C++大作业设计说明书

物理系 基科31 蒋文韬 2013011717

一、类设计说明名：

本程序设计了三个类，包括整型矩阵类，由整型矩阵类继承而来的游戏矩阵类，以及英雄榜类。以下为各类的具体设计说明：

（一）整型矩阵类（class MATRIX）：

该类用于生成行数，列数可变的，由整型变量构成的矩阵，矩阵本身为一个动态二维数组。

1、类图：

|  |
| --- |
| MATRIX |
| - m\_row: int  - m\_col: int  - m\_MAT: int \*\* |
| + MATRIX(row: int = 1, col: int = 1)  + operator()(x: int, y: int)const: int  + getRow()const: int  + getCol()const: int  + initMat(row: int = 0, col: int = 0, val: int = 0): void  + printM()const: void  + changeMatItem(x: int, y: int, input: int): void  + findInMat(toFind: int = 0)const: BOOL  + countInMat(toCount: int = 0)const: int  + deleteMat(): void |

2、数据成员说明：

int m\_row 矩阵的行数

int m\_col 矩阵的列数

int \*\* m\_MAT 矩阵本身，即二元动态数组的指针

3、函数成员说明：

MATRIX(int row = 1, int col = 1) 构造函数，默认行数与列数为一，矩阵元素值为0

operator()(int x, int y)const 重载操作符()，通过输入的行数与列数获取矩阵元素值。因其他const类型的函数中会用到它因此未返回引用，而用changeMatItem函数实现元素的更改

getRow()const 获取矩阵的行数

getCol()const 获取矩阵的列数

initMat(int row = 0, int col = 0, int val = 0) 初始化急诊全体元素为val的值，可添加行与列

printM()const 打印出矩阵，测试时用

changeMatItem(int x, int y, int input) 将矩阵x, y处的元素改为input的值

operator= (MATRIX tempM) 重载操作符=为拷贝运算

findInMat(int toFind = 0)const 查找矩阵中是否有某值

countInMat(int toCount = 0)const 对矩阵中某数的个数进行计数

deleteMat() 人工释放内存

（二）游戏矩阵类（class GameMat）：

该类用于生成用于记录游戏进行状态的矩阵。与整型矩阵类相比，其多了一个记录雷数的成员，并具有更多的操作。

1、类图：

|  |
| --- |
| MATRIX |
|  |

<<Public>>

|  |
| --- |
| GameMat |
| - m\_numOfBomb: int |
| + GameMat(int row = 1, int col = 1, int numOfBomb = 0)  + getNumberOfBomb()const: int  + GenerateGM(int addBomb = 0, int currentX = 0, int currentY = 0): void  + GenerateBCMfrom(GameMat tempM): void  + printWithColor()const: void  + printWithColor(GameMat tempGM)const: void  + manipulateLD(GameMat \*, GameMat \*, int currentX = 0, int currentY = 0 ): void  + manipulateRD(GameMat \*, GameMat \*, int currentX = 0, int currentY = 0 ): void  + manipulateBothD(GameMat \*, GameMat \*, int currentX=0, int currentY=0): void  + deleteGM(): void |

2、数据成员说明：

m\_numOfBomb 雷阵中的雷数

3、函数成员说明：

GameMat(int row = 1, int col = 1, int numOfBomb = 0) 构造函数，默认行列为1，雷数为0，且有雷数不超过行列数乘积的判断

getNumberOfBomb()const 获取雷阵中的雷数

GenerateGM(int addBomb = 0, int currentX = 0, int currentY = 0) 随机生成雷阵，可以更改雷数，默认不更改，且会避免在当前鼠标点击位置出现雷

GenerateBCMfrom(GameMat tempM) 根据当前雷阵生成雷的计数矩阵

printWithColor()const 打印出可视化后的雷阵

printWithColor(GameMat tempGM)const 根据游戏进度打印玩家看到的雷阵

manipulateLD(GameMat \*, GameMat \*, int currentX = 0, int currentY = 0 )

进行鼠标左键单击的相应操作

manipulateRD(GameMat \*, GameMat \*, int currentX = 0, int currentY = 0 )

进行鼠标右键单击的相应操作

manipulateBothD(GameMat \*, GameMat \*, int currentX=0, int currentY=0)

