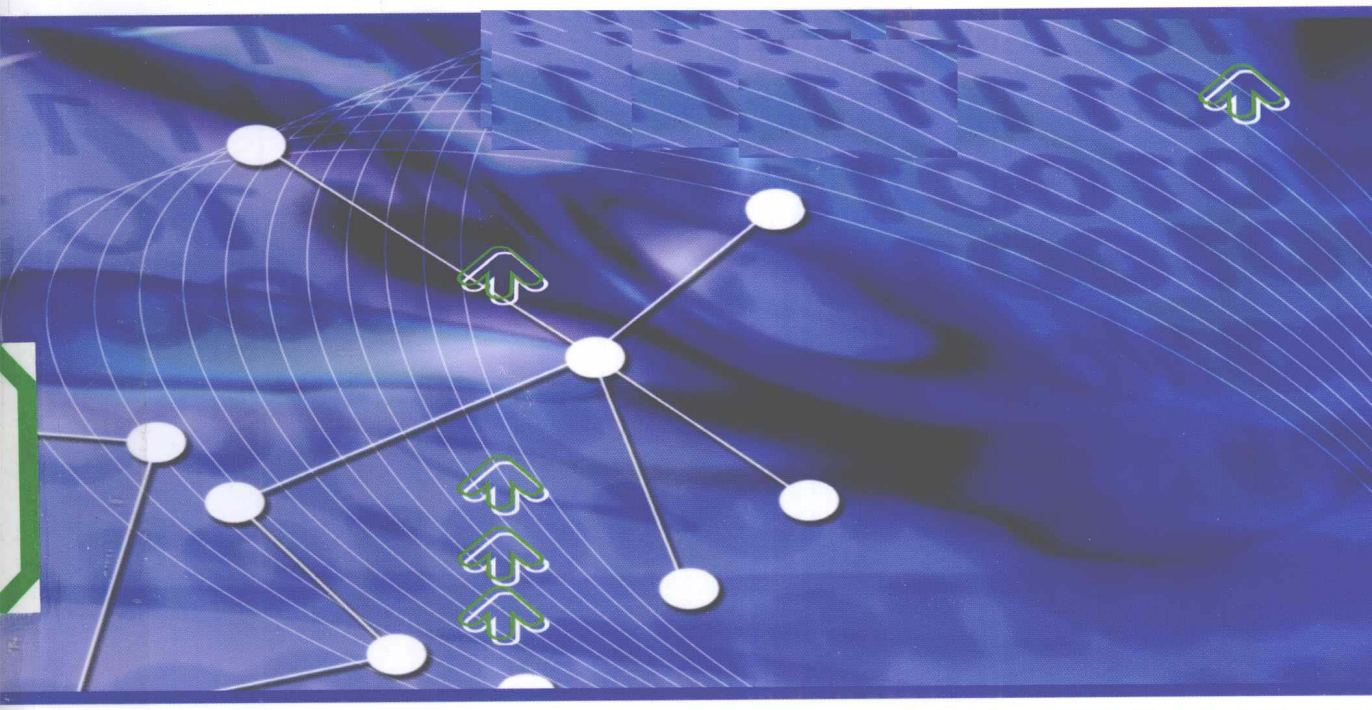
HZ BOOKS
华章教育

高等院校计算机教材系列

C++程序设计教程
基于案例与实验驱动

C++ Program Design
Based on Case and Experimental Drive

邬延辉 王小权 陈叶芳 等编著



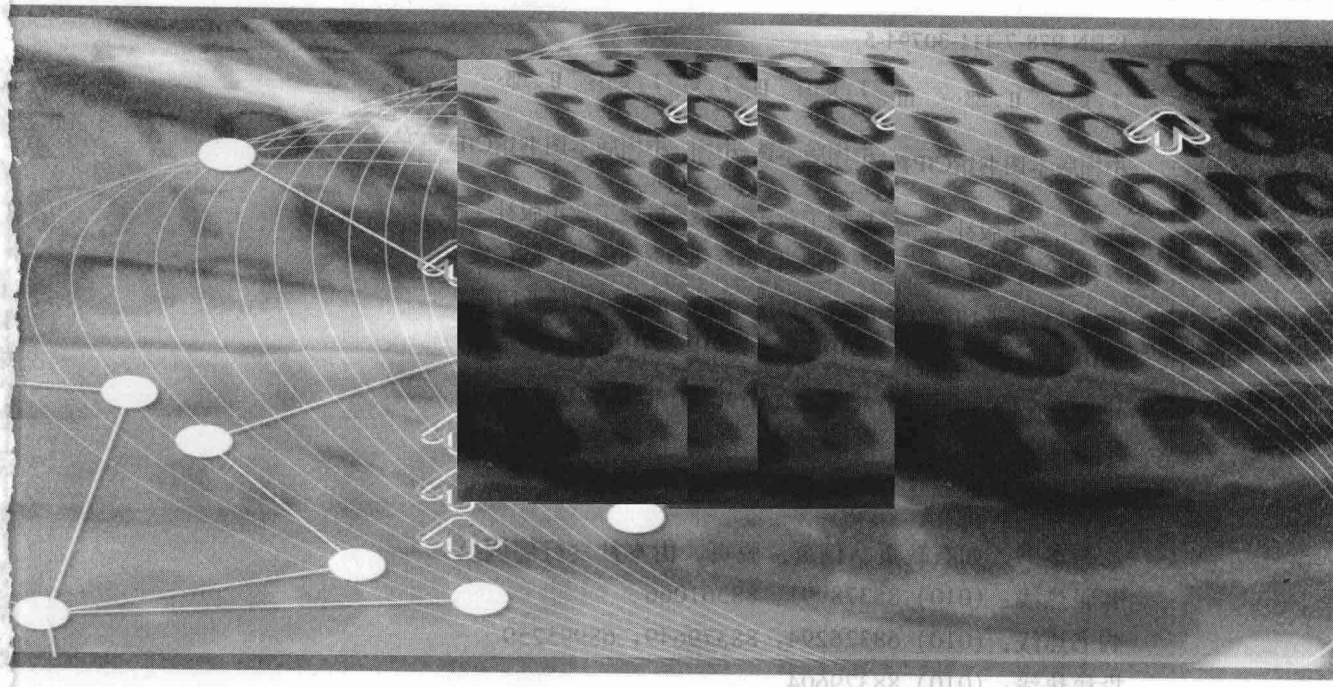
机械工业出版社
China Machine Press

高等院校计算机教材系列

C++程序设计教程
基于案例与实验驱动

C++ Program Design
Based on Case and Experimental Drive

邬延辉 王小权 陈叶芳 刘玉英 赵英刚 等编著



机械工业出版社
China Machine Press

作者结合多年的C++教学经验，根据教学大纲精心设计并且编写了本书。本书依据ANSI C++标准，阐述面向对象的程序设计思想，引出必要的语法知识，详细介绍了C++语言。针对初学者的特点，书中结合大量实例讲解面向对象程序设计的基本概念和方法，并增加了UML类图以及STL程序设计的内容。

全书分为9章，在讲解了C++基础知识之后，引导读者深入学习类与对象、继承与派生、多态性、独立编译与命名空间、模板、输入/输出流、string类以及异常处理。

本书文字流畅、概念清晰、通俗简洁，并配有多种形式的习题，适合作为高等院校相关专业C++程序设计课程的入门教材。

图书在版编目（CIP）数据

C++程序设计教程：基于案例与实验驱动 / 邬延辉等编著. —北京：机械工业出版社，2010.6

（高等院校计算机教材系列）

ISBN 978-7-111-30794-5

Ⅰ.C… Ⅱ.邬… 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（2010）第097301号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）

责任编辑：陈佳媛

北京诚信伟业印刷有限公司印刷

2010年8月第1版第1次印刷

184mm×260mm · 17.25印张

标准书号：ISBN 978-7-111-30794-5

定价：29.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

