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: «Програмування: алгоритм, програма, код. Системи числення. Двійкова система числення. Розробка та середовище розробки програми.»

***з дисципліни:*** «Основи програмування»

до:

Практичних Робіт до блоку № 6

**Виконав:**

Студент групи ШІ-11

Вербицький Юрій Віталійович

Львів 2024

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

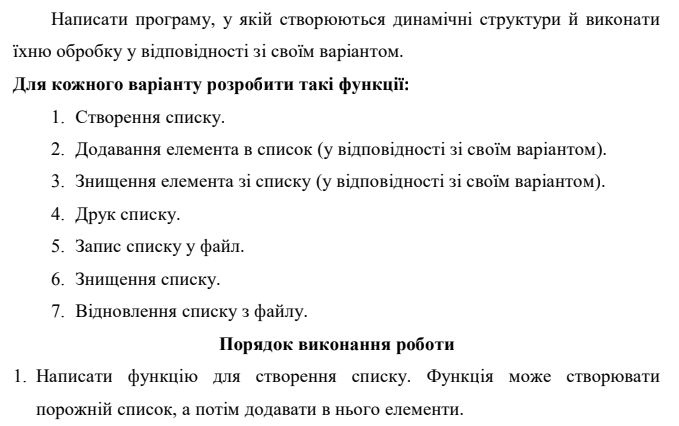
**Мета:** Засвоїти основи роботи з динамічними структурами даних, такими як черга, стек, списки та дерева. Ознайомитися з алгоритмами їх обробки для розв'язання різноманітних задач.

**Теоретичні відомості та джерела:**

Основи Динамічних Структур Даних:   
<https://studfile.net/preview/7013685/page:10/>   
Стек: <https://acode.com.ua/urok-111-stek-i-kupa/>   
Зв'язні Списки: <https://uk.myservername.com/linked-list-data-structure-c-with-illustration#Linked_List_In_C>   
Дерева: <https://foxminded.ua/binarne-derevo/>

**Виконання роботи:**

**VNS Lab 10v1:**



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

**Algotester Lab 5v2:**

В пустелі існує незвичайна печера, яка є двохвимірною. Її висота це N, ширина - M.

Всередині печери є пустота, пісок та каміння. Пустота позначається буквою О , пісок S і каміння X;

Одного дня стався землетрус і весь пісок посипався вниз. Він падає на найнижчу клітинку з пустотою, але він не може пролетіти через каміння.

Ваше завдання сказати як буде виглядати печера після землетрусу.

**Algotester Lab 7-8 v2:**

Ваше завдання - власноруч реалізувати структуру даних "Динамічний масив".

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 масиву, який треба вставити.

У третьому рядку N цілих чисел - масив, який треба вставити на позицію index.

Видалення:

Ідентифікатор - erase

Ви отримуєте 2 цілих числа - index, індекс 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 динамічному масиві.  
Визначення кількості зарезервованої пам’яті:

Ідентифікатор - capacity

Ви не отримуєте аргументів.

Ви виводите кількість зарезервованої пам’яті у динамічному масиві.

Ваша реалізація динамічного масиву має мати фактор росту (Growth factor) рівний 2.

Отримання значення i-го елементу

Ідентифікатор - get

Ви отримуєте ціле число - index, індекс елемента.

Ви виводите значення елемента за індексом. Реалізувати використовуючи перегрузку оператора []

Модифікація значення i-го елементу

Ідентифікатор - set

Ви отримуєте 2 цілих числа - індекс елемента, який треба змінити, та його нове значення. Реалізувати використовуючи перегрузку оператора []

Вивід динамічного масиву на екран

Ідентифікатор - print

Ви не отримуєте аргументів.

Ви виводите усі елементи динамічного масиву через пробіл.

Реалізувати використовуючи перегрузку оператора <<

**Class Practice Task:**

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 - insertВи 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, індекс елемента, з якого почати видалення та nn - 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 ii-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.  
Реалізувати використовуючи перегрузку оператора <<

**Class Practice Task:**

## 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

## 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

**Self Practice**

**Task** **Algotester Lab 5v2:**

У вас є карта гори розміром N×M.

