

陣列（二）

人工智慧與無線感應設備開發專班

湜憶電腦知訊顧問股份有限公司

馬傳義

排序

- 在日常生活中，最常需要處理的東西就是找資料與更新資料；尤其當資料量很大時，為了尋找資料常花費很多時間。
- 如果將資料適當分類，同類資料依特定方式排列，將來要尋找時會節省很多時間，讓別人覺得做事很有效率。
- 『排序（Sorting）』就是將多筆資料，以某一個項目（或稱欄位）當作『鍵值（key value）』，讓資料依序排列。

排序

■ 通常鍵值由小而大（遞增）或者由大而小（遞減）來排列資料；經此種方式排序，將來要搜尋（Searching）就會很快。

◆ 學校老師的點名冊或成績登記冊是依照學生的學號（鍵值）由小而大排序，老師要點名或者登錄成績會比較快。

◆ 入學分發是依照考試成績總分（鍵值）由高而低排序，由分數最高先決定所要就讀的學校與系所。

■ 排序的方法有很多種，各有其優缺點，先介紹最簡單且易懂的『**氣泡排序法（Bubble Sort）**』，其他方法請參考『資料結構（Data Structure）』相關的書籍。

排序

氣泡排序法（Bubble Sort）

- ◆ 氣泡排序法簡單易懂，但比較沒有效率。
 - 當資料量少時，與其他排序的方法比較，在排序的時間花費上沒有明顯的不同。
 - 當資料量以千萬計時，氣泡排序所花費的時間就會有很明顯差別。
- ◆ 氣泡排序法是採用相鄰資料的鍵值做比較，使資料的鍵值由左而右排列時，能由鍵值較小者排前面，而鍵值較大者排後面。
- ◆ 其進行的方式是由左而右進行兩兩比較，當左邊資料的鍵值比右邊資料的鍵值大時，即進行交換工作。
 - 在第一次循環時，鍵值最大的資料會移到最右邊。

排序

- 第二次循環時，鍵值第二大的資料，移到最右邊算過來的第二位。
 - 依此類推...
 - 最後，鍵值最小的資料會排在最左邊。
- ◆ 若有 n 個資料要做氣泡排序時，循環次數為『 $(n-1)$ 』次，比較次數則為『 $n(n-1)/2$ 』次。

排序

◆ 程式：

```
public class CH04_10
{
    public static void main(String[] args)
    {
        int[] A = { 18, 14, 12, 16, 10 };
        int i, j, k, t;
        System.out.print("A陣列排序前的順序：");
        for (i = 0; i < A.length; i++)
            System.out.print(A[i] + " ");

        System.out.println("\n");

        for (i = 3; i >= 0; i--)
        {
            for (j = 0; j <= i; j++)
```

排序

```
if (A[j] > A[j + 1])  
{  
    t = A[j + 1];  
    A[j + 1] = A[j];  
    A[j] = t;  
}
```

```
System.out.print("第 " + (4 - i) + " 次 : ");
```

```
for (k = 0; k < A.length; k++)
```

```
    System.out.print(A[k] + " ");
```

```
System.out.println();
```

```
}
```

```
}
```

```
}
```

排序

```
1 public class CH04_10
2 {
3     public static void main(String[] args)
4     {
5         int[] A = { 18, 14, 12, 16, 10 };
6         int i, j, k, t;
7         System.out.print("A陣列排序前的順序：");
8         for (i = 0; i < A.length; i++)
9             System.out.print(A[i] + "\t");
10
11         System.out.println("\n");
12
13         for (i = 3; i >= 0; i--)
14         {
15             for (j = 0; j <= i; j++)
16                 if (A[j] > A[j + 1])
17                 {
18                     t = A[j + 1];
19                     A[j + 1] = A[j];
20                     A[j] = t;
21                 }
22
23             System.out.print(" 第 " + (4 - i) + " 次：");
24             for (k = 0; k < A.length; k++)
25                 System.out.print(A[k] + " ");
26
```


排序

```
27     System.out.println();  
28     }  
29 }  
30 }
```

◆ 執行結果：

A陣列排序前的順序：18 14 12 16 10

第1次：14 12 16 10 18
第2次：12 14 10 16 18
第3次：12 10 14 16 18
第4次：10 12 14 16 18

◆ 說明：

- 行05：
 - ◆ 宣告一維整數陣列A，並同時指定陣列的初值。
- 行06：
 - ◆ 宣告變數。

排序

- 行07：
 - ◆ 螢幕輸出。
- 行08~行09：
 - ◆ 利用for迴圈，輸出A陣列中各個元素的值。
 - ◆ 陣列的長度為5（即 `A.length=5`）。
- 行11：
 - ◆ 螢幕輸出。
- 行13~行28：
 - ◆ 利用巢狀for迴圈及if，進行氣泡排序法，並顯示每一次排序的結果。
 - ◆ 行13：
 - ◆ 設定外層for迴圈（即循環的次數；計算方式為『(n-1)』次）。

排序

- ◆ 行15：

- ◆ 設定內層for迴圈（即 比較的次數；計算方式為『 $n(n-1)/2$ 』次）。

- ◆ 行16~行21：

- ◆ 進行氣泡排序法。

行16：

判斷左邊的資料是否大於右邊，

行18：

若左邊的資料大於右邊，則將右邊的資料，存入暫存變數中。

行19：

將左邊的資料存入右邊的資料位置中。

行20：

將暫存變數中的資料，存入左邊的位置中。

排序

- ◆ 行23：
 - ◆ 螢幕輸出。
- ◆ 行24~行25：
 - ◆ 利用for迴圈，輸出每次循環後，A陣列中各個元素的值。
 - ◆ 陣列的長度為5（即 `A.length=5`）。
- ◆ 行27：
 - ◆ 螢幕輸出。

排序

◆ 課堂練習：

- 請將此 (CH04_10) 程式，更改符合下列條件：
 - 1、『陣列長度』由『使用者輸入』，但必須限制在2~10之間。
 - 2、陣列的『元素值』由『使用者輸入』，但必須限制在1~100之間。
 - 3、輸入完成的陣列元素值，須進行降冪排序（最左邊的數字最大，最右邊的數字最小）。