客服热线：（010）88378991；88361066

购书热线：（010）68326294；88379649；68995259

投稿热线：（010）88379604

读者信箱：hzjsj@hzbook.com

前　言

C++语言是从C语言发展而来的一种高级程序设计语言，它具有全面兼容C语言并支持面向对象的特点，深受广大编程爱好者的喜爱。在过去的十多年里，面向对象设计方法已经成为开发大型软件的主要手段。尽管迅猛发展的Java语言对C++形成强有力的冲击，但C++语言仍然是应用最广泛的面向对象程序设计语言之一，用在高等院校的教学和业界的软件开发中。C++适合于系统级的程序设计，如编译器设计，而Java语言是开发图形界面和跨平台程序的理想语言，二者将会并存和互补。

目前，国内有许多C++教材，它们大都包含了大量的C语言教学内容，对于已经学过C语言课程的读者来讲，这些教材显得篇幅过长，不适合在有限的课堂教学学时情况下使用。本书一开始就进入类和对象的介绍，关注面向对象的设计思想，适合于已经学过C语言课程并将开始学习C++语言的读者，对于在实际工作中需要用面向对象技术来编写C++程序的工程技术人员也很有帮助。本书要求读者具有C语言基础知识，是一本面向C++初学者的入门教材。

本书主要特点

1) 将统一建模语言UML融入到每部分学习内容中，讲解例题前，用UML类图对问题进行分析，使读者掌握用面向对象方法分析复杂问题，而不仅仅是学习面向对象语法。

2) 选取大量贴近现实和现象的例题进行讲解，其中有些例题是作者全新设计出来的，这些例题可以使读者对程序产生浓厚的兴趣，增强学习效果。每章在讲述知识点和例题后，通过一个完整的应用实例来巩固所学知识点。

3) 本书每章都有形式多样的课后习题，让读者自己动手来编写C++程序，使他们从能够看懂C++程序，到能够分析问题，并动手编写程序来解决问题，彻底解决以往“看得懂、不会编”的情况。

4) 本书配备多媒体课件，有需要者可登录华章网站（www.hzbook.com）下载。

5) 本书附录A中提供分章实验指导包含大量习题，部分编程题已给出答案，可作为教师进行测验的备选题目。

致谢

本书第1章～第3章和第5章由邬延辉编写，第4章由刘玉英编写，第6章由王小权编写，第7章由陈叶芳编写，第8章和第9章由赵英刚编写。实验指导由邬延辉编写，宗亮参与了实验指导的部分整理工作。

