Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту



**Звіт**

**ро виконання лабораторних та практичних робіт блоку № 5**

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

***з дисципліни:*** «Мови та парадигми програмування»

до:

ВНС Лабораторної Роботи № 6

ВНС Лабораторної Роботи № 8

ВНС Лабораторної Роботи № 9

Алготестер Лабораторної Роботи №4

Алготестер Лабораторної Роботи №6

Практичних Робіт №5

***Виконала:***

студентка групи ШІ-13

Вітковська Марія Володимирівна

# **Тема роботи:**

Стандартна бібліотека та деталі/методи роботи з файлами. Робота із двійковими файлами. Ввід-вивід рядків. Створення й використання бібліотек.

# **Мета роботи:**

Знайомство з простими структурами даних. Робота з одновимірними та двовимірними масивами. Практика з алгоритмами обробки. Отримання навичок в обробці масивів з використанням функцій та передаванні масивів, як параметрів функцій.

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Потоковий та блоковий ввід-вивід. Базовий та рандомний файловий ввід-вивід
* Тема №2: Системи числення
* Тема №3: Бінарні Файли
* Тема №4: Створення й використання бібліотек
* Тема №5: Вектори в C++

1. Індивідуальний план опрацювання теорії:

**Тема №1: Потоковий та блоковий ввід-вивід**

* + Джерела Інформації:
    - Конспект лекцій з дисципліни «Алгоритмізація та програмування. Частина1» Л.І. Мочурад.
    - <https://www.youtube.com/playlist?list=PLVlQHNRLflP8_DGKcMoRw-TYJJALgGu4J>
    - <https://www.geeksforgeeks.org/>
    - <https://chat.openai.com/>
    - <https://acode.com.ua/urok-215-potoky-vvodu-i-vyvodu/>
    - <https://acode.com.ua/urok-220-bazovyj-fajlovyj-vvid-i-vyvid/>
    - <https://acode.com.ua/urok-221-randomnyj-fajlovyj-vvid-i-vyvid/>
  + Що опрацьовано:
    - Потоки вводу та виводу.функції, що використовуються для блочного вводу та виводу.
    - Класи і об’єкти для роботи з файлами, такі як std::fstream, std::ifstream, std::ofstream. Як відкривати, закривати, читати, писати, переміщатися та перевіряти стан файлів.
    - Формати і режими файлів (in, out, app, binary, trunc), такі як текстові, бінарні, додавання, перезапис, читання тощо. Різні способи доступу до даних у файлі. Cпособи перевірки стану файлу, такі як функції is\_open(), good(), eof(), fail().
    - Бібліотека fstream, яка дозволяє відкривати, закривати, читати, писати, переміщатися та перевіряти стан файлів.
  + Статус: Ознайомлена
  + Початок опрацювання теми: 6.12
  + Завершення опрацювання теми: 7.12

**Тема №2: Системи числення**

* + Джерела Інформації:
    - Конспект лекцій з дисципліни «Алгоритмізація та програмування. Частина1» Л.І. Мочурад.
    - <https://www.geeksforgeeks.org/>
    - <https://chat.openai.com/>
    - <https://dotnettutorials.net/lesson/introduction-to-number-system/>
    - <https://www.youtube.com/watch?v=jgcTDKjepNM>
  + Що опрацьовано:
    - Визначення та різниця між системами числення, такими як: двійкова, вісімкова, десяткова та шістнадцяткова.
  + Статус: Ознайомлена
  + Початок опрацювання теми: 7.12
  + Завершення опрацювання теми: 7.12

**Тема №3: Бінарні Файли**

* + Джерела Інформації:
    - Конспект лекцій з дисципліни «Алгоритмізація та програмування. Частина1» Л.І. Мочурад.
    - <https://www.youtube.com/playlist?list=PLVlQHNRLflP8_DGKcMoRw-TYJJALgGu4J>
    - <https://chat.openai.com/>
    - <https://www.codespeedy.com/reading-and-writing-binary-file-in-cpp/>
    - <https://www.geeksforgeeks.org/file-handling-c-classes/>
    - <https://acode.com.ua/urok-220-bazovyj-fajlovyj-vvid-i-vyvid/>
  + Що опрацьовано:
    - Формати бінарних файлів: які переваги та недоліки бінарних файлів порівняно з текстовими, як читати та записувати різні типи даних у бінарних файлах
    - Класи і об’єкти для роботи з файлами, такі як std::fstream, std::ifstream, std::ofstream. Як відкривати, закривати, читати, писати, переміщатися та перевіряти стан файлів.
    - Формати і режими файлів (in, out, app, binary, trunc), такі як текстові, бінарні, додавання, перезапис, читання тощо. Різні способи доступу до даних у файлі.
    - Бібліотека fstream, яка дозволяє відкривати, закривати, читати, писати, переміщатися та перевіряти стан файлів
  + Статус: Ознайомлена
  + Початок опрацювання теми: 10.12
  + Завершення опрацювання теми: 10.12
* **Тема №4: Створення й використання бібліотек**
  + Джерела Інформації:
    - Конспект лекцій з дисципліни «Алгоритмізація та програмування. Частина1» Л.І. Мочурад.
    - <https://www.youtube.com/playlist?list=PLVlQHNRLflP8_DGKcMoRw-TYJJALgGu4J>
    - <https://chat.openai.com/>
    - <https://acode.com.ua/pidklyuchennya-ta-vykorystannya-bibliotek-v-visual-studio/>
    - <https://acode.com.ua/statychni-i-dynamichni-biblioteky/>
  + Що опрацьовано:
    - Бібліотеки: створення й визначення функцій, компіляції в бібліотеку. Використання бібліотек: підключення бібліотек, використання функцій бібліотеки в коді.
  + Статус: Ознайомлена
  + Початок опрацювання теми: 3.12
  + Завершення опрацювання теми: 3.12

