Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту

A blue and white logo

Description automatically generated

**Звіт**

**Звіт**

**про виконання лабораторних та практичних робіт блоку № 6**

На тему: «Динамічні структури (Черга, Стек, Списки, Дерево). Алгоритми обробки динамічних структур.»

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

до:

ВНС Лабораторної Роботи № 10

Алготестер Лабораторної Роботи № 5

Алготестер Лабораторної Роботи № 7-8

Практичних Робіт № 6

**Виконав:**

Студент групи ШІ-13

Юнко Дмитро Богданович

# **Тема роботи:**

# Робота із простими структурами даних та алгоритмами обробки. Робота з одновимірними та двовимірними масивами.

# **Мета роботи:**

# Мета полягає в освоєнні простих структур даних та алгритмів обробки масивів з використанням відповідних функцій. Навчитися передавати Символи і Рядкові Змінні та Текстові Файли

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

- Тема №1: Динамічний масив.

- Тема №2: Однозв’язний та двозв’язний списки.

- Тема №3: Бінарне дерево

- Тема №4: ООП

- Тема №5: BFS and DFS

1. Індивідуальний план опрацювання теорії:

Тема №1: Динамічний масив.

* + Джерела Інформації
    - Стаття: [How do Dynamic arrays work? - GeeksforGeeks](https://www.geeksforgeeks.org/how-do-dynamic-arrays-work/)
    - Стаття: [C++ Dynamic Allocation of Arrays with Example (guru99.com)](https://www.guru99.com/cpp-dynamic-array.html)
    - Курс: Beginning C++ Programming - From Beginner to Beyond
  + Що опрацьовано:

▪ Вивчення Специфікації Динамічного Масиву:

▪ Ознайомлення з основними характеристиками динамічного масиву.

▪ Процес його алокації та реалокації в пам'яті за потреби.

▪ Операції Читання та Запису:

▪ Освоєння процесу доступу та зміни даних в динамічному масиві.

▪ Розуміння індексування елементів в масиві та його впливу на пам'ять.

▪ Cпособи зміни розміру динамічного масиву.

▪ Аналіз ситуацій, коли потрібно збільшити чи зменшити розмір масиву.

▪ Керування Пам'яттю

▪ Розуміння необхідності вивільнення пам'яті після використання.

* + Статус: Ознайомлений
  + Початок опрацювання теми: 10.12.2023
  + Звершення опрацювання теми: 10.12.2023
* Тема №2: Однозв’язний та двозв’язний списки
  + Джерела Інформації:
    - Стаття: [Introduction to Doubly Linked List – Data Structure and Algorithm Tutorials - GeeksforGeeks](https://www.geeksforgeeks.org/data-structures/linked-list/doubly-linked-list/)
    - Стаття: [Difference between Singly linked list and Doubly linked list - GeeksforGeeks](https://www.geeksforgeeks.org/difference-between-singly-linked-list-and-doubly-linked-list/)
    - Курс: Beginning C++ Programming - From Beginner to Beyond
  + Що опрацьовано:

▪ Однозв’язний Список:

▪ Структура вузла - Кожен елемент однозв’язного списку складається з вузла, який має дані та вказівник на наступний вузол у списку.

▪ Вставка - Можна додати новий вузол на початок або на кінець списку.

▪ - Можна додати новий вузол у середину списку на задану позицію.

▪ Видалення - Можна видалити вузол з початку, кінця або з середини списку.

▪ Пошук - Можна знайти вузол за його значенням або за його порядковим номером.

▪ Перебір - Можна пройтися по всіх вузлах списку для виконання різних операцій.

▪ Двозв’язний Список:

▪ Структура вузла - Кожен елемент двозв’язного списку складається з вузла, який має дані та вказівники на попередній та наступний вузли у списку.

▪ Вставка - Можна додати новий вузол на початок або на кінець списку.

▪ - Можна додати новий вузол у середину списку на задану позицію.

▪ Видалення - Можна видалити вузол з початку, кінця або з середини списку.

▪ Пошук - Можна знайти вузол за його значенням або за його порядковим номером.

▪ Перебір - Можна пройтися по всіх вузлах списку для виконання різних операцій.

* + Статус: Ознайомлений
  + Початок опрацювання теми: 01.12.2023
  + Звершення опрацювання теми: 01.12.2023
* Тема №3: Бінарне дерево
  + Джерела Інформації:
    - Стаття: [Reading and writing binary file in C++ - CodeSpeedy](https://www.codespeedy.com/reading-and-writing-binary-file-in-cpp/)
    - Стаття: [An Introduction to Object Serialization in C++ | CodeGuru](https://www.codeguru.com/cplusplus/an-introduction-to-object-serialization-in-c/)
    - Beginning C++ Programming - From Beginner to Beyond
  + Що опрацьовано:

▪ Структура вузла - Кожен вузол бінарного дерева має дані та два вказівники на своїх нащадків (лівого та правого).

▪ Операції на бінарному дереві - Можна виконувати різні операції на бінарному дереві, такі як вставка, видалення, пошук, перебір, сортування тощо.

▪ Властивості BST - BST (бінарне дерево пошуку) - це особливий вид бінарного дерева, в якому для кожного вузла всі значення в його лівому піддереві менші за його значення, а всі значення в його правому піддереві більші.

▪ Алгоритми для бінарного дерева - Можна застосовувати різні алгоритми для розв’язання задач, пов’язаних з бінарними деревами, таких як обробка арифметичних виразів, побудова індексів для баз даних, оптимізація коду для компіляторів, візуалізація графічних об’єктів тощо.

▪ Реалізація та оптимізація - Можна створювати власні класи або структури для представлення бінарного дерева в коді, а також оптимізувати операції та алгоритми для підвищення ефективності та швидкодії.

▪ Розуміння можливих проблем та виправлення помилок - Можна виявляти та усувати різні проблеми, які можуть виникати при роботі з бінарними деревами, такі як неправильна алокація пам’яті, небалансованість дерева, некоректність даних.

