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

Виконав(ла):
Студент групи ШІ-11
Зубрицький Арсеній Юрійович

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:
Ознайомитися з основними принципами та методами роботи з файлами в C++, включаючи текстові та бінарні файли. Навчитися використовувати стандартну бібліотеку для роботи з файлами, а також створювати власні бібліотеки для ефективного управління кодом у проектах.

Теоретичні відомості:

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:
- **Тема №1**: Вступ до Роботи з Файлами:
    - Джерела Інформації
        - https://www.programiz.com/c-programming/c-file-input-output
    - Що опрацьовано:
        - Ознайомився з основними операціями над файлами
        - C-style читання та запис до файлу
        - Опрацював базові приклади застосування читання та запису в файлі
    - Статус: Ознайомлений
    - Початок опрацювання теми: 25.11.2024
    - Звершення опрацювання теми: 25.11.2024
    - Витрачено часу: 20 хв
- **Тема №2**: Символи і Рядкові Змінні:
    - Джерела Інформації:
        - https://www.geeksforgeeks.org/cpp-string-functions/
    - Що опрацьовано:
        - Ознайомився з роботою char та string операціями над ними
    - Статус: Ознайомлений
    - Початок опрацювання теми: 25.11.2024
    - Звершення опрацювання теми: 25.11.2024
    - Витрачено часу: 10 хв
- **Тема №3**: Текстові Файли:
    - Джерела Інформації:
        - https://acode.com.ua/urok-220-bazovyj-fajlovyj-vvid-i-vyvid/
    - Що опрацьовано:
        - Опрацював особливості читання та запису
        - Ознайомився з обробкою рядків з файлу: getline, ignore, peek
        - ознайомився з форматуванням тексту при записі: setw, setfill, setprecision. Парсингом текстових файлів: розділення на слова, аналіз структури. Обробку помилок при роботі з файлами
    - Статус: Ознайомлений
    - Початок опрацювання теми: 25.11.2024
    - Звершення опрацювання теми: 25.11.2024
    - Витрачено часу: 40 хв
- **Тема №4**: Бінарні Файли:

    - Джерела Інформації:
        - https://acode.com.ua/urok-221-randomnyj-fajlovyj-vvid-i-vyvid/
        - https://abitap.com/6-0-tekstovi-ta-binarni-fajly/
        - https://foxminded.ua/serializatsiia/
    - Що опрацьовано:
        - Опрацював відмінності бінарних від текстових, приклади (великі дані, ігрові ресурси, зображення)
        - Читання та запис бінарних даних
        - Роботу з позиціонуванням у файлі: seekg, seekp
        - Зрозумів що таке серіалізація об'єктів у бінарний формат
    - Статус: Ознайомлений
    - Початок опрацювання теми: 25.11.2024
    - Звершення опрацювання теми: 25.11.2024
    - Витрачено часу: 50 хв
- **Тема №5**: Стандартна бібліотека та робота з файлами:
    - Джерела Інформації:
        - Лекції та практичні
    - Що опрацьовано:
        - Опрацював потоки вводу/виводу: ifstream, ofstream, fstream
        - Обробка помилок при роботі з файлами
    - Статус: Ознайомлений
    - Початок опрацювання теми: 25.11.2024
    - Звершення опрацювання теми: 25.11.2024
    - Витрачено часу: 20 хв
- **Тема №6**: Створення й використання бібліотек:
    - Джерела Інформації:
        - Лекції та практичні
    - Що опрацьовано:
        - Опрацював 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
    - Статус: Ознайомлений
    - Початок опрацювання теми: 25.11.2024
    - Звершення опрацювання теми: 25.11.2024
    - Витрачено часу: 40 хв

Виконання роботи:

1. Опрацювання завдання та вимог до програм та середовища:

Завдання №1 **Vns Lab_6_task_2_4**

2. Постановка 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.

4. Надрукувати всі слова, які співпадають з її першим словом

Завдання №2 **VNS Lab_8_task_1_4**

2. Постановка 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.

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

   Знищити усі елементи із заданим віком, додати елемент після елемента із заданим номером.

Завдання №3 **VNS Lab_9_task_1_4**

2. Постановка завдання

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

Виконати завдання.

4.

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

2) Підрахувати кількість символів в останньому слові F2.

