Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту

A blue and white logo

Description automatically generated

**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

**Виконала:**

Студентка групи ШІ-13

Щербан Ярина Олегівна

Львів - 2024

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

**Мета :** навчитись працювати з динамічними структурами, реалізувати зв’язний список та навчитись працювати з бінарними деревами.

**Теоретичні відомості:**

1. Основи Динамічних Структур Даних:
   * Вступ до динамічних структур даних: визначення та важливість
   * Виділення пам'яті для структур даних (stack і heap)
2. Стек:
   * Визначення та властивості стеку
   * Операції push, pop, top: реалізація та використання
   * Приклади використання стеку: обернений польський запис, перевірка балансу дужок
   * Переповнення стеку
3. Черга:
   * Визначення та властивості черги
   * Операції enqueue, dequeue, front: реалізація та застосування
4. Зв'язні Списки:
   * Визначення однозв'язного та двозв'язного списку
   * Принципи створення нових вузлів, вставка між існуючими, видалення, створення кільця(circular linked list)
   * Основні операції: обхід списку, пошук, доступ до елементів, об'єднання списків
   * Приклади використання списків: управління пам'яттю, FIFO та LIFO структури
5. Дерева:
   * Бінарні дерева: вставка, пошук, видалення
   * Обхід дерева: в глибину (preorder, inorder, postorder), в ширину
   * Застосування дерев: дерева рішень, хеш-таблиці
   * Складніші приклади дерев: AVL, Червоно-чорне дерево
6. Алгоритми Обробки Динамічних Структур:
   * Основи алгоритмічних патернів: ітеративні, рекурсивні
   * Алгоритми пошуку, сортування даних, додавання та видалення елементів

**Опрацювання теоретичного матеріалу :**

1. Вивчення мови C++ за допомогою сайтів : https://www.w3schools.com/, <https://acode.com.ua/>
2. Робота з блок-схемами та Draw io <https://www.programiz.com/article/flowchart-programming>
3. Опрацювала відео про бінарні дерева :

<https://youtu.be/zuuAPYiMYDA?feature=shared>

**Виконання роботи:**

1. *Опрацювання завдання та вимог до програми та середовища*

**Завдання №1 Епік 6 : Практичне завдання :**

## Опис : 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

**Завдання №2 VNS Lab 10 - Task 1 – Variant 21 :**

Опис :

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.

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

символів). Сформувати двонаправлений список. Знищити елементи перед і

після елемента із заданим ключем. Додати по К елементів у початок й у

кінець списку.

**Завдання №3 Algotester Lab 5 - Variant 1 :**

Опис : У світі Атод сестри Ліна і Рілай люблять грати у гру. У них є дошка із 8-ми рядків і 8-ми стовпців. На перетині ii-го рядкa і j-го стовпця лежить магічна куля, яка може світитись магічним світлом (тобто у них є 64 кулі). На початку гри деякі кулі світяться, а деякі ні... Далі вони обирають N куль і для кожної читають магічне заклиння, після чого всі кулі, які лежать на перетині стовпця і рядка обраної кулі міняють свій стан (ті що світяться - гаснуть, ті, що не світяться - загораються).

Також вони вирішили трохи Вам допомогти і придумали спосіб як записати стан дошки одним числом a із 8-ми байт, а саме (див. Примітки):

* Молодший байт задає перший рядок матриці;
* Молодший біт задає перший стовпець рядку;
* Значення біту каже світиться куля чи ні (0 - ні, 1 - так);

Тепер їх цікавить яким буде стан дошки після виконання N заклинань і вони дуже просять Вас їм допомогти.

**Завдання №4 Algotester Lab 7-8 - Variant 2 :**

Опис : Ваше завдання - власноруч реалізувати структуру даних "Динамічний масив".  
Ви отримаєте Q запитів, кожен запит буде починатися зі слова-ідентифікатора, після якого йдуть його аргументи.

Умова **: Для того щоб отримати 50% балів за лабораторну достатньо написати свою структуру.  
Для отримання 100% балів ця структура має бути написана як**[**шаблон класу**](https://cplusplus.com/doc/oldtutorial/templates/)**, у якості параметру використати intint.  
Використовувати STL заборонено.**

**Завдання №5 Algotester Lab 5 – Variant 2 :**

Опис : 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 SS 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.

