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:**

Студент групи ШІ-11

Боднар Денис

**Тема роботи:** Вивчення основ динамічних структур даних у C++: стек, черга, зв'язні списки та дерева, а також алгоритмів обробки цих структур, включаючи операції додавання, видалення елементів та пошук. Розгляд основних принципів виділення пам'яті, переповнення стеку, обробки подій через чергу, а також обробки дерев різних типів (бінарних, AVL, червоно-чорних дерев) за допомогою ітеративних та рекурсивних алгоритмів.

**Мета роботи:** Ознайомитися з основними динамічними структурами даних у C++, зокрема стеком, чергою, зв'язними списками та деревами, вивчити їх властивості та операції (push, pop, enqueue, dequeue тощо). Опанувати основні алгоритми пошуку, сортування, вставки та видалення елементів у цих структурах. Зрозуміти принципи виділення пам'яті для динамічних структур та їх обробку, включаючи випадки переповнення та особливості обробки складних дерев.

**Теоретичні відомості:**

Теоретичні відомості з переліком важливих тем:

* Тема №1: Основи Динамічних Структур Даних.
* Тема №2: Стек.
* Тема №3: Черга.
* Тема №4: Зв'язні Списки.
* Тема №5: Дерева.

Індивідуальний план опрацювання теорії:

Тема №1: Основи Динамічних Структур Даних.

* + Джерела:

<https://acode.com.ua/urok-111-stek-i-kupa/#toc-1>

<https://www.youtube.com/watch?v=NyOjKd5Qruk>

* + - Що опрацьовано:
    - Вступ до динамічних структур даних: визначення та важливість
    - Виділення пам'яті для структур даних (stack і heap)
    - Приклади простих динамічних структур: динамічний масив
    - Статус: Ознайомлений

Тема №2: Стек.

* + Джерела:

<https://dystosvita.org.ua/mod/page/view.php?id=888>

<https://disted.edu.vn.ua/courses/learn/13472>

<https://uk.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%8C%D1%81%D1%8C%D0%BA%D0%B8%D0%B9_%D1%96%D0%BD%D0%B2%D0%B5%D1%80%D1%81%D0%BD%D0%B8%D0%B9_%D0%B7%D0%B0%D0%BF%D0%B8%D1%81>

* + - Що опрацьовано:
    - Визначення та властивості стеку
    - Операції push, pop, top: реалізація та використання
    - Приклади використання стеку: обернений польський запис
    - Переповнення стеку
    - Статус: Ознайомлений

Тема №3: Черга.

* + Джерела:

<https://dystosvita.org.ua/mod/page/view.php?id=889>

<https://www.kostrub.online/2020/07/struktura-danyx-cherha-Queue.>html

* + - Що опрацьовано:
    - Визначення та властивості черги
    - Операції enqueue, dequeue, front: реалізація та застосування
    - Приклади використання черги: обробка подій, алгоритми планування
    - Розширення функціоналу черги: пріоритетні черги

Статус: Ознайомлений

Тема №4: Зв'язні Списки.

* + Джерела:

<https://prometheus.org.ua/cs50/sections/section6.html>

* + - Що опрацьовано:
    - 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
    - Статус: Ознайомлений

Тема №5: Дерева.

* + Джерела:

<https://purecodecpp.com/uk/archives/2483>

<https://javarush.com/ua/groups/posts/uk.4165.chervono-chorne-derevo-vlastivost-principi-organzac-mekhanzmi-vstavki>

* + - 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
    - Статус: Ознайомлений

**Виконання роботи:**

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

**Завдання №1 - VNS Lab 10 - Task 1-13**

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.

**Порядок виконання роботи**

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.

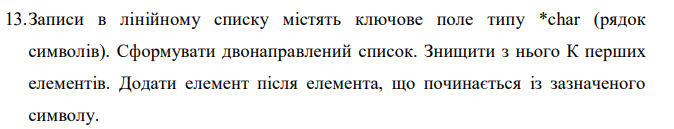
6. Написати функцію для знищення списку.

