Project Report for Data Structure

Shogi

simple program

錢宇倫
通訊二
110503511

Department of Communication Engineering,

October 26, 2022

Contents

1 Compile Result

```
whimy@LAPTOP-3EHNOILN:~/workspace/shogi$ gcc -o shogi main.c board.c chess.c stack.c -lev
whimy@LAPTOP-3EHNOILN:~/workspace/shogi$ ./shogi
File create successfully.
```

figure.1 - compile

2 Run Code

Display chess board...

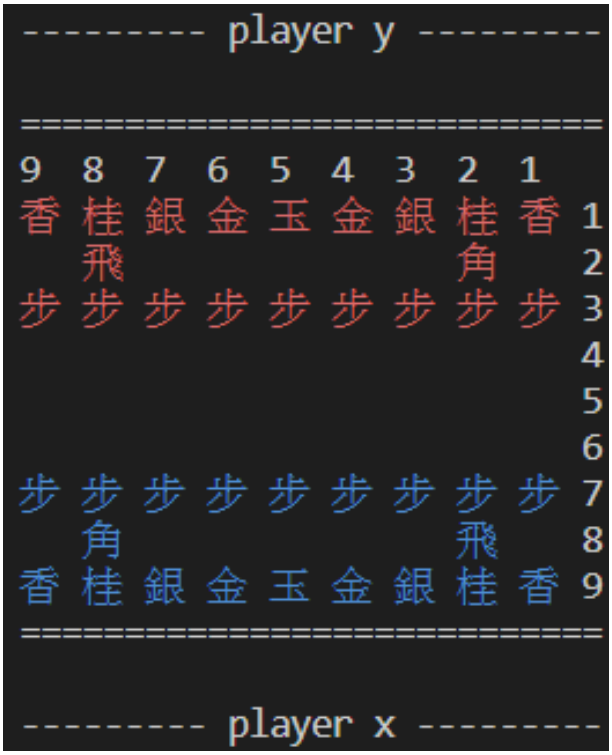

figure.2 - display chess board

1. Display board
2. Blue chess for player x and red for player y.
3. Display each player's *mochigoma*

Game display...

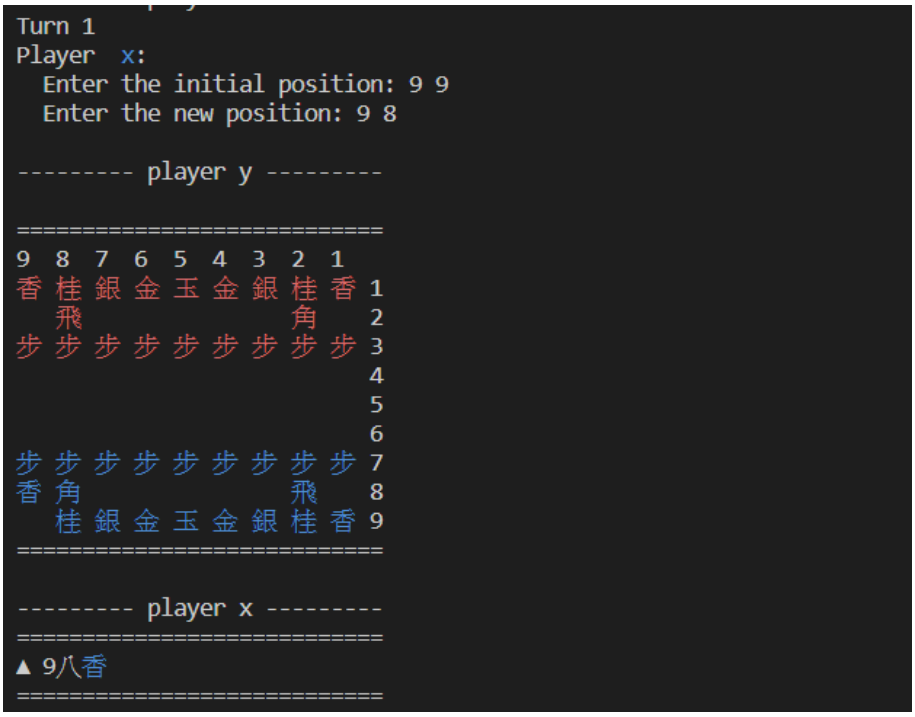

figure.3 - game display

1. Show what turn is it and whose the player now.
2. Enter the move-chess coordinate and the goal coordinate.
3. Display the board after the move.
4. Show the move in shogi notation.