1. *Дизайн та планована оцінка часу виконання завдань*

**Завдання №1 Епік 6 : Практичне завдання :**

Запланований час виконання - 2 год

**Завдання №2 VNS Lab 10 - Task 1 – Variant 21 :**

Запланований час виконання - 2 год

**Завдання №3 Algotester Lab 5 - Variant 1 :**

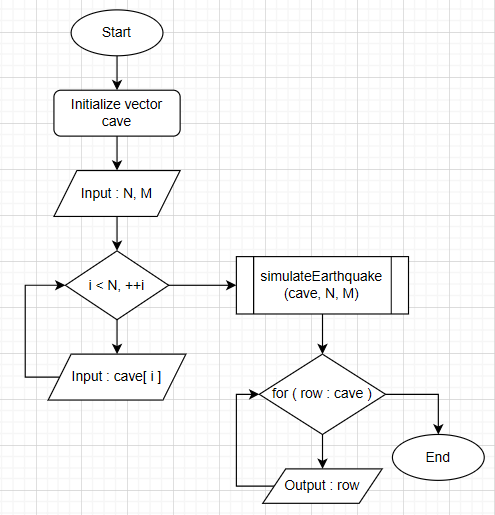
Запланований час виконання - 40 хв

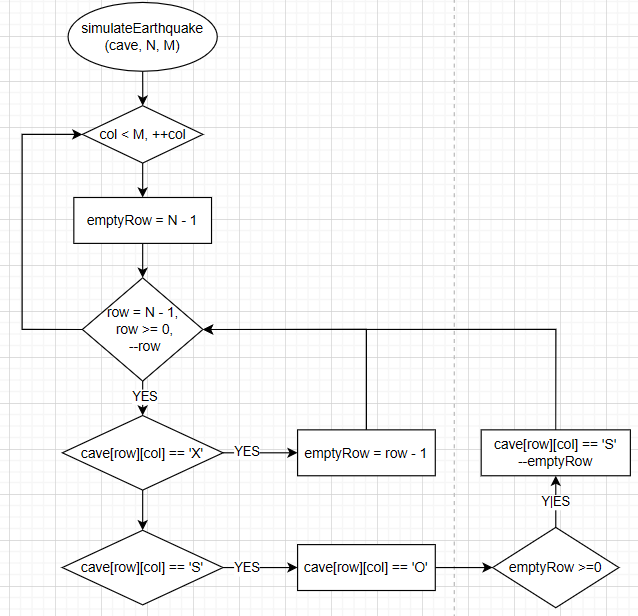
**Завдання №4 Algotester Lab 7-8 - Variant 2 :**

Запланований час виконання - 2 год

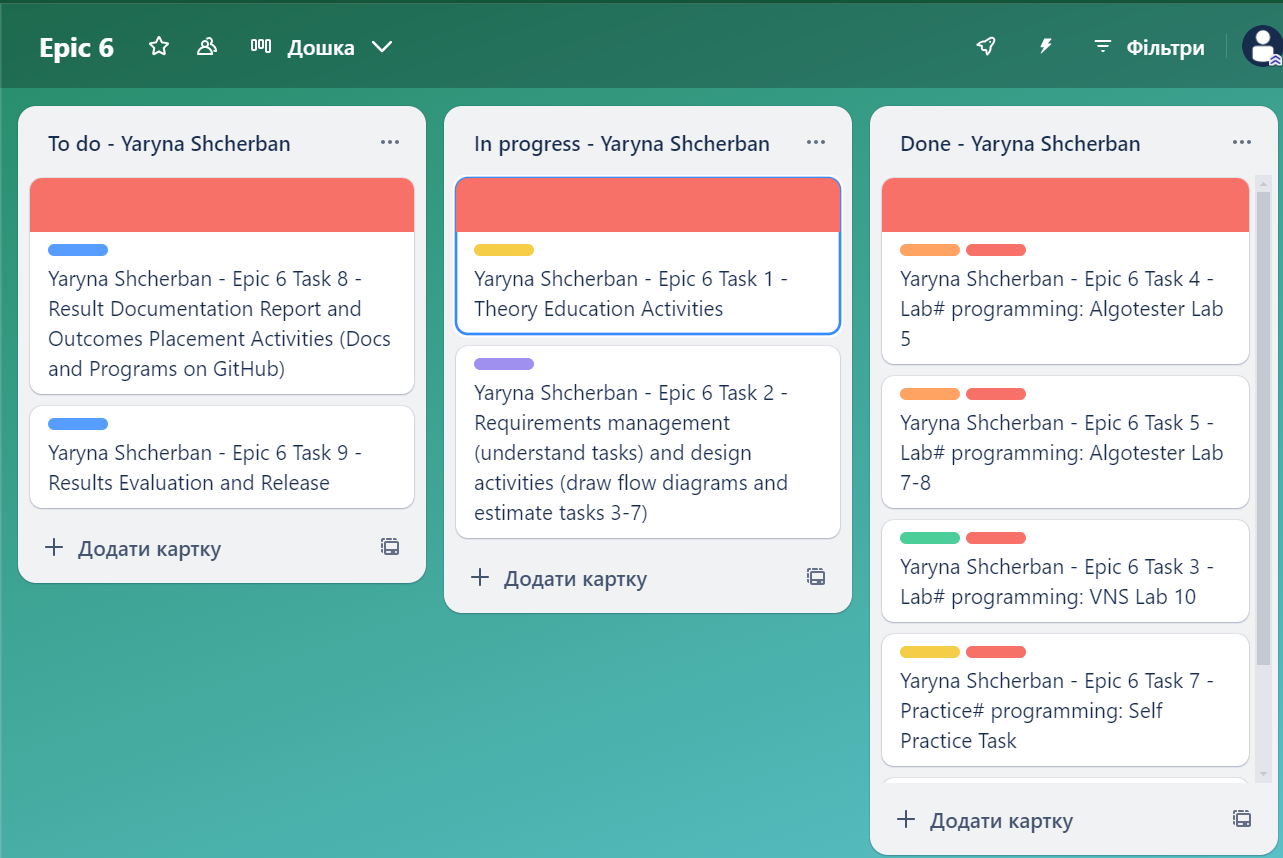
**Завдання №5 Algotester Lab 5 – Variant 2 :**

