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.»

до:

Практичних Робіт до блоку № 3

**Виконала:**

Студентка групи ШІ-12

Іванів Христина Вікторівна

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

**Мета роботи**: Ознайомитись з основними динамічними структурами даних, ознайомитись з алгоритмами їх обробки.

**Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:
2. Тема №1: Основи Динамічних Структур Даних:
3. Тема №2: Стек
4. Тема №3: Черга
5. Тема №4: Зв'язні Списки
6. Тема №5: Дерева
7. Тема №6: Алгоритми Обробки Динамічних Структур
8. Індивідуальний план опрацювання теорії:

* Тема №1: Основи Динамічних Структур Даних
* Джерела Інформації:
* Лекції О. Пшеничного
* Практичні М. Фаріон
* Уроки 58 з курсу С++ теорія з каналу «Блоган»
* Що опрацьовано:
* Лекції О. Пшеничного
* Практичні М. Фаріон
* Уроки 58 з курсу С++ теорія з каналу «Блоган»
* Статус: ознаймолена з основними динамічними структурами даних
* Тема №2: Стек
* Джерела Інформації:
* Лекції О. Пшеничного
* Практичні М. Фаріон
* Уроки 142 з курсу С++ теорія з каналу «Блоган»
* Що опрацьовано:
* Лекції О. Пшеничного
* Практичні М. Фаріон
* Уроки 142 з курсу С++ теорія з каналу «Блоган»
* Статус: ознайомлена з стеком та роботою з ним

* Тема №3: Черга
* Джерела Інформації:
* Лекції О. Пшеничного
* Практичні М. Фаріон
* Уроки 141з курсу С++ теорія з каналу «Блоган»
* Що опрацьовано:
* Лекції О. Пшеничного
* Практичні М. Фаріон
* Уроки 141 з курсу С++ теорія з каналу «Блоган»
* Статус: ознайомлена з чергою та її практичним застосуванням
* Тема №4: Зв'язні Списки
* Джерела Інформації:
* Лекції О. Пшеничного
* Практичні М. Фаріон
* Уроки 139, 140 з курсу С++ теорія з каналу «Блоган»
* Що опрацьовано:
* Лекції О. Пшеничного
* Практичні М. Фаріон
* Уроки 139, 140 з курсу С++ теорія з каналу «Блоган»
* Статус: ознайомлена з однозв’язними та двозв’язними списками
* Тема №5: Дерева
* Джерела Інформації:
* Лекції О. Пшеничного
* Практичні М. Фаріон
* Уроки 144 з курсу С++ теорія з каналу «Блоган»
* Урок #11 по С++ з ютуб каналу «Школа програмування»
* Що опрацьовано:
* Лекції О. Пшеничного
* Практичні М. Фаріон
* Урок #11 по С++ з ютуб каналу «Школа програмування»
* Уроки 144 з курсу С++ теорія з каналу «Блоган»
* Статус: ознайомлена з деревами, роботою з ними
* Тема №6: Алгоритми Обробки Динамічних Структур
* Джерела Інформації:
* Лекції О. Пшеничного
* Практичні М. Фаріон
* Що опрацьовано:
* Лекції О. Пшеничного
* Практичні М. Фаріон
* Статус: ознайомлена з алгоритмами обробки динамічних структур

**Виконання роботи:**

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

**Завдання №1 VNS.** Лабораторна робота №10

* ***Варіант завдання: 6***
* *Деталі завдання*:

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.

* *Важливі деталі для врахування*:

Записи в лінійному списку містять ключове поле типу int. Сформувати

двонаправлений список. Знищити з нього елемент із заданим номером,

додати елемент у початок списку.

**Завдання №2 VNS. Algotester task 5. V-1**

* ***Варіант завдання: 1***
* *Деталі завдання:*

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 i-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 aa із 8-ми байт, а саме:

* 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 NN заклинань і вони дуже просять Вас їм допомогти.

* *Важливі деталі для врахування*:

# Input

У першому рядку oдне число a - поточний стан дошки.

У другому рядку N - кількість заклинань.

У наступних N рядках по 2 числа Ri, Ci - рядок і стовпець кулі над якою виконується заклинання.

# Output

Одне число b - стан дошки після виконання N заклинань.

# Constraints

0≤N≤1030≤N≤103

1≤Ri,Ci≤81≤Ri,Ci≤8

0≤a,b<264

**Завдання** **№3 Algotester task 7.8. V- 2**

* ***Варіант завдання*: 2**
* *Деталі завдання*

Ваше завдання - власноруч реалiзувати структуру даних "Динамiчний масив".

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 масиву, який треба вставити. У третьому рядку N цiлих чисел - масив, 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 динамiчному масивi.

• Визначення кiлькостi зарезервованої пам’ятi:

Iдентифiкатор – capacity

Ви не отримуєте аргументiв. Ви виводите кiлькiсть зарезервованої пам’ятi у динамiчному масивi. Ваша реалiзацiя динамiчного масиву має мати фактор росту (Growth factor) рiвний 2.

