隨機漫步

丁培毅

實習目標：

1. 問題簡化
2. srand(), time(), rand() 練習
3. for, while 迴圈練習
4. 二維陣列練習
5. ASCII 編碼與字元變數加減法

隨機漫步

1. 隨機漫步 (Random Walk) 是一種數學統計模型，用來表示不規則的變動形式，在金融學中用來描述證券價格的隨機波動
2. 請撰寫一個程式，在一個 10x10 的區域中由左上角 (0,0) 出發隨機漫步，每一步可以向上下左右，但是不能走出 10x10 的區域，也不能重複已經走過的路徑，最長可以走 26 步，以 A, B, C, D, …, Z 代表走過的路徑，輸出範例如下，如第三圖也有可能還沒走到第 26 步，到 S 就沒路了：

```
ABCDGHIJKL
...EF..ONM
.......P..
.......Q..
.......RS.
.......T..
.......U..
.......V..
.......WX.
.......ZY.
```

```
ABEFGHIJK.
.CD.....L.
.......M..
.......NO.
.....VURQP
...YXWTS..
...Z......
..........
..........
..........
```

```
ABEF......
.CDG......
KJIH......
LM........
SNO.......
RQP.......
..........
..........
..........
..........
```


分析

1. 這個題目需要使用 stdlib 裡提供的虛擬亂數產生器 rand() 以及初始化函式 srand()，標準使用範例如右，如果需要產生均勻分佈的 0~(n-1) 之間的整數，將這個均勻分佈在 [0,1) 的浮點數乘 n 再轉為整數 (int)((double) rand() / (RAND_MAX+1) * n) 即可，平常偷懶一點的方法是 rand()%n，均勻分佈的特性較差。在這個例子裡，我們需要在任何時候由可以走的路徑中均勻地挑選其中之一，如下圖右中有三個方向可走，程式中就讓 n 為 3，用上式隨機挑選三個方向之一，把 S 填入陣列中

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main() {
    int i;
    srand(time(0L));
    for (i=0; i<10; i++)
        printf("%f\n", (double) rand() / (RAND_MAX+1));
    return 0;
}
```

```
0.868374
0.713309
0.858882
0.864956
0.566881
0.655721
0.240303
0.127995
0.456038
0.251930
```

P	O	N
Q	R	M
.	⓪	.

P	O	.
Q	R	①
.	⓪	.

P	②	.
Q	R	①
.	⓪	.

2. 每一段程式都有一個主要的控制架構，這個程式要求在螢幕上列印 A~Z，如果像下圖是一直列的列印，程式架構就變成很簡單的一層迴圈了，有那麼容易嗎？

A	B	C	…	Y	Z

```
for (i='A'; i<='Z'; i++) printf("%c", i);
```

會複雜很多嗎? 那麼印出右圖會複雜很多嗎? 是不是準備一個字元陣列，陣列裡放右圖的內容 (先都放 '.', 然後 'A' 放在 (1,0), 'B' 放在 (2,1), …, 不難算出各個字元的座標點)，然後一列一列印出呢?

	0	1	2	3			
0	.	.	.	D	.	…	.	.
1	A	.	C	.	E		Y	.
2	.	B	.	.	.		.	Z

```
int i, row, col, cells[3][26];
int table[4]={1,2,1,0};
for (i=0;i<26;i++)
    cells[0][i]=cells[1][i]=cells[2][i]='.';
```

```
for (i=0; i<26; i++) {
    row=table[i%4]; col=i;
    cells[row][col] = 'A'+i;
}
```

3. 變數設計
   a. 需要用一個二維字元陣列 char cells[10][10] 記錄那些格子已經走過了，一開始的時候全部初始化為 '.'，代表都沒有走過
   b. 需要一個字元變數記錄目前走到第幾步，一開始初始化為 'A'
   c. 需要一對整數變數 (row,col) 記錄目前隨機漫步字母的位置座標，內容在 (0,0) ~ (9,9) 之間

4. 程式主要為一層迴圈 (for 或是 while)，迴圈每重複執行一次代表**往前走一步**，需要更改目前的座標位置、更改記錄第幾步的變數 step、記錄在二維陣列中

字元變數也是整數變數，'A' 到 'Z' 在 ASCII 編碼表中為連續的 26 個

```
for (row=col=0,cells[0][0]='A', step='B',; step<='Z'; step++) {
   ① 檢查 cells[row][col] 的四周有幾個未走過的格子可以走
   ② 如果沒有路可以走，break 出迴圈
   ③ 隨機挑選其中一個路徑, 修改 (row,col), 修改 cells[row][col] 為 step
}
```

5. 檢查 cells[row][col] 四周，基本上就是檢查 cells[row-1][col], cells[row][col-1], cells[row+1][col], cells[row][col+1] 這四格的內容是不是 '.'，需要特別注意不要使用到超過陣列邊界的元素，以 char cells[10][10]; 的定義來說，不可以使用到 cells[-1][...], cells[…][-1], cells[10][…], cells[…][10], 如此有的時候會需要增加許多 if 的檢查敘述，為了避免這種狀況，常常我們可以把陣列加大，例如用 char cells[12][12]; 程式需要使用的 10x10 區間是 (1,1)~(10,10)，四周都有額外一個元素，設定成不是 '.' 的字元，例如空格字元，如此在檢查的時候不須特別注意邊界，檢查時也不會以為是可以走出邊界的