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:*** «Основи програмування»

до:

ВНС Лабораторної Роботи № 10

Алготестер Лабораторної Роботи № 5

Алготестер Лабораторної Роботи № 7-8

Практичних Робіт до блоку № 6

**Виконав:**

Студент групи ШІ-13

Басараб Дмитрій Богданович

Львів 2024

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

### **Мета роботи:** *Навчитись працювати з Динамічними структурами (Черга, Стек, Списки, Дерево), вивчити Алгоритми обробки динамічних структур*

### **Теоретичні відомості:**

1. C++ <https://www.youtube.com/watch?v=2UDMGCcRCjo&list=PLiPRE8VmJzOpn6PzYf0higmCEyGzo2A5g>

### **Виконання роботи:**

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

**Завдання №1 - VNS Lab 10 var 20**

*Написати програму, у якій створюються динамічні структури й виконати*

*їхню обробку у відповідності зі своїм варіантом.*

*Для кожного варіанту розробити такі функції:*

*1. Створення списку.*

*2. Додавання елемента в список (у відповідності зі своїм варіантом).*

*3. Знищення елемента зі списку (у відповідності зі своїм варіантом).*

*4. Друк списку.*

*5. Запис списку у файл.*

*6. Знищення списку.*

*7. Відновлення списку з файлу.*

*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.*

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

*символів). Сформувати двонаправлений список. Знищити елемент із заданим*

*ключем. Додати по К елементів на початок й в кінець списку.*

**Завдання №2 - Algotester Lab78v1**

# **Lab 78v1**

***Limits: 2 sec., 256 MiB***

**Ваше завдання - власноруч реалізувати структуру даних "Двозв’язний список".  
Ви отримаєте Q запитів, кожен запит буде починатися зі слова-ідентифікатора, після якого йдуть його аргументи.  
  
Вам будуть поступати запити такого типу:**

**Вставка:  
Ідентифікатор - insert  
Ви отримуєте ціле число index елемента, на місце якого робити вставку.  
Після цього в наступному рядку рядку написане число N- розмір списку, який треба вставити.  
У третьому рядку N цілих чисел - список, який треба вставити на позицію index**

**Видалення:  
Ідентифікатор - erase  
Ви отримуєте 2 цілих числа - index елемента, з якого почати видалення та n - кількість елементів, яку треба видалити.**

**Визначення розміру:  
Ідентифікатор -size  
Ви не отримуєте аргументів.  
Ви виводите кількість елементів у списку.**

**Отримання значення i-го елементу:  
Ідентифікатор - get  
Ви отримуєте ціле число - index, індекс елемента.  
Ви виводите значення елемента за індексом.**

**Модифікація значення i-го елементу:  
Ідентифікатор - set  
Ви отримуєте 2 цілих числа - індекс елемента, який треба змінити, та його нове значення.**

**Вивід списку на екран:  
Ідентифікатор - print  
Ви не отримуєте аргументів.  
Ви виводите усі елементи списку через пробіл.**

# **Input**

**Ціле число Q - кількість запитів.  
У наступних рядках Q запитів у зазначеному в умові форматі.**

# **Output**

**Відповіді на запити у визначеному в умові форматі.**

**Завдання №3 - Algotester Lab5v3**

# **Lab 5v3**

***Limits: 1 sec., 256 MiB***

**У вас є карта гори розміром 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 пік гори мав найбільше число.**

**Клітинкі 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.**

# **Input**

**У першому рядку 2 числа N та M - розміри карти**

**у другому рядку 2 числа x та y - координати піку гори**

# **Output**

**N рядків по M елементів в рядку через пробіл - висоти карти.**

**Завдання №4 - Practice task 1-3 Linked List**

# **Зв’язаний список**

## **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;

### **Пояснення прикладу**

Спочатку ми визначаємо просту структуру ***Node*** для нашого пов’язаного списку.

Потім функція ***reverse*** ітеративно змінює список, маніпулюючи наступними покажчиками кожного вузла.

***printList*** — допоміжна функція для відображення списку.

Основна функція створює зразок списку, демонструє реверсування та друкує вихідний і обернений списки.

## **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***.

### **Пояснення прикладу**

● Для пов’язаного списку визначено структуру ***Node***.

● Функція ***compare*** ітеративно проходить обидва списки одночасно, порівнюючи дані в кожному вузлі.

