全国电子信息类优秀教材

C++面向对象程序设计（第3版）

杜茂康　谢　青　主编

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

全国电子信息类优秀教材

C++面向对象程序设计（第3版）

杜茂康　谢　青　主编

電子工業出版社
Publishing House of Electronics Industry
北京 • BEIJING

内 容 简 介

本书以 C++ 11 标准为指导，深入浅出地介绍了标准 C++面向对象程序设计的相关知识，以及用 Visual C++进行面向对象的 Windows 程序设计的基本原理和方法，包括 C++对 C 语言的扩展、类、对象、友元、继承、多态性、虚函数、重载、I/O 流类库、文件、模板与 STL、C++ Windows 程序的结构、消息驱动、MFC 应用程序框架、GDI、菜单、对话框、工具栏、文档与视图等内容。

全书本着易于理解、实用性强的原则设计其内容和案例，并以一个规模较大的综合性程序的编制贯穿于 C++面向对象技术和 Windows 程序设计的全过程，引导读者理解和掌握面向对象程序设计的思想、技术、方法，以及在 Windows 程序中应用自定义类实现程序功能的软件开发方法。

本书取材新颖，内容全面，通俗易懂，可作为高等院校计算机、电子信息类专业及其他理工类相关专业的教材，也可作为 C++语言自学者或程序设计人员的参考用书。

图书在版编目（CIP）数据

C++面向对象程序设计 / 杜茂康，谢青主编. —3 版. —北京：电子工业出版社，2017.6
ISBN 978-7-121-31583-1

Ⅰ. ① C… Ⅱ. ① 杜… ② 谢… Ⅲ. ① C 语言－程序设计－高等学校－教材 Ⅳ. ① TP312.8

中国版本图书馆 CIP 数据核字（2017）第 116191 号

策划编辑：章海涛
责任编辑：章海涛　　特约编辑：曹剑锋
印　　刷：三河市鑫金马印装有限公司
装　　订：三河市鑫金马印装有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　　邮编 100036
开　　本：787×1 092　1/16　　印张：26.5　　字数：740 千字
版　　次：2007 年 5 月第 1 版
　　2017 年 6 月第 3 版
印　　次：2017 年 6 月第 1 次印刷
定　　价：52.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：192910558（QQ 群）。

第 3 版前言

面向对象编程技术降低了软件开发的复杂度，提高了软件开发的效率，让开发人员开发出高可靠性、可重用和易维护的软件，是当前软件开发的主流技术，是每个软件开发人员必须具备的技术基础。C++语言是在 C 语言基础上扩充了面向对象机制而发展起来的一种程序设计语言，程序结构灵活，代码简洁清晰，可移植性强，支持数据抽象、面向过程和面向对象程序设计。C++语言因其稳定性、高效性、兼容性和可扩展性而被广泛应用于各种领域和系统中，常被用来设计操作系统（如 UNIX、Windows、Apple Macintosh）、设备驱动程序或者其他需要在实时约束下直接操作硬件的软件。图形学和用户界面设计是使用 C++语言最深入的领域，银行、贸易、保险业、远程通信以及军事等诸多应用领域也常用 C++语言设计其应用程序的核心代码，以求软件的最佳性能和开发效率。

无论从编程思想、代码效率、程序的稳定性和可靠性，还是从语言本身的实用性来讲，C++语言都是面向对象程序设计语言的典范。学好 C++语言，不但能够用于实际的程序设计，而且有助于理解面向对象程序设计技术的精髓，再学习诸如 Java、C#、Python 之类的面向对象程序设计语言也就简单了。

多年的教学和编程实践经验给本书作者的真切体会是“**读教科书明其理，看技术书知其用**”。教科书的原理剖析和技术书的案例分析相结合有利于读者深刻地理解和掌握 C++语言程序设计的基本原理和技术，有利于读者将学到的技术用于实际的软件开发中。

本书基于这样的认知而编写，兼具 C++技术书籍和教材的特点，既比较深刻地介绍 C++面向对象的程序技术和原理，又清晰地介绍 Windows 平台下的 C++程序实现方法，且通过程序实例将两者较好地结合在一起。**书中基于 DOS 平台精心设计了一个贯穿全书大部分章节的规模较大的专业课程类管理程序 comFinl**，并不断地利用面向对象的 C++程序技术扩充该程序的功能，使之成为一个较为完整的综合程序，最终将它从 DOS 平台移植到 Windows 系统中，成为一个 Windows 应用程序。读者可借此掌握 C++应用程序的设计方法，以及将基于 DOS 平台的自定义类移植到 Windows 程序中的方法和过程。

