Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту



**Звіт**

про виконання

**Лабораторних та практичних робіт № 10**

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

***з розділу***: « Динамічні структури (Черга, Стек, Списки, Дерево). Алгоритми обробки динамічних структур»

***Виконала:***

студент групи ШІ-12

Ляшеник Христина Тарасівна

# **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, черги, 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 (Черга, Стек, Списки, Дерево).
* Тема №2: Алгоритми обробки динамічних структур.

1. Індивідуальний план опрацювання теорії:

* Тема №1: Динамічні структури (Черга, Стек, Списки, Дерево).
  + Джерела Інформації
    - Стаття. <https://acode.com.ua/urok-111-stek-i-kupa/#toc-1>
    - <https://studfile.net/preview/7013685/page:10/>
    - <https://prometheus.org.ua/cs50/sections/section6.html>
  + Що опрацьовано:
    - Опрацьовано різні типи динамічних структур таких як черга, стек, списки(однозв’язний та двозв’язний), та дерево. Опрацьовано як вони виглядають і як використовуються.
  + Статус: Ознайомлена
  + Початок опрацювання теми: 08.12.2023
  + Звершення опрацювання теми: 11.12.2023

# **Виконання роботи:**

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

Завдання №1 VNS Lab 10

* Варіант завдання-6
* 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.

Записи в лінійному списку містять ключове поле типу int. Сформувати

двонаправлений список. Знищити з нього елемент із заданим номером,

додати елемент у початок списку.

* Важливі деталі для врахування в імплементації програми – створення двонапрямленого списку

Завдання №2 Algotester Lab 5

* Варіант завдання-3
* Деталі завдання - <https://algotester.com/uk/ContestProblem/DisplayWithEditor/134644>

Завдання №3 Algotester Lab 7.8

* Варіант завдання-2
* Деталі завдання - <https://algotester.com/uk/ContestProblem/DisplayWithEditor/134648>
* Важливі деталі для врахування в імплементації програми – використання структури даних “Динамічний масив”

Завдання №4 Class Practice Work

* Варіант завдання - немає
* Деталі завдання

## Задача №1 - Реверс списку (Reverse list)

***Реалізувати метод реверсу списку:*** Node\* reverse(Node \*head);

*Умови задачі:*

-       використовувати цілочисельні значення в списку;

-       реалізувати метод реверсу;

-       реалізувати допоміжний метод виведення вхідного і обернутого списків;

## Задача №2 - Порівняння списків

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

*Умови задачі:*

-       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***.

## Задача №3 – Додавання великих чисел

Node\* add(Node \*n1, Node \*n2);

*Умови задачі:*

-       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.

## Задача №4 - Віддзеркалення дерева

TreeNode \*create\_mirror\_flip(TreeNode \*root);

*Умови задачі:*

-       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

void tree\_sum(TreeNode \*root);

*Умови задачі:*

-       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 Work

* Варіант завдання-немає
* Деталі завдання <https://algotester.com/uk/ArchiveProblem/DisplayWithEditor/29>

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

Програма №1 Vns Lab 10

* Планований час на реалізацію-2 години

Програма №2 Algotester Lab 5

* Блок-схема

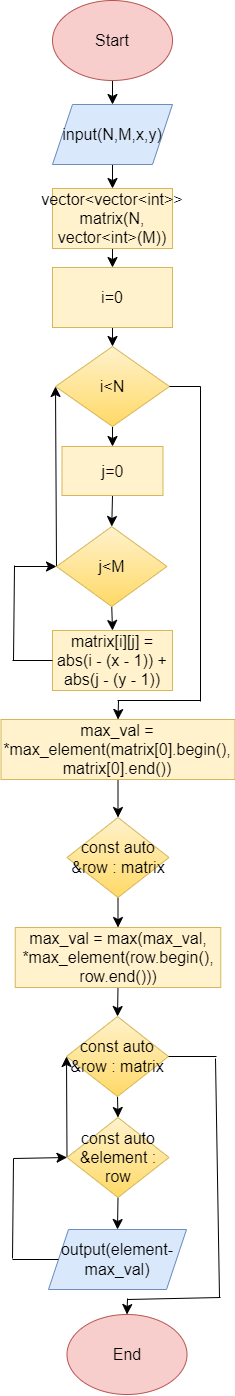


Рисунок 1 Блок-схема до Algotester Lab5

* Планований час на реалізацію-1 година

Програма №3 Algotester Lab 7.8

* Планований час на реалізацію-1.5 годин

Програма №4 Class Practice Work

* Планований час на реалізацію-2 години

Програма №5 Self Practice Work

* Планований час на реалізацію-1 година

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

Додаткової конфігурації середовища непотрібно

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

Завдання №1 Vns Lab 10

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/814/files#diff-7122829b8aa4a23d0b0e606ec094d88f39e33960e40c99625cadd2a6c21d3bb9>

#include <iostream>

#include <fstream>

using namespace std;

struct Node {

    int key;

    Node\* next;

    Node\* prev;

};

class DynamicList {

private:

    Node\* head;

public:

    DynamicList() : head(nullptr) {}

    void create\_list() {

        head = nullptr;

        cout << "Created an empty list." << endl;