本书的出版得到了宁波大学计算机科学与技术系国家高等学校特色专业建设、宁波大学教材建设等项目的经费资助，谨致谢意。

特别感谢机械工业出版社王璐编辑，她对编写工作提出了许多建设性的意见，是她的辛勤工作促成了本书的出版。

感谢读者选用本书，欢迎对本书提出批评和修改建议，编者将不胜感激。

通信地址：宁波大学信息科学与工程学院55号信箱　邬延辉

邮政编码：315211

电子邮件地址：wuyanhui@nbu.edu.cn

编者

2010年6月

教学建议

教学内容	学习要点及教学要求	课时安排
第1章　C++基础知识	了解C++语言的产生和发展过程 熟悉面向对象程序设计的基本概念，理解类、对象、封装、继承及多态性的概念 了解面向对象软件的开发步骤 了解C++程序开发过程 熟悉并掌握UML类图的描述	2
第2章　类与对象	掌握类、成员函数、对象的定义 掌握类成员的访问控制属性 理解对象的作用域和生存期 掌握构造函数和析构函数的作用和使用方法 掌握this指针的含义和用法 掌握静态成员和常成员的使用方法 掌握友元函数和友元类的定义和使用 掌握使用new、delete进行动态内存分配的技术	6
第3章　继承与派生	了解继承的目的 掌握派生类的声明 掌握派生类的访问控制 掌握派生类构造函数和析构函数的执行顺序和构造规则 理解二义性问题的本质 掌握虚继承的使用方法 理解并掌握赋值兼容规则	6
第4章　多态性	熟悉多态的分类和实现 理解静态联编和动态联编 掌握虚函数的定义、作用和使用原则 掌握纯虚函数和抽象类 理解和掌握函数重载 理解运算符重载的规则 掌握运算符重载为成员函数和友元函数的使用方法	6
第5章　独立编译与命名空间	熟悉C++程序中的多文件组织结构 理解独立编译的作用 掌握编译预处理指令的使用方法 理解和掌握命名空间的使用方法	3
第6章　模板	理解模板的概念 掌握函数模板的定义和实例化 掌握类模板的定义和实例化 熟悉类模板的默认参数 了解标准模板库STL	3
第7章　输入和输出流类库	熟悉输入/输出流概念 理解流类库结构和派生关系 掌握无格式输入/输出	4

（续）

教学内容	学习要点及教学要求	课时安排
第7章　输入和输出流类库	掌握使用ios类成员函数进行格式化输入/输出 掌握使用操纵符进行格式化输入/输出 掌握文件流的操作方法	4
第8章　string类	了解string类对象的定义 掌握string字符串的定义 掌握string类构造函数的使用方法 了解string类成员函数和操作符	2
第9章　异常处理	了解异常处理机制 熟悉异常处理规则 熟悉异常事件的多路捕获 了解异常处理中的构造与析构	2
	教学总学时建议	34

说明：

①本教材主要为计算机、电子信息类专业的本科“面向对象程序设计C++”课程而编写，要求在学习C语言之后，进行本课程的学习。建议课堂授课为34学时（包含习题课和课堂讨论等必要课堂教学环节），实验另行安排34学时，可以参考使用实验指导，根据各自的教学要求和计划学时对教材内容进行取舍。

②非相关专业的师生在使用本教材时可适当降低教学难度。若授课少于34学时，建议重点学习类与对象、继承与派生、多态性等内容，其他章节内容的学习可以适当简化。

目　录

第1章 C++基础知识

C++语言是在C语言基础上发展演变而来的一种面向对象的编程语言。本章介绍C++语言的基本概念、Java语言的基本特点、面向对象程序的开发步骤，以及UML（统一建模语言）的基础知识。通过典型的C++程序实例及详细的程序解释，让读者逐步进入精彩的C++编程世界。

1.1 面向对象程序设计语言

所有的面向对象程序设计语言，可以分为两大类：一类是混合型的面向对象程序设计语言，典型的如C++，这类语言是在传统的过程化语言中加入了各种面向对象的成分，它所强调的是运行效率；另一类是纯粹的面向对象程序设计语言，在纯粹的面向对象语言中，几乎所有的语言成分都是对象，典型的如Java，这类语言强调的是开发快速原型的能力。下面简要介绍C、C++和Java语言。

1.1.1 从C到C++

C语言是一种与自然语言及数学语言相近的高级程序设计语言，它是由贝尔实验室的Dennis M. Ritchie，于20世纪70年代在B语言的基础上进一步充实完善，发展而来的。C语言应用广泛，甚至连UNIX操作系统都是用C语言编写的。C语言具有许多优点，例如程序执行效率高，且具有高级语言和汇编语言的优点，以及结构化控制语句、编写简洁灵活、数据结构丰富等特点。C语言还可以直接访问物理地址，与汇编语言相比，具有良好的可读性和可移植性。但是随着C语言的推广和应用，其存在的一些不足之处也开始暴露出来，例如，对数据类型进行检查的机制比较弱，缺少支持代码重用，在软件工程规模日益扩大的今天，代码量越来越大，C语言已经难以适应开发大型程序的要求。同时，C语言是一种面向过程（Procedure Oriented）的编程语言，已不能满足面向对象方法开发软件的需要。

C++语言是在C语言的基础上，为克服C语言本身存在的不足之处，且支持面向对象（Object Oriented）程序设计而出现的一种通用程序设计语言，C++语言于1980年由贝尔实验室的Bjarne Stroustrup创建，它根除了C语言中存在的对数据类型检查的机制比较弱、缺少支持代码重用等问题，C++的另一个重要目标是面向对象程序设计，为此，在C++中引入了类的概念。最初的C++语言被称为“带类的C”，1983年正式命名为C++（C Plus Plus），经过后来的不断完善，形成了目前的C++。

为了使C++具有更好的可移植性，美国国家标准局（American National Standards Institute，ANSI）于1990年设立了ANSI X3J16委员会专门负责制定C++标准。不久，国际标准化组织（International Organization for Standardization，ISO）也成立了自己的委员会，负责C++标准的制定。同年，ANSI与ISO将两个委员会合并，共同合作进行C++标准化工作，委员会称为ANSI/ISO。1988年，经过委员会的不懈努力，C++的国际标准（ISO/IEC）获得了ISO、IEC（International Electrotechnical Commission，国际电工技术委员会）和ANSI的批准，这是第一个C++国际标准，称为标准C++或ANSI/ISO C++。2003年，委员会制定并发布了C++标准第2版。本书将以C++标准第2版为基础进行讲解。

