

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:**

Студент групи ШІ-11

Цяпа Остап Андрійович

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:

Опанувати вивчений матеріал на практиці, а саме попрацювати з файлами як у мові C та C++, вдосконалити роботу зі стрічками в C++.

## Теоретичні відомості:

- 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.

### 1) Індивідуальний план опрацювання теорії:

- Тема №1: Вступ до роботи з файлами:
  - o Джерела інформації:
    - Статті.
  - <https://www.youtube.com/watch?v=FeNqHytI0fA>
  - Що опрацьовано:
    - o Основні операції з файлами: відкриття, читання, запис, закриття
    - o Перевірка стану файлу: перевірка помилок, кінець файлу
    - o Базові приклади читання та запису в файл
- Запланований час на вивчення 2 години.  
Витрачений час 2 години.
- Тема №2: Символи і рядкові змінні:
  - o Джерела інформації:
    - Статті.
  - <https://www.youtube.com/watch?v=1FkTJYm-T34&t=80s>
  - Що опрацьовано:
    - o Робота з char та string: основні операції і методи
    - o Стрічкові літерали та екранування символів
    - o Конкатенація, порівняння та пошук у рядках
- Запланований час на вивчення 2 години.  
Витрачений час 2 години.
- Тема №3: Текстові файли:
  - o Джерела інформації:
    - Статті.
  - <https://www.youtube.com/watch?v=SSNJ7alki-E&t=3834s>
  - Що опрацьовано:
    - o Особливості читання та запису текстових файлів
    - o Обробка рядків з файлу: getline, ignore, peek
    - o Форматування тексту при записі: setw, setfill, setprecision
    - o Парсинг текстових файлів: розділення на слова, аналіз структури
    - o Обробка помилок при роботі з файлами
- Запланований час на вивчення 2 години.

Витрачений час 2 години.

- Тема №4:Бінарні файли:
  - Джерела інформації:
    - Статті.  
[https://www.youtube.com/watch?v=\\_h5eHf65lgs](https://www.youtube.com/watch?v=_h5eHf65lgs)
  - 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
- Запланований час на вивчення 2 години.  
Витрачений час 2 години.
- Тема № 5:Стандартна бібліотека та робота з файлами:
  - Джерела інформації:
    - Статті.  
<https://www.youtube.com/watch?v=SSNJ7alki-E&t=3932s>
  - Що опрацьовано
    - Огляд стандартної бібліотеки для роботи з файлами
    - Потоки вводу/виводу: ifstream, ofstream, fstream
    - Обробка помилок при роботі з файлами
- Запланований час на вивчення 2 години.  
Витрачений час 2 години.
- Тема №6:Створення й використання бібліотек:
  - Джерела інформації:
    - Статті.  
<https://www.youtube.com/watch?v=mnwDpO4zqLA&t=433s>
  - Що опрацьовано
    - 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
- Запланований час на вивчення 2 години.  
Витрачений час 2 години.

Також користувався Copilot який давав відповіді на конкретні питання по коду.

## Виконання роботи:

### 1. Опрацювання завдання до програм.

Завдання №1

#### VNS LAB 6 – TASK 1 (VARIANT 9)

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.

9. Надрукувати всі слова-паліндроми, які є в цьому рядку(див 1 варіант).

#### Завдання №2

##### **VNS LAB 8 – TASK 1 (VARIANT 9)**

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 "Пацієнт":

- прізвище, ім'я, по батькові;
- домашня адреса;
- номер медичної карти;
- номер страхового поліса.

Знищити елемент із заданим номером медичної карти, додати 2 елементи в початок файлу.

#### Завдання №3

##### **VNS LAB 9 – TASK 1 (VARIANT 9)**

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

- 1) Скопіювати з файлу F1 у файл F2 всі рядки, які містять тільки одне слово.
- 2) Знайти найдовше слово у файлі F2.

#### Завдання №4

##### **ALGOTESTER LAB 4 (VARIANT 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.

#### Завдання №5

##### **ALGOTESTER LAB 4 (VARIANT 3.2)**

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.

