21世纪 高等学校电子信息类专业规划教材

面向对象程序设计

周生炳 万映辉 邸晓奕 编著

清华大学出版社
http://www.tup.tsinghua.edu.cn
北京交通大学出版社
http://press.bjtu.edu.cn

21世纪高等学校电子信息类专业规划教材

面向对象程序设计

周生炳 万映辉 邸晓奕 编著

清华大学出版社
北京交通大学出版社
·北京·

内 容 简 介

本书介绍面向对象的基本理论以及如何运用C++语言实现面向对象的程序设计。本书作者根据自己学习C++的亲身体会及多年教学经验，用简单的例子和简练的叙述讲解C++编程，注重理论和实践的结合，使读者在掌握基本理论的同时，提高实际动手能力。

全书共分11章。其中第1章主要介绍面向对象的基本理论；第2章至第5章主要介绍C++程序设计语言、程序结构以及过程化程序设计基础知识；第6章至第11章主要介绍面向对象的程序设计方法，内容涉及类、重载、继承、多态、异常处理和模板等方面的知识。

本书是作者总结多年教学实践和科研开发经验写成的，可作为大学计算机专业程序设计基础课程教材，也可作为相关专业人员的参考用书。

版权所有，翻印必究。

本书封面贴有清华大学出版社激光防伪标签，无标签者不得销售。

图书在版编目（CIP）数据

面向对象程序设计/周生炳，万映辉，邸晓奕编著. —北京：清华大学出版社；北京交通大学出版社，2004.9

（21世纪高等学校电子信息类专业规划教材）

ISBN 7-81082-418-X

Ⅰ. 面… Ⅱ. ①周… ②万… ③邸… Ⅲ. 面向对象语言-程序设计-高等学校-教材 Ⅳ. TP312

中国版本图书馆CIP数据核字（2004）第093734号

责任编辑：李斌

出 版 者：清华大学出版社　邮编：100084　电话：010-62776969

北京交通大学出版社　邮编：100044　电话：010-51686045，62237564

印 刷 者：北京鑫海金澳胶印有限公司

发 行 者：新华书店总店北京发行所

开　本：185×260　印张：18.75　字数：452千字

版　次：2004年9月第1版　2004年9月第1次印刷

书　号：ISBN 7-81082-418-X / TP · 151

印　数：1～5 000册　定价：25.00元

前　言

面向对象技术是20世纪90年代以来计算机应用领域迅速发展起来的一个新生事物，它的出现被认为是程序设计方法学的一场革命，在现代程序设计思想中占有重要的地位。面向对象程序设计思想的核心是用类来表示各种概念。在这种思想影响下，诞生了一批面向对象程序设计语言，通过各种语言和程序设计范型的演化，可以清晰地看到面向对象程序设计技术的各种优势，它克服了面向过程编程方式中数据与算法完全分离的弊端，并在此基础上演化出了多种新的组织程序和表达概念的方法。

C++语言是目前应用较广的面向对象的程序设计语言，使用它可以实现面向对象的程序设计。C++被认为是目前世界上功能最强大的计算机语言之一，是通往面向对象程序设计的有效途径。对于开发高性能的软件，C++是一种卓越的程序设计语言，同时，在全世界范围内，它也是程序员的首选语言。C++并不只是一种流行的程序设计语言，它还为其他几种程序设计语言以及许多先进计算思想提供了理论基础。C#和Java这两种重要的语言都来源于C++，它的语法、风格以及设计思想在很多方面都影响着现代程序设计。

本书介绍面向对象的基本理论以及如何运用C++语言实现面向对象的程序设计。本书将从C++的基础知识开始，由浅入深地介绍该语言所有的核心内容。此外，在本书的后面还介绍了C++的高级特征。作者根据自己学习C++的亲身体会及多年教学经验，用通俗的例子和简练的叙述讲解C++编程，注重理论和实践的紧密结合，使读者在掌握基本理论的同时，提高实际动手能力。

全书共分11章。第1章主要介绍面向对象程序设计的基本理论，重点包括面向对象程序设计的基本思想、概念、特点及基本框架；第2～5章介绍C++程序设计语言、程序结构以及过程化基础，重点包括基本数据类型和表达式的概念及使用、程序结构控制的方法及函数、数组、指针等概念及使用方法；第6～11章介绍面向对象的程序设计方法，主要介绍类、重载、继承、多态、异常处理和模板等方面的知识，重点包括类的定义及面向对象系统的封装性、友元的定义及使用、通过重载和继承等机制实现面向系统的多态性、异常处理的机制与实现以及模板的概念与使用等。