**Тема №5: Вектори в C++**

* + Джерела Інформації:
    - Конспект лекцій з дисципліни «Алгоритмізація та програмування. Частина1» Л.І. Мочурад.
    - <https://www.youtube.com/playlist?list=PLVlQHNRLflP8_DGKcMoRw-TYJJALgGu4J>
    - <https://chat.openai.com/>
    - <https://www.geeksforgeeks.org/>
    - <https://acode.com.ua/urok-99-vvedennya-v-std-vector/>
  + Що опрацьовано:
* Базові операції з векторами. Оголошення векторів. Основні аспекти використання вкторів, такі як: включення бібліотеки, додавання елементів, доступ до елементів за індексом, ітерація через вектор, видалення елементів.
  + Статус: Ознайомлена
  + Початок опрацювання теми: 4.12
  + Завершення опрацювання теми: 4.12

# **Виконання роботи:**

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

**Завдання №1 Опрацювання теорії**

* Деталі завдання: опрацювання різноманітних матеріалів, аналіз та дослідження відео, статей, книг на задані теми. Практика з простими прикладами та задачками для освоєння отриманого матеріалу Організація та структурування отриманих даних для можливості ефективніше засвоювати отримані знання на практиці. Вивчення найважливіших моментів.

**Завдання №2 Вимоги та проектування за допомогою Draw.io(складання блок-схем)**

* Деталі завдання: введення завдання в Trello та їхнє переміщення в відповідну колонку в процесі виконання роботи, побудова блок-схем до завдань.

**Завдання №3 Лабораторна №6 з VNS, завдання 1**

* Варіант завдання: **24**
* Деталі завдання: Для рядка знайти кількість слів у ньому. Задано рядок, що складається із символів. Символи поєднуються в слова. Слова одне від одного відокремлюються одним або декількома пробілами. Наприкінці тексту ставиться крапка. Виконати ввід рядка, використовуючи функцію gets(s) і здійснити обробку рядка у відповідності зі своїм варіантом.

**Завдання №4 Лабораторна №8 з VNS, завдання 1**

* Варіант завдання: **24**
* Деталі завдання: 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 "Студент":

- прізвище, ім'я, по батькові;

- номер телефону;

- група;

- оцінки по 3 основних предметах.

Знищити всі елементи із групи із зазначеним номером, у яких середнє арифметичне оцінок менше заданого, додати елемент після елемента із заданим прізвищем

**Завдання №5 Лабораторна №9 з VNS, завдання 1**

* Варіант завдання: **24**
* Деталі завдання: Створити текстовий файл F1 не менше, ніж з 10 рядків і записати в нього інформацію

1) Скопіювати з файлу F1 у файл F2 всі рядки, у яких міститься не менш двох однакових слів.

2) Визначити номер слова, у якому найбільше цифр

**Завдання №6 Algotester Lab 4, варіант 2**

* Деталі завдання: Вам дано масив *a* з *N* цілих чисел.

Спочатку видаліть масиву a усі елементи що повторюються, наприклад масив [1, 3, 3, 4] має перетворитися у [1, 3, 4].  
Після цього оберніть посортовану версію масиву*a* на *K*, тобто при *K*=3 масив [1, 2, 3, 4, 5, 6, 7] перетвориться на [4, 5, 6, 7, 1, 2, 3].  
Виведіть результат.

Input:

У першому рядку цілі числа *N* та *K*

У другому рядку *N* цілих чисел - елементи масиву *a*

Output:

У першому рядку ціле число *N* - розмір множини *a*  
У наступному рядку *N* цілих чисел - множина *a*

**Завдання №7 Algotester Lab 6, варіант 2**

* Деталі завдання: У вас є шахова дошка розміром 8×8 та дуже багато фігур.

Кожна клітинка може мати таке значення:

* Пуста клітинка O
* Пішак P
* Тура R
* Кінь N
* Слон B
* Король K
* Королева Q

Вам дають позиції фігур на дошці (всі фігури одного кольору, кількість королів може бути > 1).

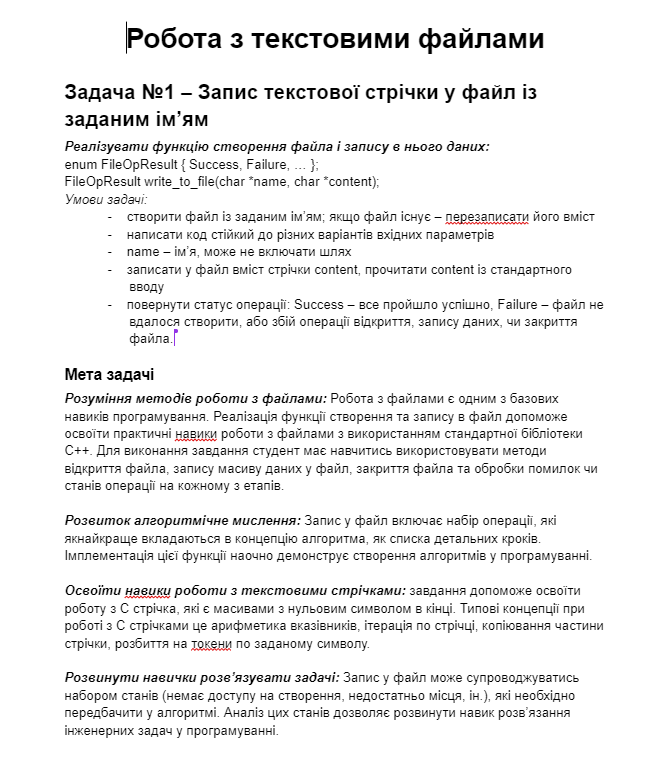
Далі йдуть Q запитів з координатами клітинки {x,y}{,}. На кожен запит ви маєте вивести стрічку si - посортовані за алфавітом букви фігур, які атакують цю клітинку (пішаки атакують вниз).

У випадку, якщо на клітинці стоїть якась фігура - виведіть символ X.