Запланований час виконання - 30 хв

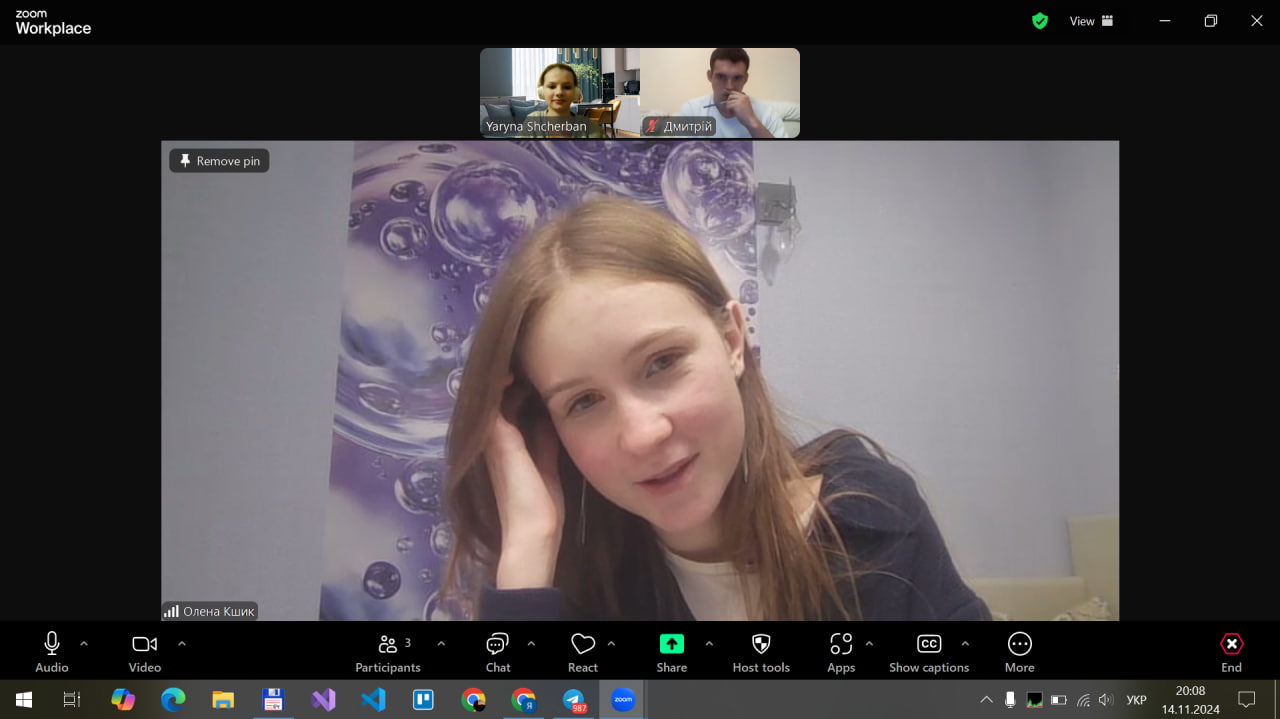




1. *Конфігурація середовища до виконання завдань:*



Створена спільна дошка в Trello

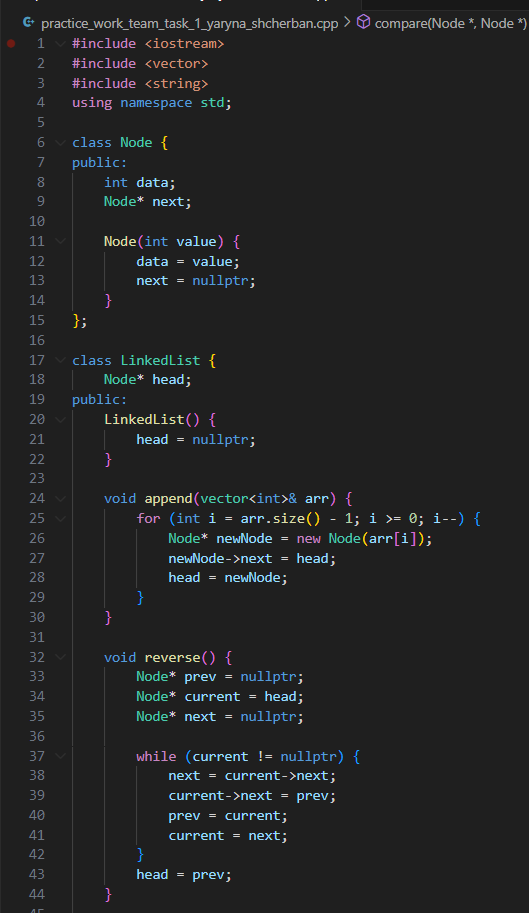


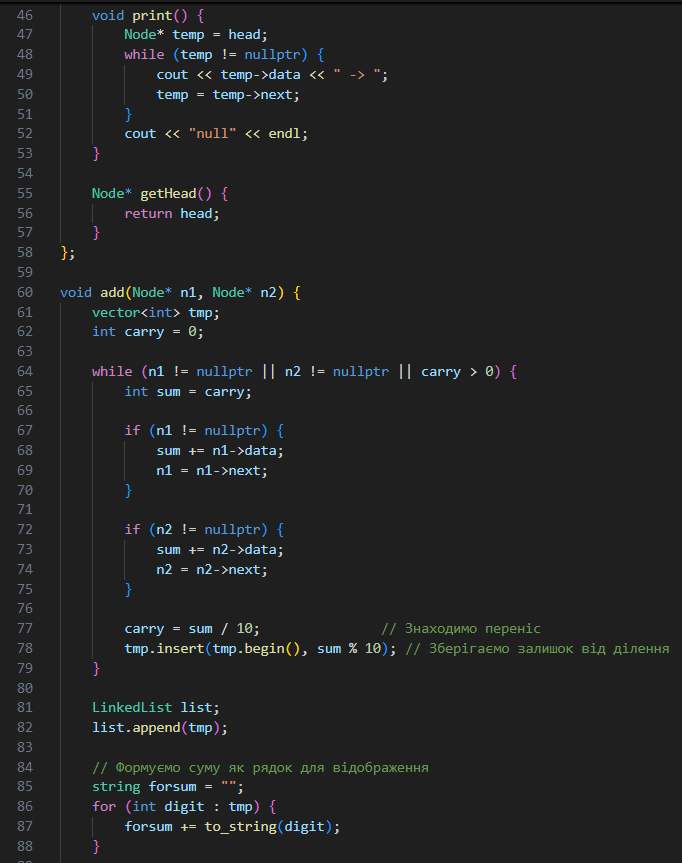
Була проведена зустріч з командою щодо практичного завдання

