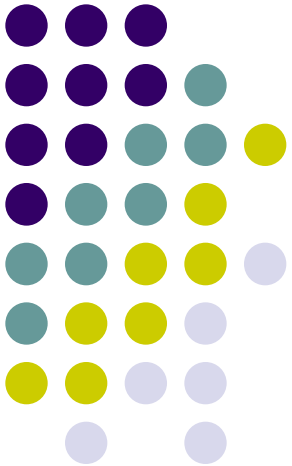
第六章
陣 列

認識陣列與一般資料型態的不同

認識一維與二維陣列

學習陣列的應用

了解陣列的儲存方式



一維陣列

- 一維陣列（1-dimensional array）可以存放多個相同資料型態的資料。
- 使用陣列必須經過兩個步驟：

  (1) 宣告陣列

  (2) 配置記憶體給該陣列
- 一維陣列的宣告與配置記憶體格式：

使用陣列的語法

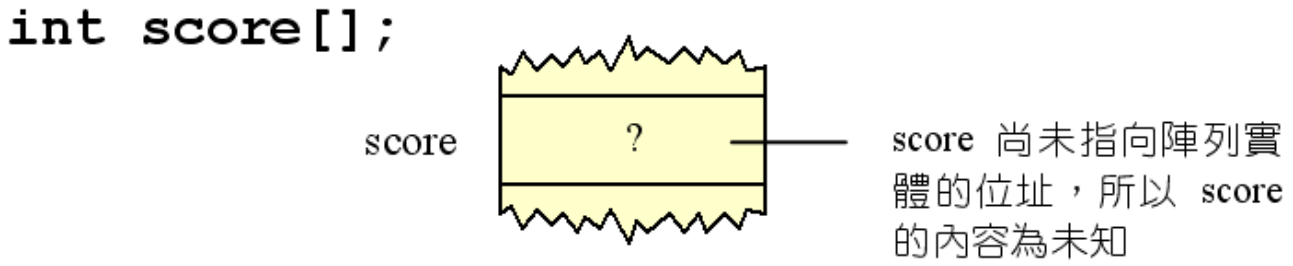
```
資料型別 陣列名稱;                  // 宣告一維陣列
陣列名稱 = new 資料型別[個數]; // 配置記憶體給陣列
```

一維陣列的宣告及使用 (1/3)

