Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені ІгоряСікорського"
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 1 з дисципліни
«Мультипарадигменне програмування»

„Проектування і аналіз алгоритмів для вирішення NP-складних задач ч.1”

Виконав(ла) *ІП-01 Адамчук Антон* _____
(шифр, прізвище, ім'я, по батькові)

Перевірив *Очеретяний О. К.* _____
(прізвище, ім'я, по батькові)

Київ 2021

1 ЗАВДАННЯ

1.1 Перше завдання:

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.

1.2 Друге завдання:

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 (1800символів).

2 ВИКОНАННЯ

2.1 Псевдокод алгоритму першого завдання

1) Початок
2) Якщо не кінець файлу, то зчитати слово
3) Кожну букву у слові перетворити в «маленьку»
4) Перевірити чи є дане слово «стоп-словом», перелік яких міститься у масиві
5) Якщо дане слово – «стоп-слово», то перейти до пункту 2
6) Перевірити чи є дане слово у масиві, де ми зберігаємо слова, які уже зустрічали та к-сть їх повторів
7) Якщо дане слово раніше уже зустрічалось, то додати одиницю до к-сті його повторів у тексті
8) Якщо дане слово раніше не зустрічалось, то додати дане слово до масиву, вказавши к-сть повторів рівним одиниці
9) Посортувати масив слів за спаданням к-сті повторів у тексті. (метод бульбашки)
10) Вивести n перших слів масиву та к-сть їх повторів у тексті, де n – константа вказана на початку програми(у даному випадку 10)

2.2 Псевдокод алгоритму другого завдання

1) Початок
2) Якщо не кінець файлу, то зчитати слово
3) Кожну букву у слові перетворити в «маленьку», та очистити слова від розділових знаків у їх завершенні(наприклад слово “HelLo,” перетвориться у “hello”)
4) Перевірити чи є дане слово «стоп-словом», перелік яких міститься у масиві(у даному випадку стоп-словом вважається тире, хоча список можна поповнити)
5) Якщо дане слово – «стоп-слово», то перейти до пункту 2

6) Перевірити чи є дане слово у масиві, де ми зберігаємо слова, які уже зустрічали, к-сть їх повторів, перелік сторінок, на яких зустрічалось дане слово
7) Якщо дане слово раніше уже зустрічалось, то додати одиницю до к-сті його повторів у тексті, номер сторінки до переліку сторінок, на яких зустрічалось дане слово
8) Якщо дане слово раніше не зустрічалось, то додати дане слово до масиву, вказавши к-сть повторів рівним одиниці, додавши номер поточної сторінки до масиву сторінок, на яких зустрічалось дане слово
9) Посортувати масив слів в алфавітному порядку
10) Вивести слова та номери сторінок, на яких вони зустрічаються, за вийнятком слів, к-сть повторів яких більша за 100.

2.3 Програмна реалізація алгоритму розв'язання першого завдання(на C/C++)

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

struct Word{
    string value;
    int countR;
};

int main(){
    int i;
    const int SHOWED_WORDS = 10;
    int countStopWords = 7;
    string stopWords[] = {
        "the","in","for","an","of","at","by"
    };

    int resultArraySize = 0;
    Word* resultArray = new Word[resultArraySize];
    string readedWord;
    ifstream inputFile("text.txt");
    readNewWord:
        if(!(inputFile>>readedWord)){
            goto endOfReadingFile;
        }
        i = 0;
        toLowerCase:
            if(!readedWord[i]){
                goto endToLowerCase;
            }
            if(readedWord[i] >= 65 && readedWord[i] <= 90){
                readedWord[i]+=32;
            }
```

```
        i++;
    goto toLowerCase;
    endToLowerCase:
    i=0;

    isStopWord:
        if(i>=countStopWords){
            goto endIsStopWord;
        }
        if(stopWords[i]==readedWord){
            goto readNewWord;
        }
        i++;
        goto isStopWord;
    endIsStopWord:
    i=0;
    isAddedToResultArray:{
        if(i<=resultArraySize){
            if(readedWord==resultArray[i].value){
                resultArray[i].countR++;
                goto endAddToResultArray;
            }
            i++;
            goto isAddedToResultArray;
        }

        resultArraySize+=1;
        i=0;
        Word* copyResultArray = new Word[resultArraySize];
        copyArray:
            if(i<resultArraySize-1){
                copyResultArray[i]=resultArray[i];
            } else {
                goto endCopyArray;
            }
        i++;
        goto copyArray;
        endCopyArray:
        copyResultArray[resultArraySize-1] = {readedWord,1};
        resultArray=copyResultArray;
    }
    endAddToResultArray:
        i=0;
        goto readNewWord;
endOfReadingFile:
inputFile.close();

int j=0;
i=0;

iCycle:
    if(i>=resultArraySize){
        goto endICycle;
    }
    j=0;
    jCycle:
        if(j>=resultArraySize){
            goto endJCycle;
        }
        if(resultArray[i].countR>resultArray[j].countR){
            Word swaper = resultArray[i];
            resultArray[i] = resultArray[j];
            resultArray[j] = swaper;
        }
```

