Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту



**Звіт**

про виконання

**Лабораторних та практичних робіт №6**

***з дисципліни:*** «Мови та парадигми програмування»

***з розділу***: «Динамічні структури (Черга, Стек, Списки, Дерево). Алгоритми обробки динамічних структур»

***Виконав:***

студент групи ШІ-14

Зімнов Олександр Сергійович

# **Тема роботи:**

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

# **Мета роботи:**

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Динамічні структури
* Тема №2: Алгоритми обробки динамічних структур

1. Індивідуальний план опрацювання теорії:

* Тема №1: Динамічні структури
  + Джерела Інформації
    - Лекції
    - Лекційний матеріал на ВНС
  + Що опрацьовано:
    - Черга
    - Стек
    - Списки
    - Дерево
  + Статус: Ознайомлений
  + Початок опрацювання теми: 30.11.2023
  + Звершення опрацювання теми: 01.12.2023
* Тема №2: Алгоритми обробки динамічних структур
  + Джерела Інформації:
    - Лекції
    - Лекційний матеріал на ВНС
  + Що опрацьовано:
    - Алгоритми для вставки, вилучення, пошуку та інших операцій
    - Linked list
  + Статус: Ознайомлений
  + Початок опрацювання теми: 01.12.2023
  + Звершення опрацювання теми: 02.12.2023

# **Виконання роботи:**

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

Завдання №1 - Theory Education Activities

* Вивчення теорії

Завдання №2 - Requirements management (understand tasks) and design activities (draw flow diagrams and estimate tasks 3-7)

* Створення діаграм для програм в draw io

Завдання №3 - Lab# programming: VNS Lab 10

* Варіант № 4
* Деталі завдання: Написати програму, у якій створюються динамічні структури й виконати їхню обробку у відповідності зі своїм варіантом. Для кожного варіанту розробити такі функції: 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. Варіант: Записи в лінійному списку містять ключове поле типу int. Сформувати однонаправлений список. Знищити з нього елемент із заданим номером, додати К елементів, починаючи із заданого номера
* Важливі деталі для врахування в імплементації програми: Функція може створювати порожній список, а потім додавати в нього елементи, після виведення функцій потрібно роздрукувати список, якщо при виведення список порожній, то потрібно вивести відповідне повідомлення

Завдання №4 - Lab# programming: Algotester Lab 5

* Варіант № 3
* Деталі завдання: У вас є карта гори розміром 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 найбільше число. Клітинкі які мають суміжну сторону з вершиною мають висоту на один меншу, суміжні з ними і не розфарбовані мають ще на 1 меншу висоту і так далі.
* Важливі деталі для врахування в імплементації програми: 1≤N,M≤103, 1≤x≤N, 1≤y≤M, спочатку варто знайти число найвищої точки, після цього згенерувати гору буде простіше

Завдання №5 - Lab# programming: Algotester Lab 7-8

* Варіант № 2
* Деталі завдання: Ваше завдання - власноруч реалізувати структуру даних "Динамічний масив".  
  Ви отримаєте Q запитів, кожен запит буде починатися зі слова-ідентифікатора, після якого йдуть його аргументи. Вам будуть поступати запити такого типу: **Вставка**: Ідентифікатор – insert. Ви отримуєте ціле число index елемента, на місце якого робити вставку. Після цього в наступному рядку рядку написане число N - розмір масиву, який треба вставити. У третьому рядку N цілих чисел - масив, який треба вставити на позицію index. **Видалення**: Ідентифікатор – erase. Ви отримуєте 2 цілих числа - index, індекс елемента, з якого почати видалення та n - кількість елементів, яку треба видалити. **Визначення розміру**: Ідентифікатор – size. Ви не отримуєте аргументів. Ви виводите кількість елементів у динамічному масиві. **Визначення кількості зарезервованої пам’яті**: Ідентифікатор – capacity. 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** i-го елементу: Ідентифікатор – get. Ви отримуєте ціле число - index, індекс елемента. Ви виводите значення елемента за індексом. Реалізувати використовуючи перегрузку оператора [][]. **Модифікація значення** i-го елементу: Ідентифікатор – set. Ви отримуєте 2 цілих числа - індекс елемента, який треба змінити, та його нове значення. Реалізувати використовуючи перегрузку оператора [][]. **Вивід динамічного масиву на екран**: Ідентифікатор – print. Ви не отримуєте аргументів. Ви виводите усі елементи динамічного масиву через пробіл. Реалізувати використовуючи перегрузку оператора <<
* Важливі деталі для врахування в імплементації програми: 0≤Q≤105, 0≤li≤105, ∥l∥≤105. **Для отримання**100%**балів ця структура має бути написана як**[шаблон класу](https://cplusplus.com/doc/oldtutorial/templates/)**, у якості параметру використати**int

Завдання №6 - Practice# programming: Class Practice Task

* Деталі завдання №1: 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
* Деталі завдання №2: bool compare(Node \*h1, Node \*h2); 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: Потрібно зробити структуру Node для даних файлу, для кожної дії потрібно створити відповідну функцію

Завдання №7 - Practice# programming: Self Practice Task

* Деталі завдання: Створити структуру, для запам’ятовування інформації про персонажа, з них Ім’я, Швидкість, Втрачене здоров’я, Втрачені рівні заклинання, Втрачені побічні здібності персонажа, Кількість можливостей лікування. Вивести меню, в якому можна створити, видалити, або змінити параметра персонажу. Також програма має вміти вивести всіх персонажів впорядкованих за швидкістю. В кінці програму можна записати в файл. А при запуску програма має зчитувати ці данні
* Важливі деталі для врахування в імплементації програми: Рівнів заклинань є 5 і їх може бути різна кількість, користувач вводить всі “втрачені” параметри як додатні, достатньо просто помножити їх на -1. Індекс персонажа може змінюватись залежно від теперішньої позиції

Завдання №8 - Result Documentation Report and Outcomes Placement Activities (Docs and Programs on GitHub)

* Створення звіту і завантажити його з програмами на GitHub

Завдання №9 - Results Evaluation and Release

* Захист звіту.

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

Програма №1 VNS Lab 10

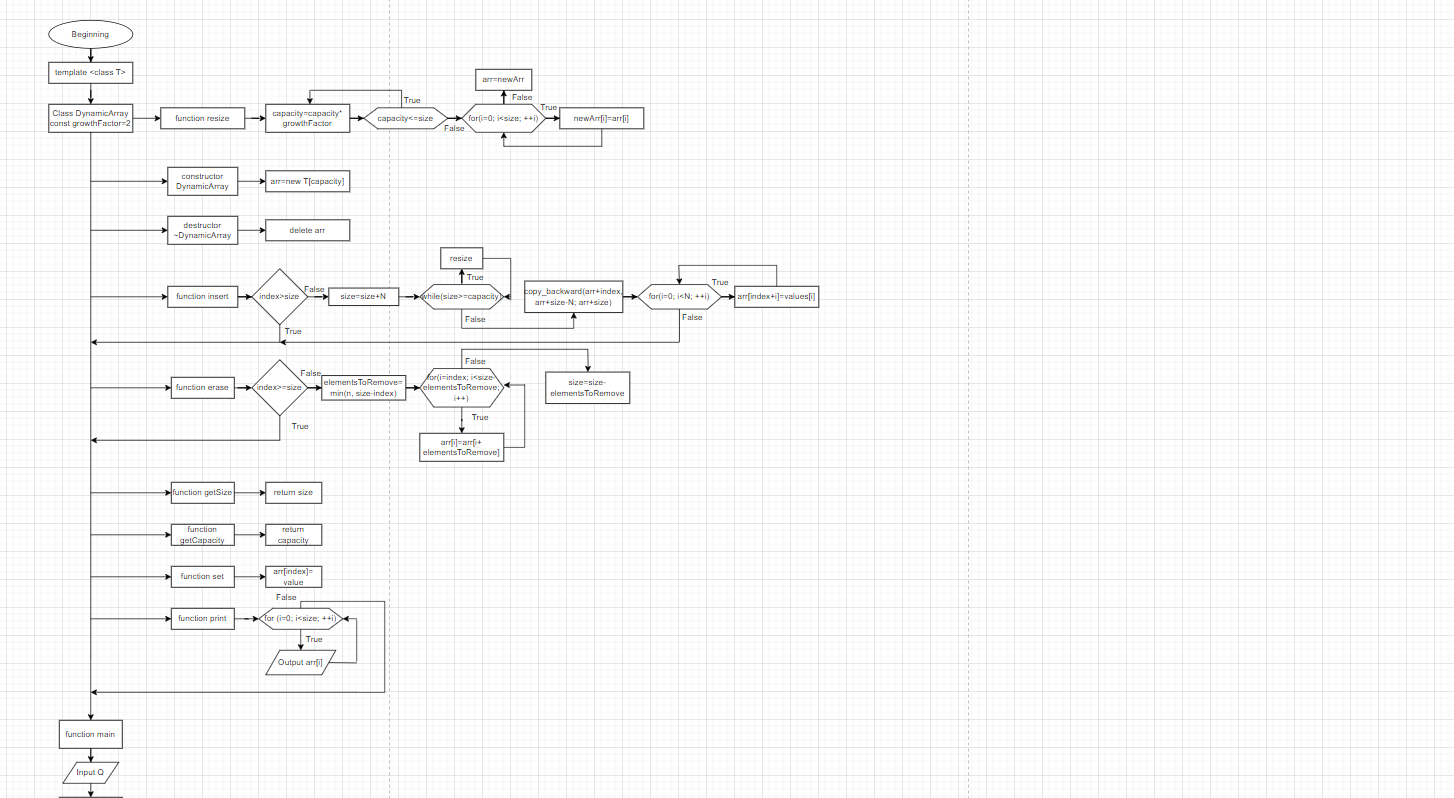
* Час на виконання: 3 години
* Важливі деталі для врахування в імплементації програми: Завдання можна виконати в одному файлі для того щоб використовувати спільні функції. Записи в списку містять поля типу int, що треба вказати в структурі

