

Звіт

По: Мультипарадигмне програмування

Лабораторна робота 1

Завдання

Практична робота складається із трьох завдань, які самі по собі є досить простими. Але, оскільки задача - зрозуміти, як писали код наші славні пращури у 1950-х, ми введемо кілька обмежень:

- Заборонено використовувати функції
- Заборонено використовувати цикли
- Для виконання потрібно взяти мову, що підтримує конструкцію GOTO

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:

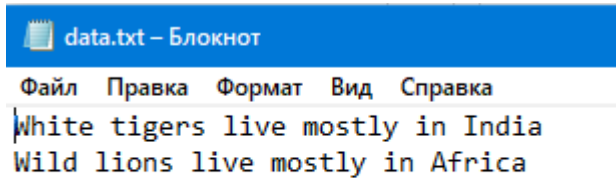
Input:

```
White tigers live mostly in India  
Wild lions live mostly in Africa
```

Output:

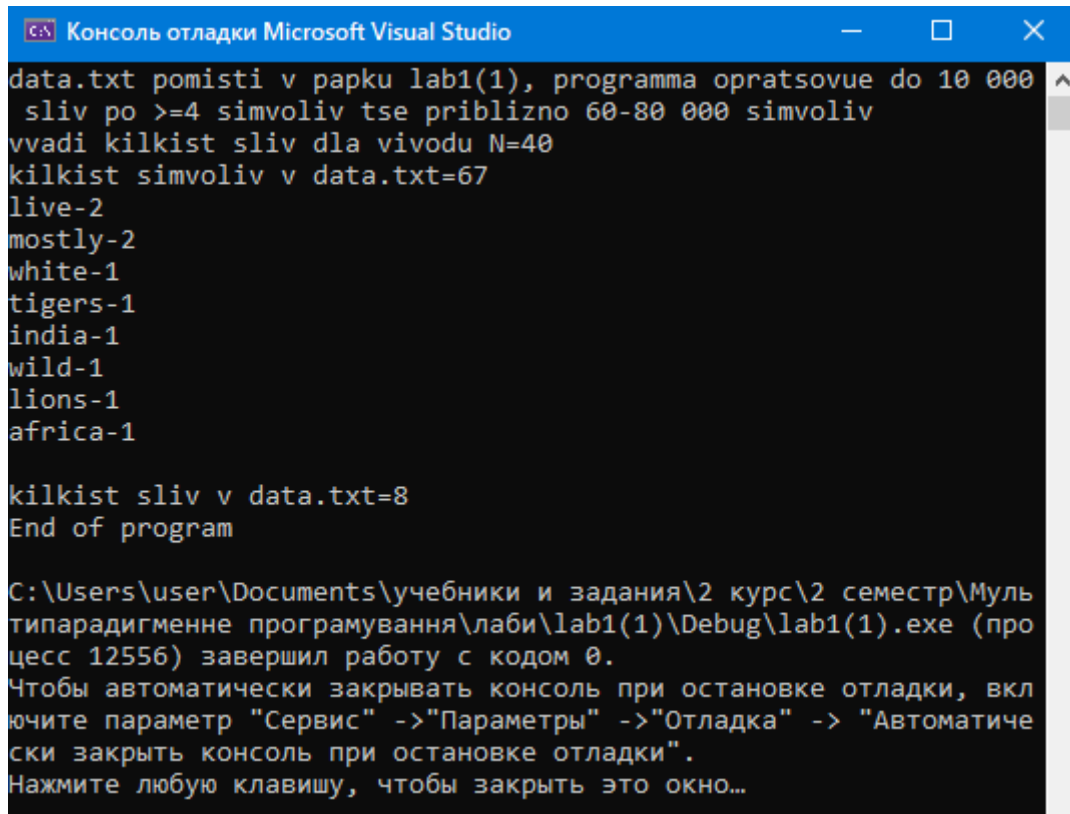
```
live - 2  
mostly - 2  
africa - 1  
india - 1  
lions - 1  
tigers - 1  
white - 1  
wild - 1
```