    }

    void add\_element(int value) {

        Node\* new\_node = new Node;

        new\_node->key = value;

        new\_node->next = nullptr;

        new\_node->prev = nullptr;

        if (head == nullptr) {

            head = new\_node;

        } else {

            Node\* temp = head;

            while (temp->next != nullptr) {

                temp = temp->next;

            }

            temp->next = new\_node;

            new\_node->prev = temp;

        }

    }

    void remove\_element(int position) {

        if (head == nullptr) {

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

            return;

        }

        Node\* temp = head;

        if (position == 1) {

            head = temp->next;

            delete temp;

            cout << "Element at position " << position << " removed from the list." << endl;

            return;

        }

        for (int i = 1; temp != nullptr && i < position - 1; ++i) {

            temp = temp->next;

        }

        if (temp == nullptr || temp->next == nullptr) {

            cout << "Position exceeds the list size." << endl;

            return;

        }

        Node\* next\_node = temp->next->next;

        delete temp->next;

        temp->next = next\_node;

        cout << "Element at position " << position << " removed from the list." << endl;

    }

    void add\_to\_beginning(int value) {

        Node\* new\_node = new Node;

        new\_node->key = value;

        new\_node->next = head;

        new\_node->prev = nullptr;

        if (head != nullptr) {

            head->prev = new\_node;

        }

        head = new\_node;

        cout << "Added element with key " << value << " to the beginning of the list." << endl;

    }

    void print\_list() {

        if (head == nullptr) {

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

            return;

        }

        Node\* current = head;

        cout << "List elements: ";

        while (current != nullptr) {

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

            current = current->next;

        }

        cout << endl;

    }

    void write\_to\_file(string filename) {

        ofstream file(filename);

        if (!file) {

            cout << "Failed to open file for writing." << endl;

            return;

        }

        Node\* current = head;

        while (current != nullptr) {

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

            current = current->next;

        }

        file.close();

        cout << "List written to file " << filename << "." << endl;

    }

    void clear\_list() {

        Node\* current = head;

        Node\* temp;

        while (current != nullptr) {

            temp = current;

            current = current->next;

            delete temp;

        }

        head = nullptr;

        cout << "List cleared." << endl;

    }

    void restore\_from\_file(string filename) {

        clear\_list();

        ifstream file(filename);

        int value;

        if (!file) {

            cout << "Failed to open file for restoring the list." << endl;

            return;

        }

        while (file >> value) {

            add\_element(value);

        }

        file.close();

        cout << "List restored from file " << filename << "." << endl;

    }

    void delete\_list() {

        clear\_list();

        cout << "List deleted." << endl;

    }

};

int main() {

    DynamicList my\_list;

    my\_list.create\_list();

    int numberOfElements;

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

    cin >> numberOfElements;

    cout << "Enter the elements: ";

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

        int element;

        cin >> element;

        my\_list.add\_element(element);

    }

    my\_list.print\_list();

    my\_list.write\_to\_file("my\_list.txt");

    int positionToRemove;

    cout << "Enter the position of the element to remove: ";

    cin >> positionToRemove;

    my\_list.remove\_element(positionToRemove);

    my\_list.print\_list();

    int elementToAdd;

    cout << "Enter the element to add at the beginning: ";

    cin >> elementToAdd;

    my\_list.add\_to\_beginning(elementToAdd);

    my\_list.print\_list();

    my\_list.write\_to\_file("my\_list.txt");

    my\_list.delete\_list();

    my\_list.print\_list();

    my\_list.restore\_from\_file("my\_list.txt");

    my\_list.print\_list();

    my\_list.delete\_list();

    return 0;

}

Завдання №2 Algotester Lab 5

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/814/files#diff-8e03a1bf6b7b728c6b2cabf7b92bab415928e8adde526ef5cad4a1ac5283512c>

#include <iostream>

#include <vector>

#include <cmath>

#include <algorithm>

using namespace std;

int main() {

    int N, M, x, y;

    cin >> N >> M >> x >> y;

    vector<vector<int>> matrix(N, vector<int>(M));

    for (int i = 0; i < N; ++i) {

        for (int j = 0; j < M; ++j) {

            matrix[i][j] = abs(i - (x - 1)) + abs(j - (y - 1));

        }

    }

    int max\_val = \*max\_element(matrix[0].begin(), matrix[0].end());

    for (const auto &row : matrix) {

        max\_val = max(max\_val, \*max\_element(row.begin(), row.end()));

    }

    for (const auto &row : matrix) {

        for (const auto &element : row) {

            cout << abs(element - max\_val) << " ";

        }

        cout << endl;

    }

    return 0;

}

Завдання №3 Algotester Lab 78

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/814/files#diff-128ac97be7c83e11f9e384c497fa7b2209ab2f1fa9725e2da907d52b753ba4e2>

#include <iostream>

using namespace std;

class DynamicArray {

private:

    int \*array;

    int currentSize;

    int currentCapacity;

    void resize(int newCapacity) {

        int \*tempArray = new int[newCapacity];

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

            tempArray[i] = array[i];

        }

        delete[] array;

        array = tempArray;

        currentCapacity = newCapacity;

    }

public:

    DynamicArray() : array(nullptr), currentSize(0), currentCapacity(1) {

        array = new int[currentCapacity];

    }

    ~DynamicArray() {

        delete[] array;

