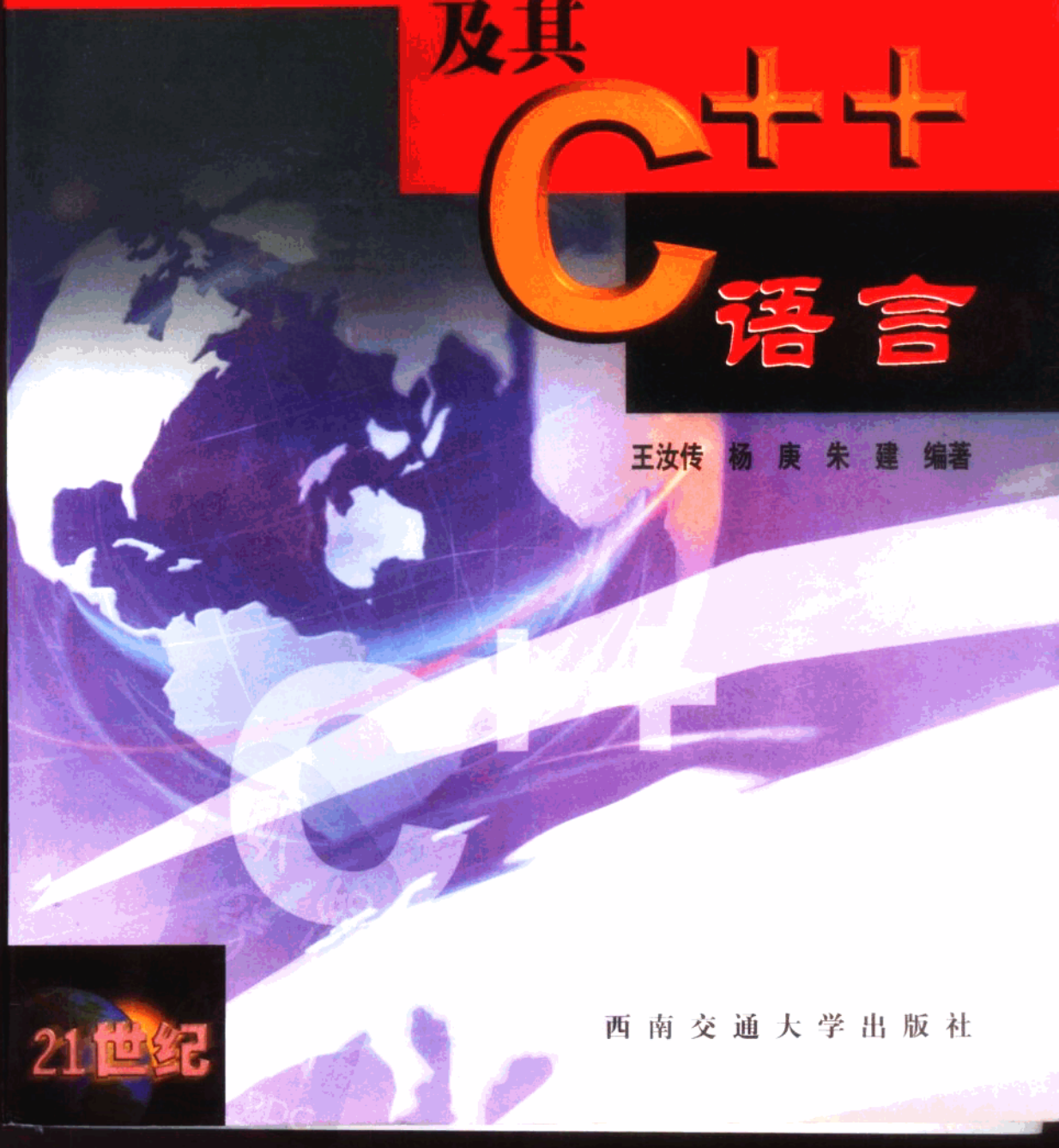
面向对象技术及其C++语言

王汝传 杨 庚 朱 建 编著



21世纪

西南交通大学出版社

前　言

面向对象技术是近年来迅速发展的一个研究领域，它对信息科学、软件工程、人工智能、系统工程和认知科学等学科都有重要的影响，尤其它应用于计算机科学与技术的各个方面，如数据库技术、操作系统设计、多媒体技术和计算机网络等。面向对象技术的出现被认为是程序设计方法学方面一场实质性革命，它与传统的结构化程序设计相比，具有许多优点。用面向对象技术进行程序设计、软件开发已成为一种时尚，这种方法从根本上改变了人们以往设计软件的思维方式，从而使程序设计者摆脱具体的数据格式和过程的束缚，将精力集中于要处理的对象的设计和研究上，大大减少了软件开发很难避免的繁杂性，提高了软件开发的效率。

