高等院校应用型本科“十三五”规划教材 · 计算机类

C++面向对象程序设计

C++ MIANXIANG DUIXIANG CHENGXU SHEJI

主编　王静

華中科技大學出版社
http://www.hustp.com

高等院校应用型本科“十三五”规划教材 · 计算机类

C++面向对象程序设计

C++ MIANXIANG DUIXIANG CHENGXU SHEJI

▶ 主　编　王　静

▶ 副主编　张秋生　刘胜艳　徐　冬

华中科技大学出版社

http://www.hustp.com

中国 · 武汉

图书在版编目(CIP)数据

C++面向对象程序设计/王静主编. —武汉：华中科技大学出版社，2017.8
ISBN 978-7-5680-3100-4

Ⅰ.①C… Ⅱ.①王… Ⅲ.①C语言-程序设计-教材 Ⅳ.①TP312.8

中国版本图书馆CIP数据核字(2017)第168997号

C++面向对象程序设计
C++ Mianxiang Duixiang Chengxu Sheji

王 静 主编

策划编辑：曾 光
责任编辑：史永霞
封面设计：孢 子
责任监印：朱 玢
出版发行：华中科技大学出版社(中国·武汉) 电话：(027)81321913
武汉市东湖新技术开发区华工科技园 邮编：430223
录 排：武汉正风天下文化发展有限公司
印 刷：武汉市籍缘印刷厂
开 本：787mm×1092mm 1/16
印 张：17.75
字 数：467千字
版 次：2017年8月第1版第1次印刷
定 价：40.00元

前言 PREFACE

面向对象程序设计是目前大型程序设计的主流方法，其具有封装、继承、多态等特点，使设计者可以方便地将现实世界的对象抽象封装在一起，并通过它所提供的接口来实现对象之间的交互，保证了对象的稳定和安全特性。为了最大限度地实现代码复用，在面向对象程序设计中又提供了继承方法，它允许子类继承父类的所有属性和方法，并可以灵活地在子类中对从父类继承来的属性和方法进行扩充和修改，实现子类的特例化；为了实现处理方法的名同义不同（函数名相同，具体处理的参数数据类型及个数及处理过程可能不相同），在面向对象程序设计中，又提供了多态性处理方法，允许对函数和运算符重载（静态多态），并提出了虚函数的概念，实现动态绑定，增强了程序处理的灵活性。

面向对象程序设计方法，对降低软件的复杂性，改善其重用性和维护性，提高软件的生产效率，有着十分重要的意义。

C++语言是在C语言的基础上，扩充了面向对象机制形成的一种面向对象程序设计语言。对于具有C语言基础的人来说，学习C++会比较容易。C++全面兼容了C语言，继承了C语言的全部优点和功能。因为C语言广泛流行，所以有面向对象机制的C++语言的出现大大促进了面向对象程序设计方法的发展。

本书以Visual C++ 2012作为主要开发平台，在C语言的基础上，紧密结合C++的标准，从C语言过渡到C++语言，涵盖了C++语言的主要特征，使初学者能够很快掌握C++。本书语言通俗，层次清晰，理论与实例结合，力求做到深入浅出，将复杂的概念用简洁浅显的语言来讲述，使读者尽快迈入面向对象程序设计的大门，迅速掌握C++程序设计的基本技能和面向对象的概念和方法，并能编写出具有良好风格的程序。

本书共11章，第1章面向对象程序设计概述，第2章C++入门，第3章类和对象Ⅰ，第4章类和对象Ⅱ，第5章组合和继承，第6章多态与虚函数，第7章运算符重载，第8章模板和命名空间，第9章输入输出流，第10章异常处理，第11章Windows程序开发概述和MFC库。本书所有例题均在VC++ 2012下调试通过。为了与C++国际标准相一致，使用标准C++的头文件，系统头文

