

C++資料結構與程式設計

堆疊與佇列(*Stack & Queue*)

NTU CSIE

大綱

堆疊 (Stack)

佇列 (Queue)

堆疊與佇列

堆疊(Stack)

- 加入(push)與刪除(pop)於同一端
- 後進先出(LIFO)
- 例子：疊盤子、發牌、走迷宮



佇列(Queue)

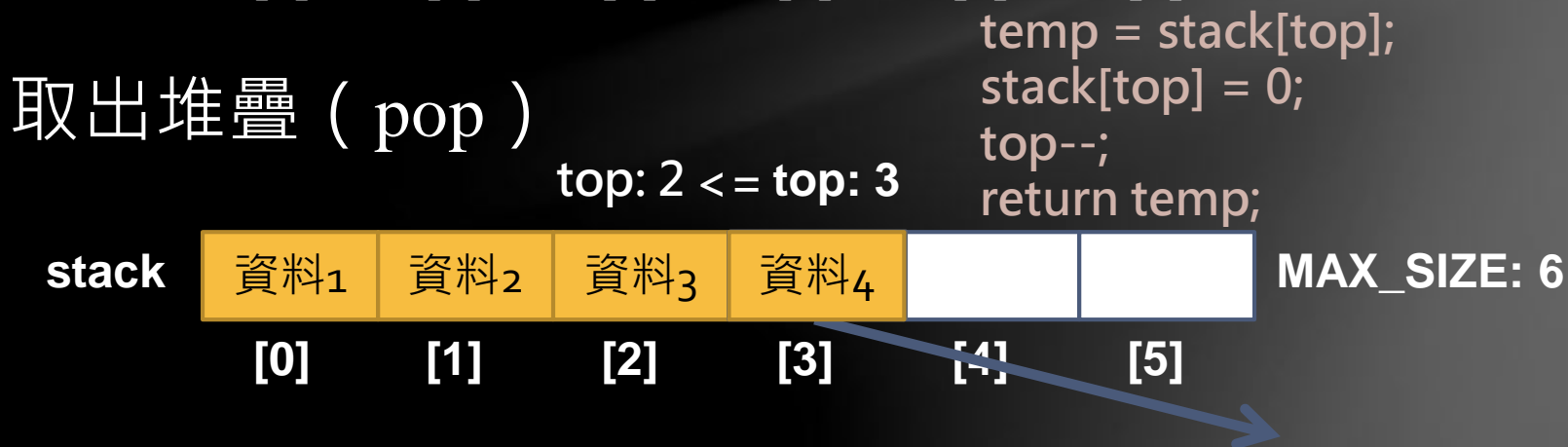
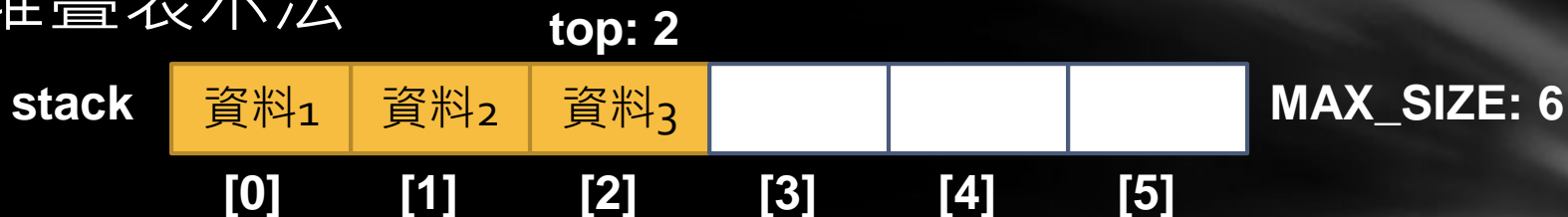
- 加入(enqueue)與刪除(dequeue)於不同端(front & rear)
- 先進先出(FIFO)
- 例子：排隊買票、坐公車、網路伺服器實作

```
<[A()B]{ }NTU>  
( )  
( ) [ 1 [ 2 [ 3 [ [ ]  
[ CSIE }
```



堆疊結構表示法(陣列)

堆疊表示法



堆疊結構表示法(陣列)

加入堆疊

```
int push(int value)
{
    if( top == MAX_SIZE-1 )
    {
        return -1;
    }
    top++;
    stack[top] = value;
    return 1;
}
```

堆疊結構表示法(陣列)

