

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



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 ІІІ-11

Гуменюк Анастасія Олександрівна

Тема роботи: Вивчення роботи з файлами у C++, зокрема текстових і бінарних файлів, а також основних операцій, таких як відкриття, читання, запис і закриття. Дослідження роботи з файловими дескрипторами, перевірки стану файлу та обробки помилок. Огляд роботи з символами та рядковими змінними (типи `char` і `string`), а також базових операцій з рядками: конкатенація, порівняння, пошук. Розгляд особливостей роботи з текстовими файлами (зчитування, обробка рядків) і форматуванням тексту. Використання стандартної бібліотеки для роботи з файлами (потoki `ifstream`, `ofstream`, `fstream`). Вивчення принципів створення власних бібліотек у C++ та правил їх структурування і застосування.

Мета роботи: Навчитися основним принципам роботи з файлами у C++ та розібратися з текстовими і бінарними файлами, включаючи операції відкриття, читання, запису та закриття. Опанувати перевірку стану файлу. Дослідити базові операції з символами та рядковими змінними, такі як конкатенація, порівняння і пошук у рядках. Навчитися формувати текстові файли при записі даних і застосовувати методи для обробки рядків з файлу. Зрозуміти принципи роботи з бінарними файлами. Ознайомитися з використанням стандартної бібліотеки для роботи з файлами (`ifstream`, `ofstream`, `fstream`). Спробувати створити власні бібліотеки у C++ та організувати їх структуру для полегшення роботи з файлами у майбутніх проектах.

Теоретичні відомості:

Теоретичні відомості з переліком важливих тем:

- Тема №1: Вступ до Роботи з Файлами.
- Тема №2: Символи і Рядкові Змінні.
- Тема №3: Текстові Файли.
- Тема №4: Бінарні Файли.
- Тема №5: Стандартна бібліотека та робота з файлами.
- Тема №6: Створення й використання бібліотек.

Індивідуальний план опрацювання теорії:

Тема №1: Вступ до Роботи з Файлами.

- Джерела:
<https://youtu.be/FeNqHytI0fA?si=uU-vKhKXGhEFFofB>
- Що опрацьовано:

- 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
- Статус: Ознайомлена
- Початок опрацювання теми: 11.11.2024.
- Звершення опрацювання теми: 11.11.2024 (1 год 23 хв.).

Тема №2: Символи і Рядкові Змінні.

- Джерела:
 - <https://www.youtube.com/watch?v=1DtZCv7xfb8&t=955s>
- Що опрацьовано:
 - Робота з char та string: основні операції і методи
 - Стрічкові літерали та екранування символів
 - Конкатенація, порівняння та пошук у рядках
- Статус: Ознайомлена
- Початок опрацювання теми: 11.11.2024.
- Звершення опрацювання теми: 11.11.2024 (35хв.).

Тема №3: Текстові Файли.

- Джерела:
 - https://youtu.be/SSNJ7alki-E?si=EAXljt_gw6hCG5hR
 - <https://acode.com.ua/urok-220-bazovyj-fajlovyj-vvid-i-vyvid/>
- Що опрацьовано:
 - Особливості читання та запису текстових файлів
 - Обробка рядків з файлу: getline, ignore, peek
 - Форматування тексту при записі: setw, setfill, setprecision
 - Обробка помилок при роботі з файлами
- Статус: Ознайомлена
- Початок опрацювання теми: 11.11.2024.
- Звершення опрацювання теми: 11.11.2024 (1 год 40 хв.).

Тема №4: Бінарні Файли.

- Джерела:
 - <https://studfile.net/preview/5994719/page:7/>
 - <https://acode.com.ua/urok-221-randomnyj-fajlovyj-vvid-i-vyvid/>
- Що опрацьовано:
 - 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
- Статус: Ознайомлена
- Початок опрацювання теми: 11.11.2024.
- Звершення опрацювання теми: 11.11.2024 (30хв.).

Тема №5: Стандартна бібліотека та робота з файлами.

- Джерела:
 - https://youtu.be/L7JGsi4sryc?si=_cEc4SG0Qu9c0NBE
 - https://youtu.be/FvbiCKvIAHo?si=TZodZ2hAGH_dKDzy
- Що опрацьовано:
 - Огляд стандартної бібліотеки для роботи з файлами
 - Потоки вводу/виводу: ifstream, ofstream, fstream
 - Обробка помилок при роботі з файлами
- Статус: Ознайомлена
- Початок опрацювання теми: 11.11.2024.
- Звершення опрацювання теми: 11.11.2024 (50 хв.).

Виконання роботи:

1) Опрацювання завдання та вимог до програми та середовища

Завдання №1 VNS Lab 6 - 10

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.

Перетворити рядок таким чином, щоб на його початку були записані слова, що містять тільки цифри, потім слова, що містять тільки букви, а потім слова, які містять і букви і цифри.

Завдання №2 VNS Lab 8 – 10

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 "Інформація":

- носій;
- об'єм;
- назва;
- автор.

Знищити перший елемент із заданим об'ємом інформації, додати елемент перед елементом із зазначеним номером.

1. Для заповнення файлу можна використовувати функцію, що формує одну структуру, зазначеного у варіанті типу. Значення елементів структури вводяться із клавіатури. Для вводу можна використовувати операцію >> і функцію gets().
2. При вводі структур можна реалізувати один з таких механізмів:
 - ввід заздалегідь обраної кількості структур (не менше 5);
 - ввід до появи структури із заданою кількістю ознак;
 - діалог з користувачем про необхідність продовжувати ввід.
3. Для запису структури у файл і читання структури з файлу використовувати функції блокового вводу/виводу fread й fwrite.
4. Для знищення/додавання елементів у файл використовувати допоміжний файл.