件不带后缀".h",使用系统库时用命名空间 std。

由于作者水平有限,时间仓促,难免有疏漏和错误之处,敬请各位专家和读者批评指正。

编　者

2017 年 3 月

目录

CONTENTS

第 1 章　面向对象程序设计概述 …… (1)
1.1　面向对象程序设计的发展历史 …… (1)
1.2　结构化程序设计概述 …… (2)
1.3　面向对象程序设计概述 …… (3)
1.4　面向对象相对面向过程的优缺点 …… (10)
1.5　其他面向对象程序设计语言 …… (11)
1.6　关于C++上机实践 …… (11)
习题 …… (18)
第 2 章　C++入门 …… (19)
2.1　C++的发展和特点 …… (19)
2.2　一个简单的C++程序 …… (20)
2.3　数据类型 …… (22)
2.4　引用(&) …… (24)
2.5　常量 const …… (28)
2.6　内联函数 …… (31)
2.7　函数的重载 …… (32)
2.8　带有默认参数的函数 …… (35)
2.9　作用域运算符:: …… (36)
2.10　强制类型转换 …… (37)
2.11　new 和 delete …… (37)
2.12　一个面向对象的C++程序 …… (40)
习题 …… (41)
第 3 章　类和对象 I …… (45)
3.1　类的定义 …… (45)
3.2　对象的定义与使用 …… (48)
3.3　构造函数与析构函数 …… (51)

3.4 对象的赋值与复制 …… (61)
3.5 自引用指针 this …… (66)
3.6 应用举例 …… (69)
习题 …… (72)
第 4 章 类和对象 Ⅱ …… (78)
4.1 对象数组与对象指针 …… (78)
4.2 向函数传递对象 …… (83)
4.3 static 与类 …… (85)
4.4 const 与类 …… (94)
4.5 友元 …… (97)
4.6 C++的多文件程序 …… (102)
4.7 应用举例 …… (104)
习题 …… (109)
第 5 章 组合和继承 …… (114)
5.1 类的组合 …… (114)
5.2 继承的概念 …… (117)
5.3 继承与组合 …… (119)
5.4 派生类的继承方式 …… (120)
5.5 派生类的构造和析构 …… (124)
5.6 派生类重载基类函数的访问 …… (128)
5.7 多继承 …… (130)
5.8 虚基类 …… (133)
5.9 应用举例 …… (139)
习题 …… (141)
第 6 章 多态与虚函数 …… (147)
6.1 多态性概述 …… (147)
6.2 基类与派生类对象之间的赋值兼容规则 …… (148)
6.3 虚函数 …… (152)
6.4 纯虚函数与抽象类 …… (157)
6.5 应用举例 …… (159)
习题 …… (161)
第 7 章 运算符重载 …… (165)
7.1 运算符重载的基本概念 …… (165)
7.2 成员函数重载运算符 …… (166)
7.3 友元函数重载运算符 …… (177)
7.4 成员函数重载运算符与友元函数重载运算符比较 …… (180)
7.5 类型转换 …… (183)
7.6 应用举例 …… (190)
习题 …… (194)

第 8 章　模板和命名空间 …… (197)
8.1　模板的概念 …… (197)
8.2　函数模板 …… (197)
8.3　类模板 …… (201)
8.4　命名空间和头文件命名规则 …… (204)
8.5　应用举例 …… (207)
习题 …… (211)
第 9 章　输入输出流 …… (215)
9.1　C++的流 …… (215)
9.2　输入输出流 …… (218)
9.3　文件的输入输出 …… (235)
9.4　应用举例 …… (243)
习题 …… (245)
第 10 章　异常处理 …… (247)
10.1　异常处理概述 …… (247)
10.2　异常处理的方法 …… (248)
10.3　异常匹配 …… (254)
10.4　标准异常及层次结构 …… (255)
10.5　应用举例 …… (255)
习题 …… (256)
第 11 章　Windows 程序开发概述和 MFC …… (258)
11.1　C++的 Windows 编程 …… (258)
11.2　MFC 应用程序 …… (265)
11.3　MFC 的类层次结构 …… (270)
11.4　MFC 类功能简介 …… (272)
习题 …… (275)
参考文献 …… (276)

