Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту



**Звіт**

про виконання

**Лабораторних та практичних робіт № 6**

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

***з розділу***: « Динамічні структури. Черга. Стек. Списки. Дерева. Алгоритми обробки динамічних структур»

***Виконала:***

студент групи ШІ-11

Камінська Єлизавета Ігорівна

# **Тема роботи:**

Ознайомитись з динамічними структурами, а саме з чергою, стеком, списками та деревами, зрозуміти їхні властивості та відмінності. Крім цього дізнатись алгоритми обробки динамічних структур.

Написати коди для практичних і лабораторних робіт з ВНС, а також задач з Algotester. Залити всі дані на гітхаб і зробити pull-request.

# **Мета роботи:**

* Theory Education Activities
* Requirements management (understand tasks) and design activities (draw flow diagrams and estimate tasks 3-7)
* Lab# programming: VNS Lab 10
* Lab# programming: Algotester Lab 5
* Lab# programming: Algotester Lab 7-8
* Practice# programming: Class Practice Task
* Practice# programming: Self Practice Task
* Result Documentation Report and Outcomes Placement Activities (Docs and Programs on GitHub)
* Results Evaluation and Release

# **Теоретичні відомості:**

1. Індивідуальний план опрацювання теорії:

* Тема №1: Theory Education Activities
  + Джерела Інформації:
    - <https://www.geeksforgeeks.org/>
    - <https://en.cppreference.com/>
    - <https://cplusplus.com/>
    - <https://www.bestprog.net/uk/2019/09/11/examples-of-using-c-tools-for-working-with-files-ua/>
    - <https://www.bestprog.net/uk/2019/09/26/c-queue-general-concepts-ways-to-implement-the-queue-implementing-a-queue-as-a-dynamic-array-ua/>
    - <https://www.bestprog.net/uk/2019/09/18/c-the-concept-of-stack-operations-on-the-stack-an-example-implementation-of-the-stack-as-a-dynamic-array-ua/>
    - <https://acode.com.ua/urok-111-stek-i-kupa/>
    - <https://allref.com.ua/uk/skachaty/Dinamichni_strukturi_danih_%28S++%29>
    - Теорія з книжки С++ Primer 5th edition
    - Теорія з книжки “Дискретна математика” Ю.В.Нікольський, В.В.Пасічник та Ю.М.Щербина
    - Теоретичні відомості з лабораторних ВНС №10
  + Що опрацьовано:
    - Дізналась що таке черга, стек, списки та дерева та в чому їхня відмінність та їхні особливості.
    - Дізналась різні алгоритми обробки динамічних структур.
  + Статус: Ознайомлена
  + Початок опрацювання теми: 06.12
  + Звершення опрацювання теми: 08.12

# **Виконання роботи:**

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

Завдання №1: VNS Lab 10

* Варіант завдання: 8
* Деталі завдання: Написати програму, у якій створюються динамічні структури й виконати їхню обробку у відповідності зі своїм варіантом.

Для кожного варіанту розробити такі функції:

1. Створення списку.

2. Додавання елемента в список (у відповідності зі своїм варіантом).

3. Знищення елемента зі списку (у відповідності зі своїм варіантом).

4. Друк списку.

5. Запис списку у файл.

6. Знищення списку.

7. Відновлення списку з файлу.

Порядок виконання роботи:

1. Написати функцію для створення списку. Функція може створювати порожній список, а потім додавати в нього елементи.

2. Написати функцію для друку списку. Функція повинна передбачати вивід повідомлення, якщо список порожній.

3. Написати функції для знищення й додавання елементів списку у

відповідності зі своїм варіантом.

4. Виконати зміни в списку й друк списку після кожної зміни.

5. Написати функцію для запису списку у файл.

6. Написати функцію для знищення списку.

7. Записати список у файл, знищити його й виконати друк (при друці повинне бути видане повідомлення "Список порожній").

8. Написати функцію для відновлення списку з файлу.

9. Відновити список і роздрукувати його.

10.Знищити список.

8. Записи в лінійному списку містять ключове поле типу int. Сформувати

двонаправлений список. Знищити з нього елемент після елемента із заданим

номером, додати К елементів у початок списку.

