Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №2**

По дисциплине «Современные платформы программирования»

**Выполнил:**

Студент 3 курса

Группы ПО-8

Лобарев А.М.

**Проверил:**

Крощенко А.А.

Брест 2024

**Цель работы**: приобрести базовые навыки работы с файловой системой.

**Вариант 16**

**Задание 1**

Напишите программу, которая читает текст построчно, а затем разбивает каждую строку на лексемы и выводит их в обратном порядке.

Код:

static void Main()

{

var path = @"..\..\..\Text.txt";

var reader = new StreamReader(path);

string? temp;

while ((temp = reader.ReadLine()) != null)

{

var lex = temp.Trim().Split(' ', StringSplitOptions.RemoveEmptyEntries);

Console.WriteLine(string.Join(" ", lex.Reverse()));

}

}

Файл:

sbs eh iuhuh shdoh fj fg

j hdf hsf s

sh shshdfjh d

es k hoeshuhge g

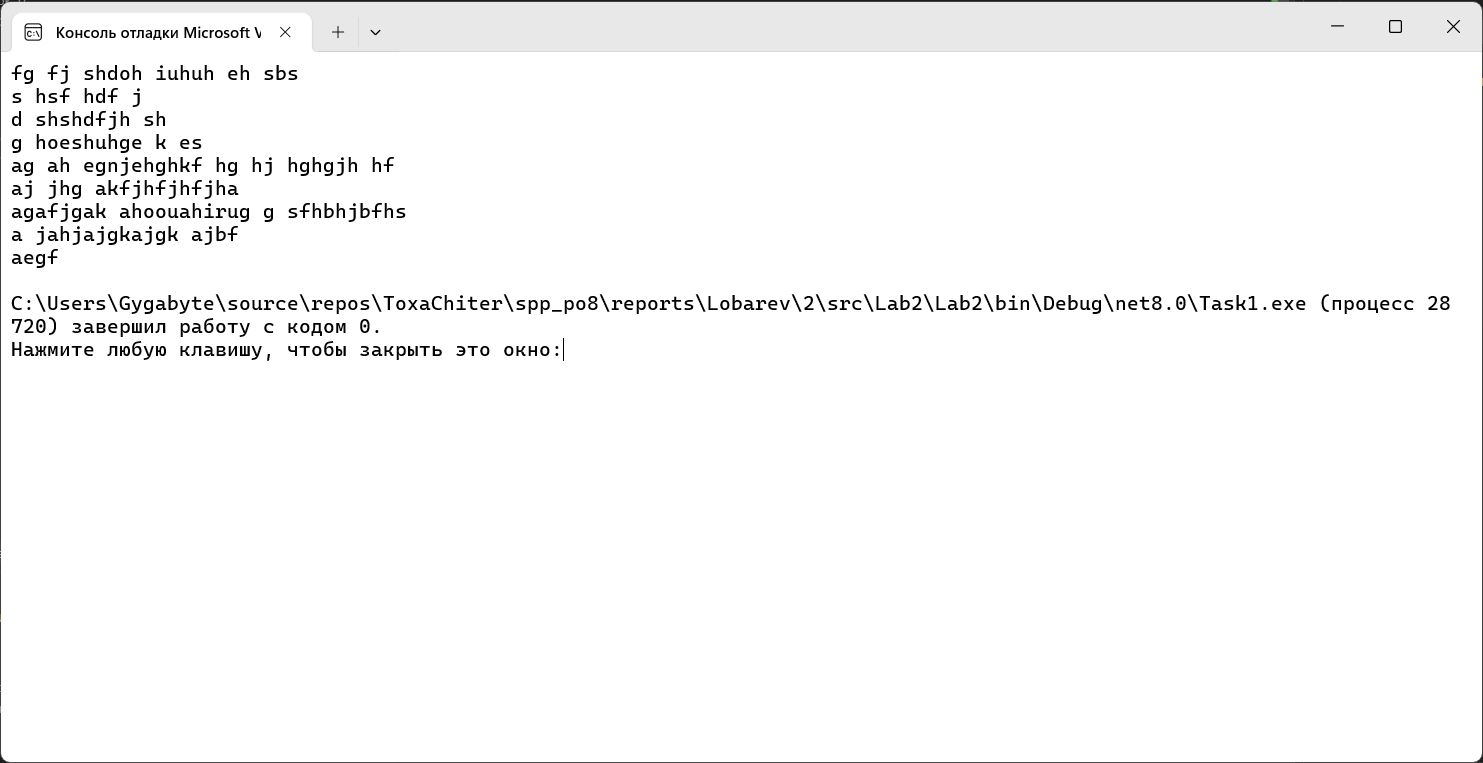
hf hghgjh hj hg egnjehghkf ah ag

akfjhfjhfjha jhg aj

sfhbhjbfhs g ahoouahirug agafjgak

ajbf jahjajgkajgk a

aegf



**Задание 2**

Утилита uniq отфильтровывает повторяющиеся строки во входном файле. Если входной файл задан как – или не задан вовсе, то чтение производится из стандартного ввода. Если выходной файл не задан, запись производится в стандартный вывод. Если одна и та же строка встречается второй и более разы, то она не записывается в вывод программы.

Формат использования: uniq [-c | -d | -u] [-i] [входной\_файл [выходной\_файл]], где ключи имеют следующее значение:

• -u Выводить только те строки, которые не повторяются на входе.

• -d Выводить только те строки, которые повторяются на входе.

• -c Перед каждой строкой выводить число повторений этой строки на входе и один пробел.

• -i Сравнивать строки без учёта регистра.

static void Uniq(TextReader reader, TextWriter writer, bool inverse, bool countRepeats, bool ignoreCase)

{

var dict = new Dictionary<string, (int, int)>();

int number = 1;

var line = reader.ReadLine();

while (line is not null and not "^Z")

{

if (ignoreCase)

{

line = line.ToLowerInvariant();

}

if (dict.TryGetValue(line, out (int, int) value))

{

value.Item1++;

dict[line] = value;

}

else

{

dict.Add(line, (1, number++));

}

line = reader.ReadLine();

}

reader.Close();

Console.Clear();

Func<KeyValuePair<string, (int, int)>, bool> func = kvp => kvp.Value.Item1 == 1;

if (inverse)

{

func = kvp => kvp.Value.Item1 > 1;

}

if (countRepeats)

{

func = \_ => true;

}

var sorted = dict.Where(func).OrderBy(kvp => kvp.Value.Item2);

foreach (var str in sorted)

{

if (countRepeats)

{

writer.WriteLine($"{str.Value.Item1} {str.Key}");

}

else

{

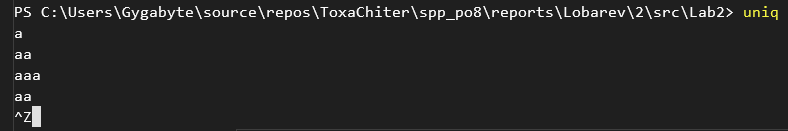
writer.WriteLine(str.Key);

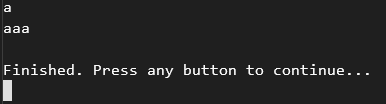
}

}

writer.Close();

}





**Вывод**: приобрел базовые навыки работы с файловой системой.