Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Кафедра систем штучного інтелекту

Звіт

про виконання лабораторних та практичних робіт блоку № 5

На тему: «Файли. Бінарні Файли. Символи і Рядкові Змінні та Текстові Файли. Стандартна бібліотека та деталі/методи роботи з файлами. Створення й використання бібліотек.»

з дисципліни: «Основи програмування»

до:

ВНС Лабораторної Роботи № 6
ВНС Лабораторної Роботи № 8
ВНС Лабораторної Роботи № 9
Алготестер Лабораторної Роботи №4
Алготестер Лабораторної Роботи №6
Практичних Робіт до блоку №5

Виконав:
Студент групи ШІ-13
Недосіка Назарій Вадимович

Львів 2024

Тема роботи: Файли. Бінарні Файли. Символи і Рядкові Змінні та Текстові Файли. Стандартна бібліотека та деталі/методи роботи з файлами. Створення й використання бібліотек.

Мета роботи: Навчитися записувати і зчитувати інформацію з файлу стилями мов C та C++. Базово розібратися що таке бібліотека і де \\ використовують.

Теоретичні відомості:

1. Лекції, практичні
2. W3schools
3. ChatGPT

Виконання роботи

Завдання 1: Class Practice Work

Задача 1

Реалізувати функцію створення файла і запису в нього даних:

enum FileOpResult { Success, Failure, … };

FileOpResult write_to_file(char *name, char *content);

Умови задачі:

- створити файл із заданим ім’ям; якщо файл існує – перезаписати його вміст
- написати код стійкий до різних варіантів вхідних параметрів
- name – ім’я, може не включати шлях
- записати у файл вміст стрічки content, прочитати content із стандартного вводу
- повернути статус операції: Success – все пройшло успішно, Failure – файл не вдалося створити, або збій операції відкриття, запису даних, чи закриття файла.

Задача 2

Реалізувати функцію створення файла і запису в нього даних:

enum FileOpResult { Success, Failure, … };

FileOpResult copy_file(char *file_from, char *file_to);

Умови задачі:

- копіювати вміст файла з ім’ям file_from у файл з ім’ям file_to; написати код стійкий до різних варіантів вхідних параметрів, обробити всі можливі варіанти відсутності одного з файлів
- file_from, file_to – можуть бути повним або відносним шляхом

- повернути статус операції: Success – все пройшло успішно, Failure – файл не вдалося створити, або збій операції відкриття, читання чи запису даних, закриття файла.

Код:

```cpp
#include <iostream>
#include <fstream>
#include <vector>

using namespace std;

enum FileOpResult
{
    Success,
    Failure,
    OpenError,
    OutputError,
    CloseError
};

FileOpResult write_to_file(char *name, char *content)
{
    fstream file(name, ios::out);
    if (!file.is_open())
        return FileOpResult::OpenError;

    file << content;
    if (file.fail())
        return FileOpResult::OutputError;

    file.close();
    if (file.bad())
        return FileOpResult::CloseError;

    return FileOpResult::Success;
}
FileOpResult copyFile(char *file_from, char *file_to)
{
    fstream file1(file_from, ios::in);
    fstream file2(file_to, ios::out);
    if (!file1.is_open() || !file2.is_open())
        return FileOpResult::OpenError;

    file2 << file1.rdbuf();

    if (file1.fail() || file2.fail())
        return FileOpResult::OutputError;

    file1.close();
    file2.close();
    if (file1.bad() || file2.bad())
        return FileOpResult::CloseError;

    return FileOpResult::Success;
}
std::string interpretCode(FileOpResult resultCode)
{
    switch (resultCode)
    {
    case 0:
        return "Success";
        break;
    case 1:
        return "Failure";
        break;
    case 2:
        return "OpenError";
        break;
    case 3:
        return "OutputError";
        break;
    case 4:
        return "CloseError";
        break;
    default:
        break;
    }
    return "";
}
int main()
{
    string name = "PracticeFile.txt", content;
    cout << "enter content for file: ";
    getline(cin, content);
    FileOpResult resultCode = write_to_file(name.data(), content.data());

    cout << "\nStatus code: " << interpretCode(resultCode) << endl;

    string name1 = "PracticeFile.txt", name2;
    cout << "\nenter name of file to copy: ";
    getline(cin, name2);
    FileOpResult resultCode2 = copyFile(name1.data(), name2.data());

    cout << "\nStatus code: " << interpretCode(resultCode2) << endl;
```

