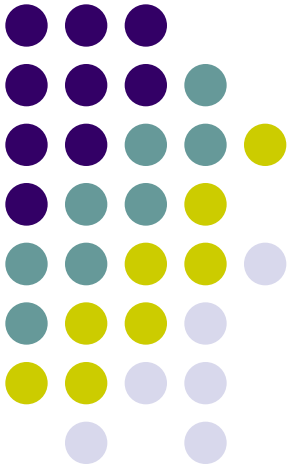
第四章
運算子、運算式與敘述

認識運算式與運算子

學習各種常用的運算子

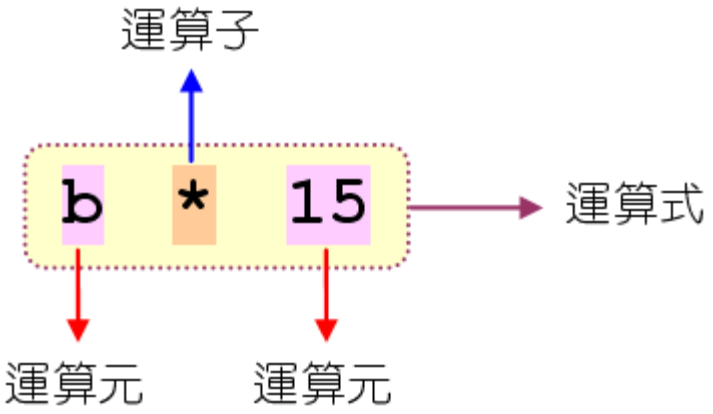
認識運算子的優先順序

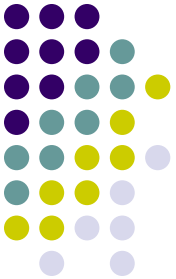
學習如何進行運算式的資料型態轉換



運算式

- 運算式由運算元（operand）與運算子（operator）組成
- 以運算式b*15為例，b與15都是運算元，而 * 為運算子：





設定運算子 (1/2)

- 要設定變數的值，可以使用設定運算子：

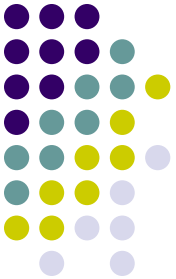
設定運算子	意義
=	設定

- 設定運算子的使用範例：

```
num=18;              // 將整數 18 設定給 num 存放

num=num+1;           // 將 num+1 的值運算之後再設定給變數 num 存放

sum=num1+num2;       // 將 num1 加上 num2 之後再設定給變數 sum 存放
```



設定運算子 (2/2)

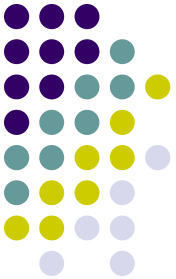
- 下面的程式碼是使用設定運算子的範例：

```
01  // prog4_1, 設定運算子「=」
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07     int num=18;     // 宣告整數變數 num,並設值為 18
08     cout << "計算前, num=" << num << endl;     // 印出 num 的值
09     num=num+1;      // 將 num 加 1 後再設定給 num 存放
10     cout << "計算後, num=" << num << endl;     // 印出計算後 num 的值
11     system("pause");
12     return 0;
13  }
```

```
/* prog4_1 OUTPUT---

計算前, num=18
計算後, num=19

--------------------*/
```



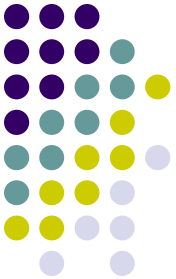
一元運算子 (1/2)

- 下面的敘述，均由一元運算子與單一個運算元所組成

```
+63;          // 表示正 63
~b;           // 表示取 b 的 1 補數
a=-b;         // 表示負 b 的值設定給變數 a 存放
!a;           // a 的 NOT 運算，若 a 為 0，則!a 為 1，若 a 不為 0，則!a 為 0
```

- 下表列出一元運算子的成員

一元運算子	意義
+	正號
-	負號
!	NOT，否
~	取 1 的補數



一元運算子 (2/2)