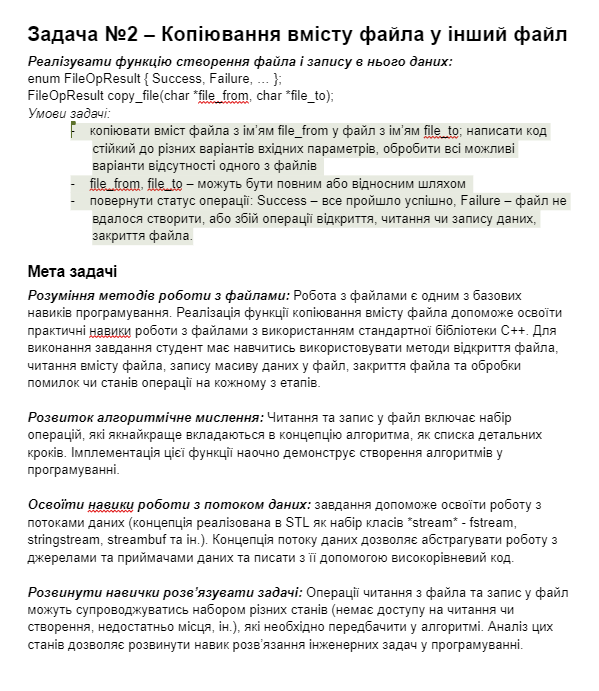
У випадку, якщо клітинку не атакують - виведіть O.

Наявніть фігури у певній клітинці не блокує атаку для іншої фігури. Тобто якщо між турою та клітинкою стоїть інша фігура - вважається що тура атакує цю клітинку.

**Завдання №8 Class Practice Task**

* Деталі завдання: 

Малюнок practice\_task\_1



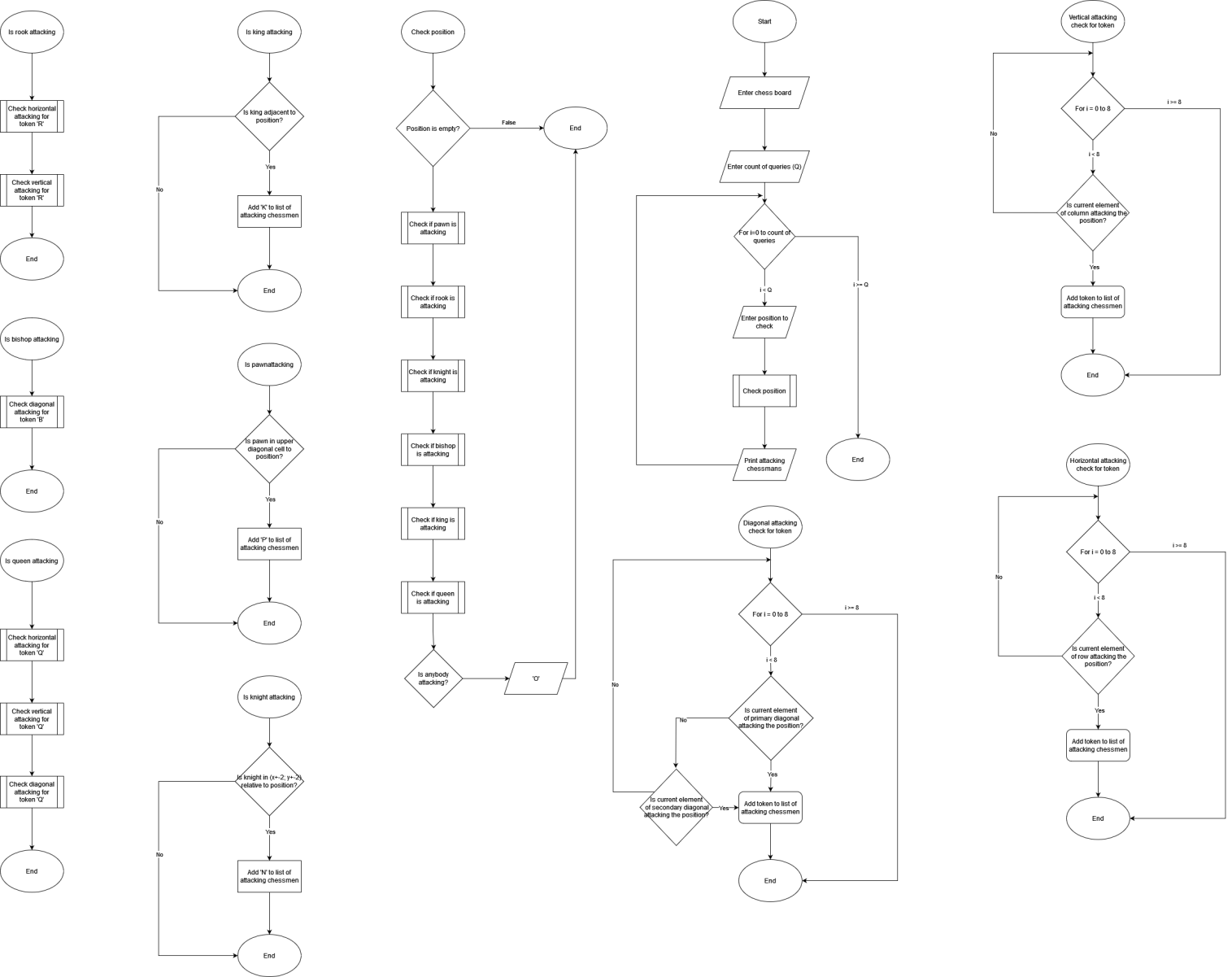
Малюнок practice\_task\_2

**Завдання №9 Self-Practice Task 1**

* Деталі завдання: Створити файл F1, у якому вказати список персонажів. Надати запит користувачу, щоб він вибрав якогось персонажа, який йому найбільше подобається з поданих. Його вибір зафіксувати, та перекинути цього персонажа у файл F2.