Вивід в терміналі:

```
enter content for file: test

Status code: Success

enter name of file to copy: copyOfFile

Status code: Success
```

Час виконання ~ 2 години

Завдання 2: VNS Lab 6 - Task 1-14

Задано рядок, що складається із символів. Символи поєднуються в слова. Слова одне від одного відокремлюються одним або декількома пробілами. Наприкінці тексту ставиться крапка. Текст містить не більше 255 символів. Виконати ввід рядка, використовуючи функцію gets(s) і здійснити обробку рядка у відповідності зі своїм варіантом.

Перетворити рядок таким чином, щоб у ньому залишилися тільки слова, що містять букви й цифри, інші слова знищити.

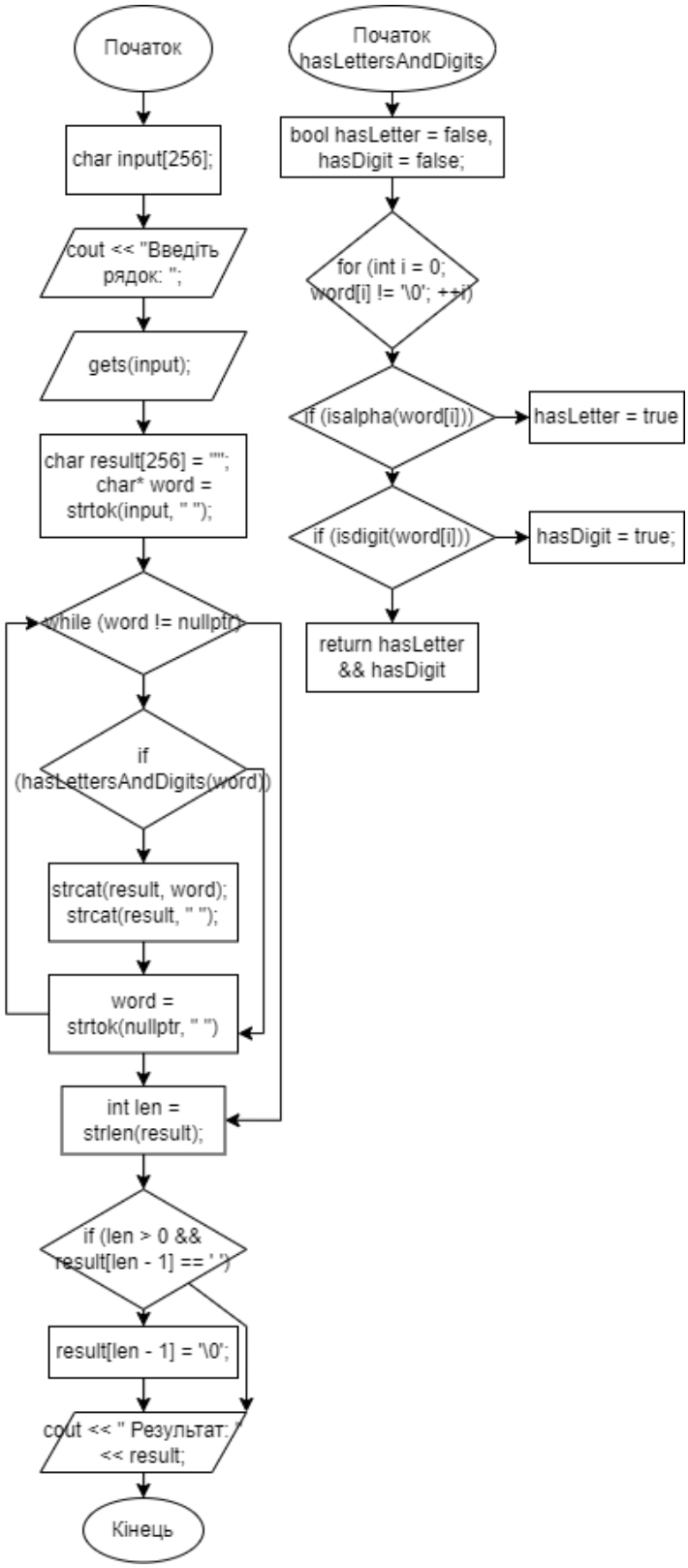
Код:

```cpp
#include <iostream>
#include <cstring>
#include <cctype>
using namespace std;

bool hasLettersAndDigits(const char* word) {
    bool hasLetter = false, hasDigit = false;

    for (int i = 0; word[i] != '\0'; ++i) {
        if (isalpha(word[i])) hasLetter = true;
        if (isdigit(word[i])) hasDigit = true;
    }

    return hasLetter && hasDigit;
}

int main() {
    char input[256];
    cout << "Введіть рядок: ";
    gets(input);

    char result[256] = "";
    char* word = strtok(input, " ");


    while (word != nullptr) {
        if (hasLettersAndDigits(word)) {
            strcat(result, word);
            strcat(result, " ");
        }
        word = strtok(nullptr, " ");
    }


    int len = strlen(result);
    if (len > 0 && result[len - 1] == ' ') {
        result[len - 1] = '\0';
    }

    cout << " Результат: " << result;
    return 0;
}
```