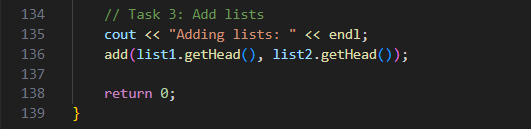
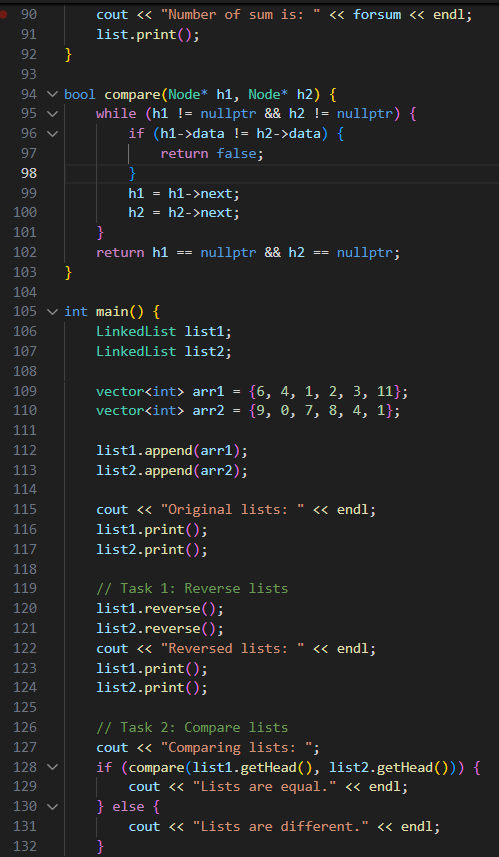
1. *Код програми з посиланням на зовнішні ресурси*

**Завдання №1 Епік 6 : Практичне завдання :**

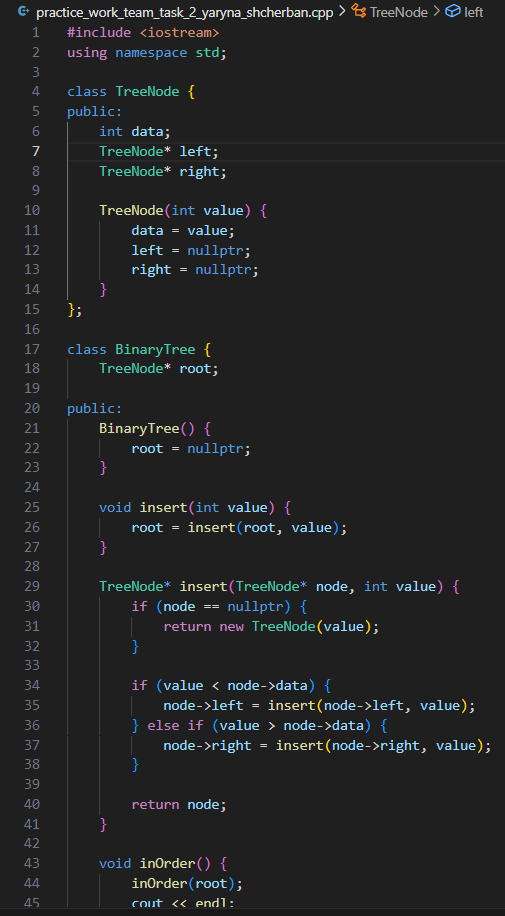
practice\_work\_team\_task\_1\_yaryna\_shcherban.cpp

