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-11

Федоришин Микола Володимирович

Львів 2024

**Тема роботи:**

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

**Мета роботи:**

Застосувати на практиці вивчений матеріал, реалізувати Linked List (Однозв’язний список), бінарне дерево.

**Теоретичні відомості:**

- Тема №1: Основи Динамічних Структур Даних.

- Тема №2: Стек.

- Тема №3: Черга.

- Тема №4: Зв’язні списки.

- Тема №5: Дерева.

- Тема №6: Алгоритми Обробки Динамічних Структур.

1. Індивідуальний план опрацювання теорії:

- Тема №1: Основи Динамічних Структур Даних:

○ Джерела інформації:

▪ Статті.

<https://www.youtube.com/watch?v=NyOjKd5Qruk&list=PLiPRE8VmJzOpn6PzYf0higmCEyGzo2A5g&index=58>

- Що опрацьовано:

○ Вступ до динамічних структур.

○ Виділення пам’яті для структур даних (stack і heap)

○ Приклади простих динамічних структур

Запланований час на вивчення 40 хвилин

Витрачений час 40 хвилин.

  - Тема №2: Стек:

○ Джерела інформації:

▪ Статті.

<https://acode.com.ua/urok-111-stek-i-kupa/>

<https://www.youtube.com/watch?v=ZYvYISxaNL0&list=PLiPRE8VmJzOpn6PzYf0higmCEyGzo2A5g&index=141>

- Що опрацьовано:

○ Визначення та властивості стеку

○ Операції push, pop, top: реалізація та використання

○ Переповнення стеку

Запланований час на вивчення 2 години.

Витрачений час 2 години.

- Тема №3: Черга:

○ Джерела інформації:

▪ Статті.

<https://www.youtube.com/watch?v=Yhw8NbjrSFA&list=PLiPRE8VmJzOpn6PzYf0higmCEyGzo2A5g&index=142>

- Що опрацьовано

○ Визначення та властивості черги

○ Операціїї dequeue, front: реалізація та застосування

○ Приклади використання черги: обробка подій, алгоритми планування

○ Розширення функціоналу черги: пріоритети черги

Запланований час на вивчення 2 години.

Витрачений час 2 години.

- Тема №4: Зв’язні списки:

○ Джерела інформації:

▪ Статті.

https://www.youtube.com/watch?v=-25REjF\_atI&list=PLiPRE8VmJzOpn6PzYf0higmCEyGzo2A5g&index=139

<https://www.youtube.com/watch?v=QLzu2-_QFoE&list=PLiPRE8VmJzOpn6PzYf0higmCEyGzo2A5g&index=140>

- Що опрацьовано

○ Визначення однозв’язного та двозв’язного списку

○ Принципи створення нових вузлів, втсавка між існуючими, видалення, створення кільця (circular linked list)

○ Основні операції: обхід списку, пошук, доступ до елементів та об’єднання списків

○ Приклади використання списків: управління пам’яттю, FIFO та LIFO структури.

Запланований час на вивчення 2 години.

Витрачений час 2 години.

- Тема № 5: Дерева:

○ Джерела інформації:

▪ Статті.

https://www.youtube.com/watch?v=qBFzNW0ALxQ&list=PLiPRE8VmJzOpn6PzYf0higmCEyGzo2A5g&index=144

- Що опрацьовано

○ Вступ до структури даних “дерево”: визначення, типи

○ 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

Запланований час на вивчення 2 години.

Витрачений час 2 години.

- Тема №6: Алгоритми Обробки Динамічних Структур:

○ Джерела інформації:

▪ Статті.

https://www.youtube.com/watch?v=mnwDpO4zqLA&t=433s

- Що опрацьовано

○ Основи алгоритмічних патернів: ітеративні, рекурсивні

○ Алгоритми пошуку, сортування даних, додавання та видалення елементів

Запланований час на вивчення 2 години.

Витрачений час 2 години.

Також користувався Chat GPT який давав відповіді на конкретні питання по коду та теорії.

**Виконання роботи:**

**1. Опрацювання завдання до програм.**Завдання №1  
**VNS LAB 10 – TASK 1 (VARIANT 14)**

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.

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

Завдання №2

**ALGOTESTER LAB 5 (VARIANT 3)**

У вас є карта гори розмiром 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 пiк

гори мав найбiльше число.

Клiтинкi якi мають сумiжну сторону з вершиною мають висоту на один меншу, сумiжнi з

ними i не розфарбованi мають ще на 1 меншу висоту i так далi.

Вхiднi данi

У першому рядку 2 числа N та M - розмiри карти

у другому рядку 2 числа x та y - координати пiку гори

Вихiднi данi

N рядкiв по M елементiв в рядку через пробiл - висоти карти.

Завдання №3  
**ALGOTESTER LAB 7-8 (VARIANT 1)**

Ваше завдання - власноруч реалiзувати структуру даних "Двозв’язний список".

Ви отримаєте Q запитiв, кожен запит буде починатися зi слова-iдентифiкатора, пiсля якого

йдуть його аргументи.

Вам будуть поступати запити такого типу:

• Вставка:

Iдентифiкатор - insert

Ви отримуєте цiле число index елемента, на мiсце якого робити вставку.

Пiсля цього в наступному рядку рядку написане число N - розмiр списку, який треба вставити.

У третьому рядку N цiлих чисел - список, який треба вставити на позицiю index.

• Видалення:

Iдентифiкатор - erase

Ви отримуєте 2 цiлих числа - index, iндекс елемента, з якого почати видалення та n -

кiлькiсть елементiв, яку треба видалити.

