

Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

ОТЧЕТ

ЛАБОРАТОРНАЯ РАБОТА №2

ССП

Выполнил
студент группы
АС-50:
Протасевич А.В.
Проверил:
Крощенко А.А.

Брест 2020

Цель работы:

приобрести базовые навыки работы с файловой системой в C#.

Вариант 8

Задание 1

Напишите программу, выполняющую чтение текстовых данных из файла и их последующую обработку: Напишите программу, считывающую текст построчно и изменяющую порядок следования слов на случайный. Строки с новым порядком слов выведите на экран.

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;

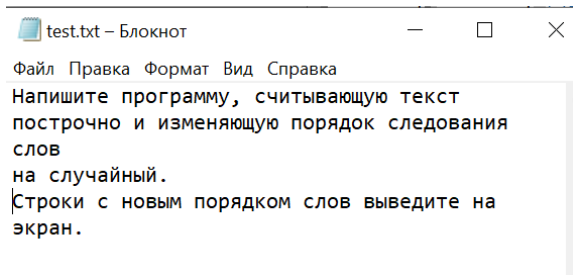
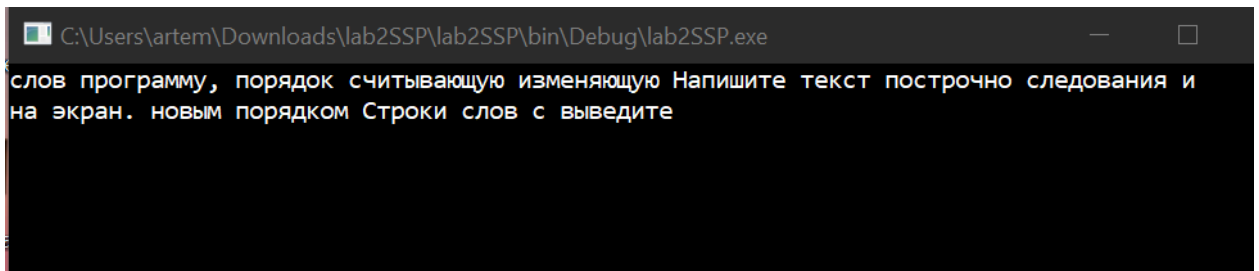
namespace lab2SSP
{
    class Program
    {
        static void Main(string[] args)
        {
            Task1();

            Console.ReadKey();
        }
        static void Task1()
        {
            string path = @"test.txt";
            Dictionary<int, List<string>> array = new Dictionary<int, List<string>>();
            using (StreamReader stream = new StreamReader(path))
            {
                int count = 0;
                while (!stream.EndOfStream)
                {
                    string line = stream.ReadLine();
                    array.Add(count++, line.Split(' ').ToList<string>());
                }
            }
            Random r = new Random();
            List<string> temp = new List<string>();
            for (int i = 0; i < array.Count; i++)
            {
                if (r.Next(0, 2) == 1)
                {
                    temp = array.ElementAt(i).Value;
                    for (int j = 0; j < temp.Count; j++)
                    {
                        int x1 = r.Next(0, temp.Count);
                        int x2 = r.Next(0, temp.Count);
                        string buff = temp[x2];
                        temp[x2] = temp[x1];
                        temp[x1] = buff;
                    }
                    foreach (var item in temp)
                    {
                        Console.Write($"{item} ");
                    }
                    Console.WriteLine();
                }
                else
                    continue;
            }
        }
    }
}
```

```

    }
}
}

```



Задание 2

Написать консольную утилиту, обрабатывающую ввод пользователя и дополнительные ключи. Проект упаковать в jar-файл, написать bat-файл для запуска.

Утилита paste выполняет слияние строк/столбцов из файлов и выводит результат в стандартный вывод.

Формат использования: paste [options] [file1 [file2]..], где ключи имеют следующее значение:

- -s Меняет положение строк со столбцами;
- -d разделитель Меняет разделитель на указанный (по умолчанию TAB)

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text.RegularExpressions;

```

```

namespace lab2_1SSP
{
    class PasteException : Exception
    {
        public PasteException(string message) : base(message) { }
    }
    class Program
    {
        static void Main(string[] args)
        {
            Task();
        }
        static void Task()
        {
            string line = "";
            while (true)
            {

