覆盖

覆盖的概念

- 父类有两个虚函数，子类重写了一个，子类的虚表有几项?
  - 答：两项
- 如果子类新增虚函数，而父类没有，则这个虚函数会存放在子类的虚表中，位于已重写的虚函数的后面
- 虚表中函数顺序是父类定义虚函数的顺序
- 当子类重写父类虚函数时，子类重写后的虚函数会替换子类虚表中父类虚函数的位置，称之为函数覆盖

重载、隐藏与覆盖

重载

```
class Testclass {
    ...
    void foo()
    {
        ...
    }

    void foo(int a)     // 重载foo
    {
        ...
    }
    ...
}
```

- 作用域相同
- 函数名相同，参数列表不同
- 返回值与调用约定**不做**考虑

隐藏

```
class A {
    ...
    void foo()
    {
        ...
    }
    ...
}

class B : public A {
    ...
```

```
    void foo()  // 隐藏父类的foo
    {
        ...
    }
    ...
}
```

- 子类函数隐藏父类函数
- 作用域不同（基类与派生类）
- 函数名相同
- 参数列表、访问权限控制关键字都**不做**考虑

覆盖

```
class A {
    ...
    virtual void foo()
    {
        ...
    }

    virtual A * bar()
    {
        ...
    }
    ...
}

class B : public A {
    ...
    virtual void foo()  // 覆盖父类的foo
    {
        ...
    }

    virtual B * bar()   // 协变，可以覆盖父类的bar
    {
        ...
    }
    ...
}
```

- 作用域不同（基类与派生类）
- 至少父类函数有`virtual`关键字
- 函数声明完全相同，特例为**协变**

协变

> 在C++中，只要原来的返回类型是指向类的指针或引用，新的返回类型是指向派生类的指针或引用，覆盖的方法就可以改变返回类型。这样的类型称为协变返回类型（Covariant returns type).

调用规则

调用者的类型来决定查找的起点

1. 在调用者的类中，查找同名函数：
    - 如果没有，则往上一层查找，如果均找不到，则报错（未定义）
    - 如果有，则不会往上面查找，可见的域就是当前找到的同名函数所在的域（函数隐藏）
2. 在当前可见的域中，找到最佳函数。（函数重载规则）
    - 如果唯一的最佳函数不是虚函数，那么该调用是直接调用。
    - 如果唯一的最佳函数是虚函数,判断调用者是否是指针或引用
        - 调用者是指针或引用，则是间接调用（函数覆盖规则）
        - 调用者不是指针或引用，则是直接调用

继承和组合

is-a

> ClassB is a ClassA
> ClassA 是抽象，ClassB 是继承。因此可以说——ClassB 是 ClassA 的子类
> 当只需要考虑 ClassA 的特性时，ClassB 和 ClassA 基本等价

has-a

> ClassB has a ClassA
> ClassB 是容器，ClassA 是部件。有时会说成——ClassA 是 ClassB 的「子类」
> ClassB 和 ClassA 一般不可相互替代