佇列(Queue)

本章學習目標

1. 讓讀者了解日常生活有許多例子都是佇列的應用。
2. 說明佇列的運作原理。

本章內容

本章重點整理

課後評量

5-1 佇列

佇列(Queue)是一種先進先出（First In First Out, FIFO）的有序串列，它與堆疊處理資料方式是不大一樣的，亦即資料處理是在不同邊進行，也就是資料由一端加入，由另一端刪除。因此，日常生活中，也有一些佇列的例子，如排隊上公車、排隊買電影票或是超市排隊付帳等，皆是先到達的先處理、後到的後處理。如圖 5-1 所示。

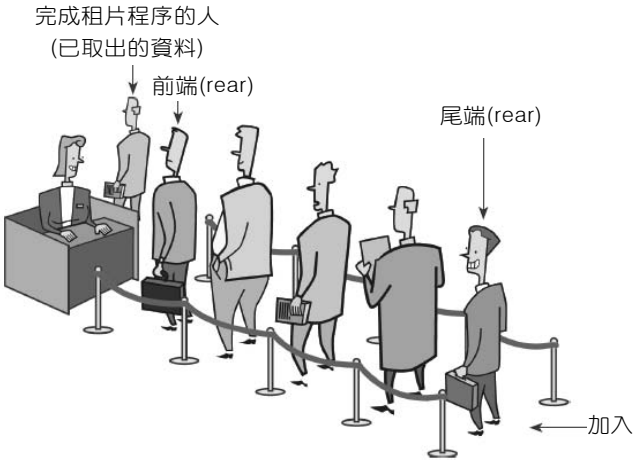


圖 5-1 佇列的例子

定 義

1. 一群相同性質元素的組合，即有序串列(Ordered List) 。
2. 具有 FIFO(First In First Out) 先進先出的性質。
3. 加入元素的動作發生在 Rear(後)端。
4. 刪除元素的動作發生在 Front(前)端。
5. Add/Delete 的動作皆發生在不同兩端。如圖 5-2 所示。

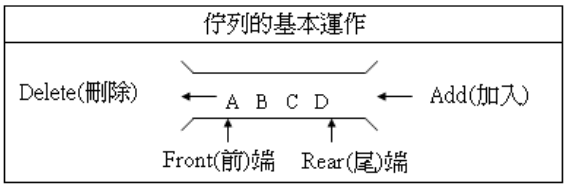


圖 5-2 佇列的基本運作

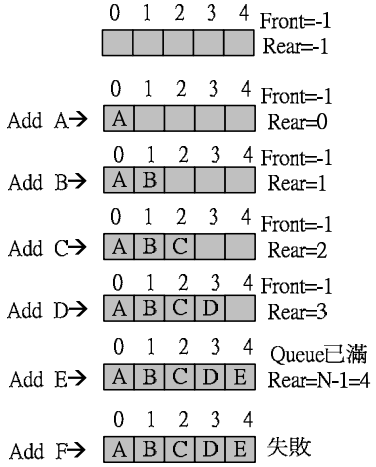
佇列常用專有名詞

1. Add(或 Enqueue)：由佇列的後端 (Rear) 加入一個新項目。
2. Delete(或 Dequeue)：從佇列的前端 (Front) 刪除一個項目。
3. IsFull：判斷佇列是否已滿，若已滿為真(True)，否則為假(False)。
4. IsEmpty：判斷佇列是否為空，若空為真(True)，否則為假(False)。

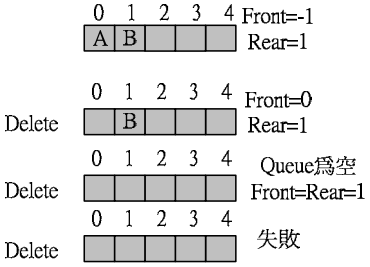
演算法

	插入元素 Add		刪除元素 Delete
01	`Procedure Add(item,Queue)`	01	`Procedure Delete(item, Queue)`
02	`begin`	02	`begin`
03	`  if (Rear=N-1)`	03	`  if (Front=Rear)`
04	`      Queue Is Full;  //Queue 已滿`	04	`Queue Is Empty;  //Queue 為空`
05	`  else`	05	`  else`
06	`   {`	06	`   {`
07	`     Rear=Rear+1;`	07	`    Front=Front+1;`
08	`     Queue[Rear]=item;`	08	`    item=Queue[Front];`
09	`   }`	09	`   }`
10	`end`	10	`End`

運作過程：



運作過程：

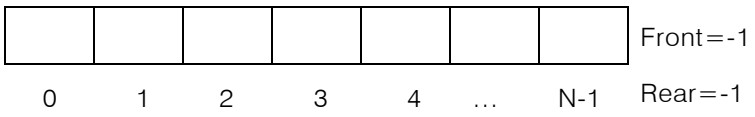


舉 例 假設有一個空的 Queue，實施下列的動作：

```
Add     A
Add     B
Add     C
Delete
Delete
Add     D
```

