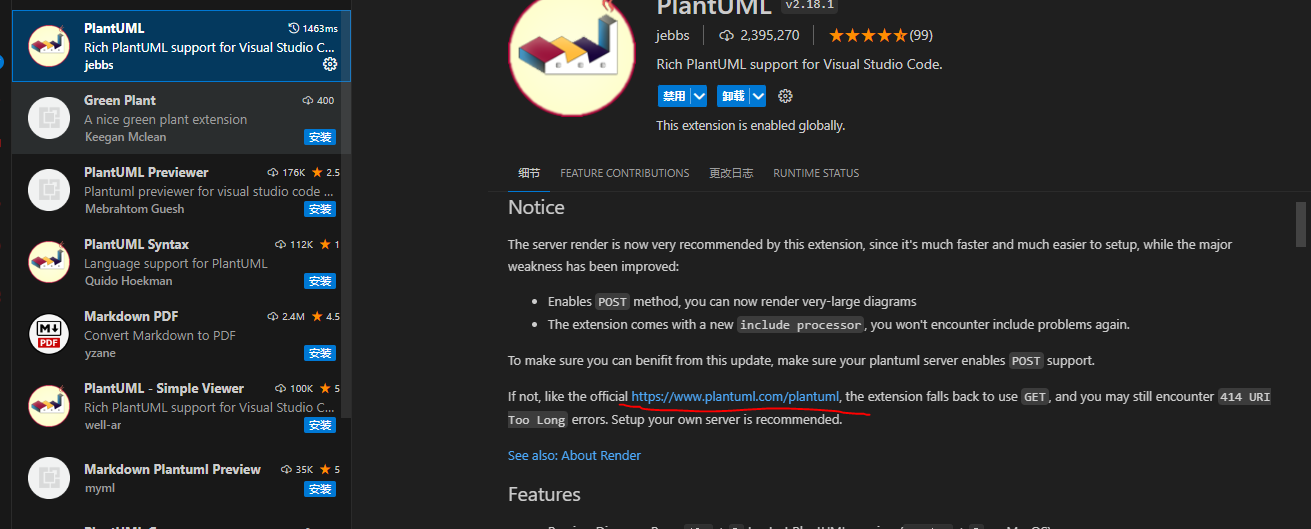
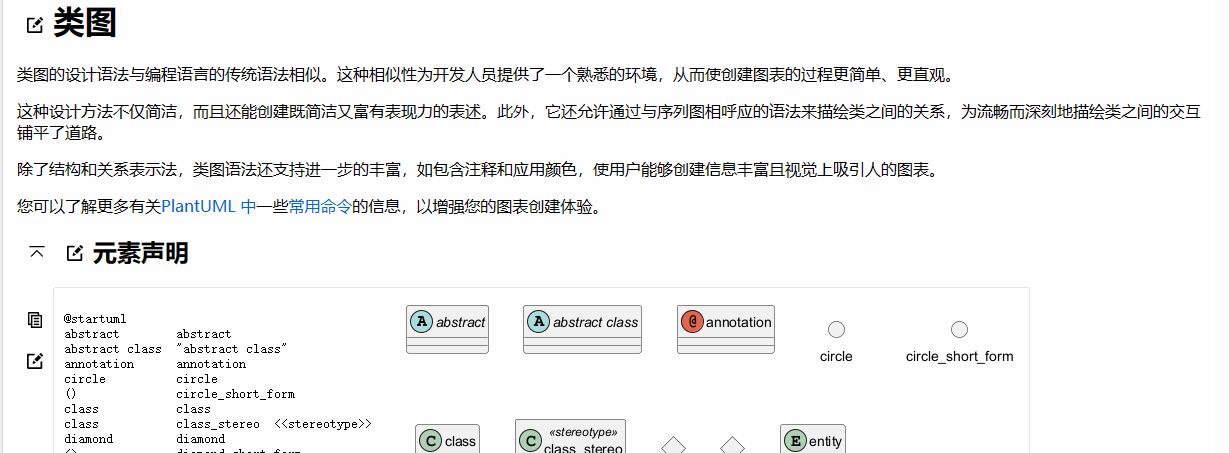
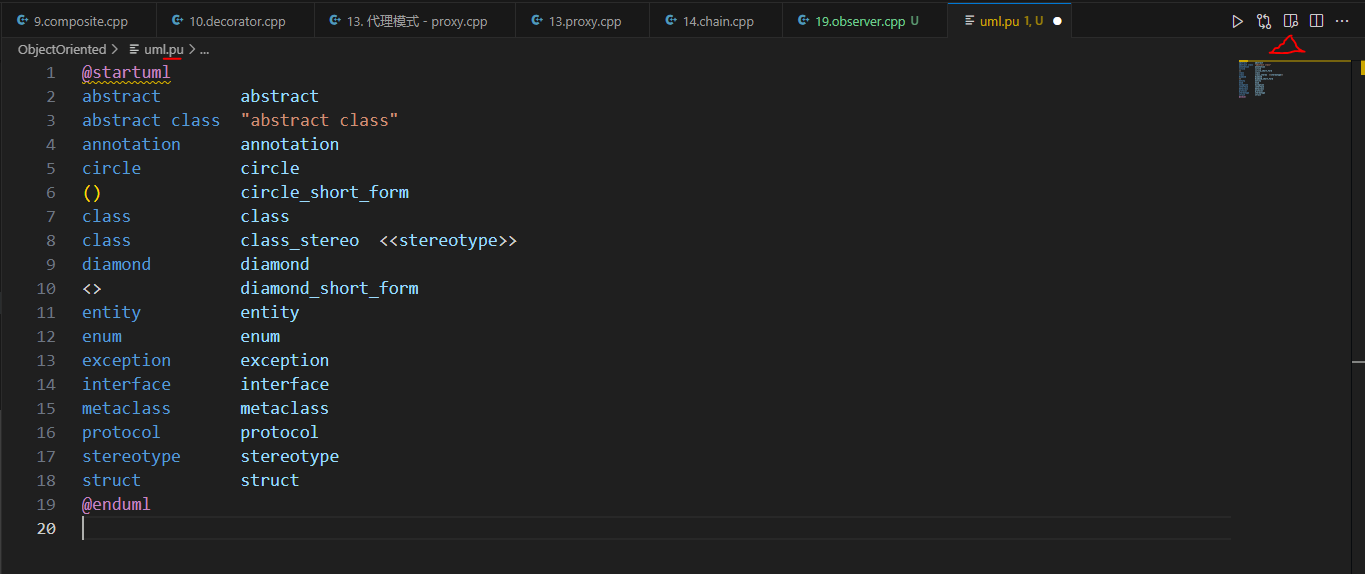
# UML类图