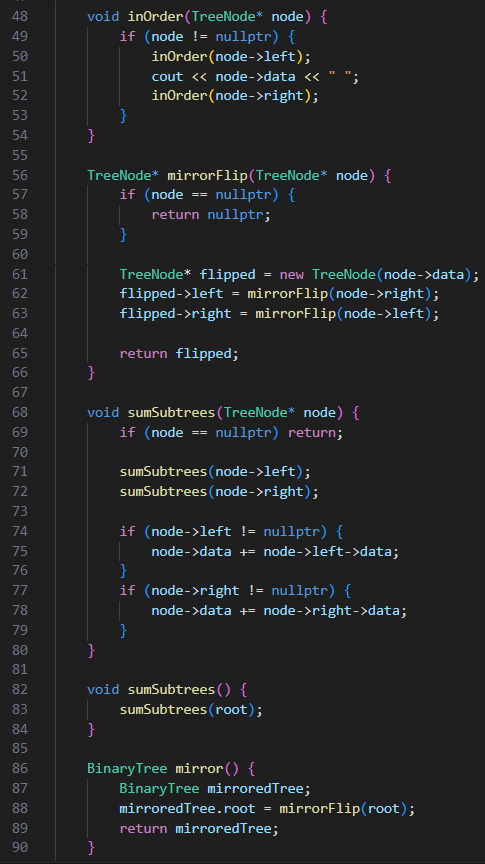


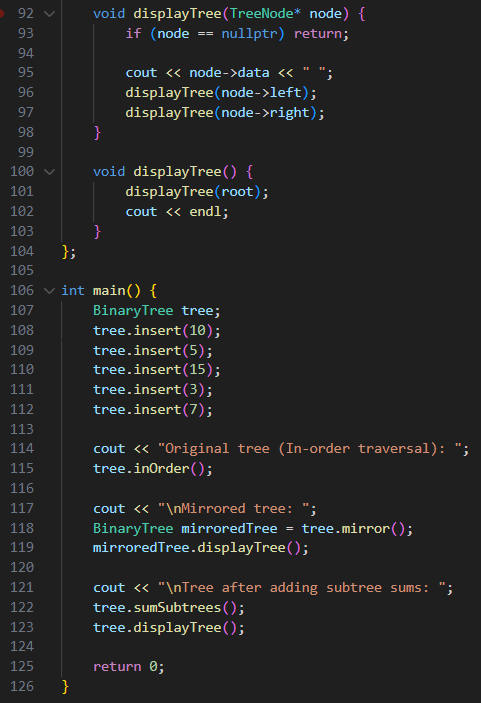




practice\_work\_team\_task\_2\_yaryna\_shcherban.cpp

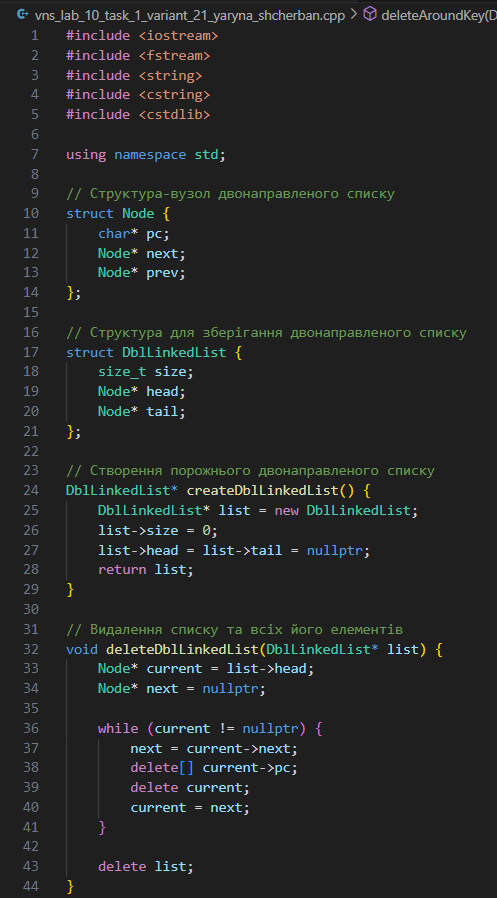