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

**Програма №5 Algotester Task 6, 2 variant**

* Блок-схема: 
* Планований час на реалізацію: 5 годин

**3. Код програм з посиланням на зовнішні ресурси:**

**Завдання №1 Лабораторна №6 з VNS, варіант 24**

#include <iostream>

#include <sstream>

#include <string>

#include <cstring>

using namespace std;

int CountWords(const string& str) {

    istringstream stream(str);

    string word;

    int wordCount = 0;

     while (stream >> word) {

        wordCount++;

    }

    return wordCount;

}

int main() {

    string input;

    cout << "Enter a line: ";

    getline(cin, input);

    int NumberOfWords = CountWords(input);

    cout << "Number of words in the entered line: " << NumberOfWords << endl;

}

VNS Lab6 Task1\_V24

**Завдання №2 Лабораторна №8 з VNS, варіант 24**

#include <iostream>

#include <fstream>

#include <string>

#include <vector>

#include <limits>

struct Student {

    std::string fullName;

    std::string phoneNumber;

    std::string group;

    std::vector<int> grades;

};

double calculateAverage(const std::vector<int>& grades) {

    if (grades.empty()) {

        return 0.0;

    }

    int sum = 0;

    for (int grade : grades) {

        sum += grade;

    }

    return static\_cast<double>(sum) / grades.size();

}

int main() {

    const int maxStudents = 5;

    const int requiredGrade = 75; // Задайте вашу константу тут

    std::vector<Student> students;

    char addAnotherStudent = 'y';

    while (addAnotherStudent == 'y' && students.size() < maxStudents) {

        Student currentStudent;

        std::cout << "Enter data for a student:\n";

        std::cout << "Full Name: ";

        std::getline(std::cin, currentStudent.fullName);

        std::cout << "Phone Number: ";

        std::getline(std::cin, currentStudent.phoneNumber);

        std::cout << "Group: ";

        std::getline(std::cin, currentStudent.group);

        std::cout << "Enter grades for 3 subjects:\n";

        for (int j = 0; j < 3; ++j) {

            int grade;

            std::cout << "Subject " << (j + 1) << ": ";

            std::cin >> grade;

            currentStudent.grades.push\_back(grade);

        }

        std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

        students.push\_back(currentStudent);

        if (students.size() < maxStudents) {

            std::cout << "Do you want to add another student? (y/n): ";

            std::cin >> addAnotherStudent;

            std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

        } else {

            std::cout << "Maximum number of students reached.\n";

        }

    }

    // Відкриття файлу для запису

    std::ofstream outputFile("students\_data.txt");

    // Перевірка чи вдалося відкрити файл

    if (outputFile.is\_open()) {

        // Запис даних у файл

        for (const auto& student : students) {

            outputFile << "Full Name: " << student.fullName << "\n";

            outputFile << "Phone Number: " << student.phoneNumber << "\n";

            outputFile << "Group: " << student.group << "\n";

            outputFile << "Grades: ";

            for (const auto& grade : student.grades) {

                outputFile << grade << " ";

            }

            outputFile << "\n\n";

        }

        // Закриття файлу

        outputFile.close();

        std::cout << "Data has been written to students\_data.txt.\n";

    } else {

        std::cerr << "Unable to open the file for writing.\n";

        return 1;

    }

    // Перевірка та видалення студентів

    for (auto it = students.begin(); it != students.end(); ) {

        if (calculateAverage(it->grades) < requiredGrade) {

            std::cout << "Average grade is below the required grade for student "

                      << it->fullName << ". Removing from the file.\n";

            // Видалення даного студента

            it = students.erase(it);

        } else {

            ++it;

        }

    }

    // Оновлення файлу зі списком студентів

    std::ofstream updatedFile("updated\_students\_data.txt");

    // Перевірка чи вдалося відкрити файл

    if (updatedFile.is\_open()) {

        // Запис оновлених даних у файл

        for (const auto& student : students) {

            updatedFile << "Full Name: " << student.fullName << "\n";

            updatedFile << "Phone Number: " << student.phoneNumber << "\n";

            updatedFile << "Group: " << student.group << "\n";

            updatedFile << "Grades: ";

            for (const auto& grade : student.grades) {

                updatedFile << grade << " ";

            }

            updatedFile << "\n\n";

        }

        // Закриття файлу

        updatedFile.close();

        std::cout << "Data has been updated and written to updated\_students\_data.txt.\n";

    } else {

        std::cerr << "Unable to open the file for writing.\n";

        return 1;

    }

    return 0;

}

VNS Lab8 Task1\_V24

**Завдання №3 Лабораторна №9 з VNS, варіант 24**

#include <iostream>

#include <fstream>

#include <sstream>

#include <unordered\_set>

#include <vector>

int main() {

    std::vector<std::string> lines; // Вектор для зберігання введених рядків

    std::string line;

    // Введення рядків вручну

    std::cout << "Enter the lines:\n";

    while (std::getline(std::cin, line) && !line.empty()) {

        lines.push\_back(line);

    }

    std::ofstream f1OutputFile("F1.txt"); // Вихідний файл F1

    std::ofstream f2OutputFile("F2.txt"); // Вихідний файл F2

    if (!f1OutputFile.is\_open() || !f2OutputFile.is\_open()) {

        std::cerr << "Can't open the file!" << std::endl;

        return 1;

    }

    std::unordered\_set<std::string> uniqueWords;

    std::vector<std::string> duplicateLines;

    for (const auto& inputLine : lines) {

        f1OutputFile << inputLine << std::endl; // Запис у файл F1

        std::istringstream iss(inputLine);

        std::unordered\_set<std::string> lineWords;

        std::string word;

        while (iss >> word) {

            if (!lineWords.insert(word).second) {

                // Знайдено дублікат слів у рядку

                duplicateLines.push\_back(inputLine);

                break;

            }

        }

    }

    // Записуємо рядки з дублікатами слів у файл F2

    for (const auto& duplicateLine : duplicateLines) {

        f2OutputFile << duplicateLine << std::endl;