● Якщо виявлено невідповідність даних або якщо довжина списків різна (один список закінчується раніше іншого), функція повертає ***false***.

● Основна функція ***main*** створює два списки та демонструє порівняння.

## **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 - Practice task 4-5 Binary Tree**

## **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

1. *Дизайн та планована оцінка часу виконання завдань:*

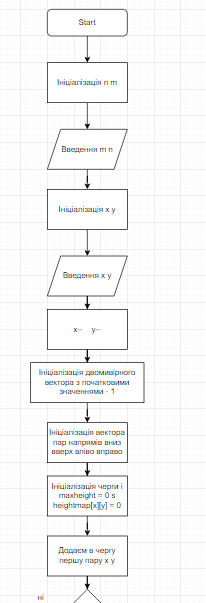
**Завдання №1 - VNS Lab 10 var 20**

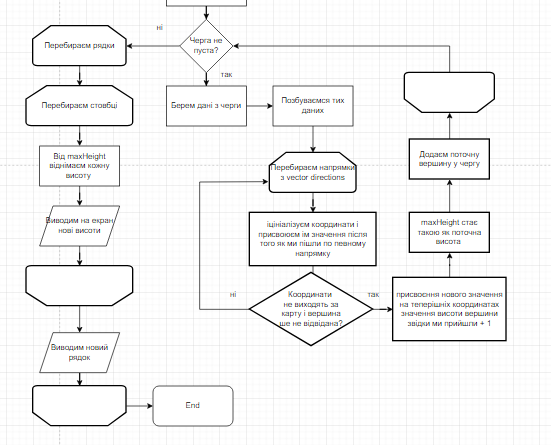
*Планований час 6 год*

**Завдання №2 - Algotester Lab78v1**

*Планований час 8 год*

**Завдання №3 - Algotester Lab5v3**

****

****

*Планований час 4 год*

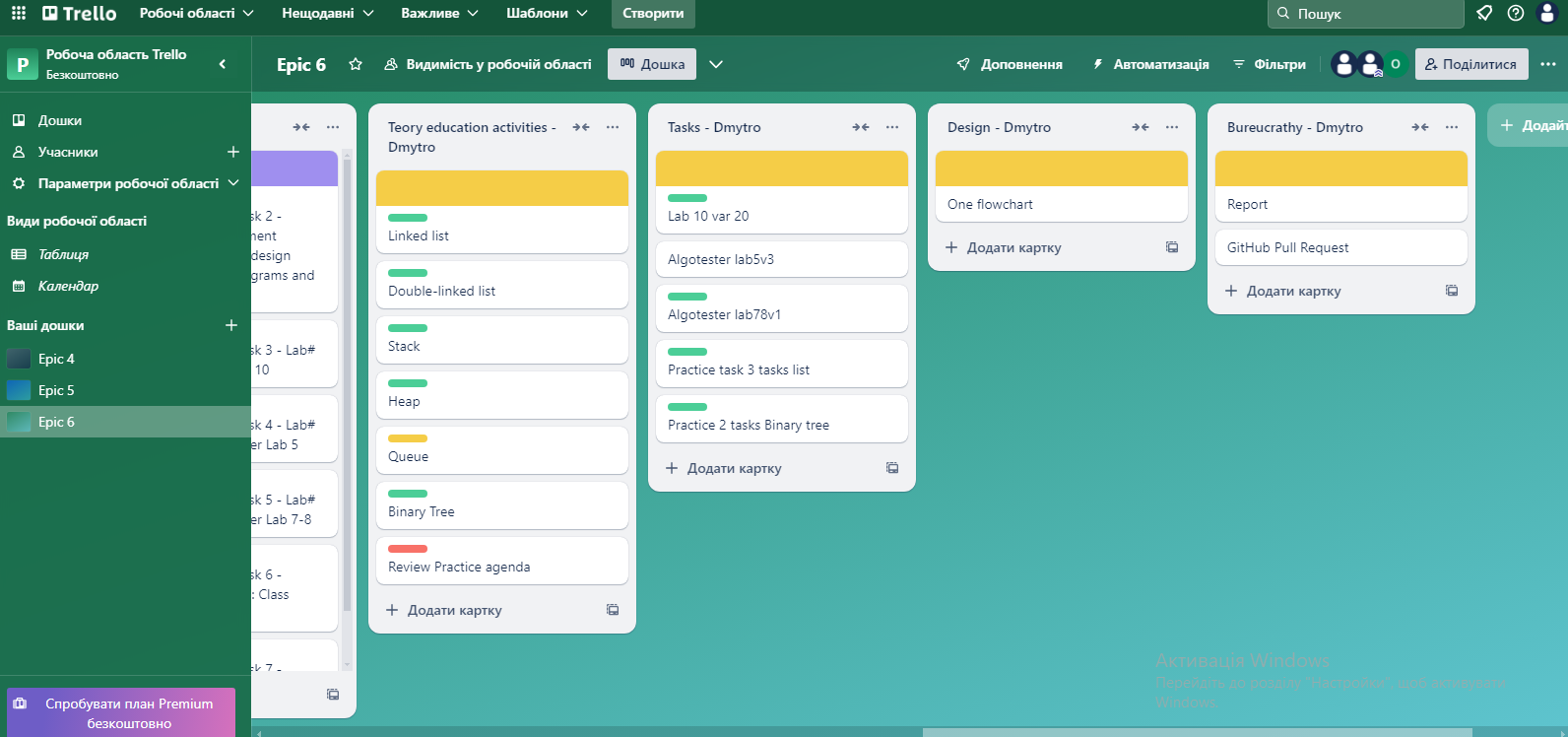
**Завдання №4 - Practice task 1-3 Linked List**

