# 第一周实验报告

我一开始的计划是直接先学Qt，直接进行游戏逻辑的框架搭建。但是很快发现Qt不容易快速上手，所以我决定改变方向，先设计好游戏的基本功能框架。

我又重新观看了一遍《最强大脑》的“燃烧的画”节目片段，自己大概总结了一下游戏的功能逻辑：

玩家将会对一幅画进行一系列操作（更合理的说是一个的矩阵），最终得到目标图画。

下面是详细的分析：

画：由个小方块组成的矩阵。

小方块：有颜色和数字这两个属性的小正方形，它们一起组合成了画。

操作：总共有四种：左旋转，右旋转、燃烧、以及自己添加的回撤。

左旋转：对整幅画进行90度的逆时针方向旋转。

右旋转：对整幅画进行90度的顺时针方向旋转。

回撤：撤回所进行的操作。

燃烧：用具体例子讲述比较好，记当前小方块为**A（其数字不为1）**，位于其正上方的小方块为**B**，位于**B**正上方的小方块为**C**，位于**A**正下方的小方块为**D**，那么当执行燃烧操作时，**A**的属性将会先向下传递给**D（D的颜色被替换为A的颜色，D的数字也被替换为A的数字-1）**。然后，**B**的属性也会以相同的方式传递给**A**，执行完后再把**D**的属性传递给**B**，以此类推。

当**A**的数字为1时，那么其颜色和数字属性则不能传递给**D**。其他小方块也是同理的。

由此考虑特判情况，位于最下面的一行无法往下流动，所以当其数值不为1时，燃烧后也会自动转化为1。

至此，基本分析清楚要实现的四个基本操作功能。我决定先用二维数组来实现，后面再执行进一步的封装操作，方便日后移植到Qt上。对于实现撤回操作，我打算使用栈是实现，将每一次执行前的矩阵存入到栈中。在执行回撤操作时，只需要不断弹出取出栈顶元素，并且弹出栈即可。

