Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту



**Звіт**

про виконання

**Лабораторних та практичних робіт № 5**

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

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

***Виконав:***

Студент групи ШІ-11

Дах Тарас

# **Тема роботи:**

Робота із двійковими файлами. Ввід-вивід рядків.

# **Мета роботи:**

Ознайомитись з організацією вводу-виводу структурованої інформації із зовнішніх носіїв. Навчитися працювати з текстовими файлами, ввід-вивід текстової інформації і її зберігання на зовнішніх носіях.

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Потоковий ввід-вивід
* Тема №2: Бінарні файли
* Тема №3: Робота з файлами

1. Індивідуальний план опрацювання теорії:

* Тема №1: Потоковий ввід-вивід.
  + Джерела Інформації
    - <http://cpp.dp.ua/potokove-vvedennya-vyvedennya/>
    - <https://acode.com.ua/urok-215-potoky-vvodu-i-vyvodu/>
  + Що опрацьовано:
    - Ввід/вивід в C++
    - Стандартні потоки С++
  + Статус: Ознайомлений
  + Початок опрацювання теми: 09.12.23
  + Звершення опрацювання теми: 11.12.23
* Тема №2: Блоковий ввід-вивід
  + Джерела Інформації:
  + <https://purecodecpp.com/uk/archives/2751> Що опрацьовано:
    - Функції, що використовуються для блочного вводу та виводу
  + Статус: Ознайомлений
  + Початок опрацювання теми: 05.12.23
  + Звершення опрацювання теми: 10.11.23

* Тема №3: Робота з файлами
  + Джерела Інформації:
    - http://www.kytok.org.ua/post/vector-u-cplusplus
    - https://www.bestprog.net/uk/2021/10/12/c-the-vector-class-methods-that-define-and-modify-the-general-characteristics-of-array-ua/
    - https://acode.com.ua/urok-99-vvedennya-v-std-vector/
  + Статус: Ознайомлений
  + Початок опрацювання теми: 10.12.23
  + Звершення опрацювання теми: 12.12.23

**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. Організація та структурування отриманих даних для можливості ефективніше засвоювати отримані знання на практиці. Вивчення найважливіших моментів.

**Завдання №2 Перегляд вимог та проектування**

* Деталі завдання

Проектування блок-схем в Draw.io.

**Завдання №3 VNS Lab 6**

* Варіант завдання: 4
* Деталі завдання : Надрукувати всі слова, які співпадають з її першим словом.

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.

**Завдання №4 VNS Lab 8**

* Варіант завдання: 4
* Деталі завдання:
* Структура "Людина":
* - прізвище, ім'я, по батькові;
* - домашня адреса;
* - номер телефону;
* - вік.

Сформувати двійковий файл із елементів, заданої у варіанті структури,

роздрукувати його вмістиме, виконати знищення й додавання елементів у

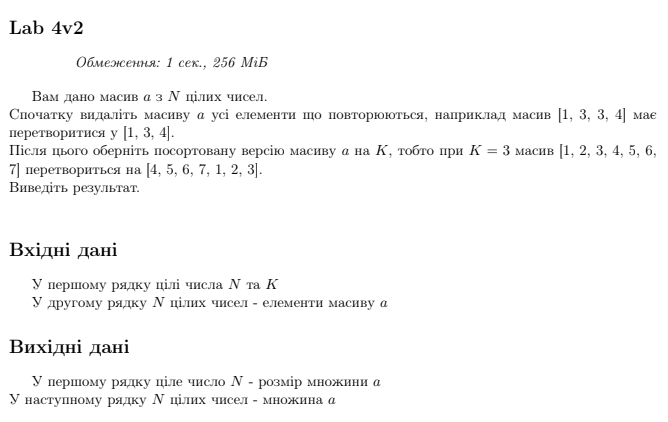
відповідності зі своїм варіантом.

**Завдання №5 VNS Lab 9**

4.

