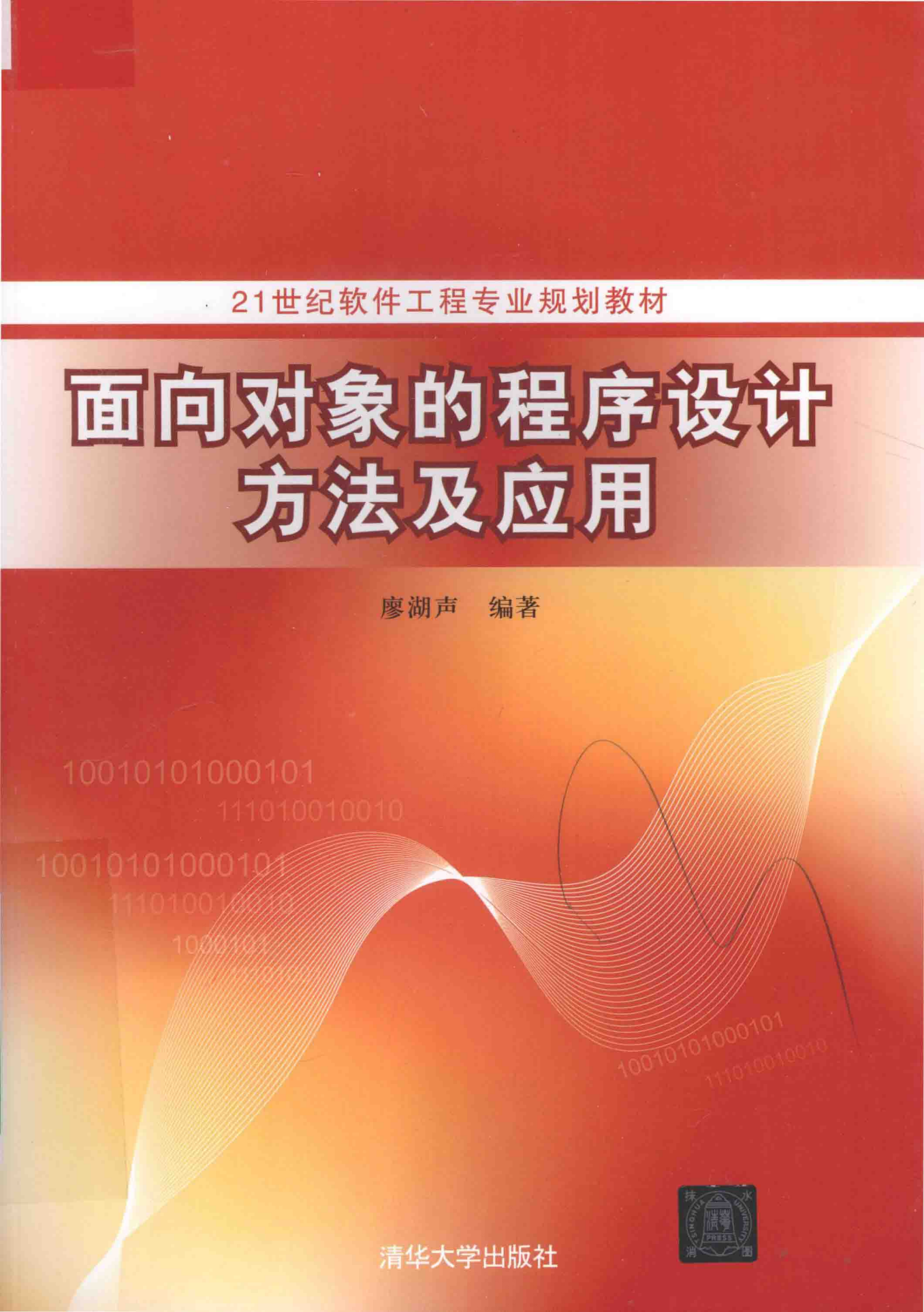
21世纪软件工程专业规划教材

面向对象的程序设计方法及应用

廖湖声　编著



清华大学出版社

21世纪软件工程专业规划教材

面向对象的程序设计方法及应用

廖湖声　编著

清华大学出版社
北京

内 容 简 介

本书以面向对象的设计方法为主线，通过大量实用的设计实例，由浅入深地介绍了对象基本概念、对象分类与设计、对象关系与复杂对象、多态性及其应用方法、软件建模和结构设计、设计模式及其应用等面向对象的方法与技术。本书采用C++语言作为程序设计语言，随着各种设计方法的讲述，逐步介绍了对象、类、继承、类模板、常用基本类库等语言功能的使用方法。

