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:**

Практичних Робіт до блоку № 5

**Виконав:**

Студент групи ШІ-11

Голейчук Іван Миколайович

Львів 2024

**Тема роботи**:  
"Основи роботи з файлами та рядковими змінними у програмуванні на C++"

**Мета роботи**:  
Розробити системне розуміння принципів роботи з файлами та рядковими змінними у мовi C++. Дослідити методи взаємодії з текстовими та бінарними файлами, а також функціональність стандартної бібліотеки для обробки файлів. Ознайомитися зі створенням власних бібліотек, організацією коду, та вивчити практичні прийоми парсингу, серіалізації даних і обробки помилок.

**Теоретичні відомості:**

Вступ до Роботи з Файлами:   
Основні операції з файлами: відкриття, читання, запис, закриття   
Робота з файловими дескрипторами   
C-style читання з файлу та запис до файлу   
Перевірка стану файлу: перевірка помилок, кінець файлу   
Базові приклади читання та запису в файл   
Символи і Рядкові Змінні:   
Робота з char та string: основні операції і методи   
Стрічкові літерали та екранування символів   
Конкатенація, порівняння та пошук у рядках   
Текстові Файли:   
Особливості читання та запису текстових файлів   
Обробка рядків з файлу: getline, ignore, peek   
Форматування тексту при записі: setw, setfill, setprecision   
Парсинг текстових файлів: розділення на слова, аналіз структури   
Обробка помилок при роботі з файлами   
Бінарні Файли:   
Вступ до бінарних файлів: відмінності від текстових, приклади (великі дані, ігрові ресурси, зображення)   
Читання та запис бінарних даних   
Робота з позиціонуванням у файлі: seekg, seekp   
Серіалізація об'єктів у бінарний формат   
Стандартна бібліотека та робота з файлами:   
Огляд стандартної бібліотеки для роботи з файлами   
Потоки вводу/виводу: ifstream, ofstream, fstream   
Обробка помилок при роботі з файлами   
Створення й використання бібліотек:   
Вступ до створення власних бібліотек у С++   
Правила розбиття коду на header-и(.h) та source(.cpp) файли   
Статичні проти динамічних бібліотек: переваги та використання   
Інтерфейси бібліотек: створення, документування, версіонування   
Використання сторонніх бібліотек у проектах

**Індивідуальний план опрацювання теорії:**

**Вступ до Роботи з Файлами:**

**Опрацьовано:**  
Розглянуто основні операції роботи з файлами: відкриття, читання, запис, закриття. Вивчено принципи роботи з файловими дескрипторами та виконання C-style операцій з файлами. Проведено аналіз перевірки стану файлів (помилки, кінець файлу) з демонстрацією базових прикладів.  
**Джерела інформації:**

* Лекції Олександра Пшеничного
* Практичні заняття
* Використання штучного інтелекту (чат GPT)
* YouTube

**Символи і Рядкові Змінні:**

**Опрацьовано:**  
Вивчено основні операції з типами даних char і string, включаючи роботу зі стрічковими літералами, екрануванням символів, конкатенацією, порівнянням та пошуком у рядках. Розглянуто приклади ефективної роботи з рядковими змінними.  
**Джерела інформації:**

* Лекції Олександра Пшеничного
* Практичні заняття
* Використання штучного інтелекту (чат GPT)
* YouTube

**Текстові Файли:**

**Опрацьовано:**  
Досліджено специфіку роботи з текстовими файлами: методи читання (getline, ignore, peek) і запису даних, використання функцій форматування тексту (setw, setfill, setprecision). Вивчено техніки парсингу файлів для аналізу їхньої структури та обробки рядків. Здійснено практику перевірки помилок при роботі з файлами.  
**Джерела інформації:**

* Лекції Олександра Пшеничного
* Практичні заняття
* Використання штучного інтелекту (чат GPT)
* YouTube

**Бінарні Файли:**

**Опрацьовано:**  
Вивчено основи роботи з бінарними файлами, їх відмінності від текстових. Ознайомлено з методами читання, запису, серіалізації даних і позиціонування у файлі (seekg, seekp). Наведено приклади використання бінарних файлів для збереження великих даних.  
**Джерела інформації:**

* Лекції Олександра Пшеничного
* Практичні заняття
* Використання штучного інтелекту (чат GPT)
* YouTube