搜尋

- 『搜尋 (Search)』的目的，是要由大量的資料中，找出特定的資料。
 - ◆ 從學生資料中，找出特定學生的電話或E-mail，即是一例。
- 搜尋方法也有很多種，在此，先介紹『**線性搜尋法 (Linear Search)**』與『**二分搜尋法 (Binary Search)**』，其他方法請參考『資料結構 (Data Structure)』相關的書籍。

搜尋

線性搜尋法（Linear Search）

- ◆ 或稱『循序搜尋法（Sequential Search）』。
- ◆ 依所要搜尋資料的鍵值，由最前面資料逐筆比較，若有鍵值相同，表示已找到資料。
- ◆ 若全部比較完畢，而沒有找到相同鍵值時，表示要搜尋資料不存在。
- ◆ 當有 n 筆資料要搜尋時，平均要比較『 $n/2$ 』次才能找到資料。
 - 實際上，所要搜尋的鍵值在所有資料的較前面，則搜尋時間較短；若所要搜尋的鍵值在所有資料的較後面，則搜尋時間較長。

搜尋

◆ 程式：

```
import java.io.*;
```

```
public class CH04_11
```

```
{
```

```
    public static void main(String[] args) throws IOException
```

```
{
```

```
    int[] account = { 18, 14, 12, 16, 10 };
```

```
    String[] name = { "王五", "張三", "陳二", "李四", "鄭一" };
```

```
    System.out.println("員工編號\t姓名");
```

```
    for (int i = 0; i < account.length; i++)
```

```
{
```

```
        System.out.print(account[i] + "\t\t");
```

```
        System.out.println(name[i]);
```

```
}
```


搜尋

```
int i, num, search_num;

System.out.print("\n使用『線性搜尋法』，請輸入要找的員工編號：");
BufferedReader keyin;
keyin = new BufferedReader(new InputStreamReader(System.in));
search_num = Integer.parseInt(keyin.readLine());
num = -1;
for (i = 0; i < account.length; i++)
    if (account[i] == search_num)
    {
        num = i;
        break;
    }
```

搜尋

```
if (num == -1)
    System.out.println("\n 查無此編號");
else
{
    System.out.println("\n員工編號\t 姓名");
    System.out.println(account[num] + "\t\t\t" + name[num]);
}
}
```

搜尋

```
1 import java.io.*;
2
3 public class CH04_11
4 {
5     public static void main(String[] args) throws IOException
6     {
7         int[] account = { 18, 14, 12, 16, 10 };
8         String[] name = { "王五", "張三", "陳二", "李四", "鄭一" };
9
10        System.out.println("員工編號\t姓名");
11        for (int i = 0; i < account.length; i++)
12        {
13            System.out.print(account[i] + "\t\t");
14            System.out.println(name[i]);
15        }
16
17        int i, num, search_num;
18
19        System.out.print("\n使用「線性搜尋法」，請輸入要找的員工編號：");
20        BufferedReader keyin;
21        keyin = new BufferedReader(new InputStreamReader(System.in));
22        search_num = Integer.parseInt(keyin.readLine());
23        num = -1;
24        for (i = 0; i < account.length; i++)
25            if (account[i] == search_num)
26            {
```

搜尋

```
27         num = i;
28         break;
29     }
30
31     if (num == -1)
32         System.out.println("\n 查無此編號");
33     else
34     {
35         System.out.println("\n員工編號\t 姓名");
36         System.out.println(account[num] + "\t\t\t" + name[num]);
37     }
38 }
39 }
```

搜尋

◆ 執行結果：

員工編號	姓名
18	王五
14	張三
12	陳二
16	李四
10	鄭一

使用「線性搜尋法」，請輸入要找的員工編號：15

查無此編號

員工編號	姓名
18	王五
14	張三
12	陳二
16	李四
10	鄭一

使用「線性搜尋法」，請輸入要找的員工編號：18

員工編號	姓名
18	王五

搜尋

◆ 說明：

- 行01：
 - ◆ 載入Java.io.*套件。
- 行05：
 - ◆ main()方法後面，加上 throws IOException。
- 行07：
 - ◆ 宣告一維整數陣列account，並同時指定陣列的初值。
- 行08：
 - ◆ 宣告一維字串陣列name，並同時指定陣列的初值。
- 行10：
 - ◆ 螢幕輸出。

搜尋

- 行11~行15：
 - ◆ 利用for迴圈，輸出account及name陣列中各個元素的值。
 - ◆ 因為兩個陣列的長度相同，所以用一個for迴圈即可輸出兩個陣列的資料。
 - ◆ 行11：
 - ◆ 利用for迴圈，輸出A陣列中各個元素的值。
 - ◆ 陣列的長度為5（即 `account.length=5`）。
 - ◆ 行13~行14：
 - 螢幕輸出。
- 行17：
 - ◆ 宣告變數。
- 行19：
 - ◆ 螢幕輸出。

搜尋

- 行20：
 - ◆ 宣告BufferedReader類別的物件。
- 行21：
 - ◆ 建立keyin物件。
- 行22：
 - ◆ 利用keyin物件的readLine()方法，讀取由鍵盤輸入的字串，透過Integer.parseInt的方法，轉換成整數後，指派給整數變數search_num。
- 行23：
 - ◆ 設定num變數的值為-1（表示預設資料未找到）。
- 行24~行29：
 - ◆ 進行線性搜尋法。
 - ◆ 行24：
 - ◆ 利用for迴圈，設定搜尋的次數。
 - ◆ 陣列的長度為5（即 `account.length=5`）。

搜尋

- ◆ 行25~行29：
判斷是否找尋到符合的資料。
行25：
若是找尋到符合的資料，
行27：
將符合資料的陣列索引，存入num變數中。
行28：
中斷迴圈的執行，跳到迴圈外的下一行，繼續執行。
- 行31~行37：
 - ◆ 判斷螢幕輸出的資料為何。
 - ◆ 行31：
 - ◆ 若num變數中的值為-1（即未搜尋到符合的資料），
 - ◆ 行32：
螢幕輸出。

