高等院校信息技术规划教材

C++面向对象程序设计（第2版）

李晋江　编著

清华大学出版社

高等院校信息技术规划教材

C++面向对象程序设计（第2版）

李晋江　编著

清华大学出版社
北京

内 容 简 介

全书详细介绍了和C++相关的C语言知识、类和对象、继承、多态、模板和运算符重载，以及面向对象设计方法的概念，结合知识点简要地讨论了几种常用的设计模式；针对重要概念精心设计了大量实例，涉及很多技巧和经验。

本书不仅可以作为高等院校C++面向对象程序设计的教材，也是希望了解C++语言和面向对象程序设计知识的专业人员的参考书。

图书在版编目（CIP）数据

C++面向对象程序设计/李晋江编著. —2版. —北京：清华大学出版社，2016
高等院校信息技术规划教材
ISBN 978-7-302-42205-1

Ⅰ. ①C…　Ⅱ. ①李…　Ⅲ. ①C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(2015)第279038号

责任编辑：白立军　战晓雷
封面设计：常雪影
责任校对：白　蕾
责任印制：李红英

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：清华大学印刷厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：34　　**字　　数**：784千字
版　　次：2012年7月第1版　2016年1月第2版　　**印　　次**：2016年1月第1次印刷
印　　数：1～2000
定　　价：59.00元

产品编号：066651-01

前言 foreword

C++是一门非常重要的语言，有着许多不同语言的特性，它甚至可以上升到思想的高度，其思想被很多其他语言借鉴和沿袭。掌握了C++，学习其他语言就非常容易了。

本书内容覆盖基本概念和方法，基本数据结构和面向对象的概念、方法和技巧。全书分为9章。第1章介绍面向对象程序设计的思想和基本概念。第2章介绍有关C++的一些知识，让读者们轻松地从C语言转到C++。第3章介绍C++的一些基本知识，如名字空间、输入输出等。第4～6章介绍面向对象的三大主要特性：封装、继承和多态。第7章介绍操作符重载，通过示例对常见的操作符重载进行分析。第8章介绍面向对象编程体系中的思想精髓——面向接口编程。第9章介绍模板相关知识。

本书全面而又系统地介绍了C++编程的基本知识，包括C++基本数据类型、基本语法和面向对象编程的基础知识和技巧。无论是刚开始编程还是已有一些编程经验，都会发现本书的精心安排使得学习C++变得快捷又轻松。

大多数教材都是按各部分内容逻辑上的先后顺序进行组织的，各知识点比较孤立，跨度较大，容易使学生产生“只见树木，不见森林”的感觉。针对这一问题，本书将知识点进一步细化分级，突出重点、难点，缩小台阶，达到深入浅出、循序渐进的目的。

本书针对已有C程序设计基础，要学习C++面向对象程序设计的读者。本书可作为高等学校C++面向对象程序设计课程的教材，也可作为工程技术人员的参考书。

本书具有以下特色：

(1) 内容整合。C、C++相融合。本书针对已有C语言基础的学生，帮助其从C语言顺利过渡到C++语言，涵盖了C++语言的主要特征，使初学者能很快学习掌握C++。

(2) 知识体系完整，教学内容由浅入深，从易到难，循序渐进，层次分明，对每个C++的理论方法从需求到应用做了详细的描述。

(3) 本书在内容组织上采用案例教学的思想，对C++中容易出

错的地方都用实例进行了讲解。

（4）本书配有电子课件、程序源代码、习题参考答案。

本书中包含大量的示例代码，其中大部分是完整程序，如无特殊声明，该程序编译和运行环境为 Visual C++ 2012、32 位 Windows 7 系统。

这里，特别感谢恩师范辉教授一直以来给予的关怀、教诲和启迪。在初稿完成后，范辉教授仔细审阅后提出了很多宝贵意见，使本书更加完善。同时，对所有曾经鼓励和帮助过我的领导、同事、专家、朋友表示诚挚的谢意。

