高等学校计算机专业规划教材

C++面向对象程序设计

杜茂康　吴　建　王　永　编著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http: //www.phei.com.cn

高等学校计算机专业规划教材

C++面向对象程序设计

杜茂康　吴　建　王　永　编著

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

前　言

面向对象编程技术从根本上改变了人们以往设计软件的思维方式，降低了软件开发的复杂度，能够开发出具有高可靠性的、可重用和易维护的软件，提高软件开发的效率，是当今及未来内软件开发的主流技术，它对信息科学、软件工程、人工智能、认知科学及系统工程等学科都将产生深远的影响。了解与掌握面向对象编程技术的基本原理和方法是进一步学习计算机应用和程序设计的基础。

C++是在C语言引入了面向对象机制而形成的一种程序设计语言，程序结构灵活，代码简洁清晰，可移植性强，支持数据抽象、面向对象程序设计和泛型程序设计。C++从其产生到现在已近30年，因其稳定性、高效性、兼容性和扩展性而被广泛应用于不同的领域和系统中，常被用来设计操作系统（如UNIX、Windows、Apple Macintosh）、设备驱动程序或者其他需要在实时约束下直接操作硬件的软件。图形学和用户界面设计是使用C++最深入的领域，银行、贸易、保险业、远程通信以及军事等诸多应用领域也常用C++设计其应用程序的核心代码，以求软件的最佳性能和开发效率。

无论从编程思想、代码效率、程序的移定性和可靠性，还是从语言本身的实用性来讲，C++都是面向对象程序设计语言的典范。学好C++，不仅能够用于实际的程序设计，而且有助于理解面向对象程序设计技术的精髓，再学习诸如Java、C#之类的面向对象程序设计语言也就简单了。

累积至今的各种C++教科书、专著和技术性书籍如满天星斗，不可胜数，其中不乏经典杰作。C++教科书按其内容的侧重点可大致分为两种主要类型：一类基于DOS平台介绍C++面向对象的特征，这类教材对C++程序的基本原理、概念、名词术语等内容的介绍全面而深刻，有利于学习者理解面向对象程序设计的基本原理和技术；另一类则侧重于Windows平台介绍C++面向对象程序设计，这类教材主要介绍Windows程序设计的基本原理，Windows的常用控件编程，以及Visual C++ Windows程序设计方法。本书充分考虑了两类教材的优劣，兼有二者的复合特征。

“**读教科书明其理，看技术书知其用**”，这是本书作者的真切体会。本书作者曾在1997—2003年间用C++结合Power Builder、Visual Basic等语言为一些电信公司开发设计了电信业务账务处理系统、电信业务受理系统、S1240光盘话单分拣系统等应用软件，这些软件的核心代码均用C++编写，稳定而高效，某些系统至今仍有效地运行在一些电信公司中。

近10年的C++教学经验和更长的编程实践让我们真切体会到：把教科书的原理剖析和技术书的案例分析相结合有利于深刻地理解与掌握C++程序设计的基本原理和技术，有利于将学到的技术用于实际的软件开发中。本书就是基于这样的认知体会而编写的。

本书复合了一些C++技术书籍和教材的特点，既比较深刻地介绍了C++面向对象的程序技术和原理，又清晰地介绍了Windows平台下的C++程序实现方法，且通过程序实例将两者较好地结合在一起。书中精心设计了一个贯穿全书大部分章节的规模较大的专业课程类管理程序comFinl，并不断地利用面向对象的C++程序技术扩充该程序的功能，使之成为一个比较完整的综合程序，并最终将它移植到Windows系统中，成为一个Windows应用程序。读者可借此掌握C++应用程序的设计方法，以及将DOS程序中的自定义类移植到Windows程序中的方法和过程。

全书共分为13章。第1～2章介绍C++的基础知识，第1章介绍面向对象程序的主要特征、C++程序的结构、数据输入/输出以及Visual C++ 6.0编程环境；第2章介绍C++对C语言非面向对象方面的扩充，主要包括指针、常量、引用、类型转换、函数重载、内联函数、作用域、名字空间及C++文件操作。