Example of error...

- **Empty position** `-> No chess to move`

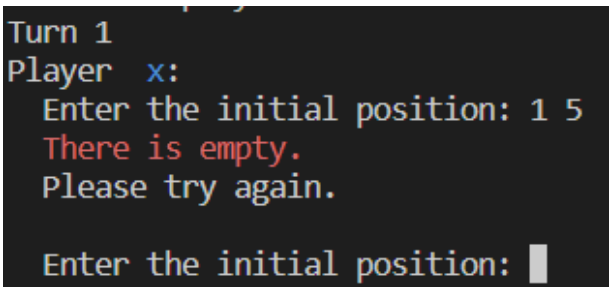

figure.4 - Error message

- **Pick enemy's chess** `-> Chess in assigned position is enemy's`

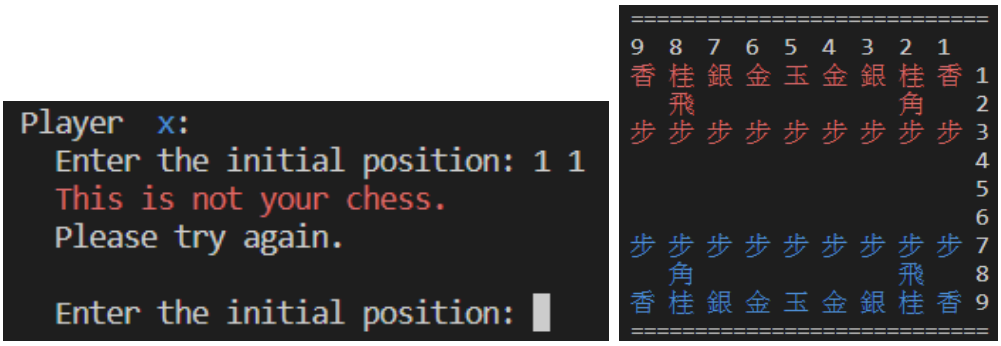

figure.5(a)(b) - (a)Error message(b)example board

- **Eat own chess** `-> Goal position point to the same player's chess`

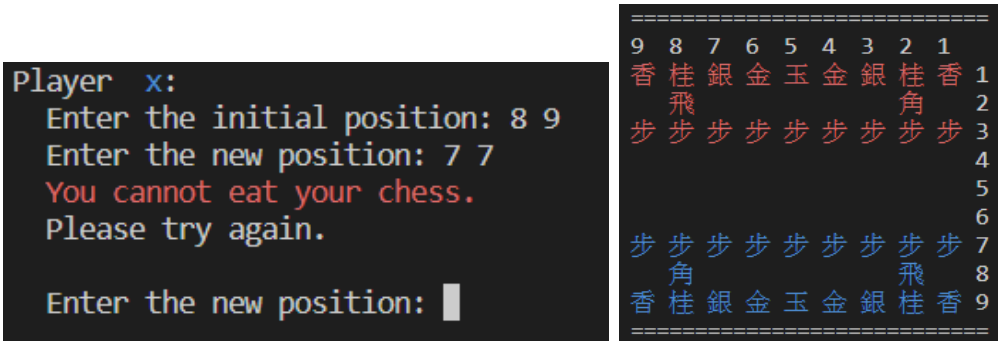

figure.6(a)(b) - (a)Error message(b)board display

- **Movement restriction**`-> Not following the movement rule`

Case 1: *Pawn* can only go on step forward each turn

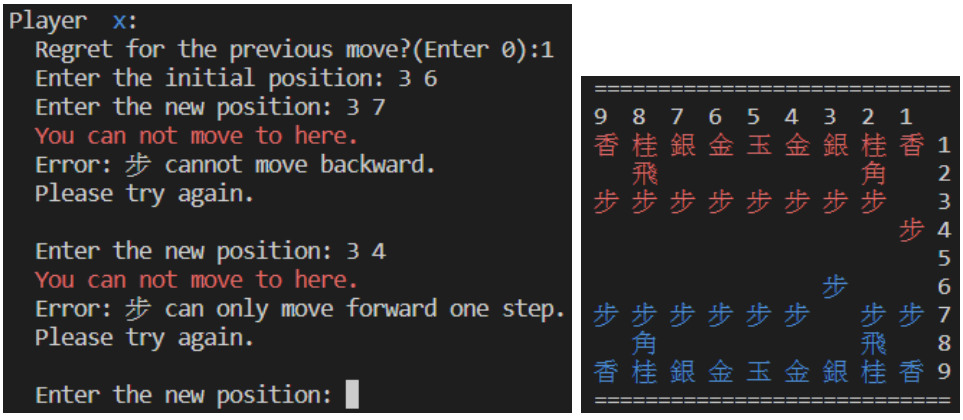

figure.7(a)(b) - (a)Error message(*Pawn* move)(b)example board

Case 2: *Bishop* can only move in a diagonal direction

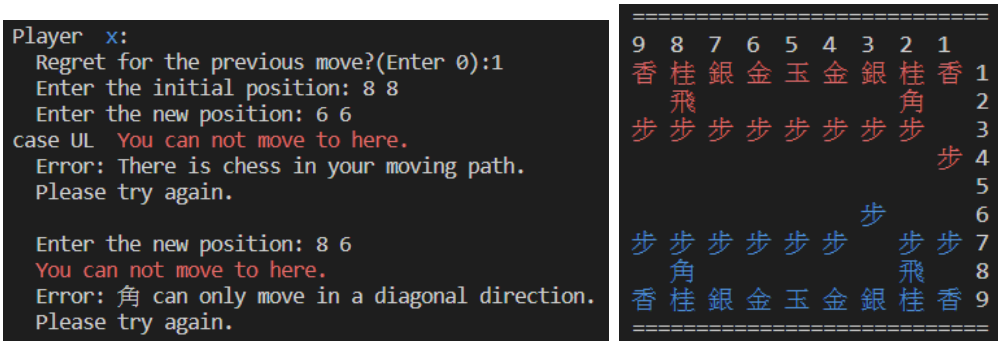

figure.8(a)(b) - (a)Error message(*Bishop* move)(b)example board

Eat chess

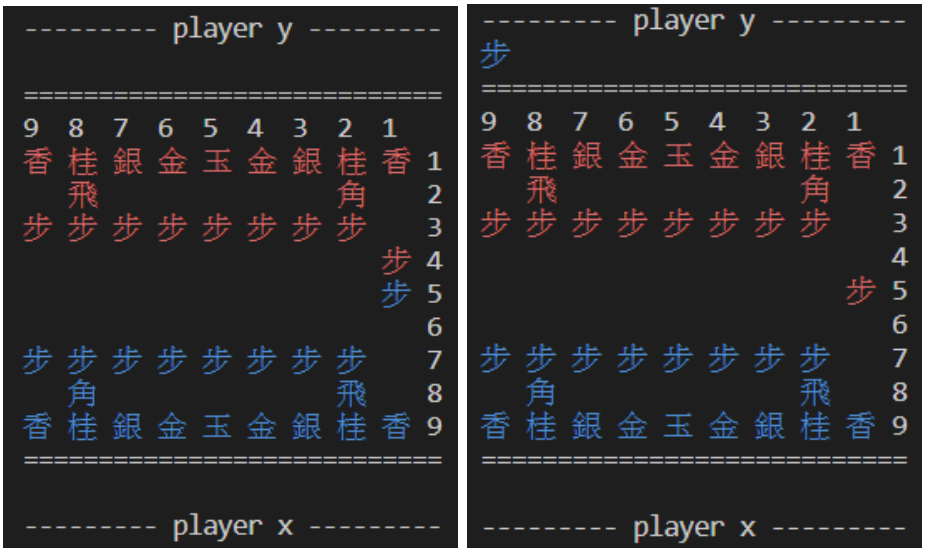

figure.9(a)(b) - (a)previous board(b)after eating chess

1. Cover the chess be ate by the present player's chess.
2. Pick the eaten chess into player's *mochigoma* bag.

Regret movement

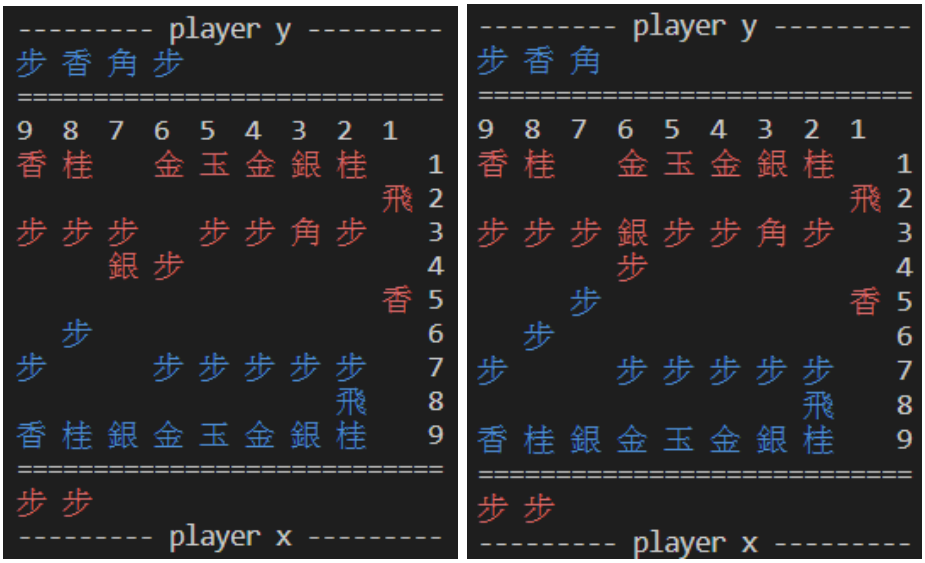

figure.10(a)(b) - (a)previous board(b)after regretting

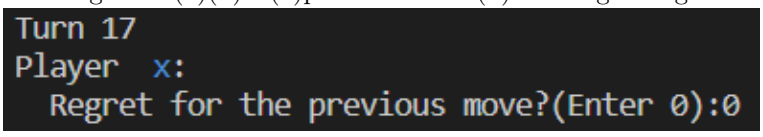