由于时间仓促及作者水平有限，本书肯定有疏漏甚至错误之处，望专家和广大读者不吝指正。

作者

2015 年 10 月

目录 Contents

第1章 chapter 1

绪　论

程序设计语言是人们为了描述计算过程而设计的一种具有语法、语义描述的记号。对计算机工作人员而言，程序设计语言是除计算机本身之外的所有工具中最重要的工具，是其他所有工具的基础。由于程序设计语言的这种重要性，从计算机问世至今的半个世纪中，人们一直在为研制更新更好的程序设计语言而努力着。

1.1 程序设计语言

程序设计语言，通常简称为编程语言，是用于书写计算机程序的语言，是一组用来定义计算机程序的语法规则，是一种被标准化的交流技巧，用来向计算机发出指令。一种计算机语言让程序员能够准确地定义计算机所需要使用的数据，并精确地定义在不同情况下所应当采取的行动。语言的基础是一组记号和一组规则。根据规则由记号构成的记号串的总体就是语言。在程序设计语言中，这些记号串就是程序。

程序设计语言有3个方面的因素，即语法、语义和语用。

(1) 语法表示程序的结构或形式，亦即表示构成语言的各个记号之间的组合规律，但不涉及这些记号的特含义，也不涉及使用者。

(2) 语义表示程序的含义，亦即表示按照各种方法所表示的各个记号的特定含义，但不涉及使用者。

(3) 语用表示程序与使用者的关系。

语言的种类千差万别，一般说来主要部分有以下4种：

① 数据成分，用以描述程序中所涉及的数据；

② 运算成分，用以描述程序中所包含的运算；

③ 控制成分，用以表达程序中的控制构造；

④ 传输成分，用以表达程序中数据的传输。

在过去的几十年间，大量的程序设计语言被发明、被取代、被修改或组合在一起。尽管人们多次试图创造一种通用的程序设计语言，却没有一次尝试是成功的。之所以有那么多种不同的编程语言存在的原因是，编写程序的初衷其实也各不相同；还有，不同程序之间的运行成本(runtime cost)各不相同。有许多语言只用于特殊用途。

程序设计语言可以从不同角度分类。

(1) 按语言级别，有低级语言和高级语言之分。

低级语言包括字位码、机器语言和汇编语言。其中，字位码是计算机唯一可直接理解的语言，但由于它是一连串的字位，复杂、烦琐、冗长，几乎无人直接使用。机器语言是直接用二进制代码指令表达的计算机语言，指令是用0和1组成的一串代码。汇编语言是机器语言中地址部分符号化的结果，即用符号代替机器语言的二进制码，于是汇编语言亦称为符号语言。这些语言的特点是与特定的机器有关，功效高，但使用复杂、烦琐、费时、易出差错。

高级程序设计语言(也称高级语言)的出现使得计算机程序设计语言不再过度地依赖某种特定的机器或环境。这是因为高级语言在不同的平台上会被编译成不同的机器语言，而不是直接被机器执行。最早出现的编程语言之一 FORTRAN 的一个主要目标就是实现平台独立。

与机器语言和汇编语言相比较，高级语言与具体计算机无关，是一种能方便描述算法过程的计算机程序设计语言。用高级语言编写的程序称为“源程序”。计算机不能直接运行源程序，通常有解释执行和编译执行两种方式在计算机上执行源程序。

① 解释执行，即让计算机运行解释程序，解释程序逐句取出源程序中的语句，对它作解释执行，输入数据，产生结果。BASIC 是典型的解释执行的语言：编写源程序(.bas)→逐行/逐语句提交给解释器→解释器逐语句翻译成机器代码→执行器逐语句执行翻译的机器代码。解释执行方式调试方便，但执行速度慢。

② 编译执行，即先运行编译程序，从源程序一次翻译产生计算机可直接执行的二进制程序(称为目标程序)；然后让计算机执行目标程序，输入数据，产生结果。C 语言是典型的编译执行语言：编写源程序(.c)→交给编译器编译(tcc.exe)→得到目标代码/机器代码(.obj)→交给连接器(link.exe/tlink.exe)将目标代码联编成可执行程序(.exe/.com)→提交给操作系统执行。编译执行的优点是执行速度快，独立执行，不需要其他应用程序支持；缺点是不利于调试，有一定的机器依赖性。

(2) 按照用户要求，有过程式语言和非过程式语言之分。

过程式语言的主要特征是，用户可以指明一列可顺序执行的运算，以表示相应的计算过程。例如，FORTRAN、COBOL、ALGOL60 等都是过程式语言，这些语言编程的结构可分为顺序结构、分支结构和循环结构。

