撲克牌型判斷

丁培毅

實習目標：

1. 練習比較複雜的條件判斷敘述, 練習以流程圖來輔助設計, 對於複雜的邏輯來說流程圖是很好的說明文件
2. 練習以陣列及迴圈來統計各種出現頻率的特性
3. 統計資料的表達方法改變時, 演算法也有機會用比較靈活的方式設計，不要被資料變數的表達方法限制死了

撲克牌型判斷

1. 請撰寫一個程式, 重複下列十次 “將 52 張撲克牌洗好 (隨機打亂), 印出前 5 張牌以及其牌型”, 要求使用者輸入 5 張牌的編號 (0~51), 印出這 5 張牌以及其牌型
2. 程式輸出範例如右：

```
♣6 ♣K ♣A ♥4 ♥5 無
♣5 ♣8 ♦3 ♦8 ♥3 兩對
♣J ♦J ♥J ♠4 ♠J 四條
♥8 ♥9 ♥10 ♥J ♥Q 同花順
♣3 ♣8 ♥6 ♥J ♠2 無
♣8 ♣10 ♦5 ♥10 ♠J 一對
♣4 ♦A ♥2 ♥7 ♠A 一對
♦4 ♦5 ♦8 ♦10 ♦Q 同花
♣4 ♦5 ♥3 ♥7 ♠6 順
♦2 ♦10 ♥2 ♠2 ♠10 葫蘆

請輸入 5 張牌：0 1 2 3 12
♣2 ♣3 ♣4 ♣5 ♣A 同花順
```

編號 code	花色 suit	UTF-8	數字 rank, value, pips
0~12	梅花 clover/club ♣	e2 99 a3	2 3 4 5 6 7 8 9 10 J Q K A
13~25	方塊 diamond/tile ♦	e2 99 a6	2 3 4 5 6 7 8 9 10 J Q K A
26~38	紅心 heart ♥	e2 99 a5	2 3 4 5 6 7 8 9 10 J Q K A
39~51	黑桃 spade/pike ♠	e2 99 a0	2 3 4 5 6 7 8 9 10 J Q K A

程式中撲克牌請以 0~51來編碼代表其花色與點數, 螢幕列印請用命令列視窗 cmd, codepage 65001 (chcp 65001, system("chcp 65001");) VC2010 請加上下列 #pragma execution_character_set("utf-8"), 並將原始檔以UTF-8 存檔

Ranking 順序	Categories 牌型	定義	判斷規則
8	Straight Flush 同花順	連續相同花色的五張牌 例如 ♣A ♣2 ♣3 ♣4 ♣5 或 ♥10 ♥J ♥Q ♥K ♥A	五張花色相同且點數連續
7	Four of a Kind 四條	四張相同點數的牌以及任意其它一張 例如 ♣9 ♥9 ♦9 ♠9 ♣J	累計每一點數的張數, 如果有某一點數出現四張
6	Full House 葫蘆	三張某一點數的牌以及二張另一點數的牌 例如 ♣3 ♥3 ♦3 ♥6 ♦6	某一點數三張且另一點數兩張
5	Flush 同花	五張相同花色但是點數沒有全部連續的牌 例如 ♦4 ♦6 ♦7 ♦10 ♦Q	累計四種花色個別的張數, 如果某一花色有五張
4	Straight 順	連續的五張牌但是有兩種以上花色 例如 ♥8 ♥9 ♦10 ♠J ♦Q	檢查每一點數的張數, 是否都是一張而且五張點數連續
3	Three of a Kind 三條	三張某一點數的牌以及二張不搭的牌 例如 ♠2 ♣6 ♦6 ♣6 ♠K	某一點數三張且另外兩種點數爲一張
2	Two Pairs 兩對	兩對相同點數的牌以及其它一張不搭的牌 例如 ♣4 ♦4 ♠9 ♦J ♠J	兩種點數都是兩張
1	One Pair 一對	一對相同點數的牌以及其它三張不搭的牌 例如 ♣4 ♦4 ♠5 ♦10 ♠K	某一點數兩張, 其它三種點數一張
0	High Card 無	沒有出現以上各種牌型的五張牌 例如 ♣3 ♦7 ♠8 ♦J ♠K	以上都沒有出現

分析

1. 這個程式除了先前的洗牌以及列印牌的花色和點數之外, 需要根據 5 張牌的花色和點數來決定是 8 種牌型裡哪一種
2. 洗完牌之後, 5 張牌放在一個整數陣列 hand[] 裡, 每一個元素是牌的編號 0~51, 花色是 hand[i]/13, 點數是 hand[i]%13+2, A 是 14 也是 1, 仔細看各種牌型的判斷規則:
   a. 需要檢查花色出現幾種, 如果某一種花色出現 5 次, 不是同花就是同花順, 其它種類花色並不影響牌型的判斷, 這個判斷可以用一個迴圈完成, 檢查第 2, 3, 4, 5 張牌的花色是不是和第一張一樣, 結果需要記錄在一個變數裡, 例如整數變數 isFlush, 1 代表花色通通相同, 0 代表不完全相同
   b. 另外一種檢查是不是同樣花色的方法: 可以設計一個整數陣列變數 suitCount[4], 每一個元素代表一種花色出現的次數, 設計一個迴圈跑 5 次, 因爲 hand[i]/13 會是 0~3, 代表花色, 所以直接運用 suitCount[hand[i]/13]++ 這樣的敘述可以統計出來各種花色的出現次數, 再另外設計一個跑 4 次的迴圈,

