國立屏東大學 資訊工程學系 物件導向程式設計

5. 運算子

5.0.1 算術運算子

算術運算子就如同我們一般在數學式子中，所使用的運算符號，例如加減乘除等。table 1為C/C++語言支援的算術運算子：

operator	意義	unary/binary
+	正	unary
-	負	unary
+	加法	binary
-	減法	binary
*	乘法	binary
/	除法	binary
%	餘除	binary

Tab. 1: Arithmetic Operators

5.0.2 指定運算子

等號「=」被稱為C/C++語言的指定運算子(assignment operator)，用以將等號右方的值指定(assign)給等號左方，我們將其稱為是右關聯(right associativity)的運算子。例如：

- i = 5;
- j = i;
- k = 10 * i + j;

要注意的是，若等號左右兩邊的資料型態不一致時，C/C++語言會進行自動的型態轉換，例如:

假設宣告有：

```
int i;
float j;

i = 83.34f; // i = 83

j = 136;    // j=136.0
```

在一個運算式中，有時可以出現一個以上的等號，例如:

- i = j = k = 0;

等同於

1. k=0;

2. j=(k=0);
3. i=(j=(k=0));

請考慮以下的程式片段，想想看其輸出結果為何？

```
i = 1;
k = 1 + (j=i);
cout << i << ", " << j << ", " << k << endl;
```

5.0.3 複合指定運算子

假設i=3，考慮以下的運算式：

- i = i + 2;

其結果是先進行等號右邊的運算，得到結果為5後，將數值5給定到等號左邊的變數i，因此，最後i的值等於5。針對這種情形，C/C++語言提供複合指定(compound assignment)運算子，例如「+=」，上面的運算式可重寫為:

- i += 2;

其它常見的複合指定運算子，還有 -=, *=, /=與 %=。這些複合指定運算子為右關聯，請考慮下列的運算式：

- i += j += k;

等同於

- i += (j += k);

5.0.4 遞增與遞減運算子

當我們需要將某個變數的值遞增時，可以寫做：

i = i + 1; 或 i += 1;

但是C/C++語言還提供++與--這兩個運算子，分別是

- ++，遞增(increment)運算子
- --，遞減(decrement)運算子

我們可以把i=i+1或i+=1，改寫為：

i++;

同理，還有i- -可以遞減i的數值。但是++與- -可以選擇為prefix operator或postfix operator，視其寫在變數的前面或後面而定。放在前面，例如++i，會先遞增i的數值，然後再傳回新的i的數值；但寫在後面，例如i++則會先傳回i現有的數值，然後才將i的值遞增。

考慮以下的程式碼，想想看輸出的結果為何？

```
i=1;
cout << ++i << endl;
cout << i << endl;
cout << i++ << endl;
cout << i << endl;
```

5.0.5 sizeof運算子

sizeof運算子可以計算型態或變數等的記憶體空間，其用法有二：在sizeof後接一組括號並在其中放置要計算空間的對象，或者直接在sizeof後放置欲計算空間的對象(無須括號)，請參考下例：

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

int main()
{
  char *month[]= {"January", "February", "March", "April", "May", "June",
      "July", "August", "September", "October", "November", "December" };

  cout << "Size of char type = " << sizeof(char) << " byte." << endl;
  cout << "Size of month array = " << sizeof month << " bytes." << endl;

  return 0;
}
```

5.0.6 關係運算子

關係運算子是一個二元的運算子，用以判斷兩個運算元(數值、函式、變數或運算式)之間的關係，其可能的關係有：大於、小於、等於、或不等於。C/C++語言提供以下的關係運算子，如table 2。

符號	範例	意義
>	a > b	a 是否大於 b
<	a < b	a 是否小於 b
>=	a >= b	a 是否大於或等於 b
<=	a <= b	a 是否小於或等於 b

Tab. 2: Relational Operators

雖然我們已經提過。C/C++語言將數值0視為false。並將其它所有非0的數值視為true。但關係運算子會以數值0代表運算結果為false。以數值1代表true。還要注意關係運算子較算術運算子的優先順序低，所以像是 x + y < i - j 等同於 (x + y) < (i - j)。在C/C++語言中。x < y < z等同於 (x < y) < z。因為關係運算子為左關聯。假設x=1, y=3, z=5。 x<y<z ⇒ (x < y) <z ⇒ 1 < z ⇒ 1。要注意的是C++已提供了bool型態，因此也可以直接使用true或false來代表運算的結果。

5.0.7 相等運算子

相等運算子是一個binary運算子，用以判斷兩個運算元(數值、函式、變數或運算式)之值是否相等。C/C++語言提供以下的關係運算子，如table 3。

符號	範例	意義
==	a == b	a 是否等於 b
!=	a != b	a 是否不等於 b

Tab. 3: Equality Operators

同樣地，相等運算子會以數值0代表運算結果為false，以數值1代表true，還要注意相等運算子與關係運算子一樣，其優先順序都較算術運算子來得低。

<note important>是 == ， 不是 = 。 千萬不要將比較兩數是否相等的==寫成=，這實在是一個常常會遇到的錯誤!建議您以後如果遇到程式執行結果錯誤，但找不出任何問題時，試試檢查一下所有的 = 與 ==，有很高的機會可以改正您的程式。</note>

5.0.8 邏輯運算子/Logical Operator

邏輯運算子共有以下三種，如table 4。

符號	意義	unary or binary
!	NOT	unary
&&	AND	binary
||	OR	binary

Tab. 4: Logical Operators

其運算結果請參考table 5的真值表：

X	Y	NOT X	X AND Y	X OR Y
0	0	1	0	0
0	1		0	1
1	0	0	0	1
1	1		1	1

Tab. 5: Truth Table

假設變數score代表c語言的修課成績，以下的邏輯運算即為檢查成績是否介於0~100：

```
((score >= 0) && (score <=100))
```

5.0.9 條件運算子

C/C++語言還有提供一種特別的運算子，稱為條件運算子(conditional operator)，可依條件決定運算式的傳回值，其語法如下：

```
expression1 ? expression2 : expression3
```

運算式的運算結果expression1的值為true(非0的數值)或false(數值0)而定，當expression1為true時，傳回expression2的值；否則當expression1為false時，傳回expression3的值。事實上，這等同於下面的if敘述：

```
if (expression1)
   result = expression2;
else
   result = expression3;
```

條件式敘述有可能是因為像「如果expression為真則…. 否則….」這樣的述敘，在程式中出現的機會很高的緣故吧！請參考以下的應用：

```
int x=1, y=2, z;
if(x>y)
   z=x;
else
   z=y;
```

上面這段程式碼是令z為x與y兩者中較大的值，如果以條件運算式改寫，則只要寫成

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

即可，是不是簡化很多？下面這行程式，假設score為學生成績，則可以簡單地檢查score是否大於100，若超過100則以100分計。

```
score = score > 100 ? 100 : score ;
```

5.0.10 優先順序與關聯性

我們將C/C++的運算子之優先順序與關聯性彙整於table 6

運算子	符號
一元運算子	+(正)、-(負)□++□--□!(NOT)□sizeof
算術運算子(乘除)	*□/□%
算術運算子(加減)	+□-
關係運算子	>=□<=□>□<
相等運算子	==□!=
邏輯運算子	&&□||
條件運算子	?:

運算子	符號
指定運算子	=□*=□/=□%=□+=□-=

Tab. 6: 各運算子的優先順序(由高至低)