#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_ROW 15

#define MAX\_COL 15

#define WHITE -1

#define BLACK 1

#define BLANK 0

void draw\_chessboardn(int row, int col, int chessboard[][MAX\_COL]);

void draw\_chessman(int type, char \*tableline);

void draw\_menu(void);

void person\_person(void);

int is\_full(int chessboard[][MAX\_COL], int row, int col);

int is\_win(int chessboard[][MAX\_COL], int row, int col);

void save\_chess(int chessboard[][MAX\_COL], int row, int col);

void replay\_chess(void);

void undo\_move(int chessboard[][MAX\_COL], int row, int col, int step);

int main() {

int choice;

draw\_menu();

while (1) {

scanf("%d", &choice);

switch (choice) {

case 1:

person\_person();

case 2:

replay\_chess();

break;

case 3:

exit(0);

break;

default:

printf("输入错误，请重新选择\n");

}

}

return 0;

}

// 绘制棋盘

void draw\_chessboardn(int row, int col, int chessboard[][MAX\_COL]) {

for (int i = 0; i < row; i++) {

if (i == 0) {

for (int j = 0; j < col; j++) {

if (j == 0)

draw\_chessman(chessboard[i][j], "┌ ");

else if (j == 14)

draw\_chessman(chessboard[i][j], "┐");

else

draw\_chessman(chessboard[i][j], "┬ ");

}

printf("\n");

} else if (i == 14) {

for (int j = 0; j < col; j++) {

if (j == 0)

draw\_chessman(chessboard[i][j], "└ ");

else if (j == 14)

draw\_chessman(chessboard[i][j], "┘ ");

else

draw\_chessman(chessboard[i][j], "┴ ");

}

printf("\n");

} else {

for (int j = 0; j < col; j++) {

if (j == 0)

draw\_chessman(chessboard[i][j], "├ ");

else if (j == 14)

draw\_chessman(chessboard[i][j], "┤");

else

draw\_chessman(chessboard[i][j], "┼ ");

}

printf("\n");

}

}

}

// 绘制棋子

void draw\_chessman(int type, char \*tableline) {

if (type == WHITE)

printf("●");

if (type == BLACK)

printf("○");

if (type == BLANK)

printf("%s", tableline);

}

// 绘制主菜单

void draw\_menu(void) {

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\* 欢迎使用五子棋 \*\*\*\*\*\*\*\n");

printf("\*\*\* 研发者：名字名字 \*\*\*\n");

printf("\*\*\* 请选择对战方式 \*\*\*\n");

printf("\* 1.人-人对战 \*\n");

printf("\* 2.复盘 \*\n");

printf("\* 3.退出 \*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("请选择：");

}

// 人人对战

void person\_person(void) {

int chessboard[MAX\_ROW][MAX\_COL] = {BLANK};

int chess\_history[MAX\_ROW \* MAX\_COL][MAX\_ROW][MAX\_COL] = {0}; // 用于保存每一步的棋盘状态

int step = 0; // 记录当前步数

int i, j;

char key;

draw\_chessboardn(MAX\_ROW, MAX\_COL, chessboard);

for (step = 1; step <= MAX\_ROW \* MAX\_COL; step++) { // 黑子先行，然后双方轮流下棋

if (step % 2 == 1) { // 当前步数为单数，黑棋落子。

printf("请黑棋落子:");

while (1) {

scanf("%d %d", &i, &j);

if (chessboard[i][j] != BLANK) {

printf("该位置已有棋子，请重新输入\n"); // 棋子只能落在空白处

continue;

}

if (i >= MAX\_ROW || j >= MAX\_COL || i < 0 || j < 0) {

printf("输入超出棋盘范围，请重新输入\n"); // 棋子坐标不可超出棋盘

continue;

}

break;

}

chessboard[i][j] = BLACK;

draw\_chessboardn(MAX\_ROW, MAX\_COL, chessboard);

if (is\_win(chessboard, MAX\_ROW, MAX\_COL) == BLACK) {

printf("黑棋胜\n");

save\_chess(chessboard, MAX\_ROW, MAX\_COL);

// 保存当前棋盘状态

return;

}

save\_chess(chessboard, MAX\_ROW, MAX\_COL);

// 保存当前棋盘状态

for (int row = 0; row < MAX\_ROW; row++) {

for (int col = 0; col < MAX\_COL; col++) {

chess\_history[step][row][col] = chessboard[row][col];

}

}

} else if (step % 2 == 0) { // 当前步数为双数，则白棋落子

printf("请白棋落子:");

while (1) {

scanf("%d %d", &i, &j);

if (chessboard[i][j] != BLANK) {

printf("该位置已有棋子，请重新输入\n"); // 棋子只能落在空白处

continue;