• Отримання значення i-го елементу

Iдентифiкатор - get

Ви отримуєте цiле число - index, iндекс елемента. Ви виводите значення елемента за iндексом. Реалiзувати використовуючи перегрузку оператора

• Модифiкацiя значення i-го елементу

Iдентифiкатор - set

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

• Вивiд динамiчного масиву на екран

Iдентифiкатор - print

Ви не отримуєте аргументiв. Ви виводите усi елементи динамiчного масиву через пробiл. Реалiзувати використовуючи перегрузку оператора

* *Важливі деталі для врахування*:

Для того щоб отримати 50% балiв за лабораторну достатньо написати свою структуру.

**Завдання** **№4 VNS. Algotester task 7.8. V- 2**

* ***Варіант завдання*: 2**
* *Деталі завдання*

Ваше завдання - власноруч реалiзувати структуру даних "Динамiчний масив".

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 масиву, який треба вставити. У третьому рядку N цiлих чисел - масив, 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 динамiчному масивi.

• Визначення кiлькостi зарезервованої пам’ятi:

Iдентифiкатор – capacity

Ви не отримуєте аргументiв. Ви виводите кiлькiсть зарезервованої пам’ятi у динамiчному масивi. Ваша реалiзацiя динамiчного масиву має мати фактор росту (Growth factor) рiвний 2.

• Отримання значення i-го елементу

Iдентифiкатор - get

Ви отримуєте цiле число - index, iндекс елемента. Ви виводите значення елемента за iндексом. Реалiзувати використовуючи перегрузку оператора

• Модифiкацiя значення i-го елементу

Iдентифiкатор - set

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

• Вивiд динамiчного масиву на екран

Iдентифiкатор - print

Ви не отримуєте аргументiв. Ви виводите усi елементи динамiчного масиву через пробiл. Реалiзувати використовуючи перегрузку оператора

* *Важливі деталі для врахування*:

Для отримання 100% балiв ця структура має бути написана як шаблон класу, у якостi

параметру використати int.

**Завдання** **№5 Class Practice Work**

# Деталі завдання

## 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.

***Розвиток алгоритмічне мислення:*** Це завдання розвиває алгоритмічне мислення. Перевертання пов’язаного списку вимагає логічного підходу до маніпулювання покажчиками, що є ключовим навиком у інформатиці.

***Засвоїти механізми маніпуляції з покажчиками:*** пов’язані списки значною мірою залежать від покажчиків. Це завдання покращить навички маніпулювання вказівниками, що є ключовим аспектом у таких мовах, як C++.

***Розвинути навички розв’язувати задачі:*** перевернути пов’язаний список непросто й вимагає творчого й логічного мислення, таким чином покращуючи свої навички розв’язування поставлених задач.

### 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***.

### 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.

***Поглиблення розуміння зв’язаних списків:*** Порівнюючи зв’язані списки, дозволяють покращити своє розуміння обходу, фундаментальної операції в обробці зв’язаних списків.

***Розуміння ефективність алгоритму:*** це завдання також вводить поняття ефективності алгоритму. Студенти вчаться ефективно порівнювати елементи, що є навичкою, важливою для оптимізації та зменшення складності обчислень.

***Розвинути базові навики роботи з реальними програми:*** функції порівняння мають вирішальне значення в багатьох реальних програмах, таких як виявлення змін у даних, синхронізація структур даних або навіть у таких алгоритмах, як сортування та пошук.

***Розвинути навик вирішення проблем і увага до деталей:*** це завдання заохочує скрупульозний підхід до програмування, оскільки навіть найменша неуважність може призвести до неправильних результатів порівняння. Це покращує навички вирішення проблем і увагу до деталей.

### 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.

### 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.

***Поглиблення розуміння зв’язаних списків:*** Застосовування зв’язаних списків для арифметичних операції з великими числами дозволяє покращити розуміння операцій з обробки зв’язаних списків.

***Розуміння ефективність алгоритму:*** це завдання дозволяє порівняти швидкість алгоритму додавання з використанням списків зі швидкістю вбудованих арифметичних операцій. Студенти вчаться розрізняти позитивні та негативні ефекти при виборі структур даних для реалізації практичних програм.

***Розвинути базові навики роботи з реальними програми:*** арифметичні операції з великими числами використовуються у криптографії, теорії чисел, астрономії, та ін.

***Розвинути навик вирішення проблем і увага до деталей:*** завдання покращує розуміння обмежень у представленні цілого числа сучасними комп’ютерами та пропонує спосіб його вирішення.

# Бінарні дерева

## 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.

***Розвиток алгоритмічне мислення:*** Це завдання розвиває алгоритмічне мислення. Прохід всіх вузлів дерева продемонструє розгортання рекурсивного виклику.

## 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.

***Розвиток алгоритмічне мислення:*** Це завдання розвиває алгоритмічне мислення. Прохід всіх вузлів дерева демонструє розгортання рекурсивного виклику.

**Завдання** **№6 Self Practice Algotester Task**

# *Деталі завдання*:

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 пік 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.

*Важливі деталі для врахування:*

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.