第3～9章介绍C++面向对象程序设计的特征、思想和方法，包括类与对象、继承与派生、虚函数、运算符重载、模板与STL程序设计、异常、文件与I/O流等内容。

第10～11章介绍Visual C++ Windows程序设计的原理和方法。第10章介绍C++ Windows程序设计的基础知识，包括Windows程序设计的常用数据结构、程序运行原理、事件驱动、消息循环、API程序设计等内容；第11章介绍MFC框架应用程序的设计原理和方法。

第12章介绍将第4～9章逐步完善的基于DOS平台的C++课程管理程序comFinal移植到Windows程序中的方法。

第13章为全书各章的习题。

本书内容全面、析理透彻、注重实用。书中精心设计了大量易于理解和富有代表性的示意图和案例程序，清晰而深入浅出地展示了C++面向对象程序设计的原理和各种技巧，颇具启发性，有利于程序设计能力的培养与提高。书中所有案例均在Microsoft Visual C++ 6.0环境中调试通过。

为了便于读者学习和教师教学，本书配有以下辅助资源：

- 例题的全部程序代码。
- 部分习题的程序代码。
- 配套的PPT电子课件。

这些资源可从电子工业出版社的华信教育资源网站（http://www.huaxin.edu.cn 或者 http://www.hxedu.com.cn）上进行下载或发邮件到 unicode@phei.com.cn 索取。

本书由杜茂康、吴建、王永、蔡红军和李昌兵编写，杜茂康编写了第1、2、3、4、5、6、7章，吴建编写了第10、11、12章，王永编写了第7章，蔡红军编写了第8章，李昌兵编写了第9章，全书由杜茂康审校和统稿。

在本书从2003年至今长达3年的编写过程中，得到了不少专家、学者、老师和同事的指导、支持和帮助。刘曜教授对本书的编著方式和章节安排提出了有益的建议，赵济林、刘跃两位教学专家结合教学规律从章节内容的详略难易设置等方面给予本书富有建设性的指导，张仿、曹慧英、罗龙艳、谢青、刘友军、武建军、何波、罗文龙等老师参与了本书编写大纲的讨论与确定，2004级信息管理与信息系统专业两位忠实的学生李明闯和王晓润仔细地阅读了本书初稿中的每一个字符、每一行代码，校正了初稿中的许多错误，并提出了许多有用的建议。在此谨向他们表示诚挚的感谢！

在本书的编写过程中阅读参考了国内外大量的C++书籍，这些书籍已被列在书后的参考文献中，在此谨向这些书籍的作者表示衷心感谢！

特别的感谢敬献给本书的策划者和责任编辑章海涛先生，没有他的策划、指导、校正和辛勤编辑，就没有本书的出版！

面向对象程序设计是一项不断发展变化的程序技术，C++更是博大精深，鉴于作者才疏学浅，水平有限，经验不足，加之时间仓促，书中一定存在不少错误和不当之处，恳请专家、同行和读者批评指正。E-mail：cqyddk@163.com。

作　者

目　录

第1章

面向对象程序设计概述

本章导读

- 计算机程序语言的发展
- 类、对象、抽象、封装、多态
- 面向过程与面向对象程序设计
- C++程序的结构
- 用 cin 和 cout 输入、输出数据
- 输出数据的格式化
- VC++实例编程

随着软件规模和复杂度的不断增加，面向过程的程序设计技术在大型软件设计方面已显示出越来越多的缺陷，而面向对象的程序设计技术则显示出了明显的优势。面向对象的程序设计技术用对象来摸拟客观世界中的事物及其行为，用消息传递来模拟对象之间的相互作用，使程序与客观世界具有很大程度的相似性，降低了软件开发的难度，适合大型的、复杂的软件设计。

1.1 计算机程序设计语言的发展

从计算机产生到现在，程序设计语言的发展过程大致经历了机器语言、汇编语言、高级语言、面向对象程序设计语言等阶段。

1. 机器语言

机器语言是最早的程序设计语言，它由计算机能够识别的二进制指令系统构成。所谓指令，就是指计算机能够识别的命令，它们是一些由 0 和 1 组合成的二进制编码。计算机硬件系统能够识别的所有指令的集合，就是它的指令系统。