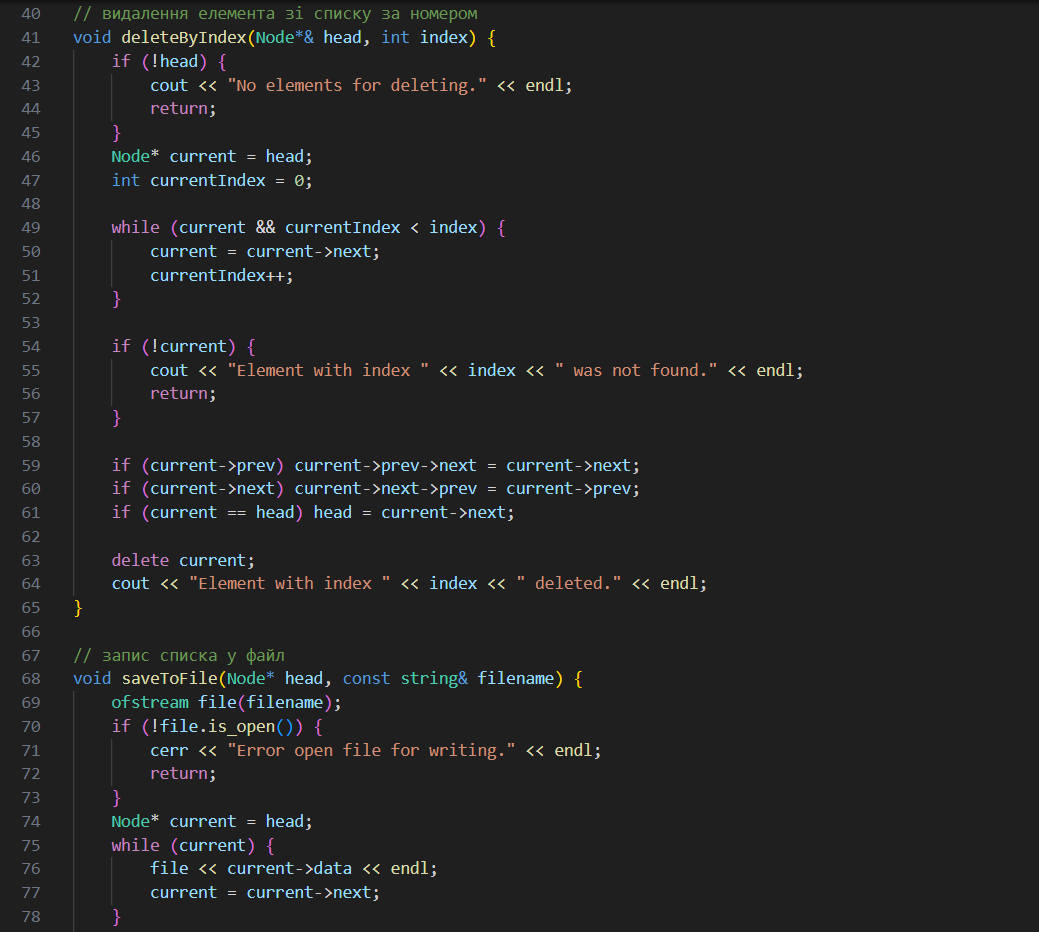
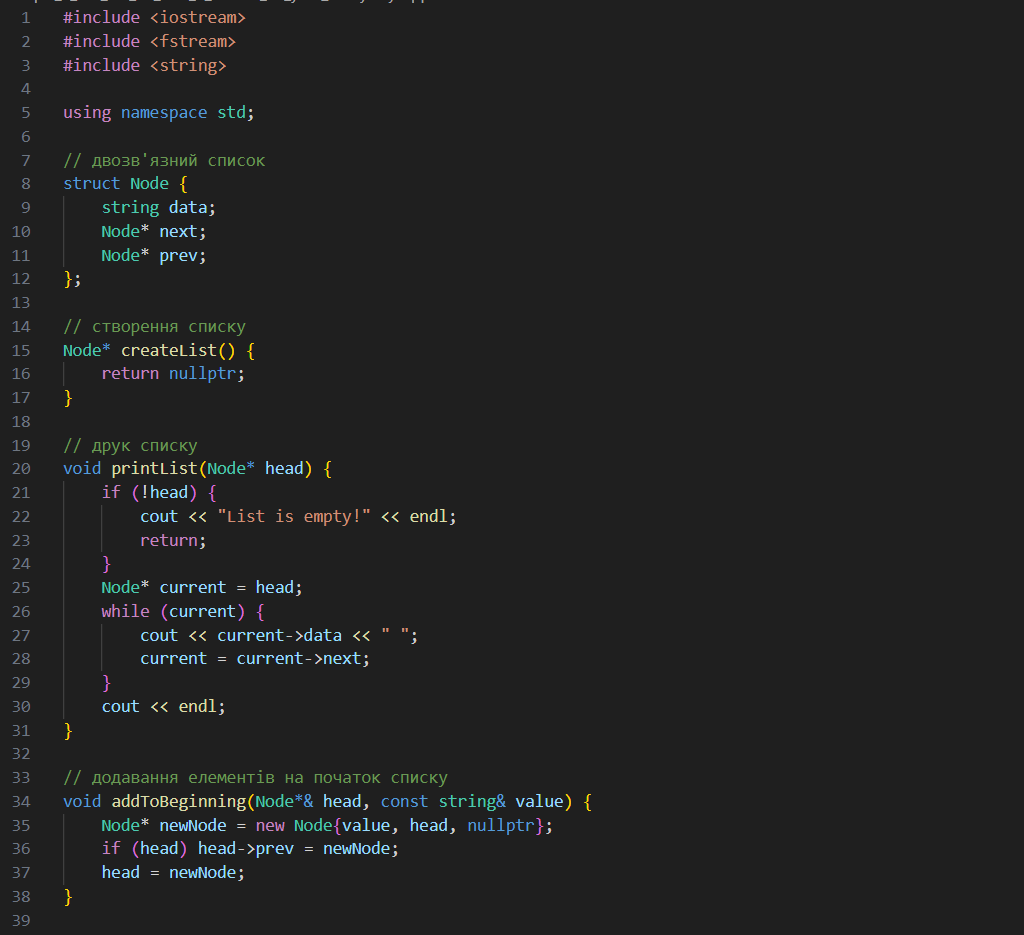
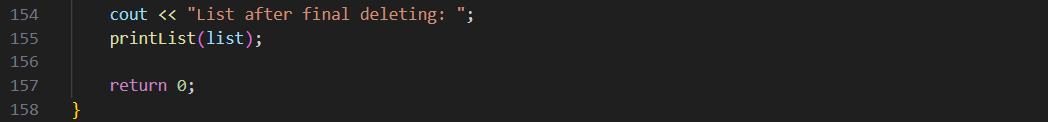
