Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту

A blue and white logo

Description automatically generated

**Звіт**

**про виконання лабораторних та практичних робіт блоку № 5**

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

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

до:

ВНС Лабораторної Роботи № 6

ВНС Лабораторної Роботи № 8

ВНС Лабораторної Роботи № 9

Алготестер Лабораторної Роботи №4

Алготестер Лабораторної Роботи №6

Практичних Робіт №5

***Виконав:***

студент групи ШІ-14

Вульчин Володимир Сергійович

# **Тема роботи:**

Перелік матеріалу: Робота з Файлами, рядкові змінні та символи, зчитування та запис інформації у файл, використання файлів, бібліотеки.

# **Мета роботи:**

Ознайомитися з вище переліченими списками тем, удосконалити свої знання в теорії та практичному застосуванні вивченого.

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* **Тема №1:** Файли. Бінарні та текстові файли. Використання файлів. Бібліотеки.
* **Тема №2:** Робота з файлами у С++.
* **Тема №3:** Символи та рядкові змінні.
* **Тема №4:** Зчитування та запис у файл.

1. Індивідуальний план опрацювання теорії:

* Тема **№1: Файли. Бінарні та текстові файли. Використання файлів. Бібліотеки.**
  + Джерела Інформації:
    - ChatGPT / Bard / Команда, Одногрупнки
    - Відео.

Відео-теорія на теми:

[Стандартна бібліотека С++.](https://youtu.be/m-WJikuZGuU?t=1014) (Відео від Блогана – базовано! )

[Бінарні файли.](https://youtu.be/2lzVB8bkM8o?t=162) (Максимум інформації – мінімум часу)

* + - Статті.

Теорія на теми:

[Бібліотеки/Класи/Заголовки.](https://acode.com.ua/urok-130-klasy-i-zagolovkovi-fajly/#toc-3) (сайт, як завжди з потужною теорією)

[Заголовкові файли.](https://acode.com.ua/urok-24-zagolovkovi-fajly/) (фактично, закріплення відео від Блогана)

* + Що опрацьовано:
    - Самостійно ознайомився з таким переліком інформації:

1. Текстові файлики.
2. Бінарні файлики.
3. Бібліотеки
4. Використання файликів.
   * Статус: Ознайомлений
   * Початок опрацювання теми: 30.11.2023
   * Звершення опрацювання теми: 12.12.2023

* Тема **№2: Робота з файлами у С++.**
  + Джерела Інформації:
    - ChatGPT / Команда, Одногрупники
    - Відео.

Відео-теорія про:

[Робота з файлами С++.](https://youtu.be/SSNJ7alki-E?t=4809) (Блоган, дає жару! Охопив інформації на життя вперед )

* + - Стаття.

Cтаття про:

[Робота з файлами С++.](https://www.bestprog.net/uk/2020/05/14/c-examples-of-working-with-text-files-modification-of-files-sort-data-in-files-convert-file-data-to-list-ua/#q02) (приклади, багато прикладів ☹)

* + Що опрацьовано:
    - Самостійно ознайомився з таким переліком інформації:

1. Файл fstream
2. Що таке потік вводу та виводу!
3. Як зламати потік 
4. Як налагодити потік 
   * Статус: Ознайомлений
   * Початок опрацювання теми: 31.12.2023
   * Звершення опрацювання теми: 12.13.2023
   * Тема **№3: Символи та рядкові змінні.**
   * Джерела Інформації:
     + ChatGPT / Команда, Одногрупники
     + Відео.

[Символи та рядки.](https://youtu.be/tLlr74WMmt4?t=742)

[Робота з символами та рядками.](https://youtu.be/zQ8YBJCpsx4?t=607)

[Детальніше про застосування бібліотеки fstream.](https://youtu.be/CBnB2fvfu_I?t=1059)

* + - Статті.

Теорія про:

[Рядки С++.](http://cpp.dp.ua/osnovni-funktsiyi-obrobky-symvolnyh-typiv/) (aCode)

[Символи С++.](http://cpp.dp.ua/osnovni-funktsiyi-obrobky-symvolnyh-typiv/) (Теж чудесний сайт, з наведенням прикладів, та ще й блок схем до них!)

Що опрацьовано:

* + - Самостійно ознайомився з таким переліком інформації:

1. Символи. , стандартна бібліотека шаблонів STL.
2. Рядочки, стандартна бібліотека шаблонів STL.
3. Бібліотека функцій string.h
   * Статус: Ознайомлений
   * Початок опрацювання теми: 30.11.2023
   * Звершення опрацювання теми: 12.11.2023

* Тема **№4: Зчитування та запис у файл.**
  + Джерела Інформації:
    - ChatGPT / Команда, Одногрупники
    - Відео.

Відео-теорія про:

[Зчитування з файлу.](https://youtu.be/SSNJ7alki-E?t=2790) (Блоган, ультує! Таймкод: [00:46:30](https://youtu.be/SSNJ7alki-E?t=2790) – [01:03:53](https://youtu.be/SSNJ7alki-E?t=3833))

[Запис у файл.](https://youtu.be/SSNJ7alki-E?t=3834) (Блоганчик! Таймкод: Таймкод: [01:03:54](https://youtu.be/SSNJ7alki-E?t=3834) – [01:09:08](https://youtu.be/SSNJ7alki-E?t=4148))

* + - Стаття.

Cтаття про:

[Зчитування/запис.](https://www.bestprog.net/uk/2019/09/11/examples-of-using-c-tools-for-working-with-files-ua/) (приклади, багато прикладів ☹)

Що опрацьовано:

* + - Самостійно ознайомився з таким переліком інформації:

1) Зчитування файлів

2) Запис файликів

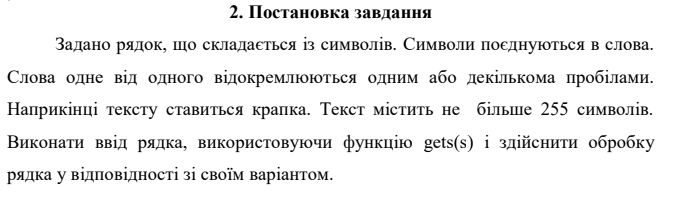
* + Статус: Ознайомлений
  + Початок опрацювання теми: 30.11.2023
  + Звершення опрацювання теми: 12.14.2023

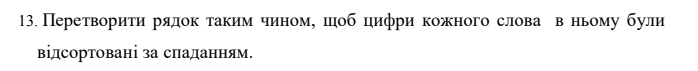
# **Виконання роботи:**

## **1. Опрацювання завдання та вимог до програм та середовища:**

Завдання **№1 VNS Lab 6 - Task 1**

* Варіант завдання: 13



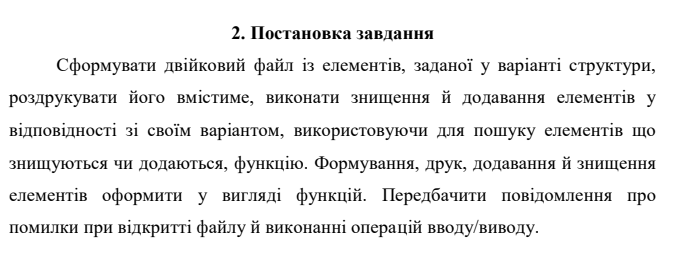


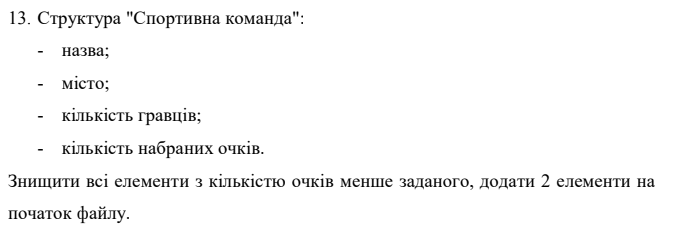
*Image 1. VNS LAB 6 – Task 1 (Умова завдання)*

* Суперове завдання, використовував констанстні змінні, токени. Та замінив функцію gets(), на getline, адже компілятор казав, що це фігня, яка не використовуюється 