以下是**Matrix**的实现（一开始没有进行封装）：

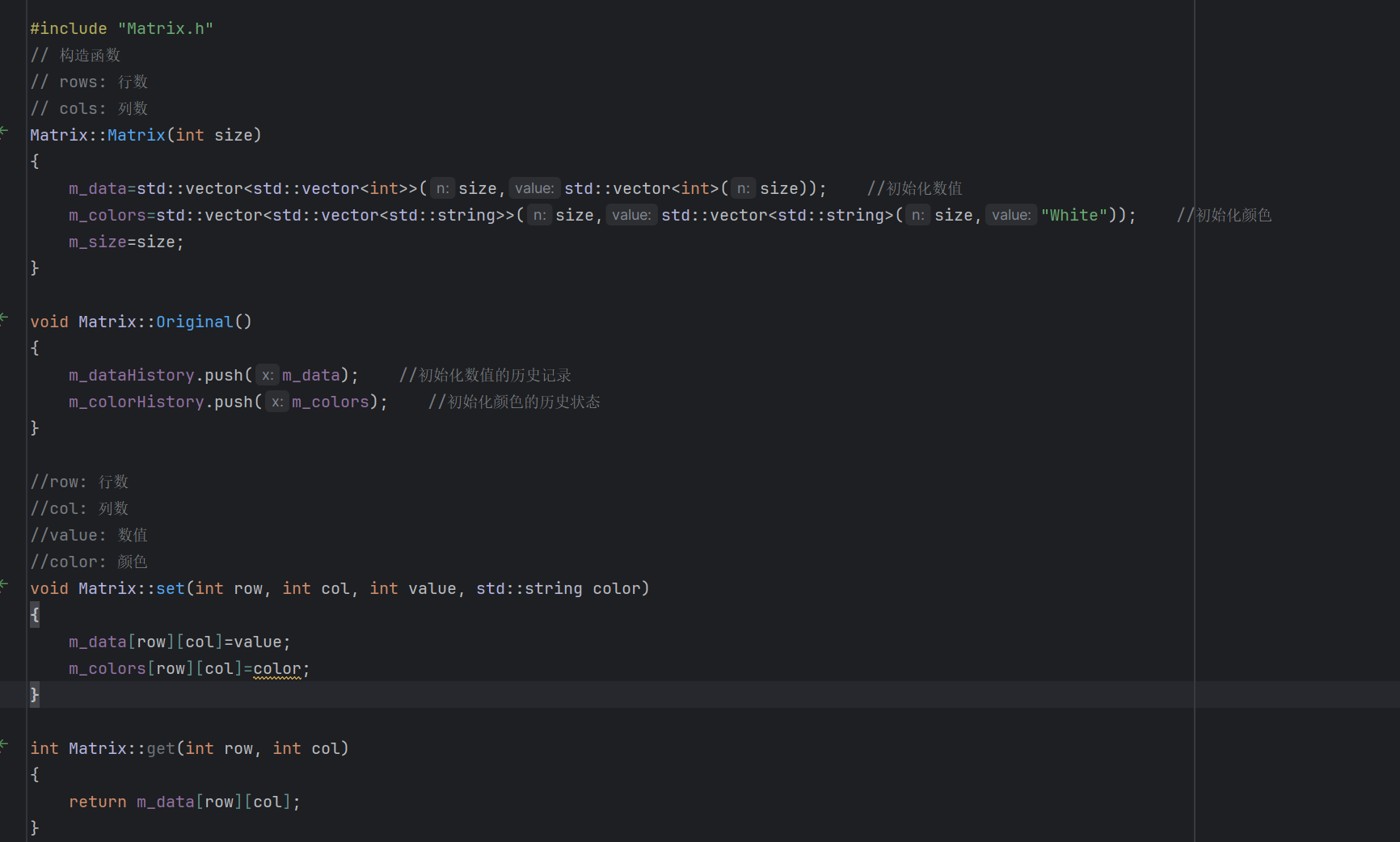
**Matrix.h**文件





