程式開發準則－史前地表最強編譯器

我們使用駝峰命名法，重點為能夠一眼就看出其中的含意，以及變數名稱的種類。

類別以英文大寫字母開頭，若有多個英文單字組成，採取大寫駝峰型 (upper camel case)

例如:

SimpleGame

MyStory

ThreadTestDrive

RunThreads

函數或變數（包括參數、屬性等）以英文小寫字母開頭，

若有多個英文單字組成，採取小寫駝峰型 (lower camel case)

例如:

actionPerformed

setUpNetworking

getField

myField

注意：Define為全大寫

1、空行

規則一：定義變數後要空行。儘可能在定義變數的同時初始化該變數，即遵循就近原則。如果變數的引用和定義相隔比較遠，那麼變數的初始化就很容易被忘記。若引用了未被初始化的變數，就會導致程式出錯。

規則二：每個函式定義結束之後都要加空行。

總規則：兩個相對獨立的程式塊、變數說明之後必須要加空行。比如上面幾行程式碼完成的是一個功能，下面幾行程式碼完成的是另一個功能，那麼它們中間就要加空行。這樣看起來更清晰。

2、空格

規則一：關鍵字之後要留空格。像 const、case 等關鍵字之後至少要留一個空格，否則無法辨析關鍵字。像 if、for、while 等關鍵字之後應留一個空格再跟左括號(，以突出關鍵字。

規則二：函式名之後不要留空格，應緊跟左括號(，以與關鍵字區別。

規則三：(向後緊跟；)、,、;這三個向前緊跟；緊跟處不留空格。

規則四：,之後要留空格。如果;不是一行的結束符號，其後要留空格。

規則五：賦值運算子、關係運算符、算術運算子、邏輯運算子、位運算子，如 =、==、！=、+=、－=、\*=、/=、%=、>>=、<<=、&=、^=、|=、>、<=、>、>=、+、－、\*、/、%、&、|、&&、||、<<、>>、^ 等雙目運算子的前後應當加空格。

注意，運算子“%”是求餘運算子，與 printf 中 %d 的“%”不同，所以 %d 中的“%”前後不用加空格。

規則六：單目運算子 !、~、++、－－、－、\*、& 等前後不加空格。

注意：

這裡的“－”和規則五里面的“－”不同。這裡的“－”是負號運算子，規則五里面的“－”是減法運算子。

這裡的“\*”和規則五里面的“\*”也不同。這裡的“\*”是指標運算子，規則五里面的“\*”是乘法運算子。

這裡的“&”和規則五里面的“&”也不同。這裡的“&”是取地址運算子，規則五里面的“&”是按位與運算子。

規則七：像陣列符號[]、結構體成員運算子.、指向結構體成員運算子->，這類操作符前後不加空格。

規則八：對於表示式比較長的 for 語句和 if 語句，為了緊湊起見，可以適當地去掉一些空格。但 for 和 if 後面緊跟的空格不可以刪，其後面的語句可以根據語句的長度適當地去掉一些空格。例如：

for (i=0; i<10; i++)

for 和分號後面保留空格就可以了，=和<前後的空格可去掉。

3、成對書寫

成對的符號一定要成對書寫，如 ()、{}。不要寫完左括號然後寫內容最後再補右括號，這樣很容易漏掉右括號，尤其是寫巢狀程式的時候。

4、縮排

縮排是通過鍵盤上的 Tab 鍵實現的，縮排可以使程式更有層次感。原則是：如果地位相等，則不需要縮排；如果屬於某一個程式碼的內部程式碼就需要縮排。

5、對齊

對齊主要是針對大括號{}說的：

規則一：{和}分別都要獨佔一行。互為一對的{和}要位於同一列，並且與引用它們的語句左對齊。

規則二：{}之內的程式碼要向內縮排一個 Tab，且同一地位的要左對齊，地位不同的繼續縮排。

例如：

#**include** <stdio.h>

**int** **main**(**void**)

{

**if** (…)

**return** 0;

}

6、程式碼行

規則一：一行程式碼只做一件事情，如只定義一個變數，或只寫一條語句。這樣的程式碼容易閱讀，並且便於寫註釋。

規則二：if、else、for、while、do 等語句自佔一行，執行語句不得緊跟其後。此外，非常重要的一點是，不論執行語句有多少行，就算只有一行也要加{}，並且遵循對齊的原則，這樣可以防止書寫失誤。

7、註釋

C語言中一行註釋一般採用//…，多行註釋必須採用/\*…\*/。註釋通常用於重要的程式碼行或段落提示。在一般情況下，程式有效註釋量必須在 20% 以上。雖然註釋有助於理解程式碼，但注意不可過多地使用註釋。

規則一：註釋是對程式碼的“提示”，而不是文件。程式中的註釋不可喧賓奪主，註釋太多會讓人眼花繚亂。

規則二：如果程式碼本來就是清楚的，則不必加註釋。例如：

i++; //i加1

這個就是多餘的註釋。

規則三：邊寫程式碼邊註釋，修改程式碼的同時要修改相應的註釋，以保證註釋與程式碼的一致性，不再有用的註釋要刪除。

規則四：當代碼比較長，特別是有多重巢狀的時候，應當在段落的結束處加註釋，這樣便於閱讀。

規則五：每一條巨集定義的右邊必須要有註釋，說明其作用。

範例：

# **include** <stdio.h>

# **include** <math.h> /\*因為要用到求平方函式sqrt()，所以要包含標頭檔案 math.h\*/

**int** **main**(**void**)

{

*//把三個係數儲存到計算機中*

**int** a = 1; *// “=”不表示相等，而是表示賦值*

**int** b = 2;

**int** c = 1;

**double** delta; *//delta存放的是b\*b - 4\*a\*c的值*

**double** x1, x2; *//分別用於存放一元二次方程的兩個解*

delta = b\*b - 4\*a\*c;

**if** (delta > 0)

{

x1 = (-b + sqrt(delta)) / (2\*a);

x2 = (-b - sqrt(delta)) / (2\*a);

printf("該一元二次方程有兩個解，x1 = %f, x2 = %f\n", x1, x2);

}

**else** **if** (0 == delta)

{

x1 = (-b) / (2\*a);

x2 = x1; *//左邊值賦給右邊*

printf("該一元二次方程有一個唯一解，x1 = x2 = %f\n", x1);

}

**else**

{

printf("無解\n");

}

**return** 0;

}