Завдання №3 VNS Lab 9 – 10

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

- 1) Скопіювати з файлу F1 у файл F2 всі рядки, які не містять слова, що починаються на одну букву.
- 2) Знайти найкоротше слово у файлі F2.

Завдання №4 Algotester Lab 4 – 3

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 посортованих за умовою чисел.

Обмеження

$$1 \leq N \leq 10^3$$

$$0 \leq a_i \leq 10^3$$

Пам'ятайте, ви маєте написати 2 варіанти розв'язку, 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 $N \cdot \log N$ часу.

Завдання №5 Algotester Lab 6 – 2

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.

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 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

$$|rowi|=N$$

$$rowi \in \{O, P, R, N, B, K, Q\}$$

$$1 \leq Q \leq 64$$

$$1 \leq x, y \leq 8$$

Завдання №6 Class Practice Task

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 навички 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 операцій, які якнайкраще вкладаються в концепцію алгоритма, як списка детальних кроків. Імплементация цієї функції наочно демонструє створення алгоритмів у програмуванні.

Освоїти навички 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.

2) *Реалізувати функцію створення файлу і запису в нього даних:*

```
enum FileOpResult { Success, Failure, ... };  
FileOpResult copy_file(char *file_from, char *file_to);
```

Умови задачі:

- копіювати вміст файлу 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 файлу допоможе освоїти практичні навички роботи з файлами з використанням стандартної бібліотеки C++. Для виконання завдання студент має навчитись використовувати методи відкриття файла, читання вмісту файлу, 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. Імплементация цієї функції наочно демонструє створення алгоритмів у програмуванні.

Освоїти навички роботи з потоком даних: завдання допоможе освоїти роботу з потоками даних (концепція реалізована в STL як набір класів `*stream*` - `fstream`, `stringstream`, `stringstreambuf` 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.

Завдання №7 Self Practice Work

Вам дано масив a з N цілих чисел.

Спочатку видаліть масиву a усі елементи що повторюються, наприклад масив $[1, 3, 3, 4]$ має перетворитися у $[1, 3, 4]$.

Після цього оберніть посортовану версію масиву a на K , тобто при $K=3$ масив $[1, 2, 3, 4, 5, 6, 7]$ перетвориться на $[4, 5, 6, 7, 1, 2, 3]$. Виведіть результат.

