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 середовище розробки програми.»

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

до:

Практичних Робіт до блоку № 6

**Виконав:**

Студент групи ШІ-13

Федів Андрій Сергійович

Львів 2024

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

**Мета:** Засвоїти основи роботи з динамічними структурами даних, такими як черга, стек, списки та дерева. Ознайомитися з алгоритмами їх обробки для розв'язання різноманітних задач.

**Теоретичні відомості:**

1. Основи Динамічних Структур Даних:
   * Вступ до динамічних структур даних: визначення та важливість
   * Виділення пам'яті для структур даних (stack і heap)
   * Приклади простих динамічних структур: динамічний масив
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
   * Приклади використання черги: обробка подій, алгоритми планування
   * Розширення функціоналу черги: пріоритетні черги
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:
   * 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
6. Алгоритми Обробки Динамічних Структур:
   * Основи алгоритмічних патернів: ітеративні, рекурсивні
   * Алгоритми пошуку, сортування даних, додавання та видалення елементів

**Індивідуальний план опрацювання теорії:**

Основи Динамічних Структур Даних

Стек

Черга

Зв'язні Списки

Дерева

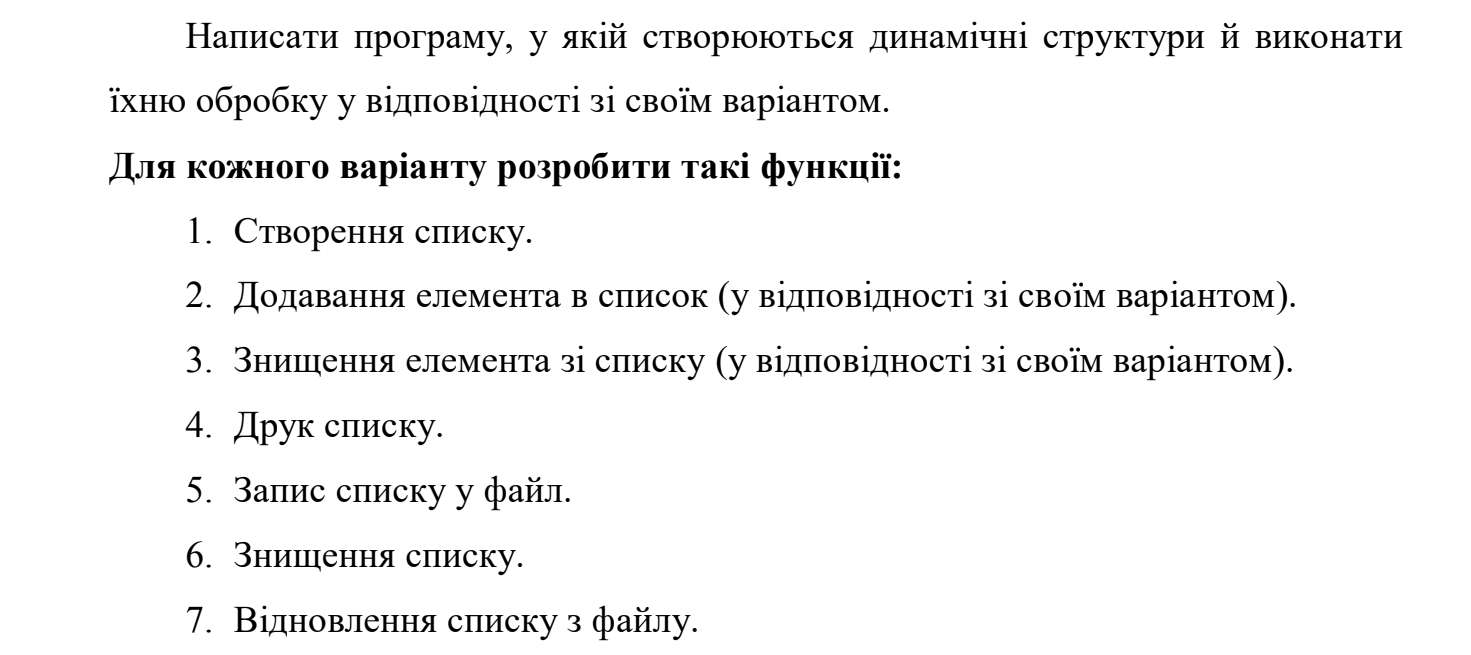
Алгоритми Обробки Динамічних Структур

1. **Джерела**:

* Chat gpt
* Список відтворення на YouTube ( <https://youtube.com/playlist?list=PLiPRE8VmJzOpn6PzYf0higmCEyGzo2A5g&si=sXvmPdnGkwvJLXUi> )
* Лекції та практичні

**Виконання роботи:**

**VNS Lab 10:**



Записи в лінійному списку містять ключове поле типу \*char (рядок

символів). Сформувати двонаправлений список. Знищити К елементів з

кінця списку. Додати елемент після елемента із заданим ключем.

**Algotester Lab 5:**

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.

