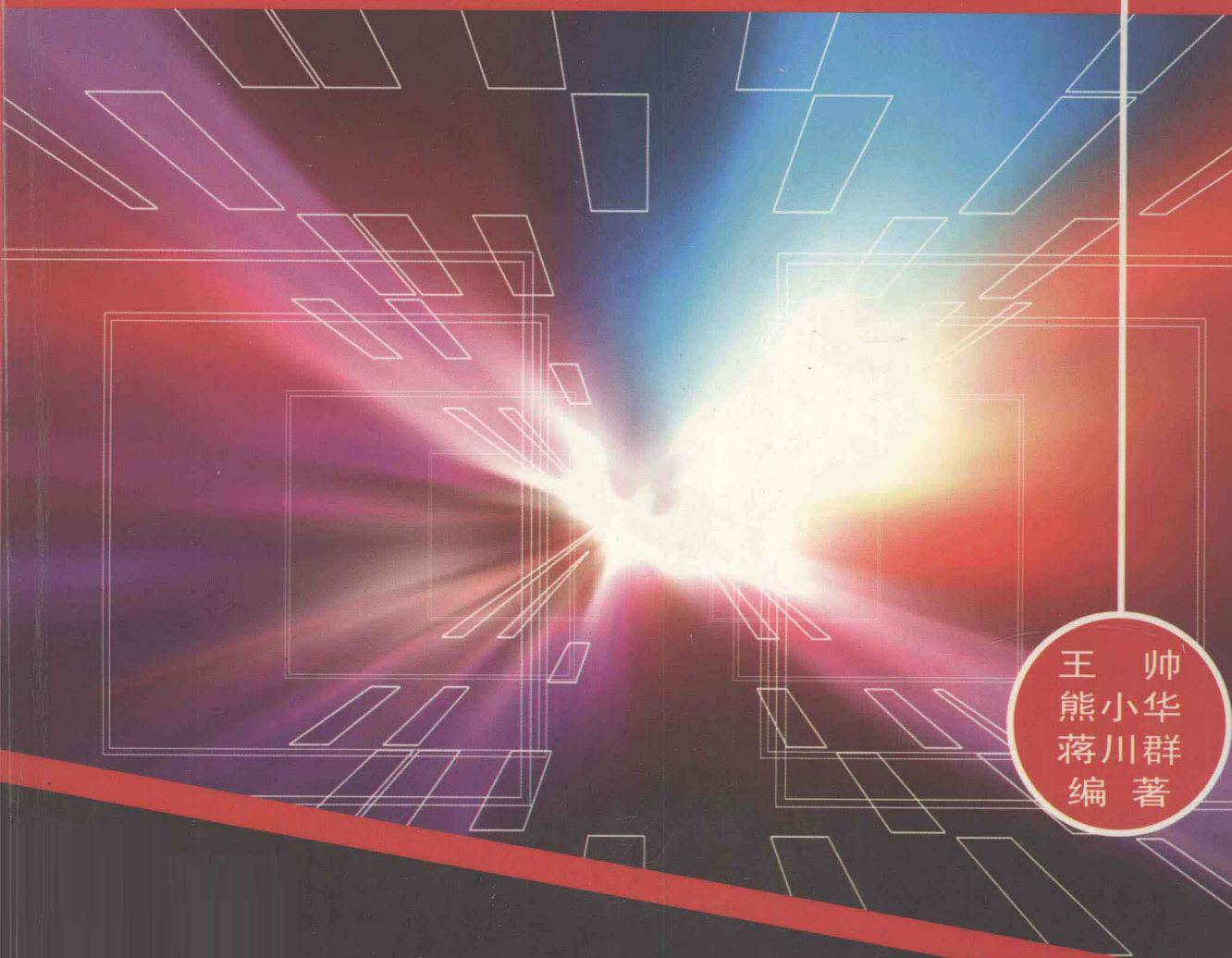
计算机应用与软件技术专业领域技能型紧缺人才培养培训系列教材

面向对象程序设计（C++）



王　帅
熊小华
蒋川群
编　著

高等教育出版社
HIGHER EDUCATION PRESS

计算机应用与软件技术专业领域技能型紧缺人才培养培训系列教材

面向对象程序设计(C++)

