將棋函式說明

此部分依照.c 檔分類函式並說明每個的功用。

一、main.c

int main(int argc, char ** argv)

main 一開始先將不同條 linkedlist 的第一個位置抓出來，並初始化對應的數值，並宣告變數(rear, front)初始值。跑 getopt function，如果讀到 n 和 s，mode = 1 進入開新局，並開啟使用者輸入的 .txt 檔；讀到 l，mode = 2 進入讀檔模式，開啟使用者輸入的 .txt 檔。

開新局模式會先呼叫 initgame 函式初始化棋盤，再呼叫 game 函式開始遊戲，遊戲結束後跳出 game 函式。將遊戲過程中尚未存入站存記憶體的每一步走法寫入 userloadptr 這個 linkedlist(因為我是兩個玩家的情況分開寫，這裡是將原本在兩個 linkedlist 的兩個玩家的走法寫在同一個 linkedlist)，使用 while 迴圈並根據 frontone 和 fronttwo 的大小，決定是要寫哪一位玩家。寫進 userloadptr 後，再將 userloadptr 的資料從頭到尾寫在 .txt 檔中，結束玩遊戲模式。

讀檔模式會先呼叫 initgame 函式初始化棋盤，再呼叫 load 函式開始讀檔。讀檔完成後結束讀檔模式。

最後進行關檔(因為不管開新局模式或讀檔模式都需要開檔)，關檔完 return 0。

void initgame(user * chessboard)

此函式為初始化棋盤，先將所有棋盤內的東西設為空白，因為後面都是使用此棋盤在終端機上印出來，目的為防止原本該空白的位置印上其他字元。使用 call by address 因為之後判斷輸贏也需呼叫當下棋盤，改動資料不因為使用 call by value 導致資料消失。

二、game.c

void game(int * rear1, int * rear2, int * front1, int *front2, int *count, user **tmpusermefrontptr, user **tmpuserfrontyouptr)

此函式為玩遊戲模式的 function，rear1 代表玩家 1(userme)的尾巴，以此類推 rear2, front1, front2(其實 linkedlist 可以不用像 array 記數字，不過我之後算要存哪一筆資料比較方便)，count 是為了要記錄兩位玩家總共下了幾手，tmpusermefrontptr 和 tmpuseryoufrontptr 是從 main 傳進來之後要改動將每一手存進去暫存記憶體的頭。

一開始呼叫 creat_user1 和 creat_user2 做出兩個玩家的初始位置，並印出當前棋盤。rear1 +1 並呼叫 addme 新增下一個 node，輸入四個座標，符合條件後將第一個數值-48(原本是 ASCII code)，使用 user1move 移動 userme，印出棋

盤。

第二手開始進入迴圈，先判斷 user2 是否需增加新的節點(可能因為悔棋只是改動資料)，判斷是否新增節點方式為若某一方開始悔棋，會記錄當下下到第幾手(cg)並存進 tmpcg，輸入 f 的話 cg+1，輸入 b 的話 cg-1，是否需要新增節點根據 cg 是否大於 tmpcg 來判斷。user2 輸入座標後，若第一個字輸入 0 代表 user1 悔棋，退回上一步(user1)，並判斷是否有吃子須補回，印出棋盤後讓 user1 重新輸入座標，若第一個字輸入 0 代表 user2 悔棋，退回上一步，並判斷是否有吃子須補回，印出棋盤後讓 user2 重新輸入座標，並印出棋盤；若第一個字輸入 s 表示儲存，將資料寫進 userloadptr 的記憶體裡，並不會儲存 s 那一行，印出儲存完畢，讓 user1 重新輸入座標；若第一個字輸入非 s 和 0，代表可以移動棋子，將第一個輸入字元轉換成數字後，呼叫 user1move 函式移動 user1，呼叫 user1eatuser2 看是否有吃子，最後呼叫 whowins 判斷是否有人勝利。判斷 user1 是否需增加新的節點(可能因為悔棋只是改動資料)，user1 輸入座標，若第一個字輸入 0 代表 user2 悔棋，和上面情況類似，只是會依先出發的人不同而有所改變。若是成功判斷有誰勝利，印出誰勝利後會跳出函式，game 函式結束。

三、load.c

void load()

和 game 函式的邏輯和做法基本上一樣，不一樣的地方在於不需要輸入座標，而是選擇需不需要讀新的走法和輸入 f 和 b。

因為邏輯都一樣就不重複敘述，如果輸入 b，會記錄當下下到第幾手(c)並存進 tmpc，輸入 f 的話 c+1，輸入 b 的話 c-1，是否需要讀新資料會根據 c 是否大於 tmpc 來判斷，大於 tmpc 代表已經超越原本的走法，需要讀新檔，game 函式是否增加新的節點也是相同的道理。

四、moving.c

void user1move(user * user11, user * user22)

