**《人工智能原理与方法》**

**实验报告**

**实验三：一字棋问题的求解**

**（极大极小分析法）**

**专业班级：12级计算机二班**

**学号：1240069 姓名：李伟康**

**实验地点： 理工楼913**

**实验时间：2014年6月8日**

实验目的：

1．加深对极大极小分析方法的理解，并掌握其在具体问题中的应用。

2．复习《数据结构》课程的相关知识，学会根据具体问题设计可解的数据结构。

3. 复习C语言的相关知识，提升自己的编程能力以及解决具体问题的能力。

实验语言环境：

1. VC++6.0（英文版）
2. 通过C语言实现

一字棋问题的求解（具体算法参考课堂ppt内容）：

此项目中所使用的极大极小分析方法均是站在用户角度进行设计的，用户在每次落子时程序会给出最优走法。项目中电脑的智能性较低。

1. **一字棋的极大极小分析法说明**

设棋局为P，估价函数为e(P).

（1）若P是A必胜的棋局，则e(P)=+∞.

（2）若P是B必胜的棋局，则e(P)= –∞.

（3）若P是胜负未定的棋局，则e(P)= e(+P)- e(-P)

其中

e(+P)表示棋局P上有可能使×成为三子一线的数目。

e(-P) 表示棋局P上有可能使○成为三子一线的数目。

1. **数据结构设计与分析**
2. **算法实现所需数据结构分析**

极大极小分析方法需要用到OPEN表和CLOSED表，结合《数据结构》的相关知识，此处利用栈来实现OPEN表和CLOSED表的构建。

1. **数据结构具体设计**

#define maxsize 100//栈长度

//数据类型转换及顺序表构建

typedef struct{

char value[3][3];//当前格局的状态

int test;//当前格局的静态估值:A\_win-B\_win

int A\_win;//A\_win的静态估值

int B\_win;//B\_win的静态估值

int location[2];//A的落子位置

}datatype;

//顺序栈定义

typedef struct{

datatype data[maxsize+1];//栈数组

int top; //栈顶指针

}Seqstack;

OPEN表和CLOSED表均用此栈进行构建。下面以OPEN表为例说明此顺序栈的具体含义：

顺序栈中的data是一个自定义的结构体类型，每一个值表示OPEN表的一个节点。

1. **算法实现设计与分析**
2. **极大极小算法实现分析**

解决方案的设计目的是模拟人脑使用算法的具体过程。因此，除了定义“静态”数据结构外，还需要定义恰当的“动态”函数。

1. **解决方案所用函数一览**

根据函数在解决方案中的重要性，将其划分为附属函数、基本函数、主要函数，每种函数种类及详细说明如下：

**A.附属函数构建**

a.void Show\_table(datatype value)//输出栈内元素

将博弈过程中的当前格局以适当格式输出。

b.bool select()//小菜单函数构建

程序执行结束后给用户的选择功能。用户输入Y，程序继续执行；用户输入N，程序终止执行。

c.void Menu()//初始菜单构建

此函数主要为用户提供良好的用户界面，并使之熟悉游戏规则。

**B.基本函数构建**

1. Seqstack \*InitStack()//初始化栈

据此函数构建OPEN表和CLOSED表，构建后的OPEN表和CLOSED表为空。

b.void Stackempty(Seqstack \*S)//判断栈是否为空

此函数主要用来检测OPEN表为空，即搜索失败的停止。若为空（不含初始化），则表示搜索失败，退出执行程序。

c.void Push(Seqstack \*S,datatype x)//入栈

此函数主要将扩展的格局依次压入OPEN表中，这些格局在压入OPEN表之前需要根据它们的获胜概率值（站在用户角度）进行排序。

d.void Pop(Seqstack \*S,datatype &x)//出栈

此函数主要用来弹出当前检测节点，其实现和基本出栈操作类似。

e.void Check\_input(int value)//检查输入是否正确

根据一字棋问题的实际情况，落字位置的输入值只能从0-2中进行选择。若用户输入错误，程序终止执行。

f.void MpSort(datatype array[],int N)//冒泡排序

根据估价函数的值对当前节点扩展的多个格局进行排序，以满足极大极小分析算法的要求。

**C.主要函数构建**

1. bool test\_location(int h,int l)//检测落子位置是否合法

电脑或用户每次落子时需要对落子位置进行判断，若位置不合法（存在覆盖或出界问题），返回值为false;反之，返回值为true。

b.void test\_value\_A()、void test\_value\_B()//当前格局的静态估值

电脑方落子后，程序自动分析用户可能落子的每一个位置。在对每个位置的情况分析后，计算出用户和电脑在当前格局各自获胜的概率值，交由格局生成函数Caculate\_value()使用。