*Планований час 5 год*

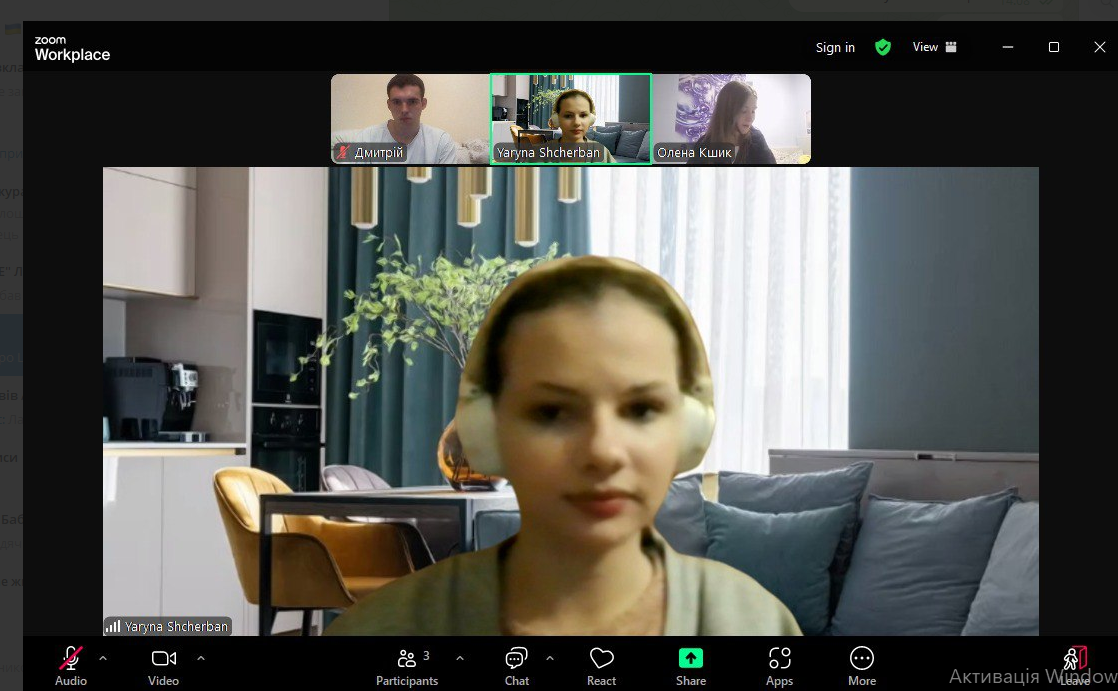
**Завдання №5 - Practice task 4-5 Binary Tree**

