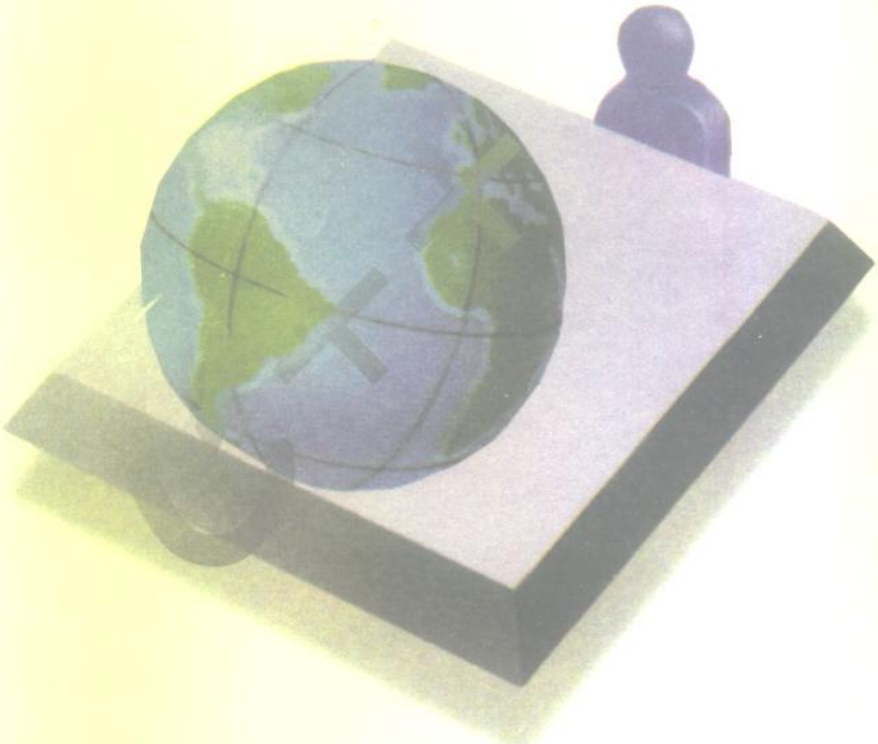
面向对象的理论与C++实践

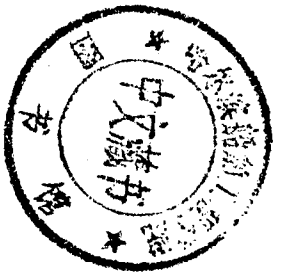
王燕 编著



清华大学出版社

面向对象的理论与C++实践

王 晓 编著



清华大学出版社

(京)新登字 158 号

内 容 简 介

本书主要介绍面向对象的理论及如何用 C++这个语言工具来实现面向对象编程。本书的特点是理论与实践紧密结合,使读者在理解理论的同时,掌握如何实际实现。

本书共分为两篇:理论篇和实践篇。理论篇共六章,主要介绍面向对象的理论。实践篇共八章,主要介绍如何用 C++实现面向对象的理论,本篇为各章设计了一个贯穿始终的实例将全书内容联系了起来。另外还简单介绍了 Windows 应用程序的面向对象实现。

本书可作为计算机专业的高年级本科生或与计算机相关专业的研究生的教材,还可以作为已经掌握了 C 语言的软件设计人员学习面向对象编程的参考书。

图书在版编目(CIP)数据

面向对象的理论与 C++实践/王燕编著. —北京:清华大学出版社,1996
ISBN 7-302-02269-0

Ⅰ. 面… Ⅱ. 王… Ⅲ. 面向对象语言,C 语言-程序设计 Ⅳ. TP312C

中国版本图书馆 CIP 数据核字(96)第 14339 号

出版者:清华大学出版社(北京清华大学校内,邮编 100084)
印刷者:北京丰华印刷厂
发行者:新华书店总店北京科技发行所
开 本:787×1092 1/16 **印张**:25 **字数**:624 千字
版 次:1997 年 2 月 第 1 版 1997 年 2 月 第 1 次印刷
书 号:ISBN 7-302-02269-0/TP・1111
印 数:0001—5000
定 价:26.00 元

前言

面向对象技术是计算机应用领域最近几年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方面的一场实质性革命。它与传统的结构化程序设计相比较，具有许多优点，但最主要的是它注意了数据与程序之间不可分割的内在联系，并把它们进行数据抽象，封装成一个统一的整体，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；大大提高了软件开发的效率和效益。

