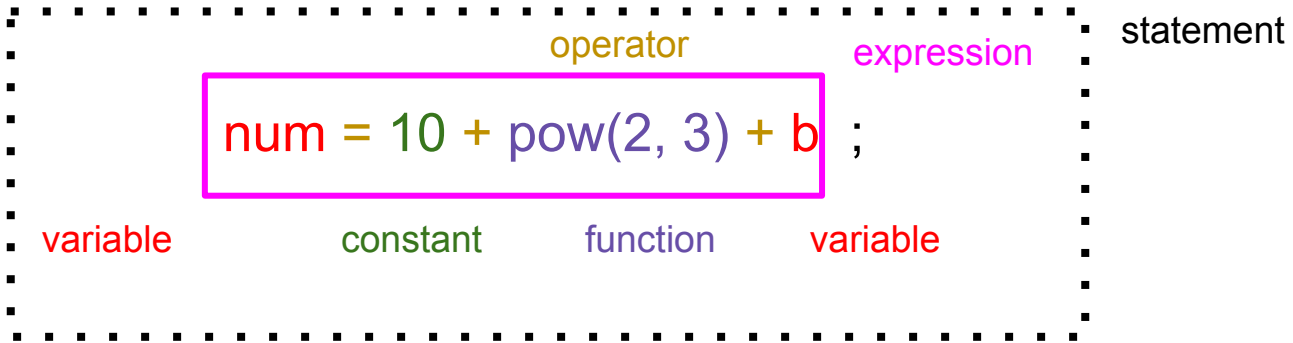
Chapter5-運算子、運算式與敘述

運算式 (Expression)

- 運算式(Expression)是由運算元(operand)以及運算子(operator)所組成 1 + 1
- 運算元(operand):
  - 可以為常數(constant)、變數(variable)、函式(function)
- 運算子(opertor):
  - 數學符號上的 +, -, *, /, 以及一些能讓 operand 進行特定運算、操作的符號



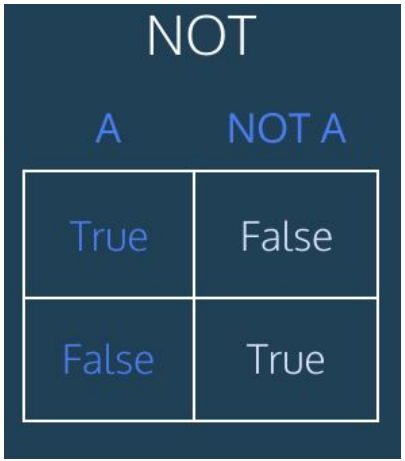
設定運算子 assignment operator

- 『=』: 為設定運算子 (assignment operator) 用來進行賦值
- 為右結合性 (right associativity) 指由右向左進行運算

```
a = b = c + 9;
```

一元運算子 (uniary operator)

- operator(運算子) 只須一個 operand(運算元) 就可以進行運算
- 一元運算子:『+』(正號)、『-』(負號)、『!』(邏輯 NOT 運算)
  - +3; /* 表示正3 */
  - -a; /* 表示負a */
  - !a; /* 表示NOT a */
- NOT 運算如右圖所示
- 在 C 語言裡非零數為 true (真)、零為 false (假)
- C 語言並無保留關鍵字 true, false
- 若想使用可 #include <stdbool.h>



可見範例程式: /lecture3/UniaryOP.c

算術運算子 (arithmatic operator)

- 數學運算中的**加、減、乘、除**以及**取餘數**的運算
- 其中要注意的是 **除(/)** 以及 **取餘數(%)**
- 除法在 int 型別下會捨棄小數部份 (取 floor)
- 如果想要保留小數部位可以先用**型別轉換**為浮點數，浮點數在除法運算中則會保留小數部份
- 取餘數(%) 顧名思義會保留餘數部份 e.g. 5%2 結果會是 1
- 但要注意的是 % 運算子只能對整數操作，若要對浮點數則需用 fmod (#include <math.h>)

可見範例程式: /lecture3/ArithmaticOP.c

C 語言的取餘數(%) 與 常見的數學 module 的定義差別

- 在正數情況下, C 語言的取餘數與常見的數學 module 並無差別
- 在負數情況下, 就有差別
- 數學定義的 module

  a ÷ b = c … d 其中 a ≡ d (mod b) 且 bc 為最大的小於等於 a 的數

  所以 -5 ≡ 1 (mod 2)

- 但在 C 語言 -5 % 2 等於 -1
- 在 C 語言中可以使用 (a%b+b)%b 的方式來跟數學常見定義的 module 做等價

可見範例程式 : /lecture3/ArithmaticOP.c

關係運算子 (relational operator)

- 利用關係運算子運算所得到的結果為布林值(0: false, 1: true)
- 可以搭配需要判斷條件的敘述 if, else if, while, for, do while, ? :

關係運算子	說明	實例
>	大於	a > b
>=	大於或等於	a >= b
<	小於	a < b
<=	小於或等於	a <= b
==	等於	a == b
!=	不等於	a != b

可見範例程式 : /lecture3/RelationalOP.c

遞增與遞減運算子 (increment & decremet operator)

