**Министерство науки и высшего образования Российской Федерации**

**ФГАОУ ВО «Севастопольский государственный университет»**

Институт информационных технологий

Кафедра «Информационные системы»

**Отчет по лабораторной работе №8**

по дисциплине «Технологии обработки информации»

Выполнил: студент группы

ИС/б-21-2-о

Мельничук В.В.

Принял:

г. Севастополь

**Лабораторная работа №8**

Тема: Алгоритмы обработки текстовых данных

Ход работы:

1. Построить облако тегов по тексту



Рисунок 1 – Входные данные

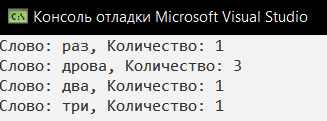


Рисунок 2 – Результат выполнения программы

Листинг:

using System;

using System.Collections.Generic;

using System.Linq;

public class TagCloudBuilder

{

public Dictionary<string, int> BuildTagCloud(string text)

{

Dictionary<string, int> tagCloud = new Dictionary<string, int>();

// Разделите текст на отдельные слова

string[] words = text.Split(new[] { ' ', ',', '.', ';', ':', '!', '?' }, StringSplitOptions.RemoveEmptyEntries);

foreach(string word in words)

{

// Проверьте, что слово является именем существительным в именительном падеже или имеет длину не менее 3 символов

if (IsNounInNominativeCase(word) || word.Length >= 3)

{

// Приведите слово к нижнему регистру

string normalizedWord = word.ToLower();

// Если слово уже присутствует в облаке тегов, увеличьте его счетчик

if (tagCloud.ContainsKey(normalizedWord))

{

tagCloud[normalizedWord]++;

}

else

{

// Добавьте слово в облако тегов со счетчиком 1

tagCloud.Add(normalizedWord, 1);

}

}

}

return tagCloud;

}

private bool IsNounInNominativeCase(string word)

{

return false;

}

}

public class Program

{

public static void Main(string[] args)

{

// Пример использования

string text = " Раз дрова, два дрова, три дрова "

TagCloudBuilder tagCloudBuilder = new TagCloudBuilder();

Dictionary<string, int> tagCloud = tagCloudBuilder.BuildTagCloud(text);

// Вывод статистики вхождения слов в текст

foreach(KeyValuePair<string, int> tag in tagCloud)

{

Console.WriteLine($"Слово: {tag.Key}, Количество: {tag.Value}");

}

}

}

1. Реализовать алгоритм вычисления разницы между двумя текстами (антиплагиат)

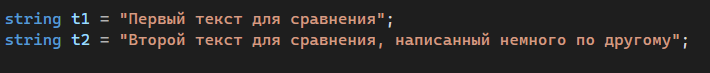


Рисунок 3 – Входные данные

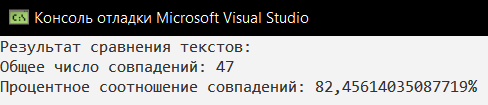


Рисунок 4 – Результат работы программы

Листинг:

using System;

public class PlagiarismChecker

{

public void CompareTexts(string text1, string text2)

{

int[,] matrix = new int[text1.Length + 1, text2.Length + 1];

// Инициализация первой строки и первого столбца матрицы

for (int i = 0; i <= text1.Length; i++)

{

matrix[i, 0] = i;

}

for (int j = 0; j <= text2.Length; j++)

{

matrix[0, j] = j;

}

// Заполнение матрицы

for (int i = 1; i <= text1.Length; i++)

{

for (int j = 1; j <= text2.Length; j++)

{

if (text1[i - 1] == text2[j - 1])

{

matrix[i, j] = matrix[i - 1, j - 1];

}

else

{

matrix[i, j] = Math.Min(Math.Min(matrix[i - 1, j] + 1, matrix[i, j - 1] + 1), matrix[i - 1, j - 1] + 1);

}

}

}

// Вычисление общего числа совпадений и процентного соотношения

int matchesCount = text1.Length + text2.Length - matrix[text1.Length, text2.Length];

double percentage = (double)matchesCount / (double)Math.Max(text1.Length, text2.Length) \* 100;

// Вывод результатов

Console.WriteLine("Результат сравнения текстов:");

Console.WriteLine($"Общее число совпадений: {matchesCount}");

Console.WriteLine($"Процентное соотношение совпадений: {percentage}%");

}

}

public class Program

{

public static void Main(string[] args)

{

string t1 = "Первый текст для сравнения";

string t2 = "Второй текст для сравнения, написанный немного по другому";

PlagiarismChecker checker = new PlagiarismChecker();

checker.CompareTexts(t1, t2);

}

}

**Вывод:** в ходе выполнения лабораторной работы были изучены подходы к строительству облаков тегов по тексту, а также к реализации проверки на антиплагиат.