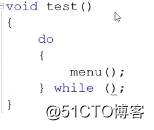
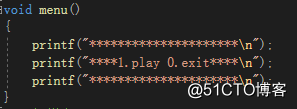
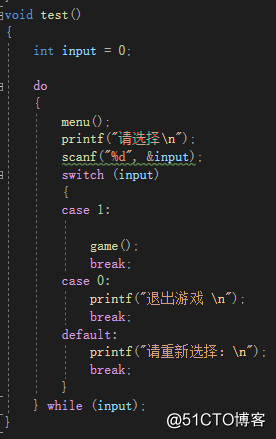
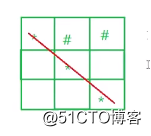
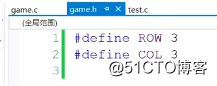
C语言数组实现三子棋  
分别创建测试模块，游戏模块和游戏头文件  
在测试模块中写入主函数  
int main （）  
{  
test （）；  
return 0；  
}

首先一局游戏我们都希望玩完一把之后可以继续玩，所以我们在测试函数中使用 do while 结构  
首先写入游戏进入时候的菜单选项  
  
1进入游戏，0退出游戏  
  
在测试函数中用switch函数对玩家输入的数字进行选择  
  
只有当input 为0时，判断条件为假，循环终止，游戏结束，当input是非零的时候，继续循环

接下来构思游戏方式  
首先游戏的棋盘是一个九宫格  
其次玩家走一步，电脑走一步，当任意一方三个棋子一条直线的时候游戏结束  
  
游戏需要存放棋子，存放\*或者#  
创建数组 存放走出的棋盘信息  
char board[ROW][COL]={0};//全部空格因为0不可打印所以是空格  
在游戏的头文件定义宏，代替数字的行列，提高代码的可读性  
  
使用头文件需要引用 #include“game.h”  
放置棋子需要打印一个空的棋盘，每个元素都是“空格”  
理想棋盘：  


初始化棋盘  
InitBoard(board,ROW,COL);  
//初始化棋盘,就不用担心一开始放置的是什么了  
在game.h中申明，在game.c中实现  
申明：void InitBoard(int board[ROW][COL],int row ,int col);  
使用：  
void InitBoard(char board[ROW][COL], int row, int col)  
{  
int i = 0;  
int j = 0;  
for (i = 0; i < row; i++)  
{  
for (j = 0; j < col; j++)  
{  
board[i][j] = ' ';  
}  
}  
}  
避免后续错误，先将棋盘打印出来显示  
封装打印函数 DisplayBoard(board, ROW, COL);  
申明：  
void DisplayBoard(char board[ROW][COL], int row, int col);  
使用：  
void DisplayBoard(char board[ROW][COL], int row, int col)  
{  
int i = 0;  
for (i = 0; i < row; i++)  
{  
//1.打印一行数据  
int j = 0;  
for (j=0;j<col;j++)  
{  
printf(" %c ", board[i][j]);//打印一个数字打印一个斜杠  
if (j < col - 1)  
printf("|");  
}  
printf("\n");

*////2.打印分割行*

**if** (i < row - 1)

{

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

{

printf("---");

**if** (j < col - 1)

printf("|");

}

printf("\n");

}

}

}  
棋盘布置好了之后，就轮到玩家落子和电脑落子  
你一步我一步你一步我一步，应该是个循环  
while(1)  
{  
//玩家下棋，范围依旧是3\*3  
PlayerMove( board, row, col)；  
//电脑下棋

}  
申明：void PlayerMove(char board[ROW][COL], int row, int col)  
使用：  
void PlayerMove(char board[ROW][COL], int row, int col)  
{  
int x = 0;  
int y = 0;  
//玩家输入坐标需要有地方存储，分别用x和y存储横纵坐标  
printf("玩家先走: \n");  
printf("请输入要走的下标： (例如->横坐标，纵坐标)\n");

**while** (1) {

scanf("%d,%d", &x, &y);

