Міністерство освіти і науки України
Національний технічний університет України «КПІ ім. Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Програмування інтелектуальних інформаційних систем

ЗВІТ

до лабораторних робіт

Виконав студент ІП-01 Хернуф Валід _____
(№ групи, прізвище, ім'я, по батькові)

Прийняв ас. Очеретяний О. К. _____
(посада, прізвище, ім'я, по батькові)

Київ 2022

1. Завдання лабораторної роботи

Завдання 1:

Обчислювальна задача тут тривіальна: для текстового файлу ми хочемо відобразити N (наприклад, 25) найчастіших слів і відповідну частоту їх повторення, упорядковано за зменшенням. Слід обов'язково нормалізувати використання великих літер і ігнорувати стоп-слова, як «the», «for» тощо. Щоб все було просто, ми не піклуємося про порядок слів з однаковою частотою повторень. Ця обчислювальна задача відома як **term frequency**.

Ось такий вигляд матимуть ввід і відповідно вивід результату програми:
Input:
White tigers live mostly in India
Wild lions live mostly in Africa

Output:
live - 2
mostly - 2
africa - 1
india - 1
lions - 1
tigers - 1
white - 1
wild - 1

Завдання 2:

Тепер, нам потрібно виконати задачу, що називається словниковим індексуванням. Для текстового файлу виведіть усі слова в алфавітному порядку разом із номерами сторінок, на яких Ці слова знаходяться. Ігноруйте всі слова, які зустрічаються більше 100 разів. Припустимо, що сторінка являє собою послідовність із 45 рядків. Наприклад, якщо взяти книгу Pride and Prejudice, перші кілька записів індексу будуть:

abatement - 89

abhorrence - 101, 145, 152, 241, 274, 281

abhorrent - 253

abide - 158, 292

2. Опис алгоритму

Завдання 1:

1. ПОЧАТОК
2. Зчитати вхідні дані (директорії файлів початкового тексту, результату, стоп-слів, кількість слів в результаті)
3. Ініціалізуємо масив стоп-слів та задаємо йому початковий розмір
4. Парсимо стоп слова:
   4.1.Зчитуємо слово з файлу стоп-слів
   4.2.ЯКЩО слово порожнє, то ПЕРЕЙТИ до пункту 5.
   4.3.Додаємо слово до масиву стоп-слів.
   4.4.ЯКЩО масив стоп-слів повний, збільшуємо його в два рази.
   4.5.ПЕРЕЙТИ до пункту 4.1.
5. Ініціалізуємо масив слів та масив кількості слів, та задаємо їм початковий розмір.
6. Зчитуємо текст та визначаємо кількість слів:
   6.1.Зчитуємо слово з текстового файлу.
   6.2.ЯКЩО слово порожнє, то ПЕРЕЙТИ до пункту 7.
   6.3. Нормалізуємо слово:
      6.3.1. Зчитуємо символ з слова.
      6.3.2. ЯКЩО символ пустий, то ПЕРЕЙТИ до пункту 6.4.
      6.3.3. ЯКЩО символ є великою літерою, то ми його перетворюємо на маленьку.
      6.3.4. ПЕРЕЙТИ до пункту 6.3.1.
   6.4. Приберемо непотрібні символи:
      6.4.1. Зчитуємо символ з слова
      6.4.2. ЯКЩО символ пустий, то ПЕРЕЙТИ до пункту 6.5.
      6.4.3. ЯКЩО символ є літерою, або символом «’», «-», «`», що знаходиться всередині слова, то зберігаємо цей символ в слові.
      6.4.4. ПЕРЕЙТИ до пункту 6.4.1.
   6.5. ЯКЩО слово після трансформацій пусте, то ПЕРЕЙТИ до пункту 6.1.
   6.6.Звіряємо зі стоп-словами:
      6.6.1. ЯКЩО ми не перевірили усі стоп-слова слова то:
         6.6.1.1. ЯКЩО слово співпадає з стоп-словом, то ПЕРЕЙТИ до пункту 6.1.
         6.6.1.2. Беремо наступне стоп слово.
         6.6.1.3.ПЕРЕЙТИ до пункту 6.6.1.
   6.7. Перевіряємо чи є в масиві таке слово:
      6.7.1.ЯКЩО ми не перевірили усі слова то:
         6.7.1.1. ЯКЩО слово співпадає з іншим словом, збільшити кількість слів на 1 та ПЕРЕЙТИ до пункту 6.1.