Завдання №4, 5 **Algotester Lab4v3**

Lab 4v3

Обмеження: 2 сек., 256 МіБ

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 a_i - елементи масиву.

Вихідні дані

У першому рядку M - кількість чисел у масиву

У другому рядку M посоротованих за умовою чисел.

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 Lab6v2**

Lab 6v2

Обмеження: 2 сек., 256 МіБ

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 s_i - 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.

Наявність 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 row_i - 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.

Завдання №7 **Робота з текстовими файлами**

Робота з текстовими файлами

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.

Мета задачі

Розуміння методів роботи з файлами: Робота з файлами є одним з базових навиків програмування. Реалізація функції створення та запису в файл допоможе освоїти практичні навики роботи з файлами з використанням стандартної бібліотеки C++. Для виконання завдання студент має навчитись використовувати методи відкриття файла, запису масиву даних у файл, закриття файла та обробки помилок чи станів операції на кожному з етапів.

Розвиток алгоритмічне мислення: Запис у файл включає набір операції, які якнайкраще вкладаються в концепцію алгоритма, як списка детальних кроків. Імплементація цієї функції наочно демонструє створення алгоритмів у програмуванні.

Освоїти навики роботи з текстовими стрічками: завдання допоможе освоїти роботу з C стрічка, які є масивами з нульовим символом в кінці. Типові концепції при роботі з C стрічками це арифметика вказівників, ітерація по стрічці, копіювання частини стрічки, розбиття на токени по заданому символу.

Розвинути навички розв’язувати задачі: Запис у файл може супроводжуватись набором станів (немає доступу на створення, недостатньо місця, ін.), які необхідно передбачити у алгоритмі. Аналіз цих станів дозволяє розвинути навик розв’язання інженерних задач у програмуванні.

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.

Мета задачі

Розуміння методів роботи з файлами: Робота з файлами є одним з базових навиків програмування. Реалізація функції копіювання вмісту файла допоможе освоїти практичні навики роботи з файлами з використанням стандартної бібліотеки C++. Для виконання завдання студент має навчитись використовувати методи відкриття файла, читання вмісту файла, запису масиву даних у файл, закриття файла та обробки помилок чи станів операції на кожному з етапів.

Розвиток алгоритмічне мислення: Читання та запис у файл включає набір операцій, які якнайкраще вкладаються в концепцію алгоритма, як списка детальних кроків. Імплементація цієї функції наочно демонструє створення алгоритмів у програмуванні.

Освоїти навики роботи з потоком даних: завдання допоможе освоїти роботу з потоками даних (концепція реалізована в STL як набір класів *stream* - fstream, stringstream, streambuf та ін.). Концепція потоку даних дозволяє абстрагувати роботу з джерелами та приймачами даних та писати з її допомогою високорівневий код.

Розвинути навички розв'язувати задачі: Операції читання з файла та запис у файл можуть супроводжуватись набором різних станів (немає доступу на читання чи створення, недостатньо місця, ін.), які необхідно передбачити у алгоритмі. Аналіз цих станів дозволяє розвинути навик розв'язання інженерних задач у програмуванні.

2. Код програм з посиланням на зовнішні ресурси:

Завдання №1

```cpp
#include <iostream>
#include <cstring>
#include <cstdlib>

using namespace std;

// Функція для пошуку та друку слів, які співпадають з першим словом
void FindMatchingWords(char str[]) {
    // Розділяємо рядок на слова, використовуючи пробіли
    char* firstWord = strtok(str, " "); // Знаходимо перше слово
    if (firstWord == nullptr) {
        cout << "Рядок не містить слів!" << endl;
        return;
    }

    cout << "Перше слово: " << firstWord << endl;
    cout << "Слова, які співпадають з першим словом: " << endl;

    // Перебираємо інші слова і порівнюємо їх з першим
    char* currentWord = strtok(nullptr, " ");
    while (currentWord != nullptr) {
        // Порівнюємо поточне слово з першим
        if (strcmp(currentWord, firstWord) == 0) {
            cout << currentWord << endl; // Друкуємо, якщо співпадає
        }
        currentWord = strtok(nullptr, " ");
    }
}

int main() {
    char s[256]; // Масив для збереження введеного рядка

    // Ввід рядка користувачем
    cout << "Введіть рядок (до 255 символів, завершуйте крапкою):";
    gets(s);  // Зчитуємо рядок

    // Видаляємо крапку з кінця рядка, якщо вона є
    size_t len = strlen(s);
    if (len > 0 && s[len - 1] == '.') {
        s[len - 1] = '\0';
    }

    // Викликаємо функцію для пошуку та друку слів
    FindMatchingWords(s);

    return 0;
}

