C语言实现三子棋

`Game one`
@TOC

一、游戏描述

三子棋我们小时候都有玩过，那时也叫圈圈叉叉、井字棋等。将正方形对角线连起来，相对两边依次摆上三个双方棋子，只要将自己的三个棋子走成一条线，对方就算输了。但是，有很多时候会出现死局的情况。即如9 * 9的宫格都已经落子，但并未满足三字一线。

> 采用程序设计方法来实现，用函数来封装各个功能的实现。

考虑如下功能：

1. 一个简易的菜单提示
2. 如何打印出棋盘
3. 数组来实现棋盘
4. 落子规则实现（人机对战）
5. 胜负规则判断

设计三个文件：

test.c ： 主函数程序
game.c ： 游戏函数实现
game.h ： 代码头文件包含，函数声明，变量声明

二、主函数main()：

> 实现一个菜单提示，作为游戏最初开始，game函数实现游戏下棋框架逻辑

```
# define _CRT_SECURE_NO_WARNINGS 1
#include "game.h"

void menu() {
    printf("-------------------------------------\n");
    printf("              1. PLAY\n");
    printf("              0. EXIT\n");
    printf("-------------------------------------\n");
}
void game() {
    char ret = 0;
    char board[ROW][COL];
    InitBoard(board, ROW, COL);  //初始化棋盘
    disPlay(board, ROW, COL);  //打印棋盘
    //游戏过程
    //玩家下棋
    while (1)
```

```
    {
        Player(board, ROW, COL);
        disPlay(board, ROW, COL);
        ret = isWin(board, ROW, COL);
        if (ret != 'C') {
            break;
        }

        Computer(board, ROW, COL);
        disPlay(board, ROW, COL);
        ret = isWin(board, ROW, COL);
        if (ret != 'C') {
            break;
        }

    }
    if (ret == '*')
    {
        printf("玩家胜出！\n");

    }
    else if (ret == '#')
    {
        printf("电脑胜出！\n");
    }
    else if(ret == 'Q')
    {
        printf("平局！\n");
    }
    disPlay(board, ROW, COL);
}
int main() {
    //二维数组来定义棋盘
    int input;
    srand((unsigned int)time(NULL));
    do
    {
        menu();
        printf("请输入你的选择：");
        scanf("%d", &input);
        switch (input)
        {
        case 1:
            game();
            break;
        case 0:
            printf("退出游戏\n");
            break;
        default:
            printf("非法的输入！");
            break;
        }
    } while (input);
    return 0;
}
```

游戏开始前菜单提示如下：

```
--------------------------------------
                1. PLAY
                0. EXIT
--------------------------------------
请输入你的选择：_
```


每次进入游戏即进入下棋落子循环，满足下棋结束条件后再次打印菜单选择是否新一轮游戏。

棋盘打印如下，每发生一个落子就判断是否满足获胜或死局，再紧接着打印棋盘。

```
--------------------------------------
                1. PLAY
                0. EXIT
--------------------------------------
请输入你的选择：1
     |     |
 ___ | ___ | ___
     |     |
 ___ | ___ | ___
     |     |
玩家下棋 -->
请输入您的落子位置：_
```