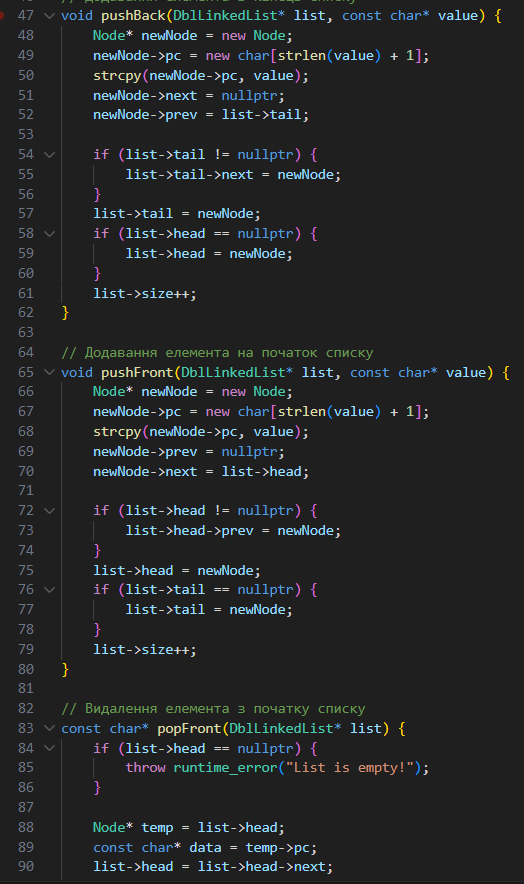


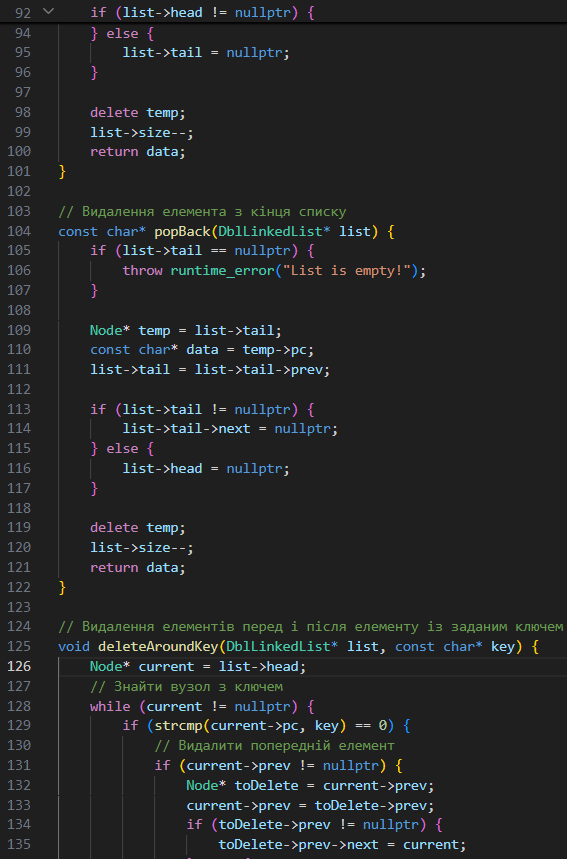


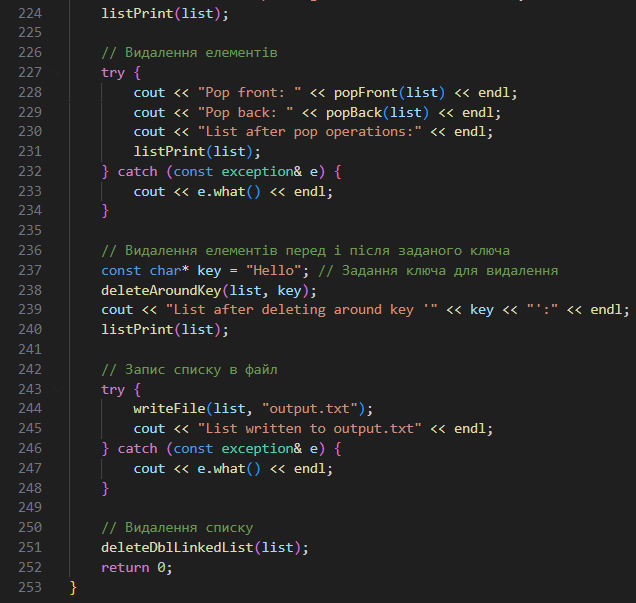
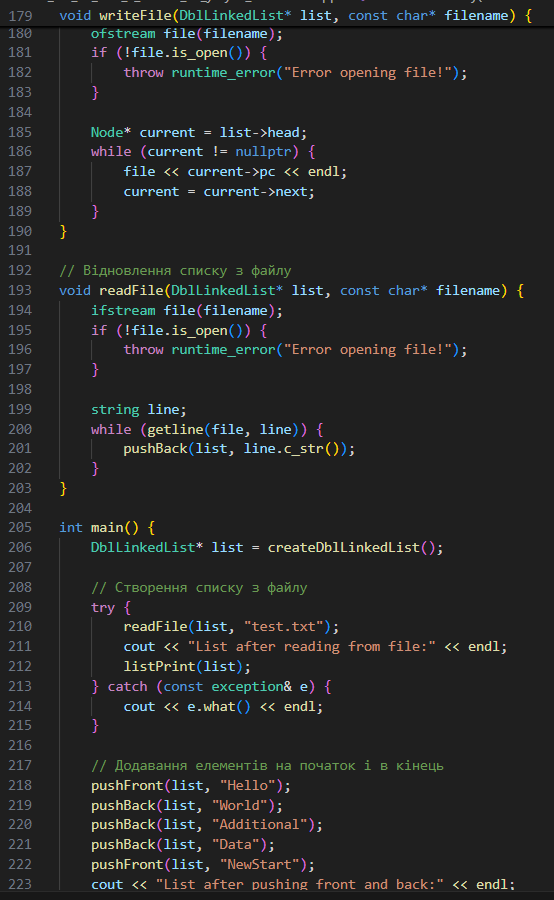
**Завдання №2 VNS Lab 10 - Task 1 – Variant 21 :**