7. Записати список у файл, знищити його й виконати друк (при друці повинне

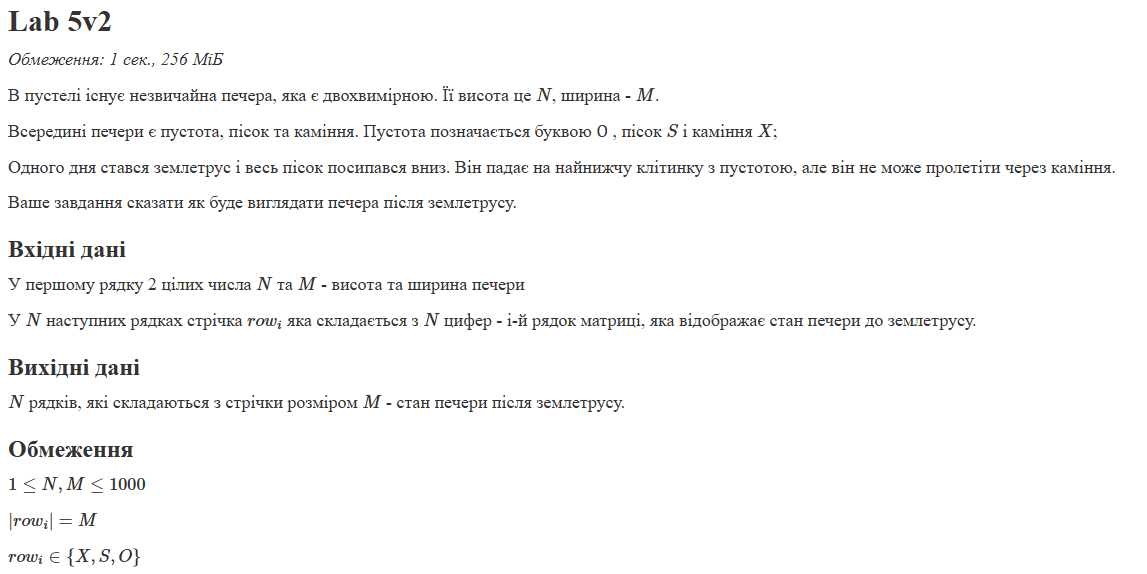
бути видане повідомлення "Список порожній").

8. Написати функцію для відновлення списку з файлу.

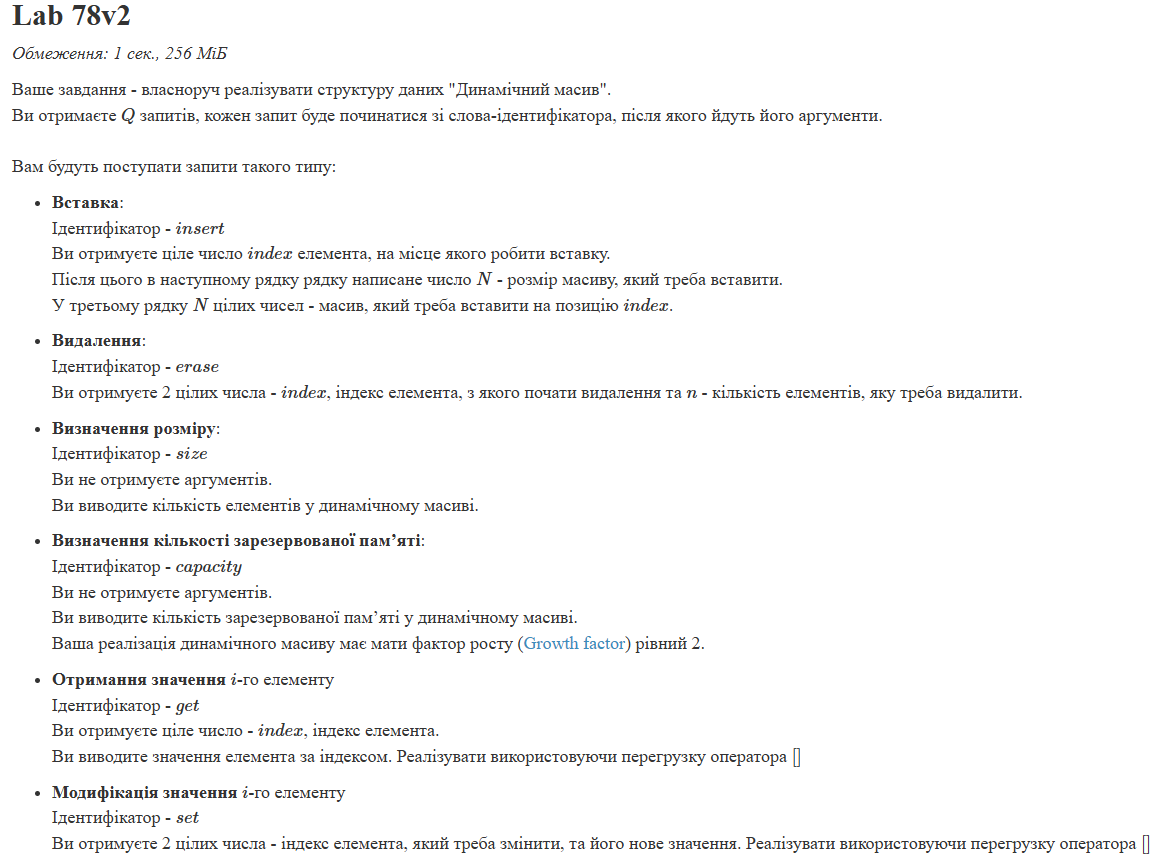
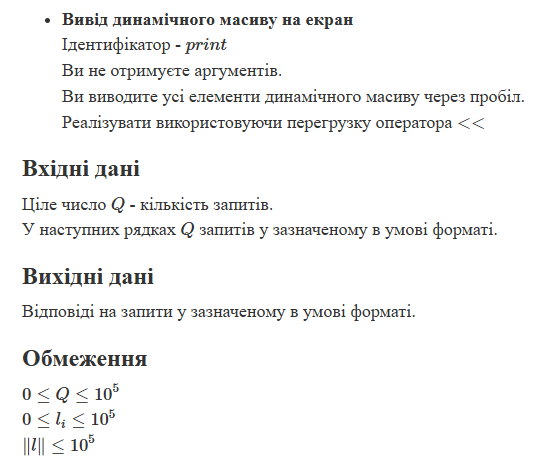
9. Відновити список і роздрукувати його.