全书共6章：第1章介绍面向对象技术的由来、面向对象的软件开发方法、程序设计方法、程序设计语言及发展趋势；第2章阐述对象的基本概念，举例介绍了4种分类设计方法；第3章基于4个设计案例深入讨论对象关系和复杂对象的分析与设计方法；第4章介绍多态性的基本概念，通过三个设计案例深入分析了多态性的应用方法；第5章基于三个设计案例介绍了面向对象的软件建模和结构设计方法；第6章介绍常用的各种设计模式，提供了两个基于设计模式的软件设计案例。

本书可作为计算机相关专业“面向对象程序设计”等本科课程以及“面向对象方法与技术”等研究生课程的教材，也可以作为专业技术人员的参考书或培训教材。同时，本书提供的应用案例及其设计方法可以作为“软件工程”、“软件开发与体系结构”等课程的教学参考资料。

图书在版编目(CIP)数据

面向对象的程序设计方法及应用/廖湖声编著. —北京：清华大学出版社，2016
21世纪软件工程专业规划教材
ISBN 978-7-302-44100-7

Ⅰ. ①面…　Ⅱ. ①廖…　Ⅲ. ①面向对象语言－程序设计－高等学校－教材　Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2016)第132416号

责任编辑：张　玥　薛　阳
封面设计：常雪影
责任校对：梁　毅
责任印制：王静怡

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦A座　　**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
课件下载：http://www.tup.com.cn，010-62795954

印 装 者：三河市吉祥印务有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：19.25　　**字　　数**：478千字
版　　次：2016年9月第1版　　**印　　次**：2016年9月第1次印刷
印　　数：1～2000
定　　价：49.50元

产品编号：068645-01

序

PREFACE

随着计算机软件变得越来越复杂，良好的设计越来越成为成功开发中的最关键因素。面向对象方法和技术提供了丰富的结构和机制，使设计者和开发者可以采用多种途径分解软件系统的复杂性，得到更好模块化的，易理解、易扩展和易维护的软件系统结构。如何把这方面的理论、方法和技术传播给本领域的入门新手，为未来软件和计算机领域的更大发展培养生力军，是计算机教育工作者和技术专家面临的一项重要挑战。廖湖声老师基于自己多年从事计算机科学技术教育，教授相关课程，以及使用面向对象技术（和 C++ 语言）开发软件系统的经验，撰写出本书，给我们提供了一个良好的范例。

在本书中，作者基于教育界、学术界和工业界对面向对象软件开发领域的经验总结，以及自己对相关领域中思想、方法和技术的深入理解，通过原理讨论、问题分析和实例程序开发过程的展示，给读者提供了大量有用信息。书中讨论采用了作者精心梳理的结构和组织方式，涵盖了非常宽泛的论题，从基本面向对象概念和基础知识开始，直到对设计模式等许多高级问题的深入讨论等，内容非常丰富。

在本书的各章里，作者都精心选择了一些重要概念和问题，首先对它们做了仔细的解释和分析，讨论了它们在思考和处理在软件结构设计中的重要作用、相关的情况和设计决策等。然后通过一个或几个有着或多或少实际意义的例子，阐释处理这些问题的基本方法和技术，为读者提供了良好的设计范例。在书中，作者还讨论了许多有助于在实际中分析和处理问题的线索，例如提出将对象分为实体型、管理型、加工型、事务型 4 类，为人们在设计所需的类时提供了有意义的思考方向。

当然，面向对象方法和技术，以及面向对象的编程技术，已经发展为一个内涵非常丰富的领域。任何一本书籍都已经不可能涵盖这个领域中的所有精华。但无论如何，本书为学过基本面向对象程序设计（和 C++ 语言）基础课程、希望进一步向软件工作者发展的人们提供了一块坚实的垫脚石，也是计算机教育中程序开发方面后续课程的一本优秀教材，是国内教育工作者在这方面的一个有益尝试。在审阅本书的过程中，我自己也得到了很多启发和新的认识，感受到了本书的魅力。我非常乐于看到，也期待着本书受到计算机领域大学生、专业教师和实际工作者的重视和欢迎。