Завдання **№2 VNS Lab 8- Task 1**

* Варіант завдання: 13



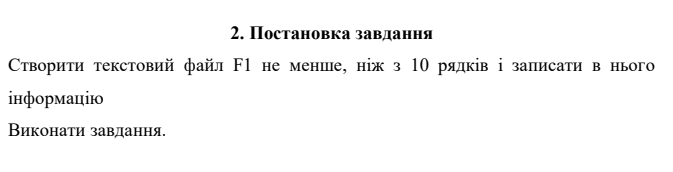


*Images 2. VNS LAB 8 – Task 1 (Умова завдання)*

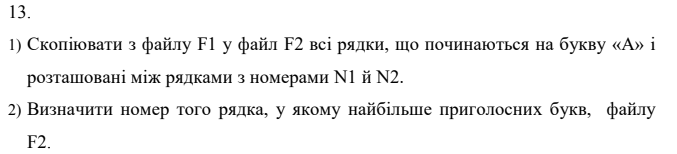
* В цьому завданні рівень цікавості пропорційний рівню складності, адже потрібно використати струтури, оголошувати багато функцій, для різних випадків, знати на хорошому рівні використання бібліотек, та інше.
* В результаті, успішно написаний код, з морем емоцій! (**P.S Для закріплення написав аналогічний код, але у випадку з рядками**)

Завдання **№3 VNS Lab 9 - Task 1**

* Варіант завдання: 13



*Image 3. VNS LAB 9 – Task 1 (Умова завдання)*

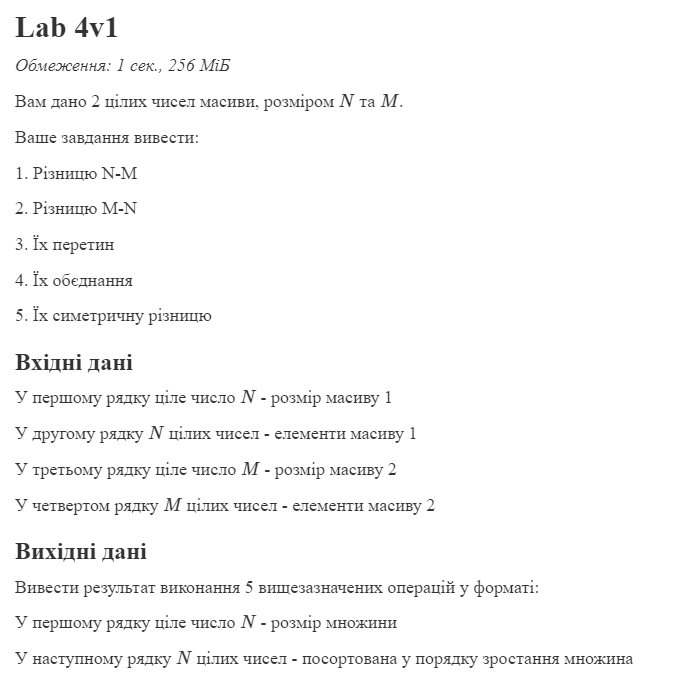
**

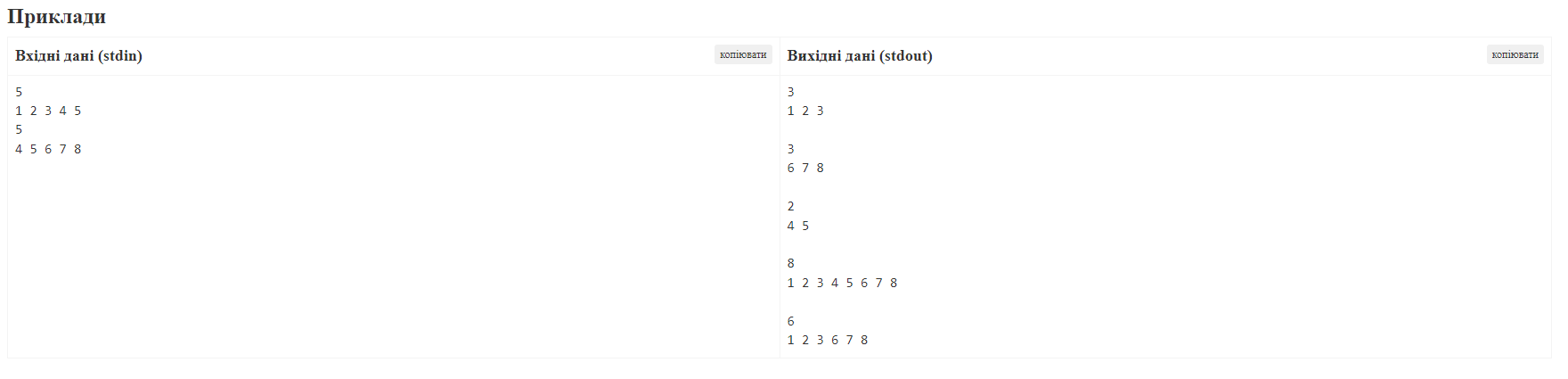
*Image 4. Algotester Lab 2v3 (Примітки, для візуалізаційного сприйняття)*

* Теж, не менш цікава задачка, де треба ознайомитись з бібліотекою (fstream) та її значення. Досить креативна задача, яка тримається в голові, ще й у сні.