全书的编写工作由周生炳统一组织实施，第1～5章由邸晓奕编写，第6～11章由万映辉编写。本书是作者总结多年教学实践和科研开发的经验写成的，全书力求做到内容全面，概念清楚，通俗易懂，并注重实用性和先进性。本书可作为大学计算机专业程序设计基础课程教材，也可作为相关专业人员的参考用书。

由于水平有限，时间仓促，书中难免有错误或不妥之处，诚恳期待读者的批评指正和建议，以供再版时参考，使本书日臻完善。

作者

2004年9月

目　录

试读结束：需要全本请在线购买：　www.ertongbook.com

第 1 章　面向对象程序设计的基本知识

面向对象程序设计是一种围绕真实世界的概念来组织模型的程序设计方法，它采用对象来描述问题空间的实体，注重数据和操作之间密不可分的内在联系，并将数据和操作进行抽象，从而封装成一个统一的整体。本章将重点介绍面向对象程序设计的基本思想和面向对象的基本概念。

1.1　面向对象程序设计的基本思想

计算机软件和硬件的不断发展是相互促进的。由于早期的计算机运行速度慢，存储容量小，且硬件价格昂贵，软件的规模一般也比较小，因此程序人员主要考虑的是如何用最少的计算机指令，达到节省内存空间、提高运算速度的目的，而很少考虑程序规范化的问题，使得编写的程序可读性和可维护性很差。随着计算机的发展，硬件成本急剧下降，处理能力却越来越强，软件所能处理的问题越来越多，越来越复杂，软件的规模也越来越大，一些大型的软件系统由个人在短期内完成已不大可能。而早期的程序设计方法具有个人随意性，未能考虑到协同工作的需要，使得软件设计周期加长、成本增高、难以维护和扩充，不便于产业化。在这样一种情况下，人们急于找到一种规范化的程序设计方法，使程序设计人员共同遵守相对统一的一种规则。随着软件规模的不断扩大，结构化程序设计思想以及面向对象程序设计思想相继产生，使程序设计逐步走向规范化的道路。在面向对象程序设计语言产生之后，面向对象程序设计逐步成为编码的主流，其中所蕴涵的面向对象的思想不断向开发过程的上游和下游发展，形成现在的面向对象分析、面向对象设计、面向对象测试等，并一起逐步发展为面向对象的软件开发方法。

1.1.1　结构化程序设计思想

结构化程序设计(Structure Programming)诞生于 20 世纪 60 年代，盛行于 20 世纪 70—80 年代，成为当时所有软件开发设计领域及每个程序员都采用的程序设计方法，它的产生和发展形成了现代软件工程的基础。

结构化程序设计从系统的功能入手，按照工程的标准和严格的规范将系统分解为若干功能模块，系统是实现模块功能的函数和过程的集合。在设计上，结构化程序设计产生自顶向下、结构清晰的树状系统结构。每个模块尽可能保持较强的独立性，各模块之间的关系尽可能简单，每个模块内部均是由顺序、选择和循环 3 种基本结构组成，其模块化实现的具体方法主要是使用子程序。

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，但它是一种面向数据和过程的设计方法。结构化程序设计把数据和过程分离为相互独立的实体，程序员在编程时必

原书缺页

试读结束：需要全本请在线购买： www.ertongbook.com

1.1.3　面向对象的程序设计语言

面向对象程序设计语言必须支持抽象数据类型和继承性。但是在某些不支持抽象数据类型的程序设计语言中，可以使用特别的编程技术和约束规则在一定程度上实现抽象数据类型，这是一种把抽象数据类型看成程序设计方法的观点。因此有人认为，用传统的程序设计语言也能在一定程度上进行面向对象程序设计，这是一种把面向对象看成编程技巧的观点，并不是真正意义上的面向对象编程。面向对象的程序设计语言规则提供了特定的语法成分来保证和支持面向对象程序设计，并且提供了继承性、多态性和动态链接机制，使得类和类库成为可重用的程序模块。

