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-14

Кормилюк Роман Олегович

# **Тема роботи:** робота з динамічними структурами (Черга, Стек, Списки, Дерево). Алгоритмами обробки та динамічними структурами.

# **Мета роботи:** отримати навички та навчитися писати код використовуючи динамічні структури (Черги, Стек, Списки, Дерево). Алгоритми обробки динамічних структур.

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема **№1:** Динамічні структури (Черга, Стек, Списки, Дерево).
* Тема **№2:** Алгоритми обробки динамічних структур.
* Тема **№3:** Бінарне дерево**.**

1. Індивідуальний план опрацювання теорії:

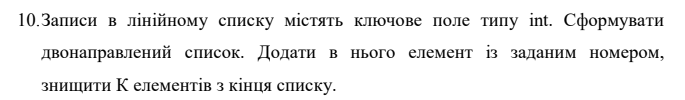
* Тема **№1: Динамічні структури (Черга, Стек, Списки, Дерево).**
  + Джерела Інформації
    - Книжка. “Мова програмування С” Браян В. Керніган, Деніс М. Річі
    - Відео: [Стек, Куча, Статична пам'ять](https://www.youtube.com/watch?v=NyOjKd5Qruk&ab_channel=%D0%91%D0%BB%D0%BE%D0%B3%D0%B0%D0%BD%E2%A6%81%D0%A3%D1%80%D0%BE%D0%BA%D0%B8%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)
    - Стаття: [Стек і Купа](https://acode.com.ua/urok-111-stek-i-kupa/)  
       [Вкладені класи. Класи. Статичне поле.](https://www.youtube.com/watch?v=ATNE6AKZ-WU&ab_channel=%D0%91%D0%BB%D0%BE%D0%B3%D0%B0%D0%BD%E2%A6%81%D0%A3%D1%80%D0%BE%D0%BA%D0%B8%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)
  + Що опрацьовано:
    - Динамічні структури (Черга, Стек, Списки, Дерево, Класи).
  + Статус: Ознайомлений частково
  + Початок опрацювання теми: 15.12
  + Звершення опрацювання теми: 19.12
* Тема **№2: Алгоритми обробки динамічних структур.**
  + Джерела Інформації
    - Стаття: [Використання структур](http://cpp.dp.ua/vykorystannya-struktur/)
    - Книжка. “Мова програмування С” Браян В. Керніган, Деніс М. Річі
    - Відео: [Програмування складних динамічних структур даних](https://www.youtube.com/watch?v=eSxLVD5vfqM&ab_channel=%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D0%B8%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)
  + Що опрацьовано:
    - Алгоритми обробки динамічних структур.
  + Статус: Ознайомлений частково
  + Початок опрацювання теми: 15.12
  + Звершення опрацювання теми: 19.12
* Тема **№3: Бінарне дерево.**
  + Джерела Інформації
    - Книжка. “Мова програмування С” Браян В. Керніган, Деніс М. Річі
    - Стаття: [Бінарні дерева](https://purecodecpp.com/uk/archives/2483)
    - Відео: [Бінарне дерево](https://www.youtube.com/watch?v=qBFzNW0ALxQ&ab_channel=%D0%91%D0%BB%D0%BE%D0%B3%D0%B0%D0%BD%E2%A6%81%D0%A3%D1%80%D0%BE%D0%BA%D0%B8%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)
  + Що опрацьовано:
    - Бінарне дерево.
  + Статус: Ознайомлений частково
  + Початок опрацювання теми: 15.12
  + Звершення опрацювання теми: 19.12

# **Виконання роботи:**

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

Завдання **№1 VNS Lab 10**

* Варіант завдання: 10



*Завдання до програми №1*

* Деталі завдання: створення списку, додавання елемента в список (у відповідності зі своїм варіантом). Знищення елемента зі списку (у відповідності зі своїм варіантом), друк списку, запис списку у файл, знищення списку, відновлення списку з файлу.
* Важливі деталі для врахування в імплементації програми: використовується динамічне виділення пам'яті з **new** та **delete**, слід пам'ятати про звільнення пам'яті, коли вона більше не потрібна. Функція **create\_list** створює список з послідовними ключами 0, 1, 2, ... n-1. Функція **delete\_element\_from\_end** видаляє елементи з кінця списку, починаючи з 1. Варто задокументувати це чітко. Функції додавання та видалення елементів: Функція **add\_element\_at\_position** не перевіряє додавання елемента на кінець списку, де **current->next** може бути **nullptr**. Подумайте про обробку цього випадку. Функція **delete\_element\_from\_end** видаляє елементи, використовуючи лічильник **k**. Якщо **k** більший за кількість елементів, видалення відбувається з-за меж списку. Слід додати перевірку, щоб уникнути помилок. Функція збереження списку в файл: **save\_list\_to\_file** відкриває файл в режимі ofstream та записує ключі, розділені пробілами. Важливо закрити файл командою **file.close()** наприкінці. Розгляньте варіанти збереження додаткової інформації про список (кількість елементів, тип даних тощо) для зручнішого завантаження.

Завдання **№2 Algotester Lab 5v2**

* Варіант завдання: 2
* Деталі завдання: потрібно сказати як буде виглядати печера після землетрусу.
* Важливі деталі для врахування в імплементації програми: у першому рядку 2 цілих числа N та M - висота та ширина печери. У N наступних рядках стрічка rowi яка складається з N цифер - i-й рядок матриці, яка відображає стан печери до землетрусу.

Завдання **№3 Algotester Lab 7-8**

* Варіант завдання: 1
* Деталі завдання: ця програма реалізує простий клас **List**, який представляє собою імітацію двонаправленого списку з використанням вектора. Програма працює в інтерактивному режимі та чекає введення користувача для виконання різних операцій над списком. Основні операції включають додавання елементу за певним індексом, видалення елементів за певним індексом та кількістю, отримання розміру списку, отримання елемента за індексом, встановлення значення елемента за індексом та виведення елементів списку.
* Важливі деталі для врахування в імплементації програми: **Insert**: Додає нові елементи вектора з вказаного індексу. **Erase**: Видаляє елементи з вектора за вказаним індексом та кількістю. **Size:** Повертає розмір вектора (кількість елементів). **Get:** Повертає значення елемента вектора за вказаним індексом. **Set:** Задає нове значення елементу вектора за вказаним індексом. **operator<<:** Перевантажений оператор виведення для виведення вмісту вектора. У функції main виконується зчитування кількості операцій **k**, а потім в циклі виконуються операції відповідно до введення користувача.

