Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту



**Звіт**

про виконання

**Лабораторних та практичних робіт № 3**

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

***з розділу***: «Одновимірні масиви. Двовимірні Масиви. Алгоритми обробки»

***Виконав:***

студент групи ШІ-14

Вульчин Володимир Сергійович

# **Тема роботи:**

Ознайомлення з одновимірними та двовимірними масивами. Знайомство з базовими алгоритми обробки даних

# **Мета роботи:**

Ознайомитися з одновимірними та двовимірними у С++. Опрацювати та застосувати алгоритми обробки для масивів.

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* **Тема №1:** Одновимірні масиви.
* **Тема №2:** Двовимірні масиви.
* **Тема №3:** Алгоритми обробки.

1. Індивідуальний план опрацювання теорії:

* Тема **№1: Одновимірні масиви.**
  + Джерела Інформації:
    - ChatGPT / Команда, Одногрупнки
    - Відео.

Відео-теорія про:

[Одновимірні масиви.](https://youtu.be/ULdbOaMBPYc?t=1382) (База)

[Одновимірні масиви](https://youtu.be/Nvby9WQmDsM?t=915). (Закріплення)

* + - Статті.

Теорія про одновимірні масиви:

[Одновимірні масиви](https://acode.com.ua/urok-77-masyvy/) (сайт, як завжди з потужною теорією)

[Одновимірні масиви](https://cherto4ka.xyz/2020/01/16/massive_begin/) (дуже сподобалась візуалізація, картинки must have 😊)

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

1. Що таке масив?
2. Приклади використання масиву.
3. Типи данних і масиви.
4. Динамічні масиви!
   * Статус: Ознайомлений
   * Початок опрацювання теми: 22.11.2023
   * Звершення опрацювання теми: 30.11.2023

* Тема **№2: Двовимірні масиви.**
  + Джерела Інформації:
    - ChatGPT / Команда, Одногрупники
    - Відео.

Відео-теорія про: [Двовимірний масив.](https://www.youtube.com/watch?app=desktop&v=V2g3B9Zbh4Q) (База)

* + - Стаття.

Cтаття про:

[Двовимірні масиви.](https://acode.com.ua/urok-81-bagatovymirni-masyvy/) (acode)

[Двовимірні масиви.](https://www.bestprog.net/uk/2017/03/04/масиви-частина-2-двовимірні-масиви-ма/) (приклади, багато прикладів ☹)

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

1. Що таке двовимірний масив?
2. Багатовимірні масиви!
3. Ініціалізація двовимірних масивів.
4. Приклади застосування двовимірних масивів
   * Статус: Ознайомлений
   * Початок опрацювання теми: 23.11.2023
   * Звершення опрацювання теми: 31.11.2023
   * Тема №3: Алгоритми обробки
   * Джерела Інформації:
     + ChatGPT / Команда, Одногрупники
     + Відео.

[Лінійний пошук.](https://youtu.be/gncUL57AHwk?t=487)

[Бінарний пошук](https://youtu.be/cVwmHF4pROg?t=931).

* + - Статті.

Теорія про:

[Алгоритми обробки одновимірних масивів.](https://cherto4ka.xyz/2020/01/21/базові-операції-обробки-одновимірни/) (aCode)

[Види сортування масивів](http://cpp.dp.ua/sortuvannya-masyviv/) (Теж чудесний сайт, з наведенням прикладів, та ще й блок схем до них!)

Що опрацьовано:

* + - Самостійно ознайомився з таким переліком інформації:

1) Які є базові алгоритми обробки одновимірних масивів?

2) Види сортування масивів

3) Лінійний пошук.

4) Бінарний пошук.

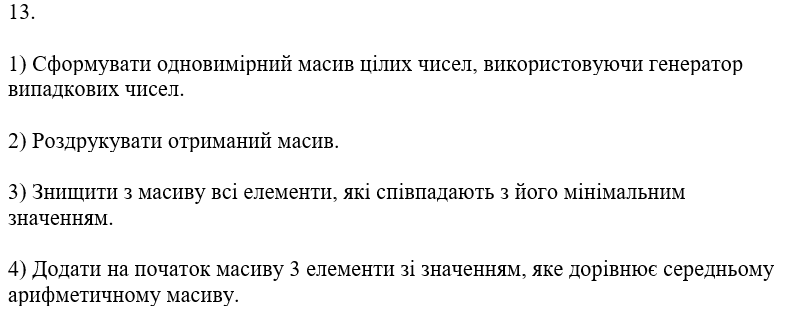
* + Статус: Ознайомлений
  + Початок опрацювання теми: 22.11.2023
  + Звершення опрацювання теми: 31.11.2023

# **Виконання роботи:**

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

Завдання **№1 VNS Lab 4 - Task 1**

* Варіант завдання: 13

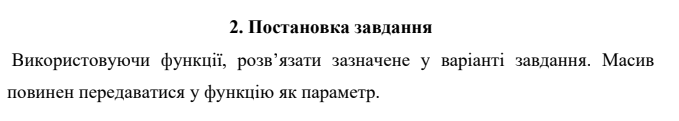


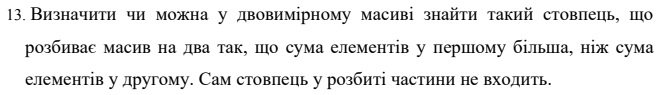
*Image 1. VNS LAB 4 – Task 1 (Умова завдання)*

* Погуглив про генератор випадкових чисел, та знайшов на платформі Reddit, пояснення генератора рандомних чисел в С++: [link-reddit-topic](https://www.reddit.com/r/learnprogramming/comments/1o4bd2/understanding_srand_in_c/)