面向对象程序语言经历了一个比较长的发展阶段。诸如“对象”和“对象的属性”这样的概念，可以追溯到 1950 年前后，它们首先出现在关于人工智能的早期著作中。然而，OOP 的实际发展却是始于 1966 年，当时 Kisten Nygaard 和 Ole-Johan Dahl 开发了具有更高级抽象机制的 Simula 语言。Simula 提供了比子程序更高一级的抽象和封装，为仿真一个实际问题，引入了数据抽象和类的概念。大约在同一时期，Alan Kay 正在尤他大学的一台个人计算机上努力工作，他希望能在计算机上实现图形化和模拟仿真。尽管由于软硬件的限制，Kay 的尝试没有成功，但他并没有放弃。20 年代 70 年代初期，他加入了 Palo AltoResearch Center 研究中心(PARC)，再次将这些想法付诸实施。

在 PARC，Alan Kay 所在的研究小组坚信计算机技术是改善人与人、人与机器之间通信渠道的关键。在这种信念的支持下，他们吸取了 Simula 的类的概念，开发出 Smalltalk 语言，1972 年 PARC 发布了 Smalltalk 的第 1 个版本。大约在此时，“面向对象”这一术语正式确定。Smalltalk 被认为是第一个真正面向对象的程序设计语言，Smalltalk 的目标是为了使软件设计能够以尽可能自动化的单元来进行。在 Smalltalk 中一切都是对象，即某个类的实例。

Smalltalk-80 是 PARC 的一系列 Smalltalk 版本的总结，发布于 1981 年，这标志着 PARC 的面向对象思想的启航。早期 Smalltalk 关于开发环境的研究导致了后来的一系列进展，如窗口(window)，图标(icon)，鼠标(mouse)和下拉式 window 环境等。Smalltalk 语言还影响了 20 世纪 80 年代早期和中期的面向对象的语言，如 Object-C(1986)、C++(1986)、Self(1987)、Eiffl(1987)和 Flavors(1986)等。面向对象的应用领域也被进一步拓宽。对象不再仅仅与名词相联系，还包括事件和过程。1990 年以来，面向对象的分析、测试、度量和管理等研究都得到了长足发展。目前对象技术的前沿课题包括设计模式(design patterns)、分布式对象系统和基于网络的对象应用等。

在 20 世纪 80 年代，C 语言成为一种极其流行、应用广泛的语言。C++是在 C 语言的基础上进行扩充，并增加了类似 Smalltalk 语言中相应的对象机制，它将“类”看作是用户自定义类型，使其扩充比较自然。C++以其高效的执行赢得了广大程序设计者的喜爱，在 C++中提供了对传统语言 C 的向后兼容性，因此，很多已有的程序原封不动或稍加改造就可以重用，许多有效的算法也可以重新利用。它是一种混合型的面向对象程序设计语言，由于它的出现，才使面向对象的程序设计语言越来越得到重视和广泛的应用。

C++有很多种类，如 MS C++，Turbo C++，Borland C++，C++ Builder 和 VC++等，每一种又有几个版本与之相对应。

综观所有的面向对象程序设计语言，可以把它们分为两类，纯粹型的面向对象语言和混合型的面向对象语言。在纯粹型的面向对象语言中，几乎所有的语言成分都是对象，这类语言强调开发快速原型的能力；而混合型的面向对象语言，是在原传统的过程化语言中加入各种面向对象的语法机制，它强调的是运行效率。

1.2　面向对象程序设计的基本概念

相对于结构化程序设计来讲，面向对象程序设计理论扩充了许多新的概念和术语。要想理解和掌握面向对象的理论，必须从最基本的概念入手，通过对最基本概念的掌握来真正认识面向对象方法的作用。本节主要介绍对象、类、消息的基本概念以及彼此之间的关系。

1.2.1　对象的基本概念

面向对象程序设计中的对象具有两方面的含义，即在现实世界中的含义和在计算机世界中的含义。

在现实世界中，可以将任何客观存在的事物都看作一个对象，如一个人、一辆汽车、一棵树，甚至一个星球。一方面，对象与对象之间存在着一定的差异，如一棵树和一辆汽车是两个截然不同的物体；另一方面，对象与对象之间可能又存在某些相似性。如一辆白色的自行车和一辆红色的自行车，两者都是自行车，具有相同的结构和工作原理，仅仅是颜色不同而已。

对象既具有一些静态的特征，如一个人的体重、身高和年龄；还具有一些动态的特征，如一个人的说话或交友习惯等。另外，每一个对象都具有一个名字以区别其他对象，如学生张三和学生李四。

在计算机世界中，对象(Object)是一个现实实体的抽象。一个对象可被认为是一个将数据(属性)和程序(方法)封装在一起的实体，这个程序产生该对象的动作或对它接受到的外界信号的反应，这些对象操作有时称为方法。