搜尋

- ◆ 行33：
 - ◆ 若num變數中的值不是-1（即 搜尋到符合的資料），
 - ◆ 行35~行36：
輸出找尋到的資料。

搜尋

二分搜尋法 (Binary Search)

- ◆ 是一種比較有效率的搜尋法。
- ◆ 此方法需先將資料依鍵值做排序，若未經排序的資料無法應用此搜尋法。
- ◆ 二分搜尋法是將 n 筆資料，先依鍵值由小而大排序後，儲存於陣列中（當然也可以由大而小排序）。
- ◆ 然後從已排序好的 n 筆資料之中間（即第 $n/2$ 筆）開始搜尋比較。
- ◆ 如果比較結果相同，表示已找到；若不同，則再從比搜尋值大或小的資料中間找起（即第 $n/4$ 筆或第 $3/4 n$ 筆）...以此類推。

搜尋

- ◆ 如果有 n 筆資料，線性搜尋法平均需要『 $n/2$ 』次比較才能找到資料；二分搜尋法，最多需要『 $\log_2 n + 1$ 』的比較次數，就可以找到資料。

例如：

有1024筆料，則最多需11次的比較，便能找到所需資料。

搜尋

◆ 程式：

```
import java.io.*;

public class CH04_12
{
    public static void main(String[] args) throws IOException
    {
        int[] account = { 18, 14, 12, 16, 10 };
        String[] name = { "王五", "張三", "陳二", "李四", "鄭一" };

        int i, j, account_t;
        String name_t;

        System.out.println("排序前的資料順序：\n");
        System.out.println("員工編號\t姓名");
        for (i = 0; i < account.length; i++)
            System.out.println(" " + account[i] + "\t\t\t" + name[i]);
    }
}
```

搜尋

```
for (i = 3; i >= 0; i--)  
    for (j = 0; j <= i; j++)  
        if (account[j] > account[j + 1])  
        {  
            account_t = account[j + 1];  
            account[j + 1] = account[j];  
            account[j] = account_t;  
            name_t = name[j + 1];  
            name[j + 1] = name[j];  
            name[j] = name_t;  
        }
```

System.out.println("\n使用『氣泡排序法』後的資料排序順序：\n");

System.out.println("員工編號\t姓名");

```
for (i = 0; i < account.length; i++)
```

```
    System.out.println(" " + account[i] + "\t\t" + name[i]);
```

搜尋

```
System.out.print("\n使用『二分搜尋法』，請輸入要找的員工編號：");
BufferedReader keyin;
keyin = new BufferedReader(new InputStreamReader(System.in));
int search_account = Integer.parseInt(keyin.readLine());

int low = 0, high = account.length - 1, mid_num = (low + high) / 2, num = -1;

while (true)
{
    System.out.println("low=" + low + " " + "high=" + high + " " +
                       "mid_num=" + mid_num);

    if (account[mid_num] == search_account)
    {
        num = mid_num;
        break;
    }
}
```

搜尋

```
if (low == high || mid_num < low || mid_num > high)
{
    num = -1;
    break;
}

if (account[mid_num] > search_account)
    high = mid_num - 1;
else
    low = mid_num + 1;

mid_num = (low + high) / 2;
}
```


搜尋

```
if (num == -1)
    System.out.println("\n查無此編號");
else
{
    System.out.println("\n 編號\t姓名");
    System.out.println(" " + account[num] + "\t\t" + name[num]);
}
}
```

搜尋

```
1 import java.io.*;
2
3 public class CH04_12
4 {
5     public static void main(String[] args) throws IOException
6     {
7         int[] account = { 18, 14, 12, 16, 10 };
8         String[] name = { "王五", "張三", "陳二", "李四", "鄭一" };
9
10        int i, j, account_t;
11        String name_t;
12
13        System.out.println("排序前的資料順序：\n");
14        System.out.println("員工編號\t姓名");
15        for (i = 0; i < account.length; i++)
16            System.out.println(" " + account[i] + "\t\t" + name[i]);
17
18        for (i = 3; i >= 0; i--)
19            for (j = 0; j <= i; j++)
20                if (account[j] > account[j + 1])
21                {
22                    account_t = account[j + 1];
23                    account[j + 1] = account[j];
24                    account[j] = account_t;
```

搜尋

```
25         name_t = name[j + 1];
26         name[j + 1] = name[j];
27         name[j] = name_t;
28     }
29
30     System.out.println("\n使用「氣泡排序法」後的資料排序順序：\n");
31     System.out.println("員工編號\t姓名");
32     for (i = 0; i < account.length; i++)
33         System.out.println(" " + account[i] + "\t\t" + name[i]);
34
35     System.out.print("\n使用「二分搜尋法」，請輸入要找的員工編號：");
36     BufferedReader keyin;
37     keyin = new BufferedReader(new InputStreamReader(System.in));
38     int search_account = Integer.parseInt(keyin.readLine());
39
40     int low = 0, high = account.length - 1, mid_num = (low + high) / 2, num = -1;
41
42     while (true)
43     {
44         System.out.println("low=" + low + " " + "high=" + high + " " + "mid_num=" + mid_num);
45
46         if (account[mid_num] == search_account)
47         {
48             num = mid_num;
49             break;
```

搜尋

```
50     }
51
52     if (low == high || mid_num < low || mid_num > high)
53     {
54         num = -1;
55         break;
56     }
57
58     if (account[mid_num] > search_account)
59         high = mid_num - 1;
60     else
61         low = mid_num + 1;
62
63     mid_num = (low + high) / 2;
64 }
65
66 if (num == -1)
67     System.out.println("\n查無此編號");
68 else
69 {
70     System.out.println("\n 編號\t姓名");
71     System.out.println(" " + account[num] + "\t\t" + name[num]);
72 }
73 }
74 }
```