1.1.2 Java

Java语言是由Sun公司在20世纪90年代初开发出来的一种纯粹的面向对象的程序设计语言。

Sun公司关于Java的定义，在“Java白皮书”中有明确说明：“Java: A simple, object oriented, distributed, interpreted, robust, secure, architecture-neutral, portable, high performance, multi-threaded, and dynamic language”。按照这个定义进行解释，“Java是一种简单的、面向对象的、分布式的、解释型的、健壮的、安全的、与体系结构无关的、可移植的、高性能的、多线程的和动态执行的语言”。Java语言最大限度地利用了网络，Java的应用程序（Applet）可以在网络上进行传输，所以Java是网络世界的通用语言。另外，Java还提供了丰富的类库，使程序员可以方便地建立自己的系统。因此，Java语言具有强大的图形、图像、动画、音视频、多线程和网络交互能力，能够在设计交互式、多媒体网页和网络应用程序方面大显身手。

Java是从C++编译器入手进行改写的，所以具有类似于C++的风格，保留了C++语言的优点，摒弃了C++中不安全且容易引发程序错误的指针；消除了C++中可能给软件开发、实现和维护带来麻烦的地方，包括冗余和二义性等问题，如操作符重载、多重继承和数据类型自动转换等；简化了内存管理和文件管理。从这个角度来看，Java是C++的简化和改进，C++程序员可以很快地掌握Java编程技术。

1.2 面向对象程序设计的基本概念

在传统的面向过程程序设计中，问题被描述为一系列需要完成的任务，函数是完成这些任务的主要手段。函数是面向过程的，它关注的是如何根据规定的条件完成指定的任务，强调的是算法。尽管每个函数都有自己的局部数据，但是在多函数程序中，有许多重要数据仍然需要放在全局数据区域中，以便所有函数都能够访问到。这种特点容易造成全局数据无意中被其他函数改动、程序的正确性不易保证等问题。在大型程序中，基于结构化程序设计的模块化系统是按照模块功能划分的函数集合。由于用户需求和软硬件技术的不断发展变化，按照功能设计出来的模块可重用性不高，导致代码量增加。

面向对象编程语言与面向过程编程语言的根本区别在于，面向对象编程语言的出发点是能够更加直接地描述客观世界中的事物（对象）以及它们之间的关系。面向对象的编程语言将客观事物看成具有属性和行为的对象，通过抽象找出同一类事物的共同属性（静态特征）和行为特征（动态特征），从而形成类。通过类的继承和多态性，可以实现代码重用，大大缩短软件开发周期，并使软件风格统一。面向对象编程语言使程序能够直接反映问题的本来面目，从而软件开发者能够利用人类认识事物的一般思维方法开发软件。

面向对象的程序设计语言经历了一个很长的发展阶段，出现了多种编程语言。Smalltalk语言、LISP语言、Ada语言、Simula 67语言、Modula-2语言、Prolog语言等，都或多或少地引入了面向对象的概念，Prolog语言用于人工智能领域，而Smalltalk是第一种真正面向对象的程序设计语言。

然而，目前应用最广泛的面向对象的程序设计语言是C++和Java，针对C++语言的集成开发环境包括Visual C++、Borland C++、C++ Bulider等，针对Java语言的集成开发环境包括JDK、JBuilder、Visual J++等。由于C++是在C语言基础上扩充而来的，具有很好的对C语言的兼容性，而C语言早已被广大程序开发者所熟悉，所以C++语言理所当然地成为应用最广的面向对象程序设计语言，同时Java语言与C++语言有极强的相似性，掌握了C++语言，也就掌握了Java语言的面向对象程序设计思想。

那么，什么是面向对象的设计方法？首先，面向对象设计方法将数据和施加于数据之上的操作方法集成在一起，形成一个相互依存、不可分离的整体，即对象。对于同一类对象抽象出其共性特征，从而形成类。类中的大多数数据（数据成员），只能用本类的方法（成员函数）进行处理。类通过外部接口与外界发生联系，对象之间通过消息进行通信。

下面简单介绍几个面向对象方法中的基本概念，仅仅通过以下介绍，就让读者完全理解这些概念，是不现实的。随着后面内容的不断深入，读者会逐步理解这些概念，最终熟练掌握和应用。

1.2.1 类

类（class）是面向对象程序设计语言的基本概念，在C++中也是如此。在现实生活中，人们常常把众多的事物归纳并划分为若干类型，这是认识客观世界常用的思维方法。比如，我们把载人数量为5～7人的、各种品牌的、使用汽油或柴油的、四个轮子的汽车统称为小轿车，也就是说，从众多的具体车辆中抽象出小轿车类。再比如，我们把一所高校所有在校的、男性或女性的、各个班级的、各个专业的本科生、研究生统称为学生，可以从众多的具体学生中抽象出学生类。

对事物进行分类时，依据的原则是抽象，将注意力集中在与当前目标有关的本质特征上，而忽略事物的非本质特征，进而找出这些事物的所有共同点，把具有共同性质的事物划分为一类，得到一个抽象的概念。日常生活中的汽车、房子、人、衣服等概念都是人们在长期的生产和生活实践中抽象出来的概念。

面向对象方法中的“类”，是具有相同属性和服务的一组对象的集合。它为属于该类的全部对象提供了抽象的描述，其内部包括属性和行为两个主要部分。

1.2.2 对象

对象（object）是现实世界中一个实际存在的事物，它可以是看得见、摸得到的物体（如一本书），也可以是无形的（如一项计划）。对象是构成现实世界的一个独立单位，它具有自己的静态特征（可以用某种数据来描述）和动态特征（对象所表现出来的行为或具有的功能）。比如，张三是现实世界中一个具体的人，他具有身高、体重（静态特征），能够思考和做运动（动态特征）。

