## 程序结构设计与分析

**Q1:简要概括读入和初始化地形地图、动物地图的方法。**

**通过File导入地图文件,Scanner读入文件,将读取到的数据转化为String类型。通过一个嵌套的循环将String中的各个数据分别赋给储存地图信息的二维数组。就完成了地图初始化。**

**Q2：如果代码包含多个类，简述每个类的作用和设计思路。**

**没有多个类。**

**Q3:详细说明关键变量的变量类型及其作用。**

|  |  |  |
| --- | --- | --- |
| **变量类型** | **变量名** | **作用** |
| **二维char数组** | **AnimalMap** | **储存动物数据** |
| **二维char数组** | **map** | **储存地形数据** |
| **boolean** | **player** | **判断当前行动方** |
| **三维char数组** | **memory** | **存放游戏历史数据** |
| **string** | **command** | **读入游戏指令** |
| **int** | **line** | **要操作的动物所处的行** |
| **int** | **row** | **要操作的动物所处的列** |
| **int** | **currentStep** | **当前已走的有效步数** |
| **int** | **lastStep** | **悔棋前的有效步数** |
| **int** | **nextStep** | **悔棋或者撤销悔棋后想要达到的步数** |

**Q4:列举程序中的主要方法并阐述其功能。简述该方法改变了哪些关键变量，改变的作用是什么（如果其作用过于明显或已经提及，可以省略）。**

**1.public static void printMap(char[][] animalMap, char[][] map)**

**打印地图.**

**2. private static int undo(int currentStep, int nextStep)**

**判断是否能够悔棋且返回悔棋后到达的步数。**

**改变currentStep到悔棋后到达的步数。便于输出当前状态下的棋盘。**

**3. private static int redo(int currentStep, int lastStep, int nextStep)**

**判断是否能够撤销悔棋且返回撤销后所到的步数。**

**改变currentStep到撤销悔棋后所到的步数。便于输出当前状态下的棋盘。**

**4. public static String findPosition(boolean players, char theAnimal, char[][] animal)**

**找到玩家输入的动物对应的行列坐标以便对其进行操作。**

**5. public static int meetWater(char[][] animal, int line, int row, char direction, char[][] map, boolean players)**

**判断玩家要操作的棋子遇见水之后如何行动。**

**6. public static int whetherMove(char animal1, char animal2, char map, boolean player)**

**判断玩家正常行进时如何行动。**

**7.public static int judgingWin(char animal[][], char map[][])**

**8. public static char[][] copyArray(char[][] animal)**

**判断吃光敌方棋子后的胜利以及占领敌方兽穴后的胜利。**

**Q5:列举并分析**几个你觉得你的程序中写的好的和写的不好的地方。

写的好的地方：

1.移动棋子的时候按照方向分类,具体到各个方向来分析棋子移动的各种情况,思路比较清晰.

2.通过line和row确定了棋子所处的行列坐标,便于对棋子进行操作.

3.在移动棋子的时候把遇到水和正常行进分开写了两个函数,判断起来更清晰.

写的不好的地方：

1.代码重复比较多,开始的时候打印地图出了一些bug,于是在每一次移动棋子之后都加了一段重复的打印地图,变换玩家等等的代码,尚未优化.

2.在移动棋子方面代码还是很多很复杂,条理不清晰.

3.在有很多条件判断用了大量单独的if.显得很乱.

4.代码风格不好.至今不清楚什么时候应该空行.注释也是自己凭感觉写的.

Q6:遇到的问题.

1.读入地图的时候没有发现我的地图原文件是只有一整行的,走了些弯路

2.打印地图之后面对复杂的游戏规则无从下手.

3.基础知识不牢固.面对int类型的返回值,我在使用switch的时候用了诸如case’0’,导致不能正常判断.知道数组是从零开始计数的,却在应用的时候总忘记.

4.在比较动物大小的时候不知道怎么比较字母和数字的大小

5.采用animal[i][j]=memory[currentStep][i][j]的方式,导致在撤销悔棋的时候无法正常识别悔棋+其他输入(如help)+撤销悔棋的操作

6.对循环的知识掌握的不够好,导致有时候陷入死循环无法继续.

7.写判断条件的时候曾经把(m!=a&n!=b)与!(m==a&n==b)看成是等价的,诸如此类在判断条件上进了很多坑.

8.规则太多,没有处理好.在一些细节上没有做好,动物会莫名其妙的吃掉不应该吃的动物.或者吃不了该吃的动物.

解决的方法

1.开始的时候毫无头绪,面对复杂的游戏规则无从下手,通过请教大神才开始慢慢有思路分类写动物的各种移动.

2.思路打开之后自己一点点摸索写各种想要实现的功能.

3.看书补基础知识.关于数字和字母的大小比较,都是通过复习书上知识慢慢实现的.

4.遇到一些bug,上网查看不懂的错误的代码.自己摸索检查代码的逻辑错误,也会向同学请教.

5.在悔棋和撤销悔棋方面lab7的code review帮了大忙 .

6.有些问题想不到好的解决方案,如开始时遇到的地图不能打印的问题,就傻傻的在每次下棋之后print地图.导致代码冗长,尽管多少解决了问题.

意见与建议:在写完pj之后真的能够明显感受到自己学到了很多.编程这件事真的需要多练习.虽然老师常常会在课后布置思考题,但是总觉得过于难,自己的水平不足以应付.希望以后能有一些难度适中的练习题.