第1章 面向对象程序设计概述

【学习目标】

(1) 了解面向对象技术的发展历史。

(2) 了解面向对象软件开发的过程。

(3) 掌握面向对象程序设计的相关术语。

(4) 掌握面向对象程序设计的特征。

(5) 了解目前常用的面向对象程序设计语言。

传统的软件开发方法曾经给软件产业带来了巨大的进步，尤其是在开发中小规模软件项目中获得了成功。但是随着硬件性能的提高和图形用户界面的推广，软件的应用更加普及与深入。当开发大型软件产品时，由于面对的问题越来越复杂，再使用传统软件开发方法，成本较高，成功率较低。

随着面向对象编程语言 Simula 67 中首次引入了类和对象的概念，人们逐渐开始注重面向对象分析和面向对象设计的研究，因此产生了面向对象方法学。到了 20 世纪 90 年代，面向对象方法学已经成为人们在开发软件时的主流软件设计方法。

1.1 面向对象程序设计的发展历史

“对象”和“对象的属性”这样的概念可以追溯到 20 世纪 50 年代初，它们首先出现于关于人工智能的早期著作中。但是出现了面向对象语言之后，面向对象思想才得到了迅速的发展。

1967 年，挪威计算中心的 Kristen Nygaard 和 Ole-Johan Dahl 开发的 Simula 67 语言被认为是最早的面向对象程序设计语言。它引入了所有后来面向对象程序设计语言所遵循的基础概念：对象、类和继承。他们对类、对象、继承和动态绑定等重要概念的首先引入，为面向对象这一当前最流行、最重要的程序设计技术奠定了基础。

Simula 67 的面向对象概念的影响是巨大而深远的。它本身虽然未能广泛流行，但在它的影响下产生的面向对象技术却迅速传播开来。

20 世纪 70 年代，Smalltalk 的问世又给面向对象的语言注入了新的血液。Smalltalk 被认为是第一个真正面向对象的语言，Smalltalk 的问世标志着面向对象程序设计方法的正式形成。

面向对象源于 Simula，真正的 OOP 由 Smalltalk 奠基。Smalltalk 现在被认为是最纯的 OOPL(面向对象程序设计语言)。

而在实践中，人们开始渐渐发现，由于 C 语言是如此深入人心，以至于当前最好的解决软件设计危机的方法并不是另外发明一种新语言去代替 C 语言，而是在它的基础上加以发展，使之可以扩展到面向对象的领域。

在这种形势下，C++于 20 世纪 80 年代初面世。C++保留了 C 语言原有的特点，同时增加了面向对象的机制。由于 C++对 C 语言的改进主要是增加了类，因此它最初被设计者称为“带类的 C”，后来为了强调它是 C 的增强版，就采用 C 语言中的自加运算符号“++”，改称它为“C++”。从 C++的名字中可以看出，C++是 C 的超集，因此 C++既

可以用于面向过程的结构化程序设计，又可以用于面向对象的程序设计，是一种功能强大的混合型的程序设计语言。

从20世纪80年代中期到90年代，是面向对象语言走向繁荣的阶段。其主要表现是大批比较实用的面向对象编程语言的涌现，例如C++、Objective-C、Object Pascal、Eiffel等。这些面向对象的编程语言分为纯OO型语言和混合型OO语言。混合型OO语言是在传统的过程式语言基础上增加了OO语言成分形成的，在实用性方面具有更大的优势。此时的纯OO型语言也比较重视实用性。现在，在面向对象编程方面，普遍采用语言、类库和可视化编程环境相结合的方式，如Visual C++、Delphi等。面向对象方法也从编程发展到设计、分析，进而发展到整个软件生命周期。