Програма №2 Algotester Lab 5

* Час на виконання: 1 година і 30 хвилин
* Важливі деталі для врахування в імплементації програми: Завдяки функції abs можна знайти максимальне значення гори, а потім завдяки цьому числу і масиву можна сформувати всі решта числа

Програма №3 Algotester Lab 7-8

* Блок схема:



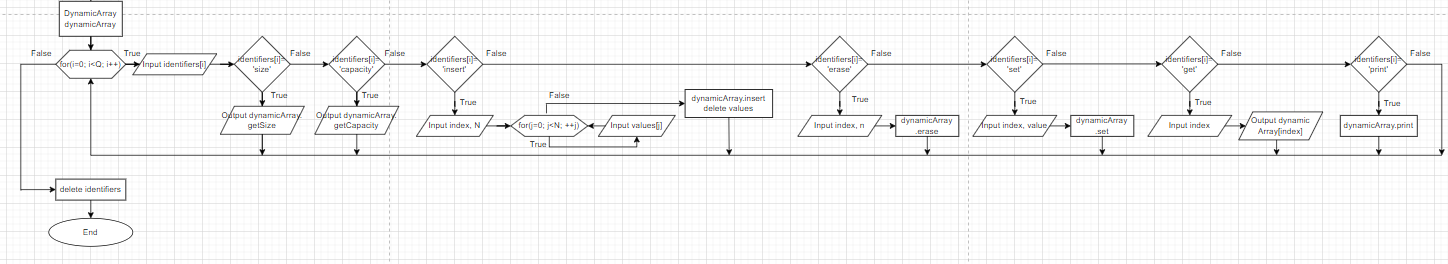


Рисунок 1: Блок схема до програми №3

* Час на виконання: 6 годин
* Важливі деталі для врахування в імплементації програми: capacity завжди має бути більше ніж size, якщо ж ні, то потрібно його домножити на growthFactor

Програма №4 Class Practice Task

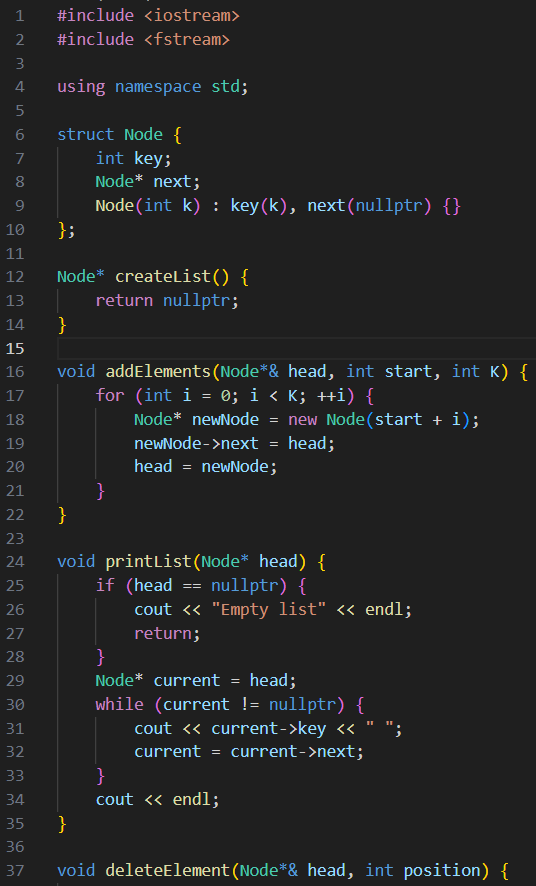
* Час на виконання: 2 години
* Важливі деталі для врахування в імплементації програми: Для зручності list був створений самою програмою, провірку можна робити до nullptr, що означає кінець списку.

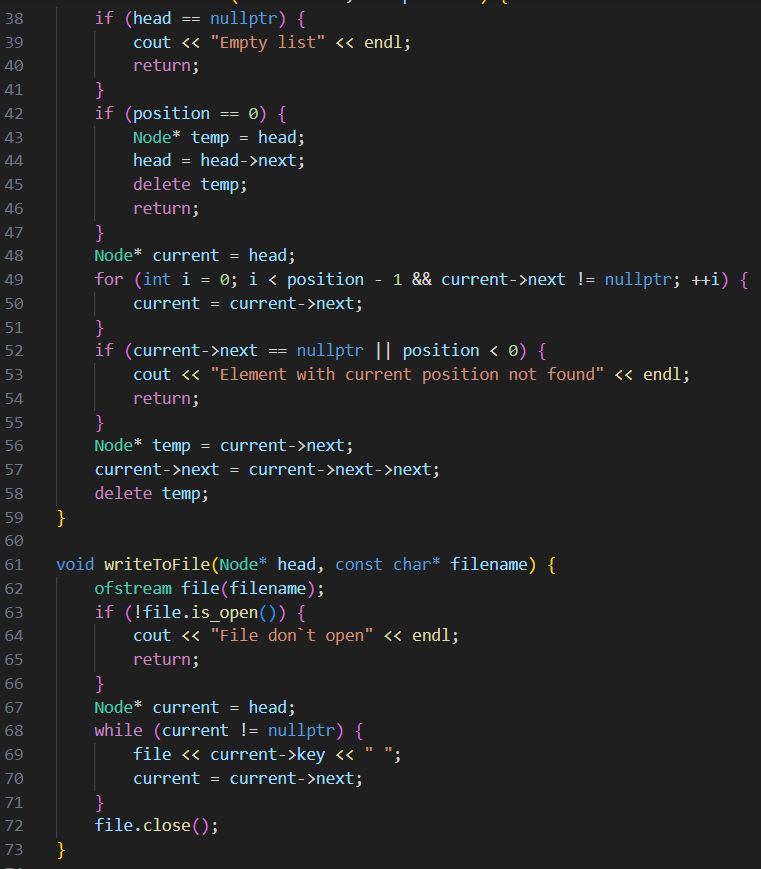
Програма №5 Self Practice Task

* Час на виконання: 3 години
* Важливі деталі для врахування в імплементації програми: Важливо зазначити, що для кожного рівня заклинання є відмінна кількість самих заклинань. Як максимум можна взяти 10 гравців, цього цілком достатньо для списку.