取出堆疊

```
int pop()
{
    int temp;

    if( top == -1 ) //判斷堆疊是否為空的
    {
        return -1;
    }
    temp = stack[top];
    stack[top] = 0;
    top--;
    return temp;
}
```

example

example2

撲克牌發排程式 ([stack_card.c](#))

使用堆疊結構實作發排程式

實作將52張撲克牌發給4位玩家, 並排序後印出發排結果

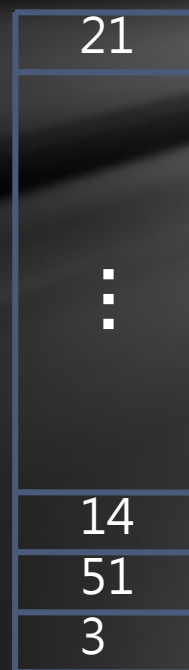
撲克牌表示：

- 0~12: 紅心 A~K
- 13~25: 方塊 A~K
- 26~38: 梅花 A~K
- 39~51: 黑桃 A~K

撲克牌花色字碼：

- 紅心: 3
- 方塊: 4
- 梅花: 5
- 黑桃: 6

堆疊大小: 52



發排：
將堆疊pop出去

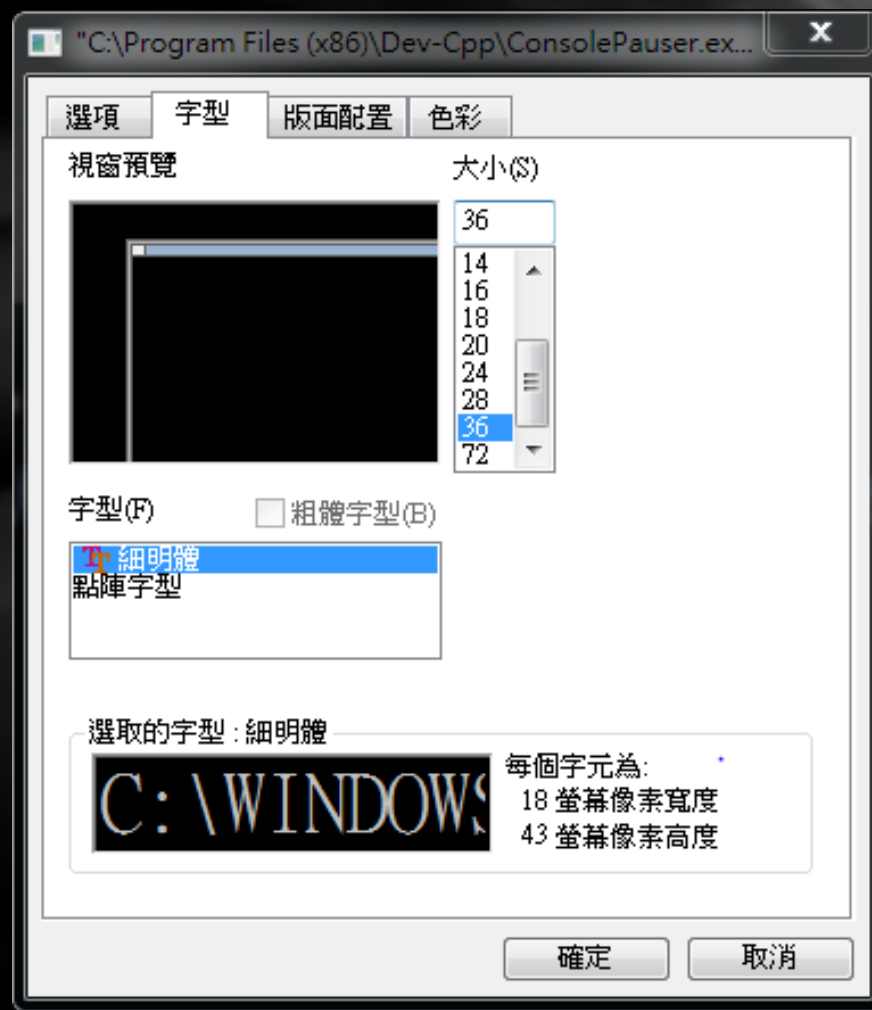
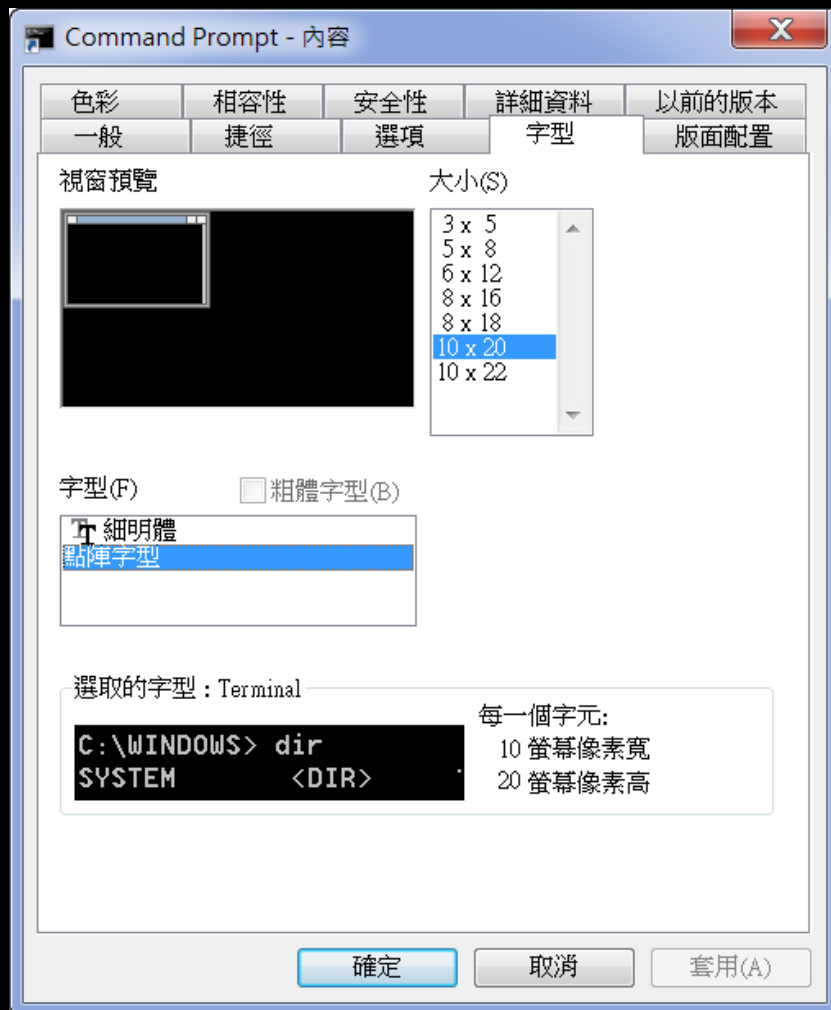
洗排：
亂數將牌號push入堆疊

