

## 1) Project Description

### 1-1) Program Flow Chart

- A、initial：先創建 45\*45 二維陣列，並將 row\*column 弄成俄羅斯方塊的可掉落區。
- B、開檔：開啟檔案並讀取參數。
- C、while 迴圈：判斷輸入是否為 End。
- D、分類：分成 19 種俄羅斯方塊。
- E、falling：判斷方塊能否進入最上排，若能則開始往下掉，直到碰到另一個方塊或是碰到底部，並停留在該位。
- F、shift：判斷方塊能否左右移動到指定位置，若能則移到該位。
- G、falling：判斷方塊能否繼續掉落，若能則繼續掉落，直到碰到另一個方塊或碰到底部，並將方塊停留在該位。
- H、update：判斷是否有任何一列皆有方塊，若有則消除此列。
- I、cout：印出結果。
- J、關檔：關閉檔案並結束。

### 1-2) Detailed Description

```
10     int row, column;
11     int tetris[45][45];
12     int start, dir;
13     int flag = 0, flag1 = 0;
14
15     for(int a=0;a<=45;a++){
16         for(int b=0;b<=45;b++){
17             tetris[a][b] = -1;
18         }
19
20     for(int a=1;a<=row;a++){
21         for(int b=1;b<=column;b++){
22             tetris[a][b] = 0;
23         }
24     }
```

A、

令 tetris[45][45] = -1，並令 row\*column 的 element = 0。

非掉落區的 element 皆 = -1，可掉落區的 element 皆 = 0，方塊的 element 皆 = 1。

```
26     ifstream fin;
27     fin.open(argv[1]);
28     char N[20];
29     fin.getline(N,20);
30     stringstream ss(N);
31     ss>>column>>row;
```

B、

先 fin.open 開啟檔案；再用 N[20]來讀取輸入的字串，並轉換成

stringstream ss。

C、

```
35 while(!fin.eof()){
```

若讀取的.data 檔內容已結束，則離開 while 迴圈。

```
46 if(kind == "T1"){
47     for(int a=2;;a++){
48         if(tetris[start+1][a] != 0 || tetris[start][a-1] != 0 || tetris[start+1][a-1] != 0 || tetris[start+2][a-1] != 0){
49             for(int b=1;;b++){
50                 if(tetris[start+1+dir][a-1+b] != 0 || tetris[start+dir][a-2+b] != 0 || tetris[start+1+dir][a-2+b] != 0 || tetris[start+2+dir][a-2+b] != 0){
51                     tetris[start+1+dir][a-1+b-1] = 1;
52                     tetris[start+dir][a-2+b-1] = 1;
53                     tetris[start+1+dir][a-2+b-1] = 1;
54                     tetris[start+2+dir][a-2+b-1] = 1;
55                     flag1 = 1;
56                     break;
57                 }
58             }
59             if(flag1 == 1){
60                 flag1 = 0;
61                 break;
62             }
63         }
64     }
65 }
66 else if(kind == "T2"){
67     for(int a=3;;a++){
68         if(tetris[start+1][a] != 0 || tetris[start][a-1] != 0 || tetris[start+1][a-1] != 0 || tetris[start+1][a-2] != 0){
69             for(int b=1;;b++){
70                 if(tetris[start+1+dir][a-1+b] != 0 || tetris[start+dir][a-2+b] != 0 || tetris[start+1+dir][a-2+b] != 0 || tetris[start+1+dir][a-3+b] != 0){
71                     tetris[start+1+dir][a-1+b-1] = 1;
72                     tetris[start+dir][a-2+b-1] = 1;
73                     tetris[start+1+dir][a-2+b-1] = 1;
74                     tetris[start+1+dir][a-3+b-1] = 1;
75                     flag1 = 1;
76                     break;
77                 }
78             }
79             if(flag1 == 1){
80                 flag1 = 0;
81                 break;
82             }
83         }
84     }
85 }
```

D、

用 if 分別包裝 19 種俄羅斯方塊，包含 T1、T2、.....。

E、

```
47 for(int a=2;;a++){
48     if(tetris[start+1][a] != 0 || tetris[start][a-1] != 0 || tetris[start+1][a-1] != 0 || tetris
```

