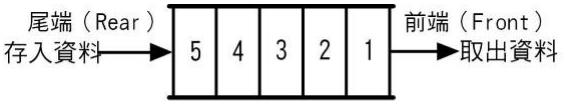
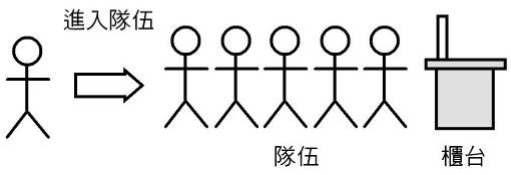
資料結構的佇列(Queues)

資訊科技系
林偉川

佇列的基礎

- 「佇列」（Queues）是一種和堆疊十分相似的資料結構，在日常生活中隨處可見的排隊人潮，例如：在郵局排隊寄信、銀行排隊存錢或電影院前排隊買票的隊伍，其組成的線性串列就是一種佇列，如下圖所示：

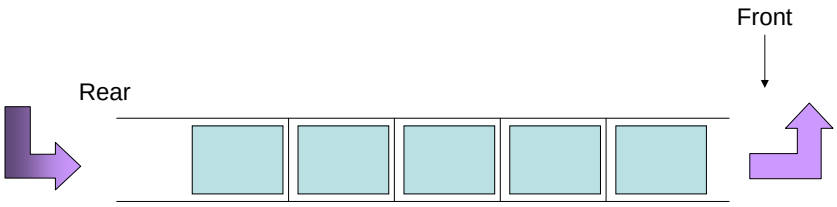




2

佇列(Queue)

- 佇列的定義
- 佇列的動作
- FIFO (First In First Out)



3

佇列的基礎

- 排隊的隊伍是在尾端（Rear）加入隊伍，如同佇列在尾端存入資料，當前端（Front）寄完信、存完錢或買完票後，人就離開隊伍，如同佇列從前端取出資料，所以佇列的基本操作，如下所示：
  - **dequeue()**：從佇列取出資料，每執行一次，就從前端取出一個資料。
  - **enqueue()**：在尾端將資料存入佇列。
  - **isQueueEmpty()**：檢查佇列是否是空的，以便判斷是否還有資料可以取出。
  - **isQueueFull()**：檢查佇列是否是滿了，以便判斷是否還可以存入資料。

4

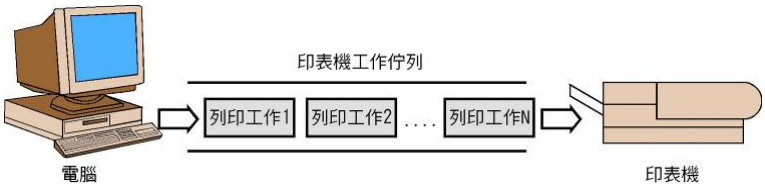
佇列的基礎-特性

- 佇列的資料因為是從尾端一一存入，佇列的內容是依序執行enqueue(1)、enqueue(2)、enqueue(3)、enqueue(4)和enqueue(5)的結果，接著從佇列依序執行dequeue()取出佇列資料，如下所示：
  dequeue()：1 dequeue()：2 dequeue()：3
  dequeue()：4 dequeue()：5
- 上述取出的順序是1、2、3、4、5，和存入時完全相同，稱為「先進先出」(First In, First Out)特性。總之，佇列擁有的特性，如下：
  - 從佇列的尾端存入資料，從前端讀取資料。
  - 資料存取的順序是先進先出，也就是先存入佇列的資料，先行取出。

5

佇列的應用

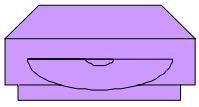
- 以計算機科學來說，佇列的主要用途是作為資料緩衝區，例如：因為電腦周邊設備的處理速度遠不如CPU，所以印表機列印報表時，需要使用佇列作為資料暫存的緩衝區，如下圖所示：



6

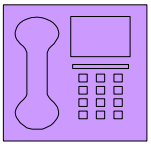
佇列的應用

- Buffer:
- 寫入磁碟的資料先儲存在電腦的記憶體緩衝區中，待緩衝區的資料到達一定的數量後,再寫入磁碟中，因為電腦的記憶體的速度比磁碟機快，如此連續寫入資料比分段寫入資料更能節省時間。



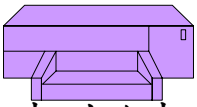
7

- Communication:



- 數據機再傳輸時就是採取佇列的方式,先到的資料先傳送,來不及傳送的資料就先暫存在 Queue 中,等到線路空出來時再繼續傳送.