Обмеження

1 ≤ N, M ≤ 103

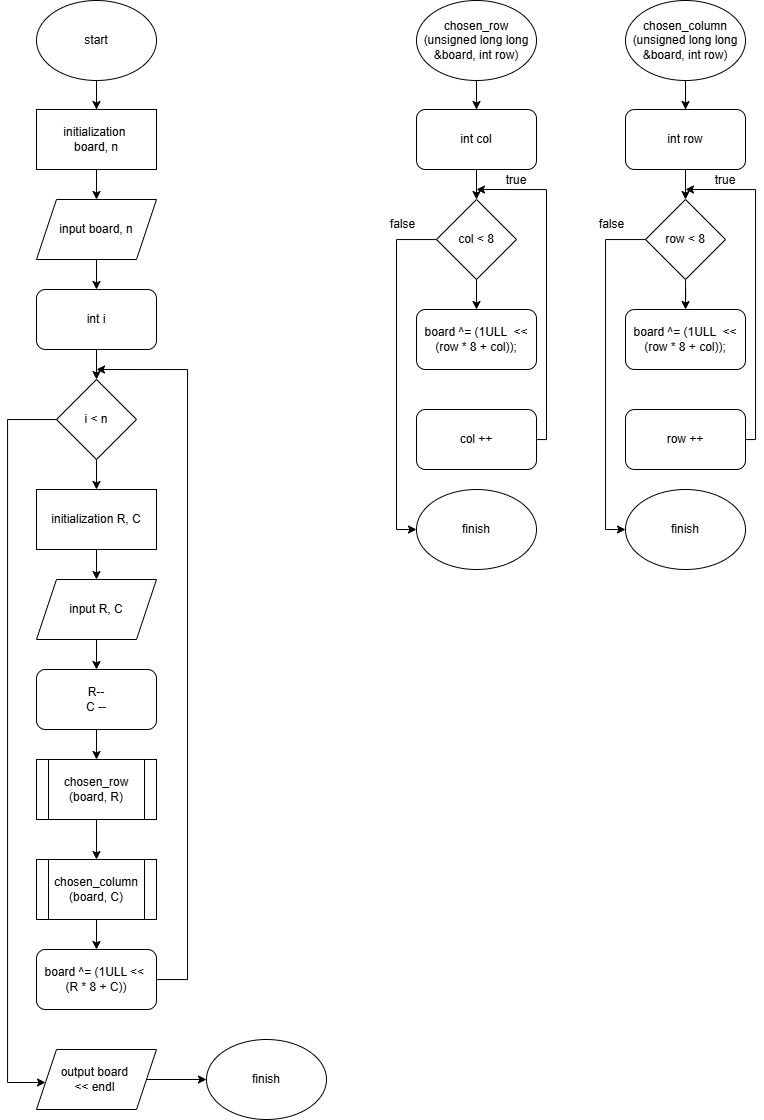
1 ≤ x ≤ N

1 ≤ y ≤ M

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

**Завдання №1 VNS.** Лабораторна робота №10

Плановий час на реалізацію: 4години  
**Завдання №2 VNS. Algotester task 5. V-1**



Плановий час на реалізацію: 4 години

**Завдання** **№3 Algotester task 7.8. V- 2**

Плановий час на реалізацію: 4 години

**Завдання** **№4 Algotester task 7.8. V- 2**

Плановий час на реалізацію: 4 години

**Завдання** **№5 Class Practice Work**

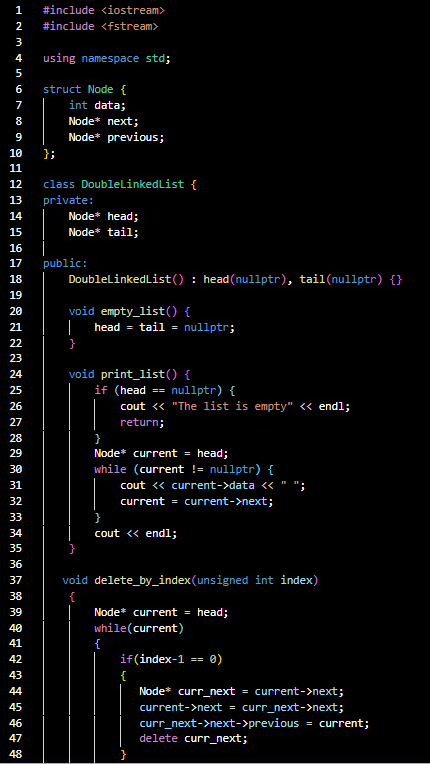
Плановий час на реалізацію: 6 годин

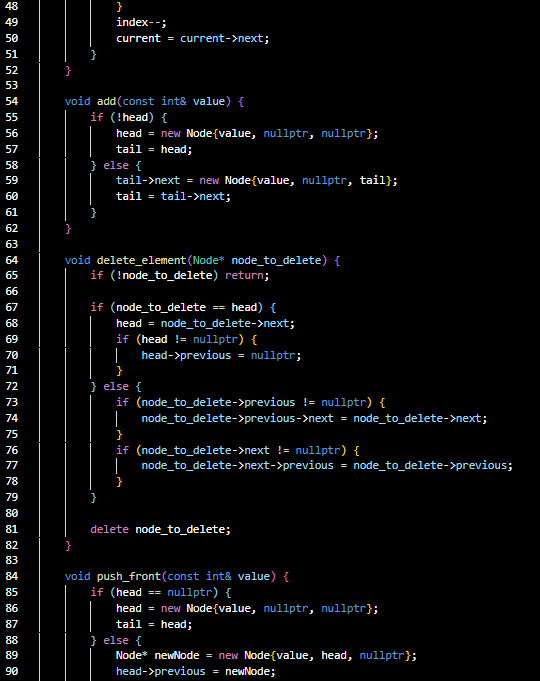
**Завдання** **№6 Self Practice Algotester Task**