面向对象方法中的对象，是描述系统中某一客观事物的一个实体，它是构成系统的一个基本单位。对象由一组属性和一组行为构成。属性是用来描述对象静态特征的数据项，而行为是用来描述对象动态特征的操作序列。类与对象的关系，就像模具与产品之间的关系，一个属于某类的对象称为该类的一个实例，比如张三就是人这个类的一个实例，或者是这个类的具体表现。

1.2.3 封装

封装（encapsulation）是面向对象程序设计3个主要特征中一个最基本的概念，另外两个是继承和多态性。封装是将数据和代码捆绑在一起，避免了外界的干扰和不确定性。在C++中，封装是通过类来实现的。类用来描述具有相同属性和方法的对象的集合，定义了该集合中每个对象所共有的属性和方法。封装也是面向对象方法中的一个重要原则，它把对象的属性和行为结合成一个独立的系统单位，并且尽可能地隐蔽对象的内部细节。这里有两个含义：第一个含义是把对象的全部属性和全部行为结合在一起，形成一个不可分割的独立单位；第二个含义是信息隐蔽，也就是尽可能隐蔽对象的内部细节，对外部世界形成一个边界或屏障，只保留有限的公有对外接口，使之与外部世界发生联系。

1.2.4 继承

继承（inheritance）是面向对象技术能够提高软件开发效率的重要原因之一，也是软件规模化的一个重要手段，它的定义是：特殊类的对象拥有其一般类的全部属性和行为，称为特殊类对一般类的继承。

继承具有重要的现实意义，它简化了人们对于现实世界客观事物的认识和描述。比如我们认识了汽车的特征之后，再考虑小轿车时，因为知道小轿车也是汽车，于是理所当然地可以认为小轿车具有汽车的全部一般特征，从而可以把精力用于发现和描述小轿车不同于一般汽车的、其独有的那些特征。

软件的规模化生产是影响软件产业发展的重要因素，其强调软件的复用性，也就是程序不加修改或者进行少许修改，就可以用在不同的地方。继承对于软件的复用具有重要意义，特殊类继承一般类，本身就是软件复用。不仅如此，如果将开发好的类作为构件放到构件库中，在开发新

系统时可以直接使用或者继承使用。

1.2.5 多态性

面向对象的通信机制是消息，面向对象技术是通过向未知对象发送消息来进行程序设计的。当一个对象发出消息时，对于相同的消息，不同的对象具有不同的反应能力。这样，一个消息可以产生不同的响应效果，这种现象称为多态性（polymorphism）。即一个名字，多种语义；相同界面，多种实现。多态机制使具有不同内部结构的对象共享相同的外部接口，通过这种方式减小代码的复杂度。

在操作计算机时，“双击鼠标左键”这个操作可以很形象地说明多态性的概念。如果发送消息“双击鼠标左键”，不同的对象会有不同的反应。比如，“文件夹”对象收到双击消息后，其产生的操作是打开这个文件夹；而“可执行文件”对象收到双击消息后，其产生的操作是执行这个文件；如果是音乐文件，会播放这个音乐；如果是图形文件，会使用相关工具软件打开这个图形。很显然，打开文件夹、播放音乐、打开图形文件需要不同的函数体。但是在这里，它们可以被同一条消息“双击鼠标左键”来引发，这就是多态性。面向对象程序设计通过继承和重载两种机制实现多态性。

多态性是面向对象程序设计的又一个基本特征，它减轻了程序员的记忆负担，使程序的设计和修改更加灵活，程序员只需要记住有限的几个接口就可以完成各种所需要的操作，程序中可以用简单的操作完成同一类体系中不同对象的操作。

1.3 面向对象软件的开发步骤

在学习具体的面向对象程序设计之前，有必要简要了解面向对象的软件开发过程，学习软件开发过程是软件工程学的一部分。在整个软件开发过程中，编写程序代码只是相对较小的一个组成部分。软件开发的决定性因素来自前期概念问题的提出，而非后期的实现问题。只有真正地识别、理解和正确表达应用问题的内在实质，才能做出好的设计方法，在具体的编程代码实现中事半功倍。

早期软件开发所面临的应用问题比较简单，从认清需要解决的问题到编程代码的实现并不太难，但是，随着计算机应用领域的飞速发展，计算机需要处理的应用问题日益复杂，软件系统的规模和复杂度空前扩大，以至于软件的复杂度和其中包含的错误已经到了软件开发人员无法控制的程度，这就是20世纪60年代的“软件危机”。软件危机的出现，有力促进了软件工程学的形成和发展。

面向对象的软件工程是面向对象方法在软件工程领域的全面应用。它包括面向对象分析（Object-Oriented Analysis，OOA）、面向对象设计（Object-Oriented Design，OOD）、面向对象编程（Object-Oriented Programming，OOP）、面向对象测试（Object-Oriented Test，OOT）和面向对象软件维护（Object-Oriented Software Maintenance，OOSM）。

在面向对象的分析阶段，需要从应用问题的陈述着手，建立起一个能够说明系统重要特性的真实情况模型。为了深入理解问题，程序分析员需要与客户协同工作。系统分析阶段应该扼要且精确地抽象出系统需要做什么，而不是关心如何去实现。面向对象的系统分析，直接用应用问题中客观存在的事物建立模型中的对象，无论是单个事物还是事物之间的关系，都保留它们的原貌，不做转换，也不打破原有界限而重新组合，因此能够很好地映射客观事物。

在面向对象的设计阶段，需要将面向对象的方法具体实现在应用问题的系统中。包括两部分工作：一是将面向对象的分析模型搬到面向对象的设计中，作为其中的一部分；二是将具体实现中的人机界面、数据存储、任务管理等因素补充进来。

