拷贝构造函数/赋值函数

2015年11月29日 星期日　　下午3:43

一、简介

拷贝构造函数是一种构造函数，是"用拷贝的方式构造"。在初始化类的对象时，通过将另一个对象拷贝过来，可以视为"赋初值"。

赋值函数，是对象直接执行赋值操作(例如: a1 = a2)时，调用的函数。

二、目录

变量声明：

下面介绍的过程中，A表示类，a1、a2…表示A的对象。

```
class A {
        Int  _id;
        char*  _data;
        String  _name;
};
```

三、详述

1、所处的地位

每个类都一些最基本的函数。

每个类只有一个析构函数、一个赋值函数，可以有一个或多个构造函数。构造函数又分很多种，其中一种就是拷贝构造函数(也称作复制构造函数，copy constructor)。

```
A(void)  //默认无参数构造函数
~A(void)   //默认无参数析构函数
A(const A &a)  //默认拷贝构造函数
A &operator  =  (const A&a)   //默认赋值函数
```

2、调用方式

拷贝构造函数和赋值函数都是将一个已有的对象A a2的值"给"A a1。看起来功能是类似的，但是两个函数在调用方式和内部实现上有些差别。使用时要注意这些细节。

2.1 拷贝构造函数调用方式

```
A a1(a2);    //很直观，就是三、1中拷贝构造函数的示例，是C++风格的调用方式
A a1 = a2;   //有C语言的风格，虽然看起来像赋值，但是实际是调用拷贝构造函数，这里容易和赋值函数弄混，代码风格不好，不如上一种调用方式。
```

2.2 赋值函数

```
A a1;
A a2;
a1 = a2;  //直接用=，是赋值函数的运算符
```

3、编译器默认的实现方式，及可能存在的问题

三、1中说的几种最基本的函数，是每个类都需要有的，如果你定义一个自己的类，并没有显式的声明、实现这些函数，编译器会按照默认的方式进行实现。

如果定义的类比较复杂，直接使用默认的方式可能出现严重的问题。我们先看看默认实现的方式。(详见参考资料1)

3.1 深拷贝与浅拷贝

深拷贝和浅拷贝的区别，详见另一篇笔记(“浅拷贝与深拷贝”)

简单的介绍就是：

对于待复制对象内部的指针，浅拷贝拷贝地址，深拷贝拷贝内容。

浅拷贝做的只是简单的指针地址拷贝，目标对象的改指针指向的内容和源对象的内容是一样的;

深拷贝会把指针对应的内存内容拷贝过去(当然，这就需要在目标对象内部开一个独立的空间)目标对象的空间内容和源对象的空间内容以后就没什么关系了，是两段相互独立的空间。

有的文章中把浅拷贝叫做位拷贝，深拷贝叫做值拷贝。

3.2 默认的拷贝构造函数/赋值函数中，使用浅拷贝

3.3 默认的拷贝构造函数/赋值函数，存在的问题

默认的拷贝构造函数/赋值函数的问题，其实也就是浅拷贝和深拷贝的问题。
如果函数没有指针类型的变量，或者实现的逻辑中对这个指针变量没有"拷贝内容"的需求，使用默认是没问题的。

但是，如果涉及到浅拷贝和深拷贝。从3.1我们看出，浅拷贝虽然轻便，但是会有一些问题。

A 目标对象a1._data数据之前的内存没有释放，而且以后也没机会释放了(因为指针已经变了)，会造成内存泄露

B a1._data与a2._data会指向同一个内存区域，改变一方，会同时改变另一方。(但是我们的预期常常不是这样的，我们预期常常是a2 只是拷贝给 a1了，并不想绑定)

C 当a1或a2都执行内存释放时，a2._data会被释放两次

3.4 结论

A 当类中有指针变量的时候，我们需要自己定义拷贝构造函数

B 拷贝构造函数和赋值函数类似，因为A 所以我们也需要自己定义赋值函数

C 三法则：如果类需要析构函数，则它也需要赋值操作符和复制构造函数

4、重载拷贝构造函数

下面提供一个重载拷贝构造函数的示例

```
A::A(const A &a2)
{
    // 允许操作a2 的私有成员_data
int length = strlen(a2._data);
_data = new char[length+1]; //第一步,在目标对象中开空间,并将源对象内容拷贝给目标对象
strcpy(_data, a2._data);
}
```

5、重载赋值函数

重载赋值函数，其实就是做一个重载运算符的操作，他是在重装我们常见的赋值符号"="

```
A:: & operator = (const A &a2)
{
        If (this == &a2){     //第一步，判断是否是自拷贝
                return *this;
        }

        delete[] this->_data; //第二步，预先释放明白对象的内存区域，不然后面没机会释放了

        Int length = strlen(a2._data);
        _data = new char[length + 1];
        strcpy(_data, a2_data); //第三步,在目标对象中开空间,并将源对象内容拷贝给目标对象

        return *this;//第四步，返回目标对象(本对象)的引用,赋值函数的额外功能,我感觉没啥用
}
```

6、偷懒的办法

如果我自己实现一个类，成员变量中有指针，但是不想写，还怕别人瞎用，怎么办。

我们可以把拷贝构造函数和赋值函数声明为private的，然后自己不实现，也不让用默认的。是不是很吊。

7、

四、参考资料

1、简洁、思路清晰，对应三、3前面的部分
http://www.cnblogs.com/kaituorensheng/p/3245522.html

2、详细、层次不清晰，对应三、4~6
http://blog.chinaunix.net/uid-25808509-id-354211.html

3、详细，可以作为上面两个的补充
http://blog.chinaunix.net/uid-28662931-id-3496326.html

4、排版很不错 http://www.jellythink.com/archives/378