**Algotester Lab 7-8 v1:**

Ваше завдання - власноруч реалізувати структуру даних "Двозв’язний список".  
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 списку, який треба вставити.  
  У третьому рядку N цілих чисел - список, який треба вставити на позицію index.
* **Видалення**:  
  Ідентифікатор - erase  
  Ви отримуєте 2 цілих числа - index, індекс елемента, з якого почати видалення та nn - кількість елементів, яку треба видалити.
* **Визначення розміру**:  
  Ідентифікатор - size  
  Ви не отримуєте аргументів.  
  Ви виводите кількість елементів у списку.
* **Отримання значення** i-го елементу  
  Ідентифікатор - get  
  Ви отримуєте ціле число - index, індекс елемента.  
  Ви виводите значення елемента за індексом.
* **Модифікація значення** ii-го елементу  
  Ідентифікатор - set  
  Ви отримуєте 2 цілих числа - індекс елемента, який треба змінити, та його нове значення.
* **Вивід списку на екран**  
  Ідентифікатор - print  
  Ви не отримуєте аргументів.  
  Ви виводите усі елементи списку через пробіл.  
  Реалізувати використовуючи перегрузку оператора <<

**Algotester Lab 7-8 v2:**

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 <<

**Class Practice Task:**

## 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;

## 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***.

## 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.

## 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

## 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

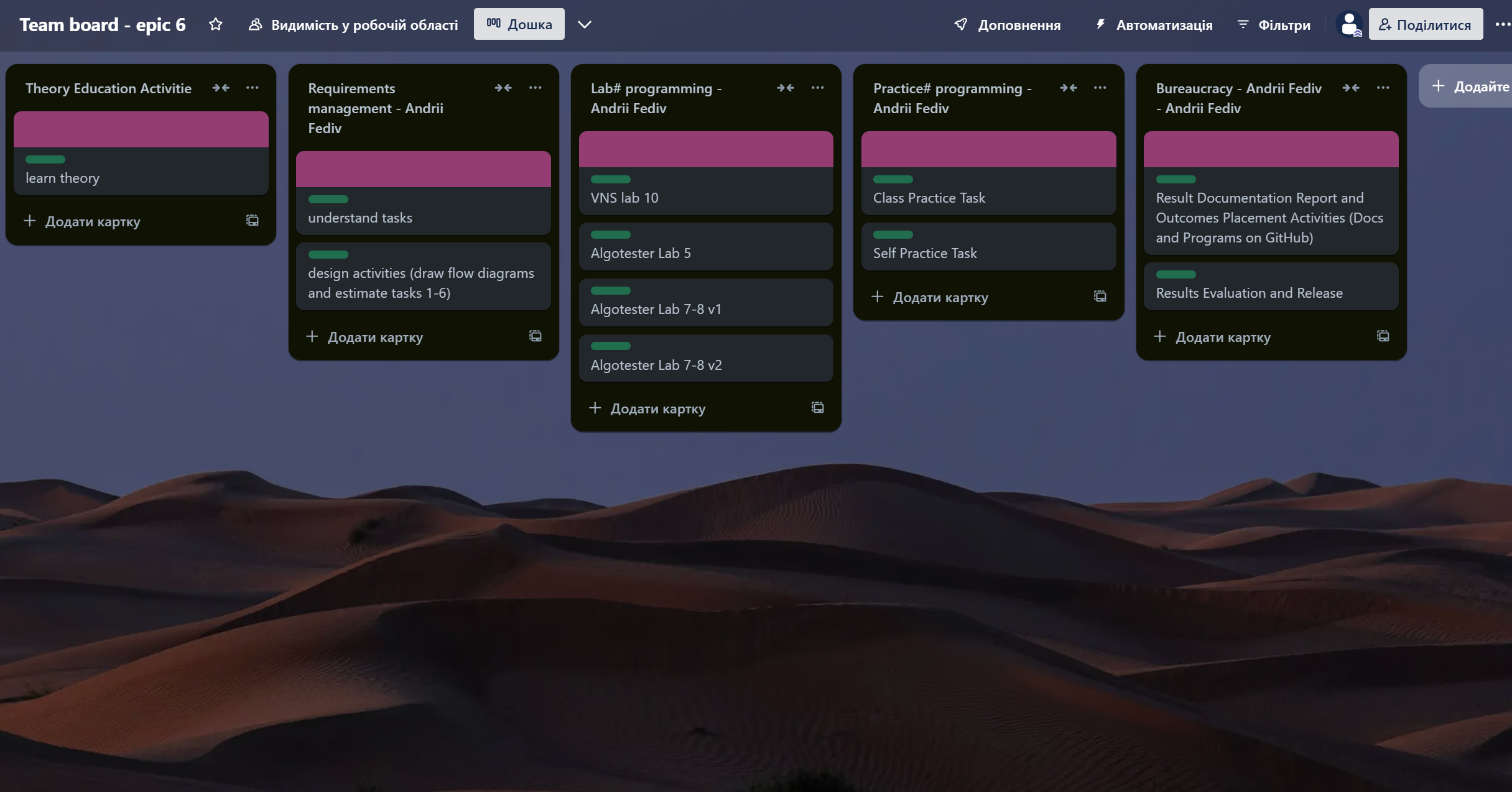
**Self Practice Task (7-8 v3):**

Ваше завдання - власноруч реалізувати структуру даних "Двійкове дерево пошуку".  
Ви отримаєте Q запитів, кожен запит буде починатися зі слова-ідентифікатора, після якого йдуть його параметри.  
  
