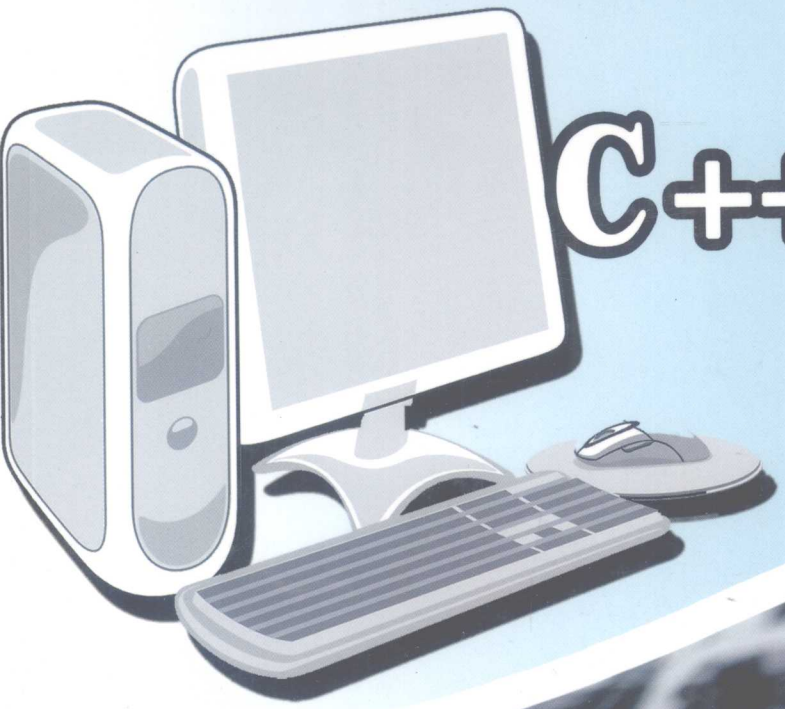
北京科海
25年 · IT技术出版专家

21世纪大学计算机系列教材



C++程序设计

邓 飞 李 瑶 主 编
李 倩 辛瑞超 胡恩福 马 芳 副主编

中国人民大学出版社
北京科海电子出版社
www.khp.com.cn

21世纪大学计算机系列教材

C++程序设计

邓 飞 李 瑶 主 编
李 倩 辛瑞超 副主编
胡恩福 马 芳

中国人民大学出版社
·北京·
北京科海电子出版社
www.khp.com.cn

图书在版编目(CIP)数据

C++程序设计/邓飞，李瑶主编.
北京：中国人民大学出版社，2009
（21 世纪大学计算机系列教材）
ISBN 978-7-300-10401-0

Ⅰ. C…
Ⅱ. ①邓… ②李…
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（2009）第 031080 号

21 世纪大学计算机系列教材
C++程序设计
邓飞　李瑶　主编

出版发行　中国人民大学出版社　北京科海电子出版社
社　　址　北京中关村大街 31 号　　邮政编码　100080
　　北京市海淀区上地七街国际创业园 2 号楼 14 层　　邮政编码　100085
电　　话　（010）82896594　62630320
网　　址　http: //www.crup.com.cn
　　http: //www.khp.com.cn（科海图书服务网站）
经　　销　新华书店
印　　刷　北京市艺辉印刷有限公司
规　　格　185 mm×260 mm　16 开本　　版　　次　2009 年 4 月第 1 版
印　　张　21.5　　印　　次　2009 年 4 月第 1 次印刷
字　　数　523 000　　定　　价　34.00 元

内容提要

本书将C++作为大学生学习程序设计的入门语言，不仅详细介绍了语言本身，而且介绍了常用的数据结构和算法。为了适应读者对C++语言的学习需要，本书系统讲解了以下内容：C++的基本数据类型与表达式，C++控制结构，函数，数组与指针，引用、结构体和共用体，类与对象，C++程序的结构，继承与派生及多态性，模板，流类库与输入输出，异常处理和命名空间，并对面向对象程序设计开发和C++的集成开发环境Visual C++ 6.0进行了介绍。

