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 \_\_\_\_\_\_\_\_\_\_»

***Виконав:***

студент групи ШІ-11

Дах Тарас Романович

# **Тема роботи:**

Вивчення базових динамічних структур. Робота з динамічними структурами. Алгоритми обробки динамічних структур.

# **Мета роботи:**

Вивчення основних понять про динамічні струкрури. Здобуття навичок в роботі з ними.

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Класи
* Тема №2: Робота з класами
* Тема №3: Однозвязні списки
* Тема №4: Двозвязні списки
* Тема №5\_: Бінарні дерева

1. Індивідуальний план опрацювання теорії:

* Тема №1: Класи
  + Джерела Інформації
    - <http://cpp.dp.ua/ponyattya-klasu/>
  + Що опрацьовано:
    - Збодуте розуміння, що таке класи
  + Статус: Ознайомлений
  + Початок опрацювання теми: 08.12.2023
  + Звершення опрацювання теми: 14.12.2023
* Тема №2: Робота з класами
  + Джерела Інформації:
    - <https://www.bestprog.net/uk/2018/03/25/c-classes-part-1-the-concept-of-class-declaring-a-data-type-class-the-object-of-the-class-classes-in-the-clr-encapsulation-of-data-in-a-class_ua/#q02>
  + Що опрацьовано:
    - Ознайомлений з основними методами роботи з класами
  + Статус: Ознайомлений частково
  + Початок опрацювання теми: 08.12.2023
  + Звершення опрацювання теми: 14.12.2023
* Тема №3: Однозвязні списки
  + Джерела Інформації:
    - <https://www.bestprog.net/uk/2022/02/11/c-linear-singly-linked-list-general-information-ua/>
    - <https://www.youtube.com/watch?v=-25REjF_atI&ab_channel=%D0%91%D0%BB%D0%BE%D0%B3%D0%B0%D0%BD%E2%A6%81%D0%A3%D1%80%D0%BE%D0%BA%D0%B8%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F>
  + Що опрацьовано:
    - Здобуте розуміння, що таке однозвязні списки
  + Статус: Ознайомлений
  + Початок опрацювання теми: 10.12.2023
  + Звершення опрацювання теми: 13.12.2023
* Тема №4: Двозвязні списки
  + Джерела Інформації:
    - <https://www.youtube.com/watch?v=QLzu2-_QFoE&t=734s&ab_channel=%D0%91%D0%BB%D0%BE%D0%B3%D0%B0%D0%BD%E2%A6%81%D0%A3%D1%80%D0%BE%D0%BA%D0%B8%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F>
  + Що опрацьовано:
    - Здобуто поняття, що таке двозвязні списки
  + Статус: Ознайомлений
  + Початок опрацювання теми: 09.12.2023
  + Звершення опрацювання теми: 14.12.2023
* Тема №5: Бінарні дерева.
  + Джерела Інформації:
    - <https://www.youtube.com/watch?v=QLzu2-_QFoE&t=734s&ab_channel=%D0%91%D0%BB%D0%BE%D0%B3%D0%B0%D0%BD%E2%A6%81%D0%A3%D1%80%D0%BE%D0%BA%D0%B8%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F>
  + Що опрацьовано:
    - Здобуто поняття, що таке бінарні дерева
  + Статус: Ознайомлений
  + Початок опрацювання теми: 10.12.2023
  + Звершення опрацювання теми: 15.12.2023