```
            j++;
            goto jCycle;
        endJCycle:
        i++;
        goto iCycle;
    endICycle:

    i=0;
    iCycle1:
        if((i>=SHOWED_WORDS)||(i>=resultArraySize)){
            goto endICycle1;
        }
        cout<< resultArray[i].value<< " - " << resultArray[i].countR<<endl;
        i++;
    goto iCycle1;
    endICycle1:
    return 0;
}
```

2.4 Програмна реалізація алгоритму розв'язання другого завдання(на C/C++)

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
// Програма написана для такої умови:
// 1800 символів = 1 сторінка

struct Word{
    string value;
    int* pages;
    int countR;
    int pagesCount;
};

int main(){
    int countStopWords = 7;
    string stopWords[] = {
        "-"
    };

    ifstream inputFile("text.txt");
    int resultArraySize = 0;
    Word* resultArray = new Word[resultArraySize];
    string readedWord;
    int symbolsCount = 0,
        k=0,
        i=0;

    readNewWord:{
        if(!(inputFile>>readedWord)){
            goto endOfReadingFile;
        }
        if(symbolsCount!=0) symbolsCount++;
        i=0;
        isStopWord:
            if(i>=countStopWords){
                goto endIsStopWord;
            }
            if(stopWords[i]==readedWord){
                symbolsCount++;
                goto readNewWord;
```

```
    }
    i++;
    goto isStopWord;
endIsStopWord:

string cleanedWord = "";
i = 0;
toLowerCase:
    if(!readedWord[i]){
        goto endToLowerCase;
    }

    if(readedWord[i]>=97 && readedWord[i]<=122){
        cleanedWord+=readedWord[i];
    }

    if(readedWord[i] >= 65 && readedWord[i] <= 90){
        readedWord[i]+=32;
        cleanedWord+=readedWord[i];
    }
    symbolsCount++;
    i++;
goto toLowerCase;
endToLowerCase:
readedWord = cleanedWord;
i=0;
isAddedToResultArray:{
    if(i<=resultArraySize){
        if(readedWord==resultArray[i].value){
            resultArray[i].countR++;
            int page1 = symbolsCount / 1800;
            if(symbolsCount % 1800>0){
                page1++;
            }
            if(resultArray[i].pages[resultArray[i].pagesCount-1]<page1){
                resultArray[i].pagesCount++;
                k = 0;
                int *pagesCopy = new int[resultArray[i].pagesCount];
                copyPages:
                    if(k<resultArray[i].pagesCount-1){
                        pagesCopy[k] = resultArray[i].pages[k];
                    } else {
                        goto endCopyPages;
                    }
                    k++;
                goto copyPages;
                endCopyPages:
                    resultArray[i].pages=pagesCopy;
                    resultArray[i].pages[resultArray[i].pagesCount-1] = page1;
            }
            goto endAddToResultArray;
        }
        i++;
        goto isAddedToResultArray;
    }

    resultArraySize+=1;
    i=0;
    Word* copyResultArray = new Word[resultArraySize];
    copyArray:
        if(i<resultArraySize-1){
            copyResultArray[i]=resultArray[i];
        } else {
            goto endCopyArray;
```

```
            }
        i++;
        goto copyArray;
        endCopyArray:
        int* page = new int[1];
        page[0] = symbolsCount / 1800;
        if(symbolsCount%1800>0){
            page[0]++;
        }
        copyResultArray[resultArraySize-1] = {readedWord,page,1,1};
        resultArray=copyResultArray;
    }
    endAddToResultArray:
        i=0;
        goto readNewWord;
}
endOfReadingFile:
inputFile.close();

int j=0;
i=0;

iCycle:
    if(i>=resultArraySize){
        goto endICycle;
    }
    j=0;
    jCycle:
        if(j>=resultArraySize){
            goto endJCycle;
        }
        if(resultArray[i].value<resultArray[j].value){
            Word swaper = resultArray[i];
            resultArray[i] = resultArray[j];
            resultArray[j] = swaper;
        }
        j++;
        goto jCycle;
    endJCycle:
    i++;
    goto iCycle;
endICycle:

j=0;
i=0;

iCycle1:
    if(i>=resultArraySize){
        goto endICycle1;
    }
    if(resultArray[i].countR>100){
        i++;
        goto iCycle1;
    }
    cout<< resultArray[i].value<< " - ";
    j=0;
    jCycle1:
        if(j>=resultArray[i].pagesCount){
            goto endJCycle1;
        }
      cout<<resultArray[i].pages[j]<<" ";
        j++;
        goto jCycle1;
    endJCycle1:
```

```
        cout<<endl;
        i++;
        goto iCycle1;
    endICycle1:

    return 0;
}
```

