03
CHAPTER

運算式和運算子

本章學習重點

- 運算式與運算子簡介
- 運算子的功能
- 運算子的優先權與結合性
- 型態轉換

本章學習範例

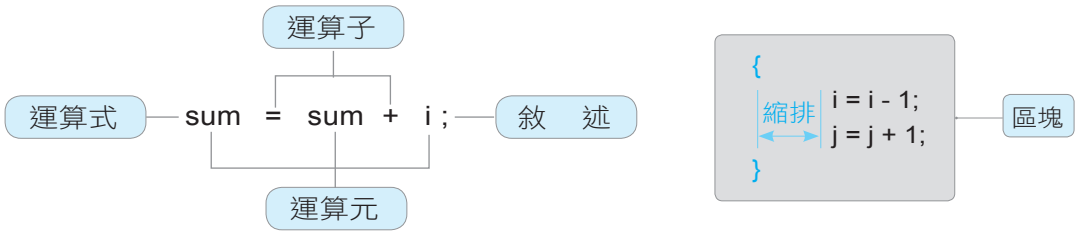
- 範例 3.2.1 兩數交換
- 範例 3.2.2 時差換算 (d050)
- 範例 3.2.2-2 買原子筆 (d827)
- 範例 3.2.2-3 分組問題 (d073)
- 範例 3.2.2-4 秒數格式轉換
- 範例 3.2.3 複合運算子
- 範例 3.2.3-2 整數的位數
- 範例 3.2.6 位元運算
- 範例 3.4.2 字元與 ASCII 碼轉換
- 範例 3.4.2-2 BMI 計算

3.1 運算式與運算子簡介

3.1.1 運算式與敘述

運算式是由運算元（operand）和運算子（operator）組成的，如左下圖，= 和 + 是運算子，sum 和 i 是運算元。敘述則是在運算式後加上分號 ;。

二個或二個以上的敘述可使用大括弧 { } 組成一個區塊（block），例如右下圖的區塊內有二個敘述，區塊內的敘述可使用 Tab 鍵向內縮排，讓程式區塊的層次更分明，更簡潔易懂，除可增加程式的可讀性，也方便除錯。



3.1.2 運算子分類

C++ 的運算子可依運算元個數或運算子功能來分類：

1. 依運算元個數分類

(1) 一元運算子（unary operator）：只需一個運算元，例如

```
x = -1 ;
```

負號 - 是一元運算子，因為 - 後面只需一個運算元。

(2) 二元運算子（binary operator）：需要兩個運算元，例如

```
x = y + z;
```

加號 + 是二元運算子，因為 + 需要兩個運算元。

(3) 三元運算子（ternary operator）：需要三個運算元，例如

```
max = x > y ? x : y ;
```

? : 是唯一的三元運算子，上面敘述的功能是「若 x > y 則 max = x，否則 max = y」，下一章會進一步說明。

2. 依運算子功能分類

分類	運算子
指定運算子	=
算術運算子	+ - * / %
複合指定運算子	+= -= *= /= %= >>= <<= &= |= ^=
遞增 / 遞減運算子	++ / --
關係運算子	== != > < >= <=
邏輯運算子	&& || !
位元運算子	>> << & |~ ^
條件運算子	?
逗號運算子	,
資料大小運算子	sizeof
位址運算子	&

運算子的結合性（associativity）是指一個運算子選擇其運算元的方向，有下列兩種

(1) 左結合性（left to right）：由左運算至右

例如 2 + 3 + 4，會先計算 2 + 3，得 5，再計算 5 + 4，得 9。

(2) 右結合性（right to left）：由右運算至左

如 i = 5，會將 5 指定給 i。

大部份的運算子都是左結合性，除了一元運算子和指定運算子（含複合指定運算子），如 ++、--、~、!、sizeof 和 =（含 +=、-=、*=、/=、%= 等）。

