Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту



**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-12

Воронін Олександр Олександрович

# **Тема роботи:**

# Ознайомлення з динамічними структурами даних, такими як черга, стек, списки та дерева, передбачає вивчення їхніх унікальних властивостей, функцій і особливостей використання.

# **Мета роботи:**

Ознайомлення з основними динамічними структурами даних, такими як стеки, списки і дерева, включає в себе зрозуміння їхніх особливостей, роботи та функцій. Вивчення операцій над цими структурами включає в себе розуміння способів додавання, вилучення та обробки даних, що зберігаються у цих структурах. Практичні навички реалізації алгоритмів з використанням динамічних структур даних в мові програмування C++ допомагають закріпити ці знання та вміння у практичному застосуванні.

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Динамічні структури (Стек, Списки, Дерево).
* Тема №2: Алгоритми обробки динамічних структур.

1. Індивідуальний план опрацювання теорії:

* Тема №1: Динамічні структури ( Стек, Списки, Дерево).
  + Джерела Інформації
    - Стаття.

<https://acode.com.ua/urok-111-stek-i-kupa/#toc-1>

<https://www.bitdegree.org/learn/linked-list-c-plus-plus>

* + - Відео.

<https://www.youtube.com/watch?v=UHxtjVsOTHc>

* + Що опрацьовано:
    - Опрацьовано різні типи динамічних структур таких як стек, списки(однозв’язний та двозв’язний), та дерево. Опрацьовано як вони виглядають і як використовуються.
  + Статус: Ознайомлений
  + Початок опрацювання теми: 08.12.2023
  + Звершення опрацювання теми: 19.12.2023
* Тема №2: Алгоритми обробки динамічних структур.
  + Джерела Інформації:
    - Стаття.

<https://prometheus.org.ua/cs50/sections/section6.html>

* + Що опрацьовано:
    - Розглянуто базові алгоритми обробки динамічних структур.
  + Статус: Ознайомлений
  + Початок опрацювання теми: 08.12.2023
  + Звершення опрацювання теми: 19.12.2023

