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

Кафедра інформатики та програмної інженерії

Мультипарадигменне програмування

ЗВІТ

до лабораторної роботи №1

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

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

Київ 2021

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:

Крок 1. Відкрити файл для зчитування. Перейти на крок 2.

Крок 2. Зчитати слово, якщо файл повністю зчитано перейти на крок 6 інакше крок 3.

Крок 3. Приведення зчитаного слова до нижнього регістру і відкидання неслівних символів. Перейти на крок 4.

Крок 4. Перевірка чи зчитане слово не є стоп-словом. Якщо це стоп слово перейти на крок 2 інакше крок 5.

Крок 5. Якщо зчитане слово уже є в масиві слів, збільшити на 1 відповідний елемент в масиві кількостей слів. Якщо слова нема в масиві записати і встановити кількість цих слів на 1. Перейти на крок 2.

Крок 6. Закрити файл. Перейти на крок 7.

Крок 7. Сортуємо масиви в порядку зменшення за значеннями кількість слів. Перейти на крок 8.

Крок 8. Відкриваєм файл. Записуємо масиви. Закриваєм файл. Перейти на крок 9.

Крок 9. Кінець алгоритму.

Завдання 2:

Крок 1. Відкрити файл для зчитування. Перейти на крок 2.

Крок 2. Визначення номера поточного рядка. Перейти на крок 3.

Крок 3. Зчитати слово, якщо файл повністю зчитано перейти на крок 7 інакше крок 4.

Крок 4. Приведення зчитаного слова до нижнього регістру і відкидання неслівних символів. Перейти на крок 5.

Крок 5. Перевірка чи зчитане слово не є стоп-словом. Якщо це стоп слово перейти на крок 2 інакше крок 6.

Крок 6. Якщо зчитане слово уже є в масиві збільшити відповідний елемент на 1 в масиві кількостей слів і якщо кількість менше рівна 101 то вписати номер поточної сторінки в матрицю сторінок. Якщо слова нема в масиві записати і встановити кількість цих слів на 1 і записати номер поточної сторінки в відповідний масив матриці сторінок. Перейти на крок 2.

Крок 7. Закрити файл. Перейти на крок 8.

Крок 8. Видаляєм з масивів елементи і з матриці масиви, відповідні словам кількість яких в тексті більше 100. Перейти на крок 9.

Крок 9. Сортуємо елементи масивів (слів і кількостей) і масиви матриці (сторінок) у алфавітному порядку за словами елементами масива слів. Перейти на крок 10.

Крок 10. Відкриваєм файл. Записуємо масиви і матрицю. Закриваєм файл. Перейти на крок 11.

Крок 11. Кінець алгоритму.

3. Опис програмного коду

Завдання 1:

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

using namespace std;
int main()
{
    string str;
    bool is_stopword = false;
    int l, k, j;
    int N = 25;

    int len_array = 10;
    string stop_words[10];
    stop_words[0] = "the ";
    stop_words[1] = "in ";
    stop_words[2] = "a ";
    stop_words[3] = "for ";
    stop_words[4] = "to ";
    stop_words[5] = "on ";
    stop_words[6] = "at ";
    stop_words[7] = "with ";
    stop_words[8] = "about ";
    stop_words[9] = "before ";

    int len_words = 0;
    int real_len_words = 20000;
    int* amount_words = new int[real_len_words];
    string* words = new string[real_len_words];



    ifstream fin("input.txt");

for1:
    if (fin >> str)
    {
        str = str + " ";
        string buf_str = "";

        //нормалізація слів і відрізання не слівних символів
        j = 0;
    for69:
        if (str[j] != ' ')
```

```
        {
            if ((str[j] >= 'A' && str[j] <= 'Z') || (str[j] >= 'a' && str[j] <= 'z')
|| (str[j] == '-'))
            {
                if ((str[j] >= 'A' && str[j] <= 'Z'))
                {
                    str[j] = str[j] + 32;
                }
                buf_str += str[j];
            }
            j++;
            goto for69;
        }
        if (buf_str[0] == ' ' || buf_str == "" || buf_str == "-")
            goto for1;
        else
            str = buf_str + " ";