- Printer Queue:

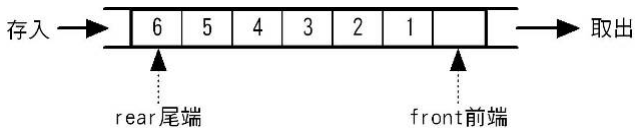


- 印表機連續列印不同資料時,先到的先列印,未列印的資料依序排在 Queue 中等待列印.

8

使用陣列建立佇列

```
#define MAXQUEUE 10      /* 佇列的最大容量 */
int queue[MAXQUEUE];     /* 佇列的陣列宣告 */
int front = -1;          /* 佇列的前端 */
int rear = -1;           /* 佇列的尾端 */
extern int isQueueEmpty();
extern int enqueue(int d);
extern int dequeue();
```



9

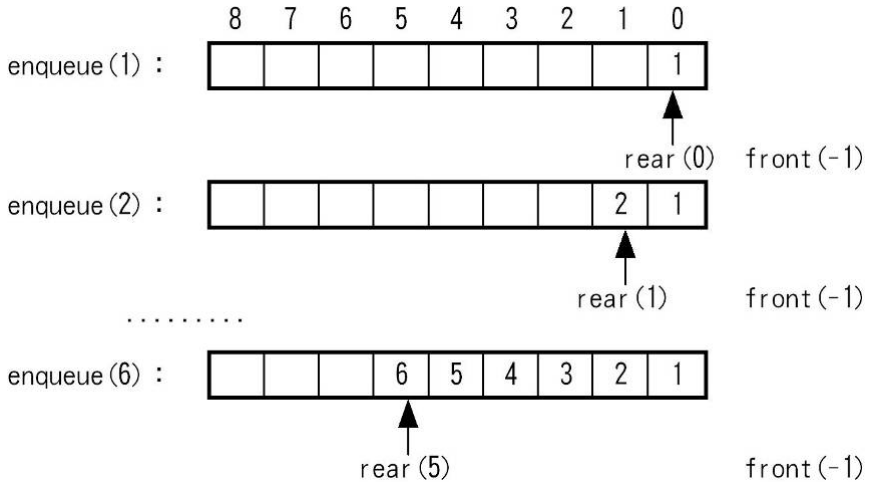
使用陣列建立佇列-存入元素

- 函數enqueue()是將資料存入佇列的rear尾端，其步驟如下所示：
  - Step 1：將尾端指標rear往前移動，也就是將指標rear加1。
  - Step 2：將值存入尾端指標rear所指的陣列元素。

```
queue[++rear] = d;
```