• Визначення розмiру:

Iдентифiкатор - size

Ви не отримуєте аргументiв.

Ви виводите кiлькiсть елементiв у списку.

• Отримання значення i-го елементу

Iдентифiкатор - get

Ви отримуєте цiле число - index, iндекс елемента.

Ви виводите значення елемента за iндексом.

• Модифiкацiя значення i-го елементу

Iдентифiкатор - set

Ви отримуєте 2 цiлих числа - iндекс елемента, який треба змiнити, та його нове значення.

• Вивiд списку на екран

Iдентифiкатор - print

Ви не отримуєте аргументiв.

Ви виводите усi елементи списку через пробiл.

Реалiзувати використовуючи перегрузку оператора <<

**Вхiднi данi**

Цiле число Q - кiлькiсть запитiв.

У наступних рядках Q запитiв у зазначеному в умовi форматi.

**Вихiднi данi**

Вiдповiдi на запити у зазначеному в умовi форматi.

**Примiтки**

Гарантується, що усi данi коректнi. Виходу за межi списку або розмiр, бiльший нiж розмiр

списку недопустимi.

Iндекси починаються з нуля.

Завдання №4  
**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;

### **Мета задачі**

***Розуміння структур даних:*** Реалізація методу реверсу для зв’язаних списків є чудовим способом для поглиблення розуміння зв’язаних списків як фундаментальної структури даних. Він заохочує практичний підхід до вивчення того, як структуруються пов’язані списки та як ними маніпулювати.  
**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***.

### **Мета задачі**

***Розуміння рівності в структурах даних:*** це завдання допомагає зрозуміти, як визначається рівність у складних структурах даних, таких як зв’язані списки. На відміну від примітивних типів даних, рівність пов’язаного списку передбачає порівняння кожного елемента та їх порядку.

**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.

### Мета задачі

***Розуміння операцій зі структурами даних:*** це завдання унаочнює практичне використання списка для обчислювальних потреб. Арифметичні операції з великими числами це окремий клас задач, для якого використання списків допомагає обійти обмеження у представленні цілого числа сучасними комп’ютерами.  
**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

### **Мета задачі**

***Розуміння структур даних:*** Реалізація методу віддзеркалення бінарного дерева покращує розуміння структури бінарного дерева, виділення пам’яті для вузлів та зв’язування їх у єдине ціле. Це один з багатьох методів роботи з бінарними деревами.  
**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

### **Мета задачі**

***Розуміння структур даних:*** Реалізація методу підрахунку сум підвузлів бінарного дерева покращує розуміння структури бінарного дерева. Це один з багатьох методів роботи з бінарними деревами.

Завдання №5  
**SELF PRACTICE WORK ALGOTESTER**

Коли Коля та Вася прийшли робити ремонт на «Екстралогіку» — першим, що вони побачили в офісі, був стіл для настільного тенісу. Поки всі інші працювали, Коля та Вася вирішили пограти. Через декілька годин прийшов директор і накричав на заробітчан через те, що вони нічим не займаються. Тож Вася і Коля мусили йти працювати.

По дорозі вони сперечалися, хто ж виграв і з яким рахунком. Оскільки вони записували результати кожної подачі, то це можна порахувати. Але оскільки гра тривала дуже довго — порахувати це вручну дуже тяжко.

Всього відбулося nn подач. Про кожну з них ми знаємо, хто переміг. За виграну подачу гравець отримує одне очко. Партія вважається виграною, коли один з гравців набере не менше одинадцяти очок з перевагою щонайменше у два очки. Наприклад, за рахунків 11:9, 4:11, 15:13 партія закінчується, а за рахунків 11:10 та 99:98 — ні. Як тільки Коля і Вася закінчили одну партію — вони починають іншу.

Знаючи, хто переміг кожної подачі — виведіть загальний рахунок по партіях в грі Коля-Вася. А якщо вони не дограли останню партію, то і її рахунок теж.

# Вхідні дані

У першому рядку задано ціле число n — загальна кількість подач.

У другому рядку задано n символів ci. ci=K, якщо ii-ту подачу виграв Коля, та ci=V, якщо i-ту подачу виграв Вася.

# Вихідні дані

У першому рядку виведіть загальний рахунок гри по партіях у форматі k:v, де k — кількість партій, у яких переміг Коля, а v — кількість партій, у яких переміг Вася.

Якщо вони не дограли останню партію, то в другому рядку в такому ж форматі виведіть рахунок останньої партії.