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

Завдання №1: VNS Lab 10





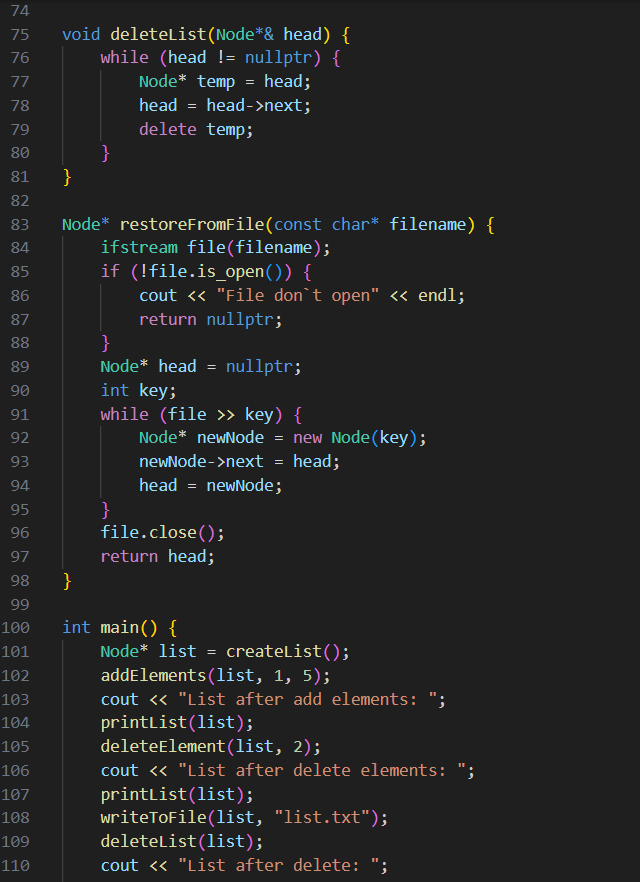
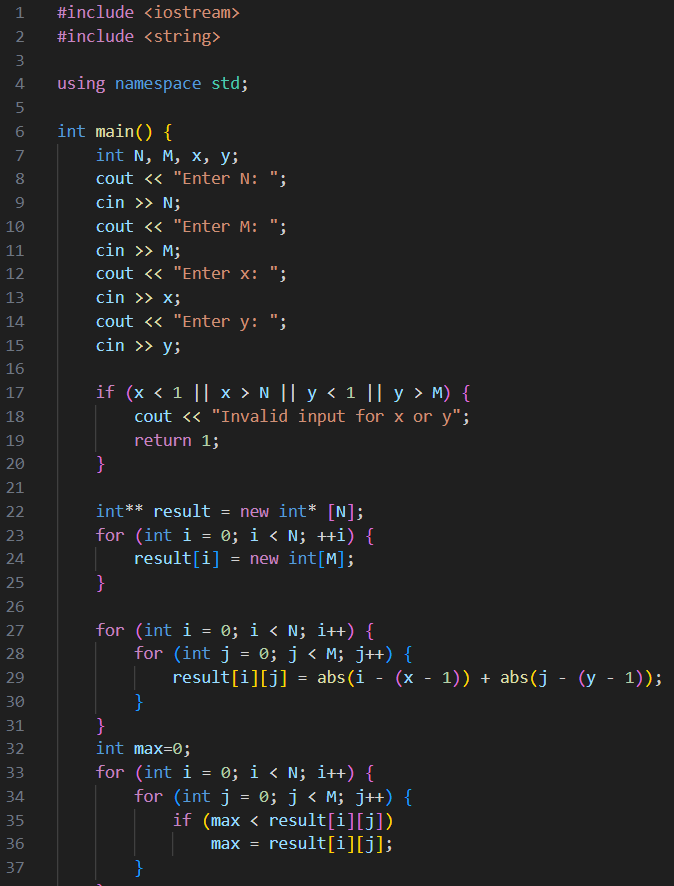




Рисунок 2: Код завдання з VNS №10

Завдання №2: Algotester Lab 5



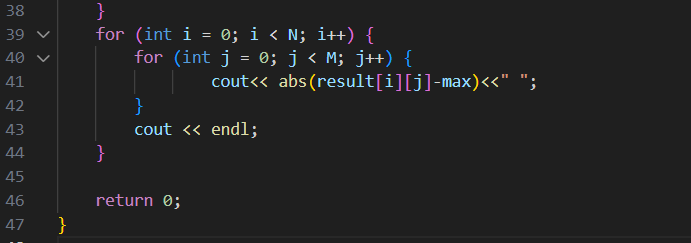
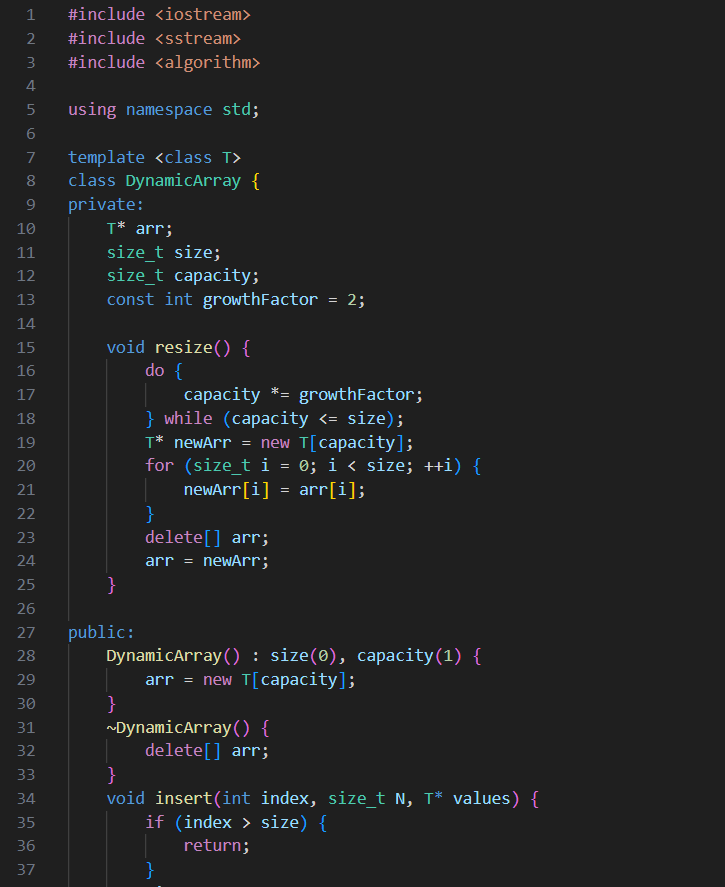
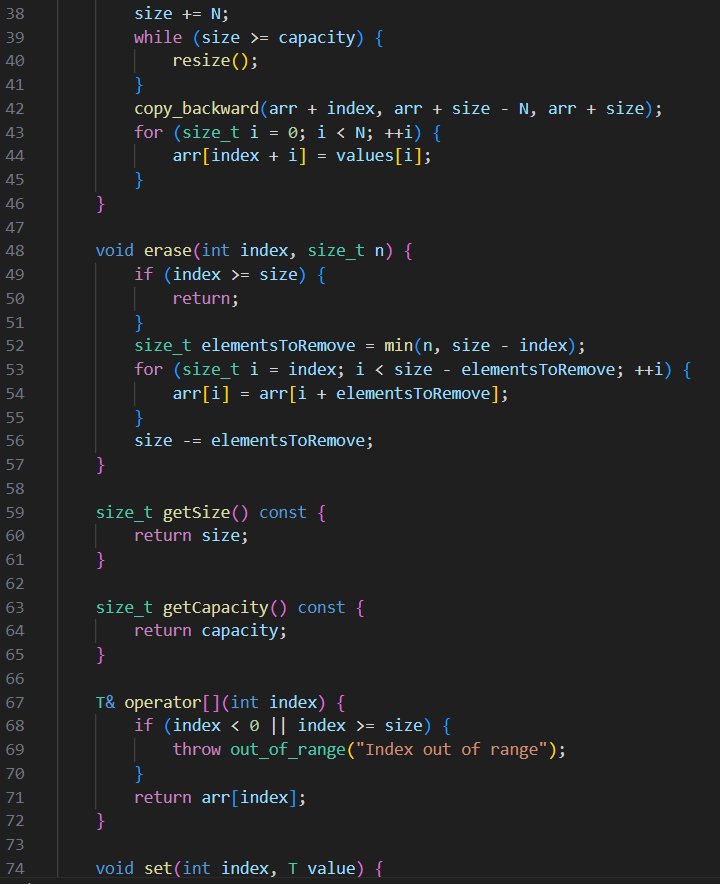
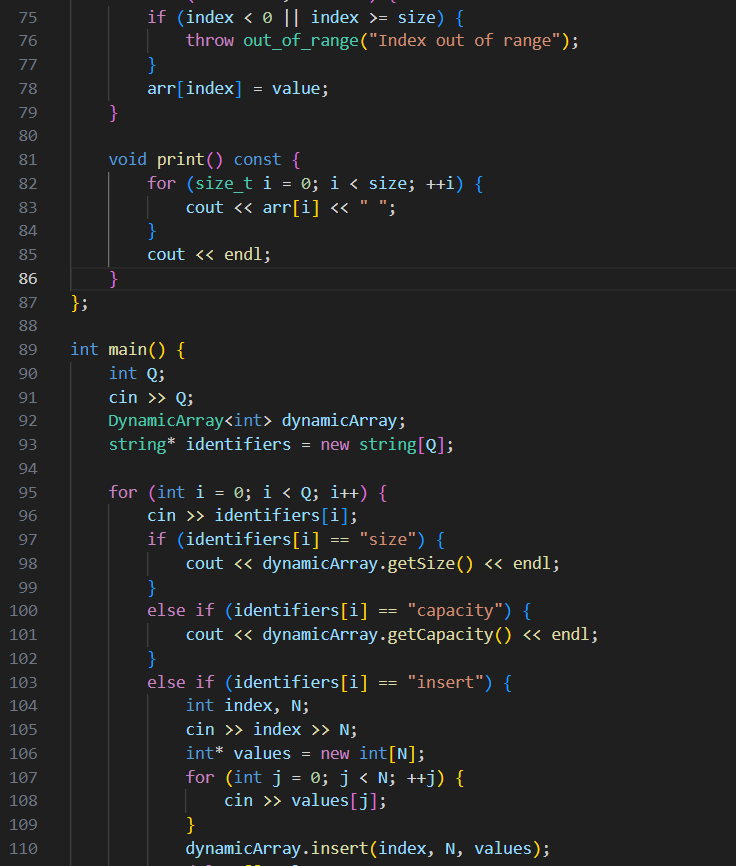