# **Виконання роботи:**

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

Завдання № 1. VNS Lab 10 – Task 1

* Варіант завдання: 24

Завдання № 2. Algotester Lab 5

* Варіант завдання: 3

Завдання № 3. Algotester Lab 78

* Варіант завдання: 2

Завдання № 4. Class Practice Work – Task 1

Зробити функцію reverse для однозвязного списку

Завдання № 5. Class Practice Work – Task 2

Порівняти 2 однозвязні списки

Завдання № 6. Class Practice Work – Task 3

Записати кожному батьківському вузлу суму підвузлів

Завдання № 7. Self Practice Work –Lab 2v3

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

Програма № 1. VNS Lab 10

* Планований час на реалізацію: 3 години

Програма № 2. Algotester Lab 5

Блок-схема

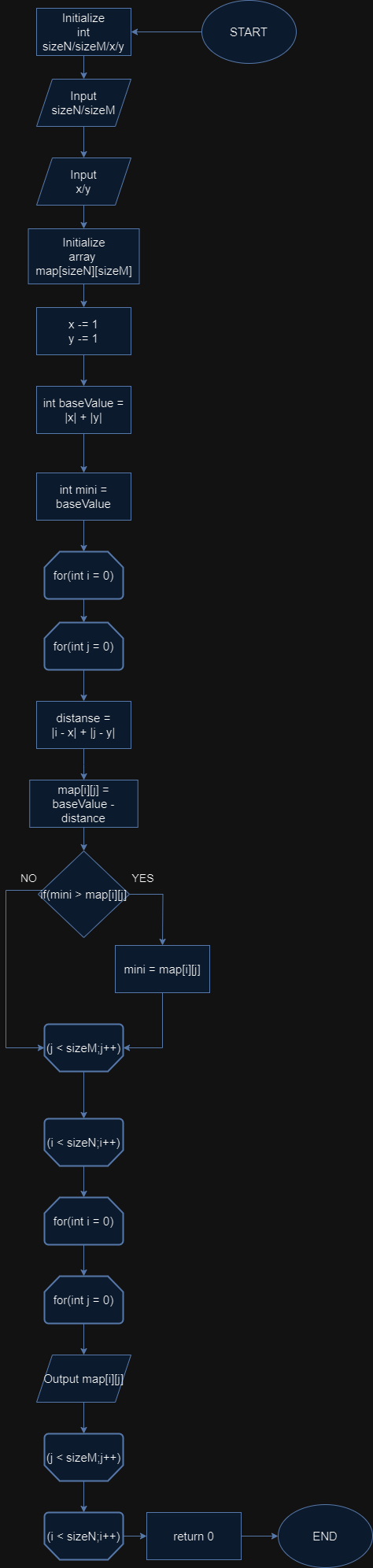


Рисунок 1.Блок схема до прогами algotester Lab 5v3

* Планований час на реалізацію: 2 години

Програма № 3. Algotester Lab 78

* Планований час на реалізацію: 6 годин

Програма № 4. Class Practice Work – Task 1

* Планований час на реалізацію: 1 години

Програма № 5. Class Practice Work – Task 2

* Планований час на реалізацію: 1 година

Програма № 6. Class Practice Work – Task 3

* Планований час на реалізацію: 1 година

Програма № 7. Self Practice Work –.Lab 2v3

Блок-схема

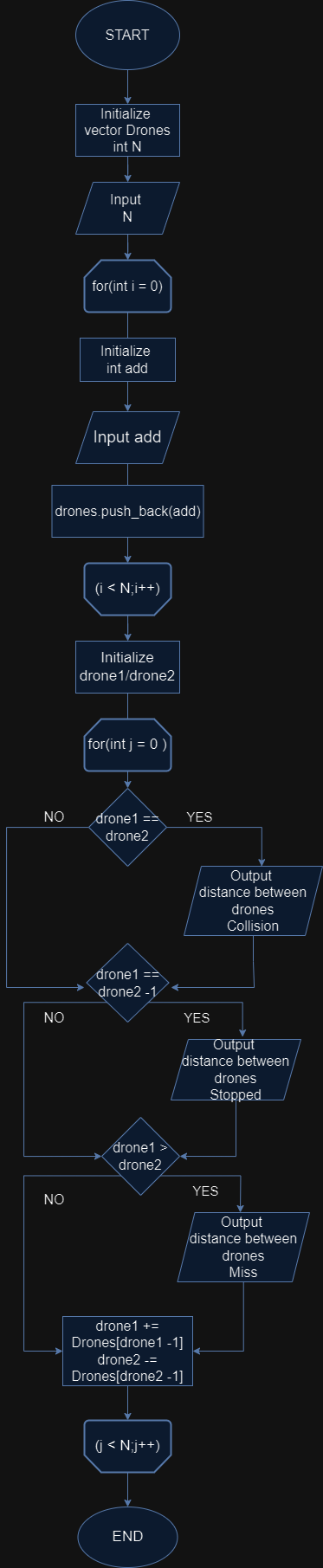


Рисунок 2.Блок схема до прогами self parctice

* Планований час на реалізацію: 1 години

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

Конфігурація середовища непотрібна.

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

**Завдання № 1. VNS Lab 10 – Task 1**

Код до програми:

#include <iostream>

#include <cstring>

using namespace std;

struct Node

{

    char\* data;

    Node\* next;

    Node\* prev;

};

struct DoubleLinkedList

{

    int size;

    Node\* head;

    Node\* tail;

};

DoubleLinkedList\* createList()

{

    DoubleLinkedList\* list = new DoubleLinkedList;

    list->size = 0;

    list->head = list->tail = nullptr;

    return list;

}

void addElementToList(DoubleLinkedList\* list, const char\* newData)

{

    Node\* newNode = new Node;

    newNode->data = strdup(newData);

    newNode->next = nullptr;

    if (list->head == nullptr) {

        newNode->prev = nullptr;

        list->head = list->tail = newNode;

    } else {

        newNode->prev = list->tail;

        list->tail->next = newNode;

        list->tail = newNode;

    }

    list->size++;

}

void addElements(DoubleLinkedList\* list)

