

# 第五章

# 陣列及在資料結構上的應用





# 本章內容

**5-1 陣列的觀念**

**5-2 一維陣列的宣告及儲存方式**

**5-3 陣列在「排序」上的實務應用**

**5-4 陣列在「搜尋」上的實務應用**

**5-5 二維陣列**

**5-6 多維陣列的觀念**

# 5-1 陣列的觀念



**【定義】**是指一群具有「相同名稱」及「資料型態」的變數之集合。

**【特性】**1.佔用連續記憶體空間。

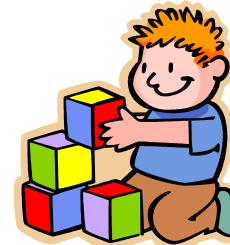
2.用來表示有序串列之一種方式。

3.各元素的資料型態皆相同。

4.支援隨機存取(Random Access)與循序存取(Sequential Access)。

5.插入或刪除元素時較為麻煩，因為須挪移其他元素。

**【示意圖】**

連續記憶體空間	各元素的資料型態皆相同
	

【例如】假設我們需要5個整數變數來存放資料時，那就必須要宣告一個A陣列為整數型態，其註標是按照順序排列從0~4共有5項，其含義如下：

```
int A [5];
```

陣列名稱 → A

陣列註標 → 0      1      2      .....      4

陣列元素 → 

A[0]	A[1]	A[2]	.....	A[4]
------	------	------	-------	------

### 【說明】

- (1) A陣列表示內共有5個陣列元素，也就是有5個變數，分別為A[0]、A[1]、A[2].....A[4]。
- (2)每一個陣列元素可以存放一筆資料。
- (3)當我們要連續輸入或輸出資料時，只須要使用「陣列」資料結構加上「迴圈」演算法就可以快速且有彈性處理資料了。

## (4)陣列內容的存取，通常是以迴圈指令配合輸入或輸出指令來進行，如下片段程式。

陣列內資料的「初始化」方式	陣列內資料的「輸出」方式
<pre>int[] A=new int[]{1,3,5,7,9}; //第一種寫法 int B[]={2,4,5,6,10}; //第二種寫法</pre>	<pre>for (int i=0;i&lt;=4;i++) {     System.out.println(B[i]); }</pre>

(5) 陣列所存放的每個資料叫做元素，若一個陣列中元素的個數為  $n$  時，表示此陣列的長度為  $n$ ，然後透過「陣列」與「註標」可以用來區分每個元素，註標是以一個數字表示，註標0是代表陣列的第一個元素，註標1代表陣列的第二個元素，.....，以此類推，則陣列的第  $n$  個元素則以註標  $n-1$  代表。

# 【實作1】



行號	程式檔名：ch5_1A.java
01	<b>public class</b> ch5_1A {
02	<b>public static void</b> main(String[] args)
03	{ //宣告
04	<b>int</b> [] A= <b>new int</b> []{1,3,5,7,9};
05	//處理
06	<b>for</b> ( <b>int</b> i=0;i<=4;i++)
07	{
08	//輸出
09	System. <b>out</b> .println(A[i]);
10	}
11	}
12	}

## 【說明】

行號04：宣告A為整數陣列，並且初值設定為1,3,5,7,9。

行號06~10：透過「迴圈」結構來顯示A陣列中的內含值。

行號11：利用System.out.println()方法來顯示A陣列中的內含值。



## 【優點】

1. 利用註標 (Index) 可以快速存取資料。

2. 一次可以處理大批資料。

(1) 利用註標 (Index) 可以快速的輸入資料。

連續「輸入」5 筆資料

```
for (i = 0; i <= 4; i++) //利用「迴圈結構」
```

```
A[i]=i*2+1; //快速「輸入資料」到「陣列」中
```

(2) 利用註標 (Index) 一次可以輸出大批的資料。

連續「輸出」5 筆資料

```
for (i = 0; i <= 4; i++) //利用「迴圈結構」
```

```
System.out.println(A[i]); //從「陣列」一次「輸出大批」的資料
```