**Стандартна бібліотека та робота з файлами:**

**Опрацьовано:**  
Досліджено можливості стандартної бібліотеки для роботи з файлами: використання потоків (ifstream, ofstream, fstream), обробка помилок, а також інтеграція сторонніх бібліотек у проєкти.  
**Джерела інформації:**

* Лекції Олександра Пшеничного
* Практичні заняття
* Використання штучного інтелекту (чат GPT)
* YouTube

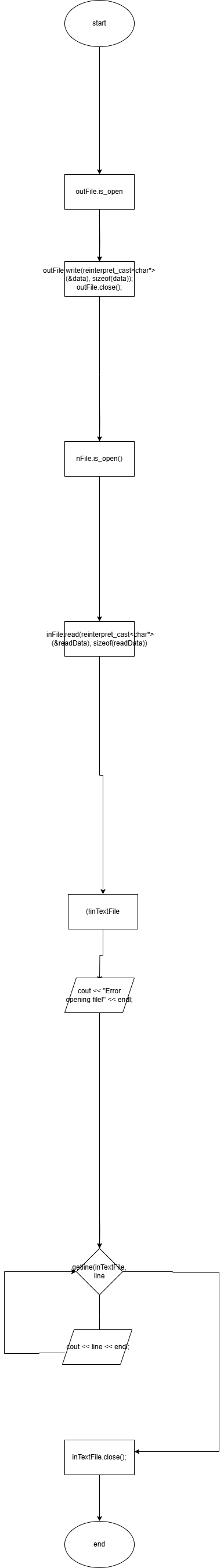
**Створення й використання бібліотек:**

**Опрацьовано:**  
Розглянуто основи створення власних бібліотек у C++, правила розподілу коду між файлами (header і source). Вивчено переваги статичних і динамічних бібліотек, принципи документування та версіонування інтерфейсів, інтеграцію бібліотек у проєкти.  
**Джерела інформації:**

* Лекції Олександра Пшеничного
* Практичні заняття
* Використання штучного інтелекту (чат GPT)
* YouTube

**Виконання роботи:**

**Task 2 - Requirements management (understand tasks) and design activities (draw flow diagrams and estimate tasks 3-9)**



Epic 5 Task 3 - Lab# programming: VNS Lab 6

#include <iostream>

#include <cstring>

#include <climits>

using namespace std;

int main() {

    char s[256];

    cout << "Enter the text: ";

    cin.getline(s, 256);

    int minlen = INT\_MAX, curlen = 0;

    for (int i = 0; s[i] != '\0'; i++) {

        if (s[i] != ' ' && s[i] != '.') {

            curlen++;

        }

        else if (curlen > 0) {

            if (curlen < minlen) minlen = curlen;

            curlen = 0;

        }

    }

    if (minlen == INT\_MAX) {

        cout << "There are no words in the text." << endl;

    }

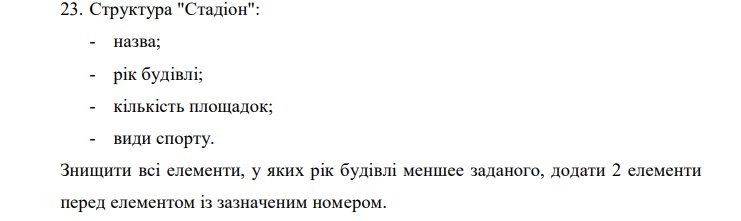
    else {

        cout << "The length of the shortest word is: " << minlen << endl;

    }

    return 0;

}

Epic 5 Task 4 - Lab# programming: VNS Lab 8

#include <iostream>

#include <fstream>

#include <vector>

#include <string>

using namespace std;

struct stadium {

    string name;

    int year;

    int fields;

    string sports;

};

void create(const string& fname, const vector<stadium>& data) {

    ofstream f(fname, ios::binary);

    if (!f) {

        cerr << "File opening error" << endl;

        return;

    }

    for (const auto& s : data) {

        f.write((char\*)&s, sizeof(stadium));

    }

    f.close();

}

void print(const string& fname) {

    ifstream f(fname, ios::binary);

    if (!f) {

        cerr << "File opening error" << endl;

        return;

    }

    stadium s;

    while (f.read((char\*)&s, sizeof(stadium))) {

        cout << "Name: " << s.name << ", Year: " << s.year

            << ", Fields: " << s.fields

            << ", Sports: " << s.sports << endl;

    }

    f.close();

