Міністерство освіти і науки України

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"

Факультет інформатики та обчислювальної техніки Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 1 з дисципліни

«Основи програмування-2. Методології програмування»

«Файли даних»

Варіант 06

Виконав студент ІП-11 Головатюк Владислав Ігорович

(шифр, прізвище, ім'я, по батькові)

Перевірив

( прізвище, ім'я, по батькові)

Київ 2022

**Лабораторна робота №1**

**Файли даних. Текстові файли**

**Мета**

***Мета роботи:*** вивчити особливості створення і обробки текстових файлів даних.

***Завдання:*** Створити два текстових файли. Переписати в новий текстовий файл рядки, які є у другому файлі, і прибрати рядки, які присутні в першому. Визначити кількість таких рядків. Вивести вміст вихідного і створеного файлів.

***Постановка задачі:*** В умовах модульності, потрібно відокремити функції/методи, які відповідаюсь 3 головним задачам: робота з файлами, консолью та вмістом файлів.

* Робота з файлами полягає в доступі до контекту для зчитування і запису в файл у двох режимах: доповнення та повний перепис вмісту.
* Робота з консолью потребує отримування вводу від користувача та сповіщення його про необхідність вводу тієї чи іншої операції.
* Робота з текстом повинна виконати основу частину завдання, тобто формування контенту вихідного файла відповідно умові та підрахунок елементів, які задовольняють цю умову.

1. **С#**
   1. **Код:**
      1. **Program.cs**

using System;

namespace Lab1

{

internal class Program

{

static void Main(string[] args)

{

string firstFileName = "firstData",

secondFileName = "secondData",

outputFileName = "outputData";

TextWorker = new TextWorker();

ConsoleWorker consoleWorker = new ConsoleWorker();

FileWorker fileWorker = new FileWorker();

bool result1 = fileWorker.CreateFile(firstFileName);

bool result2 = fileWorker.CreateFile(secondFileName);

fileWorker.CreateFile(outputFileName);

if (consoleWorker.GetWriteMode() == WriteMode.Write)

fileWorker.WriteData(firstFileName, consoleWorker.GetMultilineInput());

else

{

if (result1 == false)

consoleWorker.PrintFileContent("\nFirstData.txt content: ", fileWorker.ReadData(firstFileName));

fileWorker.AppendData(firstFileName, consoleWorker.GetMultilineInput());

}

Console.WriteLine("====================");

if (consoleWorker.GetWriteMode() == WriteMode.Write)

fileWorker.WriteData(secondFileName, consoleWorker.GetMultilineInput());

else

{

if (result2 == false)

consoleWorker.PrintFileContent("\nSecondData.txt content: ", fileWorker.ReadData(secondFileName));

fileWorker.AppendData(secondFileName, consoleWorker.GetMultilineInput());

}

Console.WriteLine("====================");

string[] fromFirstFile = fileWorker.ReadData(firstFileName);

string[] fromSecondFile = fileWorker.ReadData(secondFileName);

string[] resultContent = textWorker.SecondWithoutFirst(fromFirstFile, fromSecondFile);

Console.WriteLine("Second array without first one's elements:");

foreach (string item in resultContent)

{

Console.WriteLine(item);

}

Console.WriteLine("Count of output elements: " + resultContent.Length);

fileWorker.WriteData(outputFileName, resultContent);

Console.ReadLine();

}

}

}

* + 1. **ConsoleWorker.cs**

using System;

using System.Linq;

namespace Lab1

{

internal class ConsoleWorker

{

public string[] GetMultilineInput(ConsoleKey exitKey = ConsoleKey.E)

{

PrintInputHint(exitKey);

bool keepEntering;

string result = "";

do

{

var key = Console.ReadKey(true);

keepEntering = key.Key != exitKey || key.Modifiers != ConsoleModifiers.Alt;

if (keepEntering)

{

switch (key.Key)

{

case ConsoleKey.Enter:

result += "\n";

Console.CursorLeft = 0;

Console.CursorTop++;

break;

case ConsoleKey.Backspace:

if (Console.CursorLeft == 0)

break;

Console.CursorLeft--;

Console.Write(" ");

Console.CursorLeft--;

result = result.Remove(result.Length - 1, 1);

break;

default:

Console.Write(key.KeyChar);

result += key.KeyChar.ToString();

break;

}

}

} while (keepEntering);

Console.Write("\n");

return result.Split(new char[] { '\n' }, StringSplitOptions.RemoveEmptyEntries);