3. 較容易表達資料處理的技巧。

在資料結構中幾乎每一種結構都必須要以「陣列」為基礎，例如：  
以陣列來實作堆疊或佇列，及各種排序或搜尋方法都必須使用陣列來處理。

## 【缺點】

1. 在新增或刪除資料時，比較麻煩。
2. 當記憶體配置在宣告陣列時就固定大小，缺乏彈性。

## 【實作2】

行號	程式檔名：ch5-1B.java
01	<code>public class ch5_1B {</code>
02	<code>    public static void main(String[] args)</code>
03	<code>    { //宣告</code>
04	<code>        int[] A=new int[5];</code>
05	<code>        //處理</code>
06	<code>        for (int i=0;i&lt;=4;i++)</code>
07	<code>        {</code>
08	<code>            //輸入</code>
09	<code>            A[i]=i*2+1;</code>
10	<code>        }</code>
11	<code>        for (int i=0;i&lt;=4;i++)</code>
12	<code>        {</code>
13	<code>            //輸出</code>
14	<code>            System.out.println(A[i]);</code>
15	<code>        }</code>
16	<code>}</code>
17	<code>}</code>

利用「迴圈結構」+「陣列」  
可以快速的「輸入」資料

利用「迴圈結構」+「陣列」  
可以快速的「輸出」資料

# 5-2 一維陣列的宣告及儲存方式



在上面的單元已經了解陣列的概念與好處之後，接下來，將介紹陣列的宣告與儲存方式。陣列宣告的方式和一般變數的宣告大同小異，所不同的，便是在陣列名稱後，必須要再加上陣列註標(索引)值大小，以便向系統爭取預留足夠的主記憶體空間。