Завдання **№4 Class Practice Task**

* Варіант завдання: -
* Деталі завдання: використовувати цілочисельні значення в списку; реалізувати метод реверсу; реалізувати допоміжний метод виведення вхідного і обернутого списків;
* Важливі деталі для врахування в імплементації програми: **Node\* reverse(Node \*head)**; функція **void, return prev -** новий початок списку (кінець оригінального списку), **while.**

Завдання **№5 Class Practice Task 2**

* Варіант завдання: -
* Деталі завдання: 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***.
* Важливі деталі для врахування в імплементації програми: **bool compare(Node \*h1, Node \*h2);**

Завдання **№6 Class Practice Task 3**

* Варіант завдання: -
* Деталі завдання: 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: **Node\* add(Node \*n1, Node \*n2);**

Завдання **№7 Class Practice Task 4**

* Варіант завдання: -
* Деталі завдання: 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
* Важливі деталі для врахування в імплементації програми: **TreeNode \*create\_mirror\_flip(TreeNode \*root);**

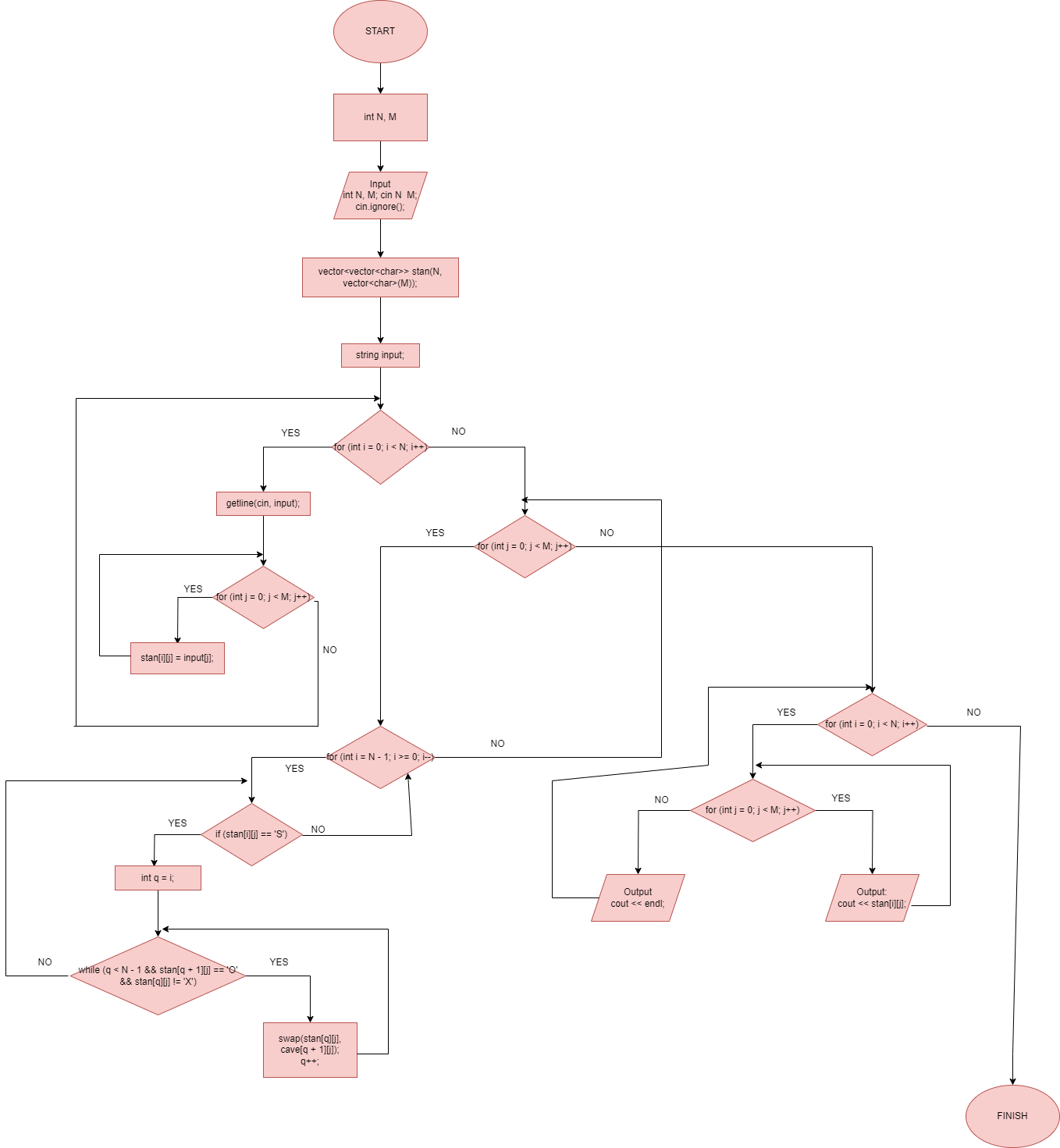
Завдання **№8 Class Practice Task 5**

* Варіант завдання: -
* Деталі завдання: 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: **void tree\_sum(TreeNode \*root);**

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

Програма **№2 Algotester Lab 5v2**

* Блок-схема:



*Блок-схема до програми №2*

* Планований час на реалізацію: 1 год.
* Важливі деталі для врахування в імплементації: використав блоки Process, Elipse, Rectangle, Parallelogram для блок-схеми. Описав у них задачу та план дій.

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

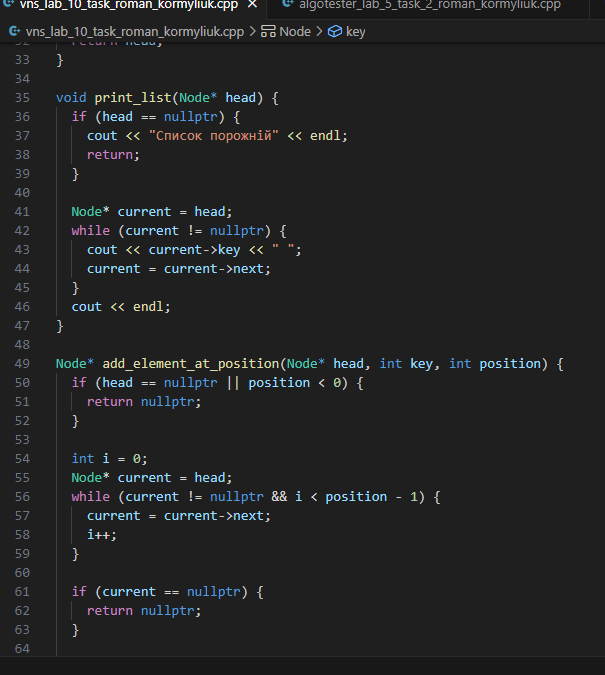
Конфігурацію не змінював.