3.2 運算子的功能

3.2.1 指定運算子（=）

指定運算子 = 是將運算式的結果指定給變數，屬於右結合性。格式如下：

變數 = 運算式

將運算式的值指定給變數

如下例，可將常數、變數、或運算式指定給另一個變數。

```
x = 1
```

將常數 1 指定給整數變數 x

```
x = y
```

將變數 y 的值指定給變數 x，x 原有的值會消失

```
x = x + y
```

將運算式 x + y 的結果指定給變數 x，x 原有的值會消失

```
int x = 1, y = 2, z;
z = x + y;
    1 + 2
      3
```

將整數 1 指定給變數 x，整數 2 指定給變數 y

將整數 x + y 的運算結果 3，指定給變數 z

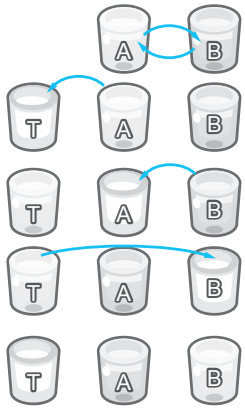
指定運算子可以同時多個連續使用，例如

```
float x, y, z ;
x = y = z = 2.5 ;
```

③ ② ①

將浮點數 2.5 指定給 z，再將 z 的值 2.5 指定給 y，再將 y 的值 2.5 指定給 x。

程式設計時，常會用到「兩數交換」，要將它寫成程式，可以想像成有 A B 二杯水，要將這兩個杯子的水互相交換。如右圖，解題步驟設計如下



1. 取出一個空杯 T。
2. 將 A 杯的水，倒入空杯 T。
3. 將 B 杯的水，倒入空杯 A。
4. 將 T 杯的水，倒入空杯 B。

步驟中倒水的動作就是指定運算，所以可將上述步驟轉換成程式碼如下

```
int temp;
temp = a;
a = b;
b = temp;
```

範例 3.2.1 兩數交換

輸入兩個整數 a, b，輸出 a, b 兩數交換的結果。

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int a, b, temp;
6      cout << " 請輸入兩個整數 ";
7      cin >> a >> b;
8      temp = a;
9      a = b;
10     b = temp;
11     cout << " 兩數交換後 a = " << a << "\tb = " << b << endl;
12     return 0;
13 }
```

兩數交換的過程

執行結果

```
請輸入兩個整數    2 6
兩數交換後        a = 6        b = 2
```

動動腦

若下列程式碼表示 a, b 兩數交換，空格內應填入甚麼運算式？

a = _____; b = a - b; a = a - b;

3.2.2 算術運算子（+, -, *, /, %）

算術運算子提供加減乘除及求餘數等運算，運算子包含如下表：

運算子	作用	運算式	結果	運算元數
+	正號	+5	+5	1
-	負號	-2	-2	1
+	加	5 + 2	7	2
-	減	5 - 2	3	2

運算子	作用	運算式	結果	運算元數
*	乘	5 * 2	10	2
/	除	16 / 3	5	2
%	餘	16 % 3	1	2

1. 例如數學式 $d = b^2 - 4ac$ 可使用以下敘述表示

```
d = b * b - 4 * a * c;
```

2. % 取餘數，又稱模數（modulus）運算子，兩個運算元必須都是整數，例如 10 % 2.5 是錯誤的語法，因為 2.5 不是整數。

3. 若兩運算元都是整數，/ 運算是求商，若其中一者為浮點數，則是除法運算。例如

```
15 / 2                                  可得   7
15.0 / 2 或 15 / 2.0 或 15.0 / 2.0      可得   7.5
```

4. 運算優先權順序

   ① 正負 + -　② 乘除餘 * / %　③加減 + -　④ 指定 =

   如下例，p 值的運算過程如下：

```
int i = 1, j = 2, k = 4, p;
p = i + 2 * j   -   22 / k;
          4          5
p = i + 4       -   5
      5         -   5
p = 0
```

   * / 運算權高於 + - =，所以先計算 2 * j 與 22 / k

   + - 運算權高於 =，所以先計算 i + 4 - 5

5. 算術運算子中，沒有指數次方運算子。

範例 3.2.2 時差換算 (d050)

小明的朋友住在美國，比台灣慢 15 個小時，他想打電話給他，但又怕半夜吵到對方。請寫一個程式，使用 24 小時制，將台灣時間換算成美國時間。

輸入：台灣時間

輸出：美國時間

解題方法

1. 若台灣時間為 h 點，美國時間應該是 (h + 24 - 15) % 24 點。
2. 例如台灣時間 21 點，美國時間應該是 (21 + 24 - 15) % 24 點，為 6 點。

