第 2 章
運算子、運算式與敘述

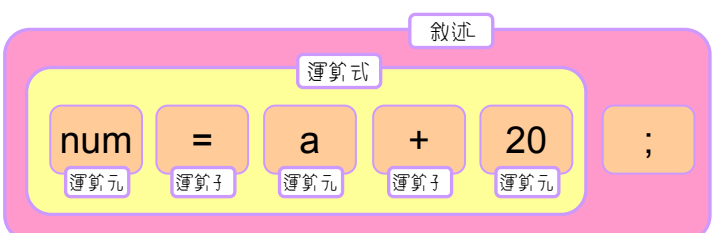
本章學習重點

- 認識運算式與運算子
- 學習各種常用的運算子
- 認識運算子的優先順序
- 學習如何進行運算式之資料型態的轉換

4.1 運算式與運算子

運算式是由運算元（operand）與運算子（operator）所組成。

下圖中 num、a 與 20 都是運算元，而「=」與「+」則為運算子：



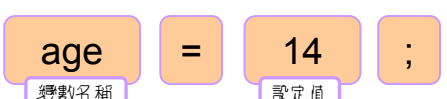
4.1.1 設定運算子

為變數設值，可使用設定運算子（=，assignment operator）。

設定運算子	意義
=	設定

表 4.1.1
設定運算子的說明

等號（=）是「設定」的意思，如下面的範例：



下面的程式碼，是設定運算子的範例：

```
01  // app4_1,設定運算子「=」
02  public class app4_1
03  {
04     public static void main(String args[])
05     {
06        int age=18; // 宣告整數變數 age,並設值為 18
07
08        System.out.println("before compute, age="+age);// 印出 age 的值
09        age=age+1;  // 將 age 加 1 後再設定給 age 存放
10        System.out.println("after compute, age="+age);// 印出計算後 age 的值
11     }
12  }

/* app4_1 OUTPUT-----
before compute, age=18
after compute, age=19
----------------------*/
```

4.1.2 一元運算子

下面的敘述，均是由一元運算子與單一個運算元所組成的。

```
+3;          // 表示正 3
~a;          // 表示取 a 的 1 補數
b=-a;        // 表示負 a 的值設定給變數 b 存放
!a;          // a 的 NOT 運算
```

下表列出了一元運算子的功能：

一元運算子	意義
+	正號
-	負號
!	NOT，否
~	取 1 的補數

表 4.1.2
一元運算子的說明

下面的程式中，可以看到變數經過「~」與「!」運算之後，所呈現的運算結果。

```
01  // app4_2,一元運算子「~」與「!」
02  public class app4_2
03  {
04     public static void main(String args[])
05     {
06        byte a=Byte.MIN_VALUE;     // 宣告變數a,並設為該型態之最小值
07        boolean b=true;            // 宣告 boolean變數b,並設為 true
08
09        System.out.println("a="+a+",~a="+(~a));  // 印出 a與~a的值
10        System.out.println("b="+b+",!b="+(!b));  // 印出 b與!b的值
11     }
12  }
```

```
/* app4_2 OUTPUT----
a=-128,~a=127
b=true,!b=false
----------------------*/
```

4.1.3 算數運算子

下表列出算數運算子的功能：

算數運算子	意義
+	加法
-	減法
*	乘法
/	除法
%	取餘數

表 4.1.3
算數運算子的說明

加法運算子「+」

下面的敘述為加法運算子的範例：

```
6+2;          // 計算 6+2
b=a+15        // 將 a 的值加 15 之後，再設定給變數 b 存放
sum=a+b+c     // 將 a,b 與 c 的值相加之後，再設定給變數 sum 存放
```

減法運算子「-」

下面的敘述為減法運算子的範例：

```
age=age-1;      // 計算 age-1 之後，再將其結果設定給 age 存放
c=a-b;          // 計算 a-b 之後，再設定給 c 存放
54-12;          // 計算 54-12 的值
```

乘法運算子「*」

下面的敘述為乘法運算子的範例：

```
b=c*3;          // 計算 c*3 之後，再將其結果設定給 b 存放
a=a*a;          // 計算 a*a 之後，再設定給 a 存放
17*5;       // 計算 17*5 的值
```

除法運算子「/」

下面的敘述為除法運算子的範例：

```
b=a/6;          // 計算 a/6 之後，再將其結果設定給 b 存放
d=c/d;          // 計算 c/d 之後，再設定給 d 存放
3/8;            // 計算 3/8 的值
```

