C++程序设计上机报告：拷贝构造函数的用法

姓名	周新斌	学号	2118140201
班级	软工 212		
目的及要求	（1）掌握类的定义和实现的方法 （2）掌握类中的构造函数的定义和调用 （2）掌握类中的拷贝构造函数定义和调用 （3）掌握类中的析构函数的定义和调用 （4）掌握静态对象的创建和释放		
上机学时	2 学时		
设备要求	（1）主要仪器设备：微型计算机 （2）软件环境：WINDOWS 2000／XP 操作系统；Visual C++ 程序设计语言		
上机内容	设计一个 Rectangle 类，类中包含： 数据域：长和宽 一个带有默认参数的有参构造函数（函数中使用 this 指针） 一个拷贝构造函数 一个析构函数 一个全局函数 Rectangle f（Rectangle r）（在其中定义个 static 对象，返回该对象） 编写一个主函数，测试类中的各个函数。验证拷贝构造函数的三种调用方式。（注意：需要调用两次 f 函数，判断静态对象的特点）		

源代码	#include<bits/stdc++.h> using namespace std; class Rectangle { private: double width; double height; public: Rectangle(double width = 0, double height = 0) { this -> width = width; this -> height = height; } Rectangle(const Rectangle &r) { this -> height = r.height + 1; this -> width = r.width + 1; cout << "拷贝构造函数被调用: " << this -> width << ' ' << this -> height << endl; } ~Rectangle() { cout << this -> width << ' ' << this -> height << " 摧毁" << endl; } void setWidth(int width) { this -> width = width; } double getWidth() { return width; } double getHeight() { return height;

	`` } }; // 参数会调用拷贝构造函数创建新的对象 Rectangle f(Rectangle r) { cout << "新的 r 对象: " << r.getWidth() << ' ' << r.getHeight() << endl; static Rectangle static_object(99, 99); cout << "static_object: " << static_object.getWidth() << ' ' << static_object.getHeight() << endl; // 返回会调用拷贝函数创建新的对象 return static_object; } int main() { Rectangle r; // 0 0 // 第一种调用拷贝构造函数 Rectangle r2(r); // 1 1 // 第二种调用拷贝构造函数 Rectangle r3 = r2; // 2 2 // 第三种调用拷贝构造函数 // 两次调用 f 验证全局对象 Rectangle R1 = f(r3); // 3 3 Rectangle R2 = f(r3); return 0; } ```
程序的输入描述	无
程序的输出结果	拷贝构造函数被调用: 1 1 拷贝构造函数被调用: 2 2 拷贝构造函数被调用: 3 3 新的 r 对象: 3 3 static_object: 99 99 拷贝构造函数被调用: 100 100 3 3 摧毁 拷贝构造函数被调用: 3 3

	新的 r 对象: 3 3 static_object: 99 99 拷贝构造函数被调用: 100 100 3 3 摧毁 100 100 摧毁 100 100 摧毁 2 2 摧毁 1 1 摧毁 0 0 摧毁 99 99 摧毁
程序难点分析	**static 对象的特点是什么：** 1. 静态对象的生命周期与程序的生命周期相同，即在程序开始执行时创建，在程序结束时销毁。 2. 静态对象存储在静态存储区域（静态数据区），不存储在函数的栈上。 3. 静态对象在程序执行之前就已经初始化，其初始化顺序与其定义顺序相关，并且只初始化一次。 4. 静态对象具有全局可见性，可以在定义它的文件之外的其他文件中访问。 5. 如果静态对象没有显式初始化，它们会被默认初始化为零值（如果是内置类型）或者调用默认构造函数进行初始化（如果是类类型）。 6. 静态对象的析构函数在程序结束时被调用，与静态对象的创建顺序相反，即在程序结束时以相反的顺序销毁。 **在你的程序里，你设计的三次调用拷贝构造函数的地方都在哪里？** 1. 创建对象时把另一个对象作为参数； 2. 在声明一个新的对象时，直接把一个已存在的对象赋值给自己； 3. 函数的参数是对象，调用函数时，会调用拷贝构造函数，创建新的对象。 **如果拷贝构造函数的参数设置为对象，而不是对象的引用，是否可行？为什么？** 在 C++中，拷贝构造函数的参数通常被设计为对象的引用（const 引用），而不是直接的对象。这是因为如果拷贝构造函数的参数是对象本身而不是引用，会导致额外的拷贝操作，从而陷入无限循环的情况。具体来说，如果拷贝构造函数的参数是对象本身，那么在调用拷贝构造函数时，又会创建一个新的对象作为参数，这样就会一直循环下去，直到耗尽内存或栈溢出。