目前已被人们研究或应用的非过程式语言范型主要有函数式语言、逻辑式语言、面向对象式语言等几种。典型的函数式语言有 LISP、APL 与 ML 等，函数式语言也叫作用式语言，纯函数式语言中不使用赋值语句，其语法形式很类似于数学上的函数，故得名。逻辑式语言也叫说明式语言、基于规则式语言，以逻辑程序设计思想为理论基础，其主要核心是事实、规则与推理机制，最有代表性的逻辑式语言是 PROLOG。面向对象式语言简称对象式语言，它与传统过程性语言的主要区别在于：在传统过程性语言中把数据以及处理它们的子程序当作互不相关的成分分别处理，而在对象式语言中则把这两者统一作为对象封装在一起进行处理。典型的面向对象程序设计语言有 C++，C++ 是在 C 的基础上扩充而成的，是 C 的超集，它在 C 的基础上扩充了类、对象、继承、运算符重载等面向对象的概念。

非过程式语言的含义是相对的，用户描述问题时不必指明解决问题的顺序。凡是用户无法指明表示计算过程的可顺序执行的语言，都是非过程式语言。但这只是一个相对的概念，也就是说随着近代程序设计技术的改进，需要用户提供的描述解决问题顺序的内容越来越少，即越来越非过程化。

(3) 按照应用范围，有通用语言和专用语言之分。目标非单一的语言称为通用语言，如 FORTRAN、C、Java 等都是通用语言。C 语言是目前流行的通用程序设计语言，是许多计算机专业人员和计算机的爱好者学习程序设计语言的首选。目标单一的语言称为专用语言，如 APT(Automatically Programmed Tool，自动编程工具)等。为了解决数控加工中的程序编制问题，20 世纪 50 年代 MIT 设计了一种专门用于机械零件数控加工程序编制的语言 APT，可用来描述零件图纸上的几何形状及刀具相对零件运动的轨迹、顺序和其他工艺参数。

(4) 按照使用方式，有交互式语言和非交互式语言之分。具有反映人-机交互作用的语言成分的称为交互式语言，如 BASIC 语言就是交互式语言。语言成分不反映人-机交互作用的称非交互式语言，如 FORTRAN、COBOL、PASCAL、C 语言等都是非交互式语言。

(5) 按照并发程度，可分为顺序语言、并发语言和分布语言之分。只含顺序成分的语言称为顺序语言，大部分过程语言如 FORTRAN、COBOL 等都属顺序语言。含有并发成分的语言称为并发语言，如并发 ADA 等都属并发语言。并发是理论计算机科学研究的热点之一，人们提出了各种并发计算模型，如 CSP、CCS、Petri 网、π-演算和进程代数等。并发程序的执行与顺序程序的执行差别在于并发程序的执行结果取决于时间和程序执行推进速度，因此如何实现进程间相互作用控制，如对共享资源存取的同步控制及通信控制，成为一个关键问题。考虑到分布计算要求的语言称为分布语言，人们对分布并行语言的研究已有较长的历史，如今比较有代表性的语言系统有 DC++ (Distributed C++)，它以支持并行计算为首要任务。DC++ 是一种面向对象的分布式程序设计语言，在 DC++ 语言中增加了一系列关于分布式程序设计的语言设施，以便于程序员灵活应用，编写出高质量的分布运行的程序。它以活动对象——进程作为分布单位，采用汇合机制实现进程间的同步和通信，并且支持进程的动态创建和撤销，同时提供了丰富的进程通信控制机制。进程组概念的引入，提供了一组进程中的各个成员与外部世界的统一接口，增强了对系统的控制，是 DC++ 语言的一大特色。分布式计算机系统的发展向程序设计语言提出了新的要求，它要求程序设计语言不仅能为基于共享变元的并行处理提供支持，而且还应该能对具有通信功能的分布处理提供支持。

面向对象程序设计以及数据抽象在现代程序设计思想中占有很重要的地位，未来的语言将不再是一种单纯的语言标准，而是向完全面向对象，更易表达现实世界，更易编写的方向发展。