6.7.1.2. Беремо наступне слово
6.7.1.3. ПЕРЕЙТИ до пункту 6.7.1.
6.8. Додаємо слово до масива слів та виставляємо кількість слів 1.
6.9. ЯКЩО масив слів заповнений, то збільшуємо його та масив кількості слів в два рази.
6.10. ПЕРЕЙТИ до пункту 6.1.
7. Сортуємо слова за кількістю:
7.1. ЯКЩО ми не перевірили всі елементи масива:
7.1.1. Знаходимо найбільший елемент в незачепленої частині масива.
7.1.2. Міняємо місцями найбільший та відповідний елемент.
7.1.3.ПЕРЕЙТИ до пункту 7.1.
8. Записуємо результат до файлу
9. КІНЕЦЬ

Завдання 2:

1. ПОЧАТОК
2. Зчитати вхідні дані (директорії файлів початкового тексту, результату, стоп-слів)
3. Ініціалізуємо масив стоп-слів та задаємо йому початковий розмір
4. Парсимо стоп слова:
4.1.Зчитуємо слово з файлу стоп-слів
4.2.ЯКЩО слово порожнє, то ПЕРЕЙТИ до пункту 5.
4.3.Додаємо слово до масиву стоп-слів.
4.4.ЯКЩО масив стоп-слів повний, збільшуємо його в два рази.
4.5.ПЕРЕЙТИ до пункту 4.1.
5. Ініціалізуємо масив слів, масив кількості слів, булевий масив, що відзначає зустрічалось слово на цій сторінці чи ні. Та масив сторінок, на яких зустрічаються слова. Задаємо їм початковий розмір.
6. Зчитуємо текст та визначаємо сторінки, на яких вони зустрічаються:
6.1. ЯКЩО нова сторінка, то заповнюємо весь булевий масив значеннями хибності.
6.2.Зчитуємо слово з текстового файлу.
6.3.ЯКЩО слово порожнє, то ПЕРЕЙТИ до пункту 7.
6.4. Нормалізуємо слово:
6.4.1. Зчитуємо символ з слова.
6.4.2. ЯКЩО символ пустий, то ПЕРЕЙТИ до пункту 6.4.
6.4.3. ЯКЩО символ є великою літерою, то ми його перетворюємо на маленьку.
6.4.4. ПЕРЕЙТИ до пункту 6.3.1.
6.5. Приберемо непотрібні символи:

6.5.1. Зчитуємо символ з слова
6.5.2. ЯКЩО символ пустий, то ПЕРЕЙТИ до пункту 6.5.
6.5.3. ЯКЩО символ є літерою, або символом «’», «-», «`», що знаходиться всередині слова, то зберігаємо цей символ в слові.
6.5.4. ПЕРЕЙТИ до пункту 6.4.1.
6.6. ЯКЩО слово після трансформацій пусте, то ПЕРЕЙТИ до пункту 6.1.
6.7.Звіряємо зі стоп-словами:
6.7.1. ЯКЩО ми не перевірили усі стоп-слова слова то:
6.7.1.1. ЯКЩО слово співпадає з стоп-словом, то ПЕРЕЙТИ до пункту 6.1.
6.7.1.2. Беремо наступне стоп слово.
6.7.1.3.ПЕРЕЙТИ до пункту 6.6.1.
6.8. Перевіряємо чи є в масиві таке слово:
6.8.1.ЯКЩО ми не перевірили усі слова то:
6.8.1.1. ЯКЩО слово співпадає з іншим словом то:
6.8.1.1.1. Збільшити кількість слів на 1.
6.8.1.1.2. ЯКЩО відповідне слову значення в булевом масиві хибне, то додаємо сторінку до відповідного елемента в масиві зі сторінками.
6.8.1.1.3. Виставляємо відповідне значення в булевом масиві на істинність. ПЕРЕЙТИ до пункту 6.1.
6.8.1.2. Беремо наступне слово. ПЕРЕЙТИ до пункту 6.7.1.
6.9. Додаємо слово до масива слів, виставляємо кількість слів 1 та додаємо сторінку до відповідного елемента з масивом сторінок.
6.10. ЯКЩО масив слів заповнений, то збільшуємо його, масив кількості слів та масив сторінок в два рази.
6.11. ПЕРЕЙТИ до пункту 6.1.
7. Сортуємо слова по алфавіту:
7.1. ЯКЩО ми не перевірили всі елементи масива:
7.1.1. Знаходимо найкращій за алфавітом елемент в незачепленої частині масива.
7.1.2. Міняємо місцями найкращій за алфавітом та відповідний елемент.
7.1.3.ПЕРЕЙТИ до пункту 7.1.
8. Записуємо слова, які зустрілися менш, ніж 100 разів до файлу разом зі сторінками.
9. КІНЕЦЬ

3. Опис програмного коду

Завдання 1:

```
#include <fstream>
```

```cpp
#include <iostream>
using namespace std;

int main()
{
	const int arrStartSize = 4;
	string txtDirectory;
	string stopWordDirectory;
	string resultDirectory;
	string temp;
	int amount;
	cout << "Enter directory of file with text: ";
	cin >> txtDirectory;
	cout << "\nEnter directory of file with stop words: ";
	cin >> stopWordDirectory;
	cout << "\nEnter directory of file where the result will be written: ";
	cin >> resultDirectory;
	cout << "\nEnter amount of most common words that you want to see: ";
	cin >> amount;
	ifstream stopWordFile(stopWordDirectory);
	string* stopWord = new string[arrStartSize];
	int currStopWordMax = arrStartSize;
	int arrStopWordSize = 0;
stopWordParse:
	stopWordFile >> temp;
	if (temp == "") goto endStopWordParse;
	temp += " ";
	stopWord[arrStopWordSize] = temp;
	temp = "";
	arrStopWordSize++;
	if (arrStopWordSize == currStopWordMax)
	{
		currStopWordMax *= 2;
		string* newArr = new string[currStopWordMax];
		int i = 0;
	fillNewArrStopWord:
		newArr[i] = stopWord[i];
		i++;
		if (i < arrStopWordSize) goto fillNewArrStopWord;
		delete[] stopWord;
		stopWord = newArr;
	}
	goto stopWordParse;
endStopWordParse:
	stopWordFile.close();
	ifstream txtFile(txtDirectory);
	string* words = new string[arrStartSize];
	int* wordsAmount = new int[arrStartSize];
	int currWordMax = arrStartSize;
	int arrWordSize = 0;
	int charsInWord;
	string withSymbols;
txtParse:
	txtFile >> temp;
	if (temp == "") goto endTxtParse;
	else
	{
		temp += " ";
		int i = 0;
	removeCapLetters:
		if (temp[i] == ' ') goto endCapLetters;
		else if (temp[i] >= 65 && temp[i] <= 90) temp[i] += 32;
		i++;
		goto removeCapLetters;
	endCapLetters:
```

```
		i = 0;
		withSymbols = "";
		charsInWord = 0;
	removeSymbols:
		if (temp[i] == ' ') goto endRemoveSymbols;
		if ((temp[i] >= 97 && temp[i] <= 122) || (charsInWord > 0 && (temp[i + 1]
>= 97 && temp[i + 1] <= 122) && (temp[i] == 39 || temp[i] == 96 || temp[i] == 45)))
		{
			charsInWord++;
			withSymbols += temp[i];
		}
		i++;
		goto removeSymbols;
	endRemoveSymbols:
		if (withSymbols == "") goto txtParse;
		temp = withSymbols + " ";
		i = 0;
	checkStopWords:
		if (i < arrStopWordSize)
		{
			if (temp == stopWord[i]) goto txtParse;
			i++;
			goto checkStopWords;
		}
		i = 0;
	checkExist:
		if (i < arrWordSize)
		{
			if (temp == words[i])
			{
				wordsAmount[i]++;
				goto txtParse;
			}
			i++;
			goto checkExist;
		}
		words[arrWordSize] = temp;
		wordsAmount[arrWordSize] = 1;
		temp = "";
		arrWordSize++;
	}
	if (arrWordSize == currWordMax)
	{
		currWordMax *= 2;
		string* newArr = new string[currWordMax];
		int* newArrAmount = new int[currWordMax];
		int i = 0;
	fillNewArrWord:
		newArr[i] = words[i];
		newArrAmount[i] = wordsAmount[i];
		i++;
		if (i < arrWordSize) goto fillNewArrWord;
		delete[] words;
		delete[] wordsAmount;
		words = newArr;
		wordsAmount = newArrAmount;
	}
	goto txtParse;
endTxtParse:
	txtFile.close();
	delete[] stopWord;
	amount = amount > arrWordSize ? arrWordSize : amount;
	int i = 0;
sortStart:
	if (i < amount)
```

```
	{
		int maxFound = 0;
		int j = i;
		int maxFoundJ;
	checkAll:
		if (j < arrWordSize)
		{
			if (maxFound < wordsAmount[j])
			{
				maxFound = wordsAmount[j];
				maxFoundJ = j;
			}
			j++;
			goto checkAll;
		}
		string tempWord = words[maxFoundJ];
		int tempAmount = wordsAmount[maxFoundJ];
		words[maxFoundJ] = words[i];
		wordsAmount[maxFoundJ] = wordsAmount[i];
		words[i] = tempWord;
		wordsAmount[i] = tempAmount;
		i++;
		goto sortStart;
	}
	ofstream resultFile(resultDirectory);
	i = 0;
writeResult:
	if (i < amount)
	{
		resultFile << words[i] << " - " << wordsAmount[i] << '\n';
		i++;
		goto writeResult;
	}
	resultFile.close();
	delete[] words;
	delete[] wordsAmount;
	return 0;
}
```

