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

Кафедра інформатики та програмної інженерії

ЗВІТ

з лабораторної роботи № 1 з дисципліни
«Мультипарадигменне програмування»

„Імперативне програмування”

Виконав студент Ткачук Святослав Васильович _____
(№ 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.

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.

2. Опис алгоритмів

Завдання 1

1. Прочитати увесь файл в 1 змінну.
2. Поелементно перебираючи текст змінної, розбити текст на слова та зберегти в масив, замінити великі літери маленькими.
3. Поелементно перебираючи масив слів, відкинути стоп слова та заповнити два масиви: перший містить усі слова тексту без повторень, другий - скільки разів зустрічається слова.
4. Відсортувати заповнені у попередньому пункті масиви за порядком спадання частоти повторення слів.

Завдання 2

1. Прочитати увесь файл в 1 змінну.
2. Поелементно перебираючи текст змінної, розбити текст на слова і зберегти їх та номера сторінок, на яких вони зберігаються,(розмір сторінки 45 рядків) в масиви; замінити великі літери маленькими.
3. Поелементно перебираючи масив слів та заповнити два масиви: перший містить усі слова тексту без повторень, другий - скільки разів зустрічається слова.
4. Упорядкувати заповнений у попередньому пункті масив за алфавітом.
5. Перебираючи масив слів без повторень та масив, який містить текст робитий на слова, виводимо слова, частота повторень яких менше 100, та їх сторінки.

3. Опис програмного коду

Завдання 1

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

using namespace std;

int main() {
    int stop_words_lengh = 3;
    int N = 25;
    ifstream file("some_text.txt");
    string text = "";

    string s;
    read_file:
    if (!file.eof()) {
        s = "";
        getline(file, s);
        text += s + '\n';
        goto read_file;
    }

    int max_words = text.size() / 2;
    string* words = new string[max_words];
    string* unique_words = new string[max_words];
    int* words_count = new int[max_words];

    int i = 0;
    initilize_word_count:
    if (i < max_words) {
        words_count[i] = 0;
```

```
        i++;
        goto initilize_word_count;
    }

    int i1 = 0, word_i = 0;
    bool is_word = false;
    text_traversal:
    if (text[i1] != '\0') {
        if ((text[i1] >= 'a' && text[i1] <= 'z') || (text[i1] >= 'A' && text[i1]
<= 'Z') || text[i1] == '\'') {
            is_word = true;
        }
        else {
            if (is_word) {
                word_i++;
                is_word = false;
            }
        }

        if (is_word) {
            if (text[i1] >= 'A' && text[i1] <= 'Z') {
                words[word_i] += text[i1] - 'A' + 'a';
            }
            else {
                words[word_i] += text[i1];
            }
        }

        i1++;
        goto text_traversal;
    }

    int i3 = 0;
    words_traversal:
```

```
if (words[i3] != "") {
    int i4 = 0;
    if (words[i3].size() <= stop_words_lengh) {
        goto miss_word;
    }

    unique_words_traversal:
    if (unique_words[i4] != "") {
        if (words[i3] == unique_words[i4]) {
            words_count[i4]++;
            goto miss_word;
        }

        i4++;
        goto unique_words_traversal;
    }
    else {
        unique_words[i4] = words[i3];
        words_count[i4]++;
    }

    miss_word:

    i3++;
    goto words_traversal;
}

int i6 = 0;
sort:
if (i6 < N) {
    string max_unique_word;
    int max_word_count = 0;
    int max_i7 = 0;

    int i7 = 0;
```

```
        step:
        if (unique_words[i7] != "") {
            if (words_count[i7] > max_word_count) {
                max_unique_word = unique_words[i7];
                max_word_count = words_count[i7];
                max_i7 = i7;
            }
            i7++;
            goto step;
        }

        words_count[max_i7] = -words_count[max_i7];
        cout << max_unique_word << "    " << max_word_count << endl;

