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

Звіт

про виконання лабораторних та практичних робіт блоку № 5

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

з дисципліни: «Основи програмування»

до:

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

Виконала:
Студентка групи ШІ-13
Осінна Єлизавета Сергіївна

Львів 2024

Тема роботи:

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

Мета роботи:

Вивчення символьних і рядкових змінних і способів їхньої обробки в мові С. Робота із двійковими файлами, організація вводу-виводу структурованої інформації і її зберігання на зовнішніх носіях. Робота з текстовими файлами, ввід-вивід текстової інформації і її зберігання на зовнішніх носіях. Освоїти практичні навики роботи з файлами з використанням стандартної бібліотеки C++, навчитись використовувати методи відкриття файла, запису масиву даних у файл, закриття файла та обробки помилок чи станів операції на кожному з етапів.

Теоретичні відомості:

1) Теоретичні відомості з переліком важливих тем:

1. 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
2. Символи і Рядкові Змінні:
    - Робота з char та string: основні операції і методи
    - Стрічкові літерали та екранування символів
    - Конкатенація, порівняння та пошук у рядках
3. 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
4. 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
5. 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
6. Створення й використання бібліотек:
    - Вступ до створення власних бібліотек у C++
    - 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:

- C++ Конкатенація рядків - W3Schools українською
- Базовий файловий ввід і вивід в C++ / aCode
- getline у C++: читання рядка
- Уроки C++ для початківців / #13 – Робота з файлами за допомогою C++
- 13 – Робота з файлами за допомогою C++

- Тема №*.1: Назва.
    - Джерела Інформації
        - Книжка.
        - Відео.
        - Стаття.
        - Курс.
    - Що опрацьовано:
        - Коментар 1
        - Коментар 2
    - Статус: Ознайомлений/ Ознайомлений частково / Не ознайомлений
    - Початок опрацювання теми: Дата
    - Звершення опрацювання теми: Дата

Виконання роботи:

1. Опрацювання завдання та вимог до програм та середовища:

Завдання № 1 VNS Lab 6

- Варіант 5
- 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. Текст містить не більше 255 символів. Виконати ввід рядка, використовуючи функцію gets(s) і здійснити обробку рядка у відповідності зі своїм варіантом.

5. Перетворити рядок таким чином, щоб спочатку в ньому були надруковані тільки букви, а потім тільки цифри, не міняючи порядку проходження символів у рядку.

Завдання № 2 VNS Lab 8

- Варіант 5
- Деталі завдання:

Сформувати двійковий файл із елементів, заданої у варіанті структури, роздрукувати його вміст, 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 "Людина":
- прізвище, ім'я, по батькові;
- рік народження;

- ріст;
- вага.

Знищити усі елементи із зазначеним ростом і вагою, додати елемент після елемента із зазначеним прізвищем.

Завдання № 3 VNS Lab 9

- Варіант 5
- Деталі завдання:

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

5.
1) Скопіювати з файлу F1 у файл F2 рядки, починаючи з K до K+5.
2) Підрахувати кількість голосних букв у файлі F2.

Завдання № 4 Algotester Lab 4

- Варіант 1
- Деталі завдання:

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

Вхідні дані
У першому рядку ціле число N - розмір масиву 1
У другому рядку N цілих чисел - елементи масиву 1
У третьому рядку ціле число M - розмір масиву 2
У четвертом рядку M цілих чисел - елементи масиву 2

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

Обмеження
$1 \leq N, M \leq 100$
$1 \leq ni, mi \leq 100$

Примітки
Пам’ятайте, ви маєте написати 2 варіанти розвязку, один з використанням засобів STL (std::set_intersection, std::set_symmetric_difference, std::set_difference, std::set_union), інший зі своєю реалізацією. Своє сортування можна не писати.

Завдання № 5 Algotester Lab 4

- Варіант 3
- Деталі завдання:

Вам дано масив, який складається з N додатніх цілих чисел.

Ваше завдання - розділити його на три частини, по остачі від ділення на 3, по зростанню остачі (тобто спочатку йдуть числа, у яких остача 0, далі числа з остачею 1 і тоді нарешті числа з остачею 2).
Далі необхідно ті елементи, остача від ділення на 3 яких парна посортувати по зростанню, а ті, у яких остача 1 - по спаданню.
Після цього видаліть усі дублікати з масиву.
Виведіть результуючий масив.

Вхідні дані
У першому рядку N - кількість чисел.
У другому рядку N чисел ai - елементи масиву.

Вихідні дані
У першому рядку M - кількість чисел у масиву
У другому рядку M посоротованих за умовою чисел.

Обмеження
$1 \le N \le 103$
$0 \le ai \le 103$

Примітки
Пам’ятайте, ви маєте написати 2 варіанти розвязку, один з використанням засобів STL (власноруч написаний компаратор або std::partition + std::sort + std::unique), інший зі своєю реалізацією. Алгоритм сортування можна вибрати будь який, окрім сортування бульбашкою і має працювати за N*logN часу.

Завдання № 6 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.

Вхідні дані
У перших 8 рядках стрічка rowi - стан i-го рядка дошки.
У наступному рядку ціле число Q - кількість записів

У наступних Q рядках 2 цілих числа x та y - координати клітинки

Вихідні дані
Q разів відповідь у наступному форматі:
Строка result - усі фігури, які атакують клітинку з запиту.

Обмеження
|rowi| = N
rowi ∈ {O, P, R, N, B, K, Q}
$1 \le Q \le 64$
$1 \le x, y \le 8$

Завдання № 7 Class Practice Work

- Деталі завдання:

Задача №1 – Запис текстової стрічки у файл із заданим ім'ям

Реалізувати функцію створення файла і запису в нього даних:
enum FileOpResult { Success, Failure, … };
FileOpResult write_to_file(char *name, char *content);
Умови задачі:

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

Задача №2 – Копіювання вмісту файла у інший файл

Реалізувати функцію створення файла і запису в нього даних:
enum FileOpResult { Success, Failure, … };
FileOpResult copy_file(char *file_from, char *file_to);
Умови задачі:

- копіювати вміст файла з ім'ям file_from у файл з ім'ям file_to; написати код стійкий до різних варіантів вхідних параметрів, обробити всі можливі варіанти відсутності одного з файлів
- file_from, file_to – можуть бути повним або відносним шляхом
- повернути статус операції: Success – все пройшло успішно, Failure – файл не вдалося створити, або збій операції відкриття, читання чи запису даних, закриття файла.

Завдання № 8 Self Practice Work

- Деталі завдання:

Lab 6v1

Вам дано N слів та число K.
Ваше завдання перечислити букви в словах, які зустрічаються в тексті більше-рівне ніж K разів (саме слово, не буква!).
Великі та маленькі букви вважаються однаковими, виводити необхідно малі, посортовані від останьої до першої у алфавіті. Букву потрібно виводити лише один раз.

