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, Дерево). Алгоритми обробки динамічних структур.»

***Виконала:***

студентка групи ШІ-13

Мостова Віта Любомирівна

# **Тема роботи:** *Динамічні структури (Черга, Стек, Списки, Дерево). Алгоритми обробки динамічних структур.*

# **Мета роботи:** *Навчитись створювати і працювати з динамічними структурами.*

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Види динамічних структур
* Тема №2: Одно- та двозв’язні списки
* Тема №3: Бінарне дерево

1. Індивідуальний план опрацювання теорії:

* Тема №1: Види динамічних структур
  + Джерела Інформації
    - <https://youtu.be/eSxLVD5vfqM?si=8OiPGfRKIazU-N8z>
    - <https://youtu.be/jH9ZE5uAal8?si=xdq9aK04b9Brv7Eq>
    - <https://youtu.be/5uAwaQL99cY?si=fn3dSlgevFYkswRp>
  + Що опрацьовано: Опрацьовано матеріали надані вище. Розібрано в чому різниця між різними динамічними структурами
  + Статус: Ознайомлена
  + Початок опрацювання теми: 6.12.2023
  + Звершення опрацювання теми: 14.12.2023
* Тема №2: Одно- та двозв’язні списки.
  + Джерела Інформації:
    - <https://youtu.be/N6dOwBde7-M?si=UrtPdTS4PvXcXLJc>
    - <https://youtu.be/vcQIFT79_50?si=9N_GPcCRgntvKedH>
  + Що опрацьовано: Опрацьовано матеріали надані вище. Попрактикувалась в імплементації linked list
  + Статус: Ознайомлена
  + Початок опрацювання теми: 6.12.2023
  + Звершення опрацювання теми: 14.12.2023
* Тема №3: Бінарне дерево
  + Джерела Інформації:
    - <https://youtu.be/Krr9J-_7IS0?si=A9QJWtCCxu9er9xI>
    - [https://youtu.be/qBFzNW0ALxQ?si=RgZR7\_W-PYHTQLz0](https://youtu.be/qBFzNW0ALxQ?si=RgZR7_W-PYHTQLz0%20)
    - <https://youtu.be/_IhTp8q0Mm0?si=xzi_Eb0rwYpFt3ib>
  + Що опрацьовано: Опрацьовано матеріали надані вище.
  + Статус: Ознайомлена
  + Початок опрацювання теми: 6.12.2023
  + Звершення опрацювання теми: 14.12.2023

# **Виконання роботи:**

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

Завдання №1 **«Лабораторна робота №10 VNS»**

* Варіант 18
* Потрібно виконати наступні пункти:

1. Створення списку.

2. Додавання елемента в список (у відповідності із варіантом).

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

4. Друк списку.

5. Запис списку у файл.

6. Знищення списку.

7. Відновлення списку з файлу.

* Записи в лінійному списку містять ключове поле типу \*char (рядок

символів). Сформувати двонаправлений список. Знищити елемент із заданим

ключем. Додати К елементів у початок списку

Завдання №2 **«Algotester Lab 5v3»**

* Дано карту гори розміром N×M та координати вершини{x,y}.
* Необхідно розмалювати карту таким чином, щоб найнижча точка мала число 0, а пік гори мав найбільше число. Клітинки які мають суміжну сторону з вершиною мають висоту на один меншу, суміжні з ними і не розфарбовані мають ще на 1 меншу висоту і так далі

Завдання №3 **«Algotester 78v3»**

* Реалізувати структуру даних "Двійкове дерево пошуку".На вхід отримується Q запитів, кожен запит буде починатися зі слова-ідентифікатора, після якого йдуть його параметри.
* Запити:
* Вставка
* **Пошук**
* **Визначення розміру**
* Вивід дерева на екран

Завдання №4 **«Practice Task 1»**

* Реалізувати метод реверсу списку: Node\* reverse(Node \*head);
* Необхідно використовувати цілочисельні значення в списку;

Завдання №5 **«Practice Task 2»**

* Реалізувати метод порівняння bool compare(Node \*h1, Node \*h2);
* Необхідно використовувати цілочисельні значення в списку;
* Реалізувати функцію, яка ітеративно проходиться по обох списках і порівнює дані в кожному вузлі. Якщо дані не збігаються або довжина списків різна, то функція повертає false

Завдання №6 **«Practice Task 3»**

* Реалізувати метод додавання Node\* add(Node \*n1, Node \*n2);
* Необхідно 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.