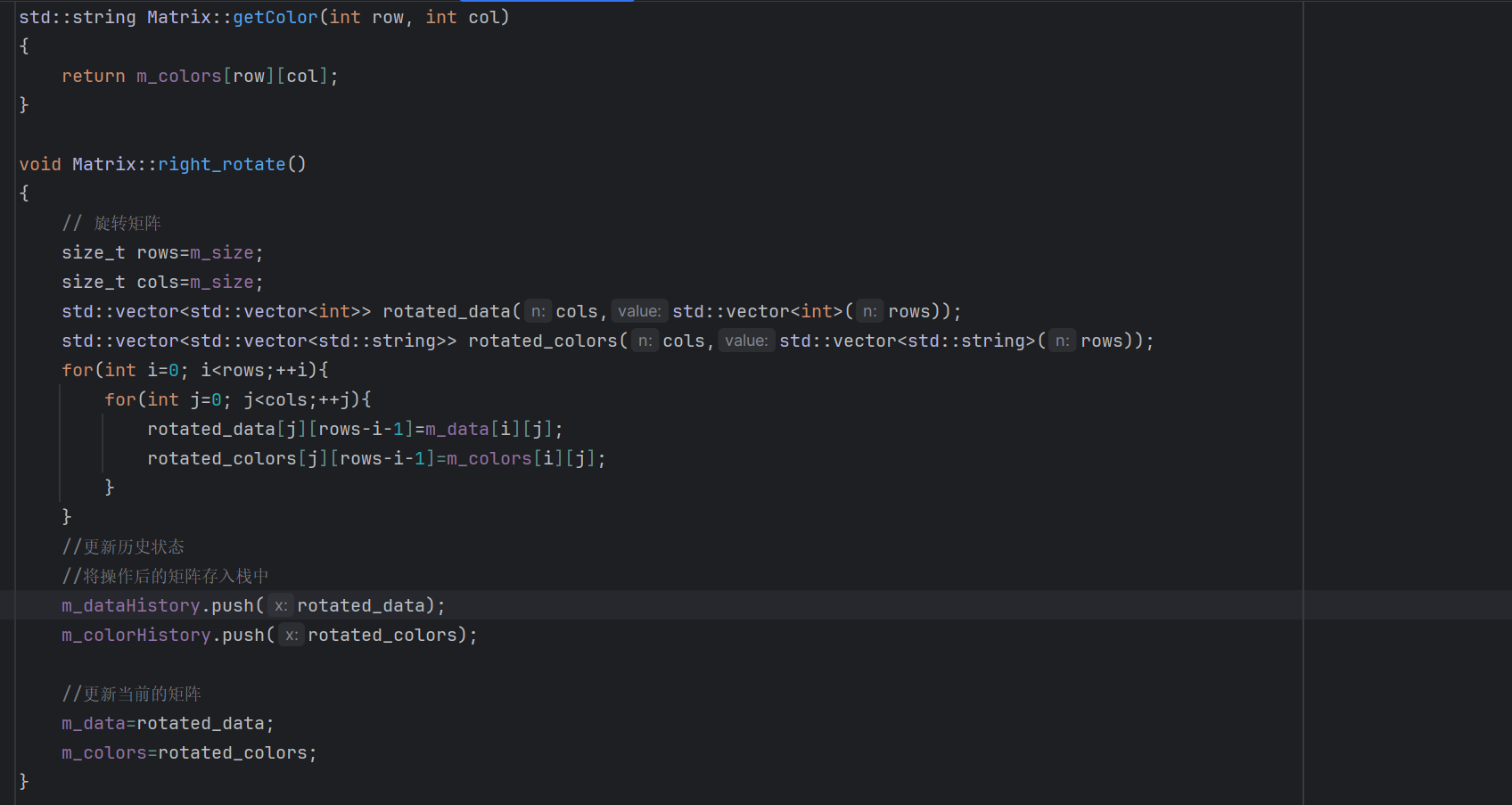
**Matrix.cpp**文件

一些接口的实现

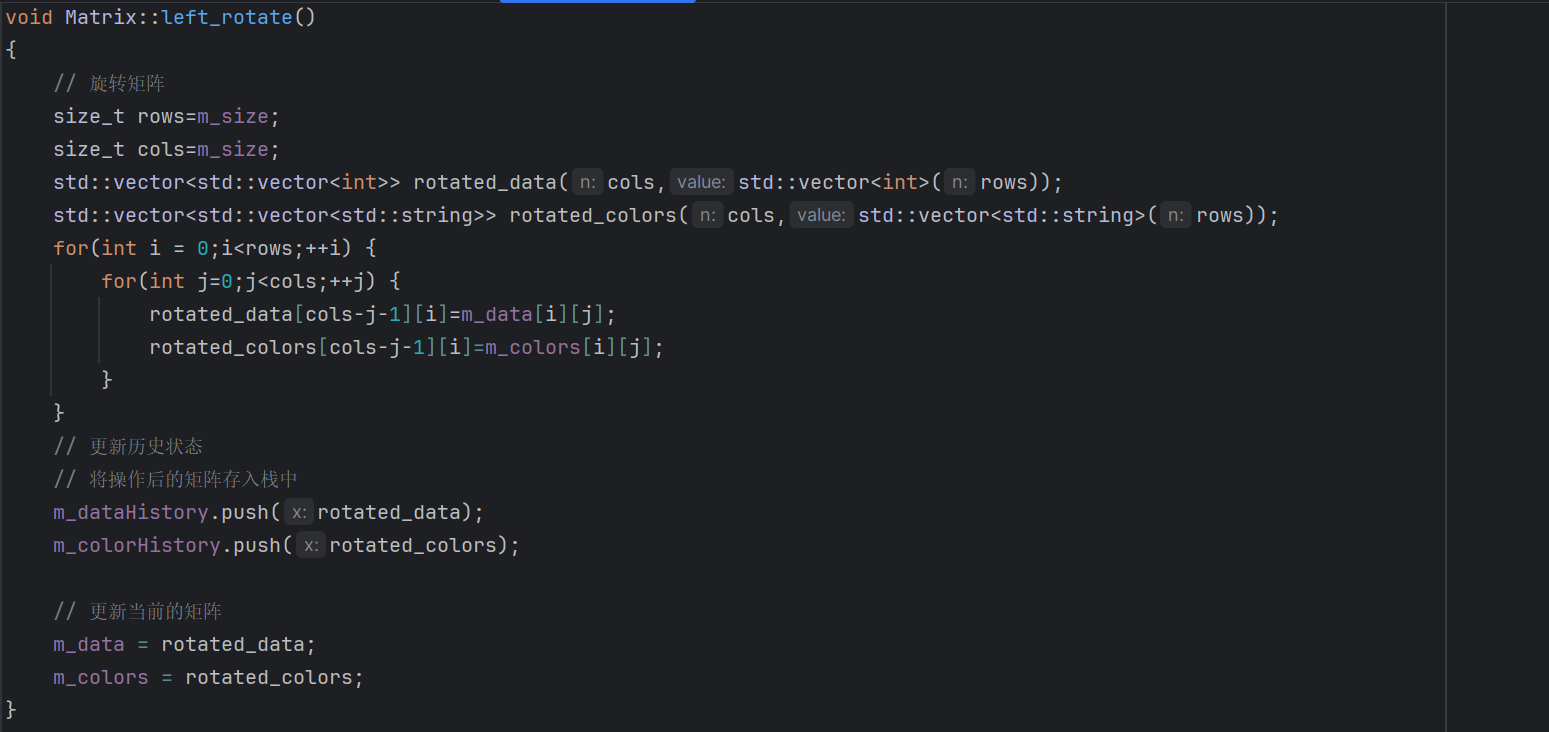


接下来是四个操作的函数

