Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту

A blue and white logo

Description automatically generated

**Звіт**

**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-12

Михальчук Антон Євгенійович

# **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:**

# Дослідити основи роботи з файлами у мові програмування C++, зокрема розглянути та освоїти принципи роботи з текстовими та бінарними файлами. Опанувати операції введення та виведення символів і рядкових змінних у файл, а також ознайомитися зі стандартною бібліотекою C++ для роботи з файлами. Навчитися створювати власні бібліотеки та використовувати їх у проектах, організовуючи код для повторного використання та покращення його структури.

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №\*.1: С++ Arrays, Data structures

1. Індивідуальний план опрацювання теорії:

* Тема №\*.1: С++ Basics
  + Джерела Інформації
    - Відео. <https://www.youtube.com/watch?v=2UDMGCcRCjo>
    - Стаття. <https://www.w3schools.com/cpp/>
  + Що опрацьовано:
    - Вивчив базовий синтаксис та семантику мови C++.
    - Особливу увагу приділяв таким темам, як стрічки та файли.
  + Статус: Ознайомлений
  + Початок опрацювання теми: 15.09.2024
  + Звершення опрацювання теми: 12.11.2024

# **Виконання роботи:**

## **1. Опрацювання завдання та вимог до програм та середовища:**

Завдання №1 VNS Lab 4 Варіант: 9

* Деталі завдання:

Задано рядок, що складається із символів. Символи поєднуються в слова.

Слова одне від одного відокремлюються одним або декількома пробілами.

Наприкінці тексту ставиться крапка.

Надрукувати всі слова-паліндроми, які є в цьому рядку

Завдання №2 VNS Lab 8 Варіант: 9

* Деталі завдання:

Структура "Пацієнт":

- прізвище, ім'я, по батькові;

- домашня адреса;

- номер медичної карти;

- номер страхового поліса.

Знищити елемент із заданим номером медичної карти, додати 2 елементи в

початок файлу.

Завдання №3 VNS Lab 9 Варіант: 9

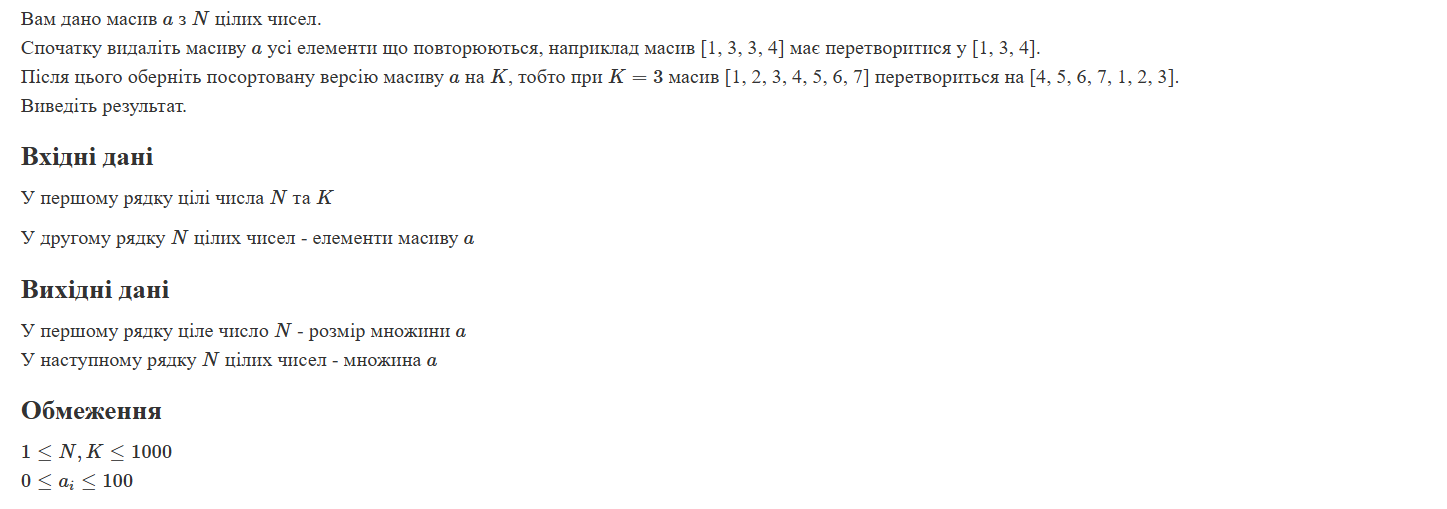
* Деталі завдання:

1) Скопіювати з файлу F1 у файл F2 всі рядки, які містять тільки одне слово.

2) Знайти найдовше слово у файлі F2.

Завдання №4 Algotester Lab 4 Варіант: 2

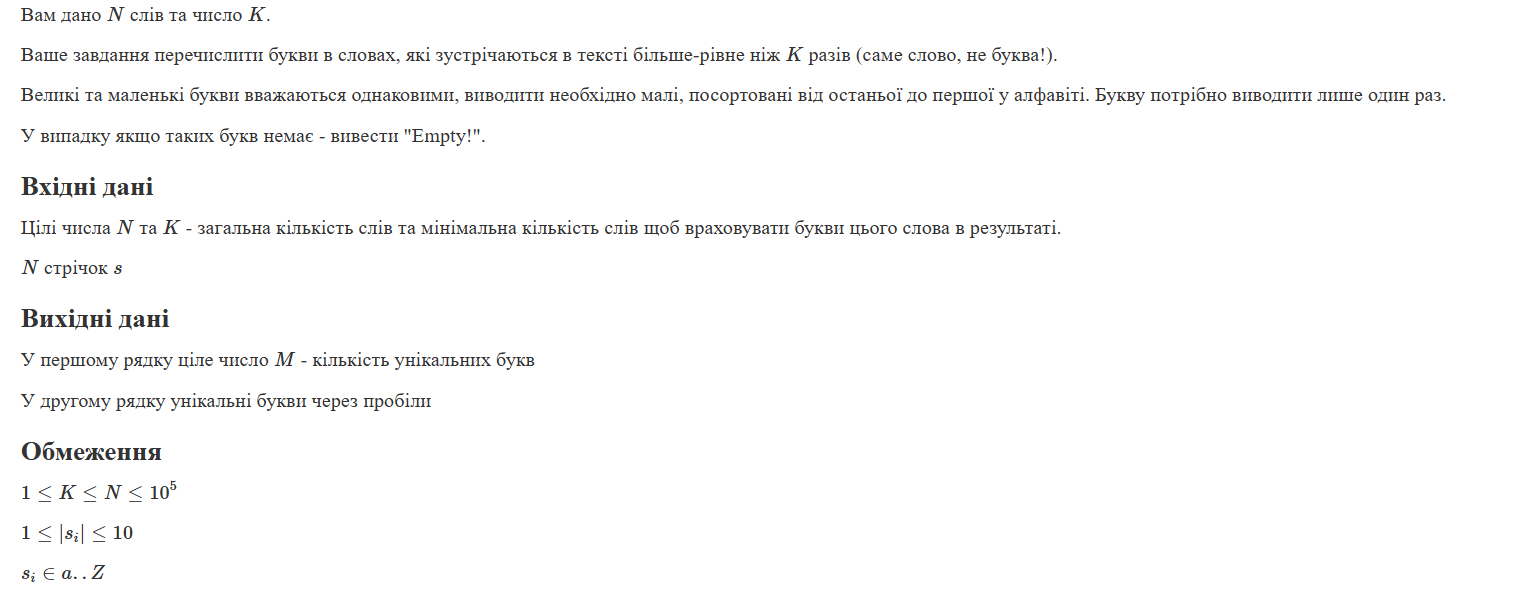
* Деталі завдання:



**Пам’ятайте, ви маєте написати 2 варіанти розвязку, один з використанням засобів STL (std::unique, std::sort, std::rotate), інший зі своєю реалізацією.**

Завдання №5 Algotester Lab 6 Варіант: 1

* Деталі завдання:



Завдання №5 Class Practice Task

* Деталі завдання:

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.

enum FileOpResult { Success, Failure, … };

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

Умови задачі:

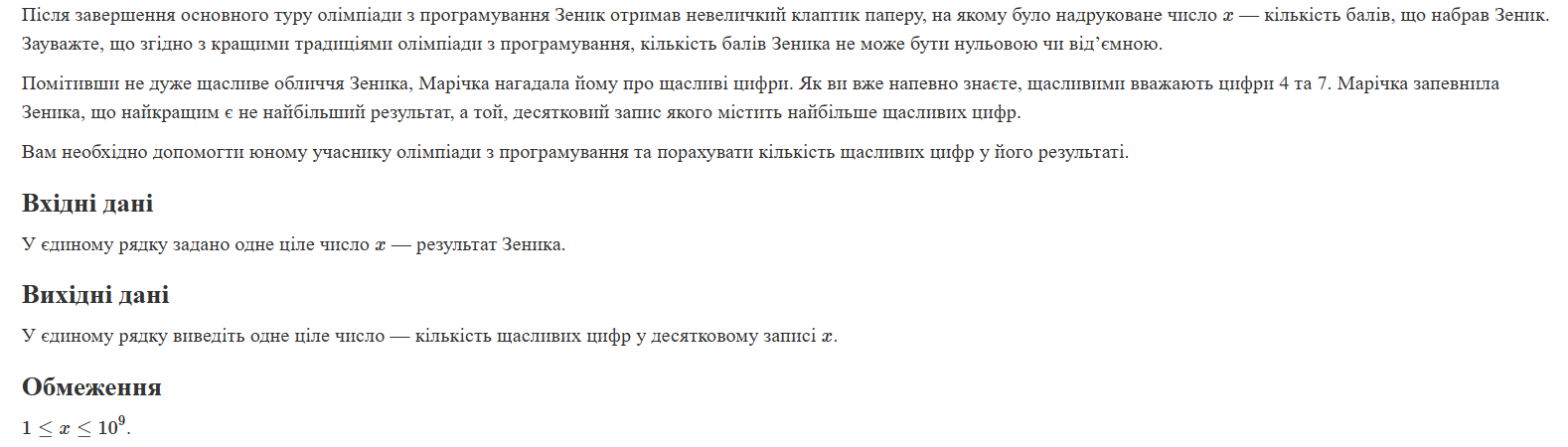
- копіювати вміст файла з ім’ям file\_from у файл з ім’ям file\_to; написати код стійкий до різних варіантів вхідних параметрів, обробити всі можливі варіанти відсутності одного з файлів

- 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.

Завдання №6 Self Practice Task Щасливий результат

* Деталі завдання:



## **2. Дизайн та планована оцінка часу виконання завдань:**

Програма №1 VNS Lab 6 Варіант: 9

* Планований час на реалізацію: 1 год.

Програма №2 VNS Lab 8 Варіант: 9

* Планований час на реалізацію: 1 год.

Програма №3 VNS Lab 8 Варіант: 9

* Планований час на реалізацію: 1 год.

Програма №4 Algotester Lab 3 Варіант: 2

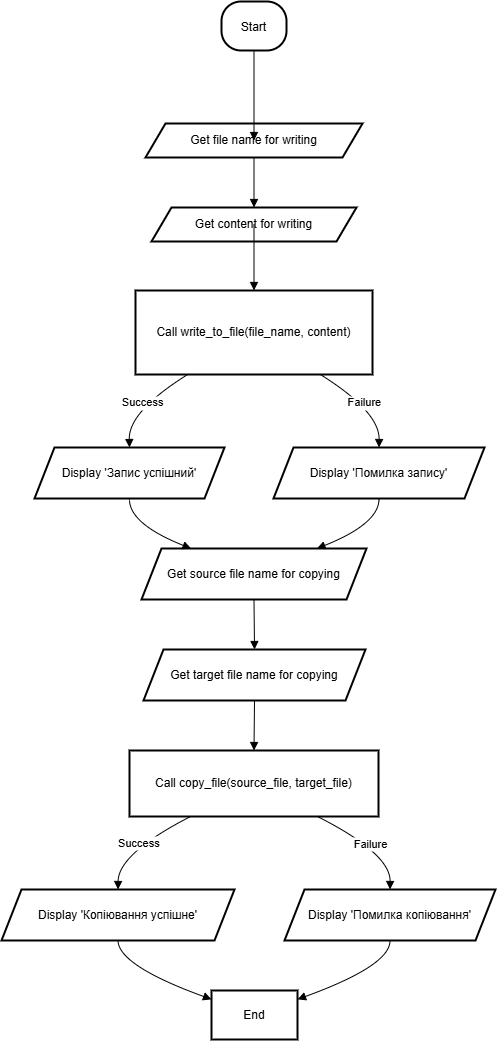
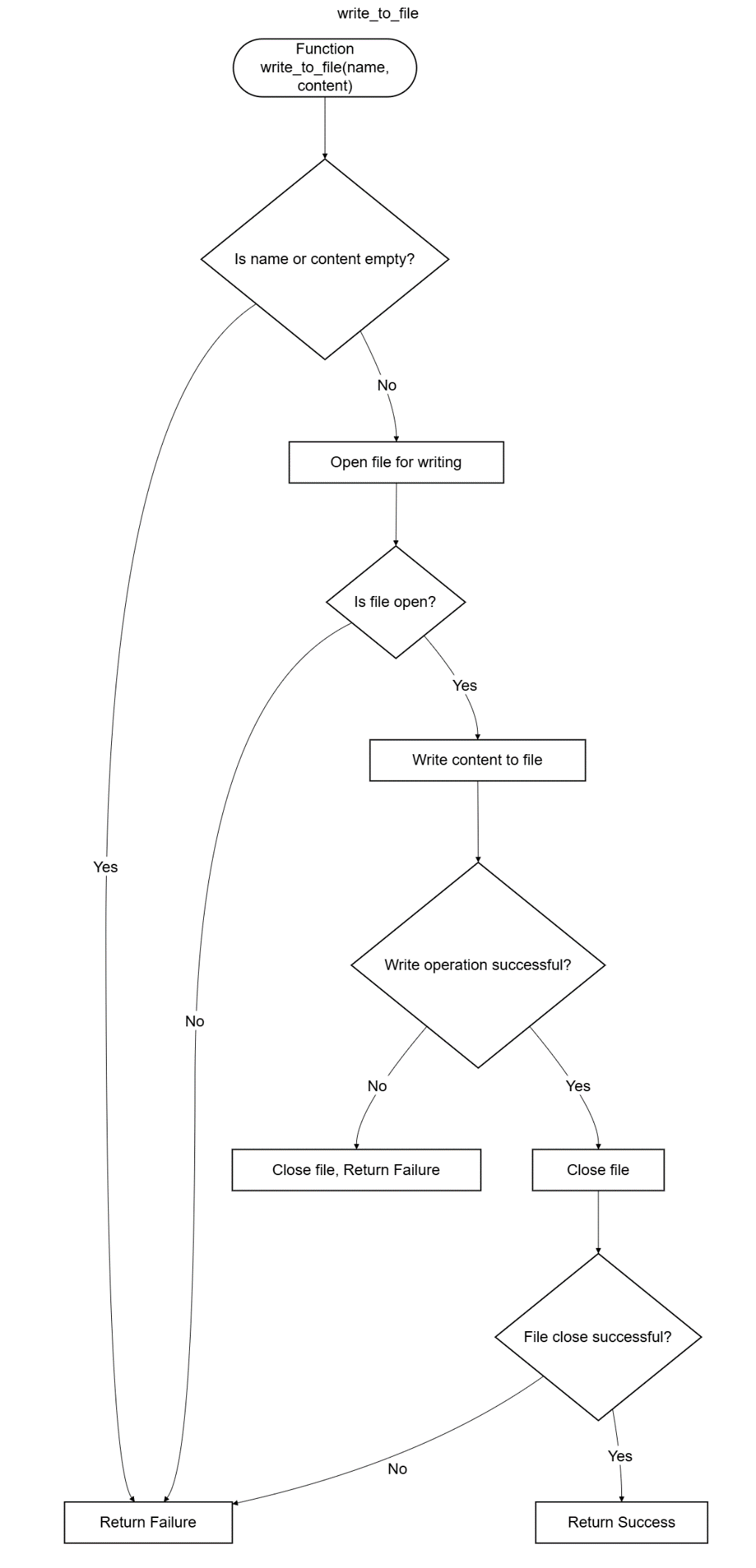
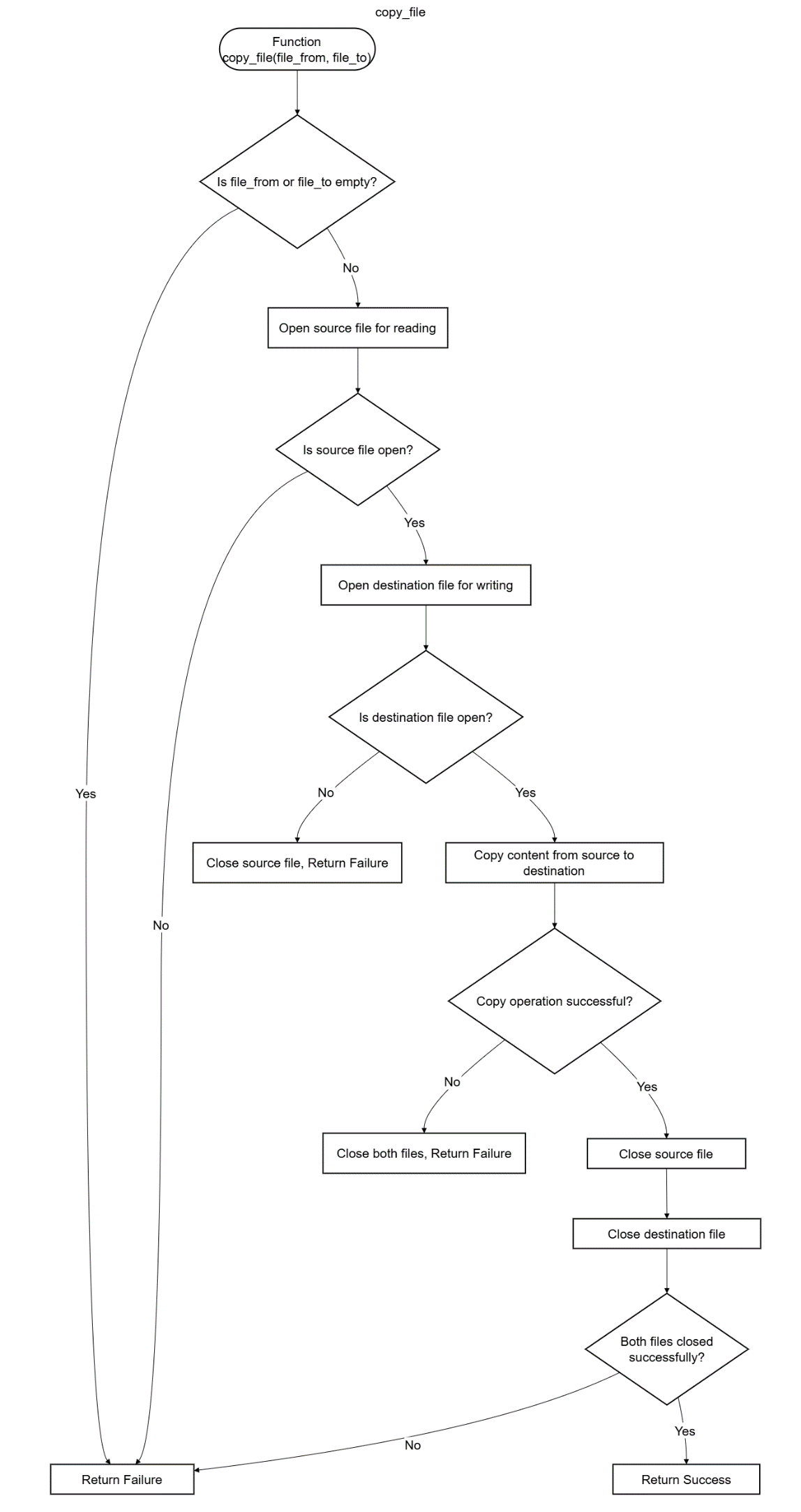
* Планований час на реалізацію: 30 хв.

Програма №5 VNS Algotester Lab 6 Варіант: 1

* Планований час на реалізацію: 50 хв.

Програма №5 Class Practice Task

* Блок-схема



* Планований час на реалізацію: 90 хв.

Програма №6 Self Practice Task Щасливий результат

* Планований час на реалізацію: 10 хв

## **4. Код програм з посиланням на зовнішні ресурси:**

Завдання №1

<https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/271/files>

#include <iostream>

#include <string>

#include <vector>

#include <sstream>

using namespace std;

bool isPalindrome(const string &str, int start, int end)

{

    if (start >= end)

    {

        return true;

    }

    if (str[start] != str[end])

    {

        return false;

    }

    return isPalindrome(str, start + 1, end - 1);

}

int main()

{

    string input;

    cout << "Enter a sentence: ";

    getline(cin, input);

    if (!input.empty() && input.back() == '.')

    {

        input.pop\_back();

    }

    istringstream iss(input);

    vector<string> words;

    string word;

    while (iss >> word)

    {

        words.push\_back(word);

    }

    cout << "Palindromes: ";

    for (int i = 0; i < words.size(); i++)

    {

        if (isPalindrome(words[i], 0, words[i].length() - 1))

        {

            cout << words[i] << " ";

        }

    }

    return 0;

}

Завдання №2

<https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/271/files>

#include <iostream>

#include <fstream>

#include <vector>

#include <string>

using namespace std;

struct Patient

{

    char surname[50];

    char name[50];

    char patronymic[50];

    char address[100];

    int medicalCardNumber;

    int insurancePolicyNumber;

};

void createFile(const string &filename, const vector<Patient> &patients)

{

    ofstream file(filename, ios::binary);

    if (!file)

    {

        cerr << "Error opening file for writing!" << endl;

        return;

    }

    for (const auto &patient : patients)

    {

        file.write(reinterpret\_cast<const char \*>(&patient), sizeof(Patient));

    }

    file.close();

}

void printFile(const string &filename)

{

    ifstream file(filename, ios::binary);

    if (!file)

    {

        cerr << "Error opening file for reading!" << endl;

        return;

    }

    Patient patient;

    while (file.read(reinterpret\_cast<char \*>(&patient), sizeof(Patient)))

    {

        cout << "Patient: " << patient.surname << " " << patient.name << " " << patient.patronymic << endl;

        cout << "Adress: " << patient.address << endl;

        cout << "Medical card number: " << patient.medicalCardNumber << endl;

        cout << "Insurance policy number: " << patient.insurancePolicyNumber << endl;

        cout << "---------------------------" << endl;

    }

    file.close();

}

void addPatientsAtStart(const string &filename, const vector<Patient> &newPatients)

{

    ifstream file(filename, ios::binary);

    if (!file)

    {

        cerr << "Error opening file for reading!" << endl;

        return;

    }

    vector<Patient> patients(newPatients);

    Patient patient;

    while (file.read(reinterpret\_cast<char \*>(&patient), sizeof(Patient)))

    {

        patients.push\_back(patient);

    }

    file.close();

    ofstream outFile(filename, ios::binary | ios::trunc);

    if (!outFile)

    {

        cerr << "Error opening file for writing!" << endl;

        return;

    }

    for (const auto &p : patients)

    {

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

    }

    outFile.close();

}

void deletePatientByCardNumber(const string &filename, int cardNumber)

{

    ifstream file(filename, ios::binary);

    if (!file)

    {

        cerr << "Error opening file for writing!" << endl;

        return;

    }

    vector<Patient> patients;

    Patient patient;

    bool found = false;

    while (file.read(reinterpret\_cast<char \*>(&patient), sizeof(Patient)))

    {

        if (patient.medicalCardNumber != cardNumber)

        {

            patients.push\_back(patient);

        }

        else

        {

            found = true;

        }

    }

    file.close();

    if (!found)

    {

        cerr << "Patient with medical record number " << cardNumber << " not found!" << endl;

        return;

    }

    ofstream outFile(filename, ios::binary | ios::trunc);

    if (!outFile)

    {

        cerr << "Error opening file for writing!" << endl;

        return;

    }

    for (const auto &p : patients)

    {

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

    }

    outFile.close();

}

int main()

{

    string filename = "patients.bin";

    vector<Patient> initialPatients = {

        {"Shevchenko", "Taras", "Hryhorovych", "Poltava", 12345, 67890},

        {"Kovalenko", "Oleksandr", "Petrovych", "Rivne", 12346, 67891}};

    createFile(filename, initialPatients);

    cout << "The initial contents of the file:" << endl;

    printFile(filename);

    int cardNumberToDelete = 12345;

    deletePatientByCardNumber(filename, cardNumberToDelete);

    cout << "File contents after deleting a patient with a medical record number " << cardNumberToDelete << ":" << endl;

    printFile(filename);

    vector<Patient> newPatients = {

        {"Doroshenko", "Mykola", "Ivanovych", "Kryvyi Rih", 12347, 67892},

        {"Lytvynenko", "Vasyl", "Anatoliyovych", "Lviv", 12348, 67893}};

    addPatientsAtStart(filename, newPatients);

    cout << "File contents after adding patients:" << endl;

    printFile(filename);

    return 0;

}

Завдання №3

<https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/271/files>

#include <iostream>

#include <fstream>

#include <vector>

#include <string>

#include <sstream>

using namespace std;

int main()

{

    ifstream file1("F1.txt");

    ofstream file2("F2.txt");

    if (!file1)

    {

        cerr << "Error opening file for reading!" << endl;

        return 1;

    }

    if (!file2)

    {

        cerr << "Error opening file for writting!" << endl;

        return 1;

    }

    string line;

    string biggestWord;

    int max = -1;

    while (getline(file1, line))

    {

        stringstream ss(line);

        std::string word;

        int wordCount = 0;

        while (ss >> word)

        {

            wordCount++;

        }

        if (wordCount == 1)

        {

            file2.write(line.c\_str(), line.size());

            if (static\_cast<int>(line.size()) > max)

            {

                biggestWord = line;

                max = line.size();

            }

            file2.put('\n');

        }

    }

    cout << "The biggest word in file2: " << biggestWord;

    file1.close();

    file2.close();

    return 0;

}

Завдання №4.1

<https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/271/files>

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

int main()

{

    int n, k;

    cin >> n >> k;

    vector<int> numbers(n);

    for (int i = 0; i < n; i++)

    {

        cin >> numbers[i];

    }

    sort(numbers.begin(), numbers.end());

    auto it = unique(numbers.begin(), numbers.end());

    numbers.resize(distance(numbers.begin(), it));

    k %= numbers.size();

    rotate(numbers.begin(), numbers.begin() + k, numbers.end());

    cout << numbers.size() << endl;

    for (int i = 0; i < numbers.size(); i++)

    {

        cout << numbers[i] << ' ';

    }

}

Завдання №4.2

<https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/271/files>

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

void bubbleSortArr(vector<int> &vec)

{

    bool swapped;

    for (int i = 0; i < vec.size() - 1; i++)

    {

        swapped = false;

        for (int j = 0; j < vec.size() - i - 1; j++)

        {

            if (vec[j] > vec[j + 1])

            {

                int temp = vec[j];

                vec[j] = vec[j + 1];

                vec[j + 1] = temp;

                swapped = true;

            }

        }

        if (!swapped)

            break;

    }

}

vector<int> uniqueArr(const vector<int> &vec)

{

    vector<int> newArr;

    for (int i = 0; i < vec.size(); i++)

    {

        bool single = true;

        for (int j = 0; j < newArr.size(); j++)

        {

            if (vec[i] == newArr[j])

            {

                single = false;

            }

        }

        if (single)

        {

            newArr.push\_back(vec[i]);

        }

    }

    return newArr;

}

void rotateArr(vector<int> &vec, int k)

{

    k %= vec.size();

    int temp;

    for (int i = 0; i < k; i++)

    {

        temp = vec[0];

        vec.erase(vec.begin());

        vec.push\_back(temp);

    }

}

int main()

{

    int n, k;

    cin >> n >> k;

    vector<int> numbers(n);

    for (int i = 0; i < n; i++)

    {

        cin >> numbers[i];

    }

    bubbleSortArr(numbers);

    numbers = uniqueArr(numbers);

    rotateArr(numbers, k);

    cout << numbers.size() << endl;

    for (int i = 0; i < numbers.size(); i++)

    {

        cout << numbers[i] << ' ';

    }

}

Завдання №5

<https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/271/files>

#include <iostream>

#include <vector>

#include <map>

#include <set>

#include <algorithm>

#include <cctype>

using namespace std;

int main()

{

    int n, k;

    cin >> n >> k;

    map<string, int> wordCount;

    vector<string> words(n);

    string word;

    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> uniqueLetters;

    for (const auto &entry : wordCount)

    {

        if (entry.second >= k)

        {

            for (char c : entry.first)

            {

                uniqueLetters.insert(c);

            }

        }

    }

    if (uniqueLetters.empty())

    {

        cout << "Empty!" << endl;

    }

    else

    {

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

        vector<char> sortedLetters(uniqueLetters.rbegin(), uniqueLetters.rend());

        for (char c : sortedLetters)

        {

            cout << c << " ";

        }

        cout << endl;

    }

    return 0;

}

Завдання №6

<https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/271/files>

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

enum FileOpResult

{

    Success,

    Failure

};

FileOpResult write\_to\_file(const string &name, const string &content)

{

    if (name.empty() || content.empty())

    {

        return Failure;

    }

    ofstream file(name, ios::out | ios::trunc);

    if (!file.is\_open())

    {

        return Failure;

    }

    file << content;

    if (!file)

    {

        file.close();

        return Failure;

    }

    file.close();

    if (!file)

    {

        return Failure;

    }

    return Success;

}

FileOpResult copy\_file(const string &file\_from, const string &file\_to)

{

    if (file\_from.empty() || file\_to.empty())

    {

        return Failure;

    }

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

    if (!src\_file.is\_open())

    {

        return Failure;

    }

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

    if (!dest\_file.is\_open())

    {

        src\_file.close();

        return Failure;

    }

    dest\_file << src\_file.rdbuf();

    if (!dest\_file)

    {

        src\_file.close();

        dest\_file.close();

        return Failure;

    }

    src\_file.close();

    dest\_file.close();

    if (!src\_file || !dest\_file)

    {

        return Failure;

    }

    return Success;

}

int main()

{

    string file\_name, content, source\_file, target\_file;

    cout << "Введіть ім'я файлу для запису: ";

    getline(cin, file\_name);

    cout << "Введіть вміст файлу для запису: ";

    getline(cin, content);

    FileOpResult write\_result = write\_to\_file(file\_name, content);

    if (write\_result == Success)

    {

        cout << "Запис успішний." << endl;

    }

    else

    {

        cout << "Помилка запису." << endl;

    }

    cout << "Введіть ім'я файлу-джерела для копіювання: ";

    getline(cin, source\_file);

    cout << "Введіть ім'я файлу-призначення для копіювання: ";

    getline(cin, target\_file);

    FileOpResult copy\_result = copy\_file(source\_file, target\_file);

    if (copy\_result == Success)

    {

        cout << "Копіювання успішне." << endl;

    }

    else

    {

        cout << "Помилка копіювання." << endl;

    }

    return 0;

}

    }

Завдання №7

<https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/271/files>

#include <iostream>

#include <string>

using namespace std;

int main()

{

    string input;

    cin >> input;

    int count = 0;

    for (char c : input)

    {

        if (c == '4' || c == '7')

            count += 1;

    }

    cout << count;

    return 0;

}

## **5. Результати виконання завдань, тестування та фактично затрачений час:**

Завдання №1

Використовуючи бібліотеку <sstream>, вдалося зчитати рядок зі слів.

Використовуючи функцію з минулого епіку для перевірки, чи є слово паліндромом, зміг знайти всі слова, які є паліндромами.

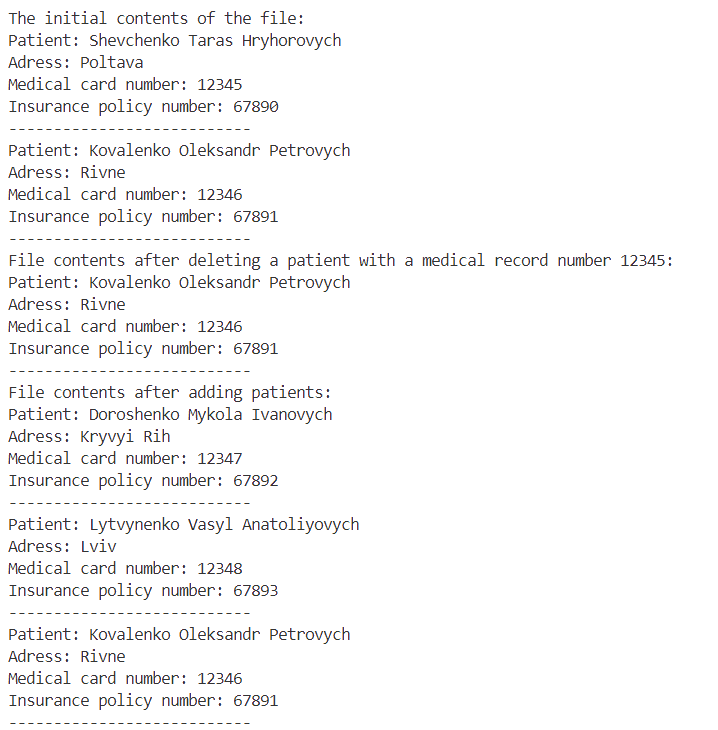


Час затрачений на виконання завдання: 35 хв.

Завдання №2

Створив структуру пацієнта з усіма потрібними данними.

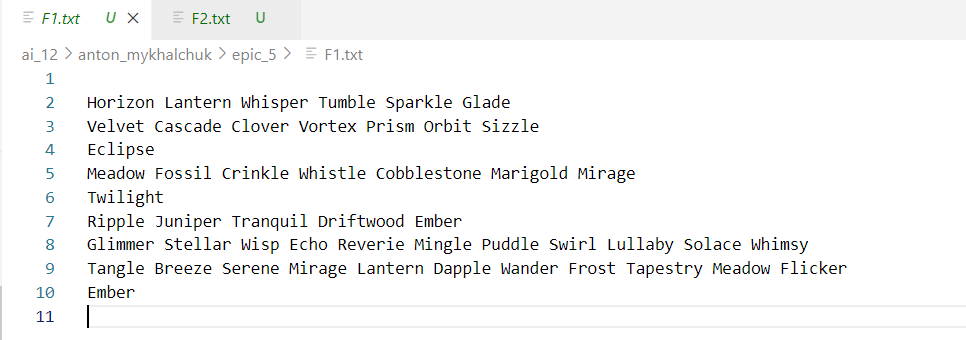
Також створив 4 функції для створення файлу, виведення інформації з файлу, додавання пацієнтів на початок файлу та видалення пацієнтів за номером картки.  
Кожна функція має перевірку, чи з файлом можливо взаємодіяти.

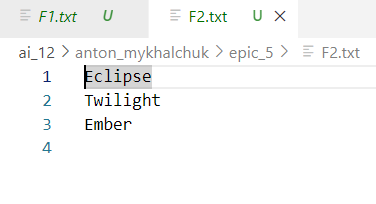


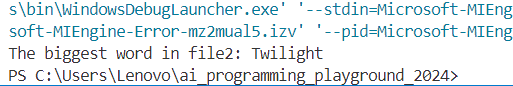
Час затрачений на виконання завдання: 2 год.

Завдання №3

З завчасно заготовленого файлу за допомогою while (getline(file1, line)) проходимось по кожному рядку. Далі за допомогою класу потоків перерахувуємо кількість слів в рядку. Якщо слово одне вписуємо його в другий файл і відразу перевіряємо, чи є воно максимальним.







Завдання №4

Для сортування у другому варіанті використовував сортування бульбашкою.



Завдання №5



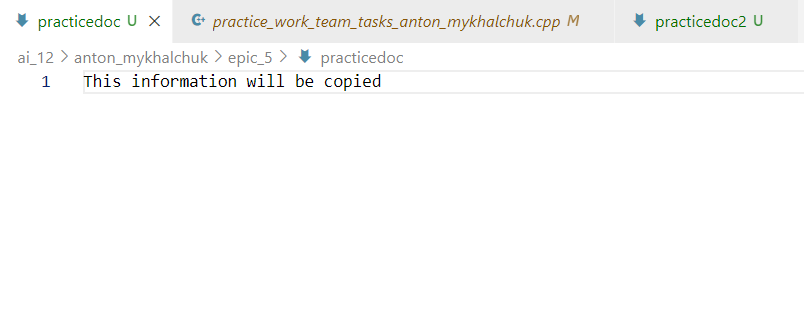
Завдання №6

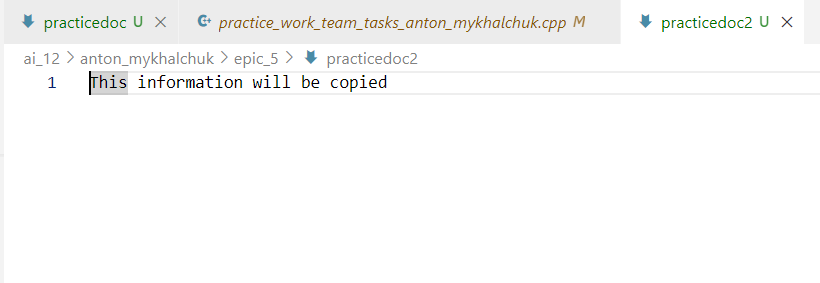
Оголошення через тип enum FileOpResult: Це тип, що дозволяє нам повертати результат операцій як Success (успішно) або Failure (неуспішно).

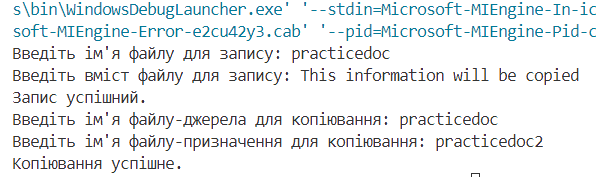
Створив функцію для запису у файл та його створення а також функцію для копіювання з одного файлу в інший контент.

Також введені перевірки, чи існує таке ім’я файлу, чи можливо його відкрити.

Коротко кажучи, цей код спочатку дозволяє вам записати текст у файл, а потім скопіювати вміст одного файлу в інший, виводячи повідомлення про успішність кожної операції.







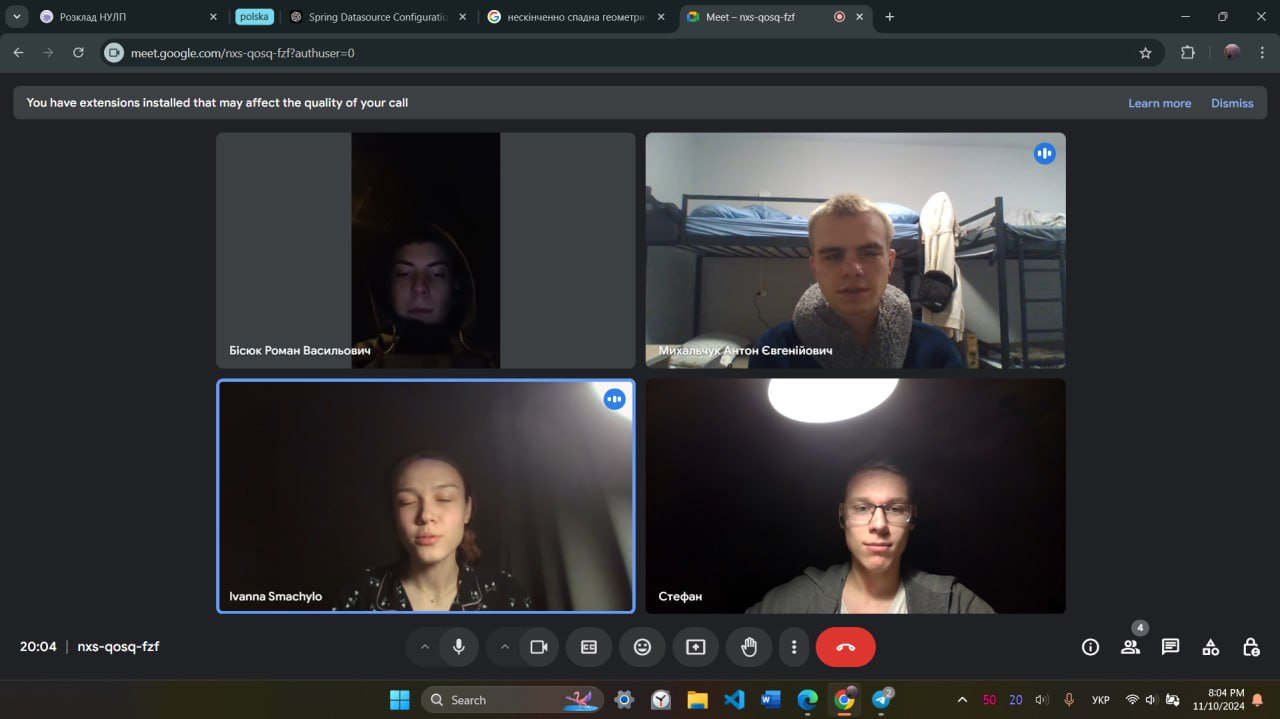
Завдання №7

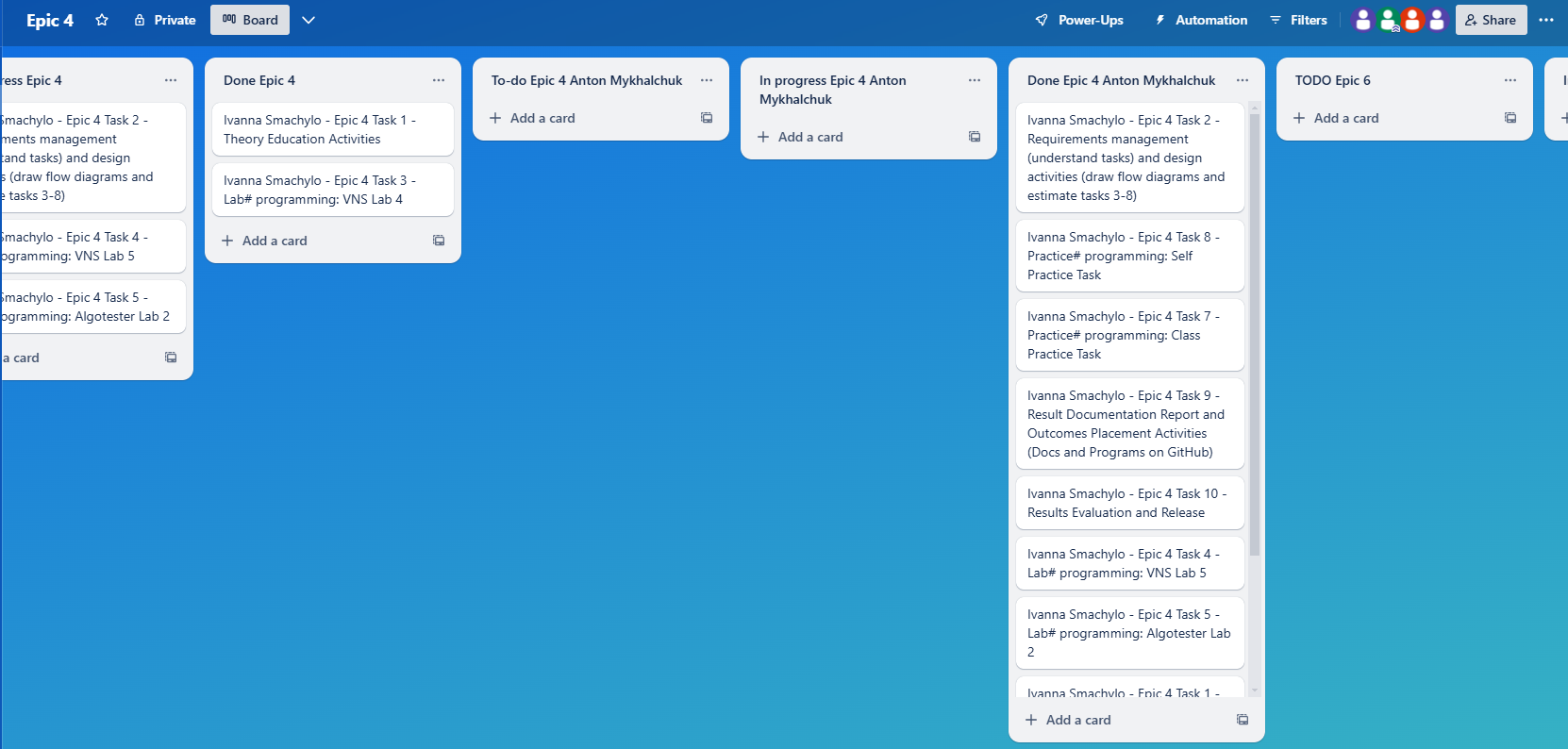
Зчитуємо рядок. За допомогою for each перебираємо кожен символ і перевіряємо, чи збігається умова задачі.

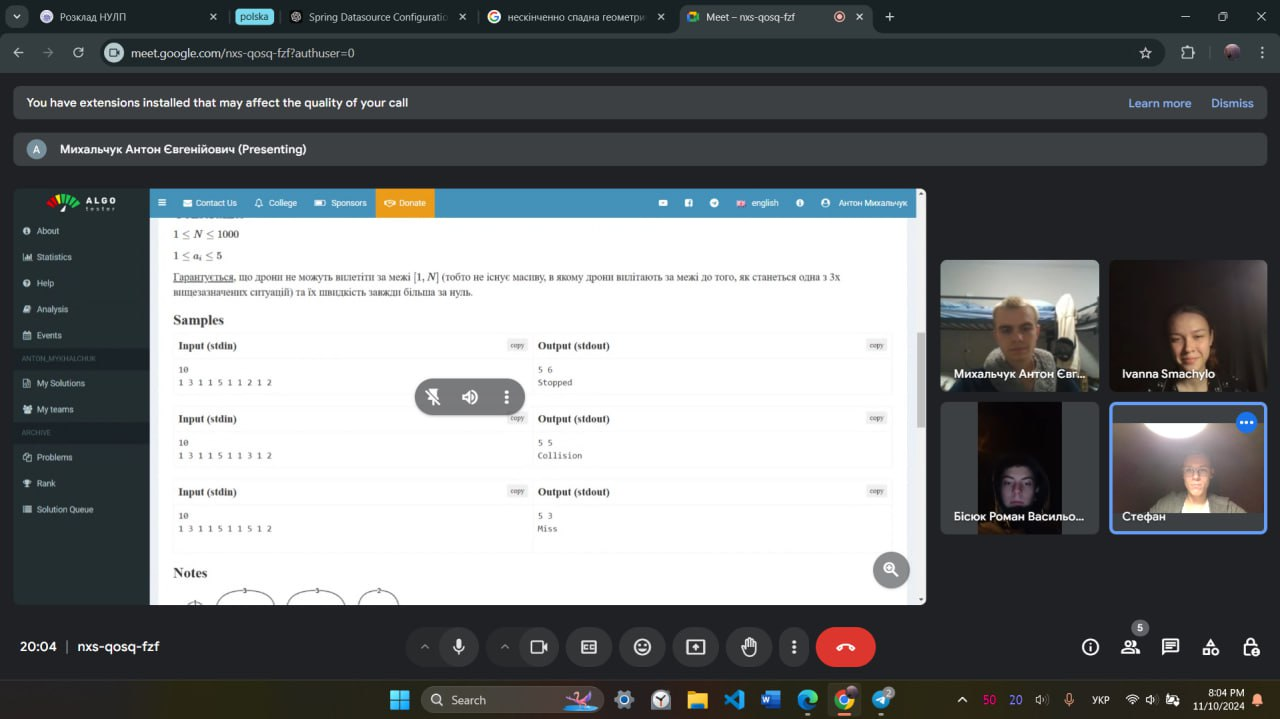
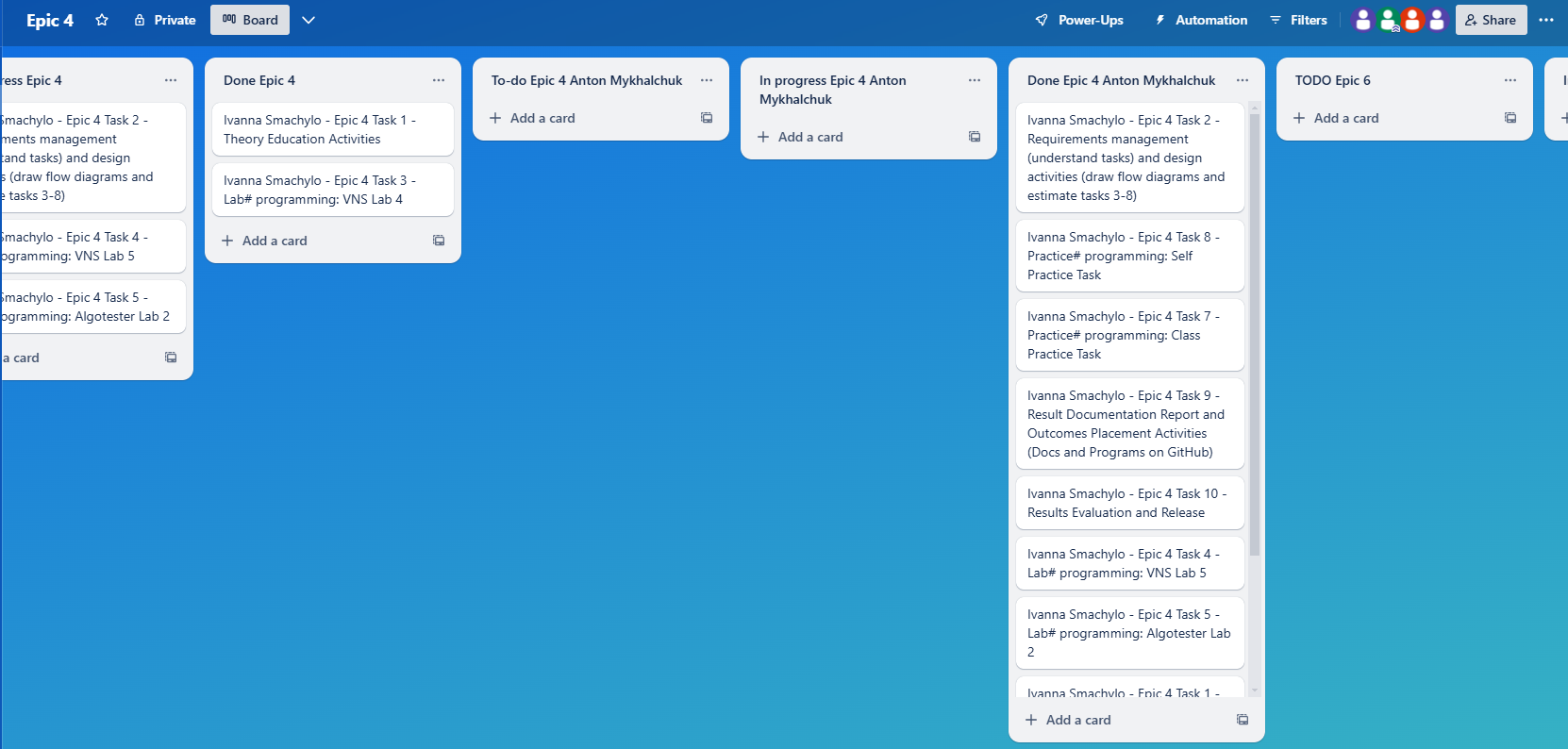
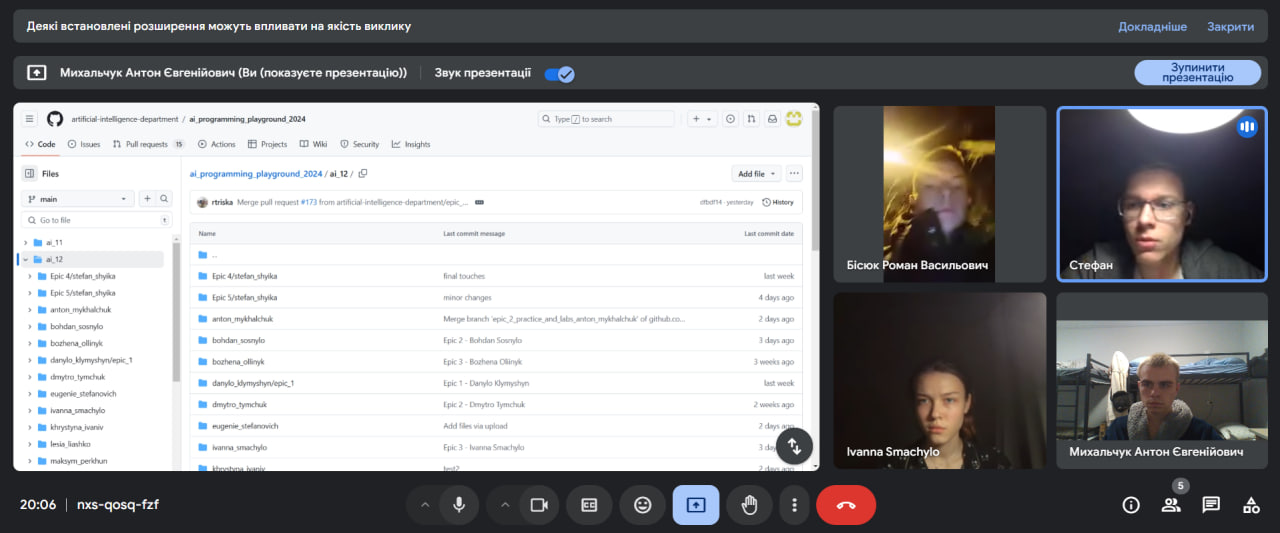