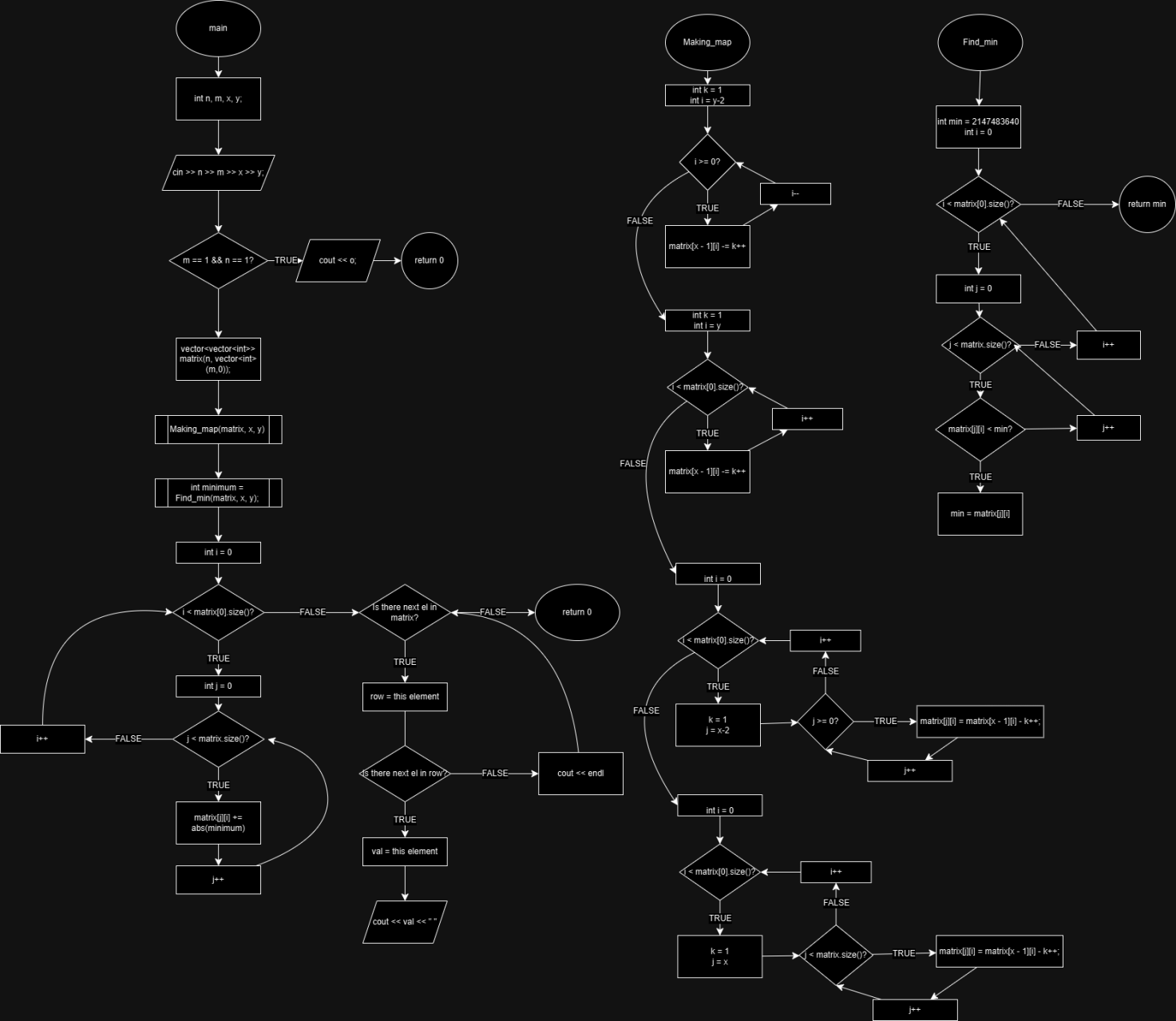
**2. Вимоги та планувальна оцінка часу виконання завдань:**

**Програма №1**

* Важливі деталі для реалізації програми.
* Усі операції з однозв'язним списком передбачають використання динамічної пам’яті (new і delete), тому важливо уникати витоків пам’яті, видаляючи вузли після видалення або очищення списку. Використовувати біблотеку fstream для запису даних у файл.
* Плановий час на реалізацію 2.5 години.

**Програма №2**

* Блок – схема

  
*Рисунок 2.1. Блок – схема до програми 1*

* Плановий час на реалізацію 2 години.

**Програма №3**

* Важливі деталі для реалізації програми.
* Зрозуміти, що таке List і навчитися його реалізовувати за допомогою вказівників та структури і перетворити це все у шаблон класу.
* Плановий час на реалізацію 5 годин.

**Програма №4**

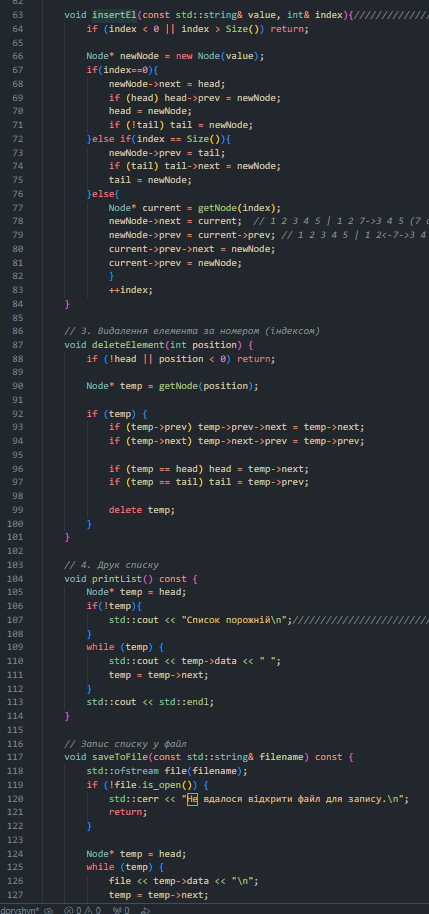
* Важливі деталі для реалізації програми.
* Зрозуміти, що таке Бінарне дерево пошуку та навчитися його реалзовувати.
* Плановий час на реалізацію 5 годин.

