# 实验报告

### 实验目的：

熟悉博弈树的算法

### 实验内容：

利用博弈树实现一字棋游戏

### 源代码：

#include <iostream>

#include <windows.h>

#include <conio.h>

#include <string>

#include <ctime>

using namespace std;

#define MAX\_NUM 1000 //计算机获胜的标志

#define NO\_BLANK -1001 //人获胜的标志

#define TREE\_DEPTH 3 //递归深度

#define NIL 1001 //根节点的函数走步评估值

void show\_meun();

class State //棋盘状态节点，一个State实例就是一个棋盘的状态节点，从而形成一颗树状结构

{

public:

int QP[3][3]; //当前棋盘数组

int e\_fun; //评分结果

int child[9]; //当前棋盘状态下的后一步的所有状态节点

int parent; //当前棋盘状态下的父母节点下标

int bestChild;//在child[9]里e\_fun最优的节点下标

};

class Tic

{

public:

int tmpQP[3][3]; //用于3层递归的临时棋盘

static int s\_count;//叶子节点的静态总数

State States[MAX\_NUM];//棋盘状态节点数组

Tic(){}

void init() //初始化棋盘，将各个位置的棋盘都置为0

{

s\_count=0;

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

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

{

States[0].QP[i][j] = 0;

}

States[0].parent = NIL;

}

void PrintQP()//棋盘界面显示

{

system("cls");

cout<<endl<<endl<<endl;

cout<<"一字棋小游戏\*"<<endl<<endl;

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

{

cout<<" ";

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

{

if(States[0].QP[i][j]==1)

cout<<"1"<<'\t';

else if(States[0].QP[i][j]==-1)

cout<<"-1"<<'\t';

else

cout<<"0"<<'\t';

}

if(i==1)

cout<<" 您： -1";

if(i==2)

cout<<" 电脑：1";

cout<<endl;

cout<<endl;

}

}

int IsWin(State s) //判断当前的棋盘状态是否有令任何一方获胜

{

int i;

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

{

if(s.QP[i][0]==1&&s.QP[i][1]==1&&s.QP[i][2]==1) return 1;

if(s.QP[i][0]==-1&&s.QP[i][1]==-1&&s.QP[i][2]==-1) return -1;

}

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

{

if(s.QP[0][i]==1&&s.QP[1][i]==1&&s.QP[2][i]==1) return 1;

if(s.QP[0][i]==-1&&s.QP[1][i]==-1&&s.QP[2][i]==-1) return -1;

}

if((s.QP[0][0]==1&&s.QP[1][1]==1&&s.QP[2][2]==1)||(s.QP[2][0]==1&&s.QP[1][1]==1&&s.QP[0][2]==1)) return 1;

if((s.QP[0][0]==-1&&s.QP[1][1]==-1&&s.QP[2][2]==-1)||(s.QP[2][0]==-1&&s.QP[1][1]==-1&&s.QP[0][2]==-1)) return -1;

return 0;

}

int e\_fun(State s)//机器智能判定评价函数

{

bool flag=true;

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

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

if(s.QP[i][j]==0) flag=false;

if(flag) return NO\_BLANK;

if(IsWin(s)==-1) return -MAX\_NUM;

if(IsWin(s)==1) return MAX\_NUM;

int count=0;

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

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

if(s.QP[i][j]==0)tmpQP[i][j]=1;

else tmpQP[i][j]=s.QP[i][j];

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

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

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

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

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

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

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

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

if(s.QP[i][j]==0)tmpQP[i][j]=-1;

else tmpQP[i][j]=s.QP[i][j];

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

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

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

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

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

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

return count;

}

bool AutoDone()

{

return false;

}

void UserInput()//获取用户的输入

{

int pos,x,y;

L1:cout<<"请输入棋子的坐标 (xy)(例如第1行第2列输入12): ";

cin>>pos;

if(cin.fail())

{

cin.clear();

cin.ignore();

cin.sync();

system("cls");

cout<<"数据读入错误。"<<endl;