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

Input

Цілі числа N та K - загальна кількість слів та мінімальна кількість слів щоб враховувати букви цього слова в результаті.
N стрічок s

Output

У першому рядку ціле число MM - кількість унікальних букв
У другому рядку унікальні букви через пробіли

Constraints

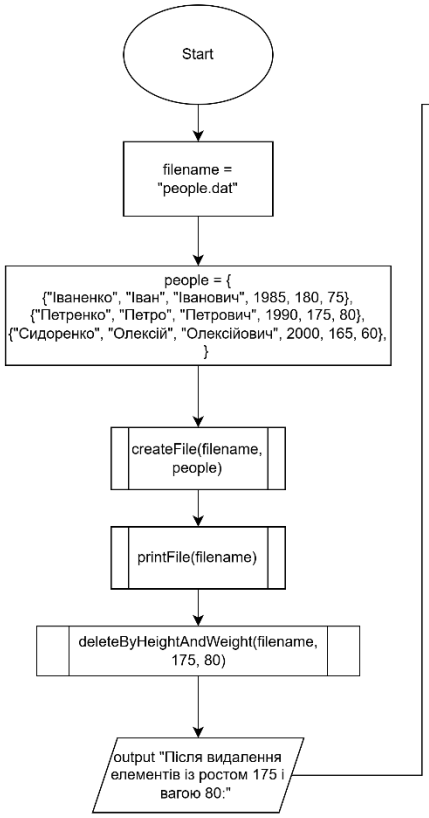
$1 \le K \le N \le 10^5$
$1 \le |s_i| \le 10$
$s_i \in a..Z$

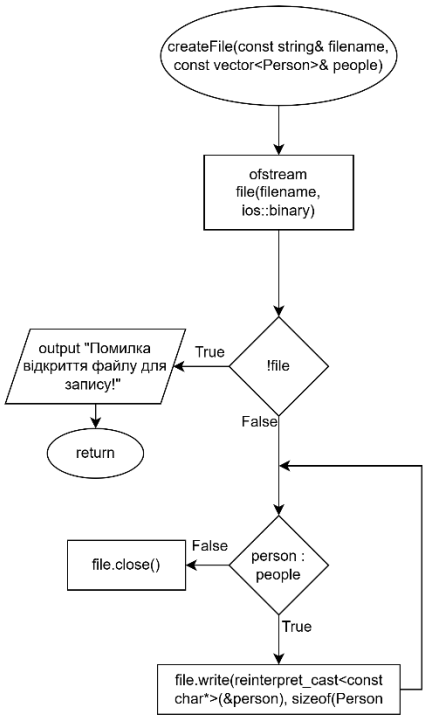
2. Дизайн та планована оцінка часу виконання завдань:

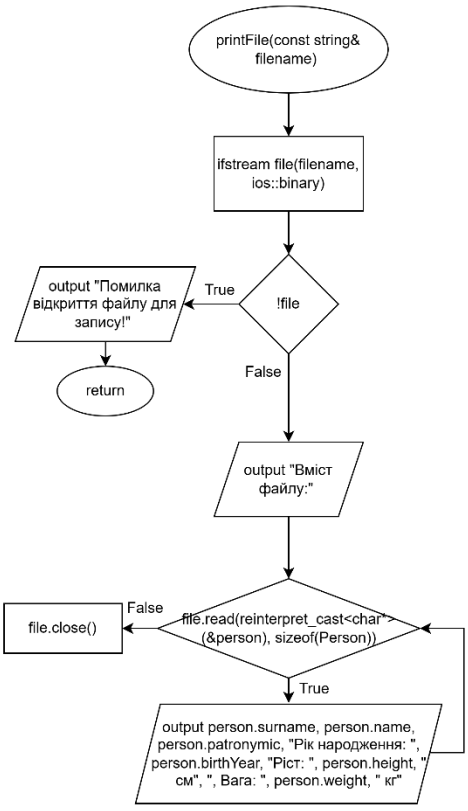
Планований час на реалізацію кожного завдання: 40 хв