Вхідні дані

У першому рядку цілі числа N та K

У другому рядку N цілих чисел - елементи масиву a

Вихідні дані

У першому рядку ціле число N - розмір множини a

У наступному рядку N цілих чисел - множина a

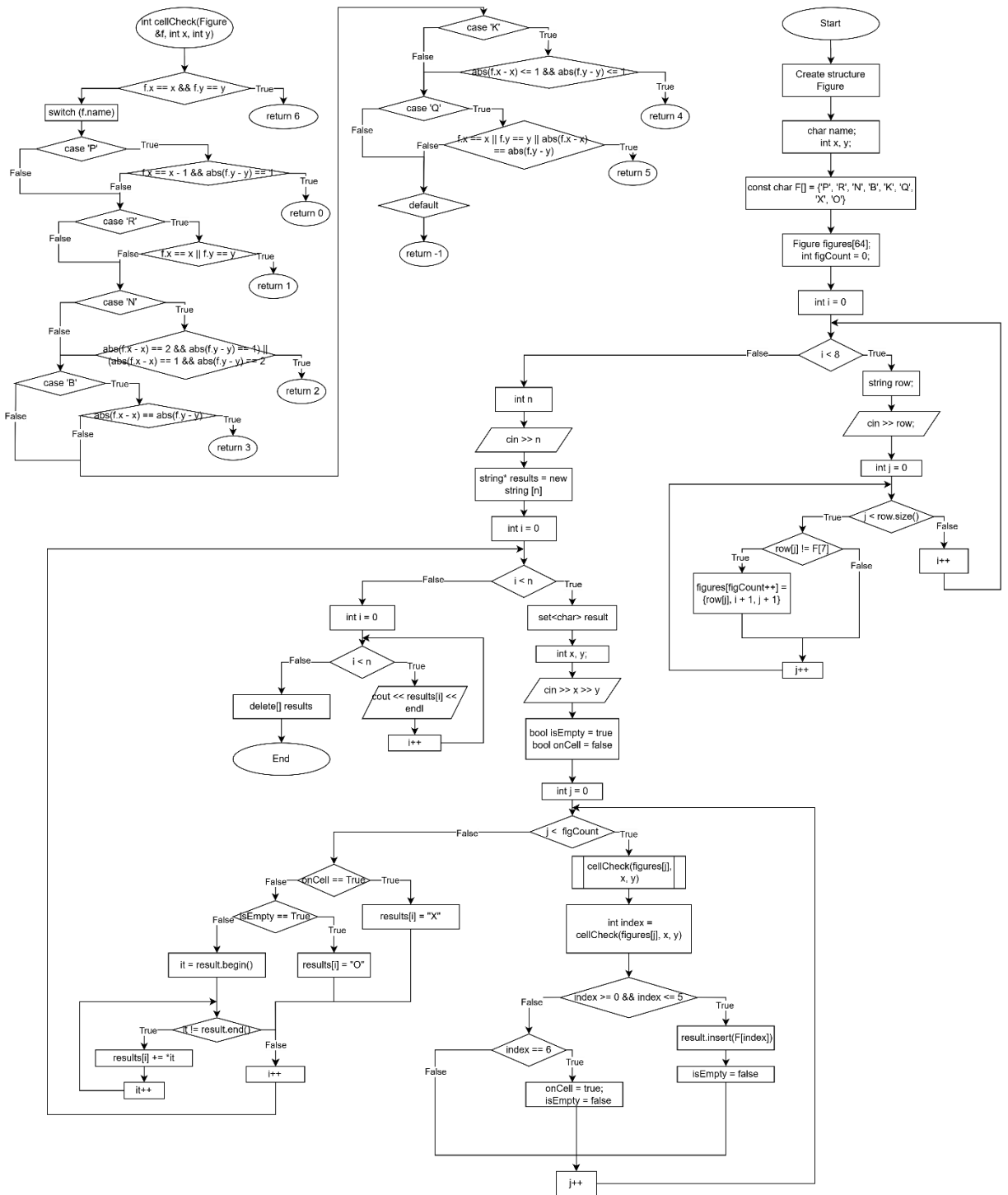
Обмеження

$$1 \leq N, K \leq 1000$$

$$0 \leq a_i \leq 100$$

2) Дизайн виконання завдань:

Завдання №5 Algotester Lab 6 – 2



3) Код програм з посиланням на зовнішні ресурси:

Завдання №1 VNS Lab 6 - 10

```

#include <stdio.h> //printf fgets
#include <string.h> //strtok strcat strcpy
#include <ctype.h> //isalpha isdigit

int wordType(const char *word) {
    int hasLetters = 0, hasDigits = 0;
    for (int i = 0; word[i]; i++) {
        if (isalpha(word[i])) hasLetters = 1;
    }
}

```

```

        if (isdigit(word[i])) hasDigits = 1;
    }
    if (hasLetters && hasDigits) return 2;
    if (hasLetters) return 1;
    if (hasDigits) return 0;
    return -1;
}

int main() {
    const int bufferSize = 255;
    char s[bufferSize], onlyDigits[bufferSize] = "", onlyLetters[bufferSize] =
"", mixed[bufferSize] = "";

    printf("Input text: ");
    fgets(s, bufferSize, stdin);
    s[strcspn(s, "\n")] = 0;

    char *word = strtok(s, " ");
    while (word != NULL) {
        switch (wordType(word)) {
            case 0: strcat(onlyDigits, word); strcat(onlyDigits, " "); break;
            case 1: strcat(onlyLetters, word); strcat(onlyLetters, " "); break;
            case 2: strcat(mixed, word); strcat(mixed, " "); break;
        }
        word = strtok(NULL, " ");
    }

    printf("Output text: %s%s%s\n", onlyDigits, onlyLetters, mixed);

    return 0;
}

```

https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/390/files#diff-6391f0757f3a4694c47a68e7c6db65504f6a4ac9457e9a53609602ba740d912f

Завдання №2 VNS Lab 8 – 10

```

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

struct INFORMATION {
    char carrier[40];
    float volume;
    char title[40];
    char author[40];
};

```

```

//к-сть записів для початкового створення файлу
const int N = 2;

void create_file(const char* filename) {
    FILE* file = fopen(filename, "wb"); // відкриваємо файл для запису в
двійковому режимі
    if (file == NULL) {
        printf("ERROR: Unable to open file for writing.\n");
        exit(1);
    }

    struct INFORMATION info; //створюємо змінну типу INFORMATION

    for (int i = 0; i < N; i++) {

        printf("Enter carrier: ");
        scanf("%s", info.carrier);
        printf("Enter volume: ");
        scanf("%f", &info.volume);
        printf("Enter title: ");
        scanf("%s", info.title);
        printf("Enter author: ");
        scanf("%s", info.author);

        //записуємо структуру в файл
        fwrite(&info, sizeof(struct INFORMATION), 1, file);
        if (ferror(file)) {
            printf("ERROR: Problem while writing to file.\n");
            fclose(file);
            exit(2);
        }
    }
    fclose(file);
}

void print_file(const char* filename) {
    FILE* file = fopen(filename, "rb"); //відкриваємо файл для читання
    if (file == NULL) {
        printf("ERROR: Unable to open file for reading.\n");
        exit(3);
    }

    struct INFORMATION info;
    printf("File contents:\n");

    //читаємо файл по одному запису
    while (fread(&info, sizeof(struct INFORMATION), 1, file) == 1) {
        printf("Carrier: %s, Volume: %.2f, Title: %s, Author: %s\n",
            info.carrier, info.volume, info.title, info.author);
    }
}

```

```

    fclose(file);
}

void delete_first_with_volume(const char* filename, float target_volume) {
    FILE* file = fopen(filename, "rb");
    if (file == NULL) {
        printf("ERROR: Unable to open file for reading.\n");
        exit(4);
    }

    FILE* temp_file = fopen("temp.dat", "wb");
    if (temp_file == NULL) {
        printf("ERROR: Unable to create temporary file.\n");
        fclose(file);
        exit(5);
    }

    struct INFORMATION info;
    int deleted = 0;

    while (fread(&info, sizeof(struct INFORMATION), 1, file) == 1) {
        if (!deleted && info.volume == target_volume) {
            deleted = 1;
        } else {
            fwrite(&info, sizeof(struct INFORMATION), 1, temp_file);
        }
    }

    fclose(file);
    fclose(temp_file);

    remove(filename);
    rename("temp.dat", filename);

    if (deleted) {
        printf("First element with volume %.2f removed.\n", target_volume);
    } else {
        printf("Element with volume %.2f not found.\n", target_volume);
    }
}

void add_information_before(const char* filename, struct INFORMATION newInfo, int
position) {
    FILE* file = fopen(filename, "rb");
    if (file == NULL) {
        printf("ERROR: Unable to open file for reading.\n");
        exit(6);
    }

    FILE* temp_file = fopen("temp.dat", "wb");