C++是一种混合型的面向对象程序设计语言。它既具有独特的面向对象特征，可以为面向对象技术提供全面支持；又具有对传统C语言的向后兼容性，很多已有的程序稍加改造就可以重用，许多有效的算法也可以继续利用。因此，目前它是实现面向对象理论与技术的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理论指导，而介绍论述面向对象技术也无法不结合对C++语言的运用。

随着经济建设和高新科技的飞速发展以及计算机应用技术的逐渐普及，科技人才的培养也在不断调整思路和规范。我国许多高等院校的计算机等专业近年来先后将《面向对象的理论与实践》列入教学计划，作为高年级专业课开设。目前专门论述面向对象的理论与技术的书籍不多，适于作高等院校教材的出版物更尚未见。一些介绍C++的书籍虽不少见，但多数只将它作为一种简单的程序设计语言对待，只着重介绍它的语法功能，而忽略它与面向对象理论的紧密联系。本书根据计算机应用专业课程教学的需要，总结与吸收国内外有关科技研究成果，试图从理论与实践的结合上介绍面向对象技术，既阐述面向对象的理论，又分析这些理论如何用C++语言来实现；对C++语言也侧重分析它的语法现象与面向对象系统各个特征的关系，使读者学完此书，既掌握了面向对象的理论，又提高用C++设计语言解决实际问题的能力。

本书分为两大部分：

第一部分为理论篇，共分为六章。重点介绍面向对象思想的由来；面向对象的基本概念；面向对象系统的各个特性；如何进行面向对象的分析和设计以及面向对象的数据库等。

第二部分为实践篇，共分为八章。重点介绍类的定义及面向对象系统的封装性；如何使用友元来访问被封装起来的类对象的私有成员；如何使用重载和虚函数概念来实现面向对象系统的多态性；如何实现继承机制；如何使用类属编程将要处理的数据的类型进行抽象，得到通用的程序版本，以及如何使用C++流库进行输入/输出流的处理。

本书在几个章节中还介绍了Windows应用程序的面向对象实现。

本书实践篇为各章设计了一个贯穿始终的实例将全书内容联系了起来。如同剥笋，结合各章节的介绍和论述，一层一层逐步地展现与分析这个实例，最后稍加补充完善，再回观全书，面向对象的理论与实践作为一个综合整体就呈现在读者面前了。

在本书的编写和出版过程中，得到了清华大学出版社的姜峰先生的热情帮助，他对本书的内容编排提出了许多宝贵的意见，使本书的结构格局更趋合理和完善。在此向他和所有支持帮助我的同志致以最衷心的谢意。

作　者

1996年4月

目 录

理 论 篇

实 践 篇

理论篇

第1章 绪 论

“面向对象”是软件程序设计中的一种新思想，由于这种新思想的引入，使我们的程序设计能更加贴近现实，并且花费更小的精力。本章主要介绍“面向对象”思想的由来和面向对象的程序设计方法。

1.1 面向对象思想的由来

“对象”一词在现实生活中经常会遇到，它表示现实世界中的某个具体的事物。

社会的不断进步和计算机科学的不断发展是相互促进的，一方面计算机科学的发展推动了社会的发展，计算机的广泛应用给整个社会生产力带来了勃勃生机；另一方面社会的发展，又给计算机科学提出了许多新的要求，计算机科学只有不断地进行自身提高和自身完善，才能适应不断进步的社会生产力的需要。随着计算机的普及应用，人们越来越希望能更直接与计算机进行交互，而不需要经过专门学习和长时间训练后才能使用它。这一强烈愿望使软件设计人员的负担越来越重，也为计算机领域自身的发展提出了新的要求。利用传统的程序设计思想无法满足这一要求，人们就开始寻求一种更能反映人类解决问题的自然方法，“面向对象”技术就是在这样的情况下产生的。

“面向对象”技术追求的是软件系统对现实世界的直接模拟，尽量实现将现实世界中的事物直接映射到软件系统的解空间。它希望用户用最小的气力，最大程度地利用软件系统来解决问题。

现实世界中的事物可分为两大部分，即物质和意识，物质表达的是具体的事物；意识描述的是某一个抽象的概念。例如“自行车”和“这辆白色的自行车”，“这辆白色的自行车”是物质，它是具体的客观存在；“自行车”是意识，它是一个抽象的概念，是对客观存在的事物的一种概括。这些现实世界中的事物可直接映射到面向对象系统的解空间，现实世界中的物质可对应于面向对象系统中的“对象”，现实世界中的意识可对应面向对象系统中的抽象概念——类。自行车在面向对象系统中可用自行车类来表达，一辆白色的自行车在面向对象系统中是一个具体的对象，是自行车类的一个实例。如图 1.1 所示。