# **Виконання роботи:**

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

Завдання №1: Lab# programming: Algotester Lab 5

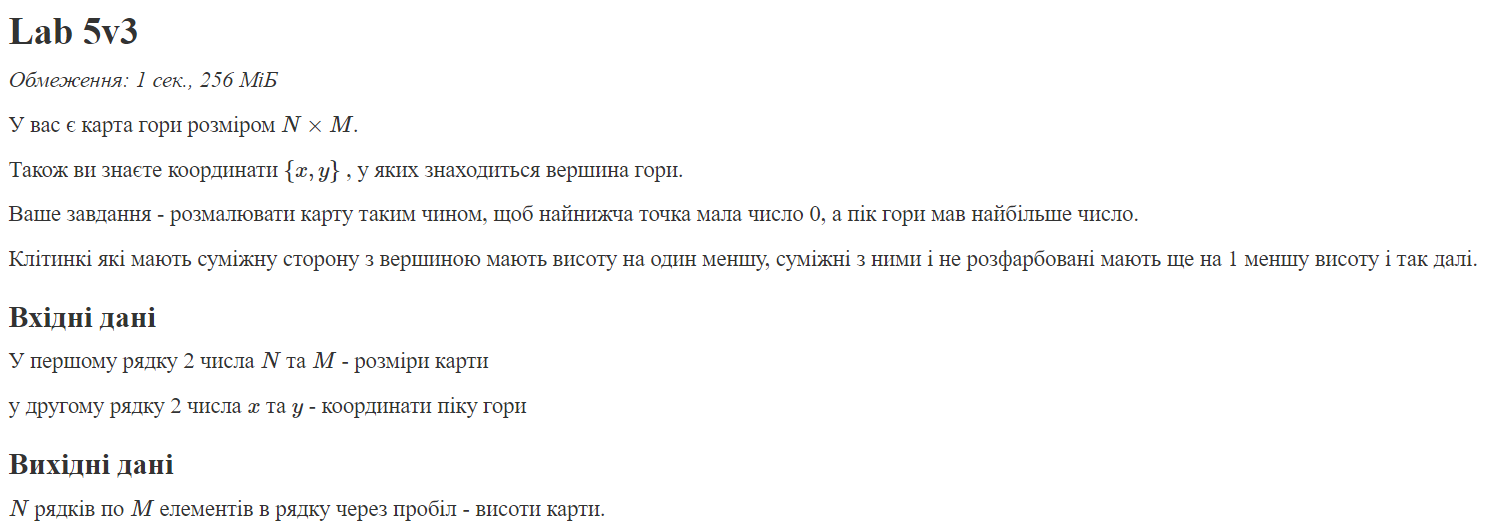


Рисунок 1 Lab# programming: Algotester Lab 5

Завдання №2:  Lab# programming: Algotester Lab 7-8

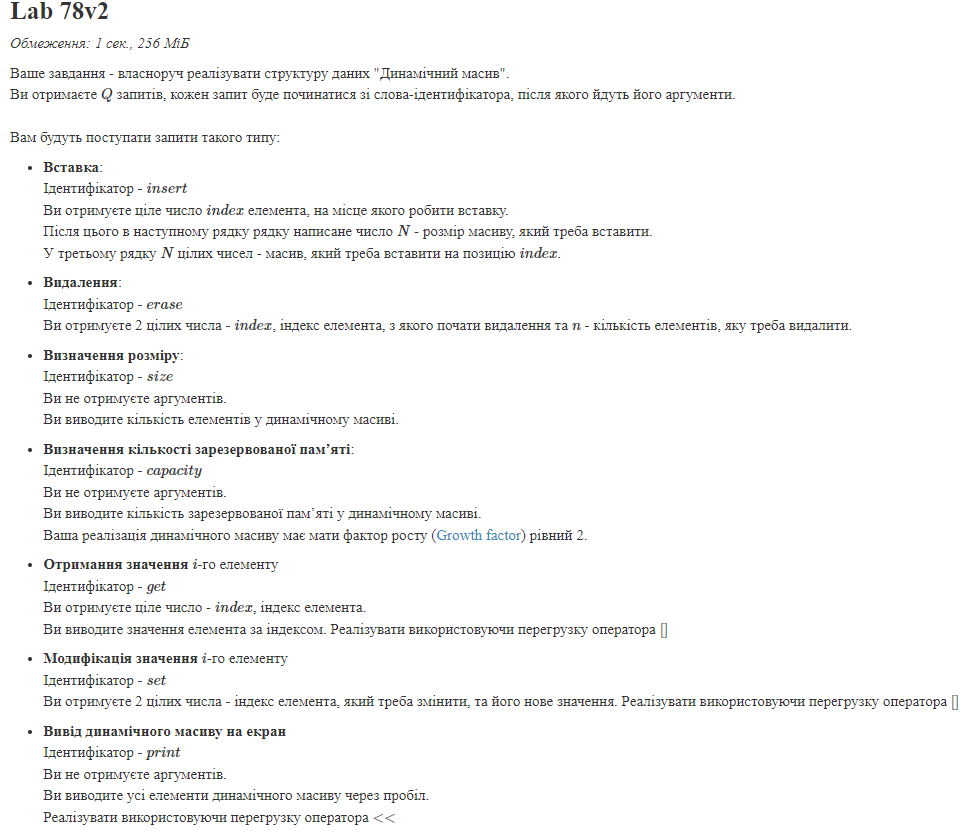


Рисунок 2 Lab# programming: Algotester Lab 7-8

Завдання №3: Lab# programming: VNS Lab 10

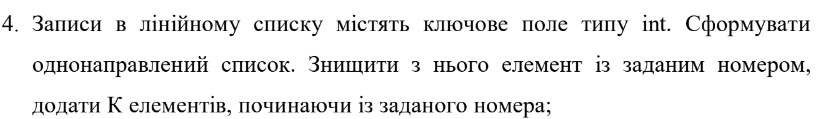


Рисунок 3 Lab# programming: VNS Lab 10

Завдання №4: Practice# programming:  Self Practice Task

Завдання №5: Practice# programming: Class Practice Task

* Варіант завдання
* 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

* Важливі деталі для врахування в імплементації програми

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

Програма №4: Practice# programming:  Self Practice Task

Блок-схема

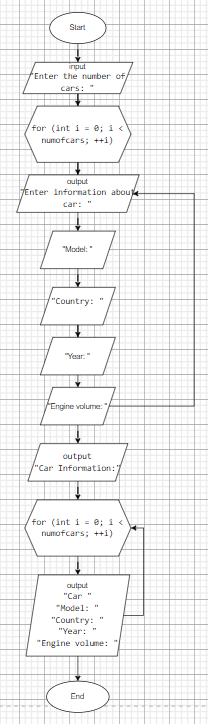


Рисунок Self Practice Task блоксхема

* Планований час на реалізацію 5 хв
* Важливі деталі для врахування в імплементації

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

Завдання №\_\_ Деталі по конфігурації середовища + скріншоти з підписами до скріншотів.

Підпис та № до блоку з скріншотами до конфігурації

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

Завдання №1: Lab# programming: Algotester Lab 5

#include <iostream>

#include <vector>

using namespace std;

int computeCellValue(int size\_n, int size\_m, int x, int y, int i, int j) {

    int xmax = (x <= size\_n / 2) ? 1 : size\_n;

    int ymax = (y <= size\_m / 2) ? 1 : size\_m;

    int max\_value = (size\_n - 1) + (size\_m - 1) - (abs(x - xmax) + abs(y - ymax));

    return max\_value - (abs(x - (i + 1)) + abs(y - (j + 1)));

}

void printMap(const vector<vector<int>>& map) {

    for (const auto& row : map) {

        for (int value : row) {

            cout << value << " ";

        }

        cout << "\n";

    }

}

int main() {

    while (true) {

        int size\_n, size\_m, x, y;

        cin >> size\_n >> size\_m >> x >> y;

        vector<vector<int>> map(size\_n, vector<int>(size\_m, 0));

