Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту

Звіт

про виконання лабораторних та практичних робіт блоку № 3

На тему: “Одновимірні масиви. Двовимірні Масиви. Вказівники та Посилання. Динамічні масиви. Структури даних. Вкладені структури. Алгоритми обробки та робота з масивами та структурами”.

З дисципліни: «Основи програмування»

до:

Практичних Робіт до блоку № 4

Виконав:

Студент групи ШІ-11

Голейчук Іван Миколайович

Львів 2024

Тема роботи: “Одновимірні масиви. Двовимірні Масиви. Вказівники та Посилання. Динамічні масиви. Структури даних. Вкладені структури. Алгоритми обробки та робота з масивами та структурами.

Мета роботи: "Ознайомлення з основами роботи з одновимірними та двовимірними масивами, вказівниками, посиланнями та динамічними масивами. Розуміння структур даних і вкладених структур, а також освоєння алгоритмів обробки даних і практичної роботи з масивами та структурами для ефективного вирішення завдань програмування."

Теоретичні відомості:

1. Класи пам'яті у C++
   - Статична пам’ять.
   - Динамічна пам’ять.
   - Поняття стеку.
   - Виділення та вивільнення пам’яті.
2. Вступ до Масивів і Вказівників:
   - Основи масивів: визначення, важливість, приклади використання.
   - Різниця між статичними та динамічними масивами.
   - Основи вказівників: що це таке, як вони працюють.
   - Взаємозв'язок між масивами та вказівниками.
   - Вступ до посилань: основні концепції та відмінності від вказівників.
3. Одновимірні Масиви:
   - Створення та ініціалізація одновимірних масивів.
   - Основні операції: індексація, присвоєння, читання.
   - Цикли та обхід масивів.
   - Використання функцій для роботи з масивами.
   - Приклади алгоритмів сортування та пошуку.
4. Вказівники та Посилання:
   - Використання вказівників для доступу до елементів масиву.
   - Арифметика вказівників.
   - Різниця між вказівниками та посиланнями в контексті функцій.
   - Динамічне виділення пам'яті з використанням вказівників.
   - Використання вказівників для створення складних структур даних.
5. Двовимірні Масиви:
   - Оголошення та ініціалізація двовимірних масивів.
   - Вкладені цикли для обходу двовимірних масивів.
   - Практичні приклади використання двовимірних масивів.
   - Передача двовимірних масивів у функції.
   - Застосування двовимірних масивів для розв'язання задач.
6. Динамічні Масиви:
   - Основи динамічного виділення пам'яті.
   - Створення та управління динамічними масивами.

- Використання операторів new та delete для управління пам'яттю.
- Реалізація змінної розмірності масивів.
- Передача динамічних масивів у функції.

7. Структури Даних:
   - Оголошення та використання структур.
   - Використання масивів та вказівників у структурах.
   - Функції для обробки даних у структурах.
   - Використання структур для представлення складних даних.
   - Вкладені структури та їх використання.
   - Об’єднання (Union)
   - Переліки (enumerations)
8. Вкладені Структури:
   - Поняття вкладених структур та їх оголошення.
   - Взаємодія з вкладеними структурами.
   - Використання вкладених структур для моделювання складних даних.
   - Передача вкладених структур у функції.
   - Приклади реального використання вкладених структур.
9. Використання структур
   - Перевантаження операторів у структурі.
   - Вивід/ввід структури (operator<<);
   - Арифметичні операції з структурами (operator+, operator-);
   - Практичні задачі на виведення структур та операції з ними
10. Алгоритми обробки та робота з масивами та структурами:
    - Алгоритми пошуку та сортування в масивах.
    - Обробка та маніпуляції з даними у структурах.
    - Використання циклів та умовних операторів для роботи з масивами та структурами.
    - Інтеграція масивів та структур у алгоритми.
- Розв'язання практичних задач з використанням масивів та структур.

Індивідуальний план опрацювання теорії:

1. Класи пам'яті у C++

Джерела інформації:

-Лекції Олександра Пшеничного;

-Практичні заняття;

-Використання штучного інтелекту (чат gpt);

-Youtube.

Що опрацьовано: Типи класів пам'яті (автоматичний, статичний, динамічний, зовнішній, регістр). Їх призначення та застосування.

2. Вступ до Масивів і Вказівників:

Джерела інформації:

-Лекції Олександра Пшеничного;

-Практичні заняття;

-Використання штучного інтелекту (чат gpt);

-Youtube.