Вам будуть поступати запити такого типу:

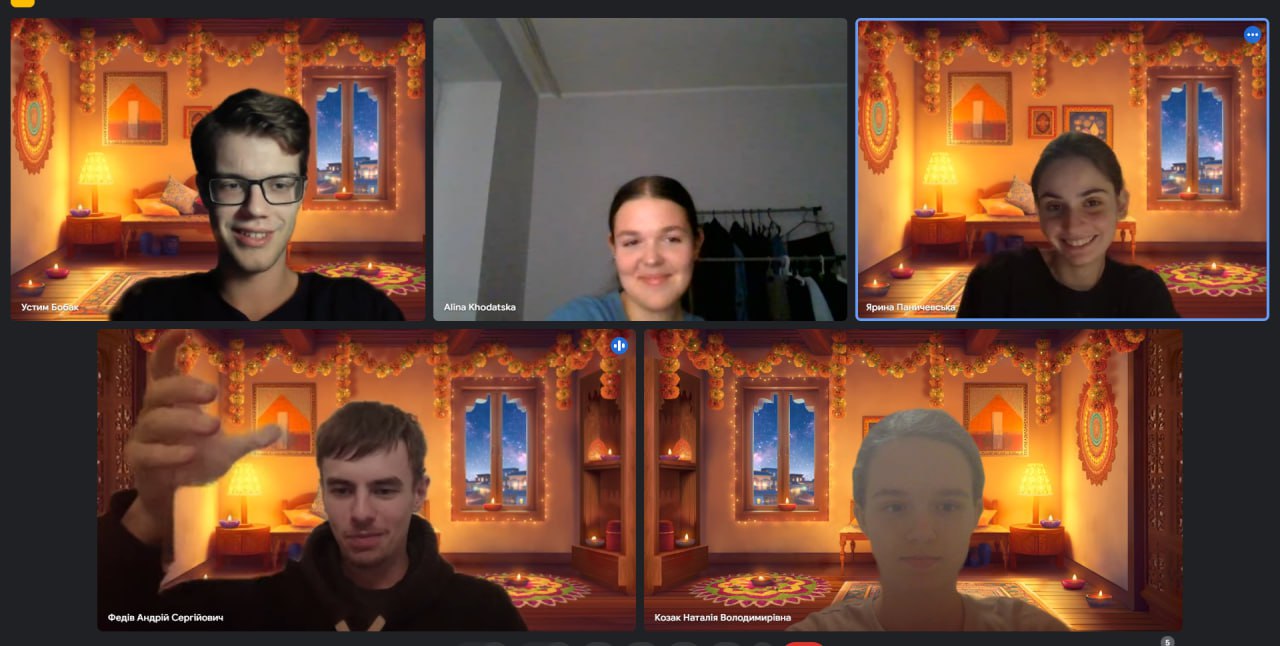
* **Вставка**:  
  Ідентифікатор - insert  
  Ви отримуєте ціле число value - число, яке треба вставити в дерево.
* **Пошук**:  
  Ідентифікатор - contains  
  Ви отримуєте ціле число value - число, наявність якого у дереві необхідно перевірити.  
  Якщо value наявне в дереві - ви виводите Yes, у іншому випадку No.
* **Визначення розміру**:  
  Ідентифікатор - size  
  Ви не отримуєте аргументів.  
  Ви виводите кількість елементів у дереві.
* **Вивід дерева на екран**  
  Ідентифікатор - print  
  Ви не отримуєте аргументів.  
  Ви виводите усі елементи дерева через пробіл.  
  Реалізувати використовуючи перегрузку оператора <<

1. Requirements management and design activities

Team Trello dashboard for task control



Team meeting in zoom



**UML-diagram block-scheme for 1 task and plannig**

VNS Lab 10 затратність ~3год

Algotester Lab 5 затратність ~1год

Algotester Lab 7-8 v1 затратність ~2год

**Діаграма знаходиться в src/algotester\_lab\_7\_8\_variant\_1\_andrii\_fediv.drawio**

Algotester Lab 7-8 v2 затратність ~1год

Class Practice Task затратність ~2.5год

Self Practice Task затратність ~1год

Код програми з посиланням на зовнішні ресурси

VNS Lab 10: **./src/vns\_lab\_10\_task\_andrii\_fediv.cpp**

Algotester Lab 5: **./src/algotester\_lab\_5\_task\_andrii\_fediv.cpp**

Algotester Lab 7-8 v1: **./src/algotester\_lab\_7\_8\_variant\_1\_andrii\_fediv.cpp**