C++是一种面向对象程序设计语言，但不是纯面向对象语言，而是混合型面向对象语言，即在过程语言中增加面向对象的结构，由于C++语言这种特性，就使得C++语言保持与C语言的兼容，从而使许多C语言代码不经修改就可以为C++语言使用。用C++语言编写的程序可读性更好，代码结构更合理，可以直接地在程序中映射问题空间的结构。C++可以为面向对象技术提供全面支持，它是实现面向对象技术比较通行和适用的手段，要想理解把握C++语言，离不开面向对象技术的指导，而介绍面向对象技术也必须结合对C++语言的运用。

随着计算机科学与技术的飞速发展和计算机应用逐渐普及，广大科技工作者和高等学校的研究生、本、专科学生迫切需要了解和掌握面向对象技术，学会用C++语言编写程序、开发软件，本书就是为了适应这一形势需要，在总结作者从事面向对象技术和C++语言研究、应用和教学与实践经验基础上，参考国内外有关资料编写而成。作者认为：只有以面向对象技术为指导进行面向对象程序设计才能充分展示面向对象程序设计语言的优越性，而现有介绍C++语言的教科书大部分只重视C++语言本身所具有的面向对象的语法特点，而没有从整体上用面向对象技术作指导。这本书是将面向对象技术与C++语言有机结合的一种尝试，其目的是为广大科技工作者及高等院校相关专业教师和学生系统学习和掌握面向对象技术和C++语言提供一本实用教材。

本书共分两篇。第一篇面向对象技术，共分四章，主要介绍面向对象技术的基本概念，面向对象技术的基本特征，面向对象技术分析和面向对象技术设计。第二篇C++语言，共分七章，主要介绍C++基本语句、函数、类和对象、派生与继承性、虚函数与多态性、C++图形程序设计基础和输入/输出流库等。

面向对象技术及其C++语言一书不但具有较深理论基础，而且也是一门实践性很强的课程。因此，本书在编写过程中力求做到概念清楚，由浅入深，论述详尽，实例丰富。书中每章后附有大量习题，以便巩固和加深所学内容。为了提高编程能力，希望参考实验指导书加强上机操作训练。

本书各篇自成体系，可根据教学对象和教学时数选学有关内容。作者建议：若面向研究生和本科高年级学生教学，可以第一篇面向对象技术为重点，适当介绍C++语言；若面向低年级本、专科学生教学，可以第二篇C++语言为重点，简单介绍第一篇的第一章和第二章的面向对象技术的基本概念和特征的内容。

象技术的基本概念和特征的内容。

本书第一章、第九章、第十章由王汝传编写；第二章、第三章、第四章由杨庚编写；第五章、第六章、第七章、第八章、第十一章由朱建编写。全书由王汝传统稿并主编。

中国科学技术大学博士生导师蔡庆生和赵保华两位教授在百忙的科研和教学工作中，从头到尾认真细致审阅了全部书稿，提出了许多有益修改意见。南京邮电学院教务处处长梅杓春教授在本书编写过程中自始至终给予作者鼓励、支持和帮助。南京邮电学院计算机科学与技术系主任王绍棣教授、居悌教授对本书编写也给予具体指导，并且同作者进行了多次讨论，提出了许多宝贵意见，使作者受益匪浅。张冬梅小姐还为书稿进行了精心排版。此外，书中还引用了其他同行工作成果，在此一并表示衷心感谢。

