高等院校软件工程专业规划教材

C++程序设计与实践

免费提供

电子教案

白忠建　编著

机械工业出版社
CHINA MACHINE PRESS

高等院校软件工程专业规划教材

C++程序设计与实践

白忠建　编著

机械工业出版社

本书详细介绍了对象和面向对象技术的概念，并围绕案例的求解，深入浅出地介绍了面向对象技术的4个核心思想（数据封装、继承、多态和泛型编程）在C++中的概念、实现机制和语法、编程方法等，其中包括类与对象、运算符重载、继承和派生、虚函数和多态性、模板和泛型编程、多继承、名字空间和异常处理，使读者能够循序渐进地掌握C++的语法以及面向对象程序设计的方法。

本书在每一章的重要知识点之后均穿插了适量的实践题，建议读者动手实践，加深对C++的理解。

本书既可作为高等学校计算机及其相关专业相应课程的教材，也可作为C++程序员的参考书。

图书在版编目（CIP）数据

C++程序设计与实践/白忠建编著. —北京：机械工业出版社，2012.4
高等院校软件工程专业规划教材
ISBN 978-7-111-37604-0

Ⅰ. ①C… Ⅱ. ①白… 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（2012）第034026号

机械工业出版社（北京市百万庄大街22号 邮政编码100037）
责任编辑：郝建伟 常建丽
责任印制：杨 曦
北京四季青印刷厂印刷

2012年5月第1版·第1次印刷
184mm×260mm·19.75印张·490千字
0001-3000册
标准书号：ISBN 978-7-111-37604-0
定价：39.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务
社服务中心：（010）88361066
销售一部：（010）68326294
销售二部：（010）88379649
读者购书热线：（010）88379203

网络服务
门户网：http://www.cmpbook.com
教材网：http://www.cmpedu.com

封面无防伪标均为盗版

出版说明

计算机技术的发展极大促进了现代科学技术的发展，明显加快了社会发展的进程。因此，各国都非常重视计算机教育。

近年来，随着我国信息化建设的全面推进和高等教育的蓬勃发展，高等院校的计算机教育模式也在不断改革，计算机学科的课程体系和教学内容更加科学和合理，计算机教材建设逐渐成熟。“十五”以来，机械工业出版社组织出版了大量计算机教材，包括“21 世纪高等院校计算机教材系列”、“21 世纪重点大学规划教材”、“高等院校计算机科学与技术‘十五’规划教材”、“21 世纪高等院校应用型规划教材”等，均取得了可喜成果，其中多个品种的教材被评为国家级、省部级的精品教材。

为了进一步满足计算机教育的需求，机械工业出版社策划开发了“高等院校规划教材”。这套教材是在总结我社以往计算机教材出版经验的基础上策划的，同时借鉴了其他出版社同类教材的优点，对我社已有的计算机教材资源进行整合，旨在大幅提高教材质量。我们邀请多所高校的计算机专家、教师及教务部门针对此次计算机教材建设进行了充分的研讨，达成了许多共识，并由此形成“高等院校规划教材”的体系架构与编写原则，以保证本套教材与各高等院校的办学层次、学科设置和人才培养模式等相匹配，满足其计算机教学的需要。

本套教材包括计算机科学与技术、软件工程、网络工程、信息管理与信息系统、计算机应用技术以及计算机基础教育等教材系列。其中，计算机科学与技术系列、软件工程系列、网络工程系列和信息管理与信息系统系列是针对高校相应专业方向的课程设置而组织编写的，体系完整，讲解透彻；计算机应用技术系列是针对计算机应用类课程而组织编写的，着重培养学生利用计算机技术解决实际问题的能力；计算机基础教育系列是为大学公共基础课层面的计算机基础教学设计的，采用通俗易懂的方法讲解计算机的基础理论、常用技术及应用。

本套教材的内容源自致力于教学与科研一线的骨干教师与资深专家的实践经验和研究成果，融合了先进的教学理念，涵盖了计算机领域的核心理论和最新的应用技术，真正在教材体系、内容和方法上做到了创新。同时，本套教材根据实际需要配有电子教案、实验指导或多媒体光盘等教学资源，实现了教材的“立体化”建设。本套教材将随着计算机技术的进步和计算机应用领域的扩展及时改版，并及时吸纳新兴课程和特色课程的教材。我们将努力把这套教材打造成为国家级或省部级精品教材，为高等院校的计算机教育提供更好的服务。

对于本套教材的组织出版工作，希望计算机教育界的专家和老师能提出宝贵的意见和建议。衷心感谢计算机教育工作者和广大读者的支持与帮助！

机械工业出版社

前　言

C++是一门非常优秀的面向对象程序设计语言，它不仅继承了C语言的全部优点，同时实现了面向对象技术的所有核心概念，使它成为很多程序员，尤其是从C转过来的程序员开发大型复杂软件的首选语言。