Завдання **№4 Algotester Lab 4v3**

* Варіант завдання: 1

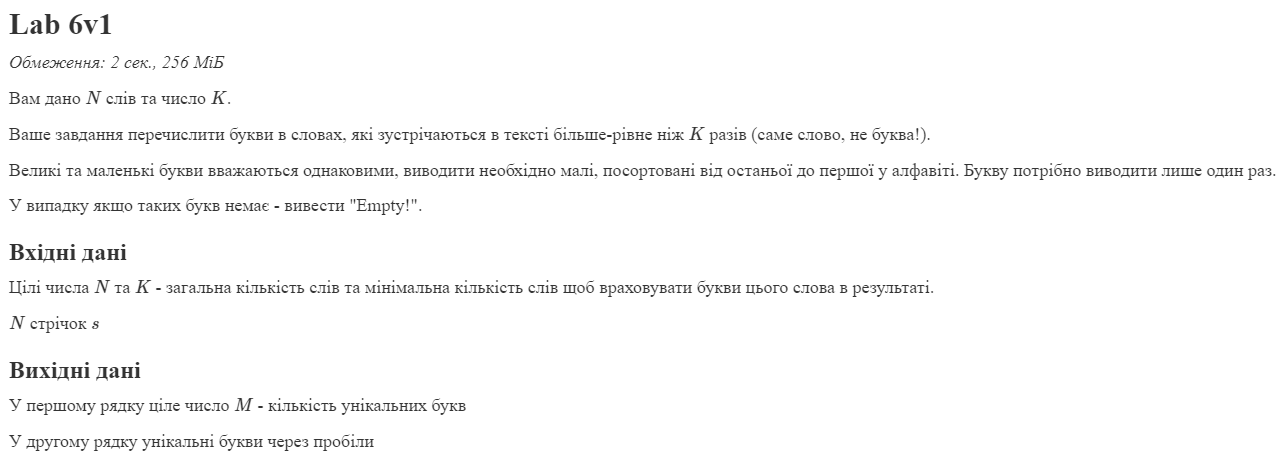


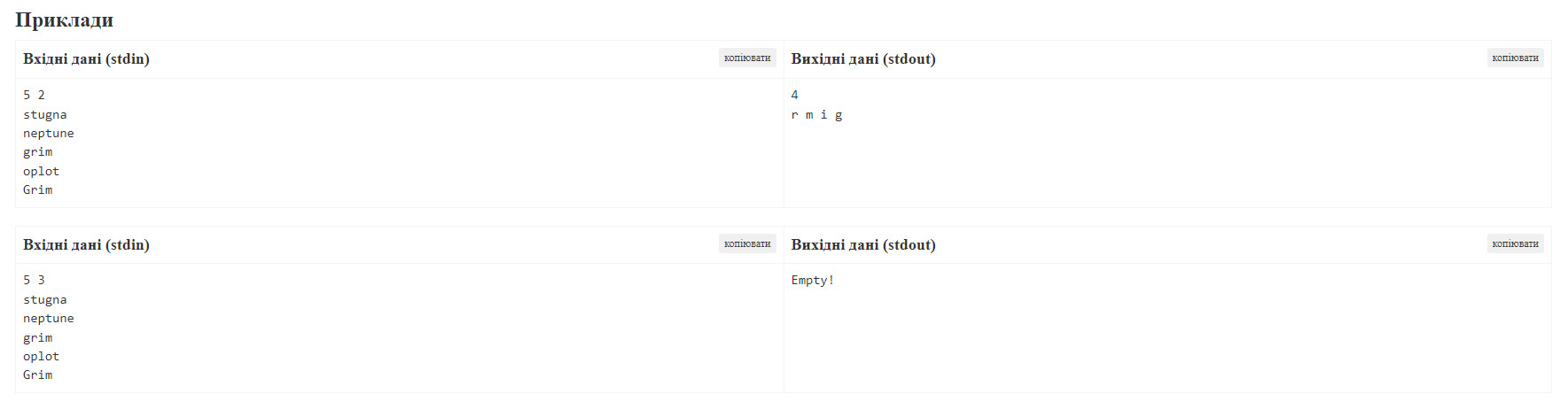


*Image 5. Algotester Lab 4v1 (Умова завдання та приклади вводу та виводу)*

Завдання **№5 Algotester Lab 6v1**

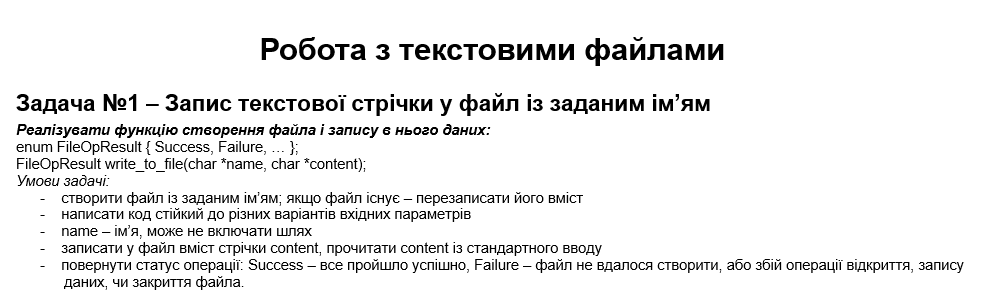
* Варіант завдання: 1

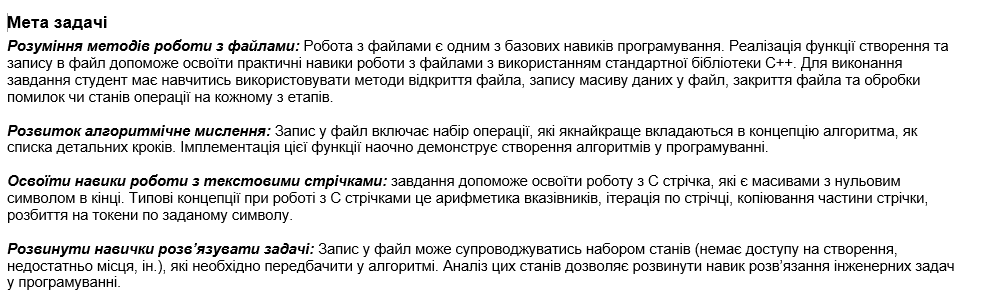


*Image 6. Algotester Lab 6v1 (Умова завдання)*

Завдання **№6 Class Practice Work (Task 1)**

* Варіант відсутній.

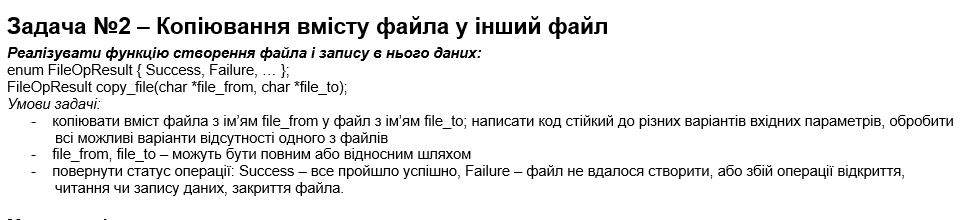


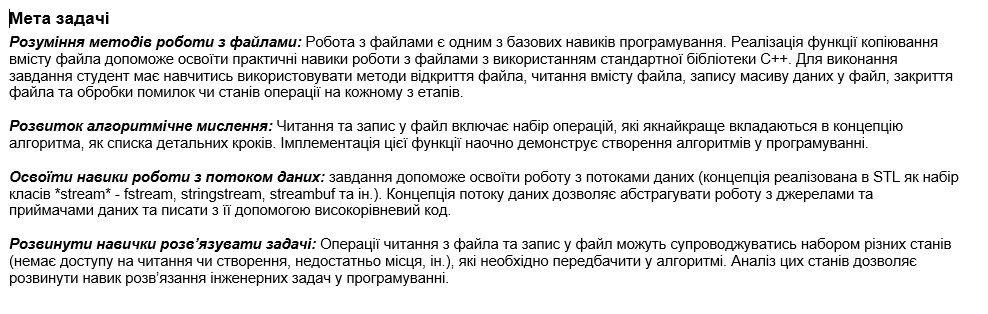
**

*Image 7. Class Practice Work (Task 1) – умова та мета завдання.*

Завдання **№7 Class Practice Work (Task 1)**

* Варіант відсутній.