Вивід в терміналі:

```
Введіть рядок: th1s is a tes7 string
Результат: th1s tes7
```

Блок схема до коду:



Час виконання ~ 1.5 години

Завдання 3:VNS Lab 8 - Task 1-14

Сформувати двійковий файл із елементів, заданої у варіанті структури, роздрукувати його вмістиме, виконати знищення й додавання елементів у відповідності зі своїм варіантом, використовуючи для пошуку елементів що знищуються чи додаються, функцію. Формування, друк, додавання й знищення елементів оформити у вигляді функцій. Передбачити повідомлення про помилки при відкритті файлу й виконанні операцій вводу/виводу.

Структура "Стадіон":
- назва;
- адреса;
- місткість;
- види спорту.

Знищити елемент із заданою назвою, додати 2 елементи після елемента із зазначеним номером.

Вивід в терміналі:

```
Введіть індекс, після якого додати два нових стадіони: 2
Введіть назву першого нового стадіону: Old Trafford
Введіть адресу першого нового стадіону: Manchester, England
Введіть місткість першого нового стадіону: 74000
Введіть види спорту для першого нового стадіону: Football
Введіть назву другого нового стадіону: Everton Stadium
Введіть адресу другого нового стадіону: Liverpool, England
Введіть місткість другого нового стадіону: 52000
Введіть види спорту для другого нового стадіону: Football
Два нових стадіони успішно додані.

Вміст після додавання двох нових елементів:
Назва: Olympic Stadium
Адреса: Kyiv, Ukraine
Місткість: 70000
Види спорту: Football

Назва: Stamford Bridge
Адреса: London, England
Місткість: 40000
Види спорту: Football

Назва: Camp Nou
Адреса: Barcelona, Spain
Місткість: 99000
Види спорту: Football

Назва: Old Trafford
Адреса: Manchester, England
Місткість: 74000
Види спорту: Football

Назва: Everton Stadium
Адреса: Liverpool, England
Місткість: 52000
Види спорту: Football
```

```
Введіть назву стадіону для видалення: Olympic Stadium
Стадіон з назвою 'Olympic Stadium' успішно видалено.

Вміст після видалення елемента:
Назва: Stamford Bridge
Адреса: London, England
Місткість: 40000
Види спорту: Football

Назва: Camp Nou
Адреса: Barcelona, Spain
Місткість: 99000
Види спорту: Football

Назва: Old Trafford
Адреса: Manchester, England
Місткість: 74000
Види спорту: Football

Назва: Everton Stadium
Адреса: Liverpool, England
Місткість: 52000
Види спорту: Football
```