```

```

    if (temp_file == NULL) {
        printf("ERROR: Unable to create temporary file.\n");
        fclose(file);
        exit(7);
    }

    struct INFORMATION info;
    int index = 1;

    //читаємо оригінальний файл
    while (fread(&info, sizeof(struct INFORMATION), 1, file) == 1) {
        //додаємо новий елемент перед певною позицією
        if (index == position) {
            fwrite(&newInfo, sizeof(struct INFORMATION), 1, temp_file); // запис
            //Нового елемента
        }
        fwrite(&info, sizeof(struct INFORMATION), 1, temp_file); // запис
        //Поточного елемента
        index++;
    }

    fclose(file);
    fclose(temp_file);

    remove(filename); // видаляємо оригінальний файл
    rename("temp.dat", filename); // перейменовуємо тимчасовий файл в
    //оригінальний

    printf("New element added before position %d.\n", position);
}

int main() {
    const char* filename = "information.dat";

    create_file(filename);

    print_file(filename);

    struct INFORMATION newInfo = {"USB", 5.0, "Programming Guide", "Smith"};
    add_information_before(filename, newInfo, 2);

    delete_first_with_volume(filename, 3.0);

    printf("\nAfter deletion and addition:\n");
    print_file(filename);

    return 0;
}

```

https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/390/files#diff-0ffc4318b012fa053084b03a88a9b265479040b267f2bc8c2aa9230345bc14

Завдання №3 VNS Lab 9 – 10

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <cstring>
#include <cctype>
#include <climits>

using namespace std;

#define MAX_WORDS 50
#define MAX_LINE_LENGTH 256

bool has_same_starting_letter(char line[]) {
    char* words[MAX_WORDS];
    int word_count = 0;

    char* token = strtok(line, " \n");
    while (token != NULL && word_count < MAX_WORDS) {
        words[word_count++] = token;
        token = strtok(NULL, " \n");
    }

    for (int i = 0; i < word_count - 1; ++i) {
        for (int j = i + 1; j < word_count; ++j) {
            if (tolower(words[i][0]) == tolower(words[j][0])) {
                return true; // Якщо слова мають однакову першу букву
            }
        }
    }
    return false; // Якщо всі перші букви різні
}

void print_shortest_word_in_file(const string& filename) {
    ifstream file(filename);
    if (!file) {
        cerr << "Не вдалося відкрити файл " << filename << ".\n";
        return;
    }

    string shortest_word;
    string word;
    size_t min_length = INT_MAX;

    while (file >> word) {
        if (word.length() < min_length) {
```

```

        min_length = word.length();
        shortest_word = word;
    }
}

file.close();

if (!shortest_word.empty()) {
    cout << "Найкоротше слово в файлі " << filename << ": " << shortest_word
<< endl;
} else {
    cout << "Файл " << filename << " порожній або не містить слів.\n";
}
}

int main() {
    ifstream inputFile("F1.txt");
    if (!inputFile) {
        cerr << "Не вдалося відкрити файл F1.txt.\n";
        return 1;
    }

    ofstream outputFile("F2.txt");
    if (!outputFile) {
        cerr << "Не вдалося створити файл F2.txt.\n";
        return 1;
    }

    char line[MAX_LINE_LENGTH];

    while (inputFile.getline(line, MAX_LINE_LENGTH)) {
        char line_copy[MAX_LINE_LENGTH];
        strcpy(line_copy, line);

        if (!has_same_starting_letter(line_copy)) {
            outputFile << line << endl;
        }
    }

    inputFile.close();
    outputFile.close();

    print_shortest_word_in_file("F2.txt");

    return 0;
}

```

https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/390/files#diff-8f2bc5a5a9576a168c198d80ec2c7149755319e23d1f93b4f7bbeac253030125

Завдання №4 Algotester Lab 4 – 3

Без STL:

```
#include <iostream>

using namespace std;

void quick_sort(int arr[], int left, int right) {
    if (left >= right) return;
    int middle = arr[(left + right) / 2];
    int i = left, j = right;
    while (i <= j) {
        while (arr[i] < middle)
            i++;
        while (arr[j] > middle)
            j--;
        if (i <= j) {
            swap(arr[i], arr[j]);
            i++;
            j--;
        }
    }
    quick_sort(arr, left, j);
    quick_sort(arr, i, right);
}

void insert_unique(int source[], int &source_size, int result_array[], int
&result_size) {
    for (int i = 0; i < source_size; ++i) {
        bool is_duplicate = false;
        for (int j = 0; j < result_size; ++j) {
            if (result_array[j] == source[i]) {
                is_duplicate = true;
                break;
            }
        }
        if (!is_duplicate) {
            result_array[result_size++] = source[i];
        }
    }
}

int main() {
    int N;
    cin >> N;
    int numbers[1000];
    for (int i = 0; i < N; ++i) {
        cin >> numbers[i];
    }
}
```