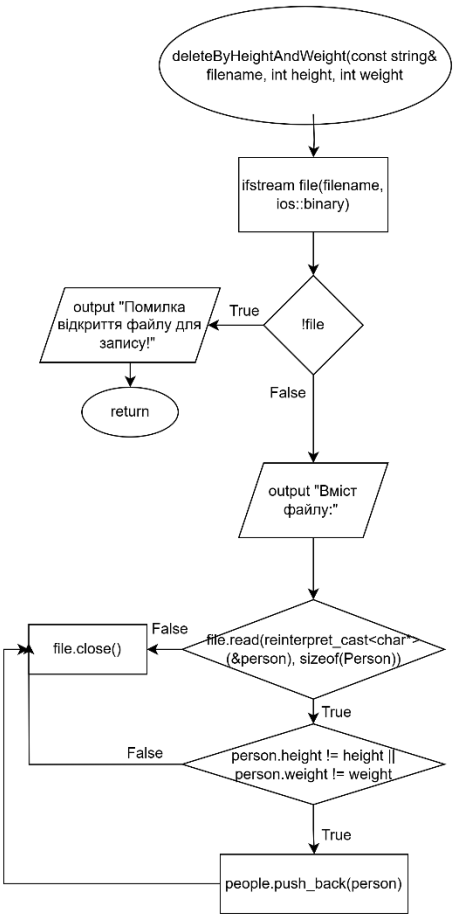
Програма № 2 VNS Lab 8

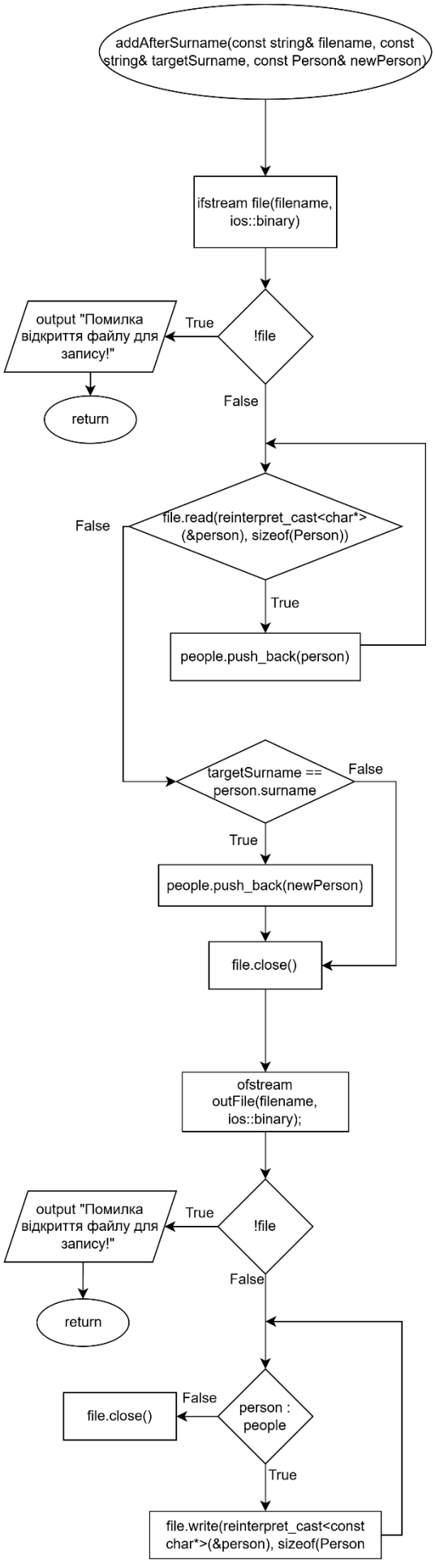
- Блок-схема


Start
filename = "people.dat"
people = { {"Іваненко", "Іван", "Іванович", 1985, 180, 75}, {"Петренко", "Петро", "Петрович", 1990, 175, 80}, {"Сидоренко", "Олексій", "Олексійович", 2000, 165, 60}, }
createFile(filename, people)
printFile(filename)
deleteByHeightAndWeight(filename, 175, 80)
output "Після видалення елементів із ростом 175 і вагою 80:"
printFile(filename)
newPerson = {"Новаченко", "Наталія", "Володимирівна", 1995, 170, 65};
addAfterSurname(filename, "Іваненко", newPerson);
output "Після додавання нового елемента після 'Іваненко':"
printFile(filename)
The end


createFile(const string& filename, const vector<Person>& people)
ofstream file(filename, ios::binary)
!file
True
output "Помилка відкриття файлу для запису!"
return
False
person : people
False
file.close()
True
file.write(reinterpret_cast<const char*>(&person), sizeof(Person


printFile(const string& filename)
ifstream file(filename, ios::binary)
!file
True
output "Помилка відкриття файлу для запису!"
return
False
output "Вміст файлу:"
file.read(reinterpret_cast<char*>(&person), sizeof(Person))
False
file.close()
True
output person.surname, person.name, person.patronymic, "Рік народження: ", person.birthYear, "Ріст: ", person.height, " см", ", Вага: ", person.weight, " кг"


deleteByHeightAndWeight(const string& filename, int height, int weight
ifstream file(filename, ios::binary)
!file
True
output "Помилка відкриття файлу для запису!"
return
False
output "Вміст файлу:"
file.read(reinterpret_cast<char*>(&person), sizeof(Person))
False
file.close()
True
person.height != height || person.weight != weight
False
True
people.push_back(person)


addAfterSurname(const string& filename, const string& targetSurname, const Person& newPerson)
ifstream file(filename, ios::binary)
!file
True
output "Помилка відкриття файлу для запису!"
return
False
file.read(reinterpret_cast<char*>(&person), sizeof(Person))
False
True
people.push_back(person)
targetSurname == person.surname
False
True
people.push_back(newPerson)
file.close()
ofstream outFile(filename, ios::binary);
!file
True
output "Помилка відкриття файлу для запису!"
return
False
person : people
False
file.close()
True
file.write(reinterpret_cast<const char*>(&person), sizeof(Person

4. Код програм з посиланням на зовнішні ресурси:

Завдання № 1 VNS Lab 6

```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
    const int MAX_LEN = 255; // Максимальна довжина рядка
    char s[MAX_LEN + 1]; // Буфер для вводу рядка (+1 для символу завершення '\0')

    cout << "Введіть рядок (до 255 символів), що закінчується крапкою:" << endl;
    cin.getline(s, MAX_LEN);

    string letters = ""; // Рядок для літер
    string digits = "";  // Рядок для цифр

    // Обробка рядка
    for (int i = 0; s[i] != '\0'; ++i) {
        if (isalpha(s[i])) {
            letters += s[i]; // Додаємо до рядка літер
        } else if (isdigit(s[i])) {
            digits += s[i];  // Додаємо до рядка цифр
        }
    }

    // Об'єднання результату
    string result = letters + digits;
    cout << "Перетворений рядок: " << result << endl;

    return 0;
}
```

Результат:

```
ebugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-djgcz1zr.gwj' '--s
rosoft-MIEngine-Error-cuyra0gs.opj' '--pid=Microsoft-MIEngine-Pid-cm1dsqqy.4tj' '--dbgExe=C:
Введіть рядок (до 255 символів), що закінчується крапкою:
rjrh7 u89r 9r 84.
Перетворений рядок: rjrhurr789984
PS D:\AI\ai_programming_playground_2024\ai_13\yelyzaveta_osinna\epic_5>
```

Час затрачений на виконання завдання 30 хв

Завдання № 2 VNS Lab 8

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

```cpp
#include <string>
using namespace std;

// Опис структури "Людина"
struct Person {
    char surname[50];
    char name[50];
    char patronymic[50];
    int birthYear;
    int height;
    int weight;
};

// Функція для створення двійкового файлу
void createFile(const string& filename, const vector<Person>& people) {
    ofstream file(filename, ios::binary);
    if (!file) {
        cerr << "Помилка відкриття файлу для запису!" << endl;
        return;
    }
    for (const auto& person : people) {
        file.write(reinterpret_cast<const char*>(&person), sizeof(Person));
    }
    file.close();
}

// Функція для друку вмісту двійкового файлу
void printFile(const string& filename) {
    ifstream file(filename, ios::binary);
    if (!file) {
        cerr << "Помилка відкриття файлу для читання!" << endl;
        return;
    }
    Person person;
    cout << "Вміст файлу:" << endl;
    while (file.read(reinterpret_cast<char*>(&person), sizeof(Person))) {
        cout << person.surname << " " << person.name << " " << person.patronymic
             << ", Рік народження: " << person.birthYear
             << ", Ріст: " << person.height << " см"
             << ", Вага: " << person.weight << " кг" << endl;
    }
    file.close();
}

// Функція для видалення елементів за заданим критерієм
void deleteByHeightAndWeight(const string& filename, int height, int weight) {
    ifstream file(filename, ios::binary);
    if (!file) {
        cerr << "Помилка відкриття файлу для читання!" << endl;
        return;
```

```
    }

    vector<Person> people;
    Person person;

    // Зчитуємо всі елементи, які не відповідають критеріям
    while (file.read(reinterpret_cast<char*>(&person), sizeof(Person))) {
        if (person.height != height || person.weight != weight) {
            people.push_back(person);
        }
    }
    file.close();

    // Перезаписуємо файл без видалених елементів
    ofstream outFile(filename, ios::binary);
    if (!outFile) {
        cerr << "Помилка відкриття файлу для запису!" << endl;
        return;
    }
    for (const auto& p : people) {
        outFile.write(reinterpret_cast<const char*>(&p), sizeof(Person));
    }
    outFile.close();
}

// Функція для додавання елемента після заданого прізвища
void addAfterSurname(const string& filename, const string& targetSurname, const
Person& newPerson) {
    ifstream file(filename, ios::binary);
    if (!file) {
        cerr << "Помилка відкриття файлу для читання!" << endl;
        return;
    }

    vector<Person> people;
    Person person;

    // Зчитуємо всі елементи
    while (file.read(reinterpret_cast<char*>(&person), sizeof(Person))) {
        people.push_back(person);
        // Додаємо нового елемента після знайденого прізвища
        if (targetSurname == person.surname) {
            people.push_back(newPerson);
        }
    }
    file.close();

    // Перезаписуємо файл з оновленими даними
    ofstream outFile(filename, ios::binary);
    if (!outFile) {
```

```
        cerr << "Помилка відкриття файлу для запису!" << endl;
        return;
    }
    for (const auto& p : people) {
        outFile.write(reinterpret_cast<const char*>(&p), sizeof(Person));
    }
    outFile.close();
}

// Головна функція для тестування
int main() {
    string filename = "people.dat";

    // Початкові дані
    vector<Person> people = {
        {"Іваненко", "Іван", "Іванович", 1985, 180, 75},
        {"Петренко", "Петро", "Петрович", 1990, 175, 80},
        {"Сидоренко", "Олексій", "Олексійович", 2000, 165, 60},
    };

    // Створення файлу
    createFile(filename, people);

    // Виведення вмісту файлу
    printFile(filename);

    // Видалення елементів з ростом 175 і вагою 80
    deleteByHeightAndWeight(filename, 175, 80);
    cout << "\nПісля видалення елементів із ростом 175 і вагою 80:" << endl;
    printFile(filename);

    // Додавання нового елемента після елемента з прізвищем "Іваненко"
    Person newPerson = {"Новаченко", "Наталія", "Володимирівна", 1995, 170, 65};
    addAfterSurname(filename, "Іваненко", newPerson);
    cout << "\nПісля додавання нового елемента після 'Іваненко':" << endl;
    printFile(filename);

    return 0;
}
```