Завдання 2:

```
#include <fstream>
#include <iostream>
using namespace std;

int main()
{
	const int arrStartSize = 4;
	string txtDirectory;
	string stopWordDirectory;
	string resultDirectory;
	string temp;
	cout << "Enter directory of file with text: ";
	cin >> txtDirectory;
	cout << "\nEnter directory of file with stop words: ";
	cin >> stopWordDirectory;
	cout << "\nEnter directory of file where the result will be written: ";
	cin >> resultDirectory;
	ifstream stopWordFile(stopWordDirectory);
	string* stopWord = new string[arrStartSize];
	int currStopWordMax = arrStartSize;
	int arrStopWordSize = 0;
stopWordParse:
	stopWordFile >> temp;
	if (temp == "") goto endStopWordParse;
```

```
		temp += " ";
		stopWord[arrStopWordSize] = temp;
		temp = "";
		arrStopWordSize++;
		if (arrStopWordSize == currStopWordMax)
		{
			currStopWordMax *= 2;
			string* newArr = new string[currStopWordMax];
			int i = 0;
		fillNewArrStopWord:
			newArr[i] = stopWord[i];
			i++;
			if (i < arrStopWordSize) goto fillNewArrStopWord;
			delete[] stopWord;
			stopWord = newArr;
		}
		goto stopWordParse;
endStopWordParse:
		stopWordFile.close();
		string* words = new string[arrStartSize];
		string* wordsPages = new string[arrStartSize];
		int* wordsAmount = new int[arrStartSize];
		bool* usedThisPage = new bool[arrStartSize];
		int currWordMax = arrStartSize;
		int arrWordSize = 0;
		int line = 0;
		int pages = 0;
		int charsInWord;
		int tempPage;
		string withSymbols;
		string strInt;
		ifstream txtFile(txtDirectory);
txtParse:
		if (txtFile.peek() == '\n')
		{
			line++;
			txtFile.get();
			goto txtParse;
		}
		if (pages != line / 45)
		{
			pages++;
			int i = 0;
		boolLoop:
			if (i < arrWordSize)
			{
				usedThisPage[i] = false;
				i++;
				goto boolLoop;
			}
		}
		temp = "";
		txtFile >> temp;
		if (temp == "") goto endTxtParse;
		else
		{
			temp += " ";
			int i = 0;
		removeCapLetters:
			if (temp[i] == ' ') goto endCapLetters;
			else if (temp[i] >= 65 && temp[i] <= 90) temp[i] += 32;
			i++;
			goto removeCapLetters;
		endCapLetters:
			i = 0;
```