**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 №6

### **ALGOTESTER LAB 6 (VARIANT 3)**

У Клінта в черговий раз виключилось світло і йому немає чим зайнятися. Так як навіть це не заставить його подивитися збережені відео про програмування на ютубі - він вирішив придумати свою гру на основі sudoku.

Гра виглядає так: Є поле розміром  $N \times N$ , в якому частина клітинок заповнена цифрами, а частина клітинок пусті (позначаються нулем). Також у нього є  $Q$  пар координат  $X$  та  $Y$ . Завданням гри є написати до кожної координати скільки чисел туди можна вписати (якщо вона пуста) і які це числа (обов'язково в посортовані по зростанню!). В клітинку можна вписати лише ті числа, які не зустрічаються в рядку та стовбці, які перетинаються у цій клітинці. Під час гри поле не міняється! Також необов'язково, щоб це було валідне sudoku! Якщо є клітинка, в яку не можна вписати ніяку цифру - виведіть 0. Також допускаються рядки та стовпці, в яких цифра записана кілька разів.

### **Вхідні дані**

У першому рядку ціле число  $N$  - розмір поля для гри

У  $N$  наступних рядках стрічка row $i$  яка складається з  $N$  цифр -  $i$ -й рядок.

Ціле число  $Q$  - кількість запитань

У наступних  $Q$  рядках 2 цілих числа  $x_j$ ,  $y_j$  - координати клітинок  $j$ -го запиту

### **Вихідні дані**

$Q$  разів відповідь у наступному форматі:

Натуральне число  $M$  - кількість цифр, які можна вписати в клітинку

$M$  цифр розділених пробілом - можливі цифри

## Завдання №7

### CLASS PRACTICE WORK

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

*Реалізувати функцію створення файлу 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 файлу у інший файл

*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.

## Завдання №8

### SELF PRACTICE WORK ALGOTESTER

До Тойлет-мена, відомого вам білоруського супергероя, дуже часто звертаються по допомогу різні люди. Проте, очевидно, усім мужній гігант допомогти не зможе — не вистачить часу. Та й не дуже хоче.

Саме тому, коли до нього приходить певне SMS-повідомлення з проханням про допомогу, він погодиться допомогти тоді й лише тоді, коли в цьому повідомленні знайдеться хоча б kk входжень рядка **TOILET**, які не перетинаються.

За заданим повідомленням ss, яке складається з великих латинських символів, виведіть **YES**, якщо Тойлет-мен погодиться допомогти людині, яка написала це повідомлення. У протилежному разі виведіть **NO**.

### Вхідні дані

У першому рядку задано одне ціле число kk — мінімальна кількість незалежних входжень рядка **TOILET**.

Другий рядок містить рядок ss — повідомлення, надіслане Тойлет-мену. Рядок містить лише великі латинські символи.

### Вихідні дані

У єдиному рядку виведіть **YES** або **NO** — відповідь на задачу.

## 2. Вимоги та планувальна оцінка часу виконання завдань: Програма №1

- Важливі деталі для реалізації програми.
- Використовувати функцію `gets()`, а також працювати з варіантом стрічок у мові C. Перевіряти на голосні букви як великі так і малі.
- Плановий час на реалізацію 1.5 години.

#### **Програма №2**

- Важливі деталі для реалізації програми.
- Працювати з файлами як у мові C, не забувати після того як відкрив файл його закрити для коректної роботи програми, а також щоразу перевіряти чи не сталася помилка під час відкриття файла для читання чи запису.
- Плановий час на реалізацію 3 години.

#### **Програма №3**

- Важливі деталі для реалізації програми.
- Працювати з файлами як у мові C++, а саме з директивою `fstream` і її функціями `fstream` та `ofstream`.
- Плановий час на реалізацію 1.5 години.

#### **Програма №4**

- Важливі деталі для реалізації програми.
- Розібратися з функціями директиви `algorithm` для сортування, об'єднання, перетину та симетричної різниці двох масивів.
- Плановий час на реалізацію 2 години.

