**扫雷-概要设计**

**1.1输入，输出设计**

**1.1.1-oj版输入输出**

**Input**

第一行一个整数 1 − 4 表示，表示初级、中级、高级、自定义。如果是自定义，则紧跟着三个整数分别表示行数 N，列数 M 和雷数 C。 从第二行开始的 N 行 M 列表示棋盘。"."表示没有雷，"\*"表示有雷，保证棋盘数据合法。 之后的每一行三个整数表示一个操作，操作类型 1, 2, 3, 4, 9 以及坐标 (x, y)（1 ≤ x ≤ N, 1 ≤ y ≤ M）， 表示对 x 行 y 列进行操作。

**Output**

对于每一个操作，第一行输出游戏提示。 第二行输出两个整数，操作次数和剩余地雷数量。 之后的 N 行 M 列输出棋盘，"."表示没点开的格子，"?"表示问题标记，"!"表示地雷标记，用 0 − 8 表 示已经点开的格子。 如果本次输入的操作不合法，则忽略这次操作，只增加操作次数但不会对棋盘进行改动。胜利或失败之 后需要在输出棋盘时，所有地雷所在位置输出"\*"，其余位置输出当前操作后的棋盘状态。

**1.1.2-图形化版输入输出**

待补充

**1.2-算法设计**

**1.2.1oj版**

**1，实现地雷区的揭开**

两个二维数组

Game\_board\_in;//存储地雷和雷数

Game\_board\_out;//0表示未打开，1表示打开，2表示标记

打印游戏盘函数：

for（循环输出）

for（循环输出）

{

if(Gameboard\_out[i][j]==0)表示已揭开

{

输出Game\_board\_in[i][j];

}

Else if 有特殊标记

输出特殊标记

Else

输出“.”

}

**2，用户核心操作函数**

opt为一个结构体，存储用户的操作数据，包括model(操作模式)，x,y(操作坐标)

operation（Game\_board\_in，Game\_board\_out，struct opt）

{

switch(model)

{

case:model1

操作1；

case：model2

操作2；

case:model3;

操作3；

case:model4;

操作4；

case:model9;

操作9；

}

}

操作1

if（该格未打开）

if(为非0数字)

打开该格

Else if 为0即周围3x3无雷

按序打开周围3x3，如果遇到0则再次调用此处递归

Else

操作无效

操作2，3 ，4直接改变外显数组实现标记

操作9，

if（判断操作有效）

{

按序打开周围未打开的格子

}

**1.3-高级数据结构设计**

**1.3.1-常量定义**

#define Game\_over 0;

#define Game\_In\_progress 1;

**1.3.2-全局变量定义**

Int Game\_state;//游戏状态-（感觉意义不大，后续版本考虑删除）

Int Col;//游戏盘列数

Int Row//游戏盘行数

**1.3.3-数据结构的定义**

Struct opt//存储用户的单次操作

{

Int x,y;

Int model;

}

**1.4-系统模块划分**

**1.4.1-oj版模块划分**

本系统程序划分为main.c,MineSweeper\_init.c,MineSweeper\_progress.c,MineSweeper\_judgestate.c

MineSweeper\_input.c,MineSweeper\_output.c

**1.模块名称：main.c**

模块功能简要描述：为游戏程序的主函数，主要负责接受用户的输入，对游戏进行初识化，执行用户的操作并进行合理的输出

**2.模块名称：MineSweeper\_init.c**

模块功能简要描述：游戏初始化函数，根据用户的输入创建合理的游戏盘并赋予要求的地雷位置

**3.模块名称：MineSweeper\_progress.c**

模块功能简要描述：游戏过程中的主要函数，也是核心函数，主要负责根据用户的输入处理游戏盘

**4.模块名称：MineSweeper\_judgestate.c**

模块功能简要描述：负责对游戏状态的判断，根据操作和游戏盘判断游戏的结束与否

**5.模块名称：MineSweeper\_input.c**

模块功能简要描述：负责接受用户的输入，包括游戏盘的初始化，后续的每一步操作等

**6.模块名称：MineSweeper\_output.c**

模块功能简要描述：负责输出游戏盘以及游戏状态，还有游戏的提示部分

模块结构图如下

图示

描述已自动生成

**1.4.2-文件及函数组成**

待补充

|  |  |  |  |
| --- | --- | --- | --- |
| 源文件 | 源文件说明 | 函数名 | 函数说明 |
| MineSweeper\_init.c | 负责雷区的初始化部分，包括创建游戏盘 | int\*\* Gameboard\_setup\_in(); | 建立游戏盘数组 |
| int\*\* game\_board\_in\_init()； | 游戏内盘建立与初始化 |
| int\*\* game\_board\_out\_init(); | 游戏外盘建立与初始化 |
| int MineNumber(int\*\* s, int row, int col); | 单元格周围雷数统计 |
| void Gameboard\_init(int\*\* arr); | 游戏内盘初始化 |
| MineSweeper\_progress.c | 负责游戏进程中的用户操作 | void User\_Operation(int\*\* arr\_out, int\*\* arr\_in); | 用户操作的核心函数 |
| int Operation1(int\*\* arr\_out, int\*\* arr\_in); | 执行操作1，左击 |
| int Operation2(int\*\* arr\_out, int\*\* arr\_in); | 执行操作2，右击一次，“！” |
| int Operation3(int\*\* arr\_out, int\*\* arr\_in); | 执行操作3，右击两次“？” |
| int Operation4(int\*\* arr\_out, int\*\* arr\_in); | 执行操作4，取消标记 |
| int Operation5(int\*\* arr\_out, int\*\* arr\_in); | 执行操作9，点开周围无标记单元格 |
| MineSweeper\_judgestate.c | 负责对游戏状态做出判断 | int State\_judge(int\*\* arr\_out, int\*\* arr\_in); | 对游戏状态做出判断，包括踩雷与否和排雷是否完毕 |
| int Mine\_judge(int\*\* arr\_out, int\*\* arr\_in); | 对雷数状态进行判断，判断排雷情况 |
| MineSweeper\_input.c | 负责读取用户输入，包含雷区初始位置的输入和用户操作输入等 | void Gameboard\_input(int\*\* s); | 获取雷区初始位置数据 |
| void Get\_User\_Operatian(); | 获取用户单步输入 |
| MineSweeper\_output.c | 负责实现游戏的输出部分 | void Gameboard\_debugout(int\*\* s); | 用于初期debug |
| void Gameboard\_Print(int\*\* arr\_out, int\*\* arr\_in); | 用于打印当前游戏盘 |