王　帅　熊小华　蒋川群　编著

高等教育出版社

内 容 提 要

本书是计算机应用与软件技术专业领域技能型紧缺人才培养培训系列教材之一，其主要内容包括类、对象、消息、封装、继承、重载、多态等面向对象程序设计的基本内容，以及 Windows 环境下面向对象编程技术等。本书的特色是通过大量的实例由浅入深地对面向对象程序设计的概念、方法和技术进行了系统的介绍，突出了实践技能的培养。教材的最后安排了配套实验，针对每节的内容提出问题，让读者进一步巩固所学的理论和方法。

本书适合作为各类高等职业技术学校、部分普通高等院校培养计算机应用与软件技术专业应用型人才的教材，也可作为程序开发和设计者的参考用书。

图书在版编目（CIP）数据

面向对象程序设计：C++／王帅，熊小华，蒋川群主编.—北京：高等教育出版社，2004.7
ISBN 7-04-014848-X

Ⅰ.面… Ⅱ.①王…②熊…③蒋… Ⅲ.C语言—程序设计—高等学校：技术学校—教材 Ⅳ.TP312

中国版本图书馆 CIP 数据核字（2004）第 063967 号

责任编辑 张尽琳　**封面设计** 吴　昊　**责任印制** 潘文瑞

书　　名	面向对象程序设计(C++)		
编　　著	王帅　熊小华　蒋川群		
出版发行	高等教育出版社	**购书热线**	010-64054588
社　　址	北京市西城区德外大街 4 号		021-56964871
邮政编码	100011	**免费咨询**	800-810-0598
总　　机	010-82028899	**网　　址**	http://www.hep.edu.cn
传　　真	021-56965341		http://www.hep.com.cn
			http://www.hepsh.com
排　　版	南京理工排版校对有限公司		
印　　刷	上海新华印刷有限公司		
开　　本	787×1092　1/16	**版　　次**	2004 年 7 月第 1 版
印　　张	15	**印　　次**	2004 年 7 月第 1 次
字　　数	349 000	**定　　价**	21.00 元

凡购买高等教育出版社图书，如有缺页、倒页、脱页等质量问题，请在所购图书销售部门联系调换。

出版说明

为实现党的十六大提出的全面建设小康社会的奋斗目标，落实《国务院关于大力推进职业教育改革与发展的决定》，促进职业教育更好地适应社会主义现代化建设对生产、服务第一线技能型人才的需要，缓解劳动力市场上制造业和现代服务业技能型人才紧缺状况，教育部、劳动和社会保障部、国防科工委、信息产业部、交通部、卫生部决定组织实施“职业院校制造业和现代服务业技能型紧缺人才培养培训工程”（教职成[2003]5号，以下简称《工程》）。《工程》的目标是：“根据劳动力市场技能型人才的紧缺状况和相关行业人力资源需求预测，在数控技术应用、计算机应用与软件技术、汽车运用与维修、护理等四个专业领域，全国选择确定500多所职业院校作为技能型紧缺人才示范性培养培训基地；建立校企合作进行人才培养的新模式，有效加强相关职业院校与企事业单位的合作，不断加强基地建设，扩大基地培养培训能力，缓解劳动力市场上技能型人才的紧缺状况；发挥技能型紧缺人才培养培训基地在探索新的培养培训模式、优化教学与训练过程等方面的示范作用，提高职业教育对社会和企业需求的反应能力，促进整个职业教育事业的改革与发展。”

《工程》实施启动以来，各有关职业院校在职业教育人才培养目标、人才培养模式以及专业设置、课程改革等方面做了大量的研究、探索和实践，取得了不少成果。为使这些研究成果能够得以固化并更好地推广，从而总体上提高职业教育人才培养的质量，我们组织了有关职业院校进行了多次研讨，根据“教育部办公厅、信息产业部办公厅关于确定职业院校开展计算机应用与软件技术专业领域技能型紧缺人才培养培训工作的通知”（教职成厅[2003]5号）中的两年制高等职业教育计算机应用与软件技术专业领域技能型紧缺人才培养指导方案，确立了“以就业为导向，以企业需求为依据”的宗旨，“以综合职业素质为基础，以能力为本位”的思路，“适应行业技术发展，以应用为目的”的体系，“以学生为主体，体现教学组织的科学性和灵活性”的风格，组织编写了一批“计算机应用与软件技术专业领域技能型紧缺人才培养培训系列教材”。这些教材结合《工程》的指导思想与目标任务，反映了最新的教学改革方向，很值得广大职业院校借鉴。