* + Статус: Ознайомлений
  + Початок опрацювання теми: 01.12.2023
  + Звершення опрацювання теми: 01.12.2023

Тема №4: ООП

* + Джерела Інформації:
    - Стаття: [OOPs (Object-Oriented Programming) in C++ - javatpoint](https://www.javatpoint.com/cpp-oops-concepts)
    - Курс: Beginning C++ Programming - From Beginner to Beyond
  + Що опрацьовано:
    - Що таке класи та об’єкти в C++, як їх оголошувати, ініціалізувати, використовувати та знищувати. Ознайомитися з конструкторами, деструкторами, модифікаторами доступу, членами класу, оператором крапка та іншими основними поняттями
    - Що таке інкапсуляція в C++, як вона дозволяє приховувати внутрішню реалізацію об’єкта від зовнішнього світу. Ознайомитися з різними рівнями доступу, такими як public, private та protected, та їх впливом на членів клас
    - Що таке спадкування в C++, як воно дозволяє створювати нові класи на основі існуючих, унаслідувавши їх властивості та методи. Ознайомитися з різними типами спадкування, такими як однорідне, багаторідне, багатоуровневе, та їх синтаксисом
    - Що таке Абстракція в C++, як вона дозволяє показувати тільки необхідні атрибути класу, приховуючи технічні деталі від користувача. Ознайомився з різними способами досягнення абстракції, такими як використання віртуальних функцій, абстрактних класів, інтерфейсів, чистих віртуальних функцій та іншими
  + Статус: Ознайомлений
  + Початок опрацювання теми: 29.11.2023
  + Звершення опрацювання теми: 29.11.2023

Тема №5: DFS and BFS

* + Джерела Інформації:
    - Стаття: [Difference between BFS and DFS - GeeksforGeeks](https://www.geeksforgeeks.org/difference-between-bfs-and-dfs/)
    - Курс: Beginning C++ Programming - From Beginner to Beyond
  + Що опрацьовано:
    - Що таке граф, які його види, як його представляти в коді, які його властивості та термінологія. Ознайомлення з різними способами зберігання графів, такими як матриця суміжності, список суміжності, список ребер тощо
    - Обхід графів за шириною BFS, як він працює, який його алгоритм. Ознайомитися з різними застосуваннями BFS, такими як пошук найкоротшого шляху, перевірка зв’язності графа, рівнева траверсалія дерева тощо. Навчитися реалізовувати BFS в C++ за допомогою черги та масиву відвіданих верши
    - Обхід графів за глибиною DFS як він працює, який його алгоритм. Ознайомитися з різними застосуваннями DFS, такими як пошук циклів, топологічне сортування, класифікація ребер, пошук компонентів сильної зв’язності тощо. Навчитися реалізовувати DFS в C++ за допомогою стеку та масиву відвіданих вершин
    - Порівняння BFS та DFS: Вивчити, які переваги та недоліки мають BFS та DFS, які їх відмінності та схожості, коли краще використовувати кожен з них. Ознайомитися з різними прикладами та задачами, які можна розв’язати за допомогою BFS або DFS
  + Статус: Ознайомлений
  + Початок опрацювання теми: 29.11.2023
  + Звершення опрацювання теми: 29.11.2023

# **Виконання роботи:**

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

**Завдання №1 Опрацювання теорії**

* Деталі завдання: опрацювання різноманітних матеріалів, аналіз та дослідження відео, статей, книг на задані теми. Організація та структурування отриманих даних для можливості ефективніше засвоювати отримані знання на практиці. Вивчення найважливіших моментів.

**Завдання №2 Проектування за допомогою Draw.io(складання блок-схем)**

* Деталі завдання:

Побудова блок-схем до завдань 3-9.

#### Завдання №3 Algotester Lab 5

* Варіант завдання: **3**
* Деталі завдання: У вас є карта гори розміром N×M.

Також ви знаєте координати {x,y}, у яких знаходиться вершина гори.

Ваше завдання - розмалювати карту таким чином, щоб найнижча точка мала число 0, а пік гори мав найбільше число.

Клітинкі які мають суміжну сторону з вершиною мають висоту на один меншу, суміжні з ними і не розфарбовані мають ще на 1 меншу висоту і так далі.

#### Завдання №4 Algotester Lab 78

* Варіант завдання: **1**

Деталі завдання:

власноруч реалізувати структуру даних "Двозв’язний список".

#### Завдання №5 VNS Lab 10 - Task 1

Варіант завдання: 1

* Деталі завдання: :

Написати програму, у якій створюються динамічні структури й виконати

їхню обробку у відповідності зі своїм варіантом.  
Записи в лінійному списку містять ключове поле типу рядок

символів. Сформувати двонаправлений список. Знищити елемент із заданим

ключем. Додати по К елементів на початок й в кінець списку.

#### Завдання №6 Class Practice 1

* Варіант завдання: -
* Деталі завдання:

  використовувати цілочисельні значення в списку;

реалізувати метод реверсу;

реалізувати допоміжний метод виведення вхідного і обернутого списків;

#### Завдання №7 Class Practice 2

* Варіант завдання: **1**
* Деталі завдання:

- використовувати цілочисельні значення в списку;

- реалізувати функцію, яка ітеративно проходиться по обох списках і порівнює дані в кожному вузлі;

- якщо виявлено невідповідність даних або якщо довжина списків різна (один список закінчується раніше іншого), функція повертає false.

#### Завдання №8 Class Practice 3

* Деталі завдання:

використовувати цифри від 0 до 9 для значень у списку;

реалізувати функцію, яка обчислює суму двох чисел, які збережено в списку; молодший розряд числа записано в голові списка (напр. 379 ⟹ 9→7→3);

функція повертає новий список, передані в функцію списки не модифікуються.

#### Завдання №9 Class Practice 4

* Деталі завдання:

      використовувати цілі числа для значень у вузлах дерева

реалізувати функцію, що проходить по всіх вузлах дерева і міняє місцями праву і ліву вітки дерева. функція повертає нове дерево, передане в функцію дерево не модифікуєтьсяa

#### Завдання №10 Class Practice 5

* Деталі завдання:

використовувати цілочисельні значення у вузлах дерева;

реалізувати функцію, яка ітеративно проходить по бінарному дереві і записує у батьківський вузол суму значень підвузлів

вузол-листок не змінює значення

значення змінюються від листків до кореня дерева

#### Завдання №11 Self Practice

* Деталі завдання:  реалізація програми, яка розбиватиме текстовий файл на чанки.

