第11章

运算符重载可以通过函数调用，也可以通过运算符调用。其他运算符重载类似。

友元：即让非成员函数能够访问私有数据。

创建过程：在类声明中，用friend修饰：friend Time operator\*(double m);不需要用类限定符限定方法，也不需要friend修饰，在类外编写函数定义：Time operator\*（double m）{}

类的转换

自动转换发生的情况：（在转换没有二义性的时候才进行）

- 将Stonewt对象初始化为double值时；

- double值赋给Stonewt对象时；

- double值传递时接受Stonewt参数；

- 返回值为Stonewt的函数返回double值时。

如果不希望这种自动类型转换可以在类声明中添加关键字explicit

如果想把Stonewt转换成double类型，就需要转换函数：

创建过程：

1.在类声明中，operator double();

2.在实现中，Stonewt::operator double(){ return pounds； }

其实就是自定义强制转换：像下面这样调用。

Stonewt wolfe(2.3);

double host = double (wolfe);

double host = (double) wolfe;

double host = wolfe;

第12章

不能在类声明中初始化静态成员变量，因为声明只是描述，不分配内存。另外初始化是在方法文件中，不是在类声明文件中，否则会出现多次初始化。

把类对象当作函数参数使用会在函数结束的时候调用析构函数。

类的静态成员，在类声明中声明，在包含类方法的文件中初始化。初始化时使用作用域运算符来指出静态成员所属的类。如果静态成员是整形或枚举型const，则可以在类声明中初始化。

值传递的时候会调用复制构造函数，如果代码里面没有复制构造函数，则会生成一个系统默认的。但是函数结束之后，就会调用析构函数。

如果是使用浅复制，可能会把原来的对象某些东西更改。

构造函数的原型如下：

className(const className& );

C++会提供一些特殊的函数，如果程序本身没有定义。

默认构造函数

默认析构函数

复制构造函数

赋值运算符

地址运算符

带参数的构造函数，如果所有参数都有默认值，则可以作为默认构造函数

但是，只能有一个默认构造函数，否则有二义性。

如果类中包含了在新对象创建时发生变化的静态数据成员，应该提供一个显示复制构造函数来处理计数问题。

如果类中包含了使用new初始化的指针成员，应当定义一个复制构造函数，以复制指向的数据，而不是指针，这被称为深度复制。

复制的另一种形式（成员复制或浅复制）只是复制指针值。浅复制仅浅浅复制指针信息，而不会深入挖掘以复制指针引用的结构。

赋值运算符，C++允许类对象赋值，通过自动为类重载赋值运算符实现的。原型如下：

className& className::operator=(const className&);

初始化对象的时候，并不一定使用赋值运算符，可能使用复制构造函数。

也可能使用复制构造函数，创建了一个临时对象，然后复制到新对象。

用nullptr来表示空指针。

构造函数如果用到new，析构函数要用到delete，如果是new[],则要用delete[]。

析构函数是可以显式调用的，使用定位new的对象可以显式调用析构函数。这样析构了自身，也不会删除了内存块。

构造函数的成员初始化列表，对于const的成员也能初始化。