星座查詢

丁培毅

實習目標：

1. 練習設計陣列變數, 字串指標陣列變數以及初始化
2. 練習運用範例來熟悉演算法的運作、設計演算法
3. 練習條件敘述的撰寫
4. 特別注意避免陣列的 overflow 以及 underflow

星座查詢

請撰寫一個程式，
要求使用者輸入其生日 月/日，判斷其星座
程式輸出範例：

```
請輸入日期 (以 月/日 的格式)：5/3
你是金牛座的，星座代碼是 1。
```

Zodiac Code	星座名	日期範圍
0	牡羊座	03/21 – 04/20
1	金牛座	04/21 – 05/20
2	雙子座	05/21 – 06/21
3	巨蟹座	06/22 – 07/22
4	獅子座	07/23 – 08/22
5	處女座	08/23 – 09/22
6	天秤座	09/23 – 10/22
7	天蠍座	10/23 – 11/21
8	射手座	11/22 – 12/21
9	摩羯座	12/22 – 01/19
10	水瓶座	01/20 – 02/19
11	雙魚座	02/20 – 03/20

分析

1. 這個題目基本上屬於資料轉換的性質，由生日的月份和日期轉換成星座以及星座代碼
2. 我們可以先研究幾個範例，藉此瞭解轉換的規則，例如：
   5月3日，在 4/21-5/20 的區間中，所以是金牛座，代碼是 1
   7月25日，在 7/23-8/22 的區間中，所以是獅子座，代碼是 4
   由於所有的日期區間是不重疊的，如果把每一個區間的起始或是結束的日期記錄下來 (由小到大排列的話可以用類似下面的迴圈找到)，任何一天的生日就可以藉由比對 (>=區間的啓始日期) 判斷是哪一個區間，再把對應陣列裡面同位置的星座名稱印出，例如

months[]/days[] 區間起始日期	4/21	5/21	6/22	7/23	
zodiacNames[]	金牛座	雙子座	巨蟹座	獅子座	

```
int i, month, day, days[12]={…}, zodiacNames[12]={…};
scanf("%d%d", &month, &day);
for (i=0; i<12; i++)
    if (month==months[i]) {
        if (day>=days[i])
            printf("%s\n", zodiacNames[i]);
        else
            printf("%s\n", zodiacNames[(i-1+12)%12]);
        break;
    }
```

不要寫成 i+11

3. 很多時候，表格的設計會直接影響演算法的設計，如果我們有如下的資料表格，因爲前面的 months[] 陣列裡的資料是完全規律化的，程式中直接計算即可，可以省略 months[] 陣列以及比對的迴圈

months[]	1	2	3	4	5	6	7	8	9	10	11	12
firstDays[]	20	20	21	21	21	22	23	23	23	23	22	22
zodiacNames[]	水瓶座	雙魚座	牡羊座	金牛座	雙子座	巨蟹座	獅子座	處女座	天秤座	天蠍座	射手座	摩羯座

輸入生日： 月 month 日 day
如果 day >= firstDays[month-1] 則 zodiacNames[month-1] 即爲其星座，
　　zodiacCode = (10 + month-1) % 12;
如果 day < firstDays[month-1] 則 zodiacNames[(month-2+12)%12] 即爲其星座　zodiacCode = (10 + month-2) % 12;

注意 1. 在這個設計中不需要使用迴圈，速度比前一個設計快很多
2. (month-2+12)%12 中先加 12 的原因：當 month-2>=0 時對結果沒有影響，但是當 month 爲 1 時 -1%12 運算結果爲 -1，但是 (-1+12)%12 爲 11 才是我們要的陣列元素位置

4. **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記憶體了。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) 。