裘宗燕

2016 年 2 月于北京

前言

PREFACE

在计算机科学与技术、软件工程等计算机类的专业教学中，面向对象技术作为主流的软件开发技术，已经成为必修的教学内容。高等院校中相当多专业的低年级教学计划中都设置了“C++ 程序设计”、“Java 程序设计”等面向对象程序设计课程，介绍程序设计语言和基本的程序设计方法，“数据结构”等核心课程的教学也普遍采用面向对象程序设计语言作为算法实现的工具。与此同时，在高年级的教学计划中也设置了“软件工程导论”、“软件设计与体系结构”等设计类课程，讲述面向对象的软件分析与设计方法。不仅如此，更多的专业选修课也普遍采用面向对象程序设计语言作为工具，介绍专用的软件开发方法，以及基于面向对象程序的软件工具。另外，在相关专业的研究生培养阶段，为了提高学生的软件开发能力，以“面向对象方法与技术”、“软件体系结构”、“软件开发方法”为代表的各种设计与实践课程也将面向对象方法与技术作为核心内容。

高校低年级的各种程序设计课程教学，侧重于算法与程序设计语言的学习，教科书中提供的程序设计案例普遍用于算法或语言功能的教学。然而，面向对象方法更多地用于程序结构的设计，现有的程序设计案例普遍规模偏小、结构简单，难以适应面向对象程序设计方法的教学需求。另外，高年级的各种软件工程课程的教学内容包含了面向对象的软件分析与设计等核心内容，其中软件构造和软件体系结构等内容的理解和掌握更需要面向对象程序设计经验的积累。

针对上述现象，作者总结了多年从事相关课程的教学积累和软件开发经验，从程序设计方法的角度重新组织了教学内容，在本书中引入了比较丰富的程序设计案例，尝试在低年级的程序设计教学和高年级的软件工程教学之间建立桥梁。对于计算机科学与技术、软件工程、信息安全、物联网、网络工程、数字媒体等专业，本书可以作为“面向对象程序设计”课程的教材。书中在讲解各种程序设计方法的同时，在各种程序案例的分析和介绍中，逐步介绍了 C++ 语言的主要语言功能。本书也可以作为“软件工程导论”、“软件设计与体系结构”、“Windows 程序设计”等本科课程的教学参考书，为这些课程的学习提供应用软件开发的设计案例。这些应用案例的设计内容已经覆盖了 UML 语言的主要功能和多数设计模式。另外，针对计算机相关专业的研究生教育，本书可以作为“面向对象方法与技术”、“软件体系结构”、“软件开发方法”等课程的教材。本书提供比较丰富的实用型程序设计案例，通过完整的设计过程讲解来介绍面向对象程序技术的应用方法，使研究生同学在复习程序设计语言和软件工程方法的同时，巩固和提高实用型软件开发的能力。

全书按照面向对象方法的内容组织，共 6 章：

第1章介绍面向对象技术的由来、程序设计方法和程序设计语言的发展、面向对象的软件开发方法和发展趋势。

第2章介绍对象的基本概念与封装性，举例介绍了实体型、管理型、加工型和事务型4种对象的设计方法。

第3章讨论对象之间的一般与特殊关系、整体与部分关系和对象关联的设计，基于二维图元对象、驾驶证管理、行车路线监测和学院班级管理系统4个设计案例，介绍了复杂对象的设计方法和MFC等可复用类库。

第4章介绍多态性的基本概念，通过数据库查询接口、广义表和通用的冒泡排序程序三个设计案例介绍了如何运用多态性来实现软件接口、异质容器和通用算法。

第5章基于网上书店、机票预订系统和XML语法分析三个设计案例，介绍了面向对象的软件建模和软件构造方法，以及若干UML图的使用方法。

第6章介绍常用的各种设计模式，提供了图形编辑程序和程序设计语言处理框架两个软件设计案例，介绍了组合模式、访问者、解释器、命令模式、抽象工厂等多种设计模式的具体应用。

本书具有以下特点：

(1) 突出面向对象设计方法的传授，而不是以程序设计语言为中心。全书按照程序设计方法的教学来组织，逐步介绍对象设计、对象关联、多态性、软件系统建模和设计模式等各种技术和方法，及其使用案例。