```

Посилання на pull-request:
https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/687/files#diff-c7e8d1e0c691b6a55a46c3297ac4b926003c0bd6b69bdcf624335a69e117522a

Завдання №2

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

using namespace std;

// Структура "Людина"
struct Person {
    char lastName[50];
    char firstName[50];
    char middleName[50];
    char address[100];
    char phoneNumber[20];
    int age;
};

// Функція для заповнення однієї структури
void fillPerson(Person& p) {
    cout << "Введіть прізвище: ";
    cin >> p.lastName;
    cout << "Введіть ім'я: ";
    cin >> p.firstName;
    cout << "Введіть по батькові: ";
    cin >> p.middleName;
    cout << "Введіть домашню адресу: ";
    cin.ignore(); // Очищаємо буфер
    cin.getline(p.address, 100);
    cout << "Введіть номер телефону: ";
    cin >> p.phoneNumber;
    cout << "Введіть вік: ";
    cin >> p.age;
}

// Функція для виведення даних про людину
void printPerson(const Person& p) {
    cout << "Прізвище: " << p.lastName << "\nІм'я: " << p.firstName << "\nПо батькові: " << p.middleName
         << "\nАдреса: " << p.address << "\nТелефон: " << p.phoneNumber << "\nВік: " << p.age << "\n----------------------" << endl;
}

```

```cpp
// Функція для запису структури у файл
void writeToFile(const char* filename, const Person& p) {
    ofstream outFile(filename, ios::binary | ios::app);
    if (!outFile) {
        cerr << "Помилка відкриття файлу для запису!" << endl;
        return;
    }
    outFile.write(reinterpret_cast<const char*>(&p), sizeof(Person));
    outFile.close();
}

// Функція для читання всіх елементів з файлу
void readFromFile(const char* filename) {
    ifstream inFile(filename, ios::binary);
    if (!inFile) {
        cerr << "Помилка відкриття файлу для читання!" << endl;
        return;
    }
    Person p;
    while (inFile.read(reinterpret_cast<char*>(&p), sizeof(Person))) {
        printPerson(p);
    }
    inFile.close();
}

// Функція для видалення людей із заданим віком
void removeByAge(const char* filename, int ageToRemove) {
    ifstream inFile(filename, ios::binary);
    if (!inFile) {
        cerr << "Помилка відкриття файлу для читання!" << endl;
        return;
    }

    ofstream tempFile("temp.dat", ios::binary);
    if (!tempFile) {
        cerr << "Помилка відкриття тимчасового файлу для запису!" << endl;
        return;
    }

    Person p;
    while (inFile.read(reinterpret_cast<char*>(&p), sizeof(Person))) {
        if (p.age != ageToRemove) {
            tempFile.write(reinterpret_cast<const char*>(&p), sizeof(Person));
        }
    }

    inFile.close();
    tempFile.close();

    remove(filename);
    rename("temp.dat", filename);
    cout << "Елементи з віком " << ageToRemove << " успішно видалено." << endl;
}
```

```cpp
// Функція для додавання нового елемента після заданого
void addAfterPerson(const char* filename, int personNumber, const Person& newPerson) {
    ifstream inFile(filename, ios::binary);
    if (!inFile) {
        cerr << "Помилка відкриття файлу для читання!" << endl;
        return;
    }

    ofstream tempFile("temp.dat", ios::binary);
    if (!tempFile) {
        cerr << "Помилка відкриття тимчасового файлу для запису!" << endl;
        return;
    }

    Person p;
    int currentIndex = 0;
    while (inFile.read(reinterpret_cast<char*>(&p), sizeof(Person))) {
        tempFile.write(reinterpret_cast<const char*>(&p), sizeof(Person));
        currentIndex++;
        if (currentIndex == personNumber) {
            tempFile.write(reinterpret_cast<const char*>(&newPerson), sizeof(Person));
        }
    }

    inFile.close();
    tempFile.close();

    remove(filename);
    rename("temp.dat", filename);
    cout << "Нова людина додана після " << personNumber << "-го елемента." << endl;
}

