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

Новосад Олена Петрівна

# **Тема роботи:**

Динамічні структури у програмуванні. Стек. Черга. Звязаний двонаправлений та однонаправлений списки. Бінарне дерево. Бінарне дерево пошуку.

# **Мета роботи:**

Зрозуміти принципи роботи із динамічними структурами даних. Детальніше дізнатися про поняття двонаправленого і однонапрaGвленого списку, та операцій роботи з ними(вставка, пошук, перебір, видалення) . Засвоїти механізми маніпуляції вказівниками. Ознайомитися із теорією щодо бінарних дерев, та особливостей будови бінарного дерева пошуку. Застосувати на практиці отримані знання.

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Динамічний масив.
* Тема №2: Черга. Стек.
* Тема №3: Однонаправлений список. Двонаправлений список.
* Тема №4: Бінарні дерева. Бінарні дерева пошуку.

1. Індивідуальний план опрацювання теорії:

* Тема №1: Динамічний масив.
  + Джерела Інформації
    - <https://acode.com.ua/urok-90-dynamichni-masyvy/>
    - Л.І. Мочурад Конспект лекцій з дисципліни «Алгоритмізація та програмування. Частина 1»
  + Що опрацьовано:
    - Поняття динамічного масиву, методи його задання. Виділення пам’яті для динамічного масиву, вивільнення пам’яті. Витік пам’яті.
  + Статус: Ознайомлений
  + Початок опрацювання теми: 09.12.2023
  + Звершення опрацювання теми: 09.12.2023
* Тема №2: Однонаправлений список. Двонаправлений список.
  + Джерела Інформації:
    - Л.І. Мочурад Конспект лекцій з дисципліни «Алгоритмізація та програмування. Частина 1»
    - <https://erudyt.net/navchalni-predmety/informatika/prohramuvannya/linijnyj-odnozvyaznyj-spysok.html>
    - <https://www.bestprog.net/uk/2022/02/16/c-linear-doubly-linked-bidirectional-list-general-concepts-ua/>
  + Що опрацьовано:
    - Поняття однозвязного та двозвязного списків. Структура вузла для обох типів. Операції над списками: вставка, виділення, пошук, перебір.
  + Статус: Ознайомлений
  + Початок опрацювання теми: 12.12.2023
  + Звершення опрацювання теми: 14.12.2023
* Тема №3: Бінарні дерева. Бінарні дерева пошуку.
  + Джерела Інформації:
    - <https://www.programiz.com/dsa/binary-tree>
    - <https://uk.myservername.com/binary-search-tree-c>
    - <https://www.youtube.com/watch?v=Gt2yBZAhsGM>
  + Що опрацьовано:
    - Поняття дерев та їх структури. Бінарне дерево, особливості його вузлів. Бінарне дерево пошуку та його властивості.
  + Статус: Ознайомлений
  + Початок опрацювання теми:12.12.2023
  + Звершення опрацювання теми: 14.12.2023

# **Виконання роботи:**

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

Завдання №1 VNS Lab 10

* Варіант 16
* Деталі завдання

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.

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

символів). Сформувати двонаправлений список. Знищити елемент із заданим

ключем. Додати К елементів у кінець списку.

Завдання №2 Algotester Lab 5

* Варіант 3
* Деталі завдання
* У вас є карта гори розмiром N × M.
* Також ви знаєте координати {x, y} , у яких знаходиться вершина гори.
* Ваше завдання - розмалювати карту таким чином, щоб найнижча точка мала число 0, а пiк
* гори мав найбiльше число.
* Клiтинкi якi мають сумiжну сторону з вершиною мають висоту на один меншу, сумiжнi з
* ними i не розфарбованi мають ще на 1 меншу висоту i так далi.
* Вхiднi данi
* У першому рядку 2 числа N та M - розмiри карти
* у другому рядку 2 числа x та y - координати пiку гори
* Вихiднi данi
* N рядкiв по M елементiв в рядку через пробiл - висоти карти.

