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 середрвище розробки програми.»

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

до:

Практичних Робіт до блоку № 6

**Виконала:**

Студентка групи ШІ-13

Бобринок Ангеліна Вадимівна

**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

**Виконання роботи:**

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

**Завдання №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.

Записи в лінійному списку містять ключове поле типу int. Сформувати

однонаправлений список. Знищити з нього К елементів, починаючи із

заданого номера, додати елемент перед елементом із заданим ключем;

**Завдання №2**

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.

**Завдання №3**

Ваше завдання - власноруч реалізувати структуру даних "Динамічний масив".  
Ви отримаєте QQ запитів, кожен запит буде починатися зі слова-ідентифікатора, після якого йдуть його аргументи.  
  
Вам будуть поступати запити такого типу:

* **Вставка**:  
  Ідентифікатор - insertinsert  
  Ви отримуєте ціле число indexindex елемента, на місце якого робити вставку.  
  Після цього в наступному рядку рядку написане число NN - розмір масиву, який треба вставити.  
  У третьому рядку NN цілих чисел - масив, який треба вставити на позицію indexindex.
* **Видалення**:  
  Ідентифікатор - eraseerase  
  Ви отримуєте 2 цілих числа - indexindex, індекс елемента, з якого почати видалення та nn - кількість елементів, яку треба видалити.
* **Визначення розміру**:  
  Ідентифікатор - sizesize  
  Ви не отримуєте аргументів.  
  Ви виводите кількість елементів у динамічному масиві.
* **Визначення кількості зарезервованої пам’яті**:  
  Ідентифікатор - capacitycapacity  
  Ви не отримуєте аргументів.  
  Ви виводите кількість зарезервованої пам’яті у динамічному масиві.  
  Ваша реалізація динамічного масиву має мати фактор росту ([Growth factor](https://en.wikipedia.org/wiki/Dynamic_array" \l "Growth_factor)) рівний 2.
* **Отримання значення** ii-го елементу  
  Ідентифікатор - getget  
  Ви отримуєте ціле число - indexindex, індекс елемента.  
  Ви виводите значення елемента за індексом. Реалізувати використовуючи перегрузку оператора [][]
* **Модифікація значення** ii-го елементу  
  Ідентифікатор - setset  
  Ви отримуєте 2 цілих числа - індекс елемента, який треба змінити, та його нове значення. Реалізувати використовуючи перегрузку оператора [][]
* **Вивід динамічного масиву на екран**  
  Ідентифікатор - printprint  
  Ви не отримуєте аргументів.  
  Ви виводите усі елементи динамічного масиву через пробіл.  
  Реалізувати використовуючи перегрузку оператора <<

**Завдання №4**

***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;

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***.

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.

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

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 №5**

Снігурочка — на сьогодні дуже велика рідкість, одним словом, вимираючий вид. А от тих Дідів Морозів порозводилося вже. Хоч бери й відстрілюй...

І ось напередодні Нового року снігурочка вирішила скористатися своїм монополістичним становищем і оголосила конкурс для Дідів Морозів. Переможець конкурсу стане якраз тим щасливчиком, якому вона складе пару в новорічну ніч.

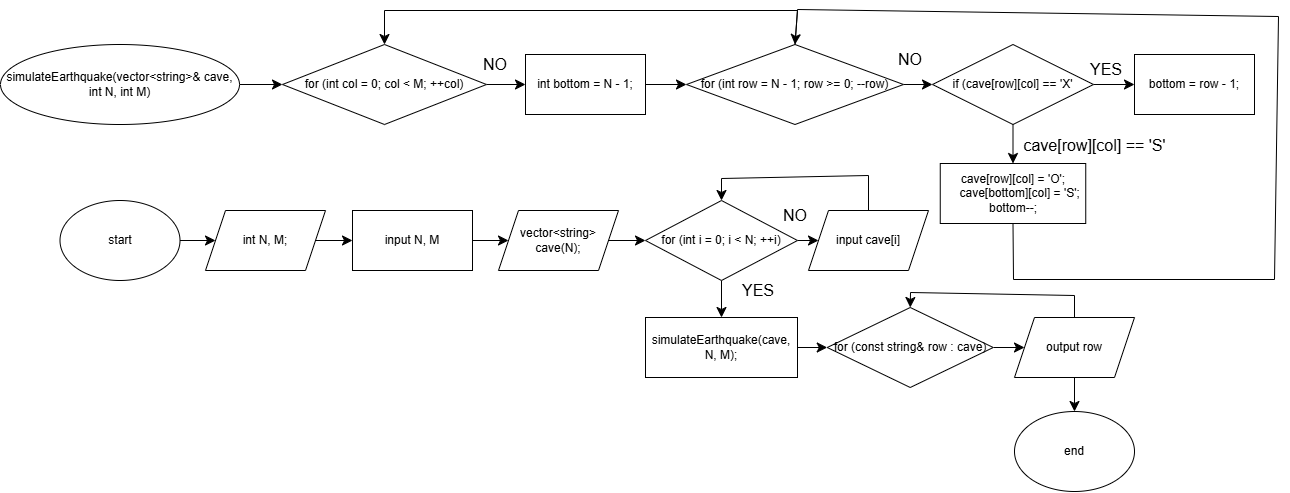
Правила конкурсу до болю прості. Кожен Дід Мороз приходить і урочисто зачитує снігурочці свій вірш, який він готував протягом довгих недоспаних ночей. Снігурочка ж у свою чергу ставить кожному з них оцінку — ціле невід’ємне число. Стратегія оцінювання снігурочки теж до болю тупа — вона підраховує кількість паліндромів у вірші дідуся-претендента і трактує її як його оцінку.

Нагадаю, що паліндром — це слово, що читається однаково як зліва направо, так і справа наліво. При цьому регістр букв не враховується.

Кількість кандидатів на новорічну ніч є величезна, от і вирішила Снігурочка автоматизувати процес оцінювання.

Вам і доведеться написати програму, яка оцінює вірш невідомого Діда Мороза.