10.Знищити список. 

**Завдання №2 - Algotester Lab 5v2**

****

**Завдання №3 - Algotester Lab 78v2**

**** ****

**Завдання №4 - Class Practice Work - Task 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.

**Завдання №5 - Class Practice Work - Task 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.

**Завдання №6 - Class Practice Work - Task 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.

**Завдання №7 - Class Practice Work - Task 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.

**Завдання №8 - Class Practice Work - Task 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

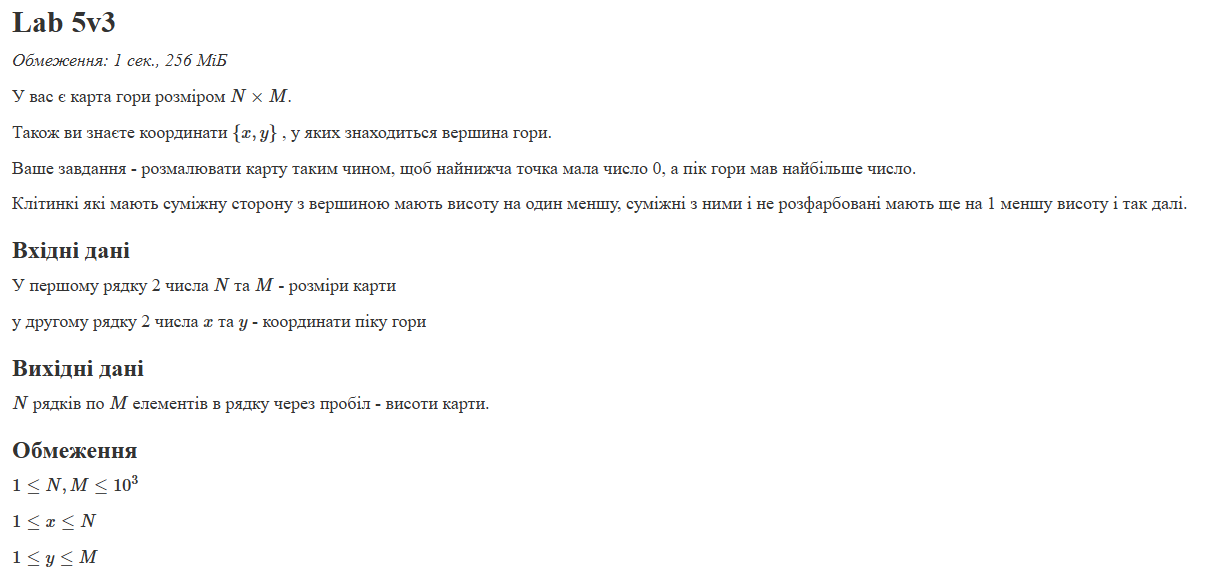
-       значення змінюються від листків до кореня дерева