    }

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

        if (currentSize + N >= currentCapacity) {

            while (currentSize + N >= currentCapacity) {

                currentCapacity \*= 2;

            }

            resize(currentCapacity);

        }

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

            array[i + N] = array[i];

        }

        for (int i = 0; i < N; ++i) {

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

        }

        currentSize += N;

    }

    void erase(int index, int n) {

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

            array[i] = array[i + n];

        }

        currentSize -= n;

    }

    int get(int index) {

        return array[index];

    }

    void set(int index, int value) {

        array[index] = value;

    }

    int size() {

        return currentSize;

    }

    int capacity() {

        return currentCapacity;

    }

    void print() {

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

            cout << array[i] << " ";

        }

        cout <<endl;

    }

};

int main() {

    int Q;

    cin >> Q;

    DynamicArray dynamicArray;

    for (int i = 0; i < Q; ++i) {

        string query;

        cin >> query;

        if (query == "insert") {

            int index, N;

            cin >> index >> N;

            int \*elements = new int[N];

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

                cin >> elements[j];

            }

            dynamicArray.insert(index, N, elements);

            delete[] elements;

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

            int index, n;

            cin >> index >> n;

            dynamicArray.erase(index, n);

        } else if (query == "size") {

            cout << dynamicArray.size() << endl;

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

            cout << dynamicArray.capacity() << endl;

        } else if (query == "get") {

            int index;

            cin >> index;

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

        } else if (query == "set") {

            int index, value;

            cin >> index >> value;

            dynamicArray.set(index, value);

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

            dynamicArray.print();

        }

    }

    return 0;

}

Завдання №4 Class Practice Work Task 1

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/814/files#diff-eb3cfd419ff9cb35d7e4752bb10a1fc0a5fe391e6ae47ba6a945b2ba51c225dc>

#include <iostream>

using namespace std;

struct Node {

    int data;

    Node\* next;

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

};

class LinkedList {

private:

    Node\* head;

public:

    LinkedList() : head(nullptr) {}

    void append(int val) {

        Node\* newNode = new Node(val);

        if (head == nullptr) {

            head = newNode;

            return;

        }

        Node\* temp = head;

        while (temp->next != nullptr) {

            temp = temp->next;

        }

        temp->next = newNode;

    }

    Node\* 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;

        return head;

    }

    void printList() {

        Node\* temp = head;

        while (temp != nullptr) {

            std::cout << temp->data << " ";

            temp = temp->next;

        }

        cout << endl;

    }

};

int main() {

    LinkedList list;

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

    int num;

    cin >> num;

    cout << "Enter " << num << " elements:" << endl;

    for (int i = 0; i < num; ++i) {

        int val;

        cin >> val;

        list.append(val);

    }

    cout << "Original List: ";

    list.printList();

    list.reverse();

    cout << "Reversed List: ";

    list.printList();

    return 0;

}

Завдання №5 Class Practice Work Task 2

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/814/files#diff-280ffea27c4aa4ca04d513b43602ee28fa49b8428aad6f5c34255534e6ce47e8>

#include <iostream>

using namespace std;

struct Node {

    int data;

    Node\* next;

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

};

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

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

        if (h1->data != h2->data) {

            return false;

        }

        h1 = h1->next;

        h2 = h2->next;

    }

    if (h1 != nullptr || h2 != nullptr) {

        return false;

    }

    return true;

}

Node\* createList(int numElements) {

    Node\* head = nullptr;

    Node\* temp = nullptr;

    int val;

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

        cout << "Enter a number: ";

        cin >> val;

        if (head == nullptr) {

            head = new Node(val);

            temp = head;

        } else {

            temp->next = new Node(val);

            temp = temp->next;

        }

    }

    return head;

}

int main() {

    int num1, num2;

    cout << "Enter the number of elements for the first list: ";

    cin >> num1;

    Node\* head1 = createList(num1);

    cout << "\nEnter the number of elements for the second list: ";

    cin >> num2;

    Node\* head2 = createList(num2);

    cout << "\nComparison result: " << boolalpha << compare(head1, head2) << endl;

    return 0;

}

Завдання №6 Class Practice Work Task 3

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/814/files#diff-bf793aedb6129ac4b22d2245a1d7c9a3488829bc2656ac9f7903c3980ada3203>

#include <iostream>

#include <sstream>

using namespace std;

struct Node {

    int data;

    Node\* next;

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

};

class LinkedList {

private:

    Node\* head;

public:

    LinkedList() : head(nullptr) {}

    void append(int val) {

        Node\* newNode = new Node(val);

        if (head == nullptr) {

            head = newNode;

            return;

        }

        Node\* temp = head;

        while (temp->next != nullptr) {

            temp = temp->next;

        }

        temp->next = newNode;

    }

    void printList() {

        Node\* temp = head;

        while (temp != nullptr) {

            cout << temp->data << " ";

            temp = temp->next;

        }

        cout << endl;

    }

    Node\* getHead() {

        return head;

    }

};

Node\* add(Node \*n1, Node \*n2) {

    LinkedList result;

    int carry = 0;

    while (n1 != nullptr || n2 != nullptr || carry != 0) {

        int sum = carry;

        if (n1 != nullptr) {

            sum += n1->data;

            n1 = n1->next;