c.void C\_action()//电脑落子函数

此函数第一次运行时（电脑走的第一步棋）通过随机数函数rand()得到电脑方的落子位置，后续执行时，则通过试探法选择出可能的落子位置。由于此函数的实现方法如此，电脑方的智能性相当有限。

d.void Caculate\_value(int h,int l)//新格局产生

借助函数test\_value\_A()、test\_value\_B()站在用户角度对当前格局进行扩展。产生的新格局暂存于data\_temp数组中，然后交由排序函数Mpsort()处理后依次压入OPEN表。

e.void test\_win()//判断游戏的获胜方

此函数通过条件判断决定游戏的最终结果（用户胜、用户败、和棋）

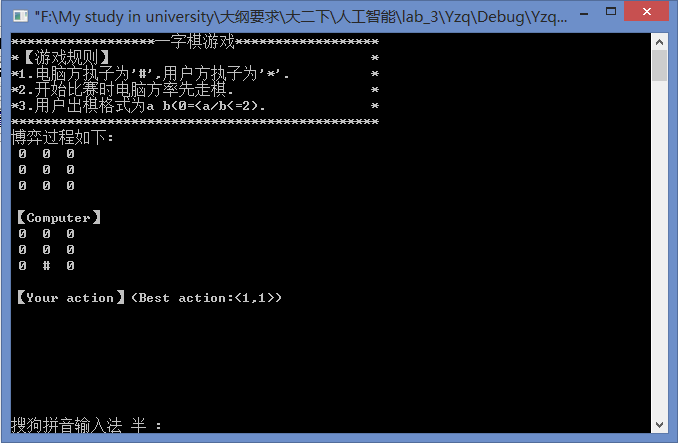
f.void Caculate\_Yzq()//一字棋问题博弈的具体过程

此函数是整个项目的总控程序，协调上述各个函数的协同运作。

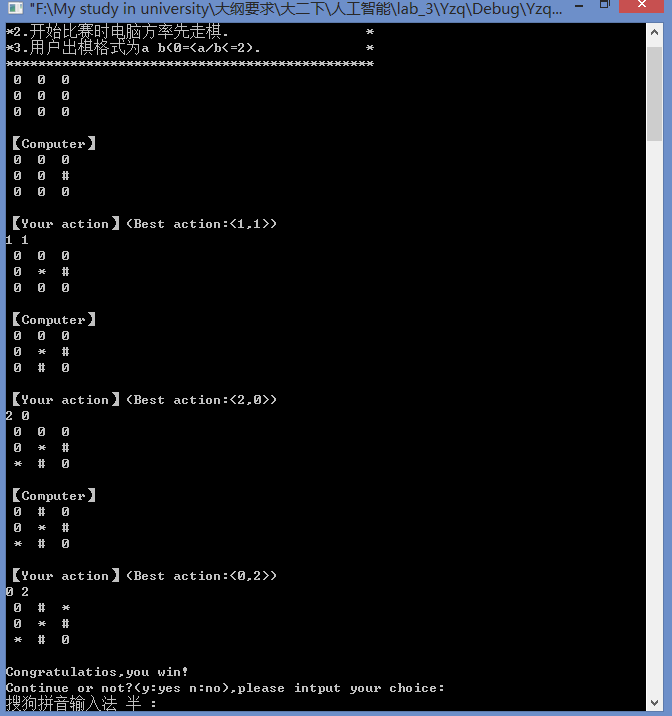
1. **解决方案整体概述**

用户输入->用户数据的初始化处理->一字棋问题的初始化->一字棋问题的具体求解->最终结果的显示

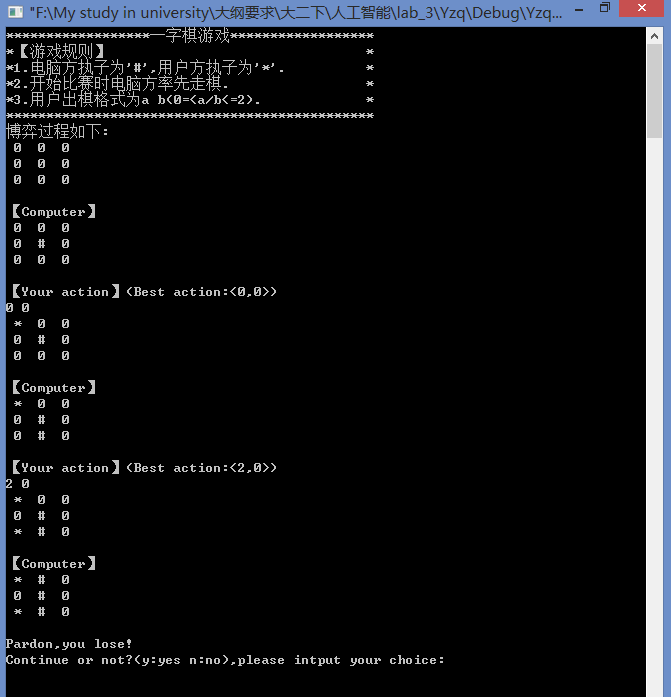
1. **程序运行结果图示说明**
2. **程序初始化界面**



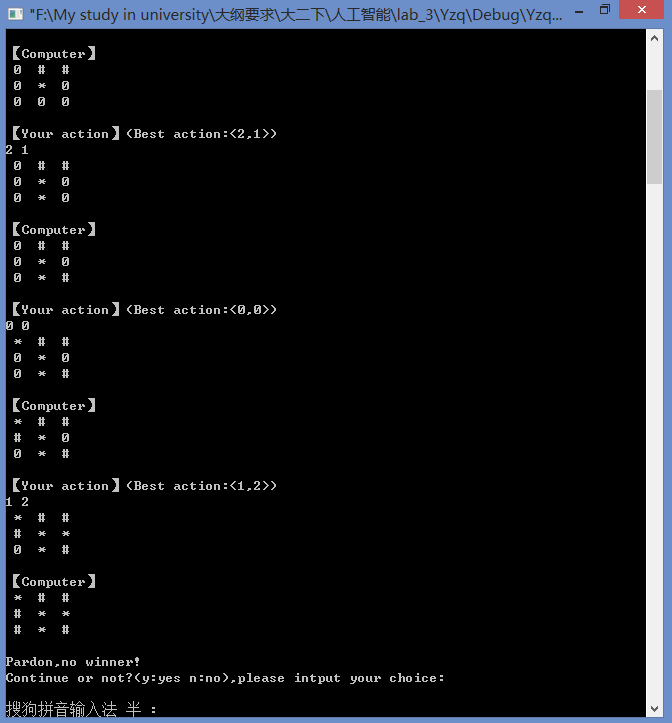