计算机语言的未来发展趋势如下：

(1) 简单性。提供最基本的方法来完成指定的任务，只需理解一些基本的概念，就可以用它编写出适合各种情况的应用程序。

(2) 面向对象。提供简单的类机制以及动态的接口模型。

（3）安全性。用于网络、分布环境时有安全机制保证。

（4）平台无关性。与平台无关的特性使程序可以方便地被移植到网络上的不同机器、不同平台。

1.2 C++ 的发展历史

C 语言是使用最广泛的语言之一，可以说 C 语言的诞生是现代程序语言革命的起点，是程序设计语言发展史中的一个里程碑。自 C 语言出现后，以 C 语言为根基的C++ 、Java 和 C# 等面向对象语言相继诞生，并在各自领域大获成功。

下面介绍从结构化程序的 C 语言发展到面向对象的 C++ 的几位非常有影响的人物：Niklaus Wirth、Dennis M Ritchie、Bjarne Stroustrup 和 Grady Booth。

1. 结构化程序设计的首创者 Wirth

Niklaus Wirth(1934 年 2 月 15 日—)有一句在计算机领域人尽皆知的名言：

“程序 = 算法 + 数据结构”(Programs = Algorithm + Data Structures)

Niklaus Wirth 凭借这一句话获得图灵奖，这个公式对计算机科学的影响程度堪比物理学中爱因斯坦的 $E=MC^2$，一个公式展示出了程序的本质。

“结构化程序设计”(structure programming)概念的要点是：不要求一步就编制成可执行的程序，而是分若干步进行，逐步求精。第一步编出的程序抽象度最高，第二步编出的程序抽象度有所降低……最后一步编出的程序即为可执行的程序。用这种方法编程看似复杂，实际上优点很多，可使程序易读、易写、易调试、易维护、易保证其正确性及验证其正确性。结构化程序设计方法又称为“自顶向下”或“逐步求精”法，在程序设计领域引发了一场革命，成为程序开发的一个标准方法，尤其是在后来发展起来的软件工程中获得广泛应用。有人评价说 Wirth 的结构化程序设计概念“完全改变了人们对程序设计的思维方式”，这是一点也不夸张的。

2. C 语言之父和 UNIX 之父 Dennis M Ritchie

Dennis M Ritchie(1941 年 2 月 15 日—2011 年 10 月 12 日)是著名的美国计算机科学家，对 C 语言和其他编程语言、Multics 和 UNIX 等操作系统的发展做出了巨大贡献，被称为 C 语言之父、UNIX 之父。

1978 年 Brian W. Kernighan 和 Dennis M. Ritchie 出版了名著《C 程序设计语言》(*The C Programming Language*)，现在此书已翻译成多种语言，成为 C 语言方面最权威的教材之一。作为一门伟大的语言，C 语言的发展颇为有趣，C 语言是借助 UNIX 操作系统的翅膀而起飞的，UNIX 操作系统也由于 C 而得以快速移植落地生根，两者相辅相成，成就了软件史上最精彩的一幕。1983 年他与肯·汤普逊一起获得了图灵奖，理由是他们“研究发展了通用的操作系统理论，尤其是实现了 UNIX 操作系统”。1999 年两人因发展 C 语言和 UNIX 操作系统的贡献而共同获得了美国国家技术奖章。

3. C++ 之父 Bjarne Stroustrup

从贝尔实验室大规模编程(Large-scale Programming)研究部门设立至 2002 年晚些时候,Bjarne Stroustrup(1950 年 12 月 30 日—)一直担任那里的负责人。

1979 年,Bjarne Stroustrup 开始开发一种语言,当时称为"C with Class",后来演化为 C++。1998 年,ANSI/ISO C++ 标准建立,C++ 的标准化标志着 Bjarne Stroustrup 倾注 20 年心血的伟大构想终于实现。他还写了一本《C++ 程序设计语言》(*The C++ Programming Language*),它被许多人认为是 C++ 的范本。