(2) 内容范围广，涵盖了面向对象程序设计方法、C++程序设计语言和常用的基本类库，也包含了初步的面向对象系统建模方法、UML语言和设计模式。

(3) 随着面向对象设计方法及其案例的讲解，由浅入深地介绍了C++语言的主要功能，不仅涵盖了基本的控制结构和数据结构，也包含了对象类、类模板等程序结构，而且介绍了微软基本类库MFC和标准模板库STL的核心内容。

(4) 提供大量设计案例，用于展示如何在应用系统设计中运用各种面向对象的设计方法。相当多的应用案例具有一定的规模，着重展示了多态性和设计模式的应用方法。

(5) 设计案例的讲解中侧重于设计思想、分析方法和设计方法，为多种应用程序的设计提供了从问题陈述、案例分析、解题思路、对象设计、算法设计、程序实现、测试案例到技术小结的完整叙述。

(6) 针对面向对象方法中对象确认等关键问题，提出了实体型、管理型、加工型和事务型4种对象的设计方法，有助于初学者掌握对象设计的方法。

本书在出版过程中，得到了北京大学裘宗燕教授的支持和帮助，还得到了清华大学出版社张玥编辑的大力支持，在此表示诚挚的感谢。

由于作者水平有限，书中难免有不妥和疏漏之处，恳请读者不吝赐教和批评指正。

编　者

目 录

CONTENTS

第1章 面向对象技术概述

1.1 面向对象技术的由来

面向对象技术属于软件工程技术，是人们为了应对“软件危机”(software crisis)而发展起来的一种软件开发技术。20世纪60年代以来，随着计算机硬件技术的发展和计算机网络的普及，计算机应用进入了越来越多的领域，软件开发的需求不断扩展，软件系统的规模不断扩大，软件系统的复杂性越来越高，从而出现了所谓的“软件危机”。

1.1.1 软件危机

所谓软件危机表现为：

(1) 软件产品难以满足用户需求。一方面，开发者和使用者对于功能需求缺少统一的认识，经常导致功能不完善、使用不便等问题。另一方面，软件的开发缺少严格的测试方法和技术，导致开发出的软件产品可靠性差。

(2) 软件开发周期难以预测。鉴于软件开发过程中，开发者需要不断确认客户的软件需求，修改设计方案；软件测试中也可能不断发现设计错误，不断地修改程序。这些现象都使得开发者难以准确地确定软件产品的开发时间，工期延误成为常态。

(3) 软件产品难以升级和维护。在软件使用过程中，应用需求仍然有可能发生变化。为了适应应用需求的变化，软件产品往往需要扩展某些功能，或者修改某些功能。然而，鉴于软件开发过程中，人们难以预测未来的变化，经常导致无法通过软件升级和软件维护来满足变化后的需求，不得不重新进行软件的开发。

(4) 软件生产率过低。软件开发过程中软件需求的反复确定、软件设计的反复修改以及程序代码的反复修改都会导致软件生产率的下降，需要投入更多的开发人员，从而使得软件开发成本难以控制，也使得软件开发难以满足各行各业对于计算机应用飞速发展的需求。

为了克服软件危机，人们开始针对软件开发、运行和维护的需求，研究系统、规范、可度量的工程化方法，逐渐形成了软件工程学科。软件工程的研究涉及软件开发技术和软件开发管理两个方面：前者试图改进软件开发方法、软件开发过程、软件开发工具与环境等，以求改善软件产品的质量，增强可维护性和可靠性，提高软件生产率，适应大型软件的开发需求；后者研究软件管理学和软件经济学，试图从工程管理、成本管理的角度来组织软件开发，控制软件开发的开销。面向对象技术就是一种十分有效、得到工业界广泛应用

的软件开发技术。

1.1.2 软件工程原则

为了降低软件开发的复杂性，增强软件产品的可维护性，各种软件开发技术的发展都符合下述软件工程原则。

1. 抽象(abstraction)

抽象是人们分析复杂问题的基本方法。这种方法要求软件的分析与设计采用分层抽象手段，提取事物的基本特征和行为，忽略具体细节，将复杂问题分解为多层相对简单的问题，可以改进软件的可理解性和管理负担。

2. 模块化(modularity)