}

void del(const string& fname, int y) {

    ifstream f(fname, ios::binary);

    if (!f) {

        cerr << "File opening error" << endl;

        return;

    }

    vector<stadium> data;

    stadium s;

    while (f.read((char\*)&s, sizeof(stadium))) {

        if (s.year >= y) {

            data.push\_back(s);

        }

    }

    f.close();

    ofstream out(fname, ios::binary | ios::trunc);

    if (!out) {

        cerr << "File opening error" << endl;

        return;

    }

    for (const auto& item : data) {

        out.write((char\*)&item, sizeof(stadium));

    }

    out.close();

}

void add(const string& fname, const stadium& e1, const stadium& e2, int pos) {

    ifstream f(fname, ios::binary);

    if (!f) {

        cerr << "File opening error" << endl;

        return;

    }

    vector<stadium> data;

    stadium s;

    while (f.read((char\*)&s, sizeof(stadium))) {

        data.push\_back(s);

    }

    f.close();

    if (pos < 0 || pos > data.size()) {

        cerr << "Invalid position" << endl;

        return;

    }

    data.insert(data.begin() + pos, { e1, e2 });

    ofstream out(fname, ios::binary | ios::trunc);

    if (!out) {

        cerr << "File opening error" << endl;

        return;

    }

    for (const auto& item : data) {

        out.write((char\*)&item, sizeof(stadium));

    }

    out.close();

}

int main() {

    string fname = "stadiums.dat";

    vector<stadium> data = {

        {"Olympic", 1990, 5, "Football, Basketball"},

        {"Lviv Arena", 2012, 3, "Football"},

        {"Dnipro Arena", 1985, 4, "Football, Volleyball"}

    };

    create(fname, data);

    cout << "File content after creation:" << endl;

    print(fname);

    int y = 2000;

    del(fname, y);

    cout << "\nFile content after deletion:" << endl;

    print(fname);

    stadium e1 = { "New Stadium 1", 2023, 6, "Football, Tennis" };

    stadium e2 = { "New Stadium 2", 2024, 8, "Basketball, Hockey" };

    int pos = 1;

    add(fname, e1, e2, pos);

    cout << "\nFile content after addition:" << endl;

    print(fname);

    return 0;

}

John Black - Epic 5 Task 5 - Lab# programming: VNS Lab 9

#include <iostream>

#include <fstream>

#include <sstream>

#include <vector>

#include <string>

#include <map>

using namespace std;

bool has\_dup(const string& s) {

    istringstream iss(s);

    map<string, int> wc;

    string w;

    while (iss >> w) {

        wc[w]++;

        if (wc[w] > 1) return true;

    }

    return false;

}

int max\_a(const string& s) {

    istringstream iss(s);

    string w;

    int max\_a = 0, pos = -1, cur = 0;

    while (iss >> w) {

        cur++;

        int cnt = 0;

        for (char c : w) {

            if (tolower(c) == 'a') cnt++;

        }

        if (cnt > max\_a) {

            max\_a = cnt;

            pos = cur;

        }

    }

    return pos;

}

int main() {

    string f1 = "F1.txt", f2 = "F2.txt";

    ofstream out(f1);

    if (!out) {

        cerr << "Error opening F1" << endl;

        return 1;

    }

    out << "This is a test line with two test words.\n";

    out << "Another line for verification.\n";

    out << "This line has no duplicate words.\n";

    out << "Here is a line with many words, many words here.\n";

    out << "This line contains the word aaaa many times.\n";

    out << "Just a meaningless line.\n";

    out << "This is another line with text.\n";

    out << "This line contains words words words.\n";

    out << "A single line.\n";

    out << "A line for testing word verification.\n";

    out.close();

    ifstream in(f1);

    ofstream out2(f2);

    if (!in || !out2) {

        cerr << "File error" << endl;

        return 1;

    }

    string line;

    while (getline(in, line)) {

        if (has\_dup(line)) {

            out2 << line << endl;

        }

    }

    in.close();

    out2.close();

    in.open(f1);

    if (!in) {

        cerr << "Error opening F1" << endl;

        return 1;

    }

    int max\_pos = -1, line\_num = 0, cur\_line = 0;

    while (getline(in, line)) {

        cur\_line++;

        int pos = max\_a(line);

        if (pos > max\_pos) {

            max\_pos = pos;

            line\_num = cur\_line;

