

1. **重载：**
2. **概念：**在同一个作用域内；函数名相同，参数列表不同（参数**个数不同**，或者参数**类型不同**，或者**参数个数和参数类型都不同**），返回值类型可相同也可不同；这种情况叫做c++的重载！注意：c语言没有函数重载的机制；
3. **c++函数重载达到的效果**：调用函数名相同的函数，会根据实参的类型和实参顺序以及实参个数选择相应的函数；
4. c++函数重载是一种静态多态（又叫做静态联编，静态绑定，静态决议）

**二、重写（又叫覆盖）**

1. 说覆盖之前先说一个概念： **虚函数：**类的**成员函数前面加virtual关键字**，则这个成员函数称为虚函数
2. **覆盖（重写）的前提条件**：父类函数为虚函数；
3. **覆盖（重写）的概念**：当在子类中定义了一个与父类**完全相同**的虚函数时，则称子类的这个函数**重写（也称覆盖）**了父类的这个虚函数。
4. **什么是在子类中定义了一个与父类完全相同的虚函数，**有两种情况：
5. 就是说子类中的虚函数和父类中的虚函数，**函数名，参数个数，参数类型，返回值类型都相同**；这种情况下子类的这个虚函数重写的父类中的虚函数，构成了重写；
6. **协变**—是说子类中的虚函数和父类中的虚函数，**函数名，参数个数，参数类型都相同，只是返回值类型不同；父类的虚函数返回父类的指针或者引用，子类虚函数返回子类的指针或者引用**；这种情况下子类的这个虚函数也重写了父类中的虚函数，也构成了重写；——我们**把这种特殊的情况叫做协变**
7. 覆盖（重写）达到的效果，**注意：**
   1. 在子类中定义了一个与父类虚函数完全相同的函数，那么这个子类的函数就是重写了父类的虚函数，此时这个子类的函数就是虚函数，如果不显示的加上virtual修饰，编译器也会默认为虚函数；
   2. 在子类中重写了父类的虚函数，那么子类对象调用该重写函数，调用到的是子类内部重写的虚函数，而并不是从父类继承下来的虚函数；（这其实就是动态多态的实现）；
   3. 在子类中重写了父类的虚函数，如果用一个父类的指针（或引用）指向（或引用）子类对象，那么这个父类的指针或用引用调用该重写的虚函数，调用的是子类的虚函数；相反，如果用一个父类的指针（或引用）指向（或引用）父类的对象，那么这个父类的指针或用引用调用该重写的虚函数，调用的是父类的虚函数

**三、重定义（隐藏）**

1. **隐藏（重定义）概念：**是指在**不同的作用域中**（分别在父类和子类中），**函数名相同，**不能构成重写的都是重定义。
2. **隐藏（重定义）的使用范围：**重定义的**不光是类的成员函数**，还可以是**类的成员变量**；
3. **隐藏（重定义）的直接效果：**如果在父类和子类中有相同名字的成员；那么在子类中。会将父类的成员隐藏；隐藏以后的直接效果就是：无论在子类的内部或者外部（通过子类成员）访问该成员；全都是访问子类的同名成员； 如果在子类内部或者外部（通过子类成员）访问同名的成员函数，则需要根据函数调用的规则来调用子类的同名成员函数；否则调用失败；