分治法也是人们分析和解决复杂问题的基本方法。这种方法根据问题的功能组织和信息结构，将软件系统划分为相对独立的模块，使模块成为可组合、可分解、可更换的基本单元。在模块划分的过程中，尽可能减少模块之间的耦合度，尽可能提高模块的内聚性，使得单一模块的内部变化难以影响软件系统整体结构，从而增强软件产品的可维护性。

3. 信息隐蔽(information hiding)

信息隐蔽原则要求在软件设计中将信息处理方法和数据组织的细节隐蔽于模块的内部，使得软件升级和软件维护过程中所产生的处理逻辑变化和数据内部结构变化仅仅影响个别模块，不影响其他模块，从而加强软件系统的可维护性、减少模块之间的相互响应、改善软件的可靠性。

4. 局部化(localization)

面向软件设计中需要实现的每个计算，局部化原则要求将该计算所使用的所有数据资源集中在同一模块中，也就是要求模块设计中为所承担的计算就近提供足够的数据。局部化原则加强了模块的内聚性，也减少了模块之间的依赖关系，有助于软件可靠性的改善。

抽象、模块化、信息隐蔽和局部化等软件工程原则为软件开发技术的发展提供了方向。不同的软件开发方法为上述原则的实现提供了不同的支撑手段，从而为软件产品的可维护性、可理解性、可靠性的实现提供了不同程度的支持，也决定了这些软件开发技术能否被工业界所接受，能否得到广泛的应用。

面向对象技术作为一种软件开发技术，同样遵循上述的抽象、模块化、信息隐蔽和局部化原则。相对于其他软件开发技术，面向对象技术提供了不同的抽象方法和模块化方法，以及更丰富的信息隐蔽方法和局部化手段，具体方法如下。

(1) 面向对象技术提供了基于对象的抽象方法。

在软件需求分析中，将客观世界中的所有事物进行分类，抽象描述为对象。在软件设计中，将系统所有组成元素都看作对象，进行分类。在分类描述中，将描述事物特征和系

统元素属性作为对象的属性，将事物的行为和系统元素的功能作为对象的行为。按照这种基于对象的抽象方法，任何系统都可以被抽象为一组具有规定行为的对象，系统工作原理也可以被抽象为对象之间的协同工作。不仅如此，对象本身的功能被抽象为一组行为，允许采用不同的方法来实现同一组行为。

(2) 面向对象技术提供了基于对象的模块化方法。

鉴于客观世界中所有事物都被看作对象，事物之间的组合关系也就是对象的组合关系。软件系统的组织结构也就是对象的组织结构。软件系统中不同数据被划分到不同的模块，不同功能也被划分到不同的模块。不同的模块组成软件系统或者子系统，子系统组成整个软件系统。

(3) 面向对象技术提供了基于对象的信息隐蔽方法。

面向对象技术规定对象之间的交互作用必须通过消息传递来实现，也就是必须通过对象外部接口进行交互控制。对象内部功能的实现逻辑、对象内部的数据组织都被隐藏在对象接口的后面，对于外部对象是不可见的。对象的这种封装性是面向对象技术的基本特征之一，是实现信息隐蔽的主要手段。

(4) 面向对象技术提供了基于对象的局部化方法。

借助于面向对象技术提供的封装性特征，对象功能的算法实现必须在对象内部。按照面向对象技术所提供的局部化方法，计算所需要的数据只能来自于对象内部的属性数据，或者来自于其他对象发送过来的消息。计算过程中可以委托其他对象完成部分计算，但是完全独立于外部对象的数据组织和计算方法。

为了实现面向对象的软件开发，人们已经研制出了各种面向对象的程序设计语言。各种常见的面向对象程序设计语言，如 C++、Java、C# 等程序设计语言都提供相应的语言功能，能够支持上述基于对象的抽象、模块化、信息隐蔽和局部化方法。目前流行的许多程序设计语言也都号称能够支持面向对象程序设计，也能够支持上述方法的使用。

1.1.3 面向对象技术的特征

人们用于克服“软件危机”的另一种方法就是设法提高软件生产率。提高软件生产率的一个途径就是加强软件复用(reuse)，尽可能构造可以重复使用的软件模块，以避免重复的设计和程序实现。不同的软件开发技术提供了不同的软件复用手段。