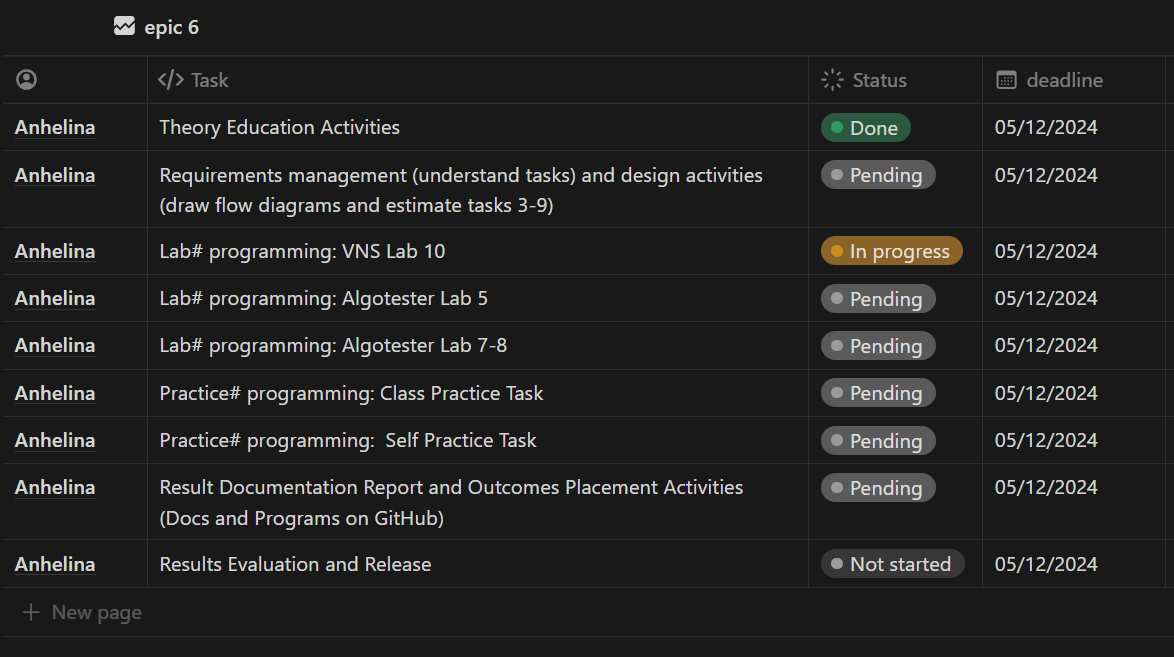
1. *Дизайн та планована оцінка часу виконання завдань:*

