Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту

A blue and white logo

Description automatically generated

**Звіт**

**Звіт**

**про виконання лабораторних та практичних робіт блоку № 6**

На тему: «Динамічні структури (Черга, Стек, Списки, Дерево). Алгоритми обробки динамічних структур.»

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

до:

ВНС Лабораторної Роботи № 10

Алготестер Лабораторної Роботи № 5

Алготестер Лабораторної Роботи № 7-8

Практичних Робіт № 6

***Виконав:***

студент групи ШІ-12

Пастухов Олександр Сергійович

# **Тема роботи:**

Ознайомлення та робота з динамічними структурами даних в мові програмування C++. Принцип роботи з бінарними деревами. Ознайомлення з поняттями Стеку, Черги та Зв’язного Списку. Ознайомлення з основними алгоритмами обробки динамічних структур.

# **Мета роботи:**

Навчитися працювати з динамічними структурами даних в мові C++. Вивчити основні алгоритми обробки динамічних структур (Черга, Стек …). Застосувати набуті знання на практиці.

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Черга в C++.
* Тема №2: Стек в C++.
* Тема №3: Однозв’язний список в C++.
* Тема №4: Двозв’язний список в C++.
* Тема №5: Бінарні дерева пошуку в C++.

1. Індивідуальний план опрацювання теорії:

* Тема №1: Черга в C++.
  + Джерела Інформації
    - Стаття: Queue in C++ STL [Queue in C++ Standard Template Library (STL) - GeeksforGeeks](https://www.geeksforgeeks.org/queue-cpp-stl/)
  + Що опрацьовано:
    - Опрацьовано теоретичний матеріал про роботу Черги в мові C++ та його застосування
  + Статус: Ознайомлений
  + Початок опрацювання теми: 20.12.2023
  + Звершення опрацювання теми: 22.11.2023
* Тема №2: Стек в C++.
  + Джерела Інформації
    - Стаття: Stack in C++ STL [Stack in C++ STL - GeeksforGeeks](https://www.geeksforgeeks.org/stack-in-cpp-stl/)
  + Що опрацьовано:
    - Опрацьовано теоретичний матеріал про роботу Стеку.
  + Статус: Ознайомлений
  + Початок опрацювання теми: 20.12.2023
  + Звершення опрацювання теми: 22.12.2023
* Тема №3: Однозв’язний список в C++.
  + Джерела Інформації:
    - Стаття: Linked List Data Structure [Linked List Data Structure - GeeksforGeeks](https://www.geeksforgeeks.org/data-structures/linked-list/)
  + Що опрацьовано:
    - Опрацьовано поняття Однозв’язного списку.
  + Статус: Ознайомлений
  + Початок опрацювання теми 20.12.2023 22.30
  + Звершення опрацювання теми: 22.12.2023 23.50
* Тема №4: Двозв’язний список в C++.
  + Джерела Інформації:
    - Стаття: Doubly Linked List [Doubly Linked List (With code)](https://www.programiz.com/dsa/doubly-linked-list)
  + Що опрацьовано:
    - Опрацьовано та вивчено основні методи роботи з Двонаправленим списком в мові програмування C++ та його застосування
  + Статус: Ознайомлений
  + Початок опрацювання теми: 20.12.2023 16.00
  + Звершення опрацювання теми: 22.12.2023 17.40
* Тема №5: Бінарні дерева пошуку в C++.
  + Джерела Інформації:
    - Стаття: Introduction to Binary Tree [Introduction to Binary Tree - Data Structure and Algorithm Tutorials - GeeksforGeeks](https://www.geeksforgeeks.org/introduction-to-binary-tree-data-structure-and-algorithm-tutorials/)
  + Що опрацьовано:
    - Опрацьовано та вивчено основні методи роботи з Бінарними деревами в мові програмування C++
    - Вивчено алгоритми обходу бінарних дерев
  + Статус: Ознайомлений
  + Початок опрацювання теми: 20.12.2023 14.50
  + Звершення опрацювання теми: 22.12.2023 17.10

# **Виконання роботи:**

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

Завдання №1 VNS Lab 10

* Варіант завдання: 19
* 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.

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

символів). Сформувати двонаправлений список. Знищити з нього елемент із

заданим ключем, додати елемент із зазначеним номером.

* Важливі деталі для врахування в імплементації програми:

Використання цілочисельних значень для запису в Список

Записи в лінійному списку містять ключове поле типу \*char (рядок символів). Сформувати двонаправлений список. Знищити К елементів із заданими номерами. Додати К елементів у початок списку.

Завдання №2 Algotester Lab 5

* Варіант завдання: 3
* Деталі завдання: <https://algotester.com/en/ContestProblem/DisplayWithEditor/134644>
* Важливі деталі для врахування в імплементації програми:

Дії над клітинками мапи використовувати двовимірний масив

Завдання №3 Algotester Lab 7-8

* Варіант завдання: 3
* Деталі завдання: <https://algotester.com/en/ContestProblem/DisplayWithEditor/134648>
* Важливі деталі для врахування в імплементації програми

Використання власноруч написаних функцій для обробки Динамічного Масиву

Завдання №4 Class Practice Work Task 1

* 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.