机器语言与计算机硬件密切相关，不同硬件系统具有不同的机器指令系统。即使完成相同的任务，在不同硬件系统（具有不同指令系统）的计算机上编写的机器语言程序也不会相同。为了编写计算机程序，程序员需要记住各种操作的机器指令代码；为了读取数据，还要

知道数据在内存中的地址。这种需要记住大量具体编码来编写程序的方法不但难于实现，而且容易出错。

2. 汇编语言

为了解决机器语言编程困难、难以记忆之类的缺点，人们用一些便于记忆的符号代替机器语言中的二进制指令代码，这就是汇编语言。从机器语言到汇编语言，虽然编写程序简单了许多，但它仍然是与机器相关的，不同机器系统的汇编语言并不相同，要在不同硬件系统（其指令系统不同）的计算机上完成相同的任务，需要编写不同的汇编程序。

机器语言与汇编语言统称低级语言，它与人类的自然思维习惯存在着较大区别，编写程序比较困难。此外，低级语言是与机器相关的，即使在不同的机器系统中完成相同的任务，也需要编写不同的程序。

3. 高级语言

随着计算机技术的发展和计算机应用的普及，出现了与人类自然语言的思维习惯很接近的计算机程序设计语言，即高级语言。高级语言屏蔽了与机器硬件相关的细节，提高了语言的抽象层次，采用具有一定含义的命名符号和容易理解的程序语句进行程序设计，不仅大大降低了程序设计的难度，而且也使程序易被人们理解。由于高级语言是与机器无关的，同一程序可以在不同计算机上运行，提高了程序的可移植性和通用性。

随着程序规模和复杂度的不断增加，程序设计方法也在不断地改进。20 世纪 60 年代末产生了影响深远的 SP（Structure Programming，结构化程序设计）程序设计思想。支持结构化程序设计的高级语言称为结构化程序设计语言。这些语言支持结构化的数据，提供了结构化的语句，以及数据抽象、过程抽象等概念，使程序设计更接近于客观事物的结构和逻辑含义，程序语言更接近于人类的自然语言。

结构化程序设计语言是一种面向过程的程序设计语言，它通过过程（某些语言有过程和函数之分，如 Pascal 语言，有些语言只有函数这一概念，如 C 语言）来抽象和模拟客观现实。过程为信息隐藏提供了可能，一个程序员可以为其他程序员编写过程，其他程序员不必了解过程的实现细节，通过过程的接口就可以操作它们。

为了实现个人通信录管理功能，程序员编写了 4 个过程（函数）：InputData，SearchAddr，SearchPhone 和 PrintData，分别用于实现通信录数据的输入、输出和数据查询功能。有许多技术可以实现通信录的数据存取，如数组、链表、队列或堆栈等，本例中采用数组存取数据。

```
struct Person{                  //用于存放个人信息的数据结构
    char name[10];
    char addr[20];
    char phone[11];
}
Person p[100];                  //保存所有个人信息的全局数组
int n=0;                        //用于保存实际人数的全局变量
void InputData(){ ......}       //初始化全局数组P，读入每个人的姓名、地址和电话
void SearchAddr(char *name){……}          //根据姓名查找地址
void SearchPhone(char *name){……}         //根据姓名查找电话号码
void PrintData(){……}    //打印输入每个人的姓名、地址和电话
```

这 4 个过程通过全局数组（即 P）共享数据，并且相互影响。如果将这些过程提供给其他程序员使用，就必须让该程序员知道他不能定义和修改全局数组（即 P），只能通过这 4 个过程存取全局数据 P。

上述个人通信管理程序代表了结构化程序设计语言的编程方法：先定义一些全局性的数据结构，然后编写一些过程对这些数据结构进行操作，其模型如图 1-1 所示。

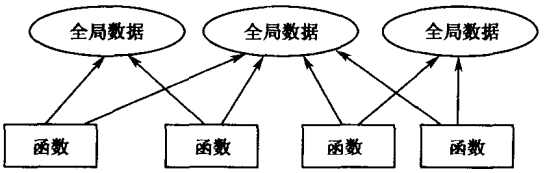

图 1-1　结构化程序设计模型

从图 1-1 的程序模型可以看出，数据和函数之间存在潜在的连接关系。某个全局数据的修改可能会引起大量操作该全局数据的函数的修改。此外，若某个函数意外修改了某个全局数据，很可能引起程序数据的混乱。