Код:

```cpp
#include <iostream>
#include <fstream>
#include <cstring>
#include <vector>

using namespace std;

struct Stadium {
    char name[50];
    char address[100];
    int capacity;
    char sport[50];
};

void createBinaryFile(const char* filename) {
    ofstream outFile(filename, ios::binary);

    if (!outFile) {
        cerr << "Не вдалося відкрити файл для запису!" << endl;
        return;
    }

    Stadium stadiums[] = {
        {"Olympic Stadium", "Kyiv, Ukraine", 70000, "Football"},
        {"Stamford Bridge", "London, England", 40000, "Football"},
        {"Camp Nou", "Barcelona, Spain", 99000, "Football"}
    };

    for (const auto& stadium : stadiums) {
        outFile.write(reinterpret_cast<const char*>(&stadium), sizeof(stadium));
    }

    outFile.close();
    cout << "Дані успішно записані в двійковий файл." << endl;
}

void printBinaryFile(const char* filename) {
    ifstream inFile(filename, ios::binary);

    if (!inFile) {
        cerr << "Не вдалося відкрити файл для читання!" << endl;
        return;
    }

    Stadium stadium;
    while (inFile.read(reinterpret_cast<char*>(&stadium), sizeof(stadium))) {
        cout << "Назва: " << stadium.name << endl;
        cout << "Адреса: " << stadium.address << endl;
        cout << "Місткість: " << stadium.capacity << endl;
        cout << "Види спорту: " << stadium.sport << endl << endl;
    }

    inFile.close();
}

void addStadiums(const char* filename, int index, const Stadium& newStadium1, const Stadium& newStadium2) {
    ifstream inFile(filename, ios::binary);
    if (!inFile) {
        cerr << "Не вдалося відкрити файл для читання!" << endl;
        return;
    }

    vector<Stadium> stadiums;
    Stadium stadium;

    while (inFile.read(reinterpret_cast<char*>(&stadium), sizeof(stadium))) {
        stadiums.push_back(stadium);
    }
    inFile.close();

    if (index < 0 || index >= stadiums.size()) {
        cerr << "Невірний індекс!" << endl;
        return;
    }

    stadiums.insert(stadiums.begin() + index + 1, newStadium2);
    stadiums.insert(stadiums.begin() + index + 1, newStadium1);

    ofstream outFile(filename, ios::binary);
    if (!outFile) {
        cerr << "Не вдалося відкрити файл для запису!" << endl;
        return;
    }

    for (const auto& s : stadiums) {
        outFile.write(reinterpret_cast<const char*>(&s), sizeof(s));
    }

    outFile.close();
```

```
    outFile.close();
    cout << "Два нових стадіони успішно додані." << endl;
}

void deleteStadiumByName(const char* filename, const char* stadiumName) {
    ifstream inFile(filename, ios::binary);

    if (!inFile) {
        cerr << "Не вдалося відкрити файл для читання!" << endl;
        return;
    }

    ofstream tempFile("temp.bin", ios::binary);

    Stadium stadium;
    bool found = false;

    while (inFile.read(reinterpret_cast<char*>(&stadium), sizeof(stadium))) {
        if (strcmp(stadium.name, stadiumName) != 0) {
            tempFile.write(reinterpret_cast<const char*>(&stadium), sizeof(stadium));
        } else {
            found = true;
        }
    }

    inFile.close();
    tempFile.close();

    if (!found) {
        cerr << "Стадіон з такою назвою не знайдений!" << endl;
        remove("temp.bin");
        return;
    }

    remove(filename);
    rename("temp.bin", filename);

    cout << "Стадіон з назвою '" << stadiumName << "' успішно видалено." << endl;
}

int main() {
    const char* filename = "stadiums.bin";

    createBinaryFile(filename);
```

```
    cout << "Вміст двійкового файлу:" << endl;
    printBinaryFile(filename);

    int index;
    cout << "Введіть індекс, після якого додати два нових стадіони: ";
    cin >> index;

    cin.ignore();
    Stadium newStadium1;
    cout << "Введіть назву першого нового стадіону: ";
    cin.getline(newStadium1.name, 50);
    cout << "Введіть адресу першого нового стадіону: ";
    cin.getline(newStadium1.address, 100);
    cout << "Введіть місткість першого нового стадіону: ";
    cin >> newStadium1.capacity;
    cin.ignore();
    cout << "Введіть види спорту для першого нового стадіону: ";
    cin.getline(newStadium1.sport, 50);

    Stadium newStadium2;
    cout << "Введіть назву другого нового стадіону: ";
    cin.getline(newStadium2.name, 50);
    cout << "Введіть адресу другого нового стадіону: ";
    cin.getline(newStadium2.address, 100);
    cout << "Введіть місткість другого нового стадіону: ";
    cin >> newStadium2.capacity;
    cin.ignore();
    cout << "Введіть види спорту для другого нового стадіону: ";
    cin.getline(newStadium2.sport, 50);

    addStadiums(filename, index, newStadium1, newStadium2);

    cout << "\nВміст після додавання двох нових елементів:" << endl;
    printBinaryFile(filename);

    char stadiumName[50];
    cout << "\nВведіть назву стадіону для видалення: ";
    cin.getline(stadiumName, 50);

    deleteStadiumByName(filename, stadiumName);

    cout << "\nВміст після видалення елемента:" << endl;
    printBinaryFile(filename);
```