        }

        if (n2 != nullptr) {

            sum += n2->data;

            n2 = n2->next;

        }

        carry = sum / 10;

        sum %= 10;

        result.append(sum);

    }

    return result.getHead();

}

int main() {

    LinkedList num1, num2;

    string number1, number2;

    cout << "Enter the first numbers: ";

    getline(cin, number1);

    istringstream iss1(number1);

    int digit;

    while (iss1 >> digit) {

        num1.append(digit);

    }

    cout << "Enter the second numbers: ";

    getline(cin, number2);

    istringstream iss2(number2);

    while (iss2 >> digit) {

        num2.append(digit);

    }

    Node\* sum = add(num1.getHead(), num2.getHead());

    cout << "Sum: ";

    while (sum != nullptr) {

        cout << sum->data << " ";

        sum = sum->next;

    }

    cout << endl;

    return 0;

}

Завдання №7 Class Practice Work Task 4

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/814/files#diff-1f0914e4aaf469c90a358b0196d50d0a4c463b972bc764e6e8ffe8dc7a407538>

#include <iostream>

using namespace std;

struct TreeNode {

    int data;

    TreeNode \*left;

    TreeNode \*right;

    TreeNode(int val) : data(val), left(nullptr), right(nullptr) {}

};

TreeNode \*create\_mirror\_flip(TreeNode \*root) {

    if (root == nullptr) {

        return nullptr;

    }

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

    newNode->left = create\_mirror\_flip(root->right);

    newNode->right = create\_mirror\_flip(root->left);

    return newNode;

}

void inorderTraversal(TreeNode \*root) {

    if (root == nullptr) {

        return;

    }

    inorderTraversal(root->left);

    cout << root->data << " ";

    inorderTraversal(root->right);

}

int main() {

    TreeNode \*root = new TreeNode(1);

    root->left = new TreeNode(2);

    root->right = new TreeNode(3);

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

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

    cout << "Original tree (inorder): ";

    inorderTraversal(root);

    cout << endl;

    TreeNode \*mirrorTree = create\_mirror\_flip(root);

    cout << "Mirror-flipped tree (inorder): ";

    inorderTraversal(mirrorTree);

    cout << endl;

    return 0;

}

Завдання №8 Class Practice Work Task 5

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/814/files#diff-492a76ebfc2e600d850f6ec50590d14cd5fac7aefd309b59ca819f1f1b8dc51e>

#include <iostream>

#include <stack>

using namespace std;

struct TreeNode {

    int data;

    TreeNode\* left;

    TreeNode\* right;

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

};

void calculateSubtreeSum(TreeNode\* root) {

    if (root == nullptr) {

        return;

    }

    stack<TreeNode\*> nodes;

    TreeNode\* current = root;

    TreeNode\* prev = nullptr;

    while (current != nullptr || !nodes.empty()) {

        while (current != nullptr) {

            nodes.push(current);

            current = current->left;

        }

        current = nodes.top();

        if (current->left == nullptr && current->right == nullptr) {

            nodes.pop();

            prev = current;

            current = nullptr;

        } else if (current->right != nullptr && current->right != prev) {

            current = current->right;

        } else {

            TreeNode\* temp = current;

            current = nullptr;

            nodes.pop();

            int sum = temp->data;

            if (temp->left != nullptr) {

                sum += temp->left->data;

            }

            if (temp->right != nullptr) {

                sum += temp->right->data;

            }

            temp->data = sum;

            prev = temp;

        }

    }

}

void inorderTraversal(TreeNode\* root) {

    if (root != nullptr) {

        inorderTraversal(root->left);

        cout << root->data << " ";

        inorderTraversal(root->right);

    }

}

int main() {

    TreeNode\* root = new TreeNode(1);

    root->left = new TreeNode(2);

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

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

    root->right = new TreeNode(5);

    cout << "Original Binary Tree: ";

    inorderTraversal(root);

    cout << endl;

    calculateSubtreeSum(root);

    cout << "Tree after computing subtree sums: ";

    inorderTraversal(root);

    cout << endl;

    return 0;

}

Завдання №9 Self Practice Work

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/814/files#diff-03c3d3eee4c6dc811f1fc80a3472c88a0f3120f4c5e604482a1ac50ab4312a01>

#include <iostream>

#include <stack>

#include <vector>

#include <utility>

using namespace std;

int main() {

    string s;

    getline(cin, s);

    stack<int> st;

    vector<pair<int, int>> brackets;

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

        if (s[i] == '(') {

            st.push(i + 1);

        } else if (s[i] == ')') {

            brackets.push\_back(make\_pair(st.top(), i + 1));

            st.pop();

        }

    }

    cout << brackets.size() << endl;

    for (auto p : brackets) {

        cout << p.first << " " << p.second << endl;

    }

    return 0;