Результат:

```
ebugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-wveykq3i.gxi' '--
rosoft-MIEngine-Error-m4ruyj3v.ok1' '--pid=Microsoft-MIEngine-Pid-x4cmvu2y.41a' '--dbgExe=C
Вміст файлу:
Іваненко Іван Іванович, Рік народження: 1985, Ріст: 180 см, Вага: 75 кг
Петренко Петро Петрович, Рік народження: 1990, Ріст: 175 см, Вага: 80 кг
Сидоренко Олексій Олексійович, Рік народження: 2000, Ріст: 165 см, Вага: 60 кг

Після видалення елементів із ростом 175 і вагою 80:
Вміст файлу:
Іваненко Іван Іванович, Рік народження: 1985, Ріст: 180 см, Вага: 75 кг
Сидоренко Олексій Олексійович, Рік народження: 2000, Ріст: 165 см, Вага: 60 кг

Після додавання нового елемента після 'Іваненко':
Вміст файлу:
Іваненко Іван Іванович, Рік народження: 1985, Ріст: 180 см, Вага: 75 кг
Новаченко Наталія Володимирівна, Рік народження: 1995, Ріст: 170 см, Вага: 65 кг
Сидоренко Олексій Олексійович, Рік народження: 2000, Ріст: 165 см, Вага: 60 кг
PS D:\AI\ai_programming_playground_2024\ai_13\yelyzaveta_osinna\epic_5>
```

Час затрачений на виконання завдання 1, 5 год

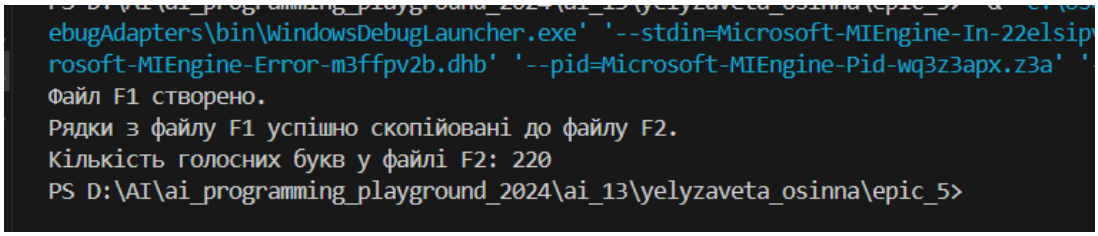
Завдання № 3 VNS Lab 9

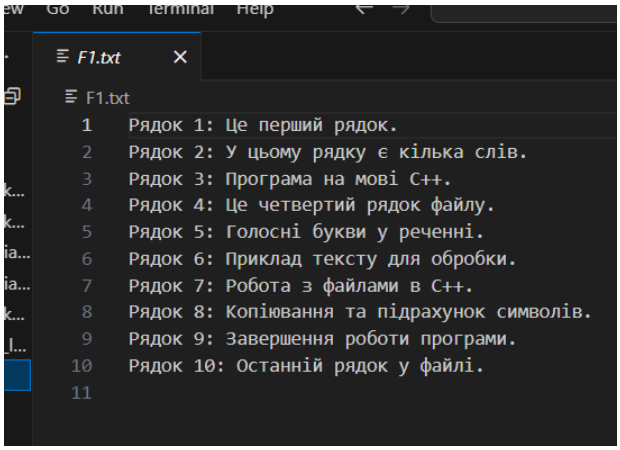
```cpp
#include <iostream>
#include <fstream>
#include <vector>
#include <cctype>
using namespace std;

// Функція для створення текстового файлу F1 з 10 рядками
void createFileF1(const string& filename) {
    ofstream file(filename);
    if (!file) {
        cerr << "Помилка відкриття файлу для запису!" << endl;
        return;
    }

    file << "Рядок 1: Це перший рядок.\n";
    file << "Рядок 2: У цьому рядку є кілька слів.\n";
    file << "Рядок 3: Програма на мові C++.\n";
    file << "Рядок 4: Це четвертий рядок файлу.\n";
    file << "Рядок 5: Голосні букви у реченні.\n";
    file << "Рядок 6: Приклад тексту для обробки.\n";
    file << "Рядок 7: Робота з файлами в C++.\n";
    file << "Рядок 8: Копіювання та підрахунок символів.\n";
    file << "Рядок 9: Завершення роботи програми.\n";
    file << "Рядок 10: Останній рядок у файлі.\n";

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

// Функція для копіювання рядків з K по K+5 з файлу F1 у файл F2
```

