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»

«Імперативне програмування»

Виконав: ІП-02 Демченко О. С.

Київ 2021

Лабораторна робота 1

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.

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.

Завдання 1

Алгоритм

Для реалізації даної задачі необхідно:

1. Посимвольно зчитати всі символи з файлу в буфер.
2. Пройтись по буферу та порахувати кількість слів.
3. Додати всі слова в масив.
4. Прибрати короткі слова, що менші за 3 символи.
5. Прибрати слова, що повторюються та порахувати кількість повторень.
6. Відсортувати слова та повторення за більшістю повторень за спаданням.
7. Вивести у файл.

Реалізація

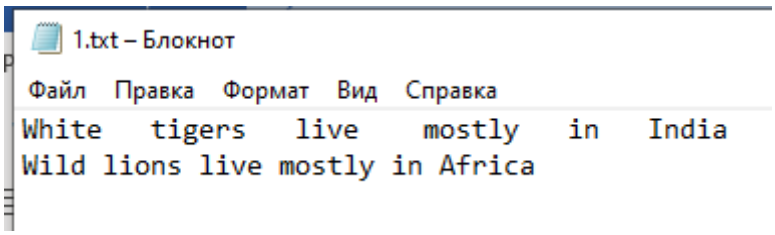
```
#include <iostream>
#include <fstream>

int main()
{
    FILE* fp;
    int i = 0, j = 0, countOfWords = 0, maxWordLength = 0;;
    char current;

    fopen_s(&fp, "1.txt", "rb");
    if (fp == NULL) return 0;

    char buffer[10000];
    int bufferLength = 0;

startReading:
    current = fgetc(fp);
    if (current == EOF)
        goto endReading;

    if (current != '\r' && current != '\n')
    {
        if ((current == ' ' && bufferLength != 0 && buffer[bufferLength - 1] != ' ')
|| current != ' ')
        {
            if (static_cast<int>(current) >= 65 && static_cast<int>(current) <= 90)
                current = static_cast<char>(static_cast<int>(current) + 32);
            buffer[bufferLength] = current;
            bufferLength = bufferLength + 1;
        }
    }
    else if (current == '\n')
    {
        buffer[bufferLength] = ' ';
        bufferLength++;
    }

    goto startReading;
endReading:
    fclose(fp);

    if (bufferLength == 0) return 0;
    if (buffer[bufferLength - 1] == ' ') bufferLength--;
```

```
    countOfWords++;
    i = 0;
checkSize:
    if (i == bufferLength) goto endChecking;

    if (buffer[i] == ' ') countOfWords++;

    i++;
    goto checkSize;
endChecking:

    int notSameWords = 0;
    auto arrWords = new std::string[countOfWords];

    i = 0;
    j = 0;

add:
    if (i == bufferLength) goto endAdding;
    if (buffer[i] == ' ' && i != 0)
    {
        j++;
        i++;
        goto add;
    }

    arrWords[j] += buffer[i];

    i++;
    goto add;
endAdding:

    int* count = new int[countOfWords];

    i = 0;

makingToZero:
    if (i == countOfWords) goto endMakingToZero;
    count[i] = 1;
    i++;
    goto makingToZero;
endMakingToZero:

    i = 0;
    j = 0;

deletingSmallWords:
    if (i > countOfWords)
        goto endDeletingSmallWords;
    if (arrWords[i].size() < 3)
        arrWords[i] = "";

    i++;
    goto deletingSmallWords;
endDeletingSmallWords:


    i = 0;
    j = 0;
count:

    if (i >= countOfWords)
    {
        goto endCounting;
```

```
    }
    if (arrWords[i] != "")
    {
        int k = i + 1;
    intern:
        if (k >= countOfWords)
            goto endIntern;
        if (arrWords[i] == arrWords[k])
        {
            arrWords[k] = "";
            count[k] = 0;
            count[i]++;
        }
        k++;
        goto intern;
    endIntern:
        int s;
    }
    i++;
    goto count;



endCounting:


    //fopen_s(&fp, "output.txt", "w");
    //fwrite()
    std::string strTemp;
    int temp;

    i = 0;
externalSort:
    if (i >= countOfWords - 1) goto endSort;

    j = 0;
internalSort:
    if (j >= countOfWords - i - 1) goto endInternal;

    if (count[j] < count[j + 1])
    {
        strTemp = arrWords[j];
        temp = count[j];
        arrWords[j] = arrWords[j + 1];
        count[j] = count[j + 1];
        arrWords[j + 1] = strTemp;
        count[j + 1] = temp;
    }
        j++;
        goto internalSort;
    endInternal:

    i++;
    goto externalSort;
endSort:

    std::ofstream outFile("output.txt");

    i = 0;
output:
    if (i >= countOfWords) goto endOutput;
    if (arrWords[i] != "")
    {
        outFile << arrWords[i] << " - " << count[i] << std::endl;
    }
```

```
    i++;
    goto output;
endOutput:

    outFile.close();
}
```