到20世纪90年代，面向对象的分析与设计方法已多达数十种，这些方法都各有所长。目前，统一建模语言UML已经成为世界性的建模语言，适用于多种开发方法。把UML作为面向对象的建模语言，不但在软件产业界获得了普遍支持，在学术界影响也很大。

在C++之后，影响巨大的就是Java和C#语言了。从某个角度来说，它们是更纯粹的面向对象语言。因为C++可以用于面向过程的结构化程序设计，而Java和C#则没有这个功能。不过，Java和C#也有自己的特点，它们都支持丰富的MetaClass，这使得一切皆对象的概念支持得越发深刻。

在面向对象发展到现今，又出现了一些重大的变革，最突出的就是动态语言的出现。动态语言也是支持面向对象技术的，最典型的动态语言有JavaScript、Python、Ruby等。它们一个重大的变化就是将类的信息改变为动态的，并提出了Ducking Type的概念，这在很大程度上提升了编程的生产力。

其实，不仅仅在程序设计方面，面向对象也在不断向其他方面渗透。1980年Grady Booch提出了面向对象设计的概念，面向对象分析由此开始。1985年，第一个商用面向对象数据库问世。1990年以来，面向对象分析、测试、度量和管理等研究都得到长足发展。

从此，全世界掀起了一股面向对象的热潮，至今盛行不衰，面向对象也逐渐成为程序设计的主流。

目前，面向对象设计方法和结构化方法仍是两种在系统开发领域相互依存的、不可替代的方法。

1.2 结构化程序设计概述

在讨论面向对象程序设计之前，我们需要讨论一下传统的软件程序设计，也称为结构化程序设计。

结构化程序设计由E. W. Dijkstra在1969年提出，是以模块化设计为中心，将待开发的软件系统采用自顶向下、逐步求精，划分为若干个相互独立的模块，这样使完成每一个模块的工作变得单纯而明确。结构化程序设计主要强调的是程序的易读性。

它将整个待解决的问题，抽象为描述事物的数据以及描述对数据进行处理的算法，或者说数据处理过程。

面向过程的程序设计思想的核心是功能的分解：

第一步要做的工作就是将问题分解成若干个称为模块的功能块；

第二步根据模块功能来设计一系列用于存储数据的数据结构；

第三步编写一些过程(或函数)对这些数据进行操作。

显然，这种方法将数据结构和过程作为两个实体来对待，其着重点在过程。该方法强调的是将一个较为复杂的任务分解成许多易于控制和处理的子任务，自顶向下顺序地完成软件开发各阶段的任务。

面向过程的程序设计的缺点之一就是数据结构和过程的分离，一旦数据结构需要变更，就必须修改与之有关的所有模块。因此，面向过程的程序的可重用性差，维护代价高。

下面我们举一个实例来进一步讨论面向过程的程序设计方法。

考虑一个学生信息系统。该系统所处理的学生类型是研究生，允许用户进行输入学生信息、输出学生信息、插入(学生)、删除(学生)、查找(学生)等操作。首先根据面向过程的程序设计方法，将学生信息系统分解成 5 个对应的模块来实现以上工作，接着，建立一个简单的数据结构：

```
struct Student{
    char Name[20];              //姓名
    long lNum;                  //学号
    float fGrade                //成绩
};
```

然后，对每个过程按照一定的操作顺序编写程序。此时，数据结构与“过程”分离。

我们考虑如果数据结构发生了一些变化会产生什么样的结果。这时如果需要再增加一种学生类型——在职研究生，则原来的程序就不能处理了，因为学生类型不同，不同类型的学生对应不同的处理，就需要重新编写程序代码。因此，面向过程的程序可重用性差，维护代价高。