自 2007 年第 1 版，本书受到了广大师生和软件开发人员的好评，得到了多所高校的认可，被选为教材，重印多次。许多读者发来求解书中疑问或习题参考答案的邮件，一些软件开发人员与作者探索了将 C++类移植到 Windows 程序中的方法，也有读者指出了书中的错误和缺限。这些是本书得以进步和持续发展的源泉。这次修订主要体现在以下几方面：

（1）以 C++ 11 标准为蓝本修订了各章节的内容。增加了 C++ 11 标准提出的新特征，包括智能指针、lamada 函数、移动对象、构造函数的继承和委托、对象初始化列表、override 和 final、pair、tuple 容器、noexcept 异常等内容，删除了过时及部分较难且不实用的内容，并对一些程序案例进行了重新设计。书中用“**11C++**”字样对首次出现的 C++ 11 新标准进行了标识，这些内容不能在 VC++ 6.0 这样的早期编译器中进行编译，必须在支持 C++ 11 规范的编译环境中才能够正常运行。

（2）注重面向对象程序设计和分析能力的培养。在介绍类、继承和多态设计时更加重视对类设计的分析，并用 UML 方法进行建模，更利于面向对象编程思维的培养与形成。

（3）按 C++ 11 新标准修订了全书例程，并在 Visual C++ 2015 环境下进行了运行测试。

本次修订保留了第 2 版的整体结构，全书共分为 12 章。第 1～2 章介绍 C++语言的基础知识。第 1 章介绍面向对象程序设计的主要特征、C++程序的结构、string 类型，数据输入/输出，以及 Visual C++ 2015 编程环境；第 2 章介绍 C++ 11 标准对 C 语言非面向对象方面的扩充，主要包括智能指针、const 和 constexpr 常量、左值和右值引用、类型转换、lamada 表达式、范围 for、函数重载、内联函数、作用域、命名空间及 C++文件操作。

第 3～9 章介绍 C++面向对象程序设计的思想、特征和方法，包括类和对象、继承和派生、虚函数、运算符重载、模板和 STL 程序设计、异常、文件和 I/O 流等内容。

第 10～11 章介绍 Visual C++ Windows 程序设计的原理和方法。第 10 章介绍 C++ Windows 程序设计的基础知识，包括 Windows 程序设计的常用数据结构、程序运行原理、消息驱动、API 程序设计等内容；第 11 章介绍 MFC 应用程序框架的设计原理和方法，包括事件函数、对话框、控件、GDI、菜单和工具栏设计等内容。

第 12 章介绍将第 4～9 章逐步完善的基于 DOS 平台的课程管理程序 comFinal 移植到 Windows 程序中的方法。在 MFC 向导创建的应用程序框架中逐步引入在 DOS 平台下完成的多个自定义类，并通过事件函数、对话框、工具栏、菜单调用这些自定义类的对象，示范了在 Windows 程序中操作自定义类、开发 Windows 应用软件的方法。

本书内容全面、析理透彻、注重实用，精心设计了易于理解和有代表性的示意图和案例程序，深入浅出地展示了 C++面向对象程序设计的原理和各种技术，并对面向对象编程过程中容易发生的误解和错误进行重点分析，颇具启发性。

本书由杜茂康、谢青、李昌兵、王永、刘友军和袁浩编写。杜茂康编写了第 1、2、3、4 章，谢青编写了第 5、6、7、8 章，李昌兵编写了第 9 章，王永编写了第 10 章，袁浩编写了第 11 章，刘友军编写了第 12 章，全书由杜茂康审校和统稿。

本书在编写过程中得到了不少专家、学者、老师和同事的指导、支持和帮助，2004 级信息管理与信息系统专业两位热爱程序技术的学生李明闯和王晓润仔细地阅读了本书第 1 版初稿中的全部内容，校正了初稿中的许多错误，广大读者也指正了本书前两版的错误和不当之处，并提出了许多有用的建议。在此，谨向他们表示诚挚的感谢！

在本书的编写过程中阅读参考了国内外大量 C++程序设计的相关书籍，这些书籍已被列在书后的参考文献中，在此谨向这些书籍的作者表示衷心感谢！

面向对象程序设计是一项不断发展变化的程序技术，C++语言更是博大精深，其标准和规范也在不断更新，鉴于作者才疏学浅，水平有限，加之经验不足，书中一定存在不少错误和不当之处，恳请专家、同行和读者批评指正。