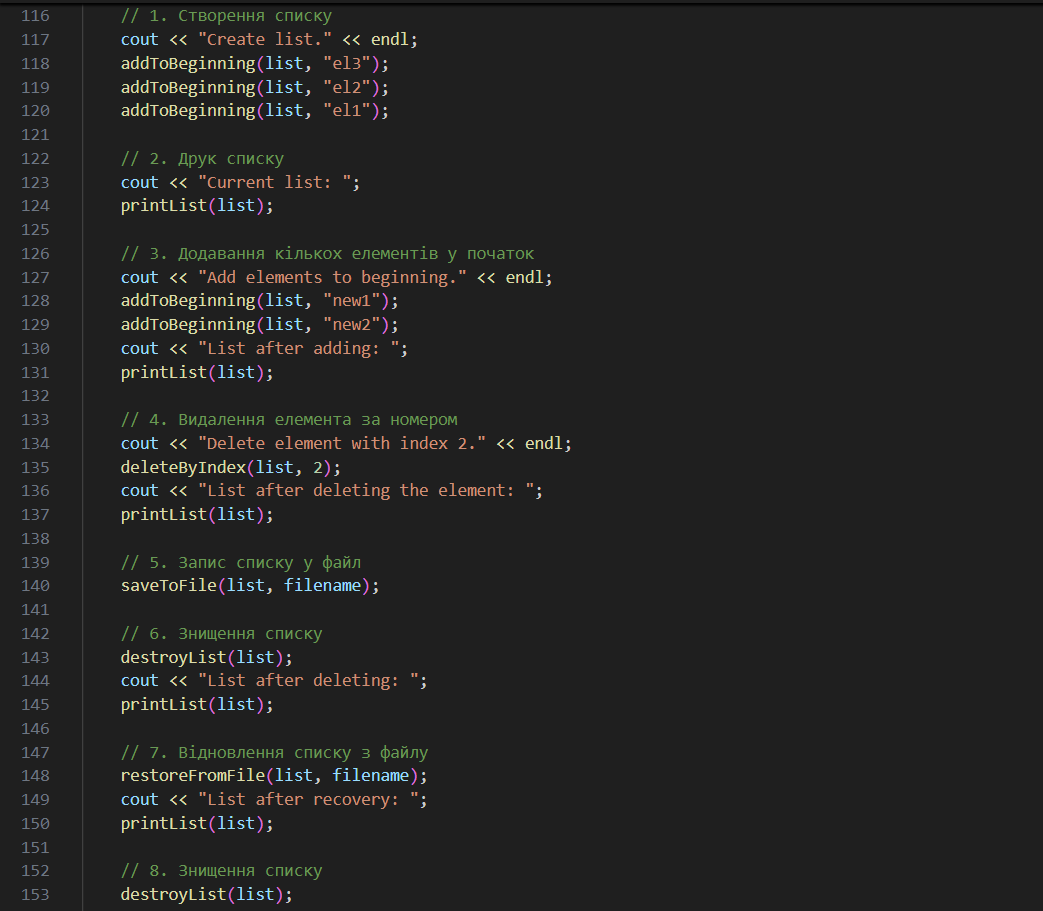
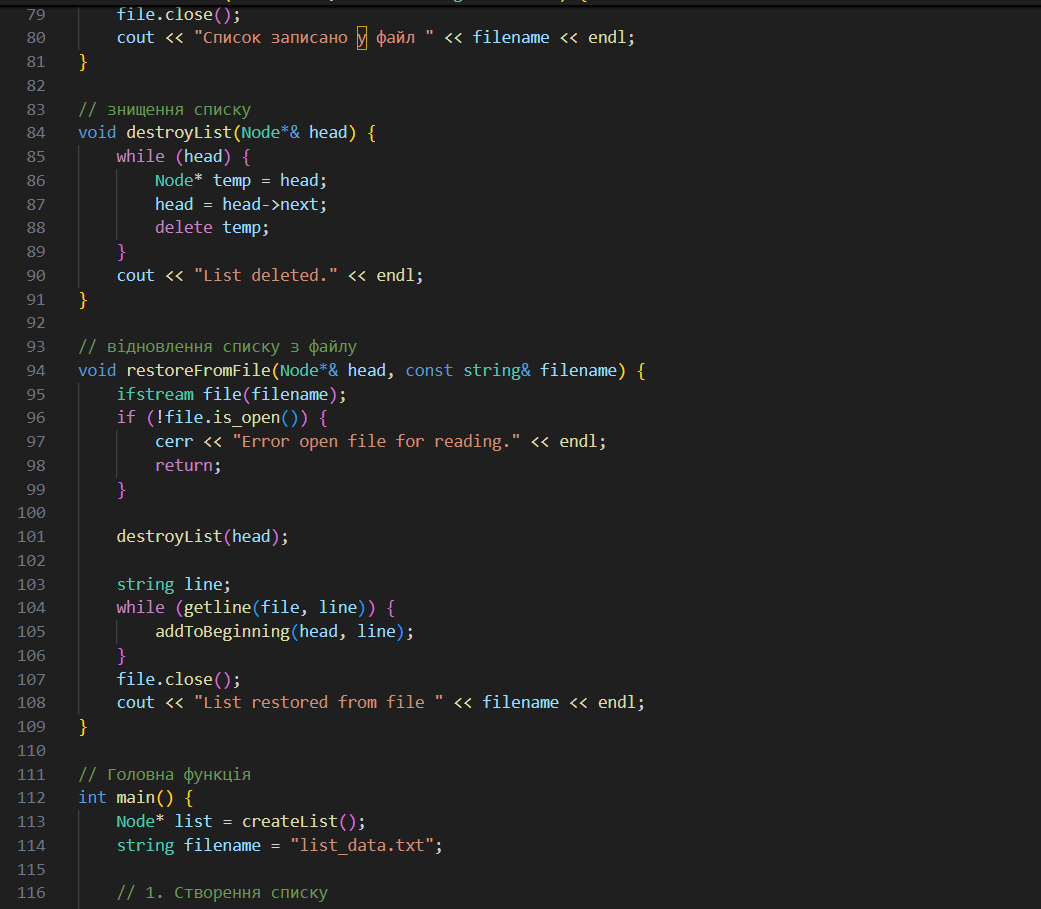
Також ви знаєте координати {x,y} , у яких знаходиться вершина гори.

