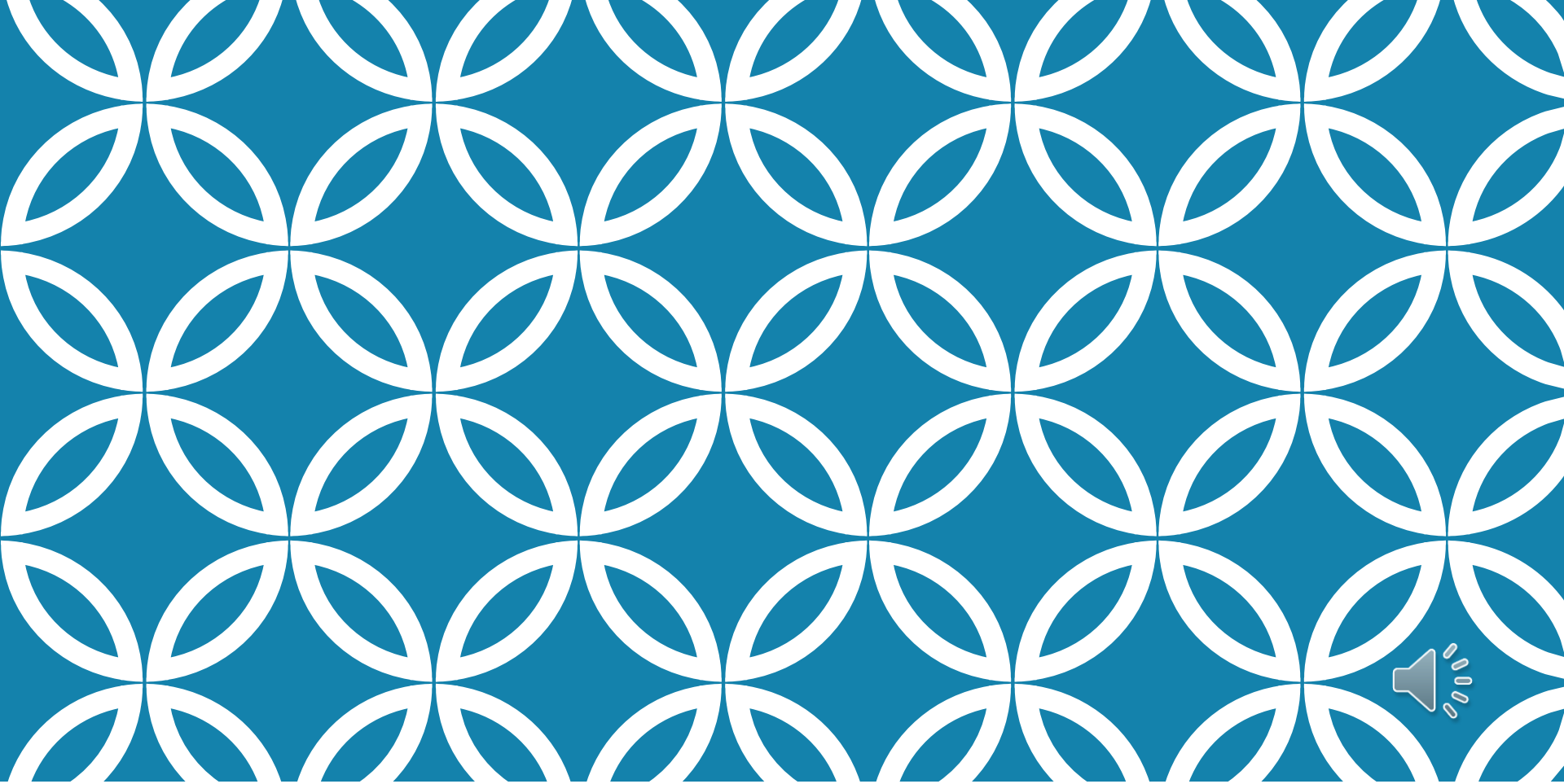


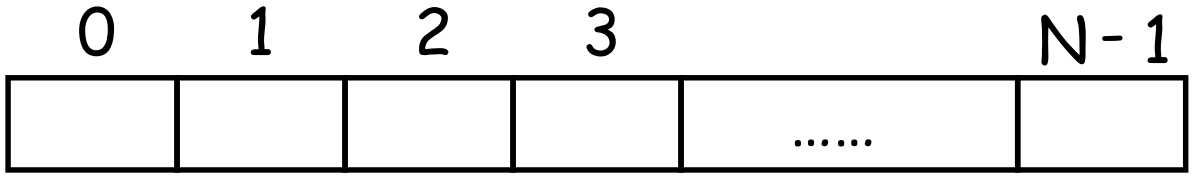
# 以陣列製作環狀佇列

資料結構
鍾宜玲

# 以陣列製作環狀佇列(CIRCULAR QUEUE)

假設放入佇列的資料為整數，且佇列最大容量是100，則陣列可以宣告如下：

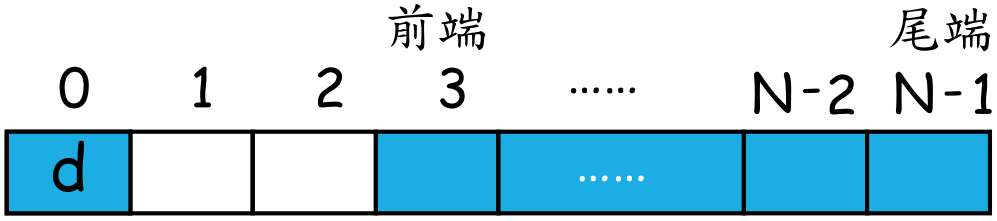
```
#define  N  100      /*  佇列大小 */
int queue [N];       /* 陣列queue當作佇列  */
```