    }

    std::cout << "Saved in  F1.txt and F2.txt." << std::endl;

    return 0;

}

VNS Lab9 Task1\_V24

**Завдання №4 Algotester Lab 4, варіант 2**

#include <iostream>

#include <vector>

#include <algorithm>

int main() {

    using namespace std;

    int n, k, b;

    cin >> n >> k;

    vector<int> uniqueSet;

    for (int i = 0; i < n; i++) {

        cin >> b;

        uniqueSet.push\_back(b);

    }

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

    auto x = unique(uniqueSet.begin(), uniqueSet.end());

    uniqueSet.erase(x,uniqueSet.end());

    if(k > uniqueSet.size()){

        k = k % uniqueSet.size();

    }

    rotate(uniqueSet.begin(), uniqueSet.begin() + k % uniqueSet.size(), uniqueSet.end());

    cout << uniqueSet.size() << endl;

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

        cout << uniqueSet[j] << " ";

    }

    return 0;

}

Algotester\_Lab\_Task\_3\_3

**Завдання №5 Algotester Lab 6, варіант 2**

#include <iostream>

#include <vector>

#include <algorithm>

#include <cstdint>

class Square {

public:

    uint8\_t x;

    uint8\_t y;

    Square(uint8\_t x, uint8\_t y) {

        if (x >= 8 || y >= 8) {

            std::cout << "Invalid square, x=" << x << ", y=" << y;

            exit(1);

        }

        this->x = x;

        this->y = y;

    }

};

class Chessboard {

private:

public:

    char table[8][8]{

            'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O',

            'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O',

            'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O',

            'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O',

            'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O',

            'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O',

            'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O',

            'O', 'O', 'O', 'O', 'O', 'O', 'O', 'O',

    };

    char getPiece(Square \*square) {

        return table[square->y][square->x];

    }

    void setPiece(Square \*square, char piece) {

        table[square->y][square->x] = piece;

    }

    bool attacks(Square \*target, Square \*attacker) {

        uint8\_t diffXAbs = std::abs(target->x - attacker->x);

        uint8\_t diffYAbs = std::abs(target->y - attacker->y);

        if (diffXAbs + diffYAbs == 0) {

            std::cout << "Asked if a square attacks itself, x=" << target->x << ", y=" << target->y;

            exit(1);

        }

        switch (getPiece(attacker)) {

            case 'O':

                return false;

            case 'P':

                return target->y == attacker->y + 1 && diffXAbs == 1;

            case 'R':

                return target->y == attacker->y || target->x == attacker->x;

            case 'N':

                return diffXAbs + diffYAbs == 3 && (diffXAbs == 1 || diffYAbs == 1);

            case 'B':

                return diffXAbs == diffYAbs;

            case 'K':

                return std::max(diffXAbs, diffYAbs) == 1;

            case 'Q':

                return diffXAbs == diffYAbs || target->y == attacker->y || target->x == attacker->x;

            default:

                std::cout << "Invalid attacker piece: " << getPiece(attacker);

                exit(1);

        }

    }

};

int main() {

    Chessboard table;

    for (int squareY = 0; squareY < 8; squareY++) {

        for (int squareX = 0; squareX < 8; squareX++) {

            char piece;

            std::cin >> piece;

            table.setPiece(new Square(squareX, squareY), piece);

        }

    }

    int i;

    std::cin >> i;

    while (i-- > 0) {

        int x, y;

        std::cin >> y >> x;

        Square target(x - 1, y - 1);

        if (table.getPiece(&target) != 'O') {

            std::cout << 'X' << std::endl;

            continue;

        }

        std::vector<char> attackers;

        for (int squareX = 0; squareX < 8; squareX++) {

            for (int squareY = 0; squareY < 8; squareY++) {

                Square attacker(squareX, squareY);

                if (table.getPiece(&attacker) != 'O' && table.attacks(&target, &attacker)) {

                    attackers.push\_back(table.getPiece(&attacker));

                }

            }

        }

        if (attackers.empty()) {

            std::cout << 'O' << std::endl;

        } else {

            std::sort(attackers.begin(), attackers.end());

            attackers.erase(std::unique(attackers.begin(), attackers.end()), attackers.end());

            for (char attacker: attackers) {

                std::cout << attacker;

            }

            std::cout << std::endl;

        }

    }

    return 0;

}

Algotester\_Lab\_Task\_3\_3

**Завдання №6 Class Practice Task №4**

#include <iostream>

#include <fstream>

#include <string>

enum FileOpResult { Success, Failure};

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

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

using namespace std;

int main(){

string fileName;

cout<<"Enter name of the file you want to write to\n";

getline(cin,fileName);

string text;

cout<<"Enter text you want to write into the file\n";

getline(cin,text);

FileOpResult resultOfWriting=write\_to\_file((fileName.c\_str()),(text.c\_str()));

if(resultOfWriting==Success){

    cout<<"Success\n";

}else{

    cout<<"Failure\n";

}

string fileNameForCopy;

cout<<"Enter name of the file you want to copy from\n";

getline(cin,fileNameForCopy);

string fileNamtoCopy;

cout<<"Enter name of the file you want to copy this into\n";

getline(cin,fileNamtoCopy);

FileOpResult resultOfCopying=copy\_file((fileNameForCopy.c\_str()),(fileNamtoCopy.c\_str()));

if(resultOfCopying==Success){

    cout<<"Success\n";

}else{

    cout<<"Failure\n";

}

    return 0;

}

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

ofstream myFile(name);

if (!myFile.is\_open()) {

        return Failure;

    }

myFile<<content;

    if(myFile.fail()){

    myFile.close();

    return Failure;

    }

else{

    myFile.close();

    return Success;

}

}

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

ifstream myReadFile(file\_from);

ofstream myCopiedFile(file\_to);

string myText;

if(!myReadFile.is\_open() || !myCopiedFile.is\_open()){

    return Failure;

}

while (getline(myReadFile, myText)) {

   myCopiedFile << myText << "\n";