Час виконання ~ 3 години

Завдання 3: VNS Lab 9 - Task 1-14

Створити текстовий файл F1 не менше, ніж з 10 рядків і записати в нього інформацію

Виконати завдання.

1) Скопіювати з файлу F1 у файл F2 всі рядки, що не містять букву «А» і розташовані між рядками з номерами N1 й N2.

2) Визначити номер того рядка, у якому найбільше голосних букв, файлу F2.

Код:

```cpp
void createFileF1(const string& filename) {
    ofstream file(filename);
    if (!file) {
        cout << "Не вдалося створити файл F1!\n";
        return;
    }
    cout << "Введіть рівно 10 рядків тексту:\n";

    string line;
    int count = 0;

    while (count < 10 && getline(cin, line)) {
        file << line << endl;
        count++;
    }

    file.close();
    cout << "Файл F1 успішно створено!\n";
}

void copyLines(const string& fileF1, const string& fileF2, int N1, int N2) {
    ifstream file1(fileF1);
    ofstream file2(fileF2);

    if (!file1 || !file2) {
        cout << "Не вдалося відкрити файли!\n";
        return;
    }

    string line;
    int lineNumber = 0;
    vector<string> copiedLines;

    while (getline(file1, line)) {
        lineNumber++;

        if (lineNumber >= N1 && lineNumber <= N2 && line.find('A') == string::npos && line.find('a') == string::npos) {
            file2 << line << endl;
            copiedLines.push_back(line);
        }
    }

    file1.close();
    file2.close();
    cout << "\nСкопійовані рядки:\n";
    for (const string& copiedLine : copiedLines) {
        cout << copiedLine << endl;
    }
}

int findLineWithMostVowels(const string& fileF2) {
    ifstream file(fileF2);
    if (!file) {
        cout << "Не вдалося відкрити файл F2!\n";
        return -1;
    }

    string line;
    int maxVowels = 0;
    int maxVowelLine = 0;
    int lineNumber = 0;

    while (getline(file, line)) {
        lineNumber++;
        int vowelCount = 0;

        for (char c : line) {
            if (tolower(c) == 'a' || tolower(c) == 'e' || tolower(c) == 'i' ||
                tolower(c) == 'o' || tolower(c) == 'u' || tolower(c) == 'y') {
                vowelCount++;
            }
        }

        if (vowelCount > maxVowels) {
            maxVowels = vowelCount;
            maxVowelLine = lineNumber;
        }
    }

    file.close();
    return maxVowelLine;
}

int main() {
    string fileF1 = "F1.txt";
    string fileF2 = "F2.txt";
```

```
    cout << "Створимо файл F1:\n";
    createFileF1(fileF1);

    int N1, N2;
    cout << "Введіть номер початкового рядка (N1): ";
    cin >> N1;
    cout << "Введіть номер кінцевого рядка (N2): ";
    cin >> N2;

    copyLines(fileF1, fileF2, N1, N2);

    int lineWithMostVowels = findLineWithMostVowels(fileF2);
    if (lineWithMostVowels != -1) {
        cout << "Номер рядка з найбільшою кількістю голосних у F2: " << lineWithMostVowels << endl;
    } else {
        cout << "Файл F2 порожній або не існує.\n";
    }

    return 0;
```