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

            for (int j = 0; j < size\_m; j++) {

                map[i][j] = computeCellValue(size\_n, size\_m, x, y, i, j);

            }

        }

        printMap(map);

        char answer;

        cin >> answer;

        if (answer != 'y') {

            break;

        }

    }

    return 0;

}

Завдання №2 Lab# programming: Algotester Lab 7-8

#include <iostream>

#include <string>

#include <algorithm>

template <typename T>

class CustomArray {

private:

    T\* elements;

    long int size;

    long int capacity;

public:

    CustomArray() : size(0), capacity(1) {

        elements = new T[capacity];

    }

    ~CustomArray() {

        delete[] elements;

    }

    void insert(long int index, long int count, T\* values) {

        if (index < 0 || index > size)

            return;

        if (size + count >= capacity) {

            long int changedCapacity = capacity;

            while (size + count >= changedCapacity) {

                changedCapacity \*= 2;

            }

            T\* changedElements = new T[changedCapacity];

            size += count;

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

                changedElements[i] = elements[i];

            }

            delete[] elements;

            elements = changedElements;

            std::copy\_backward(elements + index, elements + size - count, elements + size);

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

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

            }

            capacity = changedCapacity;

        } else {

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

                elements[i + count] = elements[i];

            }

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

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

            }

            size += count;

        }

    }

    void erase(long int index, long int count) {

        long int deleteCount = std::min(count, size - index);