// Функція для очищення файлу
void clearFile(const char* filename) {
    ofstream outFile(filename, ios::binary | ios::trunc);
    if (!outFile) {
        cerr << "Помилка відкриття файлу для очищення!" << endl;
        return;
    }
    outFile.close();
    cout << "Файл успішно очищений." << endl;
}

// Функція для видалення файлу
void deleteFile(const char* filename) {
    if (remove(filename) != 0) {
        cerr << "Помилка при видаленні файлу!" << endl;
    } else {
        cout << "Файл успішно видалено." << endl;
    }
}
```

```cpp
int main() {
    const char* filename = "people.dat";

    // Введення кількості людей, яких хочемо додати в файл
    int count;
    cout << "Введіть кількість людей: ";
    cin >> count;

    // Додаємо людей у файл
    for (int i = 0; i < count; i++) {
        Person p;
        cout << "Введіть дані для " << i + 1 << "-ї людини:" << endl;
        fillPerson(p);
        writeToFile(filename, p);
    }

    // Читання всіх елементів з файлу
    cout << "\nВміст файлу:\n";
    readFromFile(filename);

    // Видалення людей із заданим віком
    int ageToRemove;
    cout << "\nВведіть вік для видалення: ";
    cin >> ageToRemove;
    removeByAge(filename, ageToRemove);

    // Читання всіх елементів після видалення
    cout << "\nВміст файлу після видалення:\n";
    readFromFile(filename);

    // Додавання нового елемента
    Person newPerson;
    cout << "\nВведіть дані для нової людини, яку потрібно додати після певного елемента:" << endl;
    fillPerson(newPerson);
    int personNumber;
    cout << "Введіть номер елемента після якого потрібно додати нового: ";
    cin >> personNumber;
    addAfterPerson(filename, personNumber, newPerson);

    // Читання всіх елементів після додавання
    cout << "\nВміст файлу після додавання:\n";
    readFromFile(filename);

    // Очищення файлу
    char choice;
    cout << "\nЧи хочете ви очистити файл? (y/n): ";
    cin >> choice;
    if (choice == 'y' || choice == 'Y') {
        clearFile(filename);
    }

    // Видалення файлу
    cout << "\nЧи хочете ви видалити файл? (y/n): ";
    cin >> choice;
    if (choice == 'y' || choice == 'Y') {
        deleteFile(filename);
    }

    return 0;
}
```

Посилання на pull-request:
https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/687/files#diff-dbabd572f29ad3c0211cdc7c3eb0acb728aace4ccaa7aa08a55a7b99f77d12ff

Завдання №3

```cpp
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>

using namespace std;

// Функція для копіювання рядків з F1 у F2, починаючи з 4-го
void copyLinesFromFile(const string& inputFile, const string& outputFile) {
    ifstream inFile(inputFile);
    if (!inFile) {
        cerr << "Помилка відкриття файлу " << inputFile << endl;
        return;
    }

    ofstream outFile(outputFile);
    if (!outFile) {
        cerr << "Помилка відкриття файлу " << outputFile << endl;
        return;
    }

    string line;
    int lineCount = 0;

    // Читаємо рядки з файлу і записуємо в F2, починаючи з 4-го
    while (getline(inFile, line)) {
        lineCount++;
        if (lineCount >= 4) {
            outFile << line << endl;
        }
    }

    inFile.close();
    outFile.close();
}

// Функція для підрахунку кількості символів в останньому слові файлу
int countLastWordLength(const string& fileName) {
    ifstream inFile(fileName);
    if (!inFile) {
        cerr << "Помилка відкриття файлу " << fileName << endl;
        return 0;
    }

    string line;
    string lastWord;
    while (getline(inFile, line)) {
        // Розбиваємо рядок на слова
        stringstream ss(line);
        string word;
        while (ss >> word) {
            lastWord = word;  // Оновлюємо останнє слово
        }
    }

    inFile.close();