由于作者学识浅薄，编写时间仓促，书中错误及不妥之处在所难免，敬请读者批评指正。

作者

1999 年 8 月于南京

目　录

第一篇　面向对象技术

第二篇 C++语言

第一篇　面向对象技术

第一章　面向对象技术概述

第一节　面向对象技术的概念

面向对象技术概念来源于程序设计，但它现在显然不仅局限于程序设计方面，而已经发展成为软件开发领域的一种方法论，并且逐渐赢得厂商、设计人员和用户的青睐，成为目前软件开发领域的主流技术。面向对象技术不仅仅是一种新的程序设计技术，而且是一种全新的设计和构造软件的思维方法，它使计算机解决问题的方式更加类似于人类的思维方式，更能直接地描述客观世界。

目前的软件开发技术和方法远远地落后于硬件系统的发展水平，软件很难满足应用系统的需求，软件从诞生到现在虽然经历了几次变革，但总的来说，并没有突破程序的执行逻辑，发展是非常缓慢的，这些都孕育着一种全新软件的问世。

众所周知，计算机的发明和使用，对人类社会的进步和发展起到了不可磨灭的贡献，是现代文明的重要组成部分，是信息革命和产业革命的基础。然而，计算机系统本身的发展却是不平衡的，这极大地限制了计算机系统的发展和应用。从1946年第一台电子计算机问世以来，计算机的硬件已经历了四代的变化，即电子管时代、晶体管时代、集成电路时代以及大规模集成电路时代，计算机的硬件性能取得了长足的进展，速度、容量等成倍增长，而价格却成倍下降，而且，计算机的硬件水平还在突飞猛进地发展着。但这几十年来，计算机软件的性能却基本上没有发生根本性的变化，尽管各高等院校和研究机构在软件开发工具和技术方面取得了较大的进展，但这些进展并没有推广到软件开发者的日常工作中，当今的软件开发过程的研究与实际应用需求是不相适应的，程序员实际上仍在采用较原始的方式进行工作，他们编出来的程序代码缺乏一般性、可读性和可扩充性。由于软件产品不能适应诸如硬件、操作系统的改进，特别是不能适应用户要求的改变或提高等不断变化的环境，因此，软件的维护不仅占据了相当大的比重，而且非常的困难，软件的生命周期很少能达到十年以上。这一问题的实质在于软件远远落后于硬件的发展水平。目前，软件与硬件之间的差距最少也有两代处理器之多，而且这种差距还呈不断扩大之趋势。

另外，随着计算机硬件性能的提高和价格的下降，计算机得到了大规模的推广、普及和应用，软件的应用范围越来越广、软件的规模越来越大、要解决的问题越来越复杂、人们对软件的

要求越来越多且性能越来越高，因此，传统的软件开发技术、方法和工具不足以满足这些日益增长的要求，特别是组织大型的软件开发，这将使软件人员陷入困境。

大家知道，用计算机解决问题时需要用程序设计语言对问题的求解加以描述，实质上，软件是问题求解的一种表达形式。显然，如果软件能够直接地表现求解问题的方法，则软件不仅易于被人理解，而且易于维护和修改，从而可提高软件的可靠性和可维护性。此外，如果能按人们通常的思维方式来建立问题域的模型，则可以提高公共问题域中的软件模块化和重用化的可能性。面向对象技术的基本原则是：按人们通常的思维方式建立问题域的模型，设计出尽可能自然地表现求解方法的软件。

和人们认识世界的规律一样，面向对象技术认为：客观世界是由许多各种各样的对象组成的，每种对象都有各自的内部状态和运动规律，不同对象间的相互作用和联系就构成了各种不同的系统，构成了我们所面对的客观世界。可见，对象是一种普遍适用的基本逻辑结构，是一个以有组织的形式含有信息的实体。它既可以表示一个抽象的概念，也可以表示一个具体的模块，当然也就既可以表示软件，也可以表示硬件。于是，面向对象的技术既提供了一个分析、设计和实现系统的统一方法，又提供了描述、设计和实现硬件和软件系统的统一框架。

为了实现面向对象的基本原则，必须建立直接表示组成问题域的事物以及这些事物间的相互联系的概念，还必须建立适应人们一般思维方式的描述范式(paradigm)。在面向对象技术中，对象(object)和传递消息(message passing)分别是表现事物及事物间相互联系的概念。类(class)和继承(inheritance)是适应人们一般思维方式的描述范式。服务(service)是允许作用于该类对象上的各种操作。这种对象、类、消息和服务的程序设计范式的基本点在于对象的封装性(encapsulation)和继承性。通过封装能将对象的定义和对象的实现分开，通过继承能体现类与类之间的关系，以及由此带来的动态聚束(dynamic binding)和实体的多态性(polymorphism)，从而构成了面向对象技术的基本特征。