**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.

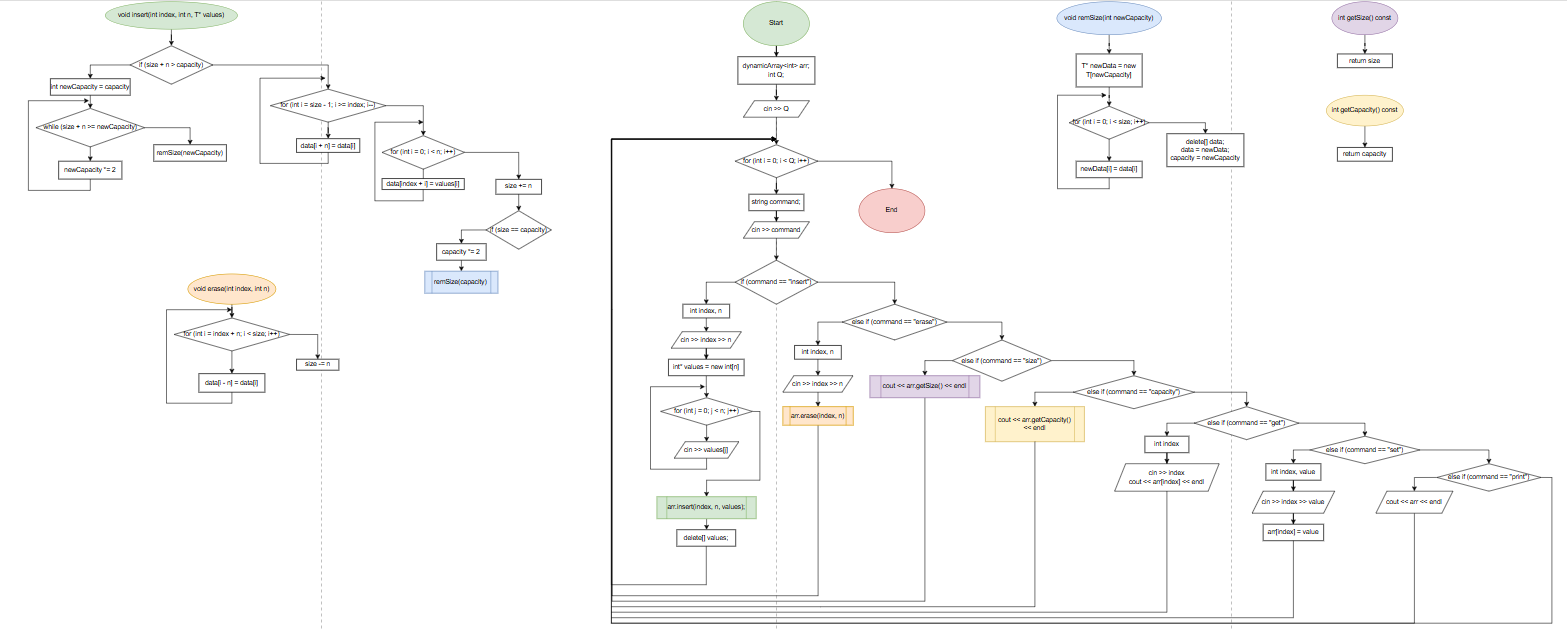
***Розвиток алгоритмічне мислення:*** Це завдання розвиває алгоритмічне мислення. Прохід всіх вузлів дерева демонструє розгортання рекурсивного виклику.

**Завдання №9 - Self Practice Work**



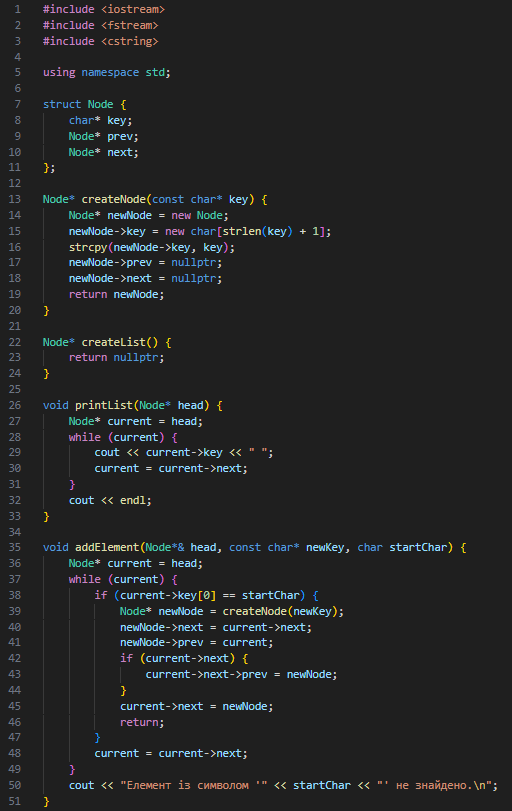
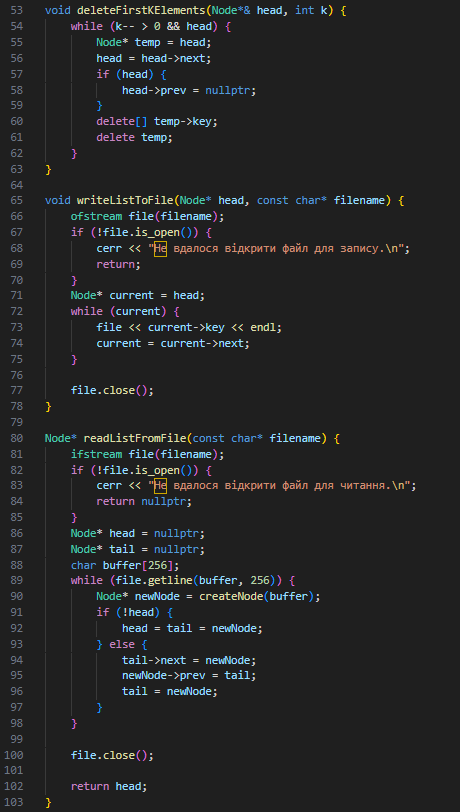
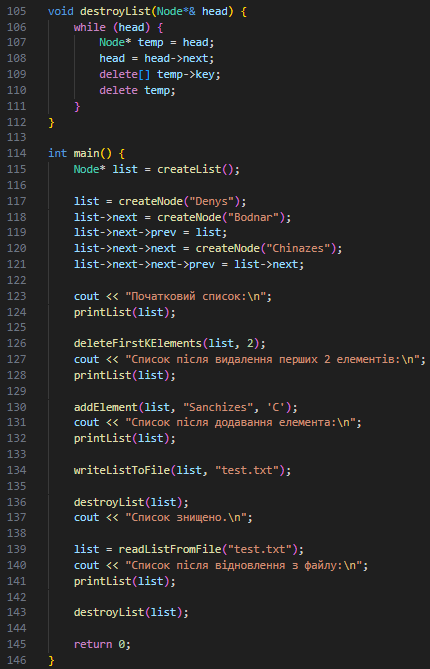
**Дизайн виконання завдань**