#### Завдання №12 Self Practice

* Деталі завдання:  покращення коду практичної з 3 епіку, з використанням свіжо здобутих знаннь про роботу з файлами

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

#### Програма № 3 Algotester Lab 5

* Блок-схема
* Планований час на реалізацію: 60хв

використання бібліотеки cmath

#### Програма №4 Algotester Lab 78

* Планований час на реалізацію: 120хв

#### Програма №5 VNS Lab 10 - Task 1

* Планований час на реалізацію 80хв

#### Програма №6 Class Practice

* Планований час на реалізацію 45 хв

#### Програма №7 Class Practice 2

* Планований час на реалізацію 30 хв

#### Програма №8 Class Practice 3

* Планований час на реалізацію 80 хв

#### Програма №9 Class Practice 4

* Планований час на реалізацію 60 хв

#### Програма №10 Class Practice 5

* Планований час на реалізацію 60 хв

#### Програма №11 Self Practice | Hide and Seek

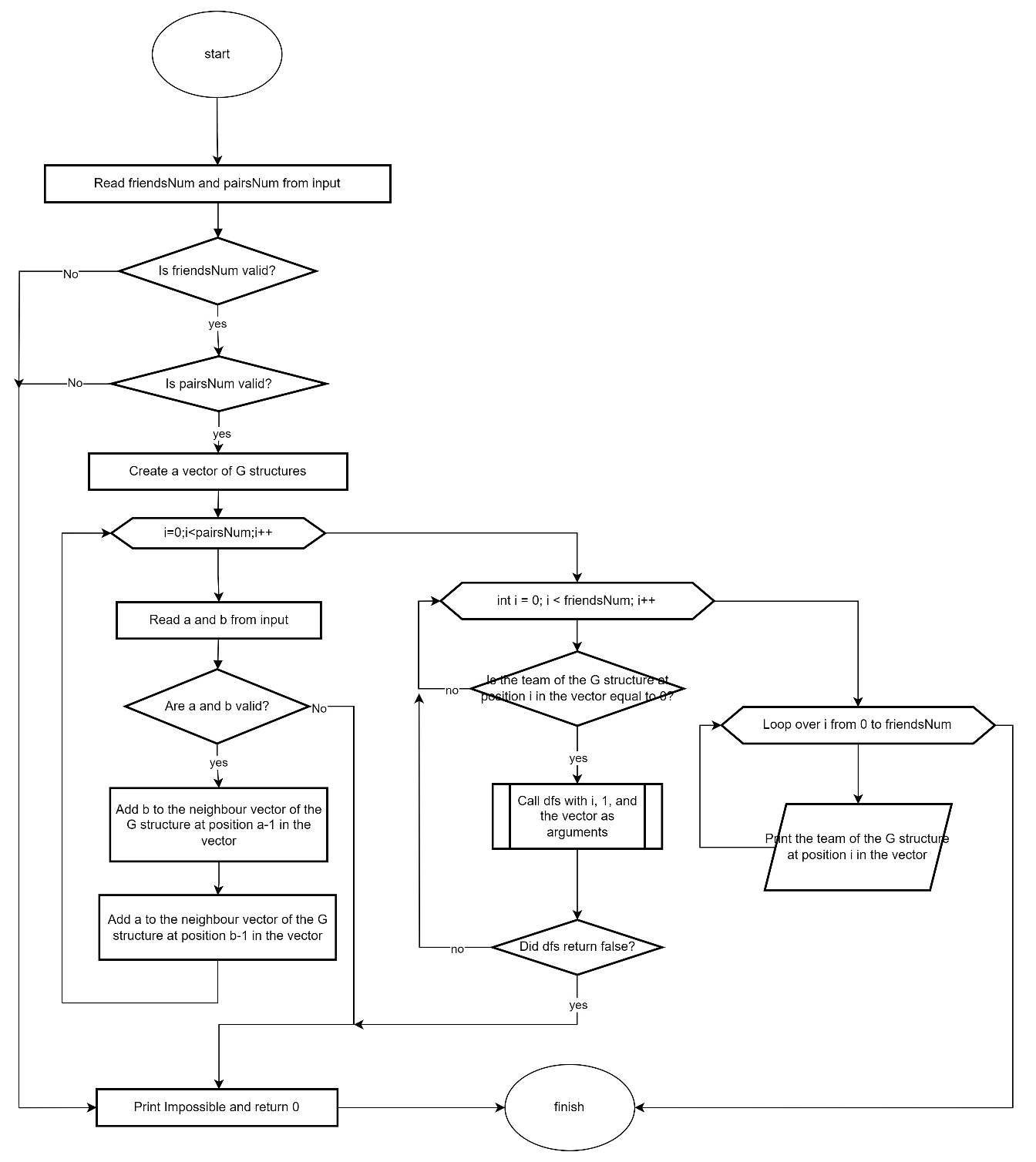


Рисунок 1 Self Practice

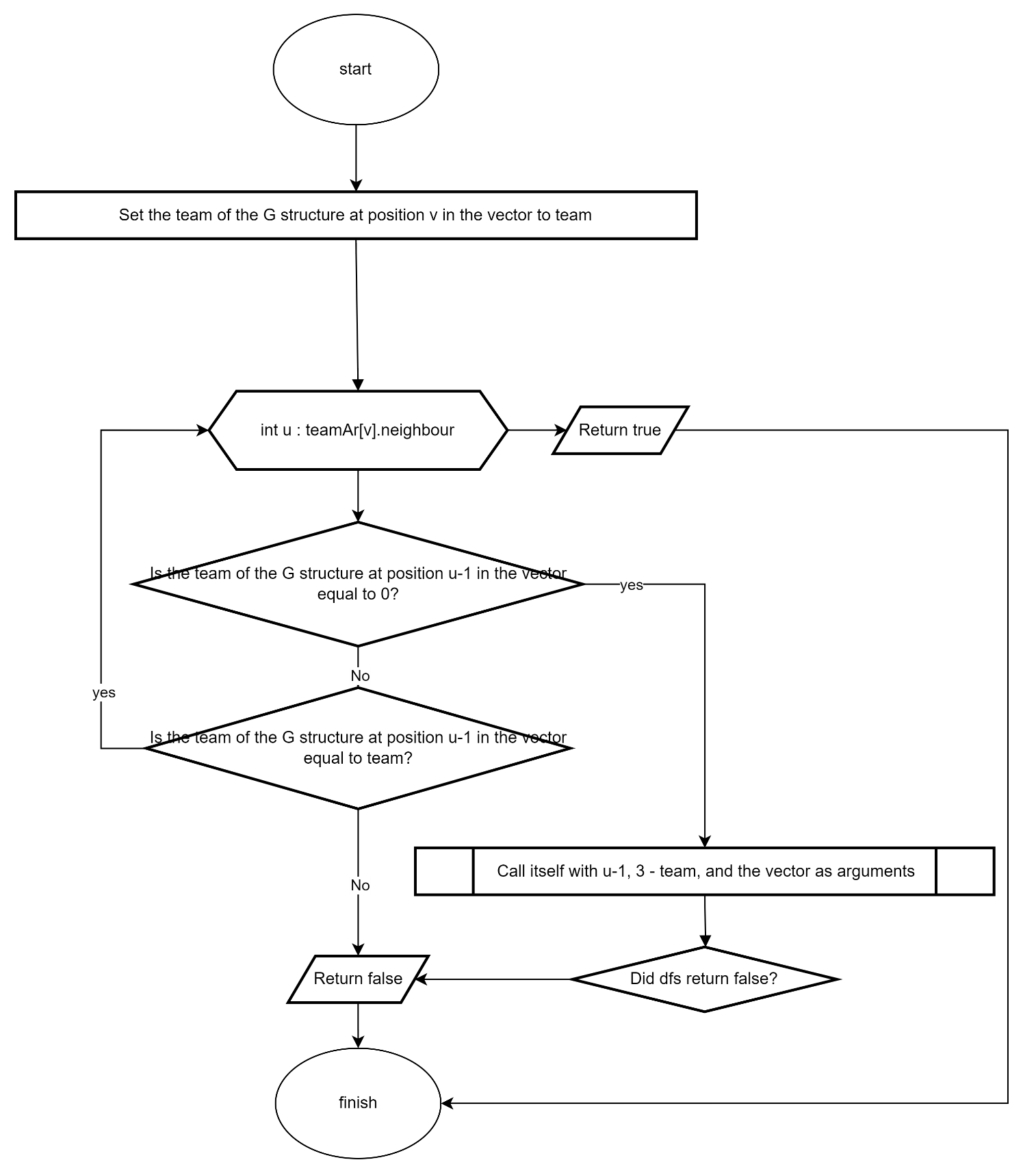


Рисунок 2 Self Practice dfs function

* Планований час на реалізацію: 60хв

## **4. Код програм з посиланням на зовнішні ресурси:** [https://github.com/artificial-intelligence-department/ai\_programming\_playground/pull/230/files#diff-097297c805c8250774272aa6616b4bbbdd5bd20fde5f2b0cf8b246d98169163d](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/230/files%23diff-097297c805c8250774272aa6616b4bbbdd5bd20fde5f2b0cf8b246d98169163d)

#include <iostream>

#include <vector>

#include <queue>

using namespace std;

void setHeight(int x, int y, int N, int M, vector<vector<int>>& ar, vector<vector<bool>>& visited) {

    queue<pair<int, int>> q;

    q.push({x, y});

    visited[x][y] = true;

    ar[x][y] = 0;

    int xDir[] = {-1, 0, 1, 0};

    int yDir[] = {0, 1, 0, -1};

    while (!q.empty()) {

        pair<int, int> current = q.front();

        q.pop();

        for (int i = 0; i < 4; i++) {

            int xi = current.first + xDir[i];

            int yi = current.second + yDir[i];

            if (xi >= 0 && xi < N && yi >= 0 && yi < M && !visited[xi][yi]) {

                ar[xi][yi] = ar[current.first][current.second] + 1;

                visited[xi][yi] = true;

                q.push({xi, yi});

            }

        }

    }

}

int main() {

    int N, M, x, y;

    cin >> N >> M >> x >> y;

    x--; y--;

    vector<vector<int>> mntn(N, vector<int>(M, 0));

    vector<vector<bool>> visited(N, vector<bool>(M, false));

    setHeight(x, y, N, M, mntn, visited);

    int maxHeight = 0;

    for (int i = 0; i < N; i++) {

        for (int j = 0; j < M; j++) {

            maxHeight = max(maxHeight, mntn[i][j]);

        }

    }

    for (int i = 0; i < N; i++) {

        for (int j = 0; j < M; j++) {

            cout << maxHeight - mntn[i][j] << " ";

        }

        cout << endl;

    }

    return 0;

}

###### Код до програми № 3 Algotester Lab 5

#include <iostream>

#include <vector>

#include <sstream>

using namespace std;

struct Node

{

    int value;

    Node \*next;

    Node \*prev;

};

class DoublyLinkedList

{

private:

    Node \*head;

    Node \*end;

    int size;

public:

    DoublyLinkedList()

    {

        head = nullptr;

        end = nullptr;

        size = 0;

    }

    ~DoublyLinkedList()

    {

        Node \*temp;

        while (head != nullptr)

        {

            temp = head;

            head = head->next;

            delete temp;

        }

    }

   void insert(int index, std::vector<int>& list) {

    Node\* current = head;

    Node\* prevNode = nullptr;

    for (int i = 0; i < index; ++i) {

        prevNode = current;

        current = current->next;

    }

    for (int i = 0; i < list.size(); ++i) {

        Node\* newNode = new Node;

        newNode->value = list[i];

        newNode->prev = prevNode;

        newNode->next = current;

        if (prevNode) {

            prevNode->next = newNode;

        } else {

            head = newNode;

        }

        if (current) {

            current->prev = newNode;

        } else {

            end = newNode;

        }

        prevNode = newNode;

        ++size;

    }

}

    void erase(int index, int n) {

    Node\* current = head;

    Node\* prevNode = nullptr;

    for (int i = 0; i < index; ++i) {

        prevNode = current;

        current = current->next;

    }

    for (int i = 0; i < n; ++i) {

        if (current == nullptr) {

            break;

        }

        Node\* nextNode = current->next;

        if (prevNode) {

            prevNode->next = nextNode;

        } else {

            head = nextNode;

        }

        if (nextNode) {

            nextNode->prev = prevNode;

        } else {

            end = prevNode;

        }

        delete current;

        current = nextNode;

        --size;

    }

}

    int getSize()

    {

        return size;

    }

    int get(int index)

    {

        Node \*current = head;

        for (int i = 0; i < index; i++)

        {

            current = current->next;

        }

        return current->value;

    }

    void set(int index, int value)

    {

        Node \*current = head;

        for (int i = 0; i < index; i++)

        {

            current = current->next;

        }

        current->value = value;

    }

    friend std::ostream &operator<<(std::ostream &os, const DoublyLinkedList &list)

    {

        Node \*current = list.head;

        while (current != nullptr)

        {

            os << current->value << ' ';

            current = current->next;

        }

        return os;

    }

};

int main()

{

    DoublyLinkedList list;

    int Q;

    cin >> Q;

    cin.ignore();

    string identifiers[Q];

    for (size\_t i = 0; i < Q; i++)

    {

        cin >> identifiers[i];

        if (identifiers[i] == "insert")

        {

            int index{0}, N{0};

            vector<int> ar;

            cin >> index >> N;

            for (size\_t i = 0; i < N; i++)

            {

                int a;

                cin >> a;

                ar.push\_back(a);

            }

            list.insert(index, ar);

        }

        if (identifiers[i] == "erase")

        {

            int index{0}, n{0};

            cin >> index >> n;

            list.erase(index, n);