为了便于读者学习和教师教学，本书配有以下教学资源：全部例题的程序代码、部分习题的程序代码、配套的电子课件。有需要者可从华信教育资源网（http://www.hxedu.com.cn）上进行下载。

作　者

目　录

第 1 章　C++与面向对象程序设计概述

面向对象程序技术用对象来模拟客观世界中的事物及其行为，用消息传递来模拟对象之间的相互作用，使程序与客观世界具有很大程度的相似性，降低了软件开发的难度，提高了软件开发的效率，适合大型的、复杂的系统开发。

本章介绍面向对象程序设计语言的基本特征、C++的数据类型、简单的 C++程序设计和数据输入、输出方法。

1.1　面向过程和面向对象程序设计

1．面向过程程序设计

早期计算机程序的规模较小，主要开发方式为个人设计、个人使用，没有什么组织原则，只需将相应的程序代码组织在一起，再让计算机执行它，就可以完成相应的程序功能。随着计算机的普及和技术发展，程序的规模和复杂度越来越大，到了 20 世纪 60 年代初期，个人软件开发方式已不能满足要求，出现了许多问题，如软件开发费用超出预算，不能按期完成软件开发，质量达不到要求，软件维护困难等。这就是所谓的软件危机。

软件危机表明个人手工编程方式已经跟不上软件开发的需求了，迫切需要改变软件的生产方式，提高软件生产率。20 世纪 60 年代末出现了影响深远的结构化程序设计（Structure Programming，SP）思想。结构化程序设计采用“自顶向下、逐步求精、模块化”的方法进行程序设计，即采用模块分解、功能抽象、自顶向下、分而治之的方法，将一个复杂、庞大的软件系统分解成为许多易于控制、处理、可独立编程的子模块。各模块可由结构化程序设计语言的子程序（函数）实现，子程序则由顺序、分支、循环三种基本结构编码完成。其基本特点是：① 按层次组织模块；② 每个模块只有一个入口，一个出口；③ 代码和数据分离，即**“程序=数据结构+算法”**。

结构化程序设计是一种以功能为中心的面向过程程序设计方法，首先将要解决的问题分解成若干功能模块，再根据模块功能设计一系列用于存储数据的数据结构，并编写一些函数（或过程）对这些数据进行操作，最终的程序是由许多函数（或过程）组成的。

在结构化程序设计中，数据和过程被分离为相互独立的程序实体，用数据代表问题空间的客体，表达实际问题中的信息；程序代码则是用于体现和加工处理这些数据的算法。在设计软件时，必须时时考虑所要处理数据的结构和类型，对不同格式的数据做相同的处理，或者对相同格式的数据做不同的处理，必须编写不同的程序，代码的可重用性较差。

此外，数据和操作过程的分离还会导致程序的可维护性也较差，原因是数据结构改变时，所有与之相关的处理过程都要进行修改，增大了程序维护的难度。

例如，假设实现一个通讯录管理软件，程序员编写了 4 个函数：inputData()，searchPhone()，printData()，searchAddr()，分别用于实现通讯录数据的输入、输出和查询功能。许多技术可以实现通讯录的数据存取，如数组、链表、队列等。一个采用数组存取数据的程序基本结构如下：

```
struct Person{                          // 用于存放个人信息的数据结构
    char name[10];
    char addr[20];
    char phone[11];
}
Person p[100];                          // 保存所有个人信息的全局数组
int n=0;                                // 用于保存实际人数的全局变量
void inputData(){…}                     // 初始化全局数组P，读入每个人的姓名、地址和电话
void searchAddr(char *name){…}          // 根据姓名查找地址
void searchPhone(char *name){…}         // 根据姓名查找电话号码
void printData(){…}                     // 打印输入每个人的姓名、地址和电话
```

4个函数通过全局数组（即P）共享数据，并且相互影响。如果将这些函数提供给其他程序员使用，就必须让该程序员知道他不能定义和修改全局数组（即P），只能通过这4个过程存取全局数据P。