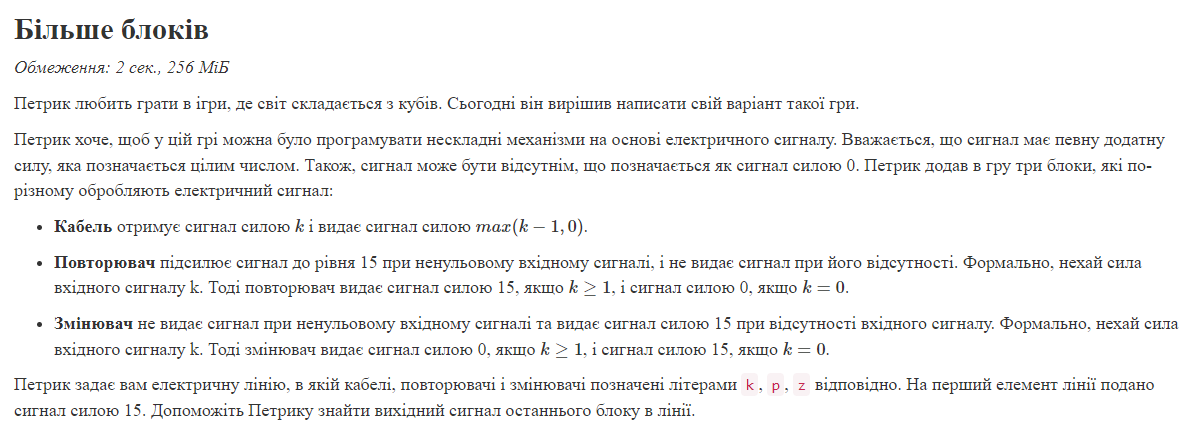


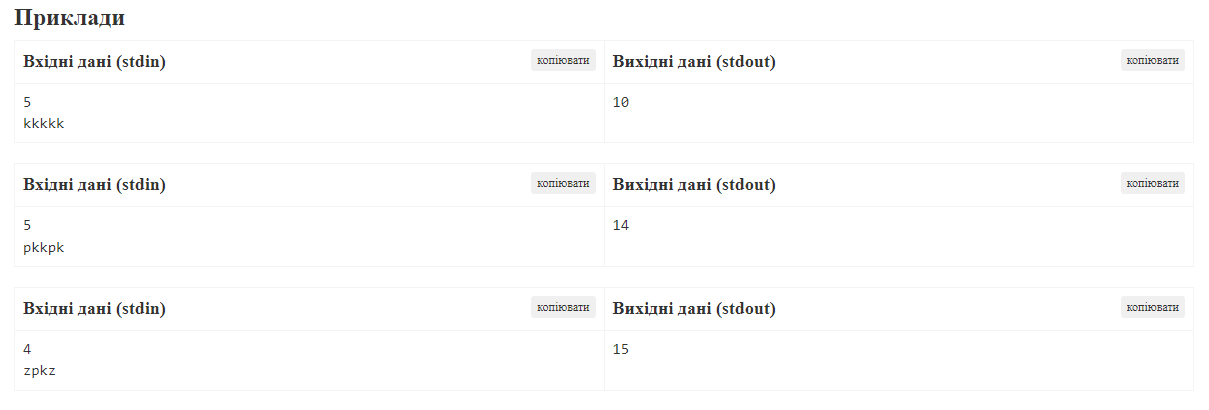
**

*Image 8. Class Practice Work (Task 2) – умова та мета завдання.*

Завдання **№8 Self Practice Work**

* [Задача Algotester: A (100). Більше блоків | The Algo Battles 2023 - Етап 7](https://algotester.com/uk/ContestProblem/DisplayWithEditor/134777) – Цікавий масив.





*Image 9. Self Practice Work – умова та мета завдання.*

* Задачка з контесту, яка пов`язана з фізикою. (бяка ), але досить не важка у написанні. Надзвичайно вигідна пропозиція, адже мало того, що отримаєш один додатковий бал (ну хоча б щось), та й ще self practice зробив! 😊

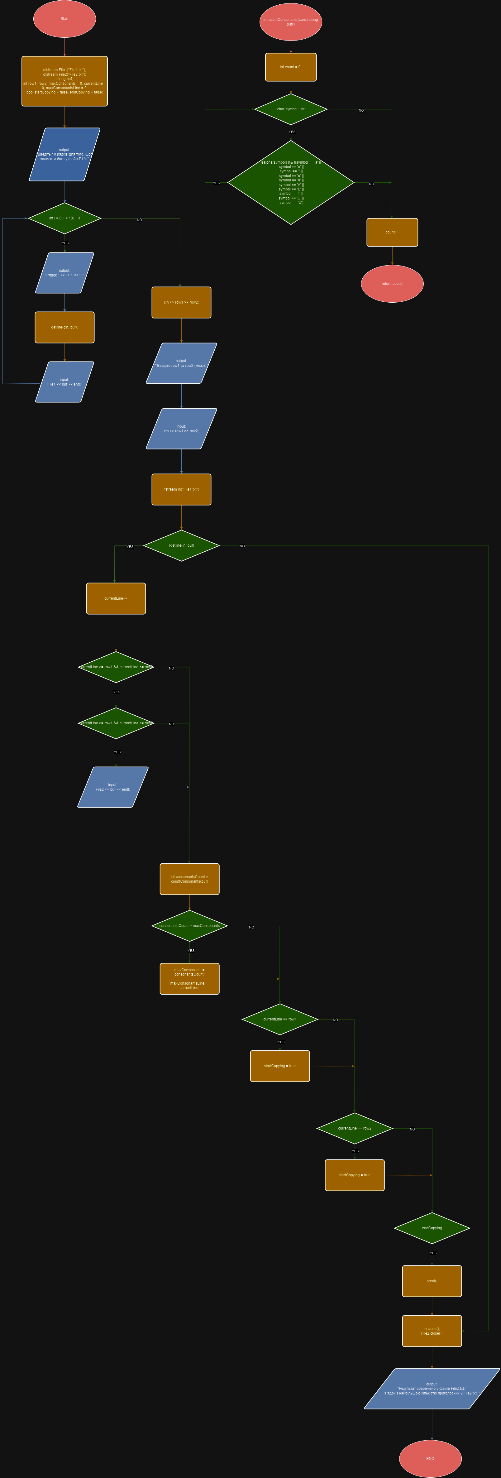
## **2. Дизайн та планована оцінка часу виконання завдань:**

Програма **№1 VNS Lab 6 - Task 1**

* Блок-схема: (відсутня)
* Планований час на реалізацію: *3* год => Затрачений час: *5* год.
* Важливі деталі для врахування в імплементації: Перетворити рядок таким чином, щоб цифри кожного слова в ньому були відсортовані за спаданням.
* Програма **№2 VNS Lab 8 - Task 1**
* Блок-схема (відсутня)
* Планований час на реалізацію: *5* год => Затрачений час: *28* год.

Важливі деталі для врахування в імплементації: 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 eлементів оформити у вигляді функцій. Передбачити повідомлення про помилки при відкритті файлу й виконанні операцій вводу/виводу.

* Програма **№3 VNS Lab 9 - Task 1**
* Блок-схема:



*Image 10. Flowchart VNS Lab 9 - Task 1*

* Планований час на реалізацію: *12* год => Затрачений час: *8* год.

Важливі деталі для врахування в імплементації: Створити текстовий файл F1 не менше, ніж з 10 рядків і записати в нього iнформацію про рядки, що починаються на букву «А» і

розташовані між рядками з номерами N1 й N2, та визначити номер того рядка, у якому найбільше приголосних букв, файлу F2.

* Програма **№4 Algotester Lab 4v1**
* Блок-схема (відсутня)
* Планований час на реалізацію: *4* год => Затрачений час: *10* год.

Програма **№5 Algotester Lab 6v1**

* Блок-схема (відсутня)
* Планований час на реалізацію: *5* год => Затрачений час: *15* год.
* Важливі деталі для врахування в імплементації:

Програма **№6 Class Practice Work (Task 1)**

* Блок-схема (відсутня)
* Планований час на реалізацію: *5* год => Затрачений час: *5* год.
* Важливі деталі для врахування в імплементації: 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, написати 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.

Програма **№7 Class Practice Work (Task 2)**

* Блок-схема (відсутня)
* Планований час на реалізацію: *12* *годин* => Затрачений час: *7 годин*. Важливі деталі для врахування в імплементації: 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.

Програма **№8 Self Practice Work**

* Блок-схемa (відсутня)

- Планований час на реалізацію: *30* хв => Затрачений час: *1* год.

## **3. Конфігурація середовища до виконання завдань:**

Використано налаштування середовища з попередніх лабораторних робіт.

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

Посилання на файл програми у пул-запиті GitHub: [pull-request-link-VNS-Lab-6-Task-1.](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/892/files#diff-0c9acf83e3dfb678e8120421358ae9f1d8b1aac8db47d0710d28f01840de6649)

Завдання **№1 VNS Lab 6 - Task 1**

#include <iostream>

#include <cstring>

#include <algorithm>

using namespace std;

// Функція для сортування цифр у слові за спаданням

void descendNum(char\* word) {

    int length = strlen(word);

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

        if (isdigit(word[i])) {

            int j = i;

            while (isdigit(word[j])) {

                ++j;

            }

            // Функція, яка відсортовує послідовність символів у слові в порядку спадання.

            sort(word + i, word + j, greater<char>()); // Тобто, ми вказуємо на діапазон символів у слові, які відповідають спадній послідовності цифр.

            i = j; // Пропускаємо вже відсортовані цифри та перейти до наступної частини рядка для подальшої обробки.