此系列教材出版后，我们还将不定期地举行相关课程的研讨与培训活动，并联合一些软件企业共同探讨人才培养目标、人才培养模式以及专业设置、课程改革，为各院校提供一个加强校企合作、交流的互动平台。

“计算机应用与软件技术专业领域技能型紧缺人才培养培训系列教材”适合高等职业学校、高等专科学校、成人高校及本科院校举办的二级职业技术学院、继续教育学院和民办高校使用。

高等教育出版社

2004年6月

前　言

本书是根据教育部等部委组织实施的“职业院校制造业和现代服务业技能型紧缺人才培养培训工程”中有关计算机应用与软件技术专业领域技能型紧缺人才培养指导方案的精神，按照高等职业技术教育技能型人才的培养目标和基本要求编写的“计算机应用与软件技术专业领域技能型紧缺人才培养培训系列教材”之一。

面向对象程序设计(OOP)是一种围绕真实世界的概念来组织模型的程序设计方法，从所处理的数据入手，抽象出对象的概念，以对象为中心而不是以服务(功能)为中心来描述系统。它把编程问题视为一个数据集合，数据相对于功能而言，具有更强的稳定性。与面向过程程序设计方法相比，面向对象程序设计方法以对象为核心，更符合人类的思维习惯，而且具有可重用性好、可维护性好、易扩充、易修改等优点。

在现代软件开发中，理解面向对象程序设计的思想和方法越来越重要。因此，掌握面向对象的基本思想和方法，是软件设计和开发的重要手段，也是应用型人才所必须掌握的一种技术。

本书编写的主要目的是通过C++语言来介绍面向对象程序设计的基本思想和方法，使读者能够理解面向对象程序设计技术并应用它来解决实际问题。书中结合大量的具体实例阐明了面向对象程序设计中的重要概念和设计方法。

本书的特色是通过大量的实例由浅入深地对面向对象程序设计的概念、方法和技术进行了系统的介绍，突出了实践技能的培养。

在内容组织和结构安排上本书也具有自己的特色。全书可分为两大部分，第一部分包括前4章，介绍面向对象程序设计的基本内容；第二部分包括5、6两章，介绍Windows环境下面向对象的编程技术。其中第1章首先对面向对象程序设计进行了概述，然后在第2章中通过一个完整范例介绍了面向对象中的重要基本概念和技术，包括类、对象、消息、构造函数、析构函数、封装、重载、继承、虚函数和多态等，使读者能够很快对面向对象技术的全貌有一个大概的了解和体会。第3章在第2章的基础上，对面向对象程序设计进行了更全面和细致的讲解，介绍了this指针、静态成员、多继承、虚基类和抽象类等较高一级的概念。第4章讲解了在实际应用中常用的通用程序设计技术，包括模板函数、模板类、容器类等内容。第5章通过学生管理和绘图程序讨论了MFC类库中的常用类。第6章给出了两个案例的面向对象设计和详细实现：人工生命游戏的模拟程序和简易画笔程序。

另外，本书的最后(第7章)安排了配套实验，针对每章的内容提出问题，让读者进一步巩固所学的理论和方法。希望读者在学习过程中，认真对待这些实验，以达到巩固和消化所学知识的目的。

本书由王帅、熊小华、蒋川群共同编写。另外，在本书的编写过程中，受到上海第二工业大学有关领导和教师的大力支持，并对本书提出了很多宝贵意见，在此一并表示诚挚的谢意。

前　言

本书适合作为各类高等职业技术学校、普通高等院校培养计算机及相关专业应用型人才的教材，也可以作为程序开发和设计者学习面向对象技术的参考用书。

由于编者水平有限，加之时间仓促，书中错误和遗漏在所难免，恳请读者批评指正。

编　者

2004 年 6 月

目　录

第1章 面向对象程序设计概述

内容提要与学习要求

面向对象(Object-Oriented)技术是目前流行的系统设计开发技术,它包括面向对象分析(Object-Oriented Analysis, OOA)和面向对象程序设计(Object-Oriented Programming, OOP)。面向对象程序设计技术的提出,主要是为了解决传统程序设计方法——结构化程序设计所不能解决的代码重用问题。

本章将简单介绍一下面向对象思想和面向对象程序设计的基本内容,包括:

- 面向对象思想的由来;
- 对象的基本特征;
- 面向过程和面向对象程序设计方法的联系和不同;
- 面向对象程序设计语言的简介;
- 面向对象建模。

要求掌握对象的概念和特点,理解面向过程和面向对象程序设计方法的联系和不同,了解面向对象语言的发展概况和面向对象建模方法。

1.1 面向对象思想的由来

长久以来,人们一直面对这样一个不合理的现象,即:认识一个系统的过程和方法同用于分析、设计和实现一个系统的过程和方法不很一致。

对一个系统的认识是一个渐进过程,是在继承了以往的有关知识的基础上多次迭代往复而逐步深化的。在这种认识的深化过程中,既包括了从一般到特殊的演绎,也包括了从特殊到一般的归纳。而目前用于分析、设计和实现一个系统的过程和方法大多数是"瀑布"型的,即后一步是实现前一步所提出的需求,或者是进一步发展前一步所得出的结果。因此越接近系统设计(或实现)的后期,如果要对系统设计(或实现)的前期的结果进行修改就越困难。同时也只有在系统设计的后期才能发现在前期所造成的一些差错。当系统越大、问题越复杂时,由于这种对系统的认识过程和对系统的设计(或实现)过程不一致所引起的困扰也就越大。

当一个有数十年实践经验的工程师回首往事时,经常把自己以往所完成的一些工程称为"遗憾工程"。这是因为当系统设计开始时(概念设计阶段),用户对系统的需求是比较笼统和抽象的,而那时设计者的设计"自由度"比较大,设计者可选用当时已有的原理、方法、工具、环境、构件和器件去设计系统。但随着时间的推进,用户对系统的需求越来越细致和具体了。但设计者对系统进行修改的"自由度"却越来越小了。还应看到,随着时间的推进,原理、方法、工具、环境、构件和器件的水平是日益提高的,而设计者采用新技术的"自由度"却受到了限制。因此,系统越大、周期越长,被称为"遗

憾工程”的可能性就越大。

为了解决上述不合理的现象，就应该使分析、设计和实现一个系统的方法尽可能地接近认识一个系统的方法。也就是说，应该使描述问题的问题空间和解决问题的方法空间在结构上尽可能地一致，使分析、设计和实现系统的方法学原理与认识客观世界的过程尽可能地一致。这就是面向对象(Object-Oriented)方法学的出发点和所追求的基本原则。

和人们认识世界的规律一样，面向对象的方法学认为：客观世界是由许多各种各样的对象(Object)所组成的，而每一个对象都有两个特征：状态(也称为属性)与行为(也称为方法)。不同对象间的相互作用和联系就构成了各种不同的系统，构成了客观世界。说得形象一些，每个组成世界的对象都是通过自己的行为来变化自身的状态，一切变化都是对象自身或对象间的协调而产生的。“面向对象”方法学其实包括：“面向对象的系统分析”、“面向对象的系统设计”、“面向对象的程序设计”、……，而在这里则主要探讨“面向对象的程序设计”。

当设计和实现一个客观系统时，如果能在满足需求的条件下把系统设计成是由一些不可变的(相对固定的)部分所组成的最小集合，那么就认为这个设计是优秀的，而这些不可变的(相对固定的)部分就被看成是一些不同的对象。

“对象”至少应该具有以下特征：

(1) 模块性。一个对象是一个可以独立存在的实体。从外部来看这个模块，只了解这个模块具有哪些功能，至于这个模块的内部状态以及如何实现这些功能的细节都是“隐藏”在模块内部的。一个模块的内部状态是不受(或很少受到)外界的影响的。同时，一个模块的内部状态的改变也不会影响到其他模块的内部状态。因此，各模块间的依赖性是很小的。所以各种模块才有可能较为独立地为各个系统所选用。

(2) 继承性和类比性。人们是通过对客观世界中的各种对象进行分类及合并等方法而来认识世界的，每个具体的对象都在它所属的某一类对象(类)的层次结构中占据一定的位置。因此，下一层次的对象应具有上一层次对象的某些属性，在面向对象方法学中把它称为是下一层次的对象继承了上一层次的对象的某些属性。另一方面，当人们发现一些不同的对象具有某些相同的属性时，也常常把它们归并成一个类，在面向对象方法学中把它称为是通过对象间的类别而实现了归类。

(3) 主动性和动态连接性。对象与传统的数据有本质的区别，它不是被动地等待外界对它施加操作。相反，它是进行处理的主体，必须发消息请求对象执行它的某个操作，处理它的私有数据，而不能从外界直接对对象的私有数据进行操作。在客观世界中，由于存在各式各样的对象以及它们之间的相互连接和作用，从而构成了各种不同的系统。因此把对象和对象间所具有的一种统一、方便、动态地连接和传递消息的能力与机制称为动态连接性。