第二节　面向对象技术的发展

面向对象技术吸取了程序设计语言和数据建模技术等有益的成果，经过近三十年的演变和发展，逐渐形成了自己的范型，为人们提供了更强的问题求解能力。

最早的计算机语言是汇编语言。汇编语言使用符号来表示机器指令，这比程序员直接使用二进制的机器指令编程显示方便多了。在50年代中期开发的FORTRAN语言是第一个具有划时代意义的程序设计语言，许多重要的程序设计语言概念，例如变量、数组和控制结构(循环和条件分枝)等被引入到程序设计语言中来。但在程序设计实践中人们发现，在一个FORTRAN程序的不同部分的变量名总容易发生混淆，COMMON数据块的使用也不便于程序的查错和修改。于是，在50年代后期，Algol语言的设计者决定在程序段内部对变量实施隔离，因此，在Algol 60语言中出现了由"Begin……End"提供的块结构(类似于C语言中的块语句)，在一个块内出现的变量，只为这个块知道，这样，对它们的使用就不会与程序中其它块中的同名变量相混淆。这是程序设计语言中第一次尝试为数据提供保护和封装。

在60年代开发的Simula 67语言现在被人们公认为是面向对象语言的鼻祖。虽然它是一个通用的程序设计语言，但直到80年代中期之前，它主要的用途却是用来进行仿真建模。Simula 67是在Algol语言的基础上开发的，它将Algol语言中的块结构概念向前发展了一步，引

人了对象的概念。对象代表着仿真中的一个实体(例如一辆汽车,一个顾客或一个服务员等),在仿真期间,一个对象可以以某种形式与其它对象通信。从概念上讲,一个对象是既包含有数据又包含有处理这些数据的操作的一个程序单元。Simula 也使用了类的概念,类用于描述特性相同或相似的一组对象的结构和行为。Simula 也支持类继承。继承将类组织成层次,允许共享结构和行为。因此,Simula 奠定了面向对象语言的基础和某些面向对象的术语。Simula 在 1986 年进行了标准化,有几个公司在其计算机平台上实现了 Simula。

为了管理大型程序的需要,在 70 年代出现了数据抽象的概念。数据抽象是一个数据结构及作用于该数据结构上操作组成的一个实体或单元,数据结构的表示被隐藏在操作接口的后面。这样,通过操作接口,外部只知道它做什么,而不知道如何做,数据如何表示也是外部不知道的。将类型的概念扩展到数据抽象,即将那些主要是针对某个类型(例如学生类型)的操作聚集起来作为一个整体来看待,并与该类型一起看作一个独立的单元,将操作的语义作为该类型的定义,数据类型的值集由操作集间接定义,就构成了一个抽象数据类型。

支持抽象数据类型的最重要的语言之一是由美国国防部主持开发的 Ada 语言,它是一个被用来开发嵌入式实时系统的语言。这个语言包含一些常见的程序控制流构造(例如,选择、循环等)和具有定义新的类型、函数和子例程的能力。Ada 语言中面向对象的构造是包。有关 Ada 是否是面向对象的语言的讨论持续了很长时间。Ada 虽然支持诸如抽象数据类型、函数和运算符重载以及参数化多态性等面向对象的机制,但由于它不全面地支持继承,因而被认为是一种基于对象的语言。

在 70 年代和 80 年代这一时期,来自于 Simula 和其它早期的原型语言中的面向对象的概念在 Smalltalk 语言中做了完整的体现。Smalltalk 是当今最有影响的面向对象语言之一。Smalltalk 语言并入了 Simula 的许多面向对象的特征,包括类、继承,并支持对象标识。而且,在 Smalltalk 语言中,信息隐藏比 Simula 更严格。