        }

    }

}

int main() {

    const int strLength = 255;

    char strInput[strLength];

    cout << "Enter the line of nubmers: " << endl;

    cin.getline(strInput, strLength);

    char\* token = strtok(strInput, " ");

    while (token != nullptr) { // Це цикл, який виконується, при отриманні нового токена. (Та превіряє чи присутні слова у рядку)

        descendNum(token);

        cout << token << " ";

        token = strtok(nullptr, " ");

    }

    return 0;

}

*Блок №1. Код до програми №1 VNS Lab 6 - Task 1*

Завдання **№2 VNS Lab 8 - Task 1**

Посилання на файл програми у пул-запиті GitHub: [pull-request-link-VNS-Lab-8-Task-1.](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/892/files#diff-ede41e13ac4c3d39e6e861fd3ff3ba8c928308dc82035fbf0a045e382764440e)

#include <fstream>

#include <vector>

#include <string>

#include <algorithm>

#include <iostream>

using namespace std;

struct Football\_Clubs {

    string city;

    string name;

    int points;

    int players;

};

// Функція для перетворення структури Футбол\_Клабс у послідовність бітів.

void serializeTeam(ofstream& stream, const Football\_Clubs& team) {

    size\_t nameSize = team.name.size();

    size\_t citySize = team.city.size();

    stream.write((char\*)&nameSize, sizeof(size\_t));

    stream.write(team.name.c\_str(), nameSize);

    stream.write((char\*)&citySize, sizeof(size\_t));

    stream.write(team.city.c\_str(), citySize);

    stream.write((char\*)&team.players, sizeof(team.players));

    stream.write((char\*)&team.points, sizeof(team.points));

}

// Функція для відновлення початкового стану структури Футбол\_Клабс із бітової послідовності.

bool deserializeTeam(ifstream& stream, Football\_Clubs& team) {

    size\_t nameSize, citySize;

    stream.read((char\*)&nameSize, sizeof(size\_t));

    if (stream.fail()) {

        return false;

    }

    char nameBuffer[nameSize + 1];

    stream.read(nameBuffer, nameSize);

    nameBuffer[nameSize] = '\0';

    team.name = nameBuffer;

    stream.read((char\*)&citySize, sizeof(size\_t));

    if (stream.fail()) {

        return false;

    }

    char cityBuffer[citySize + 1];

    stream.read(cityBuffer, citySize);

    cityBuffer[citySize] = '\0';

    team.city = cityBuffer;

    stream.read((char\*)&team.players, sizeof(team.players));

    stream.read((char\*)&team.points, sizeof(team.points));

    return !stream.fail();

}

// Функція для запису вектора футбольних команд у файл

void writeTeamsToFile(const string& filename, const vector<Football\_Clubs>& teams) {

    ofstream file(filename, ios::binary);

    if (!file) {

        cout << "Помилка відкриття файла для запису інформації!" << filename << endl;

        return;

    }

    for (const auto& team : teams) {

        serializeTeam(file, team);

    }

    file.close();

}

// Функція для зчитування вектора футбольних команд з файлу

vector<Football\_Clubs> readTeamsFromFile(const string& filename) {

    vector<Football\_Clubs> teams;

    ifstream file(filename, ios::binary);

    if (!file) {

        cout << "Помилка відкриття файла для зчитування інформації!" << filename << endl;

        return teams;

    }

    Football\_Clubs team;

    while (deserializeTeam(file, team)) {

        teams.push\_back(team);

    }

    file.close();

    return teams;

}

// Функція для виведення інформації про футбольну команду

void printTeam(const Football\_Clubs& team) {

    cout << "Name: " << team.name << ", City: " << team.city << ", Players: " << team.players << ", Points: " << team.points << endl;

}

// Функція для виведення інформації про всі футбольні команди у векторі

void printTeams(const vector<Football\_Clubs>& teams) {

    for (const auto& team : teams) {

        printTeam(team);

    }

}

// Функція для фільтрації футбольних команд за мінімальною кількістю очок

vector<Football\_Clubs> filterTeamsByPoints(const vector<Football\_Clubs>& teams, int minPoints) {

    vector<Football\_Clubs> newTeams;

    copy\_if(teams.begin(), teams.end(), back\_inserter(newTeams),

            [minPoints](const Football\_Clubs& team) { return team.points >= minPoints; });

    return newTeams;

}

// Функція для об'єднання двох векторів футбольних команд

vector<Football\_Clubs> combineTeams(const vector<Football\_Clubs>& teams1, const vector<Football\_Clubs>& teams2) {

    vector<Football\_Clubs> combinedTeams(teams2);

    combinedTeams.insert(combinedTeams.end(), teams1.begin(), teams1.end());

    return combinedTeams;

}

vector<Football\_Clubs> newTeams; // Оголошення newTeams поза main для доступності

int main() {

    // Створення футбольних команд

    vector<Football\_Clubs> teams = {

        {"Paris SG", "Paris", 15, 11},

        {"FC Porto", "Porto", 2, 11},

        {"Juventus", "Turin", 9, 11},

        {"Napoli FC", "Neapol", 1 , 11},

        {"FC Dortmund", "Dortmund", 14, 11}

    };

    // Запис футбольних команд у файл

    writeTeamsToFile("fcteams.txt", teams);

    // Зчитування футбольних команд з файлу

    teams = readTeamsFromFile("fcteams.txt");

    // Виведення початкових футбольних команд

    cout << "Команди турніру:" << endl;

    printTeams(teams);

    // Фільтрація футбольних команд з менше ніж 9 очками

    newTeams = filterTeamsByPoints(teams, 9);

    // Виведення футбольних команд після видалення

    cout << "\nФутбольні клуби, після видалення тих, які мають менше ніж 9 очок:" << endl;

    printTeams(newTeams);

    // Додавання нових футбольних команд

    vector<Football\_Clubs> newTeams2 = {

        {"PSV", "Eindhoven", 5, 11},

        {"Latium", "Rome", 5, 11}

    };

    newTeams = combineTeams(newTeams, newTeams2);

    // Виведення футбольних команд після додавання

    cout << "\nФутбольні клуби, після заміни де-яких іншими:" << endl;

    printTeams(newTeams);

    // Запис остаточних футбольних команд у файл

    writeTeamsToFile("fcteams.txt", newTeams);

    return 0;

}

*Блок №2. Код до програми №2 VNS Lab 5 - Task 1*

Завдання **№3 VNS Lab 9 - Task 1**

Посилання на файл програми у пул-запиті GitHub: [pull-request-link-VNS-Lab-9-Task-1.](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/892/files#diff-40b8d5f7ee4c1d94c130ae8ba167c781f4b2280b46d7274770ca1c470921e3e5)

#include <iostream>

#include <fstream>

#include <cctype>

using namespace std;

int countConsonants(const string &str) {

    int count = 0;

    for (char symbol : str) {

        if (isalpha(symbol) && !(symbol == 'a' || symbol == 'e' || symbol == 'i' || symbol == 'o' || symbol == 'u' || symbol == 'A' || symbol == 'E' || symbol == 'I' || symbol == 'O' || symbol == 'U')) {

            count++;

        }

    }

    return count;

}

int main() {

    ofstream File1("File1.txt");

    ofstream File2("File2.txt");

    string buf;

    int row1, row2, maxConsonants = 0, currentLine = 0, maxConsonantsLine = 0;

    bool startCopying = false, endCopying = false;

    // Вводимо 10 рядків в файл File1.

    cout << "Введіть 10 рядків для того, щоб записати його у файл F1:\n";

    for (int i = 0; i < 10; ++i) {

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

        getline(cin, buf);

        File1 << buf << endl;

    }

    File1.close();

    // Введення row1 та row2

    cout << "Введіть row1 та row2 (межі): ";

    cin >> row1 >> row2;

    // Відкриття файлу F1 для читання

    ifstream in("File1.txt");

    // Копіювання рядків з файлу File1 в File2, які знаходяться між рядками з номерами row1 і row2 і починаються на "A" або "a"

    while (getline(in, buf)) {

        currentLine++;

        if (currentLine >= row1 && currentLine <= row2) {

            if (buf[0] == 'A' || buf[0] == 'a') {

                File2 << buf << endl;

            }

        }

        // Знаходження рядка з максимальною кількістю приголосних букв

        int consonantsCount = countConsonants(buf);

        if (consonantsCount > maxConsonants) {

            maxConsonants = consonantsCount;

            maxConsonantsLine = currentLine;

        }

        // Початок копіювання після рядка row1

        if (currentLine == row1) {

            startCopying = true;

        }

        // Закінчення копіювання після рядка row2

        if (currentLine == row2) {

            endCopying = true;

        }

        // Вихід з циклу після закінчення копіювання

        if (endCopying) {

            break;

        }

    }

    // Закриття файлів

    in.close();

    File2.close();

    cout << "Результат збережено у файлі File2.txt." << endl;

    cout << "Рядок з найбільшою кількістю приголосних у File2.txt: " << maxConsonantsLine << endl;

    return 0;

}

*Блок №3. Код до програми №3 VNS Lab 9 - Task 1*

Завдання **№4 Algotester Lab 4v1**

Посилання на файл програми у пул-запиті GitHub: [pull-request-link-Algotester-Lab-4-v-1](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/892/files#diff-6a85a6fa5f96e845e7c9fa7d45022dbec042e7d3141313c2ad1dea423c0b26f6).