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

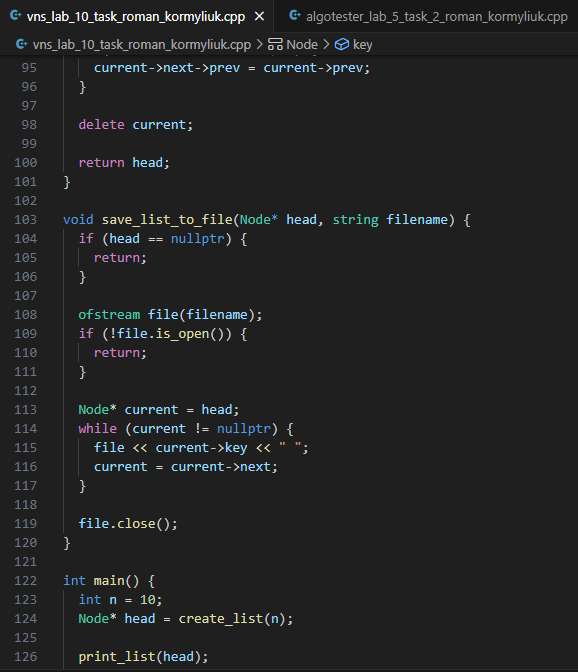
Завдання **№1 VNS Lab 10**

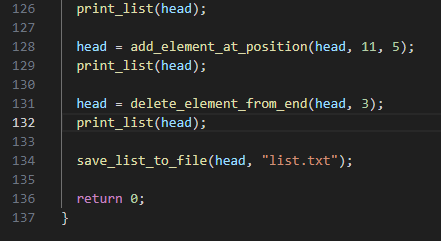
Посилання на файл програми у пул-запиті GitHub: [посилання.](http://ai_14/roman_kormyliuk/Epic%206/vns_lab_10_task_roman_kormyliuk.cpp)







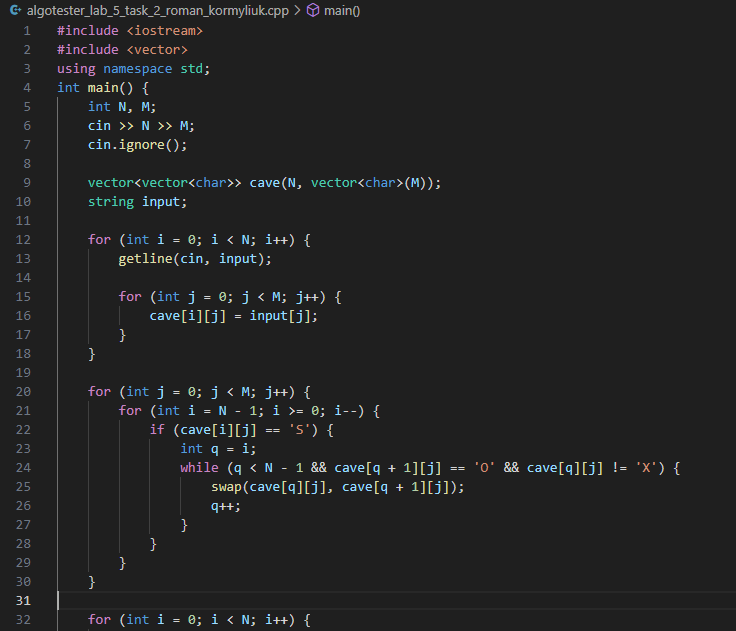


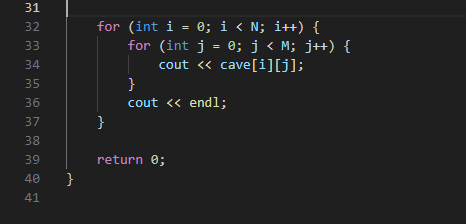


*Код програми №1*

Завдання **№2 Algotester Lab 5v2**

Посилання на файл програми у пул-запиті GitHub: [посилання.](http://ai_14/roman_kormyliuk/Epic%206/algotester_lab_5_task_2_roman_kormyliuk.cpp)