{

    char newData[100];

    int keyToAad = 1;

    cout << "Enter word to add: ";

    cin >> newData;

    cout << "Enter key around which want to add: ";

    char key[100];

    cin >> key;

    Node\* current = list->head;

    while (current != nullptr && strcmp(current->data, key) != 0) {

        current = current->next;

    }

    if (current == nullptr) {

        cout << "Element with key " << key << " not found." << endl;

        return;

    }

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

        Node\* newNode = new Node;

        newNode->data = strdup(newData);

        newNode->prev = current->prev;

        newNode->next = current;

        if (current->prev != nullptr) {

            current->prev->next = newNode;

        } else {

            list->head = newNode;

        }

        current->prev = newNode;

        list->size++;

    }

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

        Node\* newNode = new Node;

        newNode->data = strdup(newData);

        newNode->prev = current;

        newNode->next = current->next;

        if (current->next != nullptr) {

            current->next->prev = newNode;

        } else {

            list->tail = newNode;

        }

        current->next = newNode;

        list->size++;

    }

}

void deleteElementKey(DoubleLinkedList\* list)

{

    char keyToDelete[100];

    cout << "Enter the word you wana delete: ";

    cin >> keyToDelete;

    Node\* current = list->head;

    while (current != nullptr && strcmp(current->data, keyToDelete) != 0) {

        current = current->next;

    }

    if (current == nullptr) {

        cout << "Element with key " << keyToDelete << " not found." << endl;

        return;

    }

    if (current->prev != nullptr) {

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

    } else {

        list->head = current->next;

    }

    if (current->next != nullptr) {

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

    } else {

        list->tail = current->prev;

    }

    delete[] current->data;

    delete current;

    list->size--;

}

void printList(DoubleLinkedList\* list)

{

    Node\* current = list->head;

    while (current != nullptr) {

        cout << current->data << " ";

        current = current->next;

    }

    cout << endl;

}

int main()

{

    DoubleLinkedList\* list = createList();

    addElementToList(list, "How");

    addElementToList(list, "Are");

    addElementToList(list, "you");

    cout << "Initial List: ";

    printList(list);

    addElements(list);

    cout << "List after add: ";

    printList(list);

    deleteElementKey(list);

    printList(list);

    delete list;

    return 0;

}

**Завдання № 2. Algotester Lab 5 – Task 1**

Код до програми:

#include <iostream>

#include <cmath>

using namespace std;

int main()

{

    int sizeN, sizeM, x, y;

    cin >> sizeN >> sizeM;

    cin >> x >> y;

    int map[sizeN][sizeM];

    x -= 1;

    y -= 1;

    int basev = abs(x) + abs(y);

    int min = basev;

    for (int i = 0; i < N; i++)

    {

        for (int j = 0; j < M; j++)

        {

            int distance = abs(i - x) + abs(j - y);

            map[i][j] = basev - distance;

            if (min > map[i][j])

            {

                min = map[i][j];

            }

        }

    }

    for (int i = 0; i < sizeN; ++i)

    {

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

        {

            cout << map[i][j] - min << " ";

        }

        cout << endl;

    }

    return 0;

}

**Завдання № 3. Algotester Lab 78 – Task 1**

Код до програми:

#include <iostream>

using namespace std;

class DynamArray {

private:

    int \*arr;

    int size;

    int capacity;

public:

    DynamArray() : arr(nullptr), size(0), capacity(1) {

        arr = new int[capacity];

    }

    ~DynamArray() {

        delete[] arr;

    }

    void resize(int newCapacity) {

        int \*newArr = new int[newCapacity];

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

            newArr[i] = arr[i];

        }

        delete[] arr;

        arr = newArr;

        capacity = newCapacity;

    }

    void insert(int index, int n, int \*data) {

        if (size + n >= capacity) {

            while (size + n >= capacity) {

                capacity \*= 2;

            }

            resize(capacity);

        }

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

            arr[i + n] = arr[i];

        }

        for (int i = 0; i < n; i++) {

            arr[index + i] = data[i];

        }

        size = size + n;