請呈現以上佇列運作的過程。

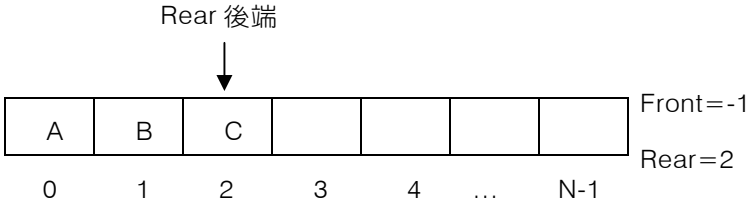
解 答 ❶ 空的 Queue



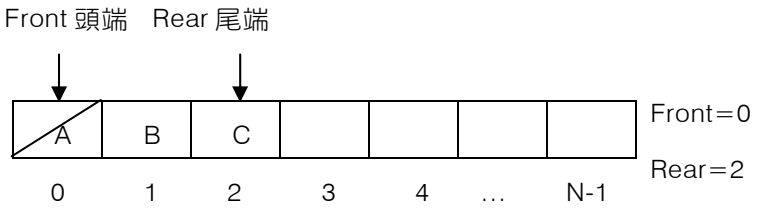
❷ Add A 到 Queue

Add B 到 Queue

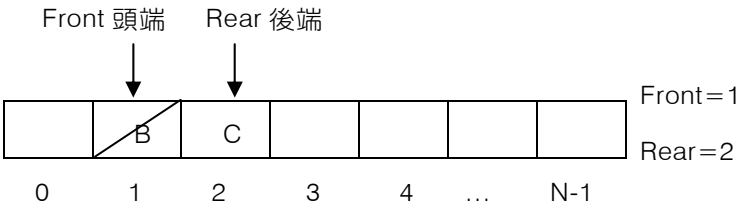
Add C 到 Queue



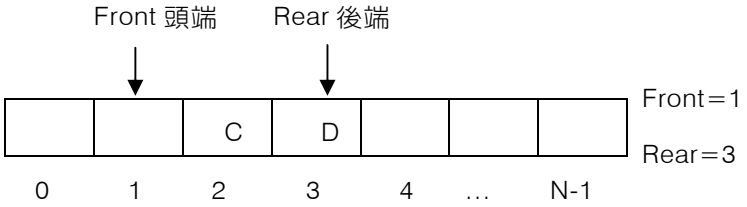
❸ Delete 從 Queue



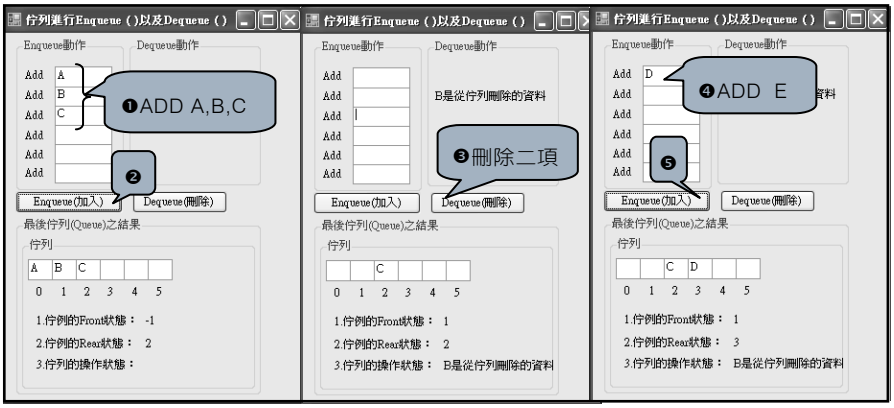
❹ Delete 從 Queue



❺ Add D 到 Queue



檔案在附書光碟中。



佇列的應用

佇列經常應用在電腦中，一般作業系統內均有一個工作佇列(job queue)，若系統不考慮工作的優先權時，則工作將依進入系統的先後順序來處理，亦即先到的工作先處理。例如：我們使用的印表機印表的工作也是採用「佇列」的特性來處理，其作法就是先送來的資料先印列，如果有兩個或兩個以上的資料，則將它們先放到工作排程佇列中等待列印。

1.作業系統的工作排程。	4.I/O 的緩衝區(Buffer)上。
2.計算機的模擬程式。	5.Priority Queue
3. 磁碟管理的輸出入(input/output)排序。	6.圖形的 BFS 廣度追蹤