Вміст текстового файлу:



```
data.txt - Блокнот
Файл  Правка  Формат  Вид  Справка
White tigers live mostly in India
Wild lions live mostly in Africa
```

Скрін виконання програми на тексті в завданні:



```
Консоль отладки Microsoft Visual Studio
data.txt помісти в папку lab1(1), programma opratsovue do 10 000
sliv po >=4 simvoliv tse priblizno 60-80 000 simvoliv
vvadi kilkist sliv dla vivodu N=40
kilkist simvoliv v data.txt=67
live-2
mostly-2
white-1
tigers-1
india-1
wild-1
lions-1
africa-1

kilkist sliv v data.txt=8
End of program

C:\Users\user\Documents\учебники и задания\2 курс\2 семестр\Муль
типарадигменне програмування\лаби\lab1(1)\Debug\lab1(1).exe (про
цесс 12556) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, вкл
ючите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Автоматиче
ски закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно...
```

Код програми на C++:

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

int main()
{
    cout << "data.txt помісти в папку lab1(1), programma opratsovue do 10 000 sliv po >=4
simvoliv tse priblizno 60-80 000 simvoliv\n";
    string line;
    string text;
    string slovo="";
    struct st_slovo { // інформація про слово
        string slovo;
        int chastota = 0;
        string storinki="";
        int last_st=0; // остання сторінка на якій зустрілось слово, потрібно для
уникнення повторів сторінок
```

```

};
st_slovo mas[10000];
int k_slov=0;
st_slovo tmp;
int n_st = 1;
int n_line=1;
int min_ind=10000;
int N;

cout << "vvadi kilkist sliv dla vivodu N=";
cin >> N;
ifstream in("data.txt"); // открываем файл для чтения
if (in.is_open())
{
    lab1: /*замінив while на мітку1*/ if(getline(in, line))
    {
        // for (int j = 0; j < line.size(); j++) замінив на мітку11
        int j = 0;
        lab11: if (j<line.size())
        {
            if (line[j] >= 'A' && line[j] <= 'Z') line[j] += 'a' - 'A'; // заміна
великих букв маленькими, щоб всі слова стали однакового регістру

            j++;
            goto lab11;
        }
        text += line + "\n";

        goto lab1;
    }
}
in.close(); // закрываем файл
cout << /*text + */"kilnist simvoliv v data.txt=" << text.size() << "\n";

//for (int i = 0; i < text.size(); i++) замінив на мітку2
int i = 0;
lab2: if (i<text.size())
{
    if (text[i] != ' ' and text[i] != '\n' and (text[i]>='a' and text[i]<='z' or
text[i]>='A' and text[i]<='Z')) { // знаходимо в тексті слово, що складається з англ букв
        slovo += (string)"" + text[i];
    }
    else { // додаємо слово в масив слів або збільшуємо частоту, якщо це слово там
вже є
        if (slovo != "") {
            if (slovo.size() <= 3) slovo = ""; // ігноруємо слова довжиною менше або
рівною 3

            // for (int j = 0; j < k_slov; j++) замінив на мітку12
            int j = 0;
            lab12: if(j<k_slov)
            { // змінюємо частоту
                if (mas[j].slovo == slovo) {
                    mas[j].chastota++;
                    if (mas[j].last_st != n_st) {
                        mas[j].storinki += (string)", " + to_string(n_st);
                        mas[j].last_st = n_st;
                    }
                    slovo = "";
                }

                j++;
                goto lab12;
            }
            if (slovo != "") { // додаємо в кінець
                mas[k_slov].slovo = slovo;
            }
        }
    }
}

```

```

        mas[k_slov].chastota++;
        mas[k_slov].storinki += to_string(n_st);
        mas[k_slov].last_st = n_st;
        slovo = "";
        k_slov++;
    }
}
if (text[i] == '\n') { // рахуємо номер сторінки
    n_line++;
    n_st = (n_line-1) / 45 + 1;
}
}

i++;
goto lab2;
};