Завдання №2: 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. Клiтинки якi мають суміжну сторону з вершиною мають висоту на один меншу, сумiжнi з ними i не розфарбовані мають ще на 1 меншу висоту i так далi.

Завдання №3: Algotester Lab 78

* Варіант завдання: 2
* Деталі завдання: Ваше завдання - власноруч реалізувати структуру даних "Динамічний масив".

Ви отримаєте Q запитів, кожен запит буде починатися зi слова-iдентифiкатора, пiсля якого йдуть його аргументи. Вам будуть поступати запити такого типу:

• Вставка:

Iдентифiкатор - insert

Ви отримуєте ціле число index елемента, на місце якого робити вставку.

Пiсля цього в наступному рядку рядку написане число N - розмір масиву, який треба вставити. У третьому рядку N цілих чисел - масив, який треба вставити на позицію index.

• Видалення:

Iдентифiкатор - erase

Ви отримуєте 2 цілих числа - index, індекс елемента, з якого почати видалення та n - кiлькiсть елементiв, яку треба видалити.

• Визначення розмiру:

Iдентифiкатор - size

Ви не отримаєте аргументів. Ви виводите кiлькiсть елементiв у динамічному масиві.

• Визначення кiлькостi зарезервованої пам'яті:

Iдентифiкатор - capacity

Ви не отримаєте аргументів.Ви виводите кількість зарезервованої пам'яті у динамічному масиві. Ваша реалiзацiя динамiчного масиву має мати фактор росту (Growth factor) рiвний 2.

• Отримання значення i-го елементу

Iдентифiкатор - get

Ви отримуєте ціле число - index, індекс елемента. Ви виводите значення елемента за індексом. Реалізувати використовуючи перегрузку оператора []

• Модифiкацiя значення i-го елементу

Iдентифiкатор - set

Ви отримуєте 2 цілих числа - індекс елемента, який треба змінити, та його нове значення. Реалізувати використовуючи перегрузку оператора []

• Вивiд динамічного масиву на екран

Iдентифiкатор - print

Ви не отримуєте аргументів. Ви виводите всi елементи динамічного масиву через пробіл. Реалізувати використовуючи перегрузку оператора <<

Завдання №4: Class Practice Task

* Деталі завдання: Бінарні дерева:
* Задача №4 - Віддзеркалення дерева

- 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 модифікується

## Задача №5 - Записати кожному батьківському вузлу суму підвузлів

- 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

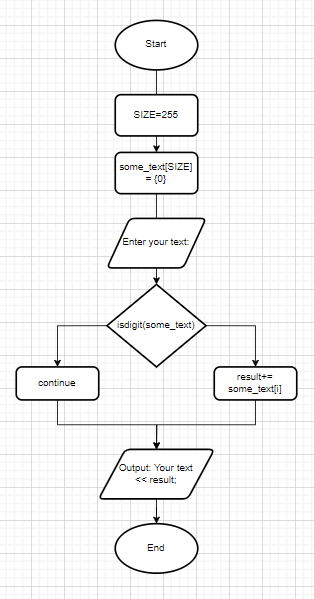
- значення змінюються від листків до кореня дерева

Завдання №5: Self Practice Task

* Деталі завдання: Ця програма запитує користувача ввести кількість студентів, а потім створює динамічний масив структур типу Student, кожна з яких має поля name (ім'я), age (вік) і gpa (середній бал). Далі програма запитує від користувача деталі для кожного студента, такі як ім'я, вік і середній бал. Після введення даних програма виводить інформацію про студентів, включаючи їхнє ім'я, вік та середній бал.

