类静态成员和非静态成员属于类还是对象

原创 ysr1980 2007-07-15 20:59:00 3098 收藏 1 展开

C++程序的内存格局通常分为四个区:全局数据区(data area),代码区(code area),栈区(stack area),堆区(heap area)(即自由存储区).全局数据区存放全局变量,静态数据和常量;所有类成员函数和非成员函数代码存放在代码区;为运行函数而分配的局部变量,函数参数,返回数据,返回地址等存放在栈区;余下的空间都被称为堆区.

根据这个解释,我们可以得知在类的定义时,类成员函数是被放在代码区,而类的静态成员变量在类定义时就已经在全局数据区分配了内存,因而它是属于类的,我们是使用这个类静态成员变量的一个拷贝.对于非静态成员变量,我们是在类的实例化过程中(构造对象)才在栈区为其分配内存,是为每个对象生成一个拷贝,所以它是属于对象的.

创建对象后会产生拷贝,拷贝后的成员函数属于对象,原本的成员函数属于类

下面我们再来讨论下静态类成员函数和非静态成员函数的区别:

静态类成员函数和非静态成员函数都是在类的定义时放在内存的代码区的,因而可以说它们都是属于类的,但是类为什么只能直接调用静态类成员函数,而非静态类成员函数(即使函数没有参数)只有类对象才能调用呢?原因是类的非静态类成员函数其实都内含了一个指向类对象的指针型参数(即this指针),因而只有类对象才能调用(此时this指针有实值).

谈到这里,我们再来看下类中的虚函数的实现:类在定义的时候建立一张虚函数分配表(vtable),再通过一个指针(ptr)指向这张表,当子类改写某个虚函数的时候,只要改写vtable中相应的虚函数即可.

通常来去获取一个类的普通成员函数或者虚函数是没有实际意义的。

当然实际上是可以获取地址，你这里输出不一样是因为类成员函数特殊的调用机制问题。说起来比较复杂，除了静态成员函数都是跟对象绑定的，调用机制是一种—thiscall的玩意。之所以用cout无法输出准确值起始是因为<<运算符没有重载那么个类型，用printf函数时可以输出这个地址值的。成员函数的地址类型跟普通函数不一样，它实际上是类似 _thiscall A::f这种类型，用cout直接输出成员函数地址会编程bool型输出1，如果像你写的那样转换成void*，我没有具体看过，但是估计肯定转换中间和普通指针是不一样的。

换句话说，类中的成员函数，起始只要static函数时跟非类成员函数时一个样子的，其他成员函数跟普通函数不是一码事