

*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: Харчук Назарій ІІІ-01
Перевірив: Очеретяний О. С.

Київ 2021

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
```

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
```

Виконання

Алгоритм першого завдання

1. Відкриваємо файл;
2. Зчитуємо слово(якщо не зчиталось, переходимо до пункту 7);
3. Переводимо слово в нижній регістр та залишаємо лише буквенні символи, “ - ” або “ ‘ ”;
4. Якщо слово пусте або є стоп-словом, переходимо до пункту 2;
5. Якщо слово ми зчитували раніше, збільшуємо кількість повторів;
6. Якщо слово раніше не зчитувалось, додаємо його до масиву і встановлюємо кількість повторів, що дорівнює 1;
7. Сортуюмо масив;
8. Виводимо перші N елементів;
9. Закриваємо файл.

Код першого завдання

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

using namespace std;

int main() {
    cout << "\tTask 1)\n";

    string word, correctWord, tempStr;
    int i, j, countStop = 22, templnt, N = 25;
    string stopWords[] = {"the", "for", "at", "a", "in", "is", "on", "are", "am", "do", "did", "to", "so",
"of", "or", "not", "and", "was", "no", "but", "has", "us" };
    int countWords = 0;
    string* allWords = new string[countWords];
    int* allCount = new int[countWords];
    string* tempAllWords;
    int* tempAllCount;

    ifstream input;
    input.open("input1.txt");

newWord:
    if (!(input >> word)) goto point;
    correctWord = "";
    i = 0;

wordCycle:
    if (!word[i]) {
        i = 0;
        goto mbStopWord;
    }

    if ((word[i] >= 'a' && word[i] <= 'z') || word[i] == '-' || word[i] == '\\') {
        correctWord += word[i++];
        goto wordCycle;
    }

    if (word[i] >= 'A' && word[i] <= 'Z') {
        correctWord += word[i++]+32;
        goto wordCycle;
    }

    i++;
    goto wordCycle;

mbStopWord:
    if(stopWords[i] == correctWord || correctWord == "" || correctWord == "-" || correctWord ==
"\\") goto newWord;

    if (i < countStop-1) {
```

```

        i++;
        goto mbStopWord;
    }

    //cout << correctWord << ";" << endl;

    i = 0;
countPoint:
    if (i < countWords) {
        if (allWords[i] == correctWord) {
            allCount[i]++;
            goto newWord;
        }
        i++;
        goto countPoint;
    }

    tempAllWords = new string[countWords+1];
    tempAllCount = new int[countWords+1];
    i = 0;
copyMass:
    if (i < countWords) {
        tempAllWords[i] = allWords[i];
        tempAllCount[i] = allCount[i];
        i++;
        goto copyMass;
    }
    tempAllWords[countWords] = correctWord;
    tempAllCount[countWords] = 1;

    countWords++;
    delete[] allWords;
    delete[] allCount;
    allWords = tempAllWords;
    allCount = tempAllCount;
    goto newWord;

point:
    i = 0;
sort1:
    if (i < countWords)
    {
        j = i + 1;
        sort2:
        if (j < countWords)
        {
            if (allCount[i] < allCount[j])
            {
                tempStr = allWords[i];
                allWords[i] = allWords[j];
                allWords[j] = tempStr;
            }
        }
    }

```

```

        templnt = allCount[i];
        allCount[i] = allCount[j];
        allCount[j] = templnt;
    }
    j++;
    goto sort2;
}
i++;
goto sort1;
}

i = 0;
outPoint:
    if (i < countWords && i < N) {
        cout << allWords[i] << " - ";
        cout << allCount[i] << endl;
        i++;
        goto outPoint;
    }