本书由浅入深，循序渐进，重点突出，对C++从基础的数据类型到高级应用都做了详细讲解。本书既可作为高等院校相关课程的教材，也可作为高级语言学习者和程序员的参考书。

前　言

本书全面介绍了C++语言。作为一本入门书，它以教程的形式对C++语言进行了清晰地讲解，并辅以丰富的示例。与大多数入门教程不同，本书对C++语言本身进行了详尽地描述，并着重介绍了目前通行的、行之有效的程序设计技巧。

本书共分为14章。

第1章　绪论：从发展的角度概要介绍了面向对象程序设计语言的产生和特点，结构化程序设计和面向对象程序设计的概念和区别，并简单介绍了什么是面向对象的软件工程，最后介绍了程序的开发过程。

第2章　Visual C++开发环境简介：主要对C++的集成开发环境Visual C++ 6.0进行了介绍。

第3章　基本数据类型与表达式：讲述C++程序设计的基础知识。首先简要介绍了构成C++语句的基本部分——字符集、关键字、标识符、操作符等，然后在此基础上介绍了基本数据类型，最后介绍了C++的运算符与表达式。

第4章　C++控制结构：讨论C++的控制语句。程序是一些按次序执行的语句。执行语句是为了要完成某个操作、修改某个数据。通过本章可以掌握各种过程化控制语句的使用。

第5章　函数：讲述了C++语言的函数。在面向对象的程序设计中，函数是模块划分的基本单位，是对处理问题过程的基本抽象单元，是对功能的抽象。本章主要从应用的角度讲述各种函数的定义和使用方法，特别是系统函数的使用方法。

第6章　数组与指针：讨论数组、指针与字符串。数组和指针是C++语言中最常用的复合（构造）类型数据，是数据和对象组织表示的最主要手段，也是组织运算的有力工具。本章首先介绍数组、指针的基本概念及动态存储分配问题，接着围绕数据和对象组织这一问题，着重讲解如何通过使用数组和指针解决数据及对象之间的联系和协调，最后还介绍了字符串这类特殊的数组和指针的处理方法。

第7章　引用、结构体与共用体：首先介绍引用的特征，它允许程序来确定把参数传递给函数的方法，然后介绍了结构体的内容，它能够使各个数据类型组合在一起，最后对共用体进行了简单的介绍。

第8章　类与对象：首先介绍C++中类和对象的定义和使用方法，然后对构造函数和析构函数进行介绍，最后讲述了对象数组与对象指针的内容，以及动态建立和释放对象的方法。

第9章　C++程序的结构：首先讲述了标识符的作用域和可见性，对象的生存期，以及使用全局变量和局部变量、类的静态成员和友元来实现数据共用，接着讨论了共用数据的保护，以及如何使用多文档结构来组织和编写程序，解决较为复杂的问题。

第10章　继承与派生：讲述类的继承特性。围绕派生过程，着重讨论不同继承方式下的派生类成员的访问控制问题、添加构造函数和析构函数，接着讨论在较为复杂的继承关系中，继承与组合的问题，最后还讨论了多重继承问题。

第11章　多态性：讲述类的另一个重要特性——多态性。多态是指同样的消息被不同

类型的对象接收时导致完全不同的行为，是对类的特定成员函数的再抽象。C++支持的多态有多种类型，重载（包括函数重载和运算符重载）和虚函数是其中主要的方式，也是我们学习的重点。

第 12 章　模板：模板是 C++语言相对较新的一个重要特性。本章介绍了模板的概念、定义和使用模板的方法，通过这些介绍，使读者能正确使用 C++中日渐庞大的标准模板类库。