Завдання №3 Algotester Lab 78

* Варіант 3
* Деталі завдання
* Ваше завдання - власноруч реалiзувати структуру даних "Двiйкове дерево пошуку".
* Ви отримаєте Q запитiв, кожен запит буде починатися зi слова-iдентифiкатора, пiсля якого
* йдуть його параметри.
* Вам будуть поступати запити такого типу:
* • Вставка:
* Iдентифiкатор - insert
* Ви отримуєте цiле число value - число, яке треба вставити в дерево.
* • Пошук:
* Iдентифiкатор - contains
* Ви отримуєте цiле число value - число, наявнiсть якого у деревi необхiдно перевiрити.
* Якщо value наявне в деревi - ви виводите Y es, у iншому випадку No.
* • Визначення розмiру:
* Iдентифiкатор - size
* Ви не отримуєте аргументiв.
* Ви виводите кiлькiсть елементiв у деревi.
* • Вивiд дерева на екран
* Iдентифiкатор - print
* Ви не отримуєте аргументiв.
* Ви виводите усi елементи дерева через пробiл.
* Реалiзувати використовуючи перегрузку оператора <<
* Вхiднi данi
* Цiле число Q - кiлькiсть запитiв.
* У наступних рядках Q запитiв у зазначеному в умовi форматi.
* Вихiднi данi
* Вiдповiдi на запити у зазначеному в умовi форматi.

Завдання №4 Class Practice Tasks 1-3

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.

Завдання №5 Class Practice Tasks 4-5

* Деталі завдання :

## 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 №54 - 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

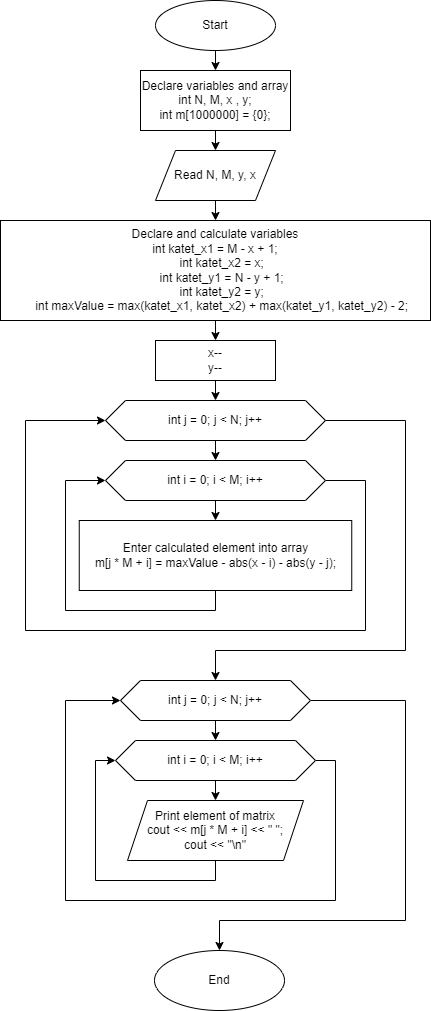
## **2. Дизайн та планована оцінка часу виконання завдань:**

Програма №1 VNS Lab 10

* Планований час на реалізацію: 2 дні

Програма №2 Algotester Lab 5

* Блок-схема:



* Планований час на реалізацію: 1 день

Програма №3 Algotester Lab 78

* Планований час на реалізацію: 2 дні

Програма №4 Class Practice Tasks 1-3

* Планований час на реалізацію: 1 день

Програма №5 Class Practice Tasks 4-5

* Планований час на реалізацію: 1 день

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

Завдання №1 VNS Lab 10

<https://github.com/artificial-intelligence-department/ai_programming_playground/blob/epic_6_practice_and_labs_olena_novosad/ai_13/olena_novosad/Epic%206/vns_lab_10_task_1_variant_1_olena_novosad.cpp>

#include <iostream>

#include <fstream>

#include <string.h>

#include <string>

using namespace std;

struct List

{

public:

    struct Node

    {

        char \*str;

        Node \*next = NULL;

        Node \*prev = NULL;

        Node(char \*arg)

        {

            auto size = strlen(arg) + 1;

            this->str = new char(size);

            strcpy(this->str, arg);

        }

        ~Node()

        {

            if (this->str != NULL)

            {

                delete[] this->str;

            }

        }

    };

private:

    Node \*first = NULL;

    Node \*last = NULL;

    void print(Node \*elem)

    {

        if (elem != NULL)

        {

            cout << "\"" << elem->str << "\"";

            if (elem->next != NULL)

            {

                cout << ", ";

            }

            print(elem->next);

        }

    }

    void remove(Node \*elem)

    {

        if (elem == this->first && elem == this->last)

        {

            this->first = NULL;

            this->last = NULL;

        }

        else if (elem == this->first)

        {

            this->first = elem->next;

            this->first->prev = NULL;

        }

        else if (elem == this->last)