Завдання **№2 VNS Lab 5- Task 1**

* Варіант завдання: 13

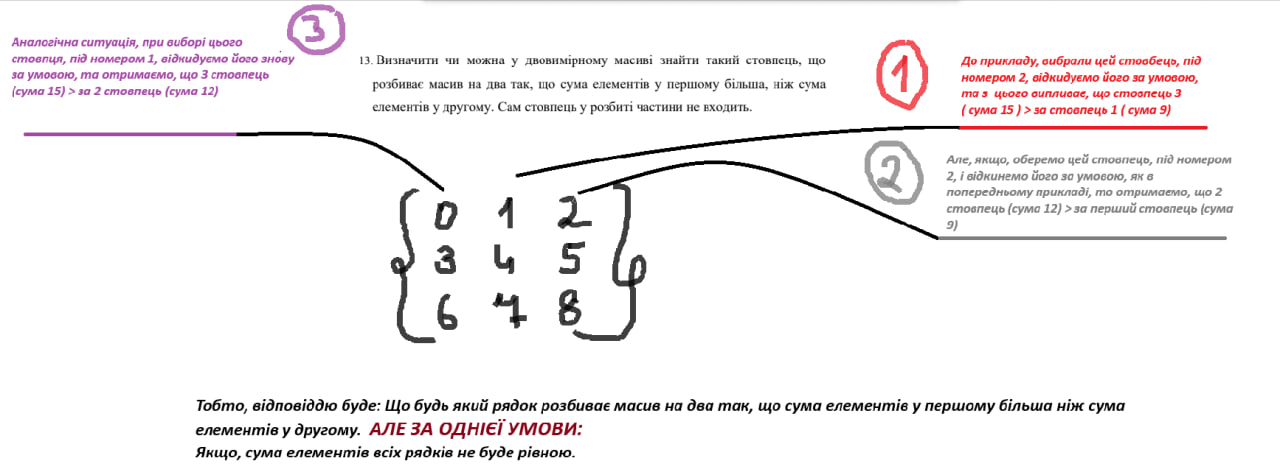




*Images 2. VNS LAB 5 – Task 1 (Умова завдання)*

* Надзвичайно цікаве завдання, над яким треба було добре поміркувати:

а) Ілюстрація ходу моїх думок на початку:



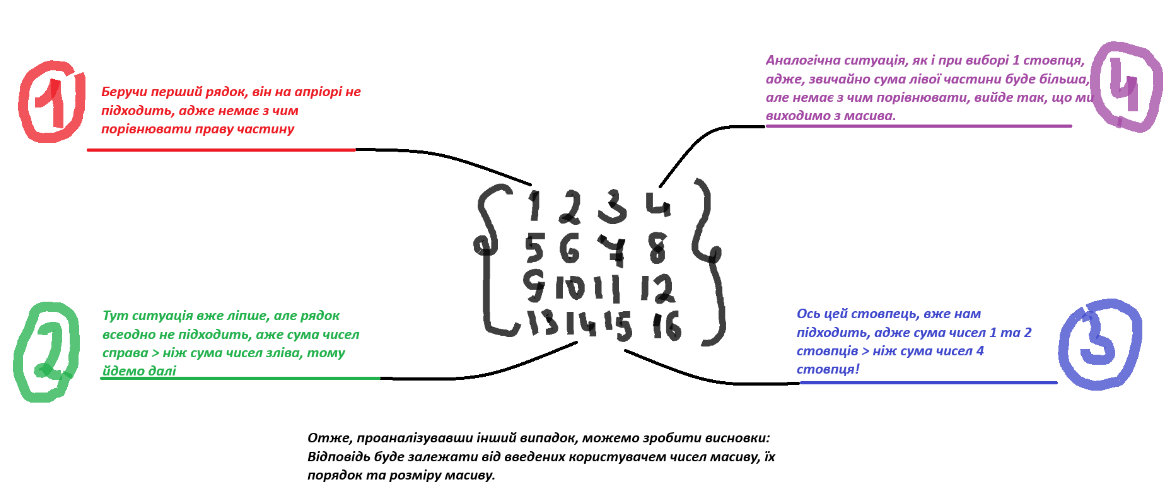
*Images 3. Роздуми над реалізацією завдання (початок)*

В теорії, хід думок частково правильний для цієї умови. Одже, поясню:

Умова вказує, що після відкидання деякого стовпця у матриці, сума елементів в першому блоці повинна бути більшою ніж у другому. Але, проблема у цьому випадку, є в тому, який стовпець ми будемо вважати першим, адже при нумеруванні стовпців, як 1 2 3 та 3 2 1, ми зіштовхуємося з двозначністю, при виборі 2-го стопвця. За однієї умови (при розташуванні 1 2 3), відповіддю буде не існування такого стовпця, а за іншої умови (3 2 1), відповіддю буде вважатись 2 стовпець. АЛЕ:

1. Таке можливе при виборі масиву користувачем, розміру: [3][3]
2. При виборі порядку стовпців.
3. При сумі елементів 1 та 3 (або 3 та 1) стовпців, які дорівнюють одне одному.

б) Ілюстрація ходу думок, коли направили на істину:

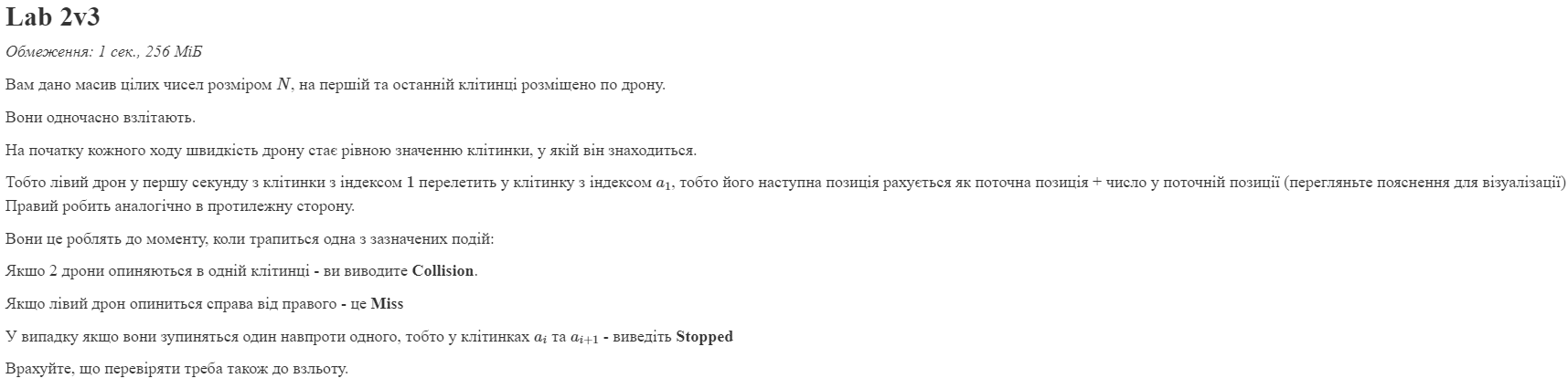


*Images 4. Роздуми над реалізацією завдання (кінець)*

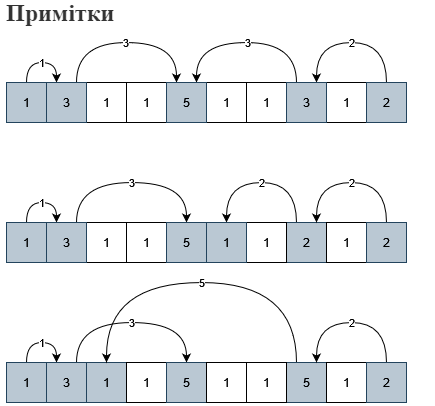
* В результаті, успішно написаний код, з морем емоцій! (**P.S Для закріплення написав аналогічний код, але у випадку з рядками**)