自从贝尔实验室的Bjarne Stroustrup博士研发出C++后，这门混合型的语言从最初的粗糙、低效演化到现在的精致和高效，尤其是在标准化后。标准化的C++更加规范、易用。此外，STL被接纳为标准库中的成员是非常令人鼓舞的，这使得C++在编写标准的、跨平台的应用中表现得更加自如。

与其他的面向对象程序设计语言相比，如Java和C#，C++最为人诟病的一个问题就是它不够纯粹。的确，C++的混合编程模式（面向过程和面向对象）有时确实让人感到有些困惑，但纯粹并非总是优点。“C++的强项恰恰在于它支持多种有效的编程风格（多种思维模型）以及它们之间的相互组合。最优雅、最有效，也最容易维护的解决方案常常涉及不止一种风格（编程模型）。如果一定要用吸引人的字眼，可以说，C++是一种多思维模型的语言。在软件开发的庞大领域中，需求千变万化，至少需要一种支持多种编程风格的通用语言，而且很可能需要一种以上。”（Bjarne Stroustrup语）。

C++语言中最吸引程序员的是它的泛型编程（Generic Programming）机制。该机制为程序员提供了编写类型无关的通用代码的强有力工具，从而使程序员能够从容应对现代大型复杂软件的编写任务。本书对这一特性做了详细的介绍。

本书的内容分为3个部分。

第1部分：1～4章，主要讲解C++的基础语法。

第2部分：5～10章，主要讲解面向对象技术在C++中的实现。

第3部分：11～13章，主要讲解面向对象技术中的高级话题。

作为讲解面向对象技术的教材，本书重点强调了面向对象技术的4个核心概念：数据封装、继承、多态和泛型编程，而对基础语法部分的介绍着墨不多。这就要求阅读本书的读者应该系统学习过C语言，或者具有一定的C语言编程经验。如果这对读者阅读本书造成了困扰，敬请谅解。

本书在讲解C++的各项知识点时，是通过一个贯穿全书的案例来展开的，并在第4章后的每一章后都附有局部解决方案。由于解决方案并不完整，因此笔者强烈建议读者在学习时，按照笔者的思路将方案补充完整，并上机实践，这样才能将面向对象技术（而不仅仅是C++的语法特性）掌握得更加牢固。此外，后一章的内容都会引用前一章的解决方案，所以，为了能充分理解每个章节的内容，请读者在完成（至少研究过）解决方案后，再进行后续章节的学习。

虽然笔者在高校从事了多年的C++教学以及C++应用研发，但仍然对这门与时俱进的语言有不能把握的地方。如果读者在书中发现错误，敬请指正，笔者不胜感激。如有高见，请发邮件至：baizj@ uestc. edu. cn。

编　者

目　录

第1章 引 论

本章要点

- 对象的概念。对象是一个主动的实体，是面向对象技术的基础概念。
- 面向过程技术与面向对象技术的异同。面向过程技术以过程（如函数和子例程等）为中心；而面向对象技术以对象为中心。
- 面向对象的核心概念。数据封装、继承和多态是每一种面向对象的程序设计语言必须实现的核心概念。现代的观点认为泛型编程也是核心之一。
- C++程序的概貌。C++程序与C程序有相似之处，但也有很大的不同，输入和输出的区别就是其中之一。

C++语言是面向对象的程序设计语言，它所支持的面向对象的概念容易将问题空间直接映射到程序空间，为程序员提供了一种与传统的结构程序设计十分不同的思维方式。因此，学习C++语言面临两个问题：

- 如何建立面向对象的思维方式?
- 如何用C++语言实现面向对象程序设计?

本章试图从对象的概念、面向对象的核心概念和面向对象程序设计等几个方面，给读者以面向对象概念的总体印象，同时抛出一个贯穿全书的案例，并在其后各章给出该案例的局部解决方案，用以串起C++的关键知识点。

1.1 什么是对象

学习一种计算机程序设计语言，其目的是为了能使用它来解决实际问题。解决方案的第一个步骤中，往往需要将实际问题进行抽象，建立起一个可行可解的模型，然后在其后的步骤中将其转换为计算机模型，最后用计算机语言编码实现。因此，下面就从观察现实世界开始学习之旅。

在现实的世界中，我们时时刻刻会面对一些客观实体，如一个叫Ken的同事、一栋大楼、一台计算机和一盆植物等，如图1-1所示。这些客体都拥有不同的特性，还拥有独特的行为，构成了外部世界。而我们，作为这些客体中的一员，会与其他客体进行交流，或者请求别的客体提供帮助。这些不依赖于人类意识而存在的客体称为“对象（Object）”。