个人通信管理程序代表了面向过程程序设计的基本编程方法：先定义一些全局性的数据结构，然后编写一些过程对这些数据结构进行操作，其模型如图1-1所示。

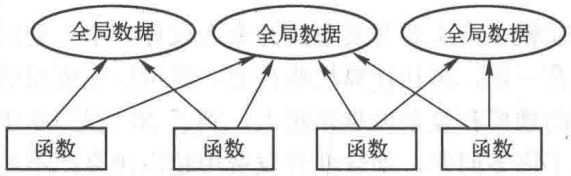

图1-1　结构化程序设计模型

从图1-1的程序模型可以看出，数据与函数之间存在潜在的连接关系。某个全局数据的修改可能引起大量操作该全局数据的函数的修改。此外，若某个函数意外修改了某个全局数据，很可能引起程序数据的混乱。例如在个人通讯录管理程序中，Person的变化会引起操作它的所有函数（如inputData()、searchAddr()等）的修改。此外，谁也没有办法限制其他程序员定义与全局数据同名的变量（如数组P），也不能限制他修改全局数组的值。当程序规模较大时，这个问题尤其突出，软件维护困难。

支持结构化程序设计的高级语言称为结构化程序设计语言，它们提供了顺序、分支和循环三种基本结构，支持面向过程的程序设计。C、Fortran、Basic等都是当前仍在广泛使用的面向过程的程序设计语言。

2．面向对象程序设计

随着计算机和网络技术的发展，软件应用的领域更加广泛，需求越来越大。同时，软件的规模和复杂度也在不断增加，升级改版的时间要求更短，面向过程程序设计技术已不能满足软件开发在效率、代码共享和更新维护等方面的需求了，取而代之的是面向对象的程序设计技术。

面向对象就是指以对象为中心，分析、设计和构造应用程序的机制，其基本观点是：计算机求解的都是现实世界中的问题，它们由一些相互联系且处于不断运动变化的事物(即对象)组成，如果在计算机中能够用对象描述问题域中的各客观事物，用对象之间的关系描述客观事物之间的联系，用对象之间的通信描述事物之间的相互交流及相互驱动，就能够将客观世界中的问题直接映射到计算机中，实现计算机系统对现实环境的真实模拟，从而解决问题。

这里涉及三方面的问题：一是如何把客观事物表示为计算机中的对象；二是如何用对象之间

的关系反映客观事物之间的联系；三是如何用对象之间的作用反映客观事物之间的交流与驱动。

对于第一个问题，面向对象技术的解决方法是：对于任何一个客观事物，用数据表示它的特征，用函数描述它的行为，并把两者结合成一个整体，称为对象，代表一个客观事物。在此可以看出，一个对象由数据和函数两部分构成。数据常被称为数据成员，函数则被称为成员函数。一个对象的数据成员通常只能通过自身的成员函数修改。

对象真实地描述了客观事物，它将数据和操作数据的过程（函数）绑在一起，形成一个相互依存、不可分离的整体（即对象），从同类对象中抽象出共性，形成类。同类对象中的数据原则上只能用本类提供的方法（成员函数）进行处理。

对于第二个问题，面向对象技术提供了继承、对象成员、对象依赖等机制来描述客观事物之间诸如父子关系、汽车与其组成部件的包含关系、某人与他的宠物狗之间的依赖关系……

对于第三个问题，则通过对象之间的消息传递机制表示客观事物之间的交流与驱动关系。禁止一个对象以任何未经允许的方式修改另一个对象的数据，如果它需要向另一个对象传递数据，或者得到它的服务，可以向该对象发送信息，对方会响应消息，执行特定函数来完成消息发送者的操作要求。

面向对象程序技术能够实现对客观事物的自然描述，反映客观世界的本来面目，使程序模块化设计更简单、自然，其基本模型如图 1-2 所示。

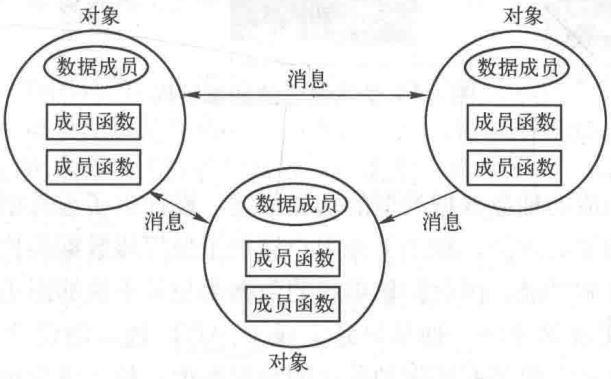

图 1-2　面向对象程序设计的程序模型

面向对象程序技术提高了软件的可靠性、可重用性、可扩展性和可维护性。因为某类对象数据的改变只会引起该类对象程序代码的改变，而与其他类型的对象无关，这就把程序代码的修改维护局限在了一个很小的范围内。由于数据和操作它的函数是一个整体，因此易被重用。在扩展某个对象的功能时，不用考虑它对其他对象的相互影响，软件功能的扩展更容易。