        {

            this->last = elem->prev;

            this->last->next = NULL;

        }

        else

        {

            Node \*prevNode = elem->prev;

            Node \*nextNode = elem->next;

            prevNode->next = nextNode;

            nextNode->prev = prevNode;

        }

        delete elem;

    }

public:

    void clear()

    {

        Node \*elem = this->first;

        while (elem)

        {

            Node \*nextElem = elem->next;

            delete elem;

            elem = nextElem;

        }

        this->first = NULL;

    }

    List()

    {

    }

    ~List()

    {

        this->clear();

    }

    void push\_back(char \*arg)

    {

        Node \*node = new Node(arg);

        if (this->first == NULL)

        {

            this->first = node;

            this->last = node;

        }

        else

        {

            this->last->next = node;

            Node \*p = this->last;

            this->last = node;

            node->prev = p;

        }

    }

    void print()

    {

        if (this->first == NULL)

        {

            cout << "List is empty!\n";

        }

        else

        {

            Node \*elem = this->first;

            print(elem);

            cout << "\n";

        }

    }

    void remove(char \*arg)

    {

        Node \*elem = this->last;

        while (elem != NULL)

        {

            Node \*prevElem = elem->prev;

            if (strcmp(elem->str, arg) == 0)

            {

                this->remove(elem);

                break;

            }

            elem = prevElem;

        }

    }

    void saveIntoFile(char \*fileName)

    {

        // відкриваємо файл

        ofstream outf{fileName, ios::trunc | ios::out};

        if (!outf)

        {

            return;

        }

        Node \*elem = this->first;

        while (elem != NULL)

        {

            Node \*nextElem = elem->next;

            // записуємо node у файл

            outf << elem->str << "\n";

            elem = nextElem;

        }

        outf.close();

    }

    void readContentFromFile(char \*fileName)

    {

        ifstream fin;

        fin.open(fileName);

        if (!fin.is\_open())

        {

            return;

        }

        while (true)

        {

            string strLine;

            getline(fin, strLine);

            if (fin.eof())

            {

                break;

            }

            else

            {

                this->push\_back((char \*)strLine.c\_str());

            }

        }

        fin.close();

    }

};

int main()

{

    List list;

    list.push\_back("Some text");

    list.push\_back("Next element");

    list.print();

    list.clear();

    list.print();

    list.push\_back("Some text");

    list.push\_back("Middle");

    list.push\_back("Some text");

    list.print();

    list.remove("Some text");

    list.print();

    list.saveIntoFile("data.txt");

    list.clear();

    list.print();

    list.readContentFromFile("data.txt");

    list.print();

    return 0;

}

Рисунок 1: Код до програми №1

Завдання №2 Algotester Lab 5

<https://github.com/artificial-intelligence-department/ai_programming_playground/blob/epic_6_practice_and_labs_olena_novosad/ai_13/olena_novosad/Epic%206/algotester_lab_5_variant_3_olena_novosad.cpp>

#include <iostream>

using namespace std;

int N, M, x , y;

int m[1000000] = {0};

int main() {

    cin >> N >> M;

    cin >> y >> x;

    int katet\_x1 = M - x + 1;

    int katet\_x2 = x;

    int katet\_y1 = N - y + 1;

    int katet\_y2 = y;

    int maxValue = max(katet\_x1, katet\_x2) + max(katet\_y1, katet\_y2) - 2;

    x--;

    y--;

    for (int j = 0; j < N; j++) {

        for (int i = 0; i < M; i++) {

            m[j \* M + i] = maxValue - abs(x - i) - abs(y - j);

        }

    }

    for (int j = 0; j < N; j++) {

        for (int i = 0; i < M; i++) {

            cout << m[j \* M + i] << " ";

        }

        cout << "\n";

    }

    return 0;

}

Рисунок 2: Код до програми №2

Завдання №3 Algotester Lab 78

<https://github.com/artificial-intelligence-department/ai_programming_playground/blob/epic_6_practice_and_labs_olena_novosad/ai_13/olena_novosad/Epic%206/algotester_lab_7_8_variant_3_olena_novosad.cpp>

#include <iostream>

#include <string>

using namespace std;

template <typename T> class Node {

public:

    T value;

    Node\* left = NULL;

    Node\* right = NULL;

    Node(T newValue) {

        this->value = newValue;

    }

    template <typename U> friend ostream& operator<<(ostream&, const Node<U>&);