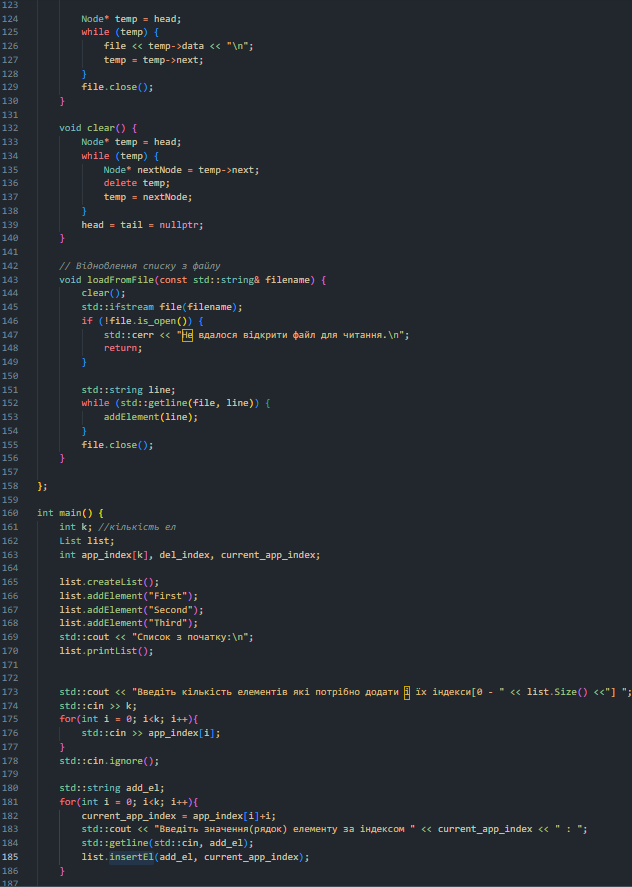
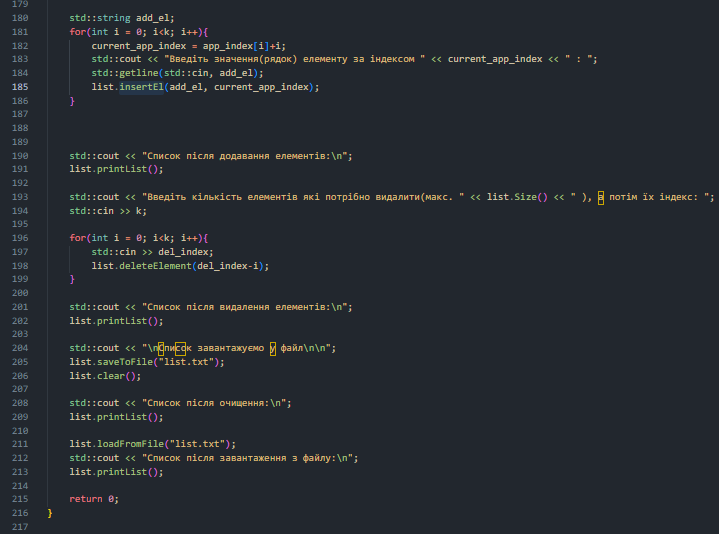
**Програма №5**

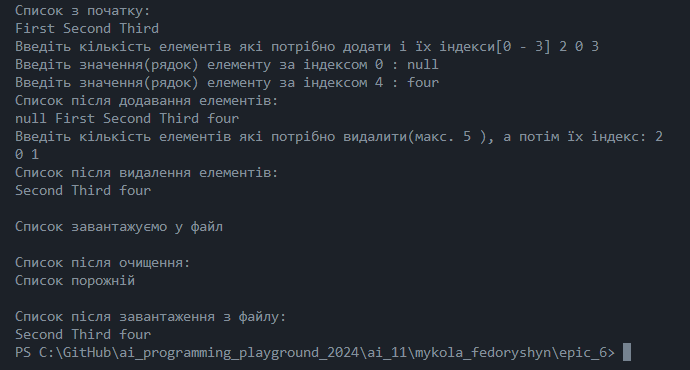
* Плановий час на реалізацію 1 година.

**3. Код програм з посиланням на зовнішні ресурси та фактично затрачений час**

**Завдання №1  
**

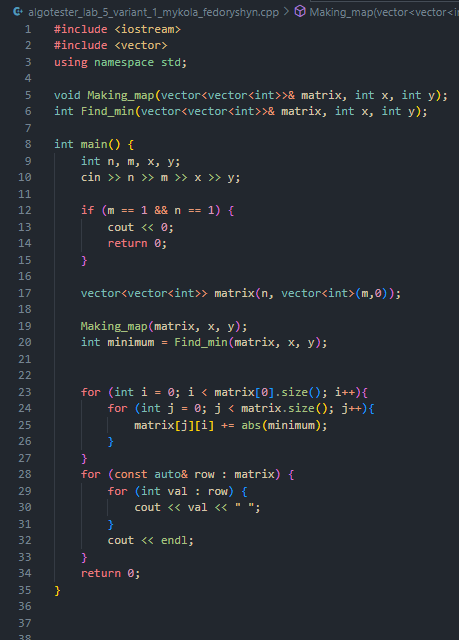
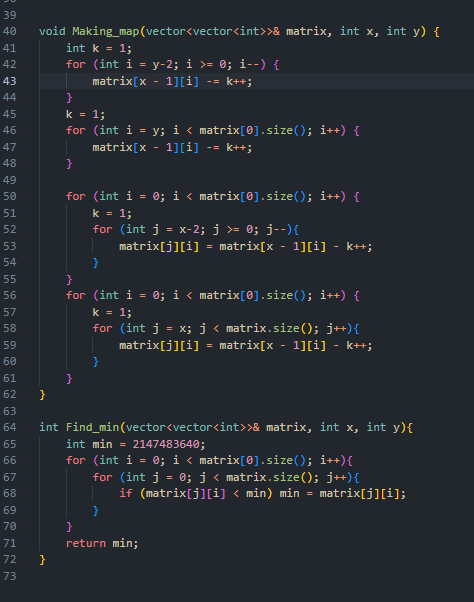
**