- 下面的程式是使用一元運算子的範例

```
01  // prog4_2, 一元運算子「~」與「!」
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07     short n=12;                    // 宣告 short 變數 n,並設為 12
08     bool b=false;                  // 宣告 bool 變數 b,並設為 false
09     cout << "n=" << n << ",~n=" << ~n << endl;    // 印出 n 與~n 的值
10     cout << "b=" << b << ",!b=" << !b << endl;    // 印出 b 與!b 的值
11     system("pause");
12     return 0;                      /* prog4_2 OUTPUT----
13  }
                                      n=12,~n=-13
                                      b=0,!b=1
                                      ----------------------*/
```

算數運算子 (1/4)

- 下表列出算數運算子的成員：

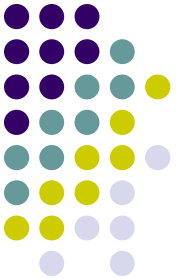
算數運算子	意義
+	加法
-	減法
*	乘法
/	除法
%	取餘數

- 加法運算子「+」
  - 將前後兩個運算元相加

```
cout << "3+8=" << 3+8;     // 直接印出運算式的值
```

- 減法運算子「-」
  - 將出現在它前面的運算元減去後面的運算元

```
age=age-10; // 將 age-10 運算後的值設定給 age 存放
b=c-a;      // 將 c-a 運算後的值設定給 b 存放
120-36;     // 計算 120-36 的值
```



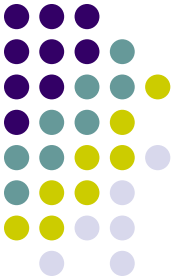
算數運算子 (2/4)

- 乘法運算子「*」
  - 將前後的兩個運算元相乘

```
b=c*8;        // 將 c*8 運算後的值設定給 b 存放
a=b*b;        // 將 b*b 運算後的值設定給 a 存放
21*5;         // 計算 21*5 的值
```

- 除法運算子「/」
  - 將前面的運算元除以後面的運算元

```
c=a/5;        // 將 a/5 運算後的值設定給 c 存放
d=b/a;        // 將 b/a 運算後的值設定給 d 存放
43/19;        // 計算 43/19 的值
```



算數運算子 (3/4)

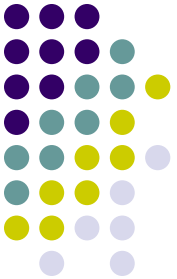
- 下面的程式裡設定兩個整數a、b，並將a/b的結果印出

```
01  // prog4_3, 除法運算子「/」
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07     int a=16;                  // 宣告 int 變數 a,並設值為 16
08     int b=7;                   // 宣告 int 變數 b,並設值為 7
09     cout << "a=" << a << ",b=" << b << endl;      // 印出 a 與 b 的值
10     cout << "a/b=" << a/b << endl;                // 印出 a/b 的值
11     cout << "a/b=" << (float)a/b << endl;   // 印出(float)a/b 的值
12     system("pause");
13     return 0;
14  }
```

```
/* prog4_3 OUTPUT----

a=16,b=7
a/b=2
a/b=2.28571

--------------------*/
```



算數運算子 (4/4)

- 餘數運算子 「%」
  - 將前面的運算元除以後面的運算元，再取其所得到的餘數

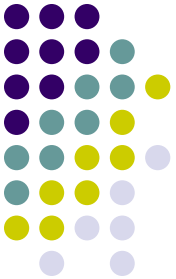
```
num=num%5;   // 將 num%5 運算後的值設定給 num 存放
c=a%b;       // 將 a%b 運算後的值設定給 c 存放
125%6;       // 計算 125%6 的值
```

```
01  // prog4_4, 餘數運算子「%」
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07     int a=123;          // 宣告 int 變數 a,並設值為 123
08     int b=6;            // 宣告 int 變數 b,並設值為 6
09     cout << a << "%" << b << "=" << a%b << endl;    // 印出 a%b 的值
10     cout << b << "%" << a << "=" << b%a << endl;    // 印出 b%a 的值
11     system("pause");
12     return 0;
13  }
```

```
/* prog4_4 OUTPUT----

123%6=3
6%123=6

---------------------*/
```



關係運算子與if敘述 (1/2)