```

int remainder0[1000], remainder1[1000], remainder2[1000];
int size0 = 0, size1 = 0, size2 = 0;

//три масиви за остачами
for (int i = 0; i < N; ++i) {
    if (numbers[i] % 3 == 0) {
        remainder0[size0++] = numbers[i];
    } else if (numbers[i] % 3 == 1) {
        remainder1[size1++] = numbers[i];
    } else {
        remainder2[size2++] = numbers[i];
    }
}

// Сортвання
quick_sort(remainder0, 0, size0 - 1);
quick_sort(remainder1, 0, size1 - 1);
quick_sort(remainder2, 0, size2 - 1);

//масив з остачею 1 по спаданню
for (int i = 0; i < size1 / 2; ++i) {
    swap(remainder1[i], remainder1[size1 - 1 - i]);
}

// об'єднання масиву
int result[1000];
int result_size = 0;

insert_unique(remainder0, size0, result, result_size);
insert_unique(remainder1, size1, result, result_size);
insert_unique(remainder2, size2, result, result_size);

cout << result_size << endl;
for (int i = 0; i < result_size; ++i) {
    cout << result[i] << " ";
}
cout << endl;

return 0;
}

```

https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/390/files#diff-9eb9ef0b8cc9db7376dd21e1b55677ac11ed19cfff45c8daf845bf9f431bcb

3 STL:

```

#include <iostream>
#include <algorithm>

using namespace std;

```

```

bool customComparator(int a, int b) {
    //остача парна сортуємо по зростанню
    if (a % 3 == 0 || a % 3 == 2) {
        return a < b;
    } else {
        return a > b; //по спаданню
    }
}

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

    int numbers[1000];
    for (int i = 0; i < N; ++i) {
        cin >> numbers[i];
    }

    int* div_by_3_0 = partition(numbers, numbers + N, [](int x) { return x % 3 == 0; });
    int* div_by_3_1 = partition(div_by_3_0, numbers + N, [](int x) { return x % 3 == 1; });

    sort(numbers, div_by_3_0, customComparator);
    sort(div_by_3_0, div_by_3_1, customComparator);
    sort(div_by_3_1, numbers + N, customComparator);

    int* unique_arr = unique(numbers, numbers + N);

    int M = unique_arr - numbers;
    cout << M << endl;

    for (int i = 0; i < M; ++i) {
        cout << numbers[i] << " ";
    }
    cout << endl;

    return 0;
}

```

https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/390/files#diff-fa28a736ed85d7e8055fa7f8b9c11b5575c0feb6f14a1e8e7fa8cccbebcf4d3c

Завдання №5 Algotester Lab 6 – 2

```

#include <iostream>
#include <cstring>
#include <cmath>

```

```

#include <set>

using namespace std;

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

const char F[] = {'P', 'R', 'N', 'B', 'K', 'Q', 'X', 'O'};

int cellCheck(Figure &f, int x, int y) {
    if (f.x == x && f.y == y)
        return 6; //фігура на клітинці
    switch (f.name) {
        case 'P': //пішак вниз по діагоналі
            if (f.x == x - 1 && abs(f.y - y) == 1)
                return 0; //P
            break;
        case 'R': //тура по горизонталі та вертикалі
            if (f.x == x || f.y == y)
                return 1; // R
            break;
        case 'N': //кінь в формі "Г"
            if ((abs(f.x - x) == 2 && abs(f.y - y) == 1) || (abs(f.x - x) == 1 &&
abs(f.y - y) == 2))
                return 2; //N
            break;
        case 'B': //слон атакує по діагоналі
            if (abs(f.x - x) == abs(f.y - y))
                return 3; //B
            break;
        case 'K': //король сусідні клітинки
            if (abs(f.x - x) <= 1 && abs(f.y - y) <= 1)
                return 4; //K
            break;
        case 'Q': //королева атакує по горизонталі, вертикалі та діагоналі
            if (f.x == x || f.y == y || abs(f.x - x) == abs(f.y - y))
                return 5; //Q
            break;
        default:
            break;
    }
    return -1;
}

int main() {
    Figure figures[64];
    int figCount = 0; //лічильник к-сті фігур

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

```

```

    string row;
    cin >> row;
    for (int j = 0; j < row.size(); j++) {
        if (row[j] != F[7]) {
            figures[figCount++] = {row[j], i + 1, j + 1}; //фігура з
координатами
        }
    }
}

int n;
cin >> n;
string* results = new string [n];

for (int i = 0; i < n; i++) {
    set<char> result;
    int x, y;
    cin >> x >> y;
    bool isEmpty = true;
    bool onCell = false;

    for (int j = 0; j < figCount; j++) {
        int index = cellCheck(figures[j], x, y);
        if (index >= 0 && index <= 5) { //атака клітинки
            result.insert(F[index]);
            isEmpty = false;
        } else if (index == 6) { //тут фігура
            onCell = true;
            isEmpty = false;
        }
    }

    if (onCell) {
        results[i] = "X"; //є фігура
    } else if (isEmpty) {
        results[i] = "O"; //не атакована
    } else {
        for (set<char>::iterator it = result.begin(); it != result.end();
++it) {
            results[i] += *it;
        }
    }
}