面向对象技术针对客观事物之间存在的一般与特殊关系，允许开发者描述对象之间的继承关系，使得描述特殊对象的模块可以复用一般对象的属性和行为。鉴于一般对象具有通用性，这就使得大量软件开发工作都是基于一般对象来开发满足特定需求的特殊对象，从而避免了一般对象的重复开发。继承性也被认为是面向对象技术的基本特征之一。各种面向对象程序设计语言及其开发环境都针对常用软件功能，以基本类库的形式提供了大量程序模块；各种应用软件支撑环境也以对象类库的形式，提供了常用的应用模块，使得开发者能够基于面向对象技术的继承性，复用基本类库提供的功能，减少软件的开发强度。

面向对象技术的另一个基本特征是多态性，表现为对象之间通过消息传递来进行交互作用，而且消息响应取决于消息的接收者，而不是消息发送者。这种特征使得开发者在

算法设计中能够聚焦于计算逻辑的实现，使用其他对象提供的服务，而不必关心哪个对象提供了这些服务。因此，这些服务可以由多种对象提供，使得算法具备通用性。多态性的使用不仅使得通用算法的描述成为可能，而且降低了对象之间的耦合度。在软件维护和升级过程中，人们可以提供新的对象来提供相同的服务功能，而不必修改通用算法，从而避免了过多的软件模块受到影响。多态性的使用事实上提高了算法的复用性。不少程序设计语言已经借助于多态性提供了常用的算法。相当多的应用软件支撑环境也以基本类库的方式，提供了本类应用通用的应用框架，要求软件开发者按照应用框架给出的通用程序结构来进行对象设计、组织和创建应用软件。

在实用软件开发中，根据基本类库提供的对象类和应用框架，借助于面向对象技术的继承性和多态性，开发者已经能够通过特殊对象设计、对象组装和配置来完成大多数软件模块的设计与实现。

由此可见，面向对象技术的几个特征：抽象性、封装性、继承性和多态性都是为了解决软件危机问题所设计的。基于对象的抽象为问题分解提供了分类描述的手段，通过事物概要的抽象描述，将客观世界的事物转化到计算机世界的系统模块。封装性将对象的属性和行为包装成独立的模块，仅允许通过接口与外部交互，通过模块功能实现的局部化实现了信息隐蔽。继承性按照事物的一般特殊关系，使得一般事物的描述能够独立于特殊对象，从而支持模块重用。多态性支持通用算法和应用框架的重用，进一步降低了对象之间的耦合性。基于对象的抽象方法事实上就是人们认识客观世界的分类方法，因此面向对象技术所产生的软件需求和系统设计更加易于理解。封装性、继承性和多态性等基本特征说明该技术具有强大的抽象描述能力，保证了软件模块的独立性，在软件设计和程序设计层面加强了软件的可维护性和可靠性。

面向对象技术最早起源于面向对象程序设计，随后逐步扩展到面向对象分析、面向对象设计和面向对象测试，形成了完整的一套软件开发方法。以下分别介绍面向对象程序设计方法、支持面向对象程序设计的程序设计语言，以及面向对象的软件开发方法。

1.2 程序设计方法的发展

在软件开发技术的发展中，人们最早关注程序设计方法的改进，力图通过改进数据结构和程序结构的组织方法来提高程序的可读性、改善程序的可维护性。众所周知，最初的计算机程序如同机器指令和汇编语言描述的程序，完全由一组顺序执行的、过程化的指令序列组成。这种程序的组织方式完全面向计算机硬件系统的冯·诺依曼(John von Neumann)体系结构，丝毫不顾及程序设计方法的使用。随着计算机应用的普及，软件复杂性问题逐渐显现，出现了最早的程序设计方法——结构化程序设计。

1.2.1 结构化程序设计

结构化程序设计(structured programming)的主要思想是采用功能分解、自顶向下、逐步求精的方法来进行程序设计。首先，针对程序设计要实现的整体功能，分析问题的内部组织，将复杂问题划分为若干个相对简单的子问题，为每个子问题的解决设置一个程序

模块。如果某个子问题仍然足够复杂，则采用相同的方法，继续进行功能分解，形成更多的子问题，设置更多的程序模块，直到分解出的子问题已经很简单，采用一个独立的函数（或子例程）即可实现时为止。这种设计方法保证每个模块具有独立的功能目标，整个系统就是多层模块的组合。这种模块化的系统易于理解，而且后期维护中程序功能的变化仅影响负责实现该功能的程序模块，对其他模块影响较少，使得程序可维护性得到改善。

