國立屏東商業技術學院 資訊工程系 物件導向程式設計

5. 運算式與算術運算子

<本章絕大部份內容與C語言一致>

- 運算式的構成
- 運算元和運算子
- 算術相關運算子
- 運算子的優先順序與關聯性

5.1 運算式、運算元與運算子

運算式(Expressions)由運算元(operands)與運算子(operators)所組成，以下是一個簡單的運算式：

```
SomeVar = i +3;
```

其中變數SomeVar和i以及資料 3，稱之為運算元(operands)而=和+稱之為運算子(operators)此運算式的動作即為將 i 的值加上 3，再將結果指定給變數SomeVar裡。

運算元可以是變數、常數、數值以及函式，甚至可以是另一個運算式。例如下面的運算式：

```
123 + 456    //123及456這兩個數值皆為運算元

r * 3.1415   //變數r與數值3.1415皆為運算元

getchar() - 'A' //getchar()函式與字元值'A'皆為運算元

rand()%10    // rand()函式與10皆為運算元
```

其中123, 456, r, 3.1415, getchar(), 'A', rand()函式與10等為運算元， +, *, -, %等為運算子。

根據其運算性質的不同，運算子可概分為算術(Arithmetic)運算子、關係(Relational)運算子、邏輯(Logical)運算子…等類別，本章將針對與算術相關的運算子加以介紹。

<note important> 注意C++語言是一個左值(left value)的程式語言，意即任何數學運算式都是先計算等號右邊的數值，再將其值指派給等號左邊。例如n=x與x=n其運算結果是不一樣的。假設n與x的數值原本分別為3與5n=x會把等號右邊的x的數值指派給左邊的n因此其結果使得n與x的數值皆為5；同理x=n會使兩者的值皆為3 </note>

5.2 算術運算子/Arithmetic Operator

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

其中unary指的是該運算子只需一個運算元，例如

```
x = +5;
y = -x;
```

binary運算子，指的是需要兩個運算元的運算，例如加、減、乘、除等，分別以+, -, *, /表示，至於%運算子則得指是進行除法後的餘數。以下為一個例子：

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

int main()
{
  int x;
  int y;

  cout << 345 + 132 << endl;

  x=123;
  y=555;
  x=x-y;

  cout << x << endl;
  cout << 12*3.1415926 << endl;
  cout << 85/8 << endl;
  cout << 85/8.0 << endl;
  cout << 85%8 << endl;
}
```

其執行結果如下：

```
477
-432
37.6991
10
10.625
5
```

優先順序與關聯Precedence and Associativity

在運算式中，運算子的優先順序如下表：

高優先	+ - (unary)
	* / %
低優先	+ - (binary)

當一個運算式中有多個運算子，且其優先順序亦相同時，所有binary的算術運算子皆為左關聯(left associative)，意即由左往右方向計算。例如：

- i - j - k 等同於 (i - j) - k
- i * j / k 等同於 (i * j) / k

至於unary的運算子，則是右關聯(right associative)，例如：

- - + i 等同於 -(+i)

5.3 指定運算子/Assignment Operator

等號=被稱為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(所以C語言被稱為是left value左值的程式語言)。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
```

等號是右關聯

在一個運算式中，有時可以出現一個以上的等號，此時等號為右關聯，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;
```

5.4 複合指定運算子/Compound Assignment

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 %=。

右關聯

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);

5.5 遞增與遞減運算子/Increment and Decrement Operators

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;
```

5.6 優先順序與關聯性

我們將本章所有出現的運算子之優先順序與關聯性彙整於table 2

Precedence	Operator	Associativity
1	++ (postfix)	left
	- - (postfix)	
2	++ (prefix)	right
	- - (prefix)	
	+ (unary)	
	- (unary)	
3	*	left
	/	
	%	
4	+	left
	-	

Precedence	Operator	Associativity
5	=	right
	*=	
	/=	
	%=	
	+=	
	-=	

Tab. 2: Precedence and Associativity of Arithmetic-related Operators