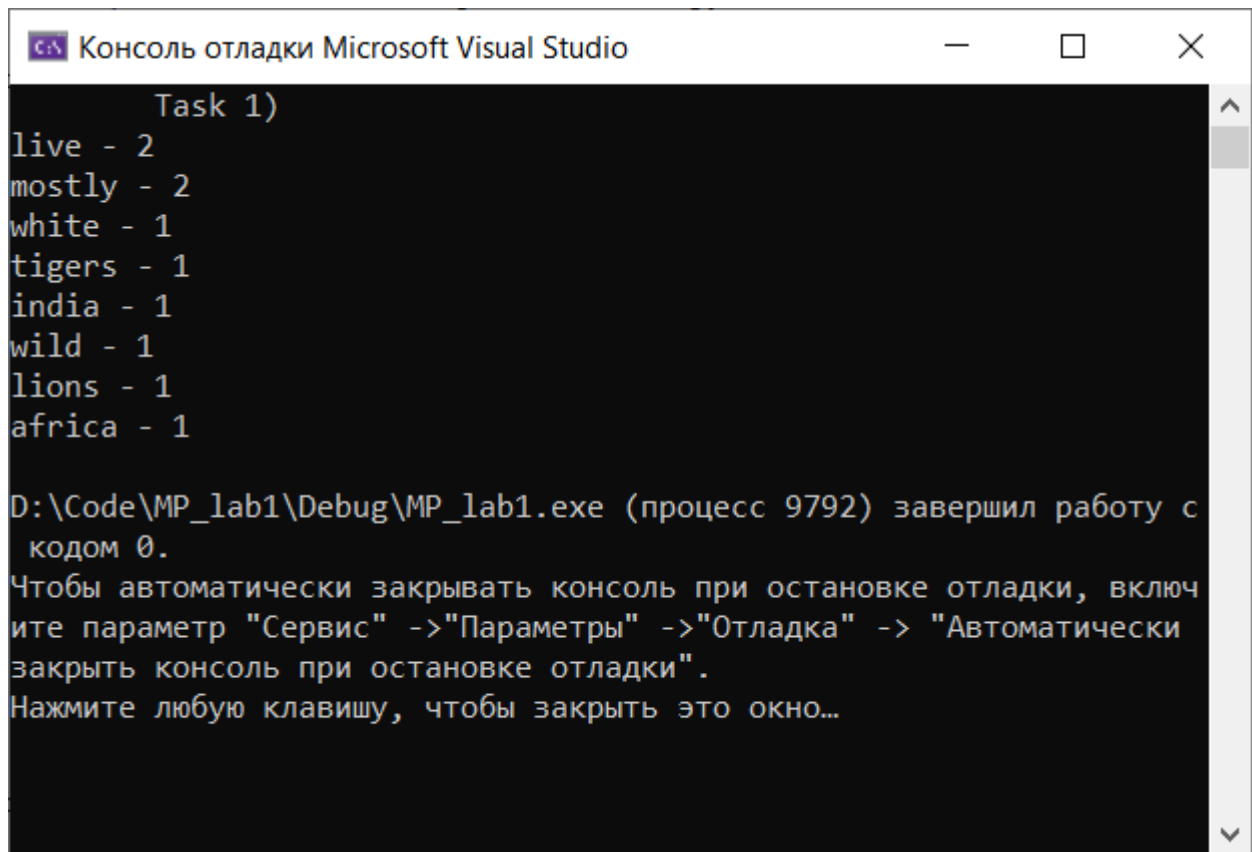
    input.close();

    delete[] allWords;
    delete[] allCount;

    return 0;
}

```

Приклад роботи першого завдання



The image shows a screenshot of the 'Консоль отладки Microsoft Visual Studio' (Microsoft Visual Studio Debug Console) window. The window has a title bar with the Visual Studio logo and standard minimize, maximize, and close buttons. The console output is as follows:

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

D:\Code\MP_lab1\Debug\MP_lab1.exe (процесс 9792) завершил работу с
кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включ
ите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Автоматически
закрывать консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно...
```

Алгоритм другого завдання

1. Відкриваємо файл;
2. Перевіряємо, чи не закінчився поточний рядок(якщо закінчився, збільшуємо кількість рядків);
3. Вираховуємо номер сторінки;
4. Зчитуємо слово(якщо не зчиталось, переходимо до пункту 9);
5. Переводимо слово в нижній регістр та залишаємо лише буквенні символи, “ - ” або “ ‘ ”;
6. Якщо слово пусте або є стоп-словом, переходимо до пункту 2;
7. Якщо слово ми зчитували раніше, збільшуємо кількість повторів та один раз записуємо номер сторінки, в якій його знайдено;
8. Якщо слово раніше не зчитувалось, додаємо його до масиву і встановлюємо кількість повторів, що дорівнює 1 та записуємо номер сторінки, в якій його знайдено;
9. Сортують масив;
10. Виводимо в алфавітному порядку слова та сторінки, на яких вони трапляються(слова, що траплялися більше 100 разів);
11. Закриваємо файл.

Код другого завдання

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

using namespace std;

int main() {
    cout << "\tTask 2)\n";

    string word, correctWord, tempStr, tempPages, strToStr;
    int i, j, countStop = 22, tempInt;
    string stopWords[] = { "the", "for", "at", "a", "in", "is", "on", "are", "am", "do", "did", "to", "so",
"of", "or", "not", "and", "was", "no", "but", "has", "us" };
    int countWords = 0;
    string* allWords = new string[countWords];
    int* allCount = new int[countWords];
    int* allLastPages = new int[countWords];
    string* allPages = new string[countWords];
    string* tempAllWords;
    int* tempAllCount;
    string* tempAllPages;
    int* tempLastPages;
    int lines = 1, pages, tempPage;

    ifstream input;
    input.open("input.txt");

newLine:

    if (input.peek() == '\n') {
        lines++;
        input.get();
        goto newLine;
    }
    //cout << lines << endl;

    pages = lines / 45 + 1;
    if (lines % 45 == 0) pages = lines / 45;

    if (!(input >> word)) goto point;
    correctWord = "";
    i = 0;

wordCycle:
    if (!word[i]) {
        i = 0;
        goto mbStopWord;
    }

    if ((word[i] >= 'a' && word[i] <= 'z') || word[i] == '-' || word[i] == '\\') {
        correctWord += word[i++];
    }
```