## 5-2.1 陣列的宣告

【目的】為了配置記憶體空間，以便讓程式有足夠的空間進行資料運算。

【語法1】**資料型態[ ] 陣列名稱=new 資料型態[陣列大小]**

【說明】(1)「陣列名稱」的命名規則和一般變數相同。

(2)「陣列大小」必須是一數字型態。

【例子】`int[] A=new int[3];`

【語法2】**資料型態 陣列名稱[ ]=new 資料型態[陣列大小]**

【例子】`int A[]=new int[3];`

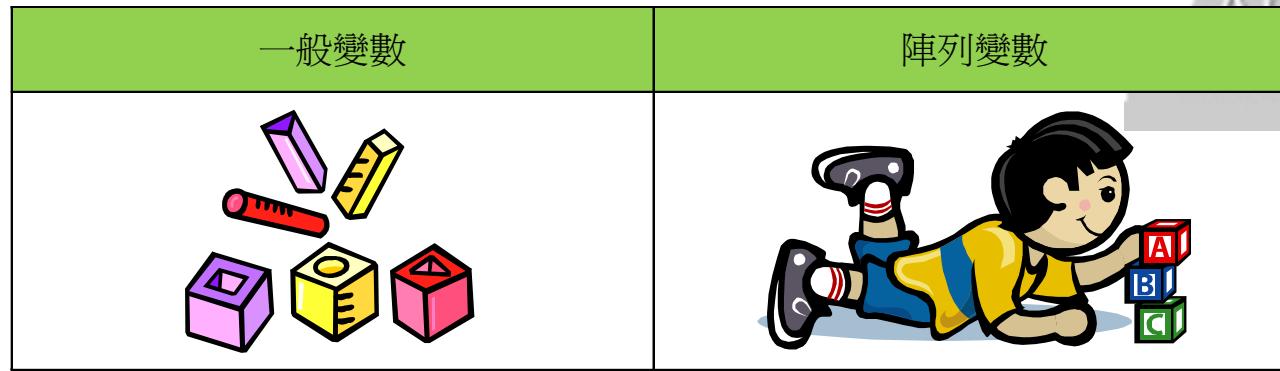
【分類】

基本上，「變數的宣告」可以分兩類：

第一類就是「一般變數」的宣告。

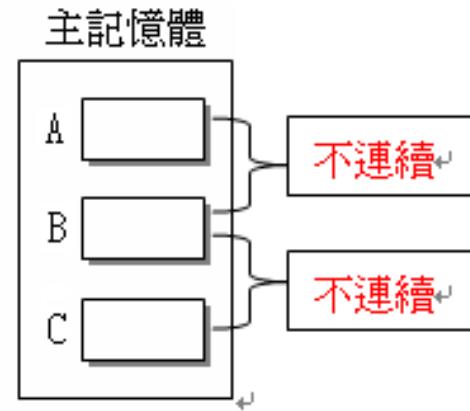
第二類則是「陣列變數」的宣告。

## 【示意圖】



### 【舉例1】「一般變數」的宣告

```
int A, B, C; //宣告三個變數(A,B,C)為整數型態
```

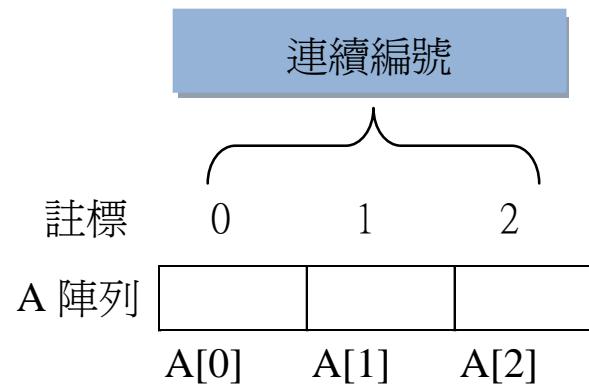


## 【說明】

以上三個變數之間都是個別獨立的記憶體空間，也就是不連續的記憶體空間的配置

## 【舉例2】「陣列變數」的宣告

`int[] A=new int[3]; //宣告一維陣列A，共有A[0]、A[1]、A[2]三個元素此時，主記憶體就會即時配置3個位置。`



## 【說明】

以上所配置位置是連續的記憶體空間，可以讓我們連續儲存多項資料，並且資料與資料之間都是按照順序排列的記憶體空間。

## 5-2.2 陣列的儲存方式

【定義】陣列名稱之後加上“註標”即可存取陣列元素。

【語法】**陣列名稱[註標]=資料來源;**

【說明】(1)「陣列名稱」的命名規則和一般變數相同。

(2)「註標」必須是一數字型態。

【舉例】宣告一個A[3]的陣列，並分別儲存10,20,30，如下所示：

```
int[] A=new int[3];
```

```
A[0]=10; //指把10指定給A陣列中的第0項的資料中
```

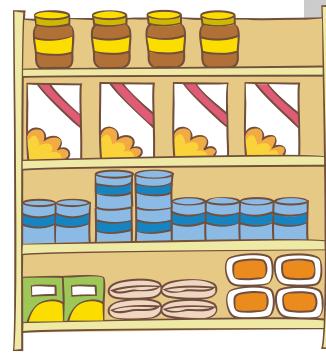
```
A[1]=20;
```

```
A[2]=30;
```

此時，主記憶體就會配置3個位置，分別存入10,20,30三項資料到陣列中，如下圖所示：

10	20	30
A[0]	A[1]	A[2]

## 【圖解說明】

存取元素	陣列儲存空間
	

## 【註標的三個型式】

陣列比變數來得有彈性，因為陣列可以利用各種型式的註標來表示：

1. 當註標是「變數」時：變數 $x=1$ 時，則 $A[x]=20$ ；
2. 當註標是「運算式」時：運算式 $x*2$ 時，則 $A[x*2]=30$ ；
3. 當註標是「陣列」時：陣列 $B[0]=1$ ，則 $A[B[0]]=20$ ；

【範例】假設我們現在想連續輸入3筆資料到A陣列中(分別是

10,20,30)，則只須要利用「註標」就可以快速的輸入資料。

行號	程式檔名：ch5-2-2.java
01	<code>public class ch5_2_2 {</code>
02	<code>    public static void main(String[] args)</code>
03	<code>    { //宣告</code>
04	<code>        int[] A=new int[3];</code>
05	<code>        //處理</code>
06	<code>        for (int i=0;i&lt;=2;i++)</code>
07	<code>            {</code>
08	<code>                //輸入</code>
09	<code>                A[i]=(i+1)*10;</code>
10	<code>            }</code>
11	<code>        }</code>
12	<code>}</code>

此時，主記憶體就會配置3個位置，分別存入10,20,30三項資料到陣列中，如下圖所示：

A 陣列	10	20	30
	A[0]	A[1]	A[2]

