浅拷贝就是之前所用到的一些拷贝，把一些对象的引用传入之后进行拷贝

浅拷贝（Shallow Copy）

- **定义**：浅拷贝仅仅复制对象的成员变量的值，如果成员变量是指针，则复制指针的值（即地址），而不复制指针所指向的数据。
- **结果**：两个对象的成员指针指向同一块内存地址，如果一个对象释放了这块内存，另一个对象仍然引用这块已经被释放的内存，导致悬挂指针或双重释放等问题。

那么深拷贝就是把传入的对象的所有指针成员分配新内存

深拷贝（Deep Copy）

- **定义**：深拷贝不仅复制对象的成员变量的值，而且为所有指针成员分配新的内存，并复制指针所指向的数据。
- **结果**：每个对象都有自己独立的一份数据副本，修改一个对象的数据不会影响另一个对象，避免了悬挂指针或双重释放的问题。

```
class MyClass {
public:
    int* ptr;
    MyClass(int value) {
        ptr = new int(value);
    }
    // 浅拷贝构造函数
    MyClass(const MyClass& other) {
        ptr = other.ptr;//仅仅只是把原来对象的值拷贝了过来，其实本质上他们还是同一个内存区域
    }
    // 深拷贝构造函数
    MyClass(const MyClass& other, bool deepCopy) {
        if (deepCopy) {
            ptr = new int(*other.ptr);//主要的区别也就是在这里
        } else {
            ptr = other.ptr;
        }
    }
    ~MyClass() {
        delete ptr;
    }
};
int main(){
    Myclass obj1(1);
    Myclass obj2(obj1);
    Myclass obj3(obj1, 1);
}
```

注意深拷贝的写法，通常这样都是在成员变量存在指针的时候使用

`int* ptr;`

`ptr = new int(*other.ptr);`

这里先用了一个关键字new，代表给后面的数值分配了一块新的内存空间

后面首先接数据类型，也就是int，跟上面一致，之后里面写传入的对象的那个属性成员变量也就是ptr，用`other.ptr`来表示，那么前面的这个*也就是代表这其实是一个指针，姑且也算是和上面的成员变量数据类型一致吧，这里要记住了。