Вивід в терміналі:

```
Створимо файл F1:
Введіть рівно 10 рядків тексту:
1 line of text
second line of text
line of text number 3
line 4 of text
text line between 4 and 6
line of text after 5
7 lineeeeeeee of text
eighth line of text
nono line of text
final text line
Файл F1 успішно створено!
Введіть номер початкового рядка (N1): 3
Введіть номер кінцевого рядка (N2): 8

Скопійовані рядки:
line of text number 3
line 4 of text
7 lineeeeeeee of text
eighth line of text
Номер рядка з найбільшою кількістю голосних у F2: 3
```

Час виконання ~ 1.5 години

Завдання 4: Algotester Lab 4

Вам дано 2 цілих чисел масиви, розміром NN та MM.
Ваше завдання вивести:
1. Різницю N-M
2. Різницю M-N
3. Їх перетин
4. Їх обєднання
5. Їх симетричну різницю

Input

У першому рядку ціле число NN - розмір масиву 1
У другому рядку NN цілих чисел - елементи масиву 1
У третьому рядку ціле число MM - розмір масиву 2
У четвертом рядку MM цілих чисел - елементи масиву 2

Output

Вивести результат виконання 5 вищезазначених операцій у форматі:
У першому рядку ціле число NN - розмір множини
У наступному рядку NN цілих чисел - посортована у порядку зростання множина.

Код з використанням STL:

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
#include <iterator>
using namespace std;

void displayVector(const vector<int>& vec) {
    cout << vec.size() << endl;
    for (const int& element : vec) cout << element << " ";
    cout << endl;
}
int main() {
    int sizeA, sizeB;
    vector<int> groupA, groupB;
    vector<int> diffA_B, diffB_A, common, allElements, symDiff;

    cin >> sizeA;
    int value;
    for (int i = 0; i < sizeA; i++) {
        cin >> value;
        groupA.push_back(value);
    }

    cin >> sizeB;
    for (int i = 0; i < sizeB; i++) {
        cin >> value;
        groupB.push_back(value);
    }

    sort(groupA.begin(), groupA.end());
    sort(groupB.begin(), groupB.end());

    set_difference(groupA.begin(), groupA.end(), groupB.begin(), groupB.end(), back_inserter(diffA_B));
    set_difference(groupB.begin(), groupB.end(), groupA.begin(), groupA.end(), back_inserter(diffB_A));
    set_intersection(groupA.begin(), groupA.end(), groupB.begin(), groupB.end(), back_inserter(common));
    set_union(groupA.begin(), groupA.end(), groupB.begin(), groupB.end(), back_inserter(allElements));
    set_symmetric_difference(groupA.begin(), groupA.end(), groupB.begin(), groupB.end(), back_inserter(symDiff));

    displayVector(diffA_B);
    displayVector(diffB_A);
    displayVector(common);
    displayVector(allElements);
    displayVector(symDiff);

