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 «Мультипарадигменне програмування»

Виконав: Гурківська Т.В.
Перевірив : ас. Очеретяний О. К.

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.

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.

2. ОПИС ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

Для виконання даної лабораторної роботи мною було використано мову програмування C++, оскільки вона підтримує усі задані обмеження. У якості середовіща розробки - Visual Studio

3. Програмний код

```
//task1.cpp
#include <iostream>,
#include <fstream>
#include <string>

using namespace std;

int main()
{
int mpi = 0;
string mp[100][2];
//=============================================================
=
//open file
string file_name;
cout << "Enter file name: "; cin >> file_name;
file_name += ".txt";
int n;
cout << "Enter amount of words: "; cin >> n;
ifstream file;
file.open(file_name);
if (file.is_open()) {
//=============================================================
=
//read words
read_string:
string w;
```

```
getline(file, w);
// cout << w << endl;
int i = 0;
read_word:
string s;
cycle_word:
if (i < w.size()) {
char ch = w[i];
ch = tolower(ch);
s += ch;
i++;
if (w[i] != ' ' and w[i] != ' ' and w[i] != '!' and w[i] != '?' and w[i] != '.' and w[i] != ',' and w[i] != ';' and w[i] != ':' and !isdigit(w[i]) and w[i] != '(' and w[i] != '[' and w[i] != '-' and w[i] != ')' and w[i] != ']' and w[i] != '_') {
goto cycle_word;
}
else {
i++;
}
}
bool flag = true;
if (mpi != 0) {
int it1 = 0;
//cout << "+++++" << endl;
check:
//cout << it1->first << endl;
if (mp[it1][0] == s) {
flag = false;
int var = stoi(mp[it1][1]);
```

```
var++;
mp[it1][1] = to_string(var);
}
it1++;
if (it1 < mpi) {
goto check;
}
}
if (s != "for " and s != "the" and flag == true and s.size()>1) {
mp[mpi][0] = s;
mp[mpi][1] = "1";
mpi++;
}
//cout << s << endl;
if (i < w.size()) {
goto read_word;
}

//cout << s << endl;
if (!file.eof()) {
goto read_string;
}

}
else {
cout << "file not found!!!!" << endl;
}
file.close();
```

```
//==========================================================
=
//sort
int i1 = 0;
int i2, var1, var2;
sort_cycle_1:
var1 = stoi(mp[i1][1]);
i2 = 0;
sort_cycle_2:
var2 = stoi(mp[i2][1]);
if (i1 != i2 and var2 < var1) {
string temp0 = mp[i1][0];
string temp1 = mp[i1][1];
mp[i1][0] = mp[i2][0];
mp[i1][1] = mp[i2][1];
mp[i2][0] = temp0;
mp[i2][1] = temp1;
}
i2++;
if (i2 < mpi - 1) {
goto sort_cycle_2;
}
else {
i1++;
if (i1 < mpi) {
goto sort_cycle_1;
}
}
```

```
//=============================================================
=
//output
string file_name1;
cout << "Enter output file name: "; cin >> file_name1;
file_name1 += ".txt";
ofstream output_file;
output_file.open(file_name1);
int it = 0;
output:
output_file << mp[it][0] << "-" << mp[it][1] << endl;
it++;
if (it < mpi and it < n) {
goto output;
}
}
//task2.cpp
#include <iostream>,
#include <fstream>
#include <string>

using namespace std;

int main()
{
int mpi = 0;
string** mp =new string *[100000];
for (int i = 0; i < 100000; i++) {
```

```
mp[i] = new string[4];
}
int page = 1;
int pi = 0;
//============================================================
=
//open file
string file_name;
cout << "Enter file name: "; cin >> file_name;
file_name += ".txt";
ifstream file;
file.open(file_name);
if (file.is_open()) {
//============================================================
=
//read words
read_string:
string w;
getline(file, w);
// cout << w << endl;
int i = 0;
read_word:
string s;
cycle_word:
if (i < w.size()) {
if (w[i] != ' ' and w[i] != ' ' and w[i] != '!' and w[i] != '?' and w[i] != '.' and w[i]
!= ',' and w[i] != ';' and w[i] != ':' and !isdigit(w[i]) and w[i] !='(' and w[i]!= '['
and w[i] != '-' and w[i] != ')' and w[i]!=']' and w[i] != '_') {
char ch = w[i];
```

```
ch = tolower(ch);
s += ch;
}
i++;
if (w[i] != ' ' and w[i] != ' ' and w[i] != '!' and w[i] != '?' and w[i] != '.' and w[i] != ',' and w[i] != ';' and w[i] != ':' and !isdigit(w[i]) and w[i] != '(' and w[i] != '[' and w[i] != '-' and w[i] != ')' and w[i] != ']' and w[i] != '_') {
goto cycle_word;
}
else {
i++;
}
}
bool flag = true;
if (mpi != 0) {
int it1 = 0;
//cout << "+++++" << endl;
check:
//cout << it1->first << endl;
if (mp[it1][0] == s) {
flag = false;
string s1 = to_string(page);
if (s1 != mp[mpi][3]) {
mp[it1][1] = mp[it1][1] + ", " + s1;
mp[it1][3] = s1;
}
int var = stoi(mp[it1][2]);
var++;
mp[it1][2] = to_string(var);
```

```
}
it1++;
if (it1 < mpi) {
goto check;
}
}
if (s != "for " and s != "the" and flag == true and s.size()>2) {
pi++;
mp[mpi][0] = s;
mp[mpi][1] = to_string(page);
mp[mpi][2] = "1";
mp[mpi][3] = to_string(page);
mpi++;
if (pi == 200) {
page++;
pi = 0;
}
}
//cout << s << endl;
if (i < w.size()) {
goto read_word;
}

//cout << s << endl;
if (!file.eof()) {
goto read_string;
}
```

```
}
else {
cout << "file not found!!!!" << endl;
}
file.close();
//=========================================================
=
//sort
int i1 = 0;
int i2, var1, var2;
sort_cycle_1:
//var1 = stoi(mp[i1][1]);
i2 = 0;
sort_cycle_2:
var2 = stoi(mp[i2][1]);
if (i1 != i2 and mp[i2][0] > mp[i1][0]) {
string temp0 = mp[i1][0];
string temp1 = mp[i1][1];
string temp2 = mp[i1][2];
mp[i1][0] = mp[i2][0];
mp[i1][1] = mp[i2][1];
mp[i1][2] = mp[i2][2];
mp[i2][0] = temp0;
mp[i2][1] = temp1;
mp[i2][2] = temp2;
}
i2++;
if (i2 < mpi - 1) {
```

```
goto sort_cycle_2;
}
else {
i1++;
if (i1 < mpi) {
goto sort_cycle_1;
}
}
//=================================================
=
//output
string file_name1;
cout << "Enter output file name: "; cin >> file_name1;
file_name1 += ".txt";
ofstream output_file;
output_file.open(file_name1);
int it = 0;
output:
int var = stoi(mp[it][2]);
if (var<100) {
output_file << mp[it][0] << "-" << mp[it][1] << endl;
}
it++;
if (it < mpi ) {
goto output;
}
output_file.close();
}
```