Завдання №7 **«Practice Task 4»**

* Реалізувати метод відзеркалення TreeNode \*create\_mirror\_flip(TreeNode \*root);
* Необхідно використовувати цілі числа для значень у вузлах дерева
* Реалізувати функцію, яка проходить по всіх вузлах дерева і міняє місцями праву і ліву вітки дерева. Функція повертає нове дерево

Завдання №8 **«Practice Task 5»**

* Реалізувати метод void tree\_sum(TreeNode \*root);
* Необхідно використовувати цілі числа для значень у вузлах дерева
* Реалізувати функцію, яка ітеративно проходить по бінарному дереві і записує у батьківський вузол суму значень підвузлів.

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

Програма №1 **«Лабораторна робота №10 VNS»**

* Планований час на реалізацію: до 3 годин

Програма №2 **«Algotester Lab 5v3»**

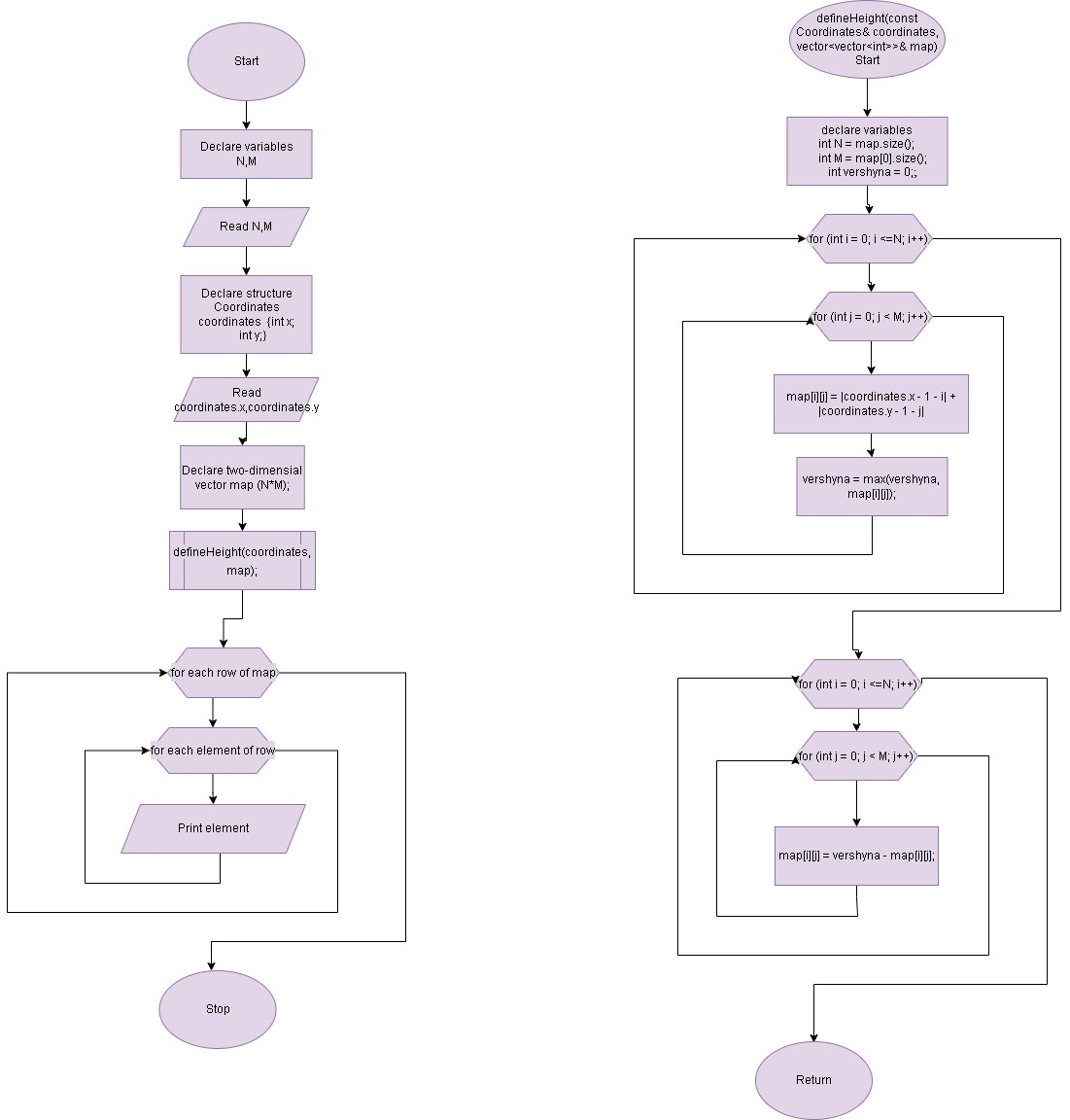


Рисунок 1: Блок-схема до завдання №2

* Планований час на реалізацію : до 2 годин

Програма №3 **«Algotester 78v3»**

* Планований час на реалізацію: до 2 годин

Програма №4 **«Practice Task 1»**

* Планований час на реалізацію: до 1 години

Програма №5 **«Practice Task 2»**

* Планований час на реалізацію: до 1 години

Програма №6 **«Practice Task 3»**

* Планований час на реалізацію: до 1 години

Програма №7 **«Practice Task 4»**

* Планований час на реалізацію: до 1 години

Програма №8 **«Practice Task 5»**

* Планований час на реалізацію: до 1 години

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

Завдання №1 **«Лабораторна робота №10 VNS»**

[**Pull request**](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/984/files%23diff-30edbb8ca81f9efb977be14b0ecd853dae6f90221130587de890a9698e2685e5)

#include <iostream>

#include <fstream>

using namespace std;

struct Node {

    string data;

    Node\* next;

    Node\* prev;

};

class LinkedList {

  private:

    Node\* head; Node\* end;

  public:

    LinkedList(){

      head = NULL;

    }

     void insert\_at\_the\_beginning(string new\_data) {

        Node\* newNode = new Node;

        newNode->data = new\_data;

        newNode->next = head;

        newNode->prev = NULL;

        if (head != NULL)

            head->prev = newNode;

        head = newNode;

    }

    void print() {

      Node\* temp = head;

      if(temp != NULL) {

        while(temp != NULL) {

          cout<<temp->data<<" ";

          temp = temp->next;

        }

        cout<<endl;

      } else {

        cout<<"The list is empty.\n";

      }

    }

    void delete\_by\_key(string key) {

    Node\* curr = head;

    while (curr != NULL && curr->data != key) {

        curr = curr->next;

    }

    if (curr != NULL) {

        if (curr->prev != NULL) { //якшо вузол не голова