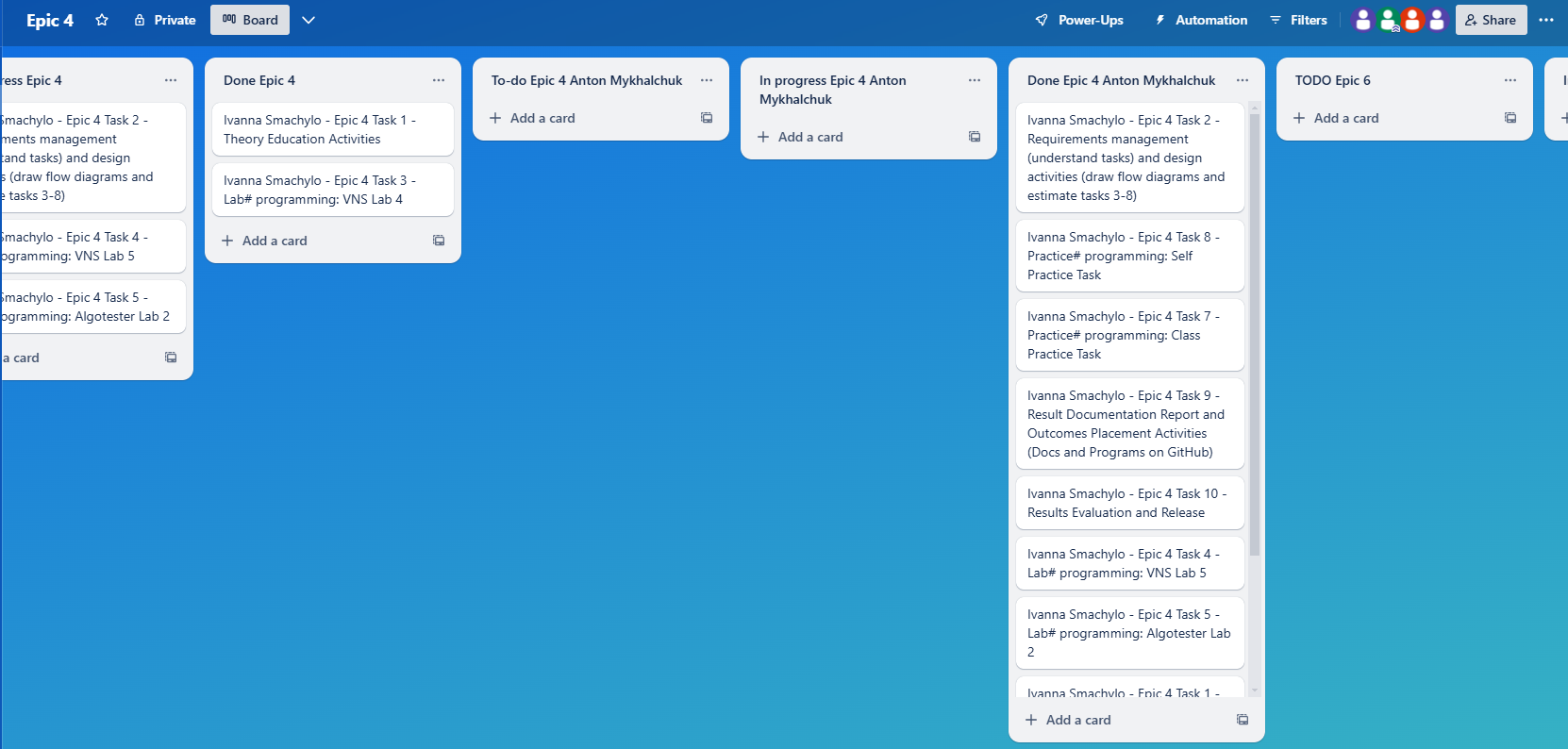
## **6. Кооперація з командою:**

* Скрін з 1-ї зустрічі по обговоренню задач Епіку та Скрін прогресу по Трелло





* Скрін з 2-ї зустрічі по обговоренню задач Епіку та Скрін прогресу по Трелло
* 
* 
* Скрін з 3-ї зустрічі по обговоренню задач Епіку та Скрін прогресу по Трелло (опційно)
* 



# **Висновки:**

У результаті виконання цієї роботи я розглянув основи роботи з файлами у C++ і практично освоїв принципи роботи з текстовими та бінарними файлами. Я вивчив, як здійснювати введення та виведення даних у файли, а також виконував різні операції з текстовими та бінарними файлами. Під час роботи було особливо корисно ознайомитися з особливостями маніпуляції символами та рядковими змінними у файлах, що є важливою частиною обробки даних у C++.

Також важливою частиною роботи було ознайомлення зі стандартною бібліотекою C++ для роботи з файлами, що значно спрощує виконання рутинних операцій, як-от запис, читання, копіювання файлів тощо. Це дозволяє будувати гнучкий і надійний код, який зручно використовувати в проєктах.

Важливим результатом роботи стало розуміння концепцій повторного використання коду через створення власних бібліотек. Це дозволяє покращити структуру програм, забезпечуючи більш організоване управління кодом та підвищуючи його ефективність і зручність використання. Я навчився створювати функції, які не тільки виконують певні операції, але й повертають статус результату операції (наприклад, успіх чи невдача), що є хорошою практикою для покращення якості та стійкості коду.

Робота з файлами різних типів, створення функцій для роботи з файлами та структурування коду у вигляді бібліотек допомогли мені краще зрозуміти, як організовувати програмний код у великих проєктах та підвищувати його придатність до розширення та повторного використання.