        }

        if (identifiers[i] == "size")

        {

            cout << list.getSize() << endl;

        }

        if (identifiers[i] == "get")

        {

            int index{0};

            cin >> index;

            cout<<list.get(index)<<endl;

        }

        if (identifiers[i] == "set")

        {

            int index{0}, newValue{0};

            cin >> index >> newValue;

            list.set(index, newValue);

        }

        if (identifiers[i] == "print")

        {

            cout << list << endl;

        }

    }

    return 0;

}

###### Код до програми № 4 Algotester Lab 78 [https://github.com/artificial-intelligence department/ai\_programming\_playground/pull/494](https://github.com/artificial-intelligence%20department/ai_programming_playground/pull/494)

#include <iostream>

#include <fstream>

#include <cstdlib>

#include <ctime>

#include <string>

using namespace std;

struct Node

{

    char\* data;

    Node \*next;

    Node \*prev;

};

class DoublyLinkedList

{

private:

    struct Node

    {

        string data;

        Node \*prev;

        Node \*next;

        Node(string data)

        {

            this->data = data;

            this->prev = nullptr;

            this->next = nullptr;

        }

    };

    Node \*head;

    Node \*end;

public:

    DoublyLinkedList()

    {

        head = nullptr;

        end = nullptr;

    }

    ~DoublyLinkedList()

    {

        deleteList();

    }

    void printList()

    {

        Node \*current = head;

        if (current == nullptr)

        {

            cout << "List is empty" << endl;

        }

        else

        {

            while (current != nullptr)

            {

                cout << current->data << " ";

                current = current->next;

            }

            cout << endl;

        }

    }

    void insert(string data)

    {

        Node \*newNode = new Node(data);

        if (head == nullptr)

        {

            head = newNode;

            end = newNode;

        }

        else

        {

            end->next = newNode;

            newNode->prev = end;

            end = newNode;

        }

    }

    void writeToFile(const string &filename)

    {

        ofstream outFile(filename);

        if (!outFile)

        {

            cerr << "Unable to open the file: " << filename << endl;

            return;

        }

        Node \*current = head;

        while (current != nullptr)

        {

            outFile << current->data << " ";

            current = current->next;

        }

        outFile.close();

    }

    DoublyLinkedList restoreFromFile(const string &filename)

    {

        ifstream inFile(filename);

        if (!inFile)

        {

            cerr << "Unable to open the file: " << filename << endl;

            return DoublyLinkedList();

        }

        DoublyLinkedList list;

        string data;

        if (inFile.peek() == EOF)

        {

            cout << "The file is empty." << endl;

            return list;

        }

        while (inFile >> data)

        {

            list.insert(data);

        }

        inFile.close();

        return list;

    }

    void deleteList()

    {

        Node \*current = head;

        while (current != nullptr)

        {

            Node \*temp = current;

            current = current->next;

            delete temp;

        }

        head = nullptr;

        end = nullptr;

    }

    void deleteNode(string key)

    {

        Node \*current = head;

        while (current != nullptr && current->data != key)

        {

            current = current->next;

        }

        if (current == nullptr)

        {

            cout << "Key not found in the list." << endl;

            return;

        }

        if (current->prev == nullptr)

        {

            head = current->next;

            if (head != nullptr)

            {

                head->prev = nullptr;

            }

            else

            {

                end = nullptr;

            }

        }

        else if (current->next == nullptr)

        {

            end = current->prev;

            if (end != nullptr)

            {

                end->next = nullptr;

            }

            else

            {

                head = nullptr;

            }

        }

        else

        {

            current->prev->next = current->next;

            current->next->prev = current->prev;

        }

        delete current;

    }

    void addKNodes(int K)

    {

        srand(time(0));

        string alphabet = "abcdefghijklmnopqrstuvwxyz";

        int len = alphabet.length();

        for (int i = 0; i < K; i++)

        {

            string data;

            int n = rand() % 10 + 1;

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

            {

                data += alphabet[rand() % len];

            }

            insert(data);

        }

        Node \*temp = end;

        for (int i = 0; i < K; i++)

        {

            string data;

            int n = rand() % 10 + 1;

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

            {

                data += alphabet[rand() % len];

            }

            insert(data);

        }

    }

};

int main()

{

    int K;

    char\* key = new char[20];

    cout<<"Enter the number of nodes to add at the beginning and at the end: ";

    cin>>K;

    cout<<"Enter the key to delete: ";

    cin>>key;

    DoublyLinkedList List;

    List.insert("banana");

    List.insert("apple");

    List.printList();

    List.deleteNode( key);

    List.printList();

    List.addKNodes(K);

    List.printList();

    List.writeToFile("test.txt");

    List.deleteList();

    List.printList();

    DoublyLinkedList newList = List.restoreFromFile("test.txt");

    newList.printList();

    return 0;

}

###### Код до програми № 5 VNS Lab 10 - Task 1

#include <iostream>

using namespace std;

namespace list

{

    class List;

    struct Node

    {

        int data;

        Node \*next;

        Node(int value) : data(value), next(nullptr) {}

        friend class List;

    };

    class List

    {

    private:

        Node \*head;

    public:

        List() : head(nullptr) {}

        ~List()

        {

            clear();

        }

        void append(int value)

        {

            Node \*newNode = new Node(value);

            if (head == nullptr)

            {

                head = newNode;

            }

            else

            {

                Node \*temp = head;

                while (temp->next != nullptr)

                {

                    temp = temp->next;

                }

                temp->next = newNode;

            }

        }

        void reverse()

        {

                Node\* prev = nullptr;

        Node\* current = head;

        Node\* next = nullptr;

        while (current != nullptr) {

            next = current->next;

            current->next = prev;

            prev = current;

            current = next;

        }

        head=prev;

        }

        void clear()

        {

            Node \*current = head;

            while (current != nullptr)

            {

                Node \*temp = current;

                current = current->next;

                delete temp;

            }

            head = nullptr;

        }

        friend ostream &operator<<(ostream &os, const List &list)

        {

            Node \*current = list.head;

            if (current == nullptr)

            {

                os << "List is empty" << endl;

            }

            else

            {

                while (current != nullptr)

                {

                    os << current->data << " ";

                    current = current->next;

                }

                os << endl;

            }

            return os;

        }

    };

}

int main()

{

    using namespace list;

    int n;

    do

    {

        cout << "Enter amount of nodes: ";

        cin >> n;

        if (n <= 0)

        {

            cout << "Amount of nodes must be over zero." << endl;

        }

    } while (n <= 0);

    cout << "Enter values: ";