**

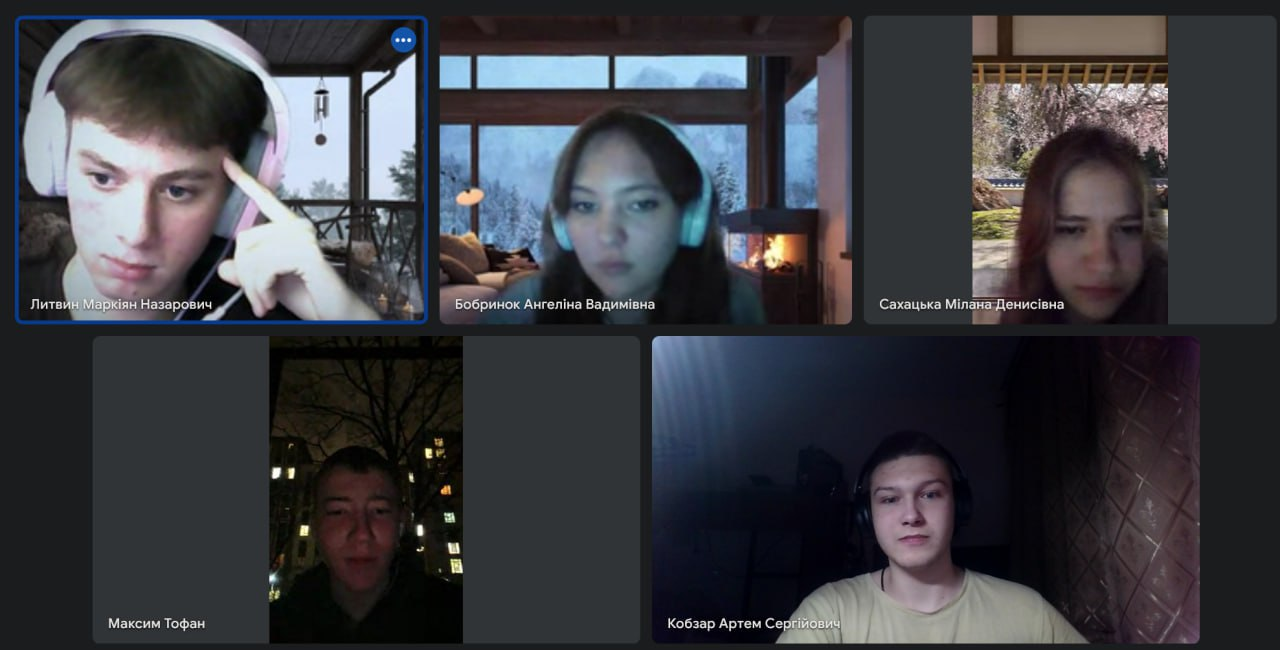
**Завдання №1**

1. *Конфігурація середовища до виконання завдань:*

Планування роботи в notion



Зустріч з командою та обговорення питань

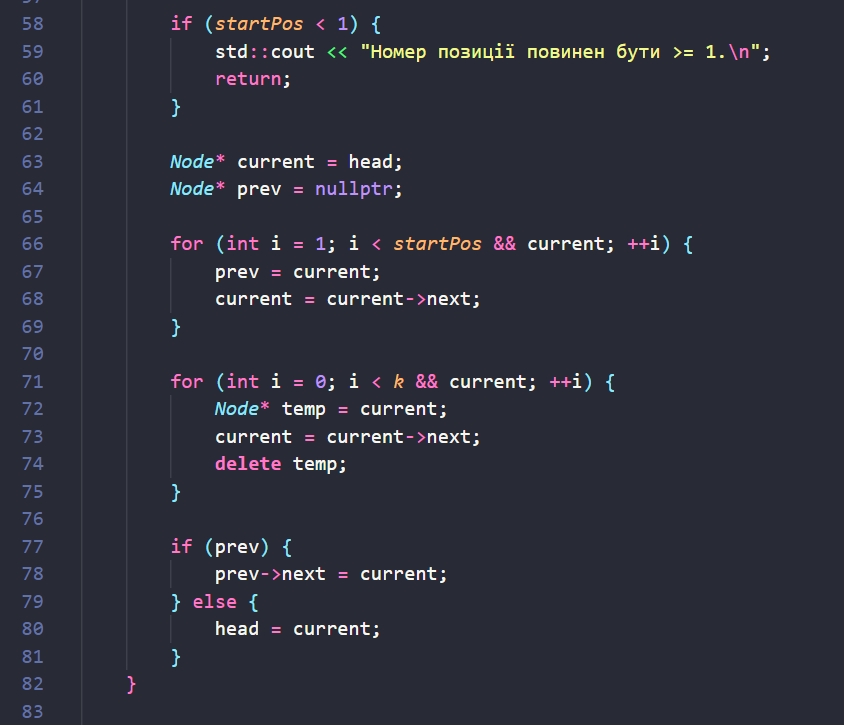


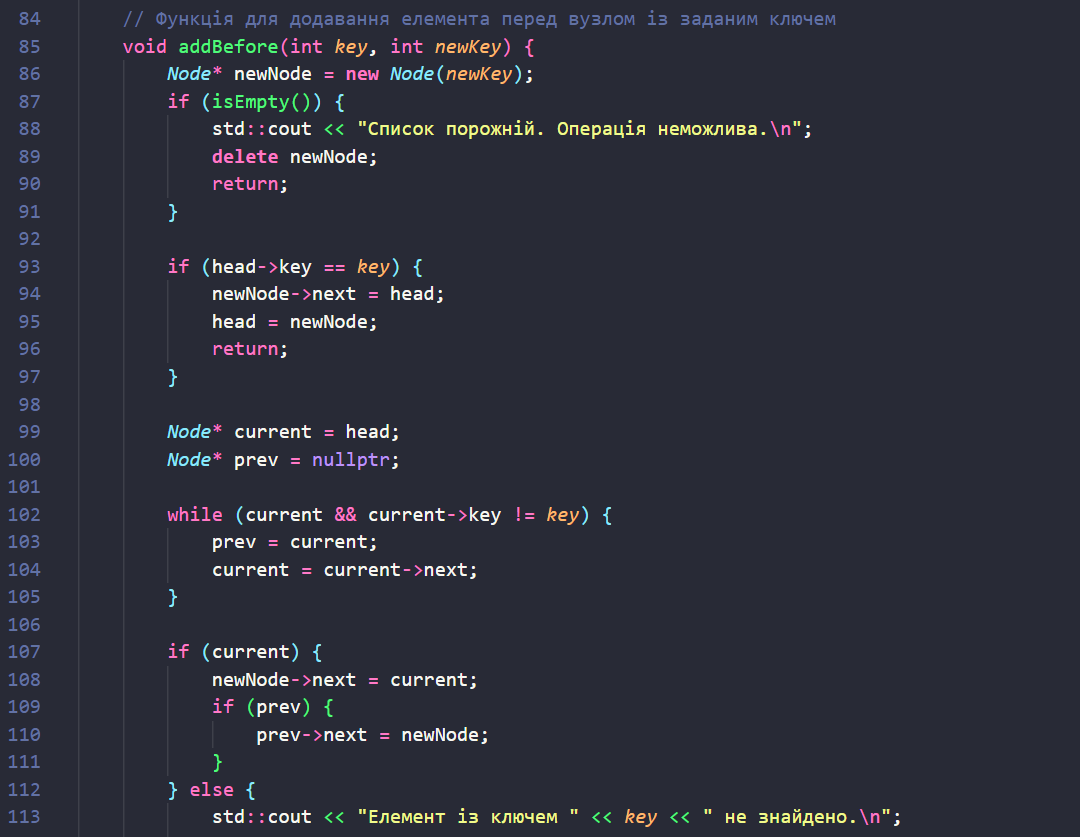
1. *Код програм з посиланням на зовнішні ресурси:*