Що опрацьовано: Основи роботи з масивами, їх оголошення, ініціалізація, зв'язок масивів і вказівників.

3. Одновимірні Масиви:

Джерела інформації:

-Лекції Олександра Пшеничного;

-Практичні заняття;

-Використання штучного інтелекту (чат gpt);

-Youtube.

Що опрацьовано: Створення, доступ до елементів, операції сортування, пошуку та обчислення.

4. Вказівники та Посилання:

Джерела інформації:

-Лекції Олександра Пшеничного;

-Практичні заняття;

-Використання штучного інтелекту (чат gpt);

-Youtube.

Що опрацьовано: Робота з вказівниками, адресація, передача параметрів за адресою, посилання як альтернатива вказівникам.

5. Двовимірні Масиви:

Джерела інформації:

-Лекції Олександра Пшеничного;

-Практичні заняття;

-Використання штучного інтелекту (чат gpt);

-Youtube.

Що опрацьовано: Ініціалізація, доступ до елементів, використання вкладених циклів, операції над таблицями.

6. Динамічні Масиви:

Джерела інформації:

-Лекції Олександра Пшеничного;

-Практичні заняття;

-Використання штучного інтелекту (чат gpt);

-Youtube.

Що опрацьовано: Виділення та звільнення пам'яті, використання new і delete, реалізація динамічних структур даних.

7. Структури Даних:

Джерела інформації:

-Лекції Олександра Пшеничного;

-Практичні заняття;

-Використання штучного інтелекту (чат gpt);

-Youtube.

Що опрацьовано: Створення та використання структур для зберігання складних типів даних.

8. Вкладені Структури:

Джерела інформації:

-Лекції Олександра Пшеничного;

-Практичні заняття;

-Використання штучного інтелекту (чат gpt);

-Youtube.

Що опрацьовано: Створення структур у структурах, обробка вкладених даних.

9. Використання структур

Джерела інформації:

-Лекції Олександра Пшеничного;

-Практичні заняття;

-Використання штучного інтелекту (чат gpt);

-Youtube.

Що опрацьовано: Реальні приклади роботи зі структурами для моделювання об'єктів.

10. Алгоритми обробки та робота з масивами та структурами:

Джерела інформації:

-Лекції Олександра Пшеничного;

-Практичні заняття;

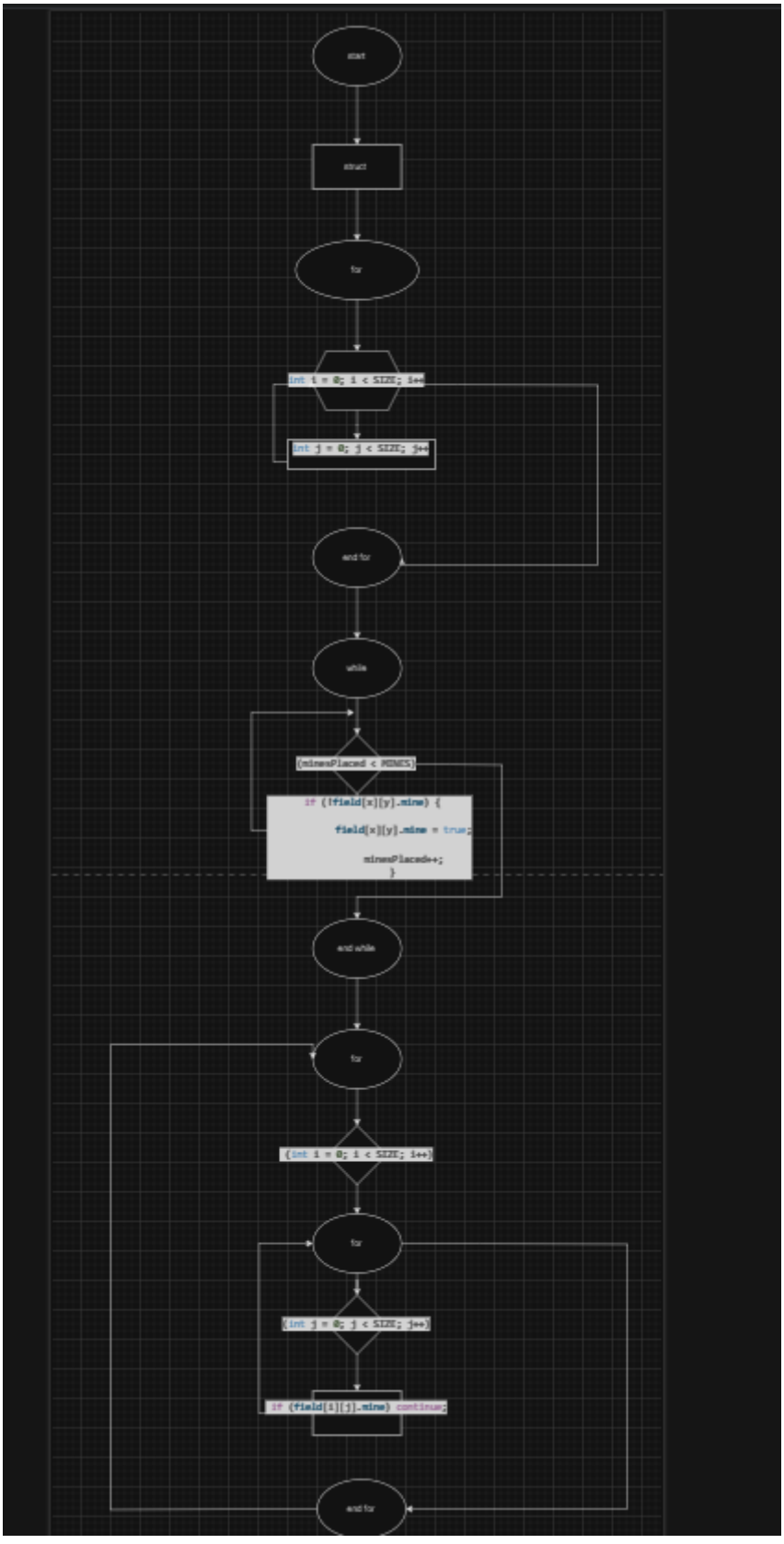
-Використання штучного інтелекту (чат gpt);