Ваше завдання - розмалювати карту таким чином, щоб найнижча точка мала число 0, а пік гори мав найбільше число.

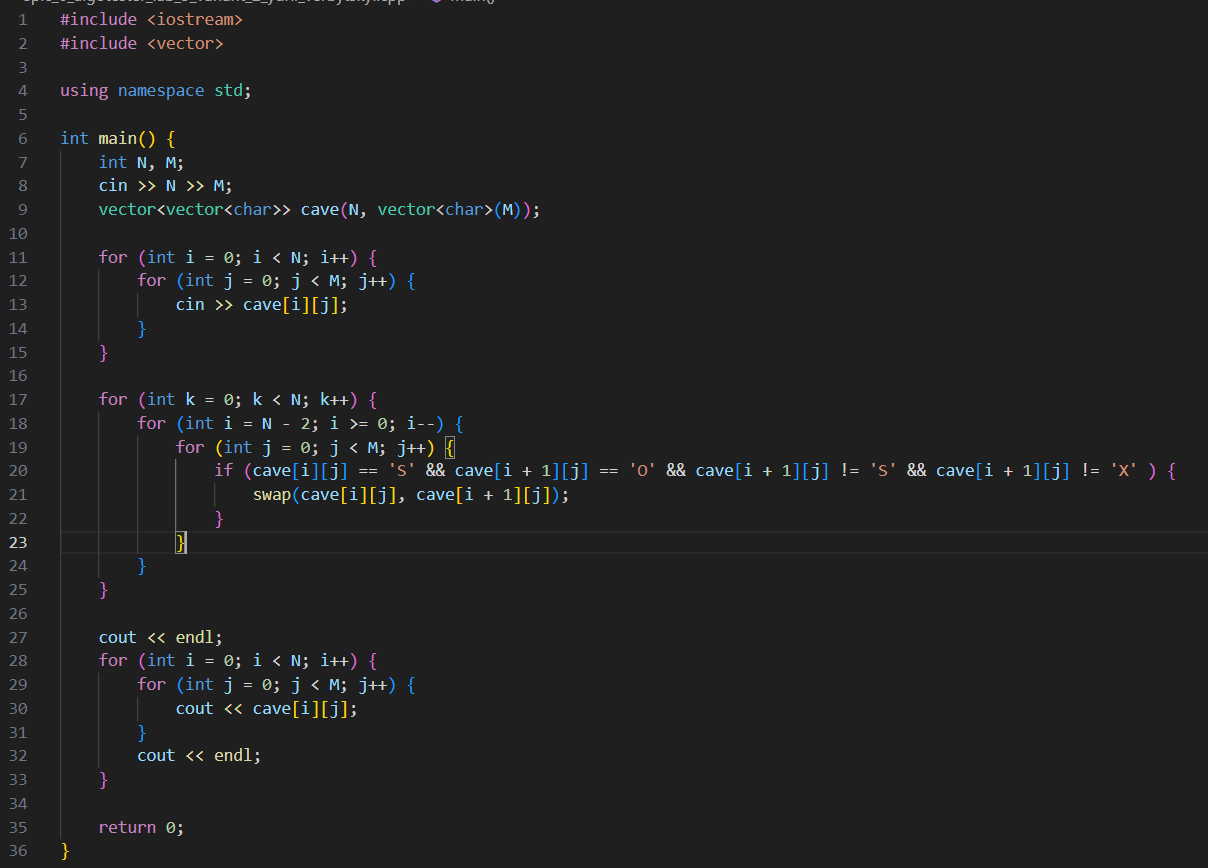
Клітинкі які мають суміжну сторону з вершиною мають висоту на один меншу, суміжні з ними і не розфарбовані мають ще на 1 меншу висоту і так далі.

