Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Кафедра систем штучного інтелекту

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
Тофан Максим Васильович

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

Джерела:

- Chat GPT
- YouTube
- Методички

Виконання роботи:

VNS Lab 10v13:

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.

Записи в лінійному списку містять ключове поле типу *char (рядок символів). Сформувати двонаправлений список. Знищити з нього K перших елементів. Додати елемент після елемента, що починається із зазначеного символу.

Algotester Lab 5v1:

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 рядка 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 заклинання, 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).

Algotester Lab 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 вводите YES, у іншому випадку NO
- **Визначення розміру**:
  Ідентифікатор - size
  Ви не отримуєте аргументів.
  Ви виводите кількість елементів у списку.
- **Вивід дерева на екран:**
  Ідентифікатор - print
  Ви не отримуєте аргументів.
  Ви виводите усі елементи дерева 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. $379 \implies 9 \to 7 \to 3$);
- 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 (5v2):

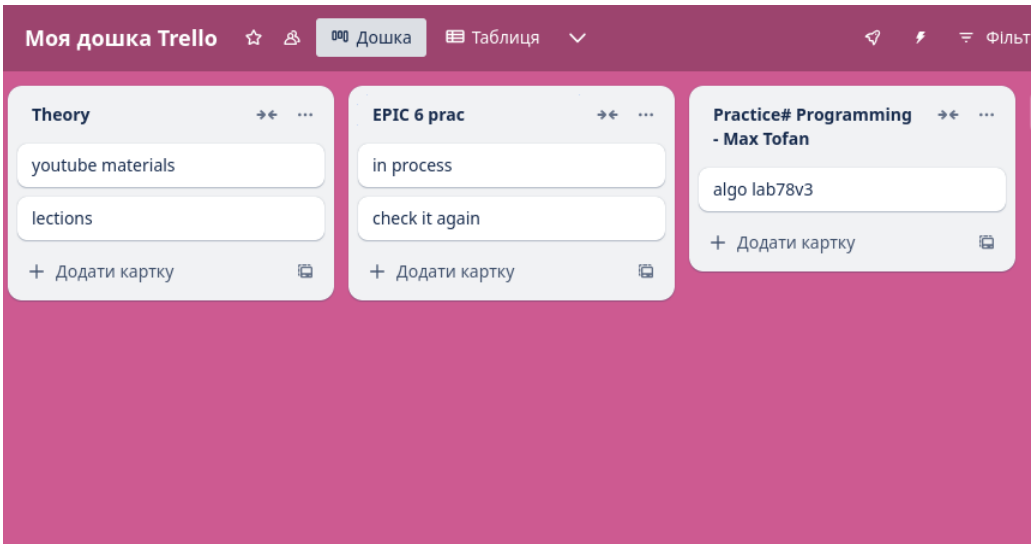
В пустелі існує незвичайна печера, яка є двохвимірною. Її висота це N, ширина — M.

Всередині печери є пустота, пісок та каміння. Пустота позначається буквою O, 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.

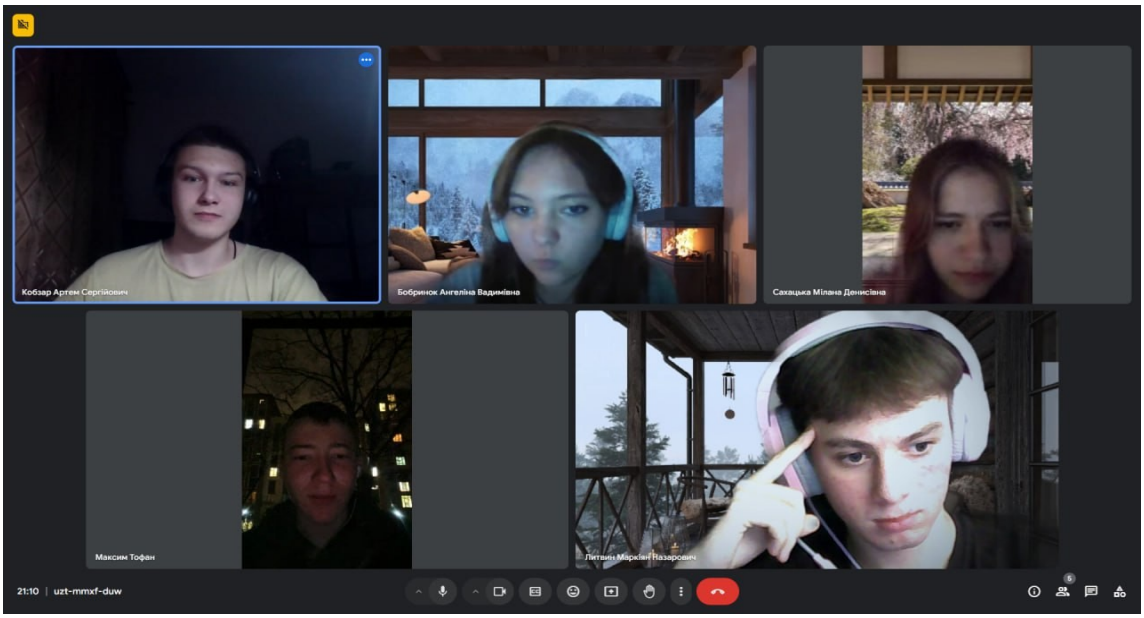
Ваше завдання сказати як буде виглядати печера після землетрусу.