Рисунок 3: Код завдання з Algotester №5

Завдання №3: Algotester Lab 7-8







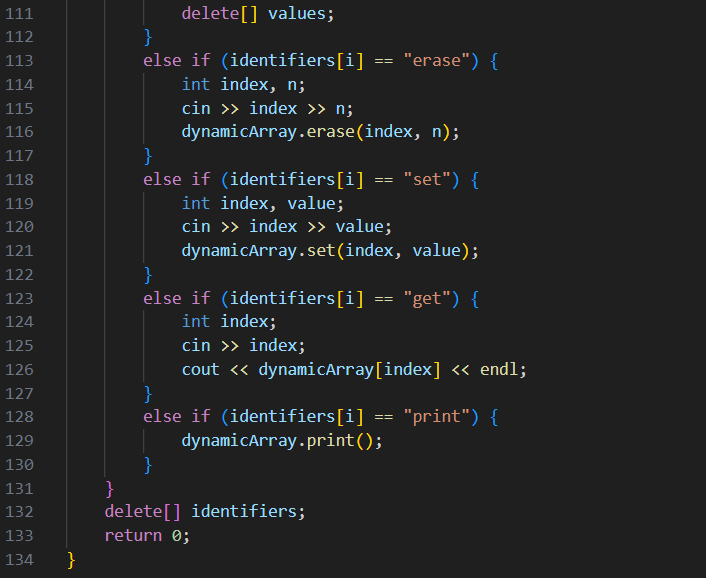
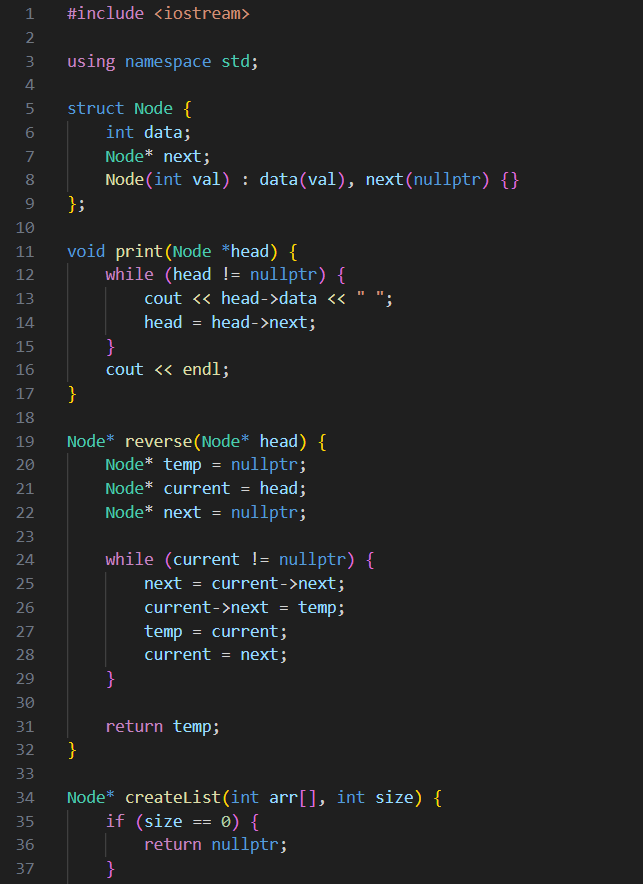
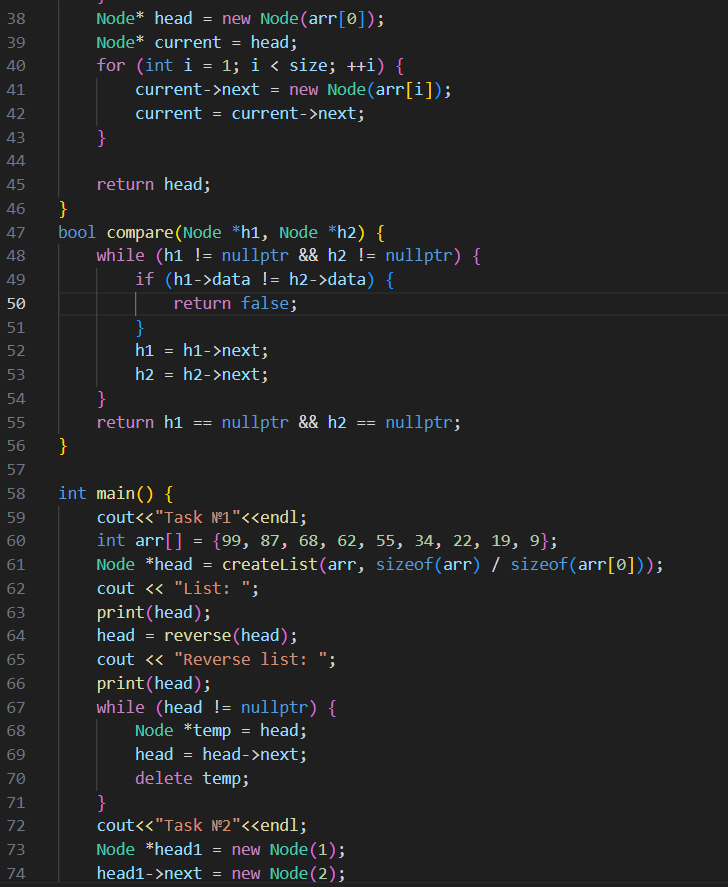