4. ОПИС АЛГОРИТМІВ

Task1

1. Відкриття файлу;
2. Ініціалізуємо строковий 2-вимірний масив, де 1 - це слово, 2- кількість слів;
3. Зчитування строки:

    3.1Зчитування слова:

        3.1.1. tolower;

        3.1.2. Якщо символ - слово , то до слова добавляється символ;

        3.1.3. Якщо слово вже є в 2-вимірному масиві, то добавляємо; +1 до 2 , де 1- це саме слово

        3.1.4. Якщо ні - то добавляємо до масива це слово та 1;

        3.1.5. Якщо строка не закінчилась, то знов зчитуємо слово

    3.2. Якщо файл не закінчився, то знов зчитуємо строку;

4. Застосування алгоритму сортування бульбашкою задля розташування слів за лексикографічним порядком;
5. Виведення слів: перевірка на ліміт виведених слів;

Task2

1. Відкриття файлу;
2. Ініціалізуємо строковий 4-вимірний масив, де 1 - це слово, 2 - список сторінок, 3- кількість слів , 4-остання сторінка ;
3. Зчитування строки:

    3.1Зчитування слова:

        Зчитування слова:

        3.1.1. tolower;

        3.1.2. Якщо символ - слово , то до слова добавляється символ;

        3.1.3. Якщо слово вже є масиві, то добавляємо та ця сторінка не дорівнює останній сторінки цього слова; +1 до 3 ,номер сторінки до 4 та добавляємо до 2 де 1- це саме слово;

        3.1.4. Якщо ні - то добавляємо до масива це слово, сторінку, 1 , сторінку;

3.1.5. Якщо строка не закінчилась, то знов зчитуємо слово

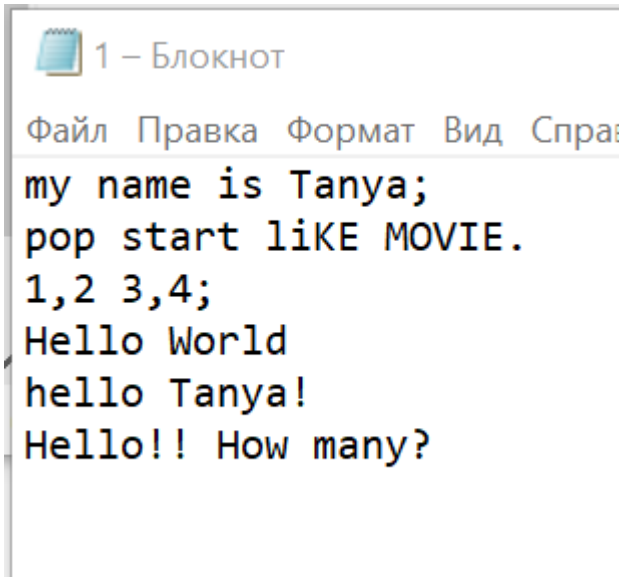
3.2. Якщо файл не закінчився, то знов зчитуємо строку;