*  Рисунок 3.1. Код до програми № 1*

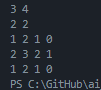
  
*Рисунок 3.2. Приклад виконання програми № 1*

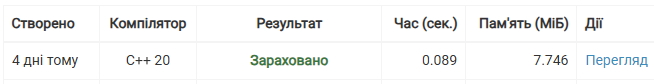
Фактично затрачений час 3 години.

**Посилання на файл у пулл реквесті**

**Завдання №2  
****

*Рисунок 3.3. Код до програми № 2*

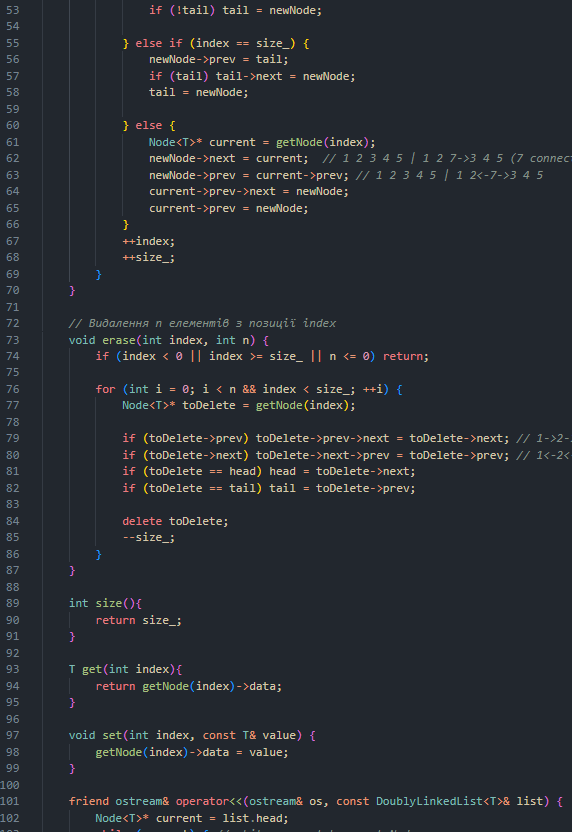
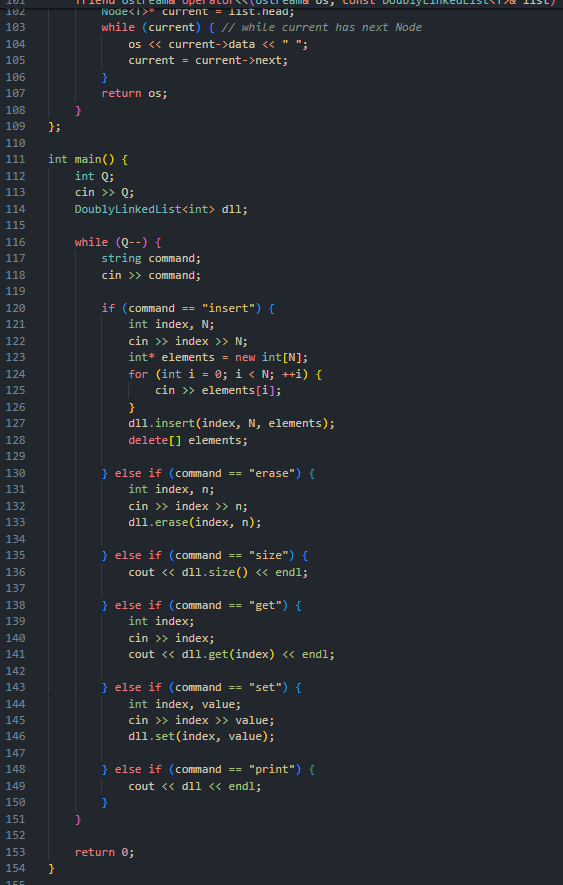
  
*Рисунок 3.4. Приклад виконання програми № 2*

  
*Рисунок 3.5. Статус задачі на алготестері*

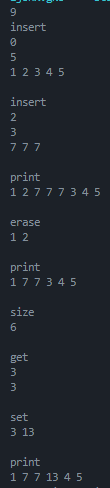
Фактично затрачений час 2 години.