1. **用户胜棋结果**



1. **用户败棋结果**



1. **和棋结果**



分析总结：

1. **主要的工作流程**
2. 用户数据的获取与初始化
3. 一字棋问题的初始化
4. 一字棋问题的具体求解
5. 最终结果的显示
6. **出现的主要问题与解决过程**

由于有了前两次处理汉诺塔问题和八数码问题的经验，此次处理一字棋问题时几乎没有什么大的困难。即使出现的一些小问题也是自己的粗心造成的。

1. **不足**

此方案主要站在用户角度通过极大极小分析法进行博弈树的构建，并未过多考虑电脑的落子情况（仅仅保证了落子的合法性和正确性）。因此，用户在使用本程序时可能会觉得电脑的低能性。另对于在某些新扩展的格局中用户获胜的概率值相等的情况，此方案并未对其进行特殊处理（仅仅按照生成次序的先后进入OPEN表），这就导致了针对某些格局此程序给出的最优走法实际上并不是最佳选择。

1. **反思**
2. 对具体算法描述时一定要紧扣算法本身，从算法出发，切忌主观臆想。
3. 明确算法和程序的关系，编程只是为了描述相关算法。构建合适易行的算法才是解决问题的根本。
4. 加深对基础知识的理解，平时学习时应加强对基础知识的练习。
5. 注意单步调试方法的使用。
6. 应挑选特征值对设计的程序进行验证，检查程序的极限承受能力，提高程序本身的正确性和容错性。
7. 长时间进行编程时应注意进行调节，切忌目不转睛地盯着电脑屏幕，一定时间后应多走动走动，缓解下眼睛的压力。