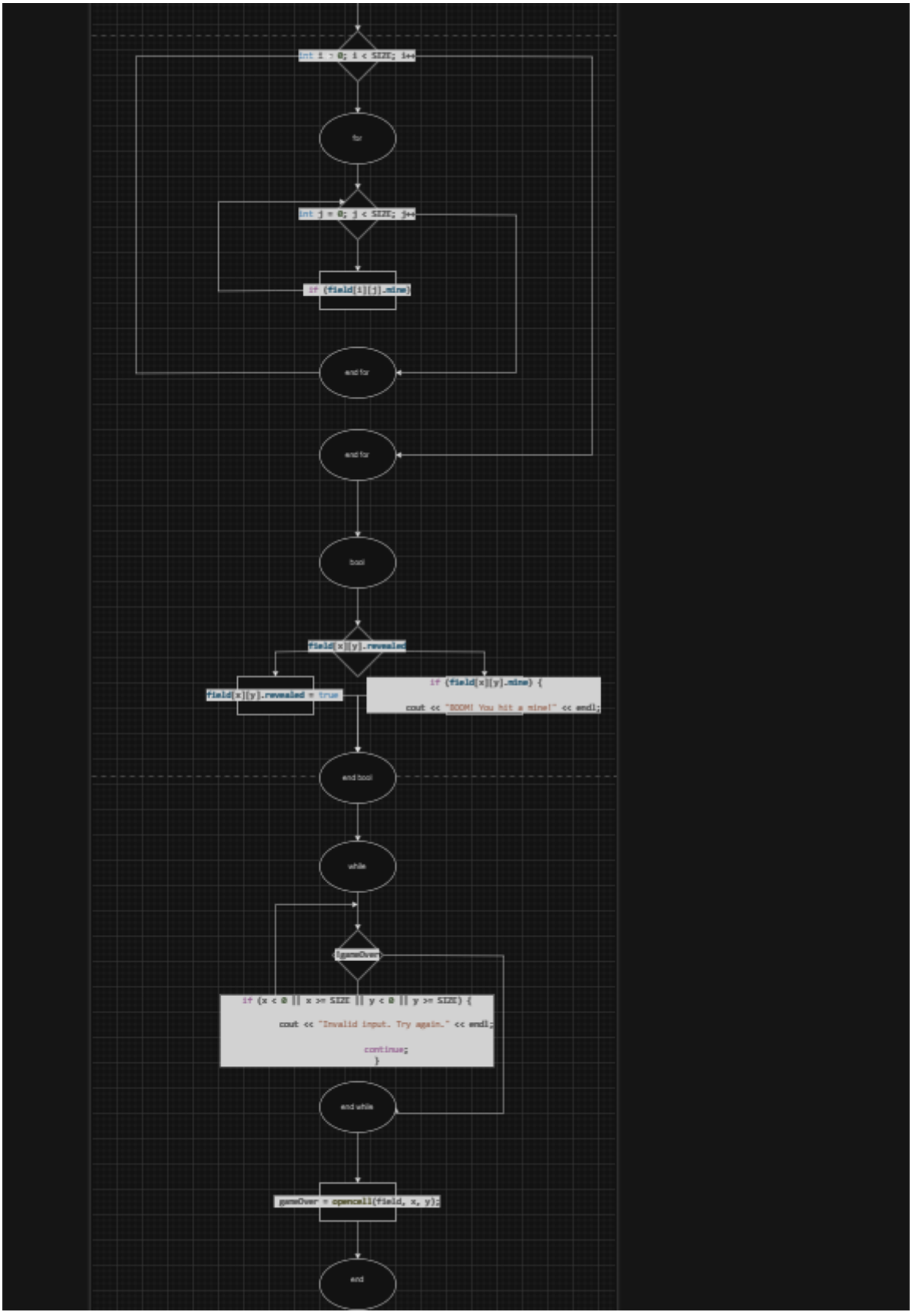
-Youtube.

