?A?B 猜數字遊戲

丁培毅

實習目標：

1. 函式設計練習
2. 不同的資料表示方法對應不同的處理演算法
3. 練習設計隨機挑選不重複數字的演算法
4. 運用迴圈設計檢查 ?A 的演算法、?B 的演算法
5. 練習兩層迴圈的設計

?A?B 猜數字遊戲

請撰寫一個程式，隨機產生四個不重複的數字作爲謎底，由使用者猜測四個不重複的數字，程式比對謎底並且輸出 mAnB 代表有 m 個數字與謎底完全相同，n 個數字有出現在謎底，但是位置不對，使用者重複猜測直到完全正確，也就是 4A0B 爲止

程式輸出範例如下：

```
請猜一個 4 位數不重覆的數字：2973
2A1B
請猜一個 4 位數不重覆的數字：3652
1A1B
請猜一個 4 位數不重覆的數字：5972
4A0B
猜對了！
```

分析

1. 由程式的輸出入和功能描述可以想像程式應該先產生謎底，然後進入一個迴圈，迴圈內重複 “輸出提示文字、讀取輸入、判斷並輸出 mAnB”，直到 4A0B 或是指定次數爲止
2. 在這個練習裡，希望大家練習運用函式建構抽象化的單元，將程式的細部步驟適當地包裝起來，在同一個函式裡 (或是任意的一小段程式中) 所處理的工作應該都是同一個細節層次的，就好像董事長和總經理談話時，不會一邊談設廠資金、政府法規，一邊夾雜電路設計的細節或是員工吃飯的時間這樣的東西，這樣子混雜處理事情是不會有效率的，通常每一個層次的工作有適當的知識、資源、與負責事項，公司內有上、中、下層的人員各自負責不同的事項，程式裡也應該是這樣規劃的，如此才能夠迅速打造各種功能，並且維護這些功能的正確性，所以像步驟 1 描述的各個事項，請嘗試以數個函式來完成
3. 這個練習裡我們也希望大家練習減少使用魔術數字，所謂的魔術數字是程式裡出現的常數，例如： 5, 20, 2.71828 等等，

3. 由於這些數字本身不容易看出它們代表的意義，容易誤用，也容易造成不一致，這會造成程式維護、擴充、和偵錯的困難。請設定一個全域的整數常數 TotalDigits，內容爲 4，代表有四個數字要猜；再設定一個全域的整數常數 TimeLimit，內容爲 10，代表使用者最多能猜的次數，猜了這麼多次還沒有完全猜對的話，使用者就輸了 (常數和變數的命名方式會有一點差別)

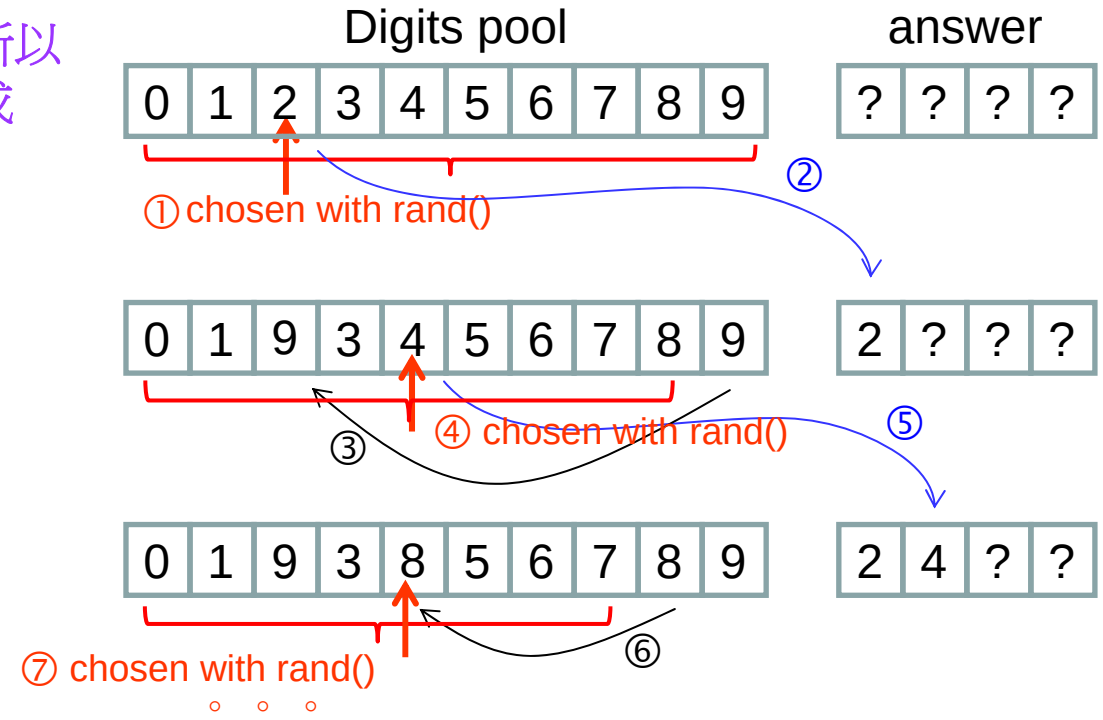
4. **產生謎底 setAnswer()**：運用 rand() 產生 TotalDigits 個不同的數字，藉由陣列參數回傳 void setAnswer(int answer[TotalDigits])