}

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

Завдання №1 Vns Lab 10

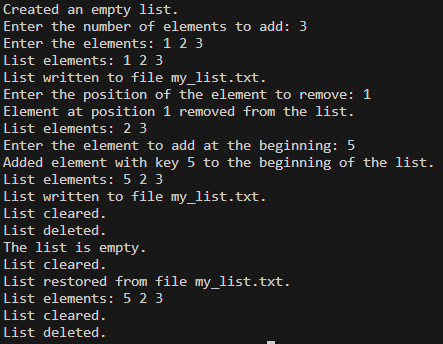


Рисунок 2 Результат виконання завдання №1

Час затрачений на виконання завдання-2 години

Завдання№2 Algotester Lab 5

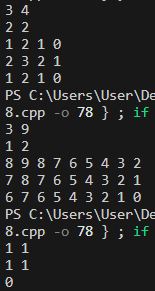


Рисунок 3 Результат виконання завдання №2

Час затрачений на виконання завдання-1 година

Завдання№3 Algotester Lab 78

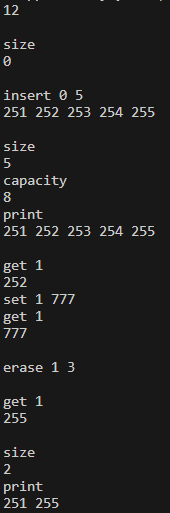


Рисунок 4 Результат виконання завдання №3

Час затрачений на виконання завдання-2 години

Завдання№4 Class Practice Work Task1

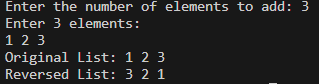


Рисунок 5 Результат виконання завдання №4

Час затрачений на виконання завдання-40 хвилин