餘數運算子「%」

下面的敘述是使用餘數運算子的範例：

```
age=age%5;      // 計算 age/5 的餘數，再把計算的結果給 age 存放
c=a%b;          // 計算 a/b 的餘數，然後把計算的結果設定給 c 存放
48%7;           // 運算 48%7 的值
```

4.1.4 關係運算子與 if 敘述

if 敘述的格式如下：

```
if (條件判斷)
    敘述;
```

格式 4.1.1
if 敘述的格式

下面是 if 敘述的範例：

```
if (i>0)
   System.out.println("i 的值大於 0");
```

下表列出了關係運算子的成員：

關係運算子	意義
>	大於
<	小於
>=	大於等於
<=	小於等於
==	等於
!=	不等於

表 4.1.4
關係運算子的說明

下面的程式是利用關係運算子來判斷該印出哪些敘述：

```
01  // app4_3,關係運算子
02  public class app4_3
03  {
04     public static void main(String args[])
05     {
06        if (9>4)          // 判斷 9>4 是否成立
07           System.out.println("9>4 成立");       // 印出傳回值
08
09        if (true)         // 判斷 true 是否成立
10           System.out.println("此行會被執行--true");      // 印出字串
11
12        if (false)         // 判斷 false 是否成立
13           System.out.println("此行會被執行--false");     // 印出字串
14
15        if (5==7) // 判斷 5 是否等於 7
16           System.out.println("5==7 成立"); // 印出字串
17     }
18  }
```

```
/* app4_3 OUTPUT---

9>4 成立
此行會被執行--true
--------------------*/
```

4.1.5 遞增與遞減運算子

下表列出遞增與遞減運算子的功能：

遞增與遞減運算子	意義
++	遞增，變數值加 1
--	遞減，變數值減 1

表 4.1.5
遞增與遞減運算子

程式執行中想要變數 a 加上 1，敘述如下：

```
a=a+1;          // a 加 1 後再設定給 a 存放
```

利用遞增運算子「++」可以寫出更簡潔的敘述：

```
a++;        // a 加 1 後再設定給 a 存放，a++為簡潔寫法
```

下面是遞增與遞減運算子的使用範例：

```
01  // app4_4,遞增運算子「++」
02  public class app4_4
03  {
04     public static void main(String args[])
05     {
06        int a=3,b=3;
07
08        System.out.print("a="+a);     // 印出 a
09        System.out.println(",a++="+(a++)+",a="+a);  // 印出 a++
10        System.out.print("b="+b);     // 印出 b
11        System.out.println(",++b="+(++b)+",b="+b);  // 印出 ++b
12     }
13  }
```

```
/* app4_4 OUTPUT---
a=3,a++=3,a=4
b=3,++b=4,b=4
--------------------*/
```

4.1.6 邏輯運算子

下表列出了邏輯運算子的功能：

邏輯運算子	意義
&&	AND，且
||	OR，或

表 4.1.6
邏輯運算子的說明

下面列出了 AND 與 OR 的真值表：

AND	T	F
T	T	F
F	F	F

OR	T	F
T	T	T
F	T	F

表 4.1.7
AND 及 OR 真值表

下面是邏輯運算子的範例程式：

```
01  // app4_5,邏輯運算子
02  public class app4_5
03  {
04     public static void main(String args[])
05     {
06        int a=53;
07
08        if ((a<0) || (a>100))
09           System.out.println("Input error!!");    // 成績輸入錯誤
10        if ((a<60) && (a>49))
11           System.out.println("Make up exam!!");   // 需要補考
12     }
13  }
```

```
/* app4_5 OUTPUT----

Make up exam!!
----------------------*/
```

4.1.7 括號運算子

括號運算子 ()可以提高括號內運算式的優先處理順序：

括號運算子	意義
()	提高括號內運算式的優先順序

表 4.1.8
括號運算子的說明

下面是括號運算子的使用範例：

```
(3+5*4)*(6-7)       // 加上括號的運算式
```

計算結果為 -23。

4.2 運算子的優先順序

下表列出了各種運算子優先順序的排列，數字愈小表示優先順序愈高。

表 4.2.1　運算子的優先順序

優先順序	運算子	類別	結合性
1	()	括號運算子	由左到右
1	[]	方括號運算子	由左到右
2	!、+（正號）、-（負號）	一元運算子	由右到左
2	~	位元邏輯運算子	由右到左
2	++、--	遞增與遞減運算子	由右到左
3	*、/、%	算數運算子	由左到右
4	+、-	算數運算子	由左到右
5	<<、>>	位元左移、右移運算子	由左到右
6	>、>=、<、<=	關係運算子	由左到右
7	==、!=	關係運算子	由左到右
8	&（位元運算的 AND）	位元邏輯運算子	由左到右
9	^（位元運算的 XOR）	位元邏輯運算子	由左到右
10	|（位元運算的 OR）	位元邏輯運算子	由左到右