```

```

        Console.WriteLine("~ ");
        line = Console.ReadLine();
        if (line.Equals("help paste") || line.Equals("help"))
        {
            ShowCommand();
        }
        else if (line.Equals("clear"))
        {
            Console.Clear();
        }
        else if (!line.Equals("quit"))
        {
            string[] array = line.Split(' ');
            if (array[0] == "")
            {
                continue;
            }
            List<string> listTextDocuments = new List<string>();
            Dictionary<int, List<string>> dictionaryOutput = new Dictionary<int,
List<string>>>();
            SelectNameTextDocuments(ref listTextDocuments, array);
            List<Dictionary<int, List<string>>> listDictionary =
GetListDictionary(listTextDocuments);
            try
            {
                if (!CheckPaste(array))
                {
                    throw new PasteException("Incorrect syntax");
                }
                if (listDictionary.Count == 0)
                {
                    throw new PasteException("TXT-File(s) not found");
                }
                if (CheckPaste(array))
                {
                    dictionaryOutput = OptionsS(listDictionary, array);
                    dictionaryOutput = OptionsD(dictionaryOutput, array);
                }

                OutputDictionary(ref dictionaryOutput);
            }
            catch (PasteException ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
        else break;
    }
}
static void ShowCommand()
{
    Console.WriteLine("paste [options] [file1 file2 ...]");
    Console.WriteLine("\n[options]:\n\t-s -> Changes the position of rows with
columns" +
"\n\t-d -> delimiter Change the delimiter to the specified one (TAB by
default)");
}
static void SelectNameTextDocuments(ref List<string> listTextDocuments, params
string[] array)
{
    foreach (var item in array)
    {
        if (item.Contains(".txt"))
        {
            listTextDocuments.Add(item);
        }
    }
}

```

```

    }
}
static bool CheckPaste(params string[] array)
{
    if (array.Length > 0)
    {
        if (array[0] == "paste")
        {
            return true;
        }
    }
    return false;
}
static bool CheckOptionsS(params string[] array)
{
    if (array.Length > 3)
    {
        for (int i = 0; i < 4; i++)
        {
            if (array[i].Contains("-s"))
            {
                return true;
            }
        }
    }
    return false;
}
static Dictionary<int, List<string>> OptionsS(List<Dictionary<int, List<string>>>
listDictionary, params string[] array)
{
    Dictionary<int, List<string>> tempDictionary = new Dictionary<int,
List<string>>>();
    List<string> tempList = new List<string>();
    if (CheckOptionsS(array))
    {
        int count = 0;
        for (int i = 0; i < listDictionary.Count; i++)
        {
            tempList = new List<string>();
            for (int list = 0; list < listDictionary.Max(z => z.Count); list++)
            {
                try
                {
                    if (list < listDictionary[i].Count)
                    {
                        string outputLine = "";
                        foreach (var item in listDictionary[i][list].ToList())
                        {
                            outputLine += item + " ";
                        }
                        tempList.Add(outputLine);
                    }
                    else
                        throw new Exception();
                }
                catch (Exception)
                {
                    continue;
                }
            }
            tempDictionary.Add(count++, tempList);
        }
        return tempDictionary;
    }
}

```

```

else
{
    int count = 0;
    for (int list = 0; list < listDictionary.Max(z => z.Count); list++)
    {
        tempList = new List<string>();
        for (int i = 0; i < listDictionary.Count; i++)
        {
            try
            {
                if (list < listDictionary[i].Count)
                {
                    string outputLine = "";
                    foreach (var item in listDictionary[i][list].ToList())
                    {
                        outputLine += item + " ";
                    }
                    tempList.Add(outputLine);
                }
                else
                {
                    throw new Exception();
                }
            }
            catch (Exception)
            {
                continue;
            }
        }
        tempDictionary.Add(count++, tempList);
    }
    return tempDictionary;
}

}

static bool CheckOptionsD(params string[] array)
{
    if (array.Length > 3)
    {
        for (int i = 0; i < 4; i++)
        {
            if (array[i].Contains("-d"))
            {
                return true;
            }
        }
    }
    return false;
}

static Dictionary<int, List<string>> OptionsD(Dictionary<int, List<string>>
listDictionary, params string[] array)
{
    Dictionary<int, List<string>> tempDictionary = new Dictionary<int,
List<string>>>();
    if (CheckOptionsD(array))
    {
        int index = -1;
        for (int i = 0; i < array.Length; i++)
        {
            if (array[i].Contains("-d"))
            {
                index = ++i;
                continue;
            }
        }
    }
}