第 13 章　流类库与输入输出：首先讲述流的概念，然后介绍流类库的结构和使用。

第 14 章　异常处理和命名空间：讲述异常处理和命名空间问题。

本书从最基础的数据类型讲起，一直到 C++的高级应用（如模板、异常处理），层次清晰，由浅入深，循序渐进，重点突出。同时本书配有《C++程序设计题解与实验指导》，注重理论与操作相结合。本书适用的读者范围很广，不管你是刚接触编程语言的新手，还是已经具有 C++语言基础的开发人员，甚至是已经熟练掌握了 C、Java 的老程序员，使用本书都会有较大的收获，迅速提升编程技能。

由于时间仓促，加之编者水平有限，不足之处在所难免，恳请广大读者不吝指正。

编者

2009 年 3 月

目　录

第1章

绪　论

计算机程序是能够被计算机识别并被计算机执行的命令集，目的是使计算机完成指定的任务。这些命令以一种特定的方式编写，这种方式必须符合所选择程序设计语言的规则。程序设计语言多种多样，但是广泛使用的只有少数几种。本章首先从发展的角度概要介绍面向对象程序设计语言的产生和特点，结构化程序设计和面向对象程序设计的概念和区别，以及什么是面向对象的软件工程，然后介绍程序的开发过程。

1.1　计算机程序设计语言的发展

语言是一套具有语法、词法规则的系统。语言是思维的工具，思维是通过语言来表述的。计算机程序设计语言是计算机可以识别的语言，用于描述解决问题的方法，供计算机阅读和执行。

1.1.1　机器语言与汇编语言

自从 1946 年 2 月世界上第一台数字电子计算机 ENIAC 诞生以来，在这短暂的 60 多年间，计算机科学得到了迅猛发展，计算机及其应用已渗透到社会的各个领域，有力地推动了整个信息化社会的发展，计算机已成为信息化社会中必不可少的工具。

计算机系统包括硬件和软件。计算机之所以有如此强大的功能，不仅因为它具有强大的硬件系统，而且依赖于软件系统。软件包括了使计算机运行所需的各种程序及有关的文档资料。计算机的工作是用程序来控制的，离开了程序，计算机将一事无成。程序是指令的集合。软件工程师将解决问题的方法、步骤编写为由一条条指令组成的程序，输入到计算机的存储设备中，计算机执行这一指令序列，便可完成预定的任务。

所谓指令，就是计算机可以识别的命令。虽然在人类社会中，各民族都有丰富的语言

来表达思想、交流感情、记录信息，但计算机却不能识别它们。计算机所能识别的指令形式，只能是简单的“0”和“1”的组合。一台计算机硬件系统能够识别的所有指令的集合，称为它的指令系统。

由计算机硬件系统可以识别的二进制指令组成的语言称为机器语言。毫无疑问，虽然机器语言便于计算机识别，但对于人类来说却是晦涩难懂，更难以记忆。在计算机发展的初期，软件工程师们只能用机器语言来编写程序。这一阶段，在人类的自然语言和计算机编程语言之间存在着巨大的鸿沟，软件开发的难度大、周期长，开发出的软件功能却很简单，界面也不友好。

不久，出现了汇编语言，它将机器指令映射为一些可以被人读懂的助记符，如 ADD、SUB 等。此时，编程语言与人类自然语言间的鸿沟略有缩小，但仍与人类的思维方式相差甚远。因为它的抽象层次太低，程序员需要考虑大量的机器细节。

尽管如此，从机器语言到汇编语言，仍是一大进步。这意味着人与计算机的硬件系统不必非得使用同一语言。程序员可以使用较适合人类思维习惯的语言，而计算机硬件系统仍只识别机器指令。那么两种语言间的沟通如何实现呢？这就需要一种翻译工具（即软件）。汇编语言的翻译软件称为汇编程序，它可以将程序员写的助记符直接转换为机器指令，然后再由计算机去识别执行。

1.1.2 高级语言

高级语言的出现是计算机编程语言的一大进步。它屏蔽了机器的细节，提高了语言的抽象层次。程序中可以采用具有一定含义的数据命名和容易理解的执行语句，这使得在书写程序时可以联系到程序所描述的具体事物。

