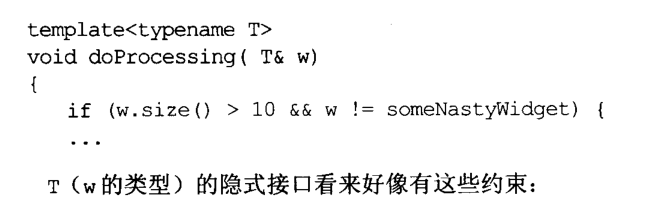


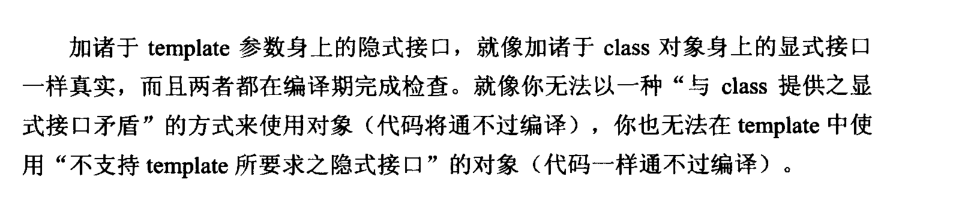
**条款41**

**隐士接口的要求在编译器阶段完成**



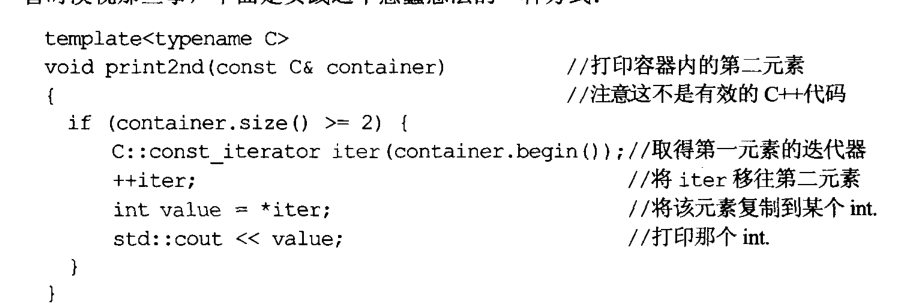
1.对于doprocessing 这个类，T 类型还是不知道的，所以不知道T类中的函数，所以来说T类型的函数是隐士接口，

2.w.size() >10 && w!= someNastyWidget 都不知道w的类型，咋知道调用size函数>10返回一个bool值呢！这个导致直接报错，所以编译器就会要求在编译阶段验证 T的类型是否支持doProcessing要求，也就是图片。