            curr->prev->next = curr->next;

        } else {

             head = curr->next;

        }

         if (curr->next != NULL) {

            curr->next->prev = curr->prev;

        }

        delete curr;

    } else {

        cout << "Element not found:(" << endl;

    }

}

    void write\_to\_file(const string& file\_name) {

        ofstream writeFile(file\_name);

        Node\* curr = head;

        while (curr != nullptr) {

            writeFile << curr->data << " ";

            curr = curr->next;

    }

        writeFile.close();

}

   void delete\_all\_nodes() {

    while (head != NULL) {

        Node\* temp = head;

        head = head->next;

        delete temp;

    }

    cout << "All nodes are deleted successfully.\n";

}

  void read\_from\_file(const string& file\_name) {

        ifstream readFile(file\_name);

        if (!readFile.is\_open()) {

            cout << "This file can not be opened:(" << endl;

            return;

        }

        delete\_all\_nodes();

        string value;

        while (readFile >> value) {

            insert\_at\_the\_beginning(value);

        }

        readFile.close();

        if (head != NULL) {

            cout << "List has been read from the file." << endl;

        } else {

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

        }

    }

};

int main() {

    LinkedList DoubleLinkedList;

    DoubleLinkedList.insert\_at\_the\_beginning("hello");

    DoubleLinkedList.insert\_at\_the\_beginning("car");

    DoubleLinkedList.insert\_at\_the\_beginning ("wallpaper");

    DoubleLinkedList.insert\_at\_the\_beginning ("shop");

    DoubleLinkedList.insert\_at\_the\_beginning ("phone");

    cout<<"The list consists of: ";

    DoubleLinkedList.print();

    DoubleLinkedList.delete\_by\_key("shop");

    cout<<"The list after deleting key element: ";

    DoubleLinkedList.print();

    DoubleLinkedList.insert\_at\_the\_beginning ("day");

    cout<<"The list after insertig a new element: ";

    DoubleLinkedList.print();

    DoubleLinkedList.write\_to\_file("list.txt");

    DoubleLinkedList.read\_from\_file("list.txt");

    DoubleLinkedList.print();

    return 0;

}

## *Програмний код до завдання №1*

Завдання №2 **«Algotester Lab 5v3»**

[**Pull request**](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/984/files%23diff-58f47abc27e2ad10187132354340a53af5a3971e009910f2332051dd9d4a9ea7)

#include <iostream>

#include <cmath>

#include <vector>

using namespace std;

struct Coordinates {

    int x;

    int y;

};

void defineHeight(const Coordinates& coordinates, vector<vector<int>>& map);

int main() {

    int N, M;

    cin >> N >> M;

