实验四

实现一字棋：

代码：

#include<iostream>

using namespace std;

int num=0; //记录棋盘上棋子的个数

int p,q;

int tmpQP[3][3]; //表示棋盘数据的临时数组，其中的元素0表示该格为空，

int cur[3][3]; //存储当前棋盘的状态

const int depth=3; //搜索树的最大深度

void Init() //初始化棋盘状态

{

for(int i=0;i<3;i++)

for(int j=0;j<3;j++)

{

cur[i][j]=0;

}

}

void PrintQP() //打印棋盘当前状态

{

for(int i=0;i<3;i++)

{

for(int j=0;j<3;j++)

cout<<cur[i][j]<<'\t';

cout<<endl;

}

}

void UserInput()//用户通过此函数来输入落子的位置，比如：用户输入31，则表示用户在第3行第1列落子。

{

int pos,x,y;

L1: cout<<"Please input your qizi (xy): ";

cin>>pos;

x=pos/10,y=pos%10;

if(x>0&&x<4&&y>0&&y<4&&cur[x-1][y-1]==0)

{

cur[x-1][y-1]=-1;

}

else

{

cout<<"Input Error!";

goto L1;

}

}

int CheckWin() //检查是否有一方赢棋（返回 0：没有任何一方赢；1：计算机赢；-1：人赢）

{ //该方法没有判断平局

for(int i=0;i<3;i++)

{

if(cur[i][0]==1&&cur[i][1]==1&&cur[i][2]==1)

{

return 1;

}

if(cur[i][0]==-1&&cur[i][1]==-1&&cur[i][2]==-1)

{

return -1;

}

}

for(i=0;i<3;i++)

{

if(cur[0][i]==1&&cur[1][i]==1&&cur[2][i]==1)

{

return 1;

}

if(cur[0][i]==-1&&cur[1][i]==-1&&cur[2][i]==-1)

{

return -1;

}

}

if((cur[0][0]==1&&cur[1][1]==1&&cur[2][2]==1)||(cur[2][0]==1&&cur[1][1]==1&&cur[0][2]==1))

{

return 1;

}

if((cur[0][0]==-1&&cur[1][1]==-1&&cur[2][2]==-1)||(cur[2][0]==-1&&cur[1][1]==-1&&cur[0][2]==-1))

{

return -1;

}

return 0;

}

int value()//评估当前棋盘状态的值（同时可以用p或q判断是否平局）

{

p=0;

q=0;

for(int i=0;i<3;i++) //计算机一方

{ //将棋盘中的空格填满自己的棋子，既将棋盘数组中的0变为1

for(int j=0;j<3;j++)

{

if(cur[i][j]==0)

{

tmpQP[i][j]=1;

}

else

{

tmpQP[i][j]=cur[i][j];

}

}

}

for(i=0;i<3;i++) //计算共有多少连成3个1的行

{

p+=(tmpQP[i][0]+tmpQP[i][1]+tmpQP[i][2])/3;

}

for(i=0;i<3;i++) //计算共有多少连成3个1的列

{

p+=(tmpQP[0][i]+tmpQP[1][i]+tmpQP[2][i])/3;

}

p+=(tmpQP[0][0]+tmpQP[1][1]+tmpQP[2][2])/3;//计算共有多少连成3个1的对角线

p+=(tmpQP[2][0]+tmpQP[1][1]+tmpQP[0][2])/3;

for(i=0;i<3;i++) //人一方

{ //将棋盘中的空格填满自己的棋子，既将棋盘数组中的0变为-1

for(int j=0;j<3;j++)

{

if(cur[i][j]==0)

{

tmpQP[i][j]=-1;

}

else

{

tmpQP[i][j]=cur[i][j];

}

}

}

for(i=0;i<3;i++) //计算共有多少连成3个-1的行

{

q+=(tmpQP[i][0]+tmpQP[i][1]+tmpQP[i][2])/3;

}

for(i=0;i<3;i++) //计算共有多少连成3个1的列

{

q+=(tmpQP[0][i]+tmpQP[1][i]+tmpQP[2][i])/3;

}

q+=(tmpQP[0][0]+tmpQP[1][1]+tmpQP[2][2])/3;//计算共有多少连成3个1的对角线

q+=(tmpQP[2][0]+tmpQP[1][1]+tmpQP[0][2])/3;

return p+q; //返回评估出的棋盘状态的值

}

int cut(int &val,int dep,bool max)//主算法部分，实现a-B剪枝的算法，val为上一层的估计值，dep为搜索深度，max记录上一层是否为极大层

{

if(dep==depth||dep+num==9) //如果搜索深度达到最大深度，或者深度加上当前棋子数已经达到9，就直接调用估计函数

{

return value();

}

int i,j,flag,temp; //flag记录本层的极值，temp记录下层求得的估计值

bool out=false; //out记录是否剪枝，初始为false

/\*if(CheckWin()==1) //如果计算机赢了，就置上一层的估计值为无穷（用很大的值代表无穷）

{

val=10000;

return 0;