在整个 20 世纪 70 年代，结构化方法在软件开发中占绝对统治地位。当时问题规模比较小，需求变化也不大。但是，到了 70 年代末期，随着计算机科学的发展和应用领域的不断扩大，人们对计算机技术的要求越来越高。问题越来越复杂，规模越来越大，需求变化越来越快，面向过程就显得有些力不从心。如上例，当根据需求变化要修改某个结构体时，与之相关的所有过程函数就不得不也要相应地进行修改；而当修改一个过程函数时，也往往会涉及其他数据结构。在系统规模较小的时候，这还比较容易解决，可是当系统规模越来越大，涉及多人协作开发的时候，结构化程序设计语言和结构化分析与设计已经无法满足用户需求的变化，于是人们开始寻找更先进的软件开发方法和技术，OOP 由此应运而生。

面向对象程序设计(OOP)技术被认为是程序设计方法学的一场实质性的革命，是程序设计方法学的一个里程碑。OOP 大大提高了软件的开发效率，减少了软件开发的复杂性，提高了软件系统的可维护性、可扩展性。

1.3 面向对象程序设计概述

面向对象程序设计思维更接近人的思维活动，按人们认识客观世界的系统思维方式，采用基于对象(实体)的概念建立模型，模拟客观世界，分析、设计、实现软件的办法，通过面向对象的理念，使计算机软件系统能与现实世界中的系统一一对应。这种思想与传统的方法有很大不同。

面向对象是将客观事物看作具有属性和行为的对象，通过对客观事物的抽象找出同一类对象的共同属性(静态属性)和行为(动态特征)，形成类。每个对象有自己的数据、操作、功能和目的。通过类的继承和派生、多态等技术，提高软件代码的可重用性。

例如，在1.2节的学生信息系统中，采用面向对象的思想，可以先定义一个学生类（研究生），包括学生的基本信息如姓名、学号、成绩等和学生所对应的相应的操作；当需要增加一种学生类型（如在职研究生）时，可以采用继承和派生的方式，在学生类的基础上派生出一个新类，这样该新类不仅可以继承学生类的所有特性，而且可以根据需要增加必要的程序代码，从而避免了公用代码的重复开发，实现了代码重用。

面向对象程序设计是一种新的程序设计范型。面向对象程序的主要结构特点是：

第一，程序一般由类的定义和类的使用两部分组成，在主程序中定义各对象并规定它们之间传递信息的规律；

第二，程序中的一切操作都是通过向对象发送消息来实现的，对象接收到消息后，启动有关方法完成相应的操作。

在面向对象的程序设计中，着重点在那些将要被操作的数据，而不是在实现这些操作的过程。数据构成了软件分解的基础，而不是功能。更重要的是，不能将数据和相应操作看成两个分离的实体，而是要把它们作为一个完整的实体来对待。数据与定义在它上面的操作构成一个整体。同时，数据本身不能被外部程序和过程直接存取，唯一的办法是通过定义在该数据上的操作，间接地实现对数据的读写，这样就实现了对信息的隐藏。

面向对象程序设计的最大优点就是软件具有可重用性。当人们对软件系统的要求有所改变时，并不需要程序员做大量的工作，就能使系统做相应的变化。

类与对象是面向对象程序设计中最重要的概念，也是一个难点，想要掌握面向对象程序设计的技术，首先就要很好地理解这两个概念。

1.3.1 面向对象的软件开发

随着软件规模的迅速增大，软件人员面临的问题十分复杂。需要规范整个软件开发过程，明确软件开发过程中每个阶段的任务，在保证前一个阶段工作的正确性的情况下，再进行下一阶段的工作。这就是软件工程学需要研究和解决的问题。面向对象的软件工程包括以下几个部分。

1. 面向对象分析

在软件工程中的系统分析阶段，系统分析员要和用户结合在一起，对用户的需求做出精确的分析和明确的描述，从宏观的角度概括出系统应该做什么（而不是怎么做）。面向对象的分析，要按照面向对象的概念和方法，在对任务的分析中，从客观存在的事物和事物之间的关系，归纳出有关的对象（包括对象的属性和行为）以及对象之间的联系，并将具有相同属性和行为的对象用一个类（class）来表示，建立一个能反映真实工作情况的需求模型。

