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

Звіт

про виконання лабораторних та практичних робіт блоку № 5

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
Матрунич Олександр Іванович

Львів 2024

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

Мета: Розібратися з основами динамічних структур даних і зрозуміти їх значення для керування пам’яттю та обробки інформації. Реалізувати основні операції для стеку, черги, зв’язних списків і дерев. Навчитися застосовувати ці структури даних у реальних задачах, таких як сортування, пошук і управління даними.

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:

- Лекції О. Пшеничного
- Практичні М. Фаріон
- Chat gpt
- Список відтворення на YouTube (https://youtube.com/playlist?list=PLiPRE8VmJzOpn6PzYf0higmCEyGzo2A5g&si=sXvmPdnGkwvJLXUi)
- Власний досвід

Виконання роботи:

1) Опрацювання завдання та вимог до програми та середовища

VNS Lab 10 - Task 1-6:

Записи в лінійному списку містять ключове поле типу int. Сформувати двонаправлений список. Знищити з нього елемент із заданим номером, додати елемент у початок списку.

Algotester Lab 5 var 3:

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.

Вхідні дані

У першому рядку 2 числа N та M - розміри карти

у другому рядку 2 числа x та y - координати піку гори

Вихідні дані

N рядків по M елементів в рядку через пробіл - висоти карти.

Class Practice Work 1:

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;

Class Practice Work 2:

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*.

Class Practice Work 3:

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. $379 \implies 9 \rightarrow 7 \rightarrow 3$);
- функція повертає новий список, передані в функцію списки не модифікуються.

Class Practice Work 4:

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

Class Practice Work 5:

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 (5.1):

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 ii-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).
Також вони вирішили трохи Вам допомогти і придумали спосіб як записати стан дошки одним числом a із 8-ми байт, а саме (див. Примітки):

- Молодший байт задає перший рядок матриці;
- Молодший біт задає перший стовпець рядку;
- Значення біту каже світиться куля чи ні (0 - ні, 1 - так);

Тепер їх цікавить яким буде стан дошки після виконання N заклинань і вони дуже просять Вас їм допомогти.

Вхідні дані

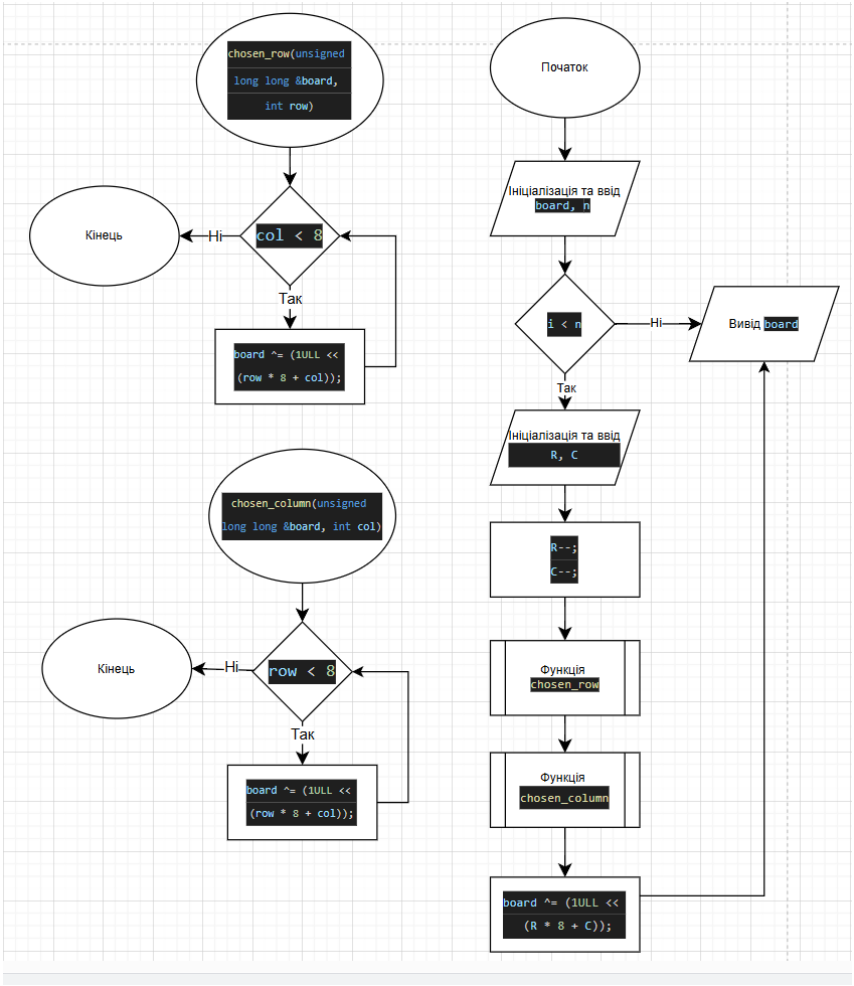
У першому рядку одне число a - поточний стан дошки.
У другому рядку N - кількість заклинань.
У наступних N рядках по 2 числа Ri, Ci - рядок і стовпець кулі над якою виконується заклинання.

Вихідні дані

Одне число b - стан дошки після виконання N заклинань.