10

使用陣列建立佇列-存入元素



11

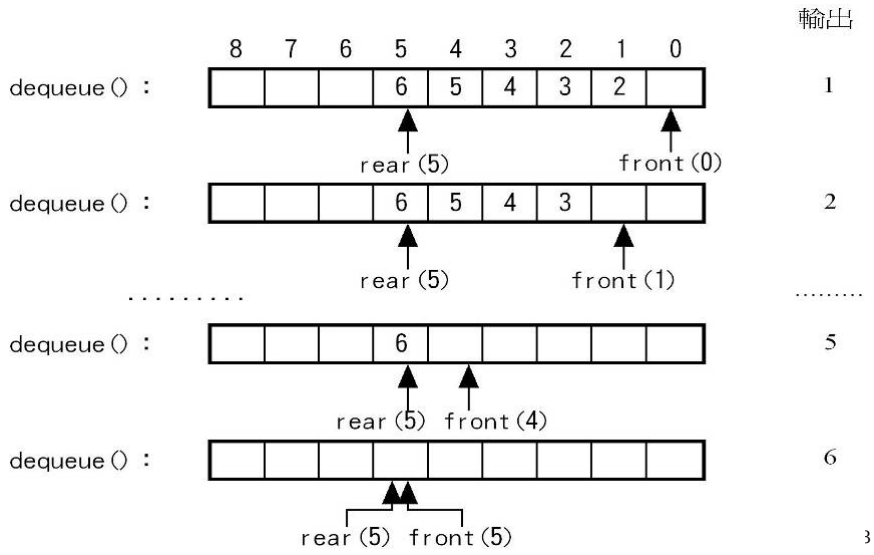
使用陣列建立佇列-取出元素

- 函數dequeue()是從佇列的front前端取出資料，其步驟如下所示：
  - Step 1：檢查佇列是否已空，如果沒有：
  - Step 2：將前端指標front往前移，即把其值加1。
  - Step 3：取出前端指標front所指的陣列元素。

    return queue[++front];

12

使用陣列建立佇列-取出元素



3

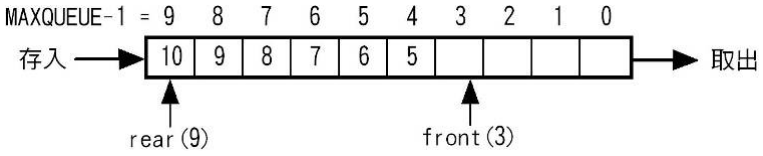
使用陣列建立佇列-佇列是否已空

- 在取出資料5後，指標front(4)與佇列rear指標(5)相差1，表示目前尚有1個元素，front指標是指向目前佇列中第1個元素的前一個元素。
- **換句話說，只需比較兩個front和rear指標是否相等，就可以知道佇列是否已空。**
- 如果front指標是指向佇列中的第1個元素，當取出資料6後，front指標就已經和指標rear相同，都是索引值5。

14

使用陣列建立佇列-問題

- 陣列實作的佇列有一個大問題，因為front和rear變數的指標都是往同一個方向遞增，如果rear指標到達一維陣列的邊界MAXQUEUE-1，就算佇列尚有一些空間，也需要位移佇列元素，才有空間存入其它佇列元素，如下圖所示：



使用鏈結串列建立佇列

```
struct Node {          /* 佇列結構的宣告 */
int data;          /* 資料 */
struct Node *next;     /* 結構指標 */
};
typedef struct Node QNode; /* 佇列節點的新型態 */
typedef QNode *LQueue; /* 串列佇列的新型態 */
LQueue front = NULL;      /* 佇列的前端 */
LQueue rear = NULL;       /* 佇列的尾端 */
extern int isQueueEmpty();
extern void enqueue(int d);
extern int dequeue();
```

使用鏈結串列建立佇列-存入元素

- 函數enqueue()將資料存入佇列，插入成為串列的最後1個節點，其步驟如下所示：
  - Step 1：建立一個新節點存入佇列資料。

    new_node = (LQueue)malloc(sizeof(QNode));

    new_node->data = d;

    new_node->next = NULL;
  - Step 2：檢查rear指標是否是NULL，如果是，表示第一次存入資料，則：
    - (1) 如果是，將開頭指標front指向新節點。

    front = new_node;
    - (2) 如果不是，將rear指標所指節點的next指標指向新節點。

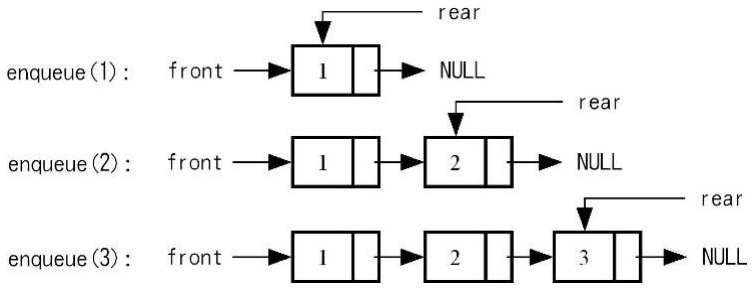
    rear->next = new_node;
  - Step 3：將後rear指標指向新節點。

    rear = new_node;