figure.11 - Asking for regretting movement

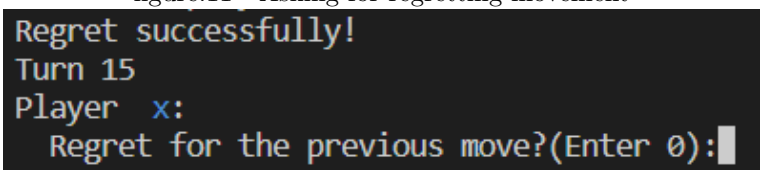

figure.12 - Regret successfully

1. Trace back to the previous turn of the player regretting.
2. If any chess had been eaten within the period,
   place it back to the board.
3. Display of turn will minus 2.(e.g. 17 to 15)

Chess manual Save the process into a CSV file

	先後手	筋	段	駒名
	先手▲	1	六	歩
	後手△	1	四	歩
	先手▲	1	五	歩
	後手△	1	五	歩
	先手▲	4	六	歩
	後手△	1	六	歩
	先手▲	1	六	香
	後手△	1	六	香

figure.13 - example CSV file

1. Save each turn in format of *shogi*'s notation.
2. Including whose chess, goal position, and which kind of chess.

3 Discussion

About pointer...
I'd tried three ways to set the chess board,which are:

1. Dynamic 2D array

```
char** board;
board = (char**)malloc(COLUMN * sizeof(char*));
for(int c = 0;c < COLUMN;c++){
    *(board + c) = (char*)malloc(ROW * sizeof(char));
}
```

figure.14 - Code of dynamic 2D array

It uses pointer's pointer,and I think it occupies much memory.Since I consider that it might use 9 char pointers to point to other 9 pointers, so I change into 1D array.

2. 1D array

```
char* board;
board = (char*)malloc(BOARD * sizeof(char));
```

figure.15 - Code of 1D array

Too much pointer and the code is so messy that it kept making errors, for example, pointer ALWAYS point to the wrong place.
Thus I change the way again.

3. static 2D array

```
char board[ROW][COLUMN];
```

figure.16 - Code of static 2D array

4 To-Improve

About code...

1. Use libev timer to count time for each turn
   (a) Show each turn takes how much time.
   (b) Timer flash each second.
2. Tidy pointer and function call.
3. Detect available path when got which chess to move, before enter the goal position.
   (a) Scan all board.
   (b) Check each position by following assigned chess movement rule.
   (c) Return whether there is any path for the assigned chess to go.

4. Renew start UI let user to do different action(e.g. play or read)
5. Drop chess system
    (a) ***Change format of *mochigoma*'s bag from stack to queue or any method that can random pop up.
    (b) Assign where to drop (position)
    (c) Assign which chess to drop (By number?)
    (d) Check the drop rule.
    (e) Drop the chess but with opposite color.
        i. Change symbol of chess(including enemy's chess).
6. Promotion system
7. Use NoSQL to save game record.

References

[1] Shogi rule

[2] Output with color in C

[3] pgn

[4] libev tutorial

[5] redis