通常将接口(类定义)放入头文件中，将实现放入源代码中

头文件中：

#ifndef \_TEST\_H

#define \_TEST\_H                 //一般是文件名的大写

头文件结尾写上一行：

#endif

在一个大的软件工程里面，可能会有多个文件同时包含一个头文件，当这些文件编译链接成一个可执行文件时，就会出现大量“重定义”的错误，这是为了防止“重定义”出现。

类的定义示例：

class Stock

{

private:

std::string company;

long shares;

double share\_val;

double total\_val;

void set\_tot(){total\_val = share\*share\_val;}

public:

void buy(long num, double price);

void show();

}

private和public，类对象的程序可以直接访问公有部分，但只能通过公有函数(或友元函数)访问私有成员。不标明，默认私有。

成员函数定义和常规函数非常相似，但有两个特征：1. 定义成员函数时，使用域解析运算符(::)来标识函数所属的类；2. 类方法可以访问private组件。如：void Stock::buy(long num, double price){…}。

定义位于类声明中的函数都将自动成为内联函数。常将短小的函数作为内联函数。使用inline限定符将外部定义的成员函数变成内联函数。

创建的每个新对象都有自己的存储空间；但一个类的所有对象共享一组类方法。

10.2.5修改实现re

构造函数和析构函数

类不能用常规方法初始化，因为数据成员是私有的，只能通过成员函数初始化和访问。因此C++提供了一种成员函数--类构造函数，用于构造新对象时，将值赋予数据成员。构造函数没有返回类型。原型如下：

Stock(const string &co, long n=0, double pr=0.0);

一种可能的定义：

Stock::Stock(const string &co, long n, double pr)

{

share = n;

share\_val = pr;

company = co;

set\_tot();

}

声明变量时，将自动调用构造函数。不能将类成员名用作构造函数的参数名。一种常见的做法是在数据成员名中加“m\_”前缀或“\_”后缀。

使用 1.显示调用：

Stock food = Stock(“world”,250,1.25);

2.隐式调用：

Stock garment(“Furry”, 50,2.5);

**构造函数被用来创造对象，但不能通过对象调用**。

默认构造函数是在未提供显式初始值时，用来创建对象的构造函数。当且仅当没有定义任何构造函数时，编译器才会提供默认构造函数。当类定义了构造函数时，程序员必须提供默认构造函数。两种定义默认构造函数的例子：

1.提供默认值

Stock(const string &co=”error”, int n=0, double pr=0.0);

2.没有参数的构造函数

Stock();

调用默认构造函数：

Stock first; //隐式调用

Stock first = Stock();//显式调用

Stock \*first = new Stock;//隐式调用

Stock first(“mycompany”);//调用(非默认)构造函数

Stock second(); //声明一个函数

析构函数是对象过期时，程序自动调用的一个函数。如果构造函数用new来分配内存，则析构函数将用delete来是否内存；一般情况下，析构函数没有任务，这是让编译器生成一个隐式的析构函数就可以。析构函数在类名前加上”~”，且没有返回值和声明类型，也没有参数，其原型为：

~Stock();

其定义可以为：

Stock::~Stock()

{

}

用构造函数赋值时，会创造一个临时对象，之后又会删除该对象。

const成员函数，不修改调用对象，声明：

void show() const;

定义：

void stock::show() const

this指针指向用来调用成员函数的对象，如定义一个比较两个stock类total\_val的成员函数，则可以这样声明：

const Stock &topval(const Stock &s) const;

上面的函数隐式地访问一个对象(调用的函数)，显式地访问另一个对象(输入参数)，三个const表明返回类型为const引用，显式访问对象为const和隐式访问对象为const。函数定义：

const Stock &topval(const Stock &s) const

{

if (s.total\_val>total\_val)

return s;

else

return \*this;

}

对象数组

Stock mystuff[4];

//初始化

Stock stocks[4]={

Stock(“co1”,12.5,20),

Stock(“co12”,14.5,30),

Stock(“co13”,15.5,20),

Stock(“co14”,12.5,50)};

作用域为类的常量re