Завдання№5 Class Practice Work Task2

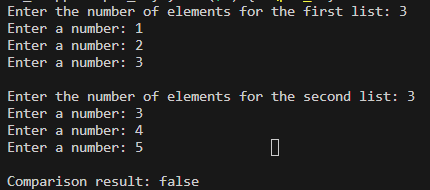


Рисунок 6 Результат виконання завдання №5

Час затрачений на виконання завдання-50 хвилин

Завдання№6 Class Practice Work Task3



Рисунок 7 Результат виконання завдання №6

Час затрачений на виконання завдання-50 хвилин

Завдання№7 Class Practice Work Task4



Рисунок 8 Результат виконання завдання №7

Час затрачений на виконання завдання-40 хвилин

Завдання№8 Class Practice Work Task5



Рисунок 9 Результат виконання завдання №8

Час затрачений на виконання завдання-50 хвилин

Завдання№9 Self Practice Work

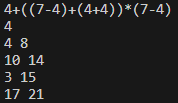


Рисунок 10 Результат виконання завдання №9

Час затрачений на виконання завдання-1 година

## **6. Кооперація з командою:**

* Скрін з 1-ї зустрічі по обговоренню задач Епіку та Скрін прогресу по Трелло

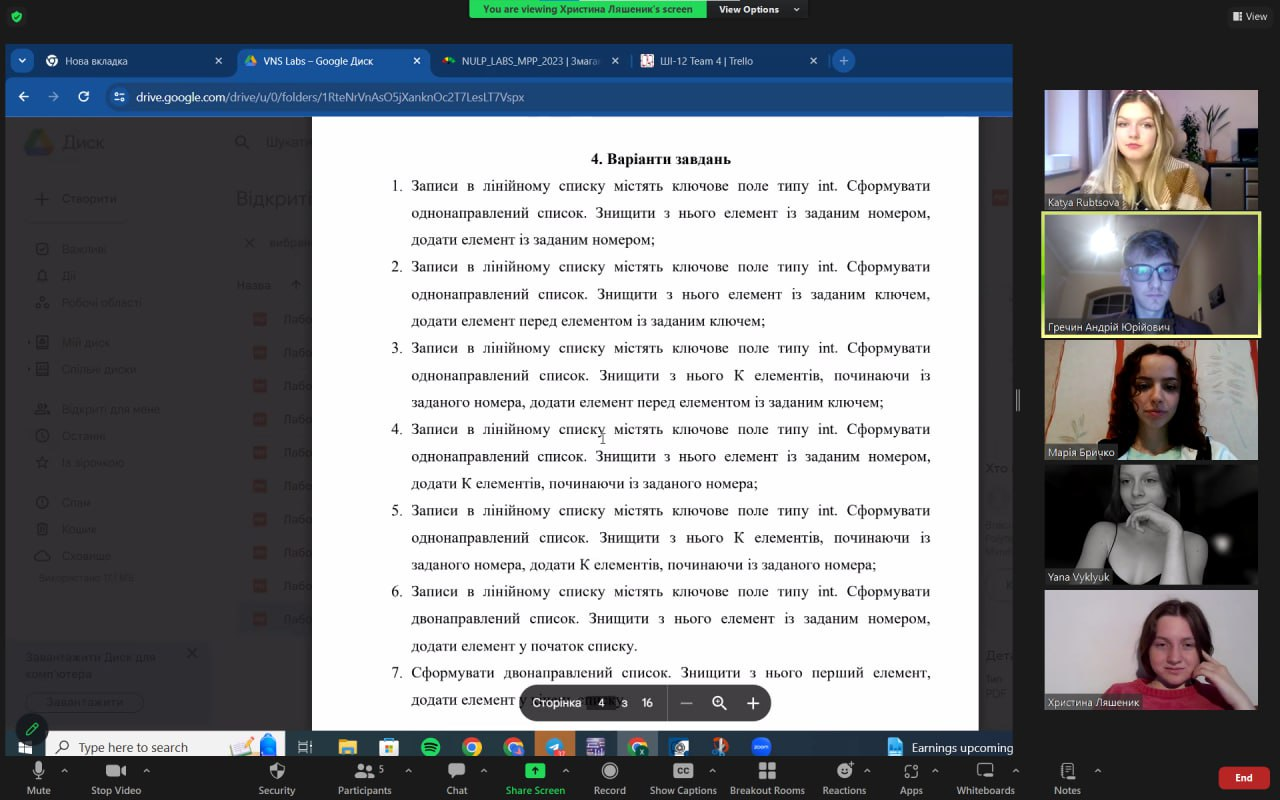


Рисунок 11 Скрін з 1-ї зустрічі з командою

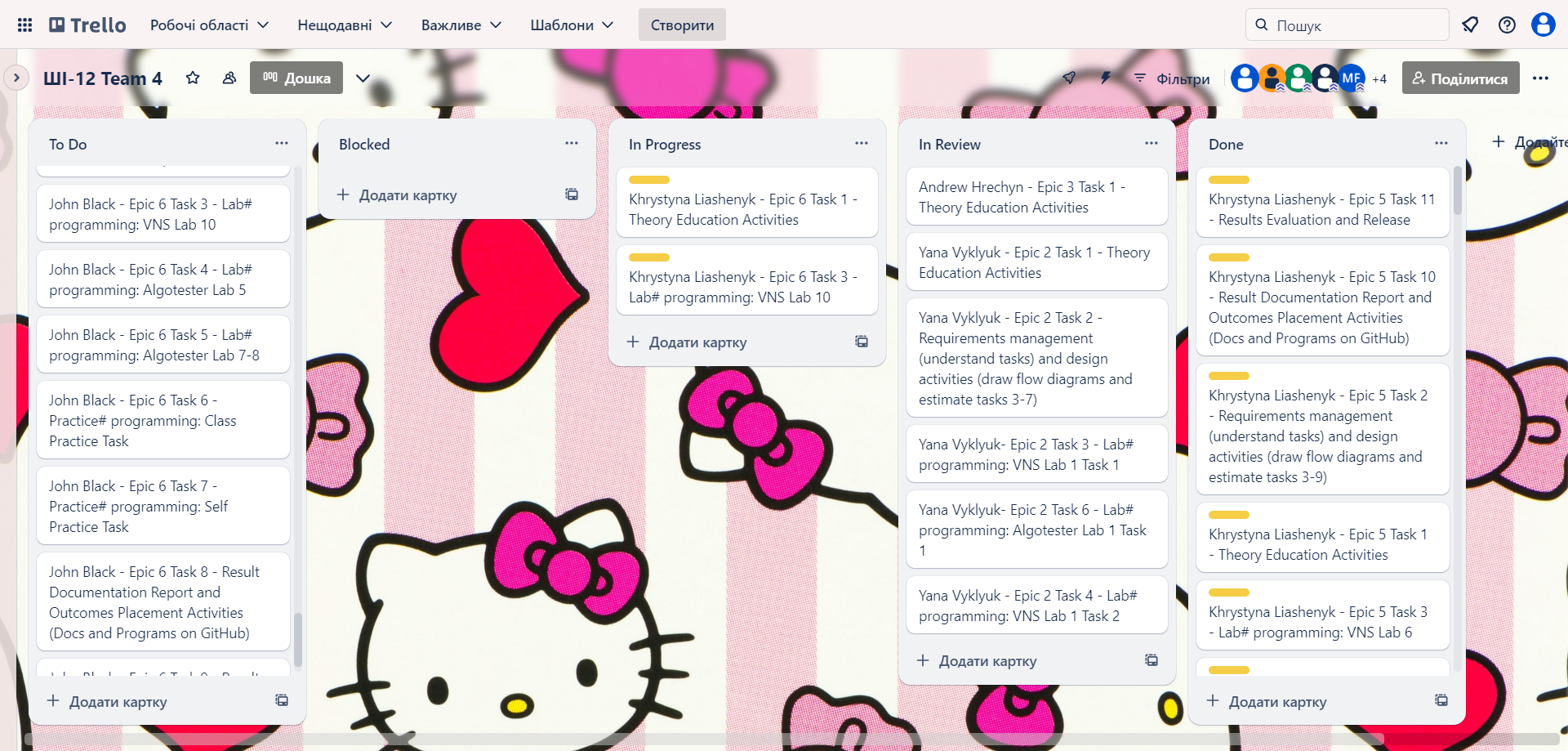


Рисунок 12 Trello на момент 1-ї зустрічі

* Скрін з 2-ї зустрічі по обговоренню задач Епіку та Скрін прогресу по Трелло

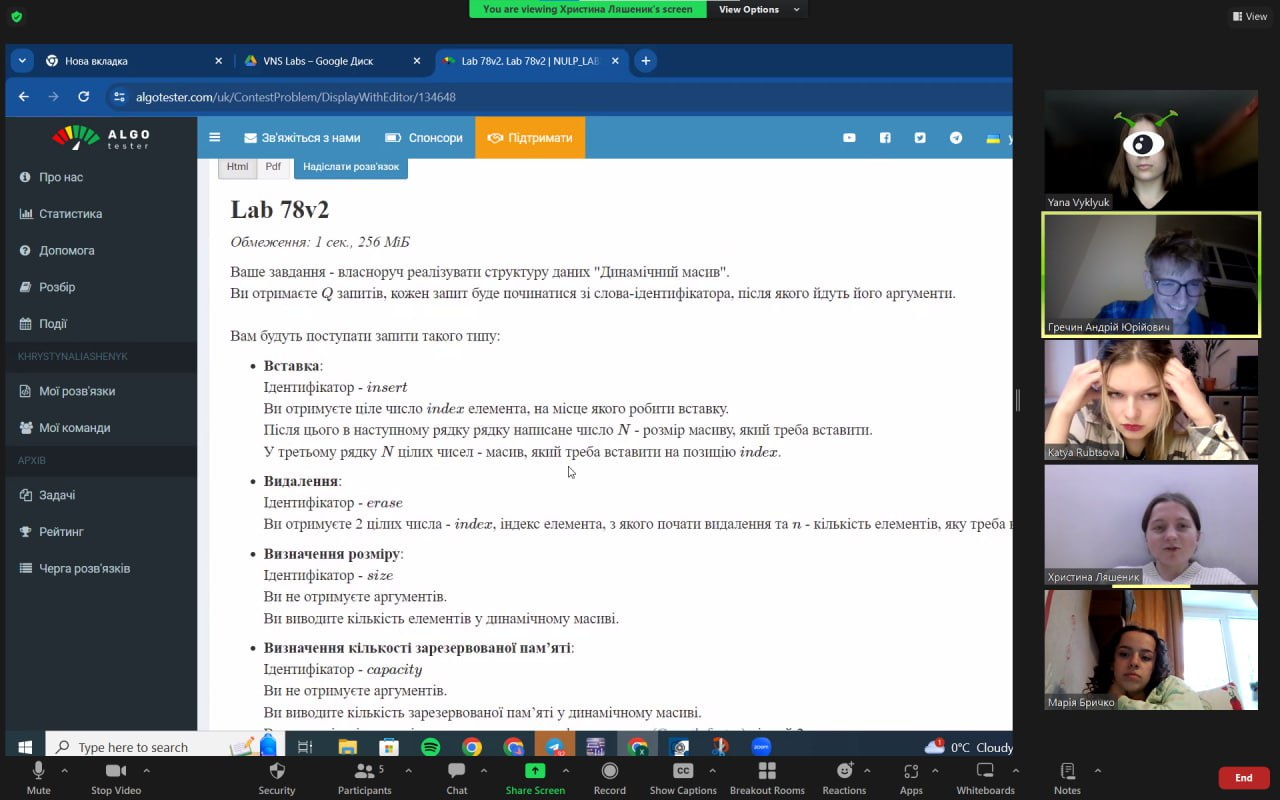


Рисунок 13 Скрін з 2-ї зустрічі з командою

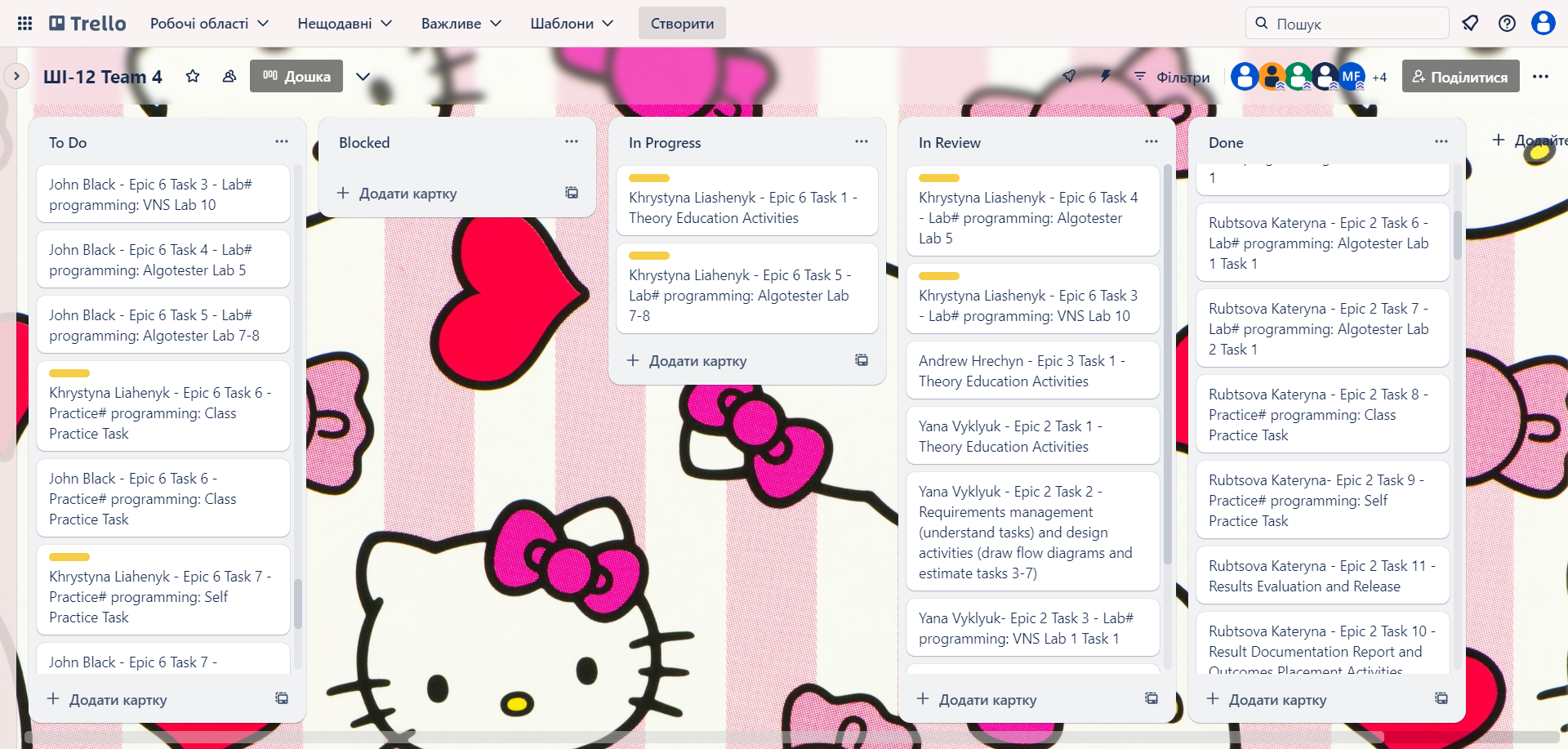


Рисунок 14 Trello на момент 2-ї зустрічі

* Скрін з 2-му коментарями від учасників команди на пул реквесті з Ревю Роботи

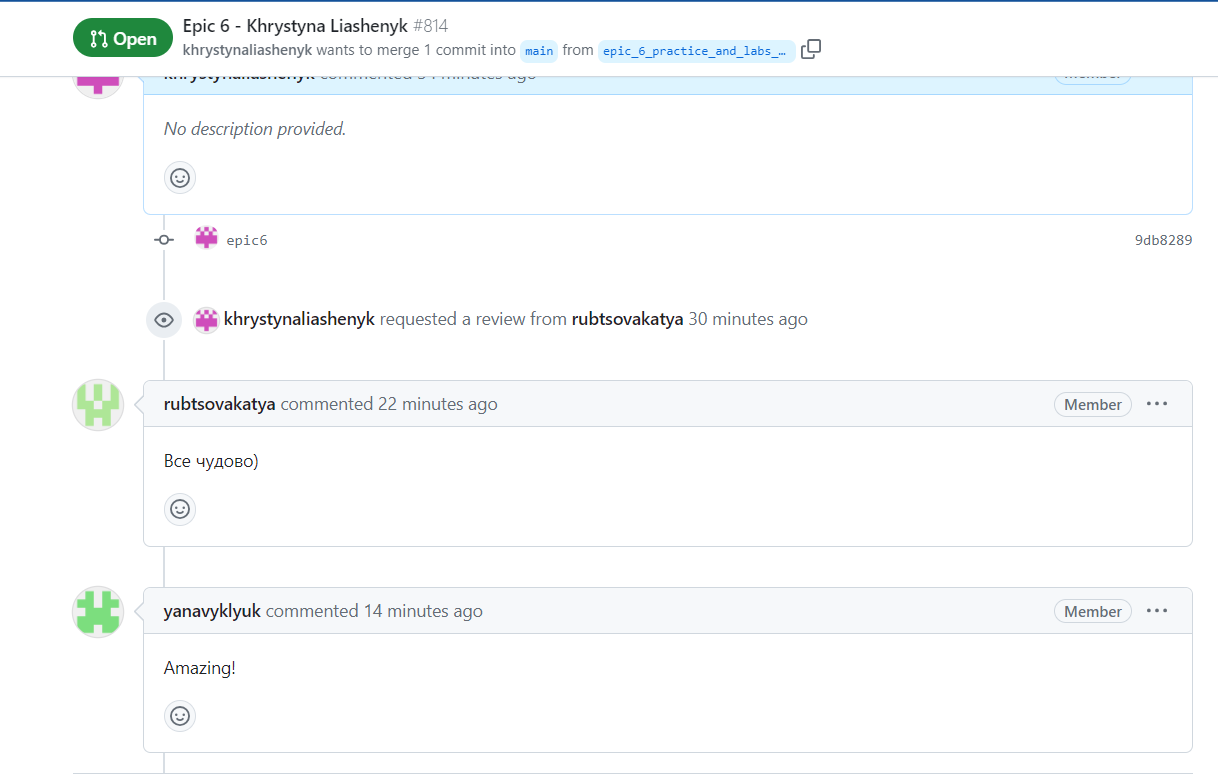


Рисунок 15 Коментарі 2 учасників команди до пул реквесту

# **Висновки:**

Виконання теоретичної та практичної роботи дозволило ознайомитися з ключовими видами динамічних структур даних, такими як стек, черга, список і дерево. Під час цього дослідження були вивчені основні операції, які можна виконувати з динамічними структурами даних, і були зміцнені навички реалізації алгоритмів на мові програмування C++. У процесі виконання практичної роботи були створені динамічні структури даних, такі як вектори з використанням динамічних масивів, списки та дерева з використанням власних вузлів. Ці структури даних були використані для вирішення задач, поставлених у рамках даної роботи, що продемонструвало їх важливість та допомогло краще зрозуміти їх внутрішню роботу.