使用 vscode插件 plantuml画，对照官网示例





创建.Pu文件,点击那个图标就行了（可以设定主题）

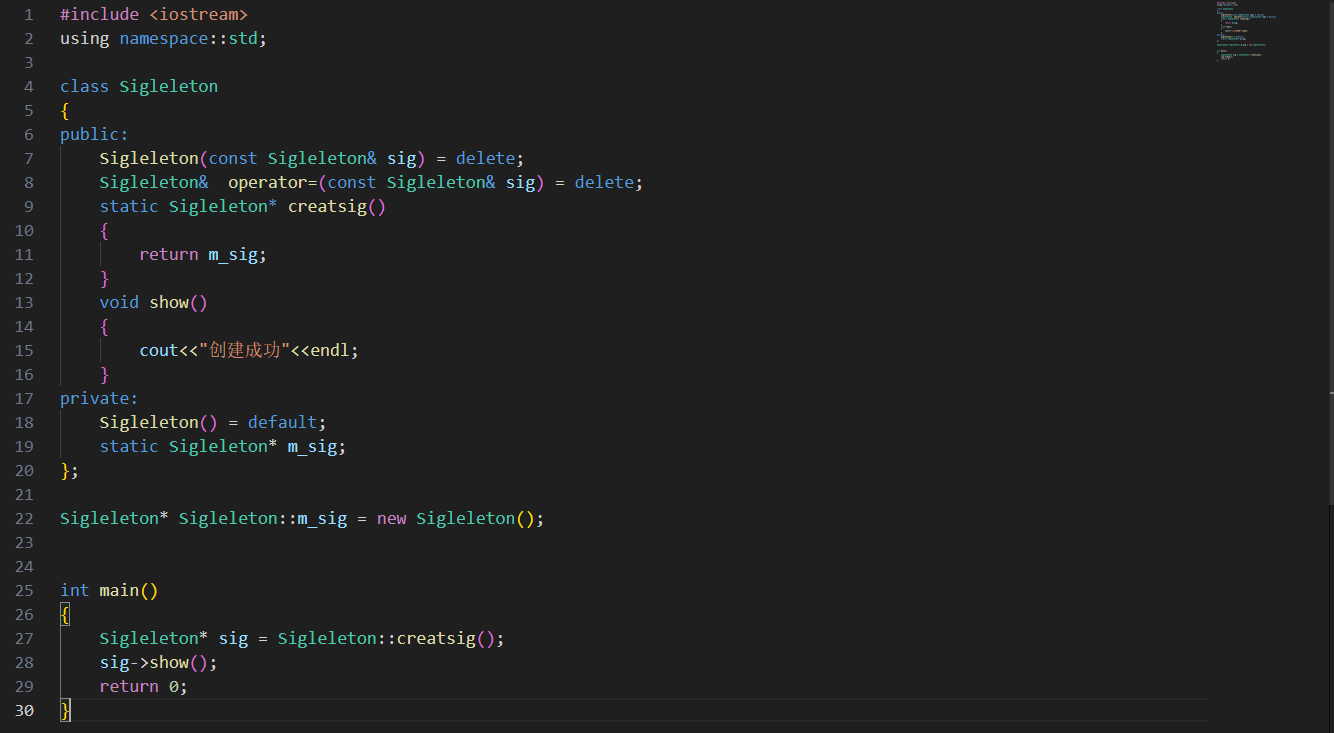


# 单例模式

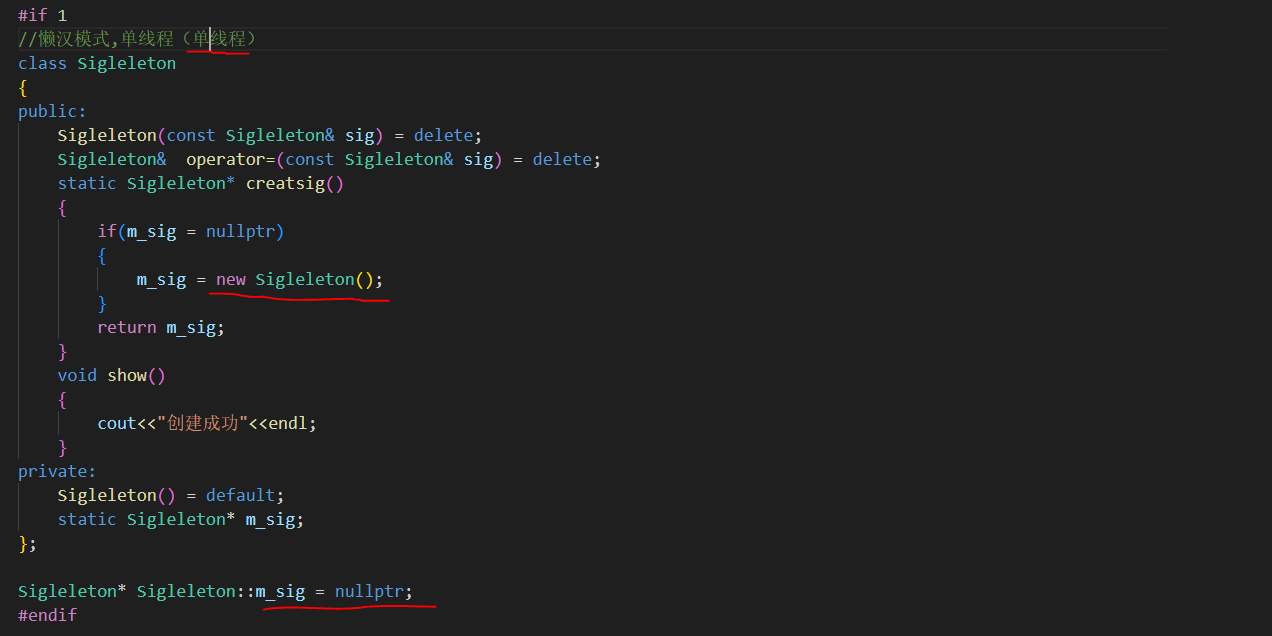
1. 重载=是有模版的
2. Static函数是不需要在全局区声明的，static成员变量才需要
3. Sigleleton不想重新写直接 = default 就行