对象是建立面向对象程序所依赖的基本单元。用更专业的话来说，所谓对象就是一种代码的实例，这种代码执行特定的功能，具有自包含或者封装的性质。在结构化程序设计中，变量可以看作是简化了的对象。换句话说，变量是仅仅具有单一属性且不具有方法的对象，这里的单一属性便是变量的取值，变量名就是对象名。

通过上面的分析，无论是现实世界中的对象还是计算机世界中的对象，它们都具有如下共同特征。

(1) 每个对象都有一个名字以区别其他对象。

(2) 每个对象都有一组状态用来描述它的某些特征。

(3) 对象通常包含一组操作，每个操作决定对象的一种功能或行为。

(4) 对象所包含的操作可能是自身内部的操作，也可能是施加于其他对象的操作。

在一个面向对象的系统中，对象是运行期的基本实体。它可以用来表示一个人或一个银行账户，一张数据表格，或其他什么需要被程序处理的东西。它也可以用来表示用户定

义的数据，如一个向量，时间或列表。在面向对象程序设计中，问题的分析一般以对象及对象间的自然联系为依据。对象在内存中占有一定空间，并且具有一个与之关联的地址。

当一个程序运行时，对象与对象之间通过互发消息来相互作用。例如，程序中包含一个 customer 对象和一个 account 对象，而 customer 对象可能会向 account 对象发送一个消息，查询其银行账目。每个对象都包含数据以及操作这些数据的代码，即使不了解彼此的数据和代码的细节，对象之间依然可以相互作用，所要了解的只是对象能够接受的消息的类型，以及对象返回的响应的类型，尽管不同的人会以不同的方法实现它们。

1.2.2　类的基本概念

类构成了面向对象程序设计的基础，它把数据和函数封装在一起，是具有相同操作功能和相同数据格式(属性)的对象抽象，它可以被看作抽象数据类型的具体实现。

在程序设计语言中，数据类型本质上是抽象的。高级程序设计语言从位、字节和字中抽象出字符、整数和实数等基本数据类型，这使程序员比使用位等设计程序方便多了，因为整数、实数的抽象比位、字节的抽象更接近人类的应用。但是在实际应用中，程序设计语言中所提供的数据类型总是有限的。例如，在 C++中没有矩阵、方程组、矢量等数据类型，也没有年龄、地址等数据类型。这些数据类型是人们应用抽象到一组对象上而得到的抽象数据类型。在程序中声明一个新类将导致产生一种新的数据类型，类的设计就是数据类型的设计。

1.2.3　类与对象的关系

简单说来，类是用户自定义的数据类型，是用来描述具有相同的属性和方法的对象的集合，它定义了该集合中每个对象所共有的属性和方法，对象是类的实例。例如，苹果是一个类，而放在桌上的那个苹果则是一个对象。对象和类的关系相当于一般的程序设计语言中变量和数据类型的关系。

对象包含数据以及操作这些数据的代码。一个对象所包含的所有数据和代码可以通过类来构成一个用户定义的数据类型。事实上，对象就是类类型(class type)的变量。一旦定义了一个类，就可以创建这个类的多个对象，每个对象与一组数据相关，而这组数据的类型在类中定义。因此，一个类就是具有相同类型的对象的抽象。例如，芒果、苹果和桔子都是 fruit 类的对象。类是用户自定义的数据类型，但在一个程序设计语言中，它和内建的数据类型行为相同，比如创建一个类对象的语法和创建一个整数对象的语法一模一样。如果 fruit 被定义为一个类，那么语句 fruit mango；就创建了一个 fruit 类的对象 mango。

1.2.4　消息

对象与对象之间并不是彼此孤立的，它们之间存在着联系，在面向对象系统中，对象之间的联系是通过消息来传递的。

消息是对象之间相互请求或相互协作的途径，是要求某个对象执行其中某个功能操作的规格说明。对象内有方法和数据，外部的用户或对象向该对象提出的服务请求，可以称为向该对象发送消息。合作是指两个对象共同承担的责任和分工，可以通过向对象发送消息来实现对象的操纵，对象收到发给它的消息后，或者执行一个内部操作(有时称为方法或

过程)，或者再去调用其他对象的操作。通常，类的公有部分描述了可以发送给对象的消息，并给出了相应的方法。

通常把发送消息的对象称为发送者，把接收消息的对象称为接收者。对象间的联系只能通过传递消息来进行。对象也只有在收到消息时才被激活，被激活后的对象代码将“知道”如何去操作它的私有数据，去完成接收到的消息要求的操作。消息具以下 3 个性质。