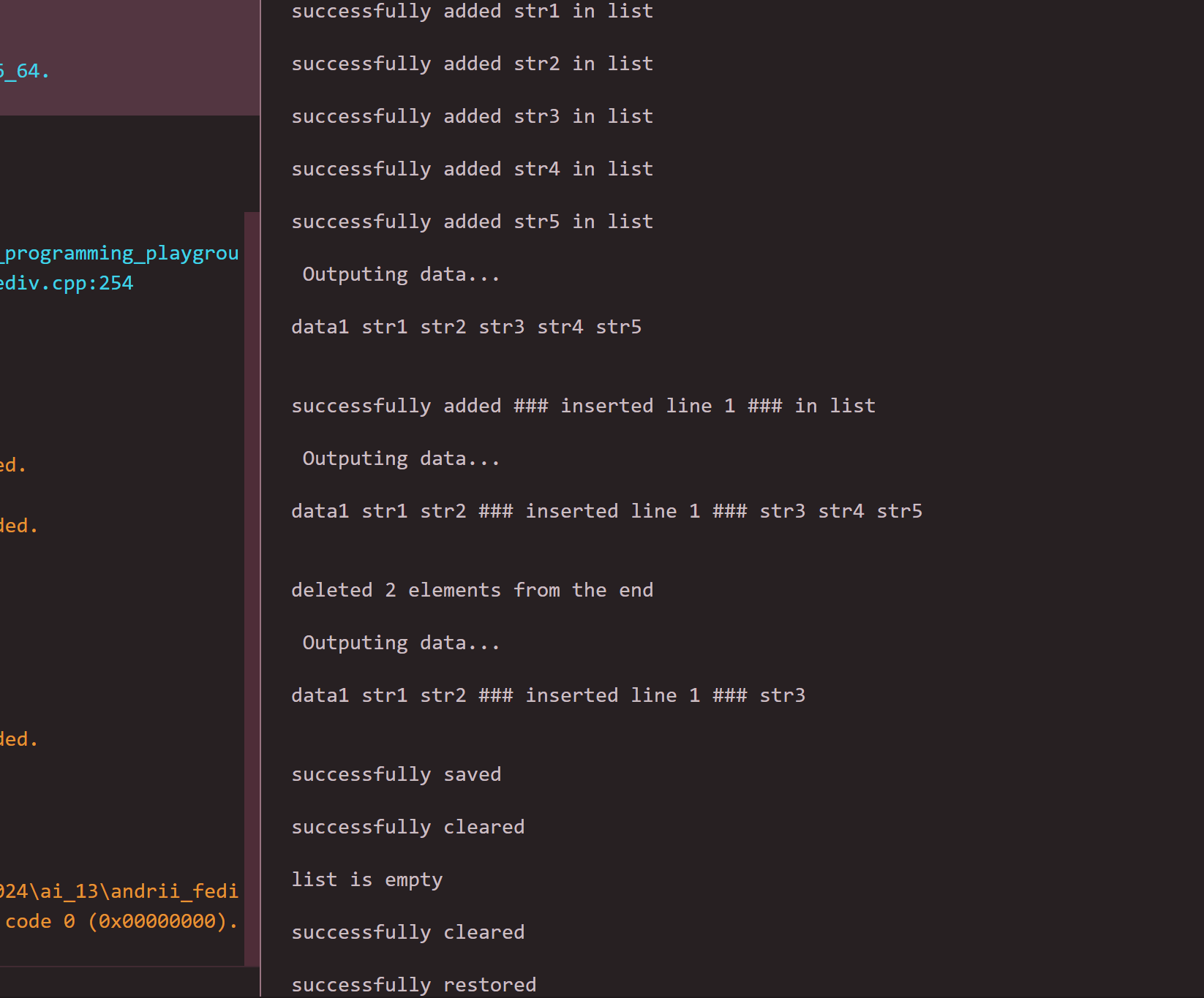
Algotester Lab 7-8 v2: **./src/algotester\_lab\_7\_8\_variant\_2\_andrii\_fediv.cpp**

Class Practice Task: **./src/practice\_work\_team\_tasks\_andrii\_fediv.cpp**

Self Practice Task: **./src/self\_practice\_work\_algotester\_task\_1\_andrii\_fediv.cpp**

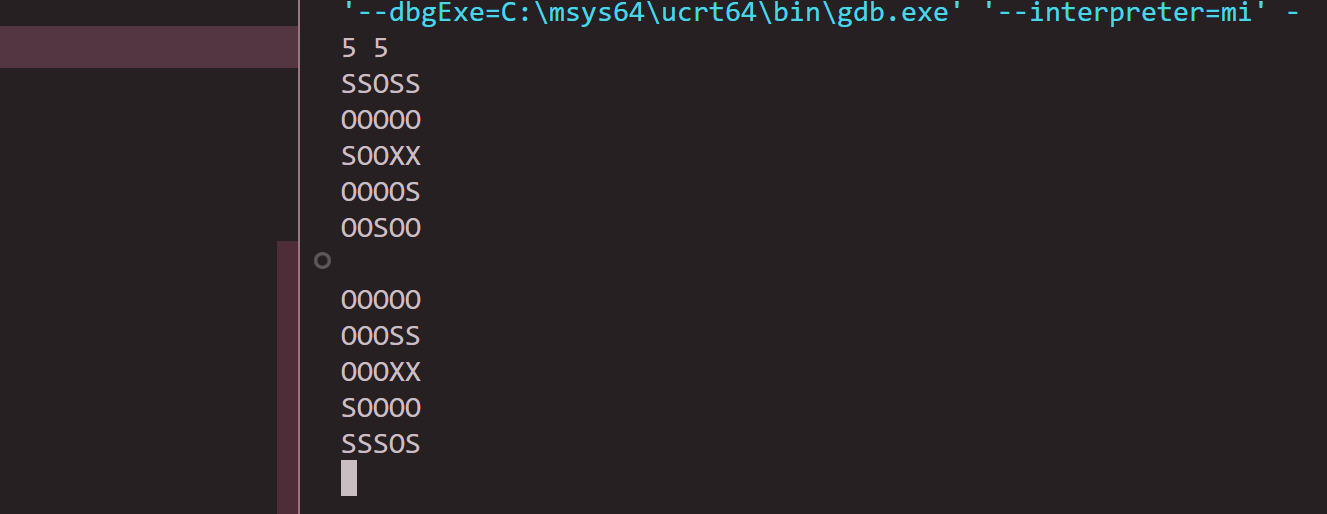
**Результати виконаних завдань, тестування та фактично затрачений час**