改完这样，饿汉模式

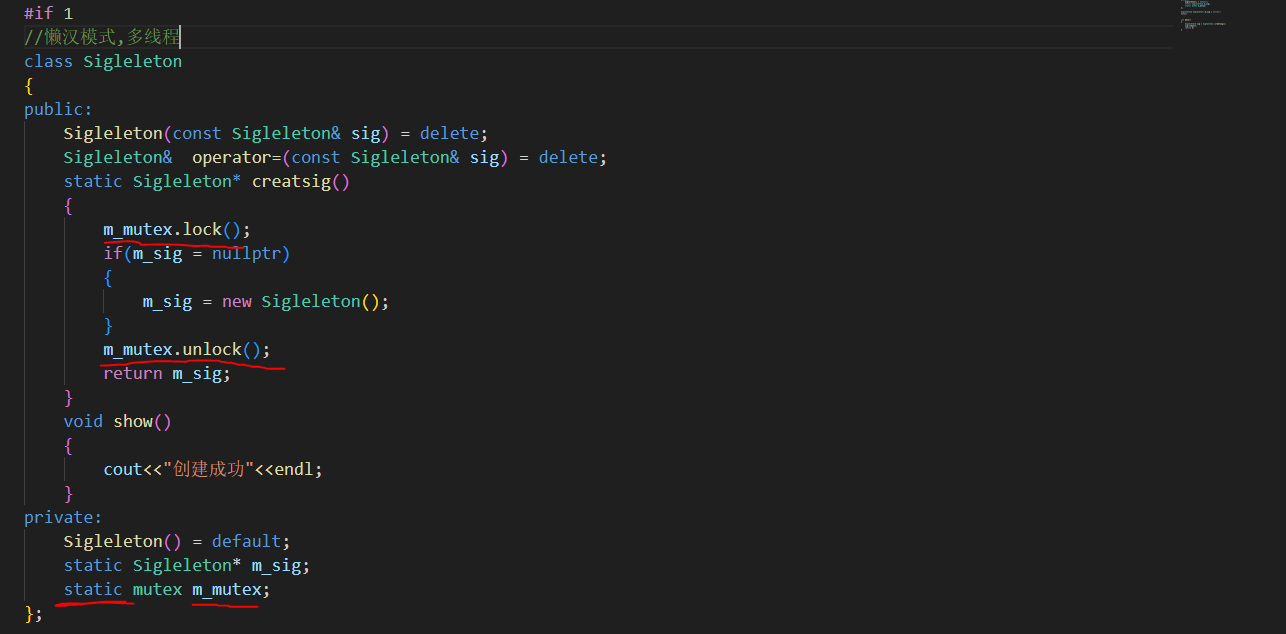


懒汉模式，单线程（多线程同时进入creatsig可能创建多个实例）

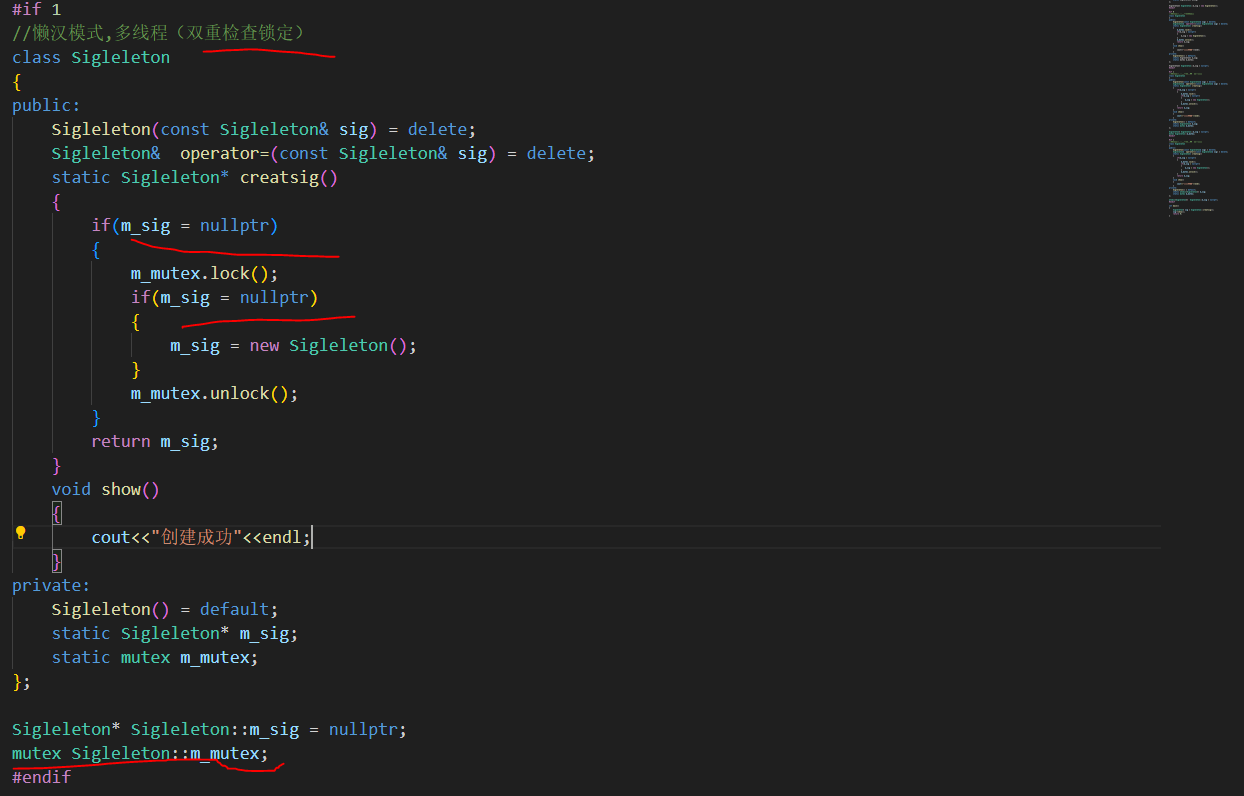


加锁：

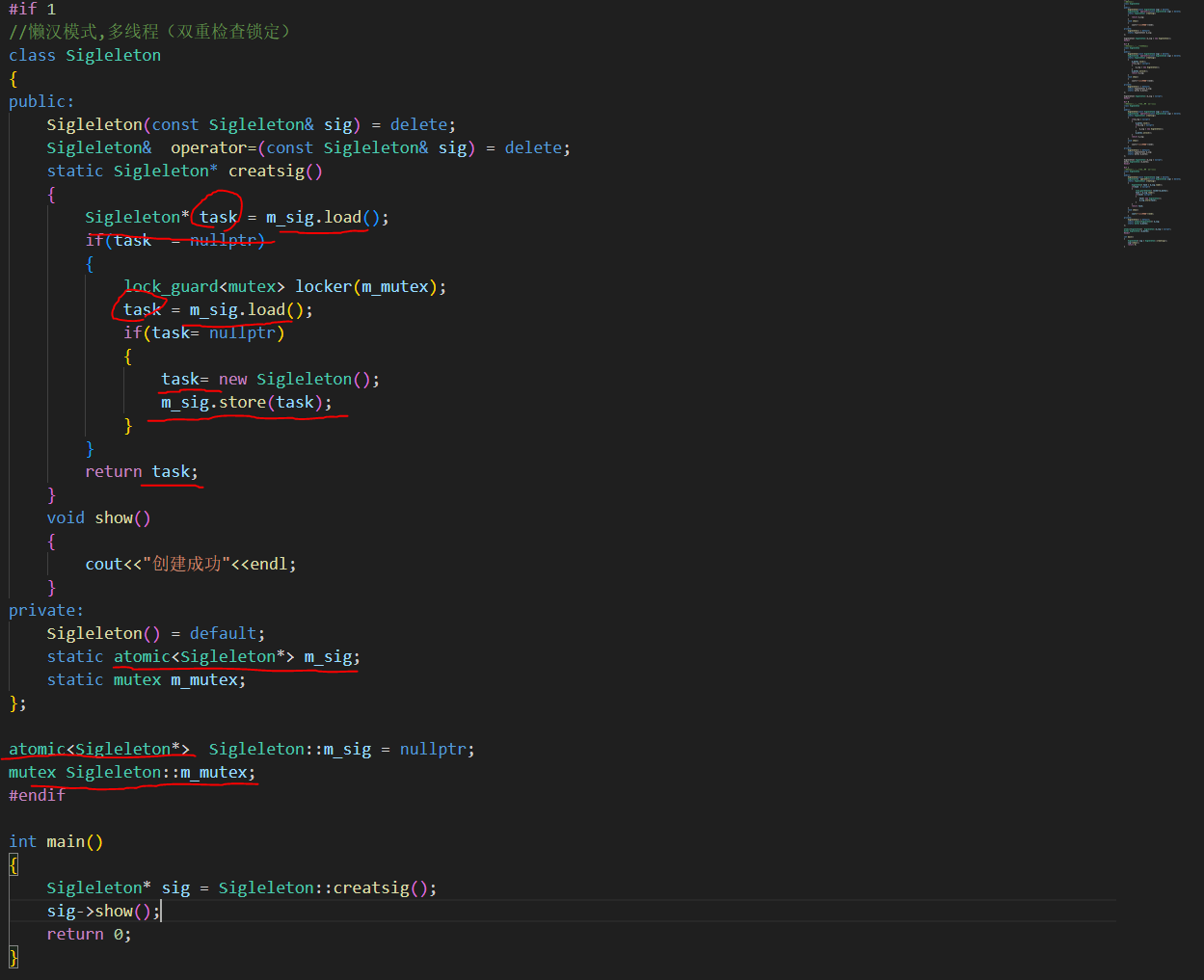
1. static 函数中使用，m\_mutex设置为static类型
2. 多线程在这里拥挤