- 一維陣列的宣告及記憶體配置：

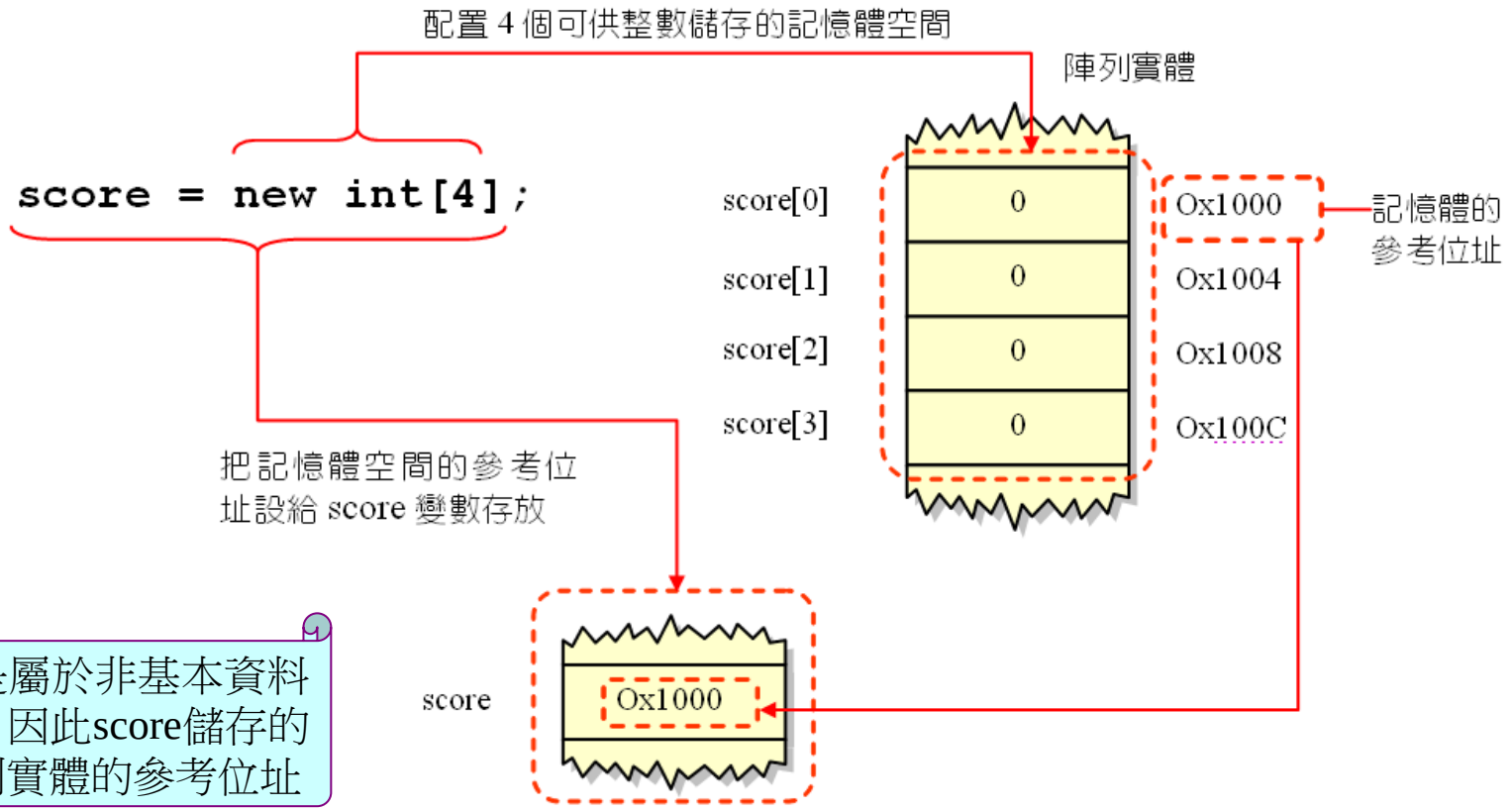
```
01   int score[];          // 宣告整數陣列 score
02   score=new int[4];     // 配置可存放 4 個整數的記憶體空間，以供陣列 score 使用
```

- 執行完第1行後，編譯器會配置一塊記憶體給它：



一維陣列的宣告及使用 (2/3)

- 第2行是記憶體配置的動作：



陣列是屬於非基本資料型態，因此score儲存的是陣列實體的參考位址

一維陣列的宣告及使用 (3/3)

- 宣告一維陣列的另一種寫法：

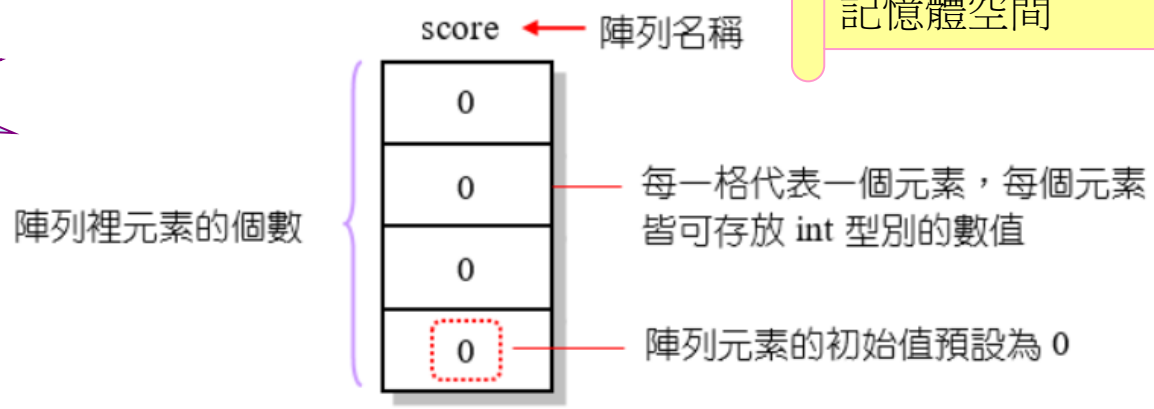
宣告陣列並配置記憶體

```
資料型別[] 陣列名稱= new 資料型別[個數];  // 宣告陣列並配置記憶體
```