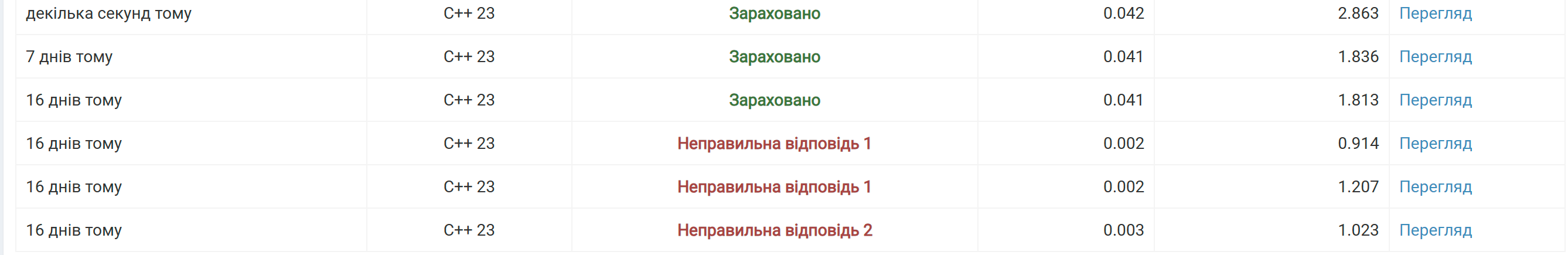
VNS Lab 10



Затратність ~8год

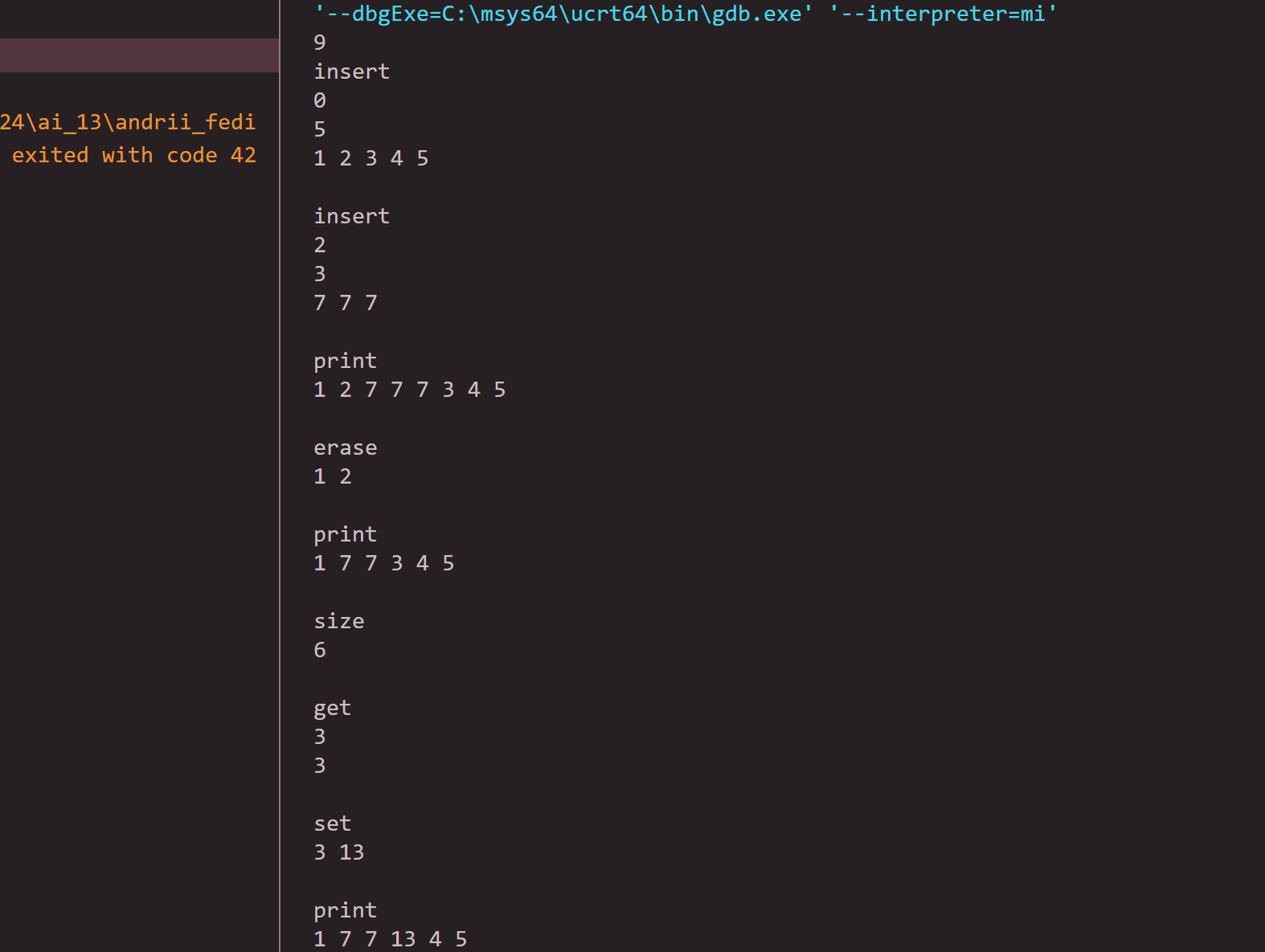
Algotester Lab 5

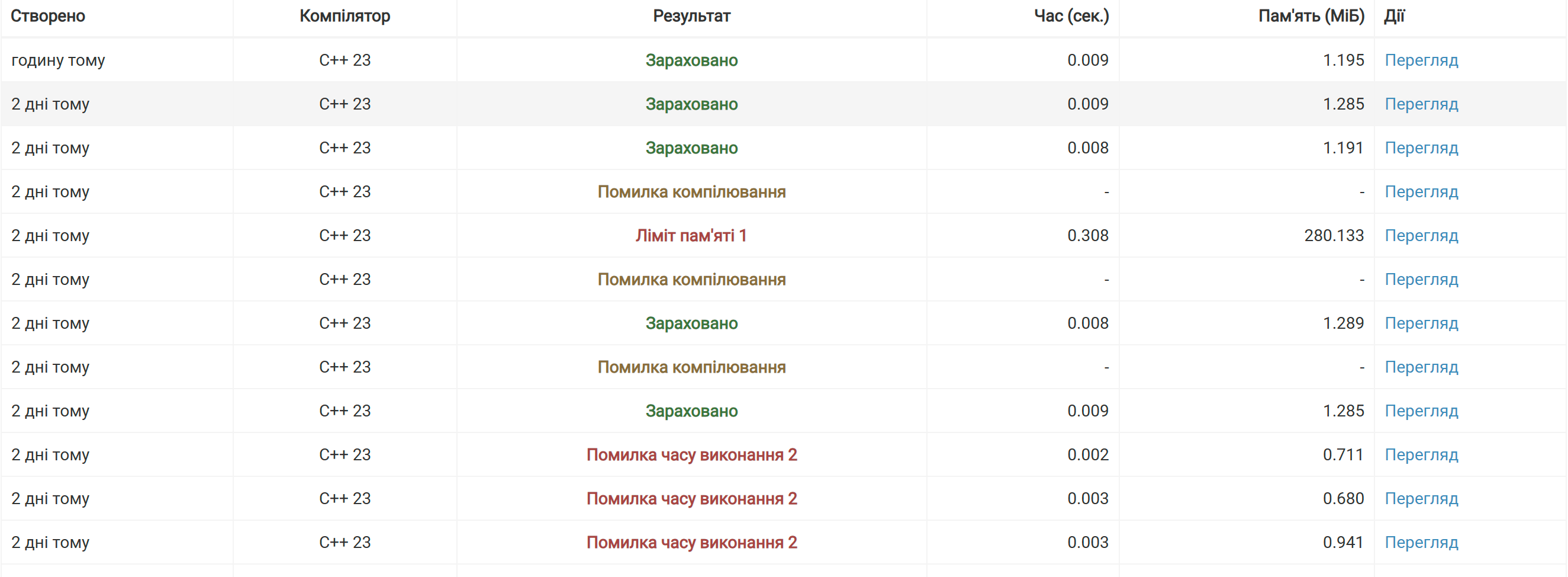




Затратність ~40хв