1) Скопіювати з файлу F1 у файл F2 рядки, починаючи з 4.

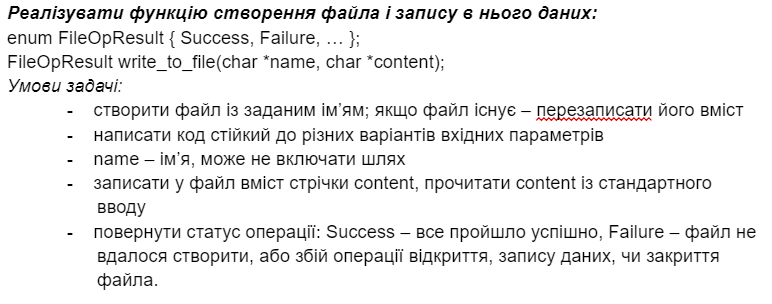
2)Підрахувати кількість символів в останньому слові F2

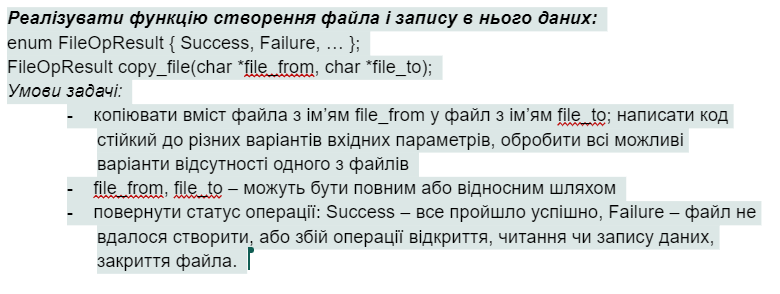
****.**Завдання №6 Algotester Lab 4**

**Завдання №7 Algotester Lab 6**

**Завдання №8 Class Practice Task**

* Деталі завдання :