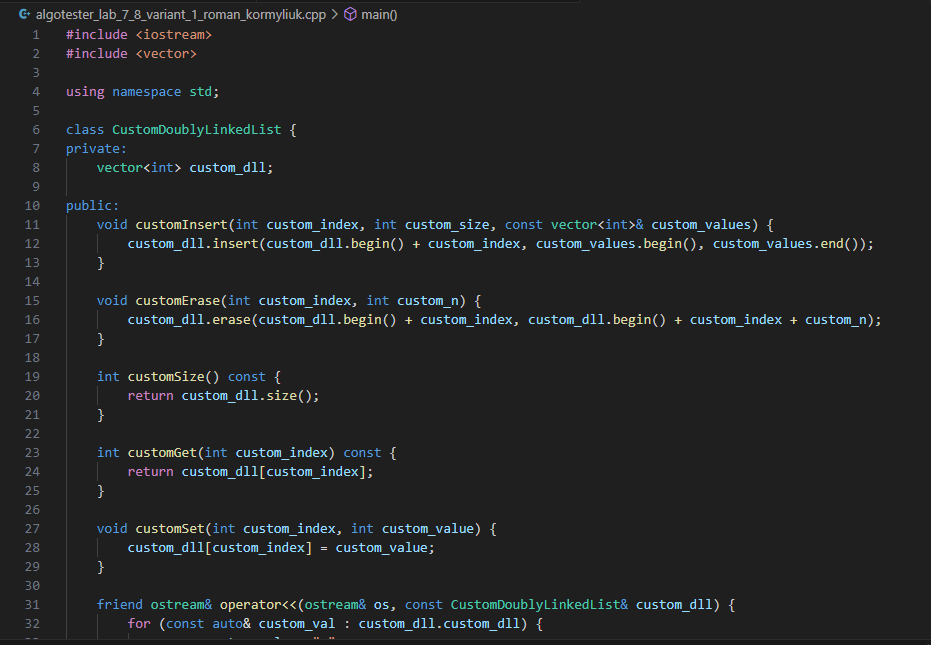


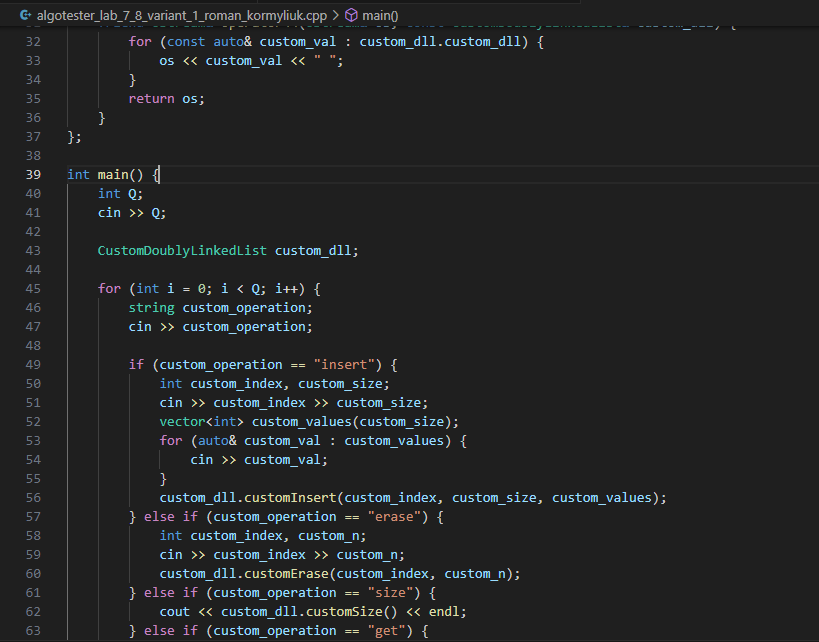


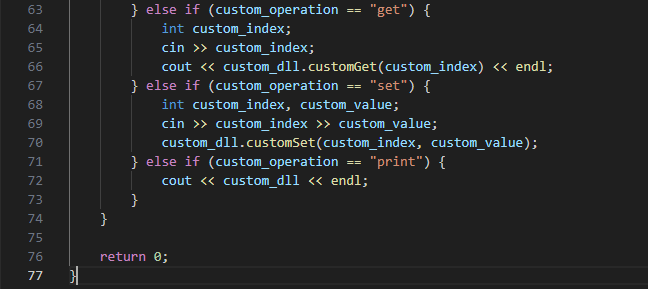
*Код програми №2*

Завдання **№3 Algotester Lab 7-8**

Посилання на файл програми у пул-запиті GitHub: [посилання.](http://ai_14/roman_kormyliuk/Epic%206/algotester_lab_7_8_variant_1_roman_kormyliuk.cpp)





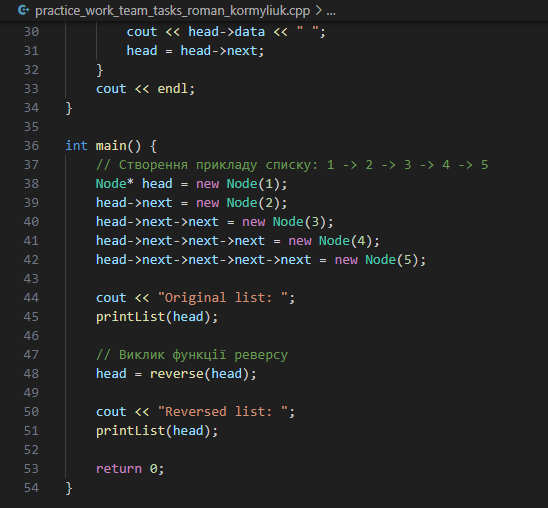


*Код до програми №3*

Завдання **№4 Class Practice Task**

Посилання на файл програми у пул-запиті GitHub: [посилання.](http://ai_14/roman_kormyliuk/Epic%206/practice_work_team_tasks_roman_kormyliuk.cpp)

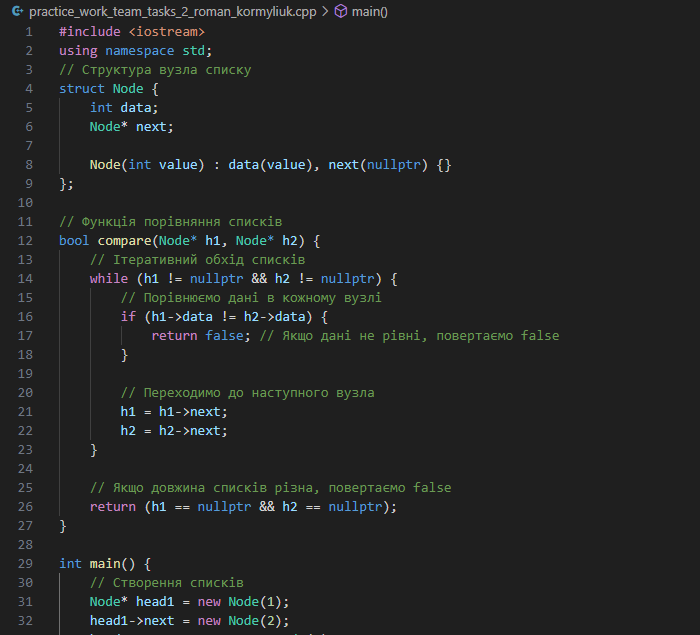
**

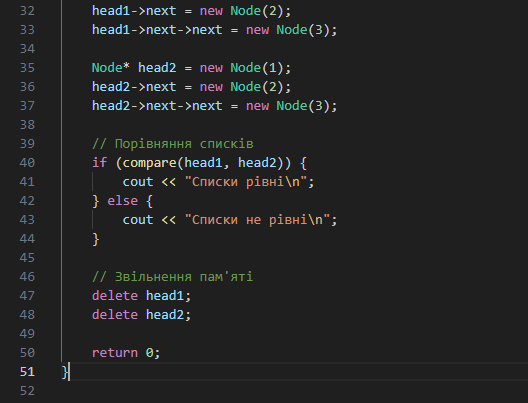
**

*Код до програми №4*

Завдання **№5 Class Practice Task 2**

Посилання на файл програми у пул-запиті GitHub: [посилання.](http://ai_14/roman_kormyliuk/Epic%206/practice_work_team_tasks_2_roman_kormyliuk.cpp)