- if敘述的格式如下：

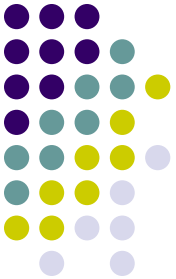
```
if (條件判斷)
   敘述;
```

- 下面的程式片段為if敘述的例子：

```
if (i>0)
   cout << "Rome was not built in a day!";
```

關係運算子成員

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



關係運算子與if敘述 (2/2)

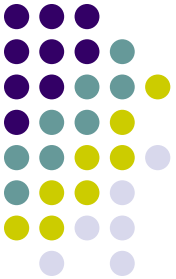
- 下面的程式是使用if敘述的完整範例

```
01  // prog4_5, 關係運算子
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07     int i;
08     cout << "Input an integer:";
09     cin >> i;
10     if (i>5)                          // 判斷 i>5 是否成立
11        cout << i << ">5 成立" << endl;  // 印出字串
12     if (i%2==0)                       // 判斷 i%2 是否等於 0
13        cout << i << "為偶數" << endl;   // 印出字串
14     if (true)                         // 判斷 true 是否成立
15        cout << "此行永遠會被執行" << endl;  // 印出字串
16     system("pause");
17     return 0;
18  }
```

```
/* prog4_5 OUTPUT---

Input an integer:7
7>5 成立
此行永遠會被執行

--------------------*/
```



遞增與遞減運算子 (1/4)

- 遞增與遞減運算子的成員

遞增與遞減運算子	意義
++	遞增，變數值加 1
--	遞減，變數值減 1

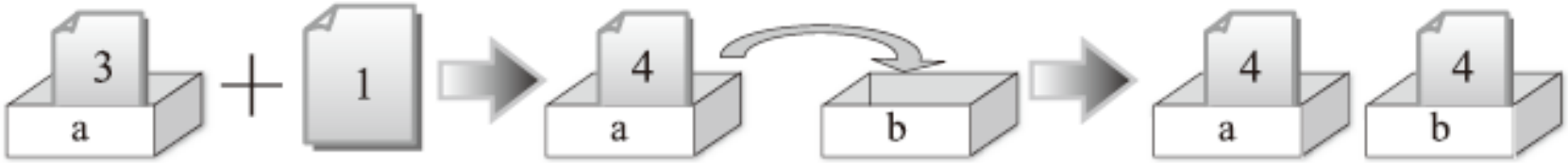
- 想讓變數i的值加上1，有下列兩種寫法

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

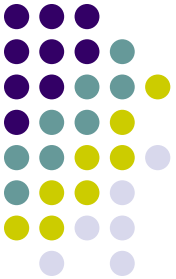
- i++與++i的區別
  - i++ 會先執行整個敘述後再將i的值加1
  - ++i 則先把i的值加1後，再執行整個敘述

遞增與遞減運算子 (2/4)

```
int a = 3;
int b = ++a;
cout << "a=" << a << "b=" << b << endl; //a=4,b=4
```




```
int a = 3;
int b = a++;
cout << "a=" << a << "b=" << b << endl; // a=4,b=3
```



遞增與遞減運算子 (3/4)

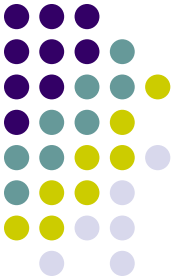
- 下面的程式可以觀察遞增運算子的使用

```
01  // prog4_6, 遞增運算子「++」在運算元之後
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07     int a=10;
08     cout << "a=" << a << endl;                    // 印出 a
09     cout << "a++*2=" << (a++*2) << endl;          // 印出 a++*2
10     cout << "a=" << a << endl;                    // 印出 a
11     system("pause");
12     return 0;
13  }
```

```
/* prog4_6 OUTPUT---

a=10
a++*2=20
a=11

--------------------*/
```



遞增與遞減運算子 (4/4)

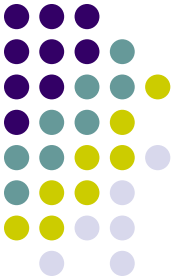
- 請比較一下遞增運算子放在運算元前後的差別

```
01  // prog4_7, 遞增運算子「++」在運算元之前
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07     int a=10;
08     cout << "a=" << a << endl;                  // 印出 a
09     cout << "++a*2=" << (++a*2) << endl;        // 印出++a*2
10     cout << "a=" << a << endl;                  // 印出 a
11     system("pause");
12     return 0;
13  }
```

```
/* prog4_7 OUTPUT---

a=10
++a*2=22
a=11

--------------------*/
```

