“程序设计基础”

课程设计报告





设计题目 一字棋游戏设计实现

姓 名 袁焕发

学 号 2019217769

专 业 物联网工程

# （一）需求和规格说明

一字棋游戏设计实现

设计实现一字棋游戏程序，人为一方，计算机为一方，人下时字符 \* 将放

在所指定的位置，而计算机下时字符 @ 将放在某一空格位置。行、列、或两对

角线有连续三个相同字符一方为胜方，也有平局情况。要求能动态演示。

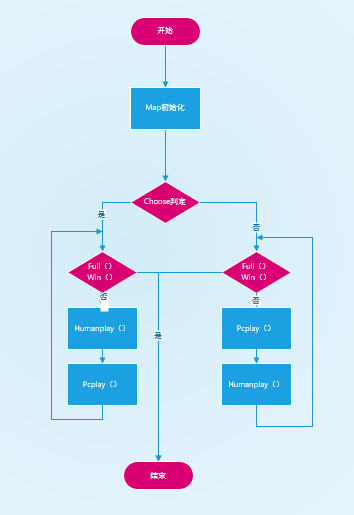
# （二）设计

### (1) 设计思想

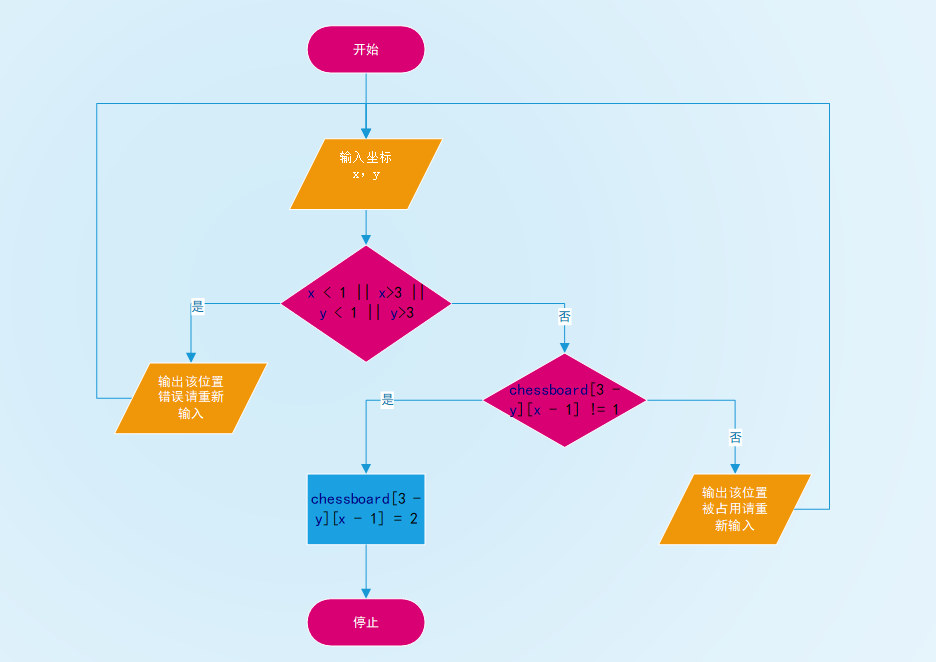
将3\*3棋盘看做是二维数组，人和电脑下棋位置分别定义不同的数值，利用横向，纵向，对角线，反对角线位置的数值之和判断是否达成胜利条件，判断棋盘是否已下满。根据棋盘上已有的棋子状态判断哪一个位置下棋赢的可能性大，哪一个位置要阻止人赢棋，根据判断结果编写电脑下棋位置。