**Код програми з посиланням на зовнішні ресурси**

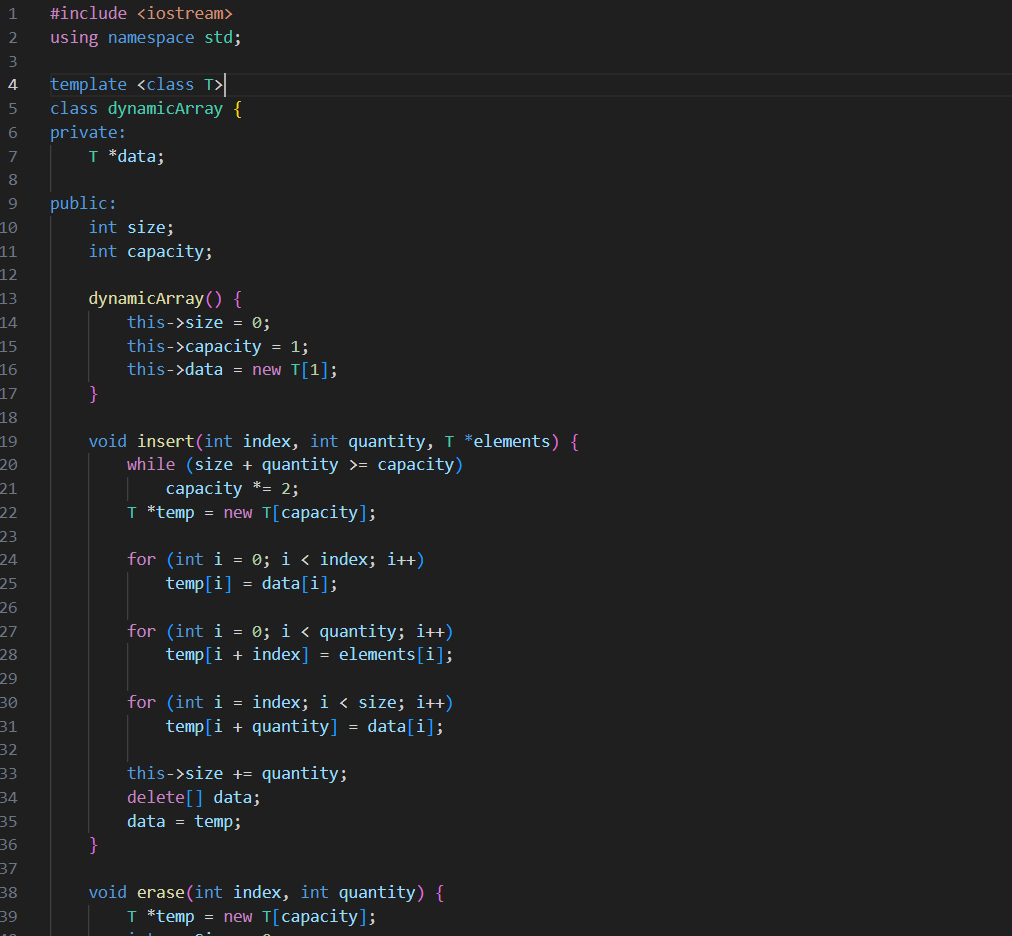
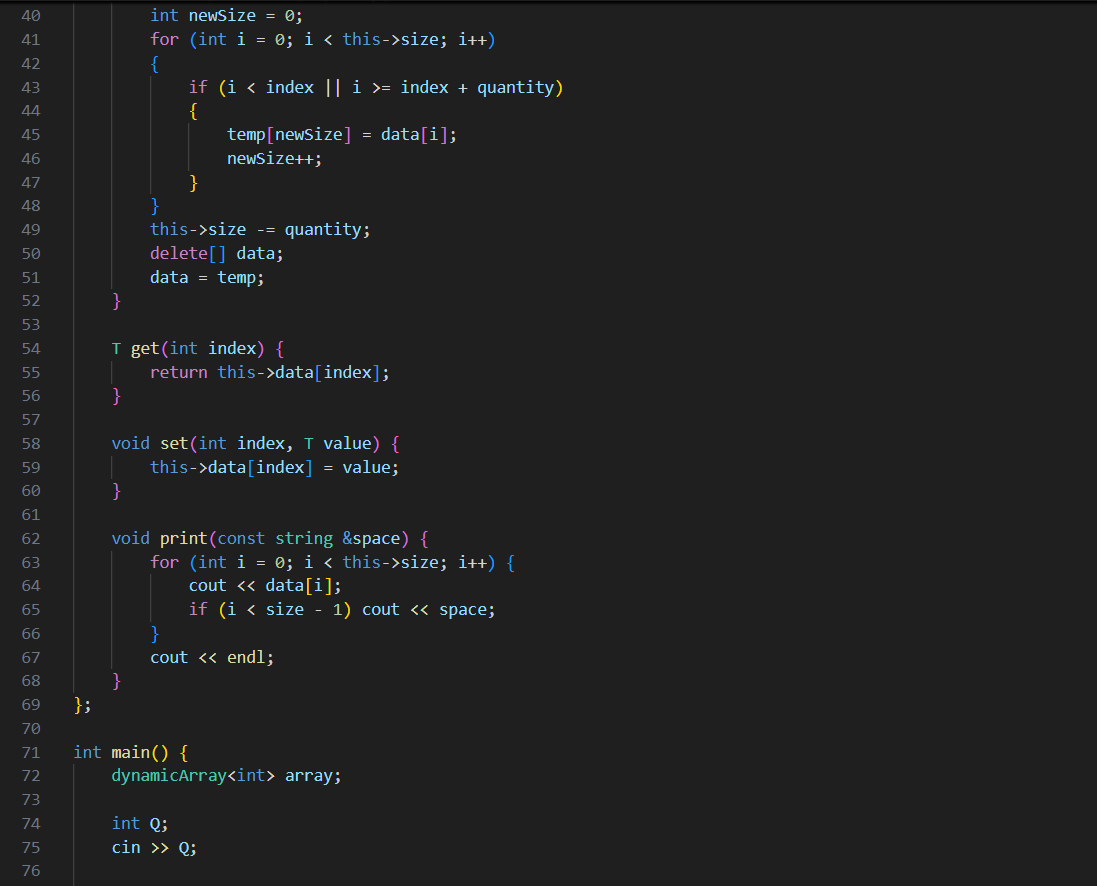
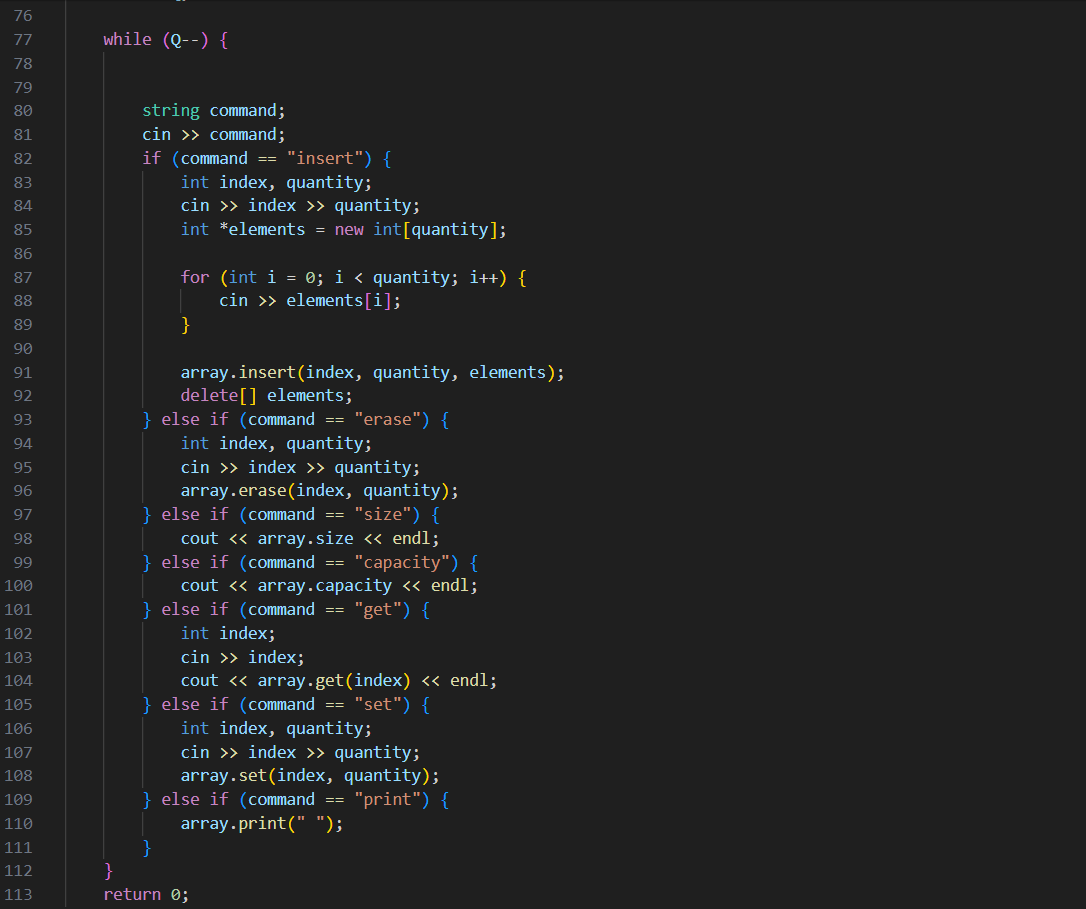
**VNS Lab 10v1:**