三、game.c游戏函数

> 实现打印棋盘；玩家落子；电脑随机落子；胜负条件判断；棋盘是否已满

```c
# define _CRT_SECURE_NO_WARNINGS 1
#include "game.h"
//初始化棋盘
void InitBoard(char board[ROW][COL], int row, int col) {
    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < col; j++)
        {
            board[i][j] = ' ';
        }
    }
}
//实现打印显示棋盘(考虑行列变化时棋盘灵活打印)
void disPlay(char board[ROW][COL], int row, int col) {
    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < col; j++)
        {
            printf(" %c ", board[i][j]);
            if (j < col - 1)
```

```
                printf("|");
        }
        printf("\n");
        //printf(" %c | %c | %c \n", board[i][0], board[i][1], board[i][2]);
        //if (i < row - 1)
        //  printf("---|---|---\n");

        //打印每行之间的分隔
        if (i < row - 1)
        {
            for (int j = 0; j < col; j++)
            {
                printf("---");
                if (j < col - 1)
                    printf("|");
            }
            printf("\n");
        }
    }
}

void Player(char board[ROW][COL], int row, int col) {
    printf("玩家下棋 -->\n");
    int x, y = 0;  //玩家坐标
    //判断坐标合法性（正整数；是否被占用）
    while (1)
    {
        printf("请输入您的落子位置：");
        scanf("%d %d", &x, &y);
        if ((x - 1 >= 0 && x - 1 <= row) && (y - 1 >= 0 && y - 1 <= row)) {
            if (board[x - 1][y - 1] == ' ') {
                board[x - 1][y - 1] = '*';
                break;
            }
            else
            {
                printf("当前位置已有棋子\n");

            }
        }
        else
        {
            printf("坐标输入非法！\n");

        }
    }
}

void Computer(char board[ROW][COL], int row, int col) {
    printf("电脑下棋 -->\n");
    //电脑随机下子
    while (1)
    {
        int x = rand() % row;
        int y = rand() % col;
```

```
        if (board[x][y] == ' ')
        {
            board[x][y] = '#';
            break;
        }
    }
}

//判断棋盘是否已满
int isFull(char board[ROW][COL], int row, int col) {
    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < col; j++)
        {
            if (' ' == board[i][j])
            {
                return 0;
            }
        }
    }
    return 1;
}
char isWin(char board[ROW][COL], int row, int col) {
    for (int i = 0; i < row; i++)
    {
        if (board[i][0] == board[i][1] && board[i][1] == board[i][2] && board[i]
[0] != ' ') {
            return board[i][0];
        }
    }

    for (int i = 0; i < col; i++)
    {
        if (board[0][i] == board[1][i] && board[1][i] == board[2][i] && board[0]
[i] != ' ') {
            return board[0][1];
        }
    }

    if (board[0][0] == board[1][1] && board[1][1] == board[2][2] && board[0][0]
!= ' ')
        return board[0][0];
    if (board[0][2] == board[1][1] && board[1][1] == board[2][0] && board[0][2]
!= ' ')
        return board[0][2];

    //在判断棋盘是否已满
    if (isFull(board, row, col) == 1)
    {
        return 'Q';  //平局
    }

    return 'C'; //继续游戏

}
```