*//对于玩家输入的坐标还需要进行判断*

*//判断x,y的合法性*

*//从玩家的角度判断，坐标是从（1,1）开始*

*//到（3,3）结束：用户友好*

**if** (x >= 1 && x <= row && y >= 1 && y <= col)

{

**if** (board[x - 1][y - 1] == ' ')

*//判断该点是不是被占有*

{

board[x - 1][y - 1] = '\*';

**break**;

}

**else**

{

printf("该坐标被占有\n");

}

}

**else**

{

printf("坐标非法，请重新输入！\n");

}

}

}  
玩家走完之后需要打印出来让玩家看  
while (1)  
{  
//玩家下棋  
PlayerMove(board,ROW,COL);  
DisplayBoard(board, ROW, COL);  
}

再设置电脑如何落子  
void ComputerMove(char board[ROW][COL], int row, int col)  
{  
同样需要在测试函数设置时间戳，生成随机坐标  
int x = 0;  
int y = 0;  
printf("电脑走; \n");  
while (1)  
{  
//落子的范围在0-2  
x = rand() % row;//用rand模上一个行，得到0-2的随机数  
y = rand() % col;  
if (board[x][y] == ' ')//判断该坐标是否被占有  
{  
board[x][y] = '#';  
break;  
}  
}  
}  
在玩家和电脑落子之后应该立马判断玩家或电脑的输赢  
封装函数  
int IsFull(char board[ROW][COL], int row, int col)  
{  
//返回1表示棋盘满了  
//返回0表示棋盘没满  
int i = 0;  
int j = 0;  
for (i = 0; i < row; i++)  
{  
for (j = 0; j < col; j++)  
{  
if (board[i][j] == ' ')//没满  
{  
return 0;  
}  
}  
}  
return 1;//满了  
}

IsWin(char board[ROW][COL], int row, int col)  
告诉我们四种游戏状态  
玩家赢，电脑赢，游戏平局，游戏继续  
所以函数需要返回字符  
//玩家赢 '\*'  
//电脑赢 '#'  
//平局 'Q'  
//谁都没赢，游戏继续 'C'  
创建char ret=0来接受返回值  
只有ret的值不等于c，就立马break跳出判断另外三种情况  
IsWin(char board[ROW][COL], int row, int col)  
{  
//一共横三，竖三，斜三，三种情况  
int i = 0;  
//横三行  
for (i = 0; i < row; i++)  
{  
if (board[i][0] == board[i][1] && board[i][1] == board[i][2]&& board[i][1] != ' ')  
{  
return board[i][1];//通过返回元素判断是玩家赢还是电脑赢  
}  
}

*//竖三列***for** (i = 0; i < col; i++)

{

**if** (board[0][i] == board[1][i] && board[1][i] == board[2][i] && board[1][i] != ' ')

{

**return** board[1][i];

}

}**if** (board[0][0] == board[1][1] && board[1][1] == board[2][2] && board[1][1] != ' ')

{

**return** board[1][1];

}**if** (board[2][0] == board[1][1] && board[1][1] == board[0][2] && board[1][1] != ' ')

{

**return** board[1][1];

}*//判断是否平局***if** (1 == IsFull(board, ROW, COL))

{

**return** 'Q';

}**return** 'C';

}

最后在game 函数中将玩家下完一步或电脑下完一步之后的棋盘打印出来，并且输出结果  
while (1)  
{  
//玩家下棋  
PlayerMove(board,ROW,COL);  
DisplayBoard(board, ROW, COL);  
//判断玩家是否赢  
ret =IsWin(board, ROW, COL);  
if (ret != 'C')  
{  
break;  
}

*//电脑下棋*

ComputerMove(board, ROW, COL);

DisplayBoard(board, ROW, COL);

*//判断电脑是否赢*

ret =IsWin(board, ROW, COL);

**if** (ret != 'C')

{

**break**;

}

}**if** (ret == '\*')

{

printf("你赢了\n");

}**else** **if** (ret == '#')

{

printf("你输了\n");

}**else** {

}

}

gitub链接：https://github.com/daydayup-ljq/Hello-World/tree/main/game1