    return 0;
```

Час виконання ~ 30хв

```
5
1 2 3 4 5
5
4 5 6 7 8
3
1 2 3
3
6 7 8
2
4 5
8
1 2 3 4 5 6 7 8
6
1 2 3 6 7 8
```

Код без засобів STL:

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

void displayVector(const vector<int>& vec) {
    cout << vec.size() << endl;
    for (const int& element : vec) cout << element << " ";
    cout << endl;
}
vector<int> customDifference(const vector<int>& a, const vector<int>& b) {
    vector<int> result;
    size_t i = 0, j = 0;
    while (i < a.size() && j < b.size()) {
        if (a[i] < b[j]) {
            result.push_back(a[i]);
            i++;
        } else if (a[i] > b[j]) {
            j++;
        } else {
            i++;
            j++;
        }
    }
    while (i < a.size()) {
        result.push_back(a[i]);
        i++;
    }
    return result;
}
vector<int> customIntersection(const vector<int>& a, const vector<int>& b) {
    vector<int> result;
    size_t i = 0, j = 0;
    while (i < a.size() && j < b.size()) {
        if (a[i] < b[j]) {
            i++;
        } else if (a[i] > b[j]) {
            j++;
        } else {
            result.push_back(a[i]);
            i++;
            j++;
        }
    }
    return result;
```

```cpp
vector<int> customUnion(const vector<int>& a, const vector<int>& b) {
    vector<int> result;
    size_t i = 0, j = 0;
    while (i < a.size() && j < b.size()) {
        if (a[i] < b[j]) {
            result.push_back(a[i]);
            i++;
        } else if (a[i] > b[j]) {
            result.push_back(b[j]);
            j++;
        } else {
            result.push_back(a[i]);
            i++;
            j++;
        }
    }
    while (i < a.size()) {
        result.push_back(a[i]);
        i++;
    }
    while (j < b.size()) {
        result.push_back(b[j]);
        j++;
    }
    return result;
}
vector<int> customSymmetricDifference(const vector<int>& a, const vector<int>& b) {
    vector<int> result;
    size_t i = 0, j = 0;
    while (i < a.size() && j < b.size()) {
        if (a[i] < b[j]) {
            result.push_back(a[i]);
            i++;
        } else if (a[i] > b[j]) {
            result.push_back(b[j]);
            j++;
        } else {
            i++;
            j++;
        }
    }
    while (i < a.size()) {
        result.push_back(a[i]);
        i++;
```

```cpp
    }
    while (j < b.size()) {
        result.push_back(b[j]);
        j++;
    }
    return result;
}
int main() {
    int sizeA, sizeB;
    vector<int> groupA, groupB;
    cin >> sizeA;
    int value;
    for (int i = 0; i < sizeA; i++) {
        cin >> value;
        groupA.push_back(value);
    }

    cin >> sizeB;
    for (int i = 0; i < sizeB; i++) {
        cin >> value;
        groupB.push_back(value);
    }
    sort(groupA.begin(), groupA.end());
    sort(groupB.begin(), groupB.end());

    vector<int> diffA_B = customDifference(groupA, groupB);
    vector<int> diffB_A = customDifference(groupB, groupA);
    vector<int> common = customIntersection(groupA, groupB);
    vector<int> allElements = customUnion(groupA, groupB);
    vector<int> symDiff = customSymmetricDifference(groupA, groupB);
    displayVector(diffA_B);
    displayVector(diffB_A);
    displayVector(common);
    displayVector(allElements);
    displayVector(symDiff);

    return 0;
}
```

Вивід в терміналі:

```
5
1 2 3 4 5
5
4 5 6 7 8
3
1 2 3
3
6 7 8
2
4 5
8
1 2 3 4 5 6 7 8
6
1 2 3 6 7 8
```

Час виконання ~ 1.5 години

Завдання 5: Algotester Lab 6

Вам дано NN слів та число KK.

Ваше завдання перечислити букви в словах, які зустрічаються в тексті більше-рівне ніж KK разів (саме слово, не буква!).

Великі та маленькі букви вважаються однаковими, виводити необхідно малі, посортовані від останьої до першої у алфавіті. Букву потрібно виводити лише один раз.

У випадку якщо таких букв немає - вивести "Empty!".

Input

Цілі числа NN та KK - загальна кількість слів та мінімальна кількість слів щоб враховувати букви цього слова в результаті.

NN стрічок ss

Output

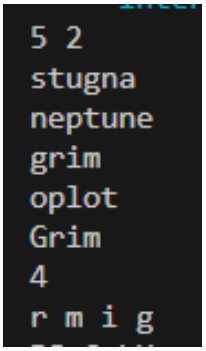
У першому рядку ціле число MM - кількість унікальних букв

У другому рядку унікальні букви через пробіли

Код:

```cpp
#include <iostream>
#include <vector>
#include <string>
#include <map>
#include <set>
#include <algorithm>

using namespace std;

int main() {
    int N, K;
    cin >> N >> K;

    vector<string> words(N);
    map<string, int> wordCount;

    for (int i = 0; i < N; ++i) {
        cin >> words[i];
        transform(words[i].begin(), words[i].end(), words[i].begin(), ::tolower);
        wordCount[words[i]]++;
    }

    set<char, greater<char>> result;

    for (const auto& pair : wordCount) {
        if (pair.second >= K) {
            for (char ch : pair.first) {
                result.insert(ch);
            }
        }
    }

    if (result.empty()) {
        cout << "Empty!" << endl;
    } else {
        cout << result.size() << endl;
        for (char ch : result) {
            cout << ch << " ";
        }
        cout << endl;
    }

    return 0;
}
```

Вивід в терміналі:



Час виконання ~ 50хв

Висновок:

У цьому епіку я навчився працювати з файлами, розібрався як працюють вектори і списки а також ознайомився із бібліотеками.