};

template <typename T> ostream& operator<<(ostream& out, const Node<T>& node) {

    if (node.left != NULL) {

        out << (\*node.left);

    }

    out << node.value << " ";

    if (node.right != NULL) {

        out << (\*node.right);

    }

    return out;

}

template <typename T> class Tree {

    Node<T>\* head = NULL;

    void calculateSize(int& s, Node<T>\* point) {

        if (point != NULL) {

            s++;

            if (point->left != NULL) {

                calculateSize(s, point->left);

            }

            if (point->right != NULL) {

                calculateSize(s, point->right);

            }

        }

    }

    void contains(bool& valueFound, T value, Node<T>\* point) {

        if (valueFound) {

            return;

        }

        if (point != NULL) {

            if (point->value == value) {

                valueFound = true;

                return;

            }

            if (point->left != NULL) {

                contains(valueFound, value, point->left);

            }

            if (point->right != NULL) {

                contains(valueFound, value, point->right);

            }

        }

    }

    void insert(Node<T>\* node, T newValue) {

        if (head == NULL) {

            head = new Node<T>(newValue);

            return;

        }

        else {

            if (newValue == node->value) {

                return;

            }

            else if (newValue < node->value) {

                if (node->left == NULL) {

                    node->left = new Node<T>(newValue);

                    return;

                }

                else {

                    insert(node->left, newValue);

                    return;

                }

            }

            else /\* means (newValue > node->value) \*/ {

                if (node->right == NULL) {

                    node->right = new Node<T>(newValue);

                    return;

                }

                else {

                    insert(node->right, newValue);

                    return;

                }

            }

        }

    }

public:

    void insert(T newValue) {

        insert(this->head, newValue);

    }

    void size() {

        int sizeValue = 0;

        calculateSize(sizeValue, head);

        cout << sizeValue << "\n";

    }

    void contains(T value) {

        bool valueFound = false;

        contains(valueFound, value, head);

        cout << ((valueFound) ? "Yes" : "No") << "\n";

    }

    template <typename U> friend ostream& operator<<(ostream&, const Tree<U>&);

};

template <typename T> ostream& operator<<(ostream& out, const Tree<T>& tree) {

    if (tree.head != NULL) {

        out << (\*tree.head);

    }

    return out;

}

int main() {

    int Q;

    cin >> Q;

    Tree<int> tree;

    for (int j = 0; j < Q; j++) {

        string operationStr;

        cin >> operationStr;

        if (operationStr == "insert") {

            int newValue;

            cin >> newValue;

            tree.insert(newValue);

        }

        else if (operationStr == "contains") {

            int findValue;

            cin >> findValue;

            tree.contains(findValue);

        }

        else if (operationStr == "size") {

            tree.size();

        }

        else if (operationStr == "print") {

            cout << tree << "\n";

        }

        else {

            cout << "unsupported operation: " << operationStr <<"\n";

        }

    }

    return 0;

}

Рисунок 3: Код до програми №3

Завдання №4 Class Practice Tasks 1-3

<https://github.com/artificial-intelligence-department/ai_programming_playground/blob/epic_6_practice_and_labs_olena_novosad/ai_13/olena_novosad/Epic%206/practice_work_task_1_2_3_olena_novosad.cpp>

#include <iostream>

#include <string>

#include <string.h>

using namespace std;

struct Node {

    int value;

    Node\* next = NULL;

    Node(int value) {

        this->value = value;

    }

};

Node\* push\_back(Node\* first, int arg) {

    Node\* node = new Node(arg);

    if (first == NULL) {

        return node;

    }

    else {

        Node\* elem = first;

        // go to end of list

        while (elem->next != NULL) {

            elem = elem->next;

        }

        elem->next = node;

    }

    return first;

}

Node\* reverse(Node\* head) {

    if (head == NULL) {

        return NULL;

    }

    else {

        Node\* elem = head;

        Node\* nextElem = elem->next;

        head->next = NULL;

        while(nextElem != NULL) {

            Node\* p = nextElem->next;

            nextElem->next = elem;

            elem = nextElem;

            nextElem = p;

        }

        return elem;

    }

}

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

    if (h1 == h2) {

        return true;

    }

    while (h1 != NULL && h2 != NULL && h1->value == h2->value) {

        h1 = h1->next;

        h2 = h2->next;

        if (h1 == h2) {

            return true;

        }

    }

    return false;

}

ostream& operator<<(ostream& out, const Node\* list) {

    if (list != NULL) {

        out << (list->value) << " " << list->next;