进行鼠标左右键同时按下的相应操作(玩家实际操作为在数字上单击)

deleteGM() 人工释放内存

（三）英雄榜类（class HEROS）：

1、类图：

|  |
| --- |
| HEROS |
| - m\_heros: HeroStruct[3][3] |
| + HEROS(){}  + getName(size: int = 0, order: int = 0)const: string  + getTime(size: int = 0, order: int = 0)const: int  + insertHero(size: int, tempName: char \*, tempTime: int = 99999): void  + getAndPrintHeroList(): void  + getHeroList(): void  + writeHeroList()const: void |

2、数据成员说明：

|  |
| --- |
| HeroStruct |
| - m\_name: char[21]  - GameTime: int |
|  |

m\_heros[3][3] HeroStruct结构体的二维数组，分别为三种传统游戏规模的前三名的玩家信息。HeroStruct组成如下：

3、函数成员说明：

getName(size: int = 0, order: int = 0)const 获取该玩家的姓名

getTime(size: int = 0, order: int = 0)const 获取该玩家的游戏时间

insertHero(size: int, tempName: char \*, tempTime: int = 99999)

将该玩家插入英雄榜，未进入英雄榜则自动不进行任何操作

getAndPrintHeroList() 从文件读取高分榜并打印

getHeroList() 从文件读取高分榜

writeHeroList()const 将高分榜写入文件

二、主要技术难点与算法设计：

这个程序在程序语言本身的功能的应用上并没有用到较为复杂的知识，如模版，STL等，但也仍有一些有一定难度的地方。在程序的编程实现过程中，遇到的较大的难点主要有三个，分别是用户点击时的操作，用户鼠标信息的获取，对用户输入的限制以及文件的读写。

在处理用户左键点击的相应操作这个难点时，在有雷，或与雷相邻的格子里点击时的情况较为简单，而在四周都没有雷的情况下，玩过扫雷的玩家应该比较了解，游戏将自动探索相连通的所有四周无雷的区域。因此程序需要实现寻找满足该条件的连通区域的算法。对这种情况，我的算法是首先标记四周无雷的被点击的点，随后多次遍历雷阵中的每一个点，判断该点是否与被标记的点相邻且满足四周无雷，是则标记，循环遍历直到一次遍历后没有新的点被标记则结束。实现这个方法后，仍有一些问题，经检查与debug均为细节上的漏误，更正后程序工作正常。在继续之后代码的编写时我又想到也可以用递归的算法解决这个问题，即左键点击函数在点击的位置四周无雷时，先标记该位置已探索，然后对四周的八个位置也调用左键点击函数，递归到点击到周围有雷的格子时便自动进入周围有雷时的处理方法。于是也对这个想法进行了实现，但程序运行情况为超出迭代深度。由于之前已经有正确的实现，因此我也没有太花费时间深究，最终仍采取了前一种方法。

在获取鼠标信息时，因为这部分涉及方面较广，与windows系统的消息机制等知识相关，我了解甚少，因此采用了在网上查阅资料与参考、修改他人代码的方式。实现这部分功能的代码主要修改自网上查找到的一段70行左右的代码。该段代码能够在控制台中让光标跟踪鼠标位置，并在鼠标左键点击时结束运行。我仔细阅读了这段代码，大致理解了其含义，其大部分的代码都是用于故障处理。我保留了它的光标跟踪鼠标的功能，并找到了它用来存储鼠标当前位置的变量，将鼠标位置接收到我的变量中以供之后使用。随后，我通过在解决方案资源管理器中搜索代码中出现的FROM\_LEFT\_1ST\_BUTTON\_PRESSED，找到了其在外部文件wincon.h中的定义，也通过阅读在其中找到了我所需的RIGHTMOST\_BUTTON\_PRESSED。最终，我将这段代码修改为能将鼠标实时位置传输给我的引用参数，并在用户点击后返回左键或右键点击的信息的函数。

用户输入这一部分，为防止用户输入乱七八糟的东西造成bug，我在需要用户输入其选择或数组的部分均使用了getch()函数，一次只接收一个字符，并通过循环条件控制只接收我想要的字符。在让用户输入自定义雷阵大小时，这种方式较为麻烦，需要挨个接收用户输入的数字字符，再将其转换为十进制数字的相应值，不过实现后程序工作良好，完全保证了在接收用户输入方面的正确性。

在文件读写这部分，涉及到三种游戏规模的前三名，以及字符串的读入与输出等，较为繁琐且容易出错，也耗费了我一定的时间，暴露出了我在字符串操作方面的不足，也通过这一部分功能的实现在这方面进行了练习与提升。