**Завдання №1**

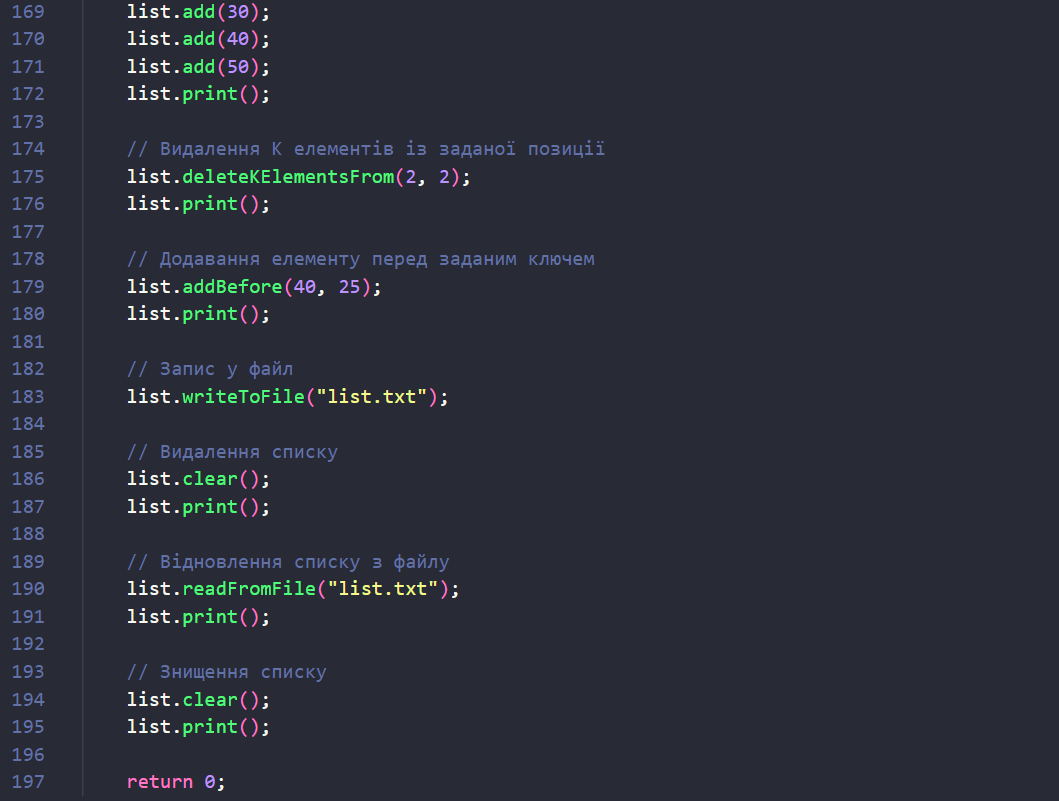
vns\_lab\_10\_anhelina\_bobrynok.cpp 







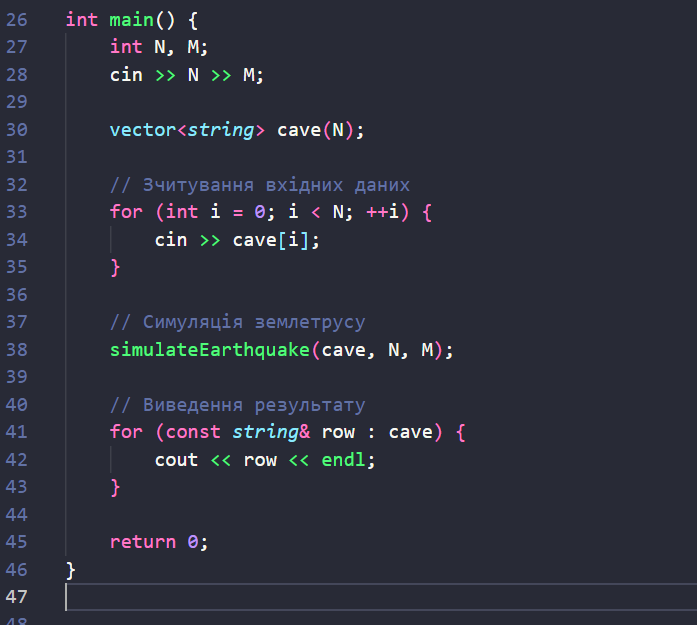




**Завдання №2**

algotester\_lab\_5\_anhelina\_bobrynok .cpp

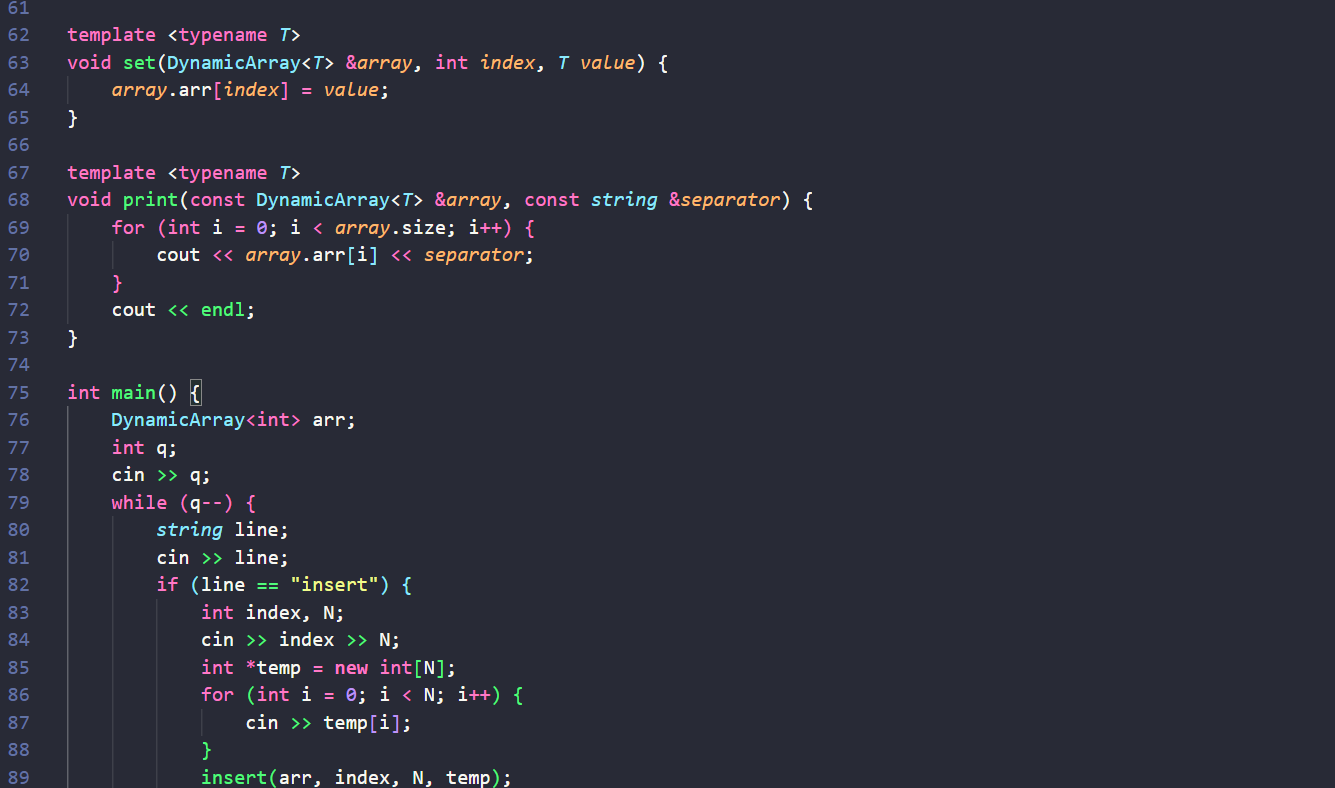




**Завдання №3**

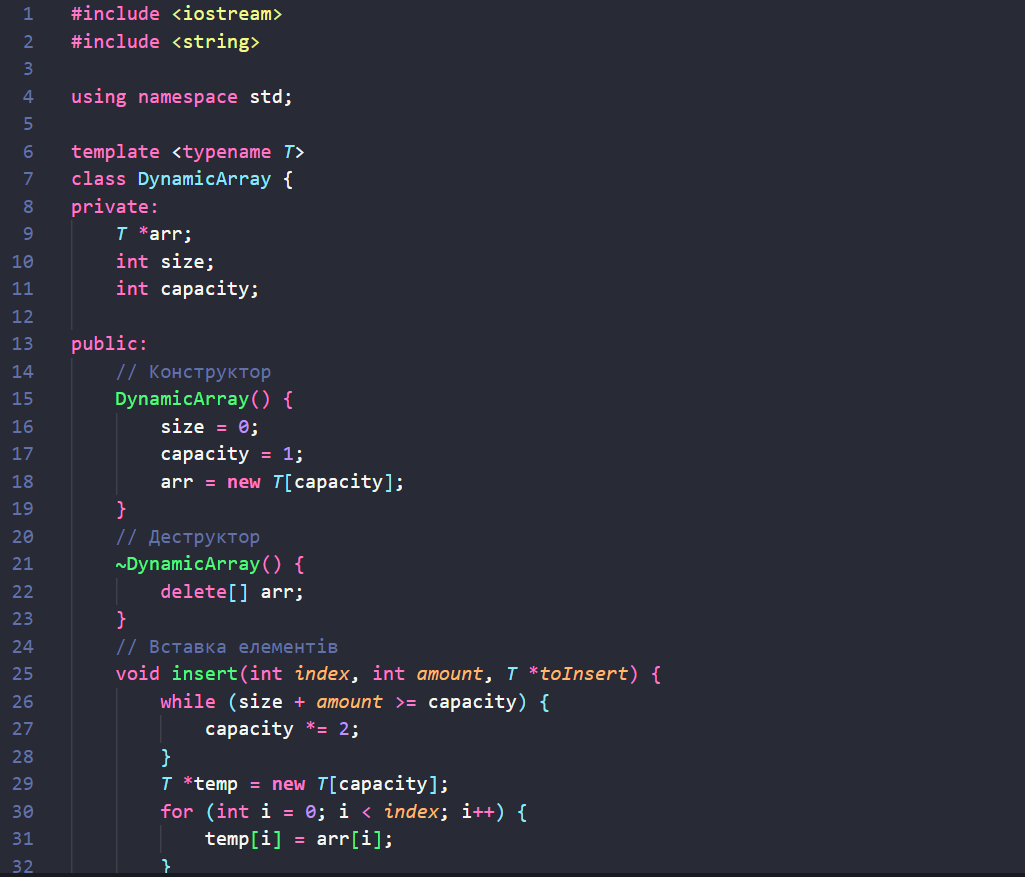
algotester\_lab\_78\_task\_1\_anhelina\_bobrynok 