4. OO(Object-Oriented)教父 Grady Booch

Grady Booch(1955 年 2 月 27 日—)是美国 Rational 软件工程公司的首席科学家和 Booch 方法的主创人。与 Rational 公司的 Ivar Jacobson、Jim Rumbauth 共同创建了一种可视化地说明和建造软件系统的工业标准语言——统一建模语言(Unified Modeling Language,UML)。世界公认这 3 个人对开发对象技术做出了许多重大的贡献。UML 在 1997 年被对象管理组织(OMG)正式确定为国际标准。Grady Booth 认为,我们能够通过不断地提升抽象级别来解决复杂问题。

C++ 的发展历史可浓缩为图 1.1。

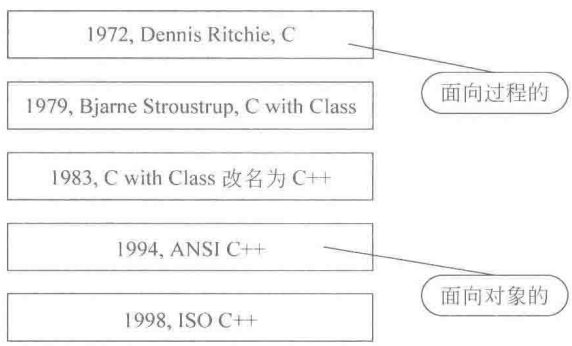
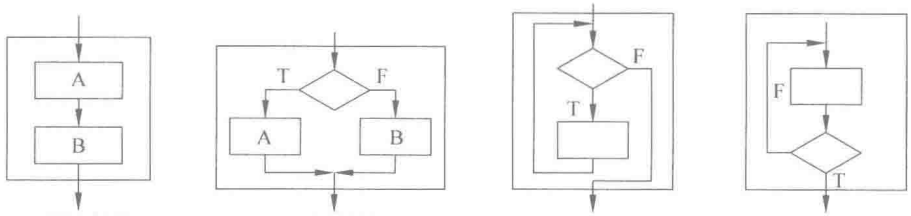


图 1.1 C++ 的发展历史

1.3 面向过程程序设计语言

所谓面向过程,就是指从要解决的问题出发,围绕问题的解决过程分析问题。面向过程分析方法考虑的是问题的具体解决步骤(解决方法)以及解决问题所需要的数据(数据的表示),所以在面向过程程序设计中,重点是设计算法(解决问题的方法)和数据结构(数据的表示和存储)。面向过程的程序有明显的开始、明显的中间过程、明显的结束,程序的编制以这个预定好的过程为中心,设计好开始子程序、中间子程序、结尾子程序,然后按顺序把这些子程序连接起来,一旦程序编制好,这个过程就确定了,程序按顺序执行。

结构化程序设计方法(Structured Programming,SP)的着眼点是“面向过程”,解决好过程,找到过程之间的联系是解决问题的关键,结构化程序设计是进行以模块功能和处理过程设计为主的详细设计的基本原则。其概念最早由 E. W. Dijkstra 在 1965 年提出的,是软件发展的一个重要的里程碑。它的主要观点是采用自顶向下、逐步细化的程序设计方法;使用 3 种基本控制结构构造程序,任何程序都可由顺序、选择、循环 3 种基本控制结构构造(见图 1.2)。结构化程序设计的特点是将程序中的数据与处理数据的方法分离。



(a) 顺序结构　(b) 选择结构　(c) 当型循环　(d) 直到型循环

图 1.2　结构化程序的基本控制结构

结构化程序设计采用自顶向下、逐步求精和模块化的分析方法。按自顶向下的方法,不是从一开始就力图触及到问题解法的细节,而应当从问题的全局出发,宏观进行需求分析,确定“做什么”以及怎样把问题分解为几个子问题或子功能,并画出层次结构图及执行流程图。下一步是在子问题一级描述算法,它是对全局算法的细化。自顶向下是指对设计的系统要有一个全面的理解,从问题的全局入手,把一个复杂问题分解成若干个相互独立的子问题,然后对每个子问题再作进一步分解,如此重复,直到每个问题都容易解决为止。

逐步细化是指程序设计的过程是一个渐进的过程,先把一个子问题用一个程序模块来描述,再把每个模块的功能逐步分解细化为一系列的具体步骤,以致能用某种程序设计语言的基本控制语句来实现。逐步细化总是和自顶向下结合使用,一般把逐步细化看作自顶向下设计的具体体现。这种细化过程可能要逐步做下去,直到细化的模块能知道具体“怎么做了”,即能写出相应的程序为止。逐步细化的基本思想是把一个复杂问题的求解过程分阶段进行,让每个阶段处理的问题都控制在人们容易解决的范围内,在进行下一层的细化前,对本阶段的正确性进行检查,及时修改存在的问题是十分必要的。图 1.3 是计算并打印 n 个数平均值的细化后的层次图。