Рисунок 4: Код завдання з Algotester №7-8

Завдання №4: Class Practice Task





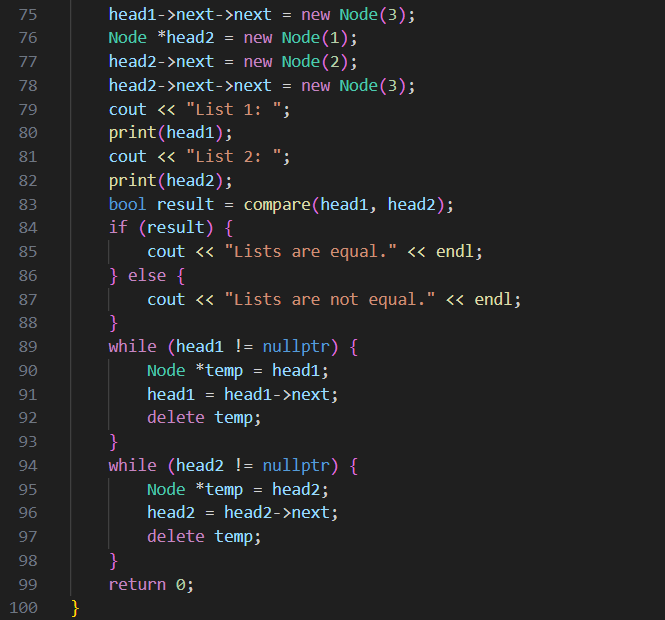
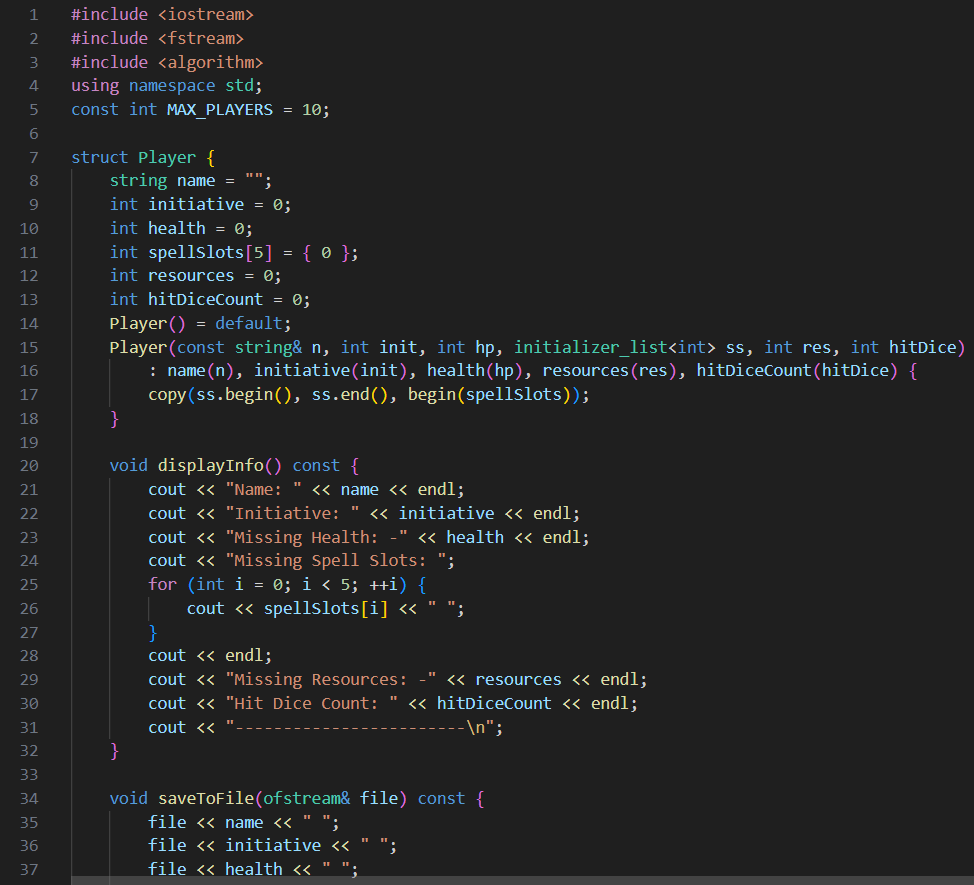
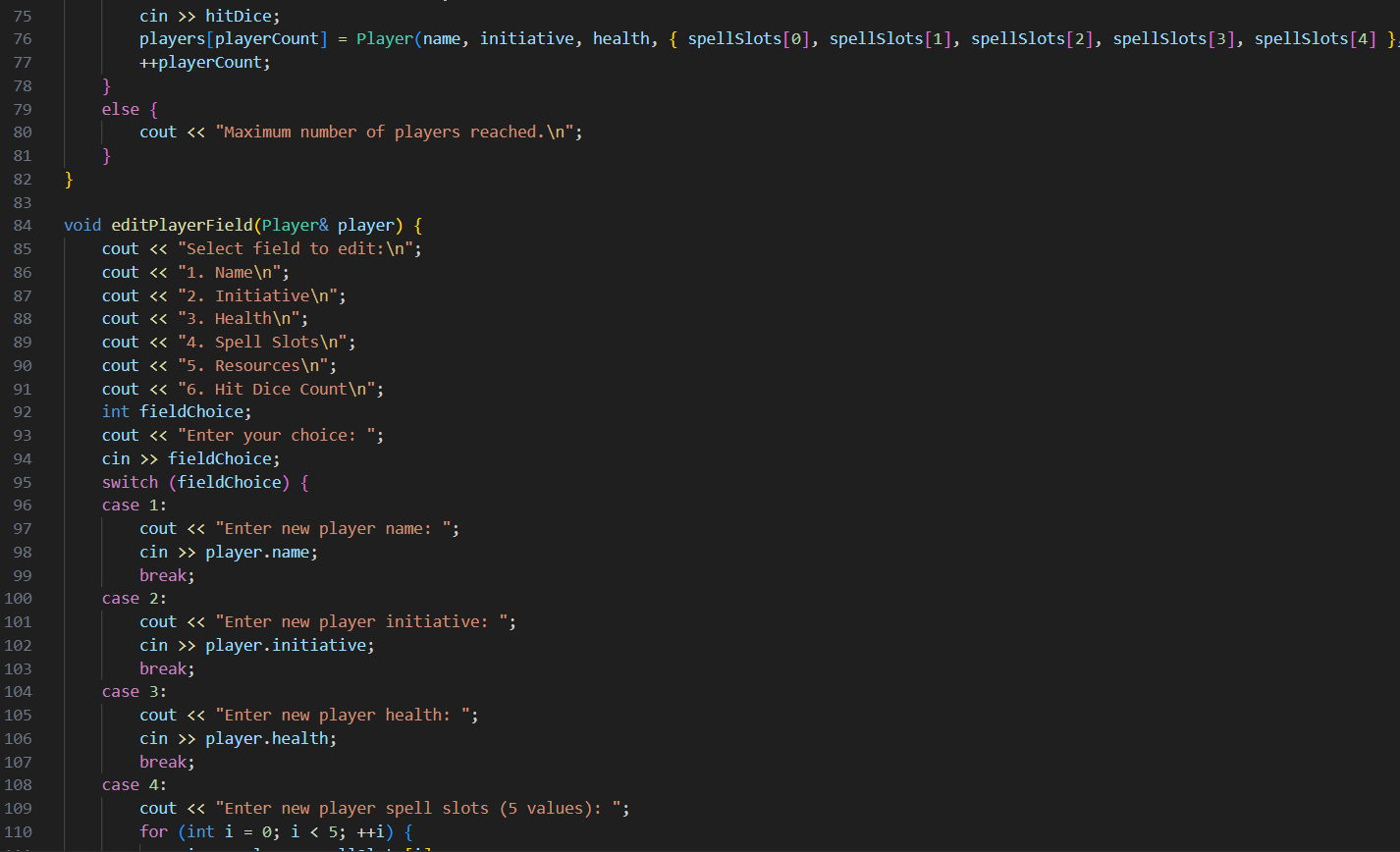