20 世纪 60 年代末开始出现的结构化编程语言进一步提高了语言的层次。结构化数据、结构化语句、数据抽象、过程抽象等概念，使程序更便于体现客观事物的结构和逻辑含义。这使得编程语言与人类的自然语言更接近。但是二者之间仍有不少差距，主要问题是程序中的数据和操作分离，不能够有效地组成与自然界中的具体事物紧密对应的程序成分。

目前应用比较广泛的几种高级语言有 FORTRAN、BASIC、PASCAL、C 等。当然，本书介绍的 C++语言也是高级语言，但它与其他面向过程的高级语言有着根本的不同。

1.1.3 面向对象的语言

面向对象的编程语言与以往各种编程语言的根本不同点在于：它设计的出发点就是为了能更直接地描述客观世界中存在的事物（即对象）以及它们之间的关系。

开发一个软件的目的是为了解决某些问题，这些问题所涉及的业务范围称为该软件的问题域。面向对象的编程语言将客观事物看作具有属性和行为（或称服务）的对象，通过抽象找出同一类对象的共同属性（静态特征）和行为（动态特征）形成类。通过类的继承与多态可以很方便地实现代码重用，大大缩短了软件开发周期，并使得软件风格统一。因此，面向对象的编程语言使程序能够比较直接地反映问题域的本来面目，使软件开发人员能够利用人类认识事物所采用的一般思维方法来进行软件开发。

面向对象的程序设计语言经历了一个很长的发展阶段。例如，LISP 家族面向对象语言、

Simula67 语言、Smalltalk 语言，以及 Ada、Modula-2 等语言，都或多或少地引入了面向对象的概念，其中 Smalltalk 是第一个真正的面向对象的程序语言。

然而，应用最广的面向对象程序语言是在 C 语言基础上扩充出来的 C++语言。由于 C++对 C 兼容，而 C 语言又早已被广大程序员所熟知，所以，C++语言也就理所当然地成为应用最广的面向对象程序语言。

1.2 C++语言概述

1.2.1 C++的产生

C++是从 C 语言发展演变而来的，因此介绍 C++就不能不先回顾一下 C 语言。C 语言最初是贝尔实验室的 Dennis Ritchie 在 B 语言基础上开发出来的，1972 年在一台 DEC PDP-11 计算机上实现了最初的 C 语言。目前比较流行的 C 语言版本基本上都是以 ANSI C 为基础的。

C 语言具有许多优点，例如语言简洁灵活、运算符和数据结构丰富、具有结构化控制语句、程序执行效率高，而且同时具有高级语言与汇编语言的优点。与其他高级语言相比，C 语言具有可以直接访问物理地址的优点，与汇编语言相比又具有良好的可读性和可移植性。因此，C 语言得到了极为广泛的应用，有大量的程序员在使用 C 语言，并且，有许多 C 语言的库代码和开发环境。

由于 C 语言毕竟是一个面向过程的编程语言，因此与其他面向过程的编程语言一样，已经不能满足运用面向对象方法开发软件的需要。C++便是在 C 语言基础上为支持面向对象的程序设计而研制的一个通用目的的程序设计语言，它是在 1980 年由贝尔实验室的 Bjarne Stroustrup 博士创建的。

研制 C++的一个首要目标是使 C++首先是一个更好的 C，所以 C++根除了 C 中存在的问题；C++的另一个重要目标就是支持面向对象的程序设计，因此在 C++中引入了类的机制。最初的 C++被称为“带类的 C”，1983 年正式取名为 C++。C++语言的标准化工作从 1989 年开始，于 1994 年制定了 ANSI C++标准草案，以后又经过不断完善，成为目前的 C++。

1.2.2 C++的特点

C++语言的主要特点表现在两个方面，一是全面兼容 C，二是支持面向对象的方法。

首先，C++的确是一个更好的 C。它保持了 C 的简洁、高效和接近汇编语言等特点，对 C 的类型系统进行了改革和扩充，因此 C++比 C 更安全，C++的编译系统能检查出更多的类型错误。