*Планований час 4 год*

1. *Конфігурація середовища до виконання завдань:*



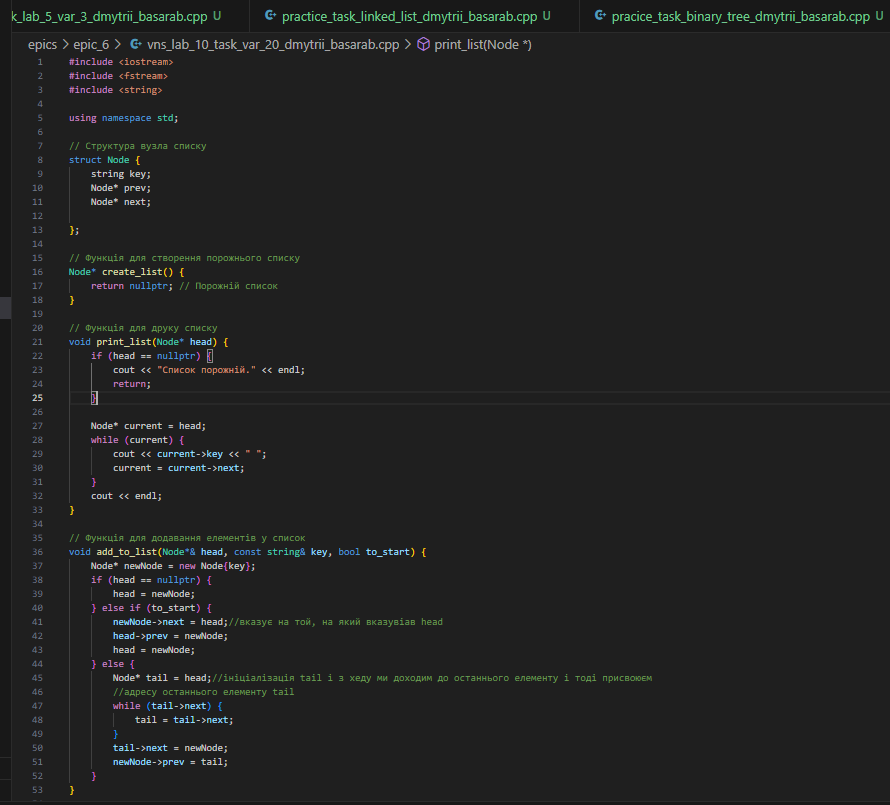
*Trello*

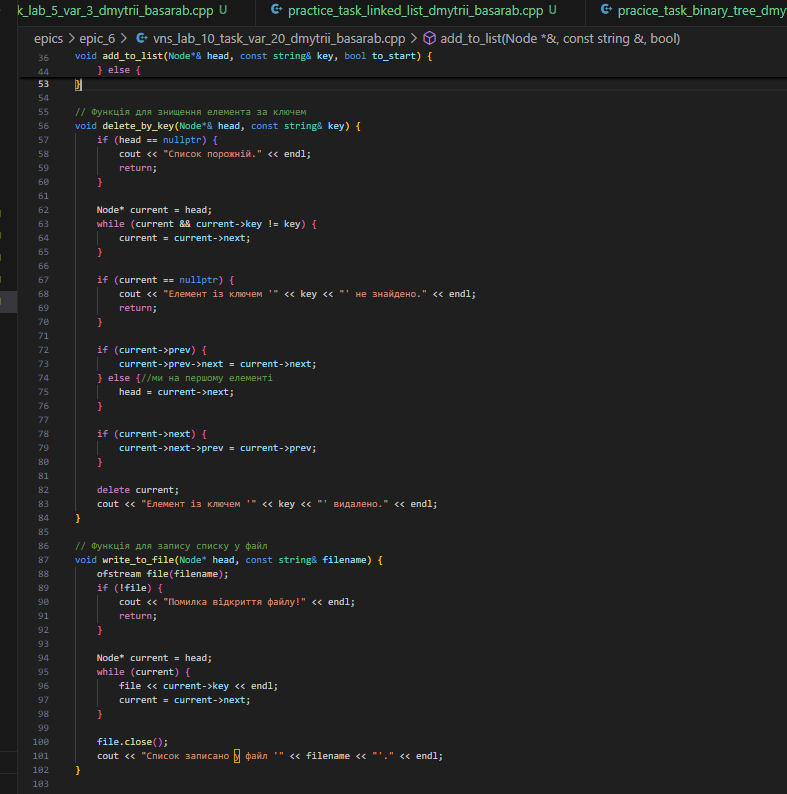


*Робота в команді*

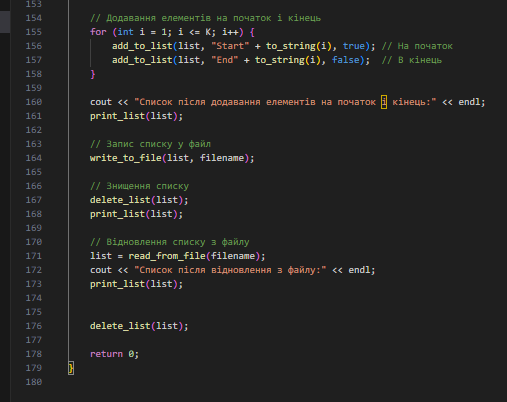
1. *Код програм з посиланням на зовнішні ресурси:*