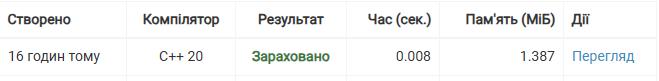
**Посилання на файл у пулл реквесті**

**Завдання №3**

***** Рисунок *

*3.6. Код до програми № 3*

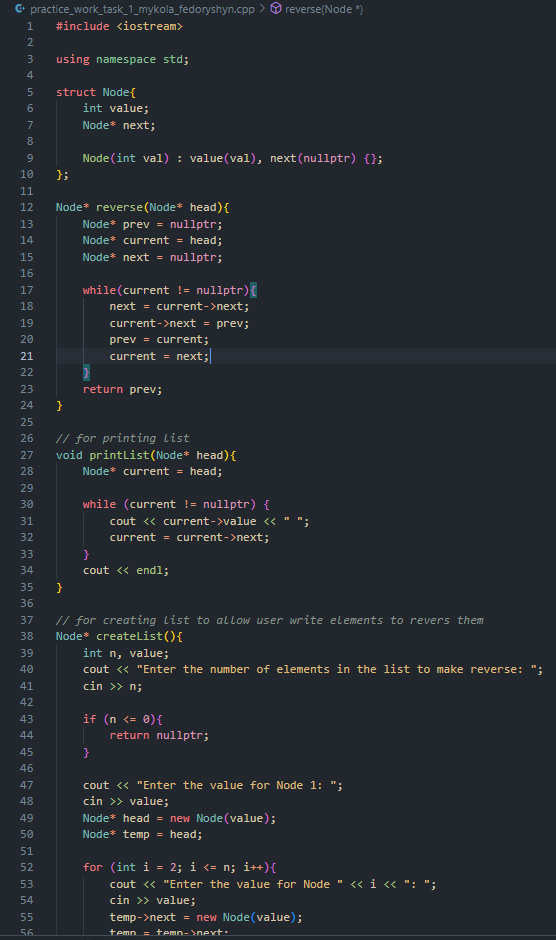
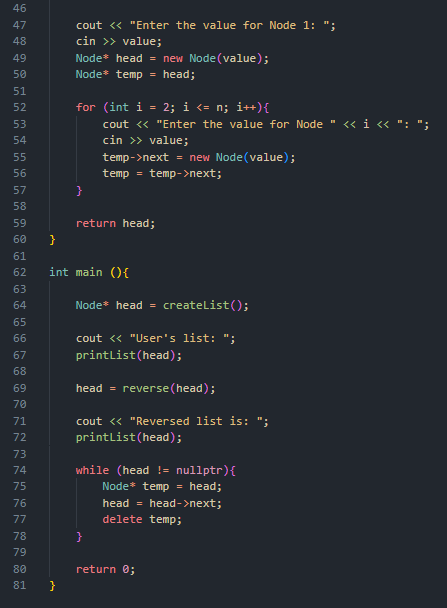
*  
Рисунок 3.7. Приклад виконання програми №3*

*  
Рисунок 3.8. Статус задачі на Algotester*

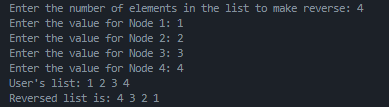
Фактично затрачений час 6 годин.

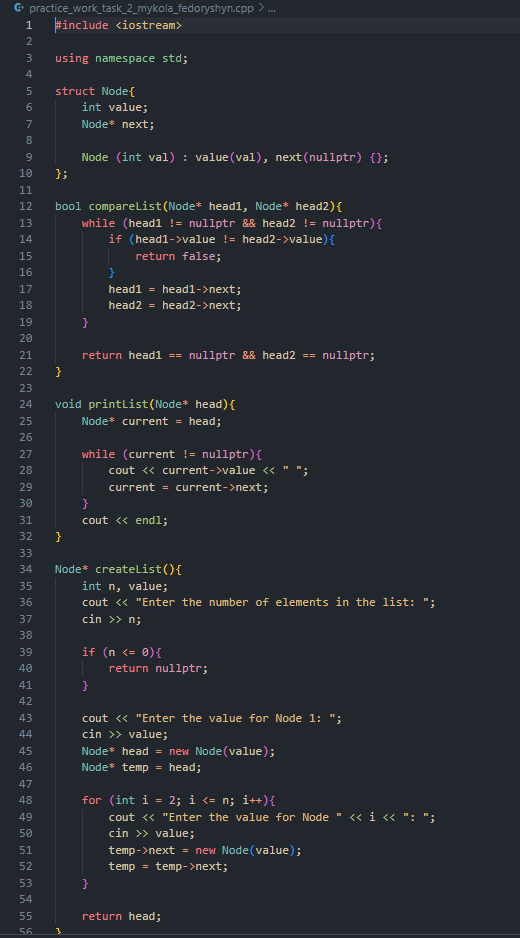
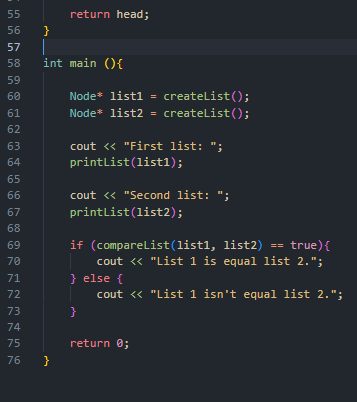
**Посилання на файл у пулл реквесті**