}

public WriteMode GetWriteMode()

{

string input = "";

string[] possibleInputs = new string[] { "a", "w" };

do

{

Console.WriteLine("Choose write mode: print \'w\' to open in write mode,\n" +

"print \'a\' to open in append mode");

input = Console.ReadLine();

} while (possibleInputs.Contains(input.ToLower()) == false);

switch (input.ToLower())

{

case "a":

return WriteMode.Append;

case "w":

return WriteMode.Write;

default:

return WriteMode.Write;

}

}

public void PrintFileContent(string header, string[] content)

{

Console.WriteLine(header + "\n====================");

foreach (var item in content)

{

Console.WriteLine(item);

}

Console.WriteLine("====================");

}

private void PrintInputHint(ConsoleKey key)

{

Console.WriteLine($"Enter text, which would be written to next file\n" +

$"[To exit entering mode press ALT+{key}]");

}

}

enum WriteMode

{

Write,

Append

}

}

* + 1. **FileWorker.cs**

using System.IO;

namespace Lab1

{

internal class FileWorker

{

public bool CreateFile(string fileName)

{

if (File.Exists($"{fileName}.txt") == false)

{

File.Create($"{fileName}.txt").Close();

return true;

}

return false;

}

public void WriteData(string fileName, string[] data)

{

if (File.Exists($"{fileName}.txt") == false)

CreateFile($"{fileName}.txt");

File.WriteAllLines($"{fileName}.txt", data);

}

public void AppendData(string fileName, string[] data)

{

if (File.Exists($"{fileName}.txt") == false)

CreateFile($"{fileName}.txt");

File.AppendAllLines($"{fileName}.txt", data);

}

public string[] ReadData(string fileName)

{

return File.Exists($"{fileName}.txt") ? File.ReadAllLines($"{fileName}.txt") : new string[0];

}

}

}

* + 1. **TextWorker.cs**

using System.Linq;

namespace Lab1

{

internal class TextWorker

{

public string[] SecondWithoutFirst(string[] first, string[] second)

{

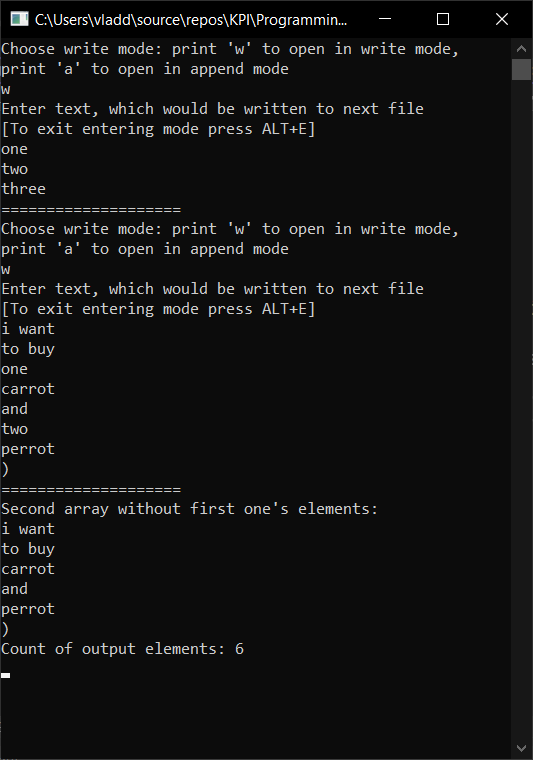
return second.Except(first).ToArray();