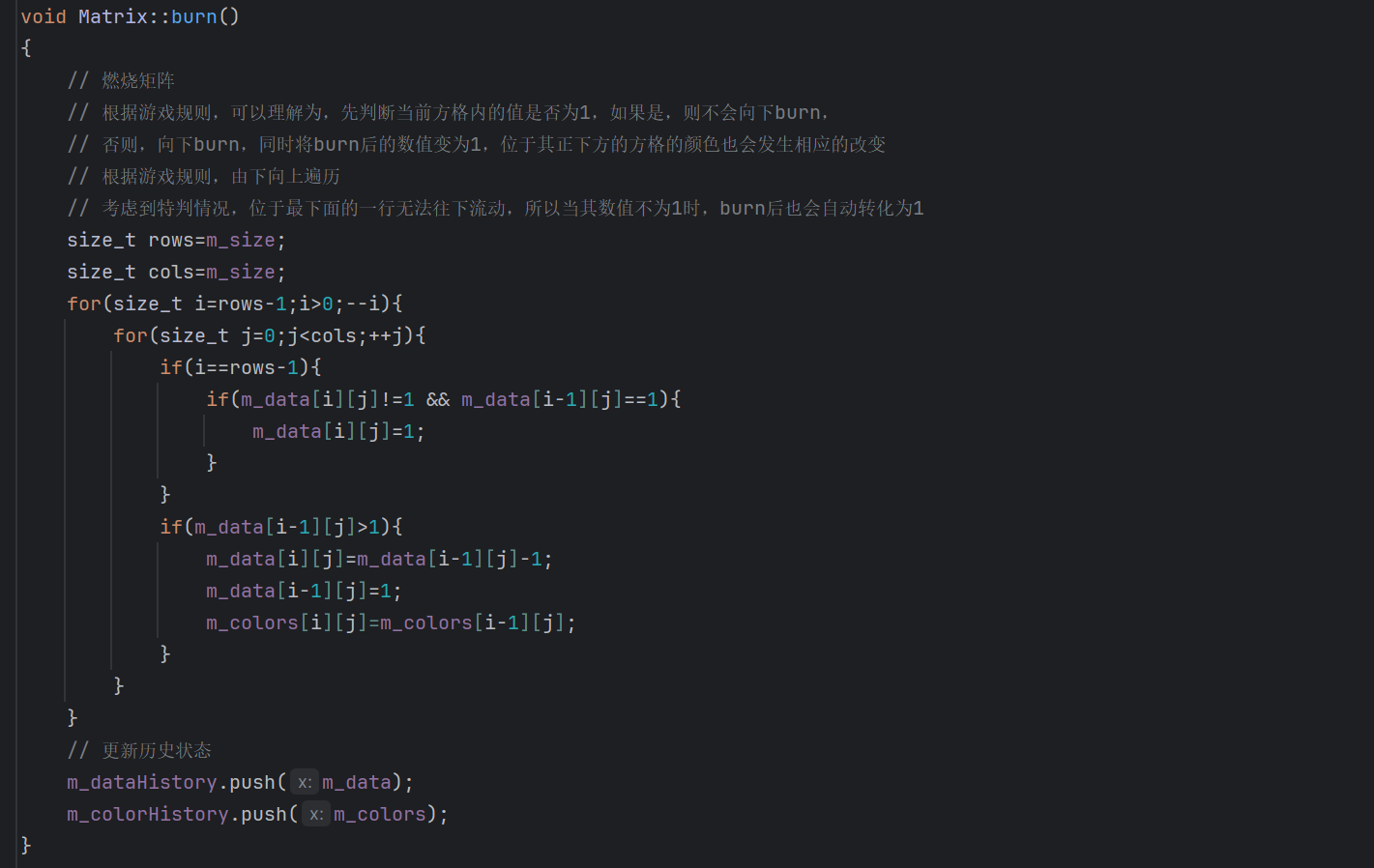
右旋转



左旋转

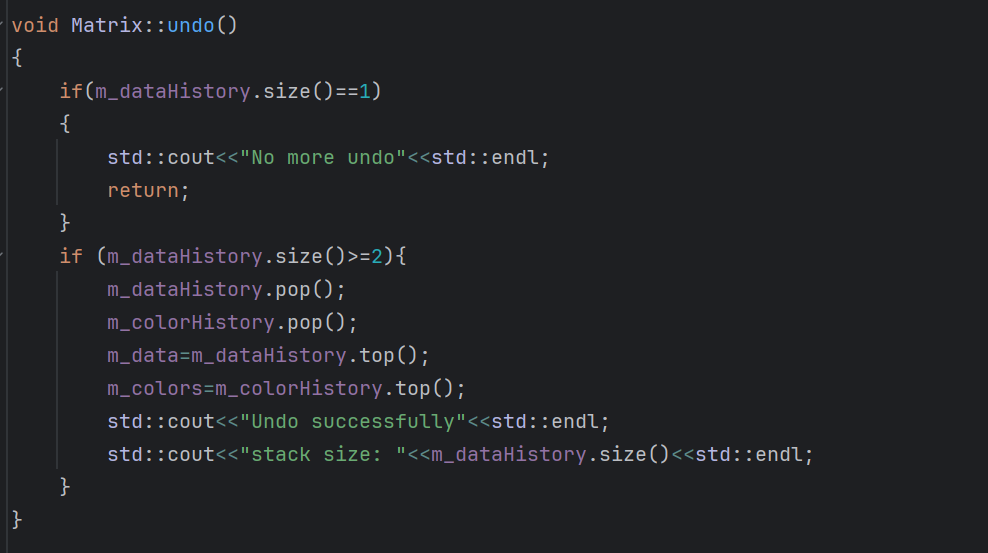


燃烧



撤回

这里主要是用来协助判断有没有出现bug