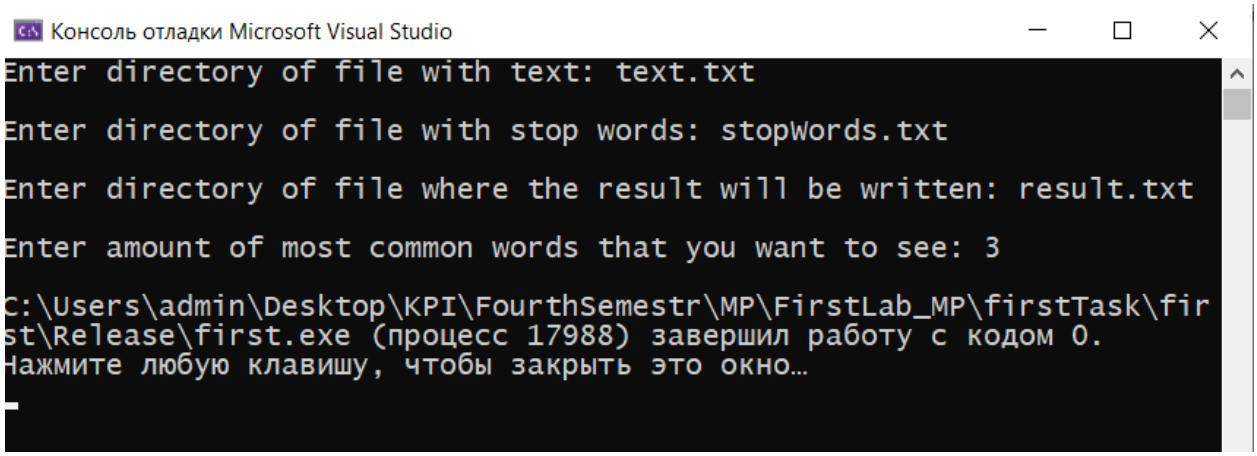
```
		withSymbols = "";
		charsInWord = 0;
	removeSymbols:
		if (temp[i] == ' ') goto endRemoveSymbols;
		if ((temp[i] >= 97 && temp[i] <= 122) || (charsInWord > 0 && (temp[i + 1]
>= 97 && temp[i + 1] <= 122) && (temp[i] == 39 || temp[i] == 96 || temp[i] == 45)))
		{
			charsInWord++;
			withSymbols += temp[i];
		}
		i++;
		goto removeSymbols;
	endRemoveSymbols:
		if (withSymbols == "")
		{
			goto txtParse;
		}
		temp = withSymbols + " ";
		i = 0;
	checkStopWords:
		if (i < arrStopWordSize)
		{
			if (temp == stopWord[i]) goto txtParse;
			i++;
			goto checkStopWords;
		}
		i = 0;
	checkExist:
		if (i < arrWordSize)
		{
			if (temp == words[i])
			{
				wordsAmount[i]++;
				if (!usedThisPage[i])
				{
					strInt = "";
					tempPage = pages + 1;
				fromIntToStr:
					if (tempPage != 0)
					{
						strInt = char(tempPage % 10 + 48) + strInt;
						tempPage /= 10;
						goto fromIntToStr;
					}
					wordsPages[i] += ", " + strInt;
				}
				usedThisPage[i] = true;
				goto txtParse;
			}
			i++;
			goto checkExist;
		}
		words[arrWordSize] = temp;
		wordsAmount[arrWordSize] = 1;
		usedThisPage[arrWordSize] = true;
		strInt = "";
		tempPage = (line / 45) + 1;
	fromIntToStr_sec:
		if (tempPage != 0)
		{
			strInt = char(tempPage % 10 + 48) + strInt;
			tempPage /= 10;
			goto fromIntToStr_sec;
		}
		wordsPages[i] += strInt;
```