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

Програма №5 : Self practise



*Блоксхема до завдання. 1: Self practise.*

* Планований час на реалізацію: 30 хв

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

Завдання №1 VNS Lab 10

#include <iostream>

#include <fstream>

using namespace std;

struct Node {

int key;

Node\* next;

Node\* prev;

};

struct DblLinkedList {

size\_t size;

Node\* head;

Node\* tail;

};

// Функція для створення порожнього списку

DblLinkedList\* nullDblLinkedList() {

DblLinkedList\* tmp = new DblLinkedList;

tmp->size = 0;

tmp->head = tmp->tail = nullptr;

return tmp;

}

// Функція для додавання елемента в кінець списку

void addElement(DblLinkedList\* list, int value) {

Node\* tmp = new Node;

tmp->key = value;

tmp->next = nullptr;

tmp->prev = list->tail;

if (list->tail) {

list->tail->next = tmp;

}

list->tail = tmp;

if (list->head == nullptr) {

list->head = tmp;

}

list->size++;

}

// Функція для друку списку

void printList(const DblLinkedList\* list) {

Node\* current = list->head;

if (current == nullptr) {

cout << "The list is empty" << endl;

} else {

while (current != nullptr) {

cout << current->key << " ";

current = current->next;

}

cout << endl;

}

}

// Функція для знищення елемента після елемента із заданим номером та додавання К елементів у початок списку

void manipulateList(DblLinkedList\* list, size\_t indexToDeleteAfter, int K) {

Node\* current = list->head;

size\_t currentIndex = 0;

while (current != nullptr && currentIndex < indexToDeleteAfter) {

current = current->next;

currentIndex++;

}

if (current != nullptr && current->next != nullptr) {

Node\* toDelete = current->next;

current->next = toDelete->next;

if (toDelete->next) {

toDelete->next->prev = current;

}

delete toDelete;

list->size--;

}

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

Node\* newElement = new Node;

newElement->key = i + 1;

newElement->next = list->head;

newElement->prev = nullptr;

if (list->head) {

list->head->prev = newElement;

}

list->head = newElement;

if (list->tail == nullptr) {

list->tail = newElement;

}

list->size++;

}

}

// Функція для знищення списку

void deleteList(DblLinkedList\*& list) {

Node\* current = list->head;

while (current != nullptr) {

Node\* toDelete = current;

current = current->next;

delete toDelete;

}

delete list;

list = nullptr;

}

// Функція для запису списку у файл

void writeToFile(const DblLinkedList\* list, const std::string& filename) {

ofstream file(filename);

if (file.is\_open()) {

Node\* current = list->head;

while (current != nullptr) {

file << current->key << " ";

current = current->next;

}

file.close();

} else {

cout << "Error opening file for writing." << endl;

}

}

// Функція для відновлення списку з файлу

void restoreFromFile(DblLinkedList\* list, const std::string& filename) {

deleteList(list);

ifstream file(filename);

if (file.is\_open()) {

int value;

while (file >> value) {

addElement(list, value);

}

file.close();

} else {

cout << "Error opening file for reading." << endl;

}

}

int main() {

DblLinkedList\* my\_list = nullDblLinkedList();

int num;

cout << "Enter the number of elements: ";

cin >> num;

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

int value;

cout << "Enter element " << (i + 1) << ": ";

cin >> value;

addElement(my\_list, value);

}

cout << "Initial list: ";

printList(my\_list);

int indexToDeleteAfter;

cout << "Enter the index after which to delete an element: ";

cin >> indexToDeleteAfter;

int numElementsToAdd;

cout << "Enter the number of elements to add to the beginning: ";

cin >> numElementsToAdd;

manipulateList(my\_list, indexToDeleteAfter, numElementsToAdd);

cout << "List after operations: ";

printList(my\_list);

writeToFile(my\_list, "output.txt");

deleteList(my\_list);

restoreFromFile(my\_list, "output.txt");

cout << "Restored list: ";

printList(my\_list);

deleteList(my\_list);

return 0;

}

Завдання №2: Algotester Lab 5

#include <iostream>

#include <vector>

#include <queue>

using namespace std;

void paint\_the\_landscape(int r, int c, int X, int Y) {

vector<vector<int>> land(r, vector<int>(c, -1));

queue<pair<int, int>> queue\_mountain;

queue\_mountain.push({X - 1, Y - 1});

land[X - 1][Y - 1] = 0;

const int x[] = {1, -1, 0, 0};

const int y[] = {0, 0, 1, -1};

while (!queue\_mountain.empty()) {

pair<int, int> current = queue\_mountain.front();

queue\_mountain.pop();

for (int i = 0; i < 4; ++i) {

int new\_X = current.first + x[i];

int new\_Y = current.second + y[i];

if (new\_X >= 0 && new\_X < r && new\_Y >= 0 && new\_Y < c && land[new\_X][new\_Y] == -1) {

land[new\_X][new\_Y] = land[current.first][current.second] + 1;

queue\_mountain.push({new\_X, new\_Y});