    Coordinates coordinates;

    cin >> coordinates.x >> coordinates.y;

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

    defineHeight(coordinates, map);

    for (const auto& row : map) {

        for (int element : row) {

            cout << element << " ";

        }

        cout << endl;

    }

    return 0;

}

void defineHeight(const Coordinates& coordinates, vector<vector<int>>& map) {

    int N = map.size();

    int M = map[0].size();

    int vershyna = 0;

    for (int i = 0; i < N; i++) {

        for (int j = 0; j < M; j++) {

            map[i][j] = abs(coordinates.x - 1 - i) + abs(coordinates.y - 1 - j);

            vershyna = max(vershyna, map[i][j]);

        }

    }

    for (int i = 0; i < N; i++) {

        for (int j = 0; j < M; j++) {

            map[i][j] = vershyna - map[i][j];

        }

    }

}

## *Програмний код до завдання №2*

Завдання №3 **«Algotester 78v3»**

[**Pull request**](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/984/files%23diff-a0f4776fca186f31b9442c1c58bb7bcac12480ef8f0e84ff56dfb5e03d3c45a9)

#include <iostream>

#include <vector>

#include <string>

using namespace std;

class TreeNode

{

public:

    int data;

    TreeNode \*lChild;

    TreeNode \*rChild;

    TreeNode(int val)

    {

        this->data = val;

        lChild = nullptr;

        rChild = nullptr;

    }

};

class Tree

{

private:

    TreeNode \*root;

    int treeSize;

    TreeNode \*aNode(TreeNode \*node, int val)

    {

        if (node == nullptr)

        {

            treeSize++;

            return new TreeNode(val);

        }

        if (val < node->data)

        {

            node->lChild = aNode(node->lChild, val);

        }

        else if (val > node->data)

        {

            node->rChild = aNode(node->rChild, val);

        }

        return node;

    }

    bool fNode(TreeNode \*node, int val) const

    {

        if (node == nullptr)

        {

            return false;

        }

        if (val == node->data)

        {

            return true;

        }

        else if (val < node->data)

        {

            return fNode(node->lChild, val);

        }

        else

        {

            return fNode(node->rChild, val);

        }

    }

    int calcSize(TreeNode \*node) const

    {

        if (node == nullptr)

        {

            return 0;

        }

        return 1 + calcSize(node->lChild) + calcSize(node->rChild);

    }

    void printTree(TreeNode \*node) const

    {

        if (node != nullptr)

        {

            printTree(node->lChild);

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

            printTree(node->rChild);

        }

    }

public:

    Tree() : root(nullptr), treeSize(0) {}

    void insert(int val)

    {

        root = aNode(root, val);

    }

    bool contains(int val) const

    {

        return fNode(root, val);

    }

    int size() const

    {

        return calcSize(root);

    }

    friend ostream &operator<<(ostream &os, Tree &bst)

    {

        bst.printTree(bst.root);

        return os;

    }

};

int main()

{

    int Q;

    cin >> Q;

    Tree bst;

    while (Q--)

    {

        string operation;

        cin >> operation;

        if (operation == "insert")

        {

            int value;

            cin >> value;

            bst.insert(value);

        }

        else if (operation == "contains")

        {

            int value;

            cin >> value;

            if (bst.contains(value))

            {

                cout << "Yes" << endl;

            }

            else

            {

                cout << "No" << endl;

            }

        }

        else if (operation == "size")

        {

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

        }

        else if (operation == "print")

        {

            cout << bst << endl;

        }

    }

    return 0;

}

## *Програмний код до завдання №3*

Завдання №4 **«Practice Task 1»**

[**Pull request**](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/984/files%23diff-7546587a7044f26595e1eb3603b97e8ef319e1dffff80bd292c824664878ef6e)

#include <iostream>

using namespace std;

struct Node {

    int data;

    Node\* next;

};

Node\* Insert(Node \*head,int data) {

    Node \*temp1 = new Node;

    temp1 -> data = data;

    temp1 -> next = nullptr;

    if (head == nullptr) head = temp1;

    else {

        Node \*temp2= head;

        while(temp2 -> next != nullptr) {

            temp2 = temp2->next;

        }

        temp2 -> next = temp1;

    }

    return head;

};

Node \*Reverse(Node \*head) {

    Node \*temp1 = new Node;

    Node \*temp2 = new Node;

    if (head->next == nullptr) {

        return head;

    }

    else {

        temp1 =Reverse(head->next);

        temp2 =head->next;

        temp2->next = head;

        head->next = nullptr;