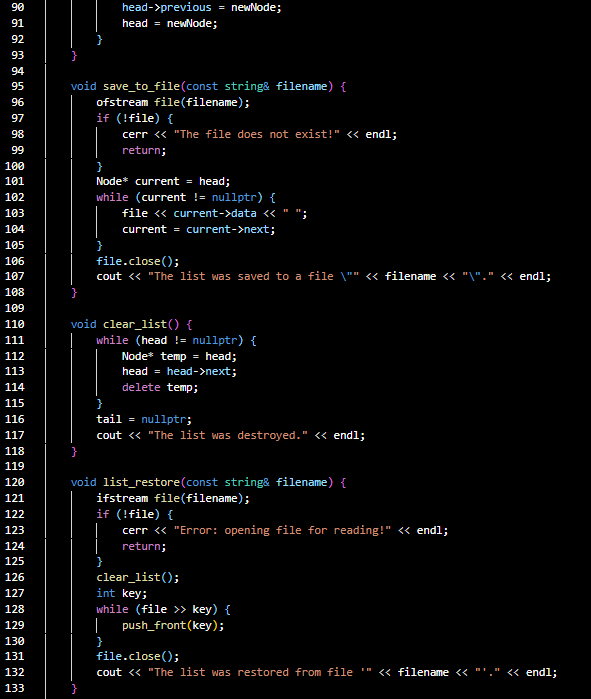
Плановий час на реалізацію: 2 години

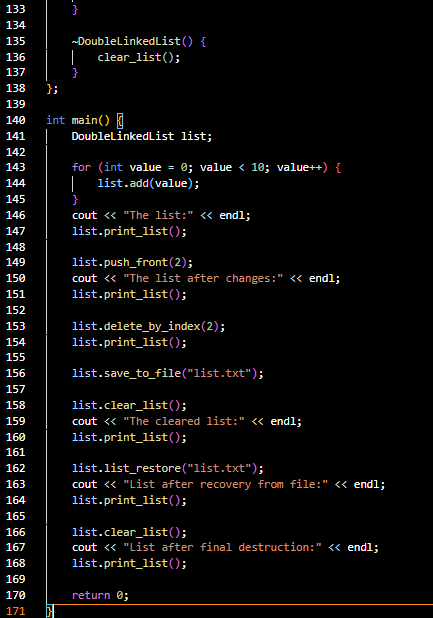
1. ***Код програми та фактичний час на реалізацію:***