for (int i = 0; i < n; i++) {
    cout << results[i] << endl;
}

delete[] results;

```

```
    return 0;
}
```

https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/390/files#diff-26c36c74f93766051876483fbc84467cbb6b6690b6c7ffcea1b16042a957dbce

Завдання №6 Class Practice Task – 1

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

using namespace std;

enum FileOpResult { Success, Failure };

FileOpResult write_to_file(const char *name, const char *content) {

    ofstream outf(name);

    if (!outf) {
        cerr << "Error: can't open for writing" << endl;
        return Failure;
    }

    outf << content;

    if (outf.fail()) {
        cerr << "Error: writing to the file failed" << endl;
        return Failure;
    }

    outf.close();
    if (outf.fail()) {
        cerr << "Error: closing the file failed" << endl;
        return Failure;
    }

    return Success;
}

int main() {
    const char *filename = "SomeText";
    char text_content[256];

    cout << "Enter the content to write into the file: ";
    cin.getline(text_content, 256);
}
```

```

FileOpResult result = write_to_file(filename, text_content);

if (result == Success) {
    cout << "File written successfully." << endl;
} else {
    cout << "Failed to write to the file." << endl;
}

return 0;
}

```

https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/390/files#diff-b86855d0f26767d9d88a812828761d520b5385cd40d7915796b0521cb55925a4

Class Practice Task – 2

```

#include <iostream>
#include <fstream>

using namespace std;

enum FileOpResult { Success, Failure };

FileOpResult copy_file(const char *file_from, const char *file_to) {

    ifstream infile(file_from, ios::binary);
    if (!infile) {
        cerr << "Error: Source file " << file_from << " could not be opened" <<
endl;
        return Failure;
    }

    ofstream outfile(file_to, ios::binary);
    if (!outfile) {
        cerr << "Error: Destination file " << file_to << " could not be created
or opened" << endl;
        return Failure;
    }

    outfile << infile.rdbuf();

    if (outfile.fail()) {
        cerr << "Error: writing to the destination file failed" << endl;
        return Failure;
    }

    infile.close();
    if (infile.fail()) {
        cerr << "Error: closing the source file failed" << endl;

```

```

        return Failure;
    }

    outfile.close();
    if (outfile.fail()) {
        cerr << "Error: closing the destination file failed" << endl;
        return Failure;
    }

    return Success;
}

int main() {
    const char *file_from = "source.txt";
    const char *file_to = "destination.txt";

    FileOpResult result = copy_file(file_from, file_to);

    if (result == Success) {
        cout << "File copied successfully." << endl;
    } else {
        cout << "Failed to copy the file." << endl;
    }

    return 0;
}

```

https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/390/files#diff-4bf08f5cfb351e69dcf5f19221dcc5cddcdc1e5c3cc72b891dc92a68832bc981

Завдання №7 Self Practice Work

```

#include <iostream>
#include <algorithm>

using namespace std;

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

    int a[1000];
    bool exists[101] = {false};

    int size = 0;

    for (int i = 0; i < N; i++) {
        int x;
        cin >> x;
    }
}

```



```

        if (!exists[x]) {
            a[size++] = x;
            exists[x] = true;
        }
    }

    sort(a, a + size);

    K %= size;
    int rotated[1000];

    for (int i = 0; i < size; i++) {
        rotated[i] = a[(i + K) % size];
    }

    cout << size << endl;
    for (int i = 0; i < size; i++) {
        cout << rotated[i] << " ";
    }
    cout << endl;

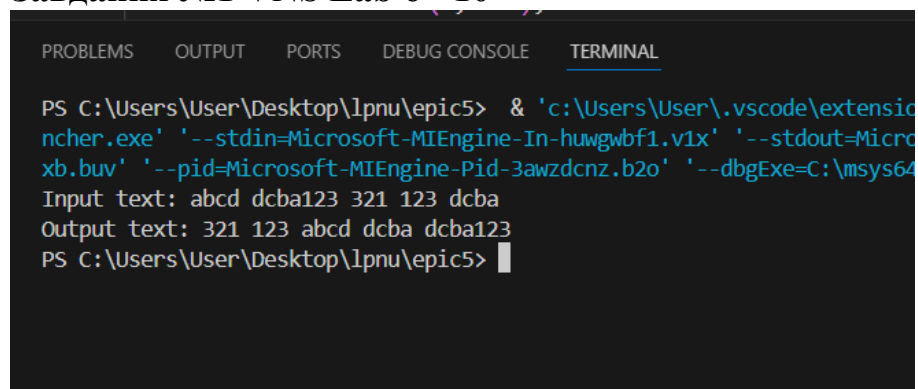
    return 0;
}

```

https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/390/files#diff-0aef8e8c7d385374588b9b76026aaebfc1ecfcfe6041bb847cb8454c88cdb58d

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

Завдання №1 VNS Lab 6 - 10



```

PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL
PS C:\Users\User\Desktop\lpnu\epic5> & 'c:\Users\User\.vscode\extensions\ms-vscode.cpptools\bin\nchcr.exe' '--stdin=Microsoft-MIEngine-In-huwwgbf1.v1x' '--stdout=Microsoft-MIEngine-Out-huwwgbf1.v1x' '--pid=Microsoft-MIEngine-Pid-3awzdcnz.b2o' '--dbgExe=C:\msys64\bin\gdb.exe'
Input text: abcd dcba123 321 123 dcba
Output text: 321 123 abcd dcba dcba123
PS C:\Users\User\Desktop\lpnu\epic5>

```

Планований час: 40 хв. Фактичний: 50 хв.