(1) 同一个对象可接收不同形式的多个消息，产生不同的响应。

(2) 相同形式的消息可以传送给不同的对象，所做出的响应可以是不同的。

(3) 消息的发送可以不考虑具体的接收者，对象可以响应消息，也可以不响应消息，对消息的响应并不是必须的。

1.3　面向对象程序语言的基本特点

面向对象系统最突出的特性就是抽象、封装、继承和多态。衡量一种高级语言是否是面向对象的程序设计语言，主要看它是否具有这几种特征。

1.3.1　抽象

抽象就是忽略一个主题中与当前目标无关的那些方面，以便更充分地注意与当前目标有关的方面。抽象并不打算了解全部问题，而只是选择其中的一部分，例如，要设计一个学生成绩管理系统，考察学生这个对象时，只需关心他的班级、学号和成绩等，而不用去关心他的身高、体重等信息。

抽象包括两个方面，一是过程抽象，二是数据抽象。过程抽象是指任何一个明确定义功能的操作都可被使用者看作单个的实体，尽管这个操作实际上可能由一系列更低级的操作来完成。数据抽象定义了数据类型和施加于该类型对象上的操作，并限定了对象的值只能通过这些操作来修改和观察。

1.3.2　封装

封装是面向对象的特征之一，是对象和类概念的主要特性。封装是把过程和数据包围起来，避免了外界的干扰和不确定性，对数据的访问只能通过已定义的界面来进行。面向对象设计始于这个基本概念，即现实世界可以被描绘成一系列完全自治、封装的对象，这些对象通过一个受保护的接口访问其他对象。一旦定义了一个对象的特性，则有必要决定这些特性的可见性，即哪些特性对外部世界是可见的，哪些特性用于表示内部状态，并在这个阶段定义对象的接口。通常，应禁止直接访问一个对象的实际表示，而应通过操作接口访问对象，这称为信息隐藏。通过这种方式，对象对内部数据提供了不同级别的保护，以防止程序中无关的部分被意外地改变或错误地使用了对象的私有部分。事实上，信息隐藏是用户对封装性的认识，封装则为信息隐藏提供支持。封装保证了模块具有较好的独立性，使得程序维护修改较为容易。对应用程序的修改仅限于类的内部，因而可以将应用程序修改带来的影响减少到最低限度。

1.3.3　继承

继承是一种联结类的层次模型，并且允许和鼓励类的重用，它提供了一种明确表述共性的方法。对象的一个新类可以从现有的类中派生，这个过程称为类继承。新类继承了原始类的特性，新类称为原始类的派生类(子类)，而原始类称为新类的基类(父类)。派生类可以从它的基类那里继承方法和实例变量，并且可以修改或增加新的方法使之更适合特殊的需要，这也体现了大自然中一般与特殊的关系。

继承性解决了软件的可重用性(reusability)问题。也就是说，人们可以向一个已经存在的类中添加新的特性，而不必改变这个类。这可以通过从这个已存在的类派生一个新类来实现。这个新的类将具有原来那个类的特性以及新的特性。而继承机制的魅力和强大就在于它允许程序员利用已经存在的类(接近需要，而不是完全符合需要的类)，并且可以某种方式修改这个类，而不会影响类中其他可利用的特性。例如，所有的 Windows 应用程序都有一个窗口，它们可以被看作都是从一个窗口类派生出来的。但是有的应用程序用于文字处理，有的应用程序用于绘图，这是由于派生出了不同的子类，各个子类添加了不同的特性。

1.3.4　多态

多态性是指允许不同类的对象对同一消息做出响应。例如，同样的加法，把两个时间加在一起和把两个整数加在一起肯定完全不同。又比如，同样的选择编辑或粘贴操作，在字处理程序和绘图程序中有不同的效果。多态性包括参数化多态性和包含多态性。多态性语言具有灵活、抽象、行为共享、代码共享的优势，解决了应用程序函数同名问题。

多态机制使具有不同内部结构的对象可以共享相同的外部接口。这意味着，虽然针对不同对象的具体操作不同，但通过一个公共的类，它们可以通过相同的方式予以调用。多态在实现继承的过程中被广泛应用。面向对象程序设计语言支持多态，术语称之为 one interface multiple method(一个接口，多个实现)。简单来说，多态机制允许通过相同的接口引发一组相关但不相同的动作，通过这种方式，可以减少代码的复杂程度。在某个特定的情况下，应该做出怎样的动作，由编译器决定，而不需要程序员手工干预。