由于 C++与 C 保持兼容，这使许多 C 代码不经修改就可以为 C++所用，用 C 编写的众多的库函数和实用软件可以用于 C++中。另外，由于 C 语言已被广泛使用，因而极大地促进了 C++的普及和面向对象技术的广泛应用。

然而，也正是由于对 C 的兼容使得 C++不是一个纯正的面向对象的语言，C++既支持

面向过程的程序设计又支持面向对象的程序设计。

C++语言最有意义的部分是其支持面向对象的特征。虽然与C的兼容使得C++具有双重特点，但它在概念上是与C语言完全不同的，我们应该使用面向对象的方式去编写程序。

如果读者具有其他面向过程高级语言的编程经验，在使用C++时应该更多地使用它的面向对象的特征。对于与C语言的兼容部分只需要适当注意一下就可以了，因为C语言与其他面向过程的语言在设计上是类似的。

如果读者是初学编程，虽然与C兼容的部分不是C++的主要方面，但你依然不能超越它。像数据类型、算法控制结构、函数等，不仅是面向过程程序设计的基本成分，也是面向对象编程的基础。因为，对象是程序的基本单位，然而对象的静态属性往往需要用某种类型的数据来表示，对象的动态属性要由成员函数来实现，而函数的实现归根到底还是算法的设计。

1.3 结构化程序设计

结构化程序设计的思想是在20世纪60年代末、70年代初为解决“软件危机”而形成的。多年来的实践证明，结构化程序设计策略确实使程序的执行效率提高了很多，并且由于减少了程序的出错率而大大减少了维护费用。

那么，什么是结构化程序设计呢？至今仍众说纷纭，还没有一个严格的又能被大家普遍接受的定义。

结构化程序设计就是一种进行程序设计的原则和方法，按照这种原则和方法可以设计出结构清晰、容易理解、容易修改、容易验证的程序。即：结构化程序设计是按照一定的原则与原理，组织和编写正确且易读的程序的软件技术。结构化程序设计的目标在于使程序具有一个合理结构，以保证和验证程序的正确性，从而开发出正确、合理的程序。

按照结构化程序设计的要求，设计出的程序设计语言称为结构程序设计语言。利用结构程序设计语言，或者说按结构化程序设计的思想和原则编制出的程序称为结构化程序。

结构化程序设计的主要特征与风格如下：

（1）一个程序按结构化设计方式构造时，一般地总是一个结构化程序，即由三种基本控制结构——顺序结构、选择结构和循环结构构成。这三种结构都是单入口/单出口的程序结构。已经证明，一个任意大且复杂的程序总能转换成这三种标准形式的组合。

（2）有限制地使用goto语句。鉴于goto语句的存在使程序的静态书写顺序与动态执行顺序十分不一致，导致程序难读难理解，容易存在潜在的错误，难于证明正确性，有人主张程序中禁止使用goto语句，但有人则认为goto语句是一种有效设施，不应全盘否定而完全禁止使用。结构化程序设计并不在于是否使用goto语句，因此作为一种折中，允许在程序中有限制地使用goto语句。

（3）借助于体现结构化程序设计思想的所谓结构化程序设计语言来书写结构化程序，并采用一定的书写格式以提高程序结构的清晰性，增进程序的易读性。

（4）强调程序设计过程中人的思维方式与规律，是一种自顶向下的程序设计策略。它通过一组规则、规律与特有的风格对程序设计细分和组织。对于小规模的程序设计，它与逐步精化的设计策略相联系，即采用自顶向下、逐步求精的方法对其进行分析和设计；对

