E94116075 工科系 115 級駱穎玟 Hw3 程式設計作業

- 實現方法
  - 概念

    引入 cp、test 和 stupidproof 函式，一個用來產生位數數字不重複的隨機亂數，一個用來比對目前所猜測的數字為多少 A 多少 B，一個用來防呆。

    1. cp 函式

       大架構是使用兩個 for 迴圈

       A. 第一個 for 迴圈：

          先產生 0-9 之間的隨機亂數，用 count 紀錄已經找到的位數有多少個，當 count=n-1 時代表已經找完

       B. 第二個 for 迴圈：

          為了尋找不重複的值。用 anss[j]存放最終會 return 回去的數，每次產生新的亂數都去與 anss 中的所有值比對，若與其中一值相同則跳出迴圈重新尋找，若皆不同，表示這次產生的亂數是可用的，將 count 加 1 並把這次產生的亂數放入 anss[j]中。

       最後，將陣列轉為四位數的表示法並 return 找到的值

    2. Test 函式

       判斷到底有幾 A 幾 B，也是由兩個 for 迴圈組成

       A. 第一個 for 迴圈

          當 guess[i]==str[i]時將 A 值+1

       B. 第二個 for 迴圈

          如果沒有，再引入一 for 迴圈檢查 guess[i]與 str 的任一位數有沒有相等，若有則將 B 值+1。

       如果結果為 4A0B 則 return 0，若不是則 retrun -1，再讓主函世界有兩個不同的回傳值判斷該執行的步驟

    3. Stupidproof 函式

       詳見延伸問題。

    4. 主函式

       在無窮迴圈當中利用 switch 判斷使用者的模式選擇

       - 模式一

         將 n 值（位數值）設為 4，先呼叫 CP 產生一隨機亂數作為謎底，將轉為字串的謎底存入全域變數 str，輸入猜測的值再將此值轉為字串後存入 guess 後呼叫 test 進行比對，使用者會不斷猜測直到 test 後的結果為

4A0B

- 模式二

  將 n 值設為 4，輸入謎底並將其轉字串後存入 str，再呼叫 cp 產生電腦所猜測之一亂數，轉字串後存入 guess，再呼叫 test 進行比對，電腦會不斷猜測直到 test 後的結果為 4A0B。

- 模式三

  概念與模式一相同，只是將 n（位數值）設為 5，使用者會不斷猜測直到 test 後的結果為 5A0B。

- 模式四

  Return 0 讓遊戲結束。

➢ 結果討論

- 使用函式、陣列簡化程式碼

在過程中我們可以發現模式一三基本上除了位數不同以外所做的事情皆相同，而模式二中僅是”主客易位”的概念，我作了以下兩步驟去優化程式：

1. 我利用 guess 與 str 去存放猜測的數字與真正的答案，這兩個陣列在三種模式中皆可使用只是在模式二時猜測的答案是用電腦提供，真正的答案是由使用者提供。
2. 我建立 cp 與 test 函式進行亂數尋找與 guess/str 陣列中元素比對，建立 stupidproof 函式去檢查前三種模式的輸入。

- 字元轉換問題

模式一中，如果最後傳回來的亂數 ans1<1000
再用 sprintf 將亂數轉換為字串時中間項放”%04d”可讓不足四位數之數字前面加 0(359 變為 0359)(模式三依此類推)

- 變數型態討論

我有想到以下兩個方法，使用第一個方法

1. 考量函式只能 return 數字，我先將亂數找到後藉由 for 迴圈轉成四位數的表示法，最後將四位數字 return 回主函式後利用 sprintf 補零並轉為字串的形式
2. 考量陣列不用回傳即可儲存在函式中的變更，我儲存 0-9 的數字在

字串陣列 1 中，不重複地抽取其中的四個後將它們存至另一字串陣列 2 中，此方法不須 return 任何數。

- 延伸問題

  - 如何達到防呆的效果

    用 stupidproof 函數達到防止以下兩種情形的目標

    1. 輸入重複的數字
       用兩個 for 迴圈檢查 guess1 裡頭的數字是否重複，若沒有則 return 0，有則 return -1
    2. 輸入的數字不是此模式所要求的位數
       #include <string.h>後即可使用 strlen(字串名稱)產生字串長度

```
int stupidproof(int n,char guesspro[]){
    if(strlen(guesspro)!=n){//if the user send the number in wrong digits.
        printf("您輸入錯的位數\n");
        printf("請重新輸入\n");
        return -1;
    }
    for(int i=0;i<n;i++){
        for (int j=0;j<n;j++){
            if(i==j){
                continue;
            }
            else if(guesspro[j]==guesspro[i]){
                printf("您輸入的%d位數字中不得有重複數字\n",n);
                printf("請重新輸入\n");
                i=n;
                return -1;
            }
        }
    }
    return 0;
}
```

我有想到如果輸入的數同時為重複位數的數且為錯的位數(如在模式 3 中輸入 2344)程式第一次只會告訴我我輸了錯的位數，第二次才會告訴我輸入了重複的數字，但我覺得在這裡並沒有嚴謹的要求只能輸入幾次，所以沒有特別去更改這個部分。