比如在个人通信录管理程序中，Person 的变化会引起操作它的所有过程（如 InputData、SearchAddr 等）的修改。此外，谁也没有办法限制其他程序员定义与全局数据同名的变量（如数组 P），也不能限制他不去修改全局数组的值。当程序规模较大时，这个问题尤其突出。

这就是面向过程程序设计的主要问题：操作数据的过程（函数）与它所操作的数据是分离的。在上面的程序中，虽然希望他人只能通过 InputData、SearchAddr、SearchPhone 和 PrintData 这 4 个函数去操作通信录全局数组 P 中的数据，但是结构化程序设计语言没有提供实现这一操作限制的语言机制，所以它不能阻止他人直接修改 P 数组中的数据，当程序规模较大时，这会使程序的设计和维护变得非常困难。

虽然存在缺陷，但结构化程序设计语言仍然广泛地应用在当前的计算机程序设计之中，较常用的有 C、Fortran、BASIC、Pascal 等语言。

4．面向对象程序设计语言

客观世界的任何事物都有自身的属性（特征）和行为，在面向过程的程序设计语言中，事物的属性被抽象成了数据，而行为则被抽象成了函数（或过程），但数据和函数是分离的。描述一个事物特征的数据能够被其他函数修改（如全局数据可被任何函数修改），这与客观事物的本来面目不相符合，因为一个事物的属性只有它自己才能改变。

面向对象程序设计的观点认为用计算机求解的都是现实世界中的问题，它们由一些相互联系，并且处于不断运动变化的事物（即对象）组成。每个事物都可以通过两个方面来刻画：描述事物状态的数据和描述事物行为的操作，应该把它们结合成一个整体，代表一个客观事物，这个整体就是对象。在对象中，描述事物行为的操作被抽象成了函数。

由此可以看出，一个对象由数据和函数两部分构成。数据常被称为数据成员，函数则被称为成员函数。一个对象的数据成员往往只能通过自身的成员函数修改。

比如前面的个人通信管理程序，在面向对象程序设计语言（如 C++）中的简易模型如下。其中的 class 是 C++用于将数据和函数组合成一个整体（即类）的语言机制。

```
class Person{                          //用于存放个人信息的数据结构
 private:
   char name[10];
   char addr[20];
   char phone[11];
 public:
   void InitData(){……}          //初始化全局数组P，读入每个人的姓名、地址和电话
   void SearchAddr(char *name){……};          //根据姓名查找地址
   void SearchPhone(char *name){……};         //根据姓名查找电话号码
 };
```

在这个程序模型中，Person 的数据 name、addr、phone 和操作数据的函数 InitData、SearchAddr、SearchPhone 被 class 捆绑在一起，它们是一个整体。程序中的任何函数只有通过 InitData、SearchAddr 和 SearchPhone 等函数才能修改或查看 Person 的 name、addr 和 phone 数据。除此之外，再无他法，这个性质是由 private 和 public 决定的。

将客观事物的属性和行为抽象成数据和操作数据的函数，并把它们组合成一个不可分割的整体（即对象）的方法能够实现对客观世界的真实模拟，反映出世界的本来面目。从客观世界中抽象出一个个对象，对象之间能够传递消息（一个对象向其他对象发出的服务请求信息），并通过特定的函数进行数据访问，禁止以任何未经允许的方式修改对象的数据，这就是面向对象程序设计的基本模式。这一过程的程序模型如图 1-2 所示。