```
 1  #include <iostream>
 2  using namespace std;
 3
 4  int main()
 5  {
 6      int h;
 7
 8      cout << "台灣時間（點）";
 9      cin >> h;
10
11      cout << "美國時間（點）" << (h + 24 - 15) % 24 << endl;
12
13      return 0;
14  }
```

直接輸出時間換算後的結果

執行結果

```
台灣時間（點）21
美國時間（點）6
```

範例 3.2.2-2 買原子筆 (d827)

原子筆一支 5 元，一打 50 元。若全班每位同學都要買一枝，最少要花多少錢？

輸入：班級學生數

輸出：所需的費用

解題方法

1. 將學生每 12 人分成一組，每組買 1 打，分組剩餘的學生，每人買 1 枝。若學生數為 n，共需買 (n / 12) 打，剩餘的學生數為 (n % 12)。
2. 所需的費用為 (n / 12) * 50 + (n % 12) * 5。

```
#include <iostream>
using namespace std;

int main()
{
    int n;
    cout << "班級學生人數 n = ";
    cin >> n;

    cout << "所需的費用為 "
         << (n / 12) * 50 + (n % 12) * 5 << "元" << endl;

    return 0;
}
```

打數所需的費用　枝數所需的費用

執行結果

```
班級學生人數 n = 42
所需的費用為 180 元
```

範例 3.2.2-3 分組問題 (d073)

上課時老師依照座號分組，每組 3 人。寫一個程式，能依座號查詢同學分到的組別。例如 8 號分到第 3 組。

輸入：座號

輸出：組別

解題方法

一組 3 人，先觀察「座號 / 3」的情形，如下表第 2 列，第 0 組只有 2 人，所以可先將座號右移兩位（座號 + 2），消除第 0 組，再除以 3，得第 3 列，便可以分配成每 3 人一組。

座號	1	2	3	4	5	6	7	8
座號 / 3	0	0	1	1	1	2	2	2
(座號 + 2) / 3	1	1	1	2	2	2	3	3

```
#include <iostream>
using namespace std;
int main()
{
    int n;
    cout << "請輸入座號";
    cin >> n;
    cout << "第" << (n + 2) / 3 << "組" << endl;
    return 0;
}
```

n + 2 是整數，所以 (n + 2) / 3 可得商

執行結果

```
請輸入座號 16
第 6 組
```

範例 3.2.2-4 秒數格式轉換

將輸入的秒數轉換成「時 : 分 : 秒」的格式。

解題方法

1. 1 小時 3600 秒，總秒數 ts 除以 3600 的商 (ts / 3600)，可得到小時 (h)。
2. 1 分鐘 60 秒，所以總秒數 ts 除以 60 的餘數 (ts % 60)，可以得到秒 (s)。
3. h 小時共 h * 3600 秒，總秒數減 h 小時的秒數 (ts – h * 3600)，為 h 小時以外剩餘的秒數，1 分鐘 60 秒，所以 (ts - h * 3600) / 60 可得到分 (m)。

