Lesson 12

1.补充：在 public 继承下，Base 的 private、protected、public 在 Derived 中的权限为不可访问、protected、public。

2.不含有任何数据成员或者虚函数的 class 或者 struct 大小为 1，含有虚函数的对象在基地址部分有一个 vptr，指向虚函数表。

3.动态绑定的原理：假设派生类和基类存在覆盖的关系，那么派生类在虚函数表中，会覆盖掉基类相应的虚函数。当程序执行的时候，根据基类指针找到 vptr，根据 vptr 找到 vtable，然后找到相应的版本去执行。所以执行的是覆盖的版本，而具体被哪个版本覆盖是由具体的对象类型所决定的，所以才实现了根据对象的具体类型去调用相应的函数。

4.当类中含有虚函数的时候，需要把析构函数设为 virtual 析构函数。

5.函数声明为 virtual，意味着需要由用户去继承，以重新实现。

6.不要试图重定义基类的非 virtual 函数。

7.虚函数尤其是纯虚函数，相当于制定了一种约定、契约，凡是继承并改写（或者实现纯虚函数）的子类，都必须遵守这一约定。

8.句柄类（Handle）：试图把一个继承体系中的多种对象放入

一个数组。采用指针会造成内存管理的极度混乱。

解决方案：

把 Animal 指针封装在一个 Handle 类中，这个 Handle 类实现的是深拷贝，这样每个 Handle 相互独立。

为了实现 Handle 的深拷贝，我们在 Animal 中添加 copy 虚函数，这样就可以通过 Animal*达到复制实际对象的目的（Dog、Cat），这是一种多态。

为了实现通过 Handle 可以操控 Animal 系列的对象，我们为 Handle 重载了->操作符，使它表现的像一个指针。（还有一种方案，在 Handle 中实现 display，然后通过 ptr 去调用 Animal 内部的 display）。

封装句柄的最终目的是在数组、vector 中封装 Animal 系列对象的多态行为。

目前这个句柄的特点：Animal 系列的继承体系对用户是可见的。

可以改进的地方：把深拷贝改为引用计数。

9.句柄类和智能指针的主要区别：

a) 智能指针主要目的在于使用 RAII 管理资源，实现资源的自动释放。

b) 句柄类也实现了智能指针的功能，但是其主要目的是为了在数组中实现多态。

10.T::size_type *p 这句话在模板中存在歧义，可以把

T::size_type 解释成一种类型，所以这里定义了一个指针 p，还可以把 T::size_type 解释成一个变量，所以这样可以看做乘法。解决方案就是在前面加上 typename， 来说明这是一个定义，而不是乘法。typename T::size_type *p;

11.编写模板类的注意点：

a) 类的声明和实现放到同一个 hpp 文件中
b) 所有函数均为 inline
c) 每个函数在类外实现时都要加上模板参数列表
d) 类名要写完整，例如 SmartPtr<T>，不能漏掉尖括号。因为 SmartPtr 不是完整的类。

12.练习：之前的 Queue 改为模板。