# DSA Assignment2、3 Shogi Report

電機4B 108501025 沈冠璋

1. Build Guide

I upload two version of this program (one is array, and the other is linked-list) you can execute them by following steps

1. Array version

gcc -o aa main2.c stk.c -lev (aa is execute filename, you can set whatever you like)

1. Linked-list version

gcc -o bb main2.c link.c -lev (bb is execute filename, you can set whatever you like)

1. Execute Guide

Because I upload two version of this program you can execute them by following steps:

1. Array version

* Play:

./(execute filename) -n -s new\_game.txt

* Load manual:

./(execute filename) -l new\_game.txt

1. Linked-list version

* Play:

./(execute filename) -n -s new\_game\_link.txt

* Load manual:

./(execute filename) -l new\_game\_link.txt

1. Flowchart

Wrong input

1. Play a new game

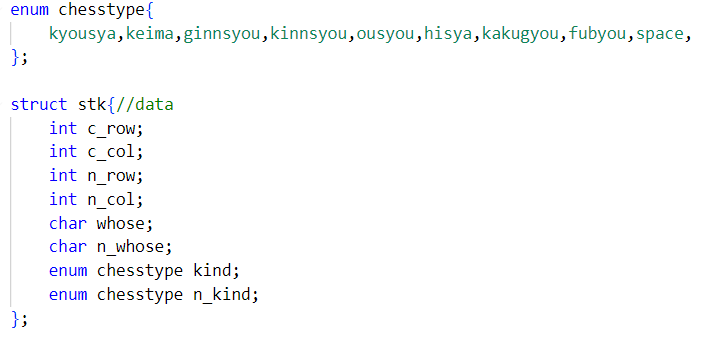
Repeat

1. Load manual
2. System Architecture

* I define a 2D array which element is “struct” to store each grid on the board

|  |  |  |
| --- | --- | --- |
| Struct | … | Struct |
| Struct | … | … |
| … | … | … |
| Struct | … | Struct |

* The struct consist of:



c\_row is the chess’s current row

n\_row is the chess’s next row if the chess will be moved

whose is on this grid the chess is belongs to which player

n\_whose is on this grid the chess will belongs to which player

kind is the chess type on this grid

n\_kind is the chess type which will be on this grid

chesstype is an enum to define the type of chess (including no type)

* In array version, each player step will be record in a stack which built in array
* In array version, each player step will be record in a stack which built in linked-list

1. Function Introduction
2. **startgame:**

Initialize the board.

1. **showboard:**

Show the board.

1. **check\_chess**

Check whether destination is out of the board.

1. **check\_direction**

Check whether player move chess in correct way.

1. **between series (between\_kyousya、between\_hisya…)**

If there is any chess between current row/column and next row/column, it will return 1.

1. **moving**

Move the chess to destination.

1. **game\_finish**

Check whether game is finished.

1. **recover**

Recover the board and turn to last situation.

1. **save**

Save data to a txt file.

1. **readmove**

(For load manual version) read next step.

1. **readback**

(For load manual version) read last step and recover