于大规模的程序设计，则与模块化程序设计策略相结合，即将一个大规模的问题划分为几个模块，每一个模块完成一定的功能。

结构化程序设计成功地为处理复杂问题提供了有力的手段。然而，到了20世纪80年代末，它的一些缺点越来越突出，主要表现在：当数据量大增时，数据与处理这些数据的方法之间的分离使程序变得越来越难以理解。对数据处理能力的需求越强，这种分离所造成的负面影响越显著。

采用结构化程序设计方法的程序员发现，每一种相对于老问题的新方法都要带来额外的开销，与可重用性相对，通常称之为重复投入。基于可重用性的思想是指建立一些具有已知特性的部件，在需要时可以插入到程序之中。这是一种模仿硬件组合方式的做法，当工程师需要一个新的晶体管时，他不用自己去创造，只要到仓库去找就行了。对于软件工程师来说，在面向对象程序设计出现之前，一直缺乏具备这种能力的工具。

1.4 面向对象程序设计

1.4.1 面向对象的方法

程序设计语言是编写程序的工具，因此程序设计语言的发展恰好反映了程序设计方法的演变过程。这里首先初步介绍一下面向对象方法的基本概念和基本思想，当您学习完本书之后，相信您会对面向对象的方法有一个深入、完整的认识。

面向对象方法是什么呢？首先，它将数据及对数据的操作方法放在一起，作为一个相互依存、不可分离的整体——对象。对同类型对象抽象出其共性，形成类。类中的大多数数据，只能用本类的方法进行处理。类通过一个简单的外部接口与外界发生关系，对象与对象之间通过消息进行通讯。这样，程序模块间的关系更为简单，程序模块的独立性、数据的安全性就有了良好的保障。另外，通过后续章节中将要介绍的继承与多态性，还可以大大提高程序的可重用性，使得软件的开发和维护都更为方便。

面向对象的方法有如此的优点，然而对于初学程序设计的人来说是否容易理解、容易掌握呢？回答是肯定的。面向对象方法的出现，实际上是程序设计方法发展的一个返璞归真过程。软件开发从本质上讲，就是对软件所要处理的问题域进行正确地认识，并把这种认识正确地描述出来。面向对象方法所强调的基本原则，就是直接面对客观存在的事物进行软件开发，将人们在日常生活中习惯的思维方式和表达方式应用在软件开发中，使软件开发从过分专业化的方法、规则和技巧中回到客观世界，回到人们通常的思维方式。

现在，简单介绍一下面向对象方法中的几个基本概念。当然我们不能期望通过几句话的简单介绍就完全理解这些概念，在本书的后续章节中，会不断帮助读者加深对这些概念的理解，以达到熟练运用。

1. 对象

从一般意义上讲，对象是现实世界中一个实际存在的事物，它可以是有形的（比如一辆汽车），也可以是无形的（比如一项计划）。对象是构成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（对象所表现的行为或具有的功能）。

面向对象方法中的对象，是系统中用来描述客观事物的一个实体，它是用来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，行为是用来描述对象动态特征的操作序列。

2. 类

把众多的事物归纳、划分成一些类，是人类在认识客观世界时经常采用的思维方法。分类所依据的原则是抽象，即忽略事物的非本质特征，只注意那些与当前目标有关的本质特征，从而找出事物的共性，把具有共同性质的事物划分为一类，得出一个抽象的概念。例如，石头、树木、汽车、房屋等都是人们在长期的生产和生活实践中抽象出的概念。

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。类与对象的关系犹如模具与铸件之间的关系，一个属于某类的对象称为该类的一个实例。

3. 封装

封装是面向对象方法的一个重要原则，就是把对象的属性和服务结合成一个独立的系统单位，并尽可能隐藏对象的内部细节。这里有两个含义：第一个含义是把对象的全部属性和全部服务结合在一起，形成一个不可分割的独立单位；第二个含义也称作“信息隐藏”，即尽可能隐藏对象的内部细节，对外形成一个边界（或者说一道屏障），只保留有限的对外接口使之与外部发生联系。