【說明】在上面的程式中，我們可以利用for迴圈來控制輸入資料的筆數，並且利用i變數來控制輸入資料的內容。

# 【實例1】請依序輸入6位同學的成績到陣列中，並計算及輸出「總和」。

## 【第一種寫法】

使用陣列，但未使用 for 迴圈演算法		程式檔名：ch5_2_2A.java
01	<pre>public class ch5_2_2A {     public static void main(String[] args)     { //宣告         int Sum;         int[] A=new int[6];         A[0]=60; A[1]=70; A[2]=80; A[3]=85; A[4]=90; A[5]=100;         //處理         Sum = A[0] + A[1] + A[2] + A[3] + A[4] + A[5];         System.out.println(Sum);     } }</pre>	

## 【程式解析】

行號04~05：宣告總分及A陣列

行號06：設定六科成績給A陣列

行號08：未使用for迴圈，來加總六科成績

行號09：顯示總分。

## 【第二種寫法】最佳

使用陣列，並使用 for 迴圈演算法

程式檔名：ch5\_2\_2B.java

```
01 public class ch5_2_2B {  
02     static int Sum;  
03     public static void main(String[] args)  
04     { //宣告  
05         int[] A=new int[6];  
06         A[0]=60; A[1]=70; A[2]=80; A[3]=85; A[4]=90; A[5]=100;  
07         //處理  
08         for (int i = 0; i<6;i++ )  
09             Sum+=A[i];  
10         System.out.println(Sum);  
11     }  
12 }
```

使用 for 迴圈演算法



## 5-2.3 使用陣列的注意事項

雖然陣列比變數來得有彈性，但是，也要注意以下事項：

- 不能夠一次讀取或指定整個陣列的資料。

現在寫一個程式，利用A陣列來存放數字10。

(1)直覺想法：以下的方法是錯誤

```
int[] A=new int[10];
```

```
A=10;           ← 不能直接指定給陣列名稱
```

原因：想把整個陣列的資料都指定為10時，電腦會產生錯誤

Error。

(2)正確方法：必須把程式改為如下：



行號	程式檔名：ch5_2_3.java
01	<b>public class ch5_2_3 {</b>
02	<b>public static void main(String[] args)</b>
03	{
04	<b>int[] A =new int[10];</b>
05	<b>for (int i=0;i&lt;10;i++)</b>
06	<b>{ //必須利用迴圈來控制，使數值10逐一的存到陣列中</b>
07	<b>A[i]=10;</b>
08	<b>System.out.println(A[i]);</b>
09	<b>}</b>
10	<b>}</b>
11	<b>}</b>

## 2. 用來指定某一項資料的註標不能超過陣列的註標範圍。

例如：**int[] A =new int[50]; // 宣告一個陣列A其註標是從0~49**

**A[-1]=100; //註標是 -1，超出陣列A的範圍**

**A[50]=100; //註標是 50，超出陣列A的範圍**



## 5-2.4 陣列的初值設定

【定義】是指宣告陣列的同時並指定初值。

【目的】可以縮短程式的長度。

【語法】資料型別[] 陣列名稱=new 資料型別[] = {陣列初值串列};

【注意】陣列宣告的同時設定初值時，不需要指定陣列的大小，因為編譯器會自動根據「陣列初值串列」大小來分配空間。

【範例1】假設宣告A是一個含有5個整數的陣列，其中初值為：

第一種寫法：int[] A =new int[] {1,2,3,4,5};

第二種寫法：int[] A ={1,2,3,4,5};

## 【範例2】承上一題，如果先宣告，後再指定初值時，則必須要指定陣列大小

```
int[] A = new int[5];  
A[0]=1; A[1]=2 ; A[2]=3 ; A[3]=4 ; A[4]=5;
```

### 【程式】利用初值設定6科成績，來計算總分

行號	程式檔名：ch5-2-4.java
01	<code>public class ch5_2_4 {</code>
02	<code>    public static void main(String[] args)</code>
03	<code>    { //宣告及初值設定</code>
04	<code>        int[] A ={ 60, 70, 80, 85, 90, 100 };</code>
05	<code>        int i,sum=0;</code>
06	<code>        //處理</code>
07	<code>        for (i = 0; i&lt;=5;i++ )</code>
08	<code>            sum+=A[i];</code>
09	<code>        //輸出</code>
10	<code>        System.out.println("Sum=" + sum);</code>
11	<code>    }</code>
12	<code>}</code>

