中国高等学校计算机科学与技术专业（应用型）规划教材

丛书主编 陈明

程序设计应用

谢书良 编著

清华大学出版社

中国高等学校计算机科学与技术专业（应用型）规划教材

丛书主编 陈明

程序设计应用

谢书良 编著

清华大学出版社

北京

内 容 简 介

本书是为学习过面向过程程序设计基础的读者编写的后续教材。全书共分 8 章，第 1 章主要介绍类和对象创建的相关概念，第 2 章集中介绍了对象和类的数据共享，第 3 章介绍了继承和派生，第 4 章介绍了多态性和虚函数，第 5 章介绍了模板和异常处理，这 5 章涵盖了C++ 面向对象程序设计的主要内容。第 6 章与第 7 章介绍了可视化程序设计的基础知识，第 8 章是体现本书特色的一章，介绍了单数据表和多数据表的“学生成绩管理系统”的设计过程和完整代码，为最后进行“课程实践”提供了两个可视化程序设计的工程样例。

本书仍按任务导引教学方法进行编写，十分注重可读性和可用性。用任务来带基础知识，既保持了知识的系统性，又使学习目的比较明确，学习效果容易检验，在激发读者学习程序设计应用知识和训练程序设计能力方面有较好的作用。本书还为授课教师提供精心设计的配套电子课件、全部例题源代码、自测练习题答案和部分题目的源代码，可在清华大学出版社网站上下载。

本书可作为高等院校涉及程序设计的相关专业“面向对象程序设计”或“工程实践”课程的教材，也可作为工程技术人员的参考用书和有志于程序设计的社会青年的自学用书。

图书在版编目（CIP）数据

程序设计应用/谢书良编著．—北京：清华大学出版社，2010.6
（中国高等学校计算机科学与技术专业（应用型）规划教材）
ISBN 978-7-302-22260-6

Ⅰ．①程…　Ⅱ．①谢…　Ⅲ．①程序设计—高等学校—教材　Ⅳ．①TP311.1

中国版本图书馆 CIP 数据核字(2010)第 046004 号

责任编辑：谢　琛　顾　冰
责任校对：梁　毅
责任印制：何　芊

出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编**：100084
社　总　机：010-62770175　　**邮　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：北京季蜂印刷有限公司
装 订 者：三河市溧源装订厂
经　　销：全国新华书店
开　　本：185×260　　**印　张**：18　　**字　　数**：421 千字
版　　次：2010 年 6 月第 1 版　　**印　　次**：2010 年 6 月第 1 次印刷
印　　数：1～4000
定　　价：28.00 元

产品编号：035222-01

编委会

Editorial Board

序 言

应用是推动学科技术发展的原动力，计算机科学是实用科学，计算机科学技术广泛而深入地应用推动了计算机学科的飞速发展。应用型创新人才是科技人才的一种类型，应用型创新人才的重要特征是具有强大的系统开发能力和解决实际问题的能力。培养应用型人才的教学理念是教学过程中以培养学生的综合技术应用能力为主线，理论教学以够用为度，所选择的教学方法与手段要有利于培养学生的系统开发能力和解决实际问题的能力。

随着我国经济建设的发展，对计算机软件、计算机网络、信息系统、信息服务和计算机应用技术等专业技术方向的人才的需求日益增加，主要包括软件设计师、软件评测师、网络工程师、信息系统监理师、信息系统管理工程师、数据库系统工程师、多媒体应用设计师、电子商务设计师、嵌入式系统设计师和计算机辅助设计师等。如何构建应用型人才培养的教学体系以及系统框架，是从事计算机教育工作者的责任。为此，中国计算机学会计算机教育专业委员会和清华大学出版社共同组织启动了《中国高等学校计算机科学与技术专业（应用型）学科教程》的项目研究。参加本项目的研究人员全部来自国内高校教学一线具有丰富实践经验的专家和骨干教师。项目组对计算机科学与技术专业应用型学科的培养目标、内容、方法和意义，以及教学大纲和课程体系等进行了较深入、系统的研究，并编写了《中国高等学校计算机科学与技术专业（应用型）学科教程》（简称《学科教程》）。《学科教程》在编写上注意区分应用性人才与其他人才在培养上的不同，注重体现应用型学科的特征。在课程设计中，《学科教程》在依托学科设计的同时，更注意面向行业产业的实际需求。为了更好地体现《学科教程》的思想与内容，我们组织编写了《中国高等学校计算机科学与技术专业（应用型）规划教材》，旨在能为计算机专业应用型教学的课程设置、课程内容以及教学实践起到一个示范作用。本系列教材的主要特点如下：