例如：作業系統的工作排程。如圖 5-3 所示。

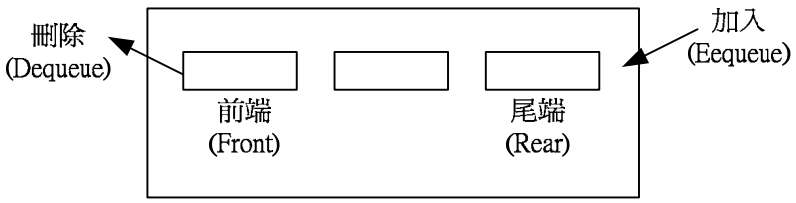


圖 5-3 作業系統的工作排程

5-2 以陣列來製作佇列

製作佇列最常用的方法就是利用一維陣列來完成。其製作的過程如下：

1. 產生佇列結構：

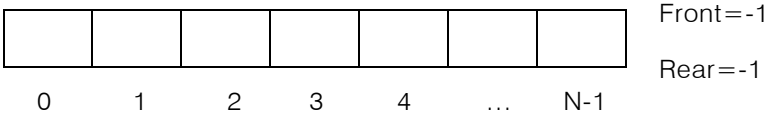
❶ 宣告一個陣列結構。

❷ 方法：Create(Queue)　//指建立一個空的 Queue

❸ 製作：利用 VB 語言

```
01  Dim N As Integer = 10              '定義佇列大小
02  Dim Queue(N) As String             '宣告及建立佇列
03  Dim Rear As Integer = -1           '宣告佇列的尾端
04  Dim Front As Integer = -1          '宣告佇列的前端
```

❹ 陣列的記憶體配置(空的 Queue)



2. 將資料加入佇列(Add 動作)：

❶ 將資料(item)加入到 Queue 中；成為 Rear 端元素。如果佇列已滿，則無法進行。

❷ 方法：Add(item, Queue)

```
01  Procedure Add(item, Queue)
02  begin
03    if (Rear=N-1)
04      Queue Is Full;  //Queue 已滿
05    else
06     {
07       Rear=Rear+1;
08       Queue[Rear]=item;
09     }
10  end
```

舉 例 假設有一個空的 Queue，實施下列的動作：

Add　A 到 Queue

Add　B 到 Queue

Add　C 到 Queue

請呈現以上佇列運作之後的 Front 與 Rear 之值。

解 答

3. 刪除資料(Delete 動作)：

❶ 指刪除 Queue 的 Front 頭端元素，如果 Queue 是空，則無法進行。

❷ 方法：Delete(item,Queue)

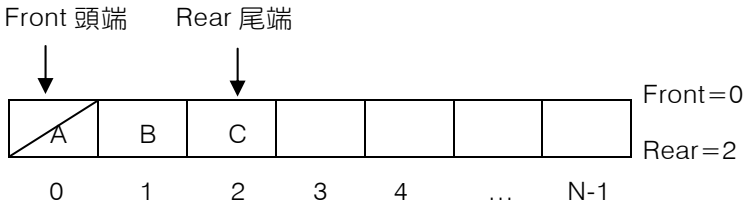
```
01  Procedure Delete(item, Queue)
02  begin
03    if (Front=Rear)
04  Queue Is Empty;  //Queue 為空
05    else
06     {
07      Front=Front+1;
08      item=Queue[Front];
09     }
10  end
```

舉 例 假設有一個 Queue，已經有 A,B,C 三個資料項，如果再實施下列的動作：

Delete 從 Queue

請呈現以上佇列運作之後的 Front 與 Rear 之值。

解 答



4. 判斷佇列是否已滿：

❶ 若使用陣列時，判斷 Queue 是否已滿(亦即 Rear 是否等於 N-1)，若是則傳回 True，否則傳回 False。

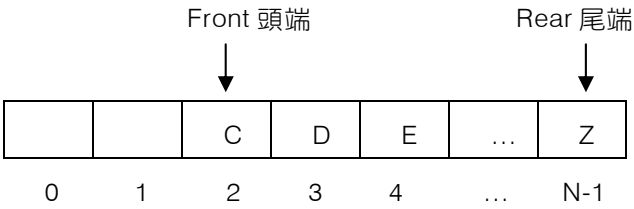
❷ 方法：IsFull(Queue)

判斷佇列是否已滿

```
01  Procedure IsFull(Queue)
02  begin
03    if (Rear=N-1)
04        return True;   //Queue 已滿
05    else
06       return False;  //Queue 尚未滿
07  end
```

缺　點　在佇列中，當 Rear=N-1 時，並不能保證 Queue 真的已滿。

分析原因



在上圖中，判斷佇列是否為滿的副程式 IsFull(Queue)，必須要額外增加一個判斷條件為：if (Front=-1)，否則像上圖中，尚有 2 個空間，而結果卻為滿。

判斷佇列是否為滿的副程式

```
01  Procedure IsFull(Queue)
02  begin
03    if (Rear=N-1) And (Front=-1)
04       return True;    //Queue 已滿
05    else
06       return False;   //Queue 尚未滿
07  end
```