}

myReadFile.close();

myCopiedFile.close();

return Success;

}

Class Practice Task

**Завдання №7 Self Practice Task 1**

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main() {

    string firstFileName = "characters.txt";

    string secondFileName = "favorite\_character.txt";

    ifstream firstFile(firstFileName);

    if (!firstFile.is\_open()) {

        cerr << "File " << firstFileName << " not found. Creating the file." << endl;

        ofstream createFirstFile(firstFileName);

        if (!createFirstFile.is\_open()) {

            cerr << "Failed to create " << firstFileName << endl;

            return 1;

        }

        createFirstFile << "Harry Potter\nBatman\nLorax\nWalter White\nHeo\n";

        createFirstFile.close();

        firstFile.open(firstFileName);

    }

    cout << "Which character do u like the most of all available?" << endl;

    cout << "Available characters:" << endl;

    string character;

    while (getline(firstFile, character)) {

        cout << character << endl;

    }

    cout << "Choose your favorite character: ";

    getline(cin, character);

    ofstream secondFile(secondFileName, ios::app);

    if (!secondFile.is\_open()) {

        cerr << "Failed to open " << secondFileName << " for writing." << endl;

        return 1;

    }

    secondFile << character << endl;

    firstFile.close();

    secondFile.close();

    cout << "Your favorite character has been added to " << secondFileName << endl;

    return 0;

}

Self Practice Task 1

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

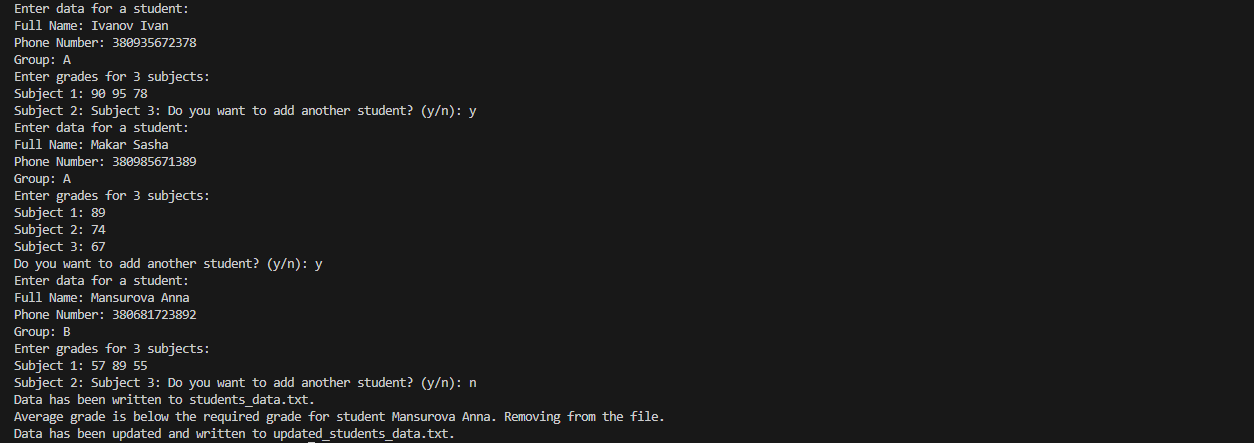
Завдання №1 **Лабораторна №6 з VNS, завдання 1 варіант 24**



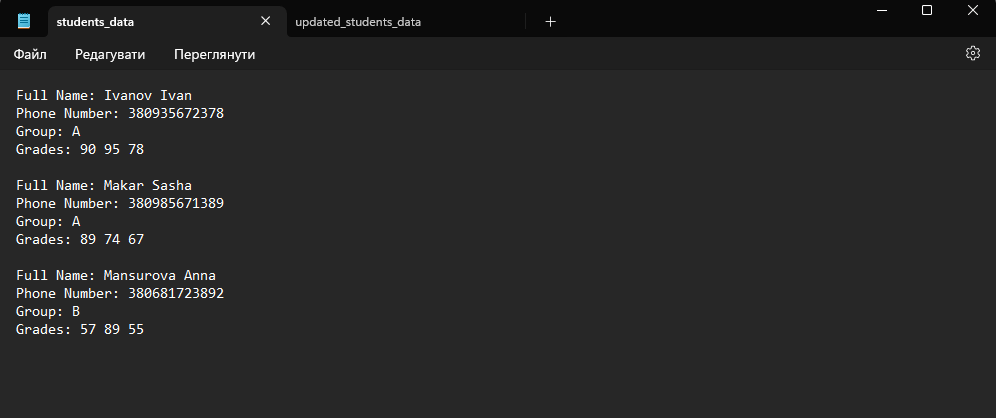
Малюнок VNS\_lab\_6\_results

Час затрачений на виконання завдання: 3 години

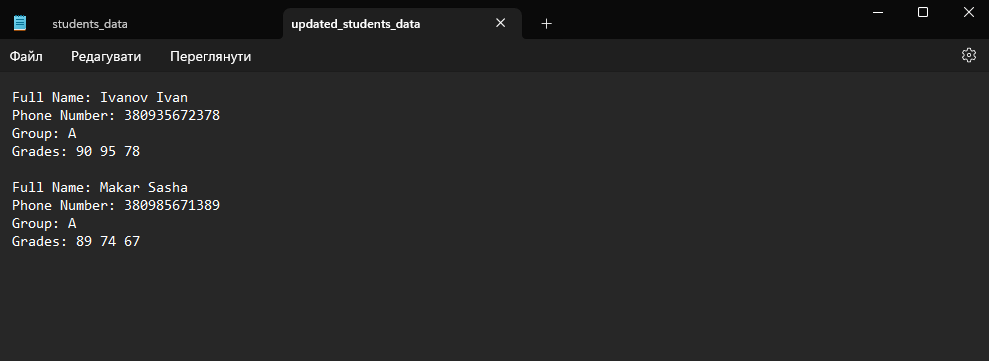
Завдання №2 **Лабораторна №8 з VNS, завдання 1 варіант 24**