4. 继承

继承是面向对象技术能够提高软件开发效率的重要原因之一，其定义是特殊类的对象拥有其一般类的全部属性与服务，称作特殊类对一般类的继承。

继承具有重要的实际意义，它简化了人们对事物的认识和描述。例如，我们认识了轮船的特征之后，再考虑客轮时，因为知道客轮也是轮船，于是可以认为它理所当然地具有轮船的全部一般特征，只需要把精力用于发现和描述客轮独有的那些特征。

继承对于软件复用有着重要意义，使特殊类继承一般类，本身就是软件复用。如果将开发好的类作为构件放到构件库中，在开发新系统时便可以直接使用或继承使用。

5. 多态性

多态性是指在一般类中定义的属性或行为被特殊类继承之后，可以具有不同的数据类型或表现出不同的行为。这使得同一个属性或行为在一般类及其各个特殊类中具有不同的语义。

例如，我们可以定义一个一般类“几何图形”，它具有“绘图”行为，但这个行为并不具有具体含意，也就是说并不确定在执行时到底画一个什么样的图（因为此时尚不知道“几何图形”到底是一个什么图形，“绘图”行为当然也就无从实现）。然后再定义一些特殊类，如“椭圆”和“多边形”，它们都继承一般类“几何图形”，因此也就自动具有了“绘图”行为。接下来，可以在特殊类中根据具体需要重新定义“绘图”，使之分别实现画椭圆和多边形的功能。进而，还可以定义“矩形”类继承“多边形”类，在其中使“绘图”实现绘

制矩形的功能。这就是面向对象方法中的多态性。

1.4.2 面向对象的软件开发

在整个软件开发过程中，编写程序只是相对较小的一部分。软件开发的真正决定性因素来自前期概念问题的提出，而非后期问题的实现。只有识别、理解和正确表达了应用问题的内在实质，才能做出好的设计，然后，才是具体的编码实现。

早期的软件开发所面临的问题比较简单，从认清要解决的问题到编程实现并不是太难的事。随着计算机应用领域的扩展，计算机所处理的问题日益复杂，软件系统的规模和复杂度空前扩大，以至于软件的复杂性和其中包含的错误已到了软件人员无法控制的程度，这就是20世纪60年代初期的“软件危机”。软件危机的出现，促进了软件工程学的形成与发展。

学习面向对象的程序设计，首先应该对软件开发和维护的全过程有一个初步的了解。因此，在这里先简要介绍一下什么是面向对象的软件工程。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，它包括面向对象的分析（OOA）、面向对象的设计（OOD）、面向对象的编程（OOP）、面向对象的测试（OOT）和面向对象的软件维护（OOSM）等主要内容。

1. 面向对象的分析

从问题的陈述着手，建立一个说明系统重要性的真实情况模型。为理解问题，系统分析员需要与客户一起工作。系统分析阶段应该扼要精确地抽象出系统必须做什么，而不是关心如何去实现。

面向对象的系统分析，直接用问题域中客观存在的事物建立模型中的对象，无论是对单个事物还是对事物之间的关系，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。

2. 面向对象的设计

在设计阶段，是针对系统的一个具体实现运用面向对象的方法。其中包括两方面的工作，一是把OOA模型直接搬到OOD，作为OOD的一部分；二是针对具体实现中的人机界面、数据存储、任务管理等因素补充一些与实现有关的部分。

3. 面向对象的编程

编程是面向对象的软件开发最终落实的重要阶段。在OOA和OOD理论出现之前，程序员要写一个好的面向对象的程序，首先要学会运用面向对象的方法来认识问题域，所以OOP被看作一门比较高深的技术。现在，OOP的工作比较简单了，认识问题域与设计系统成分的工作已经在OOA和OOD阶段完成，OOP的工作就是用一种面向对象的编程语言把OOD模型中的每个成分写出来。

尽管如此，学习面向对象的程序设计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。因此，虽然本书面向的是初学编程的读者，介绍的主要是C++语言和面向对象的程序设计方法，但仍然用了一定的篇幅通过例题介绍设计思路。