1. 完全按照《学科教程》的体系组织编写本系列教材，特别是注意在教材设置、教材定位和教材内容的衔接上与《学科教程》保持一致。

2. 每门课程的教材内容都按照《学科教程》中设置的大纲精心编写，尽量体现应用型教材的特点。

3. 由各学校精品课程建设的骨干教师组成作者队伍，以课程研究为基础，将教学的研究成果引入教材中。

4. 在教材建设上，重点突出对计算机应用能力和应用技术的培养，注重教材的实践性。

5. 注重系列教材的立体配套，包括教参、教辅以及配套的教学资源、电子课件等。

高等院校应培养能为社会服务的应用型人才，以满足社会发展的需要。在培养模式、教学大纲、课程体系结构和教材都应适应培养应用型人才的目标。教材体现了培养目标和育人模式，是学科建设的结晶，也是教师水平的标志。本系列教材的作者均是多年从事计算机科学与技术专业教学的教师，在本领域的科学研究与教学中积累了丰富的经验，他们将教学研究和科学研究的成果融入教材中，增强了教材的先进性、实用性和实践性。

目前，我们对于应用型人才培养的模式还处于探索阶段，在教材组织与编写上还会有这样或那样的缺陷，我们将不断完善。同时，我们也希望广大应用型院校的教师给我们提出更好的建议。

《中国高等学校计算机科学与技术专业（应用型）规划教材》主编

陈明

2008 年 7 月

前 言

当前，IT 技术迅猛发展，日新月异。在计算机应用日益广泛的形势下，软件的概念和程序设计的应用知识已逐渐成为人们渴求的新目标。如果说数学是“培养抽象思维的工具”，物理学是“培养逻辑思维的工具”，那么，程序设计则是“培养计算思维的工具”。有人预言，到 2050 年“计算思维”将成为全人类的主要思维方式，仅从此点而言，对于绝大多数理工类的高校学生来说，学一点程序设计基础和应用知识十分必要。

本书涵盖了C++ 面向对象程序设计和VC++ 可视化程序设计基础内容，建议用 4 课时/周讲完。

教材有如下特点：

1. 教材内容的选定考虑了面向对象程序设计的主要内容，但进行了删繁就简处理，可视化程序设计部分只围绕着简易的数据库编程进行，以适合低年级教学的要求。

2. 根据“学以致用”的原则，特别强化了“综合应用”这一章，以此作为可视化编程的入门级锤炼。选择了“学生成绩管理系统”为示范项目，培养兴趣，激发创意，为读者今后继续学习有关内容打好基础。

3. “多思考，勤上机”是学好程序设计课程的关键。本教材实验 1 到实验 4 是对应第 1 章到第 4 章的实验，实验对每次上机的目的、内容等项目均有明确的要求。考虑到使用简便，仍建议用VC++ 6.0 作为上机环境。

4. 为了对教与学提供方便，本教材备有演示文稿提供给教师教学和学生复习选用。

5. 前五章面向对象程序设计之后都设计了一套有多种题型、一定题量的自测练习题，供课堂练习使用。全部题目的参考答案，将以电子文件形式向教师提供，可以通过清华大学出版社网站下载。

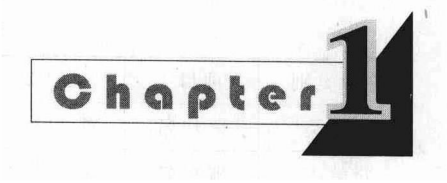
本书是《程序设计基础》的续篇，仍采用“任务导引法”思路进行编写，既有利维护教学内容的体系，也便于检测教学效果。