**Завдання №1 VNS.** Лабораторна робота №10



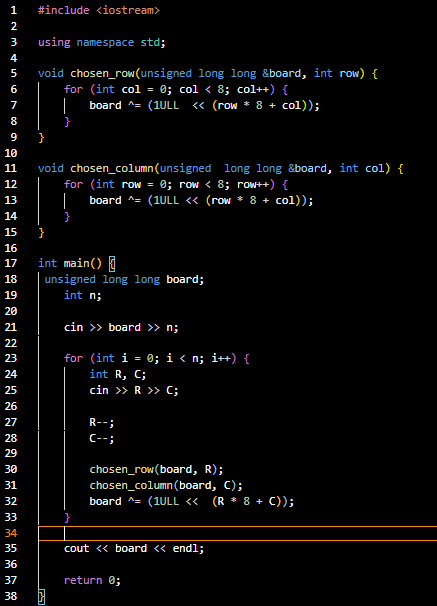






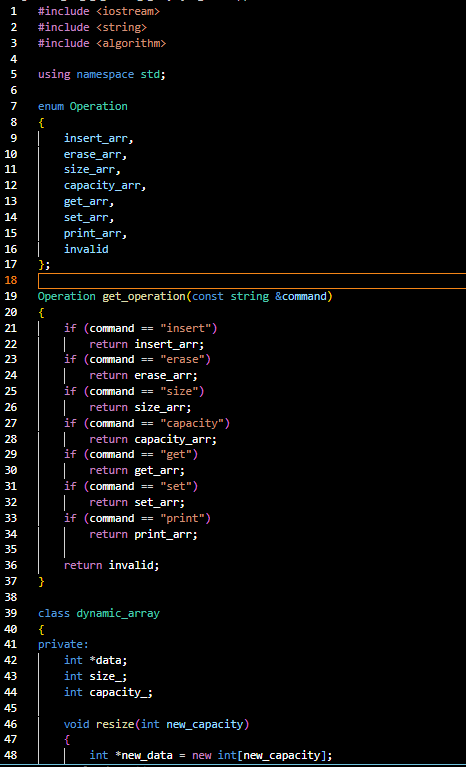
Фактичний час на реалізацію: 4 години

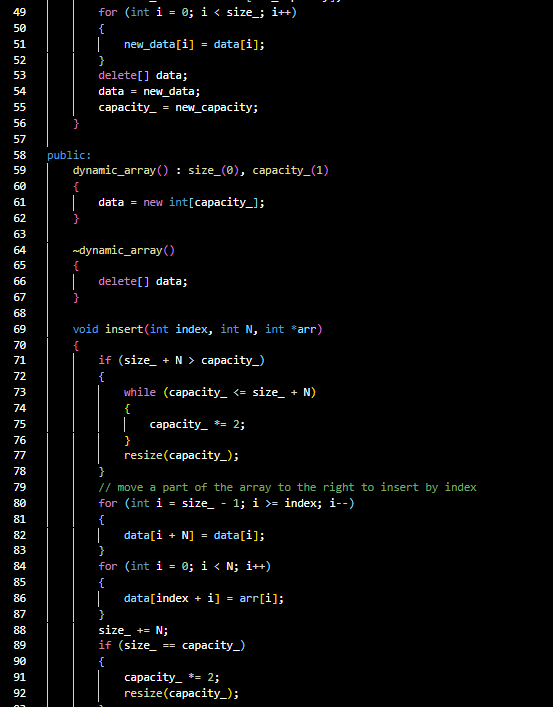
**Завдання №2 VNS. Algotester task 5. V-1**

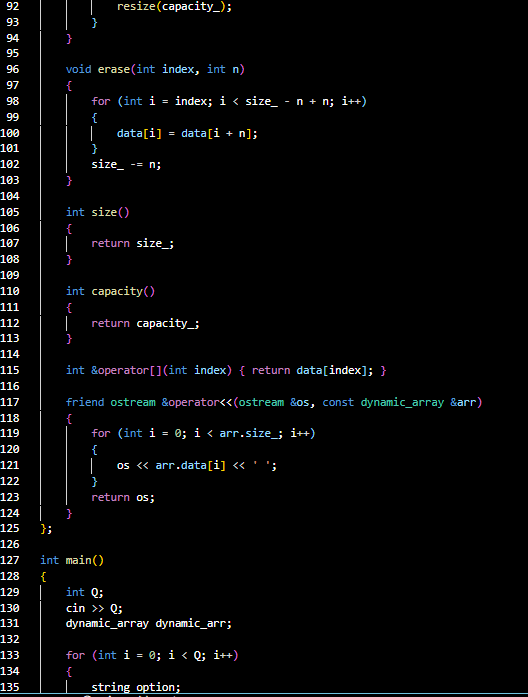
**

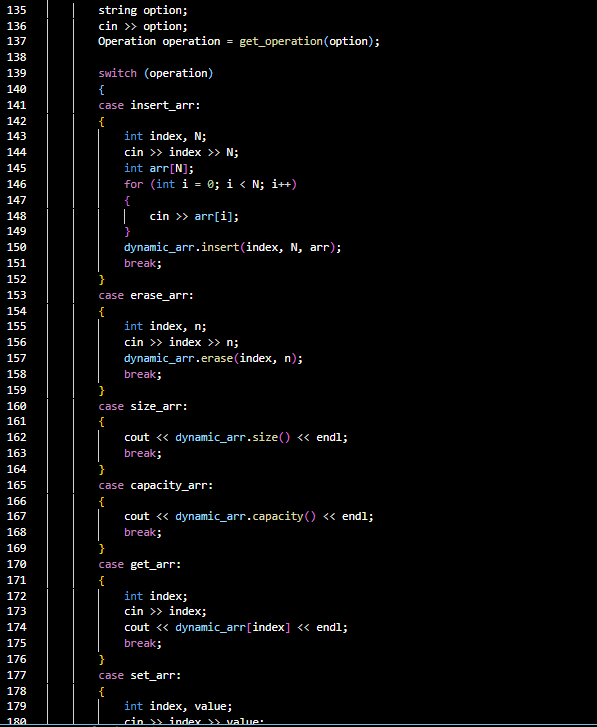
Фактичний час на реалізацію: 2 години

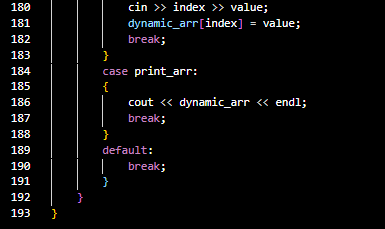
**Завдання** **№3 Algotester task 7.8. V- 2**