双重检查锁定



加入原子操作



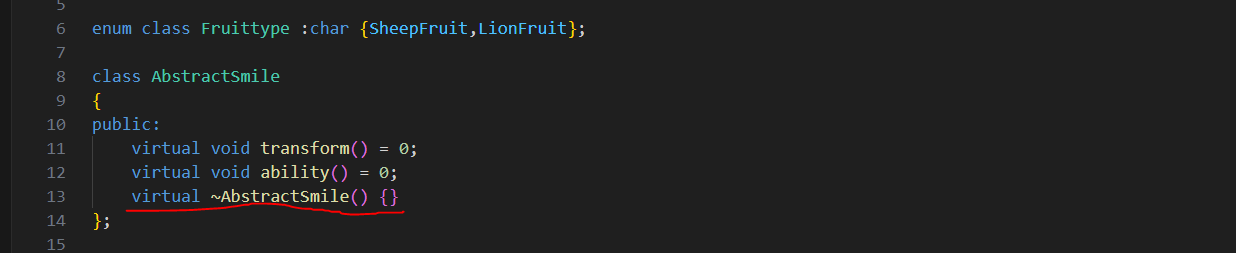
静态局部变量



# 简单工厂模式

其实就是创建新对象要从factory类中走一遍，不是绕路了吗？！？！

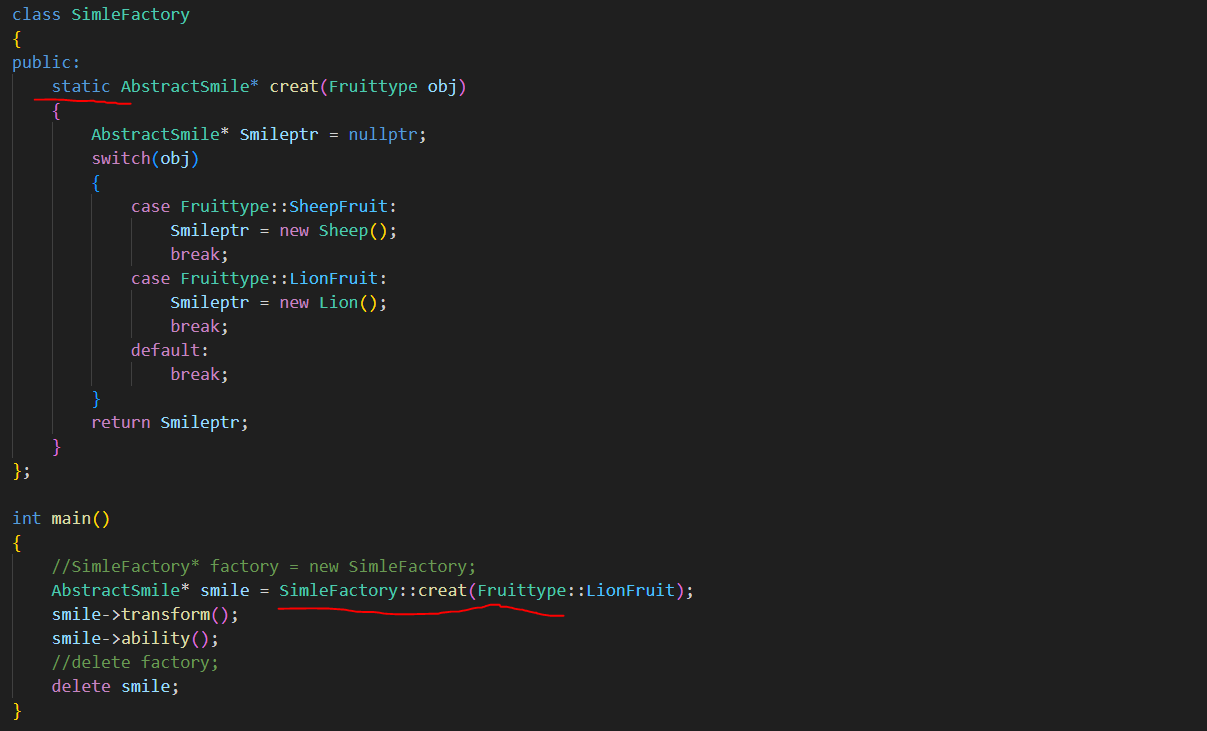
涉及到虚函数，至少要写虚析构函数



强类型枚举 Fruittype 是枚举名字，SheepFruit以char存储，使用的时候都写成 Fruittype::SheepFruit



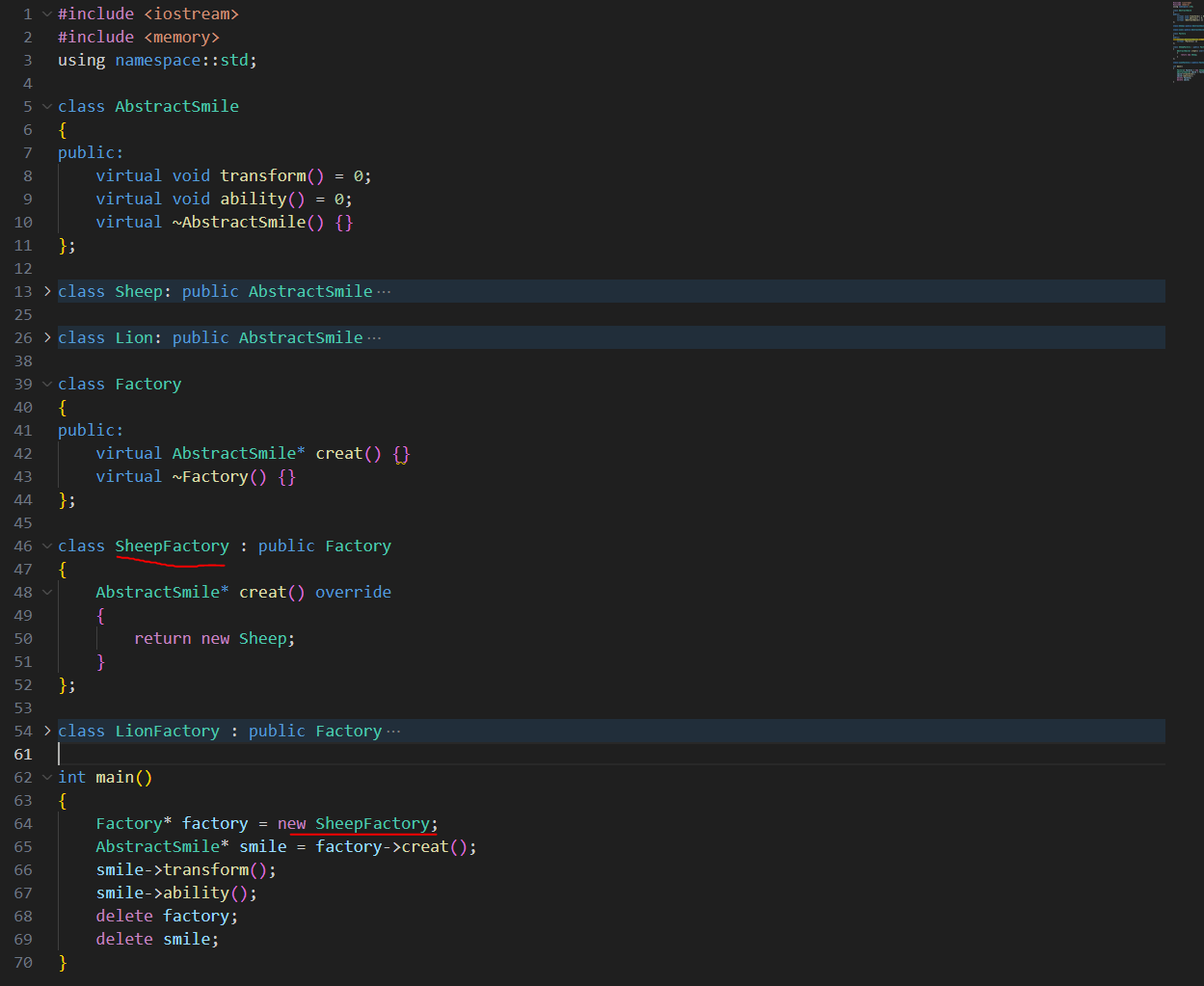
设定成静态函数，使用的时候很方便



工厂函数中需要生成三种人造恶魔果实，现在如果想要生成更多，那么就需要在工厂函数的switch语句中添加更多的case，很明显这违背了封闭原则，也就意味着需要基于开放原则来解决这个问题

# 工厂模式

需要写很多子类工厂，而且main函数中需要判定使用哪个工厂，劳动量并没有减少



缺点：在添加新产品时，需要编写新的具体产品类，而且要提供与之对应的具体工厂类，系统中类的个数将成对增加，一定程度上增加了系统的复杂度。

# 抽象工厂模式

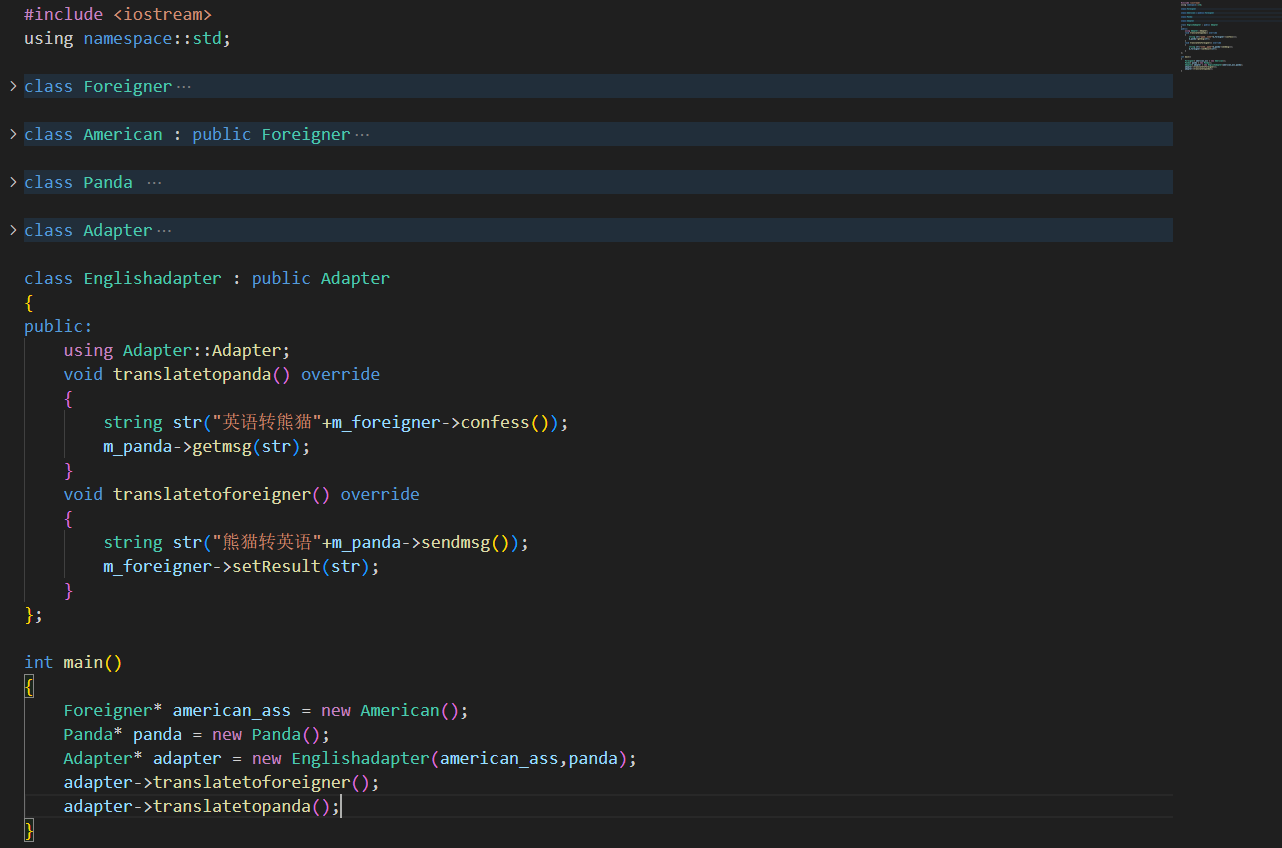
抽象工厂模式是工厂方法模式的泛化版，工厂模式是一种特殊的抽象工厂模式，在工厂模式中，每个具体工厂只能生产一种具体的产品，如海尔电视机厂只生产海尔电视机，而抽象工厂方法模式中，一个具体的工厂可以生产多个具体产品

# 适配器模式

适配器模式（Adapter Pattern）是一种结构型[设计模式](https://so.csdn.net/so/search?q=%E8%AE%BE%E8%AE%A1%E6%A8%A1%E5%BC%8F&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/h8062651/article/details/_blank)，它允许一个类的接口与另一个类的接口不兼容时，仍然能够协同工作。适配器模式的主要目的是通过提供一个中间层来转换接口，使得原本不兼容的接口能够协同工作。