Приклад роботи:



Завдання 2

Алгоритм

Для реалізації даної задачі необхідно:

1. Посимвольно зчитати файл у буфер, ігноруючи розділові знаки та цифри.
2. Створити масив з кількістю слів.
3. Записати у масив дані, що містяться в структурі [слово; сторінка].
4. Видалити слова, що повторюються більше 100 разів.
5. Видалити однакові слова, що знаходяться на одній сторінці, щоб уникнути повторень.
6. Відсортувати дані.
7. Вивести у файл.

Реалізація

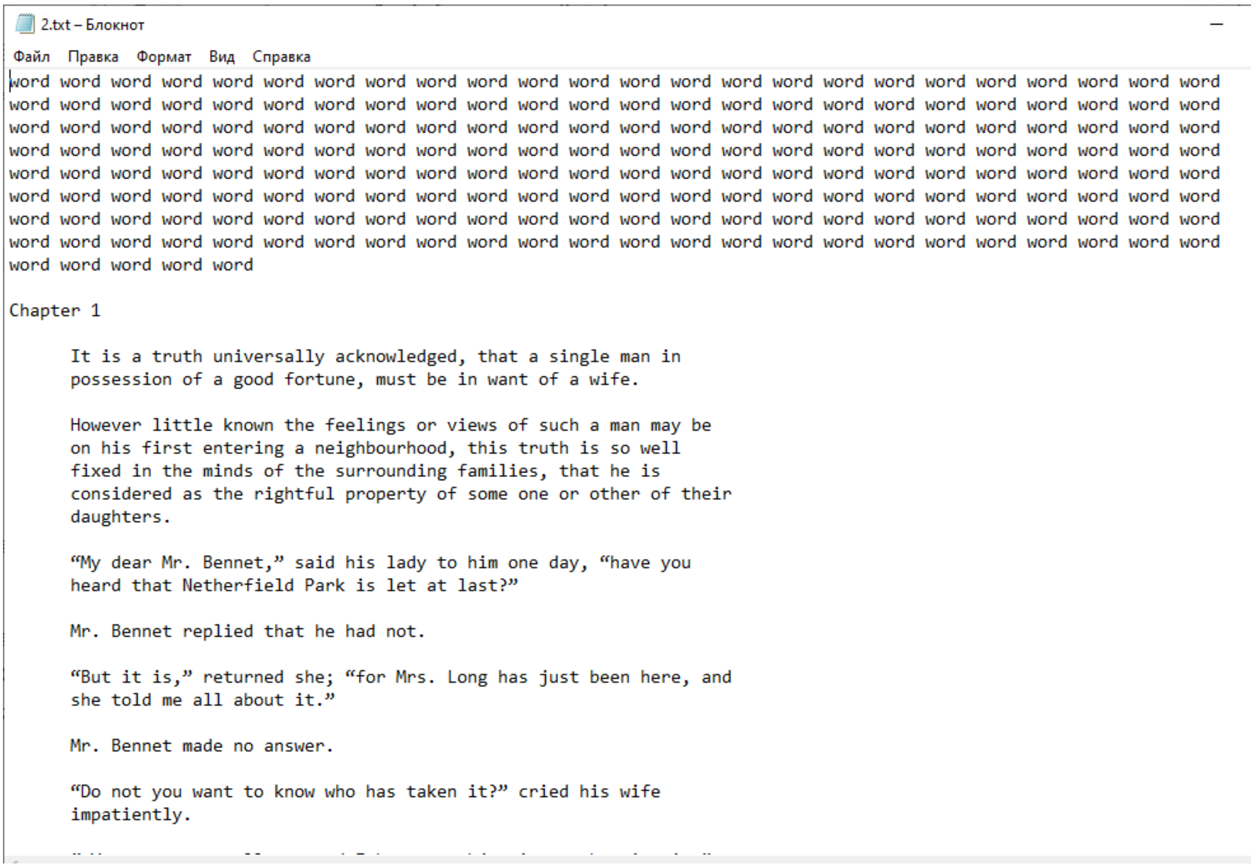
```
#include <iostream>
#include <string>
#include <fstream>

struct Word
{
    std::string word;
    int page = 0;
};

int main()
{
    FILE* fp;
    int i = 0, j = 0, countOfWords = 0;
    char current;
    int currPage = 1;
    int countLinesProccessed = 0;


    fopen_s(&fp, "2.txt", "rb");

    if (fp == NULL) return 0;

    int bufferLength = 0;
    std::string currWord = "";
    std::string allWords;
    int bufferSize = 0;

startReading:
    current = fgetc(fp);
    if (current == EOF)
        goto endReading;

    if (current != ' ' && current != ',' && current != '.' && current != '-' &&
current != ':' && current != '!' && current != '?'
        && current != '\r' && current != '\n' && current != -30 && current != -128
&& current != -99 && current != '_' && current != 'ь' && current != ';')
    {
        currWord += current;
        bufferSize++;
    }
    else
    {
        if (currWord != "" && currWord != "1" && currWord != "0" && currWord != "2"
&& currWord != "3" && currWord != "4" && currWord != "5"
            && currWord != "6" && currWord != "7" && currWord != "8" && currWord !=
"9" && current != -30 && current != -128 && current != -99 && current != '_' &&
current != 'ь')
        {
            allWords += currWord + " ";
            countOfWords++;
            bufferSize++;
            currWord = "";
        }
    }

    goto startReading;
endReading:
    fclose(fp);

    int countOfSymbols = 0;
    Word* arrayWords = new Word[countOfWords];
    i = 0;
```

```
    j = 0;
beginAddingToStruct:
    if (j >= allWords.size()) goto endAddingToStruct;


    countOfSymbols++;
    if (allWords[j] == ' ')
    {
        i++;
        j++;
        goto beginAddingToStruct;
    }
    if (static_cast<int>(allWords[j]) >= 65 && static_cast<int>(allWords[j]) <= 90)
        allWords[j] = static_cast<char>(static_cast<int>(allWords[j]) + 32);

    arrayWords[i].word += allWords[j];
    arrayWords[i].page = static_cast<int>(countOfSymbols / 1800) + 1;


    j++;
    goto beginAddingToStruct;
endAddingToStruct:


    int countWord = 0;

    i = 0;
externalDeletingOver100:
    if (i >= countOfWords) goto externalEndDeletingOver100;

    countWord = 0;
    currWord = arrayWords[i].word;

    j = 0;
internalCountSycle:
    if (j >= countOfWords) goto endInternalCountSycle;



    if (currWord == arrayWords[j].word)
    {
        countWord++;
    }

    j++;
    goto internalCountSycle;
endInternalCountSycle:


    if (countWord > 100)
    {
        j = 0;
    internalDeletingSycle:
        if (j >= countOfWords) goto endInternalDeletingSycle;


        if (arrayWords[i].word == currWord)
        {
            arrayWords[i].word = "";
            arrayWords[i].page = 0;
        }

        j++;
        goto internalDeletingSycle;
```

```
    }
endInternalDeletingSycle:
    i++;
    goto externalDeletingOver100;
externalEndDeletingOver100:

    i = 0;
deleteWithSamePages:
    if (i >= countOfWords) goto endDeleteWithSamePages;

    currWord = arrayWords[i].word;
    j = 0;
internalDeleteWithSamePages:
    if (j >= countOfWords) goto endInternalDeleteWithSamePages;

    if (currWord == arrayWords[j].word && i != j)
    {
        if (arrayWords[i].page == arrayWords[j].page)
        {
            arrayWords[j].page = 0;
            arrayWords[j].word = "";
        }
    }

    j++;
    goto internalDeleteWithSamePages;
endInternalDeleteWithSamePages:

    i++;
    goto deleteWithSamePages;
endDeleteWithSamePages:

    std::string strTemp;
    int temp;
    i = 0;
externalSort:
    if (i >= countOfWords - 1) goto endSort;

    j = 0;
internalSort:
    if (j >= countOfWords - i - 1) goto endInternal;

    if (arrayWords[j].word > arrayWords[j + 1].word)
    {
        strTemp = arrayWords[j].word;
        temp = arrayWords[j].page;
        arrayWords[j].word = arrayWords[j + 1].word;
        arrayWords[j].page = arrayWords[j + 1].page;
        arrayWords[j + 1].word = strTemp;
        arrayWords[j + 1].page = temp;
    }
    j++;
    goto internalSort;
endInternal:

    i++;
    goto externalSort;
endSort:

    std::ofstream outFile("output.txt");

    i = 0;
outputToFile:
    if (i >= countOfWords) goto endOutputToFile;
```

```
    currWord = arrayWords[i].word;

    if (currWord != "")
    {
        outFile << currWord << " - ";

        j = 0;
    internalOutputToFile:
        if (j >= countOfWords) goto internalEndOutputToFile;
        if (currWord == arrayWords[j].word)
        {
            outFile << arrayWords[j].page << " ";
            arrayWords[j].word = "";
        }
        j++;
        goto internalOutputToFile;
    internalEndOutputToFile:

        outFile << std::endl;
    }
    i++;
    goto outputToFile;
endOutputToFile:

    outFile.close();
}
```