        }

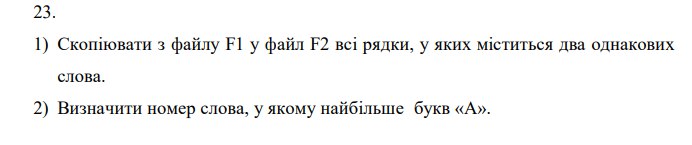
    }

    in.close();

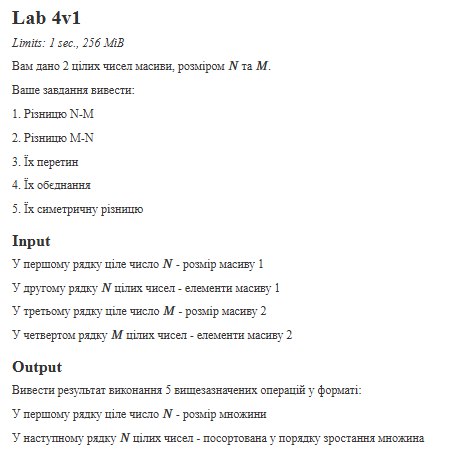
    cout << "Line number with the word having the most 'A's: " << line\_num << endl;

    return 0;

}



Epic 5 Task 6 - Lab# programming: Algotester Lab 4



#include <iostream>

#include <set>

#include <vector>

#include <algorithm>

using namespace std;

int main() {

    int N, M;

    cin >> N;

    vector<int> a(N);

    for (int i = 0; i < N; i++) {

        cin >> a[i];

    }

    cin >> M;

    vector<int> b(M);

    for (int i = 0; i < M; i++) {

        cin >> b[i];

    }

    sort(a.begin(), a.end());

    sort(b.begin(), b.end());

    vector<int> diff;

    set\_difference(a.begin(), a.end(), b.begin(), b.end(), back\_inserter(diff));

    cout << diff.size() << endl;

    for (size\_t i = 0; i < diff.size(); i++) {

        if (i > 0) cout << " ";

        cout << diff[i];

    }

    cout << endl;

    vector<int> revDiff;

    set\_difference(b.begin(), b.end(), a.begin(), a.end(), back\_inserter(revDiff));

    cout << revDiff.size() << endl;

    for (size\_t i = 0; i < revDiff.size(); i++) {

        if (i > 0) cout << " ";

        cout << revDiff[i];

    }

    cout << endl;

    vector<int> intersec;

    set\_intersection(a.begin(), a.end(), b.begin(), b.end(), back\_inserter(intersec));

    cout << intersec.size() << endl;

    for (size\_t i = 0; i < intersec.size(); i++) {

        if (i > 0) cout << " ";

        cout << intersec[i];

    }

    cout << endl;

    vector<int> unio;

    set\_union(a.begin(), a.end(), b.begin(), b.end(), back\_inserter(unio));

    cout << unio.size() << endl;

    for (size\_t i = 0; i < unio.size(); i++) {

        if (i > 0) cout << " ";

        cout << unio[i];

    }

    cout << endl;

    vector<int> symDiff;

    set\_symmetric\_difference(a.begin(), a.end(), b.begin(), b.end(), back\_inserter(symDiff));

    cout << symDiff.size() << endl;

    for (size\_t i = 0; i < symDiff.size(); i++) {

        if (i > 0) cout << " ";

        cout << symDiff[i];

    }

    cout << endl;

    return 0;

}

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

void print(const vector<int>& res) {

    cout << res.size() << endl;

    for (int x : res) cout << x << " ";

    cout << endl;

}

vector<int> un(const vector<int>& a, const vector<int>& b) {

    vector<int> res;

    int i = 0, j = 0;

    while (i < a.size() && j < b.size()) {

        if (a[i] < b[j]) res.push\_back(a[i++]);

        else if (b[j] < a[i]) res.push\_back(b[j++]);

        else {

            res.push\_back(a[i]);

            i++; j++;

        }

    }

    while (i < a.size()) res.push\_back(a[i++]);

    while (j < b.size()) res.push\_back(b[j++]);

    return res;

}

vector<int> inter(const vector<int>& a, const vector<int>& b) {

    vector<int> res;

    int i = 0, j = 0;

    while (i < a.size() && j < b.size()) {

        if (a[i] < b[j]) i++;

        else if (b[j] < a[i]) j++;

        else {

            res.push\_back(a[i]);

            i++; j++;

        }

    }

    return res;