****

****

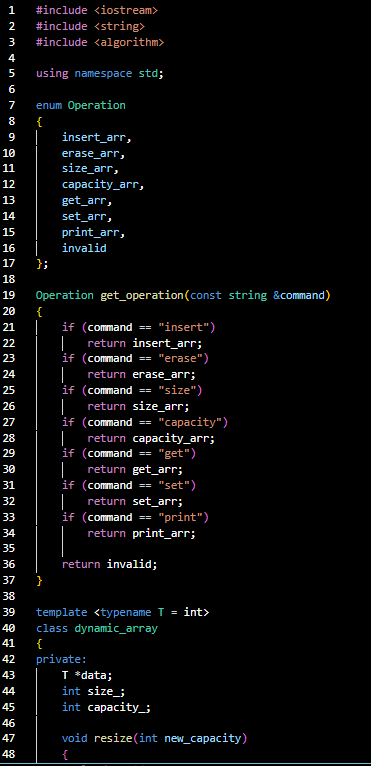
****

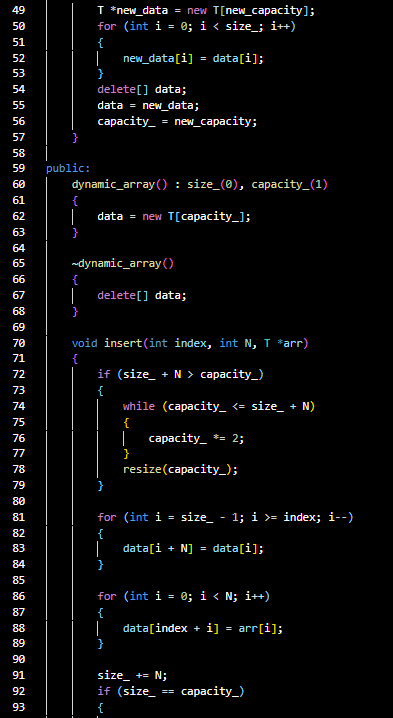
****

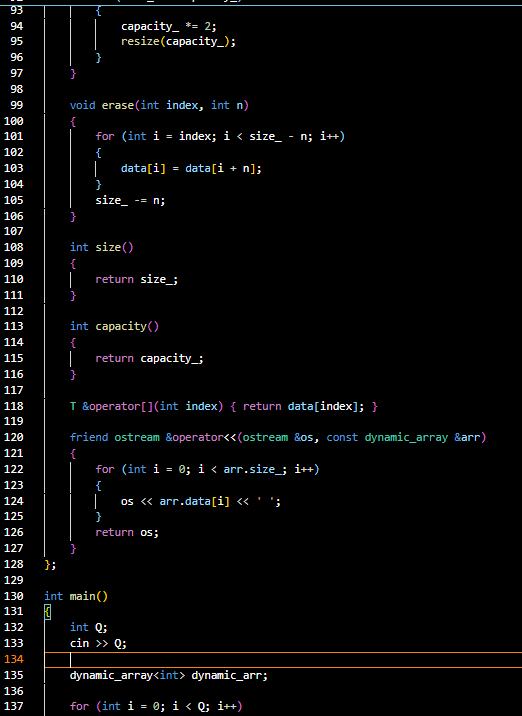
****

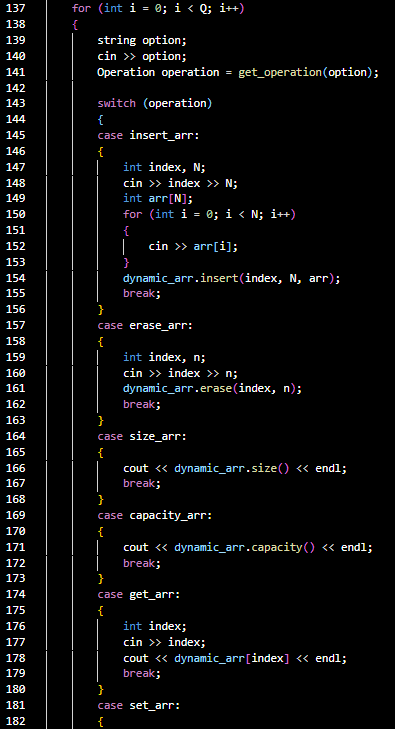
Фактичний час на реалізацію: 4 години

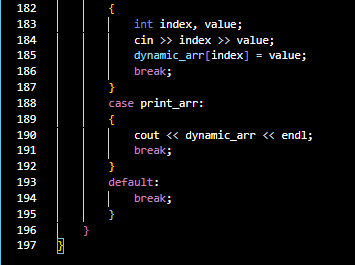
**Завдання** **№4 VNS. Algotester task 7.8. V- 2**