// for (int i = 0; i < k_slov; i++) замінив на мітку3
i = 0;
lab3: if(i<k_slov)
{ // сортуємо бульбашкою по частоті
    // for (int j = 0; j < k_slov-i-1; j++) замінив на мітку4
    int j = 0;
    lab4: if (j<k_slov-i-1)
    {
        if (mas[j].chastota < mas[j+1].chastota) {
            tmp=mas[j];
            mas[j] = mas[j+1];
            mas[j+1] = tmp;
        }

        j++;
        goto lab4;
    }

    i++;
    goto lab3;
}
// for (int i = 0; i < k_slov and i<N ; i++) замінив на мітку5
i = 0;
lab5: if(i<k_slov and i<N)
{ // виводимо слова і їх частоту
    cout << mas[i].slovo << "-" << mas[i].chastota << "\n";

    i++;
    goto lab5;
}
cout << "\nkilkist sliv v data.txt=" << k_slov << "\n";
cout << "End of program" << std::endl;
return 0;
}

// Запуск программы: CTRL+F5 или меню "Отладка" > "Запуск без отладки"
// Отладка программы: F5 или меню "Отладка" > "Запустить отладку"

```

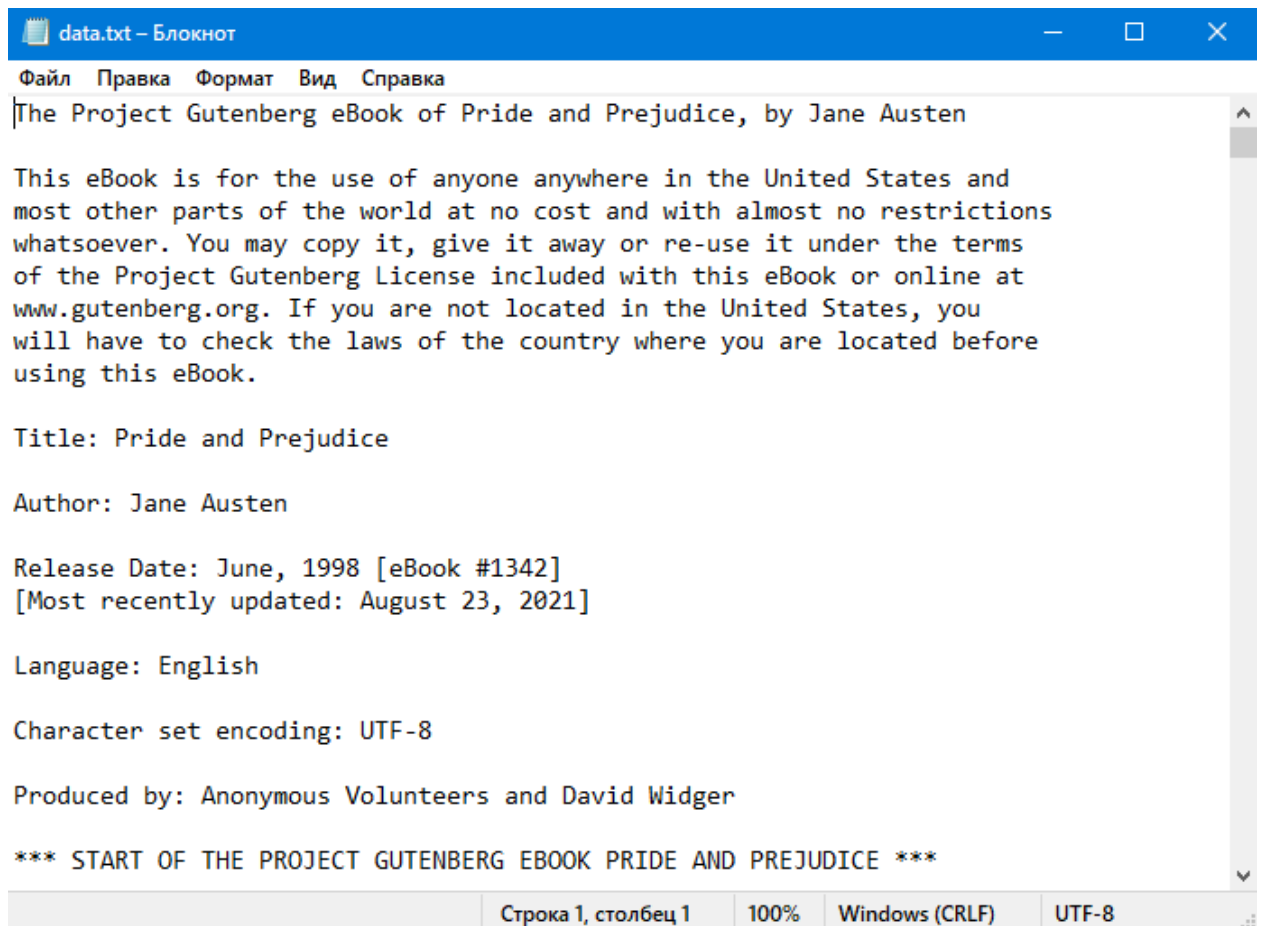
Завдання 2:

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:

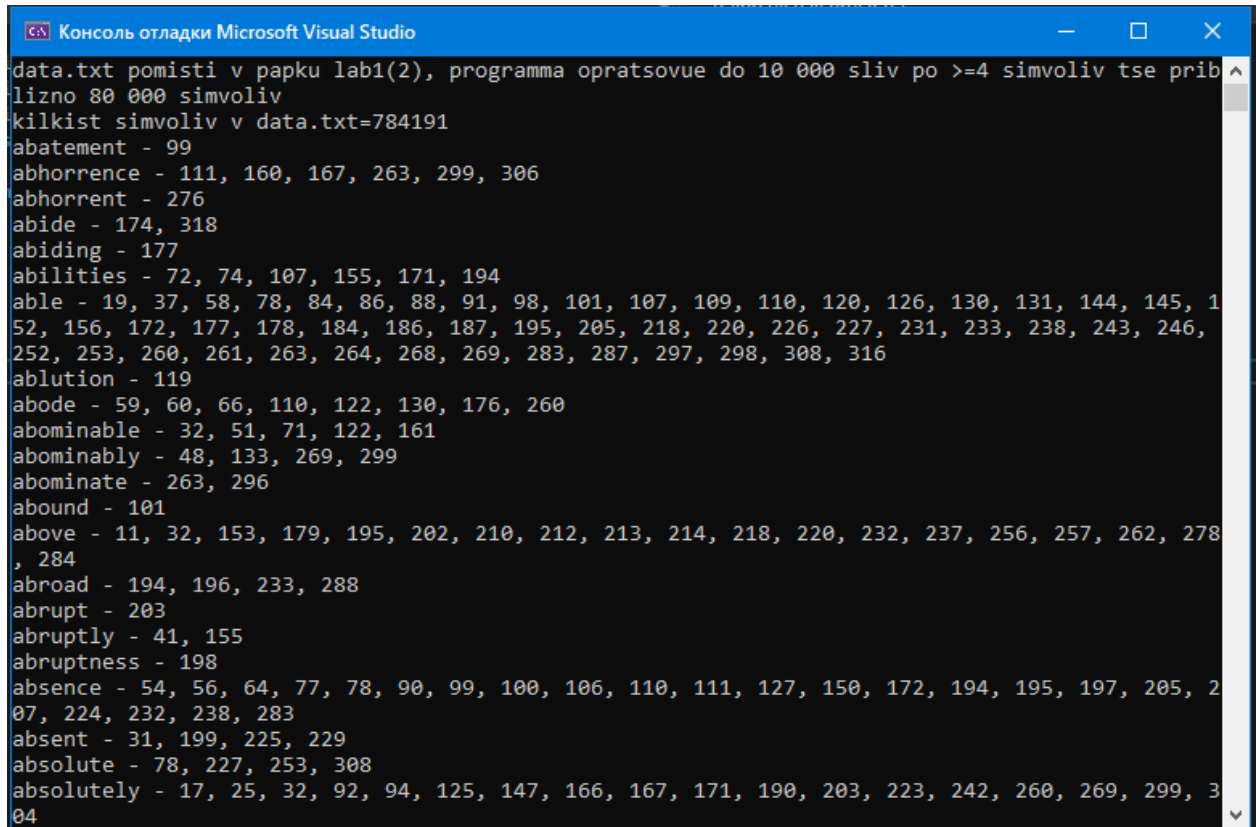
```
abatement - 89
abhorrence - 101, 145, 152, 241, 274, 281
abhorrent - 253
abide - 158, 292
```

Вміст текстового файлу взяв з *Pride and Prejudice* з інтернету:
<https://www.gutenberg.org/files/1342/1342-0.txt>

Скопіював його в data.txt.



Скрін виконання програми:



```
Консоль отладки Microsoft Visual Studio
data.txt pomisti v papku lab1(2), programma opratsovue do 10 000 sliv po >=4 simvoliv tse prib
lizno 80 000 simvoliv
kilkist simvoliv v data.txt=784191
abatement - 99
abhorrence - 111, 160, 167, 263, 299, 306
abhorrent - 276
abide - 174, 318
abiding - 177
abilities - 72, 74, 107, 155, 171, 194
able - 19, 37, 58, 78, 84, 86, 88, 91, 98, 101, 107, 109, 110, 120, 126, 130, 131, 144, 145, 1
52, 156, 172, 177, 178, 184, 186, 187, 195, 205, 218, 220, 226, 227, 231, 233, 238, 243, 246,
252, 253, 260, 261, 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, 122, 161
abominably - 48, 133, 269, 299
abominate - 263, 296
abound - 101
above - 11, 32, 153, 179, 195, 202, 210, 212, 213, 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, 106, 110, 111, 127, 150, 172, 194, 195, 197, 205, 2
07, 224, 232, 238, 283
absent - 31, 199, 225, 229
absolute - 78, 227, 253, 308
absolutely - 17, 25, 32, 92, 94, 125, 147, 166, 167, 171, 190, 203, 223, 242, 260, 269, 299, 3
04
```

Програма рахувала приблизно 5хв, мабуть бульбашка не дуже швидкий спосіб сортування.

Відстань між деякими сусідніми зустрічами слів співпадає з зразом індексу в завданні, але в цілому сторінки відрізняються, мабуть інше форматування і додаткові речі в файлі записані.

Код програми:

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

int main()
{
    cout << "data.txt pomisti v papku lab1(2), programma opratsovue do 10 000 sliv po
    >=4 simvoliv tse priblizno 80 000 simvoliv\n";
    string line;
    string text;
    string slovo = "";
    struct st_slovo { // інформація про слово
        string slovo;
        int chastota = 0;
        string storinki = "";
        int last_st = 0; // остання сторінка на якій зустрілось слово, потрібно для
        уникнення повторів сторінок
    };
    st_slovo mas[10000];
    int k_slov = 0;
    st_slovo tmp;
    int n_st = 1;
    int n_line = 1;
```

```

int min_ind = 10000;

ifstream in("data.txt"); // окриваєм файл для читання
if (in.is_open())
{
    lab1: /*замінив while на мітку1*/ if (getline(in, line))
    {
        // for (int j = 0; j < line.size(); j++) замінив на мітку11
        int j = 0;
    lab11: if (j < line.size())
    {
        if (line[j] >= 'A' && line[j] <= 'Z') line[j] += 'a' - 'A'; // заміна
        великих букв маленькими, щоб всі слова стали однакового регістру

        j++;
        goto lab11;
    }
    text += line + "\n";

    goto lab1;
}
in.close(); // закриваєм файл
cout << /*text + */"kilkist simvoliv v data.txt=" << text.size() << "\n";

//for (int i = 0; i < text.size(); i++) замінив на мітку2
int i = 0;
lab2: if (i < text.size())
{
    if (text[i] != ' ' and text[i] != '\n' and (text[i] >= 'a' and text[i] <= 'z' or
    text[i] >= 'A' and text[i] <= 'Z')) { // знаходимо в тексті слово, що складається з англ
    букв
        slovo += (string)"" + text[i];
    }
    else { // додаємо слово в масив слів або збільшуємо частоту, якщо це слово там вже
    є
        if (slovo != "") {
            if (slovo.size() <= 3) slovo = ""; // ігноруємо слова довжиною менше
            або рівною 3

            // for (int j = 0; j < k_slov; j++) замінив на мітку12
            int j = 0;
        lab12: if (j < k_slov)
        { // змінюємо частоту
            if (mas[j].slovo == slovo) {
                mas[j].chastota++;
                if (mas[j].last_st != n_st) {
                    mas[j].storinki += (string)", " + to_string(n_st);
                    mas[j].last_st = n_st;
                }
                slovo = "";
            }

            j++;
            goto lab12;
        }
        if (slovo != "") { // додаємо в кінець
            mas[k_slov].slovo = slovo;
            mas[k_slov].chastota++;
            mas[k_slov].storinki += to_string(n_st);
            mas[k_slov].last_st = n_st;
            slovo = "";
            k_slov++;
        }
    }
    if (text[i] == '\n') { // рахуємо номер сторінки