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

            elements[i] = elements[i + deleteCount];

        }

        size -= deleteCount;

    }

    long int Size() const {

        return size;

    }

    long int Capacity() const {

        return capacity;

    }

    void print() const {

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

            std::cout << elements[i] << " ";

        }

    }

    T& operator[](long int index) {

        return elements[index];

    }

};

int main() {

    using namespace std;

    long int Q;

    cin >> Q;

    CustomArray<long int> customArray;

    while (Q--) {

        string command;

        cin >> command;

        char action = command[0];

        long int index, count;

        long int\* values;

        switch (action) {

            case 'i':

                cin >> index >> count;

                values = new long int[count];

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

                    cin >> values[i];

                }

                customArray.insert(index, count, values);

                delete[] values;

                break;

            case 'e':

                cin >> index >> count;

                customArray.erase(index, count);

                break;

            case 's':

                if (command == "size") {

                    cout << customArray.Size() << endl;

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

                    cin >> index >> count;

                    if (index >= 0 && index < customArray.Size()) {

                        customArray[index] = count;

                    }

                }

                break;

            case 'c':

                cout << customArray.Capacity() << endl;

                break;

            case 'g':

                cin >> index;

                if (index >= 0 && index < customArray.Size()) {

                    cout << customArray[index] << endl;

                }

                break;

            case 'p':

                customArray.print();

                cout << endl;

                break;

            default:

        }

    }

    return 0;

}

Завдання №3 Lab# programming: VNS Lab 10

#include <iostream>

#include <fstream>

using namespace std;

// Структура елемента списку

struct Node {

    int key;

    Node\* next;

};

// Функція для створення порожнього списку

Node\* createList() {

    return nullptr;

}

// Функція для додавання елемента в список

void addElement(Node\*& head, int value) {

    Node\* newNode = new Node{value, nullptr};

    if (!head) {

        head = newNode;

    } else {

        Node\* current = head;

        while (current->next) {

            current = current->next;

        }

        current->next = newNode;

    }

}

// Функція для друку списку

void printList(Node\* head) {

    if (!head) {

        cout << "Список порожній" << endl;

    } else {

        Node\* current = head;

        while (current) {

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

            current = current->next;

        }

        cout << endl;

    }

}

// Функція для знищення елемента списку за заданим номером

void deleteElement(Node\*& head, int position) {

    if (!head) {

        cout << "Список порожній" << endl;

        return;

    }

    if (position == 1) {

        Node\* temp = head;

        head = head->next;

        delete temp;

    } else {

        Node\* current = head;

        for (int i = 1; i < position - 1 && current; ++i) {

            current = current->next;

        }

        if (!current || !current->next) {

            cout << "Помилка: елемента з номером " << position << " не існує" << endl;

            return;

        }

        Node\* temp = current->next;

        current->next = temp->next;

        delete temp;

    }

}

// Функція для запису списку у файл

void writeToFile(Node\* head, const string& filename) {

    ofstream file(filename);

    if (!file.is\_open()) {

        cerr << "Помилка відкриття файлу для запису" << endl;

        return;

    }

    Node\* current = head;

    while (current) {

        file << current->key << " ";

        current = current->next;

    }

    file.close();

}

// Функція для знищення списку

void destroyList(Node\*& head) {

    while (head) {

        Node\* temp = head;

        head = head->next;

        delete temp;

    }

}

// Функція для відновлення списку з файлу

Node\* restoreList(const string& filename) {

    ifstream file(filename);

    if (!file.is\_open()) {

        cout << "Помилка відкриття файлу для читання" << endl;

        return nullptr;

    }

    Node\* head = nullptr;

    int value;

    while (file >> value) {

        addElement(head, value);

    }

    file.close();

    return head;

}

// Функція для додавання K елементів, починаючи із заданого номера

void addKElements(Node\*& head, int position, int K) {

    Node\* current = head;

    for (int i = 1; i < position && current; ++i) {

        current = current->next;

    }

    if (!current) {

        cout << "Помилка: елемента з номером " << position << " не існує" << endl;

        return;

    }

    for (int i = 0; i < K; ++i) {

        int newValue;

        cout << "Введіть значення для нового елемента: ";

        cin >> newValue;

        addElement(current, newValue);

        current = current->next;

    }

}

int main() {

    Node\* myList = createList();