* Важливі деталі для врахування в імплементації програми

Завдання виконано на основі власноруч написаного Однозв’язного списку та власних функцій

Завдання №5 Class Practice Work Task 2

* Варіант завдання: немає
* 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 №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

* Важливі деталі для врахування в імплементації програми

Використання власноруч написаного Бінарного дерева та методів, які вимагалися в завданні

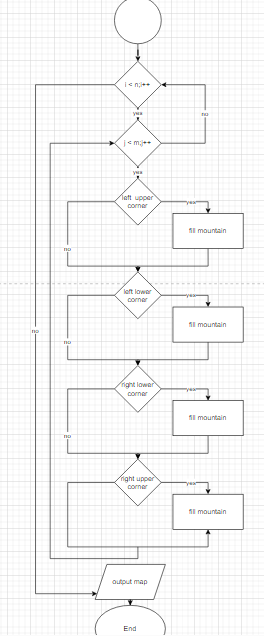
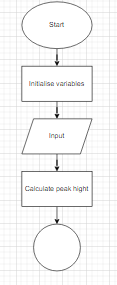
## **2. Дизайн та планована оцінка часу виконання завдань:**

Програма №1 VNS Lab 10

* Планований час на реалізацію: 120 хвилин

Програма №2 Algotester Lab 5

* Блок-схема



*Рисунок 1,2. Блок-схема до програми №2*

* Планований час на реалізацію: 100 хвилин

Програма №3 Algotester Lab 7-8

* Планований час на реалізацію: 120 хвилин

Програма №4 Class Practice Work Task 1

* Планований час на реалізацію: 200 хвилин

Програма №5 Class Practice Work Task 2

* Планований час на реалізацію: 200 хвилин

Програма №6 Self Practice Work

* Планований час на реалізацію: 40 хвилин

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

Для виконання поставлених завдань додаткова конфігурація середовища не є необхідною

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

Завдання №2 Algotester Lab 5

#include <iostream>

#include <algorithm>

using namespace std;

int main()

{

    int n, m, x, y;

    cin >> n >> m;

    cin >> x >> y;

    int mountain[n][m];

    int peak = max(x - 1, n - x) + max(y - 1, m - y);

    x--;

    y--;

    mountain[x][y] = peak;

    for (int i = 0; i < n; i++)

    {

        for (int j = 0; j < m; j++)

        {

            int dist1 = max(0, peak - (i + j));

            int dist2 = max(0, peak - (i - j));

            int dist3 = max(0, peak - (-i + j));

            int dist4 = max(0, peak - (-i - j));

            if (x - i >= 0 && x - i < n && y - j >= 0 && y - j < m)

            {

                mountain[x - i][y - j] = dist1;

            }

            if (x + i >= 0 && x + i < n && y + j >= 0 && y + j < m)

            {

                mountain[x + i][y + j] = dist2;

            }

            if (x - i >= 0 && x - i < n && y + j >= 0 && y + j < m)

            {

                mountain[x - i][y + j] = dist3;

            }

            if (x + i >= 0 && x + i < n && y - j >= 0 && y - j < m)

            {

                mountain[x + i][y - j] = dist4;

            }

        }

    }

    for (int i = 0; i < n; i++)

    {

        for (int j = 0; j < m; j++)

        {

            cout << mountain[i][j] << " ";

        }

        cout << endl;

    }

    return 0;

}

Завдання №3 Algotester Lab 7-8

#include <iostream>

struct Node {

    int value;

    Node\* left;

    Node\* right;

    Node(int val) : value(val), left(nullptr), right(nullptr) {}

};

class BinarySearchTree {

private:

    Node\* root;

    void insert(Node\*& current, int value);

    bool contains(Node\* current, int value) const;

    int size(Node\* current) const;

    void print(Node\* current) const;

public:

    BinarySearchTree() : root(nullptr) {}

    void insert(int value);

    bool contains(int value) const;

    int size() const;

    void print() const;

};

void BinarySearchTree::insert(Node\*& current, int value) {

    if (current == nullptr) {

        current = new Node(value);

    } else if (value < current->value) {

        insert(current->left, value);

    } else if (value > current->value) {

        insert(current->right, value);

    }

}

void BinarySearchTree::insert(int value) {

    insert(root, value);

}

bool BinarySearchTree::contains(Node\* current, int value) const {

    if (current == nullptr) {

        return false;

    }

    if (value == current->value) {

        return true;

    } else if (value < current->value) {

        return contains(current->left, value);

    } else {

        return contains(current->right, value);

    }

}

bool BinarySearchTree::contains(int value) const {

    return contains(root, value);

}

int BinarySearchTree::size(Node\* current) const {

    if (current == nullptr) {

        return 0;

    }

    return 1 + size(current->left) + size(current->right);

}

int BinarySearchTree::size() const {

    return size(root);

}

void BinarySearchTree::print(Node\* current) const {

    if (current != nullptr) {

        print(current->left);

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

        print(current->right);

    }

}

void BinarySearchTree::print() const {

    print(root);

    std::cout << std::endl;

}

int main() {

    int Q;

    std::cin >> Q;

    BinarySearchTree bst;

    while (Q--) {

        std::string operation;

        std::cin >> operation;

        if (operation == "insert") {

            int value;

            std::cin >> value;

            bst.insert(value);

        } else if (operation == "contains") {

            int value;

            std::cin >> value;

            std::cout << (bst.contains(value) ? "Yes" : "No") << std::endl;

        } else if (operation == "size") {

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

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

            bst.print();

        }

    }

    return 0;

}

# **Висновки:**

Під час опрацювання теоретичного матеріалу та роботи над завданнями розділу я навчився працювати з динамічними структурами даних в мові C++.Прочитав про бінарні дерева та двозв’язі та однозв’язні списки.