        //визначення чи це стоп-слово
        is_stopword = false;
        int j = 0;
    for_chech_stopword:
        if (j < len_array)
        {
            char ch = stop_words[j][0];
            ch = ch - 32;
            string stop_word = stop_words[j];
            stop_word[0] = ch;

            if (str == stop_words[j] || str == stop_word)
            {
                is_stopword = true;
                //goto backfor;
            }

            j++;
            goto for_chech_stopword;
        }

        //слово потрібно опрацьовувати (не стоп слово)
        if (!is_stopword)
        {
            //збільшення масива
            if (len_words == real_len_words)
            {
                int new_len = real_len_words * 2;
                int* new_amount_words = new int[new_len];
                string* new_words = new string[new_len];

                l = 0;
            rewrite:
                if (l < len_words)
                {
                    new_amount_words[l] = amount_words[l];
                    new_words[l] = words[l];
                    l++;
                    goto rewrite;
                }

                real_len_words = new_len;

                delete[] amount_words;
                amount_words = new_amount_words;
                new_amount_words = nullptr;
```

```
                delete[] words;
                words = new_words;
                new_words = nullptr;
            }


            //обрахунок слів
            k = 0;
        for2:

            if (k == len_words)     //нове слово
            {
                amount_words[k] = 1;
                words[k] = str;
                len_words++;
            }
            else
                if (str == words[k])       //слово вже є + кількість
                {
                    amount_words[k]++;
                }
                else                        // рухаємся далі по масиву слово не
співпало і слова в масиві не закінчилися
                {
                    k++;
                    goto for2;
                }
        }

        str = "";
        goto for1;
    }
    fin.close();


    //сортування
    k = 0;
sort_for1:
    if (k < len_words)
    {
        l = k + 1;
    sort_for2:
        if (l < len_words)
        {
            if (amount_words[k] < amount_words[l])
            {
                str = words[k];
                words[k] = words[l];
                words[l] = str;

                int buf = amount_words[k];
                amount_words[k] = amount_words[l];
                amount_words[l] = buf;
            }
            l++;
            goto sort_for2;
        }
        k++;
        goto sort_for1;
    }

    //запис в файл
    ofstream fout;
    fout.open("output.txt");
    l = 0;
```

```
    for3:
        if (l < len_words && l <= N)
        {
            fout << words[l] << "- " << amount_words[l] << "\n";
            l++;
            goto for3;
        }
        fout.close();