- 一維陣列的宣告範例：

```
int[] score=new int[4];
```

將陣列score化為圖形表示



宣告一個整數陣列score，同時配置一塊可存放4個整數的記憶體空間

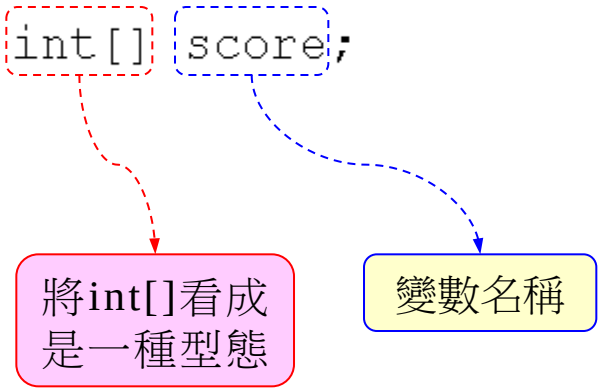
陣列的另一種宣告方式

- 稍早是以下列語法宣告score陣列：

```
int score[];                    // 宣告score陣列為整數型態
```

- 還可以用另一種語法宣告：

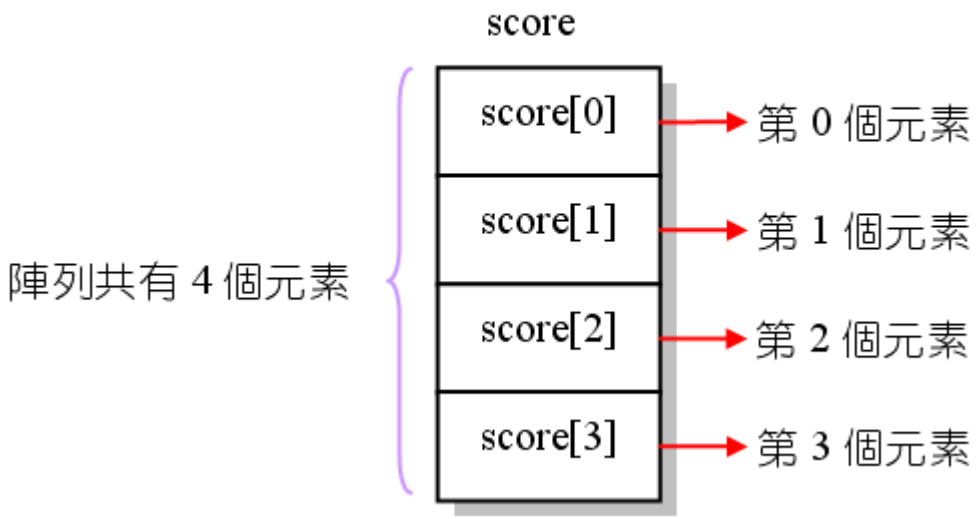
```
int[] score;                    // 宣告 score 變數，其型態為整數陣列
```

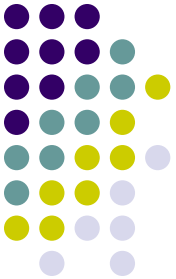


陣列元素的表示方法

- 要存取陣列裡的元素，可以利用索引值（index）
- 陣列索引值的編號是由0開始

```
int score[]=new int[4];
```





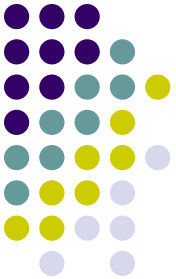
一維陣列的使用範例

- 下面的程式是一維陣列的使用範例：

```
01  // Ch6_1, 一維陣列的使用
02  public class Ch6_1{
03      public static void main(String[] args){
04          int[] a=new int[3];    // 宣告整數陣列 a,並配置可存放 3 個整數的空間
05          a[0]=9;          // 設定第 0 個元素的值為 9
06          a[1]=6;          // 設定第 1 個元素的值為 6
07
08          for(int i=0; i<a.length; i++)
09              System.out.printf("a[%d]=%d, ",i,a[i]); // 印出陣列的內容
10          System.out.printf("length=%d",a.length);    // 印出陣列長度
11      }
12  }
```