#### **Програма №5**

- Важливі деталі для реалізації програми.
- Спробувати написати програму без вбудованих функцій директиви `algorithm` (окрім `sort`), використовувати вектор для зберігання значень, цикли для того щоб проходитися по елементах у векторі.
- Плановий час на реалізацію 4 години.

#### **Програма №6**

- Важливі деталі для реалізації програми.
- Написати використовуючи функції директиви `set` та `algorithm`, а саме сортування масиву, видалення однакових елементів та оберт масиву, тобто перенесення елементів відсортованого масиву на  $K$ , ті які були в кінці наперед перенести.
- Плановий час на реалізацію 5 години.

#### **Програма №7**

- Важливі деталі для реалізації програми (задачі 1 та задачі 2).
- Для задачі 1 створити функцію `FileOpResult` яка буде приймати назву файла та його вміст і будемо записувати цей вміст у файл. Важливо перевіряти чи добре йде запис файла і якщо помилка виводити на екран відповідний помилку. Також потрібно не забувати відкривати і закривати файл, щоб не сталася помилка
- Плановий час на реалізацію 4 години.





## Завдання №1

```
1  #include <iostream>
2  #include <cstring> // Для функції gets
3  using namespace std;
4
5  // Функція для перевірки, чи є слово паліндромом
6  bool isPalindrome(const string& str, int start, int end) {
7      while (start < end) {
8          if (str[start] != str[end]) {
9              return false;
10         }
11         start++;
12         end--;
13     }
14     return true;
15 }
16
17 // Функція для друку паліндромів у рядку
18 void printPalindrome(char* str) {
19     char* word = strtok(str, " .");
20     while (word != nullptr) {
21         int len = strlen(word);
22         if (isPalindrome(word, 0, len - 1)) {
23             cout << word << endl;
24         }
25         word = strtok(nullptr, " .");
26     }
27 }
28
29 int main() {
30     char s[256];
31     cout << "Введіть рядок: ";
32     gets(s); // Використовуємо gets для зчитування рядка
33
34     printPalindrome(s); // Викликаємо функцію для друку паліндромів
35
36     return 0;
37 }
```

Рисунок 3.1. Код до програми № 1

```
Введіть рядок: Hello, I am on another level
I
level
```

Рисунок 3.2. Приклад виконання програми № 1

Фактично затрачений час 2 години.

