Containers and Inheritance

容器不容许存储不同类型的元素，所以我们不能把具有继承关系的多种类型直接存放在容器中。

```
class Base{
    private:
        int base_mem;
    public:
        Base(const int &v=0):base_mem(v){}
};
class D:public Base{
    private:
    int d_mem;
    //...
};
vector<Base> vec;
vec.push_back(Base(1));// ok

vec.push_back(D(1,2));// derived-class part will be ignored
```

> Because derived objects are "slice down" when assigned to a base-type object, containers and types related by inheritance do not mix well.

Put (Smart) Pointers, Not Objects, in Containers

When we need a container that holds objects related by inheritance, we typically define the container to hold pointers (preferably smart pointers) to the base class.

```
vector<shared_ptr<Base>> vec;
vec.push_back(make_shared<Base> (/*parms*/));

// we can convert a smart pointer to a derived type to a smart pointer
// to an base-class type.
vec.push_back(make_shared<Derived> (/*parms*/));
```

`make_shared<Derived>`returns a `shared_ptr<Derived>` object, which is converted to `shared_ptr<Base>` when we call `push_back`.