5. 判斷佇列是否為空：

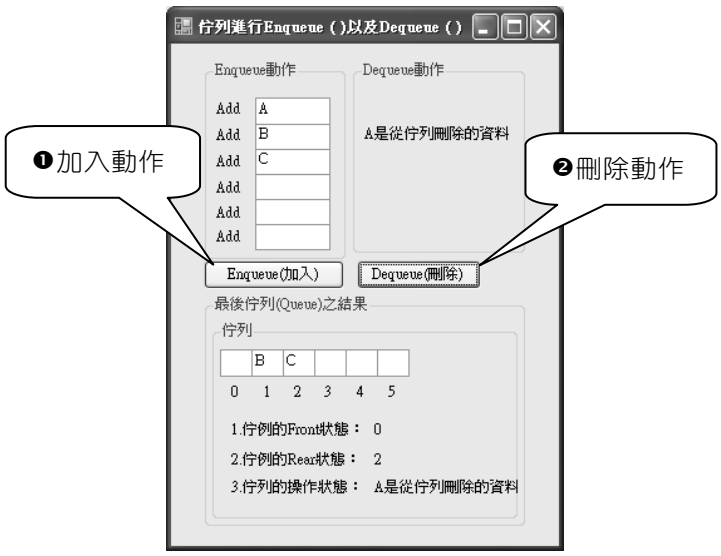
❶ 若使用陣列時，判斷 Queue 是否是空(亦即 Front=Rear)，若是則傳回 True，否則傳回 False。

❷ 方法：IsEmpty(Queue)

判斷佇列是否為空

```
01  Procedure IsEmpty (Queue)
02  begin
03    if (Front=Rear)
04       return True;    // Queue 為空
05    else
06       return False;   // Queue 不為空
07  end
```

實 例 請實作 Queue 佇列，可以讓使用者加入或刪除資料。



解 答 在附書光碟中。

5-3 環形佇列(circular queue)

由於佇列有一個問題，就是前端(Front)尚有空位時，卻再加入元素時，發現此佇列已滿。解決方法：使用環形佇列(circular queue)。如圖 5-4 所示。

定 義

1. 環狀佇列就是一種環形結構的佇列，它是利用一種 Q[0: N-1]的一維陣列，同時 Q[0]為 Q[N-1]的下一個元素。
2. 最多使用(N-1)個空間。

缺 點 佇列滿了，浪費一個空格沒有使用，若再加入一個項目時，Rear 等於 0，也就是 Front 等於 Rear，如此無法分辨此時佇列是空的還是滿的。因此，環形佇列必須浪費一個空格。當 Front 等於 Rear 時，環形佇列為空的。當(Rear+1) Mod N 等於 Front 時，環形佇列為已滿。

重要觀念

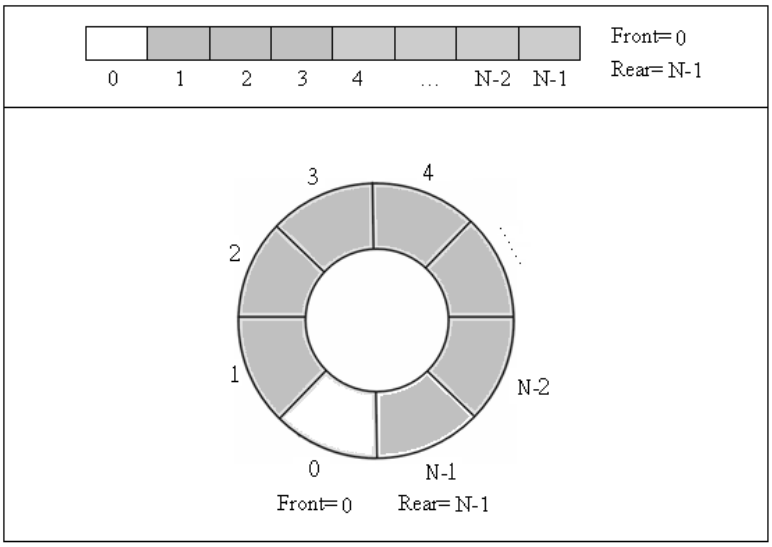


圖 5-4 環形佇列(circular queue)

5-3.1 環形佇列的基本運作功能

基本製作之方法：最多使用(n-1)個空間。

1. **產生環形佇列結構：**

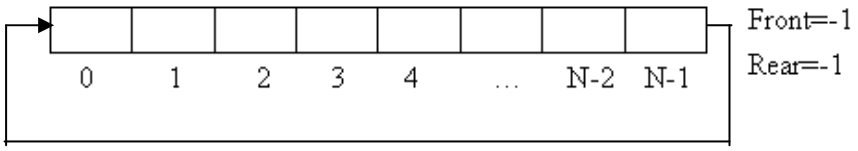
   ❶ 宣告一個陣列結構。

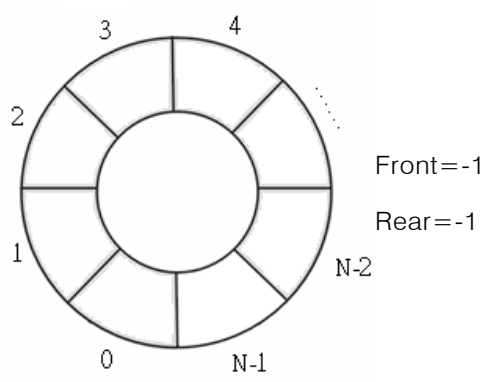
   ❷ 方法：Create(CQueue)　//指建立一個空的 Circular Queue

