Singleton.h

namespace utils {

template <typename T>

class Singleton : private NonCopyable

{

public:

static inline T& Instance()

{

// 虽然返回的是值而非指针，但其继承了noncopyable类可以有效防止拷贝，其构造函数应该不能被设计为private,让T的父类Singleton可以构造对象

// 父类Singleton需要访问子类的构造函数，而子类的构造函数必须是public，连protect都不行，那不就向外部泄露构造函数了？！

static T instance\_;

return instance\_;

}

};

} // end namespace utils

总结：singleton类是一个单例模式，继承了noncopyalbe类，因此不能被拷贝和赋值，而构造也只能是子类来构造，singleton类应该是通过instance()来获得对象（这个获得实例的函数是public的）。使用的时候，一般是继承singleton类，并且使用模板<A>传入子类的名字A，这样使用该类的instance()，就可以创建一个A类的对象，而且这个对象是单例，不可拷贝不可赋值。

通过Instance()，返回一个静态的全局对象。若再次调用，还是返回同一个对象，从而完成单例。static成员函数是用来管理static成员变量的，但这里只是用来生成一个全局静态对象，没有用来管理成员变量。这样每次需要调用全局对象就A::Instance()，就会返回这个对象。然后直接用这个对象来访问里面的数据即可（同时，A中其他成员函数也可以设置为static，这样就可以访问Instance()，从而调用这个对象的数据。例如SapMap，由于是单例，所有成员函数都写成了static，并且用Instance()来获得该对象）。