(4) 易维护性。任何一个对象都是把如何实现本对象功能的细节隐藏在该对象的内部。因此，无论是完善本对象的功能，还是改正功能实现的细节，都被局限于该对象的内部，而不会传播给外部，这就增强了对象和整个系统的易维护性。

20 世纪 70 年代末，面向对象方法学的一些基本概念就已经在系统工程领域萌发出来了，例如对于系统中的某个模块或构件可表示为问题空间的一个对象或一类对象。一些系统分析和设计人员已从实践中归纳出一系列符合面向对象方法学原理的一些分析及设计的

步骤，只是尚未得到充分和有效的软件工具的支持。

到了20世纪80年代，面向对象的程序设计方法得到了很快的发展，并显示出其强大的生命力。美国Byte杂志在1983年和1986年以专刊的形式发表了一系列介绍面向对象程序设计方法的文章，1986年在美国举办的第一届面向对象程序设计语言、系统和应用(OOPSLA '86)的国际会议，使面向对象受到世人的重视。到目前为止，从所发表的论文中可以看出，许多研究者都在期望使用面向对象的方法和技术来解决各种问题。面向对象的方法已被广泛地应用于程序设计语言、设计方法学、系统工程、操作系统、分布式系统、人工智能、实时系统、数据库、人机接口，甚至硬件设计。这种情况的出现有许多原因，动力之一是工作站技术的快速发展。现代的工作站能够支持复杂的个人程序设计环境，而且提供了基于图形的人机界面。二进制大对象(例如存储声音和图像数据等的对象)、可视编程、分布式处理和用户对控制应用程序应做什么等的需求和发展，都是面向对象计算发展的动因。

人们普遍认为，在更高的层次上、更广泛的领域内开展对面向对象方法和技术的研究将是下一阶段的一个研究热点。

1.2 面向对象程序设计与面向过程程序设计的比较

1.2.1 面向过程程序设计

面向过程程序设计方法是广为大家了解的一种程序设计方法。像C、Basic、FORTRAN、Pascal等大多数程序设计语言都属于这种面向过程的程序设计语言。在这种程序设计方法中，语言基本上是命令式的。命令是某个程序段，执行这个程序段的目的是为了更新对象，命令执行的顺序已在程序中预先作了规定。也就是说，使用过程性语言编写的程序，其功能是隐含在程序代码中的，为了搞清楚程序的功能，必须反复阅读程序，仔细分析程序的每个语句，根据该程序设计语言的语法确定语句的执行顺序，并要综合每个语句的语义及执行顺序才能推断出程序的功能。因此，理解面向过程的程序相当困难。

显然当应用系统的功能比较复杂时，应用程序的规模必然十分庞大，包含的语句很多，程序元素(数据、语句)之间的相互关系十分复杂。因此用面向过程的程序设计方法开发应用系统时，需要耗费大量人力物力，只有经过严格训练的有经验的程序员才能胜任编程工作。这样的应用系统不仅不易开发，维护起来也十分困难。所谓维护，就是在软件交付给用户使用期间，出于种种原因而对软件进行修改。

人们在开发软件的长期实践过程中，总结出一些设计原理并研究出一些系统化的技术方法，把它们用于面向过程程序设计，能够提高开发效率，增加系统的可理解性和可维护性。这些原理和技术方法，在进行面向对象程序设计时也有借鉴意义。下面简要介绍主要的设计原理和技术方法。

1. 信息隐藏

随着程序规模的增大，数据的组织成为一个重要的问题，1972 年，Parnas 提出了著名的信息隐藏原理。其基本思想是：把需求和求解的方法相分离，把相关信息——数据结构和算法集中在一个模块中，和其他模块隔离，其他模块不能随便访问这个模块的内部信息，只能通过严格定义的接口进行。在信息隐藏原理的指导下，产生了模块化程序设计。例如 Modula-2 语言就属于模块化程序设计语言。在这种程序设计方法中，程序设计的首要问题是划分模块，数据结构被隐藏在模块中。每个模块都有一个接口，模块的表示只能通过接口进行访问。

2. 模块化

模块是数据说明、可执行的语句等程序元素的集合，它是单独命名的，而且可以通过名字来访问，也就是说，可以用名字代表该模块。所谓模块化，就是把一个程序化分成若干个模块，每个模块完成一个子功能，把这些模块组装成一个整体，可以完成指定的功能。