因爲每個數字不能相同所以不能用一個簡單迴圈完成

方法一：

準備 10 個球放在袋子裡一個一個抽出來

請注意，留在陣列前端可供挑選的數字一定是不重複的



4.

```
for (i=0; i<10; i++) pool[i] = i; // 在 pool 陣列中準備好 0~9 共 10 個數字
for (i=0; i<TotalDigits; i++) { // 總共需要挑 TotalDigits 個數字
    pool 陣列中前 TotalDigits-i 個元素裡面挑一個
    把挑到的那個拷貝到 answer[i] 中 // 第 i 個數字決定好了
    挑到的那個數字不要再重用了，換一個沒有用過的數字
}
```

方法二：重複下列 TotalDigits 次

“運用 rand() 挑一個 0~9 且未挑到過的數字”

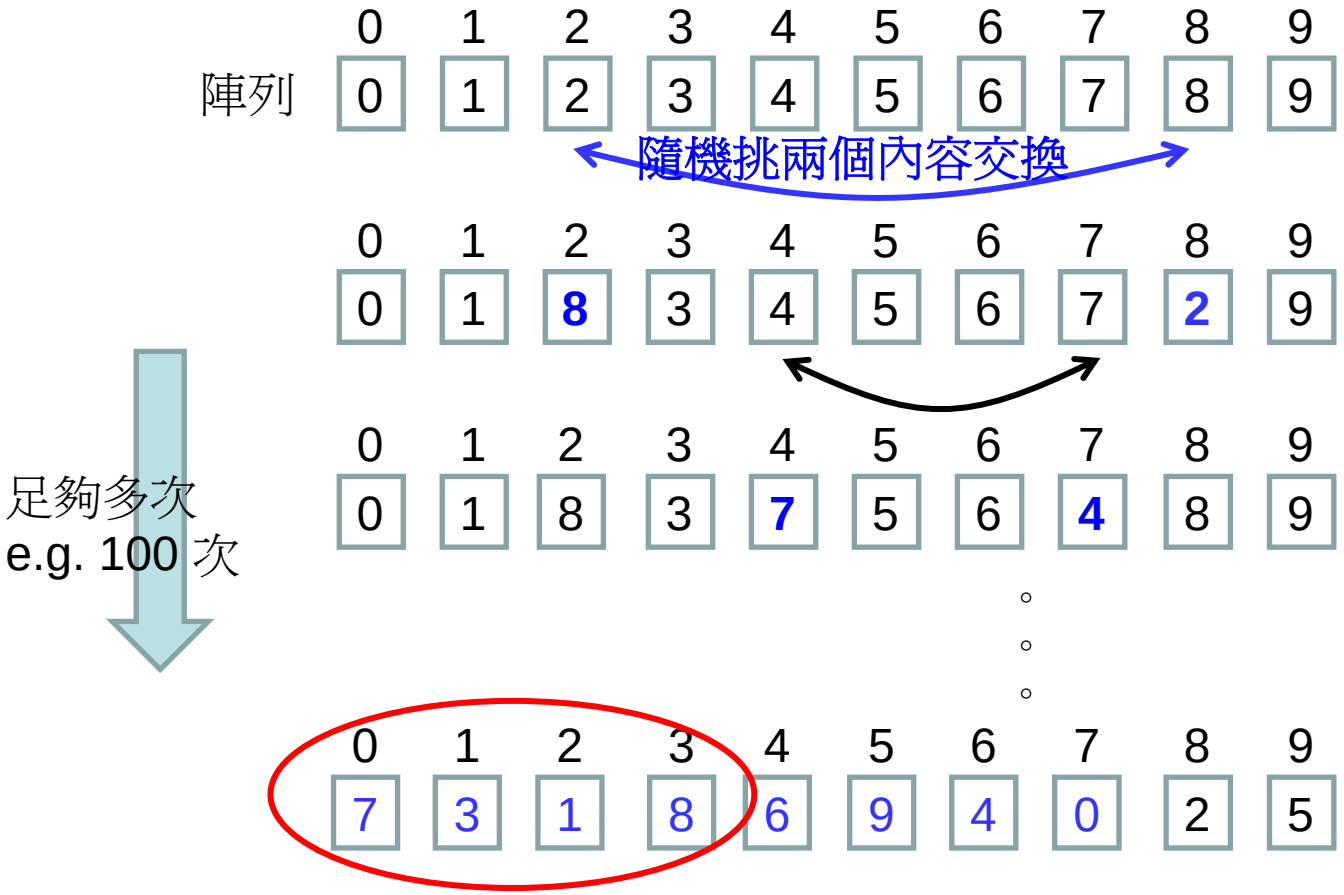
挑到的數字直接紀錄在 answer[] 陣列中，

每次挑一個數字之後可以和 answer[] 陣列裡的元素比較來判斷是 否有挑到過

```
for (i=0; i<TotalDigits; i++) {
    chosen = 運用 rand() 挑一個 0~9 的數字
    for (j=0; j<i; j++)
        if (chosen == answer[j]) {
            chosen = 運用 rand() 重新挑一個 0~9 的數字
            j = -1;