## Завдання №2

```
1  #include <iostream>
2  #include <fstream>
3  #include <cstdio>
4  #include <cstring>
5  #include <vector>
6  #include <algorithm>
7  using namespace std;
8
9  struct Patient {
10     char lastName[40];
11     char firstName[40];
12     char middleName[40];
13     char address[100];
14     int medicalCardNumber;
15     int insurancePolicyNumber;
16 };
17
18 // Функція для введення даних пацієнта з клавіатури
19 Patient inputPatient() {
20     Patient patient;
21     cout << "Введіть прізвище: ";
22     cin >> patient.lastName;
23     cout << "Введіть ім'я: ";
24     cin >> patient.firstName;
25     cout << "Введіть по батькові: ";
26     cin >> patient.middleName;
27     cout << "Введіть адресу: ";
28     cin.ignore();
29     cin.getline(patient.address, 100);
30     cout << "Введіть номер медичної карти: ";
31     cin >> patient.medicalCardNumber;
32     cout << "Введіть номер страхового поліса: ";
33     cin >> patient.insurancePolicyNumber;
34     return patient;
35 }
36
37 // Функція для збереження пацієнтів у файл
38 void savePatientsToFile(const char* filename, const vector<Patient>& patients) {
39     FILE* file = fopen(filename, "wb");
40     if (file == NULL) {
41         cerr << "Помилка при відкритті файлу для запису!" << endl;
42         exit(1);
43     }
44     fwrite(&patients[0], sizeof(Patient), patients.size(), file);
45     if (ferror(file)) {
46         cerr << "Помилка при записі у файл." << endl;
47         exit(2);
48     }
49     fclose(file);
50 }
51
52 // Функція для читання пацієнтів із файлу
53 vector<Patient> readPatientsFromFile(const char* filename) {
54     vector<Patient> patients;
55     FILE* file = fopen(filename, "rb");
```

```

56     if (file == NULL) {
57         cerr << "Помилка при відкритті файлу для читання!" << endl;
58         exit(3);
59     }
60     Patient temp;
61     while (fread(&temp, sizeof(Patient), 1, file)) {
62         patients.push_back(temp);
63     }
64     fclose(file);
65     return patients;
66 }
67
68 // Функція для друку пацієнтів
69 void printPatients(const vector<Patient>& patients) {
70     cout << "Вміст файлу:" << endl;
71     for (const auto& patient : patients) {
72         cout << patient.lastName << " " << patient.firstName << " " << patient.middleName << " "
73             << patient.address << " " << patient.medicalCardNumber << " " << patient.insurancePolicyNumber << endl;
74     }
75 }
76
77 // Функція для знищення пацієнта із заданим номером медичної карти
78 void deletePatientByMedicalCardNumber(vector<Patient>& patients, int cardNumber) {
79     auto it = remove_if(patients.begin(), patients.end(),
80         [cardNumber](const Patient& patient) { return patient.medicalCardNumber == cardNumber; });
81     if (it != patients.end()) {
82         patients.erase(it, patients.end());
83     } else {
84         cerr << "Пацієнта з номером медичної карти " << cardNumber << " не знайдено." << endl;
85     }
86 }
87
88 // Функція для додавання пацієнтів до файлу
89 void addPatientsToBeginning(const char* filename, const vector<Patient>& newPatients) {
90     vector<Patient> existingPatients = readPatientsFromFile(filename);
91     existingPatients.insert(existingPatients.begin(), newPatients.begin(), newPatients.end());
92     savePatientsToFile(filename, existingPatients);
93 }
94
95 int main() {
96     const char* filename = "patients.dat";
97     vector<Patient> patients;
98
99     // Введення пацієнтів
100     int choice;
101     do {
102         patients.push_back(inputPatient());
103         cout << "Бажаєте додати ще одного пацієнта? (1 - Так, 0 - Ні): ";
104         cin >> choice;
105     } while (choice != 0);
106
107     // Зберігаємо початкових пацієнтів у файл
108     savePatientsToFile(filename, patients);

```

```

109
110 // Зчитуємо та друкуємо пацієнтів з файлу
111 patients = readPatientsFromFile(filename);
112 cout << "Початковий список пацієнтів:" << endl;
113 printPatients(patients);
114
115 // Знищуємо пацієнта з номером медичної карти 23456
116 deletePatientByMedicalCardNumber(patients, 23456);
117
118 // Зберігаємо оновлений список пацієнтів у файл
119 savePatientsToFile(filename, patients);
120
121 // Додаємо нових пацієнтів на початок файлу
122 vector<Patient> newPatients = {
123     {"Ostap", "Tsiapa", "Andriyovych", "Ternopil, 83", 23456, 90123},
124     {"Lionel", "Messi", "Andres", "Barcelona, 10", 56789, 12345}
125 };
126 addPatientsToBeginning(filename, newPatients);
127
128 // Знову зчитуємо та друкуємо пацієнтів з файлу
129 patients = readPatientsFromFile(filename);
130 cout << "Оновлений список пацієнтів:" << endl;
131 printPatients(patients);
132
133 return 0;
134 }
135

```

Рисунок 3.3. Код до програми № 2

```

Введіть прізвище: Danyliuk
Введіть ім'я: Yulia
Введіть по батькові: Dmytrivna
Введіть адресу: Lychakivskogo
Введіть номер медичної карти: 12345
Введіть номер страхового поліса: 98765
Бажаєте додати ще одного пацієнта? (1 - Так, 0 - Ні): 0
Початковий список пацієнтів:
Вміст файлу:
Danyliuk Yulia Dmytrivna Lychakivskogo 12345 98765
Пацієнта з номером медичної карти 23456 не знайдено.
Оновлений список пацієнтів:
Вміст файлу:
Ostap Tsiapa Andriyovych Ternopil, 83 23456 90123
Lionel Messi Andres Barcelona, 10 56789 12345
Danyliuk Yulia Dmytrivna Lychakivskogo 12345 98765

```

Рисунок 3.4. Приклад виконання програми №2

Фактично затрачений час 5 годин.