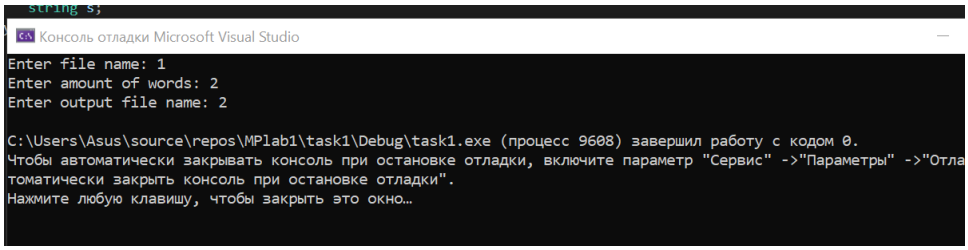
4. Застосування алгоритму сортування бульбашкою задля розташування слів за лексикографічним порядком;
5. Виведення слів;

5. СКРІНШОТИ РОБОТИ ПРОГРАМ

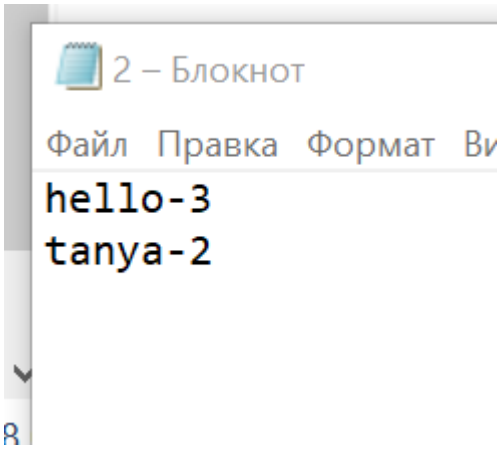
// Task1.cpp

Input:



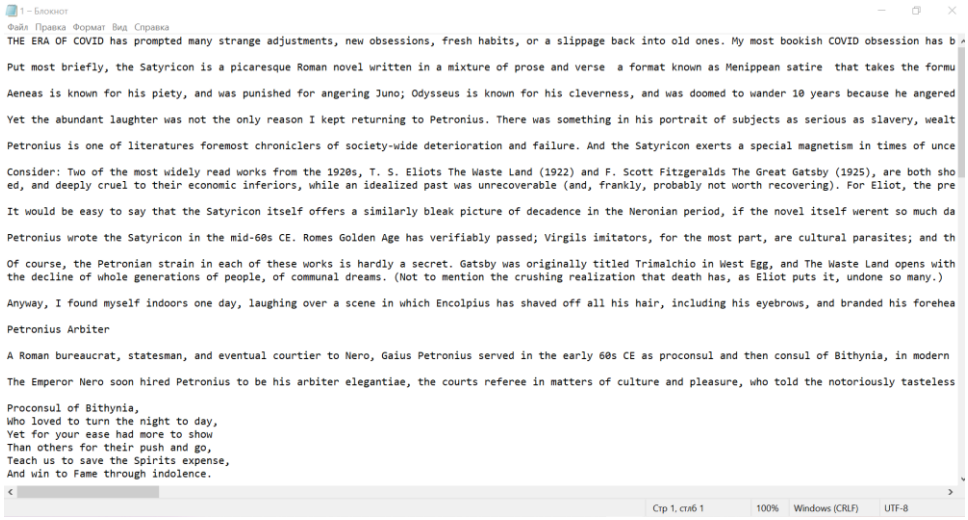


Output:



// Task2.cpp

Input:



Output:

Файл Правка Формат Вид Справка

```
abakan-10
abilities-4
ability-8, 8
able-8
abortive-2, 7
about-1, 4, 5, 6, 6, 7, 7, 7, 7, 10, 10
above-5
abundant-1, 4, 7
according-8
accordingly-2, 8
account-8
accounts-4, 5
accuracy-5
acknowledged-7
acquitted-4
across-10, 10, 10
activity-10
actually-5
adjustments-1
admirable-5
admirably-5
admiration-9
admixture-6
adrift-9
adventure-9
adventures-1
adversity-9
aeneas-1
aesthete-1, 4
aesthetic-8
afford-5
after-5, 6, 8, 8
again-8, 9
against-3, 3, 7, 8, 8, 9
```