模块化是为了使一个复杂的大型程序能被人更好地管理和更好地理解。如果一个大型程序只有一个模块组成，例如程序有几十万行语句一个接着一个罗列在一起，那么很难理解和维护，而如果把复杂的问题分解成许多容易解决的小问题，各个击破，则原来的问题也就容易解决了，这就是模块化的依据。但模块数不能无限制地增加。因为随着模块数目增加，每个模块的规模将减小，开发单个模块的工作量确实减少了，但设计和实现模块接口所需要的工作量也将增加。所以，在把程序划分成模块的时候，模块规模应该适当。模块太大则模块化带来的好处不明显；若模块太小，则模块间接口的成本过大。

采用模块化原理可以使程序结构清晰，不仅容易设计也容易阅读和理解。因为程序错误通常局限在有关的模块及它们之间的接口中，所以模块化能够提高软件的可维护性。模块化也有助于软件开发工程的组织管理，一个复杂的大型程序可以由许多程序员分工编写不同的模块。

3. 逐步求精

由于人类思维能力的限制，如果每次面临的因素太多，就不可能得到精确的思维。处理复杂系统的惟一有效方法就是用层次的方式构造和分析它。一个复杂的动态系统首先可以用一些高级的抽象概念构造和理解，这些高级概念又可以用一些较低级的概念构造和理解，如此进行下去，直到最低层次的具体元素。随着程序开发工作的进展，在程序结构每个层次中的模块表示了对程序抽象层次的一次精化。处于程序结构顶层的模块控制了程序的全部功能并影响全局；在程序结构底层的模块，完成对数据的一个具体处理。用自顶向下、逐步求精的方式分配控制，简化了程序的设计和实现，提高了程序的可理解性、可修改性和可测试性，使程序更容易维护。

关于逐步求精方法，N. Wirth 曾做过如下说明：对付复杂问题的最重要的办法是抽象。因此，对一个复杂的问题不应该立刻用计算机指令、数字和逻辑符号来表示，而应该用较自然的抽象语句来表示，从而得出抽象程序。抽象程序对抽象的数据进行某些特定的运算并

用某些合适的记号(可能是自然语言)来表示。对抽象程序作进一步的分解,并进入下一个抽象层次,这样的精细化过程一直进行下去,直到程序能被计算机接受为止。这使得程序可能是用某种高级语言或机器指令书写的。

4. 结构程序设计技术

在把一个程序划分成若干个模块并且确定了模块之间的关系以后,进一步的工作就是设计完成每个模块功能的处理过程。如何才能设计逻辑正确、性能高、易理解、易阅读的程序呢?结构程序设计技术是实现这一目标的关键技术。

结构程序设计的概念最早由 Dijkstra 提出。1965 年他在一次会议上指出:"可以从高级语言中取消 GOTO 语句"。1966 年,C. Bohm 和 G. Jacopini 证明了只用 3 种基本的控制结构就能实现任何单入口单出口的程序。这 3 种基本的控制结构是"顺序"、"循环"和"选择"控制结构。

那么什么是结构程序设计呢?目前还没有一个普遍接受的定义,比较流行的定义是:结构程序设计是一种设计程序的技术,它采用自顶向下逐步求精的设计方法和单入口单出口的控制结构。

使用结构程序设计技术设计程序的主要好处如下:

(1) 自顶向下逐步求精的方法适合人类解决复杂问题的普遍规律,因此可以显著提高开发效率。

(2) 用先全局后局部、先整体后细节、先抽象后具体的逐步求精过程开发出的程序有清晰的层次结构,因此容易阅读和理解。

(3) 不使用 GOTO 语句,只使用单入口单出口的控制结构,使得程序的静态结构和它的动态执行情况比较一致。因此程序容易阅读和理解,开发时也较容易保证程序的正确性,即使出现错误也比较容易诊断和纠正。

(4) 控制结构有确定的逻辑模式,编写程序代码只限于使用很少几种直截了当的方式,因此源程序清晰流畅,易读易懂而且容易测试。

为了能够让大家理解"面向过程"的程序设计,在此就用"伪代码"写一段程序(在本例中,主要是用于讲解面向过程的基本思考方式,而不考虑本程序是否有现实作用)。

伪代码,是指不能够直接编译运行的程序代码,它是用于语法结构讲解的一个工具,它比真正的程序代码更简明,更贴近自然语言。

一个"面向过程"的程序示例:

```
铅笔数=5
钢笔数=6
圆珠笔数=4
其他笔数=7
  ⋮
//销售程序段
铅笔数 = 铅笔数 - 1// 卖出了 1 支铅笔
```

```
其他笔数 = 其他笔数 - 2 // 卖出了 2 支其他笔
// 采购程序段
钢笔数 = 钢笔数 + 20// 新进了 20 支钢笔
圆珠笔数 = 圆珠笔数 + 10// 新进了 10 支圆珠笔
……
```