Завдання №2 VNS Lab 8 – 10

```
ncher.exe' '--stdin=Microsoft-MIEngine-In-xm505bxm.fnk' '--stdout=Micros
gh.m5h' '--pid=Microsoft-MIEngine-Pid-0gyskdd2.osn' '--dbgExe=C:\msys64\
Enter carrier: CD
Enter volume: 3
Enter title: Base
Enter author: John
Enter carrier: DVD
Enter volume: 12
Enter title: Base1
Enter author: Adam
File contents:
Carrier: CD, Volume: 3.00, Title: Base, Author: John
Carrier: DVD, Volume: 12.00, Title: Base1, Author: Adam
New element added before position 2.
First element with volume 3.00 removed.

After deletion and addition:
File contents:
Carrier: USB, Volume: 5.00, Title: Programming Guide, Author: Smith
Carrier: DVD, Volume: 12.00, Title: Base1, Author: Adam
PS C:\Users\User\Desktop\lpnu\epic5>
```

Планований час: 40 хв. Фактичний: 50 хв.

Завдання №3 VNS Lab 9 – 10

```
PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL

PS C:\Users\User\Desktop\lpnu\epic5> & 'c:\Users\User\.vsco
ncher.exe' '--stdin=Microsoft-MIEngine-In-caybp4el.vbg' '--s
4x.0ge' '--pid=Microsoft-MIEngine-Pid-adl0y3vk.3vc' '--dbgEx
Найкоротше слово в файлі F2.txt: Boy
PS C:\Users\User\Desktop\lpnu\epic5>
```

```
≡ F1.txt
1 Apple always amazing
2 Boy green pencil
3 Day is bright and beautiful
4 Sun shines strongly
5 Car drives through city
6 Lovely little red rose
7 Cold weather makes people cautious
8 Winter brings white cat
9 Birds swim near ocean
```

```
≡ F2.txt
1 Boy green pencil
2 Birds swim near ocean
3
```

Планований час: 1 год. Фактичний: 1 год.

[illegible]


```
PS C:\Users\User\Desktop\lpnu\epic5> & 'c:\Users\User\.vsco
ncher.exe' '--stdin=Microsoft-MIEngine-In-f4ohrdje.mn4' '--s
5w.erz' '--pid=Microsoft-MIEngine-Pid-3engfit0.ppk' '--dbg
File copied successfully.
PS C:\Users\User\Desktop\lpnu\epic5>
```

source.txt	destination.txt
1 copy this text	1 copy this text
2	2

Планований час: 40 хв. Фактичний: 50 хв.