    }

    void erase(int index, int n) {

        int left = size - index - n;

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

            arr[index + i] = arr[index + i + n];

        }

        size -= n;

    }

    int getSize() const {

        return size;

    }

    int getCapacity() const {

        return capacity;

    }

    int get(int index) const {

        return arr[index];

    }

    void set(int index, int data) {

        arr[index] = data;

    }

    void printArr() const {

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

            cout << arr[i] << " ";

        }

        cout << endl;

    }

};

int main() {

    int Q;

    DynamArray arr;

    cin >> Q;

    for (int i = 0; i < Q; i++) {

        string rqst;

        cin >> rqst;

        if (rqst == "size") {

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

        }

        else if (rqst == "insert") {

            int index, num;

            cin >> index >> num;

            int \*data = new int[num];

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

                cin >> data[i];

            }

            arr.insert(index, num, data);

            delete[] data;

        }

        else if (rqst == "capacity") {

            cout << arr.getCapacity() << endl;

        }

        else if (rqst == "erase") {

            int index, num;

            cin >> index >> num;

            arr.erase(index, num);

        }

        else if (rqst == "set") {

            int index, data;

            cin >> index >> data;

            arr.set(index, data);

        }

        else if (rqst == "get") {

            int index;

            cin >> index;

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

        }

        else if (rqst == "print") {

            arr.printArr();

        }

    }

    return 0;

}

**Завдання № 4. Class Practice Work – Task 1**

Код до програми:

#include <iostream>

using namespace std;

struct Node

{

    int x;

    Node\* next;

};

Node\* reverse(Node\* head) {

    Node\* cur = head;

    Node\* next = nullptr;

    Node\* prev = nullptr;

    while (cur != nullptr) {

        next = cur->next;

        cur->next = prev;

        prev = cur;

        cur = next;

    }

    return prev;

}

void printList(Node\* head) {

    while (head != nullptr) {

        cout << head->x << " ";

        head = head->next;

    }

    cout << endl;

}

int main(){

    Node\* head = new Node();

    Node\* second = new Node();

    Node\* third = new Node();

    Node\* tail = new Node();

    head->x=1;

    head ->next = second;

    second->x=2;

    second->next = third;

    third->x=3;

    third->next=tail;

    tail->x=4;

    tail->next=nullptr;

    printList(head);

    head = reverse(head);

    printList(head);

}

**Завдання № 5. Class Practice Work – Task 2**

Код до програми:

#include <iostream>

using namespace std;

class Node {

public:

    int data;

    Node\* next;

};

bool compare(Node\* h1, Node\* h2) {

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

        if (h1->data != h2->data) {

            return false;

        }

        h1 = h1->next;

        h2 = h2->next;

    }

    return (h1 == nullptr && h2 == nullptr);

}

void printLinkedList(Node\* n) {

    while (n != nullptr) {

        cout << n->data;

        if (n->next != nullptr) {

            cout << " ";

        }

        n = n->next;

    }

    cout << endl;

}

int main() {

    Node\* number1 = new Node();

    number1->data = 4;

    number1->next = new Node();

    number1->next->data = 2;

    number1->next->next = new Node();

    number1->next->next->data = 3;

    number1->next->next->next = nullptr;

    Node\* number2 = new Node();

    number2->data = 4;

    number2->next = new Node();

    number2->next->data = 2;

    number2->next->next = new Node();

    number2->next->next->data = 3;

    number2->next->next->next = nullptr;

    cout << "Number 1: ";

    printLinkedList(number1);

    cout << "Number 2: ";

    printLinkedList(number2);

    if (compare(number1, number2)) {

        cout << "Lists are equal.\n";

    } else {

        cout << "Lists are different.\n";

    }

    delete number1;

    delete number2;

    return 0;

}

**Завдання № 6. Class Practice Work – Task 3**

Код до програми:

#include <iostream>

#include <stack>

using namespace std;

class TreeNode {

public:

    int value;

    TreeNode\* left;

    TreeNode\* right;

    TreeNode(int v) : value(v), left(nullptr), right(nullptr) {}

};

void tree\_sum(TreeNode\* root) {

    if (root == nullptr) {

        return;