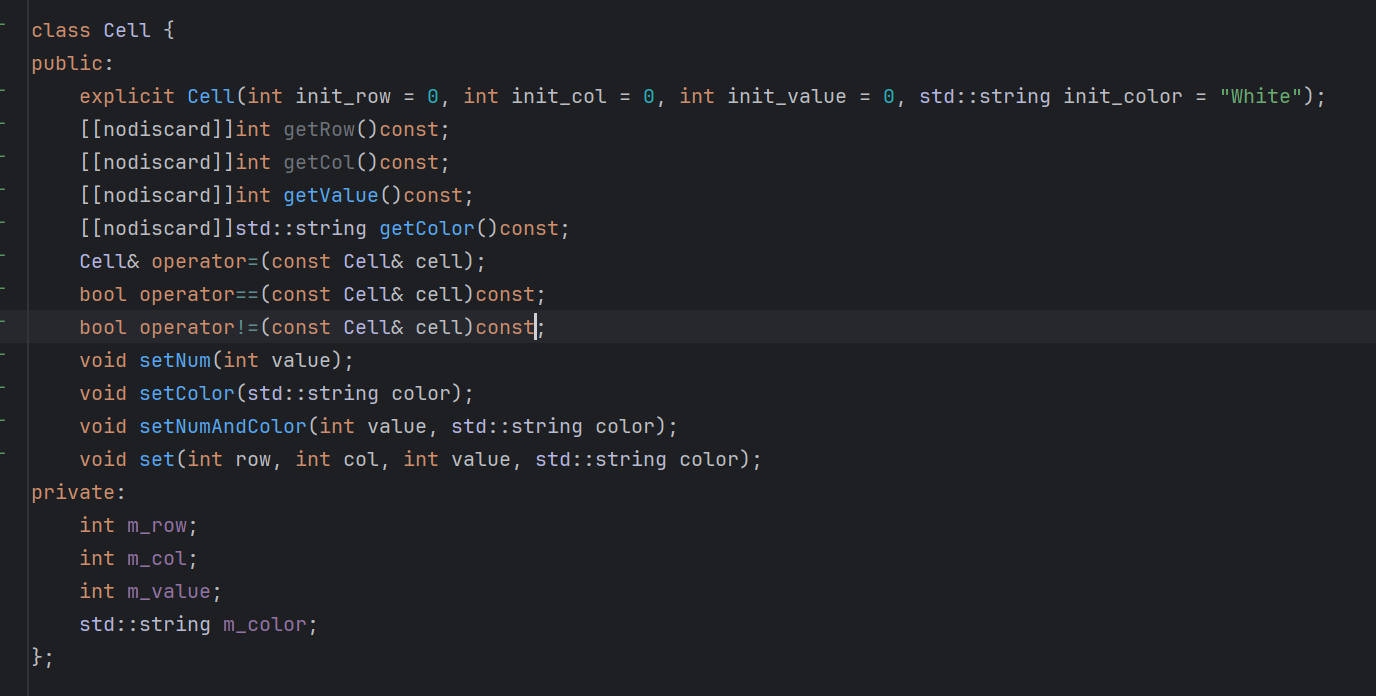


以上就大概实现功能的基本逻辑，后面再进一步实现类的封装。

重新封装了一下小方块类（命名为**Cell**）和画类(命名为