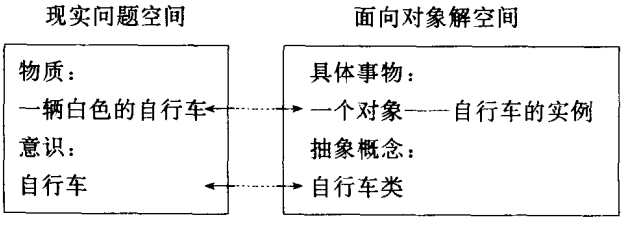


图 1.1 现实世界与面向对象系统之间的对应关系

1.2 面向对象程序设计

面向对象程序设计是软件系统设计与实现的新方法，这种新方法是通过增加软件可扩充性和可重用性，来改善并提高程序员的生产能力，并控制维护软件的复杂性和软件维护的开销。

到底什么是面向对象程序设计？在我们对它给出解释之前，需要首先讨论一下结构化程序设计。

1. 什么是结构化程序设计(Structure Programming)

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。

结构化程序设计方法虽然具有很多的优点，但它仍是一种面向数据/过程的设计方法，它把数据和过程分离为相互独立的实体，程序员在编程时必须时刻考虑所要处理的数据的格式。对于不同的数据格式即使要做同样的处理或对相同的数据格式要做不同的处理都需编写不同的程序。因此结构化程序的可重用性不好。另一方面，当数据和过程相互独立时，总存在着用错误的数据调用正确的程序模块或用正确的数据调用了错误的程序模块的可能性。因此，要使数据与程序始终保持相容，已经成为程序员的一个沉重负担。上述这些问题，结构化程序设计方法本身是解决不了的，它需要借助于我们下面要讨论的面向对象程序设计方法给予解决。

2. 什么是面向对象程序设计(Object Oriented Programming——简称OOP)

面向对象程序设计既吸取了结构化程序设计的一切优点，又考虑了现实世界与面向对象解空间的映射关系，它所追求的目标是将现实世界的问题求解尽可能简单化。

面向对象程序设计将数据及对数据的操作放在一起，作为一个相互依存、不可分割的整体来处理，它采用数据抽象和信息隐藏技术。它将对象及对对象的操作抽象成一种新的数据类型——类，并且考虑不同对象之间的联系和对象类的重用性。

例如我们可以将各种各样的自行车抽象成一个自行车类，它所包含的数据内容有架子尺寸、车轮尺寸、颜色和原材料等，它所具有的操作有转弯、移动和修理等。每一辆具体的自行车就是属于自行车类的一个对象，如图1.2所示。

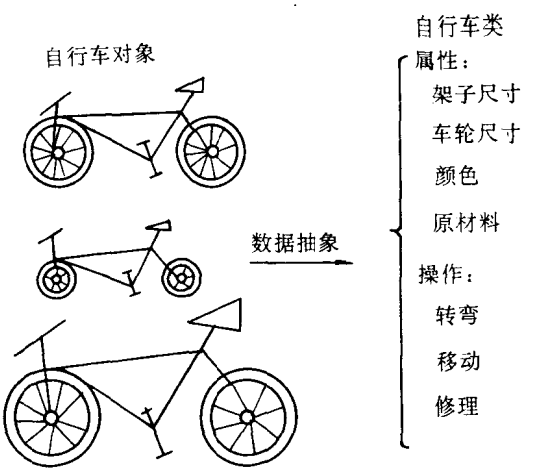


图 1.2　对象和类

面向对象程序设计优于传统的结构化程序设计，其优越性表现在，它有希望解决软件工程的两个主要问题——软件复杂性控制和软件生产率的提高，此外它还符合人类的思维习惯，能够自然地表现现实世界的实体和问题，它对软件开发过程具有重要意义。

面向对象程序设计能支持的软件开发策略有：

- 编写可重用代码；
- 编写可维护的代码；
- 共享代码；
- 精化已有的代码。

有了高质量的可重用代码就能有效地降低软件的复杂度和提高开发效率。面向对象方法，尤其是它的继承性，是一种代码重用的有效途径。开发者在设计软件时可以利用一些已经被精心设计好并且经过测试的代码，这些可重用的代码被组织和存放在程序设计环境的类库中。由于类库中的这些类的存在，使以后的程序设计过程变得简单，程序的复杂性不断降低、正确性不断加强，也越来越易于理解、修改和扩充。