```
 1  #include <iostream>
 2  using namespace std;
 3  int main()
 4  {
 5     int ts, h, m, s;
 6     cout << " 請輸入秒數 "; cin >> ts;
 7     h = ts / 3600;                          取得時數
 8     m = (ts - h * 3600) / 60;               取得分數
 9     s = ts % 60;                            取得秒數
10     cout << h << ":" << m << ":" << s << endl;
11     return 0;
12  }
```

執行結果

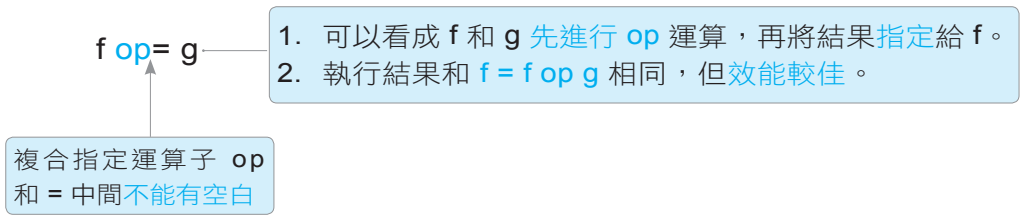
```
請輸秒數 39016
10:50:16
```

動動腦

上例中，分鐘數的運算可改成 m = ts □ 60 □ 60，小時數則可改寫成 h = ts □ 60 □ 60，□ 內應該填入何種運算子？

3.2.3 複合指定運算子

複合指定運算子是指定運算子和算術或位元運算子結合在一起的運算子。格式如下，其中運算子 op 是算術運算子 + - * / %（加減乘除餘）或位元運算子 & |~^（AND OR NOT XOR）<< >>（左移右移）之一。

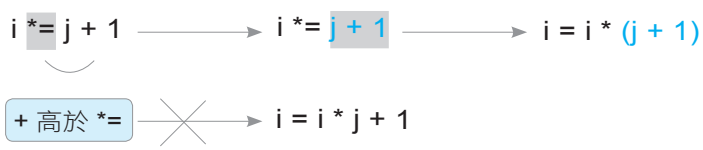


以 a += 3 為例，「+=」表示編譯器會先進行 +，再進行 =，所以 a 會先加 3，然後再將加 3 後的值指定給 a。以下是複合指定運算子的實例

複合指定運算式	一般指定運算式
i += j	i = i + j
i -= 1	i = i - 1
i /= 3	i = i / 3
i %= 2	i = i % 2

1. 使用複合指定運算子時，編譯器會直接在記憶體進行運算，不用取出變數值進行運算後，再將結果存回記憶體，所以比一般指定運算式執行效率佳。

2. 複合指定運算子是指定運算子的一種，所以算術運算子的優先權高於複合指定運算子。如下例，+ 的優先權高於 *=。



範例 3.2.3 複合運算子

依序輸入 x, y, z 三個整數，每輸入一個整數時，立即輸出所輸入之所有整數和。請使用複合運算子。

```
 1  #include <iostream>
 2  using namespace std;
 3  int main()
 4  {
 5     int x, y, z, sum;
 6     cout << "請輸入整數 x ";
 7     cin >> x;
 8     sum = x;
 9     cout << "sum = " << sum << endl;
10     cout << "請輸入整數 y ";
11     cin >> y;
12     sum += y;                    等同 sum = sum + y
13     cout << "sum = " << sum << endl;
14     cout << "請輸入整數 z ";
15     cin >> z;
16     sum += z;                    等同 sum = sum + z
17     cout << "sum = " << sum << endl;
18     return 0;
19  }
```

執行結果

```
請輸入整數 x 10
sum = 10
請輸入整數 y 15
sum = 25
請輸入整數 z 20
sum = 45
```

3.2.4 遞增 / 遞減運算子（++ / --）

遞增運算子 ++ 可提供變數加 1 的運算，遞減運算子 -- 可提供減 1 的運算。下面四個敘述都會讓 x 加 1 或減 1。

```
++x;
x++;
x += 1;
x = x + 1;
```

```
--x;
x--;
x -= 1;
x = x - 1;
```

1. 遞增 / 遞減運算子前置（prefix）與後置（postfix）是不同的。前置是指 ++ 或 -- 在變數的前面，後置是指 ++ 或 -- 在變數的後面。

順序	運算式	運算說明
前置	++i 或 --i	先加 1 或減 1
後置	i++ 或 i--	後加 1 或減 1

例如下表，若 x = 3，y = ++x 和 y = x++ 是不同的。y = ++x 會先執行 ++，再執行 =；y = x++ 會先執行 =，再執行 ++。兩者執行的結果不一樣。

順序	實例	執行順序	運算順序	運算過程	結果
前置	y = ++x	先 ++ 後 =	++x; y = x;	y = ++x → x = 4 y = x → y = 4	x = 4 y = 4
後置	y = x++	先 = 後 ++	y = x; x++;	y = x++ → y = 3 x++ → x = 4	x = 4 y = 3

2. 使用遞增 / 遞減運算子和複合指定運算子，編譯器直接在記憶體進行運算，執行效率較佳。

3.2.5 逗號運算子（,）

逗號運算子，是用來分隔敘述的，其運算優先權最低。例如