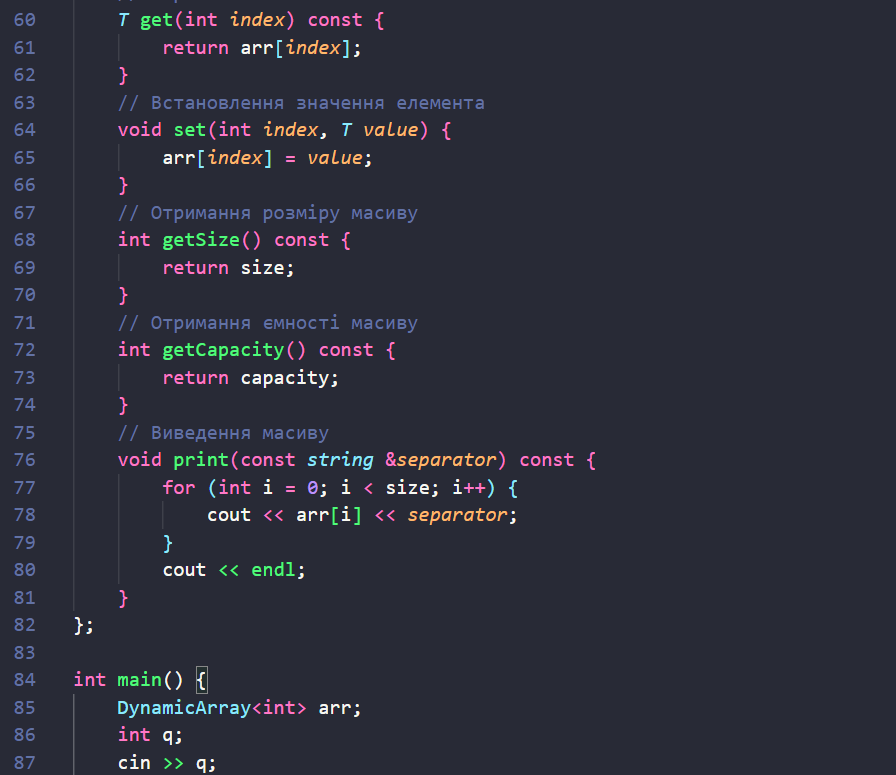


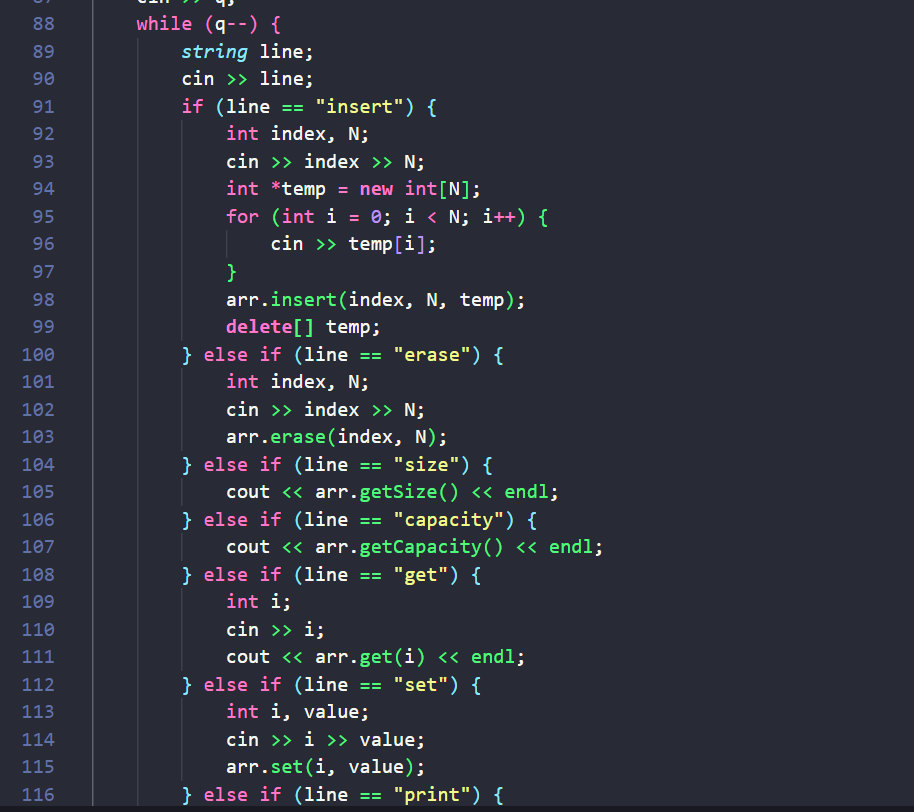


algotester\_lab\_78\_task\_2\_anhelina\_bobrynok



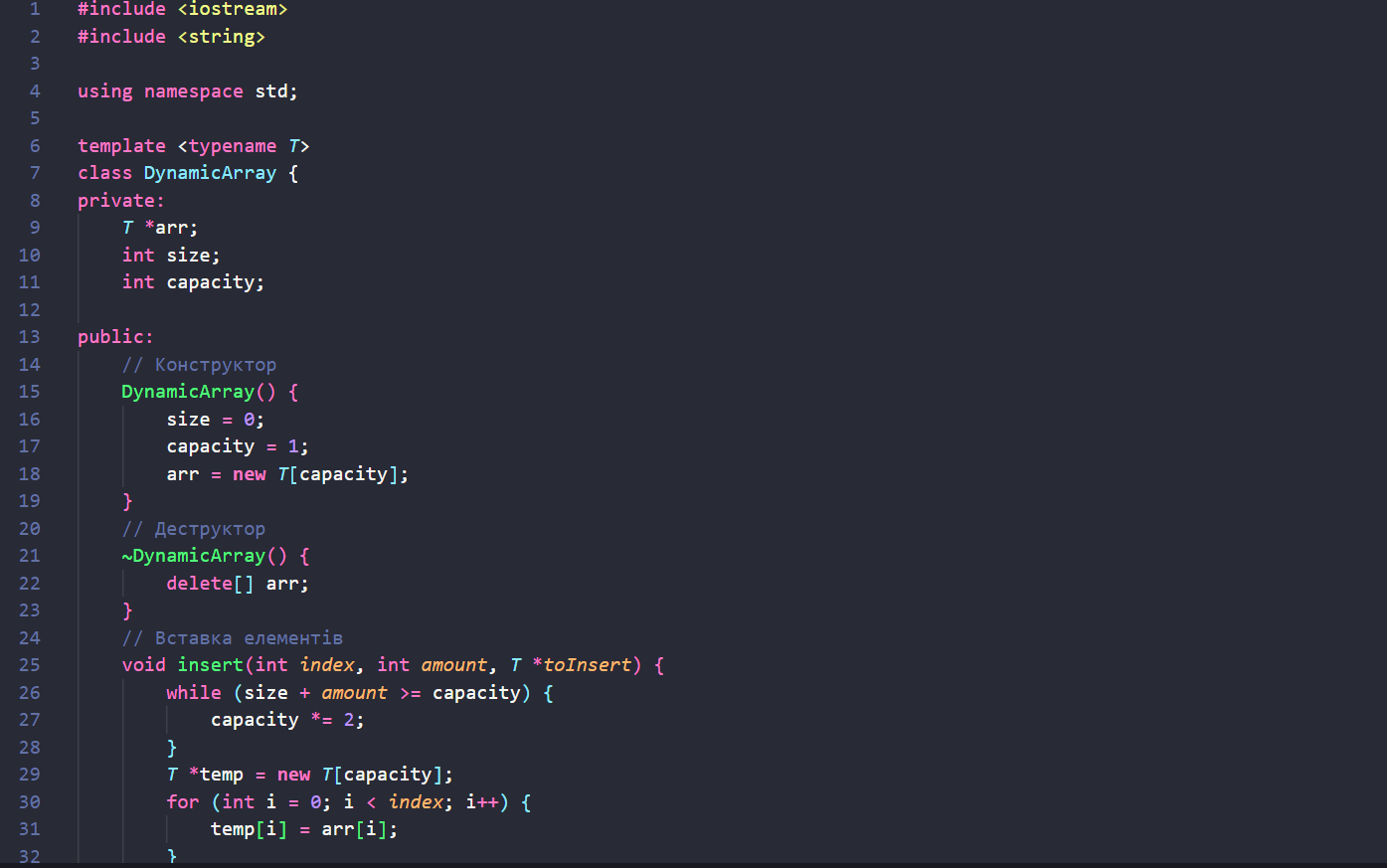