### Завдання №3

```
1  #include <iostream>
2  #include <fstream>
3  #include <sstream>
4  #include <string>
5  #include <vector>
6  #include <algorithm>
7  using namespace std;
8
9  // Функція для створення файлу F1 з інформацією
10 void createFileF1(const string filename) {
11     ofstream outFile(filename);
12     if (!outFile) {
13         cerr << "Помилка при відкритті файну для запису!" << endl;
14         return;
15     }
16     outFile << "Best\n";
17     outFile << "Cities\n";
18     outFile << "Lv\n";
19     outFile << "Ukraine\n";
20     outFile << "We without a doubt:\n";
21     outFile << "Ternopil\n";
22     outFile << "Kyiv\n";
23     outFile << "Odessa\n";
24     outFile << "And\n";
25     outFile << "Mykachevo\n";
26     outFile.close();
27 }
28
29 // Функція для копіювання рядка з одного файлу в F1 з F2
30 void copySingleWordLines(const string fileF1, const string fileF2) {
31     ifstream inFile(fileF1);
32     ofstream outFile(fileF2);
33     if (!inFile || !outFile) {
34         cerr << "Помилка при відкритті файнів!" << endl;
35         return;
36     }
37     string line;
38     while (getline(inFile, line)) {
39         stringstream ss(line);
40         string word;
41         int wordCount = 0;
42         while (ss >> word) {
43             ++wordCount;
44         }
45         if (wordCount == 1) {
46             outFile << line << "\n";
47         }
48     }
49     inFile.close();
50     outFile.close();
51 }
52
53 // Функція для знаходження найдовшого слова у файлі F2
54 string findLongestWordInFile(const string filename) {
55     ifstream inFile(filename);
56     if (!inFile) {
57         cerr << "Помилка при відкритті файну для читання!" << endl;
58         return "";
59     }
60     string longestWord;
61     string line;
62     while (getline(inFile, line)) {
63         if (line.length() > longestWord.length()) {
64             longestWord = line;
65         }
66     }
67     inFile.close();
68     return longestWord;
69 }
70
71 int main() {
72     const string fileF1 = "F1.txt";
73     const string fileF2 = "F2.txt";
74
75     // Створення файлу F1
76     createFileF1(fileF1);
77
78     // Копіюємо рядок з одним словом з файлу F1 у файл F2
79     copySingleWordLines(fileF1, fileF2);
80
81     // Знаходимо найдовше слово у файлі F2
82     string longestWord = findLongestWordInFile(fileF2);
83     cout << "Найдовше слово у файлі F2: " << longestWord << endl;
84
85     return 0;
86 }
87
```

Рисунок 3.5. Код до програми № 3

**Найдовше слово у файлі F2: Mykachevo**

Рисунок 3.6. Приклад виконання програми № 3

```
≡ F1.txt ×
ai_11 > ostap_tsiapa > epic_5 > ≡ F1.txt
1 Best
2 Cities
3 In
4 Ukraine
5 Are without a doubt:
6 Ternopil
7 Kyiv
8 Odesa
9 And
10 Mykachevo
11
```

Рисунок 3.7. У файлі F1.txt усі рядки.

```
≡ F2.txt ×
ai_11 > ostap_tsiapa > epic_5 > ≡ F2.txt
1 Best
2 Cities
3 In
4 Ukraine
5 Ternopil
6 Kyiv
7 Odesa
8 And
9 Mykachevo
10
```

Рисунок 3.8. У файлі F2.txt тільки ті рядки, які мають одне слово

Фактично затрачений час 3 години.