1) Реалізувати функцію створення файла і запису в нього даних

****2) Реалізувати функцію створення файла і запису в нього даних

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

**Програма №1 VNS Lab 8**

* Планований час на реалізацію: 2 години

**Програма №2 VNS Lab 9**

* Планований час на реалізацію: 3 години

**Програма №3 Algotester Lab 4**

* Планований час на реалізацію: 3 години

**Програма №4 VNS Lab 6**

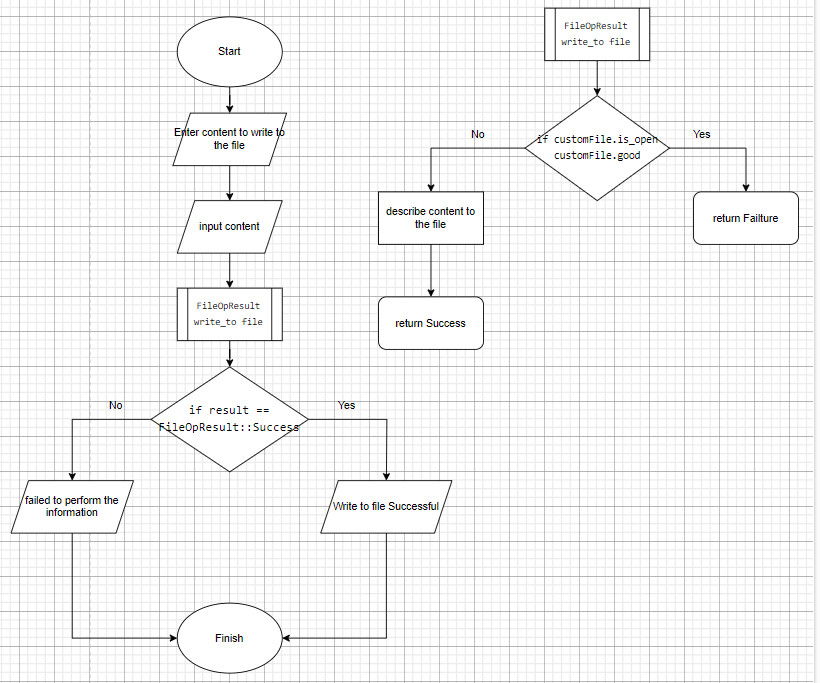
* Планований час на реалізацію: 5 годин

**Програма №5 Algotester Lab 6**

* Планований час на реалізацію: 12 годин

-

**Програма №6 Class Practice Task**

* Планований час на реалізацію: 3 години
* Блок схема:
* 

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

**Завдання №1 VNS Lab 8**

#include <iostream>

#include <fstream>

#include <cstring>

using namespace std;

// Структура для представлення інформації про людину

struct Person {

    char lastName[50];

    char firstName[50];

    char patronymic[50];

    char address[100];

    char phoneNumber[15];

    int age;

};

// Функція для виведення вмісту бінарного файлу

void printBinaryFileContents(const char\* filename) {

    ifstream inFile(filename, ios::binary);

    if (!inFile.is\_open()) {

        cerr << "Неможливо відкрити файл: " << filename << endl;

        return;

    }

    Person person;

    while (inFile.read(reinterpret\_cast<char\*>(&person), sizeof(person))) {

        cout << "Прізвище: " << person.lastName << "\n";

        cout << "Ім'я: " << person.firstName << "\n";

        cout << "По батькові: " << person.patronymic << "\n";

        cout << "Адреса: " << person.address << "\n";

        cout << "Номер телефону: " << person.phoneNumber << "\n";

        cout << "Вік: " << person.age << "\n\n";

    }

    inFile.close();

}

// Функція для видалення записів за вказаним віком

void deleteByAge(const char\* filename, int ageToDelete) {

    ifstream inFile(filename, ios::binary);  //Створює об'єкт для читання з файлу (ios::binary вказує, що файл є бінарним).

    ofstream outFile("temp.bin", ios::binary);   //Створює об'єкт для запису у новий тимчасовий файл temp.bin

    if (!inFile.is\_open() || !outFile.is\_open()) {

        cerr << "Неможливо відкрити файл для видалення." << endl;

        return;

    }

    Person person;  //Створюється об'єкт person типу Person, який використовується для зчитування даних з вхідного файлу.

    while (inFile.read(reinterpret\_cast<char\*>(&person), sizeof(person))) {   //зчитуються блоки даних розміром sizeof(person) з вхідного файлу

        if (person.age != ageToDelete) {

            outFile.write(reinterpret\_cast<char\*>(&person), sizeof(person)); //Кожний зчитаний запис порівнюється за віком з ageToDelete, і якщо вік не збігається, запис додається до тимчасового файлу.

        }

    }

    inFile.close();

    outFile.close();

    remove(filename);

    rename("temp.bin", filename);

    cout << "Особи з віком " << ageToDelete << " успішно видалено.\n";

}

// Функція для додавання запису після конкретного індексу

void addAfterIndex(const char\* filename, int index, const Person& newPerson) {

    ifstream inFile(filename, ios::binary);

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

    if (!inFile.is\_open() || !outFile.is\_open()) {

        cerr << "Неможливо відкрити файл для додавання." << endl;

        return;

    }

    Person person;

    int currentIndex = 0;

    while (inFile.read(reinterpret\_cast<char\*>(&person), sizeof(person))) {  //зчитуються блоки даних розміром sizeof(person) з вхідного файлу

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

        if (currentIndex == index) {

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

        }

        currentIndex++;   //Кожний зчитаний запис записується у тимчасовий файл. Якщо досягнутий індекс, на якому ми хочемо вставити новий запис, цей новий запис також додається у тимчасовий файл.