### （2）程序框图

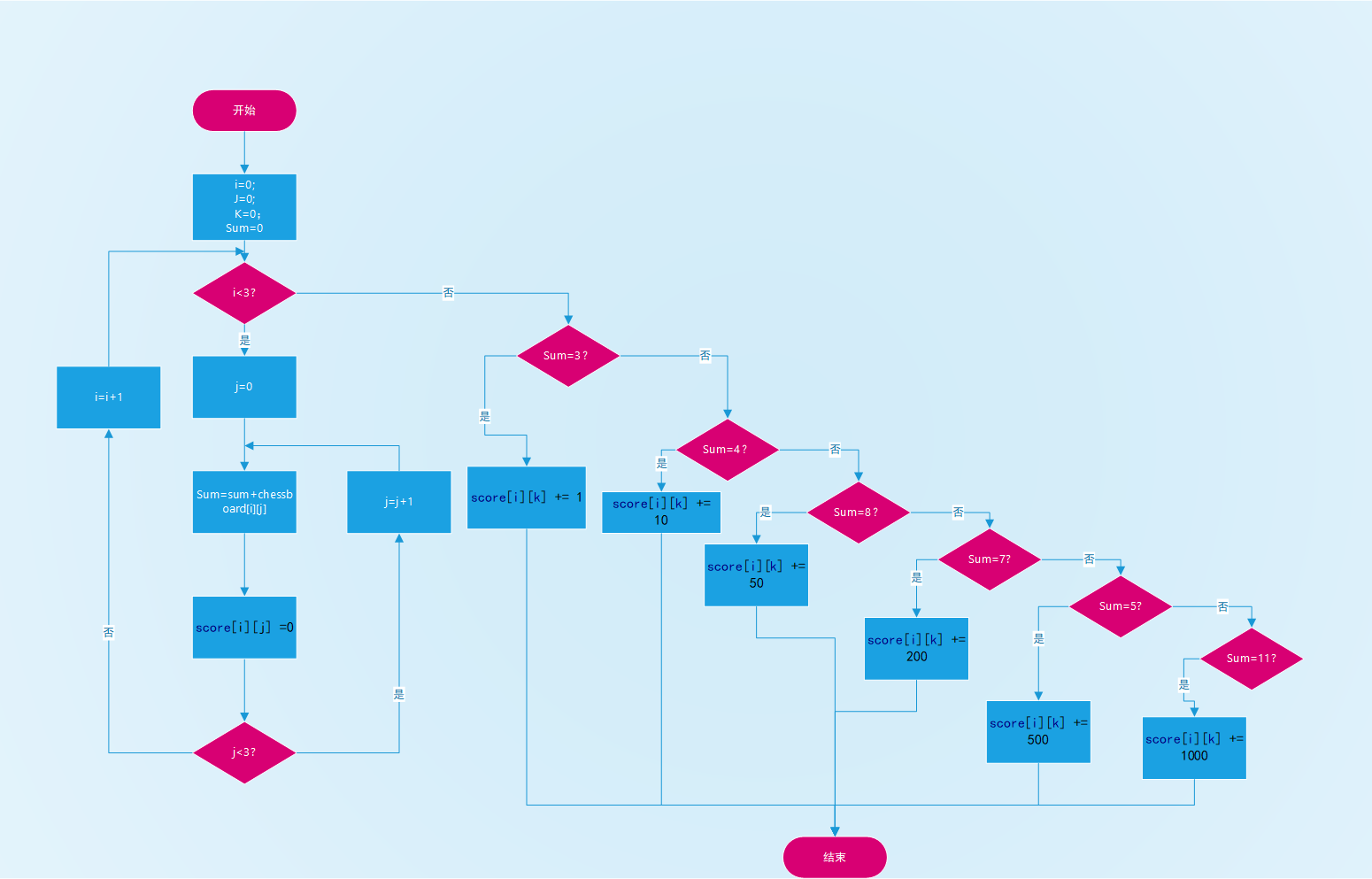
主函数

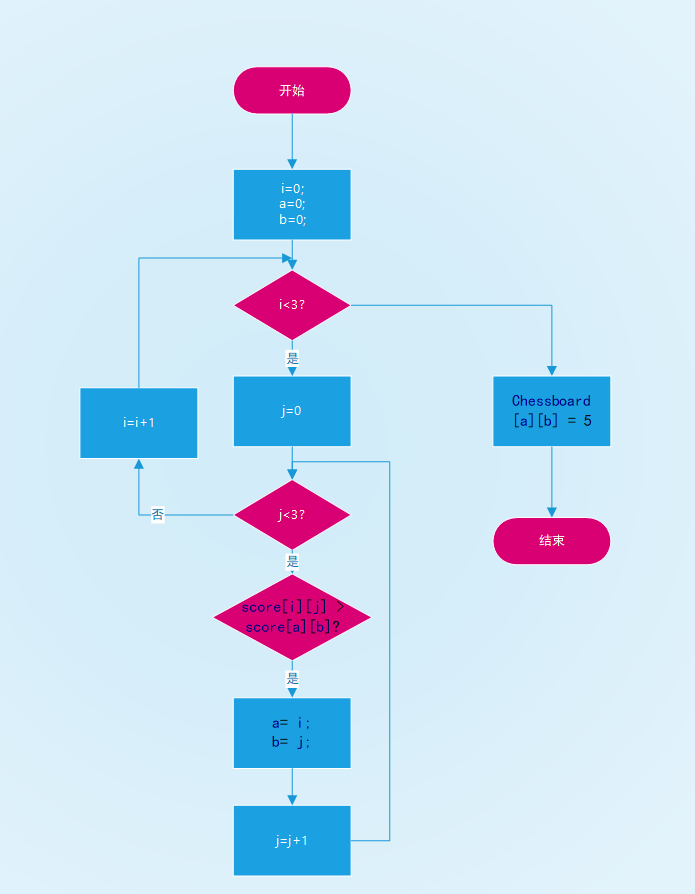


人下棋函数

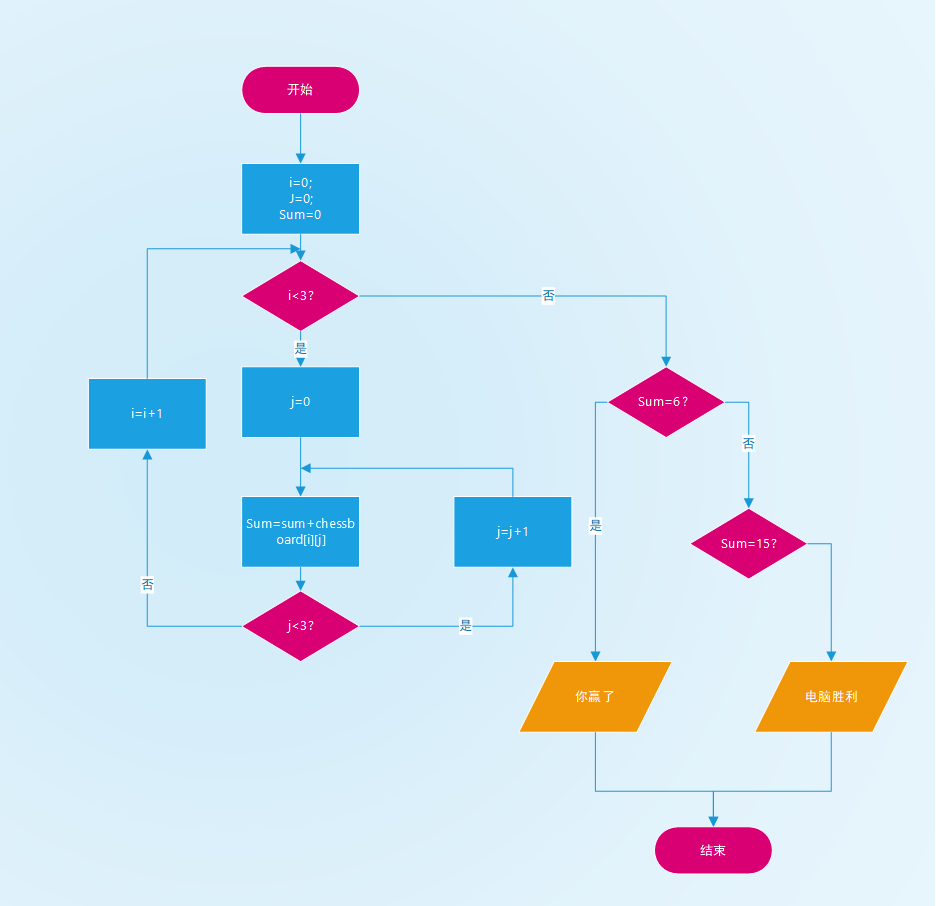


判断下棋位置重要性函数



电脑下棋函数

判断胜利函数



|  |  |  |  |
| --- | --- | --- | --- |
| 成员类别 | 类型 | 成员名 | 描述 |
| 属性 | const int | X | 常量 |
| 属性 | const int | Y | 常量 |
| 方法 | bool | Win（） | 判断谁赢了 |
| 方法 | bool | Full（） | 判断棋盘是否充满 |
| 属性 | int | Store [ ][ ] | 数组 |
| 属性 | int | Chessboard[][] | 棋盘数组 |
| 属性 | bool | choose | 判断人机先后顺序 |
| 方法 | void | map（） | 初始化棋盘 |
| 方法 | void | Showmap（） | 打印棋盘 |
| 方法 | void | Humanplay（） | 人下棋 |
| 方法 | void | Pcplay（） | 电脑下棋 |

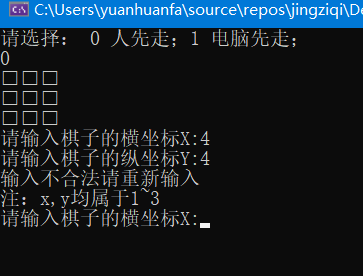
### （三）用户手册

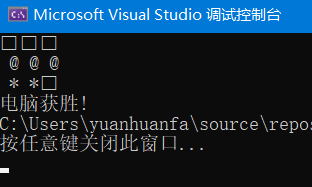
程序运行时提示用户选择下棋先后顺序，按照坐标位置选择下棋顺序，选择位置不正确会提示用户。