ASCII code

[illegible]

無法正常顯示 ♠ ♥ ♦ ♣ ?

- 命令提示字元 內容>字型>點陣字型, 大小10x20



```

0
1  ☺
2  ☺
3  ♥
4  ♦
5  ♣
6  ♣
7
8
9
10
11  ♂
12  ♀
13
14  ♪
15  ☼
16  ▶
17  ◀
18  ‡
19  !!
20  ㊦
21  $
22  _
    
```

選項 字型 版面配置 色彩

大小(S)

8 x 12
16 x 12
8 x 16
12 x 16
8 x 18
10 x 18
10 x 20
10 x 22

視窗預覽



字型(F)

☞ MS Gothic
☞ NSimSun
☞ SimSun-ExtB
☞ 細明體
點陣字型

建議針對高 DPI 顯示器使用 TrueType 字型，因為點陣字型可能無法清楚地顯示。

☐ 粗體字型(B)

選取的字型: Terminal

```

C:\WINDOWS> dir
SYSTEM          <DIR>          10-
SYSTEM32        <DIR>          10-
README   TXT    26926  10-
    
```

每個字元為:

10 螢幕像素寬度
20 螢幕像素高度

確定

取消

回家作業 (stack_card_plus.c)

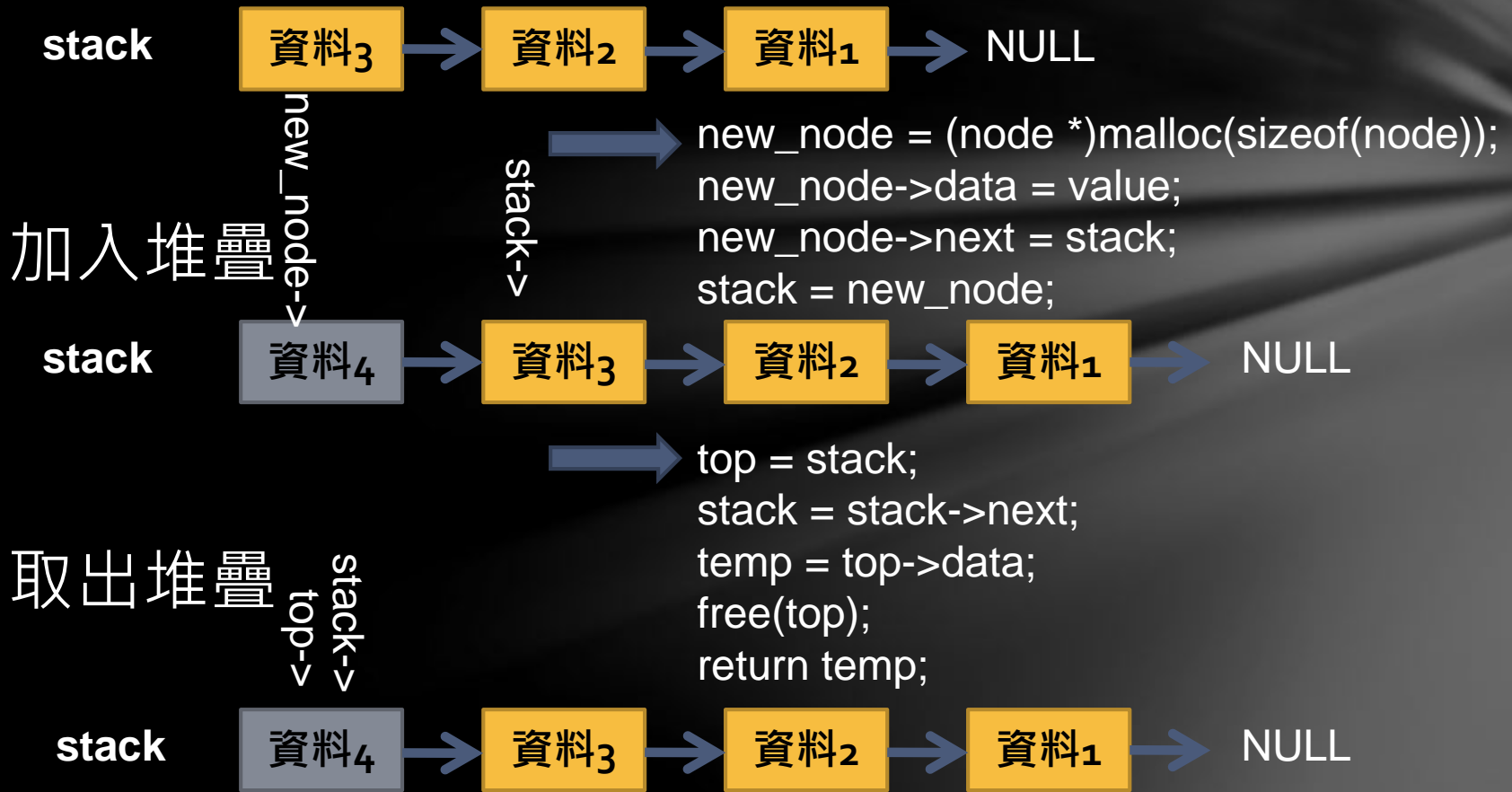
- 1.將上一撲克牌範例之排序後結果為照數字大小排序, 且花色依序為梅花,方塊,紅心,黑桃 (♣♦♥♠)
- 2.將洗牌方法改得更有效率(Hint:大樂透開獎程式)

<https://goo.gl/6SSRkN>

<https://goo.gl/nXnFY0>

堆疊結構表示法(鏈結串列)

堆疊表示法



堆疊結構表示法(鏈結串列)

加入堆疊

```
scanf("%d", &data_in);  
push(data_in);
```

```
void push(int value)  
{  
    node *new_node;  
  
    new_node = (node *)malloc(sizeof(node));  
    if( !new_node) //(new_node==NULL)判斷配置是否成功  
    {  
        printf("記憶體配置失敗!\n");  
        exit(1);  
    }  
    new_node->data = value;  
    new_node->next = stack;  
    stack = new_node;  
}
```

堆疊結構表示法(鏈結串列)