   ❸ 製作：利用 VB 語言

```
Dim N As Integer = 10          '定義環形佇列大小
Dim CQueue(N) As String        '宣告及建立環形佇列
Dim Rear As Integer = -1       '宣告環形佇列的尾端
Dim Front As Integer = -1      '宣告環形佇列的前端
```

❹ 陣列的記憶體配置(空的 Circular Queue)





2. 將資料放入環形佇列(Add 動作)：

❶ 將資料(item)加入到 Circular Queue 中。

❷ 方法：Add(item, CQueue)

❸ 製作：

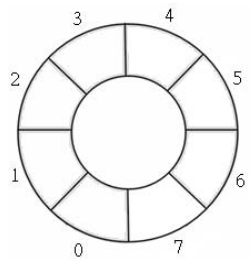
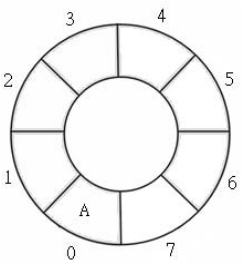
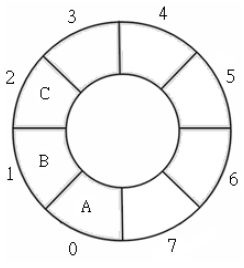
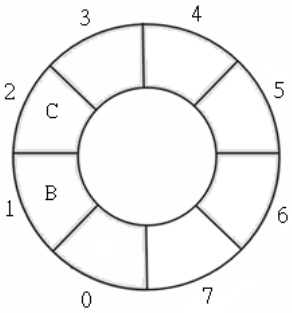
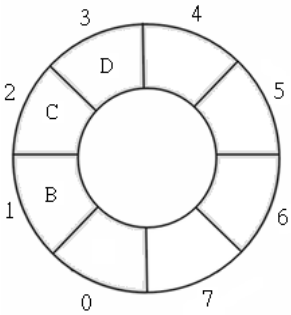
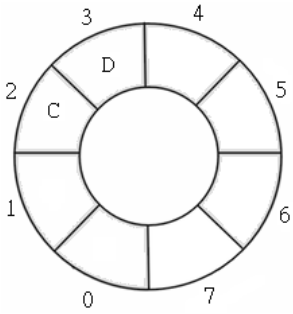
演算法	
01	Procedure Add(item,CQueue)
02	begin
03	if((Rear+1)Mod N=Front)
04	Circular Queue Is Full; //Circular Queue 已滿
05	else
06	{
07	CQueue[Rear]=item;
08	}
09	end

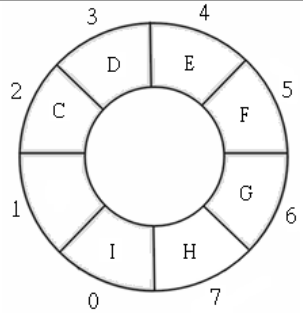
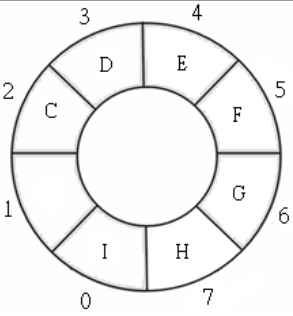
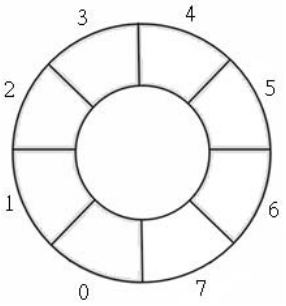
舉 例 假設有一個空的環形佇列 CQueue，實施下列的動作：

```
Add     A
Add     B,C
Delete
Add     D
Delete
Add     E,F,G,H,I
Add     J
Delete   Delete  Delete  Delete  Delete  Delete  Delete
```

請呈現以上佇列運作之後的 Front 與 Rear 之值。

解 答

 空佇列 Front=-1 Rear=-1	 加入 A Front=-1 Rear=0	 加入 B,C Front=-1 Rear=2
 刪除 Front=0 Rear=2	 加入 D Front=0 Rear=3	 刪除 Front=1 Rear=3