}

vector<int> diff(const vector<int>& a, const vector<int>& b) {

    vector<int> res;

    int i = 0, j = 0;

    while (i < a.size() && j < b.size()) {

        if (a[i] < b[j]) res.push\_back(a[i++]);

        else if (b[j] < a[i]) j++;

        else {

            i++;

            j++;

        }

    }

    while (i < a.size()) res.push\_back(a[i++]);

    return res;

}

vector<int> symDiff(const vector<int>& a, const vector<int>& b) {

    vector<int> res1 = diff(a, b);

    vector<int> res2 = diff(b, a);

    return un(res1, res2);

}

int main() {

    int N, M;

    cin >> N;

    vector<int> n(N);

    for (int i = 0; i < N; i++) {

        cin >> n[i];

    }

    cin >> M;

    vector<int> m(M);

    for (int i = 0; i < M; i++) {

        cin >> m[i];

    }

    sort(n.begin(), n.end());

    sort(m.begin(), m.end());

    vector<int> d1 = diff(n, m);

    vector<int> d2 = diff(m, n);

    vector<int> interRes = inter(n, m);

    vector<int> unRes = un(n, m);

    vector<int> symRes = symDiff(n, m);

    print(d1);

    print(d2);

    print(interRes);

    print(unRes);

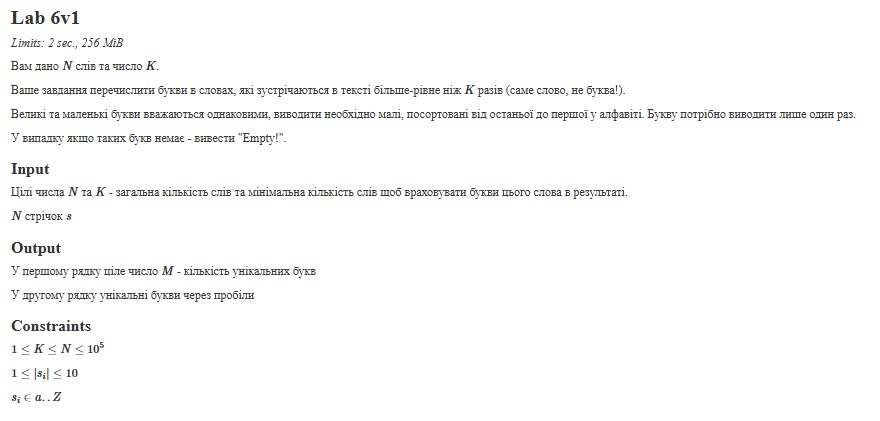
    print(symRes);

    return 0;

}

```

Epic 5 Task 7 - Lab# programming: Algotester Lab 6



#include <iostream>

#include <string>

#include <unordered\_map>

#include <set>

#include <vector>

#include <algorithm>

using namespace std;

void toLower(string& str) {

    for (char& c : str) {

        c = tolower(c);

    }

}

int main() {

    int N, K;

    cin >> N >> K;

    unordered\_map<string, int> freq;

    vector<string> words(N);

    for (int i = 0; i < N; i++) {

        cin >> words[i];

        toLower(words[i]);

        freq[words[i]]++;

    }

    set<char> letters;

    for (const auto& [word, count] : freq) {

        if (count >= K) {

            for (char c : word) {

                letters.insert(c);

            }

        }

    }

    if (letters.empty()) {

        cout << "Empty!" << endl;

        return 0;

    }

    vector<char> sortedLetters(letters.begin(), letters.end());

    sort(sortedLetters.rbegin(), sortedLetters.rend());

    cout << sortedLetters.size() << endl;

    for (size\_t i = 0; i < sortedLetters.size(); i++) {

        cout << sortedLetters[i];

        if (i != sortedLetters.size() - 1) {

            cout << " ";

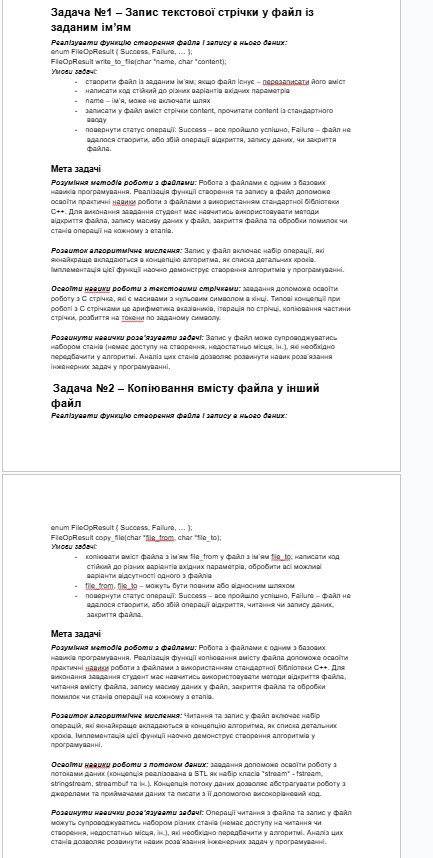
        }

    }

    cout << endl;

    return 0;

}

Epic 5 Task 8 - Practice# programming: Class Practice Tasк

#include <iostream>

#include <fstream>

#include <cstring>

using namespace std;

enum FileOpResult { Success, Failure };

FileOpResult write\_to\_file(const char\* name, const char\* content) {

    ofstream file(name);

    if (!file.is\_open()) {

        return Failure;

    }

    file << content;

    if (file.fail()) {

        return Failure;

    }

    file.close();

    if (file.fail()) {

        return Failure;

    }

    return Success;

}

int main() {

    char filename[100], content[500];

    cout << "Enter the filename: ";

    cin.getline(filename, 100);

    cout << "Enter the content to write to the file: ";

    cin.getline(content, 500);

    FileOpResult result = write\_to\_file(filename, content);

    if (result == Success) {

        cout << "File written successfully!" << endl;

    }

    else {

        cout << "Error writing file!" << endl;

    }

    return 0;

}

#include <iostream>

#include <fstream>

using namespace std;

enum FileOpResult { Success, Failure };

FileOpResult copy\_file(const char\* file\_from, const char\* file\_to) {

    ifstream src(file\_from, ios::binary);

    if (!src.is\_open()) {

        return Failure;

    }

    ofstream dest(file\_to, ios::binary);

    if (!dest.is\_open()) {

        return Failure;

    }

    dest << src.rdbuf();

    if (src.fail() || dest.fail()) {

        return Failure;

    }

    src.close();

    dest.close();

    if (src.fail() || dest.fail()) {

        return Failure;

    }

    return Success;

}

int main() {

    char file\_from[100], file\_to[100];

    cout << "Enter the source filename: ";

    cin.getline(file\_from, 100);

    cout << "Enter the destination filename: ";

    cin.getline(file\_to, 100);

    FileOpResult result = copy\_file(file\_from, file\_to);

    if (result == Success) {

        cout << "File copied successfully!" << endl;

    }

    else {

        cout << "Error copying file!" << endl;

    }

    return 0;

}

Epic 5 Task 9 - Practice# programming: Self Practice Task

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main() {

    int data = 123456;

    ofstream outFile("binary.dat", ios::binary);

    if (outFile.is\_open()) {

        outFile.write(reinterpret\_cast<char\*>(&data), sizeof(data));

        outFile.close();

    }

    int readData;

    ifstream inFile("binary.dat", ios::binary);

    if (inFile.is\_open()) {

        inFile.read(reinterpret\_cast<char\*>(&readData), sizeof(readData));

        cout << readData << endl;

        inFile.close();

    }

    ifstream inTextFile("example.txt");

    if (!inTextFile) {

        cout << "Error opening file!" << endl;

        return 1;

    }

    string line;

    while (getline(inTextFile, line)) {

        cout << line << endl;

    }

    inTextFile.close();

    return 0;

}

Робота у команді:



Ми зібрались одинраз та обговрили всі деталі епіку. Домовились що якщо в когось будуть питання то зберемось щераз.

**Висновок:** У результаті роботи вивчено основи роботи з файлами та рядковими змінними у C++. Розглянуто базові операції з файлами (відкриття, читання, запис, закриття), роботу з файловими дескрипторами та перевірку стану файлів. Досліджено обробку текстових і бінарних файлів, включаючи форматування, парсинг і серіалізацію даних.

Проаналізовано методи роботи з рядковими змінними (char, string), їх об’єднання, порівняння та пошук. Вивчено використання потоків вводу/виводу (ifstream, ofstream, fstream) і основи створення власних бібліотек у C++. Розглянуто обробку помилок для забезпечення стабільності програм.

Ці знання створюють базу для вирішення складних програмних завдань.