        }
    把 chosen 拷貝到 answer[i]
}
```

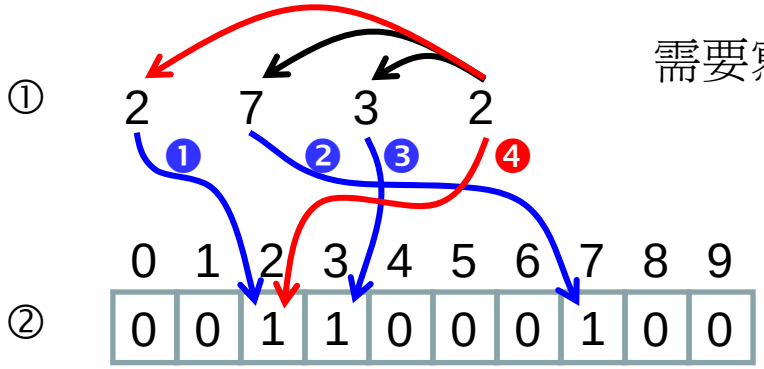
4. 方法三：

可以在一個陣列裡先放 0~9, 然後隨機挑兩個元素交換位置, 這樣子做很多次以後, 陣列裡就像是一個隨機的排列, 就像是麻將或是撲克牌洗牌一樣, 然後拿陣列前面的 TotalDigit 個元素出來使用



5. 使用者猜測數字 **guess()**：在螢幕上列印提示訊息 “請猜一個 4 位數不重覆的數字”，訊息中的 4 是 TotalDigits 的數值；由鍵盤讀入 TotalDigits 個數字，檢查是否爲不重複的 0~9 數字，將數值存放在整數陣列中，傳回呼叫端。

如下圖，檢查輸入數字是否不重複有兩種直覺的作法：



需要寫一個迴圈檢查先前的輸入

準備一個 10 個整數的陣列，每一個元素先設爲 0，每讀入一個數字時檢查對應的陣列元素是不是 0，如果是 1 就表示已經出現過了

6. 檢查猜測數字與謎底數字完全相同的有幾個 **checkA()**：以一個例子來思考作法，設計變數並且寫出變數資料的內容變化

answer[]		2	7	3	9
guess[]		4	7	9	2
countA	0	0	1	1	1
迴圈控制變數	i	0	1	2	3

寫一個 for 迴圈檢查兩個陣列對應的元素是否相同並且運用一個 countA 變數來累計相同的個數

7. 檢查猜測數字與謎底數字相同但位置不同的有幾個 checkB():