#include <iostream>

#include <vector>

#include <set>

#include <algorithm>

#include <iterator>

using namespace std;

int main() {

    // Оголошуємо змінні для розмірів двох масивів

    int N, M;

    // Вводимо розмір першого масиву

    cin >> N;

    // Оголошуємо та вводимо елементи першого масиву

    vector<int> firstArray(N);

    for (int i = 0; i < N; ++i) {

        cin >> firstArray[i];

    }

    // Вводимо розмір другого масиву

    cin >> M;

    // Оголошуємо та вводимо елементи другого масиву

    vector<int> secondArray(M);

    for (int i = 0; i < M; ++i) {

        cin >> secondArray[i];

    }

    // Сортуємо елементи обох масивів

    sort(firstArray.begin(), firstArray.end());

    sort(secondArray.begin(), secondArray.end());

    // Оголошуємо вектори для результатів операцій з масивами

    vector<int> difference\_N\_M, difference\_M\_N, intersection, union\_set, symmetric\_difference;

    // Знаходимо різницю між масивами firstArray і secondArray та зберігаємо результат в difference\_N\_M

    set\_difference(firstArray.begin(), firstArray.end(), secondArray.begin(), secondArray.end(), back\_inserter(difference\_N\_M));

    // Знаходимо різницю між масивами secondArray і firstArray та зберігаємо результат в difference\_M\_N

    set\_difference(secondArray.begin(), secondArray.end(), firstArray.begin(), firstArray.end(), back\_inserter(difference\_M\_N));

    // Знаходимо перетин масивів firstArray і secondArray та зберігаємо результат в intersection

    set\_intersection(firstArray.begin(), firstArray.end(), secondArray.begin(), secondArray.end(), back\_inserter(intersection));

    // Знаходимо об'єднання масивів firstArray і secondArray та зберігаємо результат в union\_set

    set\_union(firstArray.begin(), firstArray.end(), secondArray.begin(), secondArray.end(), back\_inserter(union\_set));

    // Знаходимо симетричну різницю масивів firstArray і secondArray та зберігаємо результат в symmetric\_difference

    set\_symmetric\_difference(firstArray.begin(), firstArray.end(), secondArray.begin(), secondArray.end(), back\_inserter(symmetric\_difference));

    // Виводимо розмір масиву difference\_N\_M та його елементи

    cout << difference\_N\_M.size() << "\n";

    for (size\_t i = 0; i < difference\_N\_M.size(); ++i) {

        int num = difference\_N\_M[i];

        cout << num << " ";

    }

    cout << "\n";

    // Виводимо розмір масиву difference\_M\_N та його елементи

    cout << difference\_M\_N.size() << "\n";

    for (size\_t i = 0; i < difference\_M\_N.size(); ++i) {

        int num = difference\_M\_N[i];

        cout << num << " ";

    }

    cout << "\n";

    // Виводимо розмір масиву intersection та його елементи

    cout << intersection.size() << "\n";

    for (size\_t i = 0; i < intersection.size(); ++i) {

        int num = intersection[i];

        cout << num << " ";

    }

    cout << "\n";

    // Виводимо розмір масиву union\_set та його елементи

    cout << union\_set.size() << "\n";

    for (size\_t i = 0; i < union\_set.size(); ++i) {

        int num = union\_set[i];

        cout << num << " ";

    }

    cout << "\n";

    // Виводимо розмір масиву symmetric\_difference та його елементи

    cout << symmetric\_difference.size() << "\n";

    for (size\_t i = 0; i < symmetric\_difference.size(); ++i) {

        int num = symmetric\_difference[i];

        cout << num << " ";

    }

    cout << "\n";

    return 0;

}

*Блок №4. Код до програми №4 Algotester 4v1*

Завдання **№5 Algotester Lab 6v1**

Посилання на файл програми у пул-запиті GitHub: [pull-request-link-Algotester-Lab-6-v-1.](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/892/files#diff-b6d7d15798c49f97ed68f0ac8c7a9bdea0b0261bf66ce104d0209b85bffcd275)

#include <iostream>

#include <map>

#include <set>

#include <vector>

#include <algorithm>

#include <string>

#include <cctype>

using namespace std;

int main() {

    // Введення кількості слів та мінімальної кількості входжень

    int totalWords, threshold;

    cin >> totalWords >> threshold;

    // Створення словника для визначення частоти кожного слова

    map<string, int> wordFrequency;

    // Зчитування слів та підрахунок їх кількості

    vector<string> words(totalWords);

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

        cin >> words[i];

        // Перетворення слова у нижній регістр

        transform(words[i].begin(), words[i].end(), words[i].begin(), ::tolower);

        // Збільшення лічильника кількості входжень слова

        wordFrequency[words[i]]++;

    }

    // Вибір букв, які належать словам, які зустрічаються не менше K разів

    set<char> uniqueLetters;

    for (const auto &word : words) {

        if (wordFrequency[word] >= threshold) {

            for (const char &letter : word) {

                uniqueLetters.insert(letter);

            }

        }

    }

    // Вивід результату

    if (uniqueLetters.empty()) {

        cout << "Empty!" << endl;

    } else {

        cout << uniqueLetters.size() << endl;

        // Вивід унікальних букв у зворотньому алфавітному порядку

        for (auto it = uniqueLetters.rbegin(); it != uniqueLetters.rend(); ++it) {

            cout << \*it;

            // Додавання пробіла між буквами (крім останньої)

            if (next(it) != uniqueLetters.rend()) {

                cout << " ";

            }

        }

        cout << endl;

    }

    return 0;

}

*Блок №5. Код до програми №5 Algotester 6v3*

Завдання **№6 Class Practice Work Task 1.**

Посилання на файл програми у пул-запиті GitHub: [pull-request-link-Class-Practice-Work-task-1.](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/892/files#diff-7701a12bbd79c5d58892e1bf737a0be5eee4c8dff2231a0c98593e0496c95c66)

#include <iostream>

#include <fstream>

using namespace std;

enum FileOpResult { Success, Failure };

FileOpResult write\_to\_file(const char\* name, const char\* content) {

    // Відкриваємо файл для запису

    ofstream file(name);

    // Перевіряємо, чи вдалося записати файл

    if (!file.is\_open()) {

        return Failure;

    }

    // Записуємо введене у файл

    file << content;

    // Перевіряємо, чи файл нормально закрився

    if (file.fail()) {

        return Failure;

    }

    // Повертаємо саксес, якщо файл було створено та все записано

    return Success;

}

int main() {

    // Вводимо ім'я файлу та самої стрічки для запису

    char filename[15];

    char content[150];

    cout << "Вкажіть ім'я файлу: ";

    cin >> filename;

    cout << "Введіть, те, чим хочете заповнити файл: ";

    cin.ignore();

    cin.getline(content, sizeof(content));

    // Викликаємо фанкшен, для запису у файл

    FileOpResult result = write\_to\_file(filename, content);

    // Виводимо відповідний результат

    if (result == Success) {

        cout << "Файл УСПІШНО створений, та у ньому записані ваші введеня! <3" << endl;

    } else {

        cout << "ПОМИЛКА створення файлу, або ж ПОМИЛКА запису. :( " << endl;

    }

    return 0;