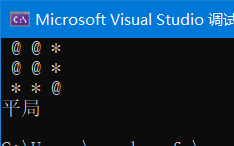
### （四）调试及测试

针对下棋先后顺序对人和电脑下棋程序先后进行调整，同时保证棋盘下满。无人胜出时输出平局。

### （五） 运行实例







### （六）心得体会

通过一字棋游戏制作，学会了进一步使用二维数组，以及对二维数组中的每个值进行求和排序等操作，对不同的情况进行分析排列，找出电脑最好的下棋位置，增加了游戏难度。对于bool函数的使用更加熟练.

### （七）对课程设计的建议

缺少对于某些题目的进一步解释以及引导，无法直接下手，没办法直接想到有一些没有接触过的算法

### 附录 源程序

// jingziqi.cpp : 此文件包含 "main" 函数。程序执行将在此处开始并结束。

#include <iostream>

#include<Windows.h>

using namespace *std*;

const int X = 3;

const int Y = 3;

int chessboard[X][Y];

int score[X][Y];

void map(); //初始化

bool choose(); //选择下棋顺序

void Showmap(); //打印棋局

bool Win(); //判断是否有一方获胜

bool Full(); //判断棋盘是否为满

void PcPlay(); //电脑下棋

void HumanPlay(); //人下棋

int main()

{

map();

if (choose())

{

Showmap();

while ((!Full()) && (!Win()))

{

HumanPlay();

*system*("cls");

Showmap();

if (Win())

break;

PcPlay();

*system*("cls");

Showmap();

}

if (Full())

*cout* << "平局" << *endl*;

}

else

{

Showmap();

while ((!Full()) && (!Win()))

{

PcPlay();

*system*("cls");

Showmap();

if (Win())

break;

if (Full())

*cout* << "平局" << *endl*;

HumanPlay();

*system*("cls");

Showmap();

}

}

return 0;

}

bool choose()

{

int m;

*cout* << "请选择： 0 人先走；1 电脑先走；" << *endl*;

*cin* >> m;

if (m == 0)

{

return true;

}

else if (m == 2)

{

return false;

}

else

{

*cout* << "选择错误，请重新输入" << *endl*;

choose();

return 0;

}}

void map()

{

for (int i = 0; i < X; i++)

for (int j = 0; j < Y; j++)

chessboard[i][j] = 1;

}

void Showmap()

{

for (int i = 0; i < X; i++)

{

for (int j = 0; j < Y; j++)

