學生成績處理

丁培毅

實習目標：

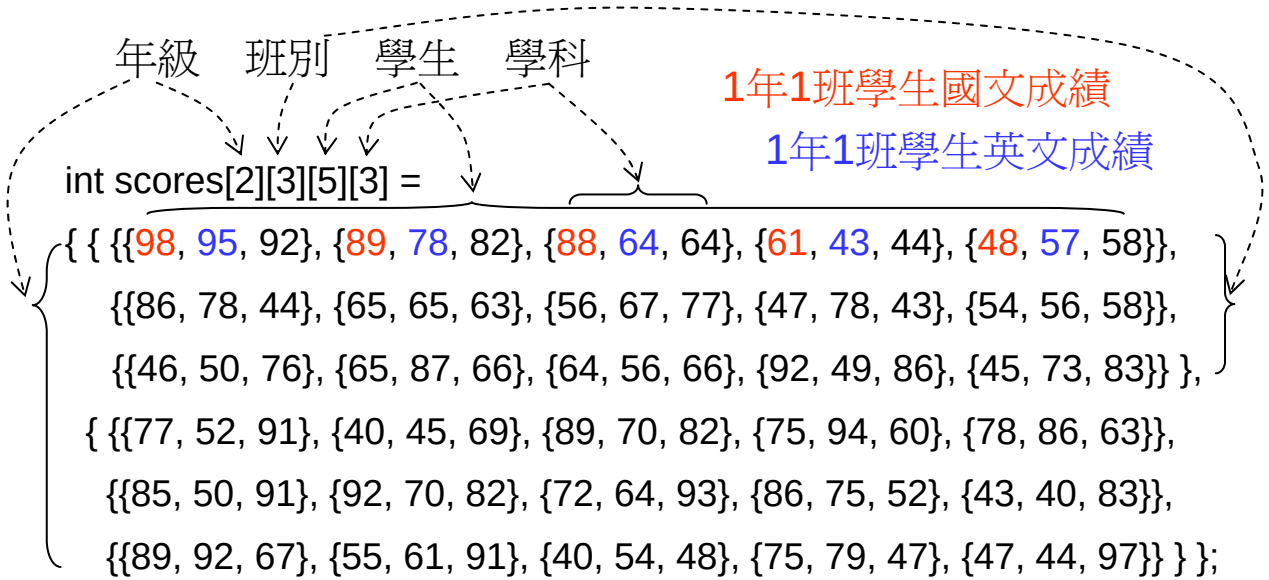
1. 練習運用多維陣列來存放資料, 配合多層迴圈來處理資料
2. 多維陣列犧牲掉用陣列名稱來區分資料的好處, 不同的資需要切換不同的註標值來存取
3. 迴圈的計數器和陣列的註標, 允許的情況下儘量使用有意義的英文名字