    }

    return temp1;

};

void PrintList(Node \*p) {

    if (p==nullptr) return;

    cout << p->data << " ";

    PrintList(p->next);

};

int main(){

    Node \*head = nullptr;

    head = Insert(head,1);

    head = Insert(head,2);

    head = Insert(head,3);

    head = Insert(head,4);

    PrintList(head);

    cout << endl;

    head = Reverse(head);

    PrintList(head);

    cout << endl;

    return 0;

}

## *Програмний код до завдання №4*

Завдання №5 **«Practice Task 2»**

[**Pull request**](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/984/files%23diff-9c3e4eb701d520b05994bab4536afe378cbde3e3bfe9de685f60792ef69a5f7e)

#include <iostream>

using namespace std;

struct Node {

    int data;

    Node\* next;

};

Node\* Insert(Node \*head,int data) {

    Node \*temp1 = new Node;

    temp1 -> data = data;

    temp1 -> next = nullptr;

    if (head == nullptr) head = temp1;

    else {

        Node \*temp2= head;

        while(temp2 -> next != nullptr) {

            temp2 = temp2->next;

        }

        temp2 -> next = temp1;

    }

    return head;

};

void PrintList(Node \*p) {  // Node \*p is a local variable

    if (p==nullptr) return;  //Exit condition

    cout << p->data << " ";       // First print the value in the node

    PrintList(p->next);              // Recursive call

};

bool Compare(Node \*head1, Node \*head2){

    Node\* h1=head1;

    Node\* h2=head2;

    while(h1 || h2){

        if(!h1 || !h2) return false;

        if(h1 ->data != h2 ->data)return false;

        h1=h1->next;

        h2=h2->next;

    }

return true;

}

int main(){

    Node\* head1 = nullptr;

    head1 = Insert(head1, 2);

    head1 = Insert(head1, 4);

    head1 = Insert(head1, 6);

    Node\* head2 = nullptr;

    head2 = Insert(head2, 2);

    head2 = Insert(head2, 4);

    head2 = Insert(head2, 6);

    cout << "List 1: ";

    PrintList(head1);

    cout << endl;

    cout << "List 2: ";

    PrintList(head2);

    cout << endl;

    if (Compare(head1, head2)) {

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

    } else {

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

    }

    return 0;

}

Success;

}

## *Програмний код до завдання №5*

Завдання №6 **«Practice Task 3»**

[**Pull request**](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/984/files%23diff-18d739f8d607827710bec86e99b26c0e035e2f57e6027395f4d26e2e099ed97a)

#include <iostream>

using namespace std;

struct Node {

    int data;

    Node\* next;

};

Node\* Insert(Node\* head, int data) {

    if (data < 0 || data > 9) {

        cout << "Invalid input. Please enter a number between 0 and 9." << endl;

        return head;

    }

    Node\* temp1 = new Node;

    temp1->data = data;

    temp1->next = nullptr;

    if (head == nullptr)

        head = temp1;

    else {

        Node\* temp2 = head;

        while (temp2->next != nullptr) {

            temp2 = temp2->next;

        }

        temp2->next = temp1;

    }

    return head;

}

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

    Node\* result = nullptr;

    Node\* ostNodeInResult = nullptr;

    int perenis = 0;

    for (; n1 || n2 || perenis; n1 = n1 ? n1->next : n1, n2 = n2 ? n2->next : n2) {

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

    perenis = sum / 10;

    Node\* newNode = new Node;

    newNode->data = sum % 10;

    newNode->next = nullptr;

    if (!result) {

        result = newNode;

        ostNodeInResult = result;

    } else {

        ostNodeInResult->next = newNode;

        ostNodeInResult = ostNodeInResult->next;

    }

}

    return result;

}

void PrintList(Node \*p) {  // Node \*p is a local variable

    if (p==nullptr) return;  //Exit condition

    cout << p->data << " ";       // First print the value in the node

    PrintList(p->next);              // Recursive call

};

int main() {

    Node\* num1 = nullptr;

    Node\* num2 = nullptr;

    num1 = Insert(num1, 9);

    num1 = Insert(num1, 7);

    num1 = Insert(num1, 3);

    num2 = Insert(num2, 1);

    num2 = Insert(num2, 6);

    num2 = Insert(num2, 9);

cout << "Number 1: ";

    PrintList(num1);

    cout<<endl;

    cout << "Number 2: ";

    PrintList(num2);

    cout<<endl;

    Node\* sum = add(num1, num2);

    cout << "Sum: ";

    PrintList(sum);

    return 0;