這個函式是在移動 user1 想要移動的棋子，首先先判斷 user1 單邊棋盤要移動的棋子原始座標是不是有東西和移動過後的座標是不是沒東西，如果符合可移動的條件，呼叫 user1chessandmove，當 user1chessandmove 回傳值為 1，代表移動符合規則，並移動棋子不符合規則的話需重新輸入座標。

void user2move(user * user22, user * user11)

和 user1move 的規則相同，只是是移動 user2。

void user1eatuser2(user * user1, user * user2)

若 user1 移動過後的座標和 user2 棋盤上某個棋子的座標重疊，代表要吃棋

子，並將當下座標和被吃掉的旗子記起來，方便悔棋。

void user2eatuser1(user * user2, user * user1)

和 user1eatuser2 移動規則相同，只是是 user2 吃掉 user1 的棋子。

int whowins(user * chessboard)

判斷是否有一方獲得勝利，將 tmp1 和 tmp2 先等於 0，搜尋整個棋盤，若是有 user1 的玉，tmp1 = 1，有 user2 的王，tmp2 = 1，如果 tmp1 = 1 且 tmp2 = 0，代表 user1 勝利，回傳 1，反之，user2 勝利回傳 2，兩方王都還在就回傳 0 繼續遊戲。

int user1chessandmove(user　* user1, user * user2)

這個函式式判斷 user1 棋子是否符合移動規則，先用 switch 找出對應棋子，並跑對應程式碼確認規則，有許多條件若符合規則回傳 1，不符合回傳 0。

int user2chessandmove(user * user2, user * user1)

和 user1chessandmove 邏輯和規則相同，只是是 user2。

user *addme(user *meptr)

這個函式使用 malloc，也就是新增一個節點給 userme 這條 linkedlist。先新增一個暫時的節點，並使用傳進來的位置將 prev 和 next 接上，將節點的各內容先等於前一個(其實只要棋盤就好，先複製前一個棋盤才能移動，不過複製座標也沒差，就當初始化)，有無吃子的地方先設為空，最後讓 userme 指到下一個位置，並回傳新的位置。

user *addyou(user *youptr)

和 addme 相同，只是是新增一個節點給 useryou(明明內容都一樣硬要多寫一個)。

user *meequal(user *meptr)

因為悔棋，所以可能不需要新增節點，可是因為悔棋所以要讓前棋盤等於前一筆資料的棋盤和座標，所以創造此函式。

user *youequal(user *youptr)

和 youequal 相同，只是是給 user2 用。

user *meequalload(user *meptr)

和 meequal 其實是相同的東西，只是用在 load 中，因為是要往前往後，已經被讀取的座標走法不必被更改，所以少了複製座標。

user *youequalload(user *youptr)

和 youequalload 相同，只是是給 user2 用。

- 其實這幾個函式都差不多，只是我太笨，寫了很多一樣的。

五、user.c

void creat_user1(user *user1)

設置第一個 user1 的內容(因為後面都是複製前一個棋盤)，並排好對應的初始位置，將吃子設為空。採大寫英文字代表 user1，對應英文字為諧音。

void creat_user2(user *user2)

設置第一個 user2 的內容(因為後面都是複製前一個棋盤)，並排好對應的初始位置，將吃子設為空。採小寫英文字代表 user1，對應英文字為諧音。

void chineseuser1 (char letter)

下面印出棋盤會使用到此函式，將大寫英文字代號轉成對應中文字，並印出藍色字體。

void chineseuser2 (char letter)

下面印出棋盤會使用到此函式，將小寫英文字代號轉成對應中文字，並印出紅色字體。

void printchess(user const * user1, user const * user2, user * chessboard)

這個函式式將 user1 和 user2 兩個半棋盤合成一個棋盤，呼叫 chineseuser1 和 chineseuser2 印出中文字在終端機上。

六、timer.c

void user1timer()

初始化 user1 並開始使用 libev 監測是否輸入 enter。ev_io_init 監測是否有 io 事件，ev_io_start 開始 stop wacher 監測；ev_timer_init 開始進行每一秒的監測，ev_timer_start 開始 timeout 監測(這兩個如果改成每一秒印一次棋盤會用到)，ev_run 將 watcher 放在迴圈裡並抓取現在時間。

void user2timer()

同理 user1timer。

void user1timeout_cb (EV_P_ ev_timer *w, int revents)

每一秒抓取一次現在時間，並將時間轉換成時分秒，照理講這個函式也要有每秒印一次棋盤，因為時間不足，目前無法完成。

void user2timeout_cb (EV_P_ ev_timer *w, int revents)

同理 user1timeout_cb。

void user1stop_cb (EV_P_ ev_io *w, int revents)

當發生 io 事件時，觸發這個 function，一開始先 system clear 清空畫面，並將兩時間相減得知這手使用時間，再加上先前已使用的時間形成新的已使用時間，並將時間轉換成時分秒後印出，離開這個 function。

void user2stop_cb (EV_P_ ev_io *w, int revents)

同理 user1stop_cb。