#### Завдання №4

З використанням STL

```

1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4  #include <set>
5  using namespace std;
6
7  int main() {
8      int N;
9      cin >> N;
10     vector<int> arr(N);
11     for (int i = 0; i < N; ++i) {
12         cin >> arr[i];
13     }
14
15     // Розділяємо масив на три частини
16     vector<int> mod0, mod1, mod2;
17     for (int num : arr) {
18         if (num % 3 == 0) mod0.push_back(num);
19         else if (num % 3 == 1) mod1.push_back(num);
20         else mod2.push_back(num);
21     }
22
23     // Сортуємо кожну частину відповідно до умов
24     sort(mod0.begin(), mod0.end());           // Сортуємо по зростанню
25     sort(mod2.begin(), mod2.end());           // Сортуємо по зростанню
26     sort(mod1.begin(), mod1.end(), greater<int>()); // Сортуємо по спаданням
27
28     // Об'єднуємо всі частини разом в правильному порядку
29     vector<int> result;
30     result.insert(result.end(), mod0.begin(), mod0.end());
31     result.insert(result.end(), mod1.begin(), mod1.end());
32     result.insert(result.end(), mod2.begin(), mod2.end());
33
34     // Видаляємо дублікати
35     auto it = unique(result.begin(), result.end());
36     result.resize(distance(result.begin(), it));
37
38     // Виводимо результуючий масив
39     cout << result.size() << endl;
40     for (int num : result) {
41         cout << num << " ";
42     }
43     cout << endl;
44     return 0;
45 }

```

Рисунок 3.9. Код до програми №4 (з STL)

```
10
1 33 4 8 6 5 2 7 5 0
9
0 6 33 7 4 1 2 5 8
```

Рисунок 3.10. Приклад виконання програми №4 (з STL)

декілька секунд тому	C++ 23	Зараховано	0.003	1.254	<a href="#">Перегляд</a>
----------------------	--------	------------	-------	-------	--------------------------

Рисунок 3.13. Статус задачі на Algotester

Фактично затрачений час 5 годин.

**Завдання №5**

Без використання STL



```

1  #include <iostream>
2  #include <vector>
3  using namespace std;
4
5  // Функція для сортування масиву за допомогою алгоритму Quick Sort
6  void quickSort(vector<int>& arr, int left, int right, bool ascending = true) {
7      int i = left, j = right;
8      int element = arr[(left + right) / 2];
9      while (i <= j) {
10         if (ascending) {
11             while (arr[i] < element) ++i;
12             while (arr[j] > element) --j;
13         } else {
14             while (arr[i] > element) ++i;
15             while (arr[j] < element) --j;
16         }
17         if (i <= j) {
18             swap(arr[i], arr[j]);
19             ++i;
20             --j;
21         }
22     }
23     if (left < j) quickSort(arr, left, j, ascending);
24     if (i < right) quickSort(arr, i, right, ascending);
25 }
26
27 int main() {
28     int N;
29     cin >> N;
30     vector<int> arr(N);
31     for (int i = 0; i < N; ++i) {
32         cin >> arr[i];
33     }
34
35     // Розділяємо масив на три частини
36     vector<int> mod0, mod1, mod2;
37     for (int num : arr) {
38         if (num % 3 == 0) mod0.push_back(num);
39         else if (num % 3 == 1) mod1.push_back(num);
40         else mod2.push_back(num);
41     }
42
43     // Сортуємо кожну частину відповідно до умов
44     if (!mod0.empty()) quickSort(mod0, 0, mod0.size() - 1);
45     if (!mod1.empty()) quickSort(mod1, 0, mod1.size() - 1, false);

```

```

46     if (!mod2.empty()) quickSort(mod2, 0, mod2.size() - 1);
47
48     // Об'єднуємо всі частини разом в правильному порядку
49     vector<int> result;
50     result.insert(result.end(), mod0.begin(), mod0.end());
51     result.insert(result.end(), mod1.begin(), mod1.end());
52     result.insert(result.end(), mod2.begin(), mod2.end());
53
54     // Видаляємо дублікати
55     vector<int> uniqueResult;
56     for (int num : result) {
57         if (uniqueResult.empty() || uniqueResult.back() != num) {
58             uniqueResult.push_back(num);
59         }
60     }
61
62     // Виводимо результуючий масив
63     cout << uniqueResult.size() << endl;
64     for (int num : uniqueResult) {
65         cout << num << " ";
66     }
67     cout << endl;
68
69     return 0;
70 }
71