*Блок №6. Код до програми №6 Class Practice Work Task 1*

Завдання **№7 Class Practice Work Task 2.**

Посилання на файл програми у пул-запиті GitHub: [pull-request-link-Class-Practice-Work-task-2.](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/892/files#diff-92420135240419a699fb63f6f1c01f33550f5717bab9be5b09a8b6595ed4d15e)

#include <iostream>

#include <fstream>

using namespace std;

enum FileOpResult { Success, Failure };

FileOpResult write\_to\_file(const char\* name, const char\* content) {

    // Відкриваємо файл для запису

    ofstream file(name);

    // Перевіряємо, чи вдалося записати файл

    if (!file.is\_open()) {

        return Failure;

    }

    // Записуємо введене у файл

    file << content;

    // Перевіряємо, чи файл нормально закрився

    if (file.fail()) {

        return Failure;

    }

    // Повертаємо саксес, якщо файл було створено та все записано

    return Success;

}

int main() {

    // Вводимо ім'я файлу та самої стрічки для запису

    char filename[15];

    char content[150];

    cout << "Вкажіть ім'я файлу: ";

    cin >> filename;

    cout << "Введіть, те, чим хочете заповнити файл: ";

    cin.ignore();

    cin.getline(content, sizeof(content));

    // Викликаємо фанкшен, для запису у файл

    FileOpResult result = write\_to\_file(filename, content);

    // Виводимо відповідний результат

    if (result == Success) {

        cout << "Файл УСПІШНО створений, та у ньому записані ваші введеня! <3" << endl;

    } else {

        cout << "ПОМИЛКА створення файлу, або ж ПОМИЛКА запису. :( " << endl;

    }

    return 0;

}

*Блок №7. Код до програми №7 Class Practice Work Task 1*

Завдання **№8 Self Practice Work**

Посилання на файл програми у пул-запиті GitHub: [pull-request-link-Self-Practice-Work.](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/892/files#diff-b7d4232b4bbae7a70b2bc83535070e9f3be44c37c23a828176da75ed194ab53b)

#include <iostream>

#include <vector>

using namespace std;

int main() {

    int n; // Вводимо розміру масиву

    cin >> n; // Зчитуємо розміру масиву з консолі

    vector<int> numbers(n); // Створення вектору цілих чисел розміром n

    // Цикл, який заповнює вектора nubmers елементами, які користувач вводить з консолі

    for (int i = 0; i < n; i++) {

        cin >> numbers[i];

    }

    // Початок циклу, який перебирає можливі значення розмежувального індексу k

    for (int k = 1; k < n; k++) {

        int sum\_left = 0;

        int sum\_right = 0;

        // Цикл, який обчислює суми елементів зліва від k (від лівої частини масиву)

        for (int i = 0; i < k; i++) {

            sum\_left += numbers[i];

        }

        // Цикл, який обчислює суми елементів справа від k (від правої частини масиву)

        for (int i = k; i < n; i++) {

            sum\_right += numbers[i];

        }

        // Виводимо "TAK", якщо сумма лівого розбиття масиву = сумі правого робиття масиву.

        if (sum\_left == sum\_right) {

            cout << "TAK" << endl;

            return 0;

        }

    }

    cout << "NI" << endl; // Виводимо "NI", якщо сумма лівого розбиття масиву != сумі правого робиття масиву.

    return 0;

}

*Блок №8. Код до програми №8 Self Practice Work (Algotester з конесту)*

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

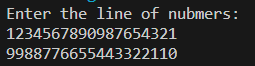
Завдання **№1 VNS Lab 6 - Task 1**



*Image 11. Результати виконання завдання №1 VNS Lab 6 - Task 1 (1 запуск)*



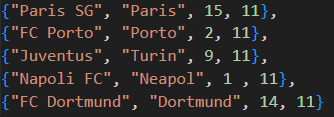
*Image 12. Результати виконання завдання №1 VNS Lab 6 - Task 1 (2 запуск)*



*Image 13. Результати виконання завдання №1 VNS Lab 6 - Task 1 (3 запуск)*

Час затрачений на виконання завдання: *5 год.*

Завдання **№2 VNS Lab 8 - Task 1**. *I*

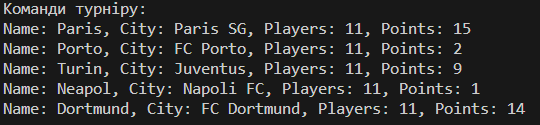
**

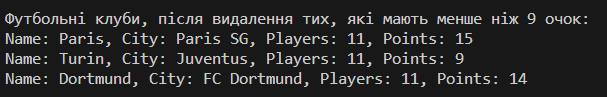
*Image 14. Умова, де вказані початкові клуби. №1 VNS Lab 8 - Task 1*

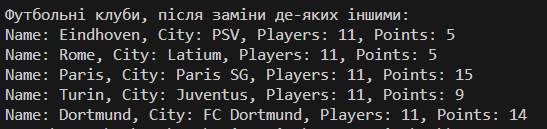
**

*Image 15. Умова, де вказані футбольні клуби, на які будуть замінятись клуби, в яких менше к-сть очок ніж 9.*

*№1 VNS Lab 8 - Task 1*

**

**

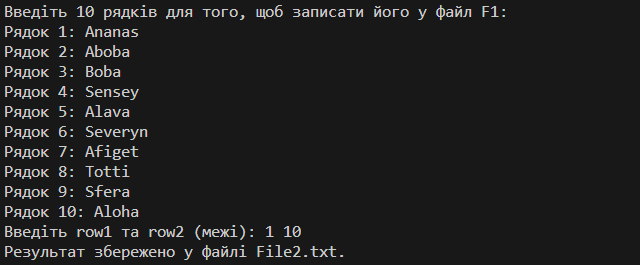
**

**

*Image 16. Результати виконання завдання №2 VNS Lab 8 - Task 1*

Час затрачений на виконання завдання: *28 год.*

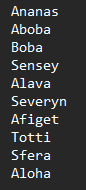
Завдання **№3 VNS Lab 9 - Task 1**

**

*Image 17. Результати виконання завдання №3 VNS Lab 9 - Task 1*

**

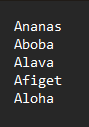
*Image 18. Результат створення файлу «File1» VNS Lab 9 - Task 1*

**

*Image 19. Результат заповнення файлу, даними веденими на початку «File2» VNS Lab 9 - Task 1*

**

*Image 20. Результат створення файлу «File2» VNS Lab 9 - Task 1*

**

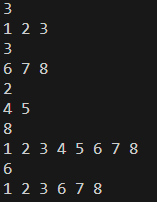
*Image 21. Результат заповнення файлу «File2», за параметрами. VNS Lab 9 - Task 1*

Час затрачений на виконання завдання: *8 год.*

Завдання **№4** **Algotester Lab 4v1**

**

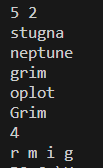
*Image 22. Результати вводу чисел (з умови алготестеру) Algotester Lab 4v1*

**

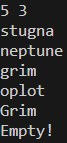
*Image 23. Результати виконання завдання Algotester Lab 4v1*

Час затрачений на виконання завдання: *10 год.*

Завдання **№5 Algotester Lab 6v1**



*Image 24. Результати виконання завдання №4 Algotester Lab 6v1 (1 запит)*



*Image 25. Результати виконання завдання №4 Algotester Lab 6v1 (2 запит)*

Час затрачений на виконання завдання: *15 год.*

Завдання **№6 Class Practice Work (Task 1)**



*Image 26. Результати виконання завдання №6 Class Practice Work Task 1 (1 запит)*



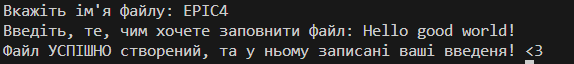
*Image 27. Результати створення файлу «EPIC5»*

*Завдання №6 Class Practice Work Task 1 (1 запит)*



*Image 28. Результати створення в файлі «EPIC5», речення, яке ввів користувач*

*Завдання №6 Class Practice Work Task 1 (1 запит)*



*Image 29. Результати виконання завдання №6 Class Practice Work Task 1 (2 запит)*