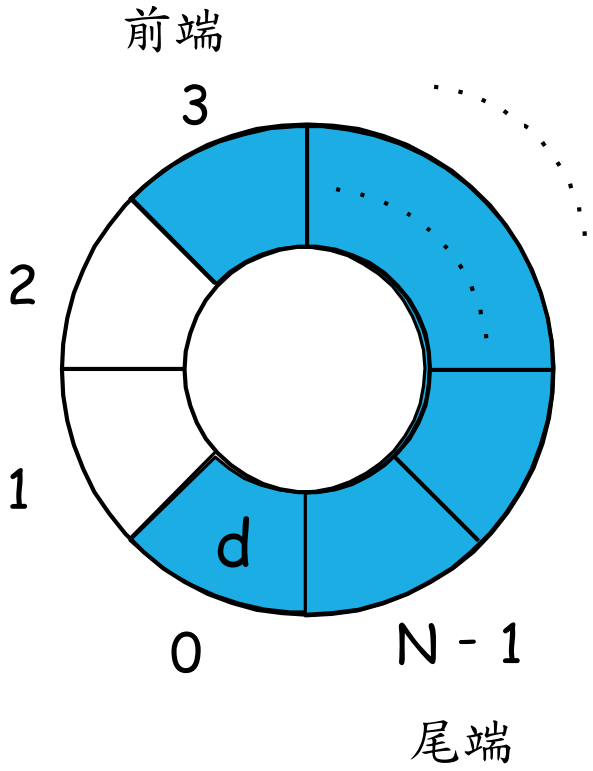


# 於環狀佇列中加入資料

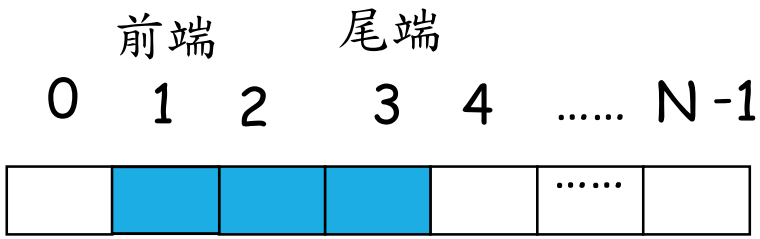
當尾端為 N-1 時，而位置0 處空著時，

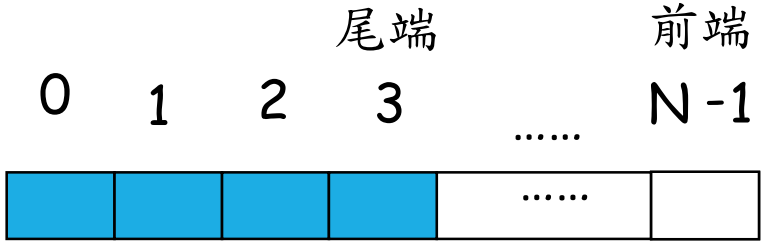
將資料加入 queue[0] 處。

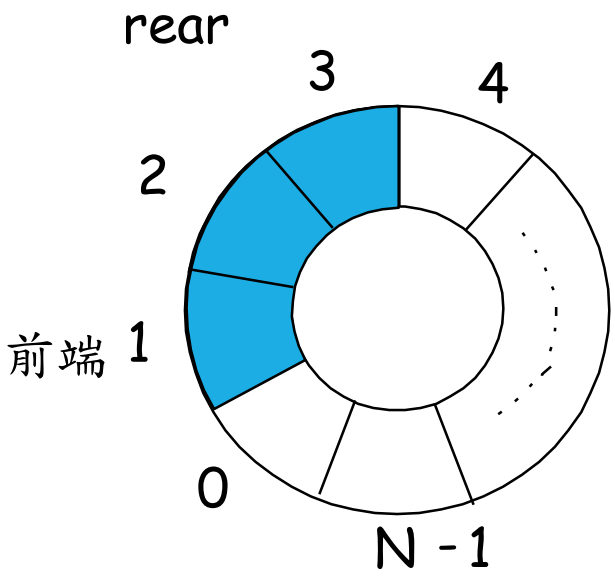


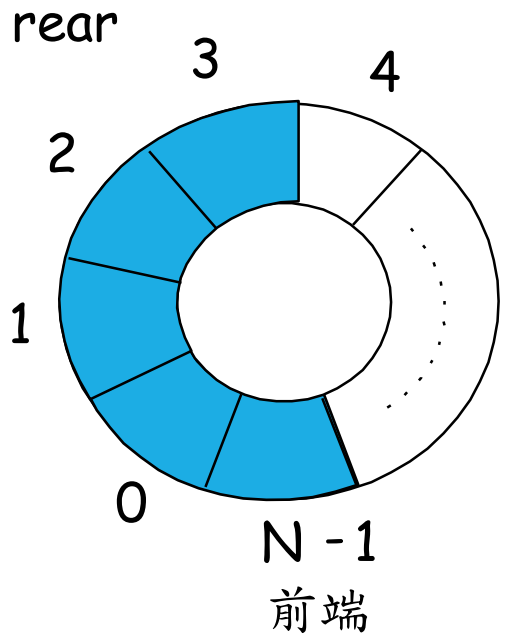



