高级C++程序分析与设计

马瑞新　田琳琳　**编著**

大连理工大学出版社

图书在版编目(CIP)数据

高级C++程序分析与设计 / 马瑞新，田琳琳编著. —大连：大连理工大学出版社，2007.7
ISBN 978-7-5611-3707-9

Ⅰ. 高… Ⅱ. ①马…②田… Ⅲ. C语言－程序设计－高等学校－教材 Ⅳ.TP312

中国版本图书馆CIP数据核字(2007)第113624号

大连理工大学出版社出版
地址:大连市软件园路80号 邮编:116023
发行:0411-84707464 邮购:0411-84703636 传真:0411-84701466
E-mail:dzcb@dutp.cn URL:http://www.dutp.cn
大连理工印刷有限公司印刷 大连理工大学出版社发行

幅面尺寸：185mm×260mm 印张：18.5 字数：426千字
2007年7月第1版 2007年7月第1次印刷

责任编辑：高智银 责任校对：达理
封面设计：宋蕾

ISBN 978-7-5611-3707-9 定价：29.80元(含1CD)

前　言

C++语言是计算机科学及相关专业的重要基础课程。如果说C语言作为编程入门课程，介绍面向过程程序设计方法的话，C++语言则提供了面向对象程序设计风格，更接近于常用的软件开发平台(如VC++)的软件开发思想。目前有很多C++语言教材，其中相当一部分篇幅介绍C++语言面向过程编程方式，考虑到本丛书已出版《基础C++程序分析与设计》，为了避免重复，本书只涵盖了C++语言面向对象程序设计部分。本书的主要特点就是以例程的方式介绍C++语言的基本内容和特点，很多难以用文字描述的复杂概念通过一个简单的程序就得以清楚地说明。

全书分为8章，从第1章到第7章以专题的方式介绍C++语言的各个组成部分，每章都是先讲述知识要点，然后分析关键例题，最后是综合练习。第1章介绍了类和对象的概念；第2章介绍了继承和派生；第3章介绍了重载；第4章介绍了多态性；第5章介绍了IO流；第6章介绍了模板；第7章介绍了异常处理；第8章是综合应用，比较全面地涵盖了面向对象程序设计的概念和技术。

本书的第1章、第2章、第7章、第8章主要由马瑞新撰写，其余章节由田琳琳撰写。在编写教材过程中得到了丘绍为和耿晓亮两位研究生的大力协助，在此表示深深谢意。

配套光盘中包括本书所有例程的代码，所有程序均在Visual C++ 6.0环境中调试通过，方便读者学习、程序调试使用。为了配合本书教学，作者做了完整的课件，如果有教学之用，可向作者发送E-mail：teacher_mrx@126.com索取。

本书可作为计算机科学及相关专业的学生学习C++语言课程的教材，也适合计算机等级考试和C++爱好者学习之用。

由于水平所限，尽管作者不遗余力，仍有可能存在错误和不足之处，敬请读者批评指正。

编　者

2007年7月

目　　录

第 1 章　类和对象

C++区别于传统编程语言的特征是：C++支持面向对象程序设计。从本章开始，我们将开始接触到使C++成为强大语言的那些特性。首先，我们将介绍一个重要的概念：类机制。读者将学会如何声明、定义以及使用类。然后讲述与类紧密相关的概念：对象。读者将会了解到对象和类在C++中是如何起到重要作用的。

类机制是现代软件产业的基石。由于采用类机制编程与传统编程方式的编程思想在方法上有着根本性的区别，因此，读者在一开始便要时刻注意使用类机制来分析问题和解决问题。

对象和类的概念必须要理解。类的定义相当基础，但是非常重要，特别是关于成员的访问控制以及储存类型的内容，一定要吃透。构造函数是类中最重要的成员函数之一，必须要弄清楚构造函数的作用、执行时机、定义以及使用方法。

1.1　类

1.1.1　从现实中抽象

现实世界中有各种各样的事物，为了研究问题的方便，我们经常针对特定的问题，对具体的复杂事物进行抽象，用较简单的模型来描述问题世界。

比如，我们想看某个电视节目，只需要打开电视机，然后选取节目所在频道，就可以观看了。我们不需要知道电视机的内部构造，不需要知道电视机如何接收信号，同样也不需要知道遥控器操作电视机的细节。一般情况下，我们只需要知道电源开关和遥控器的用法就可以观看节目了。

