教科书的原理剖析与技术书的案例分析相结合
从DOS平台完美移植至Windows平台的综合实例

C++面向对象程序设计

（第2版）

杜茂康　李昌兵
曹慧英　王　永　编著

高等学校工程创新型『十二五』规划计算机教材

工程创新 Engineering Innovation

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

高等学校工程创新型“十二五”规划计算机教材

C++面向对象程序设计（第2版）

杜茂康　李昌兵　曹慧英　王　永　编著

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

内 容 简 介

本书深入浅出地介绍了标准C++面向对象的程序设计技术，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，引导读者理解和领会面向对象程序设计的思想、技术、方法和要领，掌握在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 / 杜茂康等编著. —2版. —北京：电子工业出版社，2011.7
高等学校工程创新型“十二五”规划计算机教材
ISBN 978-7-121-13625-2
Ⅰ. ① C… Ⅱ. ① 杜… Ⅲ. ① C语言－程序设计－高等学校－教材 Ⅳ. ① TP312
中国版本图书馆CIP数据核字（2011）第094673号

策划编辑：章海涛
责任编辑：章海涛 赵晨阳 特约编辑：曹剑锋
印 刷：涿州市京南印刷厂
装 订：涿州市桃园装订有限公司
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：22 字数：620千字
印 次：2011年7月第1次印刷
印 数：4000册 定价：42.00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。

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。

服务热线：（010）88258888。

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，能够开发出可靠性高、可重用和易维护的软件，是当今及未来很长一段时间内软件开发的主流技术。了解和掌握面向对象编程的基本原理和方法是进一步学习计算机应用和程序设计的基础。

C++语言是在C语言基础上扩充了面向对象机制而逐步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、兼容性和扩展性而被广泛应用于不同的领域和系统中，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，不仅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#之类的面向对象程序设计语言也就简单了。

十多年的C++语言教学经验和更长的编程实践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”**。把教科书的原理剖析和技术书的案例分析相结合有利于深刻地理解和掌握C++语言程序设计的基本原理和技术，有利于将学到的技术用于实际的软件开发中。

本书即基于这样的认知体会而编写，兼具C++技术书籍和教材的特点。既比较深刻地介绍了C++面向对象的程序技术和原理，又清晰地介绍了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，使之成为一个比较完整的综合程序，并最终将它从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DOS程序中的自定义类移植到Windows程序中的方法和过程。

本书是《C++面向对象程序设计》的第2版。其第1版自2007年出版以来，受到广大师生和软件开发人员的好评，重印多次，得到了多所高校的认可，许多学生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邮件探索了将C++类移植到Windows程序中的方法，还有不少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。

这次修订充分采纳了广大教师和读者的意见，保留了本书第1版的整体结构，但精简了部分章节的内容，删除了一些深奥难懂且不太实用的技术原理分析，并对一些程序案例进行了重新设计，使本书更加精炼和实用。

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、数据输入/输出以及Visual C++ 6.0编程环境；第2章介绍C++

语言对 C 语言非面向对象方面的扩充，主要包括指针、常量、引用、类型转换、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特征、思想和方法，包括类与对象、继承与派生、虚函数、运算符重载、模板与 STL 程序设计、异常、文件与 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 C++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**。书中精心设计了易于理解和富有代表性的释意图和案例程序，清晰而深入浅出地展示了 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，有利于程序设计能力的培养与提高。书中**所有案例程序均在 Microsoft Visual C++ 6.0 环境中测试通过**。

本书由杜茂康、李昌兵、曹慧英、王永和张仿编写。李昌兵编写了第 1、2、3、4 章，曹慧英编写了第 5、6、7 章，张仿编写了第 8、9 章，王永编写了第 10、11 章，杜茂康编写了第 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。罗龙艳、谢青、刘友军、武建军、罗文龙等老师参与了本书编写大纲的讨论与确定，2004 级信息管理与信息系统专业的两位忠实的学生李明闯和王晓润仔细地阅读了本书第 1 版初稿中的每一个字符、每一行代码，校正了初稿中的许多错误，并提出了许多有用的建议。刘曜教授对本书的编著方式和章节安排提出了有益的建议，刘达明、刘跃两位教学专家结合教学规律从章节内容的详略难易设置等方面给予本书富有建设性的指导，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的 C++语言的相关书籍，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，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，本书配有以下辅助资源：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。