vns\_lab\_10\_task\_1\_variant\_21\_yaryna\_shcherban.cpp



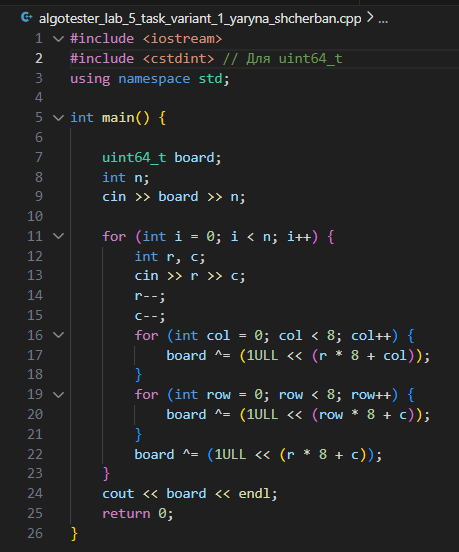






**Завдання №3 Algotester Lab 5 - Variant 1 :**

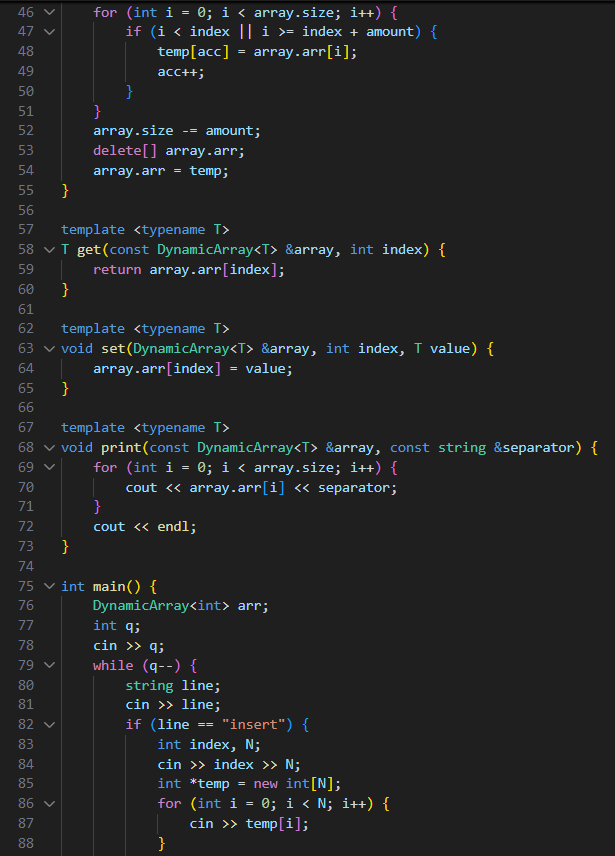
algotester\_lab\_5\_task\_variant\_1\_yaryna\_shcherban.cpp

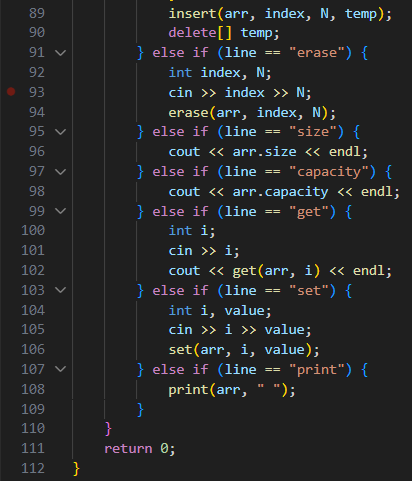


**Завдання №4 Algotester Lab 7-8 - Variant 2 :**

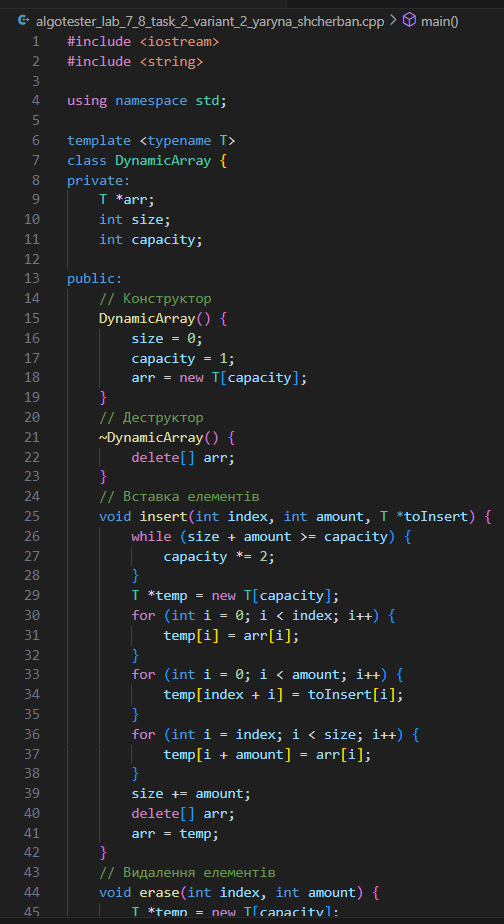
algotester\_lab\_7\_8\_task\_1\_variant\_2\_yaryna\_shcherban.cpp