在面向对象的编程阶段，由于认识问题域和设计系统成分的工作已经在前两个阶段完成，面

向对象的编程工作就是用一种面向对象的编程语言把模型中的每个成分写出来。尽管如此，在学习面向对象的程序设计时，仍然要注重学习基本的思考过程，而不能仅仅学习程序的实现技巧。

面向对象的测试任务是发现软件中的错误，任何一个软件产品在交付使用之前都要经过严格的测试过程。在面向对象的软件测试中，以类为基本单位进行测试，可以更准确地发现程序错误，提高测试效率。

无论软件经过何种严格测试，其中还是会存在错误。因此，在软件的使用过程中，需要不断地进行维护。

本书主要介绍编程方法，很多例题都比较简单，题目本身已经对问题做了详细准确的描述。建议读者在熟练掌握了C++语言编程技术之后，另外专门学习软件工程中的面向对象分析方法。

1.4 C++程序开发过程

C++程序的开发经过编辑、预处理、编译、链接、运行和调试共5个阶段。

首先是编辑（edit）阶段，编辑阶段的任务是编辑源程序，源程序是使用C++语言语句书写的程序，C++源程序文件通常带有.h和.cpp扩展名，其中.cpp是标准的C++源程序扩展名，而.h文件称为头文件。在不同的操作系统和编译环境下，使用源程序的编辑器也不同。在Linux中，常用的编辑器有vi。在Windows操作系统中，许多集成开发环境中包含了编辑器，典型的如Microsoft的Visual C++、Borland的Borland C++等。当然，程序员也可以使用其他编辑器，如记事本（notepad.exe）程序。

在编写完成源程序后，要对其进行编译（compile）。编译器负责将源程序翻译为目标代码（机器语言代码），编译过程分为词法分析、语法分析、代码生成3个阶段。

词法分析（lexical analysis）过程分析源程序语句，从语句中识别出有意义的单词，称为词法记号，词法记号是程序所使用的基本符号，是最小的程序单元。词法记号有标识符、运算符、分隔符、C++语言的关键字、各种文字等。词法分析是检查程序是否正确使用了语言的成分，如果检查到错误，将错误提示给程序员。

语法分析（syntax analysis）是对程序的形式或规则进行检查，也称为文法分析。语法分析根据高级语言程序的语法规则来识别程序的逻辑结构，如各种表达式、控制结构等。语法分析检查程序是否正确使用了语言的结构，如果检查到错误，将错误提示给程序员。根据经验，C++初学者编写程序，编译器检查出来的大部分错误都是语法错误。

代码生成（code generation）是将词法分析、语法分析过程的结果生成目标代码或目标程序，目标代码可以是机器指令代码（二进制代码），也可以是汇编语言代码或其他中间语言。目标代码文件的扩展名为.obj。

在编译器开始翻译之前，预处理器（preprocessor）会自动执行源程序中的预处理语句（命令），如include语句、define语句等。这些预处理语句是按规定在编译之前执行的语句，执行包括将其他源程序文件包含到要编译的文件中，以及进行各种文字替换的操作。

虽然目标代码是由可执行的机器指令代码组成的，但是并不能被计算机直接执行。因为C++程序中通常包含了对其他模块定义的函数和数据的引用，如标准库、自定义库或模块。C++编译器生成目标代码时，这些地方通常被忽略，链接器（linker）的功能就是将目标代码同缺失的函数链接起来，生成可执行代码，形成可执行文件，在Windows操作系统下，可执行文件具有.exe的扩展名。

在当前一些C++集成开发环境产品中（如Visual C++、Borland C++），都将程序的编辑、编译、链接集成在一个环境中，在这个环境中，编译与链接以及运行可以一起进行，但是编译、链接、运行是不同的阶段，当链接出错时，C++系统会提示链接错误。运行时，可执行文件（.exe文件）被操作系统装入内存，CPU从内存中取出执行。

通常在程序开发的各个阶段都可能出现错误，编译阶段出现编译错误，链接阶段出现链接错误，运行阶段出现运行错误。出现运行错误时，程序员需要使用调试工具debug对程序进行调试，发现程序中的逻辑错误，修改源程序，改正错误。目前的调试工具都是针对源代码进行调试。

C++程序开发过程如图1-1所示。

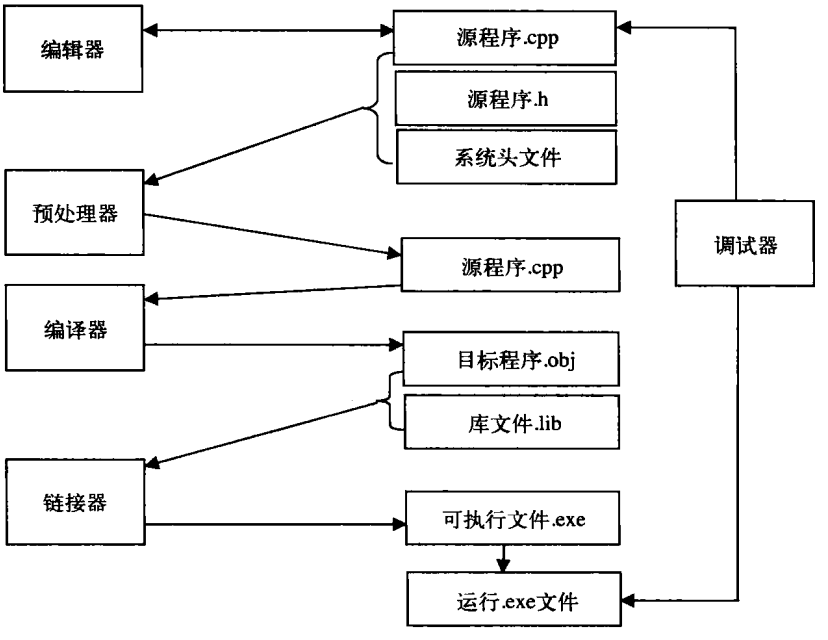


图1-1 C++程序开发过程

1.5 UML类图

通过上面的学习，我们初步了解了类和对象的基本概念，下面介绍一种称为UML类图的图形表示出这些概念，这种图形可以在面向对象设计中把对应用问题的描述直观地表示出来，方便进行交流。