1.3　面向对象程序设计语言

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待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的程序设计，这是一种把面向对象看待成编程技巧的观点，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，下面我们从几个大的家族来介绍面向对象程序语言的发展。

1. LISP 家族

LISP 是 50 年代开发出来的一种语言，它以表处理为特色，是一种人工智能语言。70 年代以来，在 LISP 基础上开发了很多 LISP 家族的面向对象语言。

(1) Flavors　它是 MIT 的 Lisp Machine 小组于 1979 年设计而成的，它的基语言是 Symbolics Common Lisp。1981，1985 年 Symbolics 公司又推出了它的新版本，它的设计目标是支持程序的模块性，便于大型复杂软件的开发，并尽可能提高运行效率。这是一个基于 LISP 的 OOP 语言版本。

(2) LOOPS　LOOPS 即 Lisp Object-Oriented Programming System，是在 InterLisp-D 环境上实现的、基于 Lisp 的 OOP 语言版本。它支持类变量、缺省值、域属性、主动值等概念，是形成 CommonLOOPS，CommonObjects 和 CLOS 语言的基础。

(3) CommonLOOPS　它是在 LOOPS 和 Flavors 的基础上实现的，其中的元对象概念有助于系统的有效实现，它还支持 OOP 的一些复杂概念，如组合对象、多重继承等。

(4) CommonObjects　它是 HP 公司经过三年(1983—1985)的努力设计实现的，是基于 CommonLisp 的 OOP 语言版本，1985 年推出新版本，新版本具有能支持多重继承、严格的封装机制、类名更换、类重定义等功能。

(5) CLOS　CLOS 即 Common Lisp Object System，它是 LISP 家族在 OOP 语言的一个分支。

2. Simula

Simula 语言是 60 年代开发出来的，在 Simula 中引入了几个面向对象程序设计语言中最重要的概念和特性，即，数据抽象的概念、类机构和继承性机制。Simula67 是它具有代表性的一个版本，70 年代发展起来的 CLU，Ada，Modula-2 等语言是在它的基础上发展起来的。

3. Smalltalk

Smalltalk 是第一个真正的面向对象程序语言，它体现了纯粹的 OOP 设计思想，是最纯的 OOP 语言。它起源于 Simula 语言。在 Smalltalk 的发展过程中推出了许多版本：

① Smalltalk-72　它是第一个正式的 Smalltalk 解释器版本。

② Smalltalk-74　它第一次加上了多窗口界面，这是 Smalltalk 系统的一大特点。

③ Smalltalk-78　在此版本中第一次采用了中间代码的设计，并用微指令实现了中间代码，大大提高了系统速度，使该系统第一次具有了可用性。

④ Smalltalk-80　是 Smalltalk 中最成功的一个版本，它在系统的设计中强调对象概念的统一，并引入和完善了类、方法、实例等概念和术语，应用了继承机制和动态连接。它被看作是一种最纯粹的面向对象程序设计语言。在其后围绕“面向对象”所进行的研究和讨论有很大一部分都是以它为基础的。

尽管 Smalltalk 不断完善，但在那个时期，面向对象程序设计语言并没有得到广泛的重视，程序设计的主流仍然为结构化程序设计语言所控制。

4. C 家族

在 80 年代，C 语言成为一种极其流行、应用非常广泛的语言。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++和 Visual C++等，每一种又有几个版本与之相对应。

除上述四大类面向对象的程序语言之外，还有一些，如 PASCAL 的面向对象版本 Turbo PASCAL，Object PASCAL，Basic 的面向对象版本 Visual Basic 等。

综观所有的面向对象程序语言，我们可以把它们分为两大类：纯粹的面向对象语言和混合型的面向对象语言。在纯粹的面向对象语言中，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言机构，它所强调的是运行效率。

1.4 面向对象方法在大型程序设计中的应用

首先我们要明确一下，什么样的程序才称为大型程序。我们讲的大型程序，并不单单指最终解决问题的程序行数，而是指待解决问题的复杂程度，当然在通常情况下，程序的大小在一定程度上反映问题的复杂程度。

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。

面向对象方法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入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。

1.5 面向对象程序设计范型

程序设计范型是指程序设计的体裁，即用程序设计语言表达各种概念和各种结构的一套设施。

当今所具有的程序设计范型有：

(1) 过程式程序设计范型　程序设计归结为选定数据结构、设计算法过程或函数。程序执行被看作各过程调用的序列。PASCAL 和 C 语言就支持此范型。

(2) 函数式程序设计范型　程序被看作“描述输入与输出之间的关系”的一个数学函数。在此范型中，完全消除状态或变量这一概念，即函数式程序设计范型是无变量的程序设计。LISP 语言就支持此范型。