**条款 42**

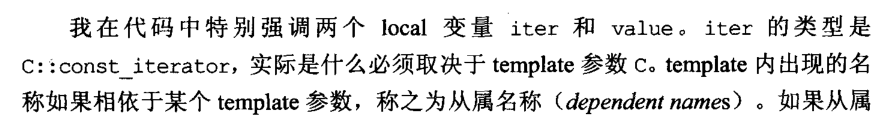
**typename让名称是否是类型名称或称变量名称得到确定**



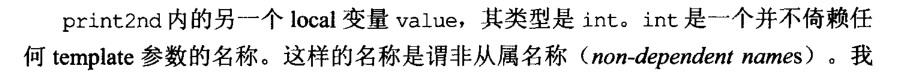
**从属名称和非从属名称**

这个模板函数print2nd存在两个变量，iter和value

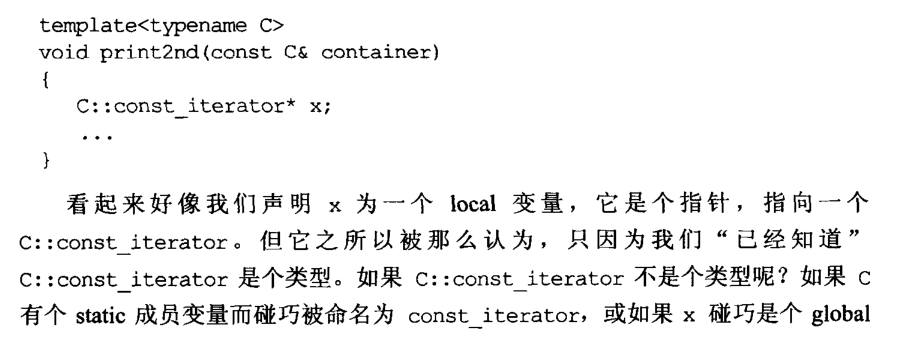
iter

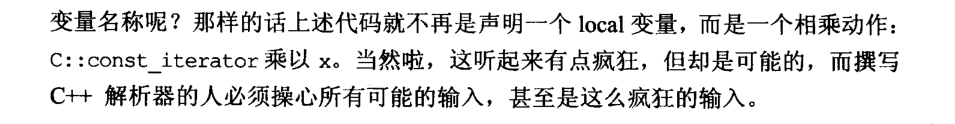


value

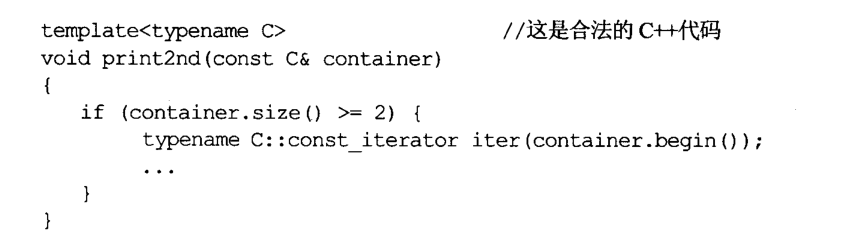


**面临问题**





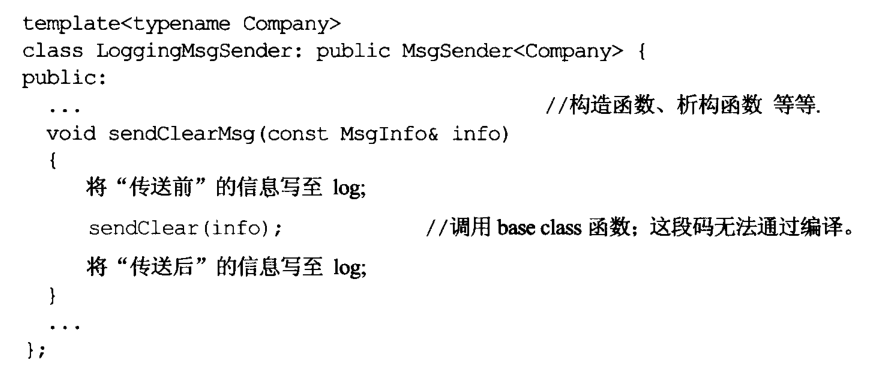
**typename可以让他定义清楚针对从属名称的问题**



**条款43**

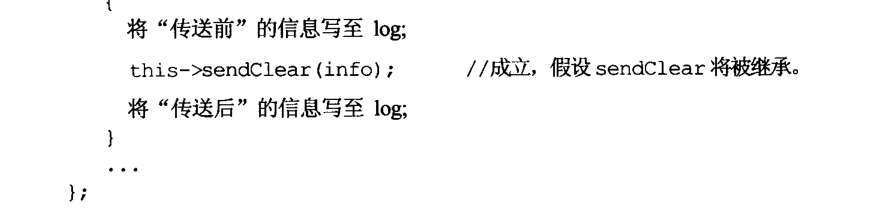
**这个类存在一定问题**

1.这个类在继承基类（且这个基类是一个模板）的时候，他的基类只在执行期被解读出来，所以，在编译器不会通过sendClear（info）这个调用，这个是基类的函数，编译期不知道MsgSender<Company>是不是含有这个函数，因为他是模板，所以存在单独特例化，这样就可以不会含有这个函数（即使原来模板含有，重写）