Що опрацьовано: Реалізація алгоритмів сортування, пошуку, аналізу даних, обчислення та їх використання для практичних завдань.

Виконання роботи:

Task 2 - Requirements management (understand tasks) and design activities


start
struct
for
int i = 0; i < SIZE; i++
int j = 0; j < SIZE; j++
end for
while
(minesPlaced < MINES)
if (!field[x][y].mine) {
field[x][y].mine = true;
minesPlaced++;
}
end while
for
(int i = 0; i < SIZE; i++)
for
(int j = 0; j < SIZE; j++)
if (field[i][j].mine) continue;
end for


int i = 0; i < SIZE; i++
for
int j = 0; j < SIZE; j++
if (field[i][j].mine)
end for
end for
bool
field[x][y].revealed
field[x][y].revealed = true
if (field[x][y].mine) {
cout << "BOOM! You hit a mine!" << endl;
end bool
while
!gameOver
if (x < 0 || x >= SIZE || y < 0 || y >= SIZE) {
cout << "Invalid input. Try again." << endl;
continue;
}
end while
gameOver = openCell(field, x, y);
end

Task 3 - Lab# programming: VNS Lab 4

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

using namespace std;

void printRingLeft(const vector<int>& ring, int start) {

    int size = ring.size();
    int index = start;


    do {

        cout << ring[index] << " ";
        index = (index - 1 + size) % size;

    } while (index != (start - 1 + size) % size);

    cout << endl;
}

void printRingRight(const vector<int>& ring, int start) {

    int size = ring.size();
    int index = start;


    do {

        cout << ring[index] << " ";
        index = (index + 1) % size;

    } while (index != (start + 1) % size);

    cout << endl;
}

void deleteMaxElements(vector<int>& ring) {

    int maxElement = *max_element(ring.begin(), ring.end());
    ring.erase(remove(ring.begin(), ring.end(), maxElement), ring.end());

}

int main() {

    int n, k;

    cout << "[illegible] [illegible] [illegible] (n): ";

    cin >> n;

    vector<int> ring(n);

    cout << "[illegible] [illegible] [illegible]: ";

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

        cin >> ring[i];

    }

    cout << "[illegible] [illegible] [illegible] (K): ";

    cin >> k;

    k = (k % n + n) % n;

    cout << "[illegible] [illegible] [illegible] [illegible] [illegible] K [illegible] K-1: ";

    printRingLeft(ring, k);


    deleteMaxElements(ring);


    cout << "[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]

    printRingRight(ring, k);




    return 0;
}
```

Task 4 - Lab# programming: VNS Lab 5

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

using namespace std;



void fill_2d_array(const vector<int>& arr, vector<vector<int>>& matrix) {

    int n = arr.size();

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

        int sum = 0, num = 1;

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

            while (sum + num > arr[j]) num++;

            matrix[i][j] = num;

            sum += num;

            num++;
        }
    }
}


void print_matrix(const vector<vector<int>>& matrix) {

    for (const auto& row : matrix) {

        for (int elem : row) {

            cout << elem << " ";
        }
        cout << endl;
    }
}

int main() {

    int n;

    cout << "[illegible] [illegible] [illegible] (N): ";

    cin >> n;

    vector<int> arr(n);

    cout << "[illegible] [illegible] [illegible]: ";

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

        cin >> arr[i];
    }

    vector<vector<int>> matrix(n, vector<int>(n, 0));

    fill_2d_array(arr, matrix);

    cout << "[illegible] [illegible] [illegible]: " << endl;

    print_matrix(matrix);



    return 0;
}
```