}

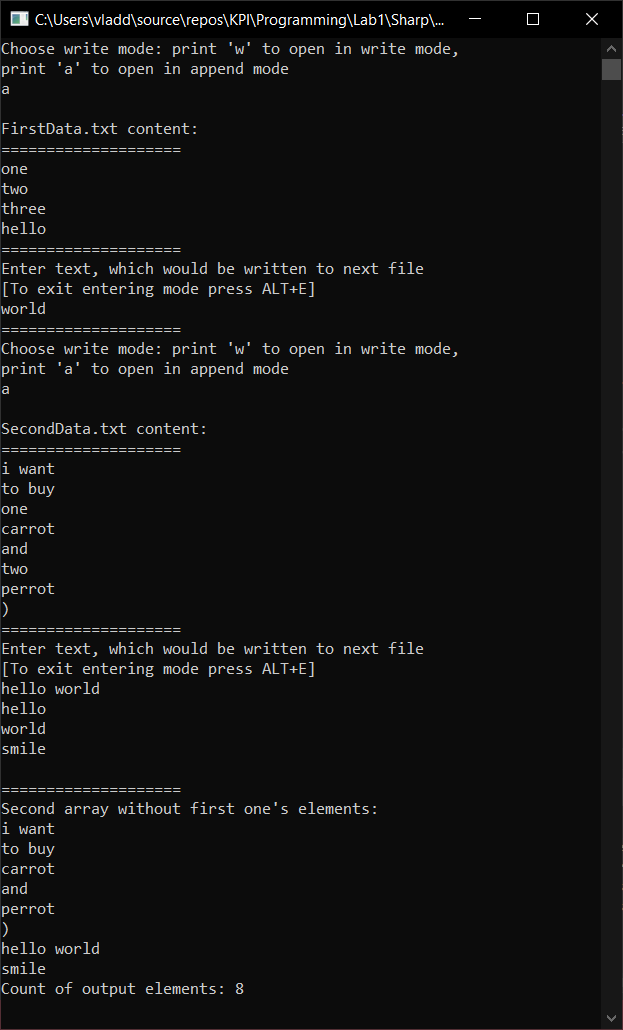
}

}

* 1. **Запуск програми**
     1. **Режим перезапису:**



* + 1. **Режим доповнення:**



1. **Python**
   1. **Код**
      1. **main.py**

**from** os **import** system  
**import** file\_worker **as** file  
**from** file\_worker **import** OpenMode  
**import** console\_worker **as** console  
**import** text\_worker **as** text  
  
  
**def** main():  
 open\_mode = console.get\_write\_mode()  
 **if** open\_mode == OpenMode.append:  
 console.print\_existing\_content(**"First file content: "**, file.read\_file(**'first.txt'**))  
 file.write\_to\_file(**'first.txt'**, open\_mode, console.get\_multiline\_input())  
 open\_mode = console.get\_write\_mode()  
 **if** open\_mode == OpenMode.append:  
 console.print\_existing\_content(**"Second file content: "**, file.read\_file(**'second.txt'**))  
 file.write\_to\_file(**'second.txt'**, open\_mode, console.get\_multiline\_input())  
  
 from\_first = file.read\_file(**'first.txt'**)  
 from\_second = file.read\_file(**'second.txt'**)  
 output\_content = text.second\_without\_first(from\_first, from\_second)  
  
 print(**'====================='**)  
 print(**'Second file, where excluded lines from first file: '**)  
 **for** line **in** output\_content:  
 print(line)  
 print(**f'Count of lines is: {**text.get\_content\_length(output\_content)**}'**)  
 file.write\_to\_file(**'output.txt'**, OpenMode.write, output\_content)  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 main()  
 system(**'pause'**)

* + 1. **file\_worker.py**

**import** os.path  
**from** enum **import** Enum  
  
  
**class** OpenMode(Enum):  
 read = **'r'** write = **'w'** append = **'a'** binary = **'b'  
  
  
def** write\_to\_file(file\_name: str, open\_mode: OpenMode, content: list[str]):  
 **if** open\_mode.value != **'w' and** open\_mode.value != **'a' and** open\_mode.value != **'b'**:  
 **return False  
 with** open(file\_name, open\_mode.value) **as** file:  
 **for** line **in** content:  
 file.write(line + **'\n'**)  
 **return True  
  
  
def** read\_file(file\_name: str) -> list[str]:  
 **if** os.path.exists(file\_name):  
 **with** open(file\_name, **'r'**) **as** file:  
 **return** file.read().split(**'\n'**)  
 **else**:  
 **return** []