Завдання №7 Self Practice Work

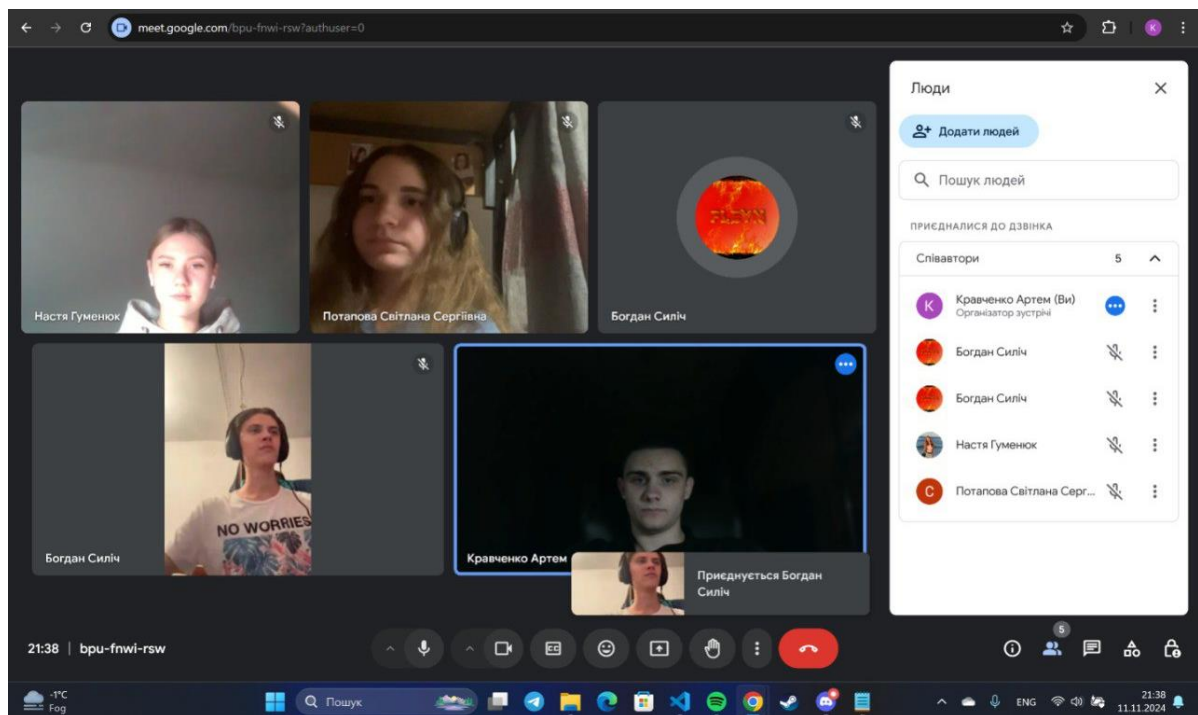
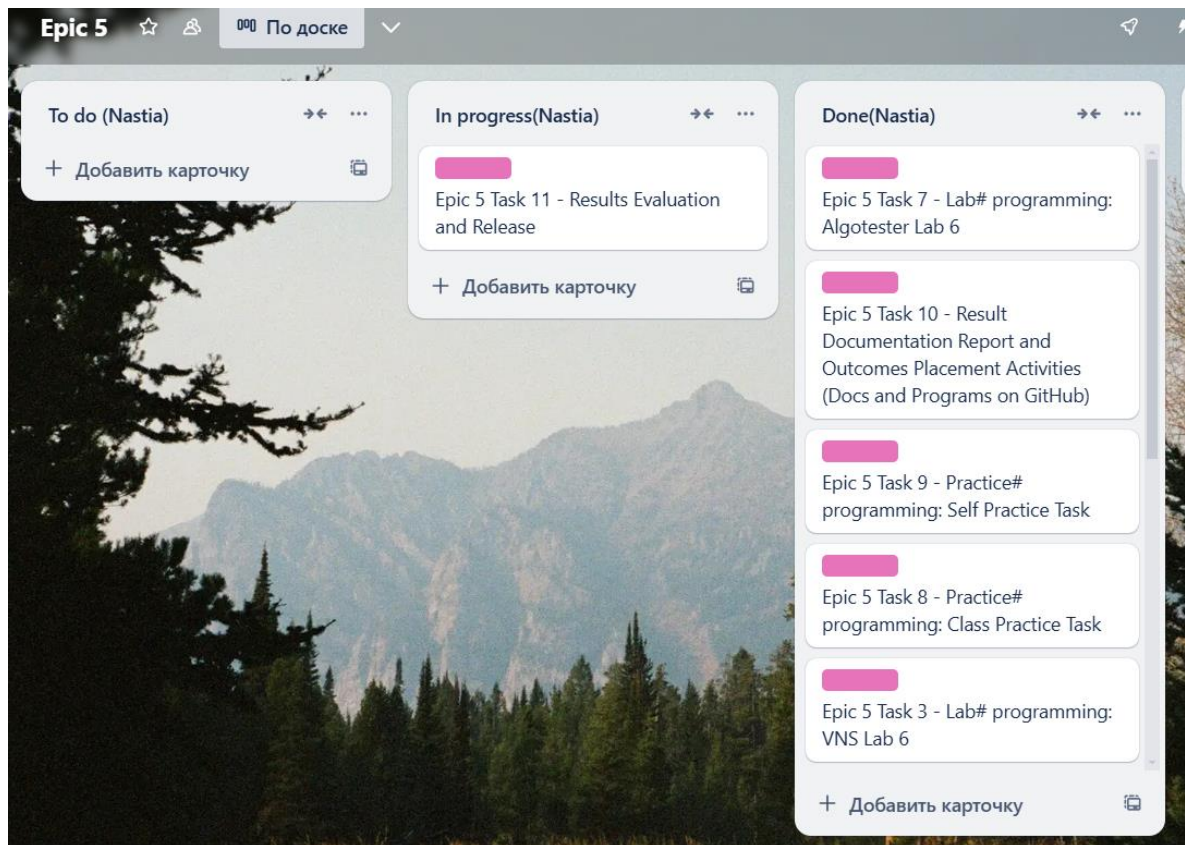
```
PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL

PS C:\Users\User\Desktop\lpnu\epic5> & 'c:\Users\User\.vsco
ncher.exe' '--stdin=Microsoft-MIEngine-In-nmhikc3d.gmq' '--s
o5.hy2' '--pid=Microsoft-MIEngine-Pid-0lnt1qoe.jaq' '--dbgEx
10 3
1 2 2 3 3 3 4 5 6 7
7
4 5 6 7 1 2 3
PS C:\Users\User\Desktop\lpnu\epic5> ^C
PS C:\Users\User\Desktop\lpnu\epic5>
PS C:\Users\User\Desktop\lpnu\epic5> & 'c:\Users\User\.vsco
ncher.exe' '--stdin=Microsoft-MIEngine-In-vaifhfw3.cof' '--s
oz.wpj' '--pid=Microsoft-MIEngine-Pid-4nane10q.fgk' '--dbgEx
10 11
5 6 2 3 1 2 3 3 4 7
7
5 6 7 1 2 3 4
PS C:\Users\User\Desktop\lpnu\epic5> █
```

Створено	Компілятор	Результат	Час (сек.)	Пам'ять (МіБ)	Дії
20 годин тому	C++ 23	Зараховано	0.003	1.414	Перегляд
20 годин тому	C++ 23	Неправильна відповідь 1	0.002	0.918	Перегляд
20 годин тому	C++ 23	Неправильна відповідь 1	0.002	0.914	Перегляд

Планований час: 40 хв. Фактичний: 40 хв.

5) *Кооперація з командою:*



Висновок: Виконуючи 5 епік, я ознайомила з основними принципами роботи з файлами у C++. Вивчила текстові та бінарні файли, зокрема операції відкриття, читання, запису та закриття. Опанувала перевірку стану файлу для обробки помилок і забезпечення надійності. Особливу увагу приділила базовим операціям із символами та рядками, таким як конкатенація, порівняння і пошук. Відпрацювала форматування текстових файлів при

записі даних і методи обробки рядків, отриманих із файлу. Дослідження принципів роботи з бінарними файлами дозволило зрозуміти їхню ефективність і застосування у програмах. Окрім цього, ознайомилася зі стандартними бібліотеками для роботи з файлами (ifstream, ofstream, fstream). Також зустрілась з командою для створення дошки Trello та обговорення поставлених задач.