**Завдання №1 - VNS Lab 10 var 20**

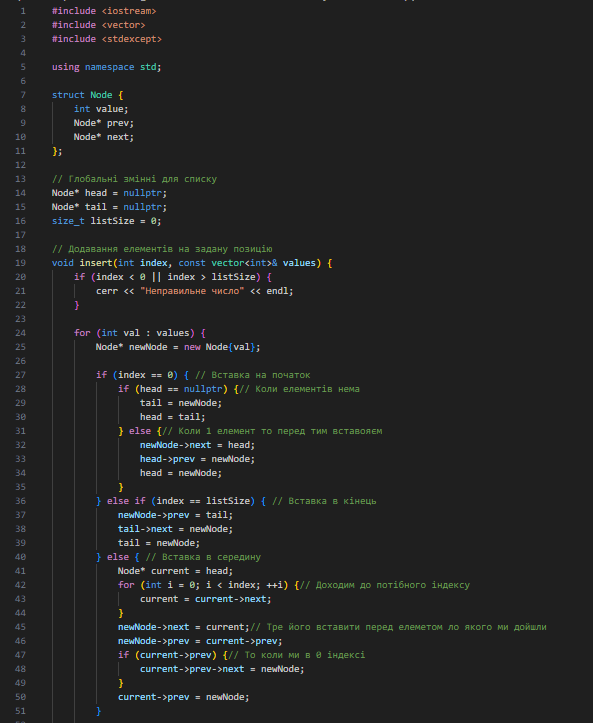
****

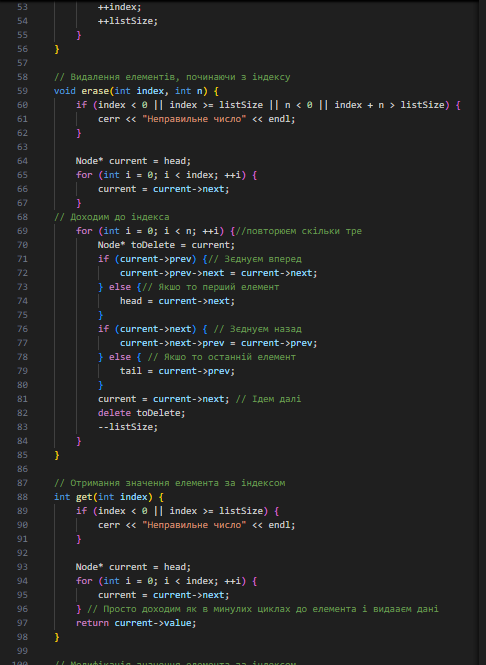
****

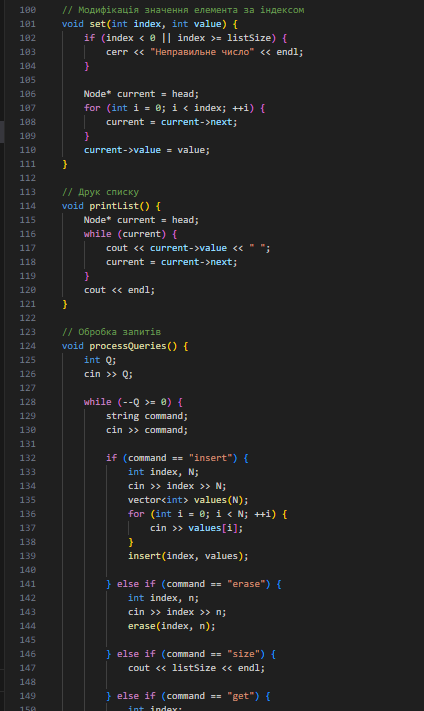