# 於環狀佇列中刪除資料



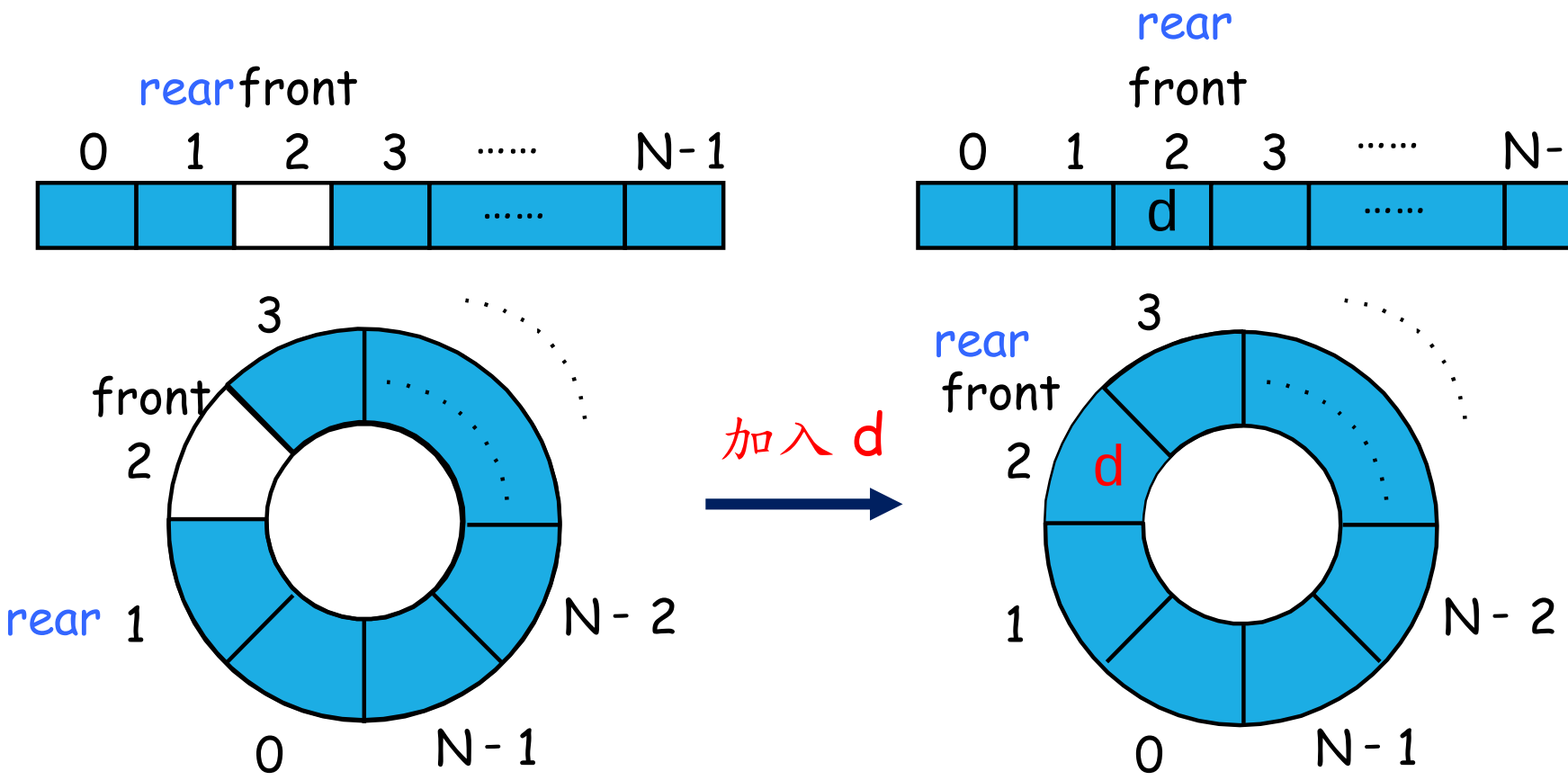



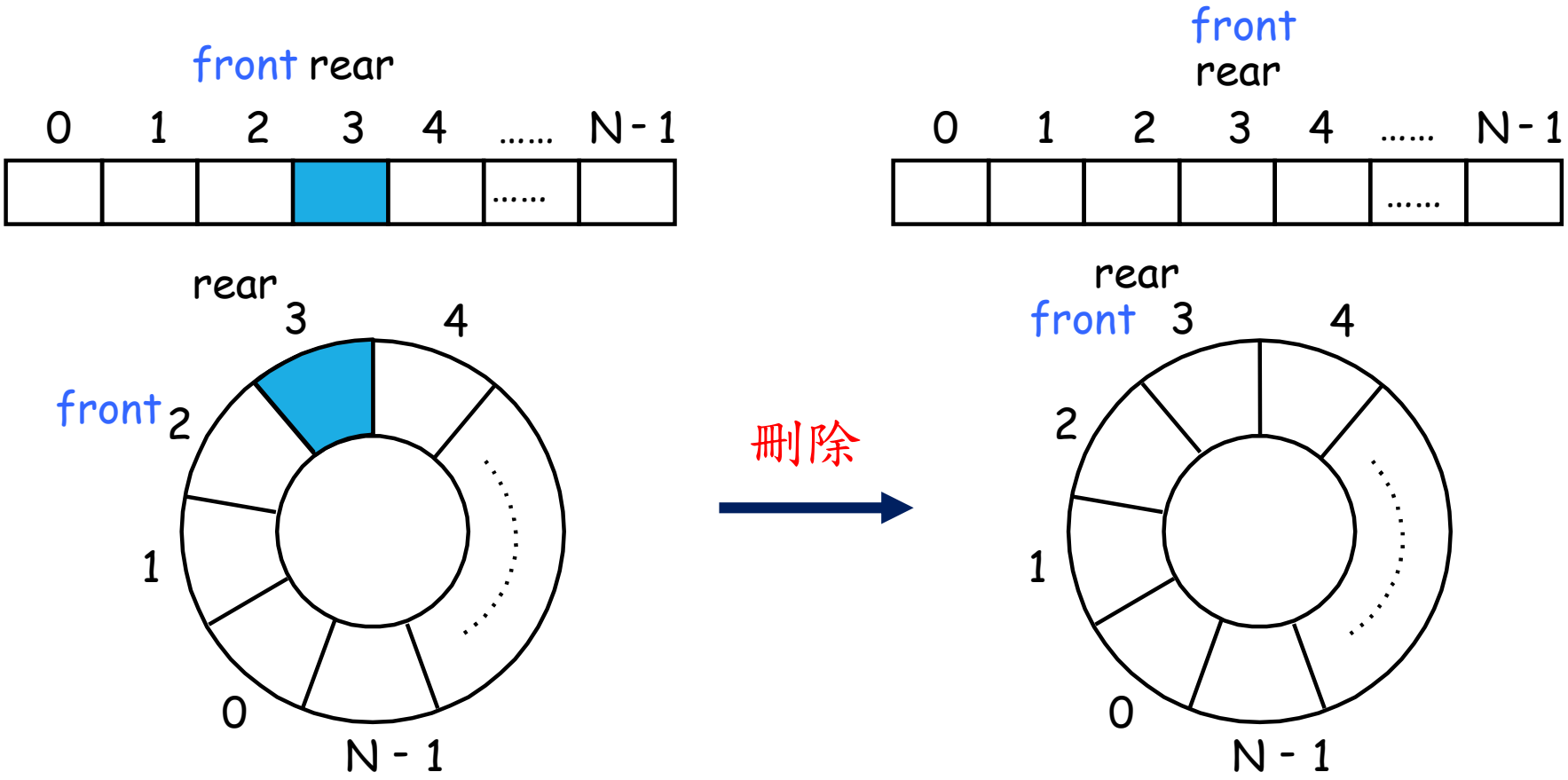
# 陣列製作佇列要點


- 佇列視為一個環，即環狀佇列。
- 需紀錄前端的**前一個位置**，例如使用 front 表示。
- 需紀錄尾端的位置，例如使用 rear 表示。
- 當僅剩最後一個空間時，視為環狀佇列已滿，不能再加入資料。

# 環狀佇列何時已滿？


rear
front
0 1 2 3 ..... N-1
.....
front
3
2
rear 1
0
N-2
N-1
加入 d
rear
front
d
3
2
1
0
N-2
N-1

# 環狀佇列何時已空？


front rear
0 1 2 3 4 ..... N-1
.....
front
rear
0 1 2 3 4 ..... N-1
.....
rear
front
3
4
2
1
0
N-1
刪除
rear
front 3
4
2
1
0
N-1