    }

    return out;

}

Node\* saveStrNumberInList(char\* str) {

    Node\* head = NULL;

    Node\* elem = NULL;

    auto size = strlen(str);

    for (int i = 0; i < size; i++) {

        char ch = str[size - i - 1];

        int n = atoi(&ch);

        Node\* node = new Node(n);

        if (head == NULL) {

            head = node;

            elem = node;

        }

        else {

            elem->next = node;

            elem = node;

        }

    }

    return head;

}

int listLength(const Node\* head) {

    int result = 0;

    const Node\* elem = head;

    while (elem != NULL) {

        result++;

        elem = elem->next;

    }

    return result;

}

ostream& operator<<(ostream& out, const Node head) {

    int length = listLength(&head);

    string result;

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

        result += " ";

    }

    int i = 0;

    const Node\* elem = &head;

    while (elem != NULL) {

        string str = to\_string(elem->value);

        result[length - 1 - i] = str[0];

        elem = elem->next;

        i++;

    }

    out << result;

    return out;

}

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

    Node\* resultHead = NULL;

    Node\* elem = NULL;

    int shift = 0;

    while (n1 != NULL || n2 != NULL) {

        int v1 = 0;

        int v2 = 0;

        if (n1 != NULL) {

            v1 = n1->value;

            n1 = n1->next;

        }

        if (n2 != NULL) {

            v2 = n2->value;

            n2 = n2->next;

        }

        int r = v1 + v2 + shift;

        int rv = r % 10;

        shift = r / 10;

        Node\* newNode = new Node(rv);

        if (resultHead == NULL) {

            resultHead = newNode;

            elem = newNode;

        }

        else {

            elem->next = newNode;

            elem = newNode;

        }

    }

    if (shift != 0) {

        Node\* newNode = new Node(shift);

        elem->next = newNode;

    }

    return resultHead;

}

int main() {

    Node\* list1 = push\_back(NULL, 5);

    push\_back(list1, 6);

    push\_back(list1, 7);

    cout << "list1: " << list1 << "\n";

    Node\* listReversed = reverse(list1);

    cout << "listReversed: " << listReversed << "\n";

    Node\* list2 = push\_back(NULL, 7);

    push\_back(list2, 6);

    push\_back(list2, 5);

    cout << "list2: " << list2 << "\n";

    cout << (compare(listReversed, list2) ? "listReversed == list2" : "listReversed != list2") << "\n";

    Node\* list3 = push\_back(NULL, 7);

    push\_back(list3, 6);

    cout << "list3: " << list3 << "\n";

    cout << (compare(listReversed, list3) ? "listReversed == list3" : "listReversed != list3") << "\n";

    Node\* list4 = push\_back(NULL, 7);

    push\_back(list4, 6);

    push\_back(list4, 7);

    cout << "list4: " << list4 << "\n";

    cout << (compare(listReversed, list4) ? "listReversed == list4" : "listReversed != list4") << "\n";

    Node\* number1 = saveStrNumberInList("485934593485934593945349932884858585");

    cout << \*number1 << "\n";

    Node\* number2 = saveStrNumberInList("123");

    cout << \*number2 << "\n";

    Node\* result = add(number1, number2);

    cout << \*result << "\n";

    return 0;

}

Рисунок 4: Код до програми №4

Завдання №5 Class Practice Tasks 4-5

<https://github.com/artificial-intelligence-department/ai_programming_playground/blob/epic_6_practice_and_labs_olena_novosad/ai_13/olena_novosad/Epic%206/practice_work_task_4_5_olena_novosad.cpp>

#include <iostream>

#include <string>

using namespace std;

struct TreeNode {

    int value;

    TreeNode\* left = NULL;

    TreeNode\* right = NULL;

    TreeNode(int newValue) {

        this->value = newValue;

    }

    friend ostream& operator<<(ostream&, const TreeNode\*);

};

ostream& operator<<(ostream& out, const TreeNode\* node) {

    if (node == NULL) {

        return out;

    }

    out << node->value << " ";

    if (node->left != NULL) {

        out << node->left;

    }

    if (node->right != NULL) {

        out << node->right;

    }

    return out;

}

TreeNode\* insert(TreeNode\* node, int newValue) {

    if (node == NULL) {

        node = new TreeNode(newValue);

        return node;

    }

    else {

        if (newValue == node->value) {

            return node;

        }

        else if (newValue < node->value) {

            if (node->left == NULL) {

                node->left = new TreeNode(newValue);

                return node;