    }

    inFile.close();

    outFile.close();

    remove(filename);

    rename("temp.bin", filename);

    cout << "Особу додано після індексу " << index << " успішно.\n";

}

// Функція для заповнення файлу інформацією про людей

void fillFile(const char\* filename, int count) {

    Person person;

    ofstream file(filename, ios::binary);

    if (!file.is\_open()) {

        cerr << "Помилка відкриття файлу для запису: " << filename << endl;

        return;

    }

    for (int i = 0; i < count; ++i) {

        cout << "\nВведіть дані для особи " << i + 1 << ":\n";

        cout << "Прізвище: ";

        cin.getline(person.lastName, sizeof(person.lastName));

        cout << "Ім'я: ";

        cin.getline(person.firstName, sizeof(person.firstName));

        cout << "По батькові: ";

        cin.getline(person.patronymic, sizeof(person.patronymic));

        cout << "Адреса: ";

        cin.getline(person.address, sizeof(person.address));

        cout << "Номер телефону: ";

        cin.getline(person.phoneNumber, sizeof(person.phoneNumber));

        cout << "Вік: ";

        cin >> person.age;

        cin.ignore(); // Очищення символу нового рядка з буфера

        file.write(reinterpret\_cast<const char\*>(&person), sizeof(person)); //Викликається file.write, який записує об'єкт person у бінарний файл;перетворення типу об'єкта на покажчик на const char, щоб його можна було записати у бінарний файл.

    }

    file.close();

}

int main() {

    const char\* filename = "people\_data.bin";

    int count = 3;

    fillFile(filename, count);

    cout << "\nПочатковий вміст файлу:\n";

    printBinaryFileContents(filename);

    int ageToDelete;

    cout << "\nВведіть вік для видалення: ";

    cin >> ageToDelete;

    deleteByAge(filename, ageToDelete);

    cout << "\nВміст файлу після видалення:\n";

    printBinaryFileContents(filename);

    int insertIndex;

    cout << "\nВведіть індекс для вставки після: ";

    cin >> insertIndex;

    cin.ignore(); // Очищення символу нового рядка з буфера

    Person newPerson;

    cout << "\nВведіть дані для нової особи:\n";

    cout << "Прізвище: ";

    cin.getline(newPerson.lastName, sizeof(newPerson.lastName));

    cout << "Ім'я: ";

    cin.getline(newPerson.firstName, sizeof(newPerson.firstName));

    cout << "По батькові: ";

    cin.getline(newPerson.patronymic, sizeof(newPerson.patronymic));

    cout << "Адреса: ";

    cin.getline(newPerson.address, sizeof(newPerson.address));

    cout << "Номер телефону: ";

    cin.getline(newPerson.phoneNumber, sizeof(newPerson.phoneNumber));

    cout << "Вік: ";

    cin >> newPerson.age;

    addAfterIndex(filename, insertIndex, newPerson);

    cout << "\nВміст файлу після додавання:\n";

    printBinaryFileContents(filename);

    return 0;

}

**Завдання №2 VNS Lab 9**

#include <iostream>

#include <fstream>

#include <string>

#include <vector>

using namespace std;

int main() {

    // Задання імен файлів

    string f1FileName = "F1";

    string f2FileName = "F2";

    // Створення та заповнення текстового файлу F1

    ofstream f1(f1FileName);

    if (!f1.is\_open()) {

        cerr << "Помилка відкриття файлу " << f1FileName << endl;

        return 1;

    }

    cout << "Введіть 10 рядків для файлу F1:\n";

    for (int i = 0; i < 10; ++i) {

        string input;

        cout << "Рядок " << i + 1 << ": ";

        getline(cin, input);

        f1 << input << endl;

    }

    f1.close();

    // Завдання 1: Копіювання рядків з F1 у F2, починаючи з 4-го рядка

    ifstream f1Read(f1FileName);

    ofstream f2(f2FileName);

    if (!f1Read.is\_open() || !f2.is\_open()) {

        cerr << "Помилка відкриття файлів для завдання 1." << endl;

        return 1;