2. 面向对象设计

根据面向对象分析阶段形成的需求模型，对每一部分分别进行具体的设计，首先是进行类的设计，类的设计可能包含多个层次（利用继承与派生）。然后以这些类为基础，提出程序设计的思路和方法，包括对算法的设计。

在设计阶段，并不牵涉某一种具体的计算机语言，而是用一种更通用的描述工具（如伪代码或流程图）来描述。

3. 面向对象编程

根据面向对象设计的结果，用一种计算机语言把它写成程序，显然应当选用面向对象的计算机语言（例如C++），否则无法实现面向对象设计的要求。

4. 面向对象测试

在写好程序后交给用户使用前，必须对程序进行严格的测试。测试的目的是发现程序中的错误并改正它。面向对象测试是用面向对象的方法进行测试，以类作为测试的基本单元。

5. 面向对象维护

因为对象的封装性，修改一个对象对其他对象影响很小。利用面向对象的方法维护程序，大大提高了软件维护的效率。

现在设计一个大的软件，严格按照面向对象软件工程的5个阶段进行，这5个阶段的工作不是由一个人从头到尾完成的，而是由不同的人分别完成的。这样，OOP阶段的任务就比较简单了，程序编写者只需要根据OOD提出的思路用面向对象语言编写出程序即可。在一个大型软件的开发中，OOP只是面向对象开发过程中的一个很小的部分。如果所处理的是一个较简单的问题，可以不必严格按照以上5个阶段进行，往往由程序设计者按照面向对象的方法进行程序设计，包括类的设计(或选用已有的类)和程序的设计。

1.3.2 面向对象程序设计方法的基本概念

面向对象程序设计(object oriented programming，OOP，面向对象编程)的一条基本原则是计算机程序是由单个能够起到子程序作用的单元或对象组合而成的，是一种把面向对象的思想应用于软件开发过程中，指导开发活动的系统方法，是建立在"对象"概念基础上的方法学。

对象是由数据和容许的操作组成的封装体，与客观实体有直接对应关系，一个对象类定义了具有相似性质的一组对象。而继承是对具有层次关系的类的属性和操作进行共享的一种方式。所谓面向对象就是基于对象概念，以对象为中心，以类和继承为构造机制，来认识、理解、刻画客观世界和设计、构建相应的软件系统。

1. 对象

对象(object)是要研究的任何事物。从一本书到一家图书馆，甚至极其复杂的自动化工厂、航天飞机等，都可看作对象，它不仅能表示有形的实体，也能表示无形的(抽象的)规则、计划或事件。每个对象皆有自己的内部状态和运动规律，如学生张三具有名字、专业、某门功课成绩等内部状态，具有吃饭、睡觉、选课、考试等运动规律。对象由数据(描述事物的属性)和作用于数据的操作(体现事物的行为)构成一个独立整体。

从程序设计者来看，对象是一个程序模块；从用户来看，对象为他们提供所希望的行为。在对象内的操作通常称为方法。

对象可指现实社会中的对象，即问题域中的对象；也可指程序中的对象，即解题域中的对象。

2. 类

类(class)是具有相似内部状态和运动规律的实体的集合(或统称、抽象)。类的概念来自于人们认识自然、认识社会的过程。在这一过程中，人们主要使用两种方法：由特殊到一般的归纳法和由一般到特殊的演绎法。在归纳的过程中，我们从一个个具体的事物中把共同的特征抽取出来，形成一个一般的概念，这就是"归类"。如昆虫、狮子、爬行动物，因为它们都能动，所以归类为动物。在演绎的过程中我们又把同类的事物，根据不同的特征分成不同的小类，这就是"分类"，如动物→猫科动物→猫→大花猫等。对于一个具体的类，它有许多具体的