            }

            else {

                insert(node->left, newValue);

                return node;

            }

        }

        else {

            if (node->right == NULL) {

                node->right = new TreeNode(newValue);

                return node;

            }

            else {

                insert(node->right, newValue);

                return node;

            }

        }

    }

}

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

    if (root == NULL) {

        return NULL;

    }

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

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

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

    return newNode;

}

int calculate(TreeNode\* node) {

    if (node == NULL) {

        return 0;

    }

    if (node->left != NULL || node->left != NULL) {

        int l = calculate(node->left);

        int r = calculate(node->right);

        return l + r;

    }

    else {

        return node->value;

    }

}

void tree\_sum(TreeNode\* root) {

    if (root == NULL) {

        return;

    }

    root->value = calculate(root);

    tree\_sum(root->left);

    tree\_sum(root->right);

}

int main() {

    TreeNode\* tree = insert(NULL, 6);

    insert(tree, 0);

    insert(tree, 10);

    insert(tree, 2);

    insert(tree, 1);

    insert(tree, 3);

    insert(tree, 11);

    cout << tree << "\n";

    TreeNode\* newTree = create\_mirror\_flip(tree);

    cout << newTree << "\n";

    tree\_sum(newTree);

    cout << newTree << "\n";

    return 0;

}

Рисунок 5: Код до програми №5

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

Завдання №1 VNS Lab 10

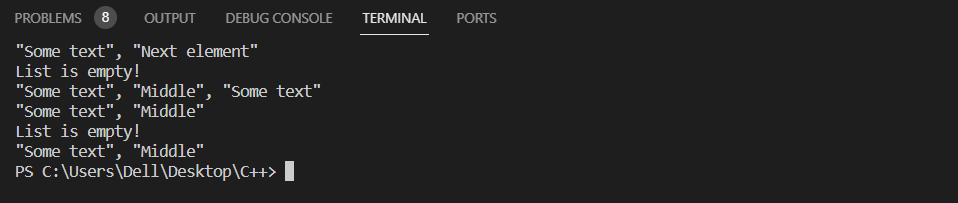


Рисунок 7: Тестування програми №1

Час затрачений на виконання завдання: 1 день

Завдання №2 Algotester Lab 5

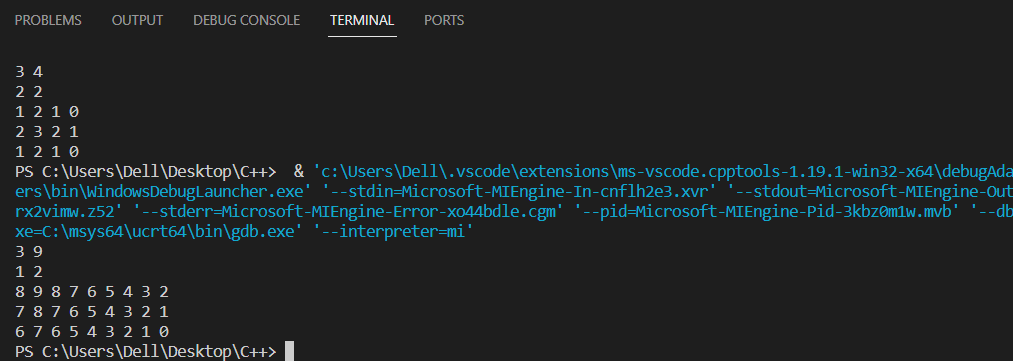


Рисунок 8: Тестування програми №2

Час затрачений на виконання завдання: 1 день

Завдання №3 Algotester Lab 78

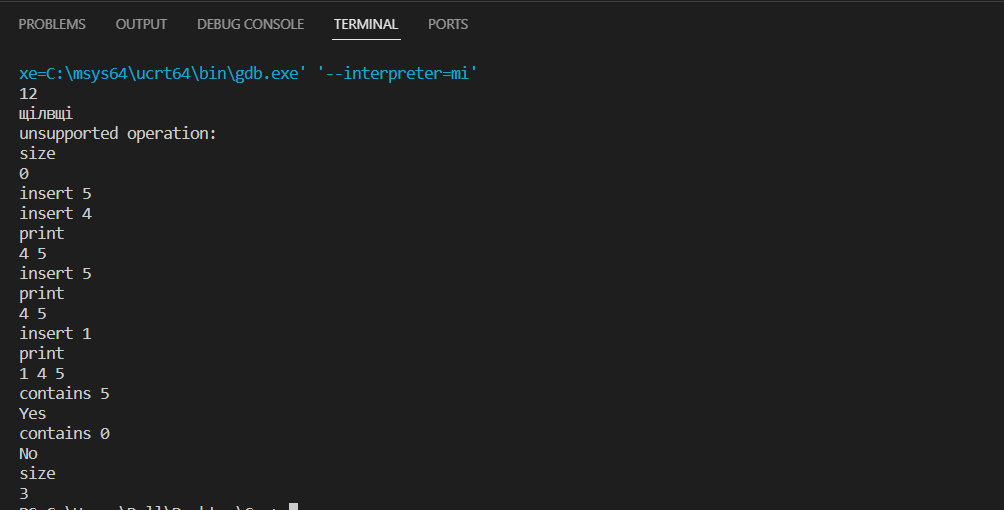


Рисунок 9: Тестування програми №3