**Завдання №4**

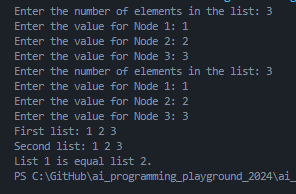
******

*Рисунок 3.9. Код до задачі номер 1*

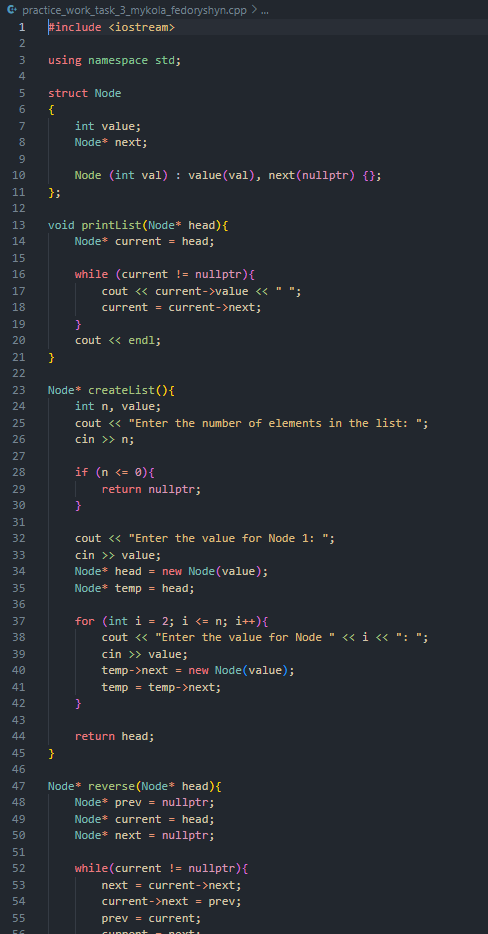
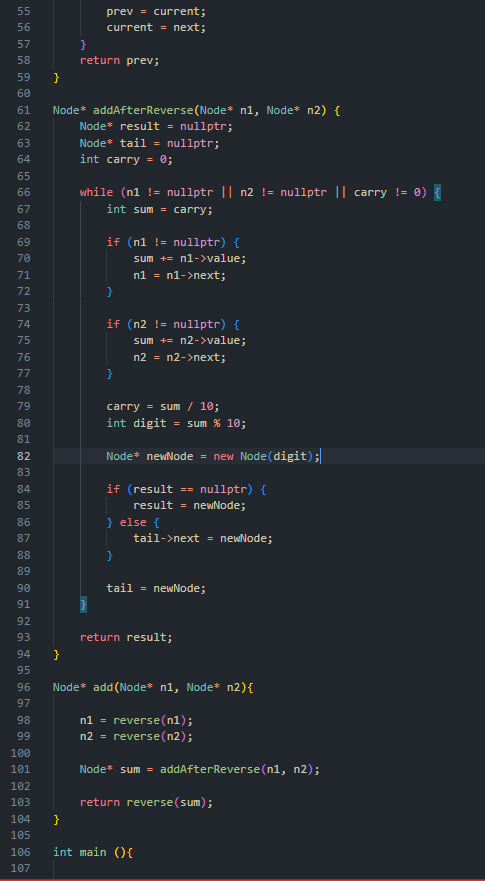
*****Рисунок 3.10. Приклад виконання задачі номер 1*

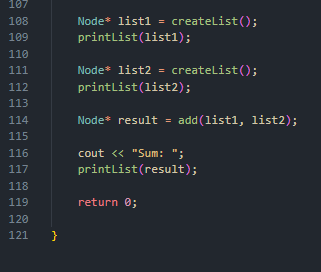
*  
*

*Рисунок 3.11. Код до задачі номер 2*

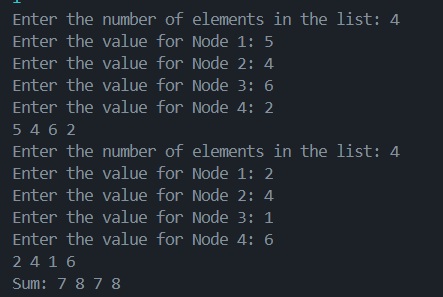
**

*Рисунок 3.12. Приклад виконання задаіч номер 2*

******

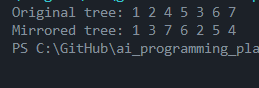
**

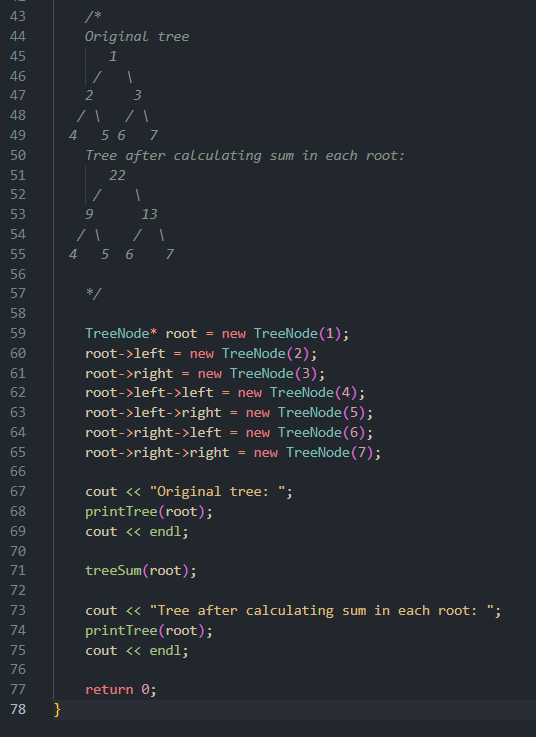
*Рисунок 3.13. Код до задачі номер 3*

**

*Рисунок 3.14. Приклад виконання програми номер 3*

**** ***Рисунок 3.15. Код до задачі номер 4*

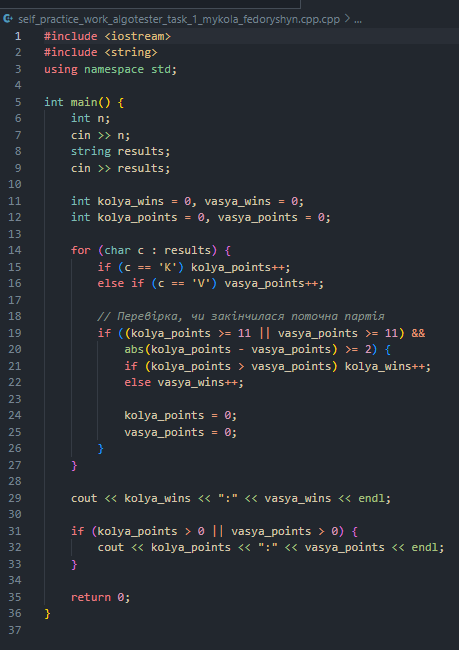
*****Рисунок 3.16. Приклад виконання задачі номер 4*