正如上面的程序段所示，在面向过程的程序中通常是：

(1) 定义变量及其初始值。

(2) 根据事件发展顺序，进行相对应的处理。

在上面的程序中，先定义了各种笔的初始总数，然后这些数量会在销售、采购时改变。这种程序的编写思路是随着事件，按照一定的过程来进行的。这种方法是一种“解题”的思维方式，把所有的需求当做一个“应用题”一样来“解答”。

传统的“面向过程”的方法学是把世界分成两个部分分别认知：

(1) 数据(Data)：用于描述各种状态的数据结构。

(2) 过程(Procedures)：就是操作这些状态数据的程序，有时也称为“算法”。

说得形象一些，它认为数据是静态的，不会自行改变的，而需要各种各样的过程来改变数据。应该说“数据结构 + 算法 = 程序”这一等式就是“面向过程”方法学的精髓。

举个例子，要将一块木头从目前的位置向北搬到距原地 10 m 的地方，首先会定义一个表示木头的数据结构，存储位置、质量等状态信息，然后再编写一个算法操作这个表示木头的数据结构，以达到目的。

通过以上介绍，会发现面向过程程序设计实质上是用计算机观点来进行程序设计工作。因为计算机工作过程是一步一步进行的，为了完成指定功能，必须告诉它详细的解题步骤，也就是必须向计算机详细描述解题算法。面向过程程序设计就是根据计算机的要求，围绕算法进行程序设计。设计者站在计算机的立场，“设身处地”地设计解题步骤，并用适当的程序设计语言表达出来。面向过程程序设计方法本质上是功能分解，也称为算法分解，具体化为“自顶向下逐步求精”。

但是计算机观点与人类观点终究有很大区别，面向过程的思维方式也并不符合人类的思维习惯。那么人的思维习惯是什么呢？人类认识世界的思路总是针对一个个具体的客观事物来认识，包括它的形状、大小等属性，以及行为、功能、动作等。例如一说到猫，马上知道它是 4 只脚的、“喵喵”叫的，而且猫的行为动作大家也很熟悉，比如，说“猫抓老鼠”，应该马上就会在脑海中出现一个相应的图像。再来考虑一个日常生活中的例子：一个人肚子饿了，只需要走进饭店点好自己要吃的饭菜，厨师自己知道该怎样做，并不需要顾客告诉他做菜的具体步骤，顾客也不需要知道做菜的具体步骤。也就是说，人类习惯的解决问题方法是把问题中的事物考虑为一个个具有行为、功能和动作的个体，采用“顾客—服务”的工作模式。不同职业的人有不同的技能，若需要完成复杂任务时，就把完成这项任务所需的各类人员找来，提出要求布置任务即可。而不需要详细说明每个人完成自己承担任务的具体步骤。

面向对象程序设计方法模拟人类习惯的解题方法，用对象分解取代功能分解，即把程序

分解成许多对象，不同对象之间通过发送消息向对方提出服务要求，接受消息的对象主动完成指定功能，程序中的所有对象分工协作，共同完成整个程序的功能。面向对象程序设计方法的提出，是对软件开发方法的一场革命，它代表了计算机程序涉及的新颖的思维方法，是目前解决软件开发面临困难的最有希望、最有前途的方法之一。下面进一步说明面向对象程序设计方法及该方法的优点。

1.2.2 面向对象程序设计

在把程序分解成模块的概念形成以后，人们很快就认识到，应该把模块中有关实现的详细信息局部化，把模块类型化，对一种类型的模块设置足够的操作集，由此形成了抽象数据类型的概念，面向对象的程序设计就是以抽象数据类型为重要基础的。

