成员函数会根据限定符的不同而对函数调用选择不同

this指针的含义

```
    struct demo1 {
        void print() {
            std::cout << a << " " << b << " " << c << std::endl;
        }
        int a, b, c;
    };

    demo1 a{1, 2, 3}; // 聚合初始化

    a.print(); // 123
    // 等价
    // print(demo1* this)
    // print(&a)
```

关于成员函数限定符调用规则

当类中只有const修定时会调用const函数

```
    struct demo2 {
        void print() const {
            std::cout << "demo2 const\n";
        }
    };

    demo2 b;
    b.print();//demo2 const
```

当同时存在两种函数时,非const对象调用非const成员函数

```
    struct demo3 {
        void print() const {
            std::cout << "demo3 const\n";
        }
        void print() {
            std::cout << "demo3\n";
        }
    };

    demo3 c;
    c.print();

    const demo3 d;
    d.print(); // "demo3 const" const对象调用const成员函数,const对象无法调用非const函
数
```

&和&&限定

```
struct demo4 {
    void print() &{
        std::cout << "demo4 &\n";
    }

    void print() &&{
        std::cout << "demo4 &&\n";
    }
};

demo4 e;
e.print();//"demo4 &"普通对象调用&版本成员函数
demo4 &f = e;
f.print();//"demo4 &"左值引用对象调用&成员函数

std::move(e).print();//"demo4 &&"右值调用&&成员函数
```