- 執行結果：

```
a[0]=9, a[1]=6, a[2]=0, length=3
```



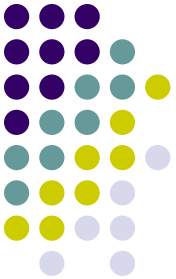
陣列的長度

- 取得陣列元素的個數（陣列長度）的格式：

取得陣列的長度
陣列名稱.length　　// 取得陣列的長度

- 如下面的程式片段：

```
a.length                    // 取得陣列a的長度
```



陣列初值的設定 (1/2)

- 在宣告時就給與陣列初值的格式：

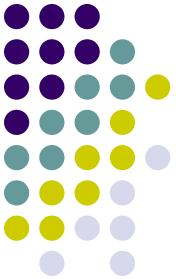
陣列初值的設定

```
資料型別[] 陣列名稱={初值 0，初值 1，…，初值 n};    // 陣列初值的設定
```

- 以上面的格式宣告時，會視初值的個數來決定陣列的長度，如下面的範例：

```
int[] days={6,8,12};        // 宣告陣列並設定初值
```

陣列元素有3個，day[0]為6，day[1]為8，day[2]為12



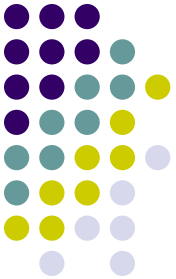
陣列初值的設定 (2/2)

- Ch6_2是一維陣列的範例：

```
01  // Ch6_2, 設定陣列初值並計算平均
02  public class Ch6_2{
03     public static void main(String[] args){
04        int sum=0;
05        int[] a={62,7,12,3,8,47};        // 宣告整數陣列 a 並設定初值
06
07        for(int i=0;i<a.length;i++)      // 計算陣列元素的和
08           sum+=a[i];
09        System.out.printf("Average = %5.2f",(float)sum/a.length);
10     }
11  }
```

- 執行結果：

```
Average = 23.17
```



for each走訪陣列元素(1/2)

- for迴圈的簡化版 for-each：

• for-each 敘述的語法

語法	說明

`for(type var: arrName){`
　敘述主體
`}`
	走訪陣列 arrName 裡的每一個元素，其中變數 var 代表走訪到的元素，其型別和 arrName 內的元素型別相同。

for each走訪陣列元素(2/2)

- 利用 for-each 比較陣列元素值的大小：

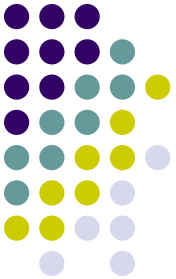
```
01  // Ch6_3, 比較陣列元素值的大小
02  public class Ch6_3{
03      public static void main(String[] args){
04          int arr[]={17,48,30,74,62};  // 宣告整數陣列 a,並設定初值
05          int max=arr[0];    // 將 max 設值為陣列的第 0 個元素
06
07          for(int i:arr){
08              if(i>max)
09                  max=i;    // 將 max 設值為目前找到的最大值
10          }
11          System.out.printf("Maximum is %d",max);  // 印出最大值
12      }
13  }
```

- 執行結果：

```
Maximum is 74
```

可寫成一般的for迴圈

```
07      for(int i; i<arr.length; i++){
08          if(arr[i]>max)
09              max=arr[i];    // 將 max 設值為目前找到的最大值
10      }
```



二維陣列的宣告

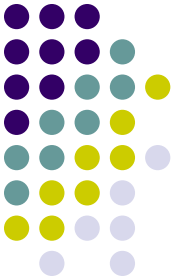
- 二維陣列的宣告與配置記憶空間的格式：

二維陣列的宣告與配置記憶體

```
資料型別[][] 陣列名稱;                    // 宣告二維陣列
陣列名稱 = new 資料型別 [列數][行數];     // 配置記憶體空間
```

- 如下面的範例：

```
int score[][];          // 宣告整數陣列 score
score=new int[4][3]; // 配置一塊記憶體空間，以供 4 列 3 行的整數陣列 score 使用
```