    // Додавання елементів у список

    addElement(myList, 23);

    addElement(myList, 20);

    addElement(myList, 30);

    addElement(myList, 45);

    addElement(myList, 56);

    // Друк списку

    cout << "Список після додавання елементів: ";

    printList(myList);

    // Запит користувача про номер елемента для видалення

    int deletePosition;

    cout << "Введіть номер елемента для видалення: ";

    cin >> deletePosition;

    // Знищення елемента за заданим номером

    deleteElement(myList, deletePosition);

    cout << "Список після видалення елемента за номером " << deletePosition << ": ";

    printList(myList);

    // Додавання K елементів, починаючи із заданого номера

    int K;

    cout << "Введіть кількість елементів для додавання: ";

    cin >> K;

    addKElements(myList, deletePosition, K);

    cout << "Список після додавання " << K << " елементів за номером " << deletePosition << ": ";

    printList(myList);

    // Запис списку у файл

    writeToFile(myList, "mylist.txt");

    // Знищення списку

    destroyList(myList);

    cout << "Список після знищення: ";

    printList(myList);

    // Відновлення списку з файлу

    myList = restoreList("mylist.txt");

    cout << "Відновлений список: ";

    printList(myList);

    // Знищення відновленого списку

    destroyList(myList);

    cout << "Список після знищення відновленого: ";

    printList(myList);

    return 0;

}

Завдання №4: Practice# programming:  Self Practice Task

#include <iostream>

#include <string>

using namespace std;

struct Cars {

    string model;

    int year;

    double enginevolume;

};

int main() {

    int numofcars;

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

    cin >> numofcars;

    Cars\* cars = new Cars[numofcars];

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

        cout <<"Enter information about car: " << (i + 1) << endl;

        cout << "Model: ";

        cin.ignore();

        getline(cin, cars[i].model);

        cout << "Year: ";

        cin >> cars[i].year;

        cout << "Engine volume: ";

        cin >> cars[i].enginevolume;

    }

    cout << endl;

    cout << "Car Information:" << endl;

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

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

        cout << "Mode: " << cars[i].model << endl;

        cout << "Year: " << cars[i].year << endl;

        cout << "Engine volume: " << cars[i].enginevolume << endl;

    }

    return 0;

}

Підпис та № до блоку з кодом програми

Завдання №5(task 1): Practice# programming: Class Practice Task

#include <iostream>

struct Node {

    int data;

    Node\* next;

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

};

Node\* reverse(Node\* head) {

    Node\* prev = nullptr;

    Node\* current = head;

    Node\* next = nullptr;

    while (current != nullptr) {

        next = current->next;

        current->next = prev;

        prev = current;

        current = next;

    }

    return prev;

}

void printList(Node\* head) {

    Node\* current = head;

    while (current != nullptr) {

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

        current = current->next;

    }

    std::cout << std::endl;

}

int main() {

    Node\* originalList = new Node(1);

    originalList->next = new Node(2);

    originalList->next->next = new Node(3);

    originalList->next->next->next = new Node(4);

    std::cout << "Вихідний список: ";

    printList(originalList);

    Node\* reversedList = reverse(originalList);

    std::cout << "Обернений список: ";

    printList(reversedList);

    // Очищення пам'яті

    while (reversedList != nullptr) {

        Node\* temp = reversedList;

        reversedList = reversedList->next;

        delete temp;

    }

    return 0;

}

Підпис та № до блоку з кодом програми

Завдання №5(task 2): Practice# programming: Class Practice Task

#include <iostream>

// Структура вузла списку

struct Node {

    int data;

    Node\* next;

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

};

// Функція порівняння двох списків

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

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

        // Порівнюємо дані в кожному вузлі

        if (h1->data != h2->data) {

            return false;

        }

        // Переходимо до наступних вузлів

        h1 = h1->next;

        h2 = h2->next;

    }

    // Якщо один зі списків закінчується раніше іншого, то вони не рівні

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

}

int main() {

    // Створення першого списку: 1 -> 2 -> 3

    Node\* head1 = new Node(1);

    head1->next = new Node(2);

    head1->next->next = new Node(3);

    // Створення другого списку: 1 -> 2 -> 3

    Node\* head2 = new Node(1);

    head2->next = new Node(2);

    head2->next->next = new Node(3);