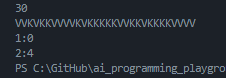
**** ***Рисунок 3.17. Код до задачі номер 5**  
Рисунок 3.18. Приклад виконання задачі номер 5*

Фактично затрачений час 5 годин.

**Посилання на файл у пулл реквесті**

**Завдання №5**

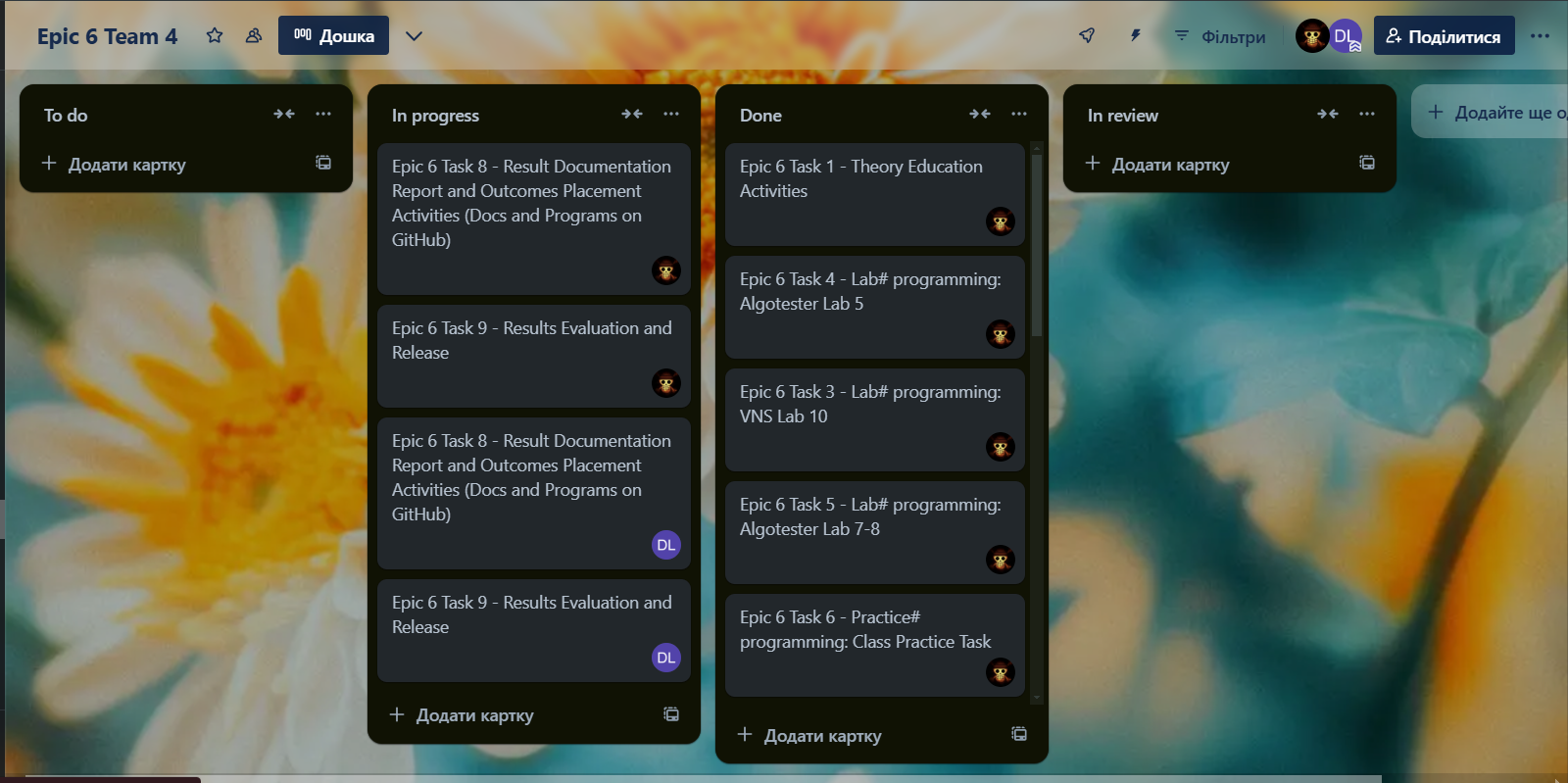
*****Рисунок 3.19. Код до програми №5*

*****Рисунок 3.20. Приклад виконання пргограми номер 5*

*****Рисунок 3.21. Статус задачі на Algotester*

Фактично затрачений час 2 години.

**4. Робота з командою:**

  
*Рисунок 4.1. Командна дошка в Trello*

**Висновок:** У межах практичних та лабораторних робіт блоку №6, я вивчив багато нового матеріалу, такого як: динамічні структури (черга, стек, списки, дерево), алгоритми обробки динамічних структур. На практиці попрацював з бінарними деревами, створив зв’язаний список. Створив дошку в Trello для комфортної роботи з командою.