面向对象程序设计方法是在前面所介绍的程序设计方法之上发展起来的，它的关键在于加入了类及其继承性，以类为样板创建对象。这种程序设计方法将计算看做是一个系统开发过程，系统由对象组成，对象通过消息传递经历一系列的状态变化以完成计算任务。如果只使用对象和消息，则可以称为基于对象的程序设计方法；如果进一步要求把所有对象都划分为类，则可称为基于类的程序设计方法，但仍然不是面向对象的程序设计方法。只有同时使用对象、类、继承和消息的方法才是真正面向对象的程序设计方法。在面向对象程序设计方法中，首要的任务是决定所需要的类，每个类应设置足够的操作，并利用继承机制显式地共享共同的特性。

熟悉 Visual Basic 的读者应该记得在使用它开发程序时，将一个个控件拖到界面上，然后修改它们的属性，使它们符合程序的需要，接着在这个控件的相应事件中填入代码。这就像儿时玩的搭积木，非常简单。这就是“面向对象”的程序设计带来的好处。

为了对“面向对象”的程序设计有一些了解，下面就再使用伪代码写一段代码，仅供比较。

一个“面向对象”的程序示例：

```
笔店
{
    铅笔;
    钢笔;
    圆珠笔;
    其他笔;

    卖出(笔种类,数量)
    {
        笔种类 = 笔种类 - 数量;
    }
    进货(笔种类,数量)
    {
```

```
            笔种类 = 笔种类 + 数量;
        }
        盘点()
        {
            打印  铅笔;  //打印出当前铅笔的库存量
            打印  钢笔;  //打印出当前钢笔的库存量
            打印  圆珠笔;  //打印出当前圆珠笔的库存量
            打印  其他笔;  //打印出当前其他笔的库存量
        }
    }
```

在这里把笔店中所拥有的各种笔以及笔店的各种行为抽象后封装在一起，形成抽象的笔店类。当一个具体的笔店对象 B1 发生了进货出货的行为时，只需给这个对象 B1 发送相应的消息即可：B1.卖出(钢笔,2)。即笔店可以根据接收到的信息，来决定自己应该如何运转。在此，笔店作为一个相对独立的实体，充分体现出自己的主动性，而不用外界来指明应该如何增加和减少笔的数量等操作的具体步骤。

虽然，使用面向对象的程序设计方法好像更繁琐，但其逻辑结构严谨、有序，符合人们的思维习惯，十分益于理解。

从以上的例子可以看出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。对象是包含现实世界物体特征的抽象实体，是一些属性及服务的一个封装体，在程序设计领域，可以用“对象 = 数据 + 作用于这些数据上的操作”这一公式来表达。

类是具有相同操作功能和相同的数据格式(属性)的对象的集合。类可以看做抽象数据类型的具体实现。抽象数据类型是数据类型抽象的表示形式。数据类型是指数据的集合和作用于其上的操作的集合，而抽象数据类型不关心操作实现的细节。从外部看，类型的行为可以用新定义的操作加以规定。类是对象集合的抽象，它规定了这些对象的公共属性和方法；对象是类的一个实例。苹果是一个类，而放在桌上的那个苹果则是一个对象。对象和类的关系类似于一般的程序设计语言中变量和变量类型的关系，也就是具体和抽象的关系。

消息是向某对象请求服务的一种表达方式。对象内有方法和数据，外部的用户或对象对该对象提出的服务请求，可以通过向该对象发送消息来实现。

面向对象的编程方法具有如下 4 个基本特征：

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，暂时不用部分细节。比如，要设计一个学生成绩管理系统，考查学生这个对象时，只要关心他的班级、学号、成绩等，而不用去关心他的身高、体重这些信息。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做单个的实体看待，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使用这些操作修改和观察。

2. 继承

继承是一种连接类的层次模型，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，并且类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。继承性很好地解决了软件的可重用性问题。比如说，所有的 Windows 应用程序都有一个窗口，它们可以看做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。

3. 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，对数据的访问只能通过已定义的界面。面向对象计算始于这个基本概念，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。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，即信息隐藏。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。

4. 多态性

多态性是指允许不同类的对象对同一消息作出响应。比如同样的加法，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—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，很好地解决了应用程序函数同名问题。

1.2.3 两种设计方法的比较

面向过程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。由于用户的需求和软、硬件技术的不断发展变化，按照功能划分设计的系统模块必然是易变的和不稳定的。这样开发出来的模块可重用性不高。

而面向对象程序设计从所处理的数据入手，以数据为中心而不是以服务（功能）为中心来描述系统。它把编程问题视为一个数据集合，数据相对于功能而言具有更强的稳定性。

面向对象程序设计同结构化程序设计相比最大的区别就在于：前者首先关心的是所要