3 ТЕСТУВАННЯ

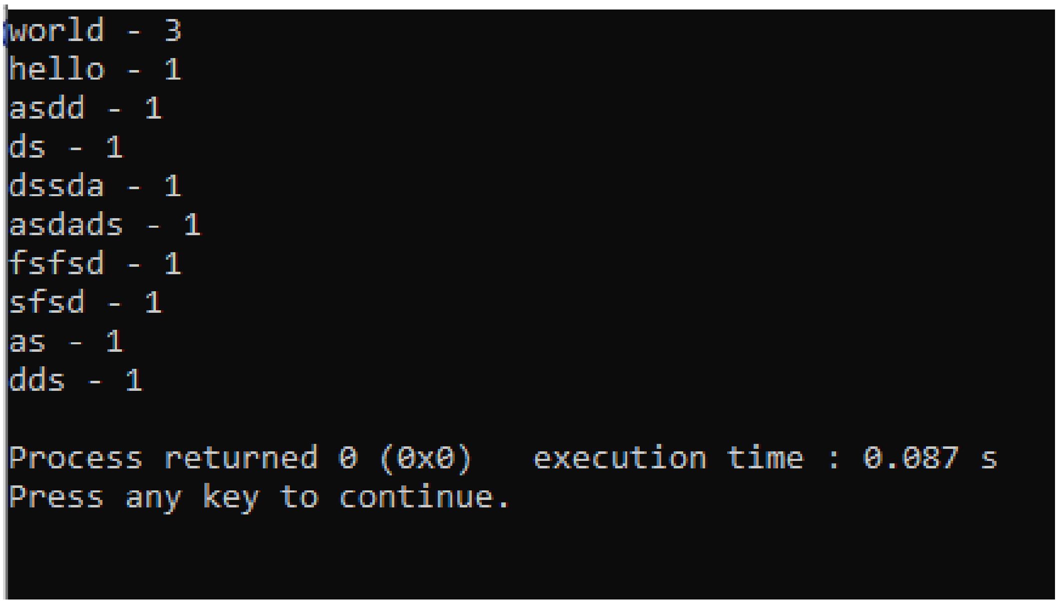
3.1 Приклад роботи програми для розв’язання першого завдання

Вхідні дані:

Файл text.txt:

```
HelLo world at World asdd ds dssda asdads fsfsd sfsd as dds qweqwe world
```

Вихідні дані:



```
world - 3
hello - 1
asdd - 1
ds - 1
dssda - 1
asdads - 1
fsfsd - 1
sfsd - 1
as - 1
dds - 1

Process returned 0 (0x0)   execution time : 0.087 s
Press any key to continue.
```

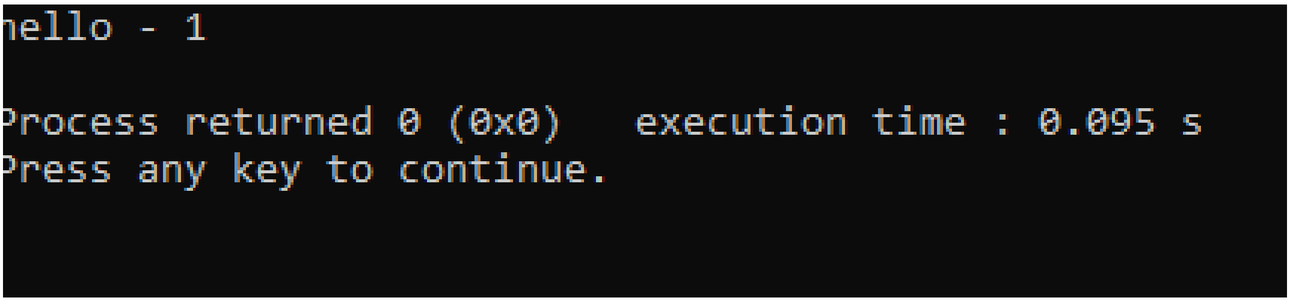
3.2 Приклад роботи програми для розв’язання другого завдання

Вхідні дані:

Файл text.txt:

```
a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a
a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a
a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a
a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a
a a a a a a a a a hello
```

Вихідні дані:



```
hello - 1

Process returned 0 (0x0)   execution time : 0.095 s
Press any key to continue.
```

4 ВИСНОВОК

Під час виконання даної лабораторної роботи я дослідив підхід написання програми з використанням оператора goto. Даний оператор дозволяє програмісту керувати потоком виконання програми. Такий підхід дуже часто використовується для оптимізації коду, що є дуже корисним, коли кожна мілісекунда на рахунку.