Smalltalk 语言极大地丰富了面向对象的概念。在 Smalltalk 中,每个东西都是对象,包括类。这也就是说,在 Smalltalk 环境中,程序设计只是向对象发送消息,一个消息可以是两个数相加,或创建一个类的新的对象,或在一个类中加入一个新的服务(数据被处理的方式)。Smalltalk 语言是一种弱类型化的语言。一个程序中的同一个对象在不同时间可以表现为不同的类型。

自从 1986 年以来,面向对象技术逐渐走出了实验室和研究部门,开始进行实际应用,在工业和商业上更多地采用面向对象技术的问题求解方法。Object-C、C++和 Eiffel 等语言都是继 Smalltalk 语言之后有广泛影响的面向对象程序设计语言。90 年代出现的程序设计语言几乎都具有面向对象的机制,JAVA 语言就是一种纯面向对象语言。面向对象技术将是 21 世纪计算机处理的主流技术。

第三节　面向对象技术的作用

面向对应技术为在 90 年代提高软件生产率起着重要作用,面向对象技术主要优点体现在解决软件开发过程中维护复杂性和提高生产率以及大型程序设计等方面。面向对象技术通过下面软件开发方法以充分发挥其优势。

(1) 编写可重用代码;

(2) 编写可维护代码；

(3) 修改代码模块；

(4) 共享代码。

当可重复使用高质量的代码时，复杂性就得以降低，生产率则得到提高。面向对象提供的机制，特别是继承，非常鼓励代码的重新使用。程序员不是拷贝及修改模块，而是利用含精致的测试过了的代码的类库来开发软件。框架不久也将为诸如图形用户界面这样的复杂领域所用。继承能很大地提高开发人员建立、扩充及维护系统的能力。

面向对象技术的最早得益者是软件开发。对开发人员来说，其当前工具不能跟上计算环境不断增长的复杂性，而面向对象技术正好提供了由新语言及工具所支持的编程方法，从而可以大大提高生产率。对用户来说，面向对象技术提供了为使应用程序更易于使用而需要的一致性与灵活性。

面向对象代表了开发与使用软件的方法上的本质改变。软件可重用意味着类可被容易地组合与修改以建立新的应用程序。将数据与服务封装在一起则改变了程序设计过程的整个性质。

在数据与函数之间的传统区分对软件开发来说是个很大的障碍。数据结构经常需要修改，控制函数则必须被加以重新考察以保证它们与数据的一致性。使数据结构与函数协调一致必须花去大量的资源，并使错误产生的机会大大提高。

面向对象技术将数据与服务封装在一起简化了此过程，方便了维护，并减少了程序设计过程中出错的可能性。

当然，为了得到这些益处，开发人员必须在其分析问题及将问题转换成程序的方式上有很大的改变。面向对象的方法与传统方法显然是有很大区别的。虽然许多人都认为面向对象的程序更易于编写，但其中的概念仍比传统方法中的概念要抽象，而且起初也较难掌握。初学者在编写了面向对象程序时常常知道程序能正确运行，却不知为什么。但以后他们终会豁然开朗的。

使用面向对象技术时，设计目标从模拟现实世界的行为转向了模拟现实世界中存在的对象及其各自的行为。如果正确使用面向对象的技术，那么所建立的应用程序的体系结构就会与问题的结构非常接近。这就使得程序开发、使用与维护更加平坦、容易与迅速。

对象相对来说易于定义、实现与维护，因为对象自然地反映了应用领域的模块性。利用模块化与继承机制，软件对象就可在以后的应用中被重复使用从而大大减少必须编写的新代码的数量。传统的应用程序可能需使用成千上万的数据记录组成预定义数据类型，但面向对象的应用程序则可从一组数目小得多的对象中加以创建。

面向对象的代码可以足够一般以不作修改就能加以重用。由于每个对象都知道如何以其适宜的方式来处理请求，所以同样的指令可以操作至许多不同的对象上。

一、复杂性的维护

将一应用分解成对用户有意义的实体及关系是传统的分析与设计方法。面向对象的方法则使此分解过程也扩展到了实现阶段。由于应用领域中的对象与软件领域中的对象有着直接的对应，所以设计与实现面向对象的应用程序较为容易。