1.2　面向对象程序语言的特征

面向对象程序设计语言经历了一个较长的演变过程，20 世纪 50 年代的 LISP 语言就引入了信息隐藏和封装机制，60 年代的 Simula 语言提出了抽象和封装，引入了数据抽象和类的概念，被认为是第一个面向对象语言（但它不具备面向对象语言的全部特征）；70 年代的 Smalltalk 则是第一个真正面向对象的程序设计语言。现在广泛使用的 C++、Eiffel、Object-C、Visual Basic、C#、Java 等都是面向对象的程序设计语言。面向对象程序设计语言具有以下几个基本特征。

1．抽象

抽象（abstract）是指有意忽略问题的某些细节和与当前求解问题无关的方面，把事物的主要

特征抽取出来，并用它来描绘客观事物。抽象的结果是形成对应客观事物的抽象数据类型，简称ADT，即 Abstract Data Type。

在现实中，人们常从特征和行为两个方面描绘客观事物，抽象也是如此。它从现实中的客观事物出发，分析同类事物应当具备的共同特征和行为，然后将它们抽取出来形成描绘该类事物的概念，是一个从个性到共性的过程。比如，要形成学生的抽象概念，就要观察现实中的各位同学，忽略他们各自的独特爱好和才能，抽取出他们作为学生的共性和行为特征，就形成了学生的抽象数据类型，如图 1-3 所示。

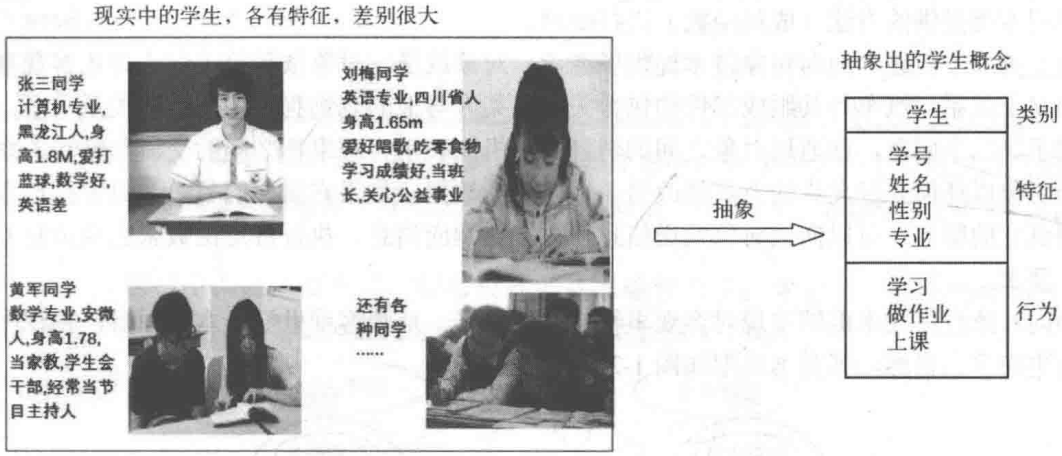

图 1-3　学生概念的抽象过程

2．封装

抽象只是给出了对形成的抽象数据类型的功能描述，设计出了该类型应该为其用户提供的各项功能，以及使用这些功能的方法，相当于为用户提供了使用该数据类型功能的接口。但抽象并没有实现任何具体的特征和功能，因此抽象形成的数据类型还不能够用于程序设计。

抽象导致了接口与实现的分离。抽象只是完成了 ADT 接口的设计，而具体实现由封装完成。封装（encapsulation）就是包装并实现抽象出的数据类型，使之成为可用于程序设计的抽象数据类型的过程。

封装是面向对象技术的重要特征，它将抽象出的特征（用数据表示）和行为（用函数表示）捆绑成一个整体，并且编码实现抽象所设计的接口功能。封装之后的 ADT 由接口和实现两部分组成。接口在外，描述了抽象类型显示给用户的外部视图，而抽象数据类型的结构和接口功能的实现细节则被封装隐藏，用户对此一无所知，也不需知道，因为这并不影响对该功能的使用。反之，封装后的抽象数据类型更便于使用，因为用户不会被复杂的内部结构和实现细节所干扰，只需向接口传递正确的参数，就能够使用所需的功能。