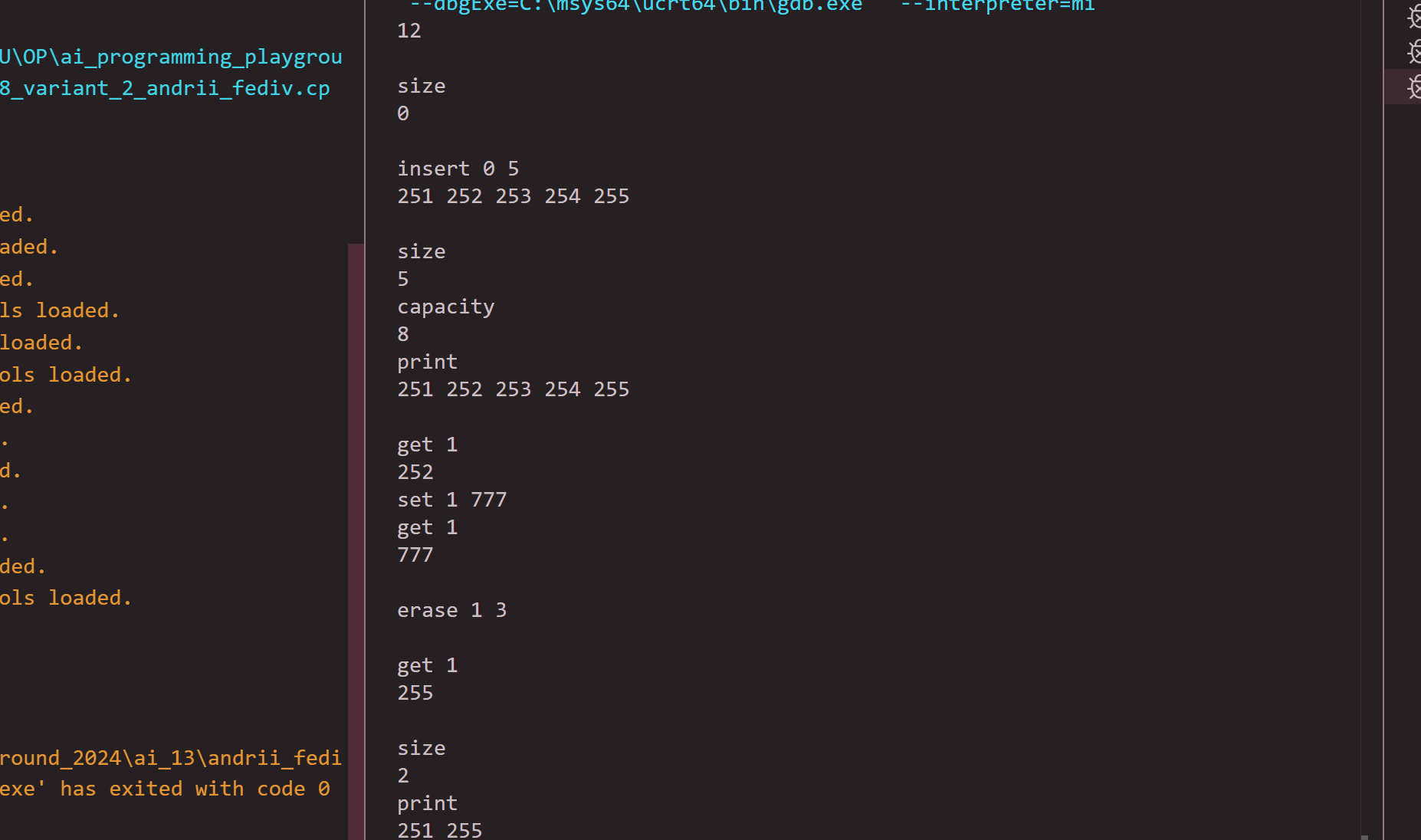
Algotester Lab 7-8 v1

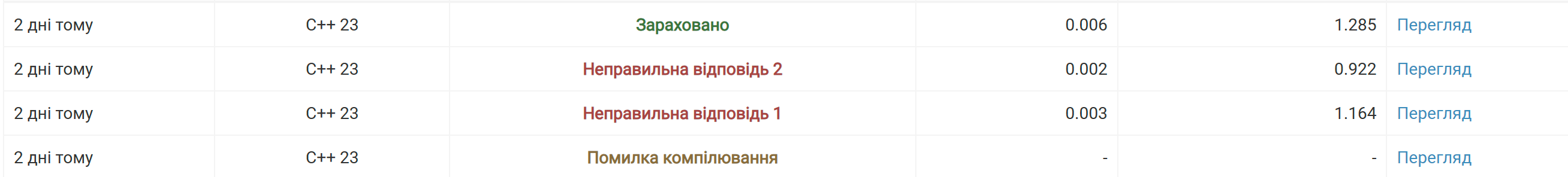




Затратність ~18год

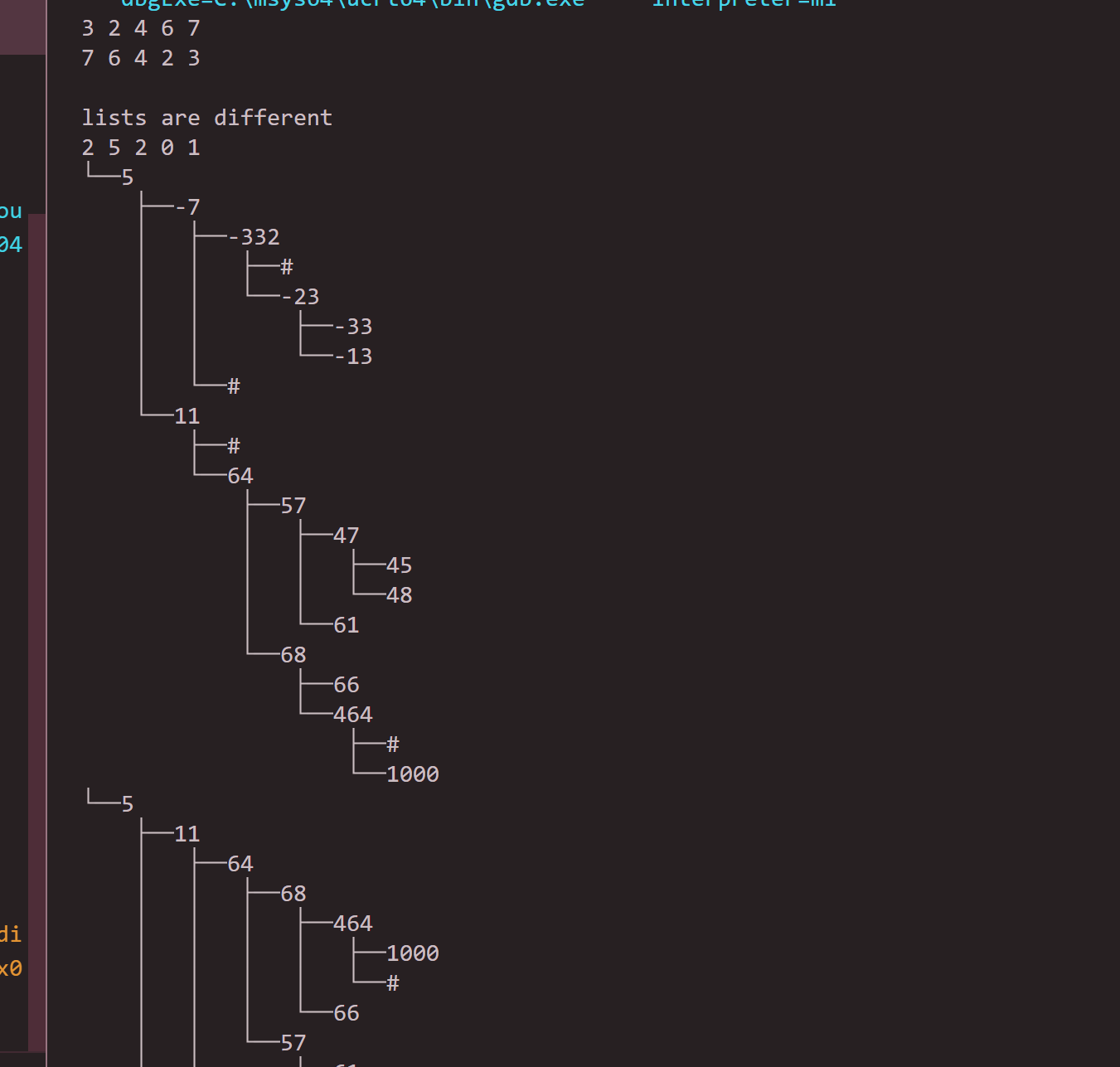
Algotester Lab 7-8 v2





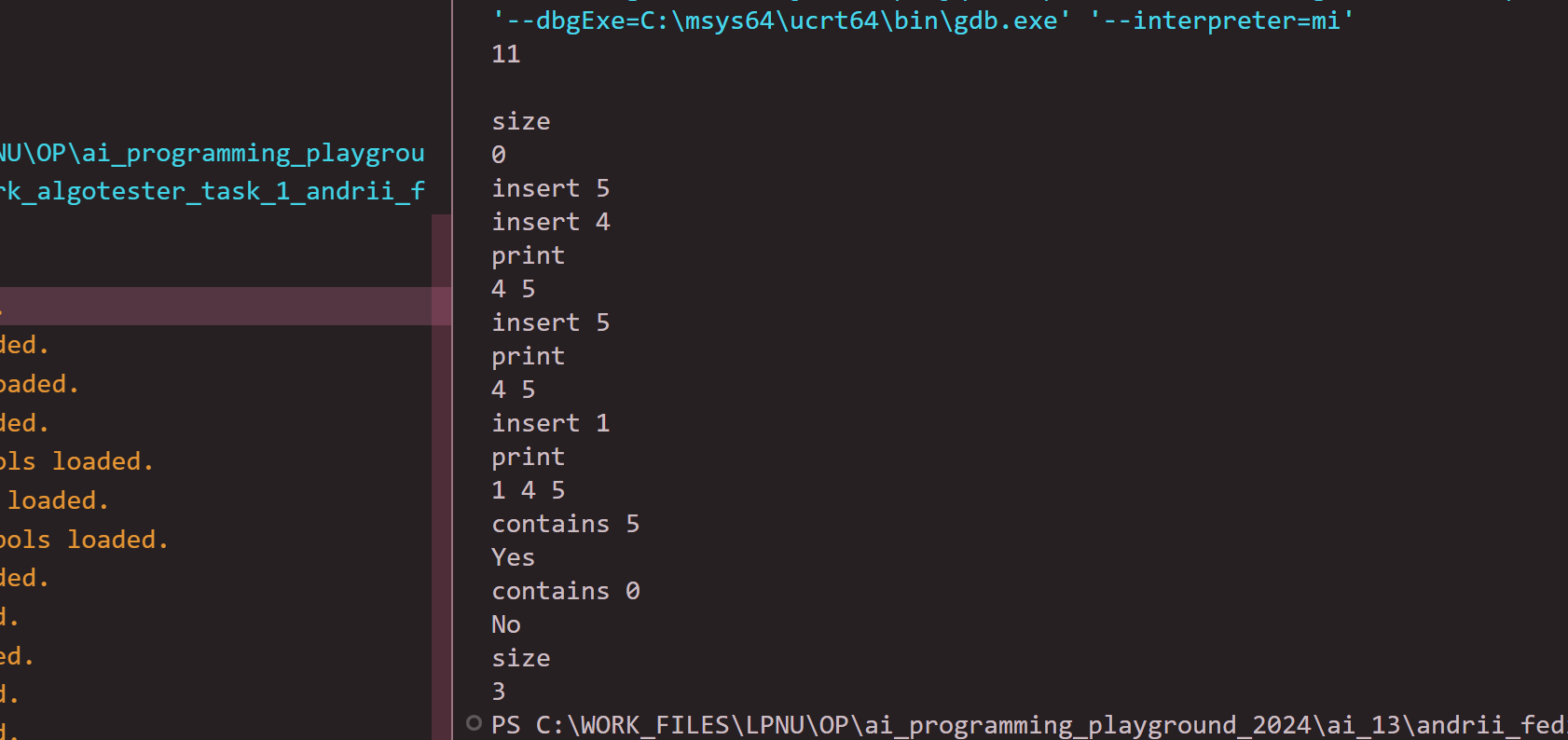