多維陣列的運用 – 學生成績處理

請撰寫一個程式，以多維陣列記錄兩個年級、每年級三個班、每班五位同學、每位同學有三個學科的成績，計算並且列印每個班每一學科的平均成績

程式輸出範例如右：

陣列的宣告與陣列內的資料如下：




```
int scores[2][3][5][3] =
{ { {{98, 95, 92}, {89, 78, 82}, {88, 64, 64}, {61, 43, 44}, {48, 57, 58}},
    {{86, 78, 44}, {65, 65, 63}, {56, 67, 77}, {47, 78, 43}, {54, 56, 58}},
    {{46, 50, 76}, {65, 87, 66}, {64, 56, 66}, {92, 49, 86}, {45, 73, 83}} },
  { {{77, 52, 91}, {40, 45, 69}, {89, 70, 82}, {75, 94, 60}, {78, 86, 63}},
    {{85, 50, 91}, {92, 70, 82}, {72, 64, 93}, {86, 75, 52}, {43, 40, 83}},
    {{89, 92, 67}, {55, 61, 91}, {40, 54, 48}, {75, 79, 47}, {47, 44, 97}} } };
```

```
1年1班學生國文成績平均爲 76.80
1年2班學生國文成績平均爲 61.60
1年3班學生國文成績平均爲 62.40
2年1班學生國文成績平均爲 71.80
2年2班學生國文成績平均爲 75.60
2年3班學生國文成績平均爲 61.20
1年1班學生英文成績平均爲 67.40
1年2班學生英文成績平均爲 68.80
1年3班學生英文成績平均爲 63.00
2年1班學生英文成績平均爲 69.40
2年2班學生英文成績平均爲 59.80
2年3班學生英文成績平均爲 66.00
1年1班學生數學成績平均爲 68.00
1年2班學生數學成績平均爲 57.00
1年3班學生數學成績平均爲 75.40
2年1班學生數學成績平均爲 73.00
2年2班學生數學成績平均爲 80.20
2年3班學生數學成績平均爲 70.00
```

分析

1. 在使用多維陣列設計程式時，需要清楚掌握每一個元素所存放的資料的意義，也需要非常清楚每一個註標代表的意義，例如前面範例中 scores[0][0][1][0] 是 1 年 1 班 座號 2 號同學的國文成績，scores[0][1][0][2] 是 1 年 2 班 座號 1 號同學的數學成績，…
2. 比較討厭的是 C 的陣列的註標由 0 開始，很多地方在解釋其意義時需要自己加 1 調整過來，例如上面 scores[0] 代表的是 1 年級所有班級所有同學的資料，scores[1] 代表的是 2 年級所有班級所有同學的資料，掌握資料代表的意義之後，就很容易地可以用迴圈來存取陣列裡多個元素，例如可以用迴圈計算 1 年 1 班數學成績的總分和平均分數

```
int i, sum=0;
double average;
for (i=0; i<5; i++)
    sum += scores[0][0][i][2];
average = sum / 5.0;
```

下面迴圈可以計算 2 年級所有學生的英文成績總分和平均分數

```
int iclass, istudent, sum=0;
double average;
for (iclass=0; iclass<3; iclass++)
    for (istudent=0; istudent<5; istudent++)
        sum += scores[1][iclass][istudent][1];
average = sum / (3.0*5.0);
```

請注意：註標變數如果使用有意義的名稱的話，可以讓程式比較容易看得懂，比較不會不小心寫到失智

3. 注意觀察這個程式要求的輸出，有幾件事需要看出來：

a. 印出的每一個成績是同班 5 位同學某一科目的平均值

b. 印出這麼多列，需要使用多層迴圈，最外層應該改變的是國文、英文、數學，第二層應該改變的是年級，第三層應該改變的是同一年級的各個班

c. 大部分列印的中文字都一樣，只有課程名稱需要改變

```
1年1班學生國文成績平均爲 76.80
1年2班學生國文成績平均爲 61.60
1年3班學生國文成績平均爲 62.40
2年1班學生國文成績平均爲 71.80
2年2班學生國文成績平均爲 75.60
2年3班學生國文成績平均爲 61.20
1年1班學生英文成績平均爲 67.40
1年2班學生英文成績平均爲 68.80
1年3班學生英文成績平均爲 63.00
2年1班學生英文成績平均爲 69.40
2年2班學生英文成績平均爲 59.80
2年3班學生英文成績平均爲 66.00
1年1班學生數學成績平均爲 68.00
1年2班學生數學成績平均爲 57.00
1年3班學生數學成績平均爲 75.40
2年1班學生數學成績平均爲 73.00
2年2班學生數學成績平均爲 80.20
2年3班學生數學成績平均爲 70.00
```

```
int iyear, iclass, istudent, icourse;
for (icourse=0; icourse<3; icourse++)
   for (iyear=0; iyear<2; iyear++)
      for (iclass=0; iclass<2; iclass++)  {
         for (istudent=0; istudent<5; istudent++)
            計算 5 個學生的成績總和
         計算平均值
         printf("%d年%d班學生%s成績平均爲 %.2f\n", iyear, iclass, 課程名稱, 平均值);
      }
```

4. 設計程式時什麼情況下會需要使用多維陣列?