		
加入 E,F,G,H,I Front=1 Rear=0 佇列已滿	加入 J Front=1 Rear=1 失敗	刪除 7 次 Front=0 Rear=0 佇列為空

3. 刪除資料(Delete 動作)：

❶ 指刪除 Circular Queue 的元素。

❷ 方法：Delete(item,CQueue)

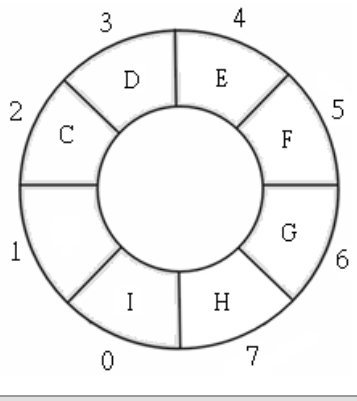
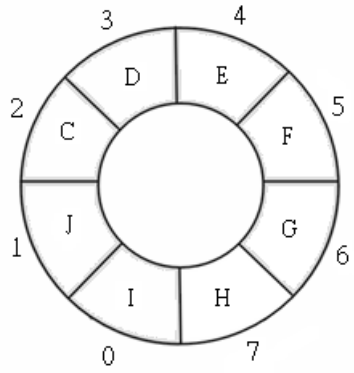
演算法	
01	Procedure Delete(item, CQueue)
02	begin
03	if (Front=Rear)
04	Circular Queue Is Empty; //Queue 為空
05	else
06	{
07	Front=(Front+1) Mod N
08	item=CQueue[Front];
09	}
10	end

❸ 分析：

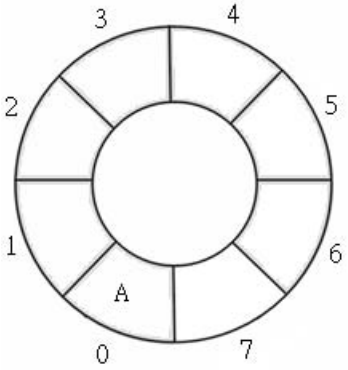
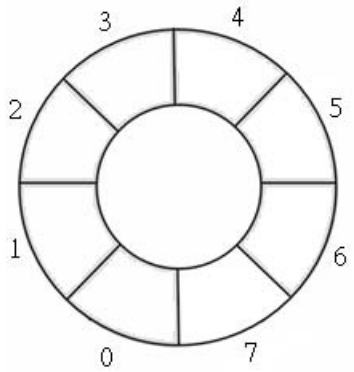
a. 若硬要使用 N 個空間，則 Rear=Front 條件式成立時，則無法區分出 Queue 是否真的 Full 或 Empty。如下圖所示：

b. 判斷 Queue 是否已滿或為空，則使用相同的條件式 (Rear=Front)。

c. 加入或刪除動作的時間複雜度均為 O(1)。

僅剩下一個空間
`Front=1`
`Rear=0`
	再加入一項資料 J 時，Rear=1
	使得 Front=Rear
	∴環形佇列已滿

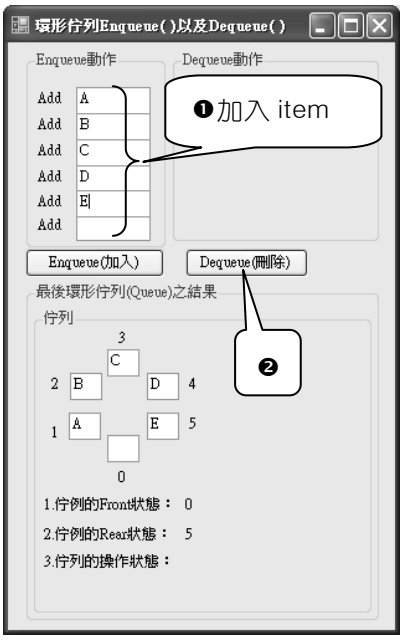
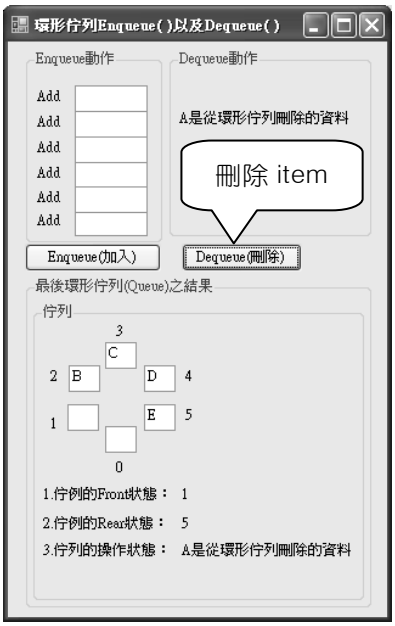
	