****

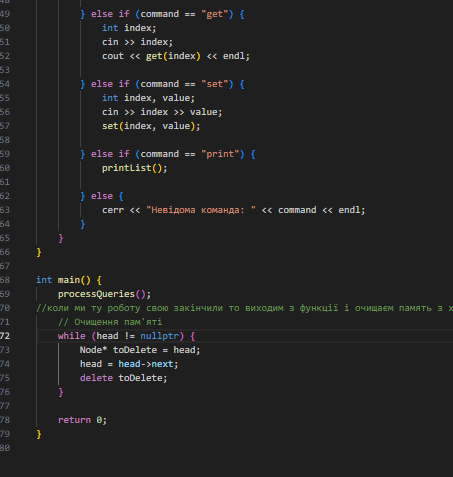
****

**Завдання №2 - Algotester Lab78v1**

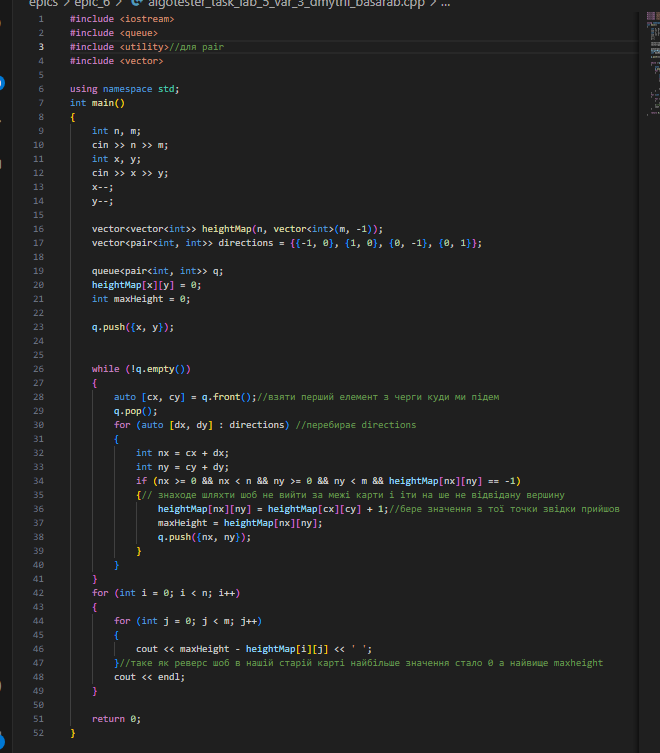
****

****

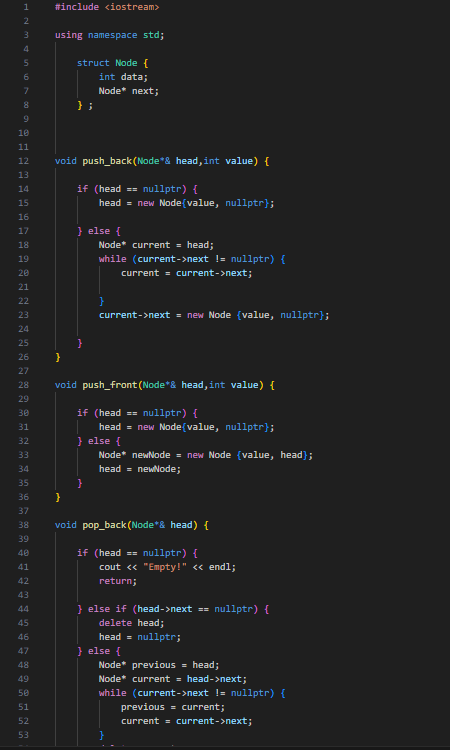
****

****

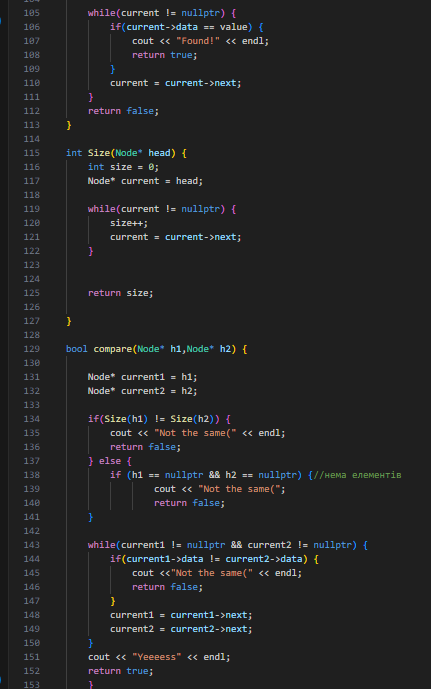
