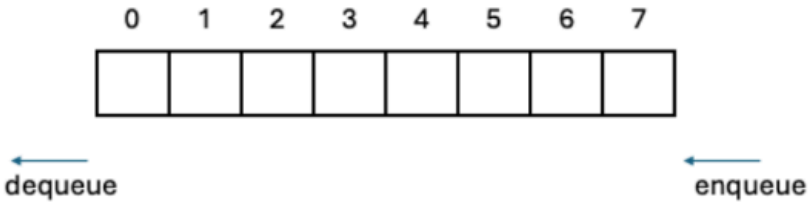
Queue

一、核心概念

一種線性資料結構，其運作遵循 **FIFO（First-In, First-Out，先進先出）**原則。

- 物理特性：具有兩個開口，一個用於進入（Input），另一個用於退出（Output），且兩者不同端。

- 生活比喻：就像結帳櫃檯的排隊人龍，先排隊的人先完成結帳離開。



二、 基本操作 (Preliminary Operations)

時間複雜度通常皆為O(1) :

操作	描述	發生位置
enqueue() / AddQ()	將新元素新增至佇列	Rear(末端)
dequeue() / DeleteQ()	將最久之前的元素移除並回傳	Front(前端)
front()	讀取最前端元素但不移除	Front(前端)
isEmpty()	檢查佇列是否為空	
isFull()	檢查佇列是否已達最大容量	

三、 兩種實現方式對比

1. 陣列實作：環狀佇列 (Circular Queue)

為了避免空間浪費（當 front 之前的空間被釋放卻無法再利用），通常使用環狀緩衝（Circular Buffer）邏輯。

- 關鍵變數：queue[MAX_SIZE]、front、rear。
- 模組運算 (Modulo Arithmetic)：利用 % 運算子讓指標在陣列末端時能「繞回」到開頭。
  - Enqueue: rear = (rear + 1) % MAX_SIZE;
  - Dequeue: front = (front + 1) % MAX_SIZE;
- 邊界條件判定：

- Empty（空）: front == rear
- Full（滿）: (rear + 1) % MAX_SIZE == front

2. 鏈結串列實作 (Linked List-based Queue)

- 優勢：動態分配記憶體，除非記憶體耗盡否則不會溢位（Overflow）。
- 指標管理：需要兩個指標 front（指向第一個節點）與 rear（指向最後一個節點）。
- 特殊情況：當 dequeue 移除最後一個節點後，佇列變空，必須將 rear 設為 NULL 。

四、 重點筆記

1. 空間利用率

- 在環狀佇列中，若使用指標判定空/滿，通常會犧牲一個儲存單元來區分 front == rear 的狀態 。
- 如果要用滿所有空間，則需要額外的一個 count 變數來記錄當前元素數量 。

2. 常見應用場景 (Real-world Applications)

「哪些情況會用到 Queue？」：

- **任務調度 (Task Scheduling)：**作業系統處理 CPU 任務順序
- **資源共享：**如印表機佇列（Printer Queues）
- **緩衝處理 (Buffering)：**I/O 數據流傳輸

3. Stack v.s. Queue

- Stack: LIFO（後進先出）
- Queue: FIFO（先進先出）

五、 靈活運用練習

Q：在環狀佇列中，如果 MAX_SIZE = 8，目前 front = 7，rear = 7。
請問此時狀態為何？若執行 Enqueue 存入一個數字 10，新的 rear 會是多少？

解析：

1. 因為 front == rear，此時佇列為空 (Empty)。
2. 執行 Enqueue 時，rear = (7 + 1) % 8 = 0。
3. 數字 10 會被存放在 queue[0] 的位置 。