```

```

        n_line++;
        n_st = (n_line - 1) / 45 + 1;
    }
}

i++;
goto lab2;
};

// for (int i = 0; i < k_slov; i++) замінів на мітку3
i = 0;
lab3: if (i < k_slov)
{ // сортуємо бульбашкою по частоті
    // for (int j = 0; j < k_slov-i-1; j++) замінів на мітку4
    int j = 0;
lab4: if (j < k_slov - i - 1)
{
    if (mas[j].chastota < mas[j + 1].chastota) {
        tmp = mas[j];
        mas[j] = mas[j + 1];
        mas[j + 1] = tmp;
    }

    j++;
    goto lab4;
}

i++;
goto lab3;
}

// for (int i = 0; i < k_slov; i++) замінів на мітку6
i = 0;
lab6: if (i < k_slov)
{ //знаходимо індекс в відсортованому по спаданню за частотою масиві, починаючи з якого
  слова зустрічаються не більше 100 разів
    if (mas[i].chastota <= 100) {
        min_ind = i;
        // break; замінів на мітку7
        goto lab7;
    }

    i++;
    goto lab6;
} lab7:

// for (int i = min_ind; i < k_slov; i++) замінів на мітку8
i = min_ind;
lab8: if (i < k_slov)
{ // сортуємо по алфавіту слова які зустрічаються не частіше 100 разів
    // for (int j = min_ind; j < k_slov - i - 1; j++) замінів на мітку9
    int j = min_ind;
lab9: if (j < k_slov - i - 1)
{
    if (mas[j].slovo > mas[j + 1].slovo) {
        tmp = mas[j];
        mas[j] = mas[j + 1];
        mas[j + 1] = tmp;
    }

    j++;
    goto lab9;
}

i++;

```



```

goto lab8;
}

// for (int i = min_ind; i < k_slov; i++) замінив на мітку10
i = min_ind;
lab10: if (i < k_slov)
{ // виводимо відсортовані слова зі сторінками, на яких вони зустрічаються
    cout << mas[i].slovo << " - " << mas[i].storinki << "\n";

    i++;
    goto lab10;
}
cout << "\nkilkist sliv v data.txt=" << k_slov << "\n";
cout << "End of program" << std::endl;
return 0;
}

// Запуск программы: CTRL+F5 или меню "Отладка" > "Запуск без отладки"
// Отладка программы: F5 или меню "Отладка" > "Запустить отладку"

```

Висновок: в ході лабораторної роботи №1 я навчився програмувати без використання циклів, замість них використав умовний оператор if і перехід на мітку goto. Спочатку написав програму з циклами, а потім переробив на мітки, щоб не заплутатись. Програма великі файли рахує довго (другий приклад рахувала 5хв), бо використав сортування бульбашкою, має обмеження в 10к слів довших 3х символів. Код програми писав на C++.