```
int i = 1, j = 2;
++i, ++j;
```

3.2.6 位元運算子（&, |,~, ^, <<, >>）

位元運算子如下表，是指逐位元（bitwise）進行運算的運算子。電腦使用二進位，因此位元運算最符合電腦運作原理，專業程式設計師常使用位元運算。

複合指定運算子、遞增 / 遞減運算子、位元運算子的執行效率較好，這是 C 系列語言較其他程式語言強的特點，初學者可能較不習慣使用，但值得深入研究，提高程式設計的能力。

運算子	運算	中文意義
&	AND	且
|	OR	或
~	NOT	反
^	XOR	互斥
<<	SHL	左移
>>	SHR	右移

1. &（AND）運算：下表為 a, b 兩個 bits 進行 & 運算顯示的結果。

a	b	a & b
0	0	0
0	1	0
1	0	0
1	1	1

只有兩個 bit 都是 1 時，& 位元運算的結果才會是 1，否則為 0。例如

$250 \ \& \ 180 = 1111\ 1010_2 \ \& \ 1011\ 0100_2 = 1011\ 0000_2 = 176$

```
  1 1 1 1   1 0 1 0
& 1 0 1 1   0 1 0 0
-----------------
  1 0 1 1   0 0 0 0
```

2. |（OR）運算：下表為 a, b 兩個 bits 進行 | 運算顯示的結果。

a	b	a | b
0	0	0
0	1	1
1	0	1
1	1	1

只要其中一個 bit 是 1 時，| 位元運算的結果就會是 1，只有兩個 bit 都是 0 時，結果才會是 0。例如

$250 \mid 180 = 1111\ 1010_2 \mid 1011\ 0100_2 = 1111\ 1110_2 = 254$

```
  1 1 1 1   1 0 1 0
| 1 0 1 1   0 1 0 0
-----------------
  1 1 1 1   1 1 1 0
```

3. ~（NOT）運算：下表為 bit a 進行 ~ 運算顯示的結果。

a	~a
0	1
1	0

結果和輸入相反，例如

$\sim 250 = \sim 1111\ 1010_2 = 0000\ 0101_2 = 5$

```
~ 1 1 1 1   1 0 1 0
-----------------
  0 0 0 0   0 1 0 1
```

4. ^（XOR）運算：下表為 a, b 兩個 bits 進行 ^ 運算顯示的結果。

a	b	a ^ b
0	0	0
0	1	1
1	0	1
1	1	0

只有兩個 bit 不同時，^ 位元運算的結果才會是 1，否則為 0。例如

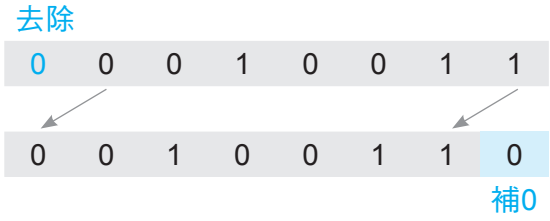
$250 \text{ ^ } 180 = 1111\ 1010_2 \text{ ^ } 1011\ 0100_2 = 0100\ 1110_2 = 78$