Час затрачений на виконання завдання: 1 день

Завдання №4 Class Practice Tasks 1-3

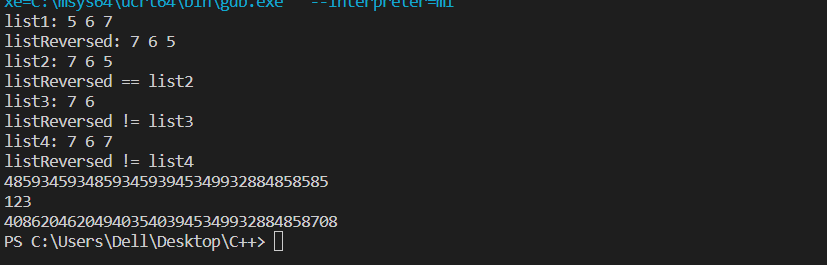


Рисунок 10: Тестування програми №4

Час затрачений на виконання завдання: 1 день

Завдання №5 Class Practice Tasks 4-5

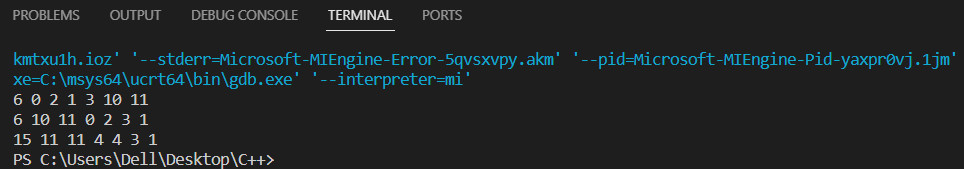


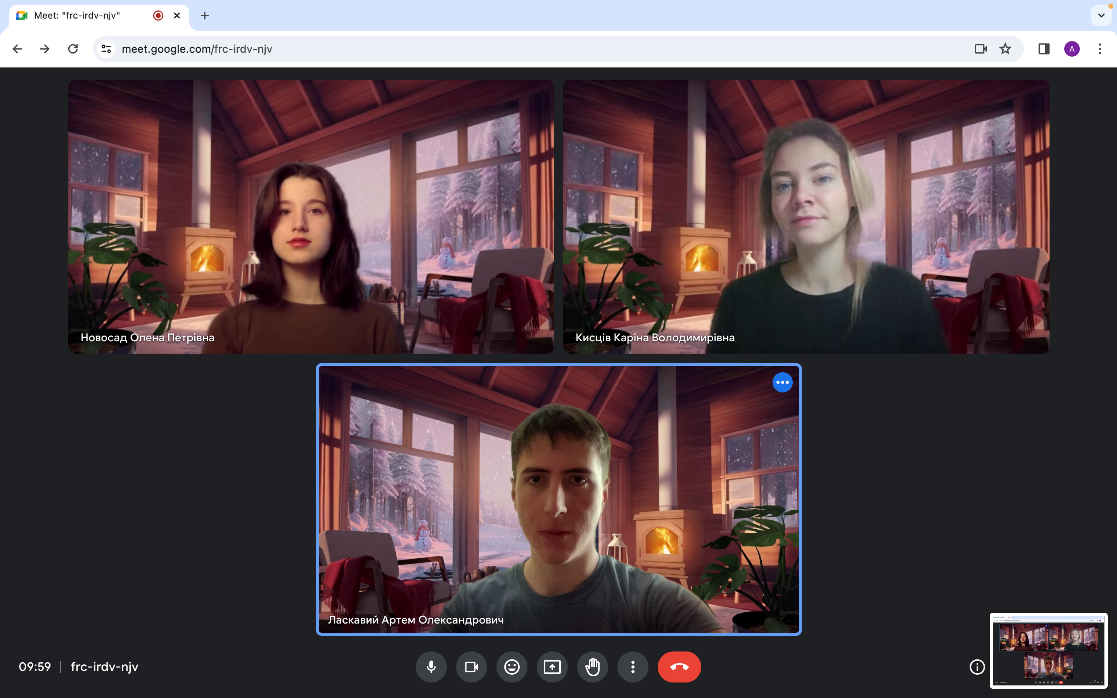
Рисунок 11: Тестування програми №5

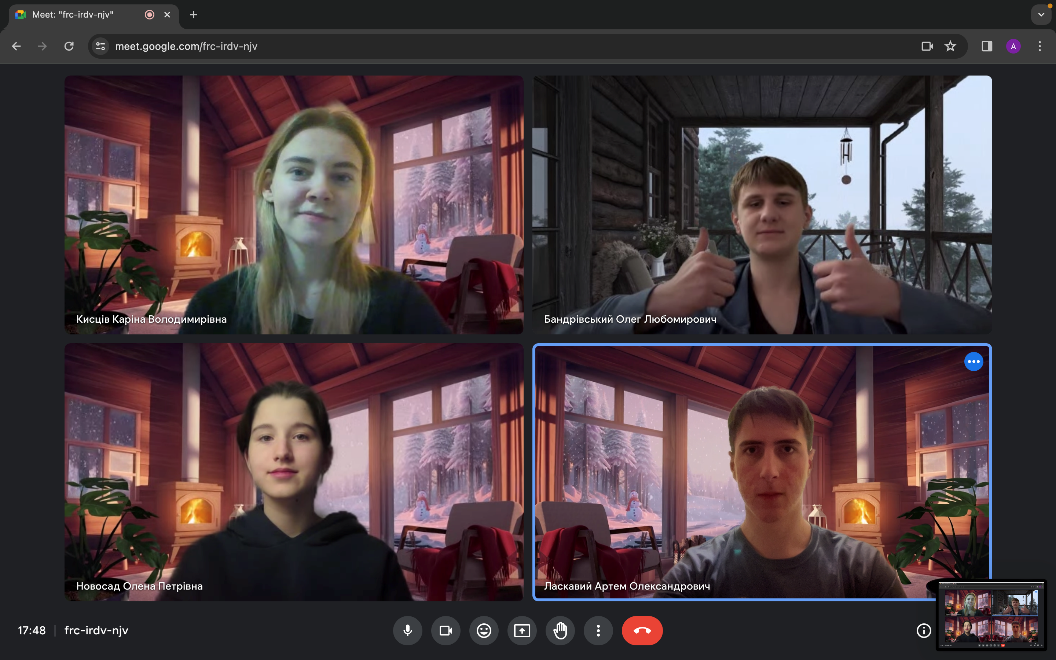
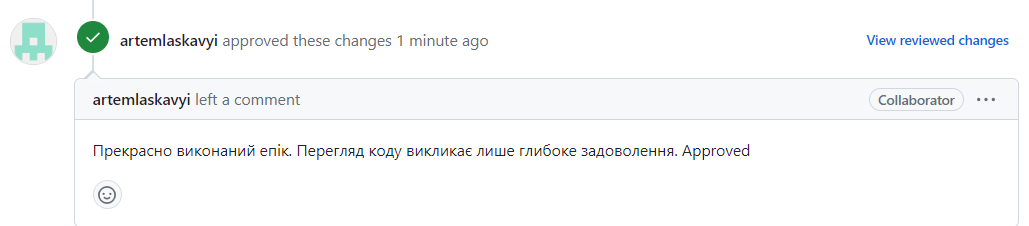
Час затрачений на виконання завдання: 1 день

## 

## **5. Кооперація з командою:**

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



* Скрін з 2-ї зустрічі по обговоренню задач Епіку та Скрін прогресу по Трелло
* 
* Скрін з 2-му коментарями від учасників команди на пул реквесті з Ревю Роботи
* 

# **Висновки:**

Під час виконання даної лабораторної роботи я детальніше дізналася про методи роботи із однонаправленими та двонаправленими списками. Засвоїла механізми маніпуляції вказівниками. Ознайомилася із поняттям бінарного дерева та бінарного дерева пошуку, застосувала це на практиці.