**Завдання №3 - Algotester Lab5v3**

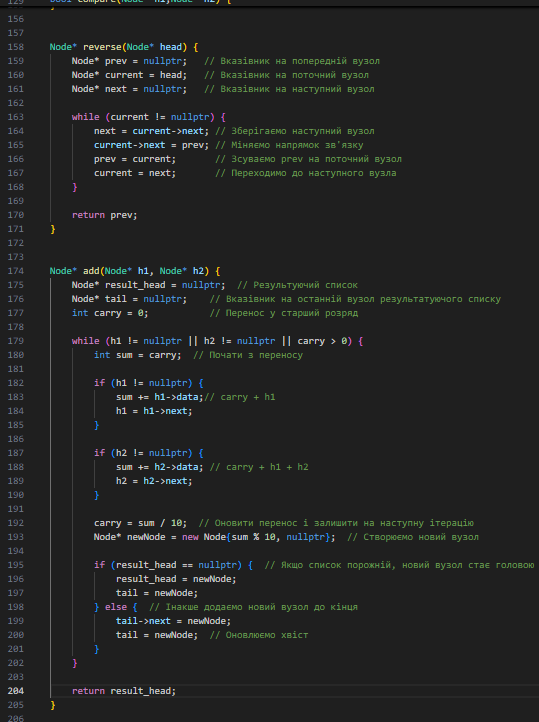
****

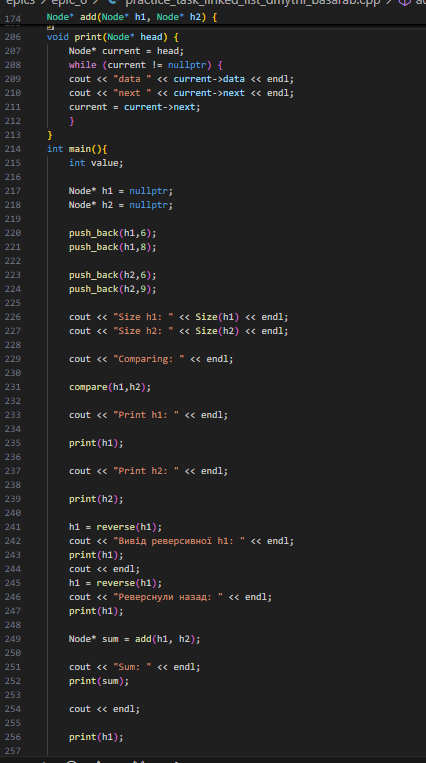
**Завдання №4 - Practice task 1-3 Linked List**