**Painting**)，并且重载了一下运算符。

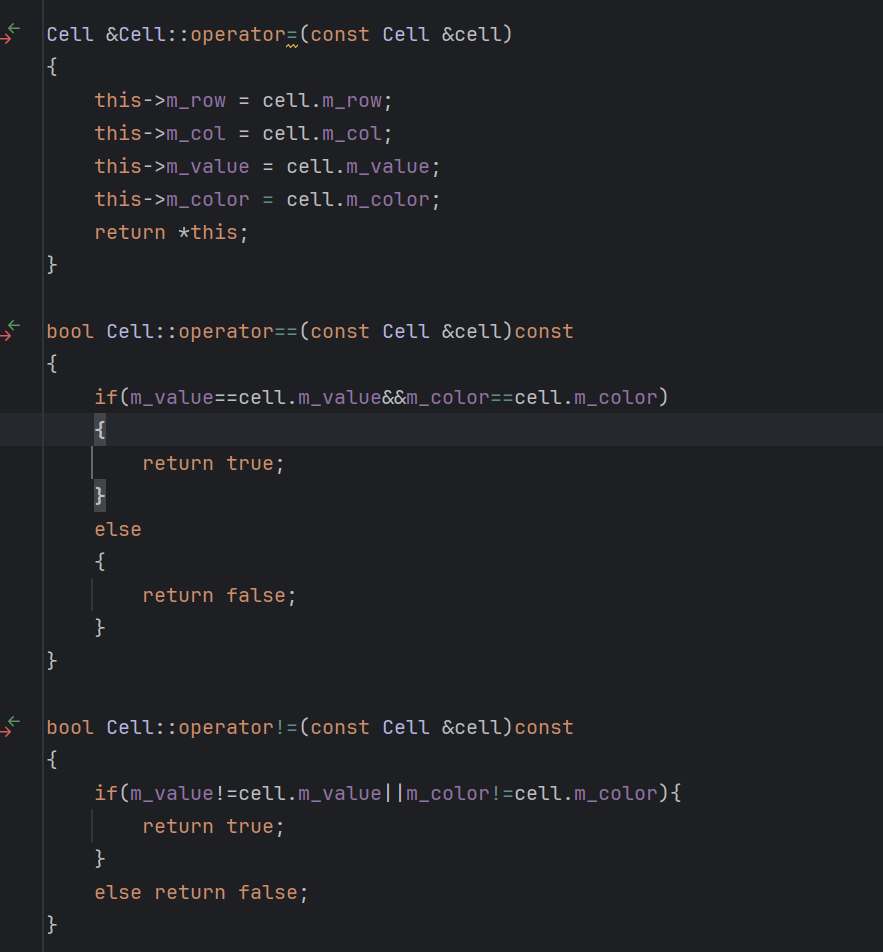
**Cell**类主要还是提供各种接口，方便**Painting**类访问其属性。