}

}

}

int max\_height = 0;

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

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

max\_height = max(max\_height, land[i][j]);

}

}

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

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

cout << max\_height - land[i][j] << " ";

}

cout << endl;

}

}

int main() {

int MIN = 1;

int MAX = 1000;

int r, c, X, Y;

cin >> r >> c;

cin >> X >> Y;

if (r >= MIN && r <= MAX && c >= MIN && c <= MAX &&

X >= MIN && X <= r && Y >= MIN && Y <= c) {

paint\_the\_landscape(r, c, X, Y);

} else {

cout << "Invalid input, please try again.";

}

return 0;

}

Завдання №3: Algotester Lab 78

#include <iostream>

#include <cstdlib>

class DynamicArray {

private:

int \*array;

int size;

int capacity;

public:

DynamicArray() : array(nullptr), size(0), capacity(1) {}

~DynamicArray() {

delete[] array;

}

void insert(int index, int N, int \*values) {

if (index < 0 || index > size) {

return;

}

if (size + N > capacity) {

int newCapacity = std::max(2 \* capacity, size + N);

int \*newArray = new int[newCapacity];

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

newArray[i] = array[i];

}

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

newArray[index + i] = values[i];

}

for (int i = index; i < size; ++i) {

newArray[index + N + i] = array[i];

}

delete[] array;

array = newArray;

capacity = newCapacity;

size += N;

} else {

for (int i = size - 1; i >= index; --i) {

array[i + N] = array[i];

}

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

array[index + i] = values[i];

}

size += N;

}

}

void erase(int index, int n) {

if (index < 0 || index >= size || n <= 0) {

return;

}

for (int i = index + n; i < size; ++i) {

array[i - n] = array[i];

}

size -= n;

}

int getSize() const {

return size;

}

int getCapacity() const {

return capacity;

}

int& operator[](int index) {

return array[index];

}

friend std::ostream& operator<<(std::ostream& os, const DynamicArray& arr);

};

std::ostream& operator<<(std::ostream& os, const DynamicArray& arr) {

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

os << arr.array[i] << " ";

}

return os;

}

int main() {

int Q;

std::cin >> Q;

DynamicArray dynamicArray;

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

std::string identifier;

std::cin >> identifier;

if (identifier == "insert") {

int index, N;

std::cin >> index >> N;

int\* values = new int[N];

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

std::cin >> values[j];

}

dynamicArray.insert(index, N, values);

delete[] values;

} else if (identifier == "erase") {

int index, n;

std::cin >> index >> n;

dynamicArray.erase(index, n);

} else if (identifier == "size") {

std::cout << dynamicArray.getSize() << std::endl;

} else if (identifier == "capacity") {

std::cout << dynamicArray.getCapacity() << std::endl;

} else if (identifier == "get") {

int index;

std::cin >> index;

std::cout << dynamicArray[index] << std::endl;

} else if (identifier == "set") {

int index, value;

std::cin >> index >> value;

dynamicArray[index] = value;

} else if (identifier == "print") {

std::cout << dynamicArray << std::endl;

}

}

return 0;

}

Завдання №4: Class Practice Task

#include <iostream>

#include <queue>

using namespace std;

struct TreeNode {

int data;

TreeNode\* left;

TreeNode\* right;

TreeNode(int val) {

data = val;

left = nullptr;

right = nullptr;

}

};

// Рекурсивна функція для створення віддзеркаленого дерева

TreeNode\* mirror\_flip(TreeNode\* root) {

if (root == nullptr) {

return nullptr;

}

TreeNode\* left\_mirror = mirror\_flip(root->left);

TreeNode\* right\_mirror = mirror\_flip(root->right);

TreeNode\* mirrored\_node = new TreeNode(root->data);

mirrored\_node->left = right\_mirror;

mirrored\_node->right = left\_mirror;

return mirrored\_node;