读者反馈：unicode@phei.com.cn。

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、复杂的软件设计。

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

1.1　面向过程与面向对象程序设计

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，不需要什么组织原则，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。随着计算机技术的发展和应用的普及，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 年代末产生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、可独立编程的子任务、子模块。各模块由顺序、分支、循环三种基本结构组成，每个模块则由结构化程序设计语言的子程序（函数）实现。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”。

在进行结构化程序设计时，首先将要解决的问题分解成若干个功能模块，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。

结构化程序设计是一种面向过程的程序设计方法，把数据和过程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，对于不同格式的数据要做相同的处理，或者对于相同格式的数据要做不同的处理，都必须编写不同的程序，代码的可重用性较差。

此外，代码和操作过程的分离还会导致程序的可维护性也较差。其原因是它把数据和处理数据的过程（函数）分为两个独立的部分，当数据结构改变时，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，假设实现一个通信录管理软件，程序员编写了 4 个函数：InputData，SearchPhone，PrintData，SearchAddr，分别用于实现通信录数据的输入、输出和数据查询功能。许多技术都可以实现通信录的数据存取，如数组、链表、队列或堆栈等，本例中采用数组存取数据。

```
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
```

这 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的编程方法：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。

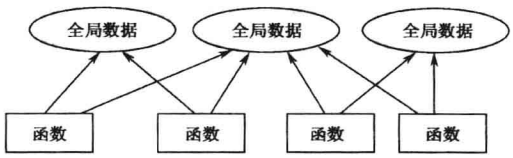

图 1-1 结构化程序设计模型

从图 1-1 的程序模型可以看出，数据和函数之间存在潜在的连接关系。某个全局数据的修改可能会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。例如在个人通信录管理程序中，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，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、Pascal、Foxpro 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

随着计算机技术的发展，软件应用的领域更加广泛，软件的规模和复杂度越来越大，软件更新的速度更快，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。

面向对象程序设计的基本观点是：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并且处于不断运动变化的事物（即对象）组成，每个事物都可以通过两方面来刻画：描述事物状态的数据和描述事物行为的函数，应该把它们结合成一个整体，代表一个客观事物，这个整体就是对象。

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。对象真实地表达了客观事物，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。

面向对象程序技术能够实现对客观世界的真实模拟，反映出世界的本来面目。从客观世界中抽象出一个个对象，对象之间能够传递消息（一个对象向其他对象发出的服务请求信息），并通过特定的函数进行数据访问，禁止以任何未经允许的方式修改对象的数据，这就是面向对象程序设计的基本模式，如图 1-2 所示。

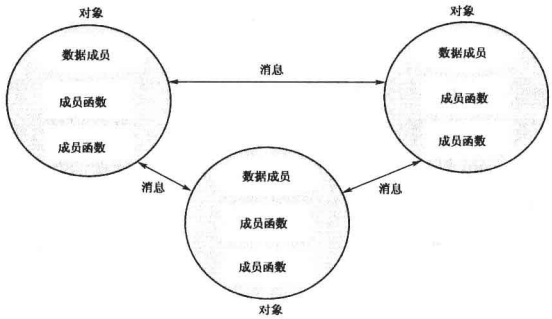

图 1-2　面向对象程序设计的程序模型

面向对象程序技术提高了软件的重用性、灵活性和扩展性，且使软件具有更好的可维护性。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

概括而言，程序的组织方式大致有两种：以功能为中心或以数据为中心。面向过程程序设计以功能为中心组织程序，而面向对象程序设计（Object-Oriented Programming，OOP）则围绕数据进行程序组织，以数据为中心，围绕数据设计程序代码，由用户定义数据及可实施于数据的操作。面向对象设计语言具有抽象、封装、继承和多态等基本特征。

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、PowerBuilder、Delphi、C#、Java 都是面向对象的程序设计语言。

1．类与对象

在现实生活中，对象是指存在的事物，如一个人、一条狗、一棵树、一台计算机等都是对象。同类对象具有相同的属性（特征）和行为，比如所有的人都有姓名、性别、眼、手脚、身高、体重等属性，有走路、讲话、打手势、学习和工作等行为；楼房有位置、楼层、房间数量、造价等属性；狗有皮毛、尾巴、四条腿等属性，有跑、吠、摇尾巴等行为；飞机能够飞行，轮船能够航行；如此等等。