        i6++;
        goto sort;
    }

    delete[] words;
    delete[] unique_words;
    delete[] words_count;
    file.close();
}
```

Завдання 2

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

using namespace std;

int main()
{
    int page_size = 45;
    ifstream file("some_book.txt");
    string text = "";
```

```
string s;
read_file:
if (!file.eof()) {
    s = "";
    getline(file, s);
    text += s + '\n';
    goto read_file;
}

int max_words = text.size();
string* words = new string[max_words];
string* unique_words = new string[max_words];

int* words_pages = new int[max_words];

int i = 0;
initialize_words_pages:
if (i < max_words) {
    words_pages[i] = 0;
    i++;
    goto initialize_words_pages;
}

int* words_count = new int[max_words];

int i2 = 0;
initialize_words_count:
if (i2 < max_words) {
    words_count[i2] = 0;

    i2++;
    goto initialize_words_count;
}
```

```
    int line = 0;
    int i1 = 0, word_i = 0;
    bool is_word = false;
    text_traversal:
    if (i1 < text.size()) {
        if ((text[i1] >= 'a' && text[i1] <= 'z') || (text[i1] >= 'A' && text[i1]
<= 'Z') || text[i1] == '\'') {
            is_word = true;
        }
        else {
            if (is_word) {
                words_pages[word_i] = line / page_size + 1;
                word_i++;
                is_word = false;
            }
        }

        if (text[i1] == '\n') {
            line++;
        }

        if (is_word) {
            if (text[i1] >= 'A' && text[i1] <= 'Z') {
                words[word_i] += text[i1] - 'A' + 'a';
            }
            else {
                words[word_i] += text[i1];
            }
        }

        i1++;
        goto text_traversal;
    }
```

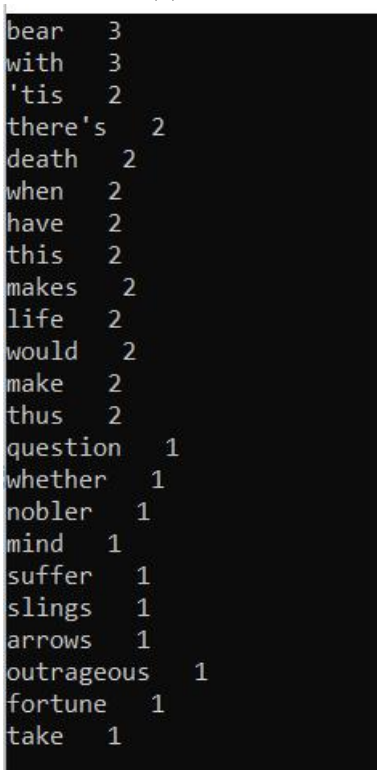
```
int i3 = 0;
words_traversal:
if (words[i3] != "") {
    int i4 = 0;

    unique_words_traversal:
    if (unique_words[i4] != "") {
        if (words[i3] == unique_words[i4]) {
            words_count[i4]++;
            goto miss_word;
        }

        i4++;
        goto unique_words_traversal;
    }
    else {
        unique_words[i4] = words[i3];
        words_count[i4]++;
    }

    miss_word:

    i3++;
    goto words_traversal;
}

bool was_replased = true;
string temp_word = "";
int temp_count = 0;
sort:
if (was_replased) {
    was_replased = false;
    int i5 = 0;
    step:
    if (words_count[i5 + 1] > 0 && i5 + 1 < max_words) {
```

```
        if (unique_words[i5] > unique_words[i5 + 1]) {
            temp_word = unique_words[i5];
            unique_words[i5] = unique_words[i5 + 1];
            unique_words[i5 + 1] = temp_word;

            temp_count = words_count[i5];
            words_count[i5] = words_count[i5 + 1];
            words_count[i5 + 1] = temp_count;
            was_replased = true;
        }

        i5++;
        goto step;
    }
    goto sort;
}

int i6 = 0;
print_words_and_pages:
if (i6 < 20) {
    if (words_count[i6] < 100) {
        cout << unique_words[i6] << " - ";
        int i7 = 0;
        print_pages:
        if (words[i7] != "") {

            if (words[i7] == unique_words[i6]) {
                cout << words_pages[i7] << ' ';
            }

            i7++;
            goto print_pages;
        }
        cout << endl;
    }
```

```
            i6++;
            goto print_words_and_pages;
        }

        file.close();
}
```

4. Скріншоти роботи програмного застосунку

Завдання 1



Завдання 2