抽象和封装是认知客观事物并把它表示成可用于程序设计的抽象数据类型的两个阶段，抽象完成抽象数据类型的整体设计，封装则实现设计。比如要做手电筒，抽象阶段的任务是了解现实，设计出手电筒的大小、形状、颜色、灯泡的规格，以及便于用户更换灯泡、电池和开关电路的三个接口，描述好接口的外形和功能。封装的任务就是按照抽象的设计结果，通过具体的电路和电子开关实现各接口的功能，并把具体的实现细节包装进电筒内部不让人们知道，只留操控开关（接口）在外，如图 1-4 所示。人们只能够通过允许的接口使用手电筒，电不足了或灯炮坏了只能通过指定的接口更换，要用光的时候打开电筒开关，不用的时候断开它。而封装在电筒内的电路和连接电筒各部件的电子线路则不让人们知道和操控。

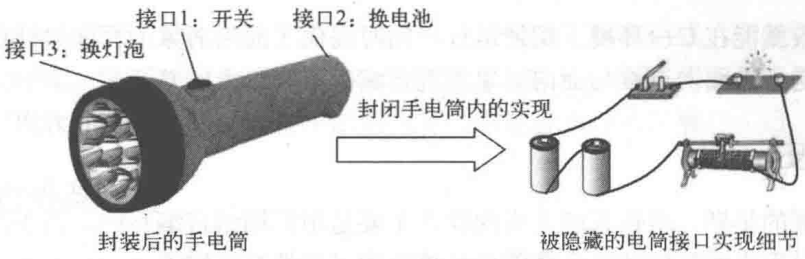

图 1-4　手电筒封装

面向对象程序用 class 实现封装，封装之后的抽象数据类型称为类，面向对象程序设计的主要任务就是对问题空间的各类客观事物进行抽象，构造出代表问题空间各类事物的 class。

3. 继承

继承源于生物界，通过继承，后代能够获得与其祖先相同或相似的特征与能力。面向对象程序设计语言也提供了类似于生物继承的语言机制，允许一个新类从现有类派生而来，新类能够继承现有类的属性和行为，并且能够修改或增加新的属性和行为，成为一个功能更强大、更能满足应用需求的类。

继承是面向对象程序设计语言的一个重要特征，是实现软件复用的一个重要手段。在面向对象程序设计中，如果一个类 B 继承了另外一个类 A，则称类 B 为子类（subclass），称类 A 为超类（superclass）。

C++的发明者 Stroustrup 认为，超类和子类这两个概念容易让人产生误解，他提出了基类和派生类这两个术语，分别用来表示超类和子类的概念。本书将引用 Stroustrup 的术语来表示派生与继承的关系。图 1-5 是表示两个类继承关系的一个简图，表示类 A 是类 B 的基类（超类，也称为父类），类 B 是类 A 的派生类（子类）。

派生类 B 继承了基类 A 的所有特征和行为，尽管类 B 只定义了数据成员 b，以及成员函数 g()。但它实际具有 a、b 两个数据成员，具有 f()和 g()两个成员函数，其中的 a 和 f()是从基类 A 继承得到的。

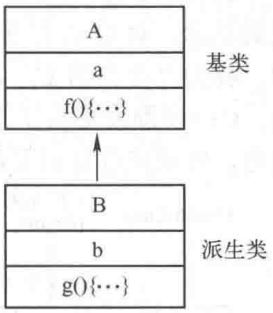

图 1-5　派生类 B 继承

继承分为单继承和多重继承，单继承规定每个子类只能有一个父类，图 1-5 就是一个单继承。多重继承允许每个子类有多个父类。继承为软件设计提供了一种功能强大的扩展机制，允许程序员基于已经设计好的基类创建派生类，并可为派生类添加基类不具有的属性和行为，极大地提高了软件复用的效率。

4. 多态

多态是面向对象程序设计语言的另一重要特征，它的意思是“一个接口，多种形态”。也就是说，不同对象针对同一种操作会表现出不同的行为。多态与继承密切相关，通过继承产生不同的类，而这些类分别对某个成员函数进行了定义，当这些类的对象调用该成员函数时会做出不同的响应，执行不同的操作，实现不同的功能，这就是多态。

1.3　C++与面向对象程序设计

C++是从 C 语言发展演变而来的，在 C 语言的基础上引入了类（class）的概念，并增加了封装、继承、多态等面向对象的语言处理机制。C++向前兼容了 C 语言程序设计，使得绝大部分 C