Coding style

1. 空白的使用

- 有些地方該用空白，但是有些地方不該用空白！以下舉一些我覺得好的程式實例來說明:

- a = 3 + 4 / 5;　　// 在運算符號，還有每個運算元 (變數，數字) 旁邊用一個空白隔開
- resp == YES; abc != 3;　// 邏輯判斷符號 (等於，不等於，或) 也用一個空白隔開
- while (i == 1); do { ... } while (i == 1); // 控制結構後面的條件判斷式、大括弧，也用一個空白隔開
- int swap(char *a, char *b) // 函式傳回值宣告之後的一個空白，還有參數列表中逗號後的空白，以及參數型態宣告後的空白。

2. 大括弧、內縮的使用 (段落編排)

- 首先，每層內縮都應該縮進 4 個空白。你可以修改編輯器的設定，讓 TAB 鍵插入 4 個空白。
- 接著，程式碼應該以這樣的方式使用括號與內縮:

```
int main()
{
    ....
    if (something == 1)
    {
        ....
    }
    else
    {
        ....
    }
}
```

- 注意到了嗎？左大括號跟著上一行，而右大括號自成一行。而且就算大括號括起來的範圍只有一行，也不寫成這個樣子:

```
if (something == 1) return;
if (something == 1)
    return;
```

- 這兩種寫法雖然省空間，但是第一種寫法在之後會比較難找到後面的那個敘述 (尤其是條件判斷非常長的時候)，而 第二種寫法要加入新敘述時還必須手動補上括號。最重要的是「這兩種長得與其他段不一樣」。

- 如果 if 的判斷非常長，可以依照下面的寫法加以斷行:

```
if ((something_one && something_two) || (other_one && other_two) ||
    (elsething_one && elsething_two))
{
    ...
}
```

- 也就是把邏輯符號 (&&, ||) 留在行尾，下一行內縮進來對齊。注意大括號還是跟在後面。

3. 變數的宣告與使用

- 幾個要點，第一個是「指標的 * 應該跟在變數名稱上」

- char* a, b, c; // 錯誤！ 只有 a 會是指標 (雖然要是只宣告一個的話就沒差)
- char *a, *b, *c; // 正確寫法
- 第二個，使用有意義的英文名稱，一看名字就知道這個變數幹什麼用的。
- 區域變數用「首字小寫，第二個字大寫」的方式，全域變數或常數則用「全大寫，底線區別單字」的方式方便閱讀。

- long maxScore; // 留意 m 小寫, S 大寫, 建議用法
- long curScoreAvg; // 現在分數的平均 (current score average)，多清楚呀 (後面不要用到其他用途上 ...)
- long MAX_SCORE; // 全域變數建議用法

- long a, b, c; // 幹什麼用的？
- long temp, tempb; // 暫存什麼東西？
- long Maxscore; // 區域還是全域變數？

4. 註解

- 註解跟程式原始碼有著同等重要的地位。在一個大程式中，註解有可能佔了 1/4 至 1/3 甚至更多的原始碼空間。 理由很簡單，多留一點當初的想法、工作流程的說明，在未來重新閱讀的時候可以更快了解程式碼的作用。
- 註解其實也有自己的一套準則可以遵守。首先是行內註解:
- // 增加 X
- x += 1;
- // 預備處理下一筆資料
- x += 1;
- // 讀取輸入檔
- ...
- // 計算標準差
- ...
- // 輸出統計圖表
- ...

- 同樣是行內註解，第一個加了不如不加 - 因為根本沒有實質意義。相較之下第二個就敘述得比較完備了。
- 接下來是說明一個函式的註解，這種註解的行數通常很多，因此使用區塊註解較為合適。留意註解中所提供的資訊:
- /**
- * CalcuateAverage
- *
- * 作者: Tiberius (2002/05/29)
- *
- * 計算學生的加權各科總平均成績。
- *
- * 輸入: people_t *student - 指向 "學生成績資料" 陣列的指標
- * weight_t *weight - 指向 "各科加權表" 陣列的指標
- * 輸出: 加權平均成績
- **/
- double CalcuateAverage(people_t *student, weight_t *weight)
- {
- ....
- }

- 詳盡的程度要到「說明書」的程度 - 別人看了就會用，而且不會用錯才行。
- 這部分的重要性，甚至有人已經寫了程式，可以把原始碼中的這個部分抓出來自動做成 輔助說明檔！雖然要用這個功能，這部分要用一些特殊的寫法，但是現在可以不管這個 部分，只要「讓人可以輕鬆看懂」就行。

- 一般來說，在檔案的開頭也要加上同樣類型的註解，說明一下這個檔案裡頭提供了 哪些功能 (函式, 物件等) 以及使用上需要留意的地方。如此一來只要把原始碼內的 註解整個閱讀過一遍，就能對程式本身有通盤的了解。

- http://www2.lssh.tp.edu.tw/~hlf/class-1/lang-c/cpp-style.html
- 關於「原始碼撰寫風格」 作者: **Tiberius**