Завдання **№3 Algotester Lab 2v3**

* Варіант завдання: 3



*Image 5. Algotester Lab 2v3 (Умова завдання)*

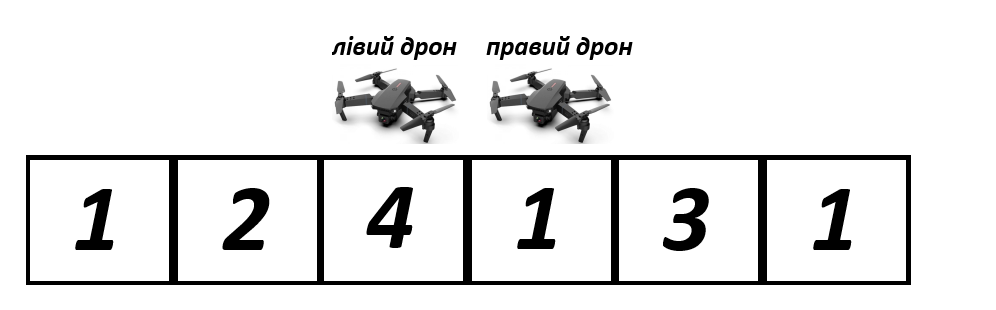
**

*Image 5. Algotester Lab 2v3 (Примітки, для візуалізаційного сприйняття)*

* Теж, не менш цікава задачка, в якій примітки допомогли зрозуміти, як пересуваються дрони при введені чисел масиву.

Отже:

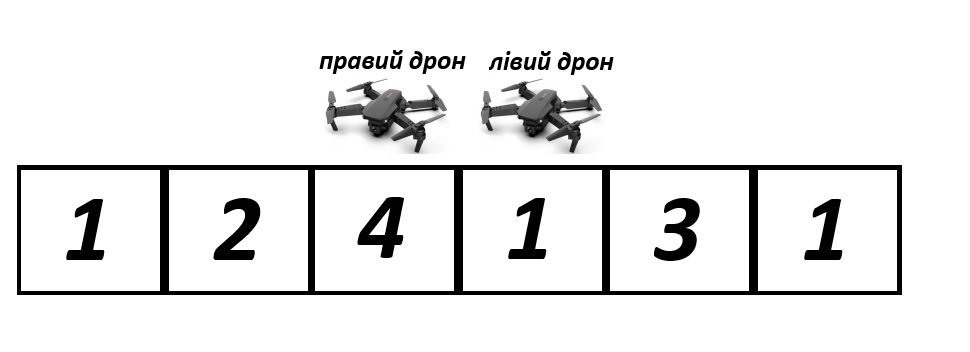
1. Виведення **STOPPED**: За умови, що дрони опиняться один навпроти одного.

****

1. Виведення **COLLISION**: За умови, що дрони опиняться один в одному.

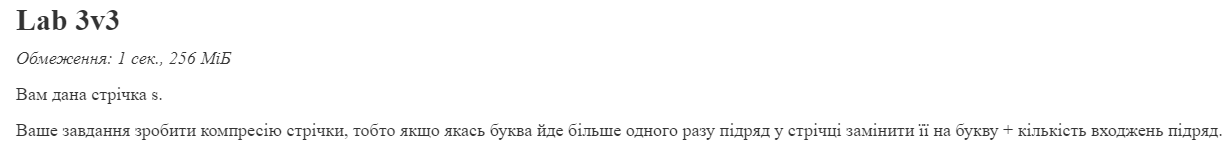


1. Виведення **MISS**: За умови, що дрони обминуться.