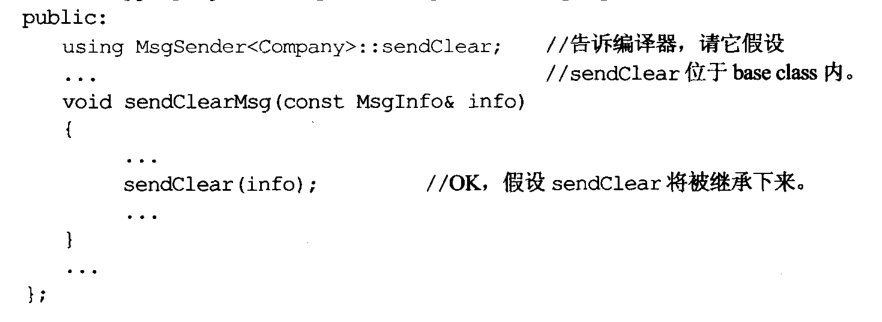


**三种方法**

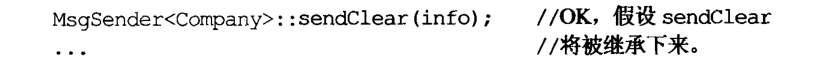
1.this-> ,假设继承了这个函数



2.using 使用



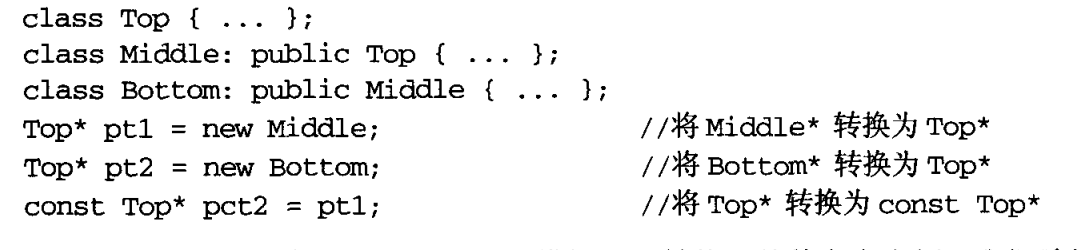
3.直接写清楚



**条款44**

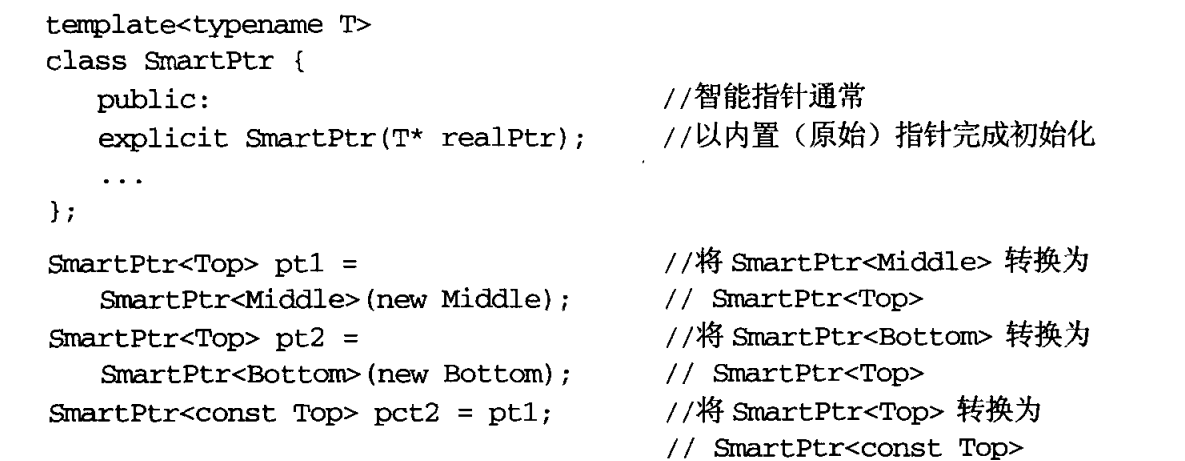
无法理解

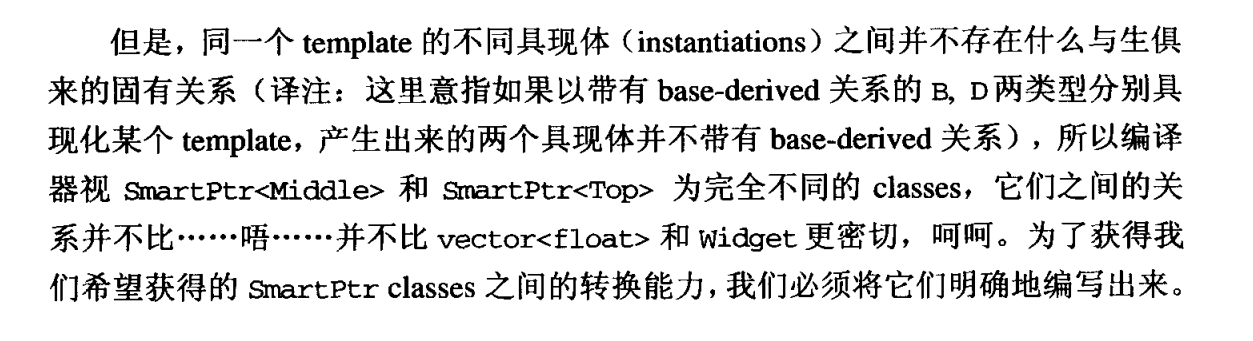
**条款45**

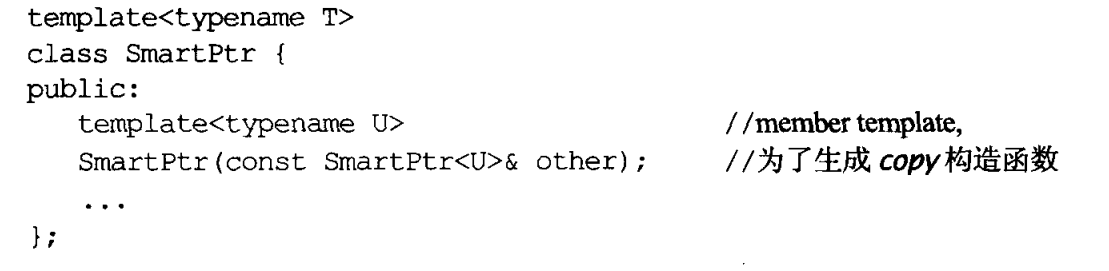


1.