優先順序	運算子	類別	結合性
11	&&	邏輯運算子	由左至右
12	||	邏輯運算子	由左至右
13	?:	條件運算子	由右至左
14	=	設定運算子	由右至左

4.3 運算式

下面的例子，均是關於運算式的一種：

```
-49
sum+2
a+b-c/(d*3-9)
```

下表列出了這些相結合的運算子：

運算子	範例用法	說明	意義
+=	a+=b	a+b 的值存放到 a 中	a=a+b
-=	a-=b	a-b 的值存放到 a 中	a=a-b
*=	a*=b	a*b 的值存放到 a 中	a=a*b
/=	a/=b	a/b 的值存放到 a 中	a=a/b
%=	a%=b	a%b 的值存放到 a 中	a=a%b

下面的運算式，皆是簡潔的寫法：

```
a++          // 相當於 a=a+1
a-=5         // 相當於 a=a-5
b%=c         // 相當於 b=b%c
a/=b--       // 相當於計算 a=a/b 之後，再計算 b--
```

我們實際練習一下這種簡潔的程式寫法：

```
01  // app4_6,簡潔運算式
02  public class app4_6
03  {
04     public static void main(String args[])
05     {
06        int a=5,b=8;
07
08        System.out.println("before compute, a="+a+" ,b="+b);
09        a+=b;       // 計算 a+=b 的值，此式相同於 a=a+b
10        System.out.println("after compute, a="+a+" ,b="+b);
11     }
12  }
```

```
/* app4_6 OUTPUT-------
before compute, a=5, b=8
after compute, a=13, b=8
------------------------*/
```

4.4 運算式的型態轉換

運算式中運算元的型態不同時，會依據下列的規則來處理型態的轉換：

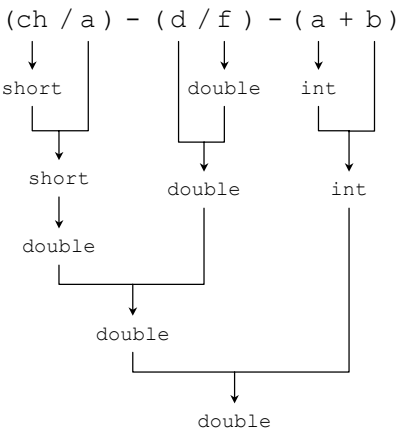
1. 佔用位元組較少的轉換成位元組較多的型態。
2. 字元型態會轉換成 short 型態（字元會取其 unicode 碼）。
3. short 型態遇上 int 型態，會轉換成 int 型態。
4. int 型態會轉換成 float 型態。
5. 運算式中的某個運算元的型態為 double，則另一個運算元也會轉換成 double 型態。
6. 布林型態不能轉換為其它的型態。

以下面的程式為例，分別宣告數個不同型態的變數，並加以運算：

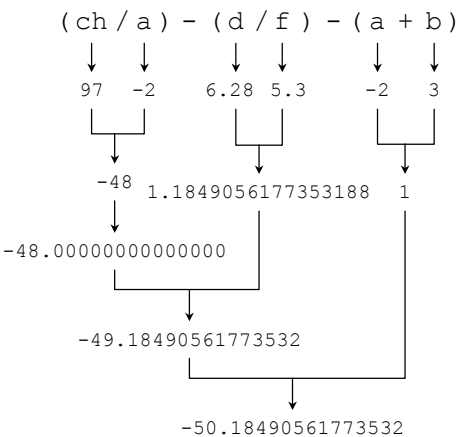
```
01  // app4_7,運算式的型態轉換
02  public class app4_7
03  {
04     public static void main(String args[])
05     {
06        char ch='a';
07        short a=-2;
08        int b=3;
09        float f=5.3f;
10        double d=6.28;
11
12        System.out.print("(ch/a)-(d/f)-(a+b)=");  // 印出結果
13        System.out.println((ch/a)-(d/f)-(a+b));
14     }
15  }

/* app4_7 OUTPUT--------------------
(ch/a)-(d/f)-(a+b)=-50.18490561773532
-------------------------------------*/
```

運算式 (ch/a)-(d/f)-(b+a) 的輸出型態是什麼？請參考下圖的解說：



再來看看運算式的運算過程：



-The End-