Завдання **№4 Algotester Lab 3v3**

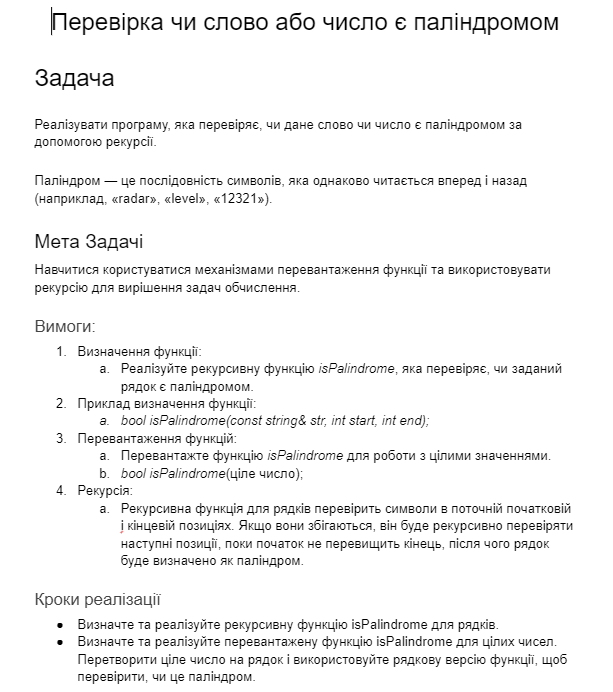
* Варіант завдання: 3



*Image 6. Algotester Lab 3v3 (Умова завдання)*

Завдання **№5 Class Practice Work**

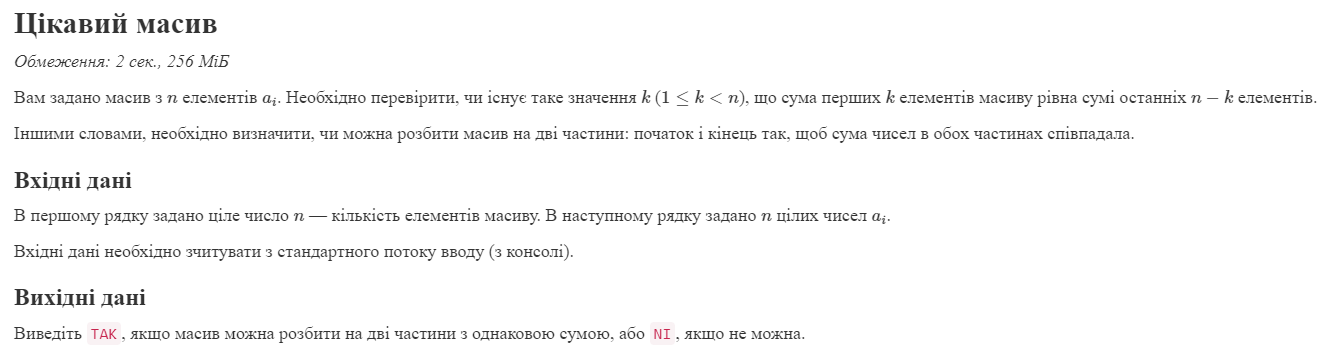
* Варіант відсутній.



*Image 7. Class Practice Work (Умова завдання)*

Завдання **№6 Self Practice Work**

* [Задача Algotester 1851](https://algotester.com/uk/ArchiveProblem/DisplayWithEditor/40910) – Цікавий масив.



*Image 8. Self Practice Work*

Використано масив, цілих чисел, прості цикли та вкладені цикли for. Спочатку не виходило, але потім глянув на вихідні дані.

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

Програма **№1 VNS Lab 4 - Task 1**

* Блок-схема: (відсутня)
* Планований час на реалізацію: *3* год => Затрачений час: *4* год.
* Важливі деталі для врахування в імплементації: Ініціалізація генератора випадкових чисел за допомогою поточного часу. Знаходження мінімального значення в масиву. Знаходження середнього значення масиву, та заміна цього значення на три перших числа в попередньому масиві.
* Програма **№2 VNS Lab 5 - Task 1**
* Блок-схема (відсутня)
* Планований час на реалізацію: *4* год => Затрачений час: *13* год.

Важливі деталі для врахування в імплементації: Відповідь буде залежити від введених користувачем чисел масиву, їх порядок та розміру масиву.

* Програма **№3 Algotester Lab 2v3**
* Блок-схема (відсутня)
* Планований час на реалізацію: *3* год => Затрачений час: *5* год.

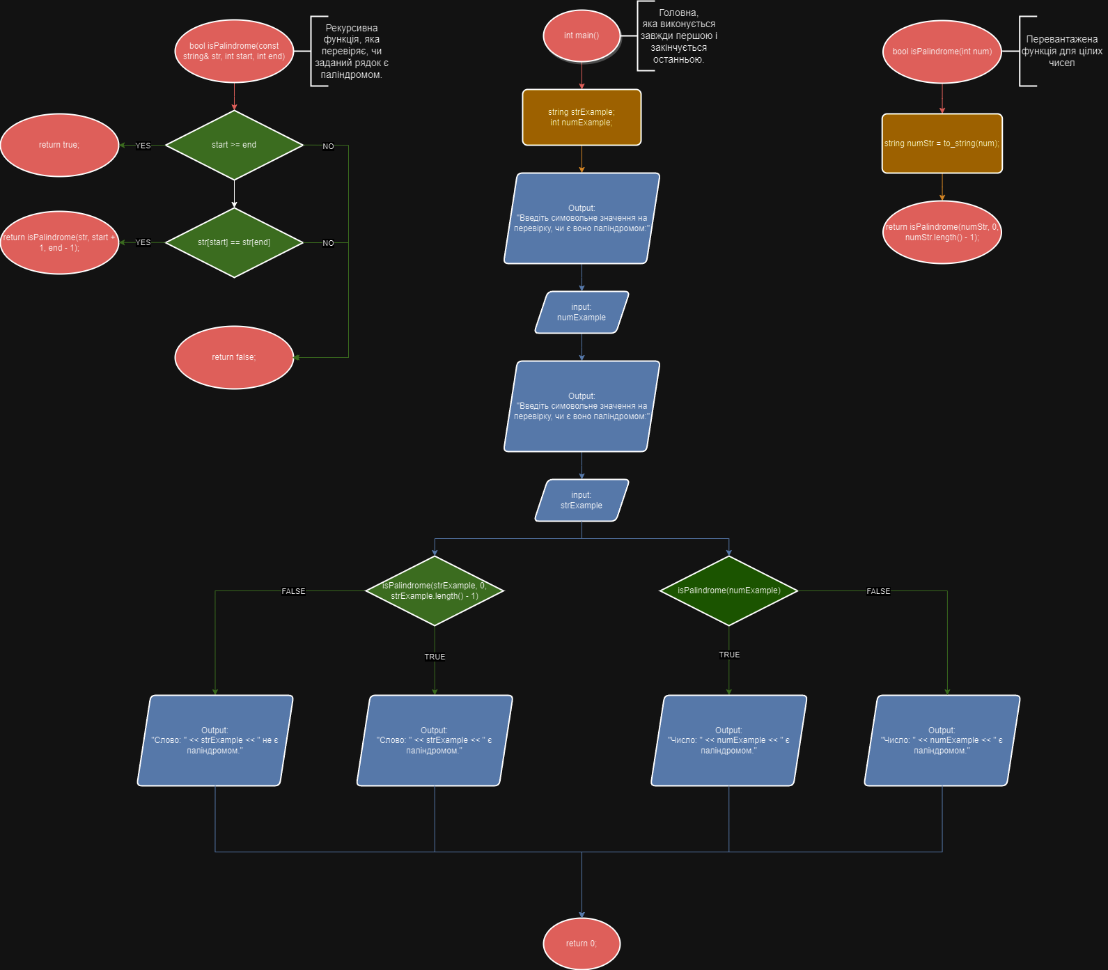
Важливі деталі для врахування в імплементації: Перевірка позицій лівого та правого дронів, для подальшого виведення одного з трьох результатів, в залежності від виконання умов.

Програма **№4 VNS Algotester Lab 3v3**

* Блок-схема (відсутня)
* Планований час на реалізацію: *2.5* год => Затрачений час: *4* год.
* Важливі деталі для врахування в імплементації: Утворено цикл, який проходить через кожний символ, який введений у рядку, у подальшому, додаємо к-сть однакових символів (якщо вони більше за 1) до змінної, та виводимо цю зміннy в термінал.