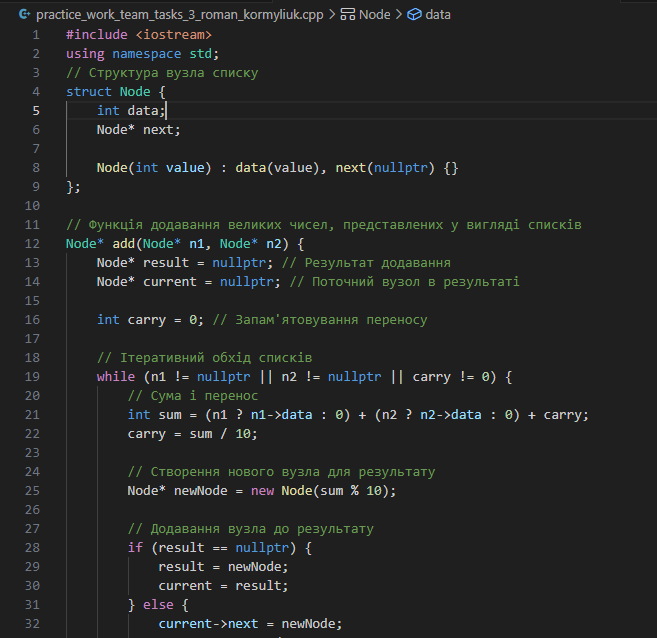
**

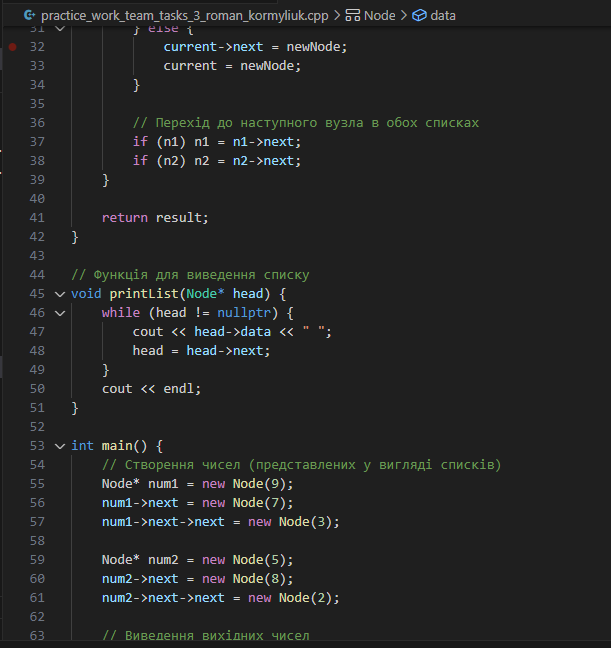
**

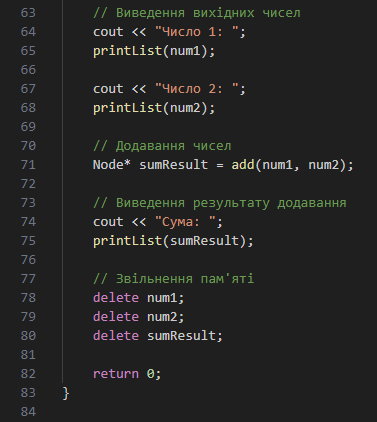
*Код до програми №5*

Завдання **№6 Class Practice Task 3**

Посилання на файл програми у пул-запиті GitHub: [посилання.](http://ai_14/roman_kormyliuk/Epic%206/practice_work_team_tasks_3_roman_kormyliuk.cpp)



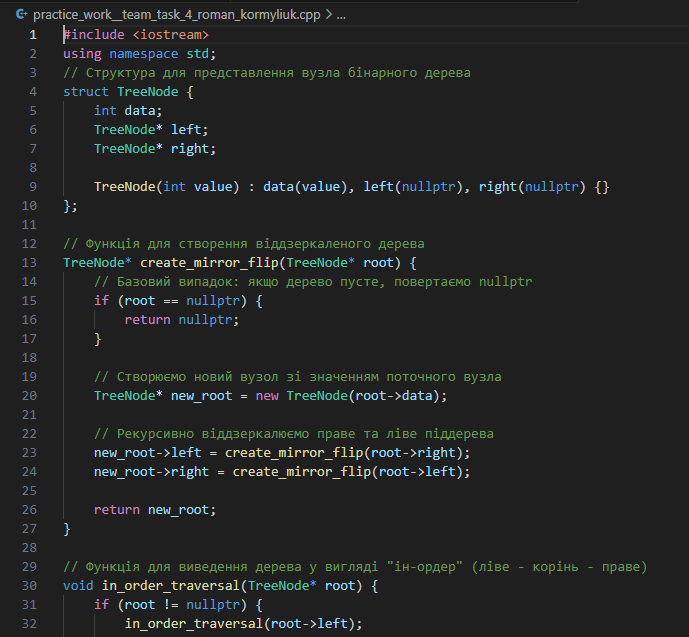


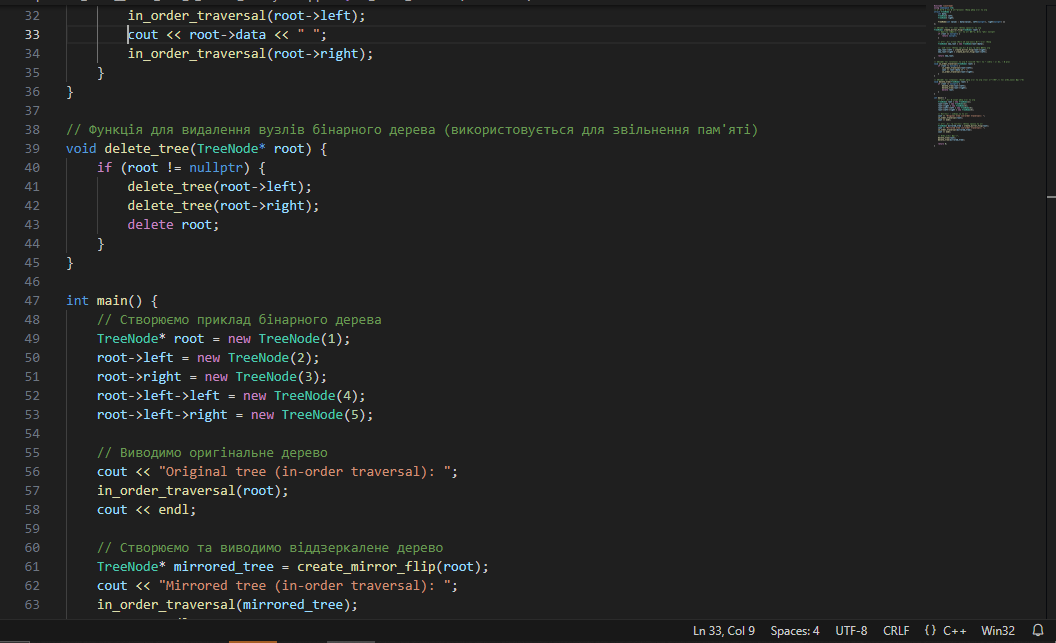


*Код до програми №6*

Завдання **№7 Class Practice Task 4**

Посилання на файл програми у пул-запиті GitHub: [посилання.](http://ai_14/roman_kormyliuk/Epic%206/practice_work__team_task_4_roman_kormyliuk.cpp)