# 5-3 陣列在「排序」上的實務應用

AUTHEN  
THEIA  
UNIVERSITY

【定義】將一組資料依使用者的需要予以重新安排其順序。

【優點】容易閱讀、利於統計分析及可以快速搜尋。

【常見的種類】

1.氣泡排序法(Bubble Sort)	2.選擇排序法(Insertion Sort)
<p>原始資料 : 3 7 1 6 8</p> <p>第一回合 :  比較4次</p> <p>第二回合 :  比較3次</p> <p>第三回合 :  比較2次</p> <p>第四回合 :  比較1次</p> <p>總共的比較次數 : <math>4+3+2+1=10</math>次</p>	<p>原始資料 : 9 7 3 1 5</p> <p>第一回合 :  </p> <p>第二回合 :  </p> <p>第三回合 :  </p> <p>第四回合 :  </p> <p>排序結果為 : 1,3,5,7,9</p>



## 5-3.1 氣泡排序法

在日常生活中，我們常常會根據某些要求做簡單的排序，而在資料結構中最普遍也最簡單的應該就是「氣泡排序法」。

### 【定義】

將兩個相鄰的資料相互做比較，若比較時發現次序不對，則將兩資料互換，依次由上往下比，而結果則會依次由下往上浮起，猶如氣泡一般。

【原理】逐次比較兩個相鄰的資料，按照排序的條件交換位置，直到全部資料依序排好為止。

【舉例】假設我們現在有5筆資料要進行排序，分別是3,7,1,6,8  
請利用氣泡排序法由小到大進行排序。

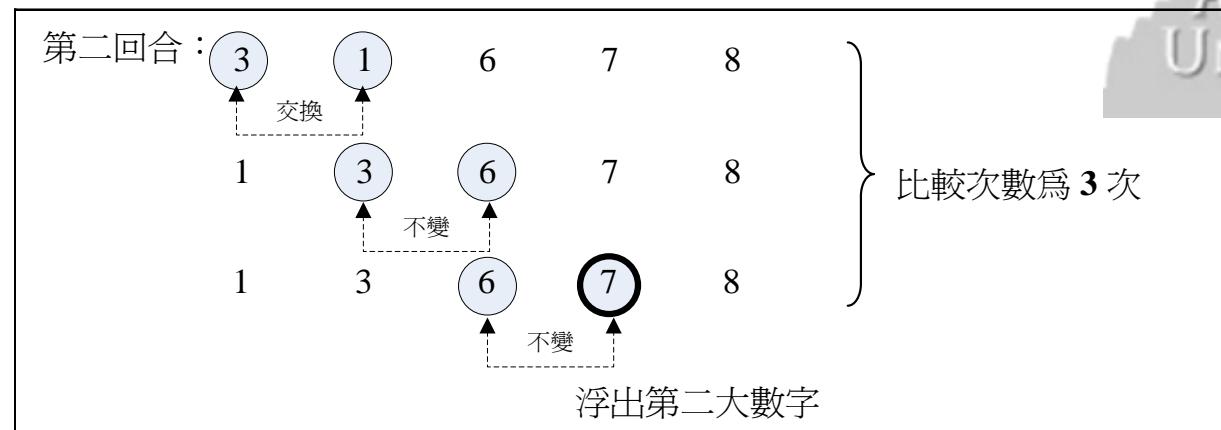
## 【圖解說明】

第一回合：



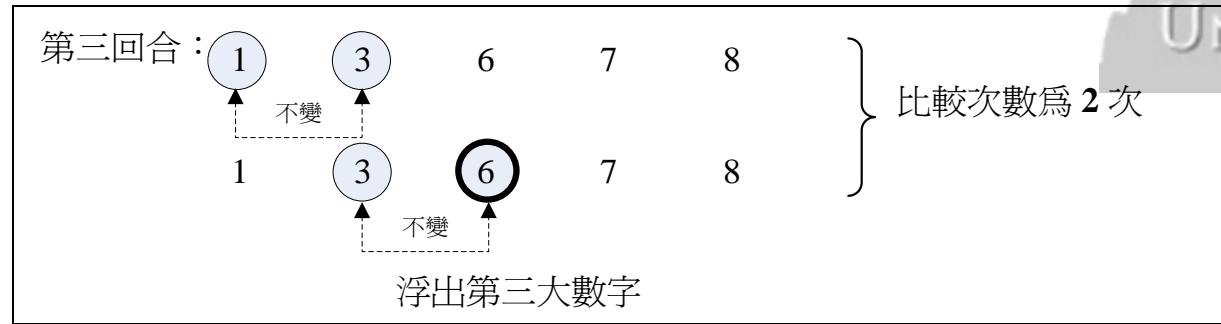
首先，3與7比較，而3小於7，所以，不交換。此時，比較次數加1。  
接下來，7與1比較，而7大於1，所以，交換。此時，比較次數再加1。  
接下來，7與6比較，而7大於6，所以，交換。此時，比較次數再加1。  
最後，7與8比較，而7小於8，所以，不交換。此時，比較次數再加1。  
因此，在完成第一回合之後，就會浮出最大數字8。