    }

    stack<TreeNode\*> nodeStack;

    TreeNode\* cur = root;

    TreeNode\* prev = nullptr;

    while (cur || !nodeStack.empty()) {

        while (cur) {

            nodeStack.push(cur);

            cur = cur->left;

        }

        cur = nodeStack.top();

        if (cur->right && cur->right != prev) {

            cur = cur->right;

        } else {

            int sum = cur->value;

            if (cur->left) {

                sum += cur->left->value;

            }

            if (cur->right) {

                sum += cur->right->value;

            }

            cur->value = sum;

            nodeStack.pop();

            prev = cur;

            cur = nullptr;

        }

    }

}

void printTree(TreeNode\* root) {

    if (root == nullptr) {

        return;

    }

    printTree(root->left);

    cout << root->value << " ";

    printTree(root->right);

}

int main() {

    TreeNode\* root = new TreeNode(5);

    root->left = new TreeNode(2);

    root->right = new TreeNode(6);

    root->left->left = new TreeNode(1);

    root->left->right = new TreeNode(3);

    cout << "Original Tree: ";

    printTree(root);

    cout << endl;

    tree\_sum(root);

    cout << "Tree after sum calculation: ";

    printTree(root);

    cout << endl;

    delete root;

    return 0;

}

**Завдання № 7. Self Practice Work – Task 3. Lab 2v3**

Код до програми:

#include <iostream>

#include <vector>

using namespace std;

int main(){

    vector<int> Drones;

    int N;

    cin >> N;

    for(int i = 0;i<N;i++){

        int add;

        cin >> add;

        Drones.push\_back(add);

    }

    int drone1 = 1;

    int drone2 = N;

    for(int j =0;j< N;j++){

       if (drone1 == drone2){

            cout << drone1 << ' ' << drone2 << endl;

            cout << "Collision";

            return 0;

        }

       else if (drone1 == drone2 - 1){

            cout << drone1 << ' ' << drone2 << endl;

            cout << "Stopped";

            return 0;

        }

         else if (drone1 > drone2){

            cout << drone1 << ' ' << drone2 << endl;

            cout << "Miss";

            return 0;

        }

        drone1 += Drones[drone1 - 1];

        drone2 -= Drones[drone2 - 1];