**Завдання №3 - Algotester Lab 78v2**

****

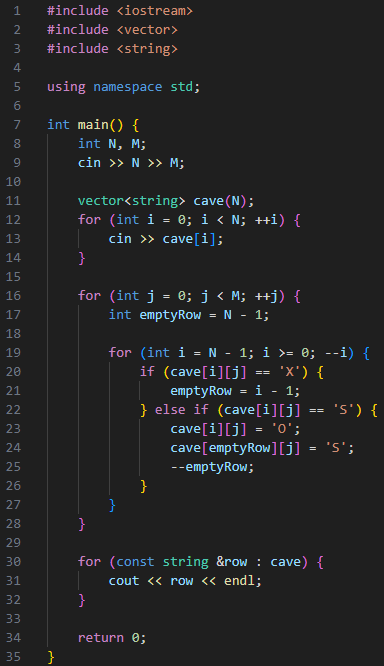
**Код програм та фактично затрачений час**

**Завдання №1 - VNS Lab 10 - Task 1-13**

**** **** 

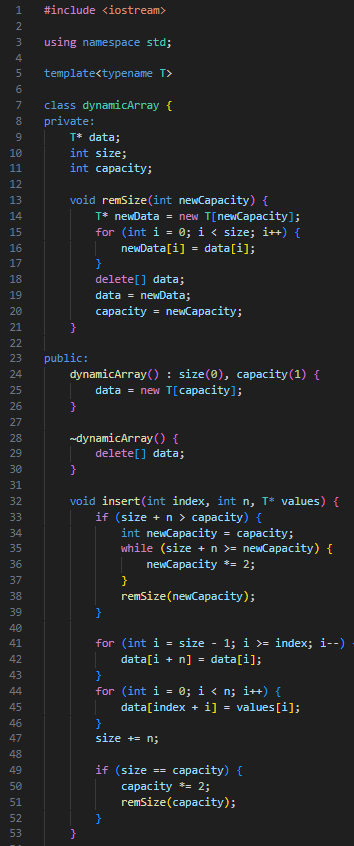
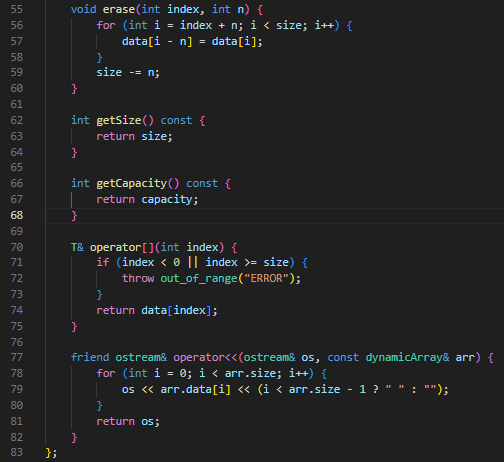
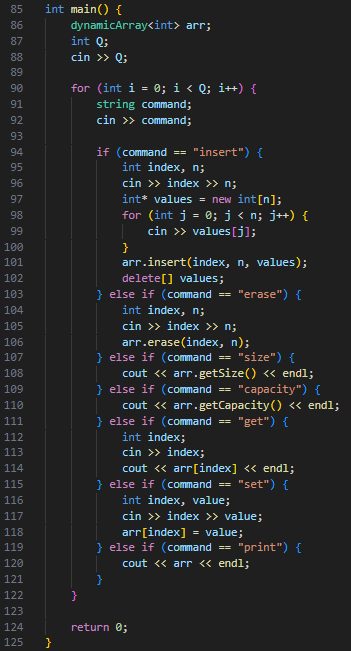
Фактично затрачений час: 4 год