这里的电源开关、遥控器以及屏幕就是对电视机的简单抽象，我们只需要知道这几部分的用法就可以完成对电视机的简单操作。而至于像电视机内部构造这样的细节，我们不必关心。

同样，在用软件解决现实世界中的问题的过程中，必然也存在着抽象。在传统的结构化程序设计中，使用层层的函数调用结构来实现抽象。还是以看电视为例，打开电视机，控制权就由电源按钮传递给电视机内部器件，内部器件通电后就开始接收图像，并显示在屏幕上；当按下遥控器时，控制权又转向遥控器，由遥控器选择频道，而后控制权转向电视机。也就是说，在这种方式下，我们操作电视机，不得不连带了解电视机的构造以及遥控器的原理。它没有隐藏内部细节，全部构造展现在你眼前，如果你是一个

通晓电视机构造的高手，你可以完成操作。但是，人总是会出错的，一旦出现错误，由于你操作的是内部的元件，电视机如果出现致命错误，损失就大了。

使用面向对象方法，就是对现实世界进行抽象，可以提供更清晰、更自然的描述；可以对对象的内部细节进行保护，免受外界破坏；复用性更好，因为使用类机制，可以将以往的问题解决方案保存起来，待有新的问题出现时，只需要从这些解决方案中“继承”即可。

下面，我们从类机制的三大特性来看这种抽象是如何实现的。

1.1.2　类的三大特性

类机制具有三大特性：即封装性、继承性和多态性。

(1)封装性：将紧密相关的数据和方法组合成类，并对类的成员访问进行了严格的管理，防止外界对类内部的破坏。这就比传统的模块机制所提供的独立性更强。

(2)继承性：经过不同程度的抽象，将分散的各种类联系起来，从而建立起一个相互联系的系统。当向系统中添加新的类时，不用从头开始写程序，只需要从已有的系统中选取相应的基类，继承它，然后添加新的功能即可。

(3)多态性：这是面向对象的一个重要特征。类定义一个接口，可以有多个不同的实现，只有在对象运行时才确定使用哪个实现。该机制给程序的编写带来了极大的灵活性，因为只需要编写一般的代码，就可以适应多种情况。

类机制的这些特性是面向对象程序设计取得成功的关键。虽然在这里，读者可能还不明白这些特性的作用，但相信随着读者编程经验的积累，会慢慢理解这些特性，并能够主动利用这些特性来培养自己的编程思维方法。

1.1.3　类的定义

1．类的定义格式

在C++语言中，使用类声明对象前应先定义其结构，类由保留字class定义。类的定义一般分为说明部分和实现部分。

类是一种用户自定义类型，它把数据和函数封装在一起。其一般定义格式如下：

```
class <类名>
{
private:
    <私有数据成员和成员函数>
protected:
    <保护数据成员和成员函数>
public:
    <公有数据成员和成员函数>
};
```

其中，class是定义类的关键字。<类名>是一个标识符，用于惟一标识一个类。一对大括号内是类的说明部分，说明该类的所有成员。类的成员包含数据成员和成员函数两部分。

下面是一个简单的类。该类由三个公有数据成员函数和三个私有数据成员组成。

```
//Ch1_1.cpp
#include <iostream.h>
class TDate
{
private:
    int   month;
    int   day;
    int   year;
public:
    void   Set(int m, int d, int y )
    {
        month = m; day = d; year = y;
    }
    int   IsLeapYear( )
    {
        return( year%4 == 0 && year%100 != 0 )||( year%400 == 0);
    }
    void   Print( )
    {
        cout << year << "." << month << "." << day << endl;
    }
};
```

2．类的作用域

作用域主要用来控制对变量的访问。对类来讲，一个类的成员函数可以不受限制地访问该类中数据成员，而在该类的作用域外对该类的数据成员和成员函数的访问则要受到一定的限制，从而体现类的封装功能。

1.1.4　数据成员和成员函数

1．数据成员

面向对象程序设计中的数据成员相当于面向过程程序设计中的变量的概念，它抽象了类对象的属性。为了体现面向对象程序设计信息的隐蔽性，数据成员均被定义为私有成员。数据成员可以使用C++语言中的任何数据类型进行定义，但需要注意以下两点：

(1)数据成员不能在定义时对其进行初始化，其初始化是在具体对象创建后由构造函数完成的。例如：int i＝1;是错误的。

(2)数据成员不能进行递归定义，即不能使用自身类的对象作为该类的数据成员。例如：

```
class A
{
private:
    A i;
};
```

是错误的。

2．**成员函数**

面向对象程序设计中的成员函数相当于面向过程程序设计中的函数，它描述了类对象的行为。成员函数向外部给出了该类对象所提供服务的接口，该接口是对封装数据成员进行操作的惟一途径。

