C 类对象的复制—拷贝构造函数 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/264/2021_2022_C___E7_B1_BB_E5_AF_B9_E8_c97_264476.htm 在学习这一章内容前我们已经学习过了类的构造函数和析构函数的相关知识，对于普通类型的对象来说，他们之间的复制是很简单的，例如： int a = 10. int b =a. 自己定义的类的对象同样是对象，谁也不能阻止我们用以下的方式进行复制,例如： #include using namespace std. class Test { public: Test(int temp) { p1=temp. } protected: int p1. }. void main() { Test a(99). Test b=a. } 普通对象和类对象同为对象，他们之间的特性有相似之处也有不同之处，类对象内部存在成员变量，而普通对象是没有的，当同样的复制方法发生在不同的对象上的时候，那么系统对他们进行的操作也是不一样的，就类对象而言，相同类型的类对象是通过拷贝构造函数来完成整个复制过程的，在上面的代码中,我们并没有看到拷贝构造函数，同样完成了复制工作，这又是为什么呢？因为当一个类没有自定义的拷贝构造函数的时候系统会自动提供一个默认的拷贝构造函数，来完成复制工作。 下面，我们为了说明情况，就普通情况而言(以上面的代码为例)，我们来自己定义一个与系统默认拷贝构造函数一样的拷贝构造函数，看看它的内部是如何工作的！ 代码如下： #include using namespace std. class Test { public: Test(int temp) { p1=temp. } Test(Test amp.c_t)就是我们自定义的拷贝构造函数，拷贝构造函数的名称必须与类名称一致，函数的形式参数是本类型的一个引用变量,且必须是引用。 当用一个已经初始化过了的自定义类类型对象去初始化另一个新构造的对象的时候，拷贝

构造函数就会被自动调用，如果你没有自定义拷贝构造函数的时候系统将会提供给一个默认的拷贝构造函数来完成这个过程，上面代码的复制核心语句就是通过Test(Test &amp.c_t)拷贝构造函数内的p1=c_t.p1.语句完成的。如果取掉这句代码，那么b对象的p1属性将得到一个未知的随机值；下面我们来讨论一下关于浅拷贝和深拷贝的问题。 就上面的代码情况而言，很多人会问到，既然系统会自动提供一个默认的拷贝构造函数来处理复制，那么我们没有意义要去自定义拷贝构造函数呀，对，就普通情况而言这的确是没有必要的，但在某写状况下，类体内的成员是需要开辟动态开辟堆内存的,如果我们不自定义拷贝构造函数而让系统自己处理，那么就会导致堆内存的所属权产生混乱，试想一下，已经开辟的一端堆地址原来是属于对象a的，由于复制过程发生，b对象取得是a已经开辟的堆地址，一旦程序产生析构，释放堆的时候，计算机是不可能清楚这段地址是真正属于谁的，当连续发生两次析构的时候就出现了运行错误。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com