}\*/

if(max) //如果上一层是极大层，本层则需要是极小层，记录flag为无穷大；反之，则为记录为负无穷大

{

flag=10000; //flag记录本层节点的极值

}

else

{

flag=-10000;

}

for(i=0;i<3 && !out;i++) //双重循环，遍历棋盘所有位置

{

for(j=0;j<3 && !out;j++)

{

if(cur[i][j]==0) //如果该位置上没有棋子

{

if(max) //并且上一层为极大层，即本层为极小层,轮到用户玩家走了。

{

cur[i][j]=-1; //该位置填上用户玩家棋子

if(CheckWin()==-1) //如果用户玩家赢了

{

temp=-10000; //置棋盘估计值为负无穷

}

else

{

temp=cut(flag,dep+1,!max); //否则继续调用a-B剪枝函数

}

if(temp<flag) //如果下一步棋盘的估计值小于本层节点的极值，则置本层极值为更小者

{

flag=temp;

}

if(flag<=val) //如果本层极值已小于上一层的估计值，则不需搜索下去，剪枝

{

out=true;

}

}

else //如果上一层为极小层，即本层为极大层,轮到计算机走了。

{

cur[i][j]=1; //该位置填上计算机棋子

if(CheckWin()==1) //如果计算机赢了

{

temp=10000; //置棋盘估计值为无穷

}

else

{

temp=cut(flag,dep+1,!max);//否则继续调用a-B剪枝函数

}

if(temp>flag)

{

flag=temp;

}

if(flag>=val)

{

out=true;

}

}

cur[i][j]=0; //把模拟下的一步棋还原，回溯

}

}

}

if(max) //根据上一层是否为极大层，用本层的极值修改上一层的估计值

{

if(flag>val)

{

val=flag;

}

}

else

{

if(flag<val)

{

val=flag;

}

}

return flag; //函数返回的是本层的极值

}

int main()//主程序

{

int m=-10000,val=-10000,dep=1; //m用来存放最大的val

int x\_pos,y\_pos; //记录最佳走步的坐标

Init();

cout<<"Qipan: "<<endl;

PrintQP();

char IsFirst;

cout<<"Do you want do first?(y/n)";

cin>>IsFirst;

while(IsFirst!='y'&&IsFirst!='n')

{

cout<<"ERROR!"<<"Do you want do first?(y/n)";

cin>>IsFirst;

}

if(IsFirst=='n')//---------------------------------计算机先走--------------------------------------

{

L5: for(int x=0;x<3;x++)

{

for(int y=0;y<3;y++)

{

if(cur[x][y]==0)

{

cur[x][y]=1;

cut(val,dep,1);//计算机试探的走一步棋，棋盘状态改变了，在该状态下计算出深度为dep-1的棋盘状态估计值val

if(CheckWin()==1)

{

cout<<"The computer put the qizi at:"<<x+1<<y+1<<endl;

PrintQP();

cout<<"The computer WIN! GAME OVER."<<endl;

return 0;

}

if(val>m) //m要记录通过试探求得的棋盘状态的最大估计值

{

m=val;

x\_pos=x;y\_pos=y;

}

val=-10000;

cur[x][y]=0;

}

}

}

cur[x\_pos][y\_pos]=1;

val=-10000;

m=-10000;

dep=1;

cout<<"The computer put the qizi at:"<<x\_pos+1<<y\_pos+1<<endl;

PrintQP();

cout<<endl;

num++;

value();

if(p==0)

{

cout<<"DOWN GAME!"<<endl;

return 0;

}

UserInput(); //玩家走一步棋

PrintQP();

cout<<endl;

num++;

value();

if(p==0)

{

cout<<"DOWN GAME!"<<endl;

return 0;

}

if(CheckWin()==-1)

{

cout<<"Conguatulations! You Win! GAME OVER."<<endl;

return 0;

}

goto L5;

}

else //--------------------------------人先走-----------------------------------

{

L4: UserInput();

PrintQP();

cout<<endl;

num++;

value();

if(q==0)

{

cout<<"DOWN GAME!"<<endl;

return 0;

}

if (CheckWin()==-1)

{

cout<<"You Win! GAME OVER."<<endl;

return 0;

}

for(int x=0;x<3;x++)

{

for(int y=0;y<3;y++)

{

if(cur[x][y]==0)

{

cur[x][y]=1;

cut(val,dep,1);

if(CheckWin()==1)

{

cout<<"The computer put the qizi at:"<<x+1<<y+1<<endl;

PrintQP();

cout<<"The computer WIN! GAME OVER."<<endl;

return 0;

}

if(val>m)

{

m=val;

x\_pos=x;y\_pos=y;

}

val=-10000;

cur[x][y]=0;

}

}

}

cur[x\_pos][y\_pos]=1;

val=-10000;

m=-10000;

dep=1;

cout<<"The computer put the qizi at:"<<x\_pos+1<<y\_pos+1<<endl;

PrintQP();

cout<<endl;

num++;

value();

if(q==0)

{

cout<<"DOWN GAME!"<<endl;

return 0;

}

goto L4;

}

return 0;

}

运行结果：