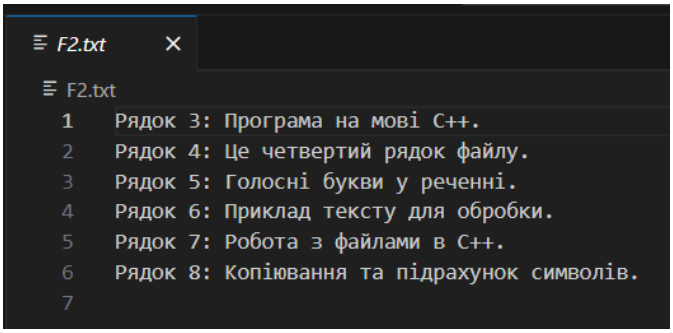
```
void copyLines(const string& fileF1, const string& fileF2, int K) {
    ifstream input(fileF1);
    ofstream output(fileF2);

    if (!input) {
        cerr << "Помилка відкриття файлу F1 для читання!" << endl;
        return;
    }
    if (!output) {
        cerr << "Помилка відкриття файлу F2 для запису!" << endl;
        return;
    }

    string line;
    int lineNumber = 0;

    while (getline(input, line)) {
        ++lineNumber;
        if (lineNumber >= K && lineNumber <= K + 5) {
            output << line << "\n";
        }
    }

    input.close();
    output.close();
    cout << "Рядки з файлу F1 успішно скопійовані до файлу F2.\n";
}

// Функція для підрахунку кількості голосних букв у файлі F2
int countVowels(const string& fileF2) {
    ifstream input(fileF2);
    if (!input) {
        cerr << "Помилка відкриття файлу F2 для читання!" << endl;
        return 0;
    }

    string vowels = "аеєиіїоуюяАЕЄИІЇОУЮЯaeiouAEIOU";
    string line;
    int vowelCount = 0;

    while (getline(input, line)) {
        for (char c : line) {
            if (vowels.find(c) != string::npos) {
                ++vowelCount;
            }
        }
    }

    input.close();
    return vowelCount;
```

```
}

// Головна функція для виконання програми
int main() {
    string fileF1 = "F1.txt";
    string fileF2 = "F2.txt";
    int K = 3; // Номер рядка, з якого починаємо копіювання

    // Створення файлу F1
    createFileF1(fileF1);

    // Копіювання рядків з K по K+5 у файл F2
    copyLines(fileF1, fileF2, K);

    // Підрахунок кількості голосних у файлі F2
    int vowelsCount = countVowels(fileF2);
    cout << "Кількість голосних букв у файлі F2: " << vowelsCount << endl;

    return 0;
}
```

Результат:







Час затрачений на виконання завдання 40 хв

Завдання № 4 Algotester Lab 4 Варіант 1

Спосіб 1:

```
#include <iostream>
```

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

int main()
{
    int N, M, element;
    cin >> N;

    vector<int> firstSet;
    for (int i = 0; i < N; i++)
    {
        cin >> element;
        firstSet.push_back(element);
    }

    cin >> M;
    vector<int> secondSet;
    for (int i = 0; i < M; i++)
    {
        cin >> element;
        secondSet.push_back(element);
    }

    cout << endl;

    // Сортування множин
    sort(firstSet.begin(), firstSet.end());
    sort(secondSet.begin(), secondSet.end());

    // Різниця: N - M
    vector<int> difference1;
    set_difference(firstSet.begin(), firstSet.end(),
                   secondSet.begin(), secondSet.end(),
                   inserter(difference1, difference1.end()));
    cout << difference1.size() << endl;
    for (int value : difference1)
    {
        cout << value << " ";
    }
    cout << endl;

    // Різниця: M - N
    vector<int> difference2;
    set_difference(secondSet.begin(), secondSet.end(),
                   firstSet.begin(), firstSet.end(),
                   inserter(difference2, difference2.end()));
    cout << difference2.size() << endl;
    for (int value : difference2)
    {
```

```
        cout << value << " ";
    }
    cout << endl;

    // Перетин множин
    vector<int> intersection;
    set_intersection(firstSet.begin(), firstSet.end(),
                     secondSet.begin(), secondSet.end(),
                     inserter(intersection, intersection.end()));
    cout << intersection.size() << endl;
    for (int value : intersection)
    {
        cout << value << " ";
    }
    cout << endl;

    // Об'єднання множин
    vector<int> unionSet;
    set_union(firstSet.begin(), firstSet.end(),
              secondSet.begin(), secondSet.end(),
              inserter(unionSet, unionSet.end()));
    cout << unionSet.size() << endl;
    for (int value : unionSet)
    {
        cout << value << " ";
    }
    cout << endl;

    // Симетрична різниця
    vector<int> symmetricDifference;
    set_symmetric_difference(firstSet.begin(), firstSet.end(),
                             secondSet.begin(), secondSet.end(),
                             inserter(symmetricDifference,
symmetricDifference.end()));
    cout << symmetricDifference.size() << endl;
    for (int value : symmetricDifference)
    {
        cout << value << " ";
    }
    cout << endl;

    return 0;
}
```

Результат:

```
ec5mhy.vrv' '--pid=Microsoft-MIEngine-
d-5l45psmv.dtq' '--dbgExe=C:\msys64\mi
w64\bin\gdb.exe' '--interpreter=mi'
5
1 0 2 5 4
4
0 2 3 7
3
1 4 5

2
3 7

2
0 2

7
0 1 2 3 4 5 7

5
1 3 4 5 7
```

Спосіб 2:

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

// Функція для знаходження різниці A - B
vector<int> difference(const vector<int>& A, const vector<int>& B) {
    vector<int> result;
    for (int a : A) {
        if (find(B.begin(), B.end(), a) == B.end()) { // Якщо a не в B
            result.push_back(a);
        }
    }
    sort(result.begin(), result.end());
    return result;
}

// Функція для знаходження перетину A ∩ B
vector<int> intersection(const vector<int>& A, const vector<int>& B) {
    vector<int> result;
    for (int a : A) {
        if (find(B.begin(), B.end(), a) != B.end()) { // Якщо a є в B
            if (find(result.begin(), result.end(), a) == result.end()) { // Уникнення
дублювань
                result.push_back(a);
            }
```

```
        }
    }
    sort(result.begin(), result.end());
    return result;
}

// Функція для об'єднання A ∪ B
vector<int> unionSet(const vector<int>& A, const vector<int>& B) {
    vector<int> result = A; // Починаємо з усіх елементів A
    for (int b : B) {
        if (find(A.begin(), A.end(), b) == A.end()) { // Якщо b не в A
            result.push_back(b);
        }
    }
    sort(result.begin(), result.end());
    return result;
}

// Функція для знаходження симетричної різниці A Δ B
vector<int> symmetricDifference(const vector<int>& A, const vector<int>& B) {
    vector<int> diff1 = difference(A, B); // A - B
    vector<int> diff2 = difference(B, A); // B - A
    vector<int> result = diff1;
    result.insert(result.end(), diff2.begin(), diff2.end()); // Поєднуємо A - B і B -
A
    sort(result.begin(), result.end());
    result.erase(unique(result.begin(), result.end()), result.end()); // Видаляємо
дублікати
    return result;
}

// Функція для друку результату
void printVector(const vector<int>& v) {
    cout << v.size() << "\n";
    for (int x : v) {
        cout << x << " ";
    }
    cout << "\n";
}

int main() {
    int N, M;

    // Зчитуємо розмір і елементи першого масиву
    cin >> N;
    vector<int> array1(N);
    for (int i = 0; i < N; ++i) {
        cin >> array1[i];
    }
```

```
    // Зчитуємо розмір і елементи другого масиву
    cin >> M;
    vector<int> array2(M);
    for (int i = 0; i < M; ++i) {
        cin >> array2[i];
    }

    // Обчислення множинних операцій
    vector<int> diff1 = difference(array1, array2);          // Різниця N - M
    vector<int> diff2 = difference(array2, array1);          // Різниця M - N
    vector<int> inter = intersection(array1, array2);        // Перетин
    vector<int> uni = unionSet(array1, array2);              // Об’єднання
    vector<int> symDiff = symmetricDifference(array1, array2); // Симетрична різниця

    // Виведення результатів
    printVector(diff1); // Різниця N - M
    printVector(diff2); // Різниця M - N
    printVector(inter); // Перетин
    printVector(uni);   // Об’єднання
    printVector(symDiff); // Симетрична різниця

    return 0;
}
```