    List list;

    for (int i = 0; i < n; i++)

    {

        int value;

        cin >> value;

        list.append(value);

    }

    cout << "Original list: " << list;

    list.reverse();

    cout << "Reversed list: "<<list  ;

    return 0;

}

###### Код до програми № 6 Class Practice

#include <iostream>

using namespace std;

class Node {

    public:

        int data;

        Node\* next;

        Node(int data) {

            this->data = data;

            this->next = nullptr;

        }

};

class LinkedList {

    private:

        Node\* head;

    public:

        LinkedList() {

            head = nullptr;

        }

        void insert(int data) {

            Node\* newNode = new Node(data);

            newNode->next = head;

            head = newNode;

        }

        bool compare(LinkedList& other) {

            Node\* h1 = head;

            Node\* h2 = other.head;

            while (h1 != nullptr && h2 != nullptr) {

                if (h1->data != h2->data) {

                    return false;

                }

                h1 = h1->next;

                h2 = h2->next;

            }

            return h1 == nullptr && h2 == nullptr;

        }

};

int main() {

    LinkedList list1;

    LinkedList list2;

    list1.insert(4);

    list1.insert(4);

    list1.insert(8);

    list1.insert(10);

    list2.insert(4);

    list2.insert(4);

    list2.insert(8);

    list2.insert(10);

    bool result = list1.compare(list2);

    if (result) {

        cout << "Lists are equal"<<endl;

    } else {

        cout << "Lists are not equal"<<endl;

    }

    return 0;

}

###### Код до програми № 7 Class Practice 2

#include <iostream>

#include <stack>

using namespace std;

class Node {

    public:

        int data;

        Node\* next;

        Node(int data) {

            this->data = data;

            this->next = nullptr;

        }

};

class LinkedList {

    private:

        Node\* head;

    public:

        LinkedList() {

            head = nullptr;

        }

        void insert(int data) {

            Node\* newNode = new Node(data);

            newNode->next = head;

            head = newNode;

        }

        LinkedList add(LinkedList& other) {

            LinkedList result;

            Node\* head\_1 = head;

            Node\* head\_2 = other.head;

            int carry = 0;

            while (head\_1 != nullptr || head\_2 != nullptr || carry != 0) {

                int sum = carry;

                if (head\_1 != nullptr) {

                    sum += head\_1->data;

                    head\_1 = head\_1->next;

                }

                if (head\_2 != nullptr) {

                    sum += head\_2->data;

                    head\_2 = head\_2->next;

                }

                int digit = sum % 10;

                carry = sum / 10;

                result.insert(digit);

            }

            return result;

        }

        void printList() {

            Node\* current = head;

            while (current != nullptr) {

                cout << current->data << " ";

                current = current->next;

            }

            cout << endl;

        }

        void printReverse() {

            stack<int> s;

            Node\* current = head;

            while (current != nullptr) {

                s.push(current->data);

                current = current->next;

            }

            while (!s.empty()) {

                cout << s.top() << " ";

                s.pop();

            }

            cout << endl;

        }

};

int main() {

    LinkedList num1;

    LinkedList num2;

    num1.insert(2);

    num1.insert(0);

    num1.insert(0);

    num2.insert(1);

    num2.insert(6);

    num2.insert(8);

    LinkedList result = num1.add(num2);

    cout << "Number 1: ";

    num1.printReverse();

    cout << "Number 2: ";

    num2.printReverse();

    cout << "Sum: ";

    result.printList();

    return 0;

}

###### Код до програми № 8 Class Practice 3

#include <iostream>

#include <queue>

using namespace std;

class TreeNode {

    public:

        int data;

        TreeNode\* left;

        TreeNode\* right;

        TreeNode(int data) {

            this->data = data;

            this->left = nullptr;

            this->right = nullptr;

        }

};

class BinaryTree {

    private:

        TreeNode\* root;

    public:

        BinaryTree() {

            root = nullptr;

        }

        void insert(int data) {

            insert(root, data);

        }

        void insert(TreeNode\*& node, int data) {

            if (node == nullptr) {

                node = new TreeNode(data);

                return;

            }

            if (data < node->data) {

                insert(node->left, data);

            } else {

                insert(node->right, data);

            }

        }

        BinaryTree create\_mirror\_flip() {

            BinaryTree mirroredTree;

            create\_mirror\_flip(root, mirroredTree.root);

            return mirroredTree;

        }

        void create\_mirror\_flip(TreeNode\* node, TreeNode\*& mirroredNode) {

            if (node == nullptr) {

                return;

            }

            mirroredNode = new TreeNode(node->data);

            create\_mirror\_flip(node->left, mirroredNode->right);

            create\_mirror\_flip(node->right, mirroredNode->left);

        }

        void in\_order\_traversal() {

            in\_order\_traversal(root);

            cout << endl;

        }

        void in\_order\_traversal(TreeNode\* node) {

            if (node != nullptr) {

                in\_order\_traversal(node->left);

                cout << node->data << " ";

                in\_order\_traversal(node->right);

            }

        }

        void printTree() {

            if (root == nullptr) {

                return;

            }

            queue<TreeNode\*> q;

            q.push(root);

            while (!q.empty()) {

                int n = q.size();

                while (n > 0) {

                    TreeNode\* node = q.front();

                    q.pop();

                    cout << node->data << " ";

                    if (node->left != nullptr) {

                        q.push(node->left);

                    }

                    if (node->right != nullptr) {

                        q.push(node->right);

                    }

                    n--;

                }

                cout << endl;

            }

        }

};

int main() {

    BinaryTree tree;

    tree.insert(4);

    tree.insert(2);

    tree.insert(6);

    tree.insert(1);

    tree.insert(3);

    tree.insert(5);

    tree.insert(7);

    cout << "Original Tree : ";

    tree.in\_order\_traversal();

    tree.printTree();

    BinaryTree mirroredTree = tree.create\_mirror\_flip();

    cout << "Mirrored Tree : ";

    mirroredTree.in\_order\_traversal();

    mirroredTree.printTree();

    return 0;

}

###### Код до програми № 9 Class Practice 4

#include <iostream>

#include<queue>

using namespace std;

class TreeNode {

    public:

        int data;

        TreeNode\* left;

        TreeNode\* right;

        TreeNode(int data) {

            this->data = data;

            this->left = nullptr;

            this->right = nullptr;

        }

};

class BinaryTree {

    private:

        TreeNode\* root;

    public:

        BinaryTree() {

            root = nullptr;

        }

        void insert(int data) {

            insert(root, data);

        }

        void insert(TreeNode\*& node, int data) {

            if (node == nullptr) {

                node = new TreeNode(data);

                return;