1. 软件开发的灵活性

一旦对象被定义，类库被扩充，程序设计过程就变得越来越容易。通过继承机制进行子类

化的过程使得程序设计变成仅对子类与超类或父类的差异进行编程的过程。

继承机制不仅影响了程序构造的效率与质量，而且还影响到程序员任务的分配。由于继承使得对对象的增删不必通过大大修改应用程序的逻辑结构即可进行，所以大型项目可被划分成若干个开发组。虽然在编程小组中对任务的分工已不是一个新概念，但面向对象的程序设计方法却有利于各程序组的独立工作——对每个对象及其服务的实现完全可以独立进行。具体工作过程如下：

一组开发人员先就一组抽象类及其相应的服务取得一致意见，然后将此类集作为其应用程序框架。抽象类说明了开发组必须实现的行为。接下来，开发人员可独立工作以创建能对某组消息进行处理并使之能方便地结合进整个应用程序的新的子类。开发组的某些成员可以实现协调消息发送的代码，另一些成员则可定义及实现新的子类，从而可使开发组发挥出最佳才干。

对每个对象的建立与修改都不必考虑其它组的成员是如何对他们的系统加以编码的。因此，工作可沿着由公共父类所提供的一致性而在各个不同的路径上进行。面向对象的开发工具还方便了对各独立成分以及整个系统的调试。这些图形丰富的环境含有诸如浏览器这样的新工具，从而可使调试更快也更有效率。此外，面向对象的语言还消除了许多常见的编码错误。总之，不断出现的开发环境伴随着强壮的类库将大大减少创建、调试及集成至大型系统的代码量。

面向对象系统提供了实际实现所需要的性能与灵活性。编程可用标准商用语言(如 C)的扩展来完成，并且面向对象的技术在某种程度上还能与过程语言一起使用。

面向对象的程序设计还扩展了可被编程的应用领域，因为它解除了对预定义数据类型的约束。面向对象的程序设计可接受复杂的数据结构。新的数据类型能不通过修改现有代码就被加入。此外，面向对象的环境还能对封装起不同的应用程序以提供一个面向对象的桥梁。

2. 可重用性

面向对象的技术可使程序员不必再三反复地编写同样的程序。通过用新的对象替换旧的元素或对象，或干脆直接在应用程序中加入新的对象，程序员就能修改一程序的功能。一般指令(消息)不需修改，因为专门的实现细节(服务与数据)驻留在对象之中。所以，每个对象都知道如何执行其自己的动作，这一概念与过程式程序设计完全不同。在过程式编程中，操作及规则都是作用到独立的数据集上的，程序员将注意力集中于语言问题；而在面向对象环境中，重要的问题是建立可在各种环境下被使用一强壮的类库或对象集。类库可提供高质量的应用程序建立模块。类不仅提供了模块化与信息隐蔽，而且也提供了为继承与多态性所进一步促进的可重用性。

可重用软件的传统技术并没有被面向对象技术取代。事实上，传统技术常常含有面向对象的成分。例如，程序骨架的概念就为抽象类的概念所蕴含。通常，传统的程序员是通过拷贝与编辑而重用代码的，而面向对象的程序员则可通过创建一子类并重载其某些方法而自动完成同样的功能。

二、生产率的提高

1. 可扩充性与可维护性

修改与扩充一个面向对象应用程序是较容易的，如图 1.1 所示，可以增加新的对象类型而不需改变已有的结构。继承特性使得可以从旧的对象中建立新对象。服务是易于修改的，因为