搜尋

◆ 執行結果：

排序前的資料順序：

員工編號	姓名
18	王五
14	張三
12	陳二
16	李四
10	鄭一

使用「氣泡排序法」後的資料排序順序：

員工編號	姓名
10	鄭一
12	陳二
14	張三
16	李四
18	王五

使用「二分搜尋法」，請輸入要找的員工編號：15

low=0 high=4 mid_num=2

low=3 high=4 mid_num=3

low=3 high=2 mid_num=2

查無此編號

搜尋

排序前的資料順序：

員工編號	姓名
18	王五
14	張三
12	陳二
16	李四
10	鄭一

使用「氣泡排序法」後的資料排序順序：

員工編號	姓名
10	鄭一
12	陳二
14	張三
16	李四
18	王五

使用「二分搜尋法」，請輸入要找的員工編號：12

low=0 high=4 mid_num=2
low=0 high=1 mid_num=0
low=1 high=1 mid_num=1

編號	姓名
12	陳二

搜尋

- 行01：
 - ◆ 載入Java.io.*套件。
- 行05：
 - ◆ main()方法後面，加上 throws IOException。
- 行07：
 - ◆ 宣告一維整數陣列account，並同時指定陣列的初值。
- 行08：
 - ◆ 宣告一維字串陣列name，並同時指定陣列的初值。
- 行10：
 - ◆ 宣告整數變數。
- 行11：
 - ◆ 宣告字串變數。

搜尋

- 行13~行14：
 - ◆ 螢幕輸出。
- 行15~行16：
 - ◆ 利用for迴圈，輸出account及name陣列中各個元素的值。
 - ◆ 因為兩個陣列的長度相同，所以用一個for迴圈即可輸出兩個陣列的資料。
 - ◆ 行15：
 - ◆ 利用for迴圈，輸出account及name陣列中各個元素的值。
 - ◆ 陣列的長度為5（即 `account.length=5`）。
 - ◆ 行16：
 - ◆ 螢幕輸出。
- 行18~行28：
 - ◆ 利用巢狀for迴圈及if，進行氣泡排序法，並顯示每一次排序的結果。

搜尋

- ◆ 行18：

- ◆ 設定外層for迴圈（即 循環的次數）。

- ◆ 行19：

- ◆ 設定內層for迴圈（即 比較的次數）。

- ◆ 行20~行28：

- ◆ 進行氣泡排序法。

- 行20：

- 判斷account陣列左邊的資料是否大於右邊，

- 行22：

- 若account陣列左邊的資料大於右邊，則將右邊的資料，存入暫存變數中。

- 行23：

- 將account陣列左邊的資料存入右邊的資料位置中。

- 行24：

- 將暫存變數中的資料，存入account陣列左邊的位置中。

搜尋

行25：

將name陣列右邊的資料，存入暫存變數中。

行26：

將name陣列左邊的資料存入右邊的資料位置中。

行27：

將暫存變數中的資料，存入name陣列左邊的位置中。

注意：

因account陣列與name陣列有相關聯（即員工編號與姓名），故account陣列有交換動作時，name陣列必須跟著要有交換動作。

- 行30~行31：

- ◆ 螢幕輸出。

- 行32~行33：

- ◆ 利用for迴圈，輸出account及name陣列中各個元素的值。

搜尋

- ◆ 因為兩個陣列的長度相同，所以用一個for迴圈即可輸出兩個陣列的資料。
- ◆ 行32：
 - ◆ 利用for迴圈，輸出account即name陣列中各個元素的值。
 - ◆ 陣列的長度為5（即 `account.length=5`）。
- ◆ 行33：
 - ◆ 螢幕輸出。
- 行35：
 - ◆ 螢幕輸出。
- 行36：
 - ◆ 宣告BufferedReader類別的物件。
- 行37：
 - ◆ 建立keyin物件。

搜尋

- 行38：
 - ◆ 利用keyin物件的readLine()方法，讀取由鍵盤輸入的字串，透過Integer.parseInt的方法，轉換成整數後，指派給整數變數search_num。
- 行40：
 - ◆ 宣告變數，並指定初值。
- 行42~64：
 - ◆ 進行二分搜尋法。
 - 行42：
利用while迴圈，進行二分搜尋法。
While的條件式為『true』，故會讓此迴圈成為『無窮迴圈』，此為特殊用法。
 - 行44：
螢幕輸出二分搜尋法執行的過程及次數。

搜尋

- ◆ 行46~行50：

- ◆ 判斷所有資料筆數的第1/2筆資料，是否符合要搜尋的資料。

行46：

若第1/2筆資料，符合要搜尋的資料，則

行48：

將該筆資料的索引值，寫入num變數中。

行49：

中斷while迴圈的執行，跳到while迴圈外的下一行，繼續執行。

- ◆ 行52~行56：

- ◆ 判斷是否所有資料都不符合要搜尋的資料。

行52：

若『第一筆資料的索引值與最後一筆資料的索引值相同』或『中間資料的索引值小於第一筆資料的索引值』或『中間資料的索引值大於最後一筆資料的索引值』，則

搜尋

行54：

將-1寫入num變數中（-1表示未找到符合的資料）。

行55：

中斷while迴圈的執行，跳到while迴圈外的下一行，繼續執行。

◆ 行58~行61：

◆ 判斷若要繼續搜尋下一循環，應該是要搜尋第1/2筆資料的左邊（要搜尋資料的索引值比第1/2筆資料的索引值小），還是右邊（要搜尋資料的索引值比第1/2筆資料的索引值大）。

行58：

若要搜尋資料的索引值比第1/2筆資料的索引值大，則

行59：

下一循環搜尋的最後一筆資料的索引值為『第1/2筆資料的索引值-1』。

搜尋

行61：

下一循環搜尋的最後一筆資料的索引值為『第1/2筆資料的索引值+1』。

- ◆ 行63：

- ◆ 計算下一循環搜尋時，第1/2資料的索引值。

- 行66~行72：

- ◆ 判斷是否有搜尋到符合的資料。

- ◆ 行66：

- ◆ 若沒有搜尋到符合的資料，則

- ◆ 行67：

- 螢幕輸出。

- ◆ 行68：

- ◆ 若有搜尋到符合的資料，則

- ◆ 行70~行71：

- 螢幕輸出。

Arrays類別

- Arrays類別是屬於「java.util」套件中。
- 其中包含一些靜態方法可以直接呼叫、直接使用。
 -
- Arrays提供許多對於陣列的處理方法。
 - 例如：
 - 排序、尋找、複製、填滿及比對等。