{

if (chessboard[i][j] == 1) //"1"代表空

*cout*<<"□";

if (chessboard[i][j] == 2) //"2"代表人

*cout*<<*setw*(2)<<"X";

if (chessboard[i][j] == 5) //"5"代表电脑

*cout*<<*setw*(2)<<"O";

}

*cout* << *endl*;

}

}

bool Win()

{

int sum = 0;

for (int i = 0; i < X; i++) //对每行判断是否获胜

{

for (int j = 0; j < Y; j++)

sum += chessboard[i][j];

if (sum == 6)

{

*cout*<<"你赢了！"<<*endl*;

return true;

}

if (sum == 15)

{

*cout*<<"电脑获胜!";

return true;

}

sum = 0;

}

for (int j = 0; j < X; j++) //对每列判断是否获胜

{

for (int i = 0; i < Y; i++)

sum += chessboard[i][j];

if (sum == 6)

{

*cout*<<"你赢了！"<<*endl*;

return true;

}

if (sum == 15)

{

*cout*<<"电脑获胜！"<<*endl*;

return true;

}

sum = 0;

}

for (int i = 0; i < X; i++) //对对角线判断是否获胜

sum += chessboard[i][i];

if (sum == 6)

{

*cout* << "你赢了！" << *endl*;

return true;

}

if (sum == 15)

{

*cout* << "电脑获胜！" << *endl*;

return true;

}

sum = 0;

for (int i = 0; i < Y; i++) //对反对角线判断是否获胜

sum += chessboard[i][2 - i];

if (sum == 6)

{

*cout* << "你赢了！" << *endl*;

return true;

}

if (sum == 15)

{

*cout* << "电脑获胜！" << *endl*;

return true;

}

return false;

}

bool Full()

{

for (int i = 0; i < X; i++)

for (int j = 0; j < Y; j++)

if (chessboard[i][j] == 1)

return false;

return true;

}

void HumanPlay()

{

int x, y;

*cout*<<"请输入棋子的横坐标X:";

*cin* >> x;

*cout*<<"请输入棋子的纵坐标Y:";

*cin* >> y;

while (x < 1 || x>3 || y < 1 || y>3)

{

*cout*<<"输入不合法请重新输入"<<*endl*

<<"注：x,y均属于1~3"<<*endl*;

*cout*<<"请输入棋子的横坐标X:";

*cin* >> x;

*cout*<<"请输入棋子的纵坐标Y:";

*cin* >> y;

}

while (chessboard[3 - y][x - 1] != 1)

{

*cout*<<"该位置已被占用！"<<*endl*

<<"请选择重新正确的位置!"<<*endl*;

*cout*<<"请输入棋子的横坐标X:";

*cin* >> x;

*cout*<<"请输入棋子的纵坐标Y:";

*cin* >> y;

}

chessboard[3 - y][x - 1] = 2;//人下棋 的位置

}

void PcPlay()

{

int sum = 0;

for (int i = 0; i < X; i++)

for (int j = 0; j <Y; j++)

score[i][j] = 0;

// 对每行进行分数统计

for (int i = 0; i < X; i++)

{

for (int j = 0; j <Y; j++)

sum += chessboard[i][j];

switch (sum) //根据人下棋情况判断下一步的重要性

{ //根据空位判断下棋位置

case 3: //1+1+1（有三个空位）；重要性：最低

for (int k = 0; k < X; k++)

{

if (chessboard[i][k] == 1)

score[i][k] += 1;

}

break;

case 4: //1+1+2有两个空位; 重要性：次低;

for (int k = 0; k < X; k++)

{

if (chessboard[i][k] == 1)

score[i][k] += 10;

}

break;

case 8: //1+2+5;有一个空位，一个人，一个pc；重要性：较低；

for (int k = 0; k < Y; k++)

{

if (chessboard[i][k] == 1)

score[i][k] += 50;

}

break;

case 7: //1+1+5;两个空位，一个pc；重要性：较高;

for (int k = 0; k <Y; k++)

{

if (chessboard[i][k] == 1)

score[i][k] += 200; //把行列的重要性比对角线高

}

break;

case 5: //1+2+2;一个空位，两个人；重要性：次高;

for (int k = 0; k <Y; k++)