    // Повертаємо довжину останнього слова
    return lastWord.length();
}
```

```
62  int main() {
63      const string inputFile = "F1.txt";
64      const string outputFile = "F2.txt";
65
66      // Створюємо файл F1 і записуємо в нього інформацію
67      ofstream outFile(inputFile);
68      if (!outFile) {
69          cerr << "Помилка відкриття файлу для запису!" << endl;
70          return 1;
71      }
72
73      // Записуємо 10 рядків у файл F1
74      outFile << "Перше слово" << endl;
75      outFile << "Друге слово" << endl;
76      outFile << "Третє слово" << endl;
77      outFile << "Четверте слово" << endl;
78      outFile << "П'яте слово" << endl;
79      outFile << "Шосте слово" << endl;
80      outFile << "Сьоме слово" << endl;
81      outFile << "Восьме слово" << endl;
82      outFile << "Дев'яте слово" << endl;
83      outFile << "Десяте слово" << endl;
84
85      outFile.close();
86
87      // Копіюємо рядки з F1 у F2, починаючи з 4-го
88      copyLinesFromFile(inputFile, outputFile);
89      cout << "Рядки скопійовано з F1 в F2, починаючи з 4-го." << endl;
90
91      // Підраховуємо кількість символів в останньому слові файлу F2
92      int lastWordLength = countLastWordLength(outputFile);
93      cout << "Кількість символів в останньому слові файлу F2: " << lastWordLength << endl;
94
95      return 0;
96  }
97
```

Посилання на pull-request:
https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/687/files#diff-be7be864a6d871bcf3377b24bda94c955ff7136078075d91e2918bd1081d0a0f

Завдання №4

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
#include <iterator>

using namespace std;

int main() {
    int N;
    cin >> N;

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

    // Розділяємо на три групи за остачею від ділення на 3
    vector<int> group0, group1, group2;

    for (int num : arr) {
        if (num % 3 == 0) group0.push_back(num);
        else if (num % 3 == 1) group1.push_back(num);
        else group2.push_back(num);
    }

    // Сортуємо за умовами: group0 та group2 по зростанню, group1 по спаданню
    sort(group0.begin(), group0.end());
    sort(group1.begin(), group1.end(), greater<int>());
    sort(group2.begin(), group2.end());

    // Об'єднуємо всі три групи
    vector<int> result;
    result.insert(result.end(), group0.begin(), group0.end());
    result.insert(result.end(), group1.begin(), group1.end());
    result.insert(result.end(), group2.begin(), group2.end());

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

    // Виводимо результат
    cout << result.size() << endl;
    for (int num : result) {
        cout << num << " ";
    }
    cout << endl;

    return 0;
}

```

Посилання на pull-request:
https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/687/files#diff-8452669e4a613d21ad5949915cf51743bf899f244e8e88a41765183097d04ab3

Завдання №5

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

using namespace std;

void sortGroup0(vector<int>& group) {
    sort(group.begin(), group.end());
}

void sortGroup1(vector<int>& group) {
    sort(group.begin(), group.end(), greater<int>());
}

void sortGroup2(vector<int>& group) {
    sort(group.begin(), group.end());
}

bool contains(const vector<int>& result, int value) {
    return find(result.begin(), result.end(), value) != result.end();
}

int main() {
    int N;
    cin >> N;

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

    // Розділяємо на три групи за остачею від ділення на 3
    vector<int> group0, group1, group2;

    for (int num : arr) {
        if (num % 3 == 0) group0.push_back(num);
        else if (num % 3 == 1) group1.push_back(num);
        else group2.push_back(num);
    }

    // Сортуємо групи
    sortGroup0(group0);
    sortGroup1(group1);
    sortGroup2(group2);

    // Об'єднуємо всі три групи
    vector<int> result;
    result.insert(result.end(), group0.begin(), group0.end());
    result.insert(result.end(), group1.begin(), group1.end());
    result.insert(result.end(), group2.begin(), group2.end());

```

```
    // Видаляємо дублікати вручну
    vector<int> finalResult;
    for (int num : result) {
        if (!contains(finalResult, num)) {
            finalResult.push_back(num);
        }
    }

    // Виводимо результат
    cout << finalResult.size() << endl;
    for (int num : finalResult) {
        cout << num << " ";
    }
    cout << endl;

    return 0;
}

```

Посилання на pull-request:
https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/687/files#diff-3f580ce006ae498decea0c018b28911951ead719febacd416f87de3c1b8ac36c

Завдання №6

```cpp
#include <iostream>
#include <vector>
#include <set>
#include <algorithm>

using namespace std;

#define BOARD_SIZE 8

// Функція перевірки, чи може пішак атакувати клітинку
bool canAttackPawn(int x, int y, int px, int py) {
    return (px == x + 1 && (py == y - 1 || py == y + 1)); // Пішак атакує вниз по діагоналях
}