17

使用鏈結串列建立佇列-存入元素

- 依序存入值1~3到佇列，可以看到rear指標一直都是指向串列的最後1個節點，如下圖所示：



18

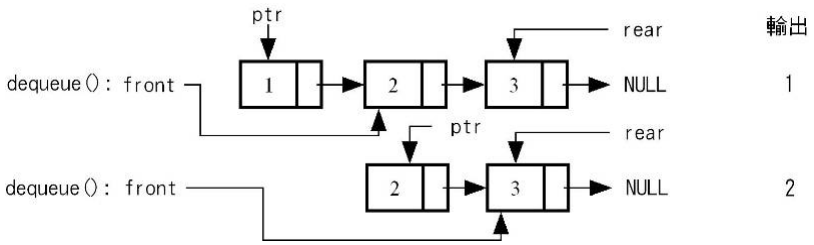
使用鏈結串列建立佇列-取出元素

- 函數dequeue()是從佇列取出資料，也就是刪除串列第1個節點，其步驟如下所示：
  - Step 1：若front等於rear指標，表示只剩一個節點，將rear設為NULL。
    ```
    if ( front == rear ) rear = NULL;
    ```
  - Step 2：將佇列的前端指標front指向下一個節點。
    ```
    ptr = front;
    front = front->next;
    ```
  - Step 3：取出第1個節點內容。
    ```
    temp = ptr->data;
    ```
  - Step 4：釋回第1個節點的節點記憶體。
    ```
    free(ptr);
    ```

19

使用鏈結串列建立佇列-取出元素

- 例如：在依序存入值1~3到佇列後，呼叫二次dequeue()函數取出佇列值，如下圖所示：



20

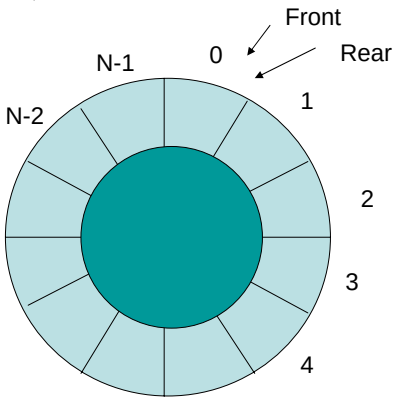
環型佇列

- 解決一般佇列問題的方法:環型佇列
- 將一般佇列(陣列)邏輯上的頭尾相接, 實體上仍然是一個一維陣列.

21

環型佇列的動作

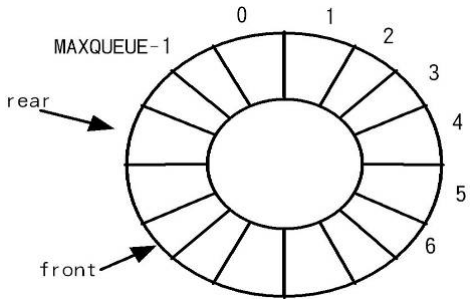
- 一維陣列共 N 個元素,製作 Q(0:n-1)的環型陣列.
- Q(n-1) 下一個元素是 0.