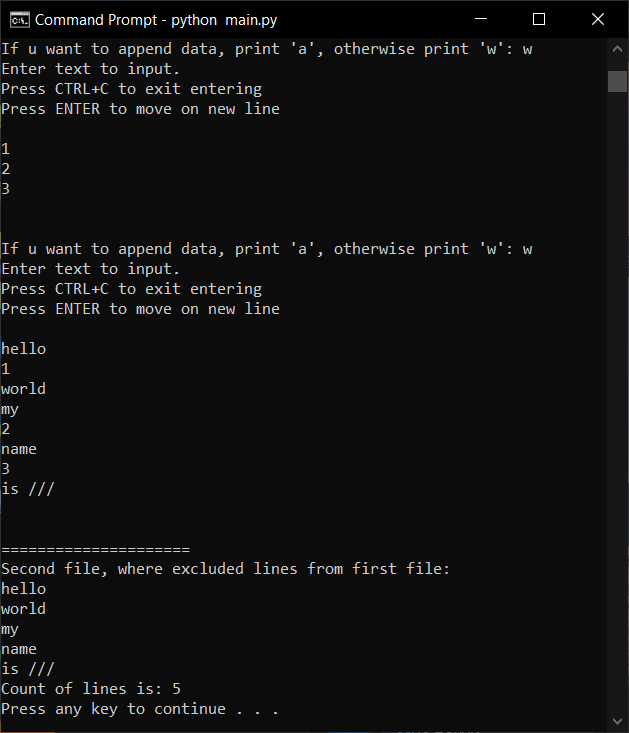
* + 1. **console\_worker.py**

**from** file\_worker **import** \*  
  
**def** get\_multiline\_input() -> list[str]:  
 print(**'Enter text to input.'  
 '\nPress CTRL+C to exit entering'  
 '\nPress ENTER to move on new line\n'**)  
 result = []  
 **while True**:  
 **try**:  
 result.append(input())  
 **except** KeyboardInterrupt:  
 print(**'\n'**)  
 **return** result  
**def** print\_existing\_content(header: str, content: list[str]):  
 **if** len(content) == 0:  
 **return** print(header)  
 print(**'==============='**)  
 **for** line **in** content:  
 print(line)  
 print(**'==============='**)  
  
**def** get\_write\_mode() -> OpenMode:  
 **return** OpenMode.append **if** input(  
 **'If u want to append data, print \'a\', otherwise print \'w\': '**) == **'a' else** OpenMode.write

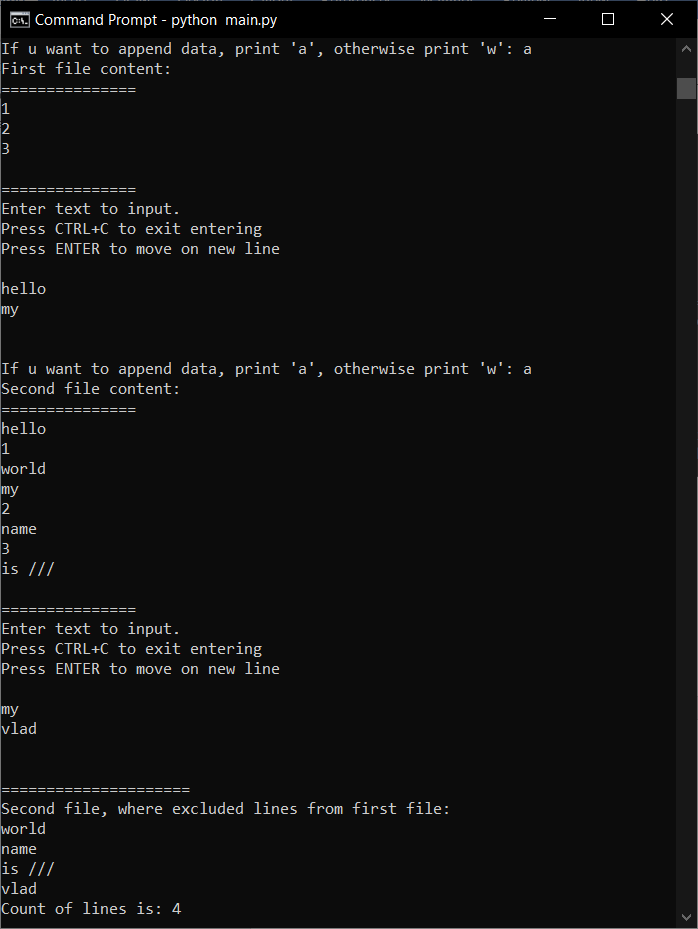
* + 1. **text\_worker.py**

**def** second\_without\_first(first: list[str], second: list[str]) -> list[str]:  
 **return** [i **for** i **in** second **if** i **not in** first]  
  
  
**def** get\_content\_length(content: list[str]) -> int:  
 **return** len(content)

* 1. **Запуск програми**
     1. **Режим перезапусу**



* + 1. **Режим доповнення**



***Висновок:*** В процесі виконання лабораторної роботи, були опановані та відпрацьовані навички у роботі із файлами, а саме запису у файл, считування та доповнення вмісту програмним шляхом на компільованій так інтерпритованій мовах.