僅剩下一筆資料
`Front=1`
`Rear=0`
	再刪除一項資料 A 時，Front=0
	使得 Front=Rear
	∴環形佇列已空

表 5-7 一般佇列與環狀佇列之比較表

	一般佇列	環狀佇列
陣列宣告	q[n]	q[n]
陣列元素	q[0],q[1],…,q[n-1]	q[0],q[1],…,q[n-1]
加入資料的位置	rear=rear+1	rear = (rear+1) mod n
刪除資料的位置	front=front+1	front = (front+1) mod n
判斷是否為空佇列	front=rear	front=rear
判斷佇列是否滿溢	rear=n-1	front=(rear+1) mod n
佇列可儲存之元素個數	n	n-1
佇列中僅剩一個元素	rear=front+1	rear=(front+1) mod n

實 例 請撰寫一個環形佇列的程式。

1.加入 5 個資料項	2.刪除 A 資料項
	

解 答 在附書光碟中。

5-3.2 環形佇列的改良

改良版製作之方法：可以使用 N 個空間。

作法：另外再增加一個變數(Tag)來判斷，每一次的動作，例如：當每次加入資料時，將 Tag 設為 1，而刪除資料時設為 0。因此，在檢查 Front 是否等於 Rear 時，先再檢查 Tag 的狀態即可。

1. **產生環形佇列結構：**

   ❶ 宣告一個陣列結構。

   ❷ 方法：Create(Queue)　//指建立一個空的 Queue

   ❸ 製作：利用 VB 語言

```
01 Dim N As Integer = 10          '定義環形佇列大小
02 Dim CQueue(N) As String       '宣告及建立環形佇列
03 Dim Rear As Integer = -1      '宣告環形佇列的尾端
04 Dim Front As Integer = -1     '宣告環形佇列的前端
05 Dim bool As Integer Tag=0 '當 CQueue 為 Full 時,則 Tag=1,否則 Tag=0。
```

2. **將資料加入環形佇列(Add 動作)：**

   ❶ 將資料(item)加入到 Circular Queue 中。

   ❷ 方法：Add(item, CQueue)

```
01 Procedure Add(item,CQueue)
02 Begin
03   Rear=(Rear+1) Mod n
04   if (Rear=Front) And (Tag=1)
05 CircularQueue Is Full;  //Queue 已滿
06   else
07    {
08       CQueue[Rear]=item;
09      if(Rear=Front) then Tag=1;
10    }
11 End
```

3. **刪除資料(Delete 動作)：**

❶ 指刪除 Circular Queue 的元素。

❷ 方法：Delete(item,CQueue)

```
01 Procedure Delete(item, CQueue)
02 begin
03   if (Front=Rear) And (Tag=0)
04       Circular Queue Is Empty;  //Queue 為空
05   else
06    {
07      Front=(Front+1) Mod n
08         item=CQueue[Front];
09      if(Rear=Front) then Tag=0;
10    }
11 end
```

4. **分析：**

優 點 最多可以利用到 N 個空間。

缺 點 Add 與 Delete 動作均須額外多了一個條件判斷式。

5-4 進階佇列

除了前面所提到的一般佇列與環狀佇列之外，在某些情況之下，也會使用到比較特殊的佇列，例如：優先佇列(priority queue)及雙向佇列(double-ended queue: deque)。

5-4.1 優先佇列(priority queue)

在優先佇列中，其元素的加入及刪除不須要遵守先進先出的特性。這種情況在日常生活中，時常遇到。例如：醫院的急診室、捷運座位提供老幼婦孺者之仁愛座等等。

定 義 為一種不必遵守佇列特性－FIFO(先進先出)的有序串列。

每一個佇列的元素，都給予一個優先順序權，其數字最大者代表有最高優先權。優先佇列可以利用陣列結構及鏈結串列來解決。

缺 點 在經常需要大量資料異動時，無法預先知道需要保留多少尾部的空間給插入的工作使用。

優先佇列結構 (採用陣列實施）。如圖 5-5 所示。

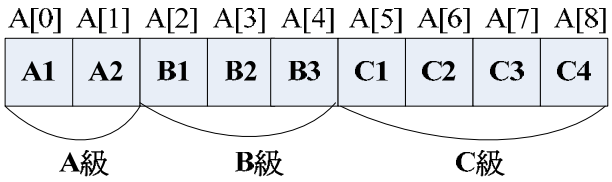


圖 5-5 優先佇列結構

其中：

A1：front_a，A 級佇列的首端

A2：rear_a ，A 級佇列的尾端

B1：front_b，B 級佇列的首端

B3：rear_b ，B 級佇列的尾端

C1：front_c，C 等級佇列的首端

C4：rear_c ，C 等級佇列的尾端

5-4.2 雙向佇列(double-ended queue：deque)

定 義 雙向佇列(Deque)是一種特殊的資料結構，它的兩端都可做加入與取出資料的動作，不像 Stack 的 LIFO 或 Queue 的 FIFO。亦即它是一種前後兩端都可輸入或取出資料的有序串列。如圖 5-6 所示。