```
    1 1 1 1   1 0 1 0
^   1 0 1 1   0 1 0 0
-------------------------
    0 1 0 0   1 1 1 0
```

5. << 左移運算（SHL）

往左移動一個變數的所有位元，移動後，最高位元會消失，最低位元補 0。例如

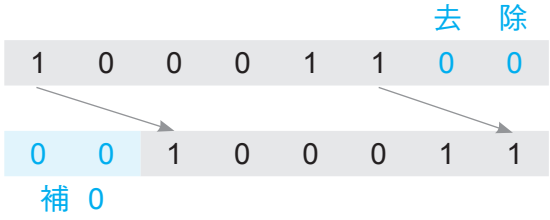
$19 << 1 \rightarrow 0001\ 0011_2 << 1 \rightarrow 0010\ 0110_2 = 38$



因此左移 1 個位元等於 ×2，左移 2 個位元等於 ×4（2^2），依此類推。

6. >> 右移運算（SHR）往右移動一個變數中的所有位元，移動之後，最低位元會消失，最高位 元補 0。例如

$140 >> 2 \rightarrow 1000\ 1100_2 >> 2 \rightarrow 0010\ 0011_2 = 35$



因此右移 1 個位元等於 / 2，右移 2 個位元等於 / 4（2^2），依此類推。

位元運算子的優先權為 ① ~ ② << >> ③ & ④ |。左移 << 和右移 >> 的位元移量若是負值，或超過運算元的 bits 數，可能會發生不可預期的結果。

十進位數所有位數都向左移動一位，等於原數 ×10，移動兩位，等於 ×100 (10^2)。向右移動一位，等於原數 ÷10，移動兩位，等於 ÷100 (10^2)。

同理，二進位數所有位數向左移動一位，等於原數 ×2，移動兩位，等於 ×4 (2^2)。向右移動一位，等於原數 ÷2，移動兩位，等於 ÷4 (2^2)，所以位元移動常被使用於乘除運算。例如

```
8 << 1 = 16 (8×2) // 0000 1000 << 1 → 0001 0000 = 16
8 << 2 = 32 (8×4) // 0000 1000 << 2 → 0010 0000 = 32
8 >> 1 = 4 (8÷2)  // 0000 1000 >> 1 → 0000 0100 = 4
8 >> 2 = 2 (8÷4)  // 0000 1000 << 2 → 0000 0010 = 2
```

以下列舉一些位元運算子的應用，大家可使用程式測試是否正確。

1. 判斷 n 是奇數或偶數

```
n & 1;
```

結果 0 為偶數，1 為奇數。

二進位奇數最右邊的位元必為 1，偶數必為 0。1 除了最右邊位元為 1 外，其餘位元皆為 0，所以 n & 1 只會留下最右邊的位元，其餘位元都會被 & 0 運算遮掉，這就是位元遮罩。

2. 找出最大的非負整數 max

```
unsigned max = ~0;
```

無正負號的整數 0 有 4 bytes 的二進位數 0，~0 會變成 4 bytes 都是二進位 1，每個 bit 都是 1，就是最大的非負整數。

3. x, y 兩數交換

```
x ^= y ^= x ^= y;
```

使用互斥複合運算子 ^=，一行敘述就能完成兩數的交換。

4. 判斷整數 n 是不是 2 的次方

```
n & -n
```

結果等於 n，則 n 是 2 的次方；不等於 n，則 n 不是 2 的次方。

5. 把整數 n 的第 5 bit 強制標記為 1

```
n | (1 << 4)
```

1 << 4 會將 1 左移 4 位元，至第 5 bit。因為第 5 bit 是 1，再和 n 進行 | 運算，運算結果的第 5 bit 也會是 1。

6. 把整數 n 的第 5 bit 強制反轉，也就是 0 變 1，1 變 0