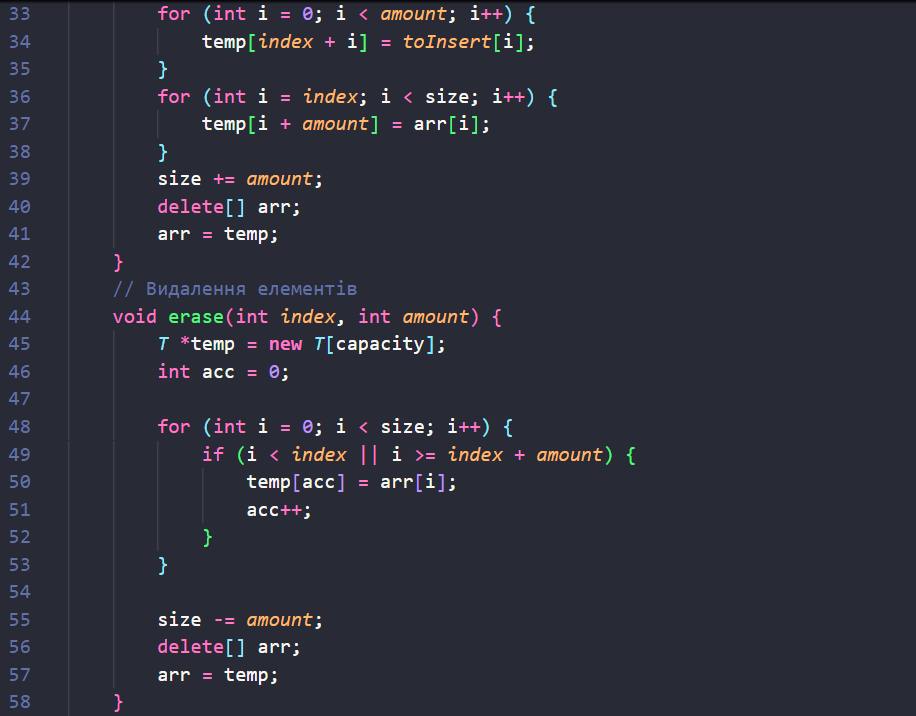


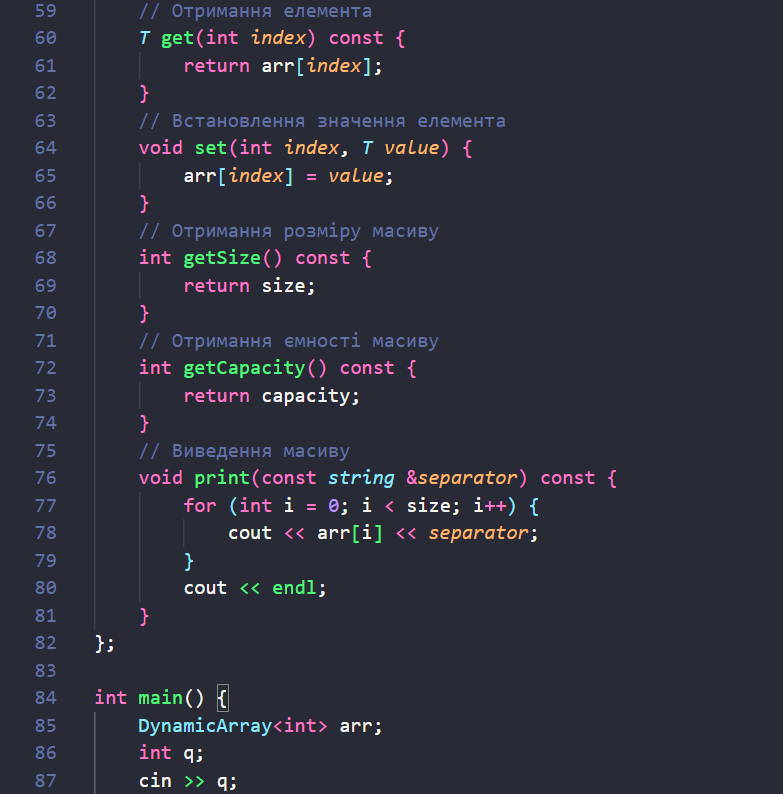


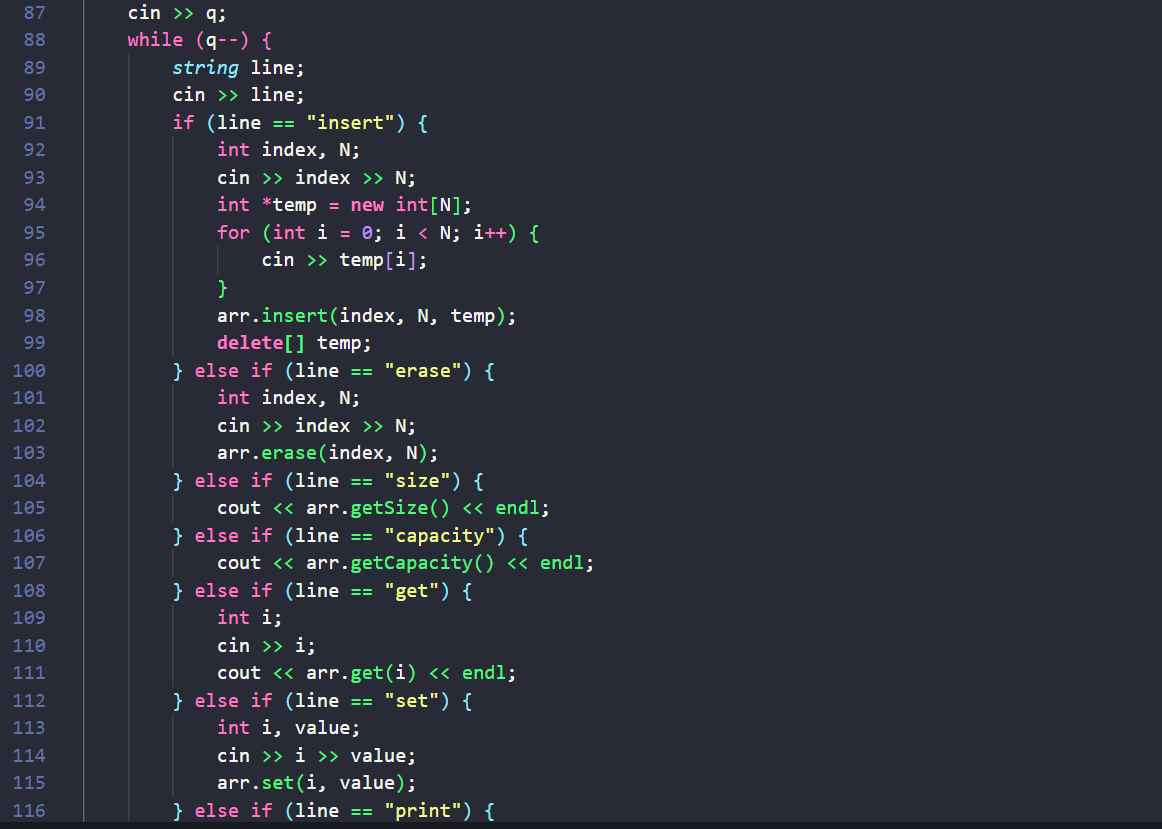
**Завдання №4**