            }

            if (data < node->data) {

                insert(node->left, data);

            } else {

                insert(node->right, data);

            }

        }

        void tree\_sum() {

            tree\_sum(root);

        }

        void tree\_sum(TreeNode\* node) {

            if (node == nullptr || (node->left == nullptr && node->right == nullptr)) {

                return;

            }

            tree\_sum(node->left);

            tree\_sum(node->right);

            node->data = (node->left ? node->left->data : 0) + (node->right ? node->right->data : 0);

        }

        void in\_order\_traversal() {

            in\_order\_traversal(root);

            cout << endl;

        }

        void printTree() {

            if (root == nullptr) {

                return;

            }

            queue<TreeNode\*> q;

            q.push(root);

            while (!q.empty()) {

                int n = q.size();

                while (n > 0) {

                    TreeNode\* node = q.front();

                    q.pop();

                    cout << node->data << " ";

                    if (node->left != nullptr) {

                        q.push(node->left);

                    }

                    if (node->right != nullptr) {

                        q.push(node->right);

                    }

                    n--;

                }

                cout << endl;

            }

        }

        void in\_order\_traversal(TreeNode\* node) {

            if (node != nullptr) {

                in\_order\_traversal(node->left);

                cout << node->data << " ";

                in\_order\_traversal(node->right);

            }

        }

};

int main() {

    BinaryTree tree;

    tree.insert(4);

    tree.insert(2);

    tree.insert(6);

    tree.insert(1);

    tree.insert(3);

    tree.insert(5);

    tree.insert(7);

    cout << "Original Tree : ";

    tree.in\_order\_traversal();

    tree.printTree();

    tree.tree\_sum();

    cout << "Tree After Sum Calculation : ";

    tree.in\_order\_traversal();

    tree.printTree();

    return 0;

}

###### Код до програми № 10 Class Practice 5

#include <iostream>

#include <vector>

using namespace std;

    struct G

    {

        int team;

        vector <int> neighbour;

    };

    bool dfs(int v, int team, vector<G>& teamAr) {

        teamAr[v].team = team;

        for (int u : teamAr[v].neighbour) {

            if (teamAr[u-1].team == 0 && !dfs(u-1, 3 - team, teamAr)) {

                return false;

            } else if (teamAr[u-1].team == team) {

                return false;

            }

        }

        return true;

    }

int main(int argc, char const \*argv[])

{

    int friendsNum {0};

    int pairsNum {0};

    cin >> friendsNum >> pairsNum;

    if(friendsNum < 1 || friendsNum > 100000){

        cout << "Impossible" << endl;

        return 0;

    }

    if(pairsNum < 1 || pairsNum > 100000){

        cout << "Impossible" << endl;

       return 0;

    }

    vector<G> teamAr(friendsNum);

    for(size\_t i=0;i<pairsNum;i++) {

        int a{0};

        int b{0};

        cin >>a >>b;

        if(a==b || a<1||b<1|| a>friendsNum|| b>friendsNum )

        {

          cout << "Impossible" << endl;

                return 0;

        }

        teamAr[a-1].neighbour.push\_back(b);

        teamAr[b-1].neighbour.push\_back(a);

    }

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

        if (teamAr[i].team == 0 && !dfs(i, 1, teamAr)) {

            cout << "Impossible" << endl;

            return 0;

        }

    }

     for(int i=0; i<friendsNum; i++)

            cout << teamAr[i].team;

    return 0;

}

###### Код до програми № 11 Self Practice | Hide and Seek

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

#### Завдання №3 Деталі по виконанню і тестуванню програми

PS D:\ED\University\C++\ai\_programming\_playground\ai\_13\dmytro\_yunko\epic\_6\output> & .\'algotester\_lab\_5\_variant\_3\_dmytro\_yunko.exe'

3 4

2 2

1 2 1 0

2 3 2 1

1 2 1 0

Algotester Lab 5

#### Завдання №4 Деталі по виконанню і тестуванню програми

PS D:\ED\University\C++\ai\_programming\_playground\ai\_13\dmytro\_yunko\epic\_6\output> & .\'algotester\_lab\_7\_8\_variant\_1\_dmytro\_yunko.exe'

5

insert

0 3

1 2 3

erase

0 2

set

0 10

size

1

print

10

Algotester Lab 78

#### Завдання №5 Деталі по виконанню і тестуванню програми

PS D:\ED\University\C++\ai\_programming\_playground\ai\_13\dmytro\_yunko\epic\_6\output> & .\'vns\_lab\_10\_task\_1\_variant\_20\_dmytro\_yunko.exe'

Enter the number of nodes to add at the beginning and at the end: 10

Enter the key to delete: apple

banana apple

banana

banana pbcc scafrn gxrnc nbwt gzaa pipyva nrahzo rupdmedpv xhlk ax pwv tbebod rwn rwk jthv fwdz xinpqc htdwk lafv a

List is empty

banana pbcc scafrn gxrnc nbwt gzaa pipyva nrahzo rupdmedpv xhlk ax pwv tbebod rwn rwk jthv fwdz xinpqc htdwk lafv a

VNS Lab 10 - Task 1

#### Завдання №6 Деталі по виконанню і тестуванню програми

PS D:\ED\University\C++\ai\_programming\_playground\ai\_13\dmytro\_yunko\epic\_6\output> & .\'practice\_work\_task\_1\_dmytro\_yunko.exe'

Enter amount of nodes: 10

Enter values: 1 2 3 4 5 6 7 8 9 10

Original list: 1 2 3 4 5 6 7 8 9 10

Reversed list: 10 9 8 7 6 5 4 3 2 1

Class Practice 1

#### Завдання №7 Деталі по виконанню і тестуванню програми

PS D:\ED\University\C++\ai\_programming\_playground\ai\_13\dmytro\_yunko\epic\_6\output> & .\'practice\_work\_task\_2\_dmytro\_yunko.exe'

Lists are equal

Class Practice 2

#### Завдання №8 Деталі по виконанню і тестуванню програми

PS D:\ED\University\C++\ai\_programming\_playground\ai\_13\dmytro\_yunko\epic\_6\output> & .\'practice\_work\_task\_3\_dmytro\_yunko.exe'

Number 1: 2 0 0

Number 2: 1 6 8

Sum: 3 6 8

Class Practice 3

#### Завдання №9 Деталі по виконанню і тестуванню програми

PS D:\ED\University\C++\ai\_programming\_playground\ai\_13\dmytro\_yunko\epic\_6\output> & .\'practice\_work\_task\_4\_dmytro\_yunko.exe'