Arrays類別

 （整個陣列的）排序：

◆ 語法：

Arrays.sort(陣列名稱);

◆ 用途：

- 對指定的陣列，做『由小至大』的排序動作。

◆ 說明：

- 陣列名稱：
 - ◆ 要排序的陣列名稱。

例如：

Arrays.sort(A);

Arrays類別

注意：

- 傳回值：
 - ◆ 無。
- 適用的資料型別：
 - ◆ byte、short、int、long、float、double、Object及char。

Arrays類別

◆ 程式：

```
import java.util.Arrays;

public class CH04_13
{
    public static void main(String[] args)
    {
        int[] account = { 18, 14, 12, 16, 10 };

        System.out.println("排序前的資料順序：");
        for (int i = 0; i < account.length; i++)
            System.out.print(account[i] + "\t");

        Arrays.sort(account);
```

Arrays類別

```
System.out.println("\n\n排序後的資料順序：");  
for (int i = 0; i < account.length; i++)  
    System.out.print(" " + account[i] + "t");  
}  
}
```

Arrays類別

```
1 import java.util.Arrays;
2
3 public class CH04_13
4 {
5     public static void main(String[] args)
6     {
7         int[] account = { 18, 14, 12, 16, 10 };
8
9         System.out.println("排序前的資料順序：");
10        for (int i = 0; i < account.length; i++)
11            System.out.print(account[i] + "\t");
12
13        Arrays.sort(account);
14
15        System.out.println("\n\n排序後的資料順序：");
16        for (int i = 0; i < account.length; i++)
17            System.out.print(" " + account[i] + "\t");
18    }
19 }
```

◆ 執行結果：

排序前的資料順序：
18 14 12 16 10

排序後的資料順序：
10 12 14 16 18

Arrays類別

◆ 說明：

- 行01：
 - ◆ 載入java.util.Arrays套件。
- 行07：
 - ◆ 宣告一維整數陣列account，並同時指定陣列的初值。
- 行09：
 - ◆ 螢幕輸出。
- 行10~行11：
 - ◆ 利用for迴圈，輸出account陣列中各個元素的值。
 - 行10：
利用for迴圈，輸出account陣列中各個元素的值。
account陣列的長度為6（即 account.length=6）。
 - 行11：
螢幕輸出。

Arrays類別


- 行13：
 - ◆ 針對account陣列的元素值，進行排序。
- 行15：
 - ◆ 螢幕輸出。
- 行16~行17：
 - ◆ 利用for迴圈，輸出account陣列中各個元素的值。
 - 行16：
 - 利用for迴圈，輸出account陣列中各個元素的值。
 - account陣列的長度為6（即 `account.length=6`）。
 - 行17：
 - 螢幕輸出。

Arrays類別

◆ 課堂練習：

- 請將此（CH04_13）程式，更改符合下列條件：
 - 1、『陣列長度』由『使用者輸入』，但必須限制在2~10之間。
 - 2、陣列的『元素值』由『使用者輸入』，但必須限制在1~100之間。
 - 3、輸入完成的陣列元素值，須進行降冪排序（最左邊的數字最大，最右邊的數字最小）。

Arrays類別

 （陣列中，指定範圍的）排序：

◆ 語法：

Arrays.sort(陣列名稱,起始索引,結束索引);

◆ 用途：

- 對指定陣列中，指定的範圍（『起始索引』到『結束索引』）做『由小至大』的排序動作。

◆ 說明：

- 陣列名稱：
 - ◆ 要排序的陣列名稱。
- 起始索引：
 - ◆ 排序範圍的起始索引值。
 - ◆ 排序範圍包含『起始索引』的位置。

Arrays類別

- 結束索引：

- ◆ 排序範圍的結束索引值。

- ◆ 排序範圍不包含『結束索引』的位置。

例如：

- ◆ `Arrays.sort(A,5,15);`

注意：

- 傳回值：

- ◆ 無。

- 適用的資料型別：

- ◆ `byte`、`short`、`int`、`long`、`float`、`double`、`Object`及`char`。

Arrays類別

◆ 程式：

```
import java.util.Arrays;
```

```
public class CH04_14
```

```
{
```

```
    public static void main(String[] args)
```

```
    {
```

```
        int[] account = { 18, 14, 12, 16, 10, 28, 22, 30, 26, 32, 36, 34, 24, 20 };
```

```
        System.out.println("排序前的資料順序：");
```

```
        for (int i = 0; i < account.length; i++)
```

```
            System.out.print(account[i] + " ");
```

```
        Arrays.sort(account, 5, 10);
```

Arrays類別

```
System.out.println("\n\n排序後的資料順序：");  
for (int i = 0; i < account.length; i++)  
    System.out.print(" " + account[i] + "t");  
}  
}
```

Arrays類別

```
1 import java.util.Arrays;
2
3 public class CH04_14
4 {
5     public static void main(String[] args)
6     {
7         int[] account = { 18, 14, 12, 16, 10, 28, 22, 30, 26, 32, 36, 34, 24, 20 };
8
9         System.out.println("排序前的資料順序：");
10        for (int i = 0; i < account.length; i++)
11            System.out.print(account[i] + "\t");
12
13        Arrays.sort(account, 5, 10);
14
15        System.out.println("\n\n排序後的資料順序：");
16        for (int i = 0; i < account.length; i++)
17            System.out.print(" " + account[i] + "\t");
18    }
19 }
```

◆ 執行結果：

排序前的資料順序：

18 14 12 16 10 28 22 30 26 32 36 34 24 20

排序後的資料順序：

18 14 12 16 10 22 26 28 30 32 36 34 24 20

Arrays類別

◆ 說明：

- 行01：
 - ◆ 載入java.util.Arrays套件。
- 行07：
 - ◆ 宣告一維整數陣列account，並同時指定陣列的初值。
- 行09：
 - ◆ 螢幕輸出。
- 行10~行11：
 - ◆ 利用for迴圈，輸出account陣列中各個元素的值。
 - 行10：
 - 利用for迴圈，輸出account陣列中各個元素的值。
 - account陣列的長度為14（即 `account.length=14`）。
 - 行11：
 - 螢幕輸出。