Результат:

Час затрачений на виконання завдання 1 год

Завдання № 5 Algotester Lab 4 Варіант 3

Спосіб 1:

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

// Основна функція
int main() {
    int N;
    cin >> N;

    vector<int> array(N);
    for (int i = 0; i < N; ++i) {
        cin >> array[i];
    }

    // Розподіл на три групи: mod0, mod1, mod2
```

```
    auto modComparator = [](int a, int b) {
        if (a % 3 != b % 3) {
            return a % 3 < b % 3; // Спочатку за остачею
        }
        if (a % 3 == 1) {
            return a > b; // Остача 1 — за спаданням
        }
        return a < b; // Інші (остача 0 або 2) — за зростанням
    };

    // Сортуємо масив з урахуванням компаратора
    sort(array.begin(), array.end(), modComparator);

    // Видаляємо дублікати
    array.erase(unique(array.begin(), array.end()), array.end());

    // Вивід результату
    cout << array.size() << "\n";
    for (int num : array) {
        cout << num << " ";
    }
    cout << "\n";

    return 0;
}
```

Результат:

```
bi' '--dbgExe=C:\msys64\mingw64\bin\gdb.exe' '--interpreter=mi'
10
1 33 4 8 6 5 2 7 5 0
9
0 6 33 7 4 1 2 5 8
PS D:\AI\ai_programming_playground_2024\ai_13\yelyzaveta_osinna\epic_5>
```

Спосіб 2:

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

// Швидке сортування (QuickSort)
void quickSort(vector<int>& arr, int left, int right, auto comparator) {
    if (left >= right) return;

    int pivot = arr[(left + right) / 2];
    int i = left, j = right;

    while (i <= j) {
        while (comparator(arr[i], pivot)) ++i;
```

```
        while (comparator(pivot, arr[j])) --j;

        if (i <= j) {
            swap(arr[i], arr[j]);
            ++i;
            --j;
        }
    }

    if (left < j) quickSort(arr, left, j, comparator);
    if (i < right) quickSort(arr, i, right, comparator);
}

// Видалення дублікатів (власна реалізація)
vector<int> removeDuplicates(vector<int>& arr) {
    vector<int> unique;
    for (int i = 0; i < arr.size(); ++i) {
        if (i == 0 || arr[i] != arr[i - 1]) {
            unique.push_back(arr[i]);
        }
    }
    return unique;
}

// Основна функція
int main() {
    int N;
    cin >> N;

    vector<int> array(N);
    for (int i = 0; i < N; ++i) {
        cin >> array[i];
    }

    // Власний компаратор
    auto modComparator = [](int a, int b) {
        if (a % 3 != b % 3) {
            return a % 3 < b % 3; // Спочатку за остачею
        }
        if (a % 3 == 1) {
            return a > b; // Остача 1 — за спаданням
        }
        return a < b; // Інші (остача 0 або 2) — за зростанням
    };

    // Сортування з використанням власного QuickSort
    quickSort(array, 0, array.size() - 1, modComparator);

    // Видалення дублікатів
    vector<int> result = removeDuplicates(array);
```

```
    // Вивід результату
    cout << result.size() << "\n";
    for (int num : result) {
        cout << num << " ";
    }
    cout << "\n";

    return 0;
}
```

Час затрачений на виконання завдання 1 год

Завдання № 6 Algotester Lab 6

```
#include <iostream>
#include <vector>
#include <set>
#include <cmath>
#include <string>

using namespace std;

struct Figure {
    char name;
    int x, y;
};

// Функція для перевірки можливості атаки
bool canAttack(const Figure &f, int x, int y) {
    switch (f.name) {
        case 'P':
            return f.x == x - 1 && abs(f.y - y) == 1;
        case 'R':
            return f.x == x || f.y == y;
        case 'N':
            return (abs(f.x - x) == 2 && abs(f.y - y) == 1) || (abs(f.x - x) == 1 &&
abs(f.y - y) == 2);
        case 'B':
            return abs(f.x - x) == abs(f.y - y);
        case 'K':
            return abs(f.x - x) <= 1 && abs(f.y - y) <= 1;
        case 'Q':
            return f.x == x || f.y == y || abs(f.x - x) == abs(f.y - y);
        default: return false;
    }
}

// Функція для зчитування шахової дошки
vector<Figure> readFigures() {
```

```
    vector<Figure> figures;
    for (int row = 1; row <= 8; ++row) {
        string line;
        cin >> line;
        for (int col = 0; col < line.size(); ++col) {
            if (line[col] != 'O') {
                figures.push_back({line[col], row, col + 1});
            }
        }
    }
    return figures;
}

// Функція для перевірки позиції
string checkPosition(int x, int y, const vector<Figure> &figures) {
    set<char> attackers;
    bool isOccupied = false;

    for (const auto &figure : figures) {
        if (figure.x == x && figure.y == y) {
            isOccupied = true;
            break;
        }
        if (canAttack(figure, x, y)) {
            attackers.insert(figure.name);
        }
    }

    if (isOccupied) {
        return "X";
    } else if (attackers.empty()) {
        return "O";
    } else {
        string result(attackers.begin(), attackers.end());
        return result;
    }
}

int main() {
    // Зчитуємо шахову дошку
    vector<Figure> figures = readFigures();

    // Зчитуємо кількість запитів
    int n;
    cin >> n;

    // Обробляємо запити
    vector<string> results;
    for (int i = 0; i < n; ++i) {
        int x, y;
```

```
        cin >> x >> y;
        results.push_back(checkPosition(x, y, figures));
    }

    // Виводимо результати
    for (const auto &result : results) {
        cout << result << endl;
    }

    return 0;
}
```

Результат:

```
MIEngine-Pid-10q5a0ej.yeu    --dbgExe=
KOOOOOOO
OOOOOOOO
OOOOOOOO
OOOOOOOO
OOOOOOOO
OOOOOOOO
OOOOOOOO
OOOOOOOO

5
1 1
1 2
2 1
3 1
X
K
K
O
```

Час затрачений на виконання завдання 1 год

Завдання № 7 Class Practice Work

Задача 1

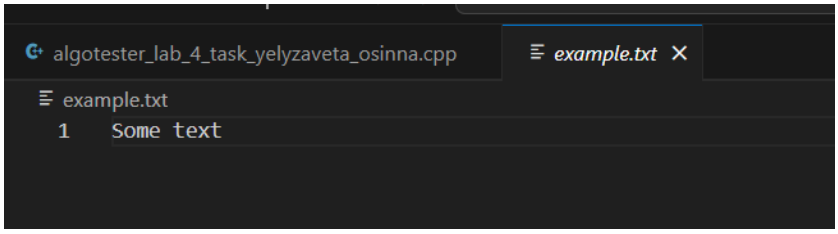
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef enum {
    Success,
    Failure
```

```c
} FileOpResult;

FileOpResult write_to_file(char *name, char *content) {
    // Перевірка на NULL для імені файлу
    if (name == NULL) {
        return Failure;
    }

    // Відкриваємо файл для запису (перезаписуємо його вміст, якщо існує)
    FILE *file = fopen(name, "w");

    // Перевірка на успішне відкриття файлу
    if (file == NULL) {
        return Failure;
    }

    // Записуємо вміст у файл
    if (content != NULL) {
        size_t written = fwrite(content, sizeof(char), strlen(content), file);
        // Перевірка на успішність запису
        if (written < strlen(content)) {
            fclose(file);
            return Failure;
        }
    }

    // Закриваємо файл
    if (fclose(file) != 0) {
        return Failure;
    }

    return Success;
}

int main() {
    char filename[256];
    char content[1024];

    // Зчитуємо ім'я файлу з вводу
    printf("Введіть ім'я файлу: ");
    fgets(filename, sizeof(filename), stdin);
    filename[strcspn(filename, "\n")] = 0; // Видаляємо символ нового рядка

    // Зчитуємо вміст для запису
    printf("Введіть вміст для запису у файл: ");
    fgets(content, sizeof(content), stdin);
    content[strcspn(content, "\n")] = 0; // Видаляємо символ нового рядка

    // Викликаємо функцію запису
    FileOpResult result = write_to_file(filename, content);
```

```
    if (result == Success) {
        printf("Файл успішно записано.\n");
    } else {
        printf("Сталася помилка під час запису у файл.\n");
    }

    return 0;
}
```

Результат:

```
Введіть ім'я файлу: example.txt
Введіть вміст для запису у файл: Some text
Файл успішно записано.
PS D:\AI\ai_programming_playground_2024\ai_13\yelyzaveta_osinna\epic_5>
```



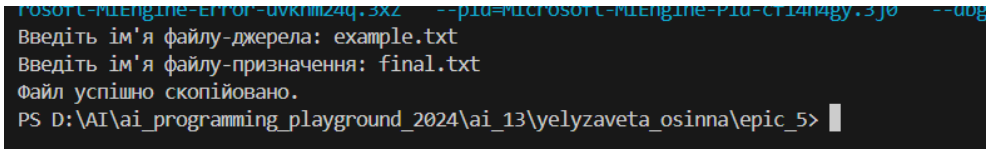
Задача 2

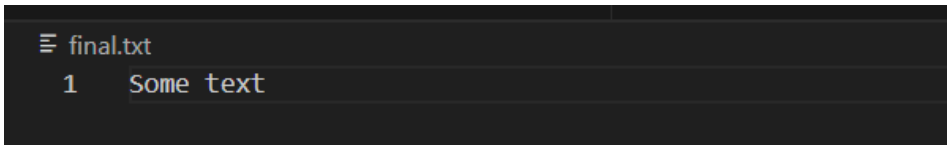
```
#include <stdio.h>
#include <stdlib.h>

typedef enum {
    Success,
    Failure
} FileOpResult;

FileOpResult copy_file(char *file_from, char *file_to) {
    // Перевірка на NULL для імен файлів
    if (file_from == NULL || file_to == NULL) {
        return Failure;
    }

    // Відкриваємо файл для читання
    FILE *src_file = fopen(file_from, "rb");
    if (src_file == NULL) {
        return Failure; // Не вдалося відкрити файл для читання
    }

    // Відкриваємо файл для запису
    FILE *dest_file = fopen(file_to, "wb");
    if (dest_file == NULL) {
        fclose(src_file); // Закриваємо джерело, якщо не вдалося відкрити файл для
запису
```

```
        return Failure; // Не вдалося створити файл для запису
    }

    // Копіюємо вміст
    char buffer[1024];
    size_t bytes_read;
    while ((bytes_read = fread(buffer, sizeof(char), sizeof(buffer), src_file)) > 0)
{
        size_t bytes_written = fwrite(buffer, sizeof(char), bytes_read, dest_file);
        if (bytes_written < bytes_read) {
            fclose(src_file);
            fclose(dest_file);
            return Failure; // Помилка запису
        }
    }

    // Перевіряємо на помилки при читанні
    if (ferror(src_file)) {
        fclose(src_file);
        fclose(dest_file);
        return Failure; // Помилка читання
    }

    // Закриваємо файли
    fclose(src_file);
    fclose(dest_file);

    return Success;
}

int main() {
    char file_from[256];
    char file_to[256];

    // Зчитуємо ім'я файлу-джерела
    printf("Введіть ім'я файлу-джерела: ");
    fgets(file_from, sizeof(file_from), stdin);
    file_from[strcspn(file_from, "\n")] = 0; // Видаляємо символ нового рядка

    // Зчитуємо ім'я файлу-призначення
    printf("Введіть ім'я файлу-призначення: ");
    fgets(file_to, sizeof(file_to), stdin);
    file_to[strcspn(file_to, "\n")] = 0; // Видаляємо символ нового рядка

    // Викликаємо функцію копіювання
    FileOpResult result = copy_file(file_from, file_to);

    if (result == Success) {
        printf("Файл успішно скопійовано.\n");
    } else {
```

```
        printf("Сталася помилка під час копіювання файлу.\n");
    }

    return 0;
}
```

Результат:





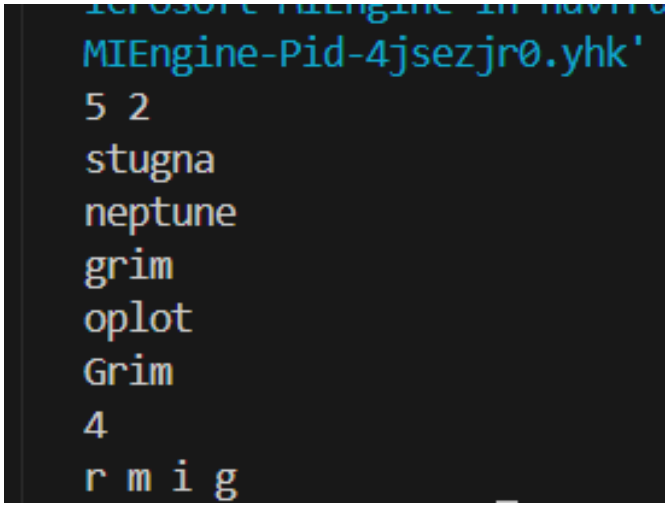
Час затрачений на виконання завдання 1, 5 год

Завдання № 8 Self Practice Work

```
#include <iostream>
#include <vector>
#include <string>
#include <unordered_map>
#include <unordered_set>
#include <algorithm>

using namespace std;

int main() {
    int N, K;
    cin >> N >> K;
    cin.ignore(); // Ігноруємо символ нового рядка після зчитування чисел

    unordered_map<string, int> word_count;

    // Зчитуємо N слів
    for (int i = 0; i < N; ++i) {
        string word;
        getline(cin, word);
        // Перетворюємо слово на малі літери
        transform(word.begin(), word.end(), word.begin(), ::tolower);
        word_count[word]++;
    }

    unordered_set<char> letter_set;

    // Збираємо літери, які зустрічаються в словах, що зустрічаються не менше ніж K
разів
```

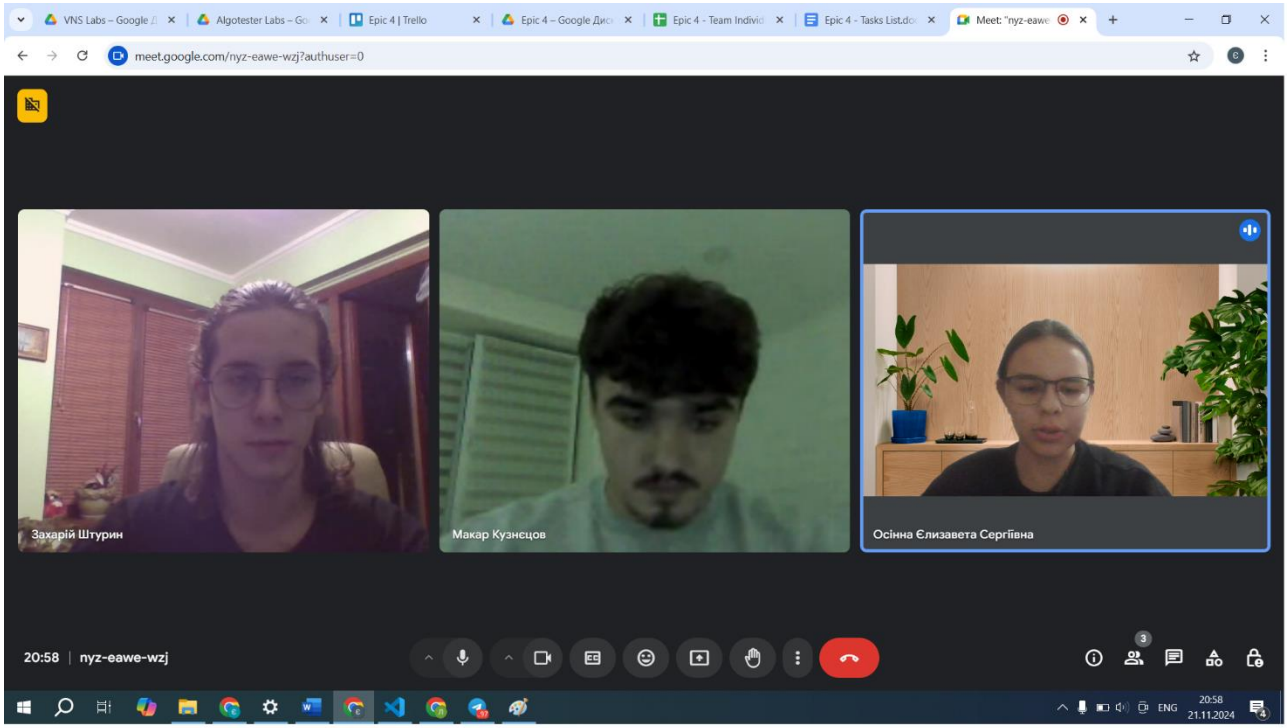
```
    for (const auto& entry : word_count) {
        if (entry.second >= K) {
            const string& word = entry.first;
            for (char c : word) {
                letter_set.insert(c);
            }
        }
    }

    // Перетворюємо множину літер у вектор для сортування
    vector<char> letters(letter_set.begin(), letter_set.end());

    // Сортуємо літери у зворотному алфавітному порядку
    sort(letters.rbegin(), letters.rend());

    // Виводимо результати
    if (!letters.empty()) {
        cout << letters.size() << endl;
        for (char c : letters) {
            cout << c << " ";
        }
        cout << endl;
    } else {
        cout << "Empty!" << endl;
    }

    return 0;
}
```

Результат:

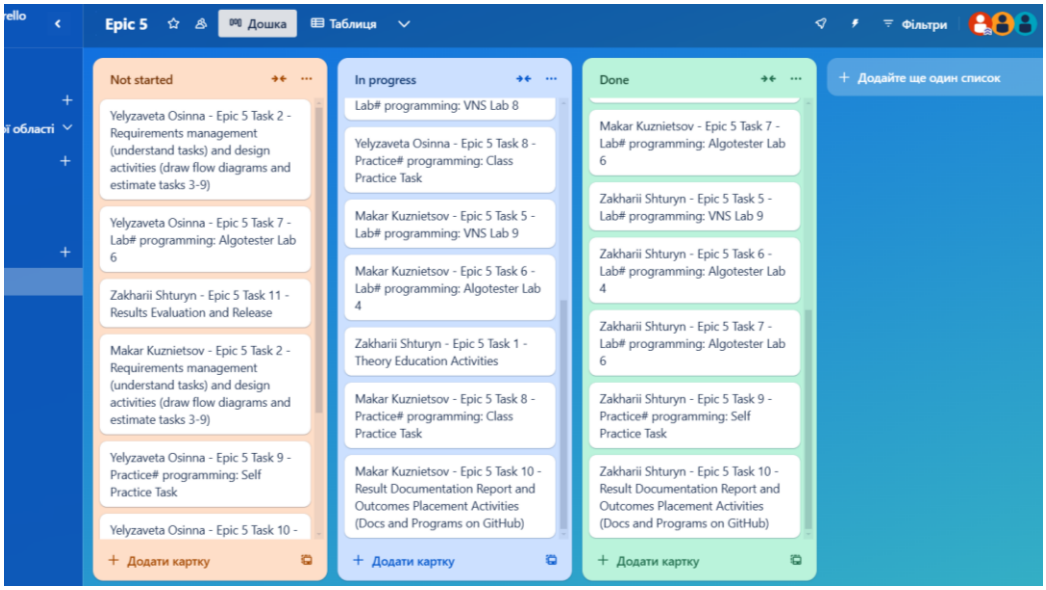


Час затрачений на виконання завдання 40 хв

7. Кооперація з командою:



Trello:



Висновки:

Виконавши цю роботу я вивчила символьні і рядкові змінні і способи їхньої обробки в мові С, опрацювала роботу із двійковими файлами, організацію вводу-виводу структурованої інформації і її зберігання на зовнішніх носіях, робота з текстовими файлами, ввід-вивід текстової інформації і її зберігання на зовнішніх носіях.