人们借助于对象的属性和行为认识客观世界，将具有相同属性和行为的客观对象归入同一类。

利用这种分类方法，人们将客观世界分成了人类、狗类、猪类、树类、草类等。

每类事物都有许多实际存在的个体，这些个体则称为对象（Object）。同类事物的不同个体，尽管有着相同的一组属性和行为，但在属性取值和行为表现上存在个体差异。比如，张三和李四都具有人类所共有的属性和行为，但他们在走路、讲话、打手势、学习和工作等行为的实施方面有各自的特点，这是行为上的个体差别；他们具有不同的名字（姓名属性不同），张三1.7米高，李四1.5米高（身高属性不同），这些是属性取值的差异。在现实中，人们借助于对象的属性和行为认识和区分不同事物。

面向对象程序设计用计算机中的软件对象模拟现实中的实际对象。它用类（class）来表示同类对象的共有属性和行为，即用类这一概念来表示客观世界中的同类事物。类是用归类方法从一个个具体对象中抽取出共同特征而形成的概念。比如，张三、李四、王二……都是学生，都有学号、姓名、班级、性别等属性，具有做作业、听课等行为。将所有同学都共有的这些属性和行为抽象出来，就构成了学生类。而具有一个类所指定的属性和行为的某个个体则称为该类的一个对象。图1-3表示了一个学生类和其中一个对象的关系。

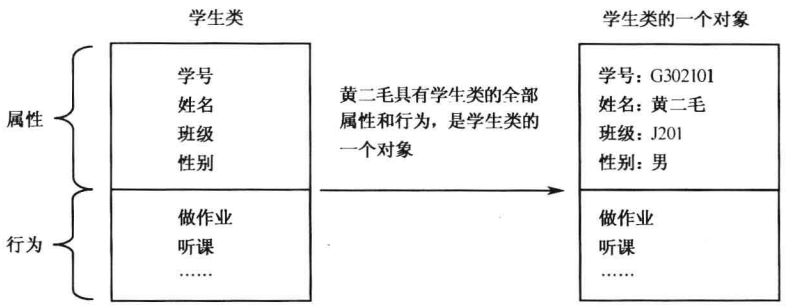

图1-3 类与对象的关系

2. 抽象与封装

抽象（abstract）是指有意忽略问题的某些细节和与当前目标无关的方面，以便把问题的本质表达得更清楚。抽象在现实生活中随处可见，如画一幅中国地图，如果不分主次把所有的山川、河流、城市、交通线路全画上去，最后的结果将是一团糟。只能通过抽象，画出中国各大省份的概况，如主要大城市、主要山脉、重要的交通线路、主要的大江大河等，有意不画出各省中不重要的县及小城镇，不画出各城市中的各条街道、公路等。道理很简单，这样更能反映出中国地理的整体面貌，让人们更加清楚地了解中国地理的分布情况。

面向对象程序设计通过数据抽象来描绘客观事物，[redacted]要特征抽取出来，有意地隐藏事物某些方面的细节，把注意力集中在事物的本质特征[redacted]把握问题的本质。数据抽象的结果将产生对应的抽象数据类型（Abstract Data Type，[redacted]

ADT把数据类型分成了接口和实现两部分。其中，[redacted]、用户能够用来完成某项任务的部分称为接口；那些对用户不可见、具体完成工作任[redacted]称为实现。这如同人们常见的DVD播放机，面板上可供用户操作使用的各种功能按钮就[redacted]的接口，各个功能按钮（如Play）完成相应功能的细节就是实现（例如，Play如何播放DV[redacted]体过程，即怎样旋转光盘，移动激光柱，读取光盘内容，并在电视机屏幕上显示光盘中的图[redacted]过程则称为Play接口的实现）。实现对用户是隐藏的，用户不知道实现的细节，也不需要知道。

数据抽象的任务是设计出清晰而足够的接口，接口必须满足用户的基本需求，且允许用户通过它访问其底层实现。抽象的结果导致了接口与实现的分离，具体可以通过数据封装实现。所谓

封装（encapsulation），就是将数据抽象的外部接口与内部实现细节分离开来，将接口显示给用户并允许其访问，将接口的实现细节隐藏起来，不让用户知道，也不允许他访问。

在面向对象程序中，封装是通过类来实现的。类是表示客观事物的抽象数据结构，它将数据及其操作函数包装成一个整体，且将内部的实现细节隐藏起来，不让类外其他对象知道，但提供了可让外界访问其功能的接口，通过接口与外界联系。