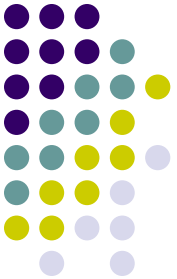


算數與設定運算子的結合 (1/3)

- 下面幾個運算式，皆是簡潔的寫法

```
a++;          // 相當於 a=a+1
b-=3;         // 相當於 b=b-3
b%=c;         // 相當於 b=b%c
```

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



算數與設定運算子的結合 (2/3)

- 下面的範例實際練習一下簡潔運算式的寫法

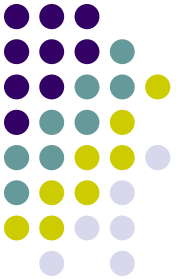
```
01  // prog4_8, 簡潔運算式
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07     int a=100,b=15;
08     cout << "a=" << a << ", b=" << b << endl;
09     a-=b;                              // 計算 a=a-b 的值
10     cout << "after a-=b, a=" << a << ", b=" << b << endl;
11     system("pause");
12     return 0;
13  }
```

```
/* prog4_8 OUTPUT---------
a=100, b=15
after a-=b, a=85, b=15
-------------------------*/
```

算數與設定運算子的結合 (3/3)

- 下表列出簡潔寫法的運算子及其範例說明

運算子	範例	執行前		說明	執行後	
		a	b		a	b
+=	a+=b	12	4	a+b 的值存放到 a 中 (同 a=a+b)	16	4
-=	a-=b	12	4	a-b 的值存放到 a 中 (同 a=a-b)	8	4
*=	a*=b	12	4	a*b 的值存放到 a 中 (同 a=a*b)	48	4
/=	a/=b	12	4	a/b 的值存放到 a 中 (同 a=a/b)	3	4
%=	a%=b	12	4	a%b 的值存放到 a 中 (同 a=a%b)	0	4
b++	a*=b++	12	4	a*b 的值存放到 a 後，b 加 1 (同 a=a*b; b++)	48	5
++b	a*=++b	12	4	b 加 1 後，再將 a*b 的值存放到 a (同 b++; a=a*b)	60	5
b--	a*=b--	12	4	a*b 的值存放到 a 後，b 減 1 (同 a=a*b; b--)	48	3
--b	a*=--b	12	4	b 減 1 後，再將 a*b 的值存放到 a (同 b--; a=a*b)	36	3



邏輯運算子 (1/5)

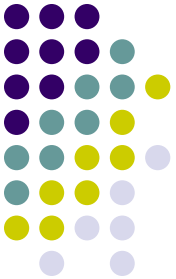
- 邏輯運算子的成員

邏輯運算子	意義
&&	AND，且
||	OR，或

- 使用&&時，兩個運算元皆為真，運算結果才會為真
- 使用 || 時，兩個運算元只要一個為真，運算結果就為真
- 下面為邏輯運算子的範例

```
(1) a>0 && b>0     // 兩個運算元皆為真，運算結果才為真

(2) a>0 || b>0     // 兩個運算元只要一個為真，運算結果就為真
```



邏輯運算子 (2/5)

- 下面的範例說明邏輯運算子如何應用在if敘述中

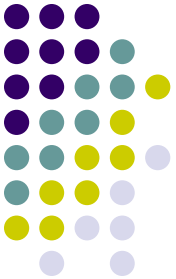
```
01  // prog4_9, 邏輯運算子
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07     int score;
08     cout << "Input your score:";                  // 由鍵盤輸入成績
09     cin >> score;
10     if ((score<0) || (score>100))
11        cout << "Input error!!" << endl;           // 成績輸入錯誤
12     if ((score<60) && (score>49))
13        cout <<"Make up exam!!" << endl;           // 需要補考
14     system("pause");
15     return 0;
16  }
```

```
/* prog4_9 OUTPUT----
Input your score:58
Make up exam!!
--------------------*/
```

「&&」及「||」的比較結果分別與「&」及「|」完全相同，但因前者在某些情況並不對第二個運算式做比較，所以執行效率較高。

邏輯運算子 (3/5)