22

環狀佇列

- 「環狀佇列」（Circular Queue）也是使用一維陣列實作的有限元素數佇列，其差異只在使用特殊技巧來處理陣列索引值，將陣列視為一個環狀結構，佇列的索引指標周而復始的在陣列中環狀的移動，如下圖所示：



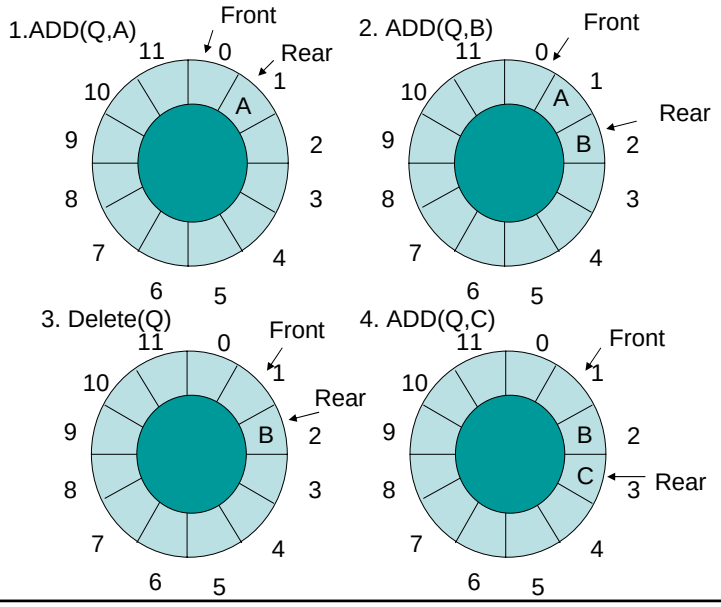
23

環狀佇列

```
#define MAXQUEUE   4      /* 佇列的最大容量 */
int queue[MAXQUEUE];      /* 佇列的陣列宣告 */
int front = -1;           /* 佇列的前端 */
int rear = -1;            /* 佇列的尾端 */
extern int isQueueEmpty();
extern int isQueueFull();
extern int enqueue(int d);
extern int dequeue();
```

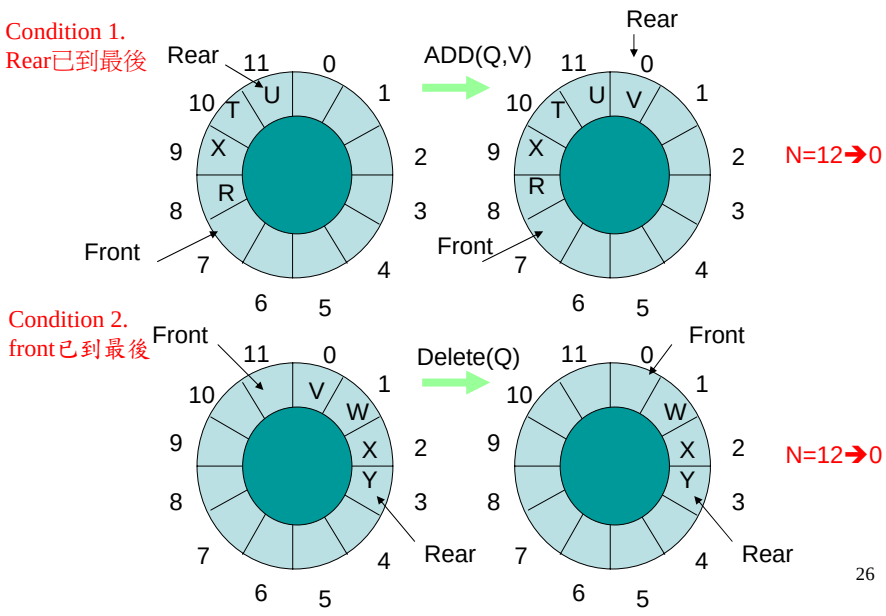
24

環型佇列操作範例(一)



25

環型佇列操作範例(二)