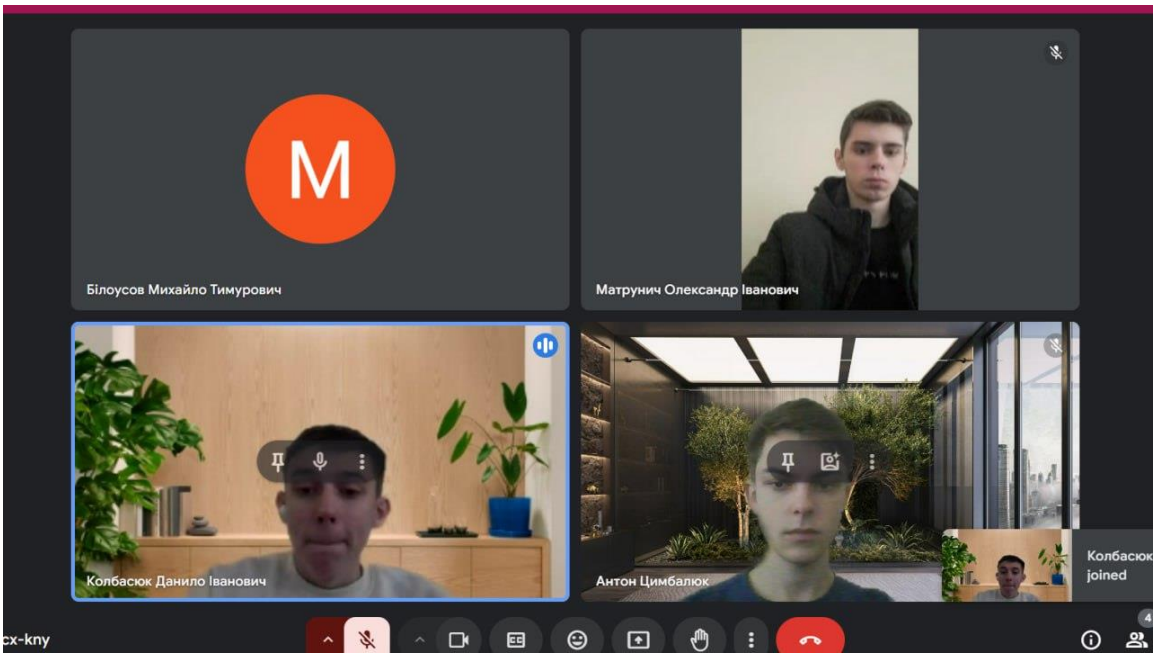
2) Дизайн

Self Practice Task (5.1):



3) Конфігурація середовища до виконання завдань:

Зустріч з командою



4) Код програми з посиланням на зовнішні ресурси

До завдання №1код ***vns_lab_10_task_oleksandr_matrunych.cpp***
До завдання №2 код ***algotester_lab_5_task_oleksandr_matrunych.cpp***
До завдання №3 код ***practice_work_team_tasks_1_oleksandr_matrunych.cpp***
До завдання №4 код ***practice_work_team_tasks_2_oleksandr_matrunych.cpp***
До завдання №5 код ***practice_work_team_tasks_3_oleksandr_matrunych.cpp***
До завдання №6 код ***practice_work_team_tasks_4_oleksandr_matrunych.cpp***
До завдання №7 код ***practice_work_team_tasks_5_oleksandr_matrunych.cpp***
До завдання №8 код ***practice_work_self_algotester_tasks_oleksandr_matrunych.cpp***

5) Результати виконаних завдань, тестування та фактично затрачений час

VNS Lab 10 - Task 1-6:

```
The list:
0 1 2 3 4 5 6 7 8 9
The list after adding an element to the front:
99 0 1 2 3 4 5 6 7 8 9
The list after deleting element at index 2:
99 0 2 3 4 5 6 7 8 9
The list was saved to "list.txt".
The cleared list:
The list is empty
The list was restored from "list.txt".
The list after recovery from file:
99 0 2 3 4 5 6 7 8 9
List after final destruction:
The list is empty
```

Час виконання: 12 год

Algotester Lab 5 var 3:

```
3 4
2 2
1 2 1 0
2 3 2 1
1 2 1 0
```

Створено	Компілятор	Результат	Час (сек.)	Пам'ять (МіБ)	Дії
хвилину тому	C++ 23	Зараховано	0.100	7.496	Перегляд

Час виконання: 1,5 год

Class Practice Work 1:

```
Original list: 1 2 3 4 5
Reversed list: 5 4 3 2 1
```

Час виконання 1.5 год

Class Practice Work 2:

```
Lists are equal.
```

```
Lists are not equal.
```

Час виконання: 1.5 год

Class Practice Work 3

```
Result: 4 2 6
```

Час виконання: 1.5 год

Class Practice Work 4

```
Mirrored tree: 1 3 2
```

Час виконання: 1.5 год

Class Practice Work 5:

```
Before tree_sum: 1 2 4 5 3 6 7
After tree_sum: 28 11 4 5 16 6 7
```

Час виконання: 1.5 год

Self Practice Task (5.1):

```
0
4
1 1
8 8
1 8
8 1
9295429630892703873
```

Створено	Компілятор	Результат	Час (сек.)	Пам'ять (МіБ)	Дії
хвилину тому	C++ 23	Зараховано	0.003	1.426	Перегляд

Час виконання: 2 год

Висновок: на лабораторній роботі я розібрався з основами динамічних структур даних і зрозумів їх значення для керування пам’яттю та обробки інформації. Реалізував основні операції для стеку, черги, зв’язних списків і дерев. Навчився застосовувати ці структури даних у реальних задачах, таких як сортування, пошук і управління даними.