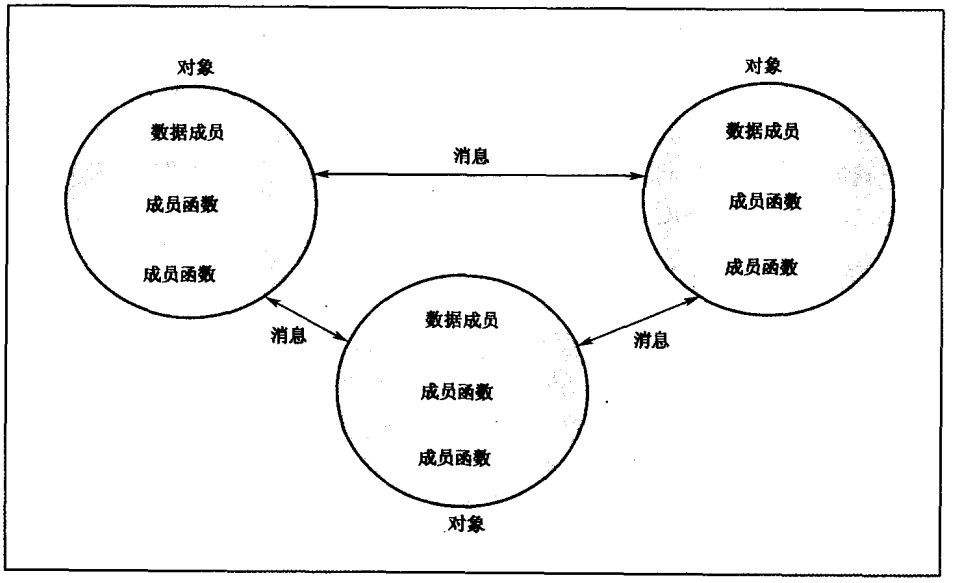

图 1-2 面向对象程序设计的程序模型

图 1-2 所示的程序模型与客观世界的本来面目非常接近，客观世界由一个个实际存在的个体组成，如果某个体要想获得另一个体的某种服务，它就会向对方发出某种请求信息，对方对接收到的请求做出某种响应。在面向对象程序设计中，客观世界的个体被抽象成了对象，一个对象能够向其他对象发送消息，也能够接收其他对象发送来的消息，并通过成员函数（也称方法）响应消息。

面向对象程序设计语言经历了一个较长的演变过程，20 世纪 50 年代的 LISP 语言就引入了信息隐藏和封装机制，20 世纪 70 年代的 Smalltalk 则是第一个真正面向对象的程序设计语言。现在广泛使用的 C++、Eiffel、Object-C、Visual Basic、PowerBuilder、Delphi、C#、Java 等都是面向对象的程序设计语言。

1.2　面向对象程序语言的特征

概括而言，程序的组织方式大致有两种：以功能为中心或以数据为中心。结构化程序以功能为中心组织程序，而面向对象程序设计（Object-Oriented Programming，OOP）则围绕数据进行程序组织，以数据为中心，围绕数据设计程序代码，由用户定义数据及可实施于数据的操作。面向对象设计语言具有抽象、封装、继承和多态等基本特征。

1.2.1　类与对象

在现实生活中，对象是指客观存在的事物，如一个人、一条狗、一棵树、一台计算机等都是对象。同类对象具有相同的属性（特征）和行为，比如所有的人都有姓名、性别、眼、双手、双脚、身高、体重等属性，有走路、讲话、打手势、学习和工作等行为；楼房有位置、楼层、房间数量、造价等属性；狗有皮毛、尾巴、四条腿等属性，有跑、吠、摇尾巴等行为；飞机能够飞行，轮船能够航行；如此等等。

人们借助于对象的属性和行为认识客观世界，将具有相同属性和行为的客观对象归入同一类。利用这种分类方法，人们将客观世界分成了人类、狗类、猪类、树类、草类等。

每类事物都有许多实际存在的个体，这些个体则称为对象（Object）。同类事物的不同个体，尽管有着相同的一组属性和行为，但在属性取值和行为表现上却存在个体差异。比如，张三和李四都具有人类所共有的属性和行为，但他们在走路、讲话、打手势、学习和工作等行为的实施方面有各自的特点，这是行为上的个体差别；他们具有不同的名字（姓名属性不同），张三 1.7 米高，李四 1.5 米高（身高属性不同），这些是属性取值的差异。在现实中，人们借助于对象的属性和行为认识和区分不同对象。