```

Рисунок 3.14. Код до програми №4 (з STL)

```

5
3 6 8 1 4
5
3 6 4 1 8

```

Рисунок 3.15. Приклад виконання програми №4 (без STL)

декілька секунд тому	C++ 23	Зараховано	0.003	1.203	Перегляд
----------------------	--------	------------	-------	-------	----------

Рисунок 3.16. Статус задачі на Algotester

Фактично затрачений час 4 години.

## Завдання №6

```

1  #include <iostream>
2  #include <vector>
3  #include <set>
4  #include <string>
5
6  using namespace std;
7
8  void readField(vector<vector<int>>& field, int N) {
9      for (int i = 0; i < N; i++) {
10         string row;
11         cin >> row;
12         for (int j = 0; j < N; j++) {
13             field[i][j] = row[j] - '0';
14         }
15     }
16 }
17
18 void processQueries(const vector<vector<int>>& field, int N, int Q) {
19     vector<string> results;
20     for (int t = 0; t < Q; t++) {
21         int x, y;
22         cin >> x >> y;
23         x = x - 1;
24         y = y - 1;
25
26         set<int> cantused;
27         for (int i = 0; i < N; i++) {
28             if (field[x][i] != 0) {
29                 cantused.insert(field[x][i]);
30             }
31         }
32
33         for (int i = 0; i < N; i++) {
34             if (field[i][y] != 0) {
35                 cantused.insert(field[i][y]);
36             }
37         }
38
39         if (field[x][y] != 0) {
40             string result = "1 " + to_string(field[x][y]);
41             results.push_back(result);
42         } else {
43             vector<int> possible;
44             for (int num = 1; num <= N; num++) {
45                 if (cantused.find(num) == cantused.end()) {

```

```

46         possible.push_back(num);
47     }
48 }
49
50     string result = to_string(possible.size());
51     for (int num : possible) {
52         result += " " + to_string(num);
53     }
54     results.push_back(result);
55 }
56 }
57
58     for (const string& res : results) {
59         cout << res << "\n";
60     }
61 }
62
63 int main() {
64     int N;
65     cin >> N;
66
67     vector<vector<int>> field(N, vector<int>(N));
68     readField(field, N);
69
70     int Q;
71     cin >> Q;
72
73     processQueries(field, N, Q);
74
75     return 0;
76 }
77
78

```

Рисунок 3.15. Код до програми №5

```

3
000
100
003
3
1 1
2 3
2 1
2 2 3
1 2
1 1

```

Рисунок 3.16. Приклад виконання програми №5

декілька секунд тому	C++ 23	Зараховано	0.003	1.234	<a href="#">Перегляд</a>
----------------------	--------	------------	-------	-------	--------------------------

Рисунок 3.17. Статус задачі на Algotester

Фактично затрачений час 5 годин.

### Завдання №7

```
1  #include <iostream>
2  #include <fstream>
3  using namespace std;
4
5  enum FileOpResult { Success, Failure };
6
7  // Завдання №1 - Запис текстової стрічки у файл із заданим ім'ям
8  FileOpResult write_to_file(const char* name, const char* content) {
9      if (name == nullptr || content == nullptr) {
10         return Failure;
11     }
12
13     ofstream file(name);
14     if (!file.is_open()) {
15         return Failure;
16     }
17
18     file << content;
19     if (file.fail()) {
20         file.close();
21         return Failure;
22     }
23
24     file.close();
25     return Success;
26 }
27
28 // Завдання №2 - Копіювання вмісту файла у інший файл
29 FileOpResult copy_file(const char* file_from, const char* file_to) {
30     if (file_from == nullptr || file_to == nullptr) {
31         return Failure;
32     }
33
34     ifstream src(file_from, ios::binary);
35     if (!src.is_open()) {
36         return Failure;
37     }
38
39     ofstream dest(file_to, ios::binary);
40     if (!dest.is_open()) {
41         src.close();
42         return Failure;
43     }
44
45     dest << src.rdbuf();
```