将面向对象程序设计的内容和可视化程序设计的一部分内容合编一书，重视基础、强化应用是顺应程序设计语言发展历史潮流的一个新的尝试，此教材虽已试用过多年，进行过反复修改，但仍可能存在不足之处，请使用者不吝赐教，使其不断完善。

谢书良

2009 年 10 月

目录



第1章 类

教学要求

- 了解过程化编程的结构体和对象化编程的类的区别；
- 了解并初步掌握类的声明方法；
- 理解并初步掌握对象的创建方法；
- 通过实例理解类的封装性和隐蔽性。

1.1 从结构体到类

前面已经讨论过，如果描述同一个对象的各数据的类型不一致，在面向过程的程序设计（又称过程化编程）中就只能用结构体了。在结构体中，默认成员的访问属性是公有，数据可以随意访问，因此数据的安全性较差，尤其是对于大型软件更是如此。为了解决这个问题，出现了面向对象的程序设计（又称对象化编程）方法。

任务 1-1 为什么要从过程化编程发展到对象化编程？类与结构体究竟有何区别呢？

下面先介绍第一次软件危机的情况。

20 世纪 60 年代末到 20 世纪 70 年代初，出现了大型软件系统，如操作系统、数据库，这给程序设计带来了新的问题。大型系统的研制需要花费大量的资金和人力，可是研制出来的产品却可靠性差、错误多、不易维护和修改。一个大型操作系统每年的工作量相当巨大，而获得的系统又常常会隐藏几百甚至几千个错误。也就是说，在大型程序设计中，投入大量的人力、物力、财力和时间开发出来的软件，其成本、效率、质量等方面都处于失控状态，尤其是软件的维护异常困难。造成的原因很多，而直接原因是出自面向过程程序设计本身的重大缺陷。这就是人们常说的历史上的第一次“软件危机”。它的问题何在呢？我们还是先从一个实际的简单例子讲起吧。

以银行对账户、账目的处理为例，一般银行的账户类型分储蓄、贷款和支票三类，处理这 3 类账户的项目不太一样，如表 1-1 所示。

面向过程程序设计是用结构体来处理的，操作项目和操作手段如表 1-2 所示。

表 1-1 银行账户处理项目

账户类别	项目 1	项目 2	项目 3	项目 4	项目 5	特　征
储蓄类	账户名	账户号	余额	年利率	账目	随时存入和支取
贷款类	账户名	账户号	余额	年利率	账目	一次支取多次存入
支票类	账户名	账户号	余额		账目	一次存入多次支取

表 1-2 操作项目和操作手段

操作项目	操作手段
查询	使用查询函数
存款 取款 转账	使用修改函数

面向过程程序设计的缺陷在哪里呢？存在问题有：

(1) 数据默认公有，易被修改。户名和账号银行掌握，易泄密而造成账户的损失。

(2) 数据和数据处理分离，管理不便，结构繁杂。

(3) 代码无法重用。

数据与数据处理分离，代码的重用性差，程序的维护十分困难。当程序规模较大时，必然会显得力不从心。

针对面向过程结构化程序设计的种种缺陷，人们提出了面向对象的程序设计方法。

面向对象程序设计是软件系统设计与实现的新方法，这种方法是通过增加软件的可扩充性和可重用性来提高程序设计者的生产能力，控制软件的复杂性，降低软件维护的开销。因此，它的应用使软件开发的难度和费用大幅度降低，已为世界软件产业带来了革命性的突破。

在客观世界中，人们处理问题都是面向对象的，对象是构成系统的基本单位。在实际社会生活中，人们都是在不同的对象中活动。

一个具体的杯子是一个对象，它的属性有口径、型号和材质等，对它的操作(或者说它的行为)是盛水等；

一部具体的汽车也是一个对象，它的属性有品牌、型号和排量等，对它的操作(或者说它的行为)是开动和转弯等；

一个具体的人同样也是一个对象，人的属性有性别、身高和体重等，人的行为(也可称为操作)有走路、吃饭等。

类则是一个抽象的概念，用来描述某一类对象所共有的、本质的属性和类的操作、行为。对象则是类的一个具体实现，又称为实例。

以杯子为例，它是描述这类对象共有的、本质的属性和操作、行为的抽象体，而大杯子和小杯子则是杯子类的某个实例，或者说是杯子类的具体对象。

对象的基本特征分为静态特征和动态特征。

例如，学生在一个班级中上课、开会、开展社团活动和文体活动等。这里的对象是班级，它的静态特征是所属系、专业，学生人数，所在教室等；它的动态特征有上课，开会，开展社团活动和文体活动等。

