重载、重写（覆盖）和隐藏

重载（overload）

重载是指在同一范围定义中的同名成员函数才存在重载关系。主要特点是函数名相同，参数类型和数目有所不同，不能出现参数个数和类型均相同，仅仅依靠返回值不同来区分的函数。重载和函数成员是否是虚函数无关。举个例子：

```
class A{
    ...
    virtual int fun();
    void fun(int);
    void fun(double, double);
    static int fun(char);
    ...
}
```

重写（覆盖）（override）

重写指的是在派生类中覆盖基类中的同名函数，重写就是重写函数体，要求基类函数必须是虚函数且：

- 与基类的虚函数有相同的参数个数
- 与基类的虚函数有相同的参数类型
- 与基类的虚函数有相同的返回值类型

举个例子：

```
// 父类
class A{
public:
    virtual int fun(int a){}
}
//子类
class B : public A{
public:
    //重写,一般加override可以确保是重写父类的函数
    virtual int fun(int a) override{}
}
```

重载与重写的区别：

- 重写是父类和子类之间的垂直关系，重载是不同函数之间的水平关系
- 重写要求参数列表相同，重载则要求参数列表不同，返回值不要求
- 重写关系中，调用方法根据对象类型决定，重载根据调用时实参表与形参表的对应关系来选择函数体

隐藏（hide）

隐藏指的是某些情况下，派生类中的函数屏蔽了基类中的同名函数，包括以下情况：

- 两个函数参数相同，但是基类函数不是虚函数。和重写的区别在于基类函数是否是虚函数。举个例子：

```cpp
// 父类
class A{
public:
    void fun(int a){
    cout << "A中的fun函数" << endl;
}
};
// 子类
class B : public A{
public:
    // 隐藏父类的fun函数
    void fun(int a){
    cout << "B中的fun函数" << endl;
    }
};
int main(){
    B b;
    b.fun(2);    // 调用的是B中的fun函数
    b.A::fun(2); // 调用A中fun函数
    return 0;
}
```

- 两个函数参数不同，无论基类函数是不是虚函数，都会被隐藏。和重载的区别在于两个函数不在同一个类中。举个例子：

```cpp
// 父类
class A{
public:
    virtual void fun(int a){
        cout << "A中的fun函数" << endl;
    }
};
// 子类
class B : public A{
public:
    //隐藏父类的fun函数
    virtual void fun(char* a){
        cout << "A中的fun函数" << endl;
    }
};
int main(){
    B b;
    b.fun(2);    // 报错，调用的是B中的fun函数，参数类型不对
    b.A::fun(2); // 调用A中fun函数
    return 0;
}
```