```

        goto wordCycle;
    }

    if (word[i] >= 'A' && word[i] <= 'Z') {
        correctWord += word[i++] + 32;
        goto wordCycle;
    }

    i++;
    goto wordCycle;

mbStopWord:
    if (stopWords[i] == correctWord || correctWord == "" || correctWord == "-" || correctWord ==
"\") goto newLine;

    if (i < countStop - 1) {
        i++;
        goto mbStopWord;
    }

    //cout << correctWord << ";\\t" << pages << endl;

    i = 0;
countPoint:
    if (i < countWords) {
        if (allWords[i] == correctWord) {
            allCount[i]++;
            if (allCount[i] <= 100 && allLastPages[i] != pages) {
                strToStr = "";
                tempPage = pages;
                toStr:
                    if (tempPage != 0) {
                        strToStr = char(tempPage % 10 + 48) + strToStr;
                        tempPage = tempPage / 10;
                        goto toStr;
                    }
                allPages[i] += ", " + strToStr ;
                //cout << "!!!\\t" <<strToStr << endl;
                allLastPages[i] = pages;
            }
            goto newLine;
        }
        i++;
        goto countPoint;
    }

    tempAllWords = new string[countWords + 1];
    tempAllCount = new int[countWords + 1];
    tempAllPages = new string[countWords + 1];
    tempLastPages = new int[countWords + 1];
    i = 0;
copyMass:

```

```

if (i < countWords) {
    tempAllWords[i] = allWords[i];
    tempAllCount[i] = allCount[i];
    tempAllPages[i] = allPages[i];
    tempLastPages[i] = allLastPages[i];
    i++;
    goto copyMass;
}
tempAllWords[countWords] = correctWord;
tempAllCount[countWords] = 1;
tempAllPages[countWords] = "";
strToStr = "";
tempPage = pages;
toStr2:
    if (tempPage != 0) {
        strToStr = char(tempPage % 10 + 48) + strToStr;
        tempPage = tempPage / 10;
        goto toStr2;
    }
tempAllPages[countWords] += strToStr;
//cout << "\t" << tempAllPages[i] << endl;
tempLastPages[countWords] = pages;

countWords++;
delete[] allWords;
delete[] allCount;
delete[] allLastPages;
delete[] allPages;
allWords = tempAllWords;
allCount = tempAllCount;
allPages = tempAllPages;
allLastPages = tempLastPages;
goto newLine;

```

point:

```
i = 0;
```

sort1:

```

if (i < countWords)
{
    j = i + 1;
    sort2:
        if (j < countWords)
        {
            if (allWords[i] > allWords[j])
            {
                tempStr = allWords[i];
                allWords[i] = allWords[j];
                allWords[j] = tempStr;

                tempInt = allCount[i];
                allCount[i] = allCount[j];
                allCount[j] = tempInt;
            }
        }
    }
}

```

```

        tempStr = allPages[i];
        allPages[i] = allPages[j];
        allPages[j] = tempStr;

        /*tempInt = allLastPages[i];
        allLastPages[i] = allLastPages[j];
        allLastPages[j] = tempInt;*/
    }
    j++;
    goto sort2;
}
i++;
goto sort1;
}

i = 0;
outPoint:
    if (i < countWords && allCount[i]<=100) {
        cout << allWords[i] << " - ";
        //cout << allCount[i] << " - ";
        //cout << allLastPages[i] << " - ";
        cout << allPages[i] << endl;
        i++;
        goto outPoint;
    }

    input.close();

    delete[] allWords;
    delete[] allCount;
    delete[] allPages;
    delete[] allLastPages;

    return 0;
}

```

Приклад роботи другого завдання

```
Консоль отладки Microsoft Visual Studio

Task 2)
a-shooting - 305
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, 152, 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, 207, 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, 242, 260, 269, 299, 304
absurd - 61, 163, 171, 296, 302
absurdities - 127, 217
absurdity - 189
abundant - 227
abundantly - 67, 85, 125
abuse - 6, 166
abused - 179, 197
abusing - 31, 299
abusive - 184, 316
accede - 166
acceded - 207
acceding - 249
accent - 188, 204, 212, 222
accents - 192, 233
accept - 10, 31, 76, 92, 94, 107, 158, 160, 161, 173, 213, 283, 289, 291, 300, 318
acceptable - 60, 92, 94, 100, 143, 256
acceptance - 94, 129, 157, 213
```