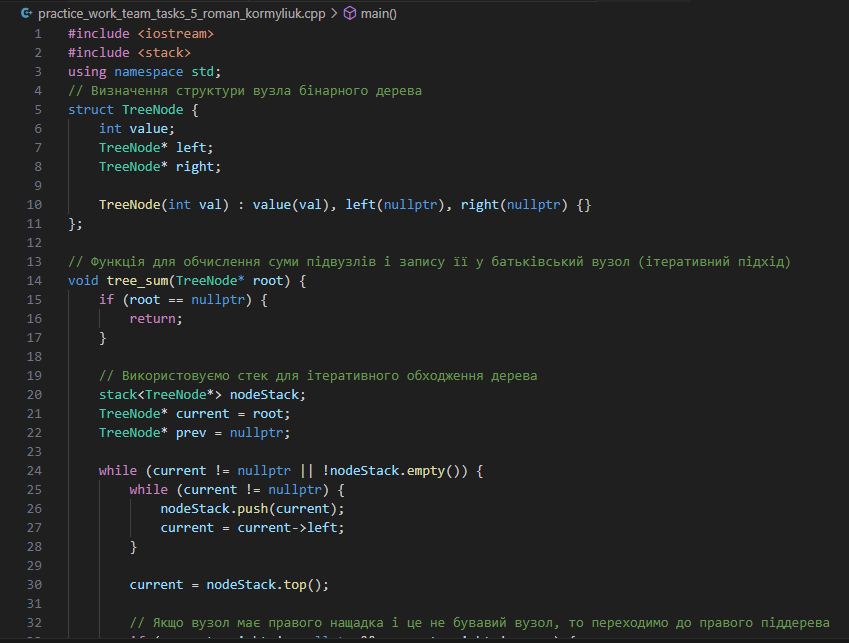


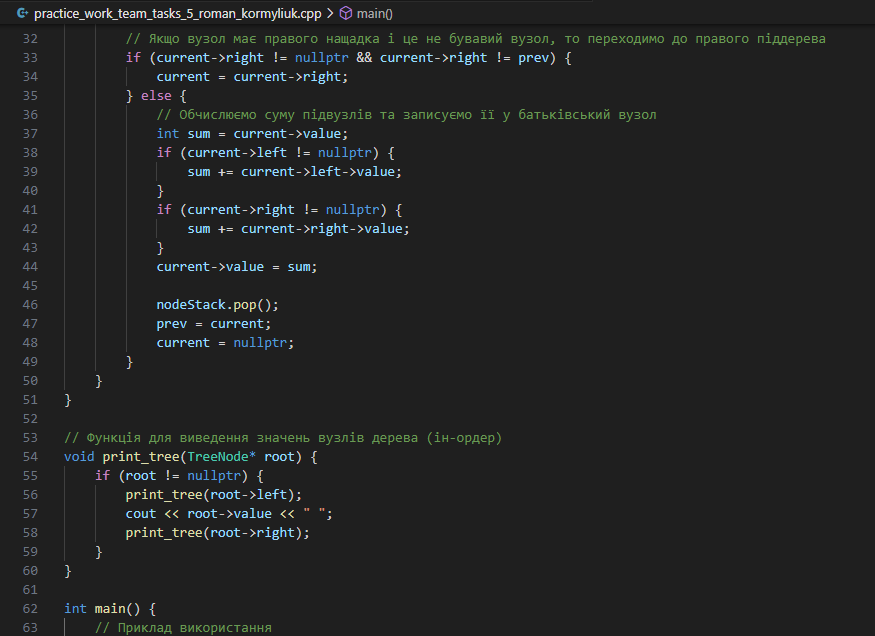


*Код до програми №7*

Завдання **№8 Class Practice Task 5**

Посилання на файл програми у пул-запиті GitHub: [посилання.](http://ai_14/roman_kormyliuk/Epic%206/practice_work_team_tasks_5_roman_kormyliuk.cpp)



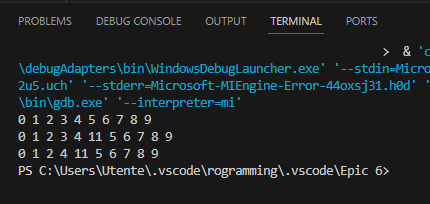


*Код до програми №8*

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

Завдання **№1 VNS Lab 10**

Деталі по виконанню і тестуванню програми:

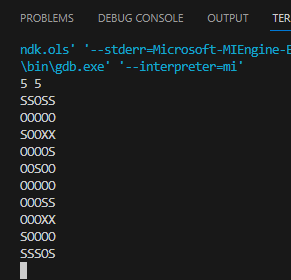


*Результат програми №1*

Час затрачений на виконання завдання: 5 хв.

Завдання **№2 Algotester Lab 5**

Деталі по виконанню і тестуванню програми:

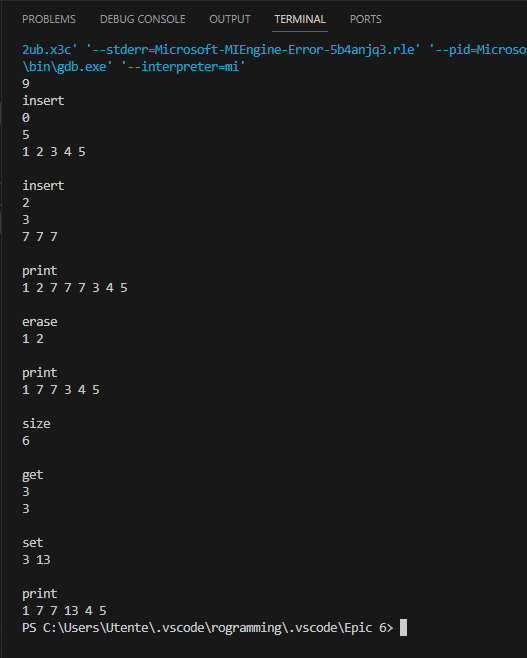


*Результат програми №2*

Час затрачений на виконання завдання: 5 хв.

Завдання **№3 Algotester Lab 7-8**

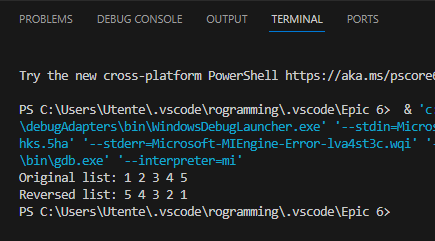
Деталі по виконанню і тестуванню програми:

  
*Результат програми №3*

Час затрачений на виконання завдання: 5 хв.

Завдання **№4 Class Practice Task**

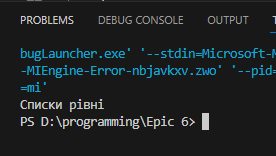
Деталі по виконанню і тестуванню програми:



*Результат програми №4*

Завдання **№5 Class Practice Task 2**

Деталі по виконанню і тестуванню програми:

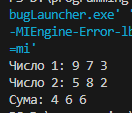


*Результат програми №5*

Час затрачений на виконання завдання: 5 хв.

Завдання **№6 Class Practice Task 3**

Деталі по виконанню і тестуванню програми:

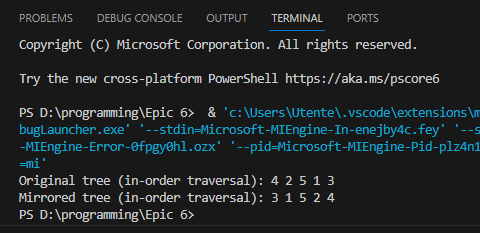


*Результат програми №6*

Час затрачений на виконання завдання: 5 хв.