在設計星座查詢程式時，我們有兩個陣列，一個是起始日期 firstDays[] 陣列，一個是星座名稱 zodiacNames[] 陣列，在兩個陣列中位置相同的是同一組的資料，如果這些資料的型態一樣，同時又希望用迴圈來處理的話，就可以用多維陣列。

以成績的資料爲例，如果我們把前面的 scores[2][3][5][3] 分成兩個陣列，firstYearScores[3][5][3] 和 secondYearSocres [3][5][3]，變成兩個陣列變數，如果希望寫迴圈把計算所有年級的資料時，就發現兩個名稱很麻煩，程式需要重複兩次，甚至如果把 firstYearScores[3][5][3] 再分爲不同科目：firstYearMandarinScores[3][5], firstYearEnglishScores[3][5], firstYearMathScores[3][5], 雖然變數名字使得裡面存放的資料變得比較清楚了，但是卻沒有辦法用迴圈來一次處理所有科目的成績了。

5. 特別注意：運用陣列設計程式時，一定要完整掌握陣列元素的註標的計算，絕對不要使用到程式沒有定義的元素，例如 int x[10]; 這個陣列就只能用 x[0], x[1], …, x[9] 這些元素，絕對不可以使用 **x[-1], x[-2], … 或是 x[10], x[11], …** 從前面的設計裡，你也留意到當元素的註標是計算出來的時後，例如 (month-2+12)%12，其實很容易一不小心就超出允許的範圍了，如果你在前面的練習裡沒有特別注意，也有可能已經使用到不該使用的東西了。編譯器沒有辦法幫你檢查出這樣的問題，甚至執行的結果也不一定會出錯，那麼爲什麼希望你特別留意呢？因爲如果你的程式裡有這種東西，你寫的程式就變成一個不定時炸彈，以後在和別人的程式整合時，就是無法維持穩定的正確性，當你的程式有這樣的狀況時，它的破壞是確定造成的，只是顯示出來的現象有的時候你會觀察到，有的時候不會而已 (通常 x[-1], x[-2], …, x[10], x[11], … 和其它的變數是重疊的，使用相同的記憶體，也許就是你程式裡的 i, j, k, month, day…，造成的現象就是你也許發現程式明明沒有修改變數 day，但是存放在裡面的數值突然改變了) 。

6. 爲了避免前頁談到的執行錯誤，爲什麼不增加一些程式碼讓 CPU 幫我們自動檢查呢? 如下圖：

```
int i, month, day, days[12]={…}, zodiacNames[12]={…};
    … // 以某種方法計算出 month 的數值
if ((month-1<0)||(month-1>=12)) {
    printf("Error using array days[%d] zodiacNames[%d]\n", month-1, month-1);
    exit(1);
}
if (day>=days[month-1])
    printf("%s\n", zodiacNames[month-1]);
else
    printf("%s\n", zodiacNames[(month-2+12)%12]);
```

實際上除非 month 是使用者的輸入，scanf("%d", &month); 否則我們很少看到這樣子的程式，因爲檢查是需要花費 CPU 時間的，如果99.999%都是對的，那麼檢查所花費的時間就變成不必要的負擔，比較常看到的是

```
#include <assert.h>
int i, month, day, days[12]={…}, zodiacNames[12]={…};
    … // 以某種方法計算出 month 的數值
assert((month-1>=0)&&(month-1<12));
if (day>=days[month-1])
    printf("%s\n", zodiacNames[month-1]);
```

assert() 敘述是程式的開發人員使用的，在交給使用者之前會由執行程式中完全移除，所以不會影響程式的執行效率，但是在多人合作時，可以增進界面之間資料傳遞的正確性