Малюнок VNS\_8\_result



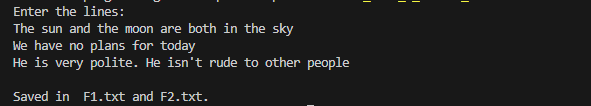
Малюнок VNS\_8\_result



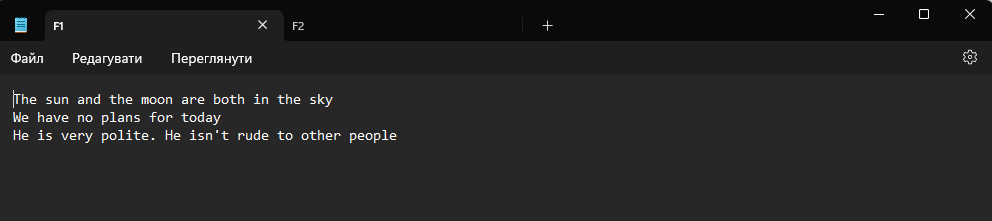
Малюнок VNS\_8\_result

Час затрачений на виконання завдання: 5 годин

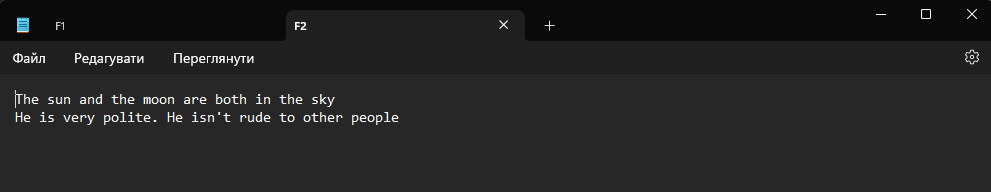
Завдання №3 **Лабораторна №9 з VNS, варіант 24**



Малюнок VNS\_9\_result



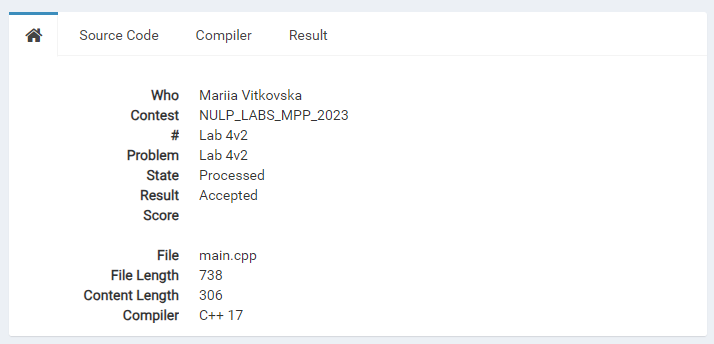
Малюнок VNS\_9\_result



Малюнок VNS\_3\_result

Час затрачений на виконання завдання: 4 години

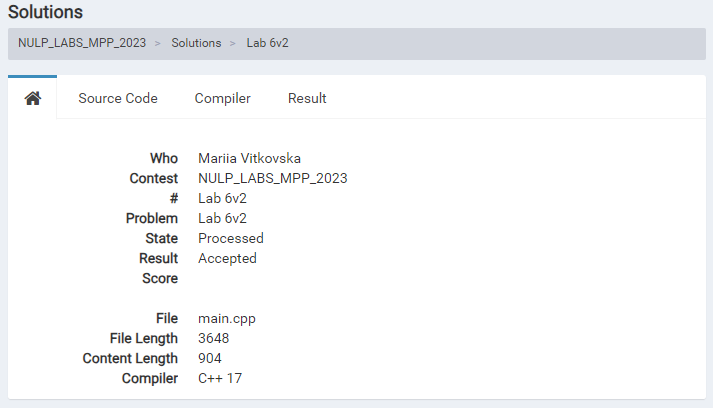
Завдання №4 **Algotester Lab 4, варіант 2**

****

Малюнок Algotester\_4\_result

Час затрачений на виконання завдання: 8 годин

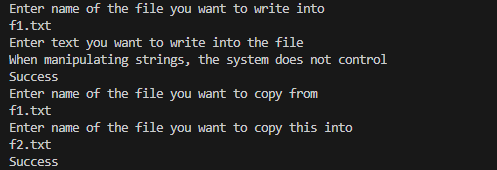
Завдання №4 **Algotester Lab 6, варіант 2**