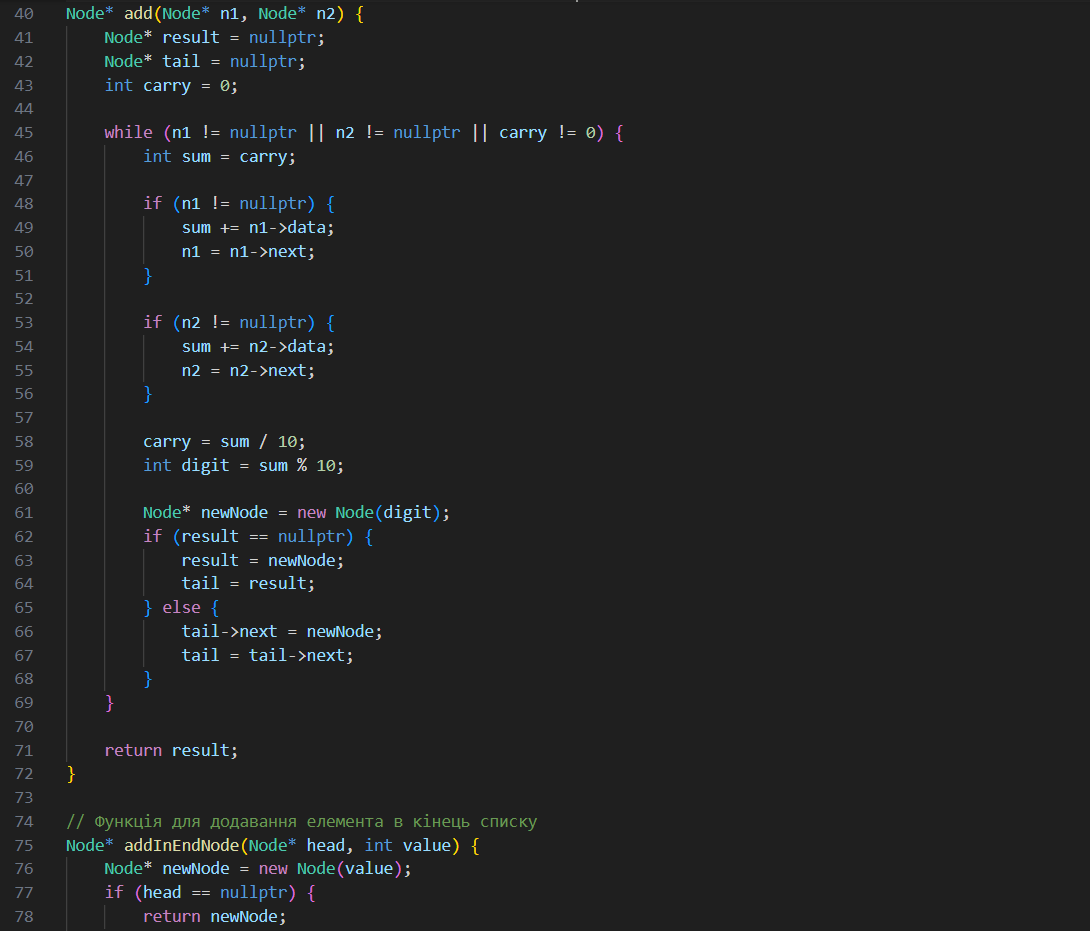
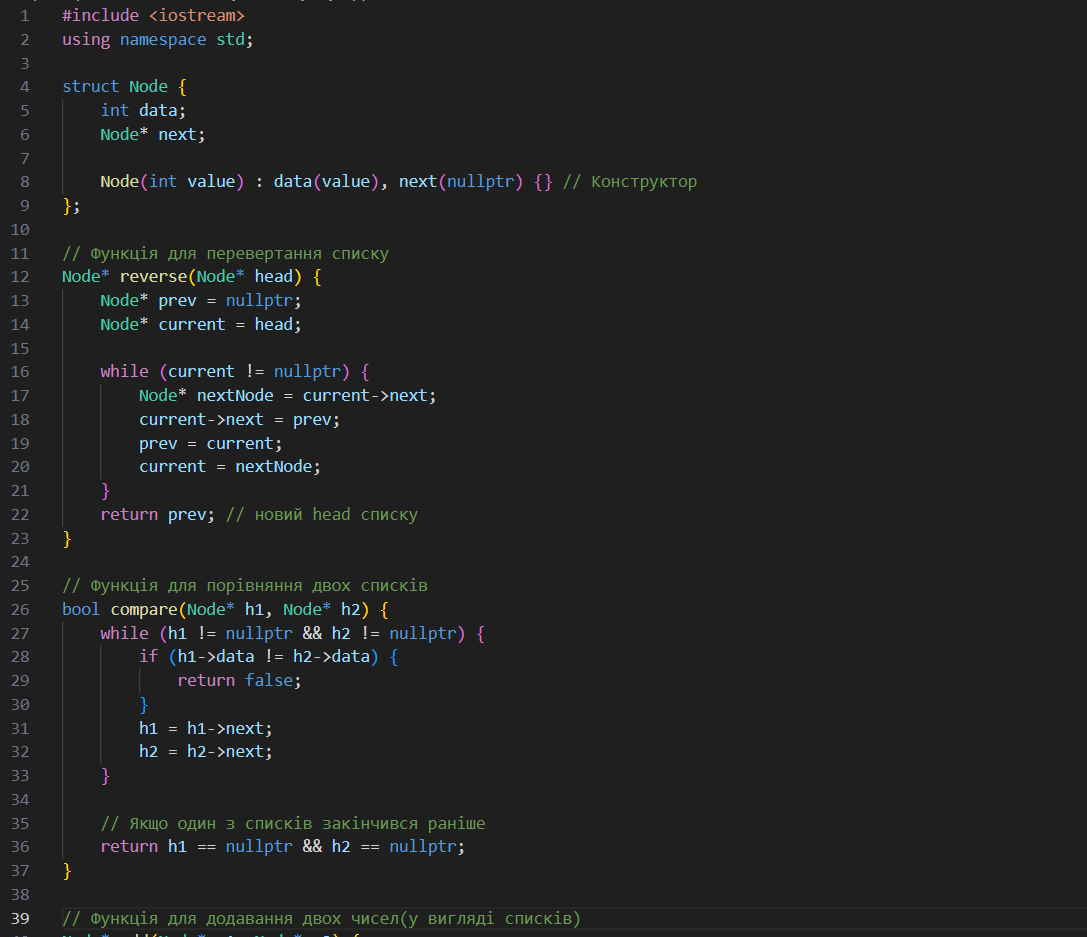
**  
**