另一種宣告方式

- 以較為簡潔的方式來宣告陣列：

二維陣列的宣告與配置記憶體

```
資料型別[][] 陣列名稱=new 資料型別 [列數][行數];   // 宣告和配置記憶體空間
```

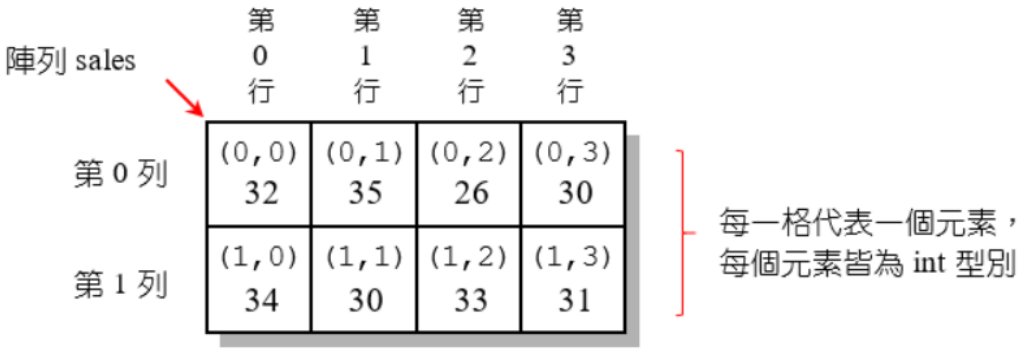
- 下面是二維陣列的宣告範例：

```
int score[][]=new int[4][3];  // 宣告整數陣列 score，同時配置一塊記憶體空間
```

二維陣列的實例 (1/2)

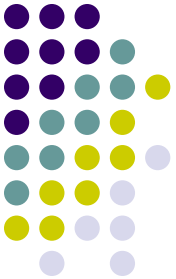
- 下表為某汽車銷售公司的車輛銷售量：

業務員	年度銷售量			
	第一季	第二季	第三季	第四季
1	32	35	26	30
2	34	30	33	31



- 上面的資料可用二維陣列儲存，宣告方式為

```
int[][] sales=new int[2][4];  // 宣告整數陣列 sales，同時配置記憶體空間
```



二維陣列的實例 (2/2)

- 二維陣列的宣告與配置記憶空間的格式：

二維陣列並設定初值的語法

```
資料型別[][] 陣列名稱={{ 第 0 列初值 },        // 二維陣列初值的設定
                       { 第 1 列初值 },
                       {  …          },
                       { 第 n 列初值 }};
```

```
int[][] sales={{32,35,26,30},
               {34,30,33,31}};
```

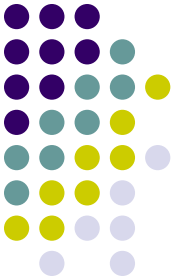
2×4 的陣列是由 2 個具有 4 個元素的一維陣列所組成

```
int[][] sales={{32,35,26,30},{34,30,33,31}};
```

2×4 的陣列

一維陣列，有 4 個元素

一維陣列，有 4 個元素



取得列數與特定列之元素的個數

- 取得二維陣列的列數，以及特定列之元素個數的語法

取得二維陣列的列數與特定元素個數之語法	
陣列名稱.length	// 取得陣列的列數
陣列名稱[列索引].length	// 取得特定列元素的個數

- 如下面的程式片段：

```
sales.length        // 取得陣列 sales 的列數，其值為 2
sales[0].length     // 取得陣列 sales 第 0 列元素的個數，其值為 4（也就是行數）
sales[1].length     // 取得陣列 sales 第 1 列元素的個數，其值為 4（也就是行數）
```

二維陣列元素的引用及存取

- 二維陣列的存取範例：

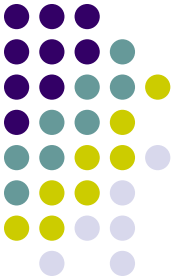
```
01  // Ch6_4, 二維陣列的存取範例
02  public class Ch6_4{
03      public static void main(String[] args){
04          int sum=0;
05          int[][] sales={{32,35,26,30},{34,30,33,31}}; // 宣告陣列並設定初值
06
07          for(int r=0;r<sales.length;r++){
08              for(int c=0;c<sales[r].length;c++){
09                  System.out.print(sales[r][c]+" ");   // 印出銷售量
10                  sum+=sales[r][c];     // 加總銷售量
11              }
12              System.out.println();     // 換行
13          }
14          System.out.printf("總銷售量為 %d 部車",sum);
15      }
16  }
```

• 執行結果：

```
32 35 26 30
34 30 33 31
總銷售量為 251 部車
```

可寫成for-each迴圈

```
07          for(int[] row:sales){// 輸出銷售量並計算總銷售量
08              for(int n:row){
09                  System.out.printf("%3d",n);
10                  sum+=n;
11              }
12              System.out.println();      // 列印換行
13          }
```