面向对象程序设计用计算机中的软件对象模拟现实中的实际对象，它用类（class）来表示同类对象的共有属性和行为，即用类这一概念来表示客观世界中的同类事物。在面向对象程序中，类是用归类方法从一个个具体对象中抽取出共同特征而形成的概念。比如，张三、李四、王二……都是学生，都有学号、姓名、班级、性别等属性，具有做作业、听课等行为。将所有同学都共有的这些属性和行为抽象出来，就构成了学生类。而具有一个类所指定的属性和行为的一个个体则称为该类的一个对象。图 1-3 表示了一个学生类和其中一个对象的关系。

1.2.2　抽象与封装

抽象（abstract）是指有意忽略问题的某些细节和与当前目标无关的方面，以便把问题的本质表达得更清楚。抽象在现实生活中随处可见，比如要画一幅中国地图，如果不分主次把

所有的山川、河流、城市、交通线路全画上去，最后的结果将是一团糟。只能通过抽象，画出中国各大省份的概况，如主要大城市，主要山脉，重要的交通线路，主要的大江大河等，有意不画出各省中不重要的县及小城镇，不画出各城市中的各条街道、公路等。道理很简单，这样更能反映出中国地理的整体面貌，让人们更加清楚地了解中国地理的分布情况。

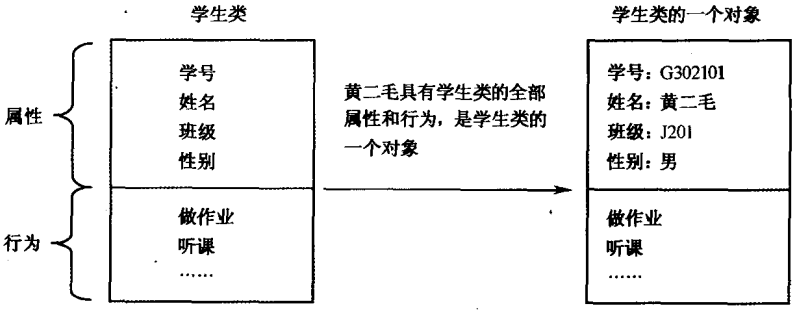

图 1-3 类与对象的关系

通过抽象把事物的主要特征抽取出来，有意地隐藏事物某些方面的细节，使人们把注意力集中在事物的本质特征上面，更能把握问题的本质。比如在设计 VCD 机器时，要考虑的主要问题如下：

- 在 VCD 机器的表面上应该设计哪些控制功能（开关、命令按钮等）？
- 应该提供哪些输入、输出功能和接口，才能使它连接到电视机、计算机、功率放大器等设备？
- 在它的遥控器面板上要设计哪些命令按钮？

此外，VCD 机器还要能播放光盘中的内容，能够向前或向后寻找特定内容（如播放 CD 时能够找到想听的歌曲），允许人们放入或取出光盘，能够在人们需要时停止或启动等。通过粗略设计，抽象出 VCD 机器的模型，如图 1-4 所示。

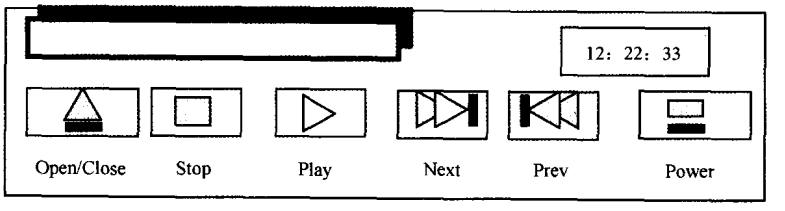

图 1-4 VCD

在进行 VCD 机器的初步设计时，忽略了具体的实现细节。比如当用户按下播放按钮 Play 时，希望播放其中的碟片，并且在连接了该机器的电视机上显示图像和声音。事实上，按下 Play 键时，VCD 要做的事情很多，比如：

- 如果 VCD 没有连接电源，它不会做出响应。
- 如果 VCD 的光驱中没有光盘，它不应该工作。
- 若 VCD 连接了电源，且光驱中有光盘，则机器内的激光束开始活动，光驱开始工作，旋转光盘。
- VCD 内的电子器件读出光盘的编码信息，将其转换成视频信号和声音信号，并将它