成员函数又称为方法。实际上，成员函数和方法是同一种实体两种不同的叫法，成员函数是面向过程程序设计中的术语，而方法是面向对象方法中的术语。

成员函数定义的一般格式为：

```
<类型><类名>::<函数名>(<参数表>)
{
    <函数体>
}
```

其中，作用域运算符“::”用于声明成员函数所从属的类。

可以在函数说明或函数定义中为形参指定一个默认值。调用时如果给出实参，则用实参初始化形参，如果没有给出实参，则采用预先定义的默认形参值。例如：

```
int add_init(int x=6,int y=30)
{
    return(x+y);
}
```

在函数add_init( )的说明中，对两个参数指定默认值。

```
add_init(10,20); //用实参来初始化形参，实现10+20
add_init(10);    //形参x采用实参值10，y采用默认值30，实现10+30
add_init( );     //x和y都采用默认值，分别为6和30，实现6+30
```

如果一个函数有多个默认参数，在形参分布中，默认参数应从右至左逐个定义。当调用函数时，只能向左匹配参数。例如：

```
void func(int a=1,int b,int c=3,int d=5);       //错误定义
void func(int a,int b=3,int c=4,int d=5);       //正确定义
```

对第二个函数的调用方法规定如下：

```
func(10,20,30,40);    //正确，调用时给出所有实参
func( );              //错误，参数a没有默认值
func(20,20);          //正确，a和b的值都是20，c和d取默认值
```

1.1.5　访问权限控制

在C++中，类是封装的单元，类通过存取控制被封装起来。存取说明符制定了类的成员的访问权限。如果不具备相应的访问权限，对该成员的访问就是非法的，编译器就可以直接指出其中的错误。

类成员从访问权限上分有以下三类：公有(public)、私有(private)和保护(protected)，其中默认为private权限。公有(public)的成员可以被程序中的任何代码访问，私有(private)的成员只能被类本身的成员函数及友元类的成员函数访问，其他类的成员函数，包括其派生类的成员函数都不能访问它们，保护(protected)的成员与私有的成员类似，只是除了类本身的成员函数和友元类的成员函数可以访问保护成员外，该类的派生类的成员也可以访问。

类的声明中可以任意变换存取说明符。也就是说，在转到public方式下进行一些公共成员的声明之后，可以再转到private方式下进行私有成员的声明。在变换到另一种说明符之前，该说明符将一直保持有效。举例如下：

```
class   A
{
private:                    //private说明符在首次出现时可以省略
    int m_a;
public:                     //切换到public说明符，以下就是公有成员
    int m_b;                //该成员是公有的
    int m_c;                //该成员仍是公有的
protected:
    int m_d;                //该成员是保护的
};
A   a;      //定义A类的对象a
a.m_a;      //错误，m_a为A的私有成员
a.m_d;      //错误，m_d为A的保护成员
a.m_b;      //正确，m_b为A的公有成员
```

基于前面对类的定义描述，定义一个student类。例如：

```
class student
{
private:
    long number;
    char name[9];
    char sex;
    float score;
public:
```

```
    void shownumber( );
    void showname( );
    void showsex( );
    void showscore( );
};
```

以上定义是student类的说明部分，下面定义了student类的实现部分。

```
void student::shownumber( )
{ cout << number; }
void student::showname( )
{ cout << name; }
void student::showsex( )
{ cout << sex; }
void student::showscore( )
{ cout << score; }
```

成员函数也可以在类体内直接定义，例如：

```
class student
{
private:
    long number;
    char name[9];
    char sex;
    float score;
public:
    void shownumber( )
    { cout << number; }
    void showname( )
    { cout << name; }
    void showsex( )
    { cout << sex; }
    void showscore( )
    { cout << score; }
};
```

使用函数有利于代码的重用，可以提高开发效率，但函数调用会降低程序的执行效率。类的成员函数分为内联函数和外联函数两种。定义在类体内的成员函数称为内联函数，调用该函数时不需要转向执行函数体，而是用函数体的代码进行切换；定义在类体外的成员函数称为外联函数。如果要将外联函数转换为内联函数，只需要在定义函数的函数头前加关键字inline。内联函数不是在调用时发生控制转移，而是在编译时将函数嵌入到每个调用语句处。例如：

```
class student
{
private:
    long number;
    char name[9];
    char sex;
    float score;
public:
    void inline shownumber( );
    void inline showname( );
    void inline showsex( );
    void inline showscore( );
};
void student::shownumber( )
{ cout << number; }
void student::showname( )
{ cout << name; }
void student::showsex( )
{ cout << sex; }
void student::showscore( )
{ cout << score; }
```

使用内联函数时应注意：

(1) 内联函数体内不能有循环语句和switch语句。

(2) 内联函数必须在第一次被调用之前定义。

1.1.6　构造函数

构造函数属于类中的成员函数，它与类具有相同的名字，并且不返回任何数据类型，也不需要返回任何值，函数的参数表和函数体是由用户根据需要编写的，是否为一个类建立重载构造函数，根据类的需要而定。