    // Спроба порівняння списків

    if (compare(head1, head2)) {

        std::cout << "Списки рівні." << std::endl;

    } else {

        std::cout << "Списки не рівні." << std::endl;

    }

    // Очистка пам'яті (звільнення вузлів)

    while (head1 != nullptr) {

        Node\* temp = head1;

        head1 = head1->next;

        delete temp;

    }

    while (head2 != nullptr) {

        Node\* temp = head2;

        head2 = head2->next;

        delete temp;

    }

    return 0;

}

Підпис та № до блоку з кодом програми

Завдання №5(task 3): Practice# programming: Class Practice Task

#include <iostream>

struct Node {

    int data;

    Node \*next;

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

};

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

    Node \*result = nullptr;

    Node \*current = nullptr;

    int carry = 0;

    while (n1 || n2 || carry) {

        int sum = (n1 ? n1->data : 0) + (n2 ? n2->data : 0) + carry;

        carry = sum / 10;

        Node \*newNode = new Node(sum % 10);

        if (!result) {

            result = newNode;

            current = result;

        } else {

            current->next = newNode;

            current = newNode;

        }

        if (n1) n1 = n1->next;

        if (n2) n2 = n2->next;

    }

    return result;

}

void printList(Node \*head) {

    while (head) {

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

        head = head->next;

    }

    std::cout << std::endl;

}

int main() {

    Node \*num1 = new Node(9);

    num1->next = new Node(7);

    num1->next->next = new Node(3);

    Node \*num2 = new Node(2);

    num2->next = new Node(4);

    num2->next->next = new Node(8);

    std::cout << "Number 1: ";

    printList(num1);

    std::cout << "Number 2: ";

    printList(num2);

    Node \*result = add(num1, num2);

    std::cout << "Sum: ";

    printList(result);

    delete num1;

    delete num2;

    delete result;

    return 0;

}

Підпис та № до блоку з кодом програми

Завдання №5(task 4): Practice# programming: Class Practice Task

#include <iostream>

// Ваша структура TreeNode

struct TreeNode {

    int val;

    TreeNode \*left;

    TreeNode \*right;

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

};

// Наша нова структура для роботи з деревом

struct TreeProcessor {

    static TreeNode\* createMirrorFlippedTree(TreeNode \*root) {

        if (root == nullptr) {

            return nullptr;

        }

        TreeNode \*newRoot = new TreeNode(root->val);

        newRoot->right = createMirrorFlippedTree(root->left);

        newRoot->left = createMirrorFlippedTree(root->right);

        return newRoot;

    }

    static void printTree(TreeNode \*root) {

        if (root == nullptr) {

            return;

        }

        printTree(root->left);

        std::cout << root->val << " ";

        printTree(root->right);

    }

};

int main() {

    // Приклад використання

    TreeNode \*originalTree = new TreeNode(1);

    originalTree->left = new TreeNode(2);

    originalTree->right = new TreeNode(3);

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

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

    std::cout << "Original Tree: ";

    TreeProcessor::printTree(originalTree);

    std::cout << std::endl;

    TreeNode \*flippedTree = TreeProcessor::createMirrorFlippedTree(originalTree);

    std::cout << "Flipped Tree: ";

    TreeProcessor::printTree(flippedTree);

    std::cout << std::endl;

    return 0;