## 第二回合：



首先，3與1比較，而3大於1，所以，交換。此時，比較次數加1。  
接下來，3與6比較，而3小於6，所以，不交換。此時，比較次數再加1。  
最後，6與7比較，而6小於7，所以，不交換。此時，比較次數再加1。  
因此，在完成第二回合之後，就會浮出第二大數字7。

### 第三回合：

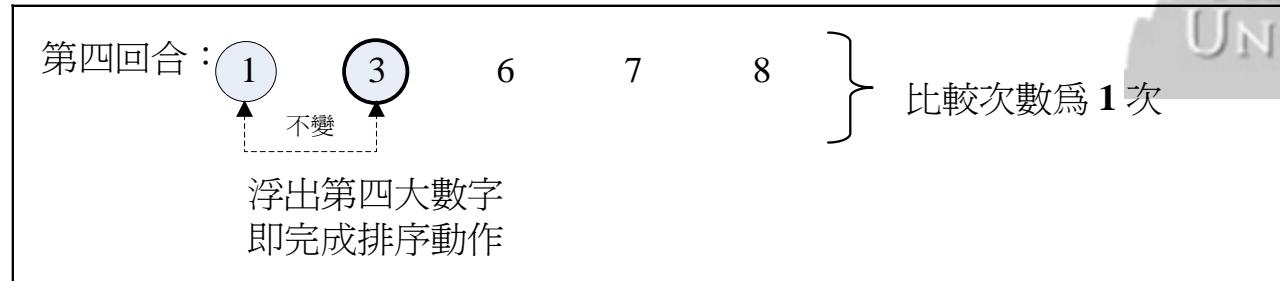


首先，1與3比較，而1小於3，所以，不交換。此時，比較次數加1。

最後，3與6比較，而3小於6，所以，不交換。此時，比較次數再加1。

因此，在完成第三回合之後，就會浮出第三大數字6。

## 第四回合：



首先，1與3比較，而1小於3，所以，不交換。此時，比較次數加1。  
因此，在完成第四回合之後，就會浮出第四大數字3。  
最後，在完成以上四回合之後，其排序結果為：1,3,6,7,8。  
並且，總共的「比較次數」為 $4+3+2+1=10$ 次。

## 【實作】

請利用「氣泡排序法」將以下的數列「由小到大」進行排序。

假設數列：3,7,1,6,8

行號	程式檔名：ch5-3-1.java
01	<code>public class ch5_3_1 {</code>
02	<code>    public static void main(String[] args)</code>
03	<code>        { //宣告及初值設定</code>
04	<code>            int i,j,temp;</code>
05	<code>            int NumArray1[]={3,7,1,6,8};</code>
06	<code>            int NumArray2[]={3,7,1,6,8};</code>
07	<code>            for(i=3;i&gt;=0;i--) //控制「比較」次數</code>
08	<code>            {</code>
09	<code>                for(j=0;j&lt;=i;j++) //控制「交換」次數</code>
10	<code>                {</code>
11	<code>                    if(NumArray2[j]&gt;NumArray2[j+1])</code>
12	<code>                    { //進行交換過程</code>
13	<code>                        temp=NumArray2[j];</code>
14	<code>                        NumArray2[j]=NumArray2[j+1];</code>
15	<code>                        NumArray2[j+1]=temp;</code>
16	<code>                    }</code>
17	<code>                }</code>
18	<code>            }</code>
19	<code>            System.out.println("Your Numbers:");</code>



```
20      for(i=0;i<5;i++)  
21          { //顯示排序前的資料  
22              System.out.print(" "+ NumArray1[i]);  
23          }  
24          System.out.println();  
25          for(i=0;i<5;i++)  
26              { //顯示排序後的資料  
27                  System.out.print(" "+ NumArray2[i]);  
28              }  
29      }  
30  }
```