若方塊中 4 塊的任一塊碰到 element = 1(其他方塊)或是 element = -1(底部)，則會停止；若還沒則繼續往下掉落。

```
49 if(tetris[start+1+dir][a-1] == 0 && tetris[start+dir][a-2] == 0 && tetris[start+1+dir][a-2] == 0
50     for(int b=1;;b++){
51         if(tetris[start+1+dir][a-1+b] != 0 || tetris[start+dir][a-2+b] != 0 || tetris[start+1+di
52             tetris[start+1+dir][a-1+b-1] = 1;
53             tetris[start+dir][a-2+b-1] = 1;
54             tetris[start+1+dir][a-2+b-1] = 1;
55             tetris[start+2+dir][a-2+b-1] = 1;
56             flag1 = 1;
57             break;
58         }
59     }
```

F、

結束 falling 後，判斷方塊所在位置 + dir(左右移動距離)是否 == 0，若 == 0 代表左右移動後不會碰到其他方塊；若 != 0 代表會碰到，則 invalid。

```

50 for(int b=1;;b++){
51     if(tetris[start+1+dir][a-1+b] != 0 || tetris[start+dir][a-2+b] != 0 || tetris[start+1+dir][
52         tetris[start+1+dir][a-1+b-1] = 1;
53         tetris[start+dir][a-2+b-1] = 1;
54         tetris[start+1+dir][a-2+b-1] = 1;
55         tetris[start+2+dir][a-2+b-1] = 1;
56         flag1 = 1;
57         break;
58     }
59 }

```

G、

結束 shift 後，判斷方塊在新位置時繼續往下掉落會否  $== 0$ ，若任一塊方塊  $!= 0$ ，代表會碰到其他方塊或底部，則停留在此位置，並將此位置的 element 改成 1。

```

541     for(int a=1;a<=column;a++){
542         for(int b=1;b<=row;b++){
543             if(tetris[b][a] == 0){
544                 flag = 1;
545                 break;
546             }
547         }
548
549         if(flag == 0){
550             for(int b=1;b<=row;b++){
551                 for(int c=a;c>=2;c--){
552                     tetris[b][c] = tetris[b][c-1];
553                 }
554                 for(int b=1;b<=row;b++){
555                     tetris[b][1] = 0;
556                 }
557             }
558         }
559     }
560 }

```

H、

判斷若有任一列皆  $== 1$ ，則  $flag = 0$ ，且令此列所有的 element = (此列-1)所有的 element，並令第一列所有的 element = 0。

```

561     ofstream cout("108062303_proj1.final");
562     for(int a=1;a<=column;a++){
563         for(int b=1;b<=row;b++){
564             cout << tetris[b][a] << ' ';
565             cout << endl;
566         }
567     }
568 }

```

I、

將輸出印到 10802303\_proj1.final 的檔案中。

```

568     fin.close();
569     return 0;
570 }

```

J、

關閉檔案並結束。

## 2) Test case Design

### 2-1) Detailed Description of the Test case

- I. 10 10 ->我想做個正方形的 block。

II.

1

1 1 1 ->我先將右下角填滿。

III.

1 1 1 1 ->將底層填平。

IV.

1

1 1 1 ->用倒 T 把左下角填滿，此時可消除最底層那列。

V.

1

1 1

1 ->繼續把左下角填滿。

VI.

```
J2 3 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0
1 1 1 0 0 0 0 0 0 0
1 1 1 1 1 0 0 0 0 1
```

1

1 1 1 ->將底層慢慢填滿。

VII.

```
S1 8 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0
1 1 1 0 0 0 0 0 1 1
1 1 1 1 1 0 0 1 1 1
```

1 1


1 1 ->將右下角填滿。

VIII.

```
0 6 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0
1 1 1 0 0 1 1 0 1 1
```


1 1

1 1 ->把最底層填滿，此時可消除最底層那列。

IX. 

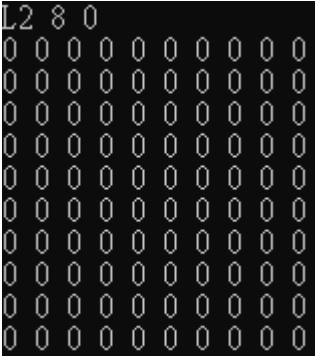
1 1 1

1 ->填補左側。

X. 


1 1 1

1 ->填補中間。

XI. 

1 1 1

1 ->填補右側，此時可消除最底層兩列。

XII. 

1 1

1 1 ->最後放下正方形大石頭，也就是我心中的大石頭。(終於做完 project1 了 :)。