每列的元素個數不同的二維陣列

- arr[]為每列元素個數不同的二維陣列：

```
int[][] arr={{31,12,14,11},          // 每列元素個數均不同的二維陣列
             {33,34,30},
             {12,81,32,14,17}};
```

- 宣告每列元素個數不同的二維陣列，但不設定初值：

```
int[][] arr=new int[3][];      // 宣告二維陣列，並指定它有 3 列
arr[0]=new int[4];             // 指定第 0 列有 4 個元素
arr[1]=new int[3];             // 指定第 1 列有 3 個元素
arr[2]=new int[5];             // 指定第 2 列有 5 個元素
```

- 為個別元素設值：

```
arr[0][1]=12;            // 設定第 0 列，第 1 個元素值為 12
arr[1][0]=33;            // 設定第 1 列，第 0 個元素值為 33
arr[2][4]=17;            // 設定第 2 列，第 4 個元素值為 17
```

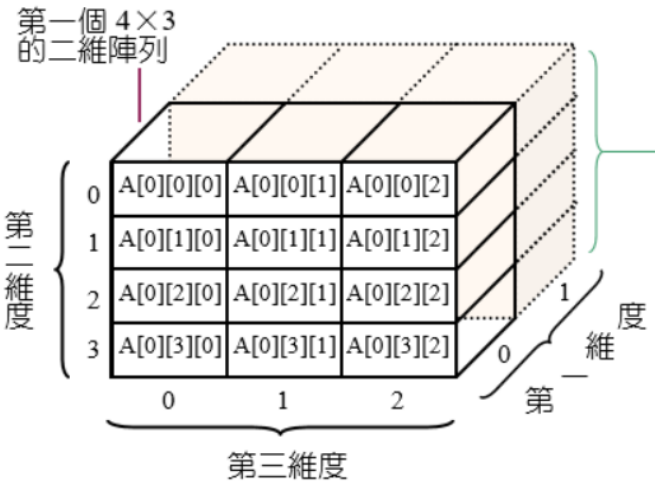
三維陣列

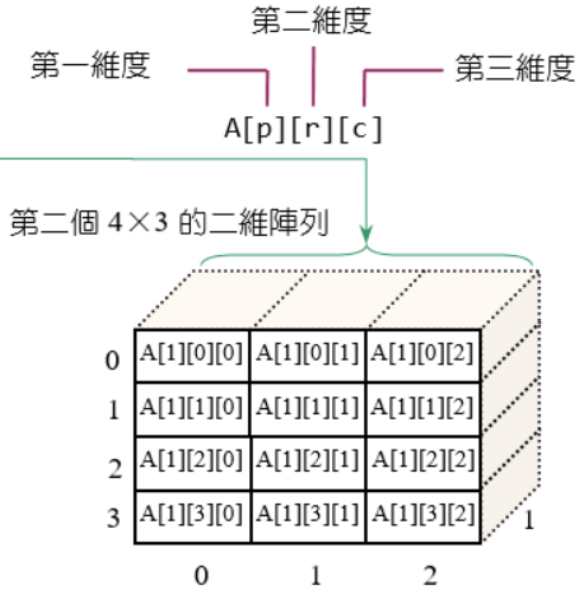
- 三維陣列的宣告範例：

```
int[][][] arr;              // 宣告三維陣列 arr
arr=new int[2][4][3];       // 2×4×3 整數陣列 arr
```

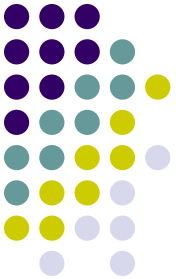
2×4×3的三維陣列可看成是由2個4×3的二維陣列所組成

也就是兩組4個橫列，3個直行的積木併在一起，組成一個立方體






```
int[][][] arr=new int[2][4][3];
```



找出最大值的範例 (1/3)

```
01  // Ch6_5, 在三維陣列裡找出最大值
02  public class Ch6_5{
03     public static void main(String args[]){
04        int arr[][][]={{{21,32,65},
05                        {78,94,76},
06                        {79,44,65},
07                        {89,54,73}},
08                       {{32,56,89},
09                        {43,23,32},
10                        {32,56,78},
11                        {94,78,45}}};
12        int p,r,c,max=arr[0][0][0];     // 設定 max 為陣列 arr 的第一個元素
13        for(p=0;p<arr.length;p++)              // 外層迴圈
14           for(r=0;r<arr[p].length;r++)        // 中層迴圈
15              for(c=0;c<arr[p][r].length;c++)  // 內層迴圈
16                 if(max<arr[p][r][c])
17                    max=arr[p][r][c];
18        System.out.println("max="+max);      // 印出陣列的最大值
19     }
20  }
```