Програма **№5 Class Practice Work**

* Блок-схема



*Image 7. Flowchart for Class Practice Work.*

* Планований час на реалізацію: *12* *годин* => Затрачений час: *1 день*.

Програма **№6 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-4-Task-1.](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/697/commits/fbb04135248384c4260c10d1630183659661c92b#diff-d91efc8a1ca60b4803ce43c61a43e80a30939ef8631d210f48f2a03c792e754f)

Завдання **№1 VNS Lab 4 - Task 1**

#include <iostream>

#include <cstdlib>

#include <ctime>

#include <algorithm>

using namespace std;

int main() {

    int arraySize = 10; // Оголошую масив array з роміром на 10 елементів.

    int array[arraySize]; // Оголошую масивy з розміром arraySize, який ми визначили вище.

    // Заповнення масиву випадковими числами

    srand(static\_cast<unsigned int>(time(0))); // Ініціалізуємо генератор випадкових чисел за допомогою поточного часу. Використовуємо static\_cast<unsigned int>, оскільки srand очікує беззнаковий цілочисельний аргумент.

    // Цикл, який проходить через всі елементи масиву і заповнює їх випадковими числами від 0 до 20.

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

        array[i] = rand() % 20; // Генеруємо числа від 0 до 20.

    }

    // Роздруковуємо початковий масив

    cout << "Початковий масив: "; // Виводимо повідомлення про те, що ми виводимо початковий масив.

    // Цикл, який виводить на екран всі елементи масиву.

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

        cout << array[i] << " ";

    }

    cout << endl; // Перевів рядок для зручності виведення.

    // Знаходимо мінімальне значення в масиві

    int minValue = \*min\_element(array, array + arraySize); //  Знаходимо мінімальне значення в масиві за допомогою min\_element та зберігаю його у змінній minValue.

    // Видаляємо всі елементи, які співпадають з мінімальним значенням

    int\* newEnd = remove(array, array + arraySize, minValue); // Переміщаємо всі елементи масиву, які = minValue, в кінець масиву, але не видаляємо їх фізично. Результат remove є ітератором, що вказує на початок "зменшеного" масиву.

    arraySize = distance(array, newEnd); // Оновлюємо arraySize до нового розміру масиву після видалення елементів, використовуючи distance.

    // Знаходимо середнє арифметичне масиву

    double average = 0.0; // Ініціалізація зміннної average для обчислення середнього арифметичного.

    // Цикл, який сумує всі елементи масиву для подальшого обчислення середнього арифметичного.

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

        average += array[i];

    }

    average /= arraySize;

    // Додаємо три нових елементи на початок масиву, кожен із значенням середнього арифметичного (заокругленого до цілого числа).

    for (int i = 0; i < 3; ++i) {

        array[i] = static\_cast<int>(average);

    }

    cout << "Змінений масив: "; // (Родруковуємо) Виводимо змінений масив.

    // Цикл, який виводить на екран всі елементи зміненого масиву.

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

        cout << array[i] << " ";

    }

    return 0;

}

}

*Блок №1. Код до програми №1 VNS Lab 4 - Task 1*

Завдання **№2 VNS Lab 5 - Task 1**

Посилання на файл програми у пул-запиті GitHub: [pull-request-link-VNS-Lab-5-Task-1.](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/697/commits/fbb04135248384c4260c10d1630183659661c92b#diff-17db4cf1470f176e30311cb1c882946c847ec85a91e4d793092785b3d24a6013)

#include <iostream>

#include <vector>

using namespace std;

int main() {

    vector<vector<int>> matrix = { // Створюємо двовимірний вектор matrix.

        {1, 2, 3, 4, 5},

        {7, 8, 9, 10, 11},

        {12, 13, 14, 15, 16},

        {17, 18, 19, 20, 21}

    };

    int rows = matrix.size(); // Визначаємо к-сть рядків у матриці, де функція size() для вектора повертає його розмір.

    int cols = matrix[0].size(); // Визначаємо кількості стовпців у матриці, та припускаємо, що всі рядки матриці мають однаковий розмір, тому визначення розміру першого рядка дає кількість стовпців.

    int count = 0; // Створюємо змінну count, для підрахунку к-сті стовпців, які задовільняють певній умові.

    // Зовнішній цикл, ітерується через стовпці матриці, залишаючи зовнішній стовпець. Тобто, це виконується для кожного стовпця, крім першого та останнього.

    for (int j = 1; j < cols - 1; ++j) {

        int sum\_left = 0; // Змінна для обчислення суми елементів стовпців зліва, від поточного стовпця j.

        int sum\_right = 0; // Змінна для обчислення суми елементів стовпців справа, від поточного стовпця j.

        // Внутрішній цикл, який ітерується через стовпці, що знаходяться ліворуч від поточного стовпця j.

        for (int k = 0; k < j; ++k) {

            // Внутрішній цикл, який ітерується через рядки матриці.

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

                sum\_left += matrix[i][k]; // Додаємо елементи, які розташовані ліворуч від поточного стовпця j.

            }

        }

        // Внутрішній цикл, який ітерується через стовпці, що знаходяться праворуч від поточного стовпця j.

        for (int k = j + 1; k < cols; ++k) {

            // Внутрішній цикл, який ітерується через рядки матриці.

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

                sum\_right += matrix[i][k]; // Додаємо елементи, які розташовані праворуч від поточного стовпця j.

            }

        }

        if (sum\_left > sum\_right) { // Перевіряємо, чи сума елементів ліворуч від поточного стовпця j більше суми елементів праворуч.

            cout << "Стовпець " << j << " ділить матрицю таким чином, що сума елементів зліва більше суми справа.\n";

            count++; // При справдженні умови, збільшуємо лічильник.

        }

    }

    cout << "Таких стовпців: " << count;

    return 0;

}

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

Завдання **№3 VNS Lab 5 - Task 1 (additional)**

Посилання на файл програми у пул-запиті GitHub: [pull-request-link-VNS-Lab-5-Task-1.](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/697/commits/fbb04135248384c4260c10d1630183659661c92b#diff-daff30516c91d704a5205436704a5f1872d3f958b6f5f5a388f30840ba7c47da) (additional)

#include <iostream>

#include <vector>

using namespace std;

int main() {

    vector<vector<int>> matrix = { // Створюємо двовимірний вектор matrix.