}

## *Програмний код до завдання №6*

Завдання №7 **«Practice Task 4»**

[**Pull request**](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/984/files%23diff-76d055c8131b07b6c936e761fba76a68a7296139c1c9b31415d308ee07875e27)

#include <iostream>

using namespace std;

struct TreeNode

{

    int data;

    TreeNode \*lChild;

    TreeNode \*rChild;

    TreeNode(int val) : data(val), lChild(nullptr), rChild(nullptr) {}

};

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

{

    if (root == nullptr)

    {

        return nullptr;

    }

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

    newRoot->rChild = create\_mirror\_flip(root->lChild);

    newRoot->lChild = create\_mirror\_flip(root->rChild);

    return newRoot;

}

void printTree(TreeNode \*node)

{

    if (node != nullptr)

    {

        cout << node->data << " ";

        printTree(node->lChild);

        printTree(node->rChild);

    }

}

int main()

{

    TreeNode \*root = new TreeNode(10);

    root->lChild = new TreeNode(6);

    root->lChild->rChild = new TreeNode(7);

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

    root->rChild = new TreeNode(12);

    cout << "Original Tree:" << endl;

    printTree(root);

    cout << endl;

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

    cout << "Mirrored Tree:" << endl;

    printTree(mirroredTree);

    cout << endl;

    return 0;

}

## *Програмний код до завдання №7*

Завдання №8 **«Practice Task 5»**

[**Pull request**](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/984/files%23diff-0881df6e502fcfb5c6f3a519caef91780b4944c292de60978a7d5c4ac64e4403)

#include <iostream>

using namespace std;

struct TreeNode

{

    int data;

    TreeNode \*lChild;

    TreeNode \*rChild;

    TreeNode(int val) : data(val), lChild(nullptr), rChild(nullptr) {}

};

int tree\_sum(TreeNode\* root)

{

    if (root == nullptr)

    {

        return 0;

    }

    int leftSum = tree\_sum(root->lChild);

    int rightSum = tree\_sum(root->rChild);

    if (root->lChild|| root->rChild)

    {

        root->data += leftSum + rightSum;

    }

    return root->data;

}

void printTree(TreeNode\* node)

{

    if (node != nullptr)

    {

        cout << node->data << " ";

        printTree(node->lChild);

        printTree(node->rChild);

    }

}

int main()

{

    TreeNode\* root = new TreeNode(8);

    root->lChild = new TreeNode(6);

    root->rChild = new TreeNode(10);

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

    root->lChild->rChild = new TreeNode(7);

    root->rChild->lChild = new TreeNode(9);

    root->rChild->rChild = new TreeNode(11);

    cout << "Original Tree:" << endl;

    printTree(root);

    cout << endl;

    tree\_sum(root);

    cout << "Updated Tree:" << endl;

    printTree(root);

    return 0;