1.4　面向对象程序设计的模式

Ernest Tello(人工智能领域的著名专家)将软件技术的演化比喻为树的生长。同树的生长一样，软件的演化具有明显的阶段性，这些阶段称为层(layer)。在过去的 40 年中，这些层逐步被建立起来，每一个层都由前一个层发展而成。图 1-2 显示了这个发展过程，但是关于树的比喻在遇到层的生命期的问题时失败了。在软件系统中，每个层都在持续的发挥作用，而在树中，只有最上面的层才有用。

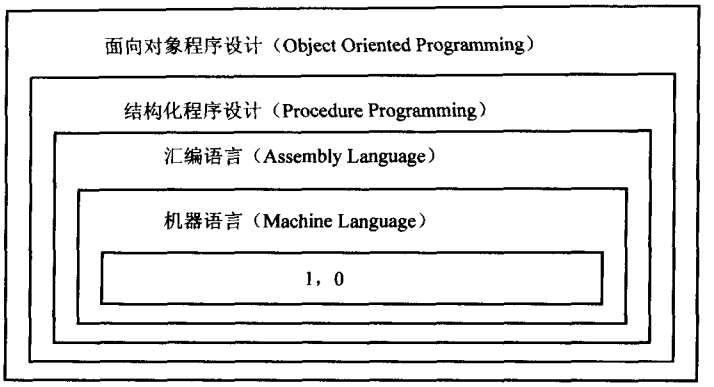

图 1-2　软件设计技术的发展

面向对象程序设计是完成程序设计工作的新方法。自从计算机发明以来，为了适应程序复杂性的不断增加，程序设计的方法有了很大的变化。汇编语言被发明出来以后，程序员们总算可以用符号表示那些机器指令，从而可以编写更长、更复杂的程序。当程序规模继续不停增长的时候，高级语言被引入，为程序员们提供了更多工具应对日益增加的复杂性。第一个被普遍使用的语言是 FORTRAN。不过虽然 FORTRAN 迈出了重大的第一步，但用它写出的代码很难说是清晰的和容易理解的。

1960 年结构化程序设计思想诞生。C 和 Pascal 等语言都大力提倡这种程序设计的方法。结构化程序设计语言使得编写较复杂的程序变得容易。但是一旦某个项目达到一定规模，即使使用结构化程序设计的方法，局势仍将变得不可控制。

在程序设计方法发展过程中，每一次重大突破都使得程序员可以应对更大的复杂性。在这条道路上迈出的每一步中，新的方法都运用和发展了以前的方法中最好的理念。今天，许多项目的规模又进一步扩大，为了解决这个问题，面向对象程序设计方法应运而生。

在详细介绍面向对象程序设计之前，先简单介绍一下面向过程程序设计的方法。在面向过程的程序设计方法中，问题被看作一系列将被完成的任务，如读、计算和打印。许多函数用于完成这些任务。问题的焦点集中于函数。图 1-3 显示了一个典型的面向过程的程序结构。分层分解的技术被用来确定一系列需要完成的任务，以解决特定的问题。

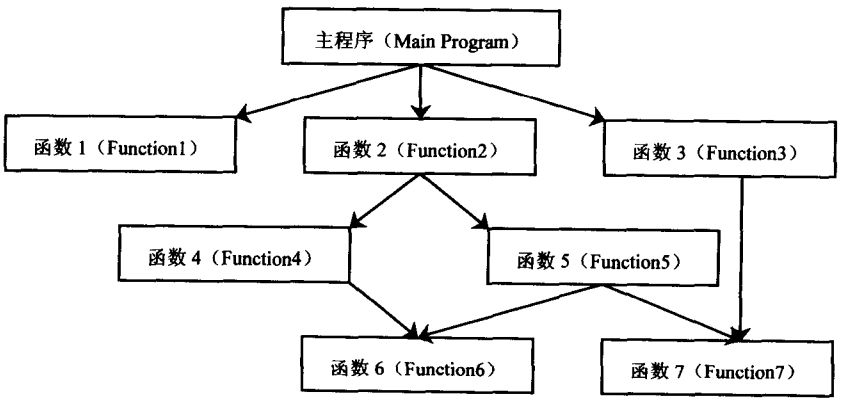

图 1-3　面向过程的程序设计模式

试读结束：需要全本请在线购买：　www.ertongbook.com