// Функція для перевірки атаки тура
bool canAttackRook(int x, int y, int rx, int ry, const vector<string>& board) {
    if (x != rx && y != ry) return false; // Тура може атакувати лише по вертикалі чи горизонталі
    if (x == rx) {
        // Перевіряємо, чи є фігури між турами по горизонталі
        int start = min(y, ry) + 1;
        int end = max(y, ry);
        for (int j = start; j < end; ++j) {
            if (board[x][j] != 'O') return false; // Фігура між турами блокує атаку
        }
    } else if (y == ry) {
        // Перевіряємо по вертикалі
        int start = min(x, rx) + 1;
        int end = max(x, rx);
        for (int i = start; i < end; ++i) {
            if (board[i][y] != 'O') return false; // Фігура між турами блокує атаку
        }
    }
    return true;
}

// Функція для перевірки атаки слона
bool canAttackBishop(int x, int y, int bx, int by, const vector<string>& board) {
    if (abs(x - bx) != abs(y - by)) return false; // Слон може атакувати лише по діагоналях
    int dx = (bx > x) ? 1 : -1;
    int dy = (by > y) ? 1 : -1;
    int i = x + dx, j = y + dy;
    while (i != bx && j != by) {
        if (board[i][j] != 'O') return false; // Фігура між слоном і клітинкою блокує атаку
        i += dx;
        j += dy;
    }
    return true;
}

// Функція для перевірки атаки конем
bool canAttackKnight(int x, int y, int nx, int ny) {
    return (abs(x - nx) == 2 && abs(y - ny) == 1) || (abs(x - nx) == 1 && abs(y - ny) == 2); // Кінь може атакувати в "букву Г"
}

// Функція для перевірки атаки королем
bool canAttackKing(int x, int y, int kx, int ky) {
    return abs(x - kx) <= 1 && abs(y - ky) <= 1; // Король атакує на одну клітинку у будь-якому напрямку
}
```

```cpp
// Функція для перевірки атаки королевою
bool canAttackQueen(int x, int y, int qx, int qy, const vector<string>& board) {
    return canAttackRook(x, y, qx, qy, board) || canAttackBishop(x, y, qx, qy, board); // Королева атакує як тура і слон
}

// Основна функція
int main() {
    vector<string> board(8);
    for (int i = 0; i < 8; ++i) {
        cin >> board[i];
    }

    int Q;
    cin >> Q;
    vector<pair<int, int>> queries(Q);
    for (int i = 0; i < Q; ++i) {
        cin >> queries[i].first >> queries[i].second;
        queries[i].first--;  // Перехід до індексації з нуля
        queries[i].second--;
    }

    // Для кожного запиту визначаємо фігури, що атакують клітинку
    for (auto& query : queries) {
        int x = query.first;
        int y = query.second;

        // Якщо в клітинці вже є фігура
        if (board[x][y] != 'O') {
            cout << "X" << endl;
            continue;
        }

        set<char> attackingFigures;

        // Перевіряємо атаки від усіх фігур на дошці
        for (int i = 0; i < BOARD_SIZE; ++i) {
            for (int j = 0; j < BOARD_SIZE; ++j) {
                if (board[i][j] == 'O') continue; // Якщо клітинка пуста

                // Перевірка на атаку кожною фігурою
                if (board[i][j] == 'P' && canAttackPawn(x, y, i, j)) attackingFigures.insert('P');
                if (board[i][j] == 'R' && canAttackRook(x, y, i, j, board)) attackingFigures.insert('R');
                if (board[i][j] == 'N' && canAttackKnight(x, y, i, j)) attackingFigures.insert('N');
                if (board[i][j] == 'B' && canAttackBishop(x, y, i, j, board)) attackingFigures.insert('B');
                if (board[i][j] == 'K' && canAttackKing(x, y, i, j)) attackingFigures.insert('K');
                if (board[i][j] == 'Q' && canAttackQueen(x, y, i, j, board)) attackingFigures.insert('Q');
            }
        }

        if (attackingFigures.empty()) {
            cout << "O" << endl;
        } else {
            for (char c : attackingFigures) {
                cout << c;
            }
            cout << endl;
        }
    }