适配器模式可以让类在现有的基础上进行复用，而不需要做出任何改变，这有助于避免大规模改写现有代码。

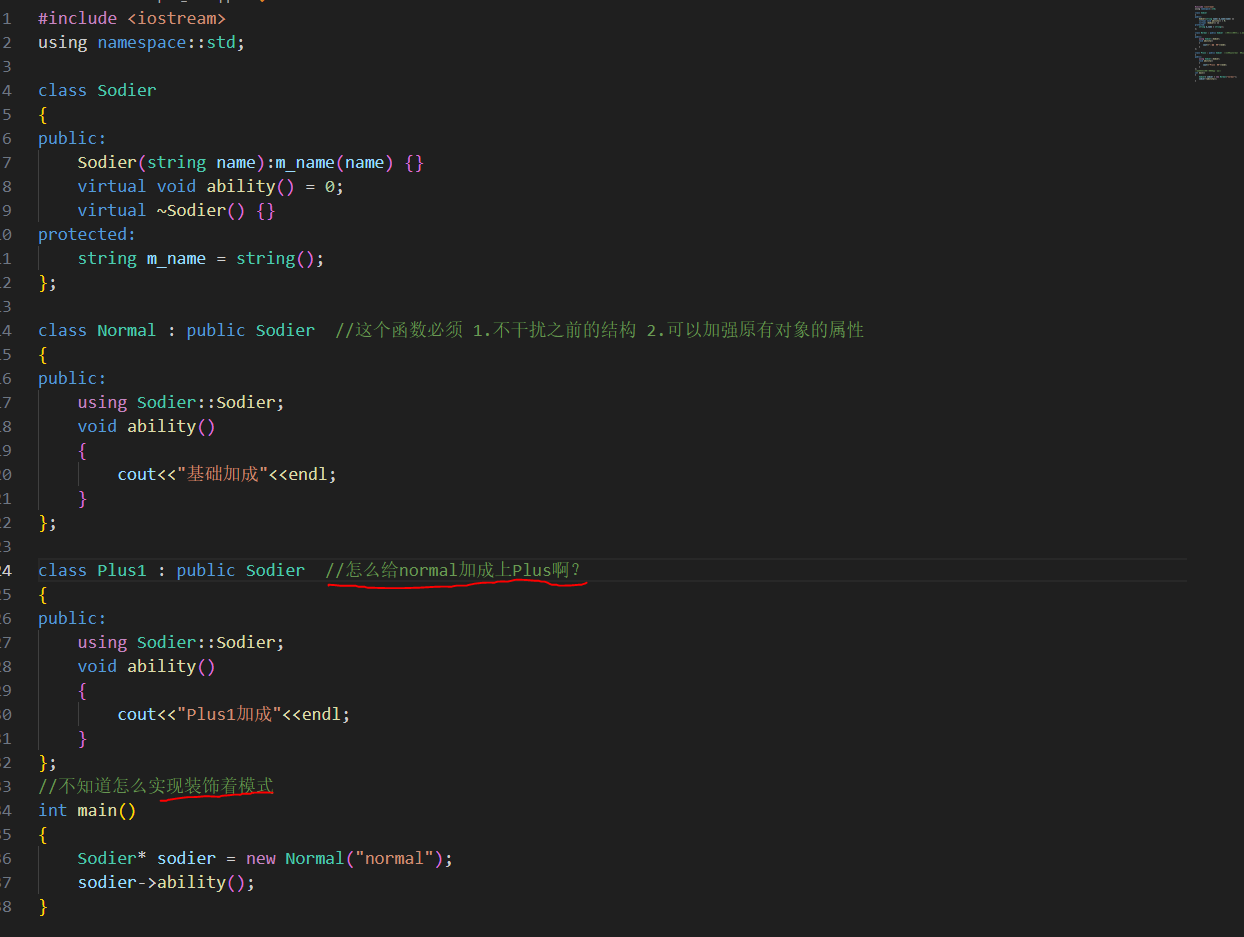
Foreigners类和Panda类本来就存在，只是相互信息不兼容，Adapter的出现不改变二者原本的结构，但是通过Adapter可以在二者之间转码。



在上面的适配器类中，同时访问了Foreigner 类和Panda 类，这样适配器类就可以拿到这两个类对象中的数据进行转译，最后再将其分别发送给对方，这样这两个不相干的没有交集的类对象之间就可以正常的沟通了。