Original Tree : 1 2 3 4 5 6 7

4

2 6

1 3 5 7

Mirrored Tree : 7 6 5 4 3 2 1

4

6 2

7 5 3 1

Class Practice 4

#### Завдання №10 Деталі по виконанню і тестуванню програми

PS D:\ED\University\C++\ai\_programming\_playground\ai\_13\dmytro\_yunko\epic\_6\output> & .\'practice\_work\_task\_5\_dmytro\_yunko.exe'

Original Tree : 1 2 3 4 5 6 7

4

2 6

1 3 5 7

Tree After Sum Calculations: 1 4 3 16 5 12 7

16

4 12

1 3 5 7

Class Practice 5

#### Завдання №11 Деталі по виконанню і тестуванню програми

PS D:\ED\University\C++\ai\_programming\_playground\ai\_13\dmytro\_yunko\epic\_6\output> & .\'self\_practice\_work\_algotester\_task\_1\_dmytro\_yunko.exe'

5 5

1 2

1 3

1 5

3 4

4 5

12212

Self Practice | Hide and Seek

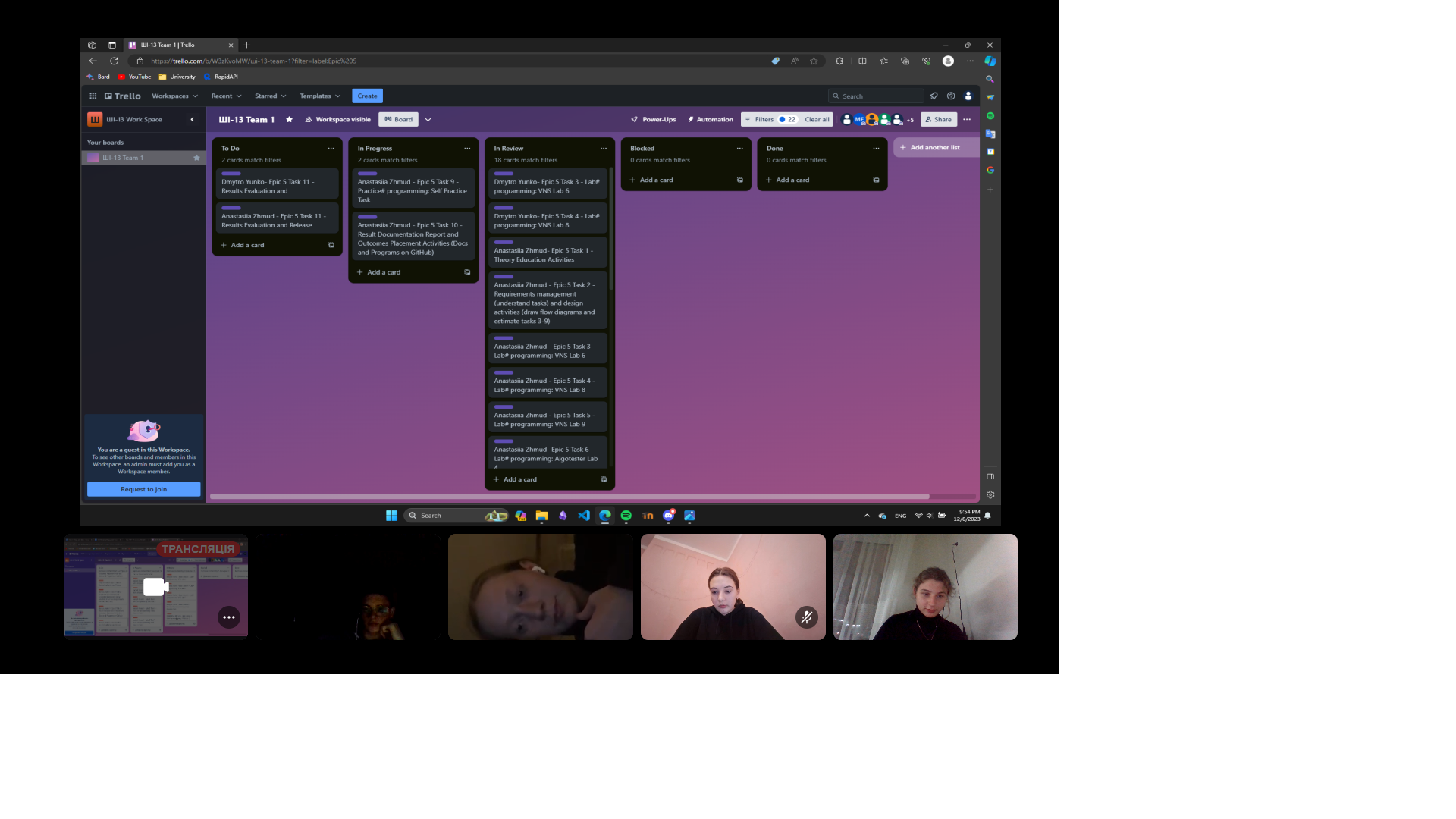


Рисунок 5 team meeting

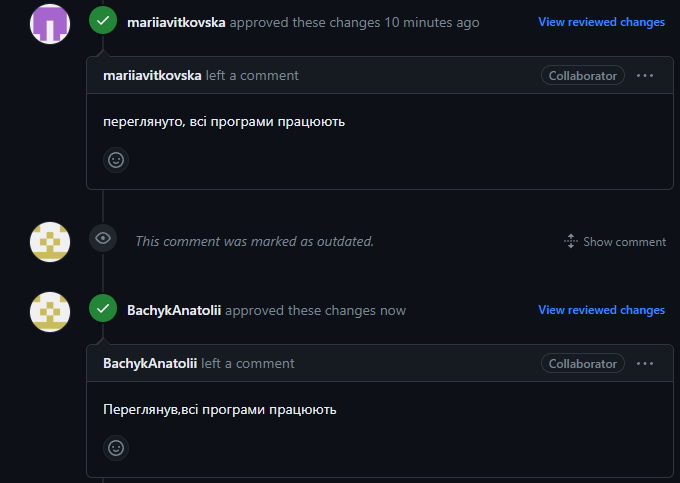


Рисунок 6 ПР звіт

Час затрачений на виконання завдання: 3дн

# **Висновки:**

У цій лабораторній робоиі я дослідив ключові аспекти роботи з файлами в C++, включаючи використання класів і об’єктів для роботи з файлами, формати і режими файлів, перевірку стану файлу, системи числення, запис та використання числових літералів, формати бінарних файлів, серіалізацію та десеріалізацію об’єктів у C++ та перетворення типів покажчиків.