Arrays類別

- 行13：
 - ◆ 針對account陣列中，將指定範圍的元素值，進行排序。
- 行15：
 - ◆ 螢幕輸出。
- 行16~行17：
 - ◆ 利用for迴圈，輸出account陣列中各個元素的值。
 - ◆ account陣列的長度為14（即 `account.length=14`）。

Arrays類別

◆ 課堂練習：

- 請將此 (CH04_14) 程式，更改符合下列條件：
 - 1、『陣列長度』由『使用者輸入』，但必須限制在2~10之間。
 - 2、陣列的『元素值』由『使用者輸入』，但必須限制在1~100之間。
 - 3、輸入完成的陣列元素值，須由『使用者』指定要排序的範圍（包含使用者所輸入的起始位置與結束位置），且須進行由降冪排序（最左邊的數字最大，最右邊的數字最小）。

Arrays類別

二分搜尋法：

◆ 語法：

Arrays.binarySearch(陣列名稱, 搜尋值);

◆ 用途：

- 對指定陣列做二分搜尋；傳回『搜尋值』在指定陣列中的『索引值』。

◆ 說明：

- 陣列名稱
 - ◆ 要搜尋資料的陣列名稱。
- 搜尋值
 - ◆ 要搜尋值。

Arrays類別

例如：

```
int num = Arrays.binarySearch(A,15);
```

注意：

- 傳回值：
 - ◆ 整數。
 - ◆ 當傳回值 <0 表示未找到符合的資料。
- 適用的資料型別：
 - ◆ byte、short、int、long、float、double、Object及char。

Arrays類別

◆ 程式：

```
import java.io.*;
import java.util.Arrays;

public class CH04_15
{
    public static void main(String[] args) throws IOException
    {
        int[] account = { 18, 14, 12, 16, 10 };
        String[] name = { "王五", "張三", "陳二", "李四", "鄭一" };

        int i, j, account_t;
        String name_t;

        System.out.println("排序前的資料順序：\n");
        System.out.println("員工編號\t姓名");
        for (i = 0; i < account.length; i++)
            System.out.println(" " + account[i] + "\t\t" + name[i]);
    }
}
```

Arrays類別

```
for (i = 3; i >= 0; i--)  
    for (j = 0; j <= i; j++)  
        if (account[j] > account[j + 1])  
        {  
            account_t = account[j + 1];  
            account[j + 1] = account[j];  
            account[j] = account_t;  
            name_t = name[j + 1];  
            name[j + 1] = name[j];  
            name[j] = name_t;  
        }
```

```
System.out.println("\n使用『氣泡排序法』後的資料排序順序：\n");
```

```
System.out.println("員工編號\t姓名");
```

```
for (i = 0; i < account.length; i++)
```

```
    System.out.println(" " + account[i] + "\t\t" + name[i]);
```

```
System.out.print("\n使用『二分搜尋法』，請輸入要找的員工編號：");
```

Arrays類別

```
BufferedReader keyin;  
keyin = new BufferedReader(new InputStreamReader(System.in));  
int search_account = Integer.parseInt(keyin.readLine());  
  
int arrNum = Arrays.binarySearch(account, search_account);  
  
if (arrNum < 0)  
    System.out.println("\n查無此編號");  
else  
{  
    System.out.println("\n 編號\t姓名");  
    System.out.println(" " + account[arrNum] + "\t\t" + name[arrNum]);  
}  
}
```

Arrays類別

```
1 import java.io.*;
2 import java.util.Arrays;
3
4 public class CH04_15
5 {
6     public static void main(String[] args) throws IOException
7     {
8         int[] account = { 18, 14, 12, 16, 10 };
9         String[] name = { "王五", "張三", "陳二", "李四", "鄭一" };
10
11         int i, j, account_t;
12         String name_t;
13
14         System.out.println("排序前的資料順序：\n");
15         System.out.println("員工編號\t姓名");
16         for (i = 0; i < account.length; i++)
17             System.out.println(" " + account[i] + "\t\t" + name[i]);
18
19         for (i = 3; i >= 0; i--)
20             for (j = 0; j <= i; j++)
21                 if (account[j] > account[j + 1])
22                 {
23                     account_t = account[j + 1];
24                     account[j + 1] = account[j];
25                     account[j] = account_t;
```

Arrays類別

```
26         name_t = name[j + 1];
27         name[j + 1] = name[j];
28         name[j] = name_t;
29     }
30
31     System.out.println("\n使用「氣泡排序法」後的資料排序順序：\n");
32     System.out.println("員工編號\t姓名");
33     for (i = 0; i < account.length; i++)
34         System.out.println(" " + account[i] + "\t\t" + name[i]);
35
36     System.out.print("\n使用「二分搜尋法」，請輸入要找的員工編號：");
37     BufferedReader keyin;
38     keyin = new BufferedReader(new InputStreamReader(System.in));
39     int search_account = Integer.parseInt(keyin.readLine());
40
41     int arrNum = Arrays.binarySearch(account, search_account);
42
43     if (arrNum < 0)
44         System.out.println("\n查無此編號");
45     else
46     {
47         System.out.println("\n 編號\t姓名");
48         System.out.println(" " + account[arrNum] + "\t\t" + name[arrNum]);
49     }
50 }
51 }
```