設定 2×4×3 陣列的初值

利用三個 for 迴圈找出陣列的最大值

- 執行結果：

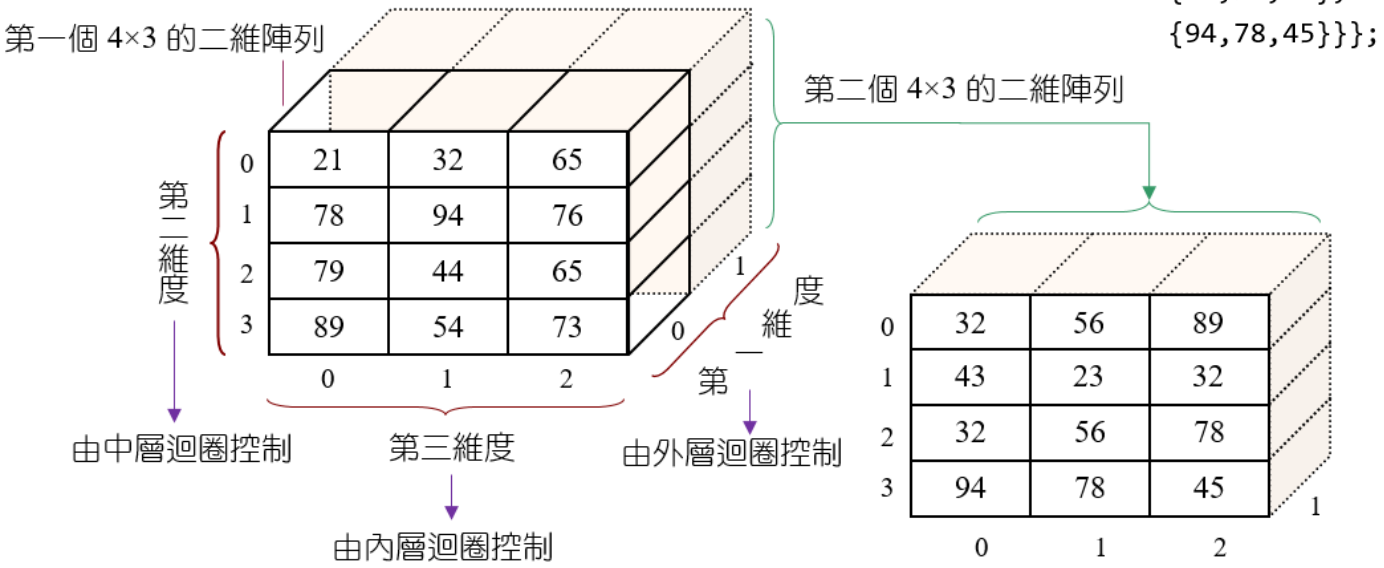
```
max=94
```

找出最大值的範例 (2/3)

- Ch6_5中三維陣列arr的示意圖：

```
int arr[][][]={{{21,32,65},
                {78,94,76},
                {79,44,65},
                {89,54,73}},
               {{32,56,89},
                {43,23,32},
                {32,56,78},
                {94,78,45}}};
```

