C++Primer学习笔记（第13章）

第十三章 拷贝控制

拷贝构造函数

- 如果一个构造函数的第一个参数是自身类型的引用，且额外的参数都有默认值，则此构造函数是拷贝构造函数
- 直接初始化调用的是构造函数，拷贝初始化调用的是拷贝构造函数

拷贝初始化发生场景：

- 将一个对象作为实参传递给一个非引用形参
- 从一个返回类型为非引用类型的函数返回一个对象
- 用花括号列表初始化一个数组中的元素或者一个聚合类中的元素

```C++
class MyClass{
    public:
    MyClass(const MyClass &c);//拷贝构造函数
};
```

重载赋值运算符

- 重载运算符本质上是函数，其由 operator 关键字后面接表示要定义的运算符的符号组成，故赋值运算符就是一个名为 operator=的函数
- 重载运算符的参数表示运算符的运算对象
- 如果运算符作为一个成员函数，其左侧运算对象就绑定到隐式的 this 参数

```C++
class MyClass{
    public:
    MyClass& operator=(const &MyClass);
};
```

合成拷贝赋值运算符

- 如果一个类未定义自己的拷贝赋值运算符，编译器会为它生成一个合成拷贝赋值运算符

析构函数

- 析构函数用来执行与构造函数相反的操作，销毁对象的非 static 成员

```C++
class MyClass{
    public:
    ~MyClass();
};
```

调用析构函数的时机：

- 变量在离开作用域的时候
- 当一个对象被销毁时
- 容器被摧毁时
- 对于动态分配的对象，当指向他的指针应用 delete 时
- 对于临时对象，当创建它的完整表达式结束时被摧毁

当指向一个对象的引用或者指针离开作用域时，析构函数不会被执行

拷贝与拷贝赋值运算符的调用时机：

```C++
MyClass m;//调用构造函数
MyClass n(m);//调用拷贝函数
MyClass t=n;//调用拷贝函数
t=m;//调用拷贝运算符
```

三五法则

- 如果这个类需要一个析构函数，几乎肯定它需要一个拷贝构造函数和一个拷贝赋值运算符（深拷贝与浅拷贝问题）
- 需要拷贝操作的类也需要赋值操作，反之亦然

阻止拷贝

- 我们可以通过将拷贝构造函数和拷贝赋值运算符定义为删除的函数来阻止拷贝
- 删除的函数：我们虽然声明了它，但不能以任何方式使用它，在函数的参数列表后面加上=delete 来指出
- 我们可以对任何函数指定=delete，但是只能对合成的默认的构造函数或拷贝控制函数使用=default
- 析构函数是不能删除的成员
- 可以使用 private 来进行拷贝控制

拷贝控制与资源管理

- 当拷贝一个像值的对象时，副本与原对象时完全独立的，改变副本不会对原对象造成任何影响
- 当我们拷贝一个像指针的类的成员时，副本与原对象使用相同的底层数据

移动构造函数

- 只需移动数据，无需复制，是一种浅拷贝，大大降低了内存新的分配

右值引用

- 用于绑定到一个将要销毁的对象
- 可以将一个右值引用绑定到表达式上，但不能直接绑定到一个左值上

```C++
int i=12;
int &r =i;
int &&rr=i;//错误，不能将一个右值引用绑定到一个左值上
int &r2 =i*42;//错误，其是一个右值
const int &r3=i*42;//正确，可以将一个const引用绑定到一个右值上 因为其不能修改是安全的
int &&rr2 =i*42;//正确
```

右值和左值引用的成员函数

- 为了阻止给一个右值赋值，常在参数列表后加一个引用限定符&
- 可以根据引用限定符来区分重载限定符

```C++
class MyClass{
    public:
    MyClass sorted()&&;//用于可改变的右值
    MyClass sorted()&;//用于任何类型
};
MyClass& ret1();//返回一个引用
MyClass ret2();//返回一个右值
//两种调用情况
ret1.sorted();
ret2.sorted();
```