practice\_work\_task\_1\_anhelina\_bobrynok.cpp







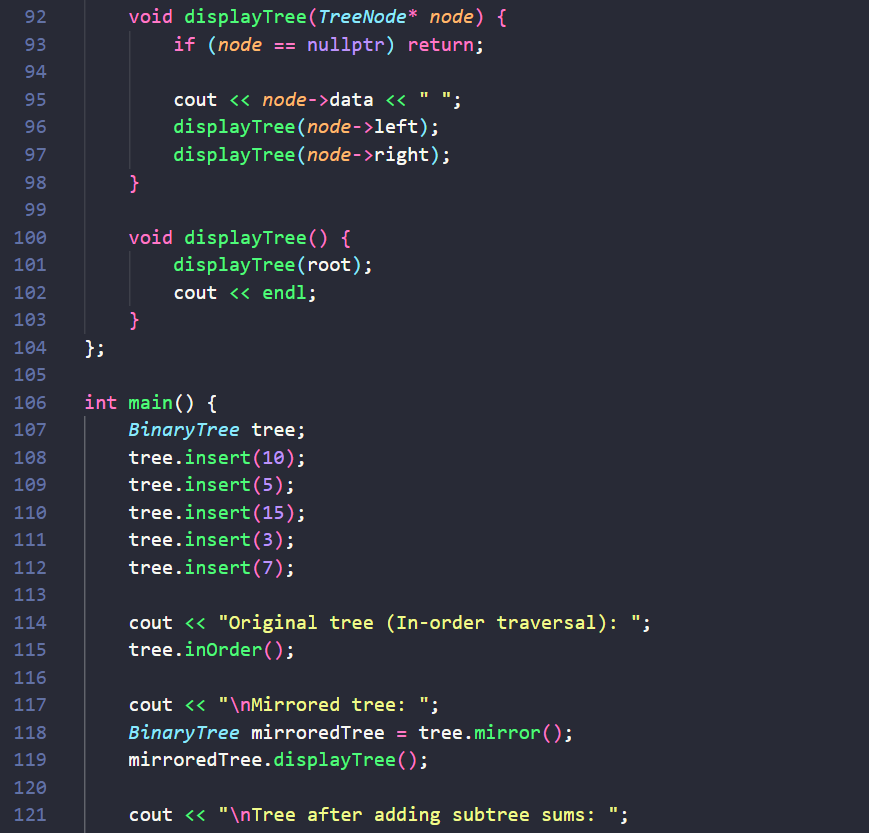


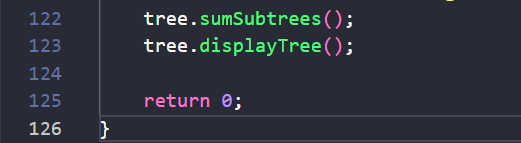
practice\_work\_task\_2\_anhelina\_bobrynok.cpp