UML是Unified Modeling Language的缩写，称为统一建模语言，它是由OMG（Object Management Group，对象管理组织）在1997年确认并开始推行、最具有代表性、被广泛采用的标准化语言。UML语言是一种复杂、庞大的系统建模语言，其目标是解决整个面向对象软件开发过程中的可视化建模问题，是软件工程中的内容，对于详细完整的介绍已经超出本书的范围，为了使读者有的放矢地了解UML语言特点，本书仅介绍简单的UML图形标识，用来描述C++语言中类、对象等核心概念，为今后的进一步学习打下良好基础。如果读者希望深入了解UML建模语言，请参考相关书籍和资料。

UML语言是一种典型的面向对象建模语言，但它不是一种编程语言，在UML语言中，用符号描述概念，概念之间的关系通过连线来描述。20世纪80年代，一大批面向对象编程语言的出现，标志着面向对象方法走向成熟。到20世纪90年代，在面向对象研究方法领域出现了许多面向对象的分析和设计方法，并且引入了各种独立于编程语言本身的标识符。在众多的建模语言中，用户很难明确区分不同语言之间的特征，因此很难找到一种比较适合自身应用特点的语言，多种建模语言各有千秋，尽管大多雷同，但是仍存在表述上的差异，极大地阻碍了用户之间的交流。因此，客观上有必要寻求一种统一的建模语言。1996年，Booch、Rumbaugh、Jacobson三人经过共同努力，发布了UML 0.9版本，1997年11月17日，OMG采纳UML 1.0作为基于面向对象技术的统一建

模语言，随即风靡信息系统世界，几乎一夜之间人们就在全世界使用UML。2003年6月，OMG通过UML 2.0版本。

显然在某些领域，文字描述不能代替图形，音乐就是一个这样的领域，系统分析和设计是另外一个这样的领域。对于系统分析和设计来说，今天可用的最佳建模语言是UML。

统一建模语言UML的重要内容是各种类型的图形，分别描述软件模型的静态结构、动态行为和模块的组织管理。本书使用UML中的图形来描述软件中类和对象以及它们之间的静态关系，使用最基本的类图（class diagram）。

在开始UML类图讲解之前，先通过King Class和Shoe Class的例子进行说明。

【例1-1】 King Class的描述。

国王乔治三世在1760～1820年间统治英国，国王路易十六世在1774～1792年间统治法国，图1-2和图1-3所示为对国王乔治三世和国王路易十六世的描述。

name:	King George III
country:	Great Britain
startOfReign:	1760
endOfReign:	1820

图1-2 英国国王乔治三世

name:	King Louis XVI
country:	France
startOfReign:	1774
endOfReign:	1792

图1-3 法国国王路易十六世

比较图1-2和图1-3，很显然，所有的国王都有一些相同的方面。国王都有名字，他们都统治一个特定的国家，他们的统治开始于某个特定的年份，并且结束于某个特定的年份。

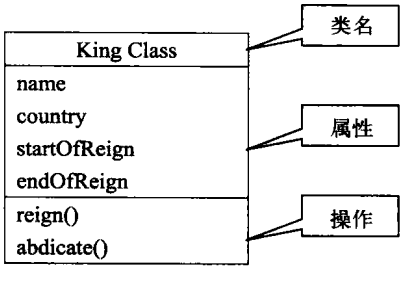


图1-4 King Class的UML描述

让我们把全部国王的集合称为King Class。图1-4描述了King Class。图1-4中间方框的4个项：name、country、startOfReign、endOfReign被称为King Class的属性。底部方框中列出了由国王在其上执行的操作。例如，国王reign（统治）；如果某位国王想停止统治，他将不得不abdicate（退位）（在面向对象编程语言中，通过在操作后面紧跟一对小括号()来区分操作和属性，如图1-4所示）。

类中所有的对象都具有相同的属性集和相同的操作集，它们只是在各自的属性值上有区别。如图1-2所示，国王乔治三世的属性是{King George III，Great Britain，1760，1820}，如图1-3所示，国王路易十六世的属性是{King Louis XVI，France，1774，1792}。

我们把国王乔治三世称为King Class的一个实例，对于国王路易十六世也是如此，也就是说，类是一个相关对象集，相应地，对象是类的一个实例。

接下来，我们考虑另外一个例子：鞋子。

【例1-2】 Shoe Class的描述。

鞋子具有一种color（颜色）（如黑色、棕色），一种size（尺码）（如36、38、40、42）和width（宽度）（如AAA、C、E），一种style（款式）（如细鞋跟、高跟鞋、平底鞋），还有制造它的material（材料）（如皮革、塑料）。Shoe Class是所有鞋子的集合的名称，图1-5所示为Shoe Class。其中中间的方框所示为鞋子的属性：color、size、width、style和material。底部的方框显示了可以在鞋子上执行的操作。也就是说，我们可以PutOn（穿上）一只鞋、TakeOff（脱