設定2×4×3陣列的初值

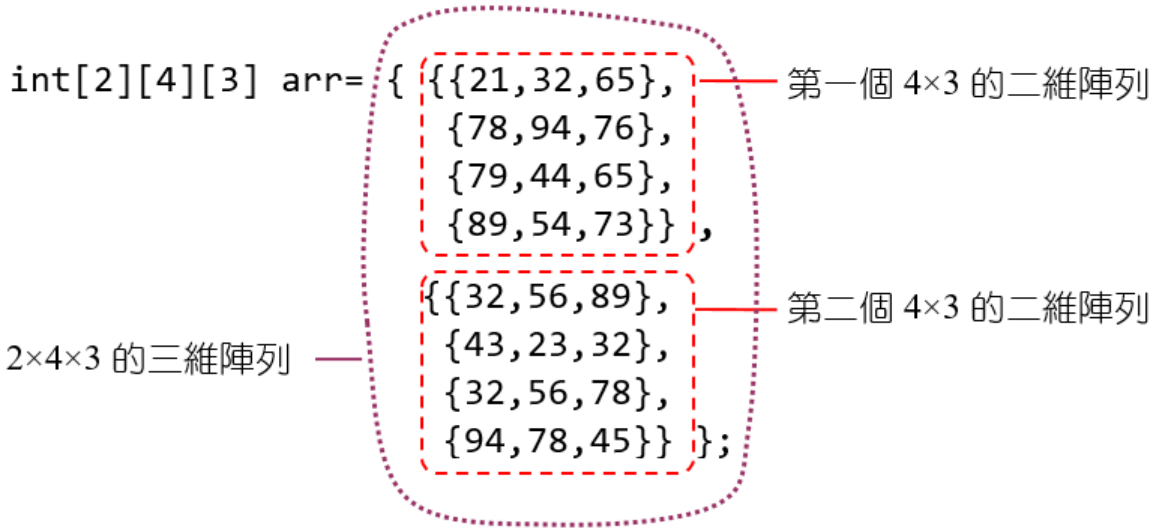


找出最大值 (3/3)

- 2×4×3的三維陣列可以寫成

  2×4×3的三維陣列 = { 4×3的二維陣列，4×3的二維陣列 }

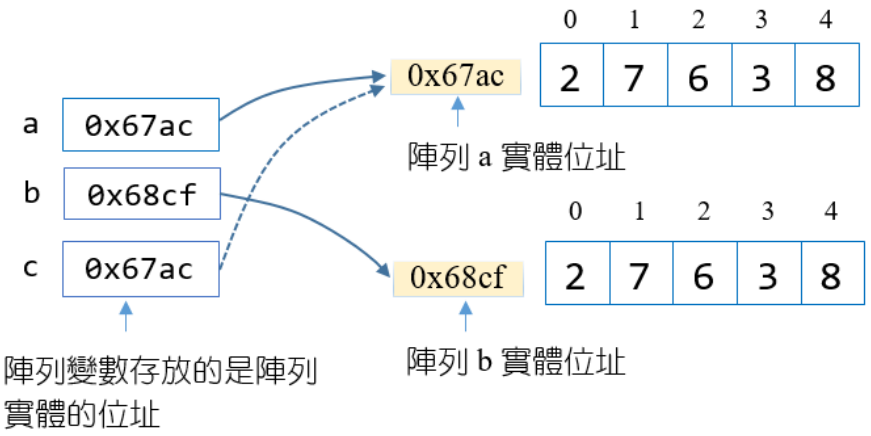
- 陣列arr初值的設定可用下圖來表示：

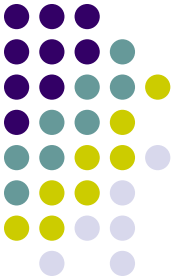


比較陣列的參考位置(1/2)

- 陣列名稱是個變數，存放的內容是陣列的實體位址
- 用「==」比較兩個陣列變數是否指向同一個陣列

```
01   int[] a={2,7,6,3,8};
02   int[] b={2,7,6,3,8};
03   int[] c=a;
```





比較陣列的參考位置(2/2)

- 下面的範例中，讓陣列變數c指向陣列變數a，更改c的內容同時也會更改到a的內容

```
01  // Ch6_6, 陣列變數的使用
02  public class Ch6_6{
03      public static void main(String[] args){
04          int[] a={2,7,6,3,8,4};    // 宣告陣列 a
05          int[] b={2,7,6,3,8,4};    // 宣告陣列 b
06          int[] c=a;                // 設定陣列變數 c 的位址指向 a
07
08          c[0]=10;   // 將 c[0]修改為 10
09          System.out.printf("a[0]=%d\n",a[0]);
10          System.out.printf("a==b: %b\n",a==b); // 判斷 a 與 b 是否相同
11          System.out.printf("a==c: %b\n",a==c); // 判斷 a 與 c 是否相同
12      }
13  }
```

- 執行結果：

```
a[0]=10
a==b: false
a==c: true
```


-The End-