构造函数的特点：

(1) 它是一种成员函数，其说明在类体内。其函数体可写在类体内，也可写在类体外。

(2) 它的名字和类名相同，定义和说明构造函数时，不必指明函数的类型。

(3) 它用于创建对象时系统自动调用，也可在程序中调用构造函数创建无名对象。

(4) 它可以有一个参数或多个参数，也可以没有参数。

(5) 它可以重载，即可以定义多个参数。

(6) 在类的定义中如果没有给出构造函数，则系统自动创建一个默认的构造函数，默认构造函数在程序中不起任何作用，它没有参数，在创建对象时，将数据成员初始化为0。

(7)它不能被继承，不能说明为虚函数，不能被显示调用，不能对构造函数进行取地址操作。

(8)它是公有成员函数。

例如，下面程序是将构造函数放在类的外部定义的：

```
//Ch1_2.cpp
#include <iostream.h>
class Date
{
public:
    Date( );      //构造函数声明
    void SetDate( int y, int m, int d );
    int IsLeapYear( );
    void PrintDate( );
private:
    int year, month, day;
};
Date:: Date( )
{ cout << "Date object initialized.\n"; }  //无参构造函数
void Date:: SetDate( int y, int m, int d )
{ year = y; month = m; day = d; }
int Date::IsLeapYear( )
{ return( year%4==0 && year%100!=0 ) || ( year%400==0 ); }
void Date::PrintDate( )
{ cout << year << "/" << month << "/" << day << endl; }
void main( )
{
    Date d;
    d.SetDate(2001,10,1);
    d.PrintDate( );
}
```

输出：

```
Date object initialized.
2001/10/1
```

在类的定义中，类的数据成员可能成为另一个类的对象，在另一个类的对象创建所调用的构造函数中，该成员函数会自动调用其构造函数。

例如，下面学校类包含学生类和老师类对象。

```
//Ch1_3.cpp
#include <iostream.h>
```

```
class Student
{
public:
    Student( )
    {
        cout << "constructing student.\n";
        SemesterHours=100;
        gpa=3.5;
    }
protected:
    int SemesterHours;
    float gpa;
};
class Teacher
{
public:
    Teacher( )
    {
        cout << "constructing teacher.\n";
    }
};
class School
{
public:
    School( )
    {
        cout << "constructing school.\n";
        noMeetings=0;
    }
protected:
    Student student;
    Teacher teacher;
    int noMeetings;
};
void main( )
{
    School sc;
    cout << "back into main.\n";
```

```
}
```

输出:

```
constructing student.
constructing teacher.
constructing school.
back into main.
```

本程序从主函数开始运行，当遇到要创建School类对象时，就调用其构造函数School()，该构造函数运行时，首先分配空间，包含一个Student对象、一个Teacher对象和一个int型数据，然后根据在类中声明的对象成员的次序依次调用其构造函数。

1.1.7　析构函数

析构函数属于类中的成员函数，它与类具有相同的名字，不过应在函数名前加上符号“~”，以同构造函数相区别。它不允许带任何参数，并且不允许带有返回类型。析构函数和构造函数调用次序相反，析构函数在对象撤销时调用，而构造函数在对象创建时调用。

析构函数的特点：

(1)它是一种成员函数，其说明在类体内。其函数体可写在类体内，也可写在类体外。

(2)它的名字和类名相同，与构造函数名的区别在于析构函数名前加“~”，表明其功能与构造函数功能相反。

(3)一个类中只能有一个析构函数。

(4)它可以被自动调用，也可以被系统调用。

在下面两种情况下，析构函数将被自动调用。

(1)一个对象当其结束生命周期时，例如一个函数体内定义的对象，当该函数结束时，自动调用析构函数释放该对象。

(2)使用new运算符创建的对象，在使用delete运算符释放该对象时，系统自动调用析构函数。

当用户没有给类定义析构函数时，系统隐含给该类定义一个析构函数，其函数体内为空，所以自动调用系统给定的析构函数时经常不执行任何操作。

由于析构函数在对象撤销时会被自动调用，所以通常利用析构函数删除对象中由指针成员所指向的动态分配的存储空间，当类中不使用动态存储空间时，通常不需要定义析构函数。

调用析构函数有下列几种情况：

(1)当变量超过作用域时，析构函数被隐式调用。

(2)当指向对象的指针超过作用域时，析构函数不能被隐式调用；如果要删除对象时，必须使用delete操作符。

(3)对于全局变量，在主函数main()终止时，自动调用析构函数。

(4)对于局部变量，当函数结束时，自动调用析构函数。