}

if (i >= MAX\_ROW || j >= MAX\_COL || i < 0 || j < 0) {

printf("输入超出棋盘范围，请重新输入\n"); // 棋子坐标不可超出棋盘

continue;

}

break;

}

chessboard[i][j] = WHITE;

draw\_chessboardn(MAX\_ROW, MAX\_COL, chessboard);

if (is\_win(chessboard, MAX\_ROW, MAX\_COL) == WHITE) {

printf("白棋胜\n");

save\_chess(chessboard, MAX\_ROW, MAX\_COL);

// 保存当前棋盘状态

return;

}

save\_chess(chessboard, MAX\_ROW, MAX\_COL);

// 保存当前棋盘状态

for (int row = 0; row < MAX\_ROW; row++) {

for (int col = 0; col < MAX\_COL; col++) {

chess\_history[step][row][col] = chessboard[row][col];

}

}

}

if (is\_full(chessboard, MAX\_ROW, MAX\_COL) == 1) {

printf("棋盘已满");

exit(0);

}

// 悔棋操作

printf("是否悔棋(y/n): ");

scanf(" %c", &key);

if (key == 'y' || key == 'Y') {

if (step <= 1) {

printf("无法悔棋\n");

continue;

}

step -= 2; // 回退两步

// 恢复棋盘状态

for (int row = 0; row < MAX\_ROW; row++) {

for (int col = 0; col < MAX\_COL; col++) {

chessboard[row][col] = chess\_history[step][row][col];

}

}

draw\_chessboardn(MAX\_ROW, MAX\_COL, chessboard);

}

}

}

// 判断棋盘是否已满

int is\_full(int chessboard[][MAX\_COL], int row, int col) {

int ret = 1;

for (int i = 0; i < row; i++) {

for (int j = 0; j < col; j++) {

if (chessboard[i][j] == BLANK) {

ret = 0; // 存在空白位置，棋盘未满

break;

}

}

}

return ret;

}

// 判断是否获胜

int is\_win(int chessboard[][MAX\_COL], int row, int col) {

int i, j;

// 横向判断

for (i = 0; i < row; i++) {

for (j = 0; j < col - 4; j++) {

if (chessboard[i][j] != BLANK && chessboard[i][j] == chessboard[i][j + 1] && chessboard[i][j] == chessboard[i][j + 2] && chessboard[i][j] == chessboard[i][j + 3] && chessboard[i][j] == chessboard[i][j + 4]) {

return chessboard[i][j];

}

}

}

// 纵向判断

for (i = 0; i < row - 4; i++) {

for (j = 0; j < col; j++) {

if (chessboard[i][j] != BLANK && chessboard[i][j] == chessboard[i + 1][j] && chessboard[i][j] == chessboard[i + 2][j] && chessboard[i][j] == chessboard[i + 3][j] && chessboard[i][j] == chessboard[i + 4][j]) {

return chessboard[i][j];

}

}

}

// 左上到右下判断

for (i = 0; i < row - 4; i++) {

for (j = 0; j < col - 4; j++) {

if (chessboard[i][j] != BLANK && chessboard[i][j] == chessboard[i + 1][j + 1] && chessboard[i][j] == chessboard[i + 2][j + 2] && chessboard[i][j] == chessboard[i + 3][j + 3] && chessboard[i][j] == chessboard[i + 4][j + 4]) {

return chessboard[i][j];

}

}

}

// 右上到左下判断

for (i = 0; i < row - 4; i++) {

for (j = 4; j < col; j++) {

if (chessboard[i][j] != BLANK && chessboard[i][j] == chessboard[i + 1][j - 1] && chessboard[i][j] == chessboard[i + 2][j - 2] && chessboard[i][j] == chessboard[i + 3][j - 3] && chessboard[i][j] == chessboard[i + 4][j - 4]) {

return chessboard[i][j];

}

}

}

return 0;

}

// 保存棋盘状态

void save\_chess(int chessboard[][MAX\_COL], int row, int col) {

FILE \*fp;

fp = fopen("chess.txt", "w");

if (fp == NULL) {

printf("保存失败\n");

return;

}

for (int i = 0; i < row; i++) {

for (int j = 0; j < col; j++) {

fprintf(fp, "%d ", chessboard[i][j]);

}

fprintf(fp, "\n");

}

fclose(fp);

printf("保存成功\n");

}

// 复盘功能

void replay\_chess(void) {

FILE \*fp;

char ch;

fp = fopen("chess.txt", "r");

if (fp == NULL) {

printf("暂无复盘记录\n");

return;

}

while ((ch = fgetc(fp)) != EOF) {

putchar(ch);

}

fclose(fp);

}