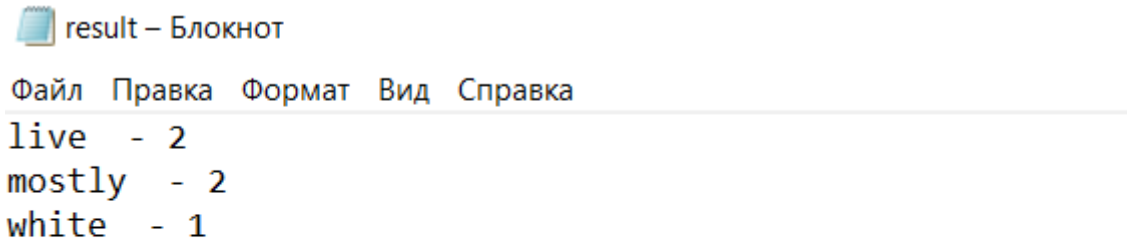
```
			temp = "";
			arrWordSize++;
		}
		if (arrWordSize == currWordMax)
		{
			currWordMax *= 2;
			string* newArr = new string[currWordMax];
			int* newArrAmount = new int[currWordMax];
			string* newArrPages = new string[currWordMax];
			bool* newUsedPages = new bool[currWordMax];
			int i = 0;
		fillNewArrWord:
			newArr[i] = words[i];
			newArrAmount[i] = wordsAmount[i];
			newArrPages[i] = wordsPages[i];
			newUsedPages[i] = usedThisPage[i];
			i++;
			if (i < arrWordSize) goto fillNewArrWord;
			delete[] words;
			delete[] wordsAmount;
			delete[] wordsPages;
			delete[] usedThisPage;
			words = newArr;
			wordsAmount = newArrAmount;
			wordsPages = newArrPages;
			usedThisPage = newUsedPages;
		}
		goto txtParse;
	endTxtParse:
		txtFile.close();
		delete[] stopWord;
		int i = 0;
	sort:
		if (i < arrWordSize)
		{
			string bestFound = "~";
			int j = i;
			int bestFoundJ;
		check:
			if (j < arrWordSize)
			{
				int k = 0;
			checkChars:
				if (bestFound[k] > words[j][k])
				{
					bestFoundJ = j;
					bestFound = words[j];
				}
				else if (bestFound[k] == words[j][k])
				{
					k++;
					goto checkChars;
				}
				j++;
				goto check;
			}
			temp = words[bestFoundJ];
			int tempAmount = wordsAmount[bestFoundJ];
			string tempPagesStr = wordsPages[bestFoundJ];
			words[bestFoundJ] = words[i];
			wordsAmount[bestFoundJ] = wordsAmount[i];
			wordsPages[bestFoundJ] = wordsPages[i];
			words[i] = temp;
			wordsAmount[i] = tempAmount;
			wordsPages[i] = tempPagesStr;
```

```
                i++;
                goto sort;
        }
        i = 0;
        ofstream resultFile(resultDirectory);
writeResult:
        if (i < arrWordSize)
        {
                if (wordsAmount[i] <= 100)
                {
                        resultFile << words[i] << "- " << wordsPages[i] << '\n';
                }
                i++;
                goto writeResult;
        }
        resultFile.close();
        delete[] words;
        delete[] wordsAmount;
        delete[] usedThisPage;
        delete[] wordsPages;
        return 0;
}
```

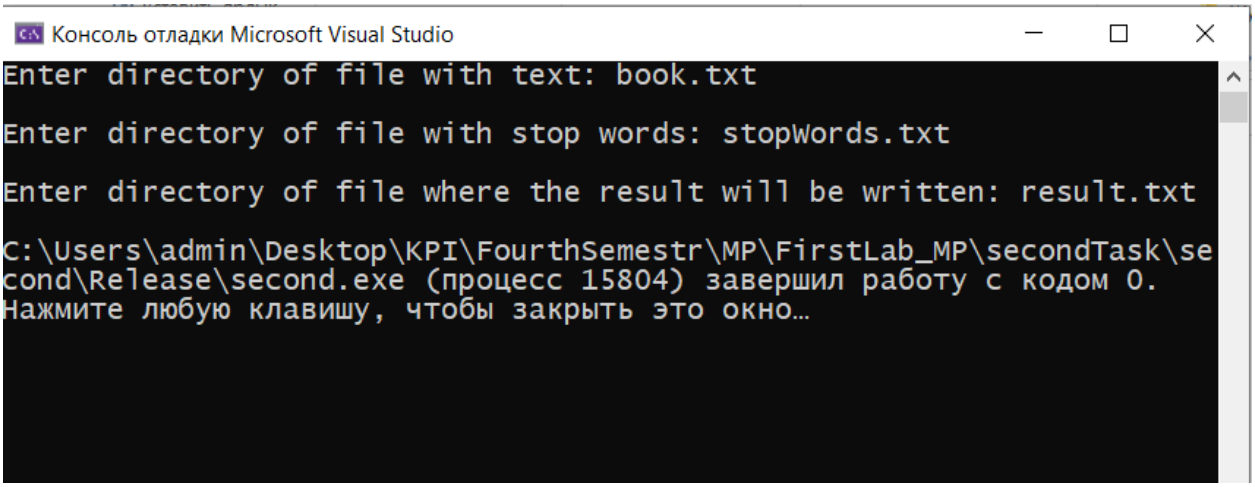
4. Скріншоти роботи програмного застосунку

Завдання 1:





Завдання 2:



result – Блокнот

Файл Правка Формат Вид Справка

```
a-shooting - 305
abatement - 99
abhorrence - 111, 160, 167, 263, 299, 306
abhorrent - 276
abide - 174, 318
abiding - 176
abilities - 72, 73, 107, 155, 171, 193
able - 19, 37, 58, 78, 84, 86, 88, 91, 97, 100, 107, 109, 110, 119, 126, 129, 131, 143, 145, 152, 156, 172, 177, 178, 183, 185, 187, 195, 205, 217, 220, 226, 227, 231, 233,
ablution - 119
abode - 59, 60, 66, 110, 122, 130, 176, 260
abominable - 32, 51, 71, 121, 161
abominably - 48, 133, 269, 299
abominate - 263, 296
abound - 101
abroad - 194, 196, 233, 288
abrupt - 203
abruptly - 41, 155
abruptness - 198
absence - 54, 56, 64, 77, 78, 90, 99, 100, 105, 106, 110, 111, 126, 150, 172, 194, 195, 197, 205, 207, 224, 232, 238, 283
absent - 30, 199, 225, 229
absolute - 77, 227, 253, 307
absolutely - 17, 25, 32, 92, 94, 125, 147, 166, 167, 171, 189, 203, 242, 260, 269, 299, 304
absurd - 60, 163, 171, 296, 302
absurdities - 127, 217
absurdity - 189
abundant - 227
abundantly - 67, 85, 125
abuse - 6, 166
abused - 179, 197
abusing - 31, 299
abusive - 184, 316
accede - 166
acceded - 207
acceding - 249
```