26

環型佇列的計算

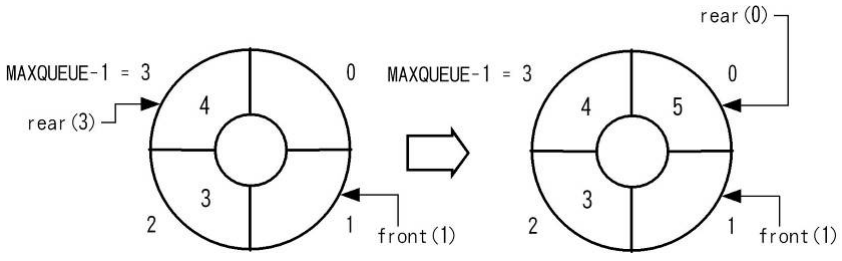
- Enqueue()
- Rear=(Rear+1) mod n
- If Rear = ((Front+1) mod n) then Queue_Full

- Dequeue()
- If Front=Rear then Queue_Empty
- Front=(Front+1) mod n

27

環狀佇列-存入元素1

- 函數enqueue()是在rear尾端將資料存入佇列，因為是環狀結構的陣列，所以當rear到達陣列邊界時，需要特別處理，如下圖所示：



28

環狀佇列-存入元素2

- MAXQUEUE為4的環狀佇列，當rear = 3時到達陣列邊界，此時再新增佇列元素5，rear++等於4，超過陣列尺寸，所以需要將它歸0，如下所示：

```
rear++;
if ( rear == MAXQUEUE ) rear = 0;
```

- ?:條件運算子，如下所示：

```
rear = ( rear+1 == MAXQUEUE ) ? 0 : rear+1;
```

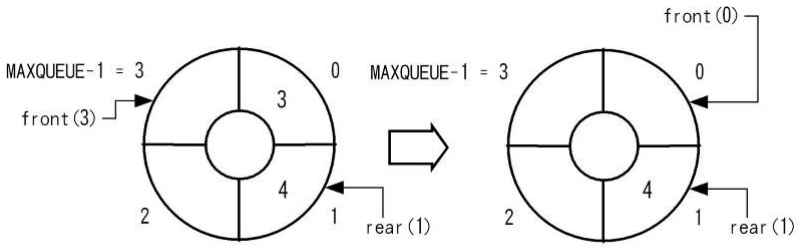
- 也可使用餘數運算，如下所示：

```
rear = (rear+1) % MAXQUEUE;
```

29

環狀佇列-取出元素1

- 函數dequeue()是在front前端從佇列取出資料，因為是環狀結構的陣列，所以當front到達陣列邊界時，需要特別處理，如下圖所示：



輸出 : 3

30

環狀佇列-取出元素2

- MAXQUEUE為4的環狀佇列，當front = 3時到達陣列邊界，此時再從佇列取出元素3，front++等於4，超過陣列尺寸，所以需要將它歸0，如下所示：

```
front++;
if ( front == MAXQUEUE ) front = 0;
```

- ?:條件運算子，如下所示：

```
front = ( front+1 == MAXQUEUE ) ? 0 : front+1;
```

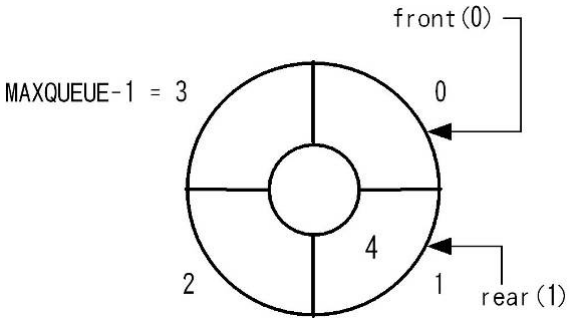
- 也可使用餘數運算，如下所示：

```
front = (front+1) % MAXQUEUE;
```