Task 5 - Lab# programming: Algotester Lab 2

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

using namespace std;


int fatigue(vector<int>& r, int n) {

    int max_val = *max_element(r.begin(), r.end());

    int min_val = *min_element(r.begin(), r.end());

    int res = max_val - min_val;

    vector<int> temp = r;

    if (count(r.begin(), r.end(), max_val) == 1) {

        temp.erase(remove(temp.begin(), temp.end(), max_val), temp.end());

        int new_max = *max_element(temp.begin(), temp.end());

        res = min(res, new_max - min_val);

    }

    temp = r;

    if (count(r.begin(), r.end(), min_val) == 1) {

        temp.erase(remove(temp.begin(), temp.end(), min_val), temp.end());

        int new_min = *min_element(temp.begin(), temp.end());

        res = min(res, max_val - new_min);
    }

    return res;
}

int main() {

    int n;

    cin >> n;

    vector<int> r(n);


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

        cin >> r[i];
    }

    cout << fatigue(r, n) << endl;

    return 0;
}

```

Task 6 - Lab# programming: Algotester Lab 3

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

using namespace std;

int main() {

    int n, m;

    cin >> n;

    vector<int> a(n);


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

        cin >> a[i];
    }

    cin >> m;

    vector<int> b(m);

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

        cin >> b[i];
    }

    int common = 0;

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

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

            if (a[i] == b[j]) {

                common++;

                break;
            }
        }
    }


    int unique = n + m - common;


    cout << common << endl;

    cout << unique << endl;




    return 0;
}
```

Task 7 - Practice# programming: Class Practice Task

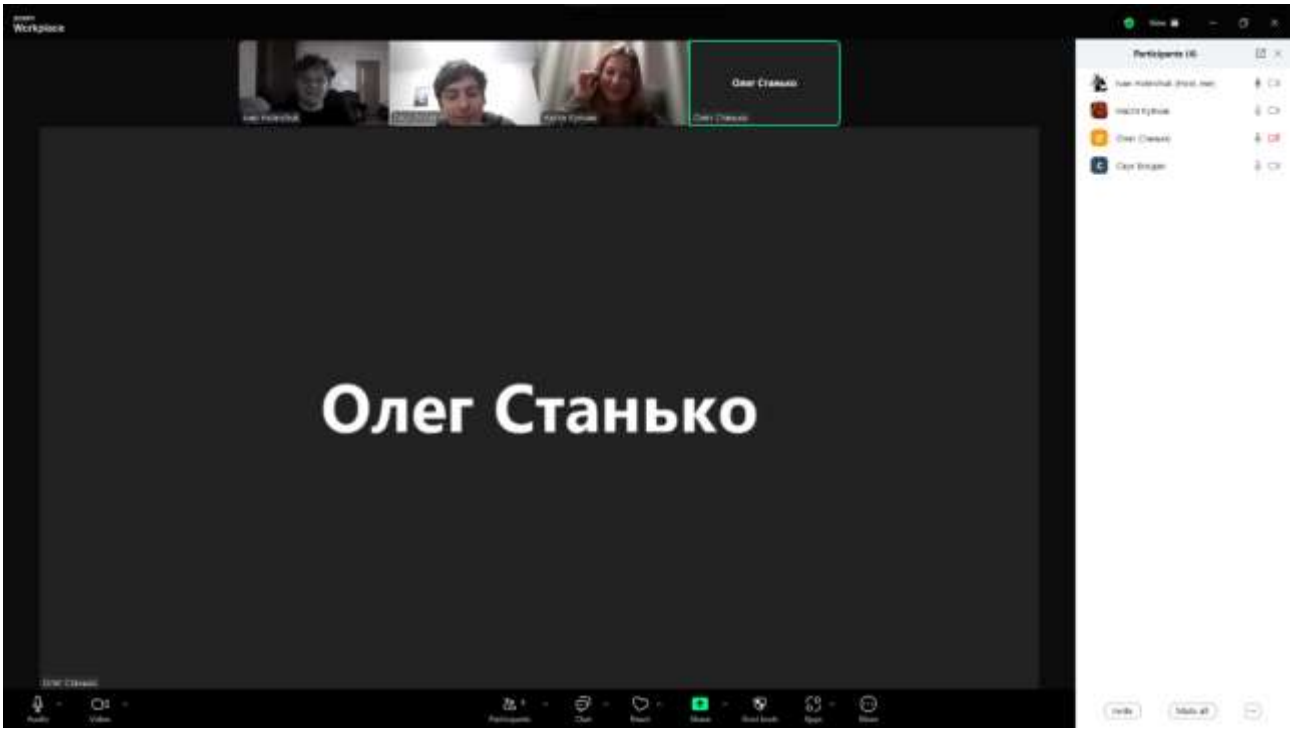
```cpp
#include <iostream>
#include <string>
#include <cmath>

using namespace std;


bool is_palindrome(const string& str, int start, int end) {

    if (start >= end) {

        return true;
    }
    if (str[start] != str[end]) {

        return false;

    }

    return is_palindrome(str, start + 1, end - 1);
}


bool is_palindrome(int num) {

    if (num < 0) return false;

    int original = num;

    int reversed = 0;


    while (num > 0) {

        int digit = num % 10;

        reversed = reversed * 10 + digit;

        num /= 10;
    }


    return original == reversed;
}
int main() {

    string str;

    cout << "Enter a string: ";

    cin >> str;

    if (is_palindrome(str, 0, str.length() - 1)) {


        cout << "The string is a palindrome." << endl;
    } else {

        cout << "The string is not a palindrome." << endl;
    }


    int num;

    cout << "Enter a number: ";

    cin >> num;


    if (is_palindrome(num)) {

        cout << "The number is a palindrome." << endl;

    } else {

        cout << "The number is not a palindrome." << endl;

    }


    return 0;
}