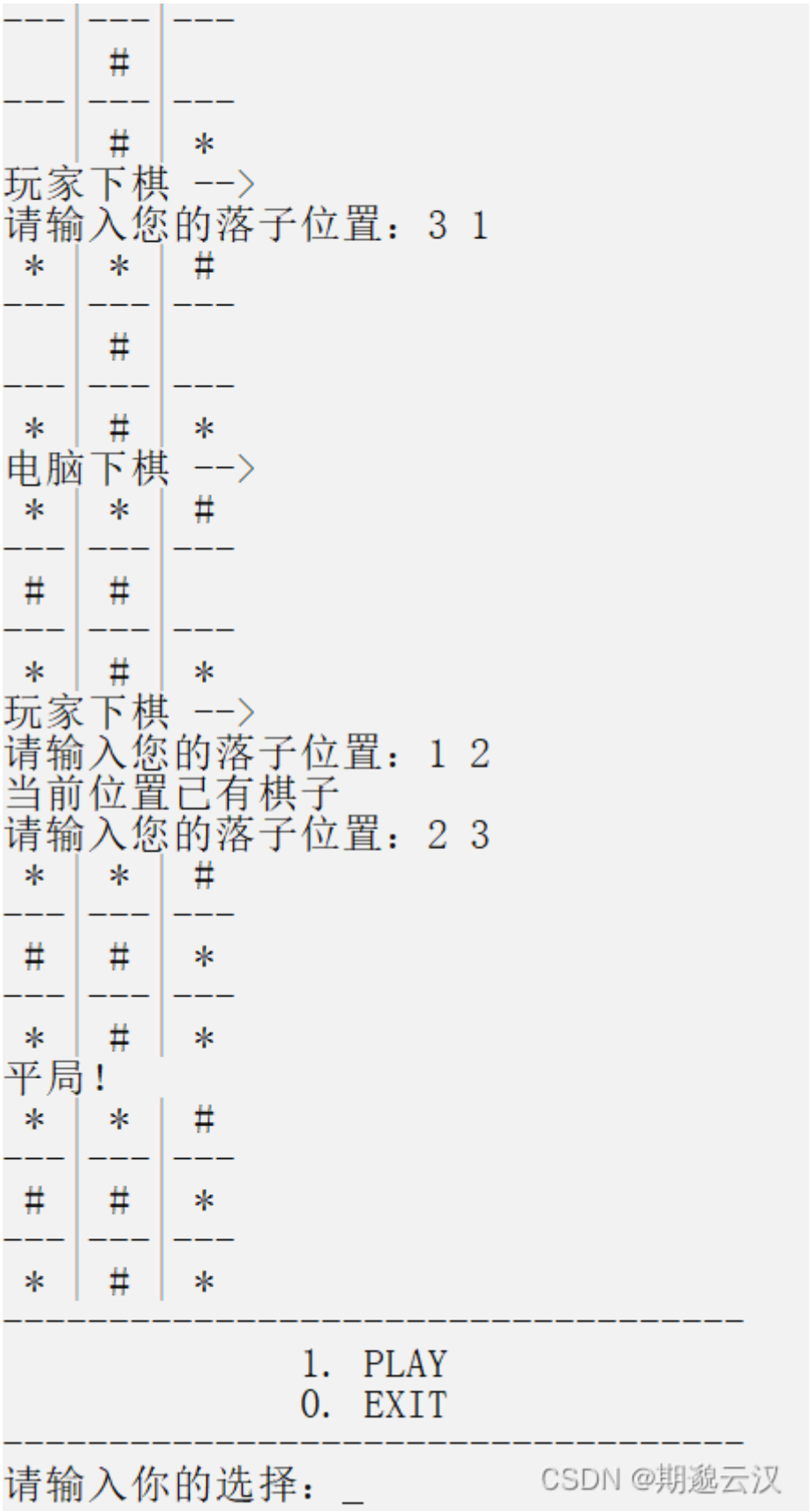
四、头文件game.h

```
#pragma once

#include <stdio.h>
#include <time.h>
#include <stdlib.h>
//将行列定义为宏
#define ROW 3
#define COL 3

//函数声明
void disPlay(char board[ROW][COL], int row, int col);  //打印棋盘
void InitBoard(char board[ROW][COL], int row, int col);  //初始化棋盘
void Player(char board[ROW][COL], int row, int col);  //玩家下棋
void Computer(char board[ROW][COL], int row, int col);  //电脑下棋
char isWin(char board[ROW][COL], int row, int col);  //判断游戏状态
```

棋局模拟

下面模拟死局过程

```
-----------------------------------
               1. PLAY
               0. EXIT
-----------------------------------
请输入你的选择：1
     |     |
  ---|---|---
     |     |
  ---|---|---
     |     |
玩家下棋 -->
请输入您的落子位置：1 1
  *  |     |
  ---|---|---
     |     |
  ---|---|---
     |     |
电脑下棋 -->
  *  |     |
  ---|---|---
     |  #  |
  ---|---|---
     |     |
玩家下棋 -->
请输入您的落子位置：3 3
  *  |     |
  ---|---|---
     |  #  |
  ---|---|---
     |     |  *
电脑下棋 -->
  *  |     |
  ---|---|---
     |  #  |
  ---|---|---
     |  #  |  *
玩家下棋 -->
请输入您的落子位置：1 2
  *  |  *  |
  ---|---|---
     |  #  |
  ---|---|---
     |  #  |  *
电脑下棋 -->
  *  |  *  |  #
```



当然，三子棋只是一个比较简单的下棋，其规则也很简单，ROW，COL行列值可以自己定义，那么相应的胜负判断规则也要改变，实现N * M棋盘实现自定义规则也并不难。