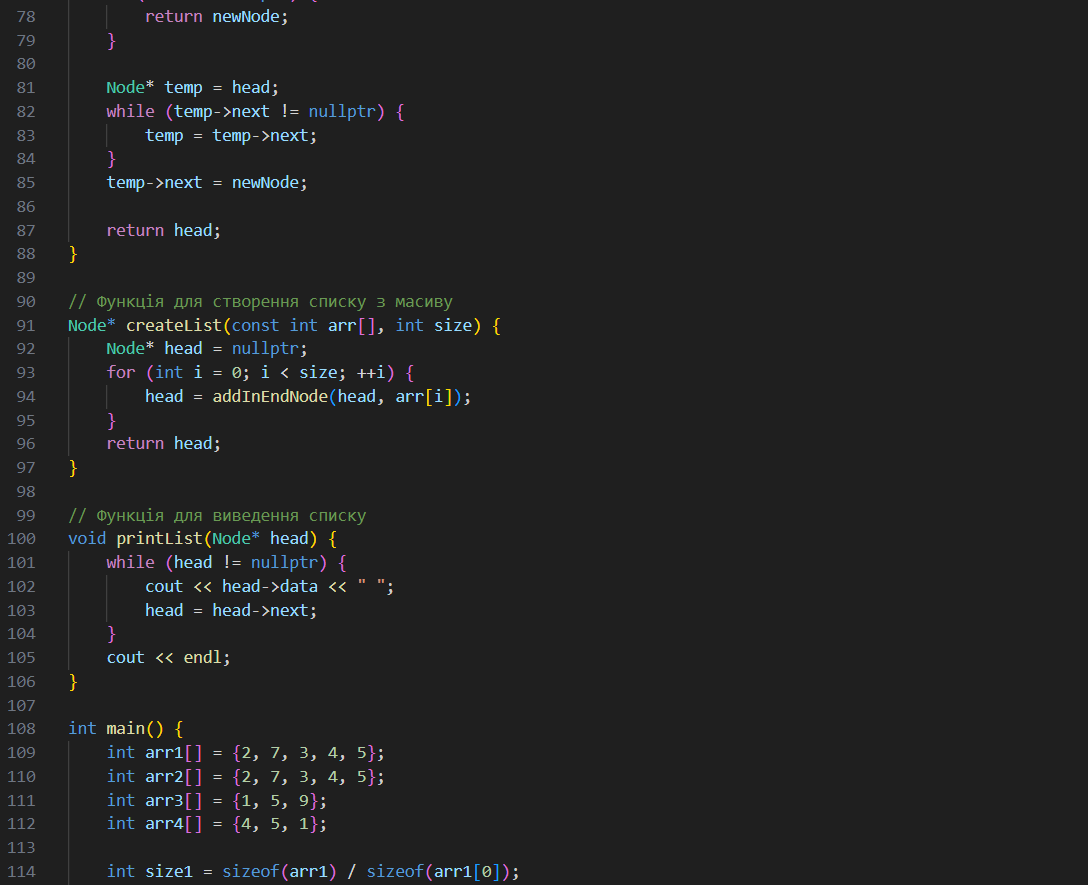
**Витрачено ~3 години**

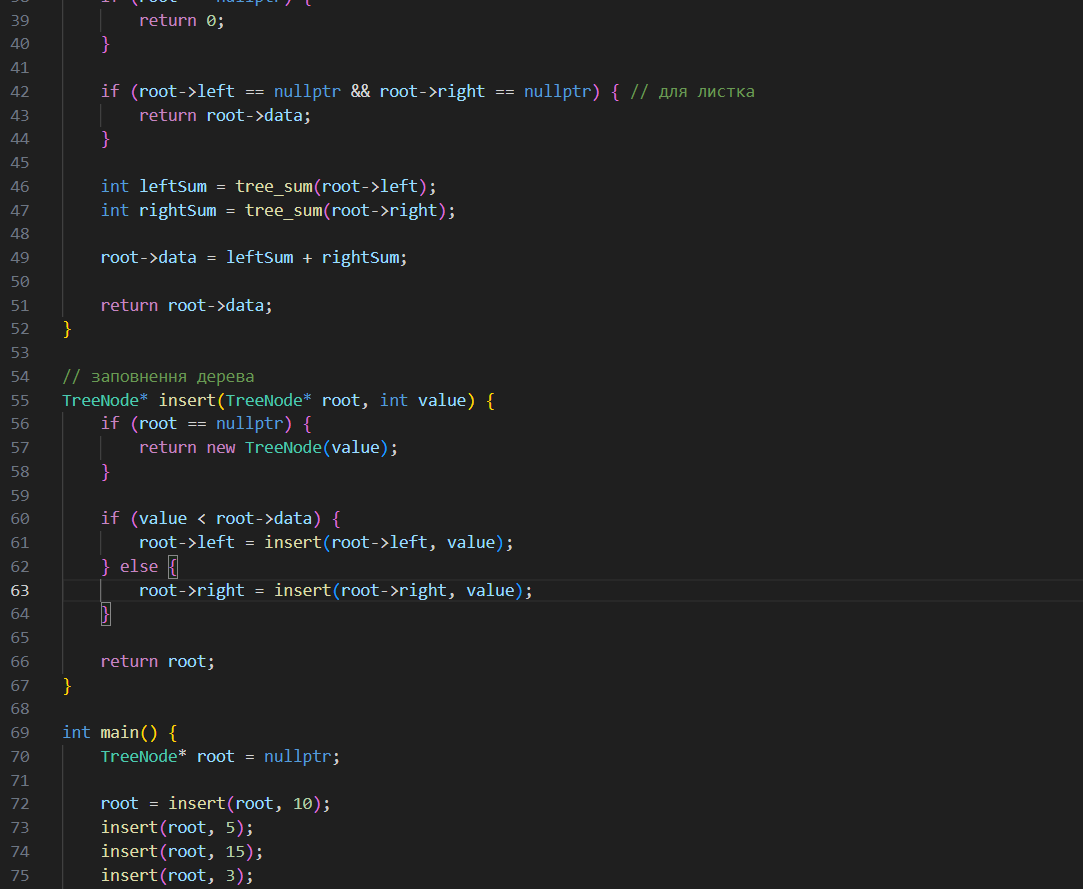
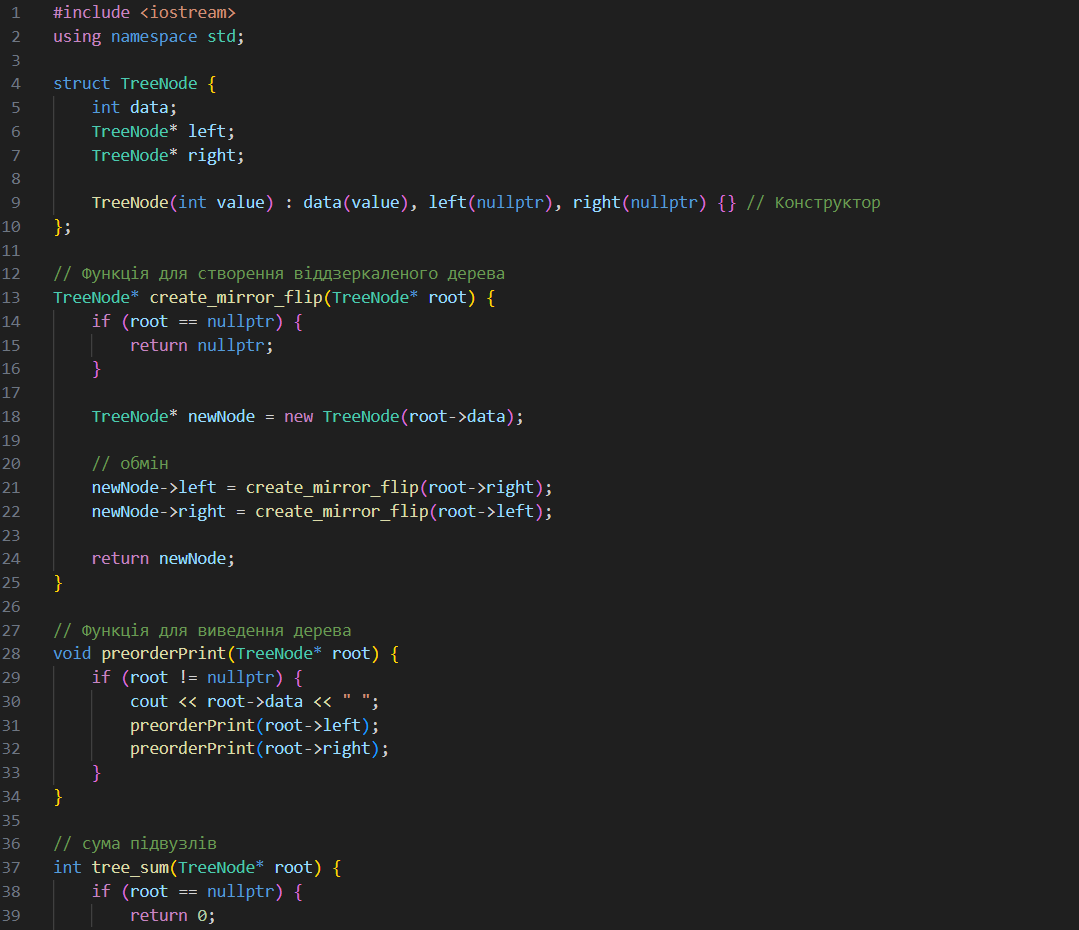
**Algotester Lab 5v2:  
 Витрачено ~1 годину**