# 装饰者模式

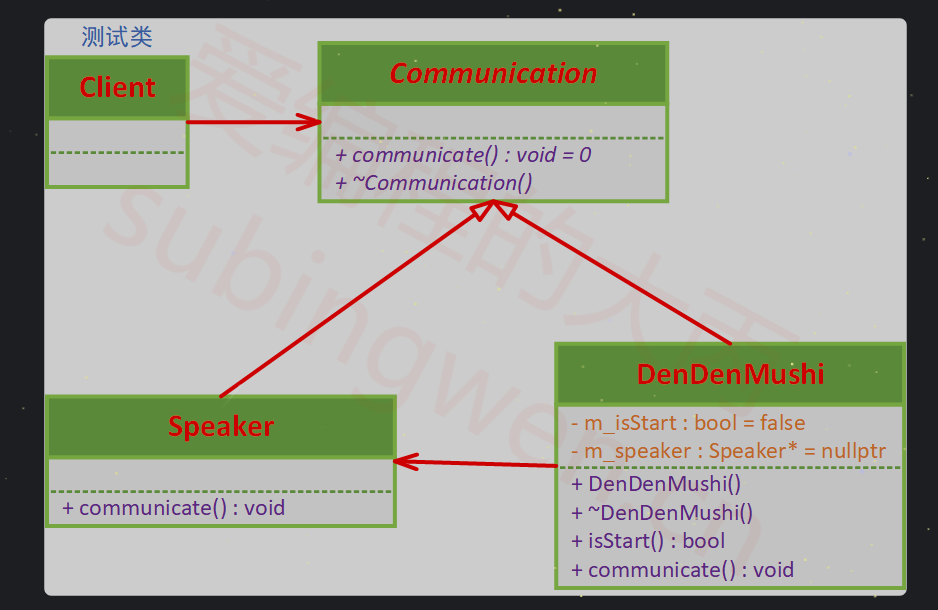
只记得是同类给同类加强属性，属于兄弟关系



C++ 中，当你继承一个类时，如果基类（如 Sodier）没有默认构造函数（即无参数构造函数），那么派生类的默认构造函数也会被标记为**已删除**。

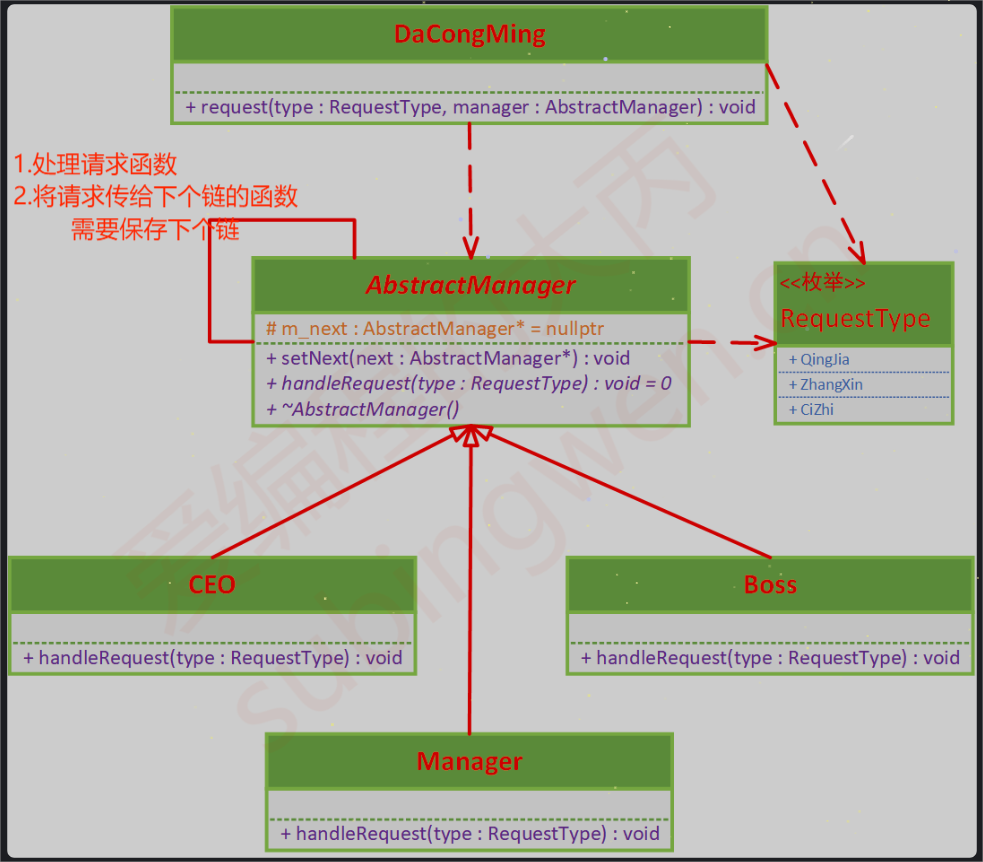
# 代理模式

代理模式和装饰着模式有个共同点，跟要操作的对象继承同一个父类，将对象当作自己的一个变量，在和对象相同的框架函数中加私货。

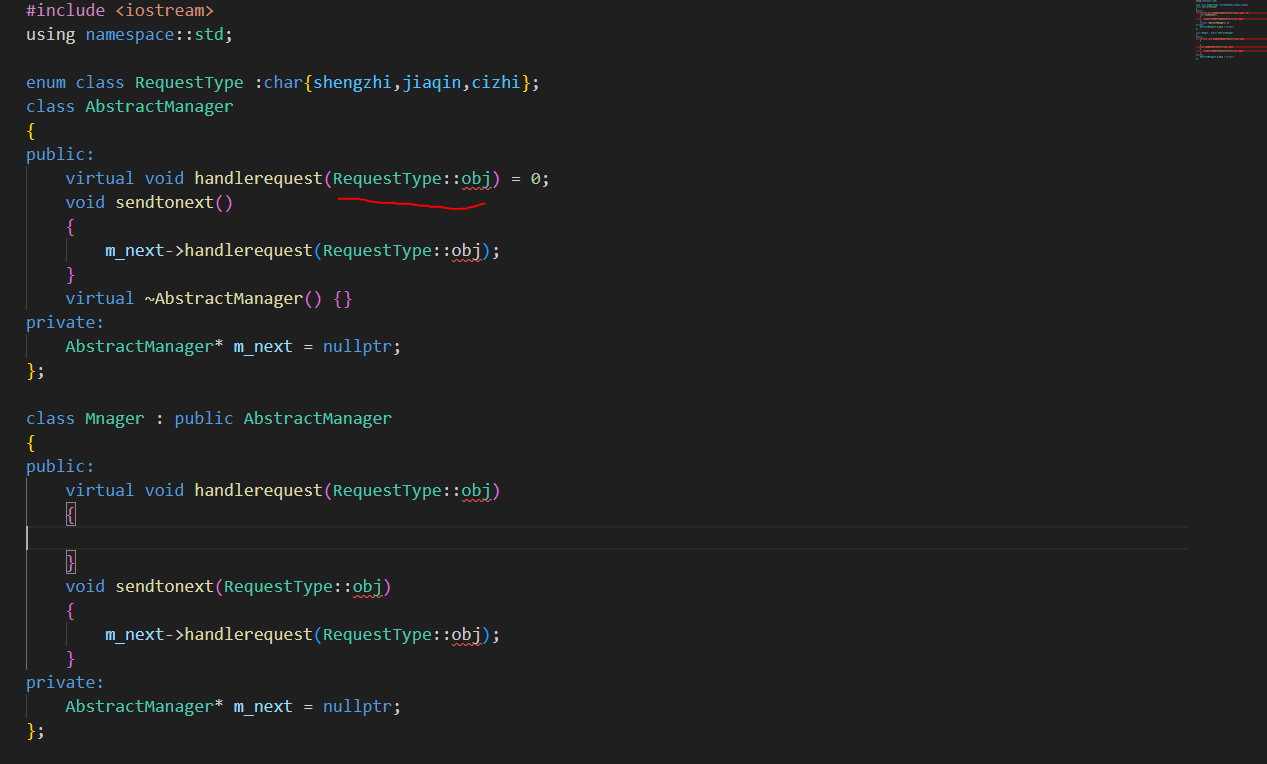


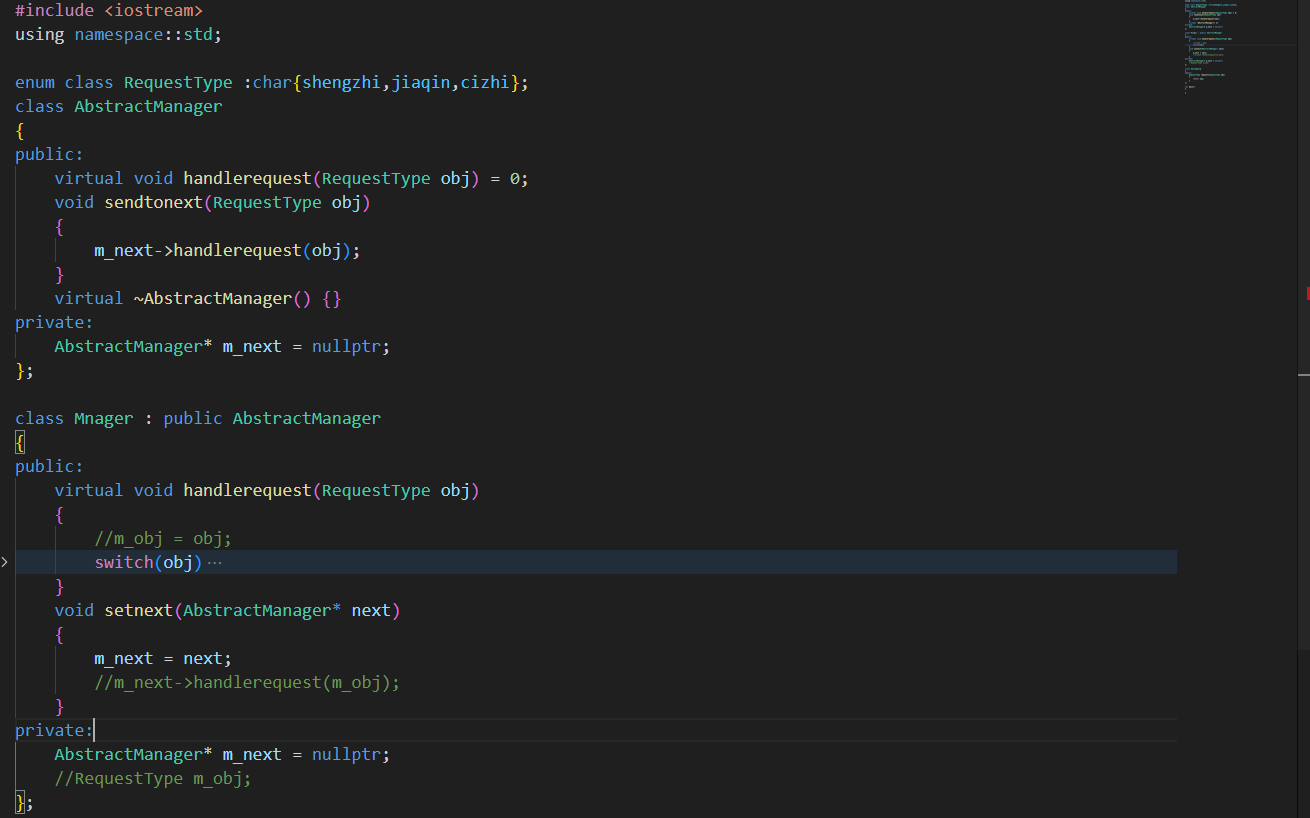
# 责任链模式

职责链(Chain Of Responsibility)模式也叫责任链模式，是一种行为型模式，用于将一个请求传递给一个链中的若干对象，哪个对象适合处理这个请求就让哪个对象来处理。职责链看起来与传统[数据结构](https://edu.csdn.net/course/detail/40020?utm_source=glcblog&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/qq_68194402/article/details/_blank)中的“链表”非常类似。



怎么在这里表示参数是个枚举类型呢？



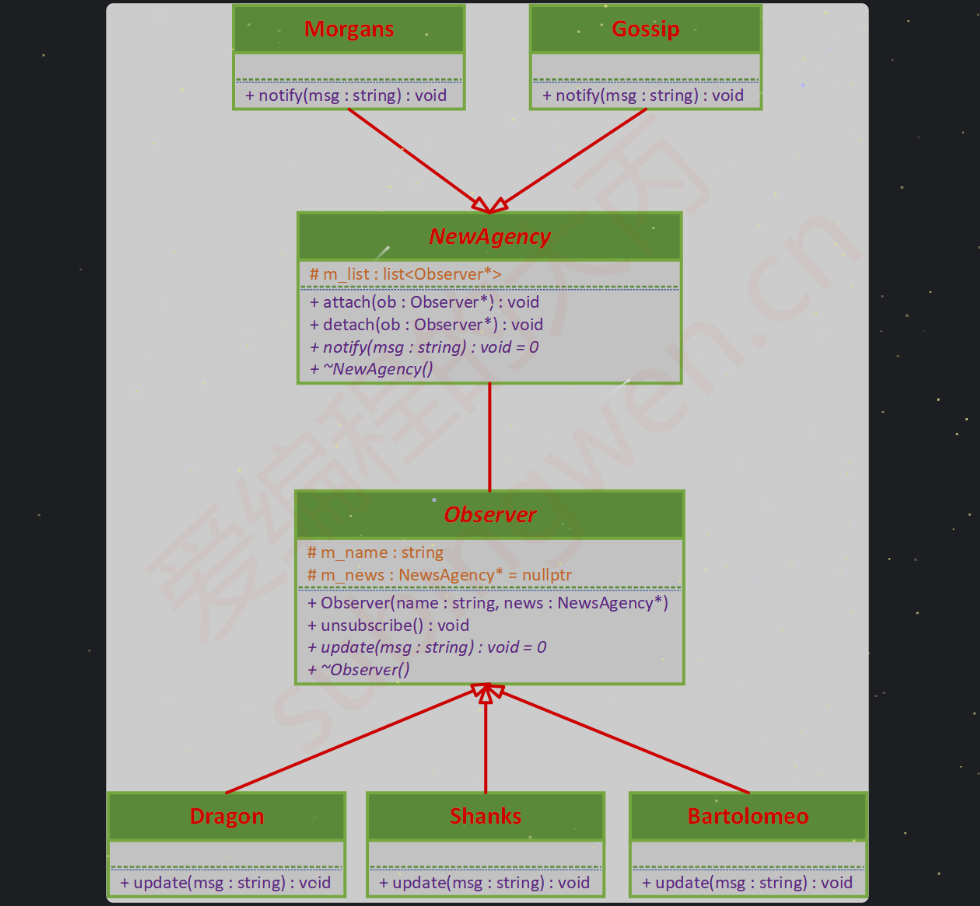


# 观察者模式

观察者模式允许我们定义一种订阅机制，可在对象事件发生时通知所有的观察者对象，使它们能够自动更新。

在[程序设计](https://so.csdn.net/so/search?q=%E7%A8%8B%E5%BA%8F%E8%AE%BE%E8%AE%A1&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/weixin_45433817/article/details/_blank)中，观察者模式通常由两个对象组成：观察者和被观察者。当被观察者状态发生改变时，它会通知所有的观察者对象，使他们能够及时做出响应，所以也被称作“发布-订阅模式”。