```

Task 8 - Practice# programming: Self Practice Task

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

using namespace std;

const int SIZE = 5;

const int MINES = 3;


struct Cell {

    bool mine;

    bool revealed;

    int adjMines;

};

void initfield(Cell field[SIZE][SIZE]) {

    srand(time(0));


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

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

            field[i][j].mine = false;

            field[i][j].revealed = false;

            field[i][j].adjMines = 0;
        }
    }


    int minesPlaced = 0;

    while (minesPlaced < MINES) {

        int x = rand() % SIZE;

        int y = rand() % SIZE;

        if (!field[x][y].mine) {

            field[x][y].mine = true;

            minesPlaced++;
        }
    }


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

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

            if (field[i][j].mine) continue;

            int count = 0;

            for (int dx = -1; dx <= 1; dx++) {

                for (int dy = -1; dy <= 1; dy++) {

                    int ni = i + dx;

                    int nj = j + dy;

                    if (ni >= 0 && ni < SIZE && nj >= 0 && nj < SIZE && field[ni][nj].mine) {
```

```cpp
void initfield(Cell field[SIZE][SIZE]) {

}

void showfield(Cell field[SIZE][SIZE]) {

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

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

            if (field[i][j].revealed) {

                if (field[i][j].mine) {

                    cout << "* ";
                }else {

                    cout << field[i][j].adjMines << " ";
                }
            } else {

                cout << "# ";
            }
        }
        cout << endl;
    }
}

bool opencell(Cell field[SIZE][SIZE], int x, int y) {

    if (field[x][y].revealed) return false;

    field[x][y].revealed = true;

    if (field[x][y].mine) {

        cout << "BOOM! You hit a mine!" << endl;

        return true;
    }

    return false;
}

int main() {

    Cell field[SIZE][SIZE];

    initfield(field);

    bool gameOver = false;

    while (!gameOver) {

        showfield(field);

        int x, y;

        cout << "Enter row and column (0 to " << SIZE - 1 << "): ";

        cin >> x >> y;

        if (x < 0 || x >= SIZE || y < 0 || y >= SIZE) {

            cout << "Invalid input. Try again." << endl;

            continue;
        }


        gameOver = opencell(field, x, y);
    }
```

Робота у команді:



З командою зібрались лиш один раз, обговорили деталі 4 епіку, та домовились зустрітись, якщо в когось будуть якісь питання.

Висновок: У процесі роботи було опрацьовано основи роботи з масивами, вказівниками, посиланнями та структурами даних. Розглянуто способи зберігання й обробки інформації за допомогою одновимірних і двовимірних масивів, динамічне керування пам’яттю, а також створення й використання структур та вкладених структур. Реалізовано алгоритми для обробки масивів і структур, що забезпечують ефективне вирішення практичних задач.