31

環狀佇列-佇列是否是空的1

- 函數isQueueEmpty()可以判斷環狀佇列是否已經空了。現在環狀佇列尚餘1個元素，如下圖所示：



32

環狀佇列-佇列是否是空的2

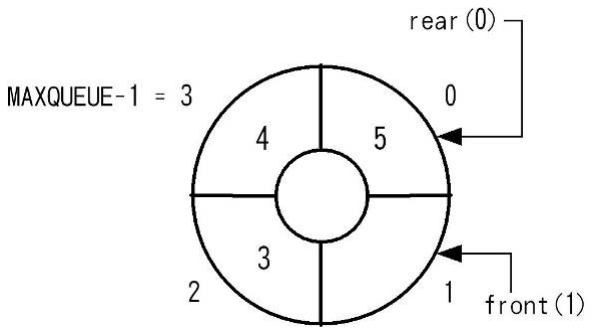
- 再執行一次dequeue()取出最後1個元素4，可以發現front和rear指標相等，換句話說，只需判斷兩個指標是否相等，就可以判斷環狀佇列是否已經空了，如下所示：

```
if ( front == rear ) return 1;
else          return 0;
```

33

環狀佇列-佇列是否已滿1

- 函數isQueueFull()可以判斷環狀佇列是否已滿。現在環狀佇列尚有1個空間沒有存入元素，如下圖所示：



34

環狀佇列-佇列是否已滿2

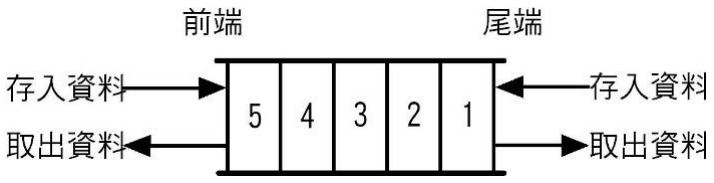
- 再執行enqueue(6)新增元素6，可以發現front和rear指標相等，沒有辦法判斷環狀佇列是已空和全滿，因為兩個指標都是指向相同索引值1。
- 所以，環狀佇列全滿就是指標rear和front相隔一個空間，換句話說，為了分辨環狀佇列是已空和全滿，其實際的儲存空間是陣列尺寸減1，如下所示：(犧牲1個位置)

```
int pos;
pos = (rear+1) % MAXQUEUE;
if ( front == pos ) return 1;
else          return 0;
```

35

雙佇列

- 「雙佇列」（Deques）是英文名稱（Double-ends Queues）的縮寫，雙佇列的二端如同佇列的前尾端，都允許存入或取出資料，如下圖所示：



36

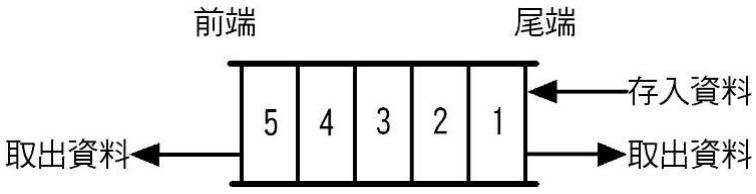
雙佇列-種類

- 雙佇列依其應用分為多種存取方式。常見的雙佇列，如下所示：
  - 輸入限制性雙佇列（Input Restricted Deque）。
  - 輸出限制性雙佇列（Output Restricted Deque）。
- 上述雙佇列是使用在電腦CPU的排程，排程在多人使用的電腦是重要觀念，因為同時有多人使用同一個CPU，而CPU在每一段時間內只能執行一個工作，所以將每個人的工作集中擺在一個等待佇列，等待CPU執行完一個工作後，再從佇列取出下一個工作來執行，排定工作誰先誰後的處理稱為「工作排程」。

37

輸入限制性雙佇列

- 輸入限制性雙佇列（Input Restricted Deque）是限制存入只能在其中一端，取出可以在兩端的任何一端，雙佇列只有一端存入，兩端都可以輸出，所以佇列輸出的結果擁有多種組合。