    return 0;
```

Посилання на pull-request:
https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/687/files#diff-b3c7d052b4f0334dfdf313205b8287690eeecd831538cfcf5c5cc6381b2560a6

Завдання №7

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

enum FileOpResult { Success, Failure };

FileOpResult write_to_file(const char* name, const char* content) {
    std::ofstream outFile(name, std::ios::trunc);
    if (!outFile) return Failure;  // Якщо файл не відкрився для запису
    outFile << content;
    if (outFile.fail()) return Failure;  // Помилка при записі
    outFile.close();
    return outFile.is_open() ? Failure : Success;  // Перевірка закриття файлу
}

FileOpResult copy_file(const char* file_from, const char* file_to) {
    std::ifstream inFile(file_from, std::ios::in);
    if (!inFile) return Failure;  // Якщо файл не відкрився для читання

    std::ofstream outFile(file_to, std::ios::trunc);
    if (!outFile) return Failure;  // Якщо файл не відкрився для запису

    outFile << inFile.rdbuf();
    if (inFile.fail() || outFile.fail()) return Failure;  // Помилка при копіюванні
    inFile.close();
    outFile.close();
    return Success;
}

int main() {
    // Задача 1: Запис в файл
    const char* filename = "output.txt";
    const char* content = "Це текст, який потрібно записати у файл.";
    if (write_to_file(filename, content) == Success)
        std::cout << "Текст успішно записано в файл!" << std::endl;
    else
        std::cout << "Не вдалося записати текст у файл." << std::endl;

    // Задача 2: Копіювання вмісту одного файлу в інший
    const char* file_from = "input.txt";
    const char* file_to = "copied_output.txt";
    if (copy_file(file_from, file_to) == Success)
        std::cout << "Вміст успішно скопійовано!" << std::endl;
    else
        std::cout << "Не вдалося скопіювати вміст файлів." << std::endl;

    return 0;
}

```

Посилання на pull-request:
https://github.com/artificial-intelligence-department/ai_programming_playground_2024

/pull/687/files#diff-1a703342c73ef2ae2c17f6dde514e7366d2958bf2f20944f2e95d50a48f0585b

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

Завдання №1

```
Введіть рядок (до 255 символів, завершуйте крапкою):dog cat fish bird dog cat.
Перше слово: dog
Слова, які співпадають з першим словом:
dog
```

Час затрачений на виконання завдання: 1 год

Завдання №2

```
Введіть кількість людей: 2
Введіть дані для 1-ї людини:
Введіть прізвище: Black
Введіть ім'я: John
Введіть по батькові:

Batkovych
Введіть домашню адресу: shevchenka_12
Введіть номер телефону: 098654726
Введіть вік: 25
Введіть дані для 2-ї людини:
Введіть прізвище: Zubrytskyi
Введіть ім'я: Arsenii
Введіть по батькові: Yuriyovych
Введіть домашню адресу: Levka-Lykyanenka
Введіть номер телефону: 0985739883
Введіть вік: 17

Вміст файлу:
Прізвище: Black
Ім'я: John
По батькові: Batkovych
Адреса: shevchenka_12
Телефон: 098654726
Вік: 25
------------------------
Прізвище: Zubrytskyi
Ім'я: Arsenii
По батькові: Yuriyovych
Адреса: Levka-Lykyanenka
Телефон: 0985739883
Вік: 17
------------------------

Введіть вік для видалення: 25
Елементи з віком 25 успішно видалено.

Вміст файлу після видалення:
Прізвище: Zubrytskyi
Ім'я: Arsenii
По батькові: Yuriyovych
Адреса: Levka-Lykyanenka
Телефон: 0985739883
Вік: 17
------------------------

Введіть дані для нової людини, яку потрібно додати після певного елемента:
Введіть прізвище: dmytro
Введіть ім'я: dzmil
Введіть по батькові: batkovych
Введіть домашню адресу: pokrovska_1
Введіть номер телефону: 063888227
Введіть вік: 23
Введіть номер елемента після якого потрібно додати нового: 0
Нова людина додана після 0-го елемента.
```

Час затрачений на виконання завдання: 4 год

Завдання №3

```
Рядки скопійовано з F1 в F2, починаючи з 4-го.
Кількість символів в останньому слові файлу F2: 10
```