**Cell.h**文件实现

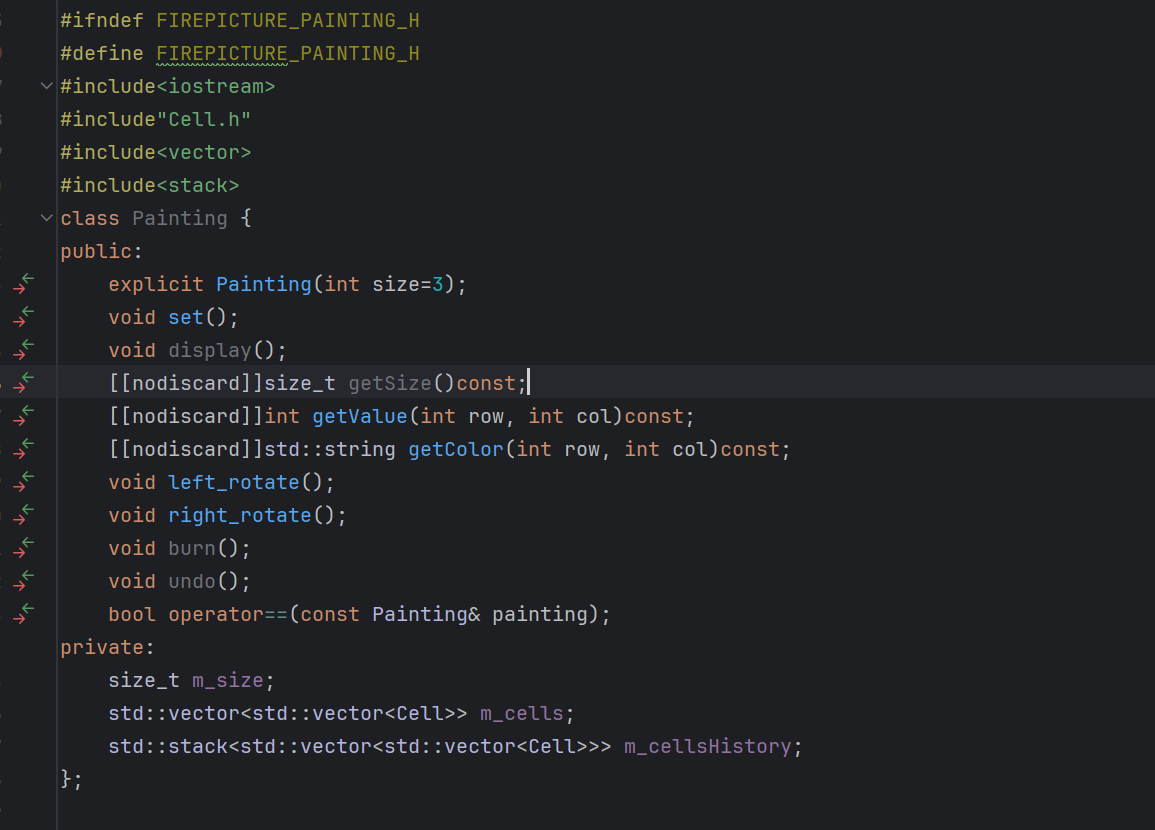


在这里我学到了一个新的知识点：Clion编译器提示我应该在**getColor**等函数前面应该加上**[[nodiscard]]**：它可以用于提示用户获取返回值，如果用户弃值了（或者没有将其转化为void），那么编译器将会抛出**warning**来警告我们获取返回值。

大部分的函数都是提供一个接口，比较重要的是重载运算符：这些关系到后面**Painting**类的一些赋值问题。



**Painting.h**文件实现



**Painting.cpp**文件的实现基本上和上面**Matrix.cpp**大同小异，这里就不展示了。

自己也看了一些Qt的博客文章和教程视频，对Qt有了一个大致的认识，但实际感觉真正实践和理论还是存在很大差别，比方说页面跳转这一功能少有网课提及，还是得自己多多寻找一些资料看（还有问题就是文章质量良莠不齐）。

由于接下来一周要参加蓝桥杯和ACM的校赛，需要花时间准备，所以这周的工作进展不是很快，请老师谅解。