Рисунок 3.19. Код до програми №7(задача 1 і 2)

```
Enter filename to write to: file_from
Enter content to write: 12345678 Hellllo Hi
File written successfully.
```

Рисунок 3.20. Приклад виконання програми №7 (задача 1)

```
1 12345678 Hellllo Hi
```

Рисунок 3.21. Запис у файлі (задача 1)

```
46
47     if (dest.fail() || src.fail()) {
48         src.close();
49         dest.close();
50         return Failure;
51     }
52
53     src.close();
54     dest.close();
55     return Success;
56 }
57
58 int main() {
59     // Тест Завдання №1
60     char filename1[256];
61     char content[1024];
62     cout << "Enter filename to write to: ";
63     cin.getline(filename1, 256);
64     cout << "Enter content to write: ";
65     cin.getline(content, 1024);
66
67     FileOpResult result1 = write_to_file(filename1, content);
68     if (result1 == Success) {
69         cout << "File written successfully." << endl;
70     } else {
71         cout << "Failed to write to file." << endl;
72     }
73
74     // Тест Завдання №2
75     char file_from[256];
76     char file_to[256];
77     cout << "Enter source filename to copy from: ";
78     cin.getline(file_from, 256);
79     cout << "Enter destination filename to copy to: ";
80     cin.getline(file_to, 256);
81
82     FileOpResult result2 = copy_file(file_from, file_to);
83     if (result2 == Success) {
84         cout << "File copied successfully." << endl;
85     } else {
86         cout << "Failed to copy file." << endl;
87     }
88
89     return 0;
90 }
91
```

Рисунок 3.22. Код до програми №7 (тест задач 1 і 2)

```
Enter source filename to copy from: file_from
Enter destination filename to copy to: file_to
File copied successfully.
```

Рисунок 3.23. Приклад виконання програми №7 (задача 2)

Фактично затрачений час 2.5 години.

## Завдання №8

```
1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  bool help_or_no(int k, const string& s) {
7      string target = "TOILET";
8      int count = 0;
9      size_t pos = 0;
10
11     while ((pos = s.find(target, pos)) != string::npos) {
12         count++;
13         pos += target.length();
14         if (count >= k) {
15             return true;
16         }
17     }
18
19     return false;
20 }
21
22 int main() {
23     int k;
24     string s;
25     cin >> k;
26     cin.ignore();
27     getline(cin, s);
28
29     if (help_or_no(k, s)) {
30         cout << "YES" << endl;
31     } else {
32         cout << "NO" << endl;
33     }
34
35     return 0;
36 }
37
38
```

Рисунок 3.24. Код до програми №8

```
3
HELPTOILETPLEASETOILETBEGYOUTOILET
YES
```

Рисунок 3.25. Приклад виконання програми №8

Створено	Компілятор	Результат	Час (сек.)	Пам'ять (МіБ)	Дії
декілька секунд тому	C++ 23	Зараховано	0.005	1.266	<a href="#">Перегляд</a>

Рисунок 3.26. Статус задачі на Algotester

Фактично затрачений час 30 хвилин.

**Посилання на пул реквест:** [Epic 5 - Ostap Tsiapa by Ostap2007ter · Pull Request #656 · artificial-intelligence-department/ai\\_programming\\_playground\\_2024](#)

**Висновок:** У рамках практичних і лабораторних робіт блоку №5 я освоїв нові теми, такі як робота з файлами, бінарними файлами, символами, рядковими змінними та текстовими файлами, а також методи їх обробки в мовах програмування C і C++. Завдяки застосуванню цього матеріалу на практиці, я поглибив розуміння принципів роботи з файлами та їхньої реалізації. Крім того, створення блок-схеми для найскладнішої задачі допомогло краще зрозуміти логіку та роботу програми.