(3) 面向约束程序设计范型　程序被看作“描述输入与输出之间的各关系”的一组方程。程序设计归结为列举事实、定义逻辑关系、以提问方式求解。在此范型中也消除了状态概念。PROLOG 语言就支持此范型。

(4) 面向对象程序设计范型。

(5) 进程式程序设计范型　它是一种从属于面向对象程序设计范型的范型。

(6) 类型系统程序设计范型。

(7) 事件程序设计范型。

什么是面向对象程序设计范型？从程序这一方面来讲，它是一个类的集合和各类之间以继承关系联系起来的结构，再加上一个主程序，在主程序中定义各对象并规定它们之间传递消息的规律。从程序的执行这一方面来看，它归结为各对象和它们之间以消息传递的方式进行着的通讯。面向对象程序设计最主要的特征是各对象之间的消息传递和各类之间的继承。

某一种程序设计语言并不一定与一种程序设计范型相对应，它也许是具备两种或多种范型的程序设计语言，即混合范型语言。例如 C++就不是纯粹的面向对象范型，而是过程式程序设计范型和面向对象程序设计范型的混合范型程序设计语言。

1.6 面向对象方法的思维科学基础

为了学习、掌握、使用一种程序设计语言，我们从这个语言所属范型的理论基础着手，可以对此程序设计语言做更深的理解。

面向对象程序的执行是模拟现实世界一部分或其想象的行为。

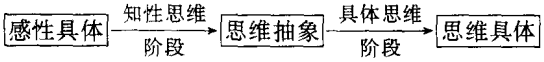
程序设计本身是人们的抽象思维和形象思维的统一，是人类的一种思维活动。

面向对象程序设计的思维活动是一种建立模型的活动。

1.6.1 建立模型

模型是用来反映现实世界中事物的特征的，任何一个模型都不可能反映客观事物的一切具体特征，它只能是对事物特征和变化规律的一种抽象，并在它所涉及的范围内更普遍、更集中、更深刻地描述客体的特征，通过建立模型而达到的抽象是人们对客体认识的深化。

建立模型的过程主要是抽象思维过程，可以把抽象思维过程概括如下：



1. 知性思维阶段

知性思维阶段是抽象思维过程的第一个阶段，它是从感性具体到思维抽象的过程。

思维是从丰富的感性材料中分解“对象”，抽象出一般规定，形成了对“对象”的某些方面、某些属性、某些特征的普遍认识。

感性具体是丰富多彩的，既具有特殊性的个别性，同时也包含一般性的个别性。在程序设计过程中，为了理解被模拟系统的复杂状态，必须分析“对象”所表现的各种现象，并且为了把握各现象的有关性质，必须进行科学抽象，即设计类及其属性和行为。

2. 具体思维阶段

具体思维阶段是从思维抽象到思维具体的过程。

抽象思维之所以要从抽象发展到具体，是因为下面两个原因：

① 从感性具体到思维抽象所获得的抽象规定，只是对“对象”的某些本质属性和特征的抽象认识，并不能全面把握这些本质属性和特征，不能揭示事物的深刻本质和规律性；

② 思维的目的在于把握具体“对象”的多样性的统一和不同规定的综合，全面把握“对象”的本质和规律，因此必须在思维中再现具体。

在程序设计过程中，在知性思维阶段，我们获得了类的抽象规定；在具体思维阶段，我们把类组织成类/子类的层次结构、继承关系。

被模拟的系统和建立的模型系统内部以及它们之间的关系如图 1.3 所示。

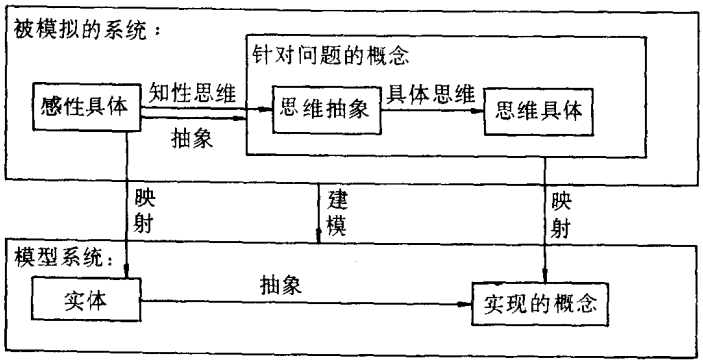


图 1.3 被模拟系统与模型间的关系