運算子	意義	範例	範例結果
! (Not)	傳回與原來比較結果相反的值，即比較結果是true，就傳回 false；比較結果是false，就傳回 true。	!(12>7) !(7>12)	0 1
& (And)	只有兩個運算元的比較結果都是 true 時，才傳回 true，其餘情況皆傳回 false。	(12>7) & (5>2) (12>7) & (5<2) (12<7) & (5>2) (12<7) & (5<2)	1 0 0 0
| (Or)	只有兩個運算元的比較結果都是 false 時，才傳回 false，其餘情況皆傳回 true。	(12>7) | (5>2) (12>7) | (5<2) (12<7) | (5>2) (12<7) | (5<2)	1 1 1 0
^ (Xor)	兩個運算元的比較結果都是 frue 或 false 時，就傳回 false；兩個運算元的比較結果一個是 true 而另一個是 false時，就傳回 true。	(12>7) ^ (5>2) (12>7) ^ (5<2) (12<7) ^ (5>2) (12<7) ^ (5<2)	0 1 1 0
&& (AndAlso)	第一個運算元為 false 時，直接傳回 false；若第一個運算元為 true 時，則繼續比較第二個運算元，其為 true 就傳回 true，否則就傳回 false。	結果與 & 相同。	
|| (OrElse)	第一個運算元為 true 時，直接傳回 true；若第一個運算元為false 時，則繼續比較第二個運算元，其為 true 就傳回 true，否則就傳回 false。	結果與 | 相同。	



邏輯運算子 (4/5)

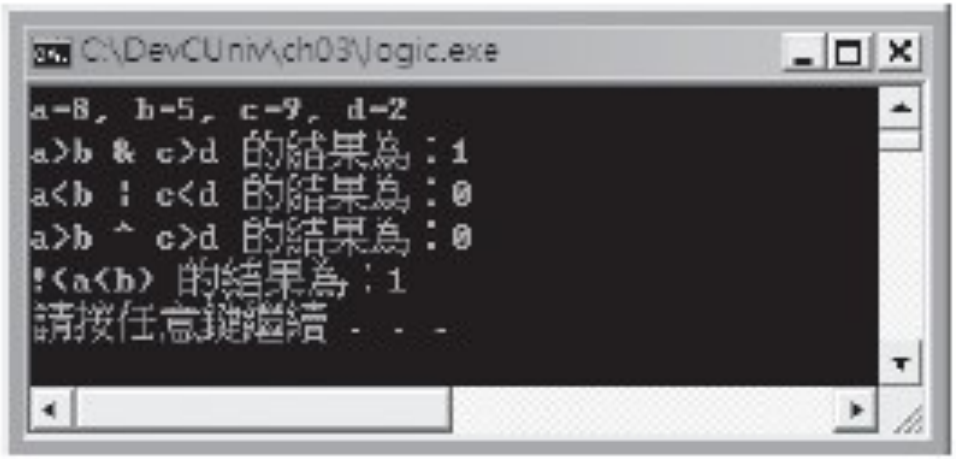
- 範例時間：邏輯運算子
  - 顯示各種邏輯運算的結果。

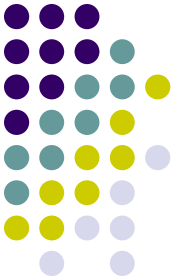
程式碼

```
1   #include <iostream>
2   using namespace std;
3   int main()
4   {
5      int a=8, b=5, c=9, d=2;
6      cout << "a=8, b=5, c=9, d=2\n";
7      cout << "a>b & c>d 的結果為：" << (a>b & c>d) << "\n";
8      cout << "a<b | c<d 的結果為：" << (a<b | c<d) << "\n";
9      cout << "a>b ^ c>d 的結果為：" << (a>b ^ c>d) << "\n";
10     cout << "!(a<b) 的結果為：" << !(a<b) << "\n";
11     system("pause");
12     return 0;
13  }
```

邏輯運算子 (5/5)

- 範例時間：邏輯運算子
  - 執行結果

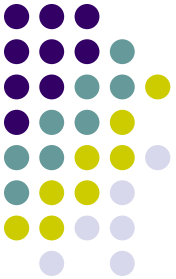




位元邏輯運算子(1/3)