    }

}

Рисунок. Результат виконання до програми № 1

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

**Завдання № 1. VNS Lab 10**

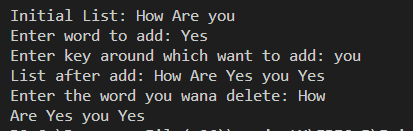


Рисунок 3. Результат виконання до програми № 1

Час затрачений на виконання завдання: 3 години

**Завдання № 2. Algotester Lab 5**

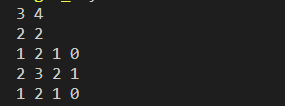


Рисунок 4. Результат виконання до програми № 2

Час затрачений на виконання завдання: 2 години

**Завдання № 3. Algotester Lab 78**

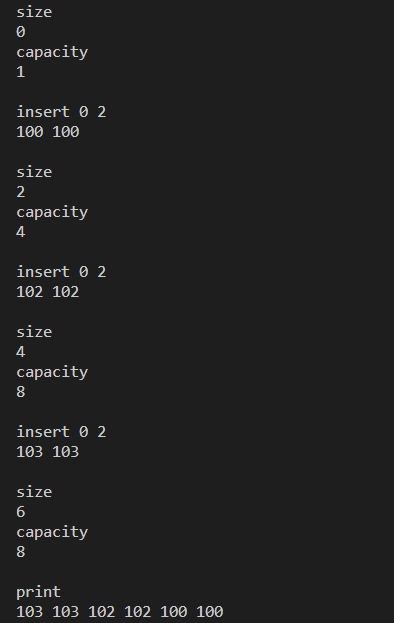


Рисунок 5. Результат виконання до програми № 3

Час затрачений на виконання завдання: 5 годин

**Завдання № 4. Class Practice Work – Task 1**



Рисунок 6. Результат виконання до програми № 4

Час затрачений на виконання завдання: 1 години

**Завдання № 5. Class Practice Work – Task 2**



Час затрачений на виконання завдання: 1 година

Рисунок 7. Результат виконання до програми № 5

**Завдання № 6. Class Practice Work – Task 3**



Рисунок 8. Результат виконання до програми № 6