Затратність ~3год

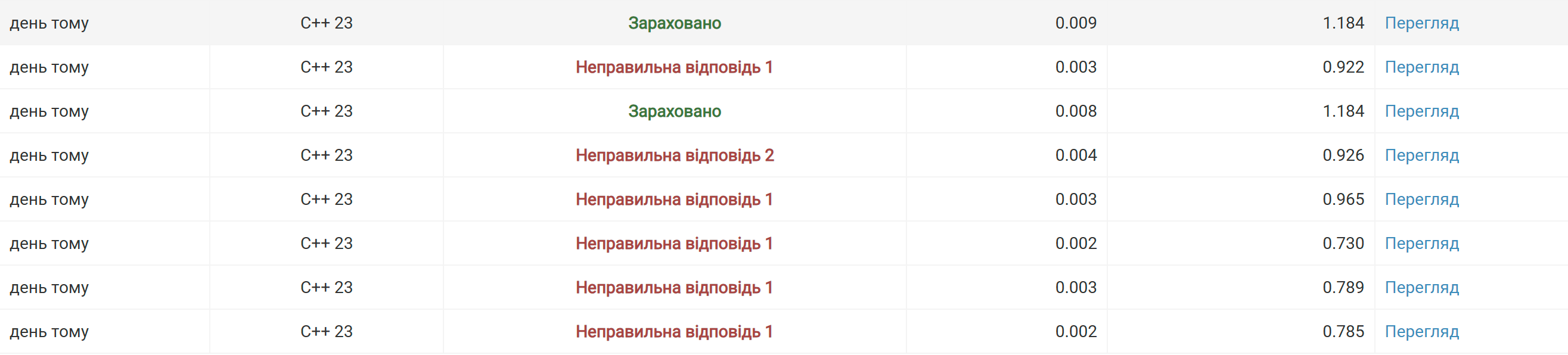
Class Practice Task



Затратність ~4год

Self Practice Task

****

****

Затратність ~1год

**Висновки:**

Я навчився застосовувати динамічні структури для ефективного зберігання та обробки даних в програмах. Також отримав розуміння алгоритмів для роботи з чергою, стеком, списками та деревами, що дозволяє вирішувати складні обчислювальні задачі.