另外，在函数内部的程序设计中，结构化程序设计方法要求仅允许使用三种控制结构，也就是图 1-1 中所示的顺序结构、选择结构和重复结构。这三种控制结构的共同特征就是只有一个入口和一个出口。因此，仅依靠这三种控制结构组成的任何一个程序模块都只有一个入口和一个出口。按照这样的程序结构，后期的程序维护中每个模块的内部修改将不会影响到其他模块的控制逻辑，从而改善了程序的可维护性。

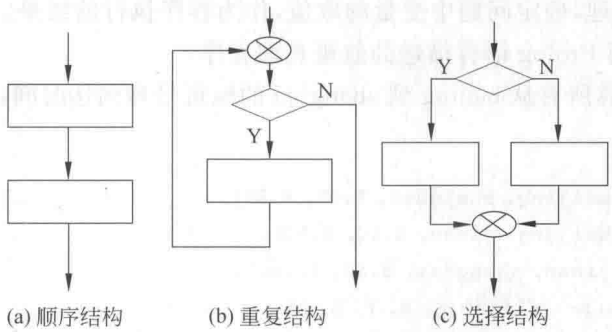


图 1-1 三种基本结构的流程图

结构化程序设计方法在工业界得到了普遍的应用，在相当长的时间内占据了程序设计的主导地位。然而，随着软件复杂性的提高，结构化程序设计方法的弱点也逐渐显现出来。

结构化程序设计方法的缺点在于仅仅考虑了功能的分层组织结构，而没有考虑数据的组织结构，没有考虑数据与功能之间的关系。事实上，任何计算机程序的用途离不开数据信息的处理。计算机程序的每个功能都是某些数据的处理功能。因此，程序所处理的数据和所实现的功能具有必然的联系。从上述原理可知，结构化程序设计完全忽略了数据和功能之间的这种关系，使得针对同一组数据的计算可能分布在不同的模块。因此，在程序设计中负责开发这些模块的每个开发者都必须了解该数据的语义和组织。在程序维护中如果数据内容和组织结构发生变化时，所有引用和创建这些数据的程序模块都会受到影响。鉴于上述缺点，人们开始探讨更好的程序设计方法。

1.2.2 逻辑型程序设计和函数式程序设计

结构化程序设计方法显然是针对过程化程序提出的。过程化程序的特点就是符合冯·诺依曼体系结构，顺序地执行程序代码；在执行中，访问或修改变量所表示的存储器内容，完成规定的计算。鉴于程序所实现的计算问题本身并没有过程化实现的要求，程序的执行逻辑中不仅包括了规定的计算逻辑，也包括了确保程序顺序执行的控制逻辑。因此，人们认为软件维护困难的根本原因之一在于这种过程化程序的程序设计范型

(programming paradigms),试图从根本上改变程序设计的方式,相继提出了逻辑型程序设计(logic programming)和函数式程序设计(functional programming)等程序设计方法。这些方法的共同特征就是仅仅要求设计者提供计算逻辑的描述,而所有控制逻辑完全由系统提供。本节简要介绍这两种程序设计方法。

1. 逻辑型程序设计方法

逻辑型程序设计方法将计算抽象为基于因果关系的推理过程,程序设计中普遍采用霍恩子句(HORN 子句)来描述计算。HORN 子句以前提和结论的形式提供一阶谓词演算功能。逻辑型程序设计的基本方法就是将计算逻辑分解为若干个谓词,利用 HORN 子句描述事实和目标之间的因果关系。针对程序中提出的问题,系统将按照给定的 HORN 子句进行推理,确定问题中变量的取值,作为程序执行的结果。

【例 1-1】 采用 Prolog 语言描述的航班查询程序。

该程序用于计算所有从 beijing 到 shanghai 的航班号和到达时间,并且允许中途转机一次。

```
flight(CA509, beijing, shanghai, 8.00, 9.30).
flight(CA510, beijing, jinan, 8.10, 8.50).
flight(CA511, jinan, shanghai, 9.40, 10.40).
goto(X, Y, N, A):-  flight(N, X, Y, D, A).
goto(X, Y, N, A):-  flight(N, X, Z, _, A1), flight(_, Z, Y, D2, A), D2>=A1+1.
?-goto(beijing, shanghai, N, A).
```