- 位元邏輯運算子如下：

運算子	意義	範例	範例結果
~ (Not)	此為單元運算子，將運算元 1 轉 0，0 轉為 1。	~1; 1= $0000,0001_2$	-2 //$1111,1110_2$ 為 -2 (2 的補數)
& (And)	只有兩個運算元的比較結果都是 1 時，才傳回 1，其餘情況皆傳回 0。	14 & 5; 14=$0000,1110_2$ 5 =$0000,0101_2$	4 // 00000100_2
| (Or)	只有兩個運算元的比較結果都是 0 時，才傳回 0，其餘情況皆傳回 1。	14 | 5; 14=$0000,1110_2$ 5 =$0000,0101_2$	15 // 00001111_2
^ (Xor)	兩個運算元的比較結果都是 1 或 0 時，就傳回 0；兩個運算元的比較結果一個是 1 而另一個是0時，就傳回 1。	14 ^ 5; 14=$0000,1110_2$ 5 =$0000,0101_2$	11 // 00001011_2
>>	運算元右移，左補 0。右移相當於除 2。	255 >>2; 255=$1111,1111_2$	63 // $0011,1111_2$
<<	運算元左移，右補 0。左移相當於乘 2。	63 << 2; 63=$0011,1111_2$	252 // $1111,1100_2$



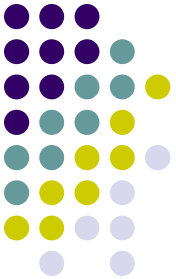
位元邏輯運算子(2/3)

- 範例時間：位元運算子
  - 讓使用者輸入兩個整數，顯示兩數各種位元運算的結果。 程式碼

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5     int a, b;
6     cout << "請輸入 a 的值(整數)：";
7     cin >> a;
8     cout << "請輸入 b 的值(整數)：";
9     cin >> b;
10    cout << "a&b 的結果為：" << (a&b) << "\n";
11    cout << "a|b 的結果為：" << (a|b) << "\n";
12    cout << "a^b 的結果為：" << (a^b) << "\n";
13    cout << "a>>2 的結果為：" << (a>>2) << "\n";
14    cout << "a<<2 的結果為：" << (a<<2) << "\n";
15    system("pause");
16    return 0;
17 }
```

位元邏輯運算子(3/3)

- 範例時間：位元運算子
  - 執行結果



括號運算子

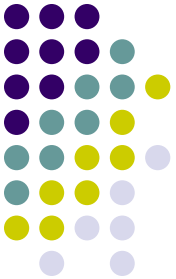
- 括號運算子 () 可用來處理運算式的優先順序

括號運算子	意義
()	提高括號中運算式的優先順序

- 括號運算子的使用範例

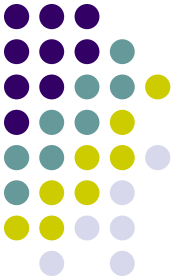
```
8-4*3+2*6                    // 未加括號的運算式

(8-4*(3+2))*6                // 加上括號的運算式
```



運算子優先順序的排列

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

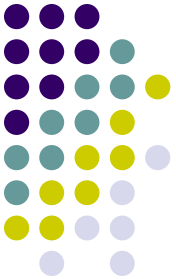


型態的轉換

- 下面的例子，均是屬於運算式的一種

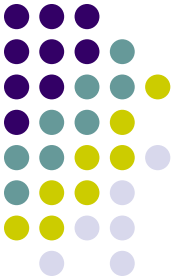
```
-2;                 // 運算式由一元運算子「-」與常數 2 組成
age+12;             // 運算式由變數 age、算數運算子與常數 12 組成
a*b-c*(d/8-3);      // 由變數、常數與運算子所組成的運算式
```

- 運算式與資料型態的轉換可分為
  - 「隱性資料型態轉換」（implicit type conversion）
  - 「顯性資料型態轉換」（explicit type conversion）



隱性資料型態轉換 (1/2)

- C++會依據下列的規則自動做資料型態的轉換
  - 轉換前的資料型態與轉換後的型態相容
  - 轉換後的資料型態之表示範圍比轉換前的型態大
- 隱性資料型態的轉換稱為自動型態轉換（automatic type conversion）
- 型態的轉換只限該行敘述
- 隱性資料型態的轉換以確保證資料精度不損失為原則



隱性資料型態轉換 (2/2)

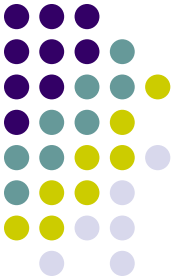
- 下面是自動形態轉換的範例