    }

    string line;

    int lineCount = 0;

    while (getline(f1Read, line)) {

        lineCount++;

        if (lineCount >= 4) {

            f2 << line << endl;

        }

    }

    f1Read.close();

    f2.close();

    cout << "Завдання 1 виконано. Результат збережено у файлі " << f2FileName << endl;

    // Завдання 2: Підрахунок кількості символів в останньому слові F2

    ifstream f2Read(f2FileName);

    if (!f2Read.is\_open()) {

        cerr << "Помилка відкриття файлу " << f2FileName << endl;

        return 1;

    }

    string lastLine;

    while (getline(f2Read, line)) {

        lastLine = line;

    }

    f2Read.close();

    // Знаходження останнього слова

    size\_t lastSpace = lastLine.find\_last\_of(' ');

    if (lastSpace != string::npos) {

        string lastWord = lastLine.substr(lastSpace + 1);

        // Виведення результату

        cout << "Останнє слово в F2: " << lastWord << endl;

        cout << "Кількість символів в останньому слові F2: " << lastWord.length() << endl;

    } else {

        cerr << "Помилка: рядок не містить пробілів." << endl;

    }

    return 0;

}

**Завдання №3 Algotester Lab 4**

#include <iostream>

#include <vector>

#include <algorithm>

int main() {

    // Введення розміру масиву та параметра K

    int size, k;

    std::cin >> size >> k;

    // Введення елементів масиву

    std::vector<int> N(size);

    for (int i = 0; i < size; ++i) {

        std::cin >> N[i];

    }

    // Сортування та видалення дублікатів

    std::sort(N.begin(), N.end());

    auto it = std::unique(N.begin(), N.end());

    N.erase(it, N.end());

    // Обертання масиву на K

    k = (k > N.size()) ? k % N.size() : k;

    std::rotate(N.begin(), N.begin() + k, N.end());

    // Виведення результату

    std::cout << N.size() << std::endl;

    for (const auto &element : N) {

        std::cout << element << " ";

    }

    std::cout << std::endl;

    return 0;

}

**Завдання №4 VNS Lab 6**

#include <iostream>

#include <vector>

using namespace std;

int main() {

    string inputString;

    cout << "Введіть рядок: ";

    getline(cin, inputString);

    vector<char> result;

    for (char ch : inputString) {

        if (isalpha(ch)) {

            result.push\_back(ch);

        }

    }

    for (char ch : inputString) {

        if (isdigit(ch)) {

            result.push\_back(ch);

        }

    }

    cout << "Відформатований рядок: ";

    for (char ch : result) {

        cout << ch;

    }

    return 0;

}

**Завдання №5 Algotester Lab 6**

#include <iostream>

#include <vector>

#include <algorithm>

struct ChessPiece {

    char type;

    int x, y;

    ChessPiece(char t, int xpos, int ypos) : type(t), x(xpos), y(ypos) {}

};

bool isOccupied(int x, int y, const std::vector<ChessPiece>& pieces) {

    return std::any\_of(pieces.begin(), pieces.end(), [x, y](const ChessPiece& piece) {

        return piece.x == x && piece.y == y;

    });

}

bool isPawn(int x, int y, const std::vector<ChessPiece>& pieces) {

    return std::any\_of(pieces.begin(), pieces.end(), [x, y](const ChessPiece& piece) {

        return piece.type == 'P' && ((x - 1 == piece.x && y - 1 == piece.y) || (x - 1 == piece.x && y + 1 == piece.y));

    });

}

bool isRook(int x, int y, const std::vector<ChessPiece>& pieces) {

    return std::any\_of(pieces.begin(), pieces.end(), [x, y](const ChessPiece& piece) {

        return piece.type == 'R' && (x == piece.x || y == piece.y);

    });