它们被放在唯一的地方，而不是分散在程序中或潜在地在程序中多次重复。因此，使用面向对象技术时，就不需要在整个过程体中搜索和替代函数和变量。

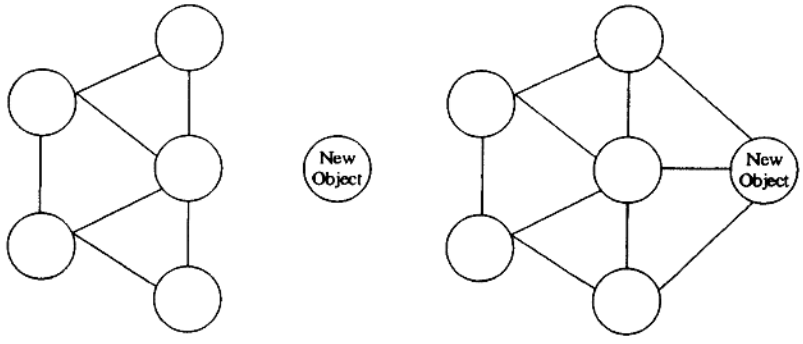


图1.1 修改一个面向对象程序不会影响程序结构

面向对象特征对于软件维护也是特别有用的。模块性使得对程序的修改变得更加有效。多态性减少了过程个数，因而也减少了维护人员需要理解的程序规模。类继承性使得可以建立一个程序的新版本而不影响旧版本。继承机制把程序修改作为子类归档，这些子类表示对超类的修改的历史。

创建一个子类，即通过仅描述其本身与其父类之间的差别而定义一个新类的面向对象技术，将使确定一个程序与其旧版本有何不同变得更加容易。一个子类集表示对一个超类的修改历史。继承机制减少了人为错误，因为对一个类的修改将会自动传播到它的所有子类。

面向对象技术还被成功地结合到传统体系中。如 C^{++} 这样的混合体系在程序设计开发工具中增加了面向对象功能，这使得程序员可以在自己熟悉的背景中利用新技术。

2. 方便用户编程

目前，许多用户都编写了大量的程序，不同面向对象技术，程序设计工作将会变得更加容易。今天的非程序员将可能会成为明天高级用户。这种建立面向对象模型和应用程序简易特性将会吸引广大的用户。实际上，用户虽然不知道应用程序内部细节，但是用户可以结合面向对象技术模型和类库中知识，用户将能够建立和扩充应用程序。对于用户来说，面向对象的优点在于具有丰富内容类库的传递，类库对真实世界过程表示很直观，并且易于修改和使用。

三、用于大型程序设计

大型程序的复杂性主要表现在两个方面：一方面，由于大型程序必须由多人合作完成，如何划分任务、估计和分配资源、掌握每个程序员的进度、控制及检查每个阶段的设计标准等，这就构成了进行大型程序设计时管理的复杂性，也是大型程序设计与单人能够完成的小型程序设计的一个重要区别；另一方面，大型程序具有大量的系统状态，我们要正确地处理它的中间状态，组织系统的程序逻辑和验证系统的正确性都是比较困难的。

对于大型程序，正确性是程序的首要目标，程序的任一微小错误都会造成重大的损失，并且大型程序设计的周期长，且需要多人合作，不可避免地要出现一些错误。

但是，对于一个大型程序，仅仅满足正确性是远远不够的，易维护性、可读性和可重用性都

是非常重要的，这是因为，要想在短期内重新开发一个大型程序是不可能的。易维护性能使程序在出现错误时，得到及时的改正；可读性又可以保证程序易于理解、便于使用和调试，从而使程序功能得以充分的发挥；可重用性可以使一个程序中开发的通用模块能够在其它程序中再次使用，从而节省开发费用，简化程序的部分复杂性。

在开发一个大型系统时，应对整个任务进行清晰的、严格的划分，使每个程序员清晰地了解自己要做的工作以及与他人的接口，使每个程序员可以独立地设计调试自己负责的模块，使各模块能顺利地应用到整个系统中去。

大型程序在设计时采用模块化的办法，将一个问题分解成若干个小问题，而每一个小问题又可以再分解成更小的问题，每个小问题都可以是一个独立的模块，这些模块都有一个清晰的抽象界面，它只说明做什么，不必说明如何去做，同时还说明模块之间的关系。这些模块构成了一种层次结构，在设计时采用自顶向下，分而治之的办法。

面向对象技术提供了一种有效的模块分解方法，进一步发展了基于数据抽象的模块化设计。并且在数据抽象和抽象数据类型之上又引人了动态连接和继承性等机制，使其更好地支持大型程序设计。

第四节　面向对象技术的应用

面向对象技术的应用实际上已十分广泛，事实上计算机科学与应用各个领域，都不同程度应用面向对象技术，除了软件开发和程序设计之外，面向对象技术至少还可以应用于以下方面：