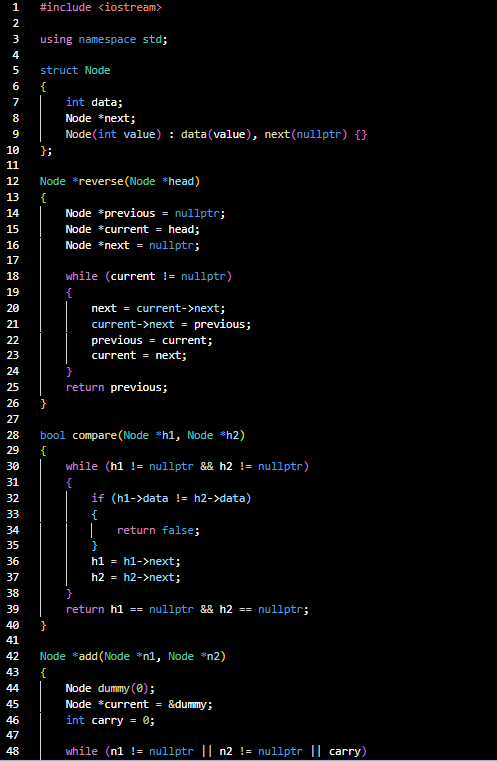


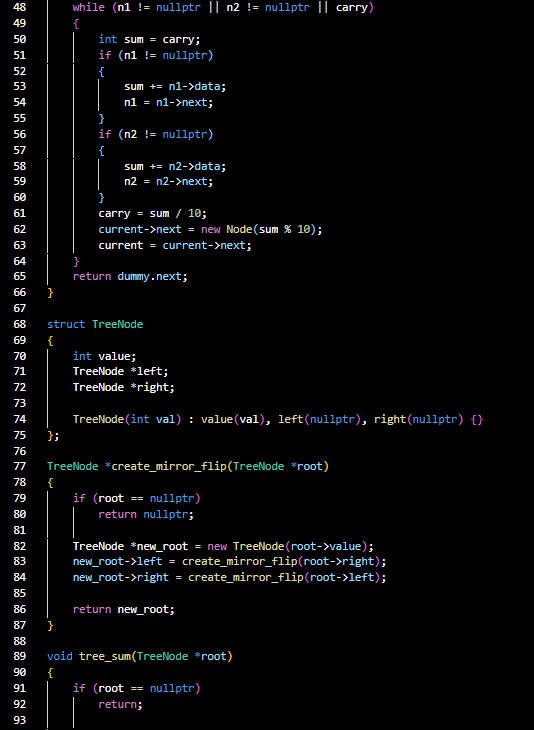


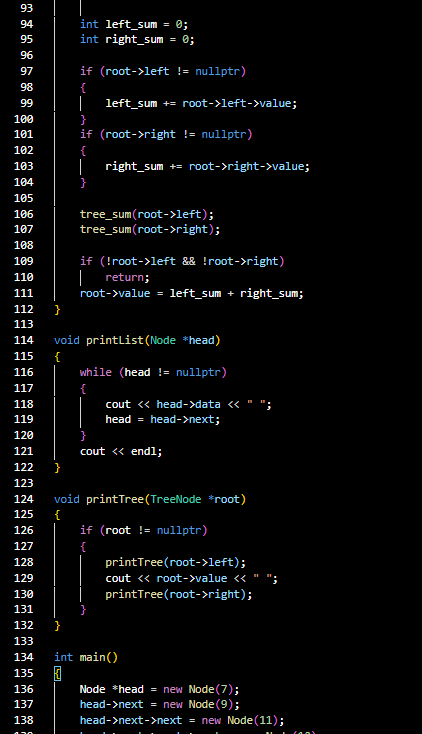


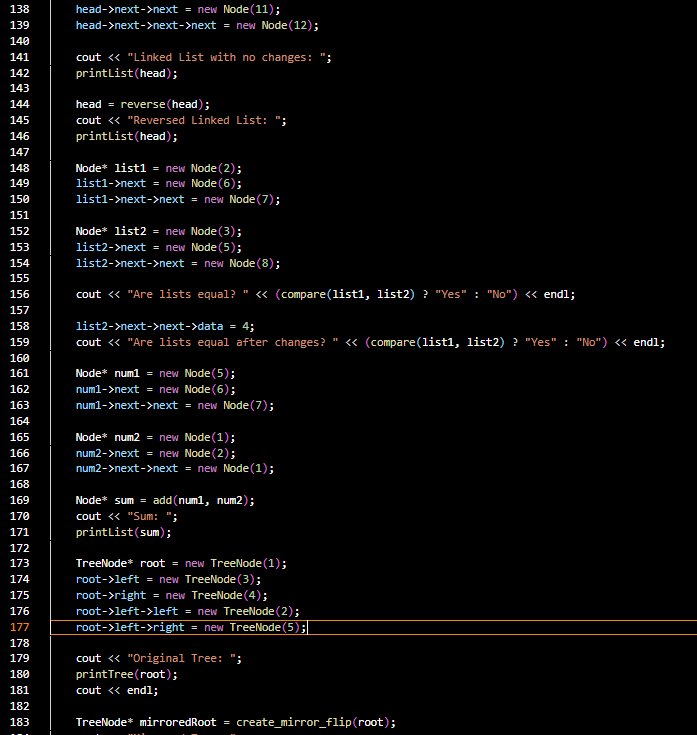
Фактичний час на реалізацію: 4 години

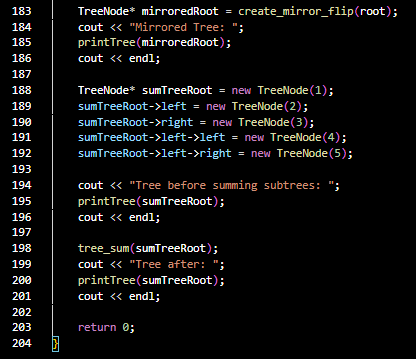
**Завдання** **№5 Class Practice Work**

****

****

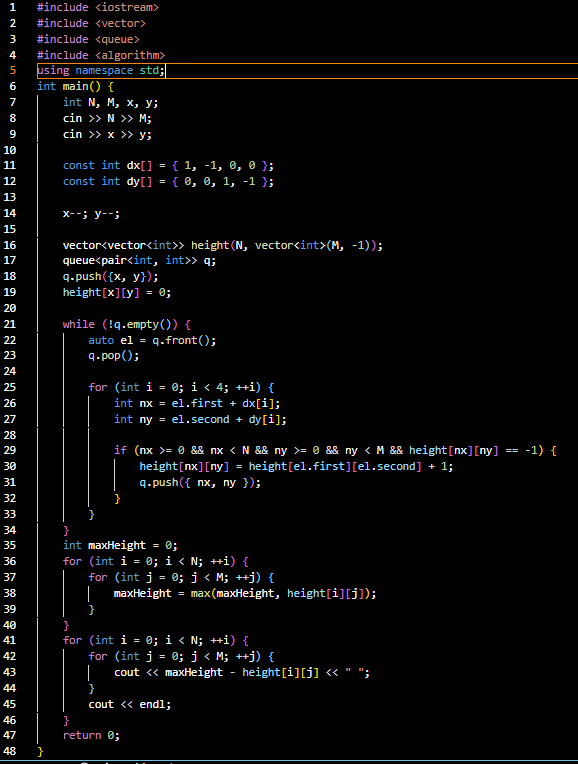
****

****

****

Фактичний час на реалізацію: 4 години

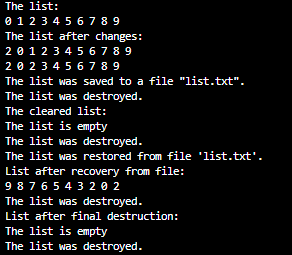
**Завдання** **№6 Self Practice Algotester Task**

****

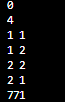
Фактичний час на реалізацію: 2 години

1. ***Результати виконання завдань, тестування:***

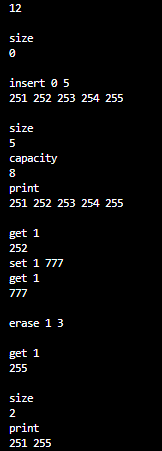