        {1, 2, 3, 4, 5},

        {6, 7, 8, 9, 10},

        {11, 12, 13, 14, 15},

        {16, 17, 18, 19, 20}

    };

    int rows = matrix.size(); // Визначаємо к-сть рядків у матриці, де функція size() для вектора повертає його розмір.

    int cols = matrix[0].size(); // Визначаємо кількості стовпців у матриці, та припускаємо, що всі рядки матриці мають однаковий розмір, тому визначення розміру першого рядка дає кількість стовпців.

    int count = 0; // Створюємо змінну count, для підрахунку к-сті рядків, які задовільняють певній умові.

    // Зовнішній цикл, ітерується через рядки матриці, залишаючи зовнішній рядок. Тобто, це виконується для кожного рядка, крім першого та останнього.

    for (int i = 1; i < rows - 1; ++i) {

        int sum\_up = 0; // Змінна для обчислення суми елементів рядків зверху, від поточного рядка i.

        int sum\_down = 0; // Змінна для обчислення суми елементів рядків знизу, від поточного рядка i.

         // Внутрішній цикл, який ітерується через рядки, що знаходяться зверху від поточного рядка і.

        for (int k = 0; k < i; ++k) {

            // Внутрішній цикл, який ітерується через стовпці матриці.

            for (int j = 0; j < cols; ++j) {

                sum\_up += matrix[k][j]; // Додаємо елементи, які розташовані зверху від поточного рядка i.

            }

        }

        // Внутрішній цикл, який ітерується через рядки, що знаходяться знизу від поточного рядка i.

        for (int k = i + 1; k < rows; ++k) {

            // Внутрішній цикл, який ітерується через стовпці матриці.

            for (int j = 0; j < cols; ++j) {

                sum\_down += matrix[k][j]; // Додаємо елементи, які розташовані знизу від поточного рядка i.

            }

        }

        if (sum\_up > sum\_down) { // Перевіряємо, чи сума елементів зверху від поточного стовпця i більше суми елементів знизу.

            cout << "Рядок " << i << " ділить матрицю таким чином, що сума елементів зверху більше суми знизу.\n";

            count++; // При справдженні умови, збільшуємо лічильник.

        }

    }

    cout << "Таких рядків: " << count;

    return 0;

}

*Блок №3. Код до програми №3 VNS Lab 5 - Task 1 (Additional)*

Завдання **№4 Algotester Lab 2v3**

Посилання на файл програми у пул-запиті GitHub: [pull-request-link-Algotester-Lab-2-v-3](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/697/commits/fbb04135248384c4260c10d1630183659661c92b#diff-c8c1c49cc68168c2d75b0a5712c7151b111d4078a8f9e06689f8f53d2aaa5c34).

#include <iostream>

using namespace std;

int main() {

  int inscribeInteger;

  int newInteger;

  cin >> inscribeInteger; // Зчитуємо у користувача змінну inscribeInteger.

  int arr[inscribeInteger]; // Масив, який вміщає в собі inscribeInteger-елементів.

  // Цикл, який повторюється inscribeInteger-разів.

  for (newInteger = 0; newInteger < inscribeInteger; newInteger++) {

    cin >> arr[newInteger]; // Зберігаємо числа введені користувачем в новий масив arr[newInteger].

  }

  int leftDrone = 1, rightDrone = inscribeInteger; // Вказуємо позицію лівого та правого дрона відповідно.

  // Нескінченний цикл, який виконується поки не зустрінеться оператор break.

  while (1) {

    if (leftDrone > rightDrone) { // Перевіряємо чи перетув лівий дрон правого, рухаючись по масиву

      cout << leftDrone << " " << rightDrone; // Виводимо поточні позиції обох дронів.

      cout << "\nMiss";

      break; // Виходимо з циклу, якщо задовільняє умова.

    } else if (leftDrone == rightDrone) { // Перевіряємо чи обидва дрона розташовані на одній позиції.

      cout << leftDrone << " " << rightDrone; // Виводимо поточні позиції обох дронів.

      cout << " \nCollision";

      break; // Виходимо з циклу, якщо задовільняє умова.

    } else if (leftDrone == rightDrone - 1) { // Перевіряємо чи лівий дрон на одну позиції позаду ніж правий дрон.

      cout << leftDrone << " " << rightDrone; // Виводимо поточні позиції обох дронів.

      cout << " \nStopped";

      break; // Виходимо з циклу, якщо задовільняє умова.

    }

    // Отримуємо значення елемента arr[leftDrone - 1]

    // Додаємо отримане значення до поточної позиції лівого дрона.

    // Призначаємо нову позицію лівому дрону.

    leftDrone += arr[leftDrone - 1];

    // Отримуємо значення елемента arr[rightDrone - 1].

    // Віднімаємоо отримане значення до поточної позиції правого дрона.

    // Призначаємо нову позицію лівому дрону.

    rightDrone -= arr[rightDrone - 1];

  }

  return 0;

}

*Блок №4. Код до програми №4 Algotester 2v3*

Завдання **№4 Algotester Lab 3v3**

Посилання на файл програми у пул-запиті GitHub: [pull-request-link-Algotester-Lab-3-v-3.](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/697/commits/fbb04135248384c4260c10d1630183659661c92b#diff-d8287de29cb51b98a885c6c4260bb7ac2b277effcf0187a24dd604998cb1a892)

#include <iostream>

#include <string>

using namespace std;

int main() {

    string input; // Змінна, для зберігання введений символів.

    cin >> input;

    string S\_compressed; // Змінна, для збереження стиснутого рядка (тобто компресії)

    // Змінна для визначення кількості однакових символів

    int count = 1;

    // Цикл, який проходить через кожний символ, який введений у рядку.

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

        // Перевіряємо, чи це перший символ або він відрізняється від попереднього

        if (i == 0 || input[i] != input[i - 1]) { // Умова, яка перевіряє, чи поточний символ відрізняється від попереднього або чи це перший символ у рядку.

            if (count > 1) { // Перевіряємо, якщо к-сть однакових символів > 1, то зберігаємо до змінної S\_compressed.

                S\_compressed += to\_string(count);

            }

            S\_compressed += input[i]; // Додаємо поточний символ до змінної S\_compressed.

            count = 1; // Скидаємо лічильник, оскільки переходимо до іншого символу.

        } else {

            // Якщо поточний символ той самий, що і попередній, збільшуємо лічильник count.

            count++;

        }

    }

    // Додаємо кількість останніх символів, якщо вони більше 1

    if (count > 1) {

        S\_compressed += to\_string(count); // Додаємо к-сть однакових симоволів (якщо вони більші за 1) до змінної S\_compressed.