1. Requirements management and design activities

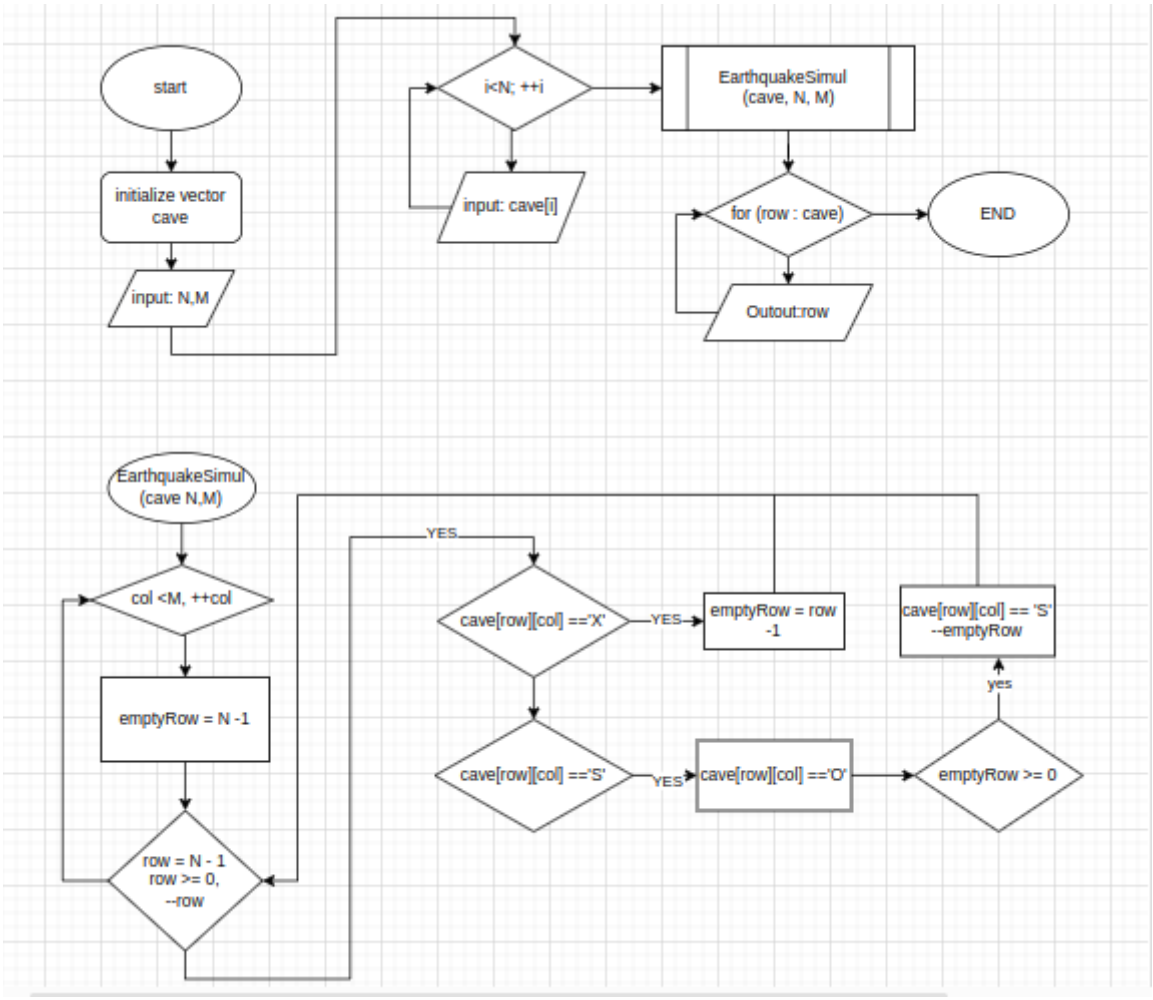
Team Trello dashboard for task control



Team meeting in zoom



UML-diagram block-scheme for 1 task and plannig



Self Practice Task затратність ~1год

Результати виконаних завдань, тестування та фактично затрачений час

VNS Lab 10

```
Початковий список: apple banana cherry
Після додавання 'blueberry' після 'banana': apple banana blueberry cherry
Після видалення 2 перших елементів: blueberry cherry
Список записано у файл 'list.txt'.
Список очищено.
Список відновлено з файлу: blueberry cherry
[1] + Done                    "/usr/bin/gdb" --interpreter=mi --tty=${Db
afaq0ti.kxk" 1>"/tmp/Microsoft-MIEngine-Out-vtlll2k3.2r3"
max@max-user:~/lpnu/epic_6$
```

Затратність ~4год

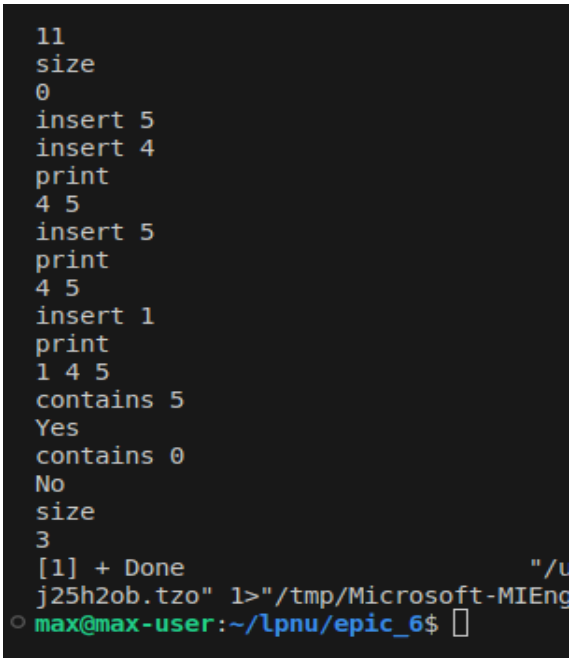
Algotester Lab 5

```
0
4
1 1
1 2
2 2
2 1
771
[1] + Done                    "
aigodd4.uon" 1>"/tmp/Microsoft-MIE
max@max-user:~/lpnu/epic_6$
```

Створено	Компілятор	Результат	Час (сек.)	Пам'ять (МіБ)	Дії
декілька секунд тому	C++ 23	Зараховано	0.003	1.387	Перегляд
27 хвилин тому	C++ 23	Зараховано	0.003	1.445	Перегляд
2 години тому	C++ 23	Неправильна відповідь 1	0.002	0.949	Перегляд