PrintQP();

goto L1;

}

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

if(x>0&&x<4&&y>0&&y<4&&States[0].QP[x-1][y-1]==0)

States[0].QP[x-1][y-1]=-1;

else

{

cout<<"数据读入错误。";

goto L1;

}

}

};

int Tic::s\_count = 0;//初始化棋盘状态节点总数，刚开始置为0

class demo : public Tic

{

public:

demo(){}

bool Judge()

{

int i,j,a=0;

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

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

if(States[0].QP[i][j]==0) a++;

if(a==0)

return true;

return false;

}

bool AutoDone()

{

int a,b,i,j,m,n,max,min,x,y;

if(IsWin(States[0])==-1)

{

cout<<"恭喜您获胜！"<<endl<<endl;

return true;system("pause");

}

a=0,b=0;

max=-10000;

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

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

States[11].QP[x][y]=States[0].QP[x][y];

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

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

{

if(States[0].QP[i][j]==0)

{ a=1;

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

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

States[a].QP[x][y]=States[0].QP[x][y];

States[a].QP[i][j]=1;

min=10000;

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

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

{

if(States[a].QP[m][n]==0)

{ b=1;

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

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

States[10].QP[x][y]=States[a].QP[x][y];

States[10].QP[m][n]=-1;

States[10].e\_fun=e\_fun(States[10]);

if(States[10].e\_fun<min) min=States[10].e\_fun;

}

}

States[a].e\_fun=min;

if(States[a].e\_fun>max)

{

max=States[a].e\_fun;

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

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

States[11].QP[x][y]=States[a].QP[x][y];

}

}

}

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

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

States[0].QP[x][y]=States[11].QP[x][y];

//计算机走棋

PrintQP();

if(IsWin(States[0])==1)

{

cout<<"抱歉你输了！"<<endl<<endl;

return true;system("pause");

}

else if(IsWin(States[0])==-1)

{

cout<<"恭喜您获胜！ "<<endl<<endl;

return true;system("pause");

}

return false;

}

};

void show\_meun() //显示主菜单

{

system("cls");

cout<<endl<<endl;

cout<<"欢迎进入一字棋游戏"<<endl;

cout<<"游戏规则:"<<endl;

cout<<"1.棋盘共九个方格"<<endl;

cout<<"2.行、列、或两对角线有连续三个相同字符一方为胜方"<<endl;

cout<<"3.人为一方，计算机为一方，计算机下时字符 1,人下时字符 -1"<<endl;

cout<<endl;

cout<<"1-开始游戏 0-结束游戏"<<endl;

cout<<endl;

}

int main()

{

char IsFirst;

bool IsFinish;

int choice=-1;//菜单选择

do

{

show\_meun();

cout<<endl<<"请输入数字选择操作: ";

cin>>choice;

if(cin.fail())

{

cin.clear();

cin.ignore();

cin.sync();

system("cls");

cout<<"数据读入类型错误。"<<endl;

choice=-1;

continue;

}

switch(choice)

{

case 0:

break;

case 1:

{

cout<<"若您为先手，请输入'y'，否则输入其他:"<<endl;

cin>>IsFirst;

demo \*p=new demo();

p->init();

cout<<"棋盘的初始状态:"<<endl;

p->PrintQP();

do

{

if(!p->Judge())

{

if(IsFirst=='y')

{

p->UserInput();p->PrintQP();

if(!p->Judge())

IsFinish=p->AutoDone();

}

else

{ IsFinish=p->AutoDone();

if(!p->Judge())

{ if(!IsFinish)

{p->UserInput();p->PrintQP();}}

}

}

if(p->Judge()) IsFinish=true;

}while (!IsFinish);

if((p->IsWin(p->States[0])==0)&&p->Judge())

cout<<"平局。。。。。。"<<endl;system("pause");

break;

}

default:

cout<<"错误：操作选择错误，请重新选择！"<<endl;

break;

}

}while(choice!=0);

return 0;

}

### 截图：