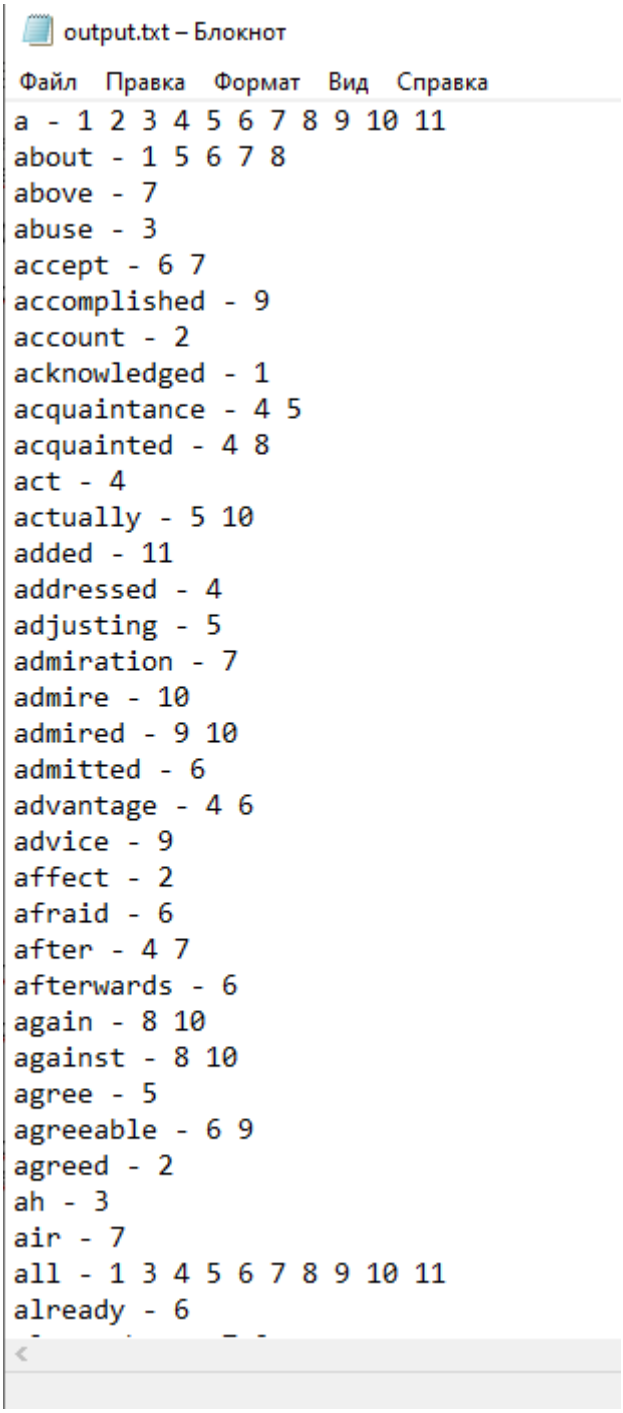
Приклад роботи:



2.txt – Блокнот

Файл Правка Формат Вид Справка

```
word word word word word word word word word word word word word word word word word word word word word word word word
word word word word word word word word word word word word word word word word word word word word word word word word
word word word word word word word word word word word word word word word word word word word word word word word word
word word word word word word word word word word word word word word word word word word word word word word word word
word word word word word word word word word word word word word word word word word word word word word word word word
word word word word word word word word word word word word word word word word word word word word word word word word
word word word word word word word word word word word word word word word word word word word word word word word word
word word word word word word word word word word word word word word word word word word word word word word word word
word word word word word

Chapter 1

        It is a truth universally acknowledged, that a single man in
        possession of a good fortune, must be in want of a wife.

        However little known the feelings or views of such a man may be
        on his first entering a neighbourhood, this truth is so well
        fixed in the minds of the surrounding families, that he is
        considered as the rightful property of some one or other of their
        daughters.

        “My dear Mr. Bennet,” said his lady to him one day, “have you
        heard that Netherfield Park is let at last?”

        Mr. Bennet replied that he had not.

        “But it is,” returned she; “for Mrs. Long has just been here, and
        she told me all about it.”

        Mr. Bennet made no answer.

        “Do not you want to know who has taken it?” cried his wife
        impatiently.
```



Висновок

Під час виконання даної лабораторної роботи я виконав 2 задачі (term frequency та словникове індексування), використовуючи C++ та конструкцій GOTO без використання динамічних структур, циклів та функцій. Набув досвіду у даній методології програмування.