**Завдання №2 - Algotester Lab 5v2**

****

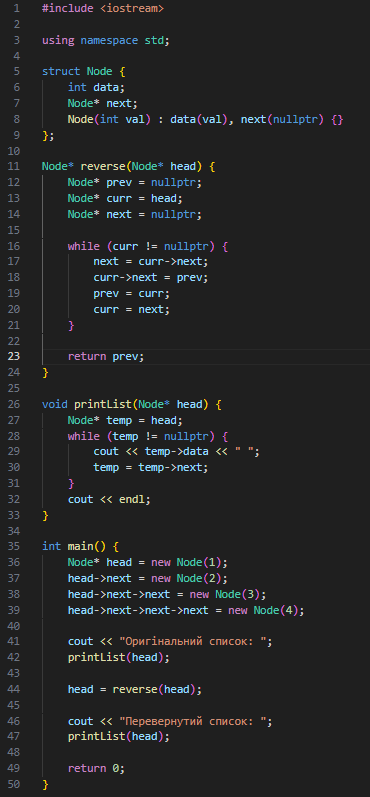
Фактично затрачений час: 2 год

**Завдання №3 - Algotester Lab 78v2**

**** **** 

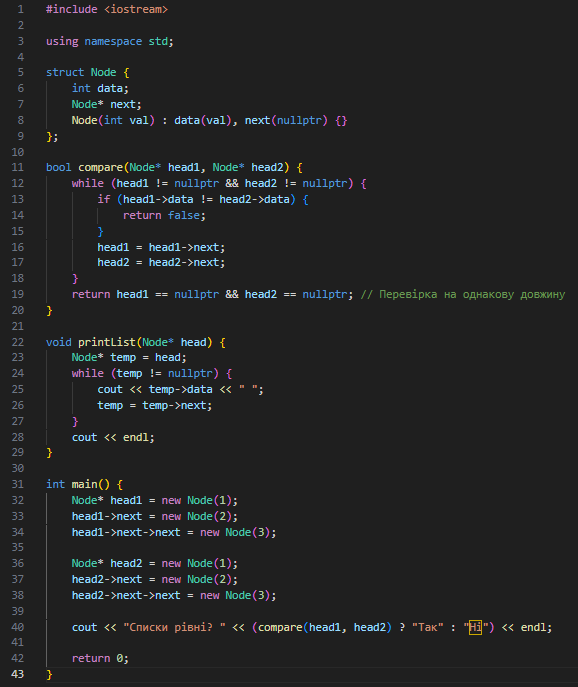
Фактично затрачений час: 3.5 год

**Завдання №4 - Class Practice Work - Task 1 - Реверс списку (Reverse list)**

****

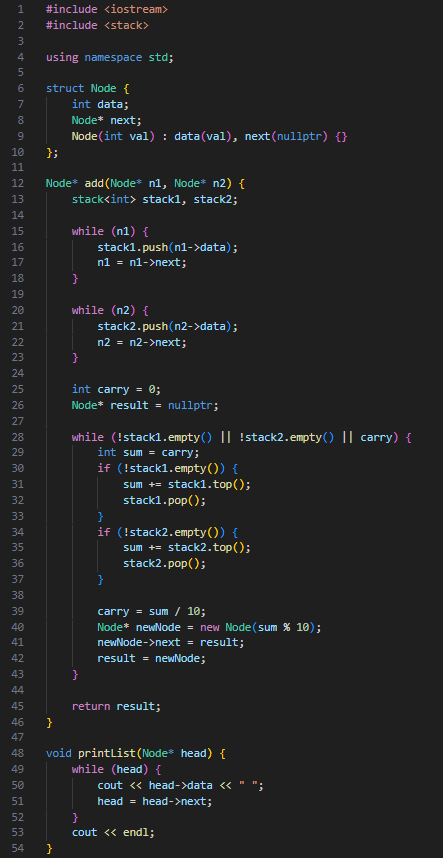
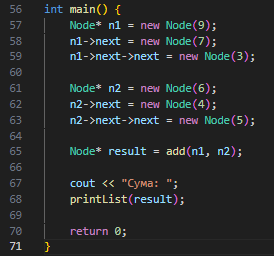
Фактично затрачений час: 2 год