单个的对象是独立存在的，然而却不是孤立的。很难想象一个由孤立对象构成的世界，那将是一片死寂。实际上，对象之间存在着一张复杂的关系网，而网中的对象随时随地都在发生信息交流，它们之间互相构成了服务与被服务的关系。可以这么说，对象加上对象之间的关系就构成了现实世界，如图1-2所示。

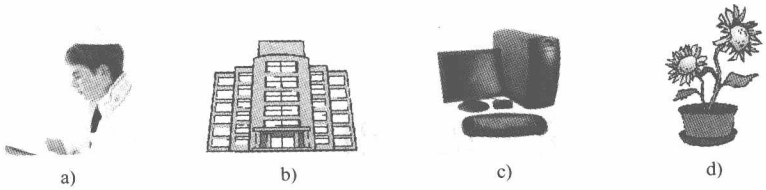

图 1-1　现实世界中对象的例子

a）Ken　b）一栋大楼　c）一台计算机　d）一盆植物

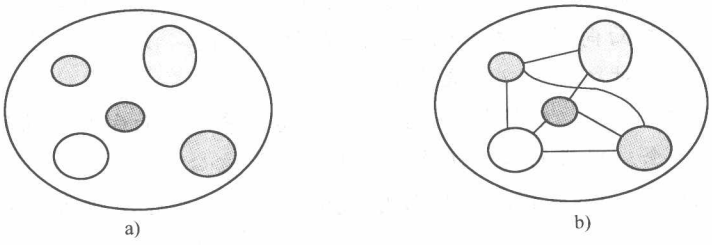

图 1-2　对象和对象间的关系构成现实世界

a）孤立对象构成的世界一片死寂　b）对象间的联系形成复杂的信息网络

现实生活中的对象总是以群体的方式出现的，正是所谓的“人以群居，物以类聚”。尽管群体中的对象具有鲜明的个性，但同一群体中的所有对象具有相似的共性和行为模式。分类学上会根据对象个性和行为的相似性而将某些对象划分在一个分类当中，然后用一个抽象的概念描述这个分类。例如，世界上所有的张三、李四、Linda 和 Peter 等就构成了“人”这个群体。这个抽象概念是在总结了大量同类对象的特性和行为模式，提炼出其中的共同特性的基础上得到的。可以看出，抽象描述了共性，属于这个抽象类别的具象（个体）无条件地拥有这些共性，而具象同时还拥有各自不同的个性。

抽象和具象有直观的辩证关系。例如，当我们听到“人”这个词，脑海中会立即浮现出脑袋、躯干和四肢组成的一种直立行走的生物，并且能说会道，会思想，这些正是“人”这种物种的外部特征。但这个“人”面部模糊，还不具有具象，因为这还不是一个对象。当提到张三时，他的音容笑貌、行为举止就立刻附着在那个模型人身上，此时“人”的概念就被具体化了，形成了一个独特的对象。从这个例子可以看出，抽象是所有对象的模板，而对象是抽象的一个具体实例。

在现实中，我们用属性和行为这一静一动两个术语来描述抽象的特性。例如，抽象人拥有姓名、年龄和性别等静态属性，拥有思维、行走和说话等动态行为，而所有这些都不能用具体的值或方式来描述。只有在描述一个特定的对象时，人的属性和行为才会具体化。例如，对象张三，年龄 22，性别男，按某种非常有张三特色的方式思维、行走和说话。这样一来，对象就真正活起来了。图 1-3 示意了抽象和具象的关系。

一个对象是一个主动的实体，它能够主动发起动作，从而引起它内部状态的改变。一个显而易见的例子就是，张三是用他自己的双腿靠自己的意识支配行走的，而一旦走了几步后，他所处的位置就发生了改变。

