

## Assignment 2 110503513 通訊二 王家欣

### 一、編譯結果

#### 1. 開新檔

```
● cindy0727@LAPTOP-SIEN7KHU:~/projects/HW_2$ gcc main.c user.c moving.c -o main -lm
○ cindy0727@LAPTOP-SIEN7KHU:~/projects/HW_2$ ./main -n -s game.txt
```

#### 2. 讀檔

```
● cindy0727@LAPTOP-SIEN7KHU:~/projects/HW_2$ gcc main.c user.c moving.c -o main -lm
○ cindy0727@LAPTOP-SIEN7KHU:~/projects/HW_2$ ./main -l record.txt
```

### 二、執行結果

#### 1. 開新檔

```
● cindy0727@LAPTOP-SIEN7KHU:~/projects/HW_2$ gcc main.c user.c moving.c -o main -lm
○ cindy0727@LAPTOP-SIEN7KHU:~/projects/HW_2$ ./main -n -s game.txt
```

1	2	3	4	5	6	7	8	9	
香	桂	銀	金	王	金	銀	桂	香	1
	飛						角		2
步	步	步	步	步	步	步	步	步	3
									4
									5
									6
步	步	步	步	步	步	步	步	步	7
	角						飛		8
香	桂	銀	金	玉	金	銀	桂	香	9

依據圖上座標 User1 輸入想移動的棋子的原始座標空格移動後的位置座標:

#### 2. 讀舊檔

```
● cindy0727@LAPTOP-SIEN7KHU:~/projects/HW_2$ ./main -l record.txt
```

1	2	3	4	5	6	7	8	9	
香	桂	銀	金	王	金	銀	桂	香	1
	飛						角		2
步	步	步	步	步	步	步	步	步	3
									4
									5
									6
步	步	步	步	步	步	步	步	步	7
	角						飛		8
香	桂	銀	金	玉	金	銀	桂	香	9

輸入f代表下一步,輸入b代表上一步:f

3. 吃子、判斷勝利後結束程式

```
依據圖上座標 User1 輸入想移動的棋子的原始座標空格移動後的位置座標:3 7 1 5
1 2 3 4 5 6 7 8 9
香 桂 銀 金 角 金 銀 桂 香 1
飛 2
步 步 步 步 步 步 步 3
步 步 步 4
步 步 5
步 6
步 步 步 步 步 7
飛 8
香 桂 銀 金 玉 金 銀 桂 香 9

user1勝利!!!!

cindy0727@LAPTOP-SIEN7KHU:~/projects/Hw_2$
```

4. 讀檔時上一步和下一步

```
輸入f代表下一步,輸入b代表上一步:b
1 2 3 4 5 6 7 8 9
香 桂 銀 金 王 金 銀 桂 香 1
飛 2
步 步 步 步 步 步 步 步 3
步 4
步 5
步 步 6
步 步 步 7
角 飛 8
香 桂 銀 金 玉 金 銀 桂 香 9
```

```
輸入f代表下一步,輸入b代表上一步:f
1 2 3 4 5 6 7 8 9
香 桂 銀 金 王 金 銀 桂 香 1
飛 2
步 步 步 步 步 步 步 步 3
步 4
步 5
步 步 6
步 步 步 7
角 飛 8
香 桂 銀 金 玉 金 銀 桂 香 9
```

5. 讀檔結束

```
輸入f代表下一步,輸入b代表上一步:f
1 2 3 4 5 6 7 8 9
香 桂 銀 金 角 金 銀 桂 香 1
飛 2
步 步 步 步 步 步 步 步 3
步 4
步 5
步 步 6
步 步 步 步 步 7
飛 8
香 桂 銀 金 玉 金 銀 桂 香 9

輸入f代表下一步,輸入b代表上一步:f

復盤結束!
cindy0727@LAPTOP-SIEN7KHU:~/projects/Hw_2$ f
```

### 三、實驗困難

1. 問題：一開始跑程式時一直出現 `segmentation fault`。

解答：一開始只宣告一個 `pointer` 並沒有告知 `pointer` 指向的位置，所以導致執行程式時出現 `segmentation fault`，之後在 `pointer` 加上初始位置後有改善問題，由此問題更了解 `pointer` 的運作，避免錯誤。

2. 問題：使用 `quene` 方式讀寫資料。

解答：設定四個 `int` 的變數，分別為 `front1, front2, rear1, rear2`，因為這次作業是使用陣列完成，所以只需要將 `rear` 或 `front` 加減就能順利讀寫不同位置的記憶體，雖然還沒使用到 `push` 或 `pop` 函式，卻能輕而易舉了解函式運作，對改成動態記憶體配置更佳容易。

3. 尚未完成：`libev`、輸入 `S` 存取檔案、復盤時吃子問題、讀檔問題。

### 四、心得

這次的作業嘗試了多新東西，除了學習將棋基本規則，也第一次使用 `call by address` 的方式存取參數，雖然一開始因為不清楚 `pointer` 的用法所以一直出現 `segmentation fault`，不過順利執行後卻時比過往方式更方便。