```
01  // prog4_10, 型態自動轉換
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07     int a=45;
08     float b=2.3f;
09     cout << "a=" << a << ", b=" << b << endl;     // 印出 a、b 的值
10     cout << "a/b=" << a/b << endl;                // 印出 a/b 的值
11     system("pause");
12     return 0;
13  }
```

```
/* prog4_10 OUTPUT---

a=45, b=2.3
a/b=19.5652

----------------------*/
```

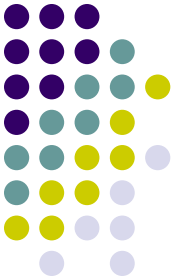


顯性資料型態轉換 (1/2)

- 如果希望計算的結果是不同的型態，可進行顯性轉換

```
（欲轉換的資料型態） 變數名稱;
```

- 顯性型態轉換也稱為強制型態轉換



顯性資料型態轉換 (2/2)

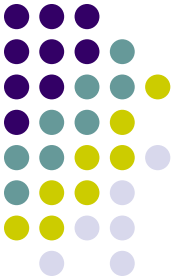
- 下面的程式說明在C++裡，整數與浮點數是如何轉換

```
01  // prog4_11, 顯性資料型態轉換
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07     int a=36,b=7;
08     cout << "a=" << a << ", b=" << b << ", ";     // 印出 a、b 的值
09     cout << "a/b=" << (a/b) << endl;              // 印出 a/b 的值
10     cout << "a=" << a << ", b=" << b << ", ";     // 印出 a、b 的值
11     cout << "a/b=" << (float)a/b << endl;    // 印出(float)a/b 的值
12     system("pause");
13     return 0;
14  }
```

```
/* prog4_11 OUTPUT-------

a=36, b=7, a/b=5
a=36, b=7, a/b=5.14286

-----------------------*/
```



型態轉換的寫法

- 下面是三種型態轉換的寫法：

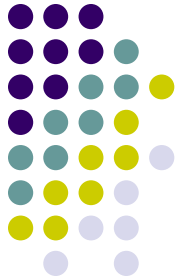
```
(1)    (float)a/b            // 將整數 a 強制轉換成浮點數，再與整數 b 相除
(2)    a/(float)b            // 將整數 b 強制轉換成浮點數，再以整數 a 除之
(3)    (float)a/(float)b     // 將整數 a 與 b 同時強制轉換成浮點數
```

- 縮小轉換（narrowing conversion）
  - 將變數設值成一個大於該型態可以表示範圍時

    例如