Завдання **№7 Class Practice Task 4**

Деталі по виконанню і тестуванню програми:

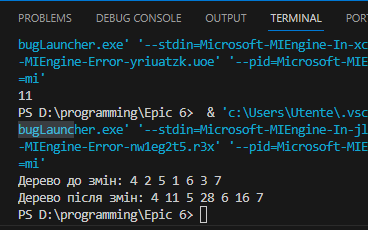


*Результат програми №7*

Час затрачений на виконання завдання: 5 хв.

Завдання **№8 Class Practice Task 5**

Деталі по виконанню і тестуванню програми:

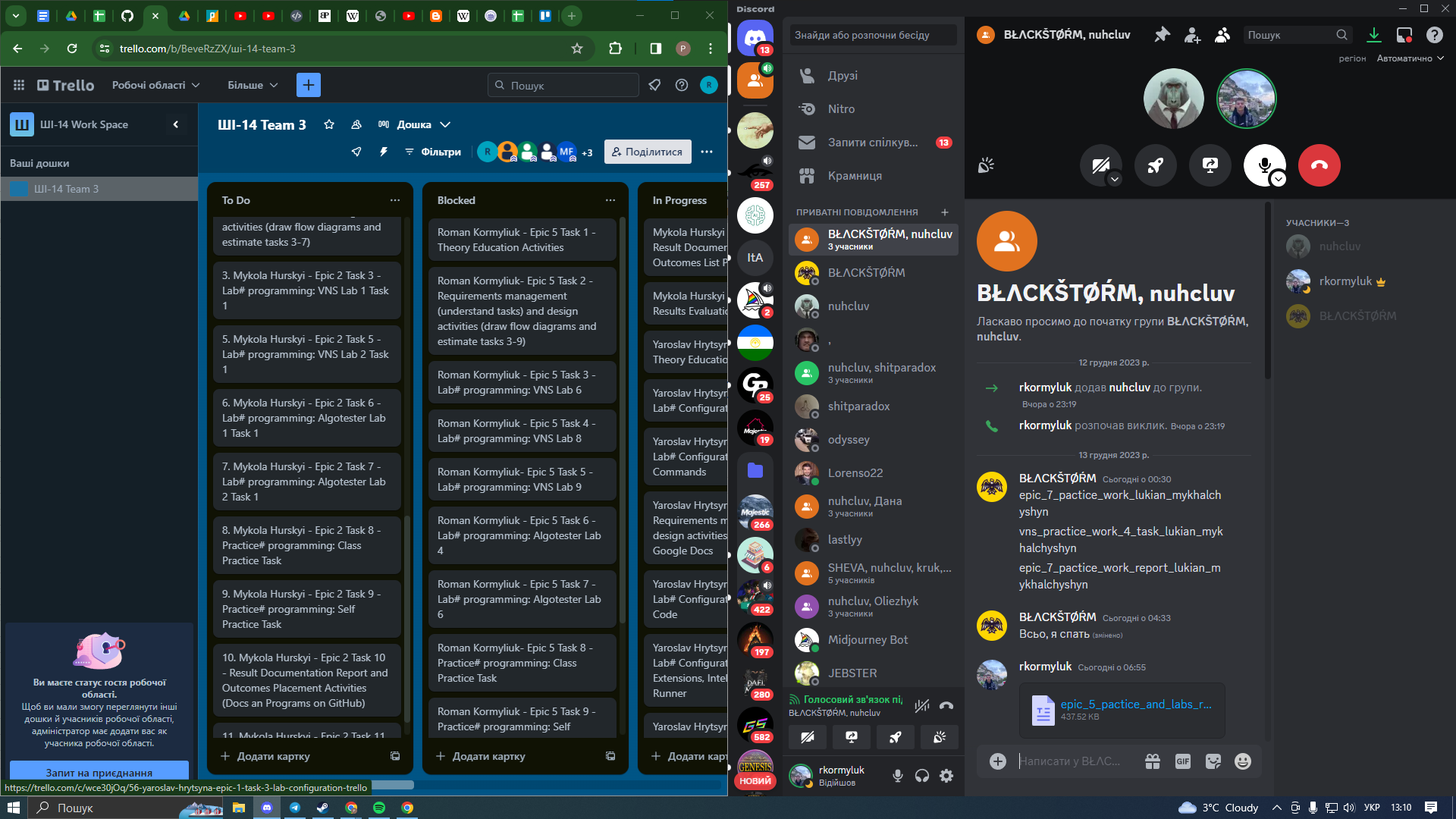


*Результат програми №8*

Час затрачений на виконання завдання: 5 хв.