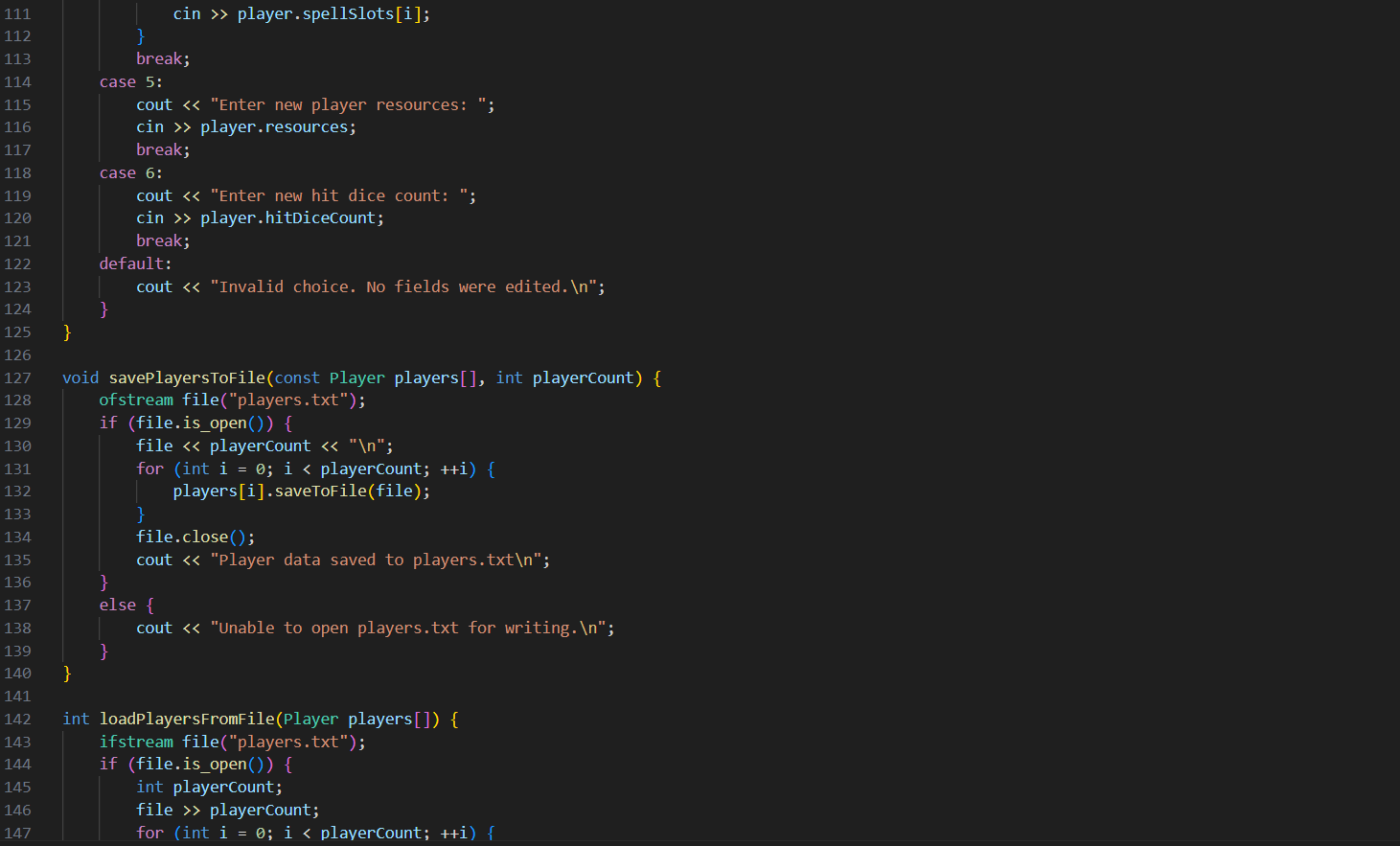
Рисунок 5: Код програми з практичної роботи