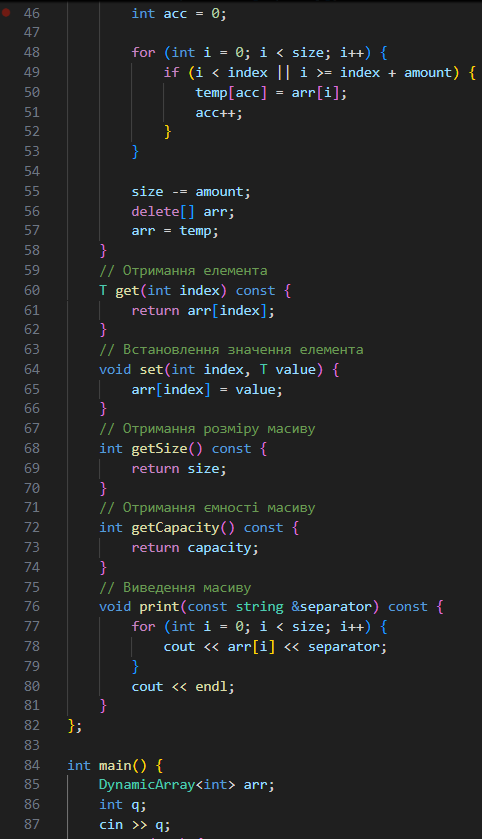


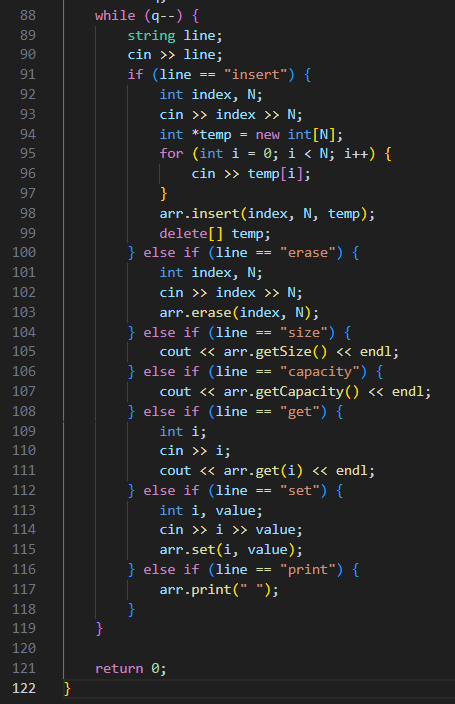




algotester\_lab\_7\_8\_task\_2\_variant\_2\_yaryna\_shcherban.cpp







**Завдання №5 Algotester Lab 5 – Variant 2 :**

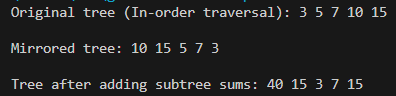
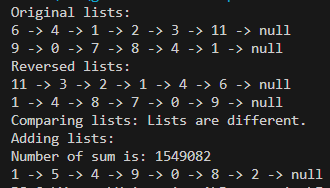
practice\_work\_self\_algotester\_tasks\_yaryna\_shcherban.cpp



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

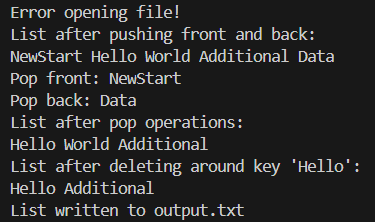
**Завдання №1 Епік 6 : Практичне завдання :**

Фактично затрачений час – 2.5 год



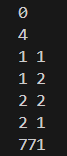
**Завдання №2 VNS Lab 10 - Task 1 – Variant 21 :**

Фактично затрачений час – 2.5 год



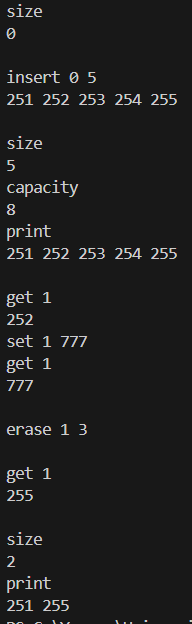
**Завдання №3 Algotester Lab 5 - Variant 1 :**

Фактично затрачений час – 40 хв



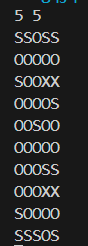
**Завдання №4 Algotester Lab 7-8 - Variant 2 :**

Фактично затрачений час – 3 год



**Завдання №5 Algotester Lab 5 – Variant 2 :**

Фактично затрачений час – 40 хв



Висновок : У ході виконання роботи було розглянуто основні динамічні структури даних: черги, стеки, зв’язні списки та бінарні дерева. Було реалізовано алгоритми для додавання, видалення та обробки елементів у зв’язному списку, а також операції з бінарним деревом, такі як вставка елементів, дзеркальне відображення та обчислення сум піддерев.

Посилання на Pull Request : <https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/599>