Час затрачений на виконання завдання: 1 год

Завдання №4

```
9
4 5 10 9 8 7 6 3 11
9
3 6 9 10 7 4 5 8 11
```

Час затрачений на виконання завдання: 1 год

Завдання №5

```
9
2 3 4 6 7 23 33 8 9
9
3 6 9 33 7 4 2 8 23
```

Час затрачений на виконання завдання: 40 хв

Завдання №6

```
ROOOOPOO
OOOOOOOO
OOOOOOOO
OOOOOOOO
OOOOOOOO
OOOOOOOO
OOOOOOOO
OOOOOOOO

4
1 1
1 5
1 6
1 8
X
R
X
O
```

Час затрачений на виконання завдання: 3 год

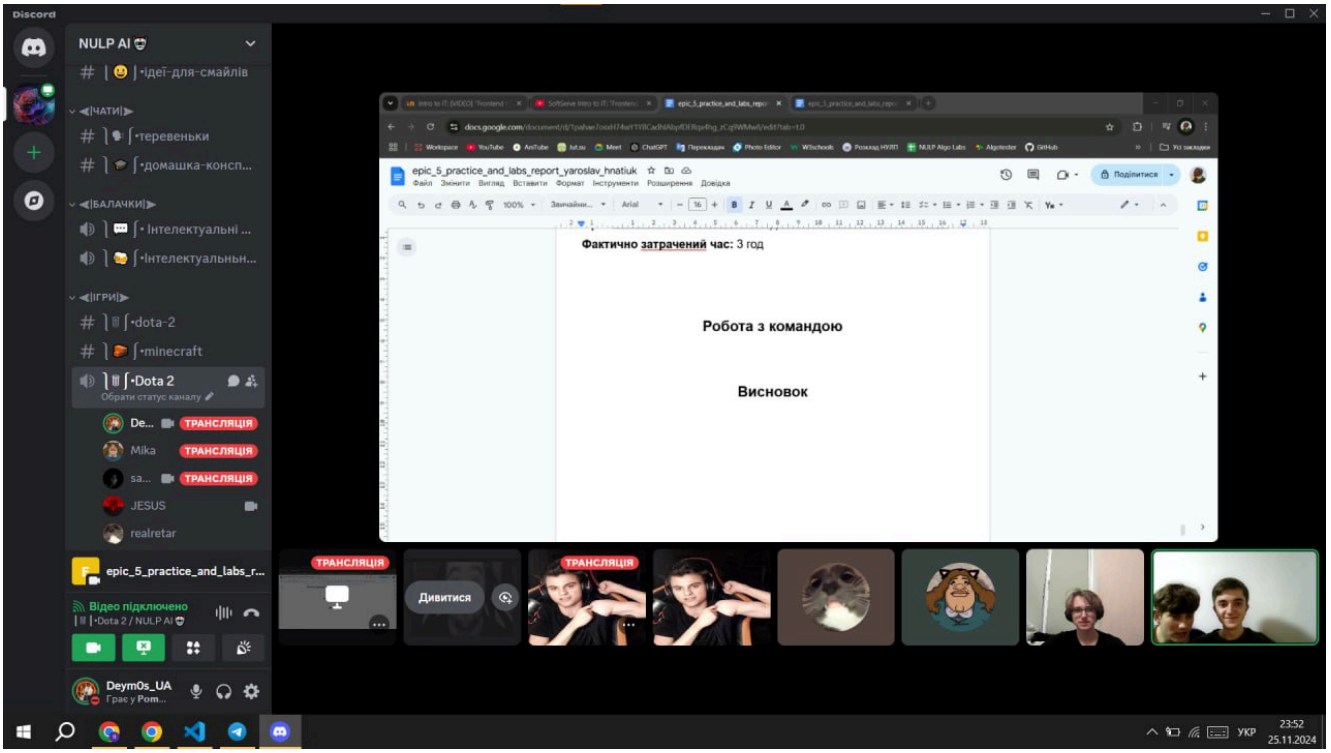
Завдання №7

```
Текст успішно записано в файл!
Вміст успішно скопійовано!
```

Час затрачений на виконання завдання: 1 год

6. Кооперація з командою:

- Скрін з зустрічі по обговоренню задач Епіку та Скрін прогресу по Трелло



Висновки: В ході виконання робіт з epic_5 я опрацював Файли. Бінарні Файли. Символи і Рядкові Змінні та Текстові Файли. Стандартна бібліотека та деталі/методи роботи з файлами. На практиці застосував та закріпив вивчені знання.