再例如，人们所熟悉的计算机也是一个对象，它的静态特征（或者说属性）有CPU、内存、硬盘、主板、显卡、声卡、键盘、鼠标、光驱等；它的动态特征（或者说行为）有打字、上网、游戏、编程、处理图像、听音乐、欣赏影视节目等。可以说，计算机的组成部件和计算机所做的各种事情共同描述了一部计算机。

类是具有共同特征的对象的抽象，例如：

- 教师：肩负传道、授业、解惑重任的一类人。
- 学生：接受思想教育、道德教育、专业教育、人文教育的一类人。

教师和学生同属于人类，他们是人类的两个属性和行为各不相同的对象（也可称实例）。对类的成员的访问级别可分为公有、私有和保护三级。而且若没有申明，类默认为私有，结构体默认的是公有。

类具有抽象性、隐蔽性和封装性的特征。类的隐蔽性就体现在外界不能直接访问私有成员。

例如：银行将储户的账目、密码、姓名和存款余额定为私有成员，封装在类中，外界无法直接访问，这就保障了储户的利益。

在面向对象（账户）程序设计中具体的做法是：

- 对象的属性：户名账号设置为公有数据成员，而利率和账目余额设置为私有数据成员。
- 对象的行为：有查询、存款、取款或转账等。查询是通过查询余额函数来实现的，而存款、取款或转账等则是通过修改账目函数来实现的。当然，这两个函数应该设置为公有成员函数。

主函数通过对象用一级密码调用查询余额函数，用于查询余额；用二级密码调用修改账目函数，用于修改账目。

这样做有什么必要呢？我们不妨设想一下，如果你有一张银行卡被人拾到了，尽管他不知道取款密码，无法在自动取款机上通过取款密码查到你的卡上（或者说你的账户上）还有多少余额，但在大多数银行柜台上通过储蓄员不难问到，这是因为大多数银行都没有设置查询密码；然而在有些银行例如上海浦东发展银行就问不到，原因很简单，因为这家银行除设置了取款密码之外，同时又设置了查询密码。不知道查询密码，储蓄员也打不开你的账目，也无法查到你的存款余额。由于银行的访问储户账目程序的差异，储户账目的安全性有所不同。后者的安全性之高，显而易见。

封装性使对象的数据得到了保护，所以说封装性是“面向对象”程序设计的重要特征。类是一个封装体，在其中封装了该对象的属性和操作。通过限制对属性和操作的访问权限，可以将属性“隐藏”在类的内部，公有函数作为对外的接口，在对象之外只能通过这一接口对对象进行具体的操作。

C++就是通过建立数据类型——类来支持封装和数据隐藏。封装性增加了对象的独

立性，从而保证了数据的可靠性。一个定义完好的类可以作为独立模块使用。

对象的属性和行为总是紧密联系在一起的，属性用数据（即变量）来描述，行为则是数据的处理，要通过函数来实现。数据和对数据的处理，在面向过程程序设计中两者是分离的，而在面向对象的程序设计中两者是合一的，都封装在类体中。

封装性就是指将数据（变量）和数据处理（函数）都封装在类体内。可以理解为是把变量和相关的函数集中在一个有孔的容器中，只有在孔的边缘处的数据与函数才能与外界相通（这便是指所有的公有的成员），而其余的（指私有和保护的成员）均不受外界的影响，这个容器就是类。

在程序设计与实现中，程序设计方法正在从面向过程走向面向对象，使编程语言与自然语言之间以及程序设计方法与实际解决问题方式之间的距离越来越近。这就意味着软件开发人员可以用更接近自然的思维方式，用更少的精力去完成同样的工作。