Затратність ~40хв

Algotester Lab 7-8 v3



Створено	Компілятор	Результат	Час (сек.)	Пам'ять (МіБ)	Дії
декілька секунд тому	C++ 23	Зараховано	0.008	1.316	Перегляд
7 хвилин тому	C++ 23	Неправильна відповідь 1	0.002	0.930	Перегляд

Затратність ~5год

Class Practice Task

```
3 2 4 6 7
7 6 4 2 3

lists are different
2 5 2 0 1
5 -7 -332 -23 -33 -13 11 64 57 47 45 48 61 68 66 464 1000 5 11 64 68 464 1000 66 57 61 47 48 45 -7 -332 -23 -13
33 Tree sum: 1528
[1] + Done                    "/usr/bin/gdb" --interpreter=mi --tty=${DbgTerm} 0<"/tmp/Microsoft-MIEngine-In-
djdglsk.hjw" 1>"/tmp/Microsoft-MIEngine-Out-5up5wnxw.fbr"
max@max-user:~/lpnu/epic_6$
```

1

Затратність ~4год

Self Practice Task

```
5 4
KKOKK
JJOOO
OOOOO
SSOSS
OOSOO

KKOK
JJOO
OOOO
OOOO
SSSS
[1] + Done                    "/usr/bin/
```

Створено	Компілятор	Результат	Час (сек.)	Пам'ять (МіБ)	Дії
декілька секунд тому	C++ 23	Зараховано	0.041	3.039	Перегляд

Затратність ~1год

Висновки:

Я освоїв використання динамічних структур для ефективного зберігання та обробки даних у програмах, а також здобув знання алгоритмів для роботи з чергами, стеком, списками та деревами, що допомагає вирішувати складні обчислювальні задачі.
.