模块化是结构化程序的重要原则。所谓模块化就是把大程序按照功能分为较小的程序。模块是在问题细化过程中划分的,要符合人类解决问题的自然习惯,一般以按功能划分为宜,其功能应尽量单一。另外,问题细化时要注意利用已有的模块来提高编程效率。还有,为了提高程序的易读、易改、易维护性,模块划分有两点是必须考虑的:其一是耦合,即模块间相互联系的紧密程度;其二是内聚性,即模块内部成分间相互联系的紧密和相关程度。好的模块应该有松散的耦合和很强的内聚性。图 1.4 为工资管理系统模块划分示意图。

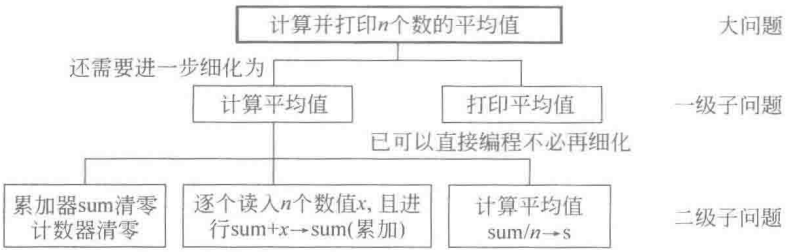


图 1.3 计算并打印平均值程序模块结构图示

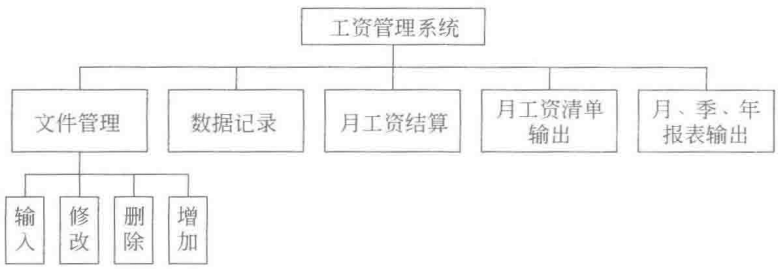


图 1.4 工资管理系统程序模块结构图示

1.4 面向对象程序设计语言

面向对象的程序设计方法(Object Oriented Programmiing,OOP)是一次程序设计方法的革命,它把设计方法从复杂烦琐的编写程序代码的工作中解放了出来,符合人的思维方式和现实世界。面向对象思想把整个世界看做是由具有行为的各种对象组成的,任何对象都具有某种特征和行为。面向对象程序设计可用于设计和维护越来越庞大和越来越复杂的软件,能满足对软件的可维护、可移植、可扩充、可重用的多项要求。

面向对象程序设计是当今众多的计算机语言中最具有特色且别具一格的一种程序设计范型,它与其他计算机语言的程序设计风格迥然不同。面向对象是一种程序设计方法学,它把软件开发过程中所处理的实体都视为对象(见图 1.5)。这些对象可组成不同类的集合,类用来刻画软件系统中所有作为基础数据的行为。出自每一类的各对象用调用该类的各方法(method)来加以处理,即发送消息(message)给这些对象,这些消息表示在该对象集合上所要采取的各种动作。对象通过消息传递与其他对象发生相互作用。面向对象语言必须具有封装性、抽象性、多态性和继承性 4 个特性。

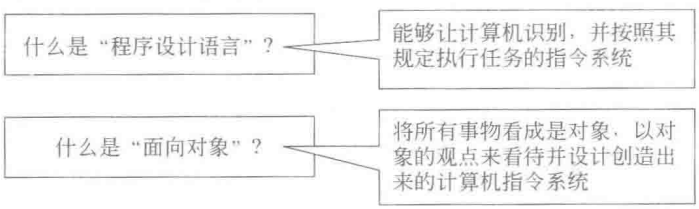


图 1.5 面向对象程序设计方法学