**极小-极大搜索**

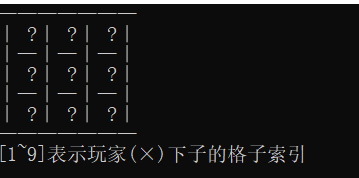
前面的六次实验课，我基本把所有基础一点儿的或者难一点的算法都一一实现了，相当于是利用了所有实验课夯实了一下我算法的基础和提高了一下自己代码实现的能力，感觉收获良多，写了很多的实验报告，再其中反应了我的学习中的心得，再这个过程中我感觉到自己提高了很多。

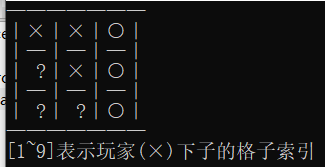
这次是最后一次实验课，我选择了一个更多的偏向于实践的代码，那就是再棋盘上的极小极大搜索算法，我是写的一个井字棋的程序，在这个程序中玩家可以和计算机对战，让电脑有了最基础的智能，但实际上就是在计算机下棋之前，它将所有情况进行搜索，然后根据我给它设定的规则，执行最有利于计算机的那一个步骤。

其实这个算法理论上还可以应用在五子棋中，但是五子棋的棋盘太大，需要落的子太多，导致搜索的情况太多，计算机不能再可以接受的时间中算出所有的情况并且进行比较，所以对于五子棋的程序一般利于更加高效，但是思想更加复杂的算法。

井字棋只有9个格子，它的博弈树可以完全展开，我们根本不需要去考虑减枝，夜不语要去设置搜索的深度，所以实现起来就比较简单。

**运行截图：**





**代码**：~~(代码实现中还有个小bug，就是再运行过程中计算机可能会输，我实在没查出来，哪里的条件设置又问题)~~

#include <Windows.h>

#include <iostream>

#include <vector>

using namespace std;

int Max(int &xiazhiPos);

int Min(int &xiazhiPos);

#define ply 3//X

#define com 4//O

//定义棋盘数组，1，2表示横竖线，0表示棋子

int qiju\_ui[7][7]=

{

1,1,1,1,1,1,1,

2,0,2,0,2,0,2,

2,1,2,1,2,1,2,

2,0,2,0,2,0,2,

2,1,2,1,2,1,2,

2,0,2,0,2,0,2,

1,1,1,1,1,1,1,

};

//定义棋局数组，九个格子

int qiju[3][3]=

{

0,0,0,

0,0,0,

0,0,0,

};

//用于判断棋子在横、竖、斜方向上是否连成3子的数组

int xydir[4][2]={

1,0,

1,1,

0,1,

-1,1,

};

//用于存放棋子坐标及方向的结构体

struct Node

{

int x;

int y;

int dir\_x;

int dir\_y;

int type;

Node(int x,int y,int dir\_x,int dir\_y,int type){

this->x=x;

this->y=y;

this->dir\_x=dir\_x;

this->dir\_y=dir\_y;

this->type=type;

}

};

//将着子存入数组中，当数组长度达到3，表明已连成3子

vector<Node> buf;

//双方轮流下子的变量，0表示X下子，1表示O下子，游戏中X是先手

bool xiazhi=0;

//绘制函数

void draw()

{

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

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

qiju\_ui[i\*2+1][j\*2+1]=qiju[i][j];

}

}

for (int i=0;i<7;i++){

for (int j=0;j<7;j++){

if (qiju\_ui[i][j]==1){

cout<<"—";

}

else if (qiju\_ui[i][j]==2){

cout<<"｜";

}

else if (qiju\_ui[i][j]==ply){

cout<<"×";

}

else if (qiju\_ui[i][j]==com){

cout<<"○";

}

else{

cout<<" ?";

}

}

cout<<endl;

}

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//判断棋子是否连成3子?

void three\_bix(int x,int y,int dir\_x,int dir\_y,int type,bool &over,bool first)

{

if (first==true){

if (qiju[y][x]==ply || qiju[y][x]==com){

for (int i=0;!over && i<4;i++){

buf.clear();

buf.push\_back(Node(x,y,dir\_x,dir\_y,qiju[y][x]));

three\_bix(x+xydir[i][0],y+xydir[i][1],xydir[i][0],xydir[i][1],qiju[y][x],over,false);

}

}

}

else{

if (y<0 || x<0 || y>=3 || x>=3){

return;

}

if (qiju[y][x]==0){

return;

}

if (qiju[y][x]==type){

buf.push\_back(Node(x,y,dir\_x,dir\_y,type));

if (buf.size()==3){

over=true;

return;

}

else{

three\_bix(x+dir\_x,y+dir\_y,dir\_x,dir\_y,type,over,false);

}

}

}

}

//qi\_over函数就是评价函数，返回值为真时，表示已经搜索到叶子节点，棋局结束，并返回评价值，反之，继续搜索

bool qi\_over(int &result)//-1:○赢 ? 0:平局 ? 1:×赢 ??

{

bool full=true;

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

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

bool over=false;

three\_bix(j,i,0,0,0,over,true);

if (over==true){

if (buf.front().type==com){

result=-1;

return true;

}

else if (buf.front().type==ply){

result=1;

return true;

}

}

else{

if (qiju[i][j]==0){

full=false;

}

}

}

}

if (full){

result=0;

return true;

}

return false;

}////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//以下四个函数是玩家与电脑下子和撤消下子

void player\_xiazhi(int pos) //pos从1开始

{

int x=(pos-1)%3;

int y=(pos-1)/3;

qiju[y][x]=ply;

}

void player\_back(int pos)

{

int x=(pos-1)%3;

int y=(pos-1)/3;

qiju[y][x]=0;

}

void com\_xiazhi(int pos)

{

int x=(pos-1)%3;

int y=(pos-1)/3;

qiju[y][x]=com;

}

void com\_back(int pos)

{

int x=(pos-1)%3;

int y=(pos-1)/3;

qiju[y][x]=0;

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

//主要的核心算法，极小极大算法

int Max(int &xiazhiPos)

{

int best = -999;

int result;

if (qi\_over(result)==true){//棋局结束【×赢,○赢,平局】

return result;

}

for (int i=0;i<9;i++){

if (qiju[i/3][i%3]==0){

player\_xiazhi(i+1);

int val = Min(xiazhiPos);

player\_back(i+1);

if (val>best){

best = val;

xiazhiPos=i+1;

}

if (val==1){

break;

}

}

}

return best;

}

int Min(int &xiazhiPos)

{

int best = 999;

int result;

if (qi\_over(result)==true){//棋局结束【×赢,○赢,平局】

return result;

}

for (int i=0;i<9;i++){

if (qiju[i/3][i%3]==0){

com\_xiazhi(i+1);

int val = Max(xiazhiPos);

com\_back(i+1);

if (val < best){

best = val;

xiazhiPos=i+1;

}

if (val==-1){

break;

}

}

}

return best;

}

int MinMax(int &xiazhiPos)

{

if (xiazhi == 0){ //X先手，我们以X来返回评价值，X取最大

return Max(xiazhiPos);

}

else{//O取最小

return Min(xiazhiPos);

}

}////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

int main(int argc, \_TCHAR\* argv[])

{

while(1){

system("cls");

draw();

cout<<"[1~9]表示玩家(×)下子的格子索引"<<endl;

if (xiazhi==0){//玩家

int pos;

cin>>pos;

player\_xiazhi(pos);

}

else{

int xiazhiPos;

MinMax(xiazhiPos);

com\_xiazhi(xiazhiPos);

}

xiazhi=!xiazhi; //轮流下子

}

return 0;

}