{

if (chessboard[i][k] == 1)

score[i][k] += 500;

}

break;

case 11: //1+5+5;一个空位，两个pc；重要性：最高;

for (int k = 0; k < Y; k++)

{

if (chessboard[i][k] == 1)

score[i][k] += 1000;

}

break;

}

sum = 0;

}

// 对每列进行分数统计

for (int j = 0; j <X; j++)

{

for (int i = 0; i < Y; i++)

sum += chessboard[i][j];

switch (sum)

{

case 3: //1+1+1（有三个空位）；重要性：最低

for (int k = 0; k < Y; k++)

{

if (chessboard[k][j] == 1)

score[k][j] += 1;

}

break;

case 4: //1+1+2有两个空位; 重要性：次低;

for (int k = 0; k < Y; k++)

{

if (chessboard[k][j] == 1)

score[k][j] += 10;

}

break;

case 8: //1+2+5;有一个空位，一个人，一个pc；重要性：较低，

for (int k = 0; k <Y; k++)

{

if (chessboard[k][j] == 1)

score[k][j] += 50;

}

break;

case 7: //1+1+5;两个空位，一个pc；重要性：较高;

for (int k = 0; k < Y; k++)

{

if (chessboard[k][j] == 1)

score[k][j] += 200;

}

break;

case 5: //1+2+2;一个空位，两个人；重要性：次高;

for (int k = 0; k < Y; k++)

{

if (chessboard[k][j] == 1)

score[k][j] += 500;

}

break;

case 11: //1+5+5;一个空位，两个pc；重要性：最高;

for (int k = 0; k <Y; k++)

{

if (chessboard[k][j] == 1)

score[k][j] += 1000;

}

break;

}

sum = 0;

}

// 对对角线进行分数统计

for (int i = 0; i <X; i++)

sum += chessboard[i][i];

switch (sum)

{

case 3:

for (int i = 0; i < Y; i++)

{

if (chessboard[i][i] == 1)

score[i][i] += 1;

}

break;

case 4:

for (int i = 0; i <Y; i++)

{

if (chessboard[i][i] == 1)

score[i][i] += 10;

}

break;

case 8:

for (int i = 0; i < Y; i++)

{

if (chessboard[i][i] == 1)

score[i][i] += 50;

}

break;

case 7:

for (int i = 0; i < Y; i++)

{

if (chessboard[i][i] == 1)

score[i][i] += 100;

}

break;

case 5:

for (int i = 0; i <Y; i++)

{

if (chessboard[i][i] == 1)

score[i][i] += 500;

}

break;

case 11:

for (int i = 0; i <Y; i++)

{

if (chessboard[i][i] == 1)

score[i][i] += 1000;

}

break;

}

// 对反对角线进行分数统计

sum = 0;

for (int i = 0; i < X; i++)

sum += chessboard[i][2 - i];

switch (sum)

{

case 3:

for (int i = 0; i < Y; i++)

{

if (chessboard[i][2 - i] == 1)

score[i][2 - i] += 1;

}

break;

case 4:

for (int i = 0; i <Y; i++)

{

if (chessboard[i][2 - i] == 1)

score[i][2 - i] += 10;

}

break;

case 8:

for (int i = 0; i <Y; i++)

{

if (chessboard[i][2 - i] == 1)

score[i][2 - i] += 50;

}

break;

case 7:

for (int i = 0; i < Y; i++)

{

if (chessboard[i][2 - i] == 1)

score[i][2 - i] += 100;

}

break;

case 5:

for (int i = 0; i < Y; i++)

{

if (chessboard[i][2 - i] == 1)

score[i][2 - i] += 500;

}

break;

case 11:

for (int i = 0; i < Y; i++)

{

if (chessboard[i][2 - i] == 1)

score[i][2 - i] += 1000;

}

break;

}

//根据重要性选择下棋位置

int a= 0, b= 0;

for (int i = 0; i < X; i++)

for (int j = 0; j < Y; j++)

{

if (score[i][j] > score[a][b]) //遍历找出重要性最高的下棋点

{

a= i;

b= j;

}

}

chessboard[a][b] = 5;//电脑下棋位置

}