    }

    // Виводимо стиснутий рядок

    cout << S\_compressed << endl;

    return 0;

}

*Блок №5. Код до програми №4 Algotester 3v3*

Завдання **№5 Class Practice Work.**

Посилання на файл програми у пул-запиті GitHub: [pull-request-link-Class-Practice-Work.](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/697/commits/fbb04135248384c4260c10d1630183659661c92b#diff-53aa29e9ee6b7fbfb4155fd1b3b3f5184b89f9977bc80e890d4f9dc24b5bcd04)

#include <iostream>

#include <string>

using namespace std;

bool isPalindrome(const string& str, int start, int end) { // Рекурсивна функція, яка перевіряє, чи заданий рядок є паліндромом.

    if (start >= end) { // Перевіряємо, чи дойшли до кінця області рядка, для перевірки паліндрома. (Тобто start >= end)

        return true; // Якщо, умова справдилась, виклик рекурсивної функції зупиняється, і повертається значення true. (Тобто, рядок який залишився для перевірки, складається лише з одного символу або є пустим, і тому вона вважається паліндромом.)

    }

    if (str[start] == str[end]) { // Перевіряємо чи перший символ (позиція старт) = остананньому символу (позиція end)

        return isPalindrome(str, start + 1, end - 1); // Якщо умова виконується, то функція викликає саму себе (тобто рекурсія), і збільшує початкову позицію на 1, та зменшує кінцеву на 1.

    }

    return false; // Якщо хоча б один символ не співпадає, то рядок не є паліндромом

}

bool isPalindrome(int num) { // Перевантажена функція для цілих чисел

    string numStr = to\_string(num); // Перевторюємо ціле число (num) в рядок (string). Використовуємо для цього функцію to\_string

    // Використовуємо рядкову версію функції для перевірки, чи це паліндром

    return isPalindrome(numStr, 0, numStr.length() - 1); // Таким чином, ми використовуємо рекурсивну функцію, яка вже була визначена для рядків, для перевірки, чи ціле число є паліндромом.

}

int main() {

    string strExample;

    int numExample;

    cout << "Введіть симовольне значення на перевірку, чи є воно паліндромом:" << endl;

    cin >> strExample;

    cout << "Введіть чисельне значення на перевірку, чи є воно паліндромом:" << endl;

    cin >> numExample;

    if (isPalindrome(strExample, 0, strExample.length() - 1)) { // Виклик функції з параметрами, для перевірки на паліндром усього рядка, з першого (srtExample, 0) до останнього символу (strExample.length() - 1).

        cout << "Слово: " << strExample << " є паліндромом." << endl;

    } else {

        cout << "Слово: " << strExample << " не є паліндромом." << endl;

    }

    if (isPalindrome(numExample)) { // Виклик функції, для перевірки числа на паліндром. Тобто функція перетворює ціле число в рядок і викликає рекурсивну функцію для перевірки паліндрома цього рядка.

        cout << "Число: " << numExample << " є паліндромом." << endl;

    } else {

        cout << "Число: " << numExample << " не є паліндромом." << endl;

    }

    return 0;

}

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

Завдання **№6 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/697/commits/fbb04135248384c4260c10d1630183659661c92b#diff-a84fd209ef2ea49170d6e9759021aebc5fb9bb045902c8bd64db8333e2faa25f)

#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 №7. Код до програми №7 Self Practice Work (Algotester 0021)*

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

Завдання **№1 VNS Lab 4 - Task 1**



*Image 11. Результати виконання завдання №1 VNS Lab 4 - Task 1 (1 запуск)*



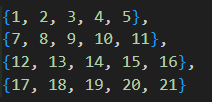
*Image 12. Результати виконання завдання №1 VNS Lab 4 - Task 1 (2 запуск)*



*Image 13. Результати виконання завдання №1 VNS Lab 4 - Task 1 (3 запуск)*

Час затрачений на виконання завдання: *4 год.*

Завдання **№2 VNS Lab 5 - Task 1**. *I*

**

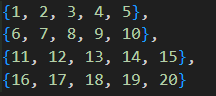
*Image 14. Умова, за якою відбуваються обчислення №1 VNS Lab 5 - Task 1*

**

*I Image 15. Результати виконання завдання №2 VNS Lab 5 - Task 1*

Час затрачений на виконання завдання: *13 год.*

Завдання **№3 VNS Lab 5 - Task 1 (additional)**

**

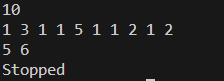
*Image 16. Умова, за якою відбуваються обчислення №1 VNS Lab 5 - Task 1 (additional)*

**

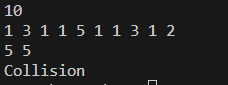
*Image 17. Результати виконання завдання №3 VNS Lab 5 - Task 1* *(additional)*

Час затрачений на виконання завдання: *13 год.*

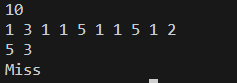
Завдання **№3** **Algotester Lab 2v3**

**

*Image 18. Результати виконання завдання №3 Algotester Lab 2v3 (Stopped)*

**

*Image 19. Результати виконання завдання №3 Algotester Lab 2v3 (Collision)*

**

*Image 20. Результати виконання завдання №3 Algotester Lab 2v3 (Miss)*

Час затрачений на виконання завдання: *5 год.*

Завдання **№4 Algotester Lab 3v3**



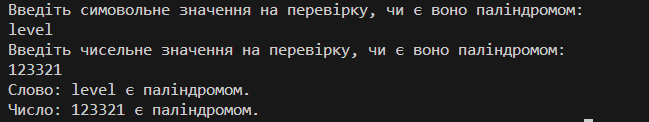
*Image 21. Результати виконання завдання №4 Algotester Lab 3v3 (1 запит)*



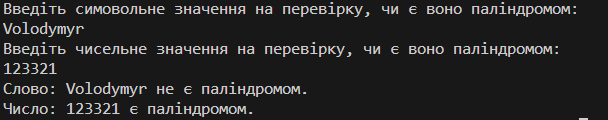
*Image 22. Результати виконання завдання №4 Algotester Lab 2v3 (2 запит)*