38

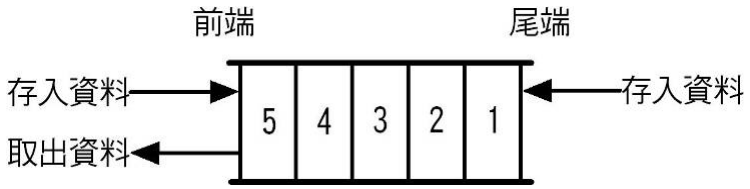
輸入限制性雙佇列

```
#define MAXQUEUE   10  /* 佇列的最大容量 */
int deque[MAXQUEUE];   /* 佇列的陣列宣告 */
int front = -1;        /* 佇列的前端 */
int rear = -1;         /* 佇列的尾端 */
extern int isDequeEmpty();
extern int isDequeFull();
extern int endeque(int d);
extern int dedeque_rear();
extern int dedeque_front();
```

39

輸出限制性雙佇列

- 輸出限制性雙佇列（Output Restricted Deque）是限制取出只能在一端，卻可以從兩端的任何一端存入元素，如下圖所示：



40

輸出限制性雙佇列

```
struct Node {           /* 佇列結構的宣告 */
int data;           /* 資料 */
struct Node *next;     /* 結構指標 */
};
typedef struct Node QNode;/* 雙佇列節點的新型態 */
typedef QNode *LDeque; /* 串列雙佇列的新型態 */
LDeque front = NULL;      /* 雙佇列的前端 */
LDeque rear = NULL;       /* 雙佇列的尾端 */
extern int isDequeEmpty();
extern void endeque_rear(int d);
extern void endeque_front(int d);
extern int dedeque();
```

41

請寫出答案

1. 原佇列=(a,b,c)，enqueue(d); enqueue(e); dequeue(); 請列出佇列中處理資料的過程及最後佇列內容為何?
2. 原佇列=(10,20,30) enqueue(dequeue()+40); printf("%d\n",dequeue()); printf("%d\n",dequeue()-dequeue()); enqueue(dequeue()+5); printf("%d\n",dequeue()); 列出佇列中處理資料的過程及最後佇列內容為何?
3. enqueue(3); enqueue(4); printf("%d\n",dequeue()+5); enqueue(5); enqueue(6); enqueue(7); enqueue(8); enqueue(dequque()+dequeue()); printf("%d\n",dequeue()*2);列出佇列中處理資料的過程即最後佇列內容為何?

42

請寫出答案

- 現有一個環狀佇列提供enqueue()和dequeue()函數，請寫出下列主程式main()的執行結果，如下所示：

```
void main(){  enqueue(5);  enqueue(3);  enqueue(4);
  printf("[%d]", dequeue());   enqueue(dequeue());
  enqueue(dequeue()+4);
  while ( !isQueueEmpty() )
    printf("[%d]", dequeue());
}
```

43

請寫出答案

- 現有一個環狀佇列大小為0-6，目前front=2、rear=5，佇列內容為(a,b,c)，請寫出下列front、rear的值，並寫出其執行結果
  1. dequeue()，front=? 、rear=? 、取出值=?
  2. enqueue(d)，front=? 、rear=?
  3. enqueue(e)，front=? 、rear=?
  4. dequeue()，front=? 、rear=? 、取出值=?

44

請寫出答案

- 現有一個環狀佇列大小為0-7，目前front=3、rear=6，佇列內容為(e,f,g)，請寫出下列front、rear的值，並寫出其執行結果

1. enqueue(a)，front=? 、rear=?
2. enqueue(d)，front=? 、rear=?
3. dequeue()，front=? 、rear=? 、取出值=?
4. enqueue(b)，front=? 、rear=?
5. dequeue()，front=? 、rear=? 、取出值=?
6. enqueue(c)，front=? 、rear=?
7. enqueue(h)，front=? 、rear=?
8. dequeue()，front=? 、rear=? 、取出值=?
9. dequeue()，front=? 、rear=? 、取出值=?
10. dequeue()，front=? 、rear=? 、取出值=?

45

請寫出答案

- 現有一個環狀佇列大小為0-5，，請寫出下列front、rear的值，並畫出佇列內容：

```
void main(){   enqueue(10);  enqueue(20);  enqueue(30);
  enqueue(dequeue()+40); printf("[%d]", dequeue());
  printf("[%d]", dequeue()-dequeue());
}
```

46