- C 語言定義 ++, -- 符號分別為遞增1、遞減1運算子
- 遞增遞減運算子只能加在變數 (variable) 上，可擺在變數的左、右邊
- e.g. a++; 等價於 a = a+1　a--; 等價於 a = a-1
- 注意: 擺在變數的左邊或右邊的執行行為不一樣!!!
- 擺在左邊(口訣: 先++)，則會先進行遞增減後在執行 statement
- 擺在右邊(口訣: 後++)，則會先執行 statement 後在進行遞增減

可見範例程式 : /lecture3/IncrementDecrementOP.c

邏輯運算子 (logical operator)

- C 語言中定義 && 為 AND operator，定義 || 為 OR operator
- 真值表如圖所示
- 值得注意的是 C 語言採用 lazy evaluation 的方式 (或稱 short circuit)
- A && B: 如果 A 是 false 則不會在執行 B
- A || B: 如果 A 是 true 則不會在執行 B

Truth table for OR , AND, and NOT operations
True = 1, False = 0

A	!A	B	!B	A || B	A&&B
0	1	0	1	0	0
1	0	0	1	1	0
0	1	1	0	1	0
1	0	1	0	1	1

illustrates the applications of Boolean operators

可見範例程式 : /lecture3/LogicalOP.c

括號運算子 parentheses operator ()

- 運來提高(到最高)運算式的優先度
- 與一般的數學括號無異 ()

三元運算子 (Ternary Operator)

- C 語言中三元運算子只有其中一個, 就是 ? :
- 語法: A ? B : C, 其中 A, B, C 分別為 expression (運算式)
- 語意: A 為判斷式
  - 當 A 為 true (非零) 則執行 B 不執行 C 且整個三元運算元的結果為 B 的結果
  - 當 A 為 false (零) 則執行 C 不執行 B 且整個三元運算元的結果為 C 的結果
  - e.g. A ? B : C ? D : E;

複合賦值運算子 (compound assignment operator)

运算符	功能说明	示例
+=	加赋值复合运算符	a+=b 等价于 a=a+b
-=	减赋值复合运算符	a-=b 等价于 a=a-b
*=	乘法赋值复合运算符	a*=b 等价于 a=a*b
/=	除法赋值复合运算符	a/=b 等价于 a=a/b
%=	求余赋值复合运算符	a%=b 等价于 a=a%b
&=	位与赋值复合运算符	a&=b 等价于 a=a&b
|=	位或赋值复合运算符	a|=b 等价于 a=a|b
^=	位异或赋值复合运算符	a^=b 等价于 a=a^b
>>=	位右移赋值复合运算符	a>>=b 等价于 a=a>>b
<<=	位左移赋值复合运算符	a<<=b 等价于 a=a<<b

運算子優先度 (precedence)

- Bitwise 運算子之後章節會在講
- 優先度最低的是 Comma operator 逗號運算子

Operator	Description	Associativity

`()`
`[]`
`.`
`->`
	Parentheses (grouping)
	Brackets (array subscript)
	Member selection via object name
	Member selection via pointer
		left-to-right

`++ --`
`+ -`
`! ~`
`(type)`
`*`
`&`
`sizeof`
	Unary preincrement/predecrement
	Unary plus/minus
	Unary logical negation/bitwise complement
	Unary cast (change *type*)
	Dereference
	Address
	Determine size in bytes
		right-to-left

`* / %`	Multiplication/division/modulus	left-to-right
`+ -`	Addition/subtraction	left-to-right
`<< >>`	Bitwise shift left, Bitwise shift right	left-to-right
`< <=` `> >=`	Relational less than/less than or equal to Relational greater than/greater than or equal to	left-to-right
`== !=`	Relational is equal to/is not equal to	left-to-right
`&`	Bitwise AND	left-to-right
`^`	Bitwise exclusive OR	left-to-right
`|`	Bitwise inclusive OR	left-to-right
`&&`	Logical AND	left-to-right
`||`	Logical OR	left-to-right
`?:`	Ternary conditional	right-to-left

`=`
`+= -=`
`*= /=`
`%= &=`
`^= |=`
`<<=`
`>>=`
	Assignment
	Addition/subtraction assignment
	Multiplication/division assignment
	Modulus/bitwise AND assignment
	Bitwise exclusive/inclusive OR assignment
	Bitwise shift left/right assignment
		right-to-left

`,`	Comma (separate expressions)	left-to-right

逗號運算子 (comma operator)

- 逗號運算子為一個雙元運算子, 優先度最低的運算子, ~~存在感也最低~~
- 注意: 逗號『, 』同時也可當作分隔符號 (seperator)
- 語法: A, B 其中 A, B 為運算式
- 語意: A, B 中會執行 A 在執行 B 並把 B 的結果當作整個運算式的結果
- 在某些情境下逗號運算子蠻好用的, 讓整個程式看起來清楚簡潔

```
string s; // In C++
while(read_string(s), s.len() > 5)
{
   //do something
}
```

運算式的型別轉換

- 在執行運算式的過程中，若有不同型別運算式裡，C 語言處理的規則是以不流失資料為前提來進行資料間的型別轉換 i.e. 就是在運算前先把小範圍資料型別轉換為範圍較大的型別，在執行運算

```
char ch = ‘a’;
short s = -2;
int i = 3;
float f = 5.3f;
double d = 6.28;

(ch / s) - (d / f) - (s + i);
```