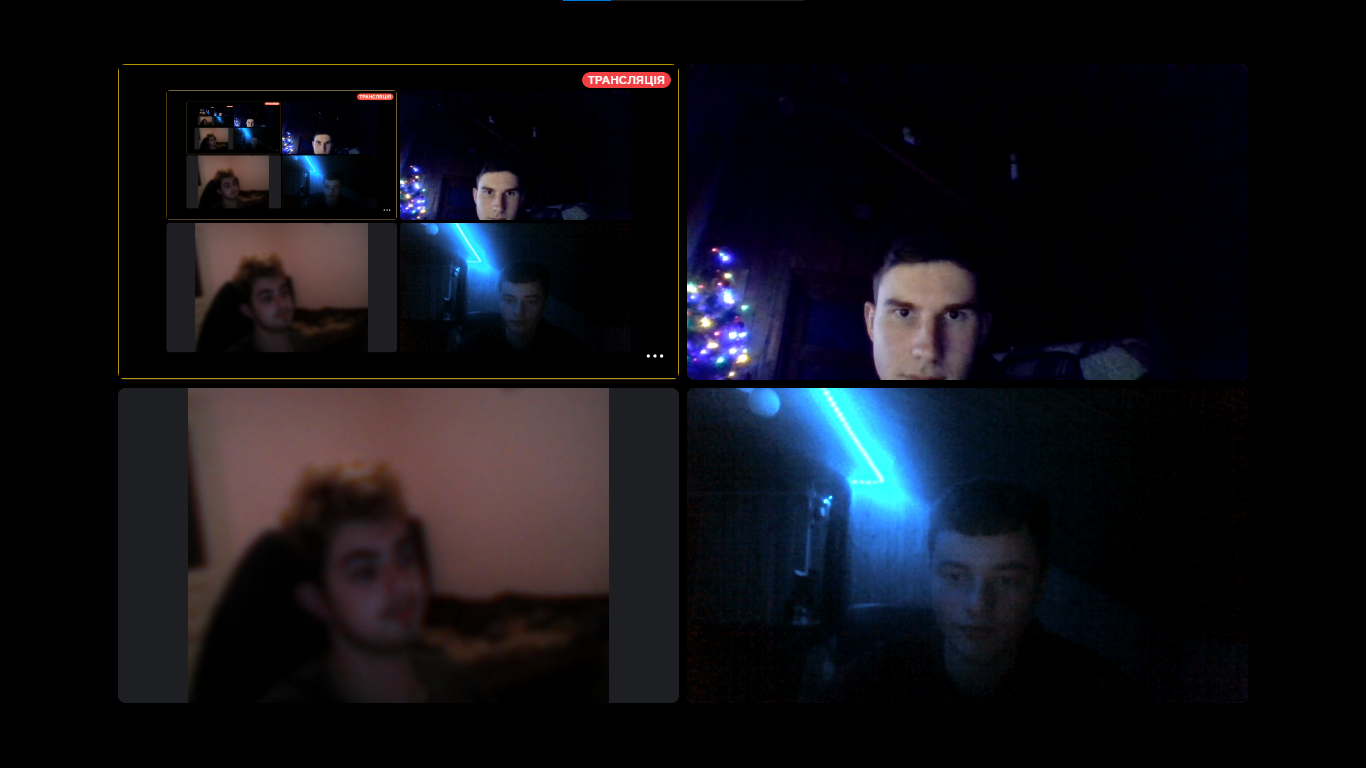
## **6. Кооперація з командою:**

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



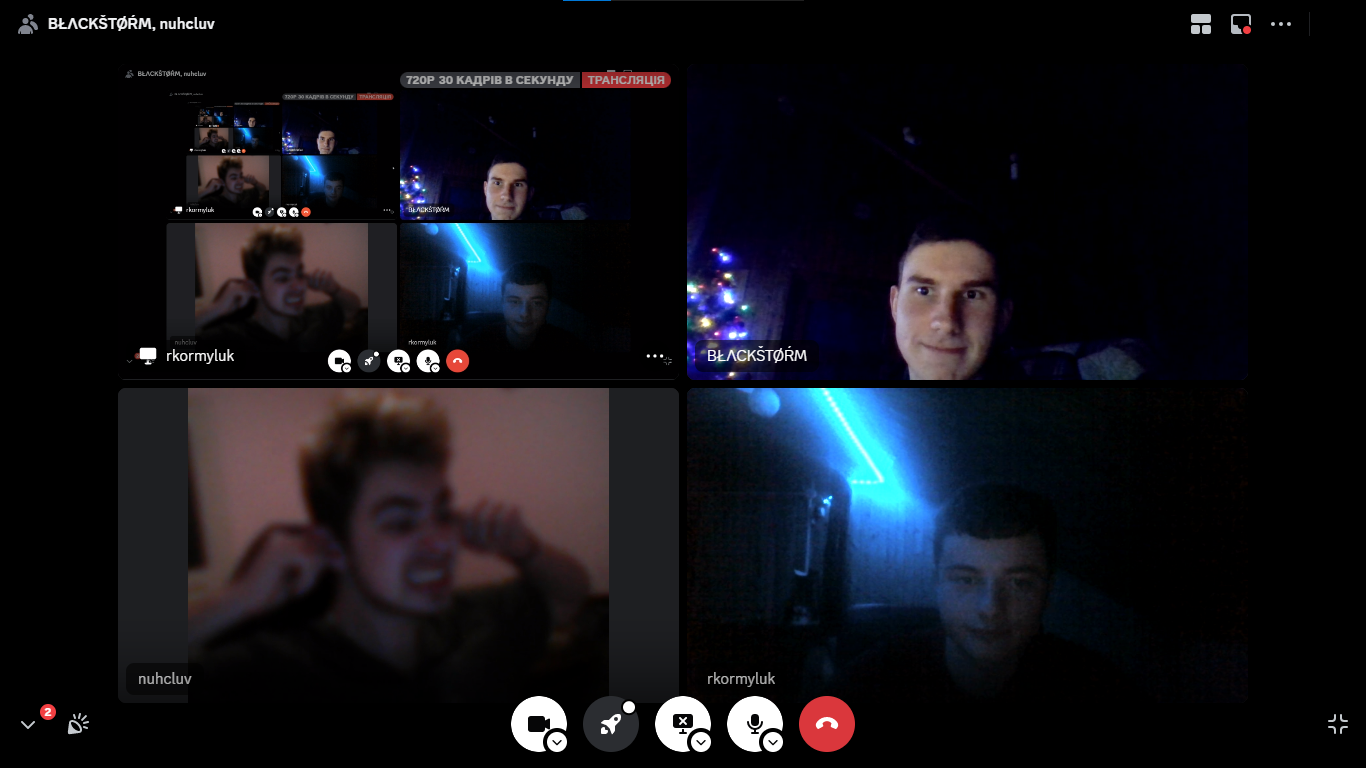
*Виконання завдань та робота в Trello*

* Скрін з 2-ї зустрічі по обговоренню задач Епіку та



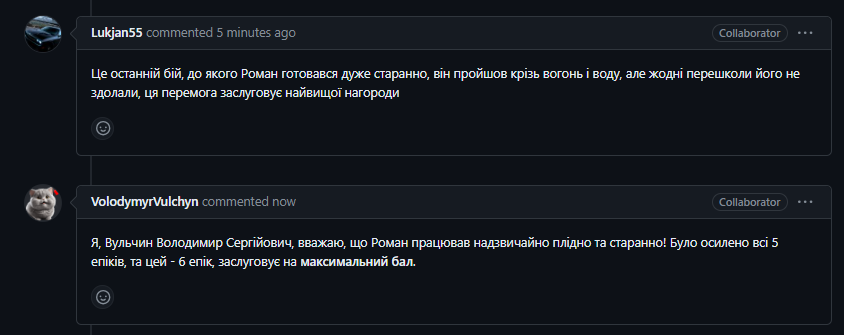
*Обговорення завдань*

* Скрін з 3-ї зустрічі по обговоренню задач Епіку



*Обговорення про закінчення звіту*

* Скрін з коментарями від учасників команди на пул реквесті з Ревю Роботи



*Коментарі тімейтів в ПР*

# **Висновки:**

# Отже, працюючи над Epic№6, я отримав базові знання про те, що таке бінарні дерева, як з ними працювати, як вони працюють. Що таке динамічні структури (Черга, Стек, Списки, Дерево), алгоритми обробки динамічних структур. Знання які я отримав протягом лабораторної, я старався використати в програмах, які я виконував, задачі, стосувалися різноманітних аспектів програмування. Одна з програм була спрямована на роботу із списком, де були реалізовані операції вставки, видалення, отримання розміру та виведення списку. Інша програма вирішувала завдання з реверсування односпрямованого списку. Обидві програми використовують структури даних, такі як вектор та вузол списку, Також я використав заннання з минулих лабораторних робіт, так як: цикли, умовні оператори, функції та робота з пам'яттю. Програми відображають роботу з класами, структурами та функціями для реалізації конкретних завдань. Для мене ця лабораторна робота була найдовшою в опрацюванні, завдяки своїм тімейтам я розібрався та виконав завдання.

# **Посилання на ПР:** [**PR**](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/1150)