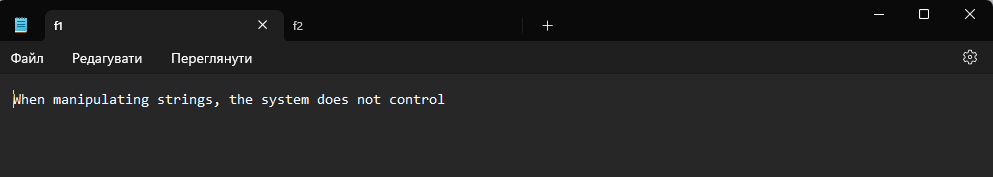
Малюнок Algotester\_6\_result

Час затрачений на виконання завдання: 10 годин

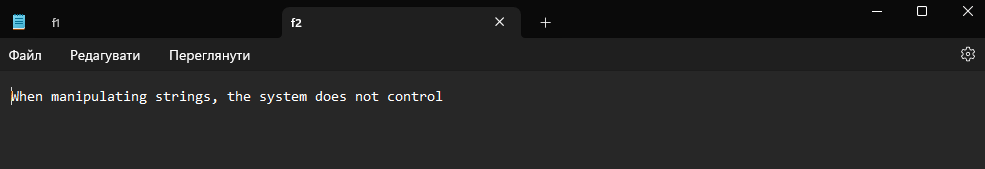
Завдання №5 **Class-Practice Task**



Малюнок Practice\_task\_result



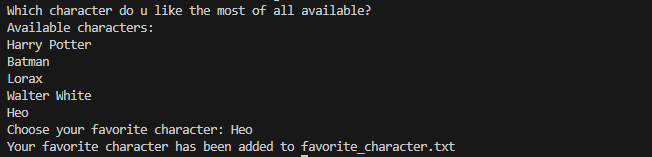
Малюнок Practice\_task\_result



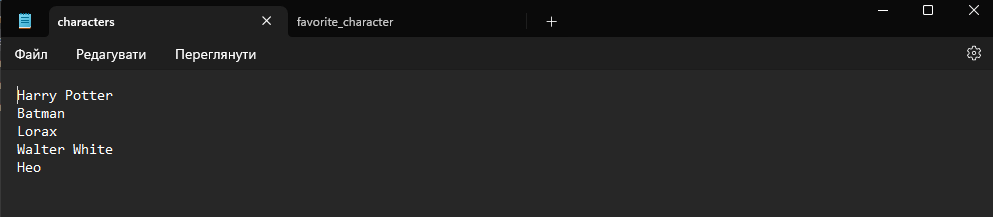
Малюнок Practice\_task\_result

Час затрачений на виконання завдання: 5 годин

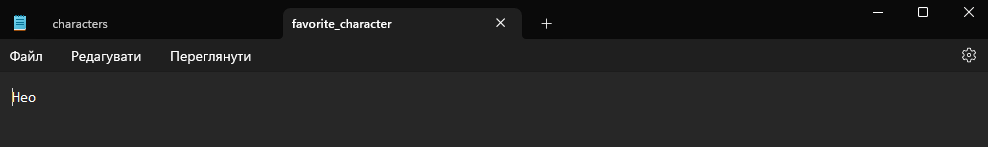
Завдання №6 **Self-Practice Task 1**



Малюнок Self\_practice\_result



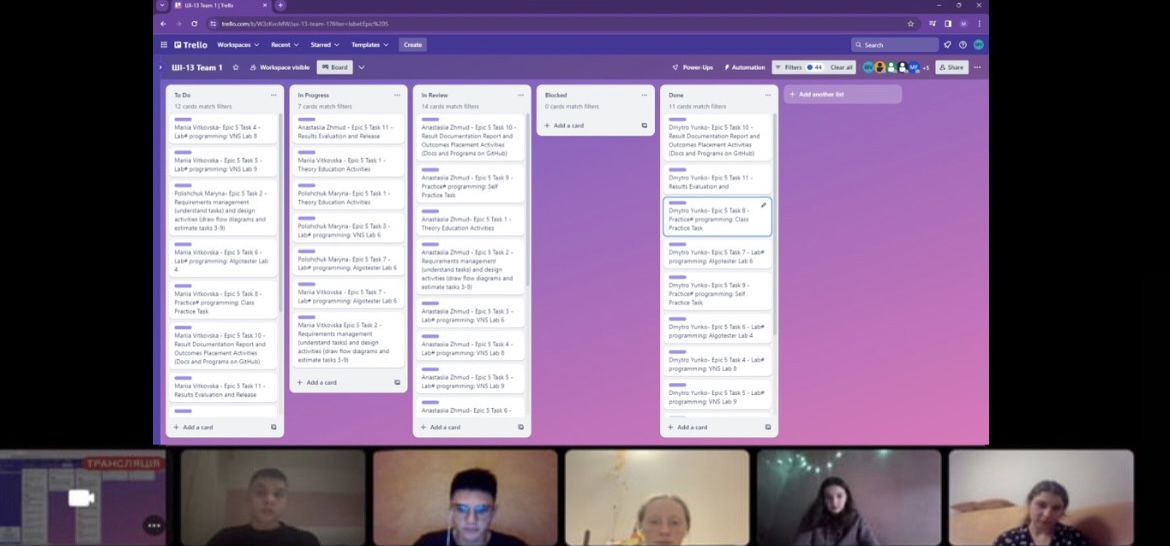
Малюнок result2



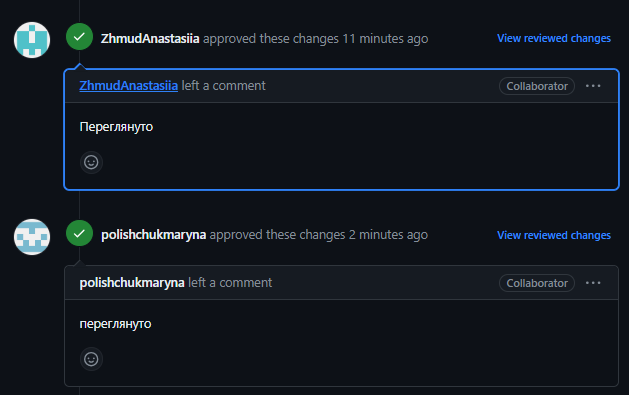
Малюнок result3

Час затрачений на виконання завдання: 30 хвилин

## **5. Кооперація з командою:**



Скрін зустрічі з командою при обговоренні тасків в Епіку №5 та прогресу в Trello

****

Малюнок Approve from teammates

**Pull-request:** **https://github.com/artificial-intelligence-department/ai\_programming\_playground/pull/936#pullrequestreview-1782607779**

# **Висновки:**

В процесі вивчення тем «Потоковий та блоковий ввід-вивід. Базовий та рандомний файловий ввід-вивід», «Бінарні Файли Вектори в C++», «Системи числення» та «Створення й використання бібліотек», «Вектори в C++» було отримано всі теоретичні знання, які включають основи одновимірних та двовимірних масивів, роботу з вказівниками та методи обробки масивів за допомогою функцій, і виконано додаткові нескладні завдання для освоєння матеріалу для подальшого виконання практичних задач у цій лабораторній роботі. Ці навички надали змогу ефективно розв’язати поставлені задачі та покращити якість коду. Виконавши всі надані завдання, було засвоєно теоретичний матеріал, який можна буде ефективно використовувати в подальших роботах та завданнях.