Arrays類別

◆ 執行結果：

排序前的資料順序：

員工編號	姓名
18	王五
14	張三
12	陳二
16	李四
10	鄭一

使用「氣泡排序法」後的資料排序順序：

員工編號	姓名
10	鄭一
12	陳二
14	張三
16	李四
18	王五

使用「二分搜尋法」，請輸入要找的員工編號：8

查無此編號

Arrays類別

排序前的資料順序：

員工編號	姓名
18	王五
14	張三
12	陳二
16	李四
10	鄭一

使用「氣泡排序法」後的資料排序順序：

員工編號	姓名
10	鄭一
12	陳二
14	張三
16	李四
18	王五

使用「二分搜尋法」，請輸入要找的員工編號：16

編號	姓名
16	李四

Arrays類別

◆ 說明：

- 行01：
 - ◆ 載入Java.io.*套件。
- 行02：
 - ◆ 載入java.util.Arrays套件。
- 行06：
 - ◆ main()方法後面，加上 throws IOException。
- 行08：
 - ◆ 宣告一維整數陣列account，並同時指定陣列的初值。
- 行09：
 - ◆ 宣告一維字串陣列name，並同時指定陣列的初值。
- 行11：
 - ◆ 宣告整數變數。

Arrays類別

- 行12：
 - ◆ 宣告字串變數。
- 行14~行15：
 - ◆ 螢幕輸出。
- 行16~行17：
 - ◆ 利用for迴圈，輸出account及name陣列中各個元素的值。
 - ◆ 因為兩個陣列的長度相同，所以用一個for迴圈即可輸出兩個陣列的資料。
 - ◆ 行16：
 - ◆ 利用for迴圈，輸出account及name陣列中各個元素的值。
 - ◆ 陣列的長度為5（即 `account.length=5`）。
 - ◆ 行17：
 - ◆ 螢幕輸出。

Arrays類別

- 行19~行29：
 - ◆ 利用巢狀for迴圈及if，進行氣泡排序法，並顯示每一次排序的結果。
 - ◆ 行19：
設定外層for迴圈（即 循環的次數）。
 - ◆ 行20：
設定內層for迴圈（即 比較的次數）。
 - ◆ 行21~行29：
進行氣泡排序法。
 - 行21：
判斷account陣列左邊的資料是否大於右邊，
 - 行23：
若account陣列左邊的資料大於右邊，則將右邊的資料，存入暫存變數中。
 - 行24：
將account陣列左邊的資料存入右邊的資料位置中。

Arrays類別

行25：

將暫存變數中的資料，存入account陣列左邊的位置中。

行26：

將name陣列右邊的資料，存入暫存變數中。

行27：

將name陣列左邊的資料存入右邊的資料位置中。

行28：

將暫存變數中的資料，存入name陣列左邊的位置中。

注意：

因account陣列與name陣列有相關聯（即員工編號與姓名），故account陣列有交換動作時，name陣列必須跟著要有交換動作。

Arrays類別

- 行31~行32：
 - ◆ 螢幕輸出。
- 行33~行34：
 - ◆ 利用for迴圈，輸出account及name陣列中各個元素的值。
 - ◆ 因為兩個陣列的長度相同，所以用一個for迴圈即可輸出兩個陣列的資料。
 - ◆ 行33：
 - ◆ 利用for迴圈，輸出account即name陣列中各個元素的值。
 - ◆ 陣列的長度為5（即 `account.length=5`）。
 - ◆ 行34：
 - ◆ 螢幕輸出。
- 行36：
 - ◆ 螢幕輸出。

Arrays類別

- 行37：
 - ◆ 宣告BufferedReader類別的物件。
- 行38：
 - ◆ 建立keyin物件。
- 行39：
 - ◆ 利用keyin物件的readLine()方法，讀取由鍵盤輸入的字串，透過Integer.parseInt的方法，轉換成整數後，指派給整數變數search_num。
- 行41：
 - ◆ 進行二分搜尋法，並將找到符合資料的索引值，存入變數arrNum中。
- 行43~行49：
 - ◆ 判斷是否有搜尋到符合的資料。
 - 行43：
若沒有搜尋到符合的資料，則

Arrays類別

行44：

螢幕輸出。

行45：

若有搜尋到符合的資料，則

行47~行48：

螢幕輸出。

Arrays類別

 填滿：

◆ 語法：

Arrays.fill(陣列名稱,要填入的值);

◆ 用途：

- 將指定的資料填入指定的陣列中。

◆ 說明：

- 陣列名稱：
 - ◆ 要填入元素值的陣列名稱。
- 要填入的值：
 - ◆ 要填入元的素值。

Arrays類別

例如：

```
Arrays.fill(A,2);
```

注意：

- 傳回值：

- ◆ 無。

- 適用的資料型別：

- ◆ byte、short、int、long、float、double、boolean、Object及char。

Arrays類別

◆ 程式：

```
import java.util.Arrays;
```

```
public class CH04_16
```

```
{
```

```
    public static void main(String[] args)
```

```
    {
```

```
        int A[] = new int[5];
```

```
        System.out.println("預定初始值：");
```

```
        for (int i = 0; i < A.length; i++)
```

```
            System.out.print(A[i] + "t");
```

```
        System.out.println("\n\n將陣列中的所有元素值，更改為指定的元素  
            值中...");
```

```
        Arrays.fill(A, 5);
```

Arrays類別

```
System.out.println("\n修正後初始值 : ");  
for (int i = 0; i < A.length; i++)  
    System.out.print(A[i] + " ");  
}  
}
```

Arrays類別

```
1 import java.util.Arrays;
2
3 public class CH04_16
4 {
5     public static void main(String[] args)
6     {
7         int A[] = new int[5];
8
9         System.out.println("預定初始值：");
10        for (int i = 0; i < A.length; i++)
11            System.out.print(A[i] + "\t");
12
13        System.out.println("\n\n將陣列中的所有元素值，更改為指定的元素值中...");
14        Arrays.fill(A, 5);
15
16        System.out.println("\n修正後初始值：");
17        for (int i = 0; i < A.length; i++)
18            System.out.print(A[i] + "\t");
19    }
20 }
```

Arrays類別

◆ 執行結果：

預定初始值：

0 0 0 0 0

將陣列中的所有元素值，更改為指定的元素值中…

修正後初始值：

5 5 5 5 5

◆ 說明：

- 行01：
 - ◆ 載入java.util.Arrays套件。
- 行07：
 - ◆ 宣告一維整數陣列A，並同時指定陣列的長度為5。
- 行09：
 - ◆ 螢幕輸出。