}

## *Програмний код до завдання №8*

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

Завдання №1 **«Лабораторна робота №10 VNS»**

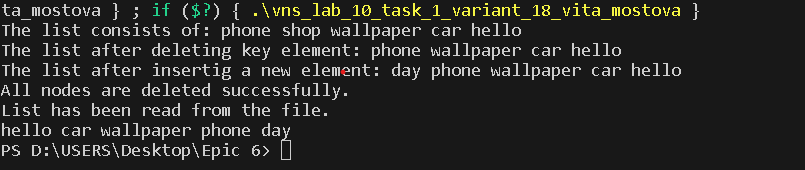


Рисунок 2:Приклад виконання завдання 1

Час затрачений на виконання завдання: до 3 години

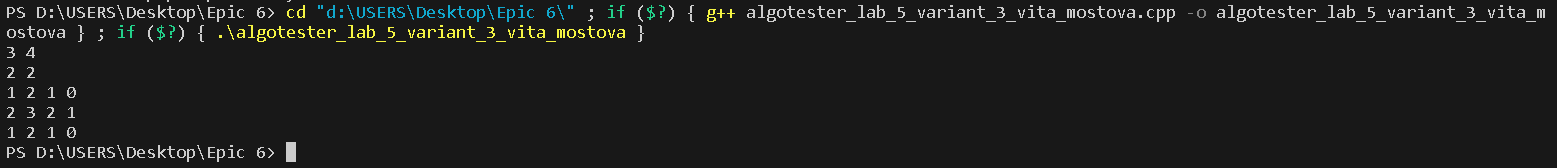
Завдання №2 **«Algotester Lab 5v3»**

Рисунок 3:Приклад виконання завдання 2



Скрін з алготестера

Час затрачений на виконання завдання: до 1 години

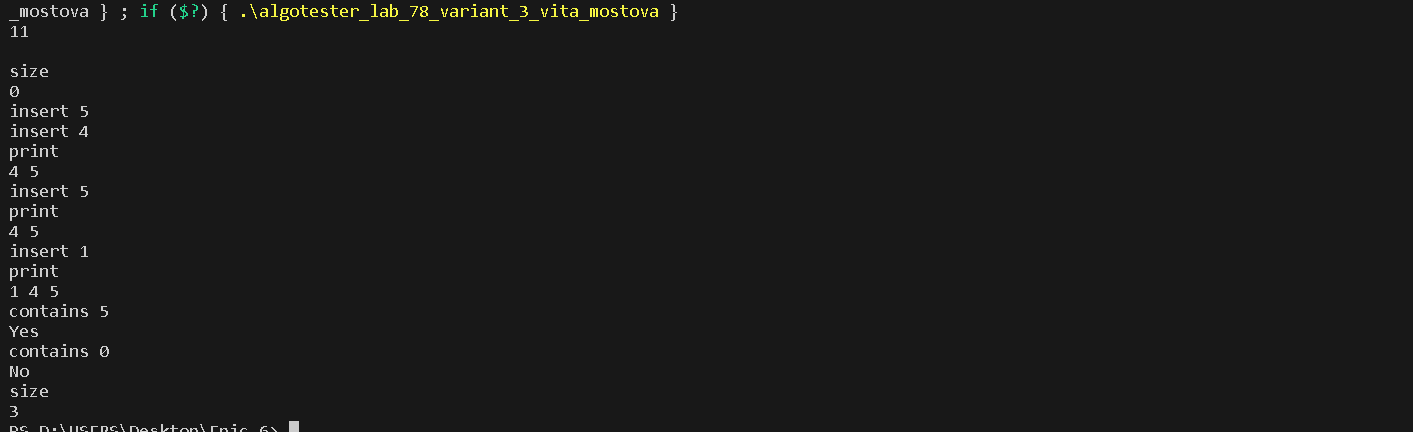
Завдання №3 **«Algotester 78v3»**

Рисунок 4:Приклад виконання завдання 3



Скрін з алготестера

Час затрачений на виконання завдання: близько 2 годин

Завдання №4 **«Practice Task 1»**

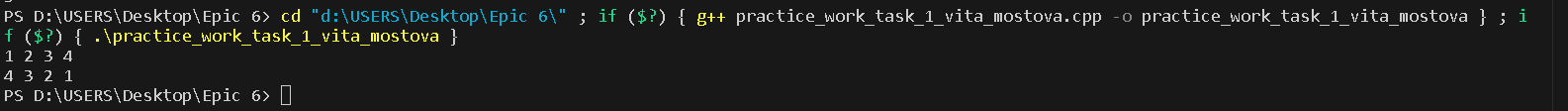
****

Рисунок 5:Приклад виконання завдання 4

Час затрачений на виконання завдання: близько години

Завдання №5 **«Practice Task 2»**

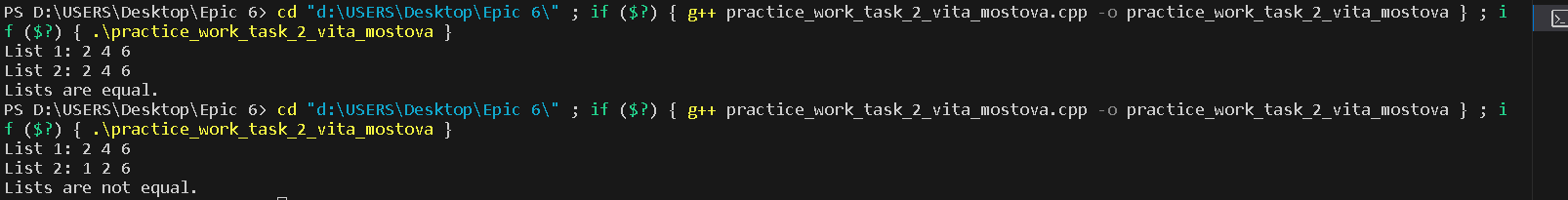


Рисунок 6:Приклад виконання завдання 5

Час затрачений на виконання завдання: близько 30 хвилин