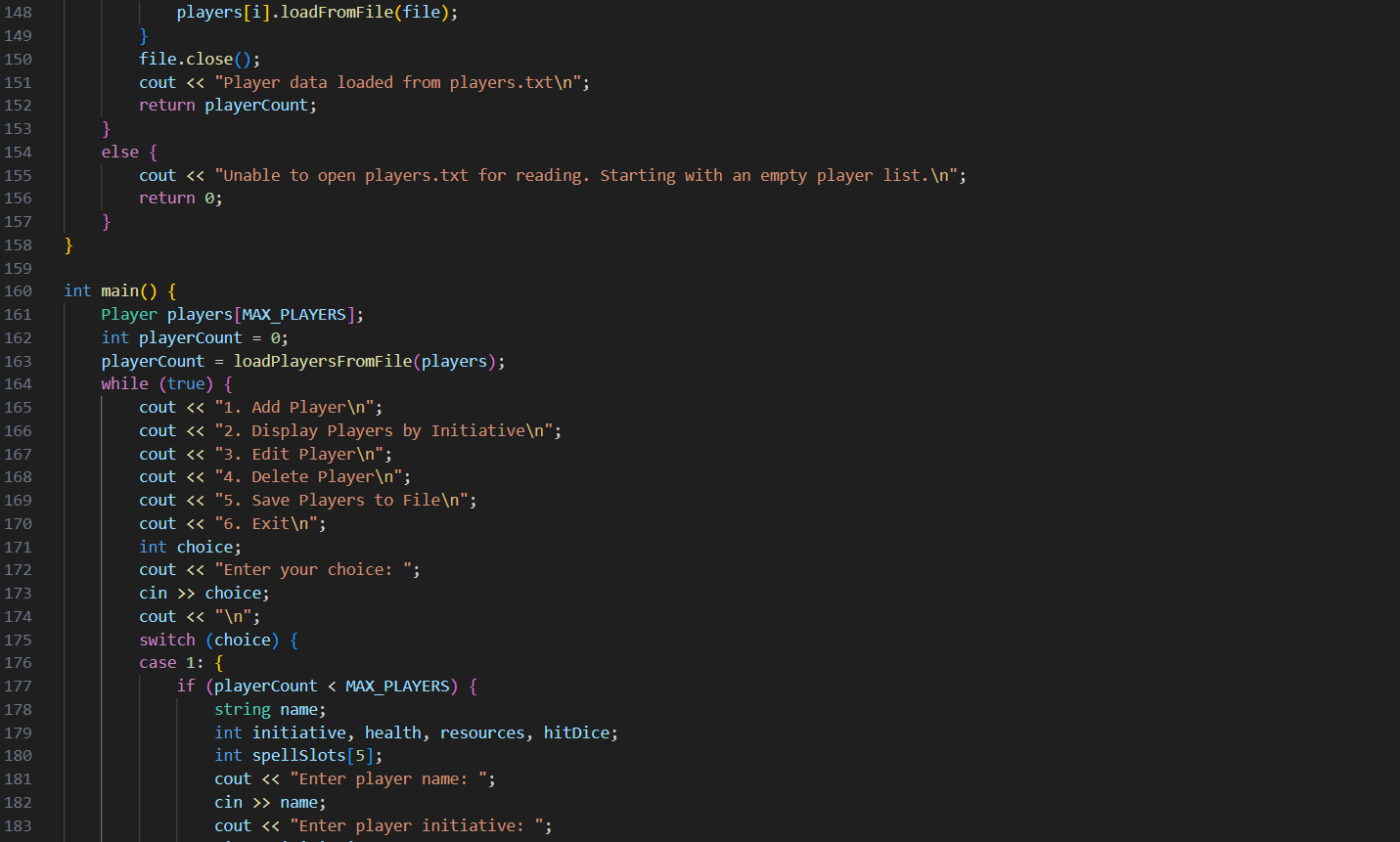
Завдання №5: Self Practice Task



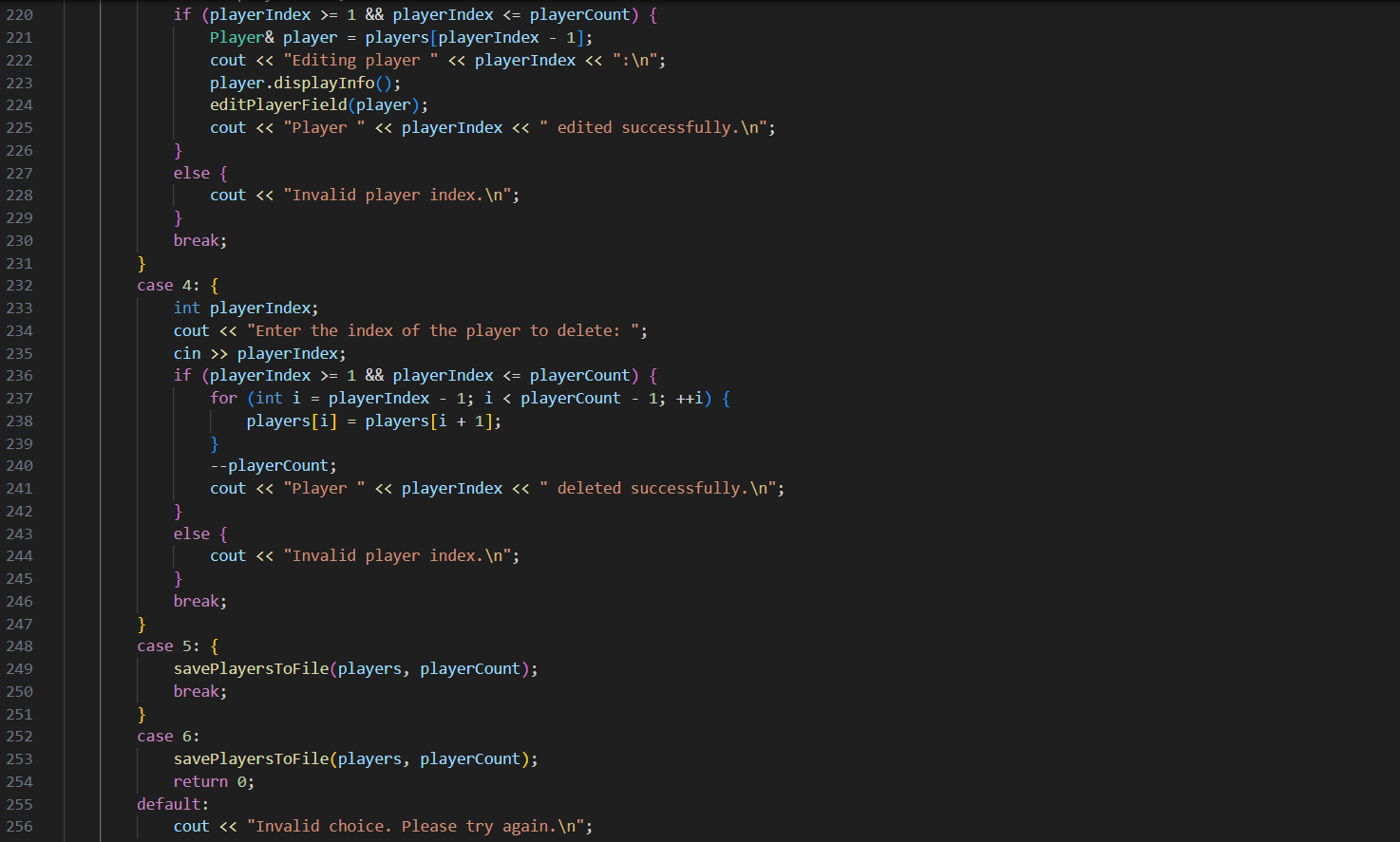












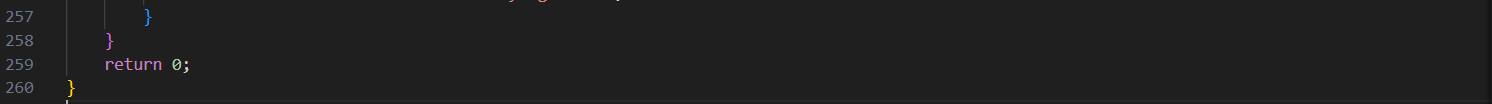


Рисунок 6: Код програми для самостійного опрацювання

Посилання на програми в GitHub: <https://github.com/Zimnov1/Repo/tree/main>

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

Завдання №1: VNS Lab 10

Результати виконання програми:

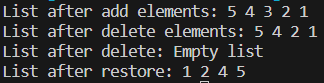


Рисунок 7: Результати виконання програми з VNS №10

Деталі по виконанню:

* Програма перевіряє чи задана позиція існуюча
* Програма перевіряє чи файл відкрився коректно

Час затрачений на виконання завдання: 2 години

Завдання №2: Algotester Lab 5

Результати виконання програми:

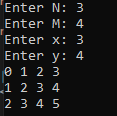


Рисунок 8: Результати виконання програми з Algotester №5

Деталі по виконанню:

* Програма провіряє валідність введення даних

Час затрачений на виконання завдання: 45 хвилин

Завдання №5: Algotester Lab 7-8

Результати виконання програми:

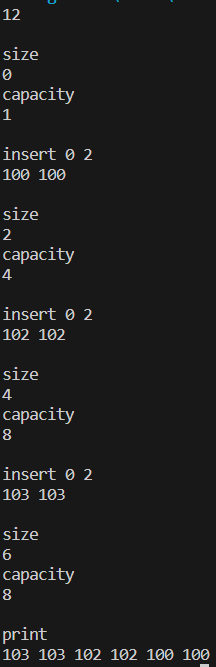


Рисунок 9: Результати виконання програми з Algotester №7-8

Деталі по виконанню:

* Було використані публічні і приватні частини класу, а також конструктори і деструктори.

Час затрачений на виконання завдання: 4 години

Завдання №6: Class Practice Task

Результати виконання програми:

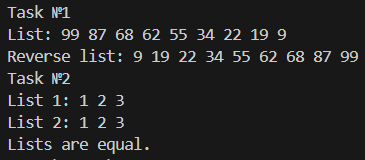


Рисунок 10: Результати виконання практичної роботи

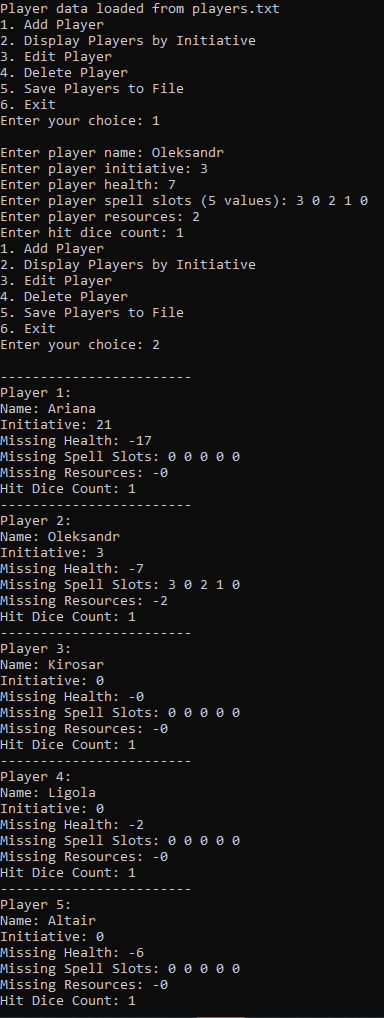
Деталі по виконанню:

* Доступ до кожного елементу списку виконується завдяки ->next

Час затрачений на виконання завдання: 1 година 30 хвилин

Завдання №7: Self Practice Task

Результати виконання програми:



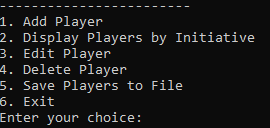


Рисунок 11: Результати виконання роботи для самостійного опрацювання

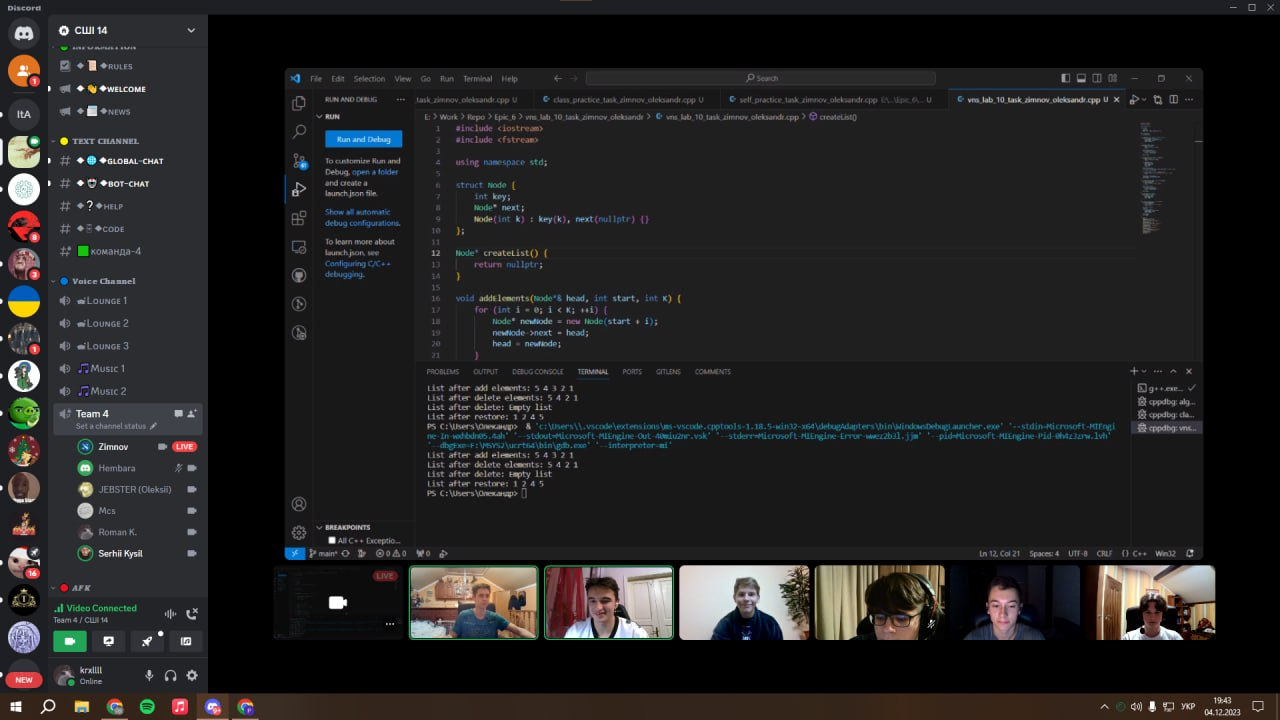
Деталі по виконанню:

* Потрібно було створити шаблон за замовчуванням
* Програма зберігає дані при виході, щоб запобігти втраті даних

Час затрачений на виконання завдання: 3 години

## **5. Кооперація з командою:**

Знімки конференції:



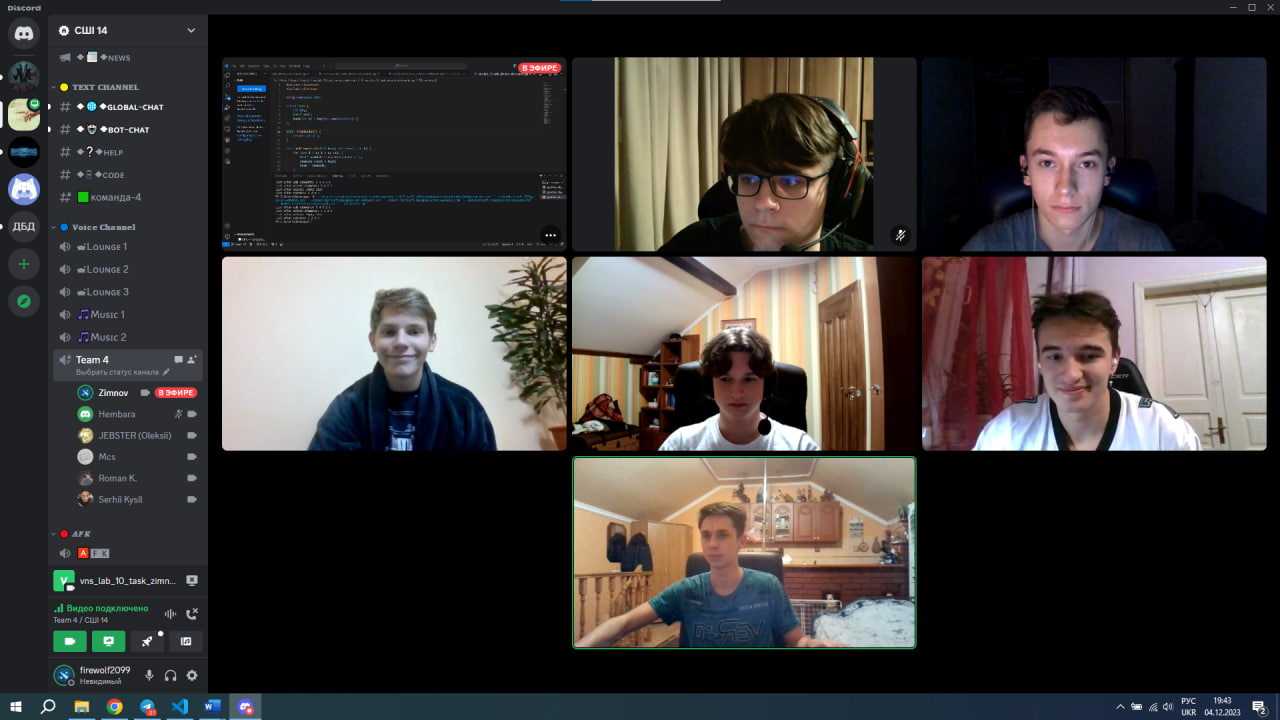


Рисунок 12: Знімки зустрічі з командою

Знімок Trello:

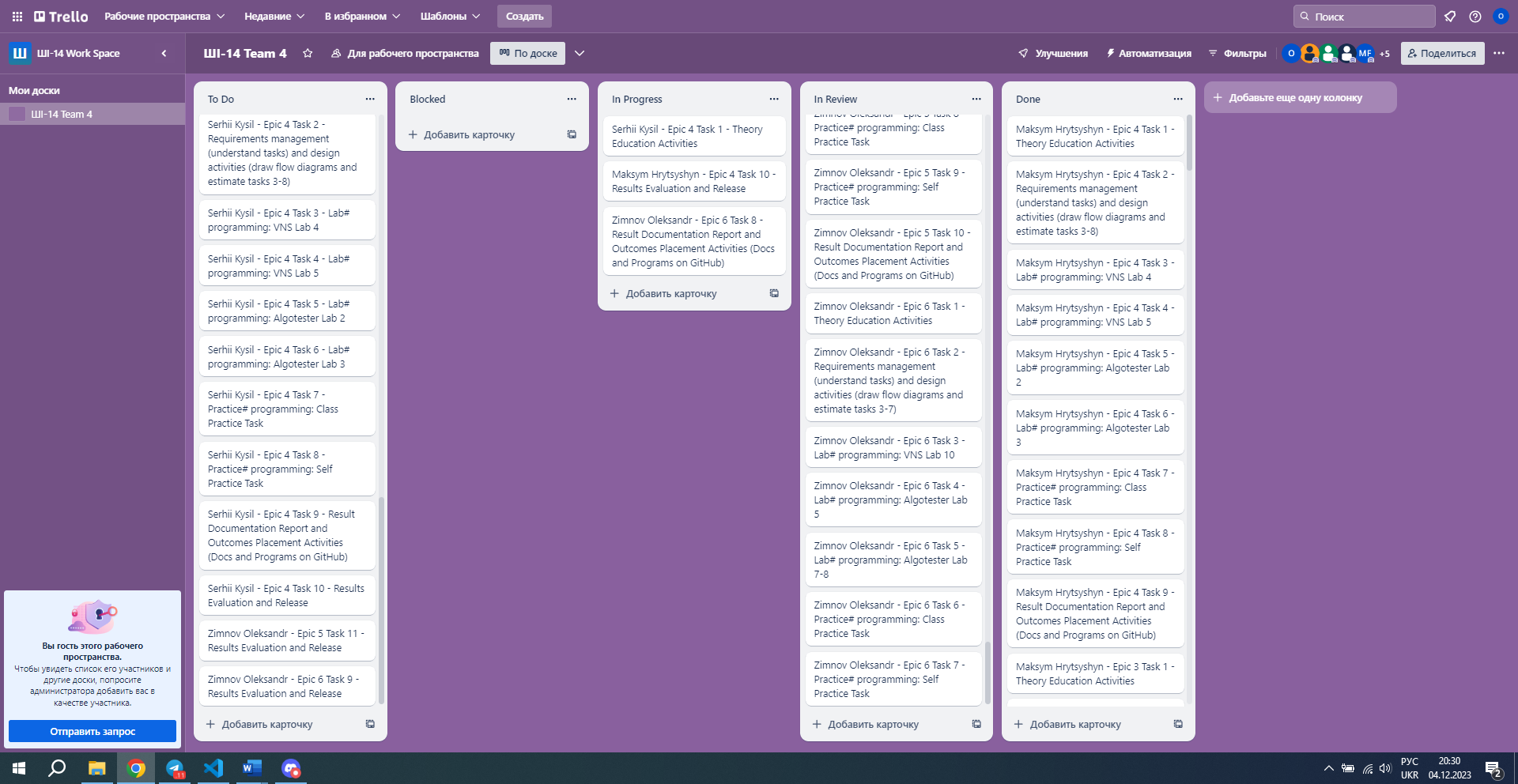


Рисунок 13: Завдання в Trello

Знімок коментарів від команди:

Рисунок 14: Коментарі від команди

# **Висновки:**

# Протягом лабораторних та практичних робіт №6було вивчено динамічні структури, такі як черга, стек, списки і дерево, а також алгоритми обробки динамічних структур

Всі завдання були відслідковані в дошці на Trello. Були онлайн зустрічі з командою, в яких ми владнали всі непорозуміння.

Посилання на Pull request: https://github.com/artificial-intelligence department/ai\_programming\_playground/pull/507