## 【執行結果】

```
Your Numbers:  
3 7 1 6 8  
1 3 6 7 8
```

## 【說明】

行號04：宣告i,j,temp為整數變數，其中i變數是用來控制「比較」次數，而j變數是用來控制「交換」次數，而temp變數是用來暫存兩數交換的中間值。

行號05：宣告NumArray1為陣列變數，用來顯示「原始資料」的陣列，  
並且設定初值串列。

行號06：宣告NumArray2為陣列變數，用來進行排序使用的陣列，並  
且設定初值串列。

行號07~18：利用雙重迴圈用來進行「氣泡排序法」。

行號07：外層迴圈用來控制「比較」次數。

行號09：內層迴圈用來控制「交換」次數。

行號11~16：用來進行「兩數交換」過程。

行號19~23：用來顯示「排序前」的資料。

行號24：換行

行號25~28：用來顯示「排序後」的資料。

## 5-3.2 選擇排序法（Selection Sort）



### 【引言】

在氣泡排序法中，每找到一個比目前數值大或小的數字時，就必須執行資料的交換，這樣的方法容易將時間浪費在資料的交換上。因此我們可以採用改良的方式，也就是利用「選擇排序法」。

### 【定義】

在找到比目前大或小的數字時，先記錄其位置或索引值，待確定後再進行資料的交換，而這樣的方法我們稱之為選擇排序法（Selection Sort）。

【原理】第一回合由資料中選取最小的資料和第一個資料對調、第二回合由資料中選取第二小的資料和第二個資料對調(因最小的資料已排到第一個位置)、依此循環直到最後一個資料，即完成資料的排序。

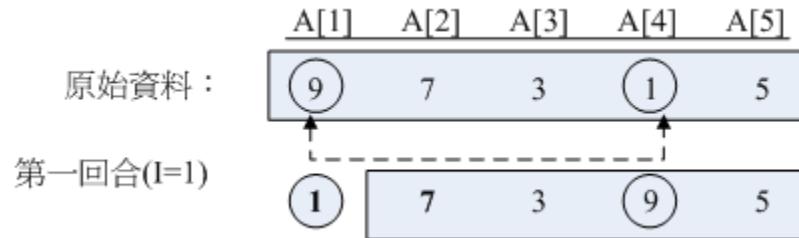
【舉例】假設我們現在有5筆資料要進行排序，分別是9,7,3,1,5  
請利用選擇排序法由小到大進行排序。



### 【圖解說明】

第一回合：共有5組數字

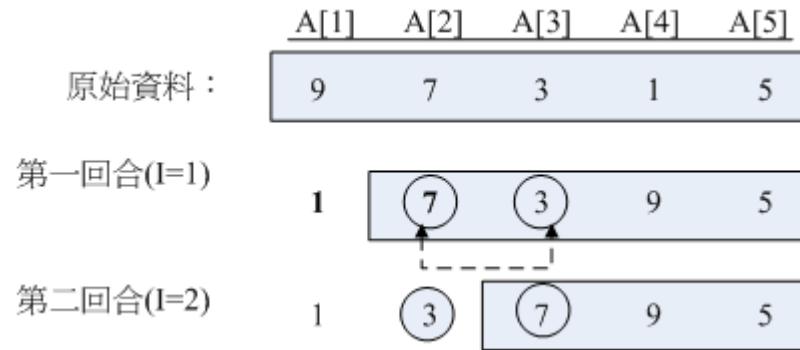
從第一個資料，由左至右掃瞄資料一次，選取出「最小資料1」和「第一個資料9」進行交換