```
n ^ (1 << 4)
```

1 << 4 運算後的第 5 bit 是 1，如果 n 的第 5 bit 是 1，1 ^ 1 為 0，結果的第 5 bit 會是 0。同理，如果 n 的第 5 bit 是 0，0 ^ 1 為 1，結果的第 5 bit 會是 1。因此此運算會將整數 n 的第 5 bit 強制反轉。

位元運算比一般運算執行效能好，但由上面例子可以發現，這些程式並不易閱讀，引用時，最好輔助註解說明，避免程式的後續維護困難。

3.3 運算子的優先權與結合性

各運算子的運算優先權及結合性（associativity）如下表。

序	運算子	說明	結合性
1	::	範圍解析	-
2	++	後置遞增	左
	--	後置遞減	
	( )	函數呼叫	
	[]	陣列標註	
	->	指標成員	
3	++	前置遞增	右
	--	前置遞減	
	+	正號	
	-	負號	
	~	位元取反	
	!	邏輯 NOT	
	sizeof	資料的大小	
	&	變數的位址	
	*	指標指向的值	
	new	動態記憶體配置	
	delete	動態記憶體釋放	
	(類型)	強制轉換類型	
4	.* ->*	指標存取	
5	*	乘	左
	/	除	
	%	餘數	
6	+	加	
	-	減	
7	<<	位元左移	
	>>	位元右移	

序	運算子	說明	結合性
8	<	小於	左
	<=	小於等於	
	>	大於	
	>=	大於等於	
9	==	相等	
	!=	不相等	
10	&	位元 AND	
11	^	位元 XOR	
12	|	位元 OR	
13	&&	邏輯 AND	
14	||	邏輯 OR	
15	=	指定	右
	*=	複合指定	
	/=		
	%=		
	+=		
	-=		
	>>=		
	<<=		
	&=		
	^=		
	|=		
	? :	條件	
16	,	逗號	左

1. 同一順序之運算子具有相同的優先權。例如 *、/ % 有相同的優先權。
2. 無法確定優先權時，可使用小括號 ( ) 安排運算的順序。
3. 除一元運算子及指定運算子（含複合指定）是右結合外，其他都是左結合。

學習挑戰

一、選擇題

1. (　　) 若 a = 2, b = 3, c = 4, d = 5，則 b / a + c / b + d / b 之值為何？

   (A) 3　　(B) 4　　(C) 5　　(D) 6

2. (　　) 執行下列敘述後，p 值為何？

   ```
   int i = 1, j = 2, k = 4, p;
   p = i + 2 * j – i / k;
   ```

   (A) 6　　(B) 4　　(C) 5　　(D) 2

3. (　　) 若 n = 583，執行下列敘述後，下列何者不正確？

   ```
   a = n % 10; n /= 10; b = n % 10; c = n / 10;
   ```

   (A) a = 3　　(B) b = 8　　(C) c = 5　　(D) n = 5

4. (　　) 若 x = 3，執行運算式 y = ++x 後，x, y 之值為

   (A) 4, 4　　(B) 3, 3　　(C) 4, 3　　(D) 3, 4

5. (　　) 下列哪一個敘述無法將變數 x 的值加 1 ？

   (A) x = x + 1;　(B) x =+ 1;　(C) x++;　(D) ++x;

6. (　　) 72 & 23 =

   (A) 0　　(B) 1　　(C) 64　　(D) 23

7. (　　) 22 | 23 =

   (A) 22　　(B) 23　　(C) 45　　(D) 1

8. (　　) 使用位元運算子，a = b * 128 可以寫成 a = ？

   (A) b << 6　　(B) b << 7　　(C) b >> 6　　(d) b >> 7

9. (　　) 下列那一個運算式可作為奇偶數的判斷？

   (A) n & 1　　(B) n | 1　　(C) n << 1　　(D) n >> 1

10.(　　) a = sizeof(char), b = sizeof(int), c = sizeof(double), d = sizeof(bool)，何者最大？

(A) a　　(B) b　　(C) c　　(D) d

11.(　　) 執行下列敘述後，y 值為何？

int a = 2, y ; float b = 0.6; y = a + b;

(A) 2　　(B) 2.6　　(C) 語法錯誤　　(D) 編譯錯誤

二、應用題

1. 使用三元運算子表示敘述「若 a > b 則 min = b，否則 min = a」。
2. 寫出三種將 a, b 兩數交換的敘述。
3. 百貨公司周年慶推出消費每滿 1000 元，就折抵 100 元。寫一程式，輸入消費金額後，能輸出應付金額。
4. 小明到商店購物，花了 n 元 (<= 1000)，他拿出一張千元紙鈔，請寫一程式，能輸出最多要找回幾張 500, 100 元紙鈔，幾個 50, 10, 5, 1 元銅板。
5. 輸入一梯形的上底、下底、高，輸出此梯形的面積。
6. 寫一程式，輸入一個 6 位數後，能輸出其奇位數與偶位數和的差。例如輸入 201856，奇位數的和 6 + 8 + 0 = 14，偶位數的和 5 + 1 + 2 = 8，所以輸出 14 - 8 = 6。
7. XOR 運算子 ^ 可以做為資料加解密運算。寫一程式，輸入任一字元與某個整數 後，將此字元連續與此數 XOR 運算兩次，觀察會有何結果。