Час затрачений на виконання завдання: *4 год.*

Завдання **№5 Class Practice Work**



*Image 23. Результати виконання завдання №5 Class Practice Work (1 запит)*



*Image 24. Результати виконання завдання №5 Class Practice Work* *(2 запит)*

Час затрачений на виконання завдання: *1 день.*

Завдання **№6 Self Practice Work**



*Image 25. Результати виконання завдання №6 Self Practice Work (1 умова)*

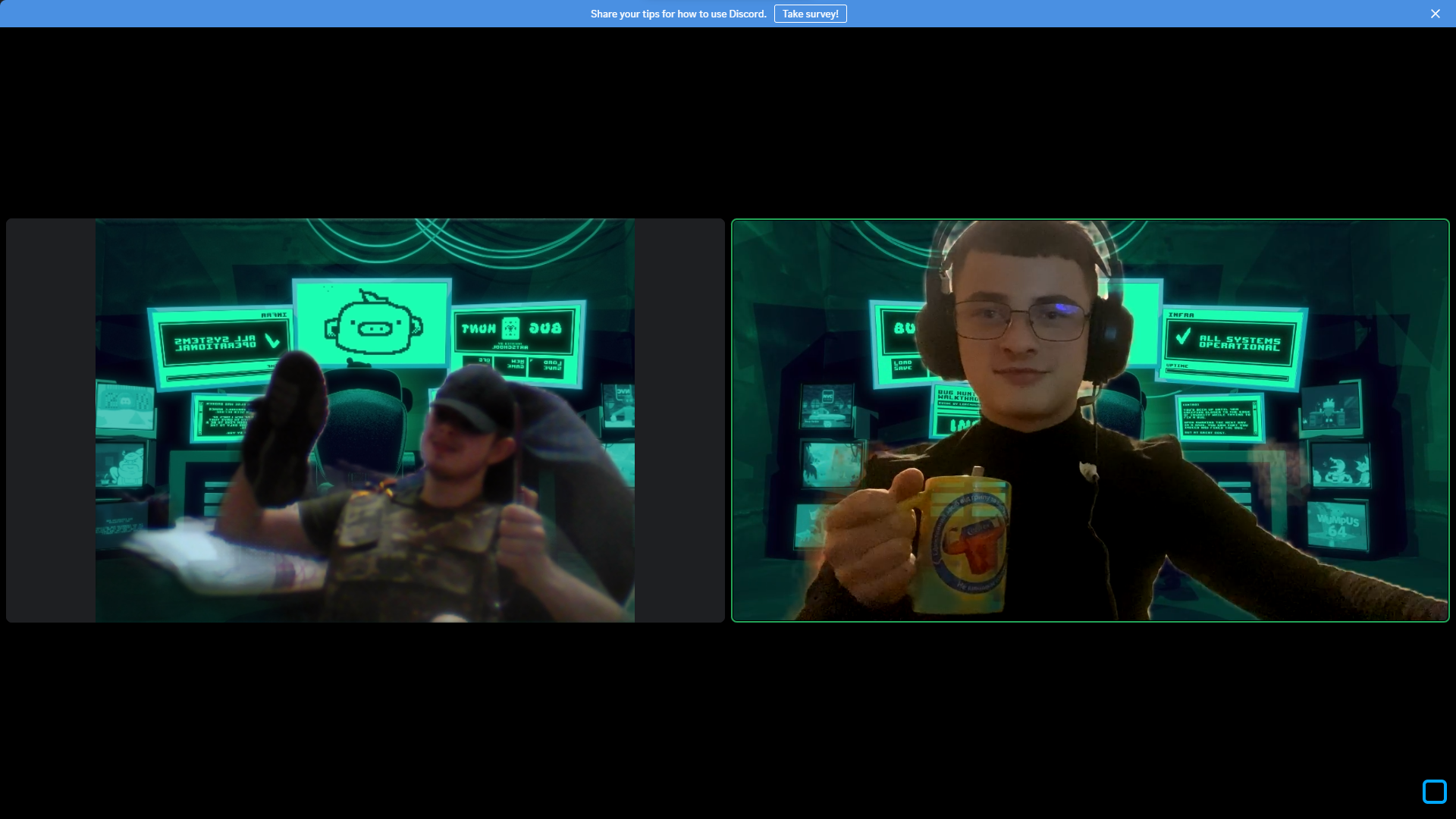


*Image 26. Результати виконання завдання №6 Self Practice Work* *(2 умова)*

Час затрачений на виконання завдання: *1 год.*

## **6. Кооперація з командою:**

Час затрачений на обговорення з командою: *15 год.*



*Image 27. Кооперація з командою (12.4.2023).*



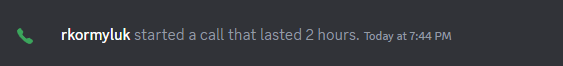
*Image 28. Кооперація з командою (11.30.2023)*



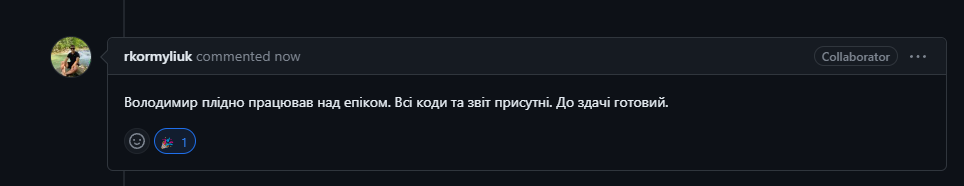
*Image 29. Кооперація з командою (12.2.2023)*

**

*Image 30. Кооперація з командою (12.3.2023)*

**

*Image 31. Кооперація з командою (12.4.2023)*

**

*Image 30. Коментар від побратима по команді (12.5.2023)*

# **Висновки:**

Під час вивчення тем, зокрема одновимірних та двовимірних масивів, а також алгоритмів обробки, було здобуто знання та навички у роботі з даними структурами даних.

Вивчення одновимірних масивів дозволило освоїти базові концепції індексації та роботи із збереженням даних в послідовних блоках пам'яті. Це стало фундаментом для подальшого розуміння та реалізації більш складних алгоритмів та програм.

Вивчення двовимірних масивів розширило розуміння використання простору для зберігання інформації та надало навички роботи з матрицями.

Алгоритми обробки стали ключовим компонентом вивчення, дозволяючи розробляти ефективні та оптимізовані рішення для різноманітних завдань.

У підсумку, вивчення цих тем дало хороший фундамент для подальшого розвитку навичок програмування C++, розкриваючи можливості роботи з масивами різної структури та застосування алгоритмів для ефективної обробки даних.