Arrays類別

- 行10~行11：
 - ◆ 利用for迴圈，輸出A陣列中各個元素的值。
 - ◆ 陣列的長度為5（即 `account.length=5`）。
- 行13：
 - ◆ 螢幕輸出。
- 行14：
 - ◆ 將『5』填入『A』陣列中。
- 行16：
 - ◆ 螢幕輸出。
- 行17~行18：
 - ◆ 利用for迴圈，輸出A陣列中各個元素的值。
 - ◆ 陣列的長度為5（即 `account.length=5`）。

Arrays類別

 比對：

◆ 語法：

Arrays.equals(陣列名稱1,陣列名稱2);

◆ 用途：

- 比較兩個陣列的內容值是否相同。

◆ 說明：

- 陣列名稱1、陣列名稱2：
 - ◆ 要比較的陣列。

例如：

```
boolean x = Arrays.equals(A, B);
```


Arrays類別

注意：

- 傳回值：

- ◆ Boolean值。

- true表示兩陣列內容值相等。

- false表示兩陣列內容值不相等。

- 適用的資料型別：

- ◆ byte、short、int、long、float、double、Object及char。

Arrays類別

◆ 程式：

```
import java.util.Arrays;

public class CH04_17
{
    public static void main(String[] args)
    {
        int A[] = { 55, 24, 31, 98 };
        int B[] = { 55, 24, 31, 98 };
        int C[] = { 45, 2, 3, 88, 77 };

        boolean x,y,z;

        x = Arrays.equals(A, B);
        y = Arrays.equals(A, C);
        z = Arrays.equals(B, C);
```

Arrays類別

```
System.out.println(" A[]和B[]是否相同：" + x);  
System.out.println(" A[]和C[]是否相同：" + y);  
System.out.println(" B[]和C[]是否相同：" + z);  
}  
}
```

Arrays類別

```
1  import java.util.Arrays;
2
3  public class CH04_17
4  {
5      public static void main(String[] args)
6      {
7          int A[] = { 55, 24, 31, 98 };
8          int B[] = { 55, 24, 31, 98 };
9          int C[] = { 45, 2, 3, 88, 77 };
10
11         boolean x,y,z;
12
13         x = Arrays.equals(A, B);
14         y = Arrays.equals(A, C);
15         z = Arrays.equals(B, C);
16
17         System.out.println(" A[]和B[]是否相同：" + x);
18         System.out.println(" A[]和C[]是否相同：" + y);
19         System.out.println(" B[]和C[]是否相同：" + z);
20     }
21 }
```

Arrays類別

◆ 執行結果：

```
A[]和B[]是否相同：true  
A[]和C[]是否相同：false  
B[]和C[]是否相同：false
```

◆ 說明：

- 行01：
 - ◆ 載入java.util.Arrays套件。
- 行07：
 - ◆ 宣告一維整數陣列A，並同時指定陣列的長度為4。
- 行08：
 - ◆ 宣告一維整數陣列B，並同時指定陣列的長度為4。
- 行09：
 - ◆ 宣告一維整數陣列C，並同時指定陣列的長度為5。

Arrays類別

- 行11：
 - ◆ 宣告Boolean變數。
- 行13：
 - ◆ 比較A、B陣列是否相同，並將比較結果，存入變數x中。
- 行14：
 - ◆ 比較A、C陣列是否相同，並將比較結果，存入變數y中。
- 行15：
 - ◆ 比較C、C陣列是否相同，並將比較結果，存入變數z中。
- 行17~行19：
 - ◆ 螢幕輸出。

綜合練習（1）

題目：樂透彩號產生器

要求：

- 1、以亂數方式，產生1~42間不可重複的六個號碼。
- 2、利用了一維陣列來儲存產生的六個亂數號碼。

提示：

在JAVA中，求 $X \sim Y$ 的範圍之間的亂數方法：

`(int)(Math.random() * (Y-X+1)) + X;`

綜合練習 (2)

題目：計算學生成績分佈圖

要求：

- 1、由使用者輸入任意人數的JAVA成績。
- 2、計算JAVA的總成績及總平均。
- 3、以星號代表該級距的人數，繪製學生成績分佈圖。

提示：

- 總平均計算到小數兩位。
- 分數級距分為0~9、10~19、20~29、30~39、40~49、50~59、60~69、70~79、80~89、90~99、100，共11個級距。

綜合練習 (3)

題目：於陣列中插入新的資料，並依降冪排列

要求：

1、建立一個一維陣列。

➤ 陣列內容為75、24、98、76、55、13。

2、由使用者決定要新增幾筆資料。

3、由使用者輸入的新增資料。

4、將新的陣列，以降冪方式排列。

資料來源

- 蔡文龍、何嘉益、張志成、張力元，JAVA SE 10基礎必修課，台北市，碁峰資訊股份有限公司，2018年7月，出版。
- 吳燦銘、胡昭民，圖解資料結構-使用Java(第三版)，新北市，博碩文化股份有限公司，2018年5月，出版。
- Ivor Horton，Java 8 教學手冊，台北市，碁峰資訊股份有限公司，2016年9月，出版。
- 李春雄，程式邏輯訓練入門與運用---使用JAVA SE 8，台北市，上奇科技股份有限公司，2016年6月，初版。
- 位元文化，Java 8視窗程式設計，台北市，松崗資產管理股份有限公司，2015年12月，出版。
- Benjamin J Evans、David Flanagan，Java 技術手冊 第六版，台北市，碁峰資訊股份有限公司，2015年7月，出版。
- 蔡文龍、張志成，JAVA SE 8 基礎必修課，台北市，碁峰資訊股份有限公司，2014年11月，出版。
- 陳德來，Java SE 8程式設計實例，台北市，上奇科技股份有限公司，2014年11月，初版。
- 林信良，Java SE 8 技術手冊，台北市，碁峰資訊股份有限公司，2014年6月，出版。
- 何嘉益、黃世陽、李篤易、張世杰、黃鳳梅，徐政棠譯，JAVA2 程式設計從零開始--適用JDK7，台北市，上奇資訊股份有限公司，2012年5月，出版。