#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <unistd.h> // 用于usleep函数

#define BOARD\_SIZE 15

typedef struct {

char board[BOARD\_SIZE][BOARD\_SIZE];

} GobangBoard;

void initBoard(GobangBoard \*board) {

for (int i = 0; i < BOARD\_SIZE; i++) {

for (int j = 0; j < BOARD\_SIZE; j++) {

board->board[i][j] = '.';

}

}

}

void printBoard(GobangBoard board) {

printf("\n ");

for (int i = 0; i < BOARD\_SIZE; i++) {

printf("%2d ", i);

}

printf("\n");

for (int i = 0; i < BOARD\_SIZE; i++) {

printf("%2d ", i);

for (int j = 0; j < BOARD\_SIZE; j++) {

printf("%c ", board.board[i][j]);

}

printf("\n");

}

printf("\n");

}

int countSameInDirection(GobangBoard board, int x, int y, int dx, int dy, char player) {

int count = 0;

for (int i = x, j = y; i >= 0 && i < BOARD\_SIZE && j >= 0 && j < BOARD\_SIZE && board.board[i][j] == player; i += dx, j += dy) {

count++;

}

for (int i = x - dx, j = y - dy; i >= 0 && i < BOARD\_SIZE && j >= 0 && j < BOARD\_SIZE && board.board[i][j] == player; i -= dx, j -= dy) {

count++;

}

return count;

}

int checkWin(GobangBoard board, char player) {

for (int i = 0; i < BOARD\_SIZE; i++) {

for (int j = 0; j < BOARD\_SIZE; j++) {

if (board.board[i][j] == player) {

if (countSameInDirection(board, i, j, 0, 1, player) >= 5) return 1;

if (countSameInDirection(board, i, j, 1, 0, player) >= 5) return 1;

if (countSameInDirection(board, i, j, 1, 1, player) >= 5) return 1;

if (countSameInDirection(board, i, j, 1, -1, player) >= 5) return 1;

}

}

}

return 0;

}

void twoPlayerMode(GobangBoard \*board) {

char players[] = {'X', 'O'};

char players[] = {'X', 'O'};

int currentPlayer = 0;

int moveX, moveY;

while (1) {

printBoard(\*board);

printf("玩家 %c，请输入行和列（用空格隔开）: ", players[currentPlayer]);

while (scanf("%d %d", &moveX, &moveY) != 2 || moveX < 0 || moveX >= BOARD\_SIZE || moveY < 0 || moveY >= BOARD\_SIZE || board->board[moveX][moveY] != '.') {

printf("输入错误，请重新输入行和列（用空格隔开）: ");

while (getchar() != '\n'); // 清除输入缓冲区

}

board->board[moveX][moveY] = players[currentPlayer];

if (checkWin(\*board, players[currentPlayer])) {

printBoard(\*board);

printf("玩家 %c 获胜！\n", players[currentPlayer]);

break;

}

currentPlayer = (currentPlayer + 1) % 2;

}

}

void computerMove(GobangBoard \*board, char computerPlayer) {

int emptyCells[BOARD\_SIZE \* BOARD\_SIZE][2];

int count = 0;

for (int i = 0; i < BOARD\_SIZE; i++) {

for (int j = 0; j < BOARD\_SIZE; j++) {

if (board->board[i][j] == '.') {

emptyCells[count][0] = i;

emptyCells[count][1] = j;

count++;

}

}

}

if (count > 0) {

int randomIndex = rand() % count;

board->board[emptyCells[randomIndex][0]][emptyCells[randomIndex][1]] = computerPlayer;

}

}

void humanVsComputerMode(GobangBoard \*board) {

char humanPlayer = 'X';

char computerPlayer = 'O';

int moveX, moveY;

while (1) {

printBoard(\*board);

printf("你（%c），请输入行和列（用空格隔开）: ", humanPlayer);

while (scanf("%d %d", &moveX, &moveY) != 2 || moveX < 0 || moveX >= BOARD\_SIZE || moveY < 0 || moveY >= BOARD\_SIZE || board->board[moveX][moveY] != '.') {

printf("输入错误，请重新输入行和列（用空格隔开）: ");

while (getchar() != '\n'); // 清除输入缓冲区

}

board->board[moveX][moveY] = humanPlayer;

if (checkWin(\*board, humanPlayer)) {

printBoard(\*board);

printf("恭喜你获胜了！\n");

break;

}

printf("电脑正在思考...\n");

usleep(500000); // 模拟思考时间

computerMove(board, computerPlayer);

if (checkWin(\*board, computerPlayer)) {

printBoard(\*board);

printf("很遗憾，电脑获胜了！\n");

break;

}

}

}

int main() {

GobangBoard board;

initBoard(&board);

srand((unsigned int)time(NULL));

int choice;

printf("\n\n欢迎来到五子棋游戏！\n");

printf("1. 双人对战\n2. 人机对战\n请选择游戏模式: ");

scanf("%d", &choice);

switch (choice) {

case 1:

twoPlayerMode(&board);

printf("感谢你的游戏，再见！\n");

break;

case 2:

humanVsComputerMode(&board);

printf("感谢你的游戏，再见！\n");

break;

default:

printf("无效选择，再见！\n");

}

return 0;

}