程序的计算结果是

```
N=CA509, D=9.30
N=CA510, D=10.40
```

程序中谓词 flight 代表航班信息,给出了航班号、起点终点以及起飞降落时间;flight 的三个定义给出了三条航班信息。谓词 goto 用于计算从起点 X 到终点 Y 的航班号码 N 和到达时间 A;goto 的定义给出了两条计算规则。第一条规则说明存在直飞航班时,可以直接获得结果;第二条说明对于以 X 为起点的任意航班和以 Y 为终点的任意航班,如果前者的终点 Z 与后者的起点相同,而且后者的起飞时间比前者的到达时间晚 1 小时以上,则返回前者的航班号 N 和后者的到达时间 A。程序中的符号"? -"表示问题,询问从 beijing 到 shanghai 的航班 N 和到达时间 D。

由此可见,逻辑型程序设计中仅仅描述了计算逻辑,不关心程序执行顺序,也没有执行指令。程序中只有变量的绑定,没有变量存储的概念,不存在赋值语句;没有循环语句,重复计算完全依靠递归来实现。整个程序描述都是数学逻辑,易于理解,可靠性也很强。

2. 函数式程序设计方法

函数式程序设计的基本观点就是将任何计算功能都抽象为函数的组合,其基本方法是采用表达式、自定义函数、内置函数来描述计算逻辑;其主要特征在于支持引用透明性,

也就是对于程序中的表达式、函数等任何片段，不论何时进行求值，只要给定的参数值相同，则必然得到相同的结果。在函数式程序设计中，没有赋值语句，函数是程序的基本模块，整个程序由函数调用表达式组合而成，循环处理完全依靠函数的递归调用实现，程序求值的结果就是计算结果。同时，函数式程序设计将函数看作一种特殊的数据对象，可以作为其他函数的参数，也可以作为函数的返回值。这些数学性质使得函数式程序非常简洁、易于进行分析和变换，经常用于程序设计语言语义的推理和验证。

【例 1-2】 Scheme 语言描述的斐波那契数列计算程序。

该程序用于对给定整数 n 计算斐波那契数列 fib(0)、fib(1)、…、fib(n)中所有偶数组成的序列，计算结果为(0 2 8 34 …)。

```
(define(even-fibs n)
  (define(next k)
    (if   (>k n)
          empty
          (let   ((v(fib k)))
            (if(even?v)(cons v(next(+k 1)))
              (next(+k 1)) ))))
  (next 0))
```

程序中定义了两个函数 even-fibs 和 next，使用了内置函数 fib、cons 和 even，分别用于求斐波那契数 fib(k)、构造序列和判断参数 v 是否为偶数。函数 next 是一个局部函数，定义在函数 even-fibs 内部，用于计算 fib(k)开始的偶数序列。let 表达式提供了局部变量绑定的功能。这里 fib(k)的计算结果被绑定到变量 v。next 表示的计算逻辑就是判断 v 是否为偶数，条件成立则使用 v 构造输出结果序列，否则通过 next 的递归调用直接计算后续的斐波那契偶数序列。

由此可见，函数式程序设计完全不使用赋值语句，仅仅通过函数定义、函数调用、序列运算、let 变量绑定和 if 表达式等少数语言功能来进行程序描述。这种函数式程序文法简单，程序易于理解，数学性质好，程序可靠性高。

逻辑型程序设计和函数式程序设计的共同特征是程序描述简洁，数学性质好，可靠性高。这些程序设计方法在人工智能、计算机语言处理等领域得到了广泛的应用。然而，这些程序设计方法并没有得到工业界的广泛接受。究其原因主要有两点：一是此类程序的执行效率低下，内存开销大，早期的计算机硬件系统性能难以满足要求。近年来，计算机硬件技术发展已经基本上克服了这个问题，能够满足逻辑型程序设计和函数式程序设计的需求。另一个原因在于逻辑型程序设计和函数式程序设计都未提供循环结构，而是采用递归处理的方式描述迭代处理等计算要求。然而，递归程序设计的难度较大，相当多的程序设计者能力有限，难以承担这种开发任务。程序设计者技能增长的速度总是赶不上软件开发需求的增长速度。

1.2.3 抽象数据类型和面向对象程序设计

针对结构化程序设计方法仅仅考虑功能分解的弱点，人们开始关注数据和数据处理