**问题所在，模板类，他们的继承关系不存在，所以需要我们自己实现隐士转换**

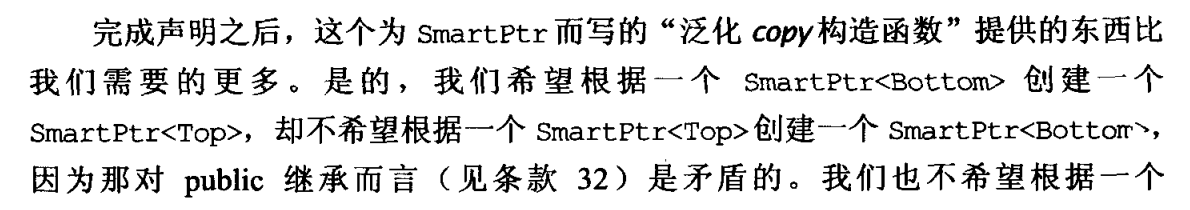


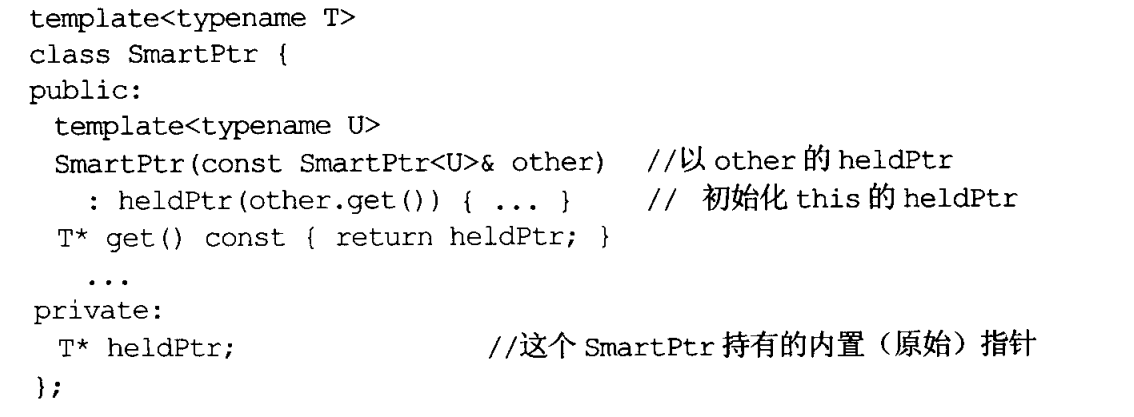




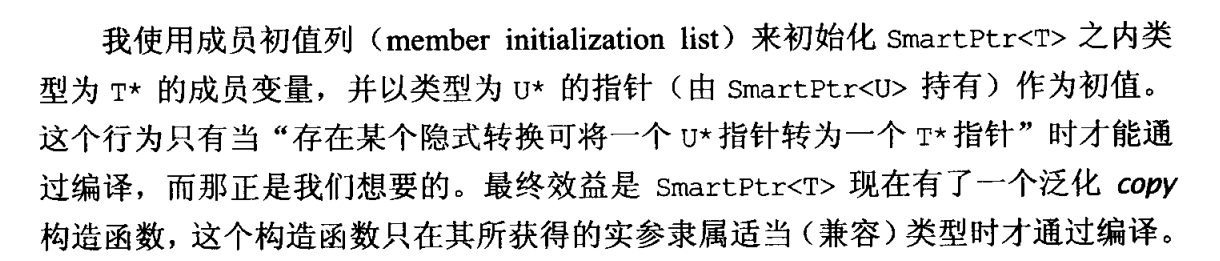
解决方法

模板《T》接受一个模板《U》同时不是模板的继承转变也是采用的隐士转换，所以这里依旧采用





解决思路



1.模板化的赋值函数也可以采用这种转换

2.这种操作不会改变语言规则，模板化的初始化函数，如果你没有实现非模板化的初始函数，编译器会帮你实现一个，但是像前面如果T和U类型一样，模板化初始函数就得到具象化，就是和非模板初始化函数一样。就不会帮你实现了。

**条款46 需要类型转换时把模板（需要执行隐士转换的模板函数）定义成firend函数**

