Разработка простого бота для Telegram с использованием языка Python»

|  |  |  |
| --- | --- | --- |
| Выполнила: |  | Проверил: |
| студент группы ИУ5-33Б |  | преподаватель каф. ИУ5 |
| Хабленко Инна |  | Гапанюк Ю. Е. |
|  |  |  |

Москва, 2023 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
4. Дамерау-Левенштейна (с учетом перестановок соседних символов).
5. Модифицировать предыдущую лабораторную работу, вместо поиска подстроки используется вычисление расстояния Левенштейна.
6. 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

# Текст программы

using System;

namespace LevenshteinDistanceCalculator{

public class LevenshteinDistance{

public int Calculate(string word\_1, string word\_2){

if (string.IsNullOrEmpty(word\_1)){

return word\_2?.Length ?? 0;

}

if (string.IsNullOrEmpty(word\_2)){

return word\_1.Length;

}

int Length\_1 = word\_1.Length;

int Length\_2 = word\_2.Length;

int[,] distanceMatrix = new int[Length\_1 + 1, Length\_2 + 1];

for (int i = 0; i <= Length\_1; i++){

distanceMatrix[i, 0] = i;

}

for (int j = 0; j <= Length\_2; j++){

distanceMatrix[0, j] = j;

}

for (int i = 1; i <= Length\_1; i++){

for (int j = 1; j <= Length\_2; j++){

int cost = (word\_2[j - 1] == word\_1[i - 1]) ? 0 : 1;

distanceMatrix[i, j] = Math.Min(

Math.Min(distanceMatrix[i - 1, j] + 1, distanceMatrix[i, j - 1] + 1),

distanceMatrix[i - 1, j - 1] + cost);

}

}

return distanceMatrix[Length\_1, Length\_2];

}

public int Print(string word\_1, string word\_2, int MAX, int distance){

if (distance < MAX){

Console.WriteLine($"Дистанция между '{word\_1}' и '{word\_2}': {distance}");

}

return 0;

}

}

class Program{

static void Main(string[] args){

int MAX = 2;

LevenshteinDistance distanceCalculator = new LevenshteinDistance();

string word\_1 = "cat";

string word\_2 = "can";

int distance = distanceCalculator.Calculate(word\_1, word\_2);

distanceCalculator.Print(word\_1, word\_2, MAX, distance);

word\_2 = "sad";

distance = distanceCalculator.Calculate(word\_1, word\_2);

distanceCalculator.Print(word\_1, word\_2, MAX, distance);

}

}

}

# Результат