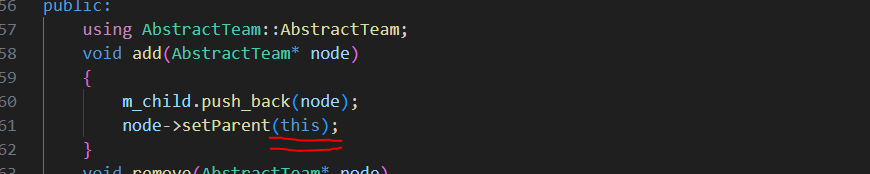
这种一般要将两个父类先写前面，之后再各自定义自己的子类。

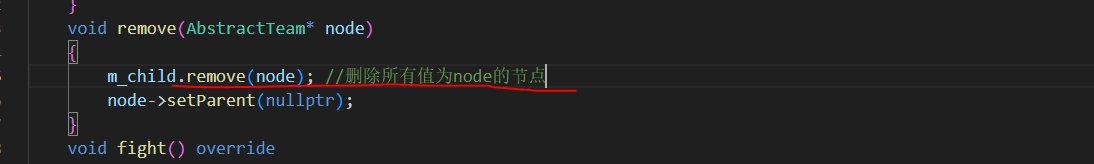


# 策略模式

Luffy类中有Abstract类成员，这个成员又有四五个儿子，luffy根据情况对Abstrac实例化，对应的函数中调用的具体函数不同，没太大意思

# 组合模式





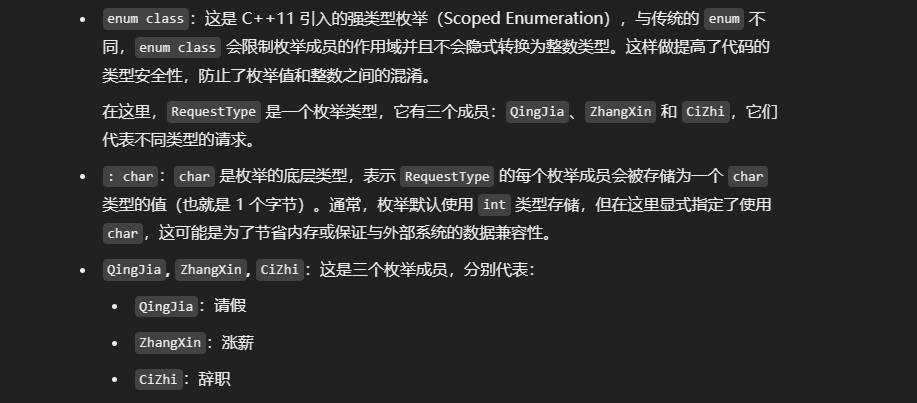
责任链模式：

enum class RequestType:char {QingJia, ZhangXin, CiZhi}; //强枚举类型

Enum class 是强枚举类型

RequestType 枚举类型

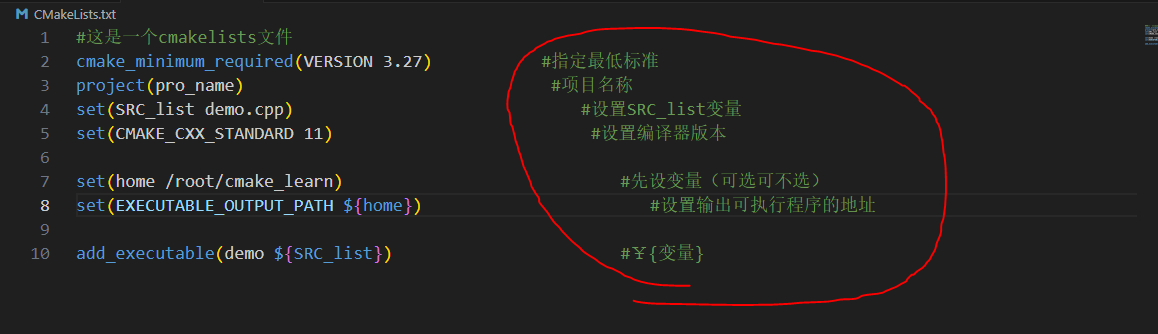
：char 指定类型



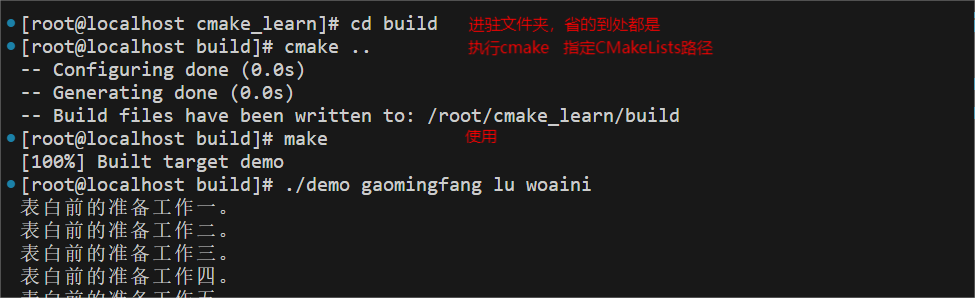


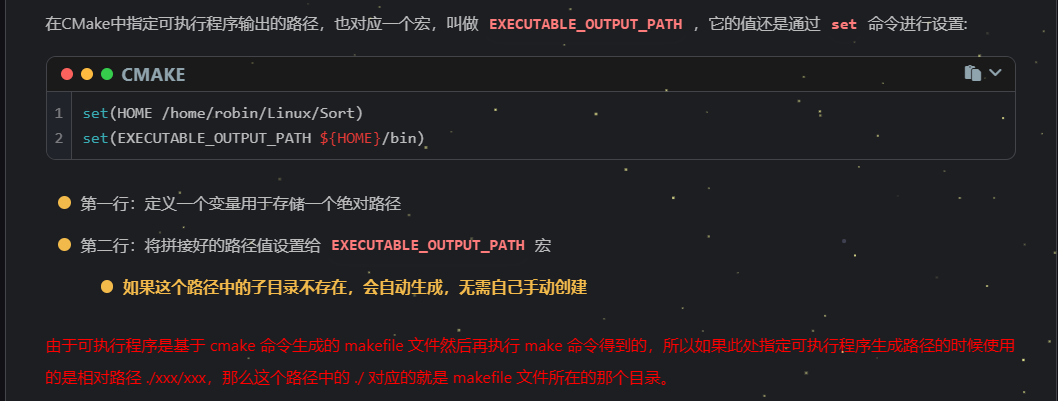
# CMAKE

CMake是开源、跨平台的[构建工具](https://edu.csdn.net/cloud/sd_summit?utm_source=glcblog&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/weixin_43717839/article/details/_blank)，可以让我们通过编写简单的配置文件去生成本地的Makefile，这个配置文件是****独立于运行平台和编译器****的，这样就不用亲自去编写Makefile了，而且配置文件可以直接拿到其它平台上使用，无需修改，非常方便。