**Завдання №5 - Class Practice Work - Task 2 - Порівняння списків**

****

Фактично затрачений час: 2 год

**Завдання №6 - Class Practice Work - Task 3 - Додавання великих чисел**

**** ****

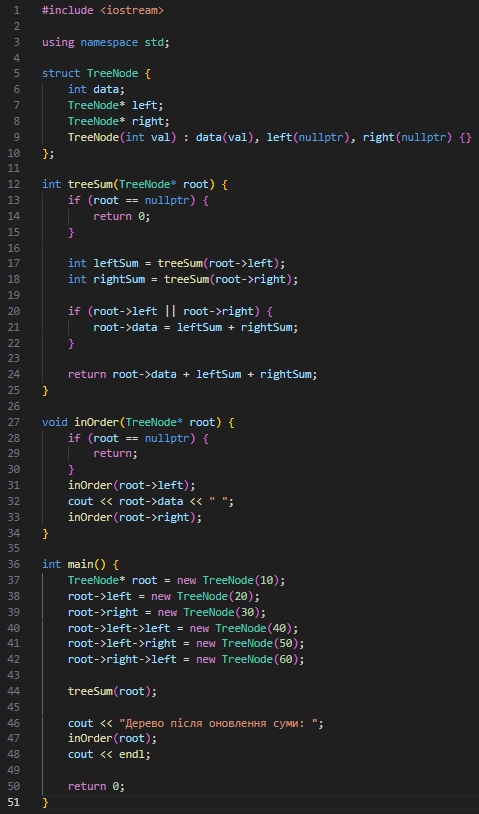
Фактично затрачений час: 3 год

**Завдання №7 - Class Practice Work - Task 4 - Віддзеркалення дерева**

****

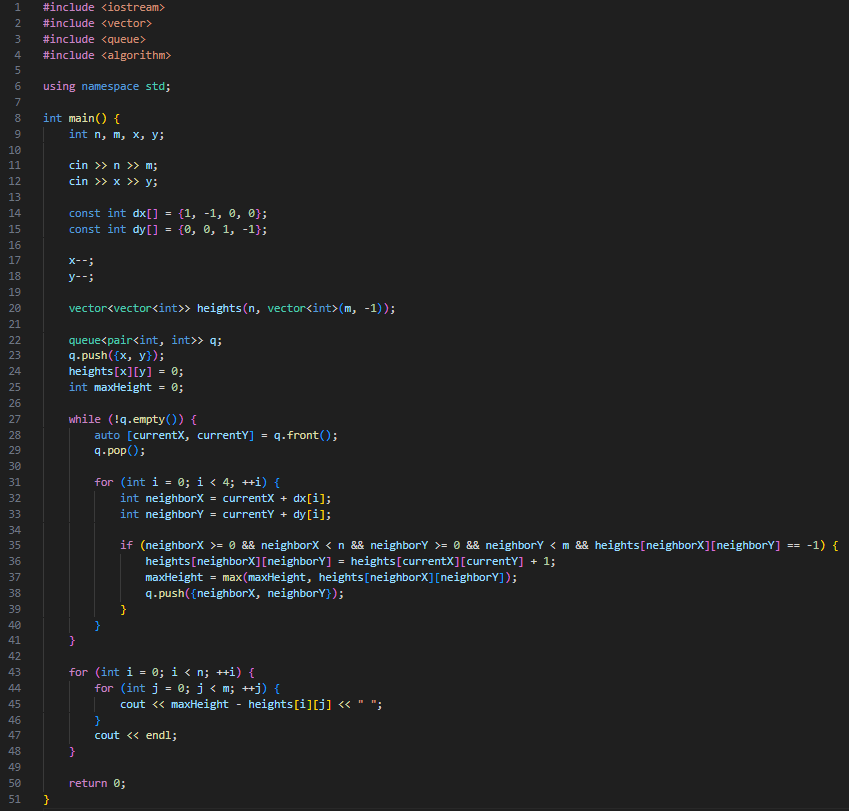
Фактично затрачений час: 4 год

**Завдання №8 - Class Practice Work - Task 5 - Записати кожному**

****

Фактично затрачений час: 4 год

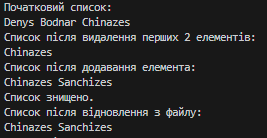
**Завдання №9 - Self Practice Work**

****

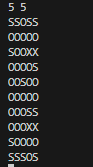
Фактично затрачений час: 3.5 год

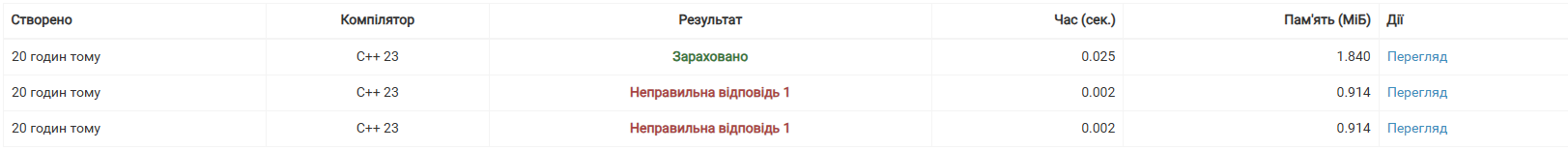
**Результати виконання завдань**

**Завдання №1 - VNS Lab 10 - Task 1-13**

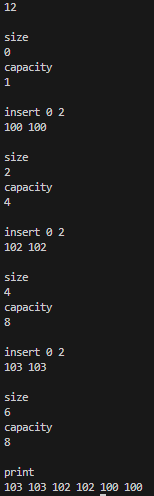
****

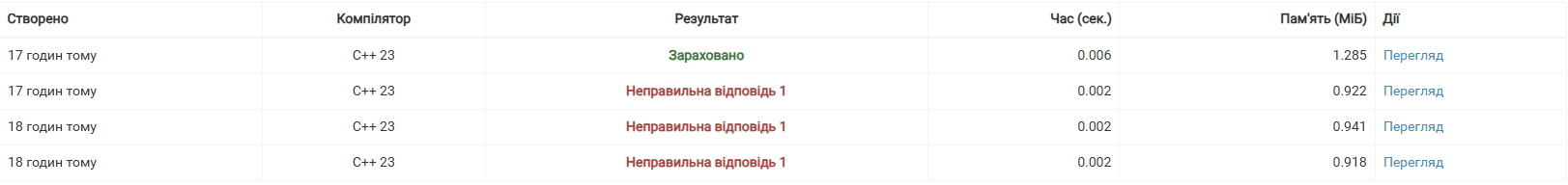
**Завдання №2 - Algotester Lab 5v2**

****

****

**Завдання №3 - Algotester Lab 78v2**

****

****

**Завдання №4 - Class Practice Work - Task 1 - Реверс списку (Reverse list)**

****

**Завдання №5 - Class Practice Work - Task 2 - Порівняння списків**

****

**Завдання №6 - Class Practice Work - Task 3 - Додавання великих чисел**

****

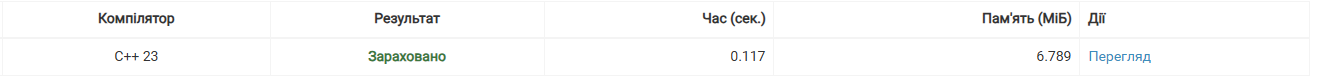