}

bool isKnight(int x, int y, const std::vector<ChessPiece>& pieces) {

    int knight\_dx[] = {-2, -1, 1, 2, 2, 1, -1, -2};

    int knight\_dy[] = {1, 2, 2, 1, -1, -2, -2, -1};

    for (int i = 0; i < 8; ++i) {

        int nx = x + knight\_dx[i];

        int ny = y + knight\_dy[i];

        if (std::any\_of(pieces.begin(), pieces.end(), [nx, ny](const ChessPiece& piece) {

                return piece.type == 'N' && piece.x == nx && piece.y == ny;

            })) {

            return true;

        }

    }

    return false;

}

bool isBishop(int x, int y, const std::vector<ChessPiece>& pieces) {

    int dx[] = {-1, -1, 1, 1};

    int dy[] = {-1, 1, -1, 1};

    for (int k = 0; k < 4; ++k) {

        for (int j = 1; j < 8; ++j) {

            int nx = x + dx[k] \* j;

            int ny = y + dy[k] \* j;

            if (std::any\_of(pieces.begin(), pieces.end(), [nx, ny](const ChessPiece& piece) {

                    return piece.type == 'B' && piece.x == nx && piece.y == ny;

                })) {

                return true;

            }

        }

    }

    return false;

}

bool isKing(int x, int y, const std::vector<ChessPiece>& pieces) {

    int king\_dx[] = {-1, -1, -1, 0, 0, 1, 1, 1};

    int king\_dy[] = {-1, 0, 1, -1, 1, -1, 0, 1};

    for (int i = 0; i < 8; ++i) {

        int nx = x + king\_dx[i];

        int ny = y + king\_dy[i];

        if (std::any\_of(pieces.begin(), pieces.end(), [nx, ny](const ChessPiece& piece) {

                return piece.type == 'K' && piece.x == nx && piece.y == ny;

            })) {

            return true;

        }

    }

    return false;

}

bool isQueen(int x, int y, const std::vector<ChessPiece>& pieces) {

    return std::any\_of(pieces.begin(), pieces.end(), [x, y](const ChessPiece& piece) {

        return piece.type == 'Q' && (x == piece.x || y == piece.y || std::abs(x - piece.x) == std::abs(y - piece.y));

    });

}

int main() {

    std::vector<ChessPiece> pieces;

    for (int i = 0; i < 8; ++i) {

        std::string row;

        std::cin >> row;

        for (int j = 0; j < 8; ++j) {

            if (row[j] != 'O') {

                pieces.push\_back(ChessPiece(row[j], i, j));

            }

        }

    }

    int Q;

    std::cin >> Q;

    std::vector<std::pair<int, int>> queries;

    for (int m = 0; m < Q; m++) {

        int a, b;

        std::cin >> a >> b;

        queries.push\_back({a - 1, b - 1});

    }

    for (const auto& query : queries) {

        int x = query.first;

        int y = query.second;

        std::string answer;

        if (isOccupied(x, y, pieces)) {

            std::cout << "X" << std::endl;

        } else {

            for (char figure : "PRNBKQ") {

                switch (figure) {

                    case 'P':

                        if (isPawn(x, y, pieces)) {

                            answer += figure;

                        }

                        break;

                    case 'R':

                        if (isRook(x, y, pieces)) {

                            answer += figure;

                        }

                        break;

                    case 'N':

                        if (isKnight(x, y, pieces)) {

                            answer += figure;

                        }

                        break;

                    case 'B':

                        if (isBishop(x, y, pieces)) {

                            answer += figure;

                        }

                        break;

                    case 'K':

                        if (isKing(x, y, pieces)) {

                            answer += figure;

                        }

                        break;

                    case 'Q':

                        if (isQueen(x, y, pieces)) {

                            answer += figure;

                        }

                        break;

                }

            }

            if (answer.empty()) {

                std::cout << "O" << std::endl;

            } else {

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

                std::cout << answer << std::endl;

            }

        }

    }

    return 0;