說明：1.第一回合找出方框中的最小值。  
2.將最小值與目前陣列元素互換。

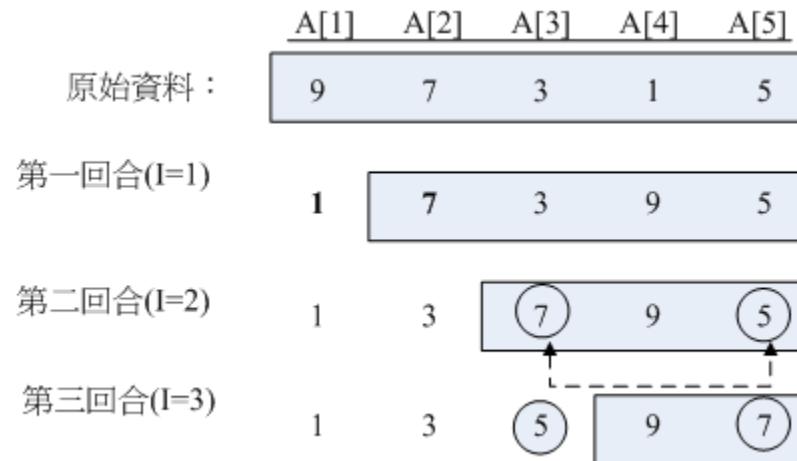
## 第二回合：只剩下後面4組數字要進行比對

從第二個資料，由左至右掃瞄資料一次，選取出「第二小的資料3」和「第一個資料7」進行交換



### 第三回合：只剩下後面3組數字要進行比對

從第三個資料，由左至右掃瞄資料一次，選取出「第三小的資料5」和「第一個資料7」進行交換



## 第四回合：只剩下後面2組數字要進行比對

從第四個資料，由左至右掃瞄資料一次，選取出「第四小的資料7」和  
「第一個資料9」進行交換



最後，在完成以上四回合之後，其排序結果為：1,3,5,7,9

## ⊕ 「選擇排序法」的演算法如下：

演算法：選擇排序法的過程		程式檔名：ch8-3
01	Procedure SelSort(int A [], int n)	
02	Begin	
03	for (i = 0; i < n - 1; i++) //控制排序 n-1 個回合	
04	{	
05	Min = i; //設定最小值	
06	for (j = i + 1; j <= n; j++)	
07	if (A[j] < A[Min])	
08	Min = j;	
09	//相鄰兩個的資料交換位置	
10	Temp = A[i];	
11	A[i] = A[Min];	
12	A[Min] = Temp;	
13	}	
14	}	
15	End	
16	End Procedure	

# 【實例】先產生10個亂數值，其範圍為10~100之間，再利用「選擇排序法」進行由小到大排序。

## 【解答】



題目：選擇排序法	程式檔案名稱	ch5-3-2.java
01 <b>public class</b> ch5_3_2 02 { 03 <b>public static int</b> Num=10; 04 <b>public static int</b> A[]= <b>new int</b> [Num]; 05 <b>public static void</b> main(String[] args) 06     { 07         RandomNum();                        //呼叫產生10個亂數值的副程式 08         SelSort(A, Num);                    //呼叫選擇排序法的副程式 09         PrintSelSort(A, Num);              //呼叫選擇排序後的結果之副程式 10     } 11 12     //產生10個亂數值之副程式 13 <b>public static void</b> RandomNum() 14     { 15 <b>int</b> i; 16         System.out.println("產生10個亂數值：" );		



```
17  for(i=0;i<Num;i++)
18  {
19      A[i] = (int)(Math.random()*90)+10; //產生10~100的整數亂數值
20      System.out.print(A[i] + " ");
21  }
22  System.out.println();
23 }
24
25 //選擇排序法之副程式
26 public static void SelSort(int A[], int n) //選擇排序法之副程式
27 {
28     int i, j, Temp, NP = 0;
29     for (i = 0; i <n - 1; i++)
30     {
31         NP = i;
32         for (j = i; j <n; j++)
33             if (A[j] > A[NP])
34                 NP = j;
35         {//相鄰兩個的資料交換位置
36             Temp = A[i];
37             A[i] = A[NP];
38             A[NP] = Temp;
39         }
40     }
41 }
```

```
42
43 //列印排序後的結果之副程式
44 public static void PrintSelSort(int A[], int n)
45 {
46     int i;
47     System.out.println("排序10個亂數值：");
48     for (i = 0; i < Num; i++)
49         System.out.print(A[i] + " ");
50 }
51 }
```

## 【執行結果】

產生10個亂數值：

90 97 61 30 38 23 97 25 86 83

排序10個亂數值：

97 97 90 86 83 61 38 30 25 23