**Завдання №1 VNS.** Лабораторна робота №10

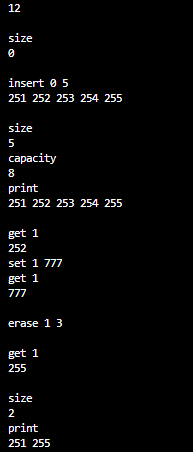


**Завдання №2 VNS. Algotester task 5. V-1**

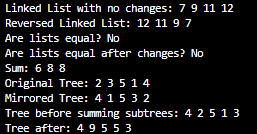
******

**Завдання №3 Algotester task 7.8. V - 2**



**Завдання №4 Algotester task 7.8. V- 2**  

**Завдання** **№5 Class Practice Work**

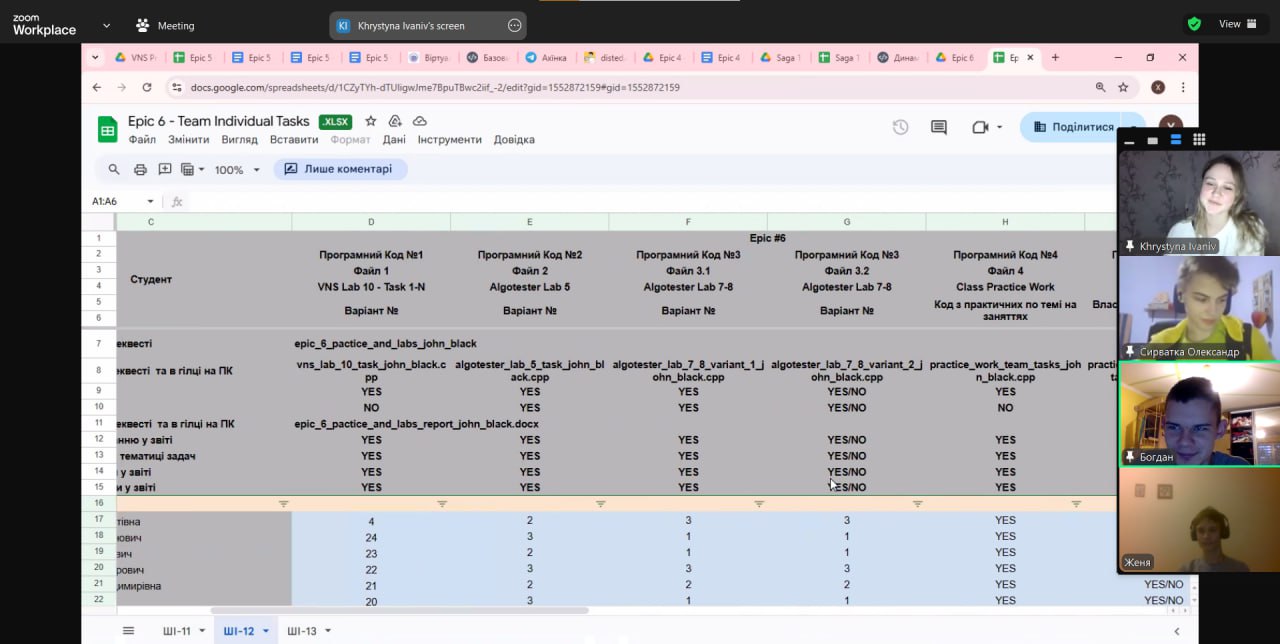
****

**Завдання** **№6 Self Practice Algotester Task**

******

***5 . Кооперація з командою:***

***Провели зустріч у зумі, обговорили деталі виконянна завдань***

******

***Висновок:***Під час виконання епіку *я* ознайомилась з динамічними структурами (Черга, Стек, Списки, Дерево) та алгоритмами обробки динамічних структур.