取出堆疊

```
int pop()
{
    node *top;
    int temp;

    if( (data_out = pop()) == -1 )
        printf("堆疊是空的\n");
    else
        printf("取出堆疊內容: %d\n", data_out);

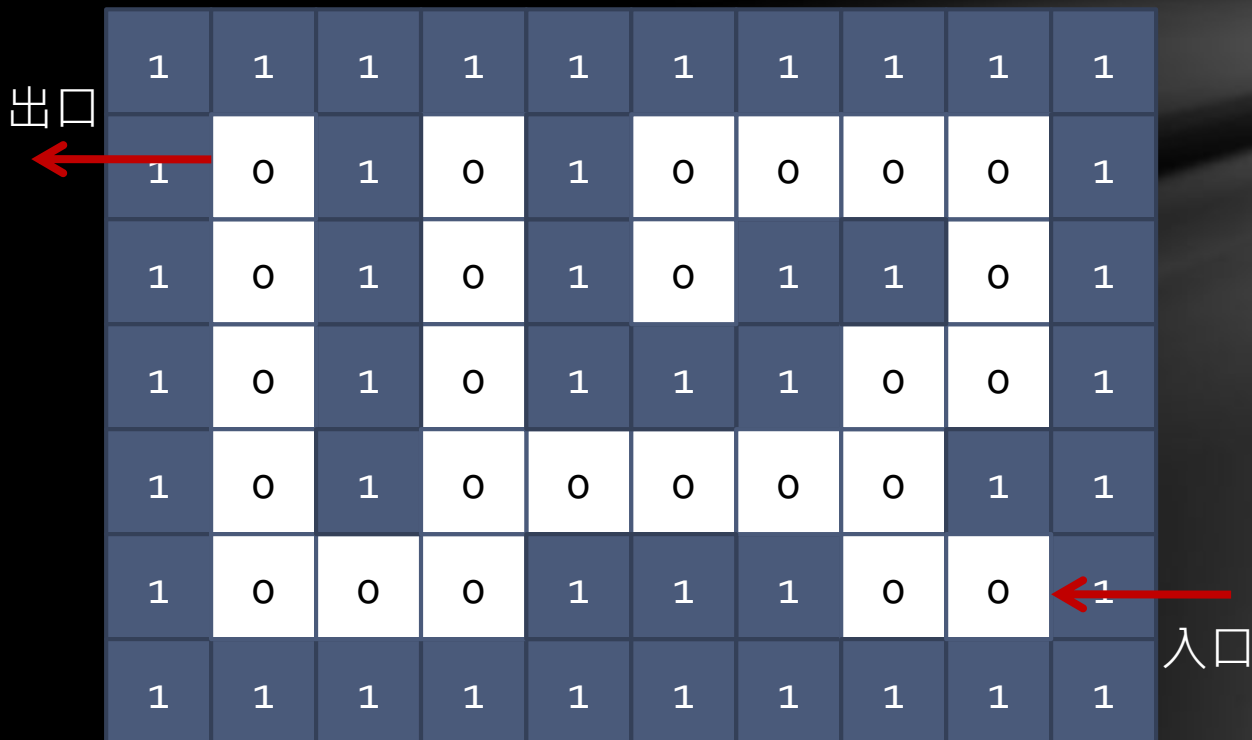
    if( stack != NULL ) //判斷堆疊是否為空的
    {
        top = stack;
        stack = stack->next;
        temp = top->data;
        free(top);
        return temp;
    }
    else
        return -1;
}
```

example

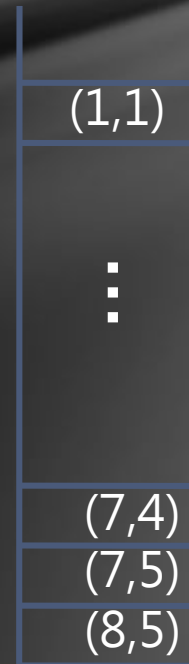
走迷宮問題 ([stack_maze.c](#))

▶ 使用堆疊結構實作走迷宮問題

- 走法: 每次都把目前位置存到堆疊, 然後走下一步
- 下一步順序: 上, 下, 左, 右
- 無路可走: 從堆疊中取出上一位置, 看看有沒路走



堆疊大小: ?