****

****

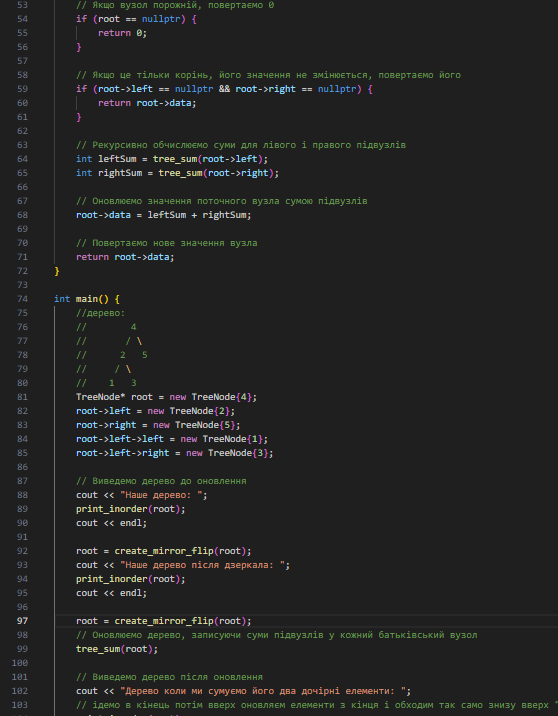
****

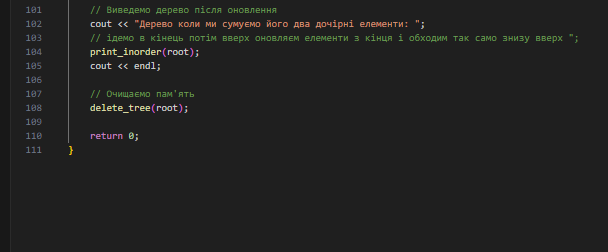
****

****

**Завдання №5 - Practice task 4-5 Binary Tree**

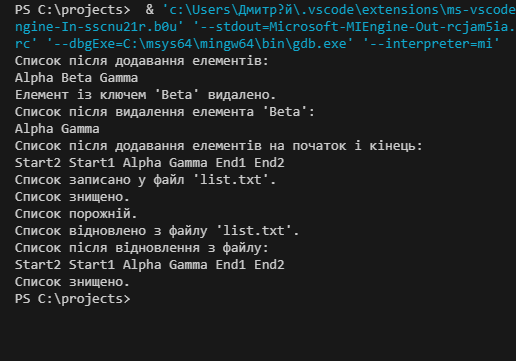
****

****

****

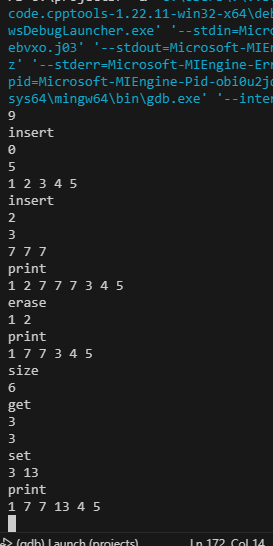
1. *Результати виконання завдань, тестування та фактично затрачений час:*

**Завдання №1 - VNS Lab 10 var 20**

****

*Фактично затрачений час 6 год*

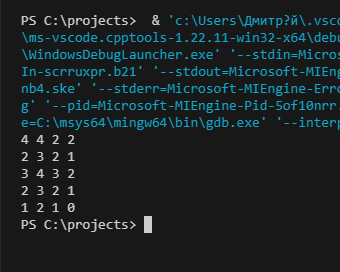
**Завдання №2 - Algotester Lab78v1**

****

****

*Фактично затрачений час 5 год*

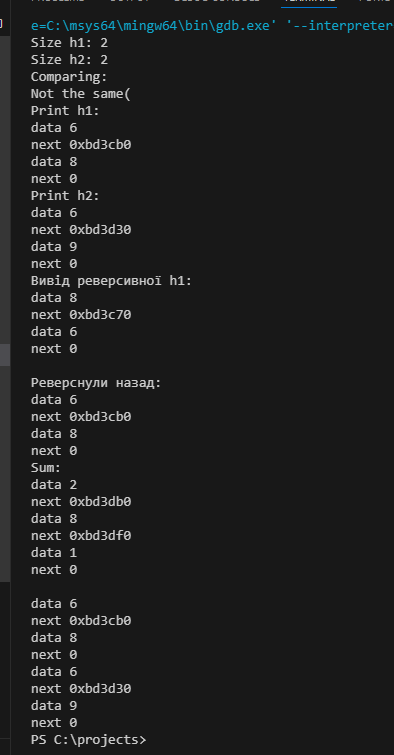
**Завдання №3 - Algotester Lab5v3**

****

****

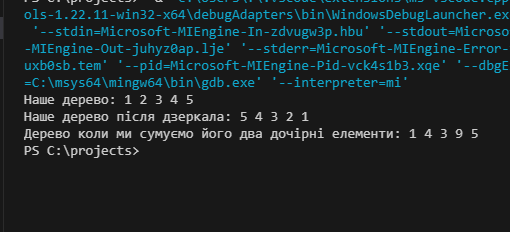
*Фактично затрачений час 4 год*

**Завдання №4 - Practice task 1-3 Linked List**

****

*Фактично затрачений час 7 год*

**Завдання №5 - Practice task 4-5 Binary Tree**

****

*Фактично затрачений час 8 год*

### **Висновки:** *На цій лабораторній роботі я навчився працювати з Динамічними структурами (Черга, Стек, Списки, Дерево), вивчив Алгоритми обробки динамічних структур*