Час затрачений на виконання завдання: 1 година

**Завдання № 7. Self Practice Work – Lab 2v3**

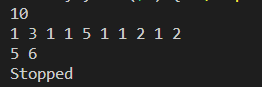


Рисунок 9. Результат виконання до програми № 7

Час затрачений на виконання завдання: 1 година

## **6. Кооперація з командою:**

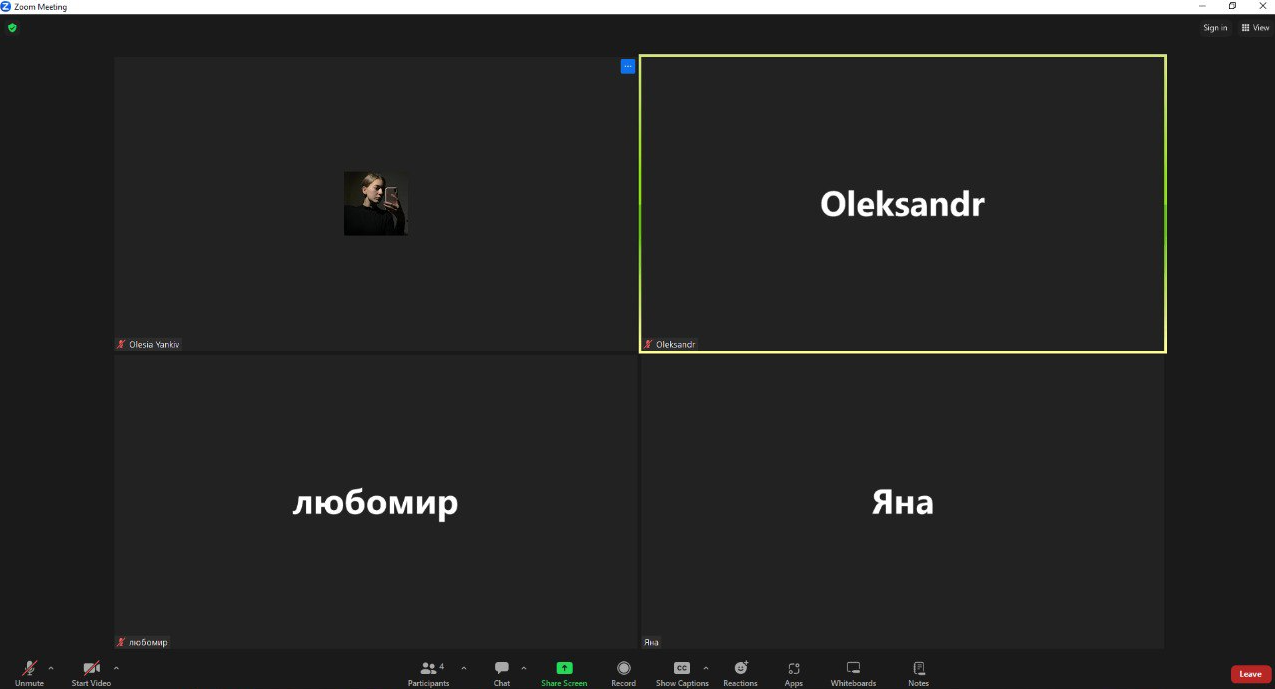


Рисунок 10. Зустріч з командою для обговорення питань по завданнях

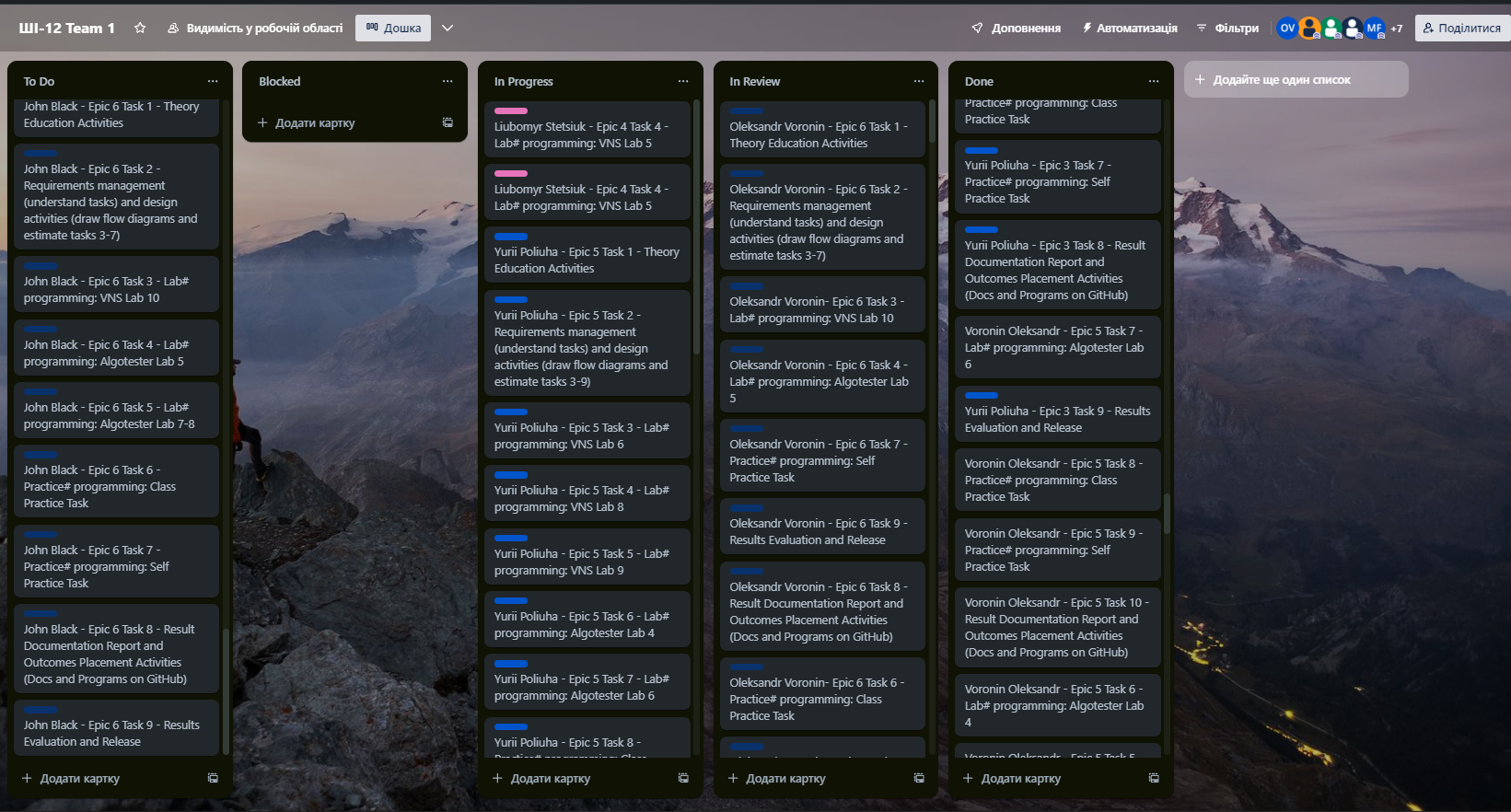


Рисунок 11. Прогрес в Trello

# **Висновки:**

Виконуючи дані практичні і лабораторні роботи, я ознайомився з основними динамічними структурами даних, такими як стеки, черги, списки і дерева, зрозумів їхні особливості, роботу та функції. Вивчення операцій над цими структурами включило в себе розуміння способів додавання, вилучення та обробки даних, що зберігаються у цих структурах. Практичні навички реалізації алгоритмів з використанням динамічних структур даних в мові програмування C++ допомогли закріпити ці знання та вміння у практичному застосуванні.