走錯:
座標pop出去

走一步:
座標push入堆疊

練習

(stack_maze_ex1.c) (stack_maze_ex2.c)

- 將上一迷宮問題範例解答之路徑座標印出



The screenshot shows a Windows command prompt window with the following text:

```
D:\Dropbox\台大課業\系統訓練班\C++資料結構與程式設計進階班\PPT\Part II 物件導向程式設計...
迷宮路徑為:
1 1 1 1 1 1 1 1 1 1
1 2 1 3 1 3 3 3 3 1
1 2 1 3 1 3 1 1 3 1
1 2 1 3 1 1 1 3 3 1
1 2 1 2 2 2 2 2 1 1
1 2 2 2 1 1 1 2 2 1
1 1 1 1 1 1 1 1 1 1
迷宮路徑座標:
[ 8, 5]
[ 7, 5]
[ 7, 4]
[ 6, 4]
[ 5, 4]
[ 4, 4]
[ 3, 4]
[ 3, 5]
[ 2, 5]
[ 1, 5]
[ 1, 4]
[ 1, 3]
[ 1, 2]
[ 1, 1]
請按任意鍵繼續 . . .
```

<https://jgirl.ddns.net/problem/0/2037>

- 請自行設計一15x10之迷宮，並檢驗其結果是否正確？

大綱

堆疊 (Stack)

佇列 (Queue)

佇列結構表示法(陣列)

```
rear++; size++;  
if(rear==MAX_SIZE) // 檢查是否超過界限  
    rear = 0; // 重頭開始  
queue[rear] = value; // 存入佇列
```

佇列表示法

front: -1

rear: 2

queue

資料₁

資料₂

資料₃

MAX_SIZE: 6

size: 3

[0]

[1]

[2]

[3]

[4]

[5]

加入佇列

front: -1

rear: 2 => rear: 3

資料₄

queue

資料₁

資料₂

資料₃

資料₄

MAX_SIZE: 6

size: 3 -> 4

[0]

[1]

[2]

[3]

[4]

[5]

front++; size--;

temp = queue[front];

queue[front] = 0;

return temp; // 取出佇列資料

取出佇列

front: -1 => front: 0

rear: 3

queue

資料₁

資料₂

資料₃

資料₄

MAX_SIZE: 6

size: 4 -> 3

[0]

[1]

[2]

[3]

[4]

[5]

佇列結構情況判斷(陣列)

佇列為空



佇列已滿



佇列結構表示法 (陣列)

加入佇列

```
scanf("%d", &data_in);  
if(enqueue(data_in)==-1)  
    printf ("佇列已滿\n");  
else  
    printf("已存入資料: %d\n", data_in);
```

```
int enqueue(int value)  
{  
    if( size == MAX_SIZE ) // 檢查佇列是否已滿  
        return -1;  
    rear++;  
    size++;  
    if(rear==MAX_SIZE) // 檢查是否超過界限  
        rear = 0; // 重頭開始  
    queue[rear] = value; // 存入佇列  
    return 1;  
}
```

佇列結構表示法(陣列)

取出佇列

```
int dequeue()  
{  
    int temp;  
  
    if( (data_out = dequeue()) == -1 )  
        printf("佇列是空的\n");  
    else  
        printf("取出佇列內容: %d\n", data_out);  
  
    if(size == 0) // 檢查佇列是否是空的  
        return -1;  
    front++;  
    size--;  
    if(front==MAX_SIZE) // 檢查是否超過界限  
        front = 0; // 重頭開始  
    temp = queue[front];  
    queue[front] = 0;  
    return temp; // 取出佇列資料  
}
```

<https://goo.gl/Qreu9o>

<https://tinyurl.com/y7exfude> 描標版

佇列結構表示法(鏈結串列)

佇列表示法



new_node->data = value;
new_node->next = NULL;
if(front == NULL)
 front = new_node;

加入佇列

else

rear->next = new_node;



取出佇列

top = front;
front = front->next;
temp = top->data;
free(top);
return temp;



佇列結構表示法(鏈結串列)

加入佇列

```
int enqueue(int value)
{
    node *new_node;

    new_node = (node *)malloc(sizeof(node));
    if( !new_node )
    {
        printf("記憶體配置失敗!\n");
        exit(1);
    }
    new_node->data = value;
    new_node->next = NULL;
    if(front==NULL)
        front = new_node;
    else
        rear->next = new_node;
    rear = new_node;
    return 1;
}
```


佇列結構表示法(鏈結串列)

取出佇列

```
int dequeue()
{
    node *top;
    int temp;

    if(front != NULL)
    {
        top = front;
        front = front->next;
        temp = top->data;
        free(top);
        return temp;
    }
    else
        return -1;
}
```

練習

- 實作鏈結串列堆疊程式([stack_list.c](#))
- 可
 - 存入
 - 取出
 - 列出所有內容 <https://goo.gl/Y3r4QW>
- 實作鏈結串列佇列程式([queue_list.c](#))
- 可
 - 存入
 - 取出
 - 列出所有內容 <https://goo.gl/a89Ani>