例如，前面的个人通信管理程序，在面向对象程序设计语言（如 C++）中的简易模型如下，其中的 class 是用于将数据和函数组合成一个整体（即类）的语言机制。

```
class Person{                                    //用于存取个人信息的数据结构
private:
   char name[10];
   char addr[20];
   char phone[11];
public:
   void initData(){……}                          //读入姓名、地址和电话
   char* getAddr(){……};                         //返回地址
   char* getPhone(){……};                        //返回电话号码
};
```

在这个程序模型中，Person 的 name、addr、phone 和操作数据的函数 initData()、getAddr()、getPhone()被 class 捆绑在一起，它们是一个整体，这就是封装。其中，private 区域中的数据定义对外部程序是隐藏的，public 区域中的函数是提供给外部函数访问 Person 类功能的接口。程序中的任何函数只有通过 public 区域中的 initData()、getAddr()和 getPhone()等函数才能修改或查看 Person 的 name、addr 和 phone 数据。除此之外，再无他法。

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概念。本书将引用 Stroustrup 的术语来表示派生与继承的关系。图 1-4 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）。

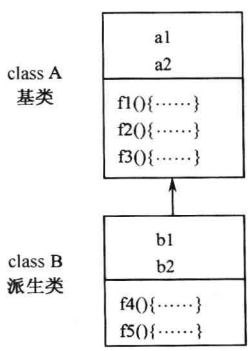

图 1-4　派生类 B 继承 A

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 b1、b2 两个数据成员，以及 f4 和 f5 两个成员函数。但它实际具有 a1、a2、b1、b2 四个数据成员，具有 f1、f2、f3、f4、f5 五个成员函数，其中的 a1、a2、f1、f2、f3 是从基类 A 继承得到的。

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 1-4 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所不

具有的属性和行为，极大地提高了软件复用的效率。

继承具有以下几个优点：

⊙继承能够清晰地体现相似类之间的层次结构关系。

⊙继承能够减少代码和数据的重复冗余度，增强程序的重用性。

⊙继承能够通过增强一致性来减少模块间的接口和界面，提高程序的易维护性。

⊙继承是自动传播代码的有力工具。

⊙继承是一种在普通类的基础上构造、建立和扩展新类的最有效手段。

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，通过继承产生的不同类，而这些类都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。它在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程序可以不加修改就能在C++环境下编译运行，同时提供了面向对象的程序设计机制，支持面向对象程序设计，是一种面向过程与面向对象的混合编程语言。

1．C++简史

在计算机发展的早期，操作系统之类的软件主要是用汇编语言编写的。由于汇编语言依赖于计算机硬件系统，用它编写的软件系统的可移植性和可读性都比较差。

UNIX系统最初也是用汇编语言编写的。为了提高UNIX系统的可移植性和可读性，1970年，美国AT&T贝尔实验室的Ken Thompson以BCPL（Basic Combined Programming Language）为基础，设计了非常简洁且与硬件很接近的B语言，用该语言改写了UNIX，并在PDP-7上实现了它。

B语言是一种无类型的语言，直接对机器字进行操作，过于简单，且功能不强。在1972年到1973年间，贝尔实验室的Dennis Ritchie对B语言进行了改造，添加了数据类型的概念，设计了C语言，并在1973年和Thompson用C语言重写了UNIX 90%以上的代码，这就是UNIX 5。在此之后，C语言又进行了多次改进，1975年UNIX 6发布后，C语言突出的优点引起了世人的普遍关注。1977年，不依赖于具体机器指令的、可移植的C语言出现了。

C语言简洁、灵活，具有丰富的数据类型和运算符，具有结构化的程序控制语句，支持程序直接访问计算机的物理地址，具有高级语言和汇编语言的双重特点。1978年以后，C语言已先后被移植到了大、中、小及微型计算机上。伴随着UNIX系统在各种类型的计算机上的实现和普及，C语言逐渐成了最受欢迎的程序设计语言之一。

但是C语言本身也存在一些缺陷，类型检查机制较弱，缺乏支持代码重用的语言结构，不适合大型软件系统的开发设计，当程序规模大到一定程度时，就很难控制程序的复杂性了。

1979年，贝尔实验室的Bjarne Stroustrup借鉴了Simula（较早的一种面向对象程序设计语言）类的概念，对C语言进行了扩展和创新，将Simula的数据抽象和面向对象等思想引入了C语言中，称为“带类的C”，这就是C++的早期版本。1983年，“带类的C”正式改名为C++。

C++发明至今，主要经历了三次修订，每次修订都增加和修改了一些内容。第一次修订发生