Завдання №6 **«Practice Task 3»**

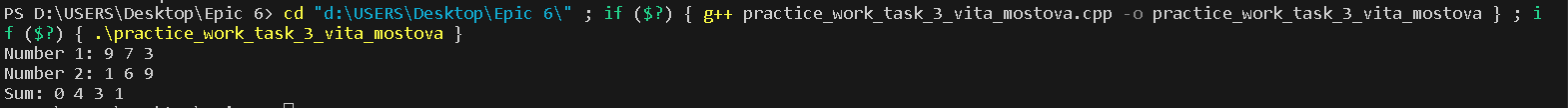


Рисунок 7:Приклад виконання завдання 6

Час затрачений на виконання завдання: близько 30 хвилин

Завдання №7 **«Practice Task 4»**

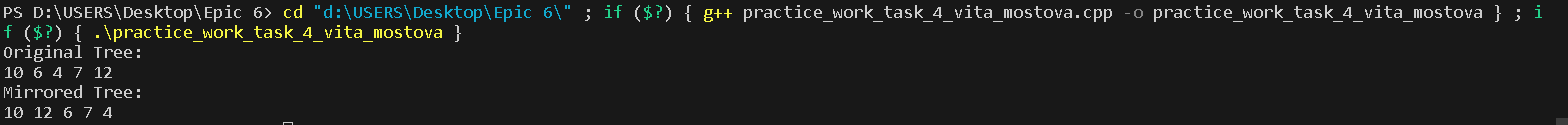


Рисунок 8:Приклад виконання завдання 7

Час затрачений на виконання завдання: близько 40 хвилин

Завдання №8 **«Practice Task 5»**

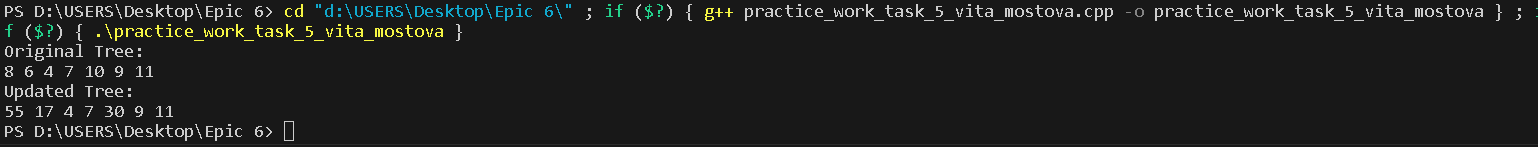


Рисунок 9:Приклад виконання завдання 8

Час затрачений на виконання завдання: близько години

## **5. Кооперація з командою:**

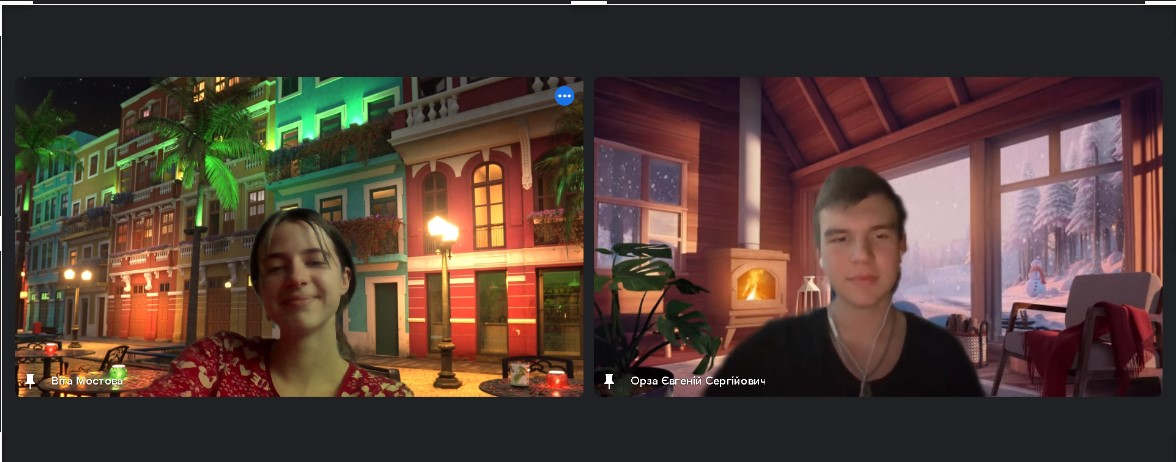


Рисунок 10:Скрін з командою)

# **Висновки:**

Під час виконання даних практичних та лабораторних робіт з епіка 6 я змогла зрозуміти основні відмінності в різних типах динамічних структур. Також вдалось попрактикуватись в реалізації зв’язних списків та бінарних дерев. Було пропрацьовано різні алгоритми обробки динамічних структур, а саме: створення списку, друк списку, вставка елемента в список, пошук по бінарному дереву, обхід дерева.