2.3 運算子與運算優先順序

在掌握了基本輸入、輸出之後，我們已經具備「將資料讀進電腦」，「將處理後資料送回外界」的能力，接下來重點就是中間的「處理」，也就是運算的部分。

首先我們要認識兩個名詞：

- **運算元(operands)**
- **運算子(operator)**

以 2+3 為例，`2` 和 `3` 都是運算元，`+` 是運算子。

我們可以把運算子想成是「運算符號」，運算元則是「運算的對象」。

指定(assign)運算子

在 C++ 中，`=` 不是 **等於(equal)** 而是 **指定(assign)**

`a = 3` 是「把 3 **指定** 給 a 這個變數」，而非表示「a 和 3 的值是相等的」

執行完這行後，變數 a 的值就會變成 3。

```
a = 3;
cout << a; // 3
```

運算子 `=` 會將其右側的運算結果，指定到左側的儲存空間。

例如 `v = 3+5` 是把 3+5 的運算結果指定給變數 v。

也就是說如果 `=` 的右側不是單純的值而是運算式，要先完成運算後，再將運算結果指定給其左側的變數。 所以，執行完這行後，變數 v 的值就會變成 8。

```
v = 3+5;
cout << v; // 8
```

想想看，以下這段程式執行後的輸出為何？

```
int a;
a = 3;
a = a + 2;

cout << a << endl;
```

其中第 2 行是把 3 指定給 a，所以執行後 a 的值為 3。

第 3 行因為 `=` 的右側是運算式 `a + 2`，因此要先完成這個運算，目前 a 的值是 3，所以 `a+2` 的運算結果是 5。

接著可以想像第3行變成 `a = 5`，所以整行執行後，a 的值為 5。

千萬不要用數學符號的角度把 `=` 當成「等於」去看待 `a = a + 2`，這行敘述，否則你會看不懂它。

練習

以下這段程式執行後，a 和 b 的值各為何？

```
int a = 3;
int b = 5;

a = a + b;
b = a - b;
a = a - b;
```

```
cout << "a = " << a << endl;
cout << "b = " << b << endl;
```

算術(arithmetic)運算子

「乘、除」 運算子的優先權高於「加、減」運算子，也就是在運算時會 **先乘除後加減**。

例如：`1+2*3-4;` 的運算結果是 `3`。

和數學運算式一樣，可以加上括號指定優先運算的部分。

例如：`(1+2)*(3-4);` 的運算結果是 `-3`。

有個需要留意的地方是，不同於我們在數學課中用小括號、中括號、大括號來一層層的指定優先運算的層次。C++裡只有小括號，不管是幾層都是用小括號來表示。中括號、大括號這兩個符號是用在其他地方。

數學課裡的 $(1+2)\times[(3-4)+2]$

在 C++ 裡是 `(1+2)*((3-4)+2)`

整數除法

除法運算子 `/` 有個需要注意的地方，如果它的左、右側運算元都是整數型別，那麼其運算結果也會是整數。

`cout << 5/2;` 的輸出不會是 `2.5`，而是`2`.

但 `cout << 5.0/2;`、`cout << 5/2.0;`、`cout << 5.0/2.0;` 的輸出，都是 `2.5`。

取餘數

另一個只能用在整數型別的 **模數(modulo)** 運算子 `%`，是用來計算兩整數相除後的餘數。

```
cout << 8%3 << endl; // 2
cout << 6%2 << endl; // 0
cout << 3%7 << endl; // 3
```

在許多演算法中，取餘數是很重要的運算，所以這是個很重要的運算子。

關係(Relational)運算子

下一個單元開始，我們的程式碼將不再只是單純按順序一行一行執行下去，它開始可以按照條件選擇接下來要執行哪一條分支路線。

例如：如果「a大於0」那麼.....否則.....。

那個條件在當下有兩種可能：

- 若成立，表示其為 **真(true)**
- 若不成立，表示其為 **偽(false)**

關係運算子的運算結果是 **布林值(Boolean)**。不同於整數型別的值有多種可能的值 `... -2, -1, 0, 1, 2, ...`，布林型別的值只有兩種 `true`, `false`。

以下是六個關係運算子的作用。

名稱	運算子	範例	範例運算結果
等於	`==`	`2==3`	false
不等於	`!=`	`2!=3`	true
大於	`>`	`2>3`	false
小於	`<`	`2<3`	true

名稱	運算子	範例	範例運算結果
大於等於	`>=`	`2>=3`	false
小於等於	`<=`	`2<=3`	true

邏輯(logical)運算子

前面提到的關係運算子，可以讓我們判斷一個條件是否成立，例如：判斷成績是否及格。

```
score>=60
```

但有時狀況比較複雜一點，例如：要產生補考名單時，我們要確認成績有沒有落在這個區間 $40\leq score<69$。也就是有2個條件要同時成立。

這時候我們可以用 **And 邏輯運算子** `&&`。

```
score>=40 && score<60
```

當 `score>=40` 為 true 而且 `score<60` 為 true，則整條式子的運算結果為 true。

Or 邏輯運算子 `||` 則是只要其中一個運算是 true，則整條式子的運算結果就是 true。例如：只要國文或英文成績大於等於80分，就發給獎學金。

```
chinese>=80 || english>=80
```

Not 邏輯運算子 `!` 可以把邏輯狀態反轉，就是把 true 變成 false，把 false 變成 true。

例如：fileIsOpen 若為 true，表示檔案已開啟。程式在讀取檔案內容前，要確定檔案已開啟。我們想在檔案未開啟時顯示錯誤訊息，只要在這個狀態為 true 時即可。

```
!fileIsOpen
```

因為當 fileIsOpen 為 false 時，把它用 Not 反轉後就是 true。

位元(bitwise)運算子

`&`, `|`, `^`, `~`, `>>`, `<<` 這幾個運算子可以在位元層級做運算，也就是對每一個 bit 做運算。我們會在之後對二進位系統有一定了解後再來討論。

其他的運算子

C++ 還有許多運算子，可以參考這張網上的 [運算子和優先順序圖表]。

Revision #28
★Created 3 May 2023 00:01:56 by huihui
Updated 4 March 2024 12:08:42 by huihui