```
double a=5.0;
int num=a;          // 縮小轉換
```

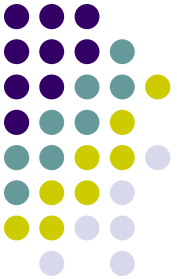
- 在轉換的過程中可能會因此漏失資料的精確度



運算式的型態轉換 (1/3)

- 型態不合的情況時，會依據下列的規則來處理型態的轉換：
  - 佔用的位元組較少的轉換成位元組較多的型態。

    如short型態（2 bytes）遇上int型態（4 bytes），會轉換成int型態。
  - 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
  - 布林型態不能轉換至其它的型態



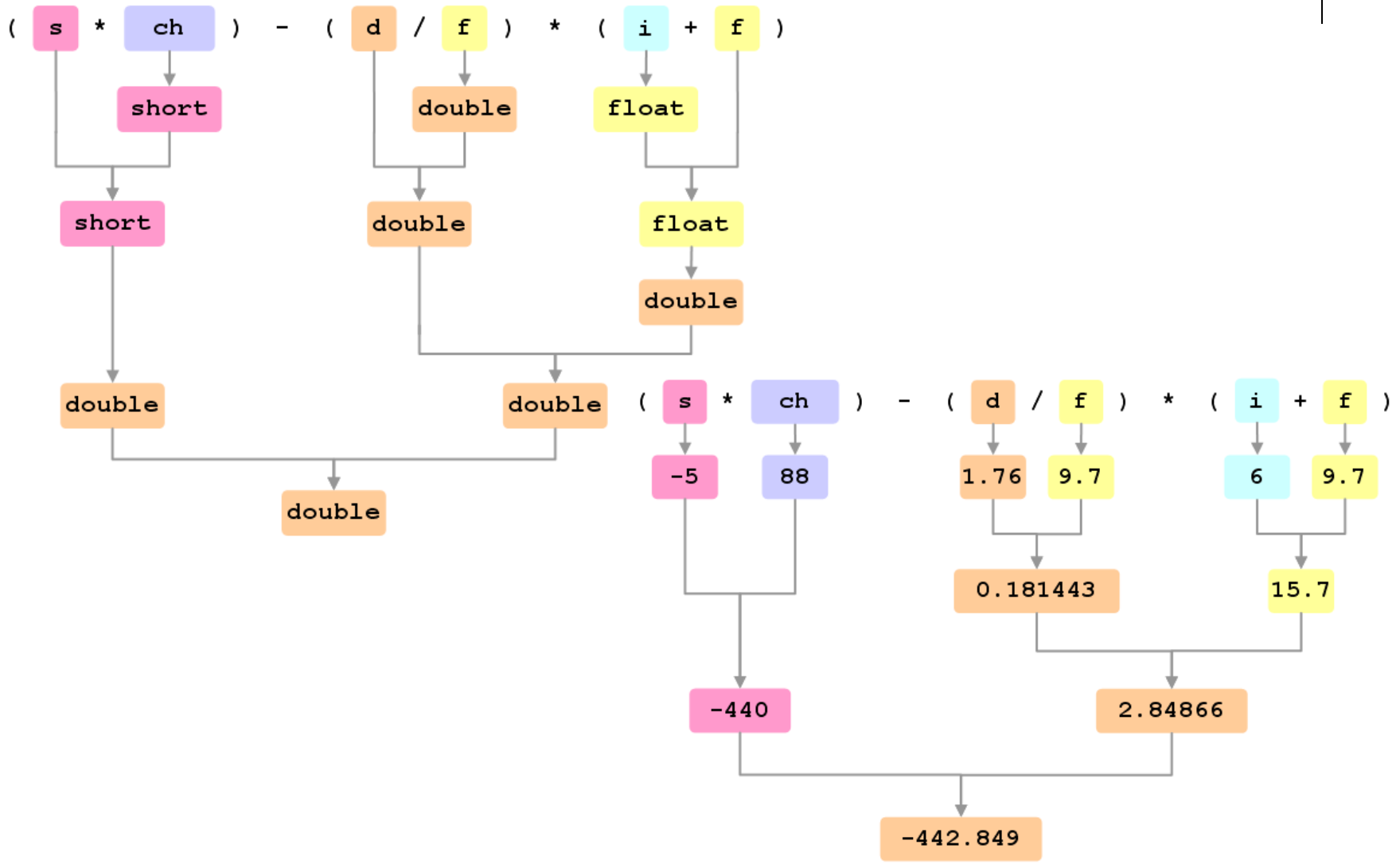
運算式的型態轉換 (2/3)

- 下面的範例是不同型態的變數，它們之間運算的結果

```
01  // prog4_12, 運算式的型態轉換
02  #include <iostream>
03  #include <cstdlib>
04  using namespace std;
05  int main(void)
06  {
07     char ch='X';
08     short s=-5;
09     int i=6;
10     float f=9.7f;
11     double d=1.76;
12     cout << "(s*ch)-(d/f)*(i+f)=";              // 印出結果
13     cout << (s*ch)-(d/f)*(i+f) << endl;
14     system("pause");
15     return 0;
16  }
```

```
/* prog4_12 OUTPUT----------
(s*ch)-(d/f)*(i+f)=-442.849
----------------------------*/
```

運算式的型態轉換 (3/3)


(s * ch) - (d / f) * (i + f)
short
double
float
short
double
float
double
double
double
double
(s * ch) - (d / f) * (i + f)
-5
88
1.76
9.7
6
9.7
0.181443
15.7
-440
2.84866
-442.849


-The End-