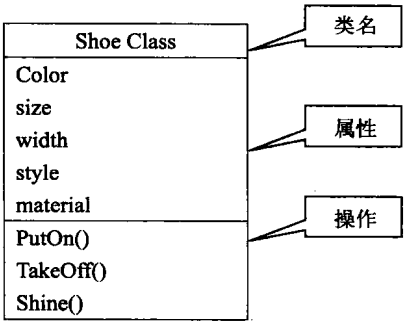


图1-5 Shoe Class的UML描述

掉）那只鞋以及Shine（擦亮）那只鞋。

图1-5显示了用UML表示的Shoe类，其中有3个方框，顶部的方框包含类名，中间的方框包含那个类某个实例的属性名称，最下面的方框包含可以由那个类的某个实例在其上执行的操作的名称。在C++类的概念中，类的属性被称为数据成员，而在类上执行的操作被称为成员函数。

下面介绍用UML完整地表示一个类属性（数据成员）和操作（成员函数）的方法。

根据UML图的详细程度，每个数据成员可以包含其访问控制属性、名称、类型、默认值和约束特性，最简单的情况是只表示出类的名称，其他部分都是可选的。在UML中，数据成员的语法为：

```
[访问控制属性] 名称 [重数] [：类型] [=默认值] [{约束特征}]
```

在数据成员的说明中，至少必须指定数据成员的名称。

- 访问控制属性：分为public（公有类型）、private（私有类型）和protected（保护类型）3种，分别对应于UML图中的“+”、“-”和“#”。
- 名称：标识数据成员的字符串。
- 重数：可以在名称后面的方括号内添加属性的重数。
- 类型：表示该数据成员的种类。可以是基本数据类型，如int（整型）、bool（布尔型）等，也可以是用户自定义的类型，还可以是某个类。
- 默认值：是赋予该数据成员的初始值。
- 约束特征：是用户对该数据成员性质一个约束的说明。如“{只读}”说明它具有只读属性。

在Shoe Class中，数据成员size可以描述为：

```
-size : int
```

访问控制属性“-”表示它是私有数据成员，其名称为“size”，类型为“int”，没有默认值和约束特征。

再比如，表示某个类的一个public类型数据成员，名为“size”，类型为Area，其默认值为(100, 100)。

```
+size: Area = (100, 100)
```

每个成员函数可以包含其访问控制属性、名称、参数表、返回类型和约束特征，最简单的情况是只表示出它的名称，其余部分都是可选的，根据图的详细程度选择使用。UML规定成员函数表示的语法为：

```
[访问控制属性] 名称 [(参数表)] [:返回类型] [{约束特征}]
```

在成员函数的说明中，至少必须指定成员函数的名称。

- 访问控制属性：分为public（公有类型）、private（私有类型）和protected（保护类型）3种，分别对应于UML图中的“+”、“-”和“#”。
- 名称：标识成员函数的字符串。
- 参数表：含有由逗号分隔的参数，其表示方法为“名称：类型=默认值”格式给出函数的形参列表，其格式与源文件（.cpp文件）中不同。
- 返回类型：表示该成员函数返回值的类型。它可以是基本数据类型，如int、float、bool等，也可以是用户自定义的类型，还可以是某个类，或上述类型的指针。
- 约束特征：是用户对该成员函数性质一个约束的说明。

在Shoe Class中，成员函数（操作）PutOn可以描述为：

```
+PutOn() : void
```

访问控制属性“+”表示它是公有成员函数，其名称为“PutOn”，返回类型为“void”，没有约束特征。

下面将King Class和Shoe Class的UML类图重画，如图1-6和图1-7所示。

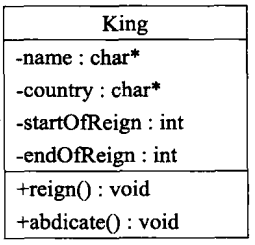


图1-6 King类的完整UML类图表示

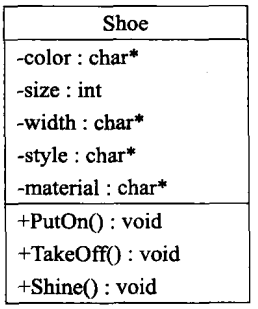


图1-7 Shoe类的完整UML类图表示

使用UML表示类有一种更为简洁的方法，它是隐藏类的数据成员（属性）和成员函数（操作或行为）的表示方法，这种表示方法简单但是信息量很少。图1-8和图1-9所示为King类和Shoe类简洁的UML图。

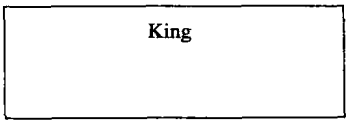


图1-8 King类的简洁UML表示

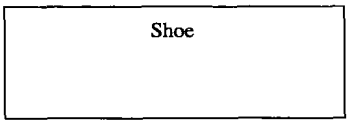


图1-9 Shoe类的简洁UML表示

不同表示方法的使用场合不同，主要取决于绘制该图形的目标，如果我们要详细描述类的成员及其访问控制属性，应当使用完整的UML类图表示；如果我们的着眼点在于类之间的关系，并不关心类内部的东西（比如在程序设计初期划分类的时候），则应该使用简洁的UML表示。

1.6 一个简单的C++程序

一个C++程序是由一个或多个源文件组成的。每一个C++源程序通常都是以.cpp为扩展名，由编译预处理指令、数据或数据结构定义，以及若干函数组成的。下面通过程序代码分析C++程序的组成和结构。

【例1-3】 一个C++示范程序。

```
//1-3.cpp
// A simple C++ program.
/* This is another comment style. */
#include<iostream>
using namespace std;
int main()
{
     int number_of_pods, peas_per_pod, total_peas;

     cout << "Press return after entering a number.\n";
     cout << "Enter the number of pods:\n";
     cin >> number_of_pods;
     cout << "Enter the number of peas in a pod:\n";
     cin >> peas_per_pod;

     total_peas = number_of_pods * peas_per_pod;

     cout << "If you have ";
     cout << number_of_pods;
```