对象和其他对象是有联系的，它们之间要产生互动，从而驱动问题向能够解决的方向发展。例如，在张三与李四的交流过程中，张三在讲，李四在听、在想；或者二者交换角色。

总之，一方发起一个动作，刺激另一方发起响应动作，这样一来一往，双方的交流就得以顺利进行。

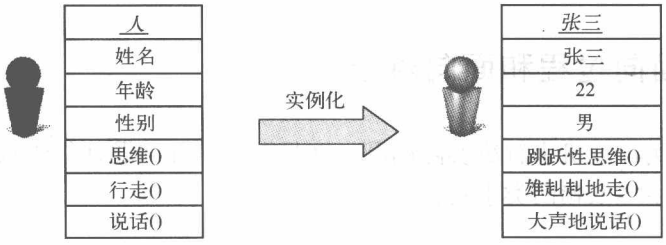

图 1-3　抽象和具象（对象）的关系

从以上观点出发，可以发现，真实世界中充满了对象，并且一切皆可成为对象。这对我们处理问题有非常重要的指导意义。

在理解了现实世界中的对象后，我们转过来观察在计算机世界中是怎样处理对象这一概念的。

我们都知道，应用计算机的根本目的就是为了解决现实问题。而现实世界是与计算机世界有差异的。不过，有一点是非常肯定的，就是我们试图利用计算机技术仿真真实问题的解决方案。既然是仿真，那么为现实问题建造的抽象问题模型就和计算机抽象模型有非常直接的联系。也就是说，现实世界中的抽象一定会用某种计算机技术来描述，而现实对象一定能在计算机模型中找到其映像，不过，这个映像必须用计算机能够理解的方式存在。

现实的对象是一种存在，那么这个对象在计算机中的影响也必须是一种存在。这很自然地令我们联想到了计算机中能够体现“存在”这一概念的机制：内存。显然，现实对象的映像肯定是被保存在内存中的，否则它只能是一个虚无缥缈的概念，无法操控。基于此，我们可以简单地为计算机处理的对象下一个定义：一段带有特定类型的内存。这个定义告诉我们以下 3 个事实：

1）对象要占据内存。

2）对象属于某种类型。类型是一种重要信息，它描述了对象要占据多大的内存、对象的细节在这段内存中如何分布，以及对象能够参与的运算。

3）对象在大多数情况下是可以被改变的；不能改变的对象为某种特定的目标存在。

根据以上描述，读者可能很容易地联想到程序设计中常用到的“变量”，因为变量实实在在拥有上述特征。基于此，可以说这些变量就是对象。只不过，由于变量缺乏主动的行为，因此它还只能是一个简单对象。

有了对象的概念后，解决问题的计算机程序就会围绕对象来进行设计。不过，不同的计算机程序设计技术对对象这个概念会有不同的看法，因而设计模式也会有所不同。目前，我们通常采用的设计技术有两种：面向过程的和面向对象的设计技术，这两种技术用两类对应的程序设计语言实现。

但无论采用哪种技术，无论使用何种语言，解决问题永远是所有计算机程序唯一的目标。

习题 1.1

除了书中的例子，在现实世界中，还能发现哪些事物可以称为对象？

1.2 什么是面向过程和面向对象

通常，人们利用计算机来解决实际问题。在早期，实际问题往往以数值计算为主体。我们知道，数值计算主要依赖两类事物：

- 数据
- 计算方法

对以上两者关系的处理方式衍生了多种程序设计方法，而其中最流行的，就是面向过程和面向对象的方法。

1.2.1 面向过程方法

面向过程的观点以计算方法为重。这显然是一种注重过程的观点，也就是这种程序设计方法得名的缘由。

在早期的程序设计技术中，对过程的仿真成为最主要的关注点。这种观点在一些程序设计语言，如 C 语言中得以体现。在这些语言的程序中，最显著的语法成分是“过程/子例程（procedure/subroutine）”，又称为“函数（function）”，它们是程序的主宰。与此对应的，待处理的数据并没有仿真实际对象，而是退化成仅包含基本属性的最小数据包，其中没有包含对象应有的行为。实际上严格地说，对象的行为还是存在的，只不过它们从对象中分离出来了，成为一个个独立的过程。当对象要发起一个动作时，一般通过过程（函数）调用（procedure/function call）来完成，对象是这个过程的一个参数（parameter）。通俗地说，就是动作作用于对象之上。这样一来，对象从一个主动的实体“沦落”为一个被动的附属品，人“行走”变成了“被行走”。这或多或少显得有些不自然。

过程和函数最直观的区别在于：函数一般要计算出一个（或一些）结果，并且要将结果返回到调用这个函数的地方；而过程往往不会返回结果。

面向过程技术将对象及其行为截然分开的特性有其合理性，但存在着相当大的弊端：

1）描述对象特性的数据包没有任何或者只有很弱的保护措施。也就是说，任何人可以直接访问数据包中的元素，而不需要任何特殊手段。现实的情况是：对象的有些属性是应该被隐蔽的、外部不可见的，一个明显的例子就是员工的薪资。

2）对象的属性和行为之间的联系非常松散，这降低了客户程序员（即那些使用数据包的程序员，而他们不是数据包的创建者）对整体逻辑的可理解程度。

3）如果现实模型和面向过程的计算机模型之间存在映射关系，那么这个映射关系是有些“扭曲”的，正如图 1-4 所示描述的那样。

在程序设计方面，面向过程程序设计采用了“自顶向下，逐步求精”的方式，模块化（或结构化）成为其最重要的思维方法，其具体设计步骤为

1）整个软件系统功能逐步细化为多个小的功能。