5. void \*\* (\*d) (int &, char \*\*(\*)(char \*, char \*\*))....如何看懂複雜的宣告…

在這裡，我們介紹兩種方式來看懂複雜的宣告。第一種要判斷的是：常數與指標混合使用時，到底const修飾的是指標還是指標所指的變數? 第二種是面對如標題所示這種複雜的宣告時，我們要怎麼讀懂它。  
  
5.1 常數與指標的讀法

const double \*ptr;  
double \*const ptr;  
double const\* ptr;  
const double \*const ptr;

以上幾個宣告，到底const修飾的對象是指標，還是指標所指向的變數呢?  
其實，關鍵在於：\*與const的前後關係!  
當\*在const之前，則是常數指標，反之則為常數變數。因此，

const double \*ptr; // ptr指向常數變數  
double \*const ptr; // ptr是常數指標  
double const\* ptr; // ptr指向常數變數  
const double \*const ptr; // 指向常數變數的常數指標

事實上，在The C++ Programming Language中有提到一個簡單的要訣：由右向左讀!!讓我們用這個要訣再來試一次。

const double \*ptr; // ptr is a pointer points to double, which is a constant  
double \*const ptr; // ptr is a constant pointer points to double  
double const\* ptr; // ptr is a pointer points to constant double  
const double \*const ptr; // ptr is a constant pointer points to double, which is a constant

結果完全相同 :-)  
  
5.2 複雜宣告的讀法 void \*\* (\*d) (int &, char \*\*(\*)(char \*, char \*\*)).......  
其實閱讀C/C++中複雜的宣告有點像是讀英文的長句子，看多了，自然知道句子是怎麼構造出來的。  
但對於句子還不熟的人，難免得藉助文法來拆解一個句子。關於C語言複雜宣告的解析文法，最令我印象深刻的，莫過於印度工程師Vikram的"[The right-left rule](http://www.codeproject.com/cpp/complex_declarations.asp#right_left_rule)"。他是這麼說的：  
「從最內層的括號讀起，變數名稱，然後往右，遇到括號就往左。當括號內的東西都解讀完畢了，就跳出括號繼續未完成的部份，重覆上面的步驟直到解讀完畢。」  
舉個例子：void \*\* (\*d) (int &amp;, char\*)依下面方式解讀：  
1. 最內層括號的讀起，變數名稱: d  
2. 往右直到碰到) : (空白)  
3. 往左直到碰到( :是一個函數指標  
4. 跳出括號，往右，碰到(int &, char\*): 此函式接受兩個參數：第一個參數是reference to integer，第二個參數是character pointer。  
5. 往左遇上void \*\*: 此函式回傳的型態為pointer to pointer to void。  
==> d是一個函式指標，指向的函式接受int&和char\*兩個參數並回傳void\*\*的型態。  
如何，是不是好懂很多了呢?  
  
標題中的void \*\* (\*d) (int &, char \*\*(\*)(char \*, char \*\*))其實和上面的例子幾乎一樣，只是函式的第二個參數又是一個函式指標，接受char\*和char\*\*兩個參數並回傳char\*\*的型態。