# 陣列製作環狀佇列的問題

環狀佇列滿時

- (front == rear) 成立
- 不能再加入資料

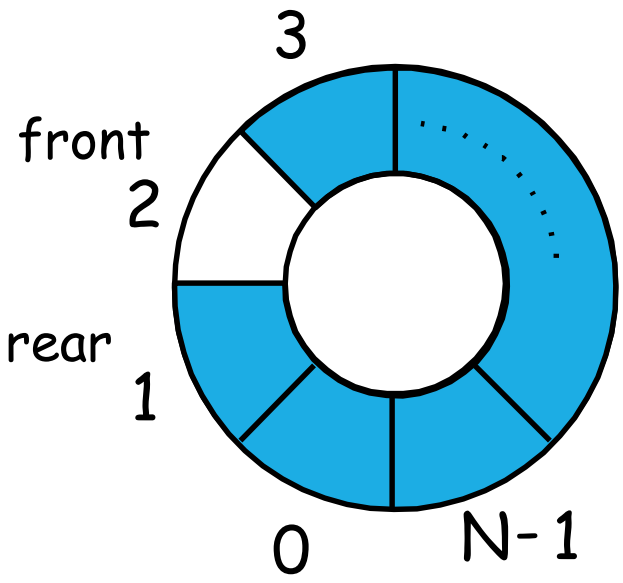
環狀佇列空時

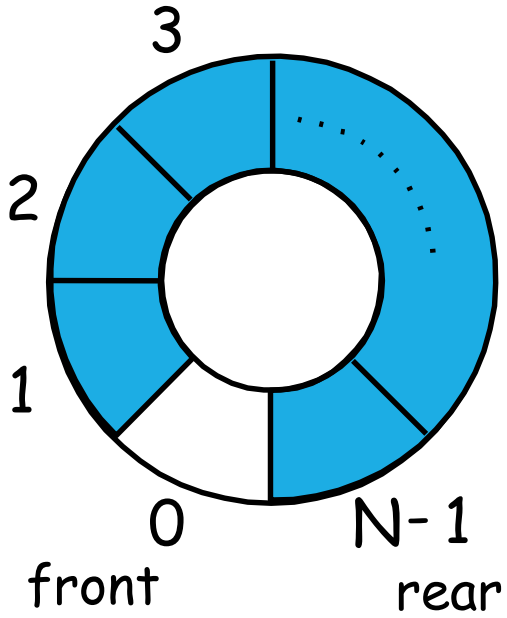
- (front == rear) 成立
- 不能刪除資料

當(front == rear) 成立時，佇列是滿的還是空的？

下列為兩種環狀佇列視為已滿的情形。

此時 (rear+1)%N == front 成立



# 陣列製作環狀佇列

環狀佇列已滿 ⇔

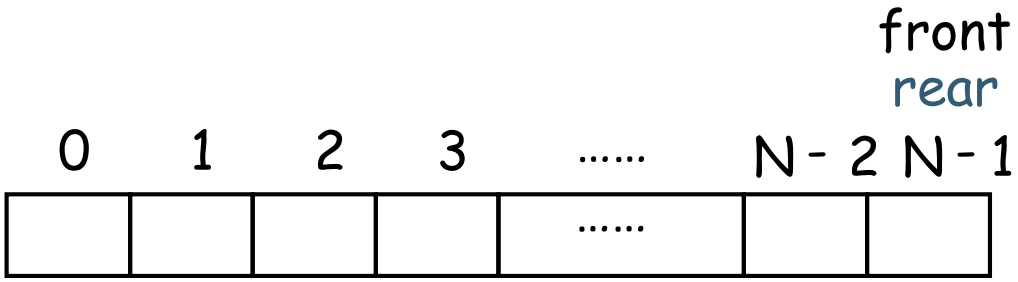
( (rear+1) % N == front ) 成立

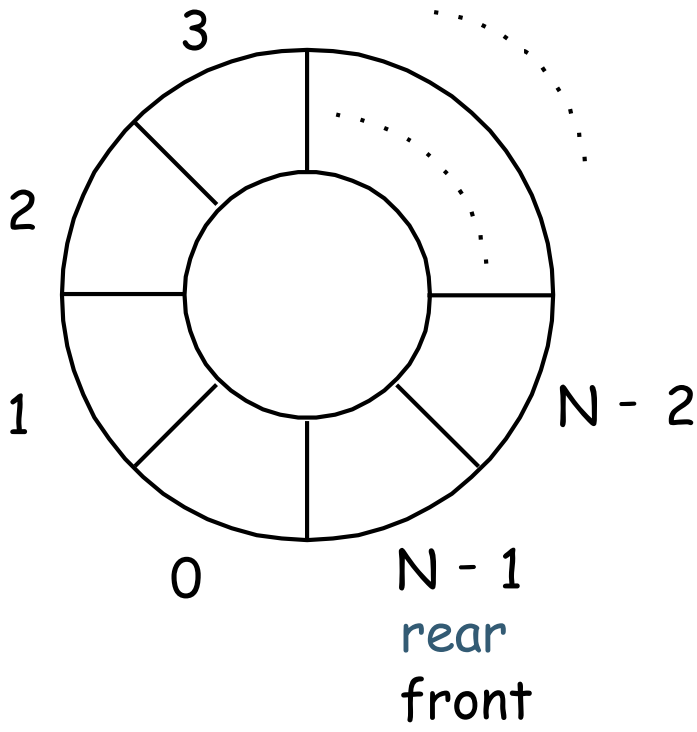
環狀佇列已空 ⇔

( rear == front ) 成立

# 使用陣列製作環狀佇列

```
#define N 100
int queue[N];
int front = N-1, rear = N-1;
```



# 加入資料(ADD TO A CIRCULAR QUEUE)

```
void add(int d)
{
  if ( front == (rear+1)%N ){
      printf("circular queue is full.\n");
    exit(1);
  }
  rear = (rear+1)%N;
  queue[rear] = d;
}
```

檢查環狀佇列是否滿了

否則rear向後移一格，
新資料加入rear位置內。

# 刪除資料(DELETE FROM A CIRCULAR QUEUE)

front變數表示真正前端元素的前一個位置

```
int delete ( )
{
  if ( front == rear ) {
      printf ("Circular queue is empty.\n");
   exit(1);
  }
  front = (front+1)%N;
  return queue[front];
}
```

檢查環狀佇列是否空了

front向後移一格，
取出front位置之資料