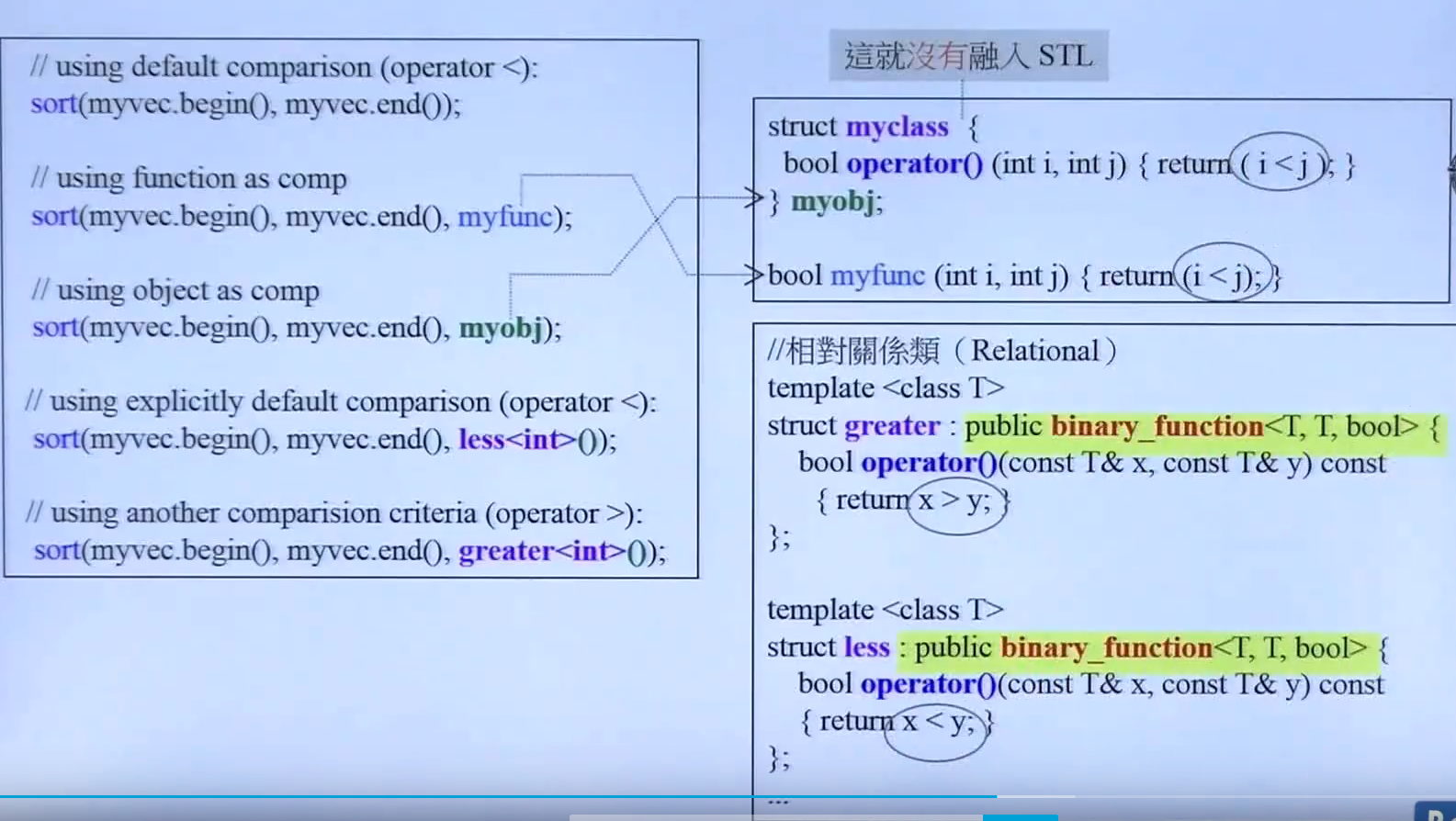
**仿函数**

**定义**

实际上我们定义的是一个类，在类中我们实现小括号的函数，这个类的被用在算法中，看起来像函数，所以我们称他为仿函数。



我们注意左边第三行第三个参数 myfunc 函数，这是我们自己实现的，我们的确重新实现了小括号，但是注意我们没有继承相关类，就像图片中所说的一样这个仿函数没有融入STL。

关于仿函数的调用，前面的仿函数的定义已经说明了他是一个类，所以我们要调用这个类的对象，也就是图片左边第四行的less<int>()这个临时对象。

**为什么我们的仿函数需要继承 unary-function（一个参数）和binary-function（两个参数）**

观看unary-function和binary-function源码我们可以发现，我们这俩个类重新命名了很多名称，这个操作其实与迭代器相似，这些名称的定义其实是回答一些问题，是为了让算法问仿函数一些问题，算法委托适配器来问仿函数，所以这样的继承实际上是为了更好的更加方便的改进仿函数，不继承意味没法有被改进的机会，同时我们不继承会在运行时遇到一些问题，