概括起来说，面向对象程序设计有如下优点：

- 与人类习惯的思维方式一致；
- 可重用性好；
- 可维护性好。

正因为面向对象程序设计有众多的优点，所以今天程序设计方法逐步由面向程序设计发展为面向对象程序设计。

1.2 类的声明和对象的创建

任务 1-2 声明一个正方形类和一个圆类，分别求它们的面积。

如何进行类的声明呢？类的声明的形式是：

```
class  类名                                    //类头
{                                              //类体起始点
    数据成员;
    成员函数;
};                                             //类体终止点
```

代码中最后一行的“;”是类体结束符。

如前所述类是抽象的，类的成员必须通过类的对象来进行访问、处理。对象必须先创建，创建对象的形式如下：

```
类名  对象名;
```

可以通过一个简单程序来作一说明。

例 1-1 类的声明示例。

```
#include<iostream.h>
class Date                                     //声明了一个名为日期的类
```

```
{
public:                                    //类的成员均具备有公有访问属性
    int year;                              //公有成员变量
    int month;
    int day;
    void print()                           //公有函数成员
    {
        cout<<year<<"/"<<month<<"/"<<day<<endl;
    }
};
void main()
{
    Date t;                                //创建日期类的对象
    cin>>t.year;                           //通过对象访问(输入)成员的值
    cin>>t.month;
    cin>>t.day;
    t.print();                             //通过对象访问(输出)成员的值
}
```

如果通过键盘输入：

```
2009
04
20
```

程序运行结果如下：

```
2009
4
20
2009/4/20
```

注意：

(1) 类声明中的私有、保护和公有等关键字可以在任意位置出现任意次，但把所有的私有成员和公有成员归类放在一起，程序显得更清晰。

(2) 若未注明成员的访问限制类别，系统将默认为私有，必须通过公有成员函数来访问。

(3) 由于类是抽象的，在声明类时系统尚未给类的数据成员分配存储空间，因此不能在类的声明中给数据成员赋初值。

1.3 成员函数

类的成员函数与一般函数的区别如下：

(1) 成员函数是属于一个类的成员，出现在类体中。

(2) 成员函数可以被指定为私有、公有和受保护，一般应指定为 public。

(3) 成员函数可以访问类中任何成员，可以引用在其作用域中有效的数据，函数名则是类的对外接口。

成员函数是类的实现，为了使类体简洁，规范的做法是将成员函数的声明和定义分开。即一般成员函数的表示方式是将成员函数在类体内声明，在类体外定义。此时，必须在函数名前，加上“类域标记”：

类名::

用来说明函数是属于哪个类的。

这种表示方式，不仅可以减小类体的长度，使类体清晰，便于阅读。而且有助于把类的接口和类的实现细节相分离，隐藏了实现其功能的执行细节，以提高软件工程的质量。

例 1-2 成员函数声明和定义分离示例。

源程序如下：

```
#include<iostream.h>
class Date
{
    int year;                                    //默认私有
    int month;
    int day;
public:
    void set();                                  //成员函数的声明
    void print();
};
void Date::set()                                 //成员函数的定义
{
    year=2009;
    month=04;
    day=20;
}
void Date::print()
{
    cout<<"year:"<<year<<endl;
    cout<<"month:"<<month<<endl;
    cout<<"day:"<<day<<endl;
}
void main()
{
    Date d;
    d.set();
    d.print();
    cout<<sizeof(Date)<<endl;
```

```
}
```

程序运行结果如下：

```
year:2009
month:4
day:20
12
```

不难看出，第二个程序类体要显得简洁。最后测出类的大小是12个字节，这正好是日期类三个整型私有成员变量所需要的存储空间。可见，此空间是在创建了类的对象之后，系统为类的所有成员变量所开辟的，成员函数并不存放在这块空间中。

有了以上两个例题作基础，就可以很好地完成任务1-2：先声明一个正方形类型和一个圆类型，然后分别求它们的面积。

源程序如下：

```
#include<iostream.h>
class Square
{
    int length;
public:
    void set(int l);
    int area();
};

class  Circle
{
    int length;
public:
    void set(int l);
    double area();
};
void Square::set(int l)
{
    length=l;
}
void Circle:: set(int l)
{
    length=l;
}
int Square::area()
{
    return length * length;
}
double Circle::area()
```