}

Підпис та № до блоку з кодом програми

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

Завдання №1: Lab# programming: Algotester Lab 5

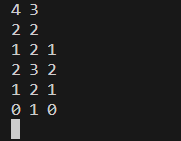


Рисунок 5 Lab# programming: Algotester Lab 5

Час затрачений на виконання завдання 2 години

Завдання №2:  Lab# programming: Algotester Lab 7-8

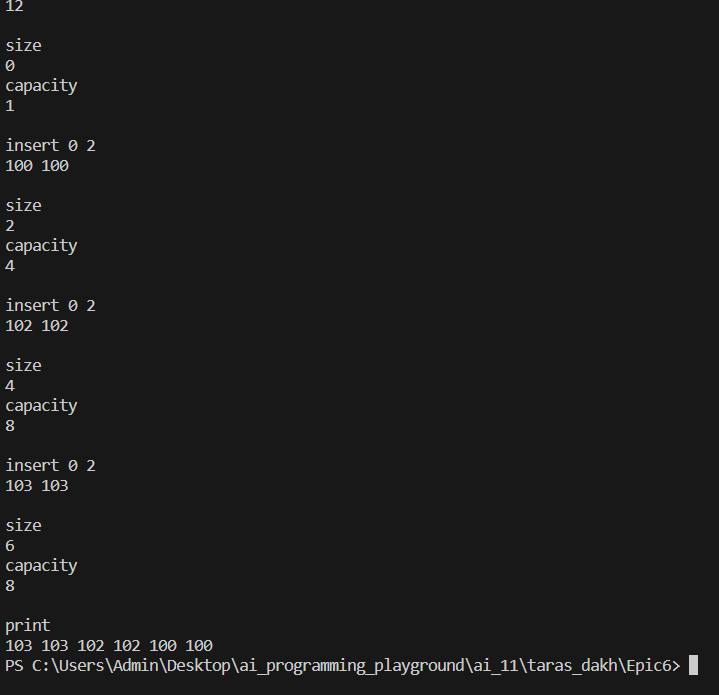


Рисунок 6 Lab# programming: Algotester Lab 7-8

Час затрачений на виконання завдання 6 годин

Завдання №3: Lab# programming: VNS Lab 10

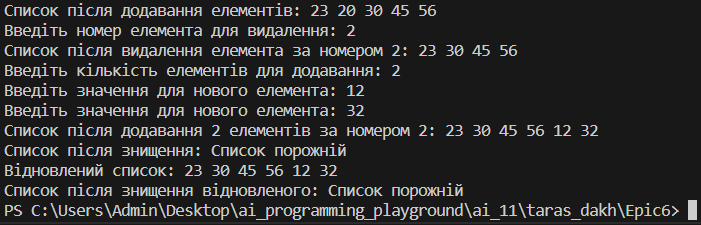


Рисунок 7 Lab# programming: VNS Lab 10

Час затрачений на виконання завдання 4 години

Завдання №4: Practice# programming:  Self Practice Task

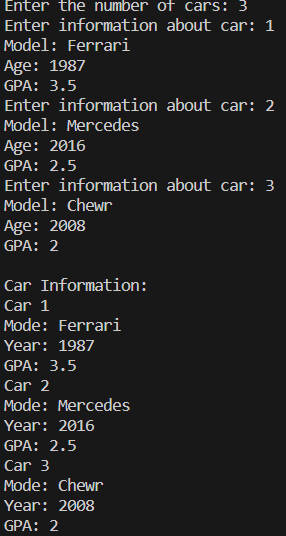


Рисунок 8 Practice# programming: Self Practice Task

Час затрачений на виконання завдання 40 хвилин

Завдання №5(task 1): Practice# programming: Class Practice Task

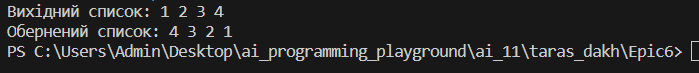


Рисунок 9 : Practice# programming: Class Practice Task

Час затрачений на виконання завдання 1 година

Завдання №5(task 2): Practice# programming: Class Practice Task



Рисунок 10 Practice# programming: Class Practice Task2

Час затрачений на виконання завдання 2 години

Завдання №5(task 3): Practice# programming: Class Practice Task

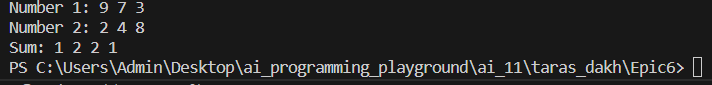


Рисунок 11 Practice# programming: Class Practice Task3

Час затрачений на виконання завдання 2 години

Завдання №5(task 4): Practice# programming: Class Practice Task

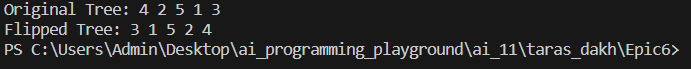


Рисунок 12 Practice# programming: Class Practice Task 4

Час затрачений на виконання завдання 1 година

# **Висновки:**

Під час виконання даної роботи я дізнався, що таке таке динамічні структури, які є їхні види, і для чого вони використовуються. Для цього я створив елементарні програми для виконання певних алгоритмів, з їхнім використанням,

ПР: <https://github.com/artificial-intelligence-department/ai_programming_playground/pull/1088>