```

```

        if (array[index].Contains(".txt"))
        {
            for (int i = 0; i < listDictionary.Count; i++)
            {
                for (int j = 0; j < listDictionary[i].Count; j++)
                {
                    listDictionary[i][j] += "\t";
                }
            }
        }
        else if (Regex.IsMatch(array[index], @"[\a\b\f\n\r\t\v\\"]"))
        {
            int count = 0;
            Dictionary<string, string> dictionaryEscapes = new Dictionary<string,
string>()
            {
                {"\\a", "\a" },
                {"\\b", "\b" },
                {"\\f", "\f" },
                {"\\n", "\n" },
                {"\\r", "\r" },
                {"\\t", "\t" },
                {"\\v", "\v" },
            };
            List<string> number = new List<string>();
            foreach (var key in dictionaryEscapes.Keys)
            {
                if (array[index].Contains(key))
                {
                    number.Add(key);
                }
            }
            for (int i = 0; i < listDictionary.Count; i++)
            {
                for (int j = 0; j < listDictionary[i].Count; j++)
                {
                    if (count >= number.Count)
                        count = 0;
                    listDictionary[i][j] =
listDictionary[i][j].Remove(listDictionary[i][j].Length - 1, 1) +
dictionaryEscapes[number[count++]];
                }
            }
        }
        else if (Regex.IsMatch(array[index], @"\W"))
        {
            int count = 0;
            List<char> symbol = array[index].ToList<char>();
            for (int i = 0; i < listDictionary.Count; i++)
            {
                for (int j = 0; j < listDictionary[i].Count; j++)
                {
                    if (count >= symbol.Count)
                        count = 0;
                    listDictionary[i][j] =
listDictionary[i][j].Remove(listDictionary[i][j].Length - 1, 1) + symbol[count++];
                }
            }
        }
        return listDictionary;
    }
    static List<Dictionary<int, List<string>>> GetListDictionary(List<string> names)
    {


```

```

        List<Dictionary<int, List<string>>> listDictionary = new List<Dictionary<int,
List<string>>>>();
        try
        {
            Dictionary<int, List<string>> dictionary;
            foreach (var file in names)
            {
                using (StreamReader stream = new StreamReader(file))
                {
                    dictionary = new Dictionary<int, List<string>>>();
                    int count = 0;
                    while (!stream.EndOfStream)
                    {
                        string line = stream.ReadLine();
                        dictionary.Add(count++, line.Split(' ').ToList<string>());
                    }
                    listDictionary.Add(dictionary);
                }
            }
            return listDictionary;
        }
        catch (FileNotFoundException ex)
        {
            Console.WriteLine(ex.Message);
            return listDictionary;
        }
    }

    static void OutputDictionary(ref Dictionary<int, List<string>> dictionary)
    {
        for (int i = 0; i < dictionary.Count; i++)
        {
            for (int j = 0; j < dictionary[i].Count; j++)
            {
                Console.Write(dictionary[i][j]);
            }
            Console.WriteLine();
        }
    }
}

```

 compile.bat – Блокнот

Файл Правка Формат Вид Справка

csc /t:exe Program.cs

start Program.exe


```
C:\Users\artem\Downloads\lab2SSP\lab2_1SSP\bin\Debug\lab2_1SSP.exe
~ help paste
paste [options] [file1 file2 ...]

[options]:
    -s -> Changes the position of rows with columns
    -d -> delimiter Change the delimiter to the specified one (TAB by default)
~ paste names.txt numbers.txt
Mark Smith 555-1234
Bobby Brown 555-9876
Sue Miller 555-6743
Jenny Igotit 867-5309
~ paste -d names.txt numbers.txt
Mark Smith      555-1234
Bobby Brown     555-9876
Sue Miller      555-6743
Jenny Igotit    867-5309
~ paste -s names.txt numbers.txt
Mark Smith Bobby Brown Sue Miller Jenny Igotit
555-1234 555-9876 555-6743 867-5309
~ paste -s -d names.txt numbers.txt
Mark Smith      Bobby Brown      Sue Miller      Jenny Igotit
555-1234        555-9876        555-6743        867-5309
~ paste -d \t\n names.txt numbers.txt
Mark Smith
555-1234
Bobby Brown
555-9876
Sue Miller
555-6743
Jenny Igotit
867-5309
~ paste
TXT-File(s) not found
~ erty
Incorrect syntax
~
```

Вывод: приобрел базовые навыки работы с файловой системой в C#.