}

**Завдання №6 Class Practice Task 1**

#include <iostream>

#include <fstream>

#include <string>

enum FileOpResult { Success, Failure };

class FileWriter {

public:

    FileWriter(const std::string &fileName) : fileName(fileName) {}

    FileOpResult writeToCustomFile(const std::string &content) {

        std::ofstream customFile(fileName);

        if (!customFile.is\_open()) {

            std::cerr << "Помилка відкриття файлу " << fileName << std::endl;

            return FileOpResult::Failure;

        }

        customFile << content;

        if (!customFile.good()) {

            std::cerr << "Помилка запису в файл " << fileName << std::endl;

            customFile.close();

            return FileOpResult::Failure;

        }

        customFile.close();

        return FileOpResult::Success;

    }

private:

    std::string fileName;

};

int main() {

    std::string customFileName = "custom\_example.txt";

    std::string fileContent;

    std::cout << "Введіть вміст для запису в файл: ";

    std::getline(std::cin, fileContent);

    FileWriter fileWriter(customFileName);

    FileOpResult result = fileWriter.writeToCustomFile(fileContent);

    if (result == FileOpResult::Success) {

        std::cout << "Success" << std::endl;

    } else {

        std::cout << "Failure" << std::endl;

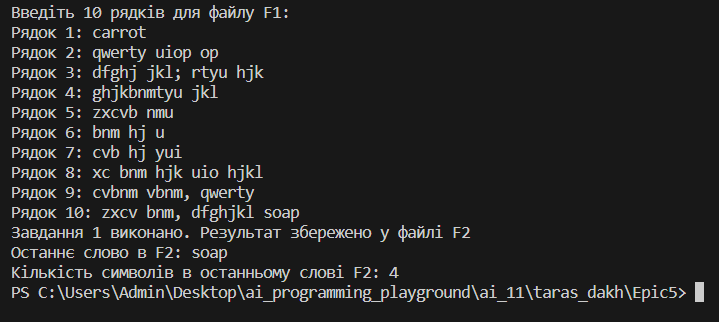
    }

    return 0;

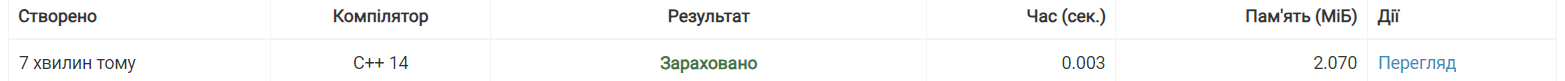
}

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

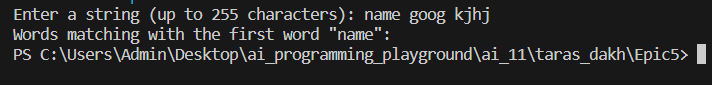
**Завдання №1** **VNS Lab 9**

****

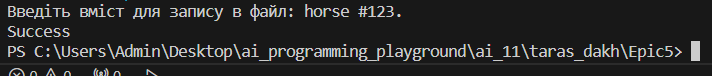
**Завдання №3 Algotester Lab 4**



**Завдання №4 VNS Lab 6**

****

**Завдання №6 Class Practice Task**

****

**Завдання №7 Algotester lab6**

****

# **Висновки:**

В ході виконання лабораторних та практичних завдань, що стосувалися взаємодії з двійковими файлами та операцій введення-виведення рядків, мені вдалося успішно ознайомитися з організацією обміну структурованою інформацією через зовнішні носії. Робота над цими етапами дозволила мені отримати практичні навички роботи з текстовими файлами, виконанням операцій введення-виведення текстової інформації, а також зберіганням даних на зовнішніх носіях. Цей досвід виявився надзвичайно корисним, допомагаючи зрозуміти процес зберігання та обробки інформації в різних форматах, зокрема, вивчити особливості роботи з двійковими файлами та текстовими рядками.