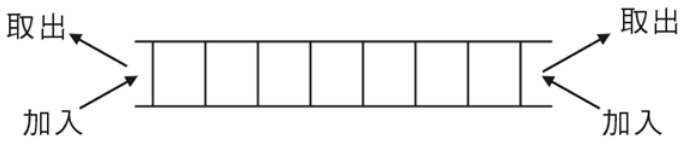


圖 5-6 雙向佇列(Deque)

舉 例 假設我們循序輸入一串數字：1,2,3,4,5,6,7，請問是否可以輸出 5174236 呢？

解 答 循序輸入一串數字：1,2,3,4,5,6,7

順序如下：

1	2	3	4	6	7	5

輸出：先輸出 5，再輸出 1，又輸出 7，但是，下一個輸出不是"2"就是"6"，無法輸出"4"，因此，在循序輸入 1,2,3,4,5,6,7 的情況之下，無法得到 5174236 的輸出結果。

本章重點整理

- 佇列(Queue)：是一種先進先出（First In First Out, FIFO）的有序串列。例子：如排隊上公車、排隊買電影票。
- 【佇列(Queue)的定義】
  1. 一群相同性質元素的組合，即有序串列(ordered List) 。
  2. 具有 FIFO(First In First Out) 先進先出的性質。
  3. 加入元素的動作發生在 Rear(尾)端。
  4. 刪除元素的動作發生在 Front(前)端。
  5. Add/Delete 的動作皆發生在不同一端。
- 【佇列的應用】

1.作業系統的工作排程。	4.I/O 的緩衝區(Buffer)上。
2.計算機的模擬程式。	5.Priority Queue
3.磁碟管理的輸出入(input/output)排序。	6.圖形的 BFS 廣度追蹤

- 【環形佇列的定義】
  1. 環狀佇列就是一種環形結構的佇列，它是以一種 Q(0: N-1)的一維陣列，同時 Q(0)為 Q(N-1)的下一個元素。
  2. Front(前端)永遠是以逆時鐘方向指向佇列中第一個元素的前一個位置，Rear(尾端)則指向佇列目前的最後位置。
  3. 最多使用(N-1)個空間。
- 優先佇列(priority queue)：在優先佇列中，其元素的加入及刪除不需要遵守先進先出的特性。例如：醫院的急診室。

- 【優先佇列的定義】為一種不必遵守佇列特性－FIFO(先進先出)的有序串列。
  1. 插入任意鍵值的動作。
  2. 刪除最大或最小鍵值動作。
  3. 當各元素以輸入先後次序為優先權時，就是一般的佇列。
  4. 如是以輸入先後次序做為最不優先權時，此優先佇列即為一堆疊。
- 雙向佇列(double-ended queue: dequeue)：它的兩端都可做置入與取出資料的動作，不像 Stack 的 FILO 或 Queue 的 FIFO。亦即它是一種前後兩端都可輸入或取出資料的有序串列。

課後評量

一、基本題《題庫來源：四技、研究所及高普考》

1. 請詳細寫出「佇列」的定義為何？及其應用實例？至少列出 5 項。
2. 假設要實作「佇列」時須考慮哪些基本操作(Operation)？至少列出 5 項。
3. 試比較「堆疊」和「佇列」在應用上之差異。
4. 請寫出 Add 插入元素到佇列的演算法。
5. 請寫出從佇列 Delete 刪除元素的演算法。
6. 假設有一個空的 Queue，實施下列的動作：
   Add　A 到 Queue
   Add　B 到 Queue
   Add　C 到 Queue
   Add　D 到 Queue
   Delete

   請呈現以上佇列運作之後的 Front 與 Rear 之值。
7. 當一開始狀態為 front=rear=-1 的環狀佇列，若加入 5(分別為 A、B、C、D、E)個資料，刪除 3 個資料，最後 front、rear 分別為若干？

二、進階題

1. 若以陣列來實作「佇列」，如何判斷「佇列滿溢」及「佇列空了」？
2. 若以陣列來實作「環狀佇列」，如何判斷「佇列滿溢」及「佇列空了」？新增資料、刪除資料之位置為何？
3. 試說明將 1,2,3,4 四個數字依此一次序分別經由堆疊(stack)與佇例(queue)做排列，問各有多少種不同的排列方式？請也分別寫出這些排列的結果。
4. 如果 MAX_ITEM=10，請畫出環狀佇列變化的情形，每個變化都必須標出 front 和 rear 的值。Enqueue(A)、Enqueue(B)、Enqueue(C)、Dequeue( )、Enqueue(D)、Dequeue( )、Dequeue( )、Enqueue(E)
5. 利用雙向佇列(deque)循序輸入 1、2、3、4、5、6、7，試問是否能夠得到 5174236 的輸出排列？並說明其過程或理由。