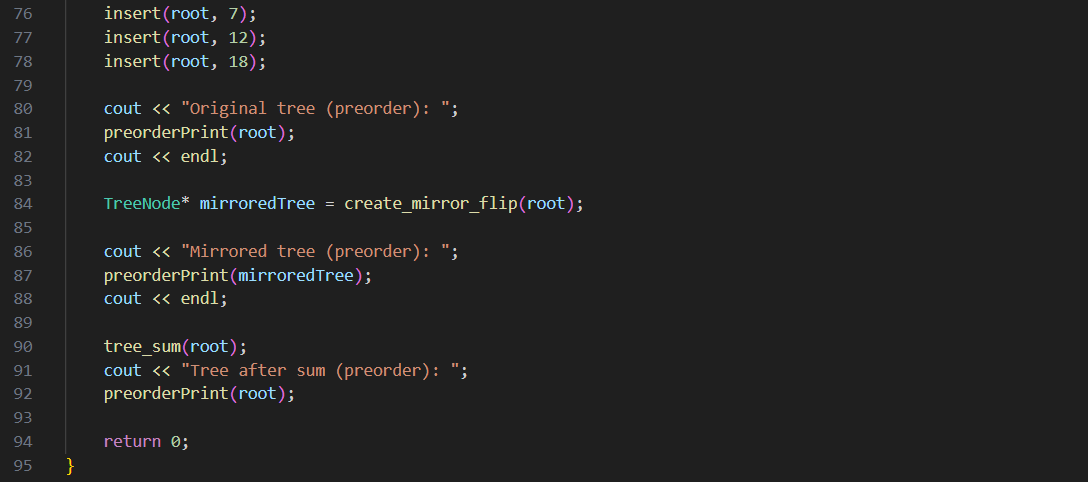
**Algotester Lab 7-8 v2:**

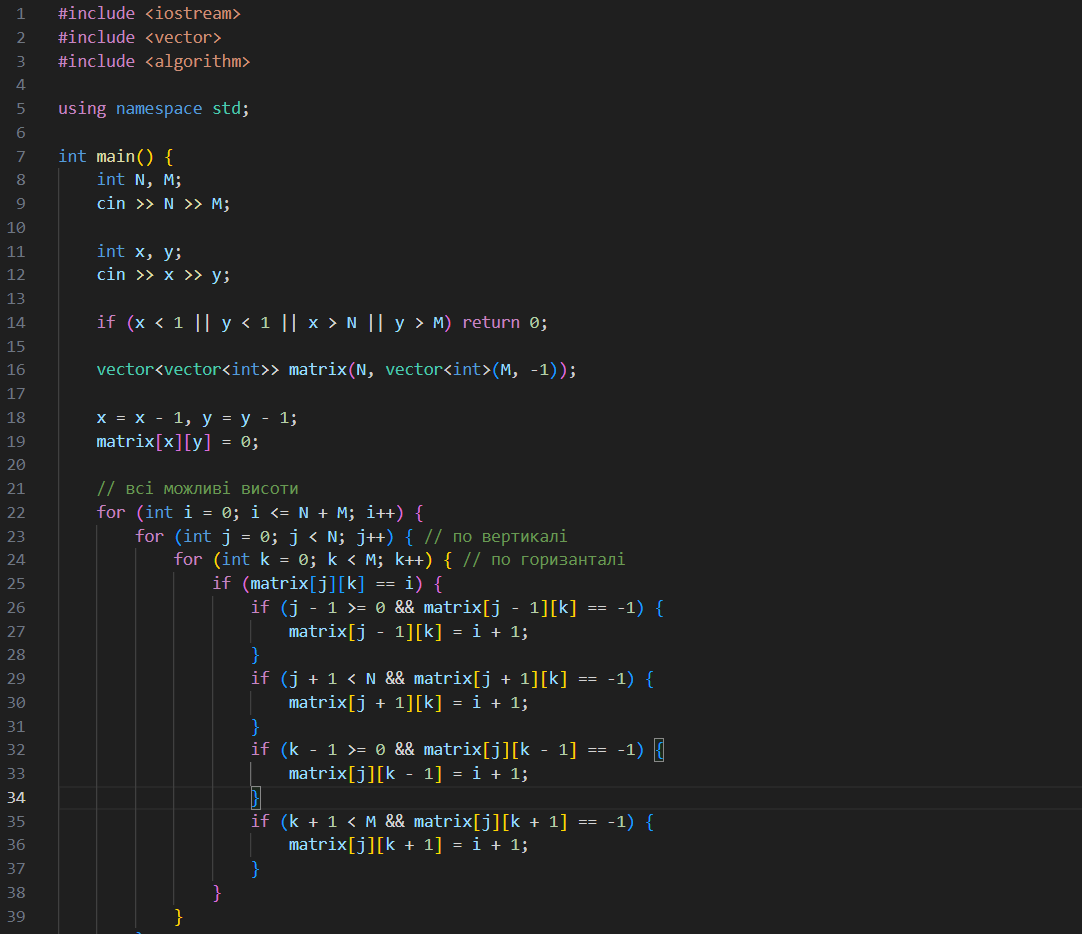
**  
  
  
Витрачено ~4 години**

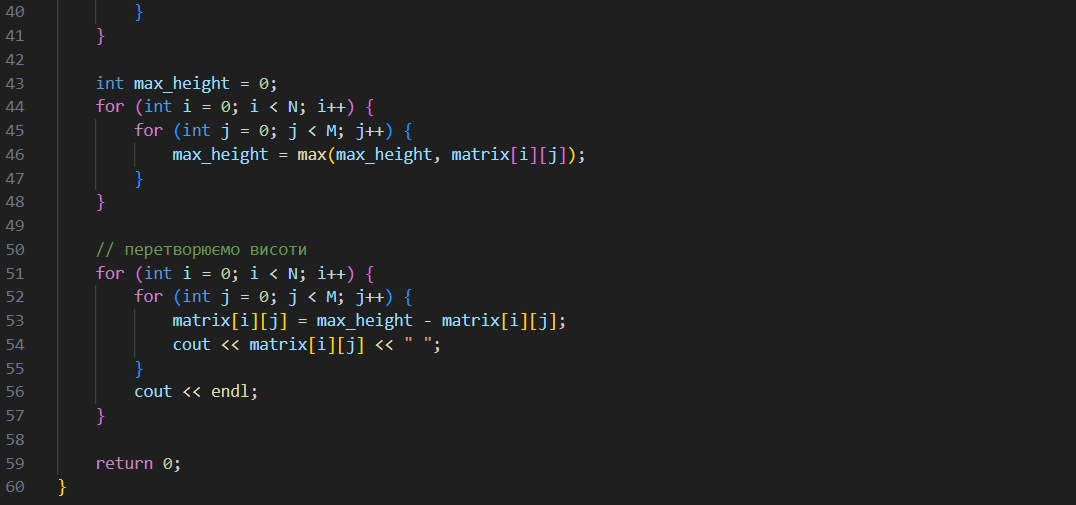
**Class Practice Task 1: **

**Витрачено ~4 години**

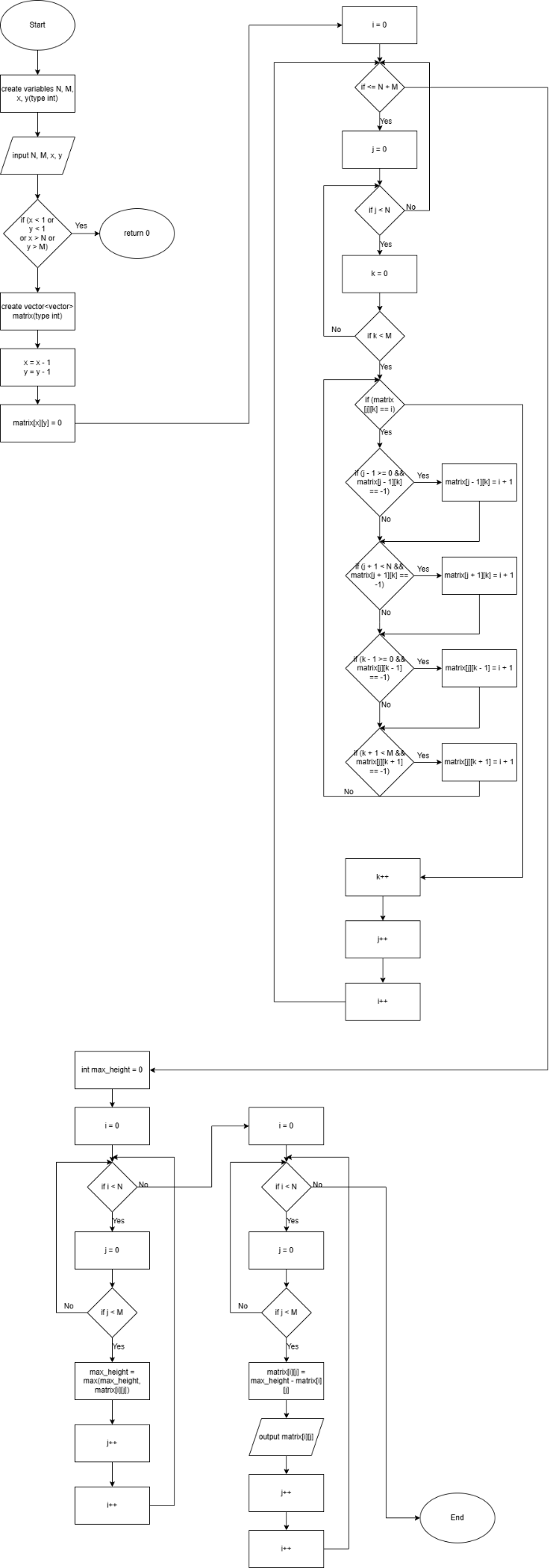
**Class Practice Task 2: **

**  
Витрачено ~3,5 години**

**SelfPractice:  **

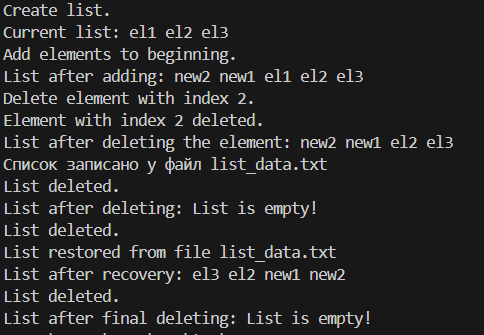
**Витрачено ~3 години**

**Block-scheme for task Algotester Lab 5v3**

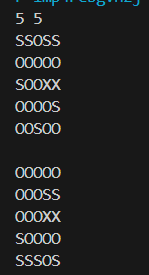


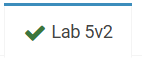
**Результати виконаних завдань, тестування та фактично затрачений час**

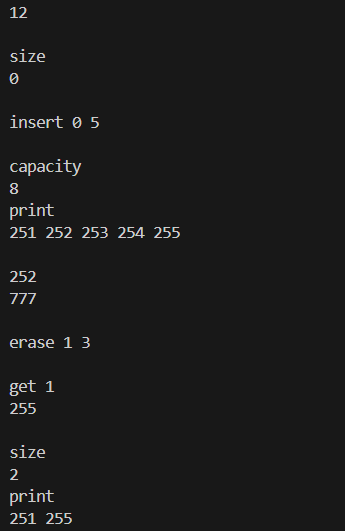
**VNS Lab 10v1: ~4 години**



**Algotester Lab 5v2: ~1 годину**

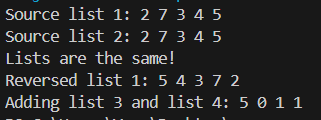
****



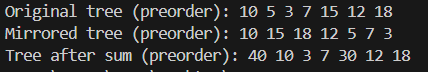
**Algotester Lab 7-8 v2: ~5 годин  
**



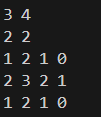
**Class Practice Task 1: ~2 години**

****

**Class Practice Task 2: ~3 години**

****

**SelfPractice:  50 хвилин**

**Висновок:**

Отже, в межах цього епіка я старався зрозуміти, що таке списки, дерева та як їх реалізовувати в коді. Практикувався з записом даних у файли, а також покращив роботу з масивами та алгоритмами.

Посилання на pull request: