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

З лабораторної роботи №1

Виконав студент *ІП-01 Смислов Даніл* _____
(шифр, прізвище, ім'я, по батькові)

Перевірив *ас. Очеретяний О.К.* _____
(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.Ініціалізуємо та заповнюємо масив стоп-слів

2.Зчитуємо з файлу слово, якщо воно порожнє(кінець файлу) то переходимо до пункту 10.

3.Додаємо до кінця слова символ ' ', щоб розуміти де знаходиться кінець слова.

4.Якщо в слові зустрічаються букви верхнього регістру - приводимо їх до нижнього, використовуючи ASCII коди.

5.Видаляємо зі слова усі зайві знаки, крім букв нижнього регістру. А також дефісу, апострофів, якщо після них йдуть ще букви та вони не на початку слова. Якщо зі слова не вдалось дістати підходящі літери - переходимо до пункту 2.

6.Перевіряємо чи це не стоп-слово. Якщо це стоп-слово - переходимо до пункту 2.

7.Якщо це слово вже зустрічалось то інкрементуємо його кількість входжень та переходимо до пункту 2.

8.Додаємо це слово в наш список як нове, прирівнюємо кількість його входжень до 1.

9.Якщо масив слів переповниться на наступному кроці то розширимо розмір цього масиву на 5, перепишемо старі елементи в нові(збільшені) масиви, чистимо пам’ять, переходимо до пункту 2.

10.Закриваємо потік читання з файлу. Сортуємо алгоритмом бульбашки наші масиви.

11.Обираємо скільки слів будемо виводити.

12.Виводимо слова та їх кількість в консоль та у файл.

13.Закриваємо потік писання в файл, чистимо пам'ять.

Завдання 2:

1.Ініціалізуємо та заповнюємо масив стоп-слів

2.Перевіряємо чи наступний взятий символ буде символом кінця рядка. Якщо це так то інкрементуємо кількість рядків, дістаємо з потоку цей символ, переходимо до пункту 2.

3.Якщо ми перейшли на нову сторінку то інкрементуємо номер поточної сторінки. Масив, що відповідає за те, чи слово було на цій сторінці, заповнюємо значеннями false, адже це нова сторінка і на ній ми ще не читали жодних слів.

4.Зчитуємо з файлу слово, якщо воно порожнє(кінець файлу) то переходимо до пункту 13.

5.Додаємо до кінця слова символ ' ', щоб розуміти де знаходиться кінець слова.

6.Якщо в слові зустрічаються букви верхнього регістру - приводимо їх до нижнього, використовуючи ASCII коди.

7.Видаляємо зі слова усі зайві знаки, крім букв нижнього регістру. А також дефісу, апострофів, якщо після них йдуть ще букви та вони не на початку слова. Якщо зі слова не вдалось дістати підходящі літери - переходимо до пункту 2.

8.Перевіряємо чи це не стоп-слово. Якщо це стоп-слово - переходимо до пункту 2.

9.Якщо це слово вже зустрічалось то інкрементуємо кількість його входжень. Якщо це слово ще не зустрічалось на даній сторінці то переводимо цифри номера його сторінки в символи, за допомогою ASCII кодів та записуємо номер поточної сторінки в рядок,

в якому записані номери сторінок де це слово зустрічається.

10.Ставимо true в масиві, що відповідає за те, що це слово вже зустрічалось на поточній сторінці, щоб ми не записали цьому слову кілька однакових номерів сторінки, переходимо до пункту 2.

11.Ставимо true в масиві, що відповідає за те, що це слово вже зустрічалось на поточній сторінці, додаємо нове слово в масив слів, переводимо номер сторінки в строку. Кількість зустрічання цього слова прирівнюємо до 1. Інкрементуємо загальну кількість різних слів.

12.Якщо масив слів переповнився, то розширимо розмір цього масиву на 5, перепишемо старі елементи в нові(збільшені) масиви, чистимо пам'ять, переходимо до пункту 2.

13.Закриваємо потік читання з файлу. Сортуємо алгоритмом бульбашки наші масиви.

14.Виводимо слова, що зустрічались меньше 101 разу, та номери сторінок, на яких вони зустрічаються, в консоль та файл.

15.Чистимо пам'ять та закриваємо потік писання в файл.

3. Програмний код

Завдання 1:

```
#include <iostream>
#include <fstream>

using namespace std;
int main()
{
    int expectedWordsToShow;
    cout << "How many words you want to show?\n";
    cin >> expectedWordsToShow;

    int stopWordsSize = 18;
    string* stopWords = new string[stopWordsSize]; //list of stop words
    stopWords[0] = "in";
    stopWords[1] = "on";
    stopWords[2] = "up";
    stopWords[3] = "an";
    stopWords[4] = "a";
    stopWords[5] = "are";
    stopWords[6] = "is";
    stopWords[7] = "has";
    stopWords[8] = "so";
    stopWords[9] = "be";
    stopWords[10] = "were";
    stopWords[11] = "do";
    stopWords[12] = "did";
    stopWords[13] = "can";
    stopWords[14] = "for";
    stopWords[15] = "to";
    stopWords[16] = "and";
    stopWords[17] = "the";

    ifstream inFile("input.txt");
    int maxNumberWords = 10;
    int currentNumberWords = 0;
    string* allWords = new string[maxNumberWords]; //list of all words
    int* allNumbers = new int[maxNumberWords]; //list of all repetitions of words
```

```
    int numberOfWordsToDisplay = 25;

    string temp = "";
wordFromFile:
    inFile >> temp;
    if (temp == "") { //if it is end of file
        goto wordsReadingEnd;
    }
    else {
        temp += " ";
        int i = 0;
    normalization:
        if (temp[i] >= 65 && temp[i] <= 90)temp[i] += 32; //make symbol lower if it is
upper case
        i++;
        if (temp[i] != ' ') goto normalization;
        i = 0;
        string buffer = "";
        bool afterChar = false;
    deleteSigns:
        if (temp[i] >= 97 && temp[i] <= 122 || (temp[i] == '-' || temp[i] == '\'' ||
temp[i] == '`') && (temp[i + 1] >= 97 && temp[i + 1] <= 122) && afterChar) {
            buffer += temp[i];
            afterChar = true;
        }
        i++;
        if (temp[i] != ' ') goto deleteSigns;
        if (buffer == "") {
            temp = "";
            goto wordFromFile;
        }
        temp = buffer + " ";
        i = 0;
    checkStop: //check if it is stop word
        if (i < stopWordsSize) {
            if ((stopWords[i]+' ') == temp) {
                temp = "";
                goto wordFromFile;
            }
            i++;
            goto checkStop;
        }
        i = 0;
    countDuplication:
        if (i < currentNumberWords) {
            if (allWords[i] == temp) {
                allNumbers[i] += 1;
                temp = "";
                goto wordFromFile;
            }
            i++;
            goto countDuplication;
        }
        allWords[currentNumberWords] = temp;
        allNumbers[currentNumberWords] = 1;
        currentNumberWords++;
        temp = "";
        if (currentNumberWords + 1 == maxNumberWords) { //check if massive is overflow
            maxNumberWords += 5; //expand to 5 extra words
            //initialize new arrays
            string* tempWords = new string[maxNumberWords];
```

```
            int* tempNumbers = new int[maxNumberWords];
            i = 0;
        newArrays:
            if (i < currentNumberWords) {
                tempWords[i] = allWords[i];
                tempNumbers[i] = allNumbers[i];
                i++;
                goto newArrays;
            }
            delete[] allWords;
            delete[] allNumbers;
            allWords = tempWords;
            allNumbers = tempNumbers;
        }
        goto wordFromFile;
    }
wordsReadingEnd:
    inFile.close();
    int i = 0;
    externalCycle: //bubble sort
    if (i < currentNumberWords - 1) {
        int j = 0;
    internalCycle:
        if (j < currentNumberWords - i - 1) {
            if (allNumbers[j] < allNumbers[j + 1]) {
                int tempNumber = allNumbers[j];
                allNumbers[j] = allNumbers[j + 1];
                allNumbers[j + 1] = tempNumber;
                string tempWord = allWords[j];
                allWords[j] = allWords[j + 1];
                allWords[j + 1] = tempWord;
            }
            j++;
            goto internalCycle;
        }
        i++;
        goto externalCycle;
    }
    int wordsToShow;
    if (expectedWordsToShow > currentNumberWords) { //choose how many words we have to
output
        wordsToShow = currentNumberWords;
    }
    else {
        wordsToShow = expectedWordsToShow;
    }
    i = 0;
    ofstream outFile("output.txt");
output:
    if (i < wordsToShow) {
        cout << allWords[i] << " - " << allNumbers[i] << endl;
        outFile << allWords[i] << " - " << allNumbers[i] << endl;
        i++;
        goto output;
    }
    outFile.close();
    delete[] allWords;
    delete[] allNumbers;
      }
```

Завдання 2:

```cpp
#include <iostream>
#include <fstream>

using namespace std;
int main()
{
	string fileName;
	cout << "Input the file name to index\n";
	cin >> fileName;
    int stopWordsSize = 18;
    string* stopWords = new string[stopWordsSize]; //list of stop words
    stopWords[0] = "in";
    stopWords[1] = "on";
    stopWords[2] = "up";
    stopWords[3] = "an";
    stopWords[4] = "a";
    stopWords[5] = "are";
    stopWords[6] = "is";
    stopWords[7] = "has";
    stopWords[8] = "so";
    stopWords[9] = "be";
    stopWords[10] = "were";
    stopWords[11] = "do";
    stopWords[12] = "did";
    stopWords[13] = "can";
    stopWords[14] = "for";
    stopWords[15] = "to";
    stopWords[16] = "and";
    stopWords[17] = "the";

    ifstream inFile(fileName);
    int maxNumberWords = 10;
    int currentNumberWords = 0;
    string* allWords = new string[maxNumberWords]; //list of all words
    string* allPages = new string[maxNumberWords]; //list with pages where the word is
    int* numberOfPages = new int[maxNumberWords];  //list with numbers of repetitions
of words
    bool* wasOnThisPage = new bool[maxNumberWords]; //bool list if the word was on
current page
    int lineNumber = 0;
    int pageNumber = 0;

wordFromFile:
    if (inFile.peek() == '\n') { //check if it is end of line
        lineNumber++;
        inFile.get();
        goto wordFromFile;
    }
    if (lineNumber / 45 != pageNumber) {  //if the new page
        pageNumber++;
        int i = 0;
    newPage:
        wasOnThisPage[i] = false;
        i++;
        if (i < maxNumberWords)goto newPage;
    }
    string temp = "";
    inFile >> temp;
    if (temp == "") {
        goto wordsReadingEnd;
    }
```

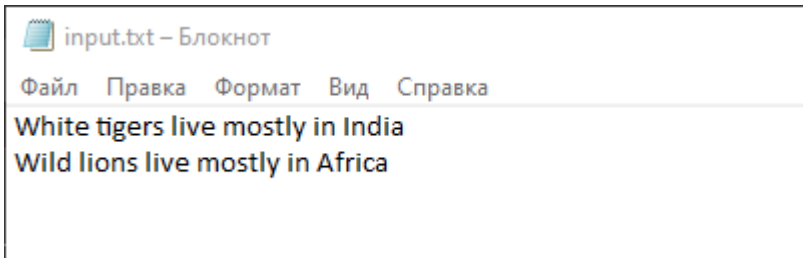
```
    else {
        temp += " ";
        int i = 0;
    normalization:
        if (temp[i] >= 65 && temp[i] <= 90)temp[i] += 32; //if it is an upper case -
make it lower
        i++;
        if (temp[i] != ' ') goto normalization;
        i = 0;
        string buffer = "";
        bool afterChar = false;
    deleteSigns: //delete the unnecessary signs from words
        if (temp[i] >= 97 && temp[i] <= 122 || (temp[i] == '-' || temp[i] == '\'' ||
temp[i] == '`') && (temp[i + 1] >= 97 && temp[i + 1] <= 122) && afterChar) {
            buffer += temp[i];
            afterChar = true;
        }
        i++;
        if (temp[i] != ' ') goto deleteSigns;
        if (buffer == "") {
            temp = "";
            goto wordFromFile;
        }
        temp = buffer + " ";
        i = 0;
    checkStop: //check if it is a stop word
        if (i < stopWordsSize) {
            if ((stopWords[i] + ' ') == temp) {
                temp = "";
                goto wordFromFile;
            }
            i++;
            goto checkStop;
        }
        i = 0;
    countDuplication:
        string tempStr = "";
        if (temp == allWords[i]) {
            numberOfPages[i]++;
            if (!wasOnThisPage[i]) {
                int tempPageNumber = pageNumber + 1; //logical page number
            toString:
                if (tempPageNumber != 0) {
                    int number = tempPageNumber % 10;
                    tempStr = char(number + 48) + tempStr; //ASCII code of number
                    tempPageNumber /= 10;
                    goto toString;
                }
                allPages[i] += ", " + tempStr;
            }
            wasOnThisPage[i] = true;
            temp = "";
            goto wordFromFile;
        }
        i++;
        if (i < currentNumberWords)goto countDuplication;
        tempStr = "";
        wasOnThisPage[currentNumberWords] = true;
        allWords[currentNumberWords] = temp;
        numberOfPages[currentNumberWords] = 1;
        int tempPageNumber = pageNumber + 1;
```

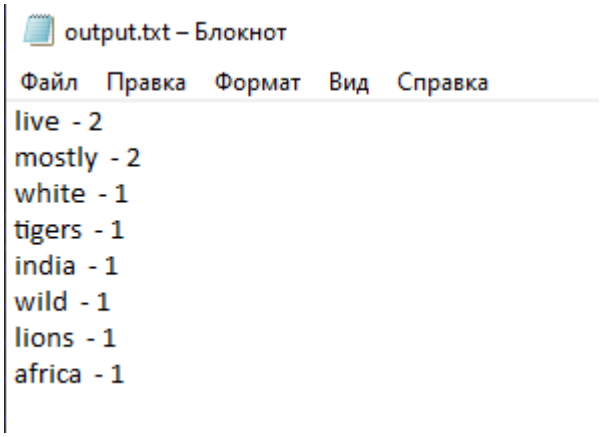
```
    toString2:
        if (tempPageNumber != 0) {
            int number = tempPageNumber % 10;
            tempStr = char(number + 48) + tempStr; //ASCII code of number
            tempPageNumber /= 10;
            goto toString2;
        }
        allPages[currentNumberWords] += tempStr;
        currentNumberWords++ ;
        if (currentNumberWords == maxNumberWords) { //if the massive is overflow
            maxNumberWords += 5; //expand to 5 extra words
            //initialize new arrays
            string* tempWords = new string[maxNumberWords];
            string* tempPages = new string[maxNumberWords];
            int* tempNumbers = new int[maxNumberWords];
            bool* tempWas = new bool[maxNumberWords];
            int i = 0;
        copyToNewArrays:
            if (i < currentNumberWords) {
                tempWords[i] = allWords[i];
                tempPages[i] = allPages[i];
                tempNumbers[i] = numberOfPages[i];
                tempWas[i] = wasOnThisPage[i];
                i++;
                goto copyToNewArrays;
            }
            delete[] allWords;
            delete[] allPages;
            delete[] numberOfPages;
            delete[] wasOnThisPage;
            allWords = tempWords;
            allPages = tempPages;
            numberOfPages = tempNumbers;
            wasOnThisPage = tempWas;
        }
        temp = "";
        goto wordFromFile;
    }
wordsReadingEnd:
    inFile.close();
    ofstream outFile("output.txt");
    int i = 0;
externalCycle:  //bubble sort
    if (i < currentNumberWords) {
        int j = i + 1;
    internalCycle:
        if (j < currentNumberWords) {
            bool swap = 0;
            int k = 0;
        checkSwap:
            if (allWords[i][k] != ' ' && allWords[j][k] != ' ') {
                if (allWords[i][k] == allWords[j][k]) {
                    k++;
                    goto checkSwap;
                }
                else {
                    if (allWords[i][k] > allWords[j][k]) swap = 1;
                }
            }
            else {
                if (allWords[i][k] != ' ') swap = 1;
```

```
            }
            if (swap) {
                string tempWord = allWords[i];
                allWords[i] = allWords[j];
                allWords[j] = tempWord;
                string tempPages = allPages[i];
                allPages[i] = allPages[j];
                allPages[j] = tempPages;
                int tempNumber = numberOfPages[i];
                numberOfPages[i] = numberOfPages[j];
                numberOfPages[j] = tempNumber;
            }
            j++;
            goto internalCycle;
        }
        i++;
        goto externalCycle;
    }
    i = 0;
output:
    if (numberOfPages[i] <= 100) {
        cout << allWords[i] << " - " << allPages[i] << endl;
        outFile << allWords[i] << " - " << allPages[i] << endl;
    }
    i++;
    if (i < currentNumberWords) goto output;
    delete[] allWords;
    delete[] allPages;
    delete[] numberOfPages;
    delete[] wasOnThisPage;
    outFile.close();
      }
```

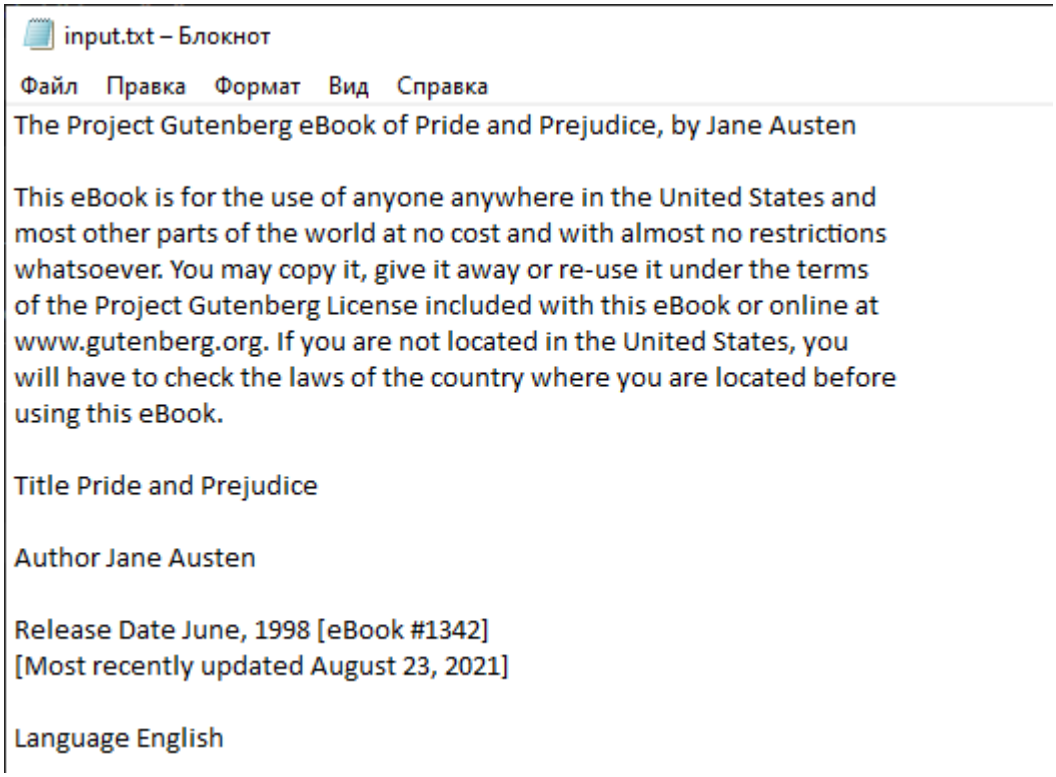
4. Приклади виконання програм

Завдання 1:




output.txt – Блокнот
Файл Правка Формат Вид Справка
live - 2
mostly - 2
white - 1
tigers - 1
india - 1
wild - 1
lions - 1
africa - 1

Завдання 2:


input.txt – Блокнот
Файл Правка Формат Вид Справка
The Project Gutenberg eBook of Pride and Prejudice, by Jane Austen
This eBook is for the use of anyone anywhere in the United States and
most other parts of the world at no cost and with almost no restrictions
whatsoever. You may copy it, give it away or re-use it under the terms
of the Project Gutenberg License included with this eBook or online at
www.gutenberg.org. If you are not located in the United States, you
will have to check the laws of the country where you are located before
using this eBook.
Title Pride and Prejudice
Author Jane Austen
Release Date June, 1998 [eBook #1342]
[Most recently updated August 23, 2021]
Language English

```
Input the file name to index
input.txt
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, 238, 243, 246, 252, 253, 259, 260, 263, 264, 268, 269, 283,
287, 297, 298, 308, 316
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
above  - 11, 32, 153, 179, 195, 202, 209, 212, 214, 218, 220, 232, 237, 256, 257, 262, 278, 284
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
```

output.txt – 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,
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
above - 11, 32, 153, 179, 195, 202, 209, 212, 214, 218, 220, 232, 237, 256, 257, 262, 278, 284
abroad - 194, 196, 233, 288
abrupt - 203
abruptly - 41, 155