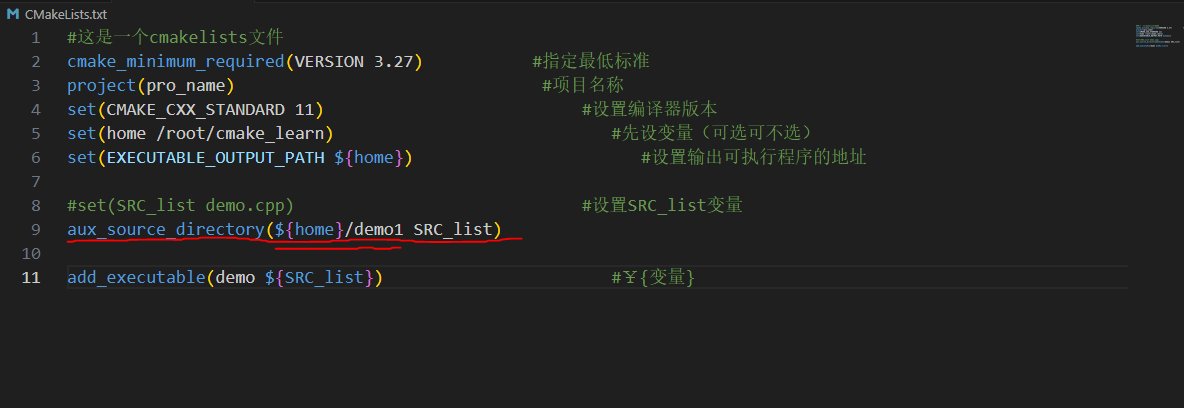
创建CMakeLists.txt: touch CMakeLists.txt

使用方法

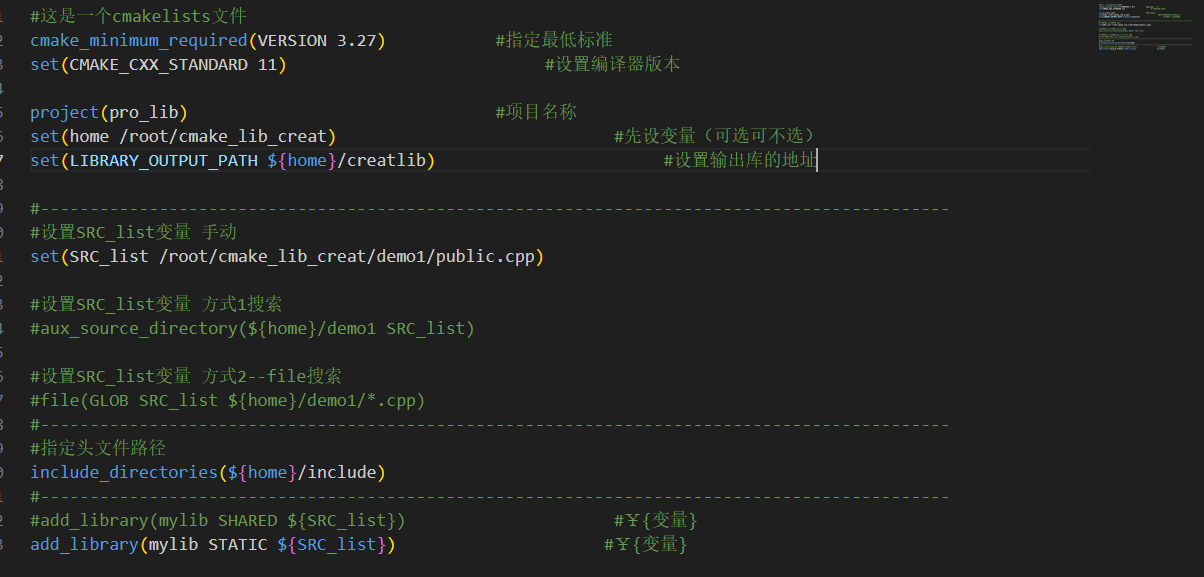




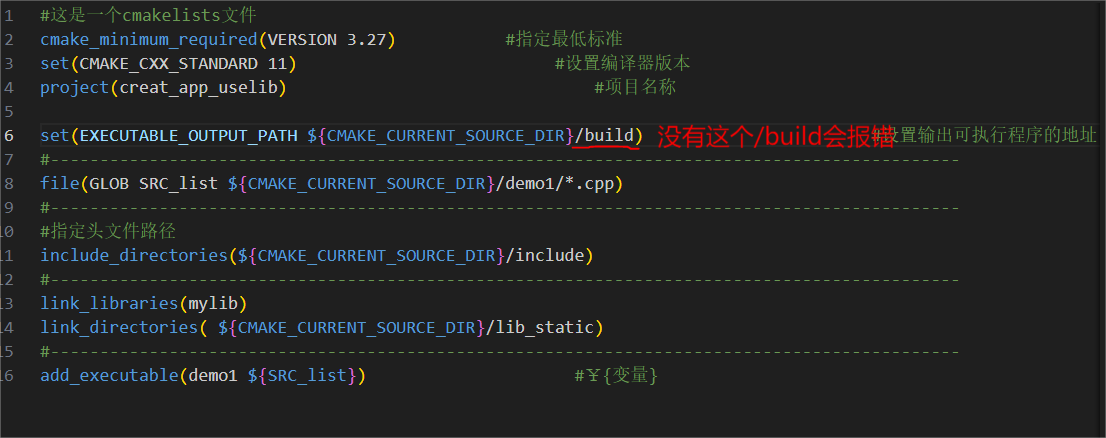
方式一搜索指定目录中所有源文件



制作库文件



使用库需要提供库文件和头文件



链接动态库写在最后



Message



## 字符串

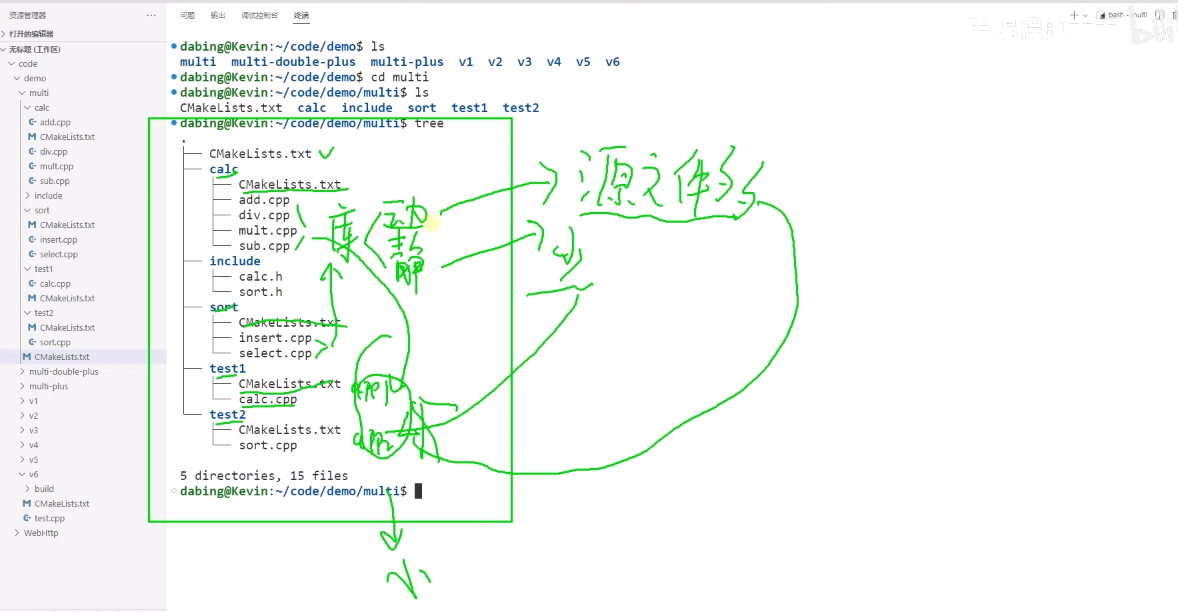
set(变量名1 ${变量名1} ${变量名2} ...)

## 嵌套的CMake

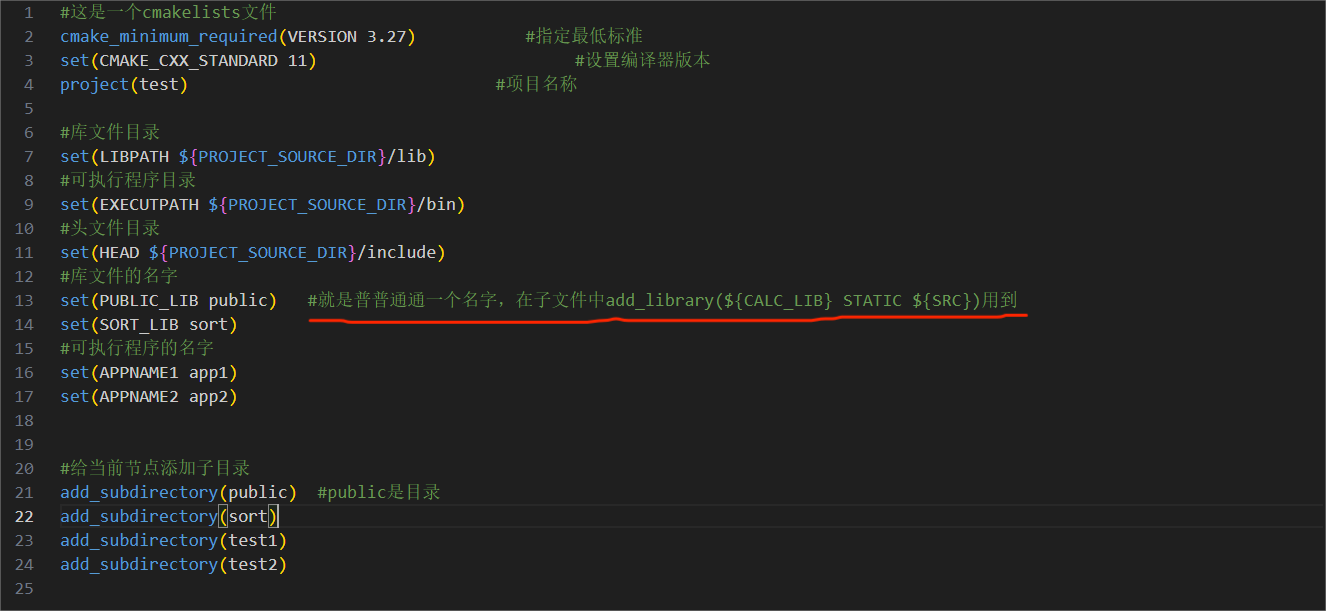
在 CMake 中是用add\_subdirectory建立父子节点关系的：

add\_subdirectory(source\_dir [binary\_dir] [EXCLUDE\_FROM\_ALL])

source\_dir：指定了子目录的位置，剩下两个参数忽略



最主要是定义的几个add\_subdirectory,上面变量都是点缀



其实就是执行下面四行

Vscode ctrl+shift+L 同时选择所有形同变量