同樣地用一個範例來構思程式的細部作法，下面投影片裡不用動畫表示，而是詳細列出變數裡面資料的改變方式，希望最後寫出來的程式執行時變數裡面的資料完全和這張表格相同

answer[]		2				7				3				9			
guess[]		4				7				9				2			
countB	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	2	2
迴圈控制變數 i		0	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3
迴圈控制變數 j		~~0~~	1	2	3	0	~~1~~	2	3	0	1	~~2~~	3	0	1	2	~~3~~

每一個 answer[i] 都可能有貢獻 (和某一個不同位置的 guess[] 相同)
所以需要一個外層的迴圈，迴圈控制變數是 i

每一個 answer[i] 都需要和 guess[0], guess[1], …, 比較 (不需要和相同位置的 guess[i] 比較，上圖中打叉叉的位置)，所以在外層迴圈執行第 i 次時，需要一個內層的迴圈，迴圈控制變數是 j

基本上是 answer[i] 和 guess[j] 比較, i≠j

8. 請注意，一般在 C 程式裡傳遞一個陣列到函式裡時，比較少像前面說明裡面直接用一個全域的常數讓所有的函式直接使用

```
const int TotalDigits = 4;
…
void setAnswer(int answers[TotalDigits]) {
    …
}
```

通常會把陣列裡有多少個可以使用的資料 (甚至常常會把陣列總共有多少個可存放的單元) 一起傳遞進去

```
void setAnswer(int nDigits, int answers[]) {
    …
}
```

如此在函式裡可以看到所有使用到的變數的定義, 不像前面使用 TotalDigits 時需要往前看到底型態是什麼, 代表什麼意義。另外, 函式的參數是陣列時, 陣列的大小可以不寫, 寫了其實編譯器只是檢查是不是一個非負 (VC 要求 >0) 的整數運算式, 這個數值其實完全不會使用到, 不會檢查傳入的陣列是不是這麼大，也不會拿來檢查程式有沒有用超過允許的範圍

9. 大家都很希望程式一寫完第一次測試時，所有功能都是對的，可是實際狀況離這個目標相當遠，程式第一次測試的時候好像完全不是那麼一回事，通常都和設計時預期的表現差很多!!! 怎麼辦? 隨便找一個學長、隨便問網路上的高手，大概都會說 – 多練習，唉! 早就知道的答案

這個答案一定是對的，反正人人都可以多練習，也不是對你的差別待遇，萬一真的一直沒有辦法突破，大概就是練習不夠吧，永遠都是一個完美的解釋，我們來看看有沒有其它的看法… 以剛才的程式來說，根據前面的分析你也許會寫右圖的程式，邏輯上看起來是對的，但是它要能夠正常運作的話在 setAnswer(answer) 執行之後，陣列 answer[] 裡的數字應該要合理，你會多寫一個迴圈把資料印出來檢查嗎?? 當執行 guess(number) 以後或是 checkA(), checkB() 也是這樣，設計的過程裡加進去的會比除錯時加進去的有效果! 試看看吧，也許一開始就別想太美好才是比較實際的，還是一句老話，欲速則不達

```
setAnswer(answer);
while (不是 4A0B) {
   guess(number);
   n = checkA(number, answer);
   …
}
```

延伸練習

1. 交換電腦與使用者的角色，由使用者挑選謎底的四位數字，由電腦猜測，使用者回覆 mAnB，直到電腦猜對為止

   兩種基本方法：

   a. 消去法：每次猜測四個數字，首先把 5040 種可能性都表列出來，使用者回覆 mAnB 以後，逐一把所猜數字和 5040 種比對，刪除不符合的可能性，由剩下的可能性中猜測

   b. 推理法：假設猜的這個數字是正確答案，那麼這個數應該符合已經猜測的數及其結果。例如已經猜 1234 得到 0A0B，那麼下一步就不能猜含有 1234 中任一數字的數，因為如果正確答案包含 1234 中任一數字，出題者不會說 1234 是0A0B。再假設如果出題者說 5678 是 0A1B 的話，則正確答案必須只包含 5、6、7、8 其中一個數字，且如果有 5、6、7、8 的話，5 的位置必須不在(左邊數起)第一個位置、6 的位置必須不在(左邊數起)第二個位置、7 的位置必須不在(左邊數起)第三個位置、8 的位置必須不在(左邊數起)第四個位置。