2. 計算 suitCount[] 陣列中最大值, 如果是 5 就是同花, 把 isFlush 變數設爲 1 記錄下這個狀況, 否則設爲 0

c. 接下來需要檢視這 5 張牌的點數, 這個時候不需要管花色, 例如某一點數出現 4 張, 某一點數出現 3 張, 某一點數出現 2 張, 或是 5 張牌的點數連續, 這些狀況都會影響牌型的決定, 需用程式來分辨或是篩選這些狀況: 原本陣列 hand[] 裡面記錄的是 0~51, 如果改記錄爲點數 hand[i]%13+2, 由這 5 個點數資料雖然可以直接判斷上面各種情況是否成立, 例如判斷是否有 4 張牌點數相同, 可以運用兩層迴圈來做

```
for (i=0, isFourOfAKind=0; i<5&&!isFourOfAKind; i++) {
    for (j=0, count=0; j<5; j++)
        if (point[i]==point[j]) count++;
    if (count == 4) isFourOfAKind = 1;
}
```

其它幾個判斷也不見得比較簡單, 可以思考嘗試改變一下資料的表示方法, 程式有機會可以變成比較簡化、比較直覺

2. d. 定義一個陣列來統計各種點數出現的次數, 有十三種點數所以可以定義一個 13 個元素的陣列 int rank[13] = {0}; 寫一個迴圈統計各種點數 (不管花色) 出現的次數

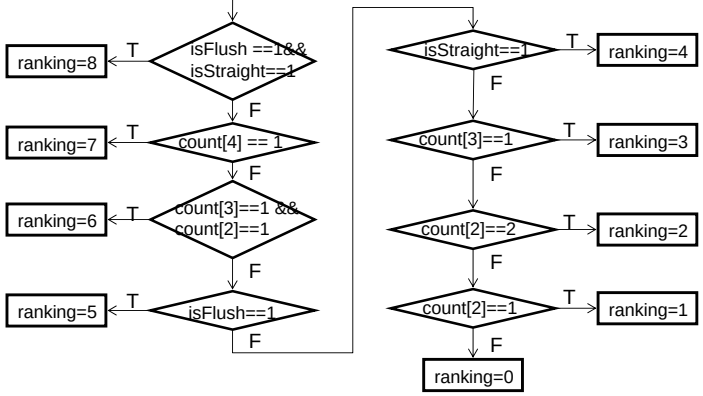
```
for (i=0; i<5; i++)
    rank[hand[i]%13]++;
```

所以 rank[0] 中記錄的是點數 2 的牌的張數, rank[12] 記錄的是點數 A 的牌的張數。有了這個統計各種點數出現次數的資料陣列以後, 想要檢查是否有兩種點數各自有兩張牌, 就可以運用下列迴圈來累計有 0~4 張牌的點數有幾種, 然後可以據以判斷部份牌型

```
int count[5]={0};
for (i=0; i<13; i++)
    count[rank[i]]++;
if (count[4]==1) ranking = 7; // 四條
if (count[2]==2) ranking = 2; // 兩對
```

想要檢查是不是有五張連續點數, 可以嘗試用兩層迴圈來做,可以讓外層迴圈變數代表起始的點數, 內層迴圈變數則代表連續的五個點數, 如果都是 1 張的話, 就是一個順子, 設定 isStraight=1。不過這個時候有一個比較特殊的狀況: “Ace 這張牌可以當作 1 點也可以當作 14 點”, 如果我們調整一下陣列 rank[], 讓它有 14 個元素, 中記錄點數 A 的牌的張數, rank[1] 中記錄的是點數 2 的牌的張數, …, rank[12] 中記錄的是點數 K 的牌的張數, rank[13] 中再一次記錄點數 A 的牌的張數, 有了這樣的設計以後, 判斷順子就不會漏掉以 A 爲起始的了

2. e. 接下來應該要仔細思考如何判斷這 8 種牌型, 因爲要判斷的特徵有多種, 所以直接用 if/else 敘述寫起來會比較複雜, 這時候建議可以用流程圖來輔助設計, 例如:



流程圖在結構化的程式設計裡面用途相當有限, 輔助設計比較複雜的流程判斷是一個不錯的用途, 用圖形化的方式表示時大部份人比較容易掌握全盤的邏輯, 比較不容易混亂, 這張圖是一個很好的說明文件, 比程式碼本身表達得清楚