}

// Допоміжна функція для виведення дерева (обхід у глибину)

void printTree(TreeNode\* root) {

if (root != nullptr) {

printTree(root->left);

cout << root->data << " ";

printTree(root->right);

}

}

// Функція для запису кожному батьківському вузлу суми підвузлів

void tree\_sum(TreeNode\* root) {

if (!root) {

return;

}

// Використовуємо обхід у ширину для ітеративного обходу дерева

queue<TreeNode\*> q;

q.push(root);

while (!q.empty()) {

TreeNode\* current = q.front();

q.pop();

int sum = 0;

if (current->left) {

sum += current->left->data;

q.push(current->left);

}

if (current->right) {

sum += current->right->data;

q.push(current->right);

}

if (current->left || current->right) {

current->data = sum;

}

}

}

void deleteTree(TreeNode\* root) {

if (root != nullptr) {

deleteTree(root->left);

deleteTree(root->right);

delete root;

}

}

int main() {

TreeNode\* root = new TreeNode(1);

root->left = new TreeNode(2);

root->right = new TreeNode(3);

root->right->left = new TreeNode(4);

root->right->right = new TreeNode(5);

cout << "Origin tree: ";

printTree(root);

cout << endl;

TreeNode\* mirrored\_root = mirror\_flip(root);

cout << "Mirrored tree: ";

printTree(mirrored\_root);

cout << endl;

// Виклик функції для запису суми підвузлів

tree\_sum(root);

// Виведення значень вузлів після змін

cout << "Modified tree values:" << endl;

cout << " " << root->data << endl;

cout << " / \\" << endl;

cout << " " << root->left->data << " " << root->right->data << endl;

cout << "/ \\" << endl;

cout << root->left->left->data << " " << root->left->right->data << endl;

delete root->right->right;

delete root->right->left;

delete root->left;

delete root->right;

delete root;

deleteTree(root);

deleteTree(mirrored\_root);

return 0;

}

Завдання №5: Self Practice Task

#include <iostream>

#include <string>

using namespace std;

struct Student {

string name;

int age;

double gpa;

};

int main() {

int numStudents;

cout << "Enter the number of students: ";

cin >> numStudents;

Student\* students = new Student[numStudents];

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

cout << "Enter details for student " << (i + 1) << endl;

cout << "Name: ";

cin.ignore();

getline(cin, students[i].name);

cout << "Age: ";

cin >> students[i].age;

cout << "GPA: ";

cin >> students[i].gpa;

}

cout << endl;

cout << "Student Information:" << endl;

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

cout << "Student " << (i + 1) << endl;

cout << "Name: " << students[i].name << endl;

cout << "Age: " << students[i].age << endl;

cout << "GPA: " << students[i].gpa << endl;

}

delete[] students;

return 0;

}

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

Завдання №1:VNS Lab 10

Enter the number of elements: 7

Enter element 1: 3

Enter element 2: 7

Enter element 3: 1

Enter element 4: 3

Enter element 5: 0

Enter element 6: 9

Enter element 7: 2

Initial list: 3 7 1 3 0 9 2

Enter the index after which to delete an element: 2

Enter the number of elements to add to the beginning: 3

List after operations: 3 2 1 3 7 1 0 9 2

Час затрачений на виконання завдання: 2 год.

Завдання №2: Algotester Lab 5

3 9

1 2

8 9 8 7 6 5 4 3 2

7 8 7 6 5 4 3 2 1

6 7 6 5 4 3 2 1 0

Час затрачений на виконання завдання: 2 год

Завдання №3: Algotester Lab 78

Час затрачений на виконання завдання: 2 год

Завдання №4: Class Practice Task

Mirrored tree: 5 3 4 1 2

Modified tree values:

5

/ \

2 9

/ \

Час затрачений на виконання завдання: 1.5 год

Завдання №5: Self Practice Task

Enter the number of students: 2

Enter details for student 1

Name: Jane

Age: 15

GPA: 9.8

Enter details for student 2

Name: Sofy

Age: 17

GPA: 10.5

Student Information:

Student 1

Name: Jane

Age: 15

GPA: 9.8

Student 2

Name: Sofy

Age: 17

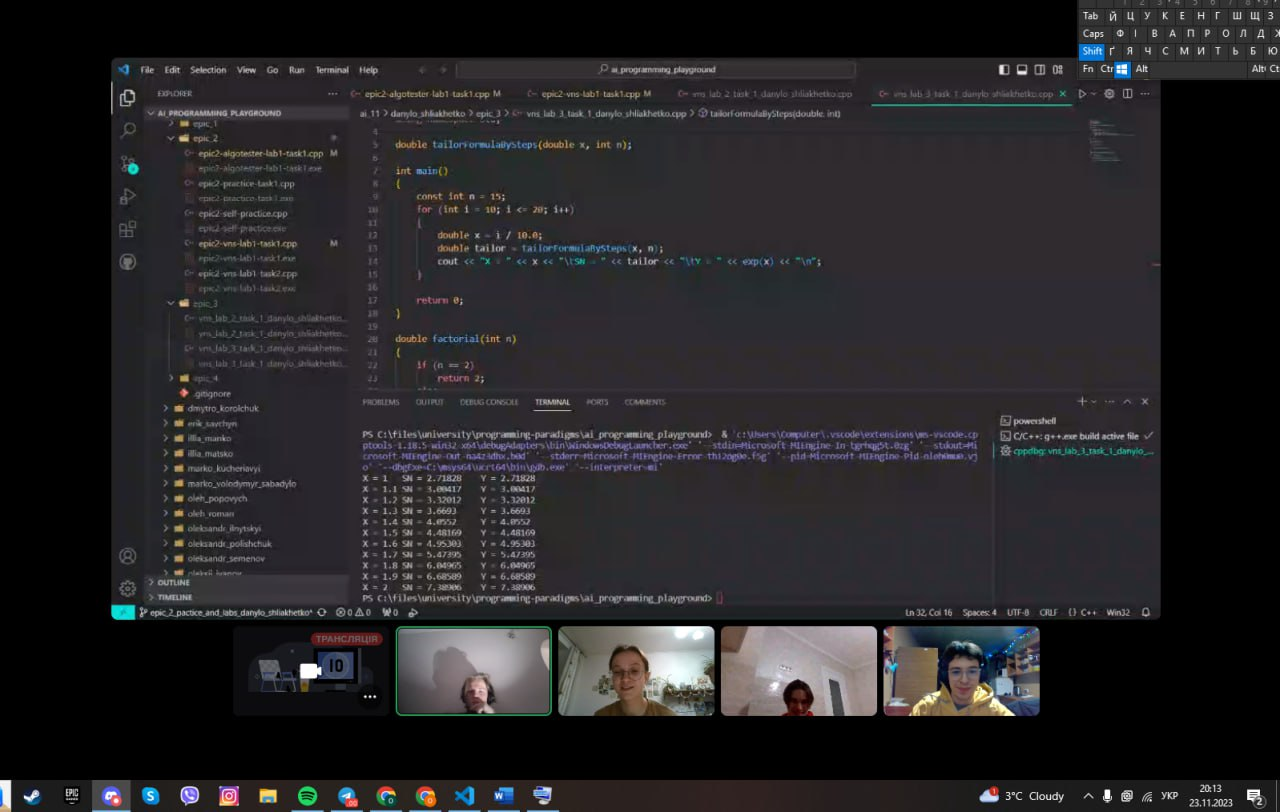
GPA: 10.5

Час затрачений на виконання завдання: 1 год

## **5. Командна робота та комунікація:**

Наша команда цього тижня провела лише 1 зустріч, але дуже грунтовну:

На ній ми обговорили план на цей епік, завантажили таски у трело, а також обговорили питання стосовно коду.



*Зустріч у діскорді.*

# **Висновки:**

# Протягом опрацювання цього епіку я вивчила новий матеріал, який стосується динамічних структур: списки, дерева, черга та стек.Також дізналась про їхні відмінності, ситуації їхнього використання та алгоритми їхньої обробки. Крім цього я використала ці теоретичні знання на практиці, пишучи програми з ВНС та Алготестера. Всі файли включно зі звітом запушила на гітхаб.