1. 数据库技术

数据库的发展已经历了几个阶段，其数据模型主要是层次数据模型、网状数据模型和关系数据模型。而面向对象的数据模型是近几年来研究热点，面向对象技术的引人无疑将对数据库技术进一步发展产生积极的影响，面向对象数据库管理系统由于引人面向对象技术的机制，例如继承关系，比传统的数据库管理系统具有许多优越性。

2. 操作系统设计

用面向对象技术设计新一代操作系统，特别是分布式操作系统，事实上，这方面现已取得了不少成果。例如操作系统 V、Amoeba、Nexus、Eden、Argus、Clouds 等，都是一些著名的面向对象的分布式操作系统。

3. 系统模拟

面向对象与模拟更有历史渊源。最早的面向对象程序设计语言 Simula 就是为模拟而设计的。离散事件模拟是当前主要的模拟模型，这种模型是事件驱动的，而事件发生的主要标志是消息的到达。系统中持有状态的实体和使状态改变的操作正好可用对象来描述。除了顺序模拟外，并行模拟或分布模拟更要采用面向对象技术。

4. 人机交互界面设计

面向对象技术广泛应用于图形用户界面(GUI)、多媒体界面与系统、可视化程序设计等。由于对象的概念很容易描述诸如窗口、图形、图像、图符、按钮等实体，所以面向对象技术在这一领域的应用十分活跃。事实上，面向对象技术已成为人机交互界面设计中的主流技术。

5. 人工智能和知识工程

面向对象技术可广泛用于人工智能、知识工程。如用于知识表示和知识库，得到面向对象知识表示和面向对象知识库。又如面向对象程序设计所具有的模块性和数据隐藏、多态性等特点，允许异构型知识结构和推理机制的无缝集成。还有，对象所具有的良好的通讯能力是实现分布式、多代理智能系统的自然选择。所以，面向对象技术在人工智能、知识工程中也将发挥重要作用。

6. 计算机辅助软件工程(CASE)

计算机辅助软件工程(CASE)的发展更是直接受到面向对象方法的影响。CASE 工具的核心是存储各种与项目有关信息的公用库(repository)，它的结构多数是以面向对象的基本思想组织的。新的 CASE 工具往往支持多种描述方法，既支持传统的数据流图、ER 图、结构图等，也支持最新的以面向对象方法为基础的各种图表和其它表达方法。

7. 多媒体技术

多媒体技术是一个十分热门的领域。当我们考察多种介质，多种形式的信息处理时，实体的概念，封装的概念，消息传递的概念，无疑会给我们带来方便，使得系统或应用软件的结构更加清晰、简明，开发工作更有秩序，开发效率与质量进一步提高。

8. 计算机网络

作为计算机技术和通信技术的结合，计算机网络的应用日益广泛。通信技术中的许多传统的概念与方法，与面向对象方法有着相通之处，这就使得面向对象技术在计算机网络领域中的应用显得十分自然。不论是网络的设计还是控制和管理都可以从面向对象方法中得到启发和益处。

9. 计算机辅助教学(CAI)

计算机辅助教学也是一个迅速发展领域，教学内容组织、各种表达形式的使用，都是面向对象技术得到有效运用的方面。

第五节　面向对象程序设计

90 年代以来，面向对象程序设计(Object-Oriented Programming，简称 OOP)异军突起，迅速地在全世界流行，并一跃成为程序设计的主流技术。面向对象程序设计是软件系统设计与实现的新方法，面向对象程序设计和传统结构化程序设计(Structure Programming 简称 SP)比较起来，有许多优越性，到底什么是面向对象程序设计，在我们对它给出解释之前，需首先讨论一下结构化程序设计。

结构化程序设计是 60 年代诞生的，在 70 年代到 80 年代已遍及全球，成为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。

结构化程序设计的设计思路是：自顶向下、逐步求精；其程序结构是按功能划分若干个基本模块，这些模块形成一个树状结构；各模块之间的关系尽可能简单，在功能上相对独立；每一模块内部均是由顺序、选择和循环三种基本结构组成；其模块化实现的具体方法是使用子程序。

结构化程序设计由于采用了模块分解与功能抽象，自顶向下、分而治之的手段，从而有效地将一个较复杂的程序系统的设计任务分成许多易于控制和处理的子任务，这些子任务都是可独立编程的子程序模块。这些子程序中的每一个都有一个清晰的界面，使用起来非常方便。