        return 0;
    }
```

Завдання 2:

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

using namespace std;
int main()
{
    string str;
    bool is_stopword = false;
    int l, k, j;

    int len_array = 10;
    string stop_words[10];
    stop_words[0] = "the ";
    stop_words[1] = "in ";
    stop_words[2] = "a ";
    stop_words[3] = "for ";
    stop_words[4] = "to ";
    stop_words[5] = "on ";
    stop_words[6] = "at ";
    stop_words[7] = "with ";
    stop_words[8] = "about ";
    stop_words[9] = "before ";

    int len_words = 0;
    int real_len_words = 20000;
    int* amount_words = new int[real_len_words];
    string* words = new string[real_len_words];


    int** pages_words = new int* [real_len_words];
    l = 0;
    for6:
    if (l < real_len_words)
    {
        pages_words[l] = new int[101];
        l++;
        goto for6;
    }

    int current_string = 0;


    ifstream fin("input.txt");

for1:

        //перевірка номера рядка
check_cur_str:
    if (fin.peek() == '\n')
```

```
    {
        current_string++;
        fin.get();
        goto check_cur_str;
    }

        //зчитати слово
    if (fin >> str)
    {

        str = str + " ";
        string buf_str = "";

                //нормалізація слів і відрізання не слівних символів
        j = 0;
    for69:
        if (str[j] != ' ')
        {
            if ((str[j] >= 'A' && str[j] <= 'Z') || (str[j] >= 'a' && str[j] <= 'z') ||
(str[j] == '-'))
            {
                if ((str[j] >= 'A' && str[j] <= 'Z'))
                {
                    str[j] = str[j] + 32;
                }
                buf_str += str[j];
            }
            j++;
            goto for69;
        }
        if (buf_str[0] == ' ' || buf_str == "" || buf_str == "-")
            goto for1;
        else
            str = buf_str + " ";

        //визначення чи це стоп-слово
        is_stopword = false;
        int j = 0;
    for_chech_stopword:
        if (j < len_array)
        {
            char ch = stop_words[j][0];
            ch = ch - 32;
            string stop_word = stop_words[j];
            stop_word[0] = ch;

            if (str == stop_words[j] || str == stop_word)
            {
                is_stopword = true;
                //goto backfor;
            }

            j++;
            goto for_chech_stopword;
        }

        //слово потрібно опрацьовувати (не стоп слово)
        if (!is_stopword)
        {


            //збільшення масивів
            if (len_words == real_len_words)
            {
                int new_len = real_len_words * 2;
```

```
            int* new_amount_words = new int[new_len];
            string* new_words = new string[new_len];

            int** new_pages_words = new int* [new_len];
            l = 0;
        for7:
            if (l < len_words)
            {
                new_pages_words[l] = pages_words[l];
                l++;
                goto for7;
            }
            else
                if (l >= len_words && l < new_len)
                {
                    new_pages_words[l] = new int[101];
                    l++;
                    goto for7;
                }

            l = 0;
            rewrite:
            if (l < len_words)
            {
                new_amount_words[l] = amount_words[l];
                new_words[l] = words[l];
                l++;
                goto rewrite;
            }

            real_len_words = new_len;

            delete[] amount_words;
            amount_words = new_amount_words;
            new_amount_words = nullptr;

            delete[] words;
            words = new_words;
            new_words = nullptr;

            delete[] pages_words;
            pages_words = new_pages_words;
            new_pages_words = nullptr;

        }



        //обрахунок слів
        k = 0;
    for2:
        if (k == len_words)     //нове слово
        {
            amount_words[k] = 1;
            words[k] = str;
            pages_words[k][0] = (current_string / 45) + 1;
            len_words++;
        }
        else
            if (str == words[k] && amount_words[k] < 101)       //слово вже є +
кількість (таких слів було знайдено менше 101)
            {
                pages_words[k][amount_words[k]] = (current_string / 45) + 1;
                amount_words[k]++;
            }
```

```
                else                        // рухаємся далі по масиву слово не співпало
і слова в масиві не закінчилися
                {
                    k++;
                    goto for2;
                }
        }

        str = "";
        goto for1;
    }
    fin.close();


            //Чистка слів більше 100
    int n_len = len_words;
    int* n_amount_words = new int[n_len];
    string* n_words = new string[n_len];
    int** n_pages_words = new int* [n_len];

    l = 0;
    j = 0;
    clean:
    if (l < len_words)
    {
        if (amount_words[l] <= 100)
        {
            n_words[j] = words[l];
            n_amount_words[j] = amount_words[l];
            n_pages_words[j] = pages_words[l];
            j++;
        }
        else
        {
            delete[] pages_words[l];
            pages_words[l] = nullptr;
        }
        l++;
        goto clean;
    }
    len_words = j;

    delete[] amount_words;
    amount_words = n_amount_words;
    n_amount_words = nullptr;

    delete[] words;
    words = n_words;
    n_words = nullptr;

    delete[] pages_words;
    pages_words = n_pages_words;
    n_pages_words = nullptr;


    //сортування
    k = 0;
sort_for1:              //вибір першого слова (зовн цикл)
    if (k < len_words)
    {
        l = k + 1;
    sort_for2:                //вибір другого слова (внутр цикл)
        if (l < len_words)
        {
                    // перевірка чи два слова потрібно поміняти місцями за алфавітом
```

```
            bool is_swap = false;
            j = 0;
            sort_words:
            if (words[k][j] != ' ' || words[l][j] != ' ')
            {
                if (words[k][j] == words[l][j])
                {
                    j++;
                    goto sort_words;
                }
                else
                {
                    if (words[k][j] > words[l][j])
                    {
                        is_swap = true;
                    }
                }
            }
            else            //одне з слів закінчилося
            {
                if(words[l][j] != ' ')          //друге слово коротше і співпадає з
частиною першого
                    is_swap = true;
            }

                //зміна місцями слів
            if (is_swap)
            {
                str = words[k];
                words[k] = words[l];
                words[l] = str;

                int buf = amount_words[k];
                amount_words[k] = amount_words[l];
                amount_words[l] = buf;

                int* b = pages_words[k];
                pages_words[k] = pages_words[l];
                pages_words[l] = b;
            }


            l++;
            goto sort_for2;
        }
        k++;
        goto sort_for1;
    }

        //запис в файл
    ofstream fout;
    fout.open("output.txt");
    l = 0;
forfout:
    if (l < len_words)
    {
        fout << words[l] << "-";
        if (amount_words[l] <= 100)
        {
            k = 0;
        forfout2:
            if (k < amount_words[l] && k == 0)
            {
                fout << " " << pages_words[l][k];
                k++;
```

```
                goto forfout2;
            }
            else
                if(k < amount_words[l] && k > 0)
                {
                    fout << ", " << pages_words[l][k];
                    k++;
                    goto forfout2;
                }
            fout << "\n";
        }
        l++;
        goto forfout;
    }
    fout.close();

    return 0;
}
```

4. Скріншоти роботи програмного застосунку