**Завдання №7 - Class Practice Work - Task 4 - Віддзеркалення дерева**

****

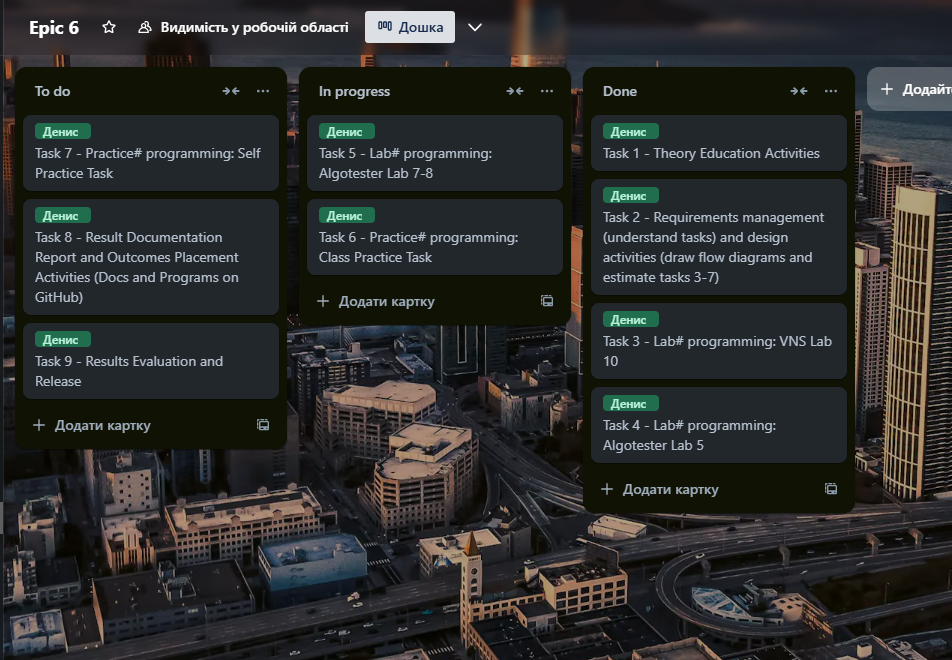
**Завдання №8 - Class Practice Work - Task 5 - Записати кожному батьківському вузлу суму підвузлів**



**Завдання №9 - Self Practice Work**

****

**Кооперація з командою**

****

**Висновок:**Виконуючи шостий епік, я ознайомився з основними принципами роботи з динамічними структурами даних у C++, зокрема стеком, чергою, зв'язними списками та деревами. Вивчив їх властивості та основні операції, такі як додавання і видалення елементів. Опанував алгоритми для маніпуляцій з цими структурами, включаючи ітеративні та рекурсивні підходи. Особливу увагу приділив принципам виділення пам'яті для динамічних структур, розумінню їх обробки, а також роботі з переповненням стеку і обробці подій через чергу. Дослідження складніших типів дерев, таких як бінарні, AVL та червоно-чорні, дозволило краще зрозуміти їх застосування та переваги в реальних задачах. Виконання цих завдань поглибило моє розуміння принципів роботи з динамічною пам'яттю та покращило навички використання алгоритмів для ефективної обробки даних.