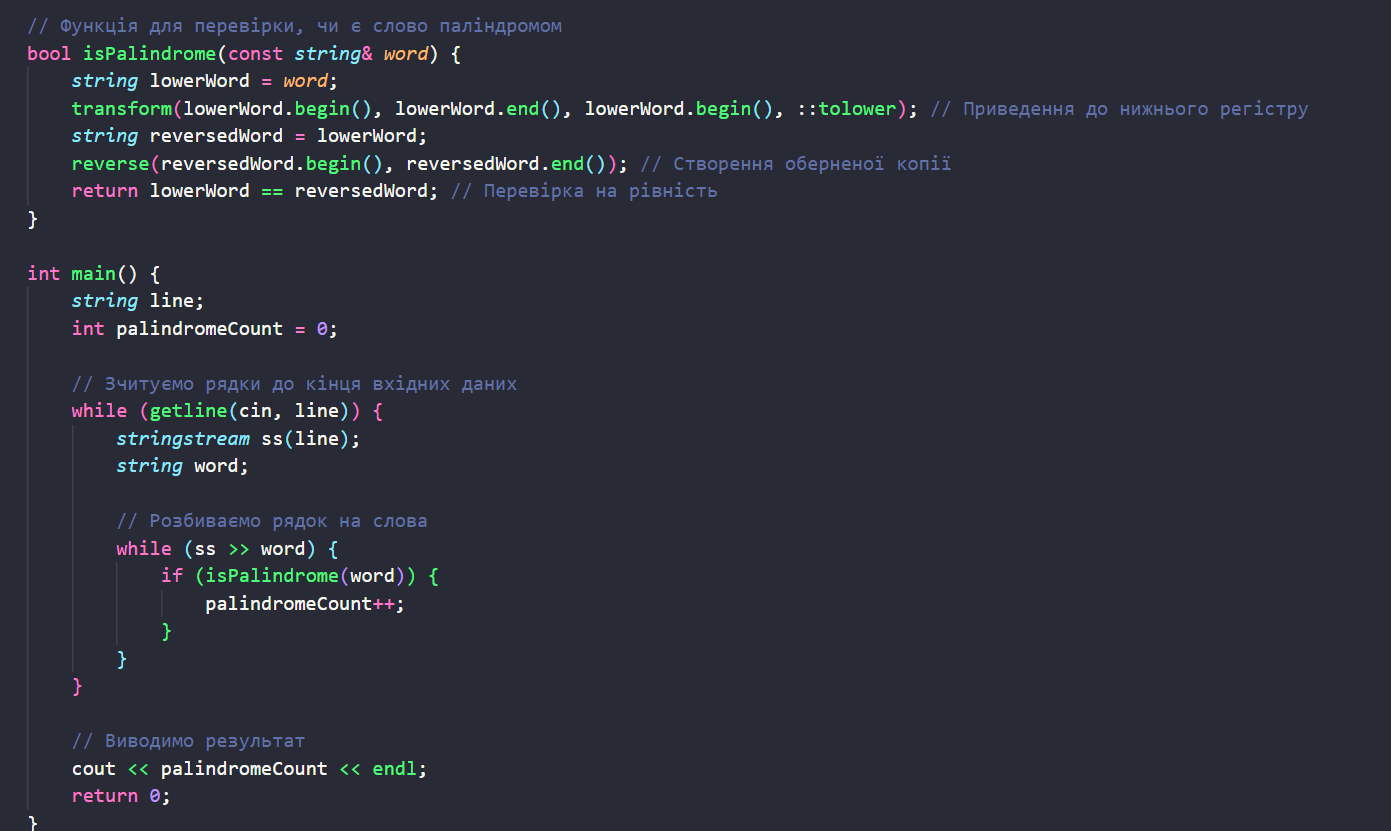






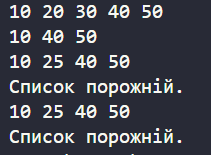
**Завдання №5**

self\_practice\_work\_anhelina\_bobrynok.cpp

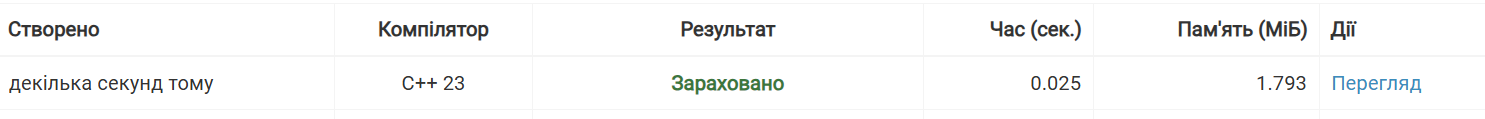


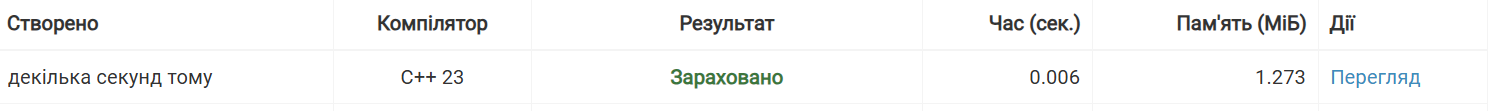
*5)Результати виконання завдань, тестування та фактично затрачений час:*

**Завдання №1**

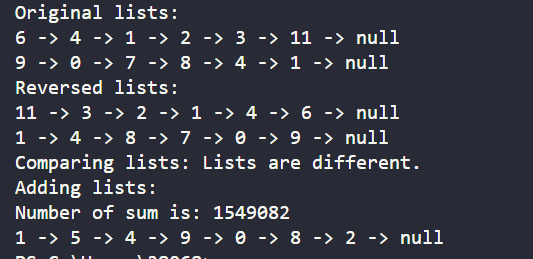


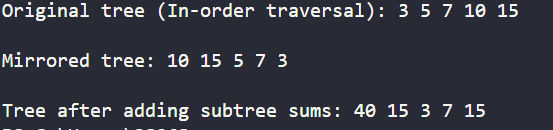
**Завдання №2**

**Завдання №3**

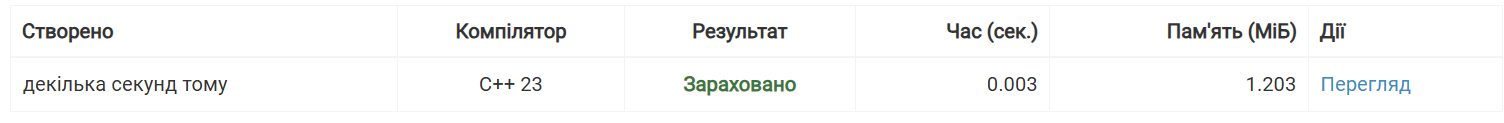


**Завдання №4**





**Завдання №5**



**Висновки:** Завдяки цій лабораторній роботі я мала змогу навчитись працювати з динамічними структурами, реалізувати зв’язний список та навчитись працювати з бінарними деревами.