*Image 30. Результати створення файлу «EPIC4»*

*Завдання №5 Class Practice Work Task 1 (1 запит)*



*Image 31. Результати створення в файлі «EPIC5», речення, яке ввів користувач.*

*Завдання №6 Class Practice Work Task 1 (2 запит)*

Час затрачений на виконання завдання: *5 год.*

Завдання **№7 Class Practice Work (Task 2)**



*Image 32. Результати виконання завдання №7*

*Class Practice Work Task 2 (1 запит)*



*Image 33. Початковий файл «EPIC5», з якого копіюємо дані.*

*№7 Class Practice Work Task 2*



*Image 34. Результати виконання завдання*

*№7 Class Practice Work Task 1 (1 запит)*



*Image 35. Файл «EPIC4» в який копіюємо дані, з файлу «EPIC5».*

*№7 Class Practice Work Task 2 (1 запит)*



*Image 36. Результати виконання завдання*

*№7 Class Practice Work Task 2 (1 запит)*

Час затрачений на виконання завдання: *7 год.*

Завдання **№8 Self Practice Work**



*Image 37. Результати виконання завдання №6 Self Practice Work (1 запит)*



*Image 38. Результати виконання завдання №6 Self Practice Work* *(2 запит)*

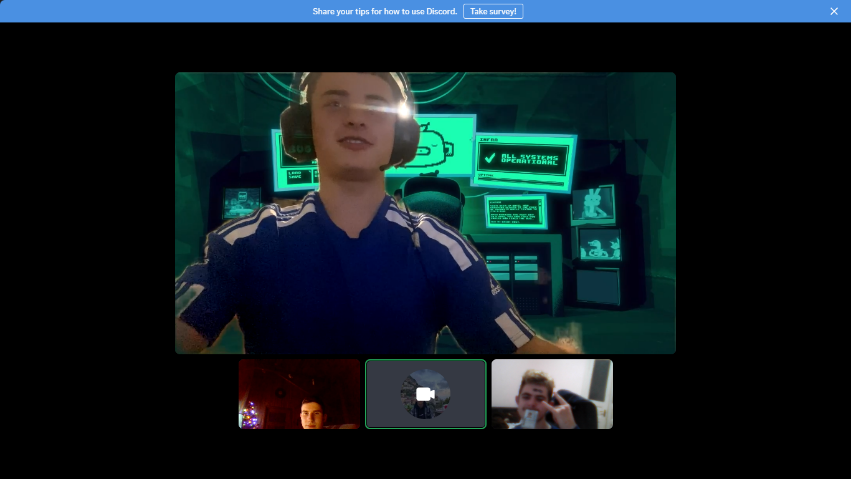


*Image 39. Результати виконання завдання №6 Self Practice Work (3 запит)*

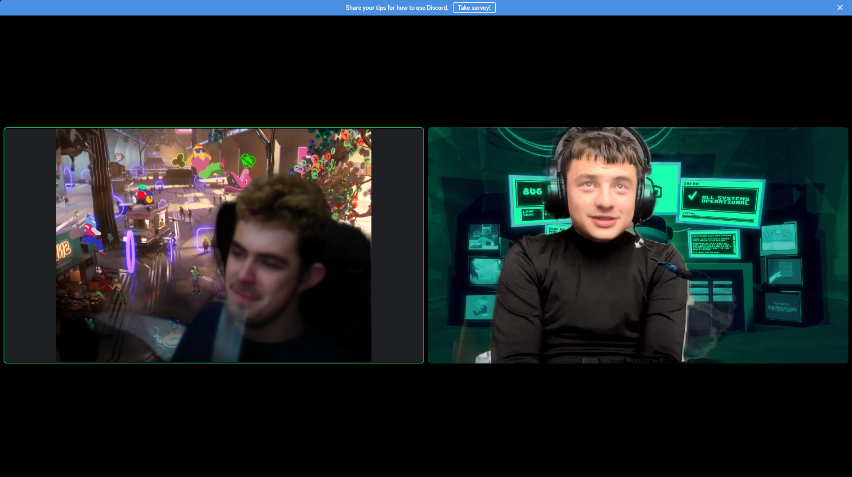
Час затрачений на виконання завдання: *1 год.*

## **6. Кооперація з командою:**

Час затрачений на обговорення з командою: *55 год.*



*Image 40. Кооперація з побратимами по нещастю (12.14.2023).*



*Image 41. Кооперація з Ромою в 5 годині ранку. (12.13.2023). P.S Досі триває 😊*



*Image 42. Кооперація з попаяним Ромою, після 30 годин підряд роботи над 5 епіком (12.4.2023).*



*Image 43. Лук’ян після 30 спроб зробити VNS LAB 8. (Fatal Face Error)*



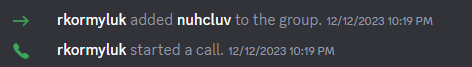
*Image 44. Лук’ян після 23092428-спроби, але вдалої зробити VNS LAB 8; (return succesed;)*



*Image 45. Кооперація з командою (12.07.2023)*



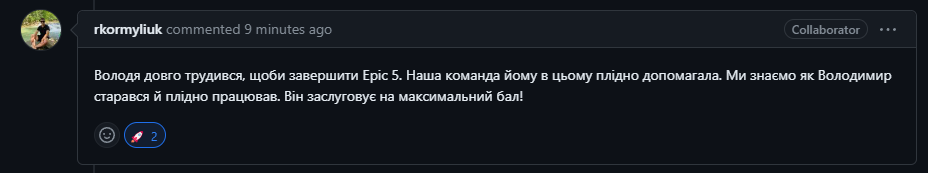
*Image 46. Кооперація з командою (12.08.2023)*

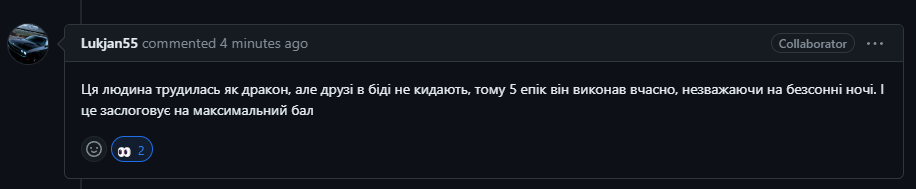
**

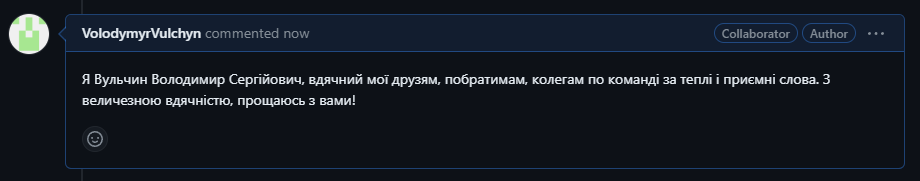
*Image 47. Кооперація з командою і бро Лук’яном (12.14.2023)*

*P.S На момент виконання звіту (12.14.2023 04:34:03) цей дзвінок ще триває.*

*І скільки триватиме цей дзвінок, нікому не відомо.*

**

**

**

*Image 48. Коментар від побратима по команді та бро з іншої команди. (12.14.2023)*

# **Висновки:**

Під час вивчення тем "Файли. Системи числення. Символи і Рядкові Змінні та Текстові Файли" я отримав глибоке розуміння основних концепцій програмування та обробки даних, а також мені почало снитись програмування!

У розділі "Файли" я вивчив, як ефективно працювати з введенням та виведенням даних, зберігати та відновлювати інформацію за допомогою файлів.

Системи числення дали мені знання про представлення чисел у різних системах, що є важливим для оптимізації алгоритмів та взаємодії з операціями.

Розділ "Символи і Рядкові Змінні" дозволив мені вивчити основи обробки текстових даних, працювати з символами та виконувати операції над рядками.

Навчання в розділі "Текстові файли" дало можливість розширити мої навички читання та запису структурованих даних у текстових файлах, що є важливим для роботи з реальними даними.