个体，我们就管这些个体叫作“对象”。类的内部状态是指类集合中对象的共同状态，类的运动规律是指类集合中对象的共同运动规律。如柏拉图对人做如下定义：人是没有毛、能直立行走的动物。在柏拉图的定义中，“人”是一个类，具有“没有毛、能直立行走”等一些区别于其他事物的共同特征；而张三、李四、王五等一个个具体的人，是“人”这个类的一个个“对象”。

类是对象的模板，即类是对一组有相同数据和相同操作的对象的定义。一个类所包含的方法和数据描述一组对象的共同属性和行为。类是在对象之上的抽象，对象则是类的具体化，是类的实例。类可有其子类，也可有其他类，形成类层次结构。

在面向对象的软件技术中，“类”就是对具有相同数据和相同操作的一组相似对象的定义。也就是说，类是对具有相同属性和行为的一个或多个对象的描述，通常在这种描述中也包括对怎样创建该类的新对象的说明。通俗地讲，类是对具有相同属性和行为的一组相似的对象的抽象。

如 1.2 节中的例子，我们可以定义一个学生类。

```
class Student
{
   int Num;          //学号
   char sName[20];   //姓名
   float fGrader;    //成绩
public:
   void input();     //输入学生信息
   void print();     //输出学生信息
};
```

3. 实例

实例(instance)是一个类所描述的一个具体的对象。例如，通过“大学生”类定义一个具体的对象学生王明就是大学生类的一个实例，就是一个对象。学号：160201；姓名：王明；高等数学成绩：80，这些就是对象中的数据。输入学生信息、输出学生信息等操作就是对象中的操作。

实例就是由某个特定的类所描述的一个具体的对象。实际上，类是建立对象时使用的“模板”，按照这个模板所建立的一个个具体的对象，就是类的实际例子，简称实例。

注意：使用“对象”这个术语，既可以指一个具体的对象，也可以泛指一般的对象；但是，使用“实例”这个术语，必然是指一个具体的对象。

类和对象之间的关系是抽象和具体的关系。类是对多个对象进行综合抽象的结果，对象是类的个体实物，一个具体的对象是类的一个实例。

4. 属性

属性，就是在类中定义的数据。它是对客观世界实体所具有的性质的抽象。例如，Student 类中所定义的表示学生的学号、姓名、成绩的数据成员就是 Student 类的属性。类的每个实例都有自己特有的属性值。如上例中学生王明的学号：160201；姓名：王明；高等数学成绩：80，就是实例王明自己特有的属性。

5. 消息

现实世界中的对象不是孤立存在的实体，它们之间存在着各种各样的联系，正是它们之间的相互作用、联系和连接，才构成了世间各种不同的系统。

在面向对象程序设计中，对象之间也需要联系，我们称为对象的交互。面向对象程序设计技术必须提供一种机制，允许一个对象与另一个对象的交互，这种机制称为消息传递。一个对象向另一个对象发出的请求称为“消息”。当对象接收到发给它的消息时，就调用有关的方法，执行相应的操作。

例如，zhangsan 是 Student 类的对象，当要求他输入自己的个人信息时，在 C++中应该向他发下列信息：

```
zhangsan. input();
```

其中 zhangsan 是接收消息的对象的名字，input 是消息名。当对象 zhangsan 接收到这个消息后，确定应完成的操作并执行之。

一般情况下，我们称发送消息的对象为发送者或请求者，接收消息的对象为接收者或目标对象。发送者发送消息，接收者通过调用相应的方法对消息做出响应。这个过程不断重复，系统不停地运转，最终得到相应的结果。

消息具有以下三个性质：

(1) 同一个对象可以接收不同形式的多个对象，做出不同的响应；

(2) 相同形式的消息可以传递给不同的对象，所做出的响应可以是不同的；

(3) 接收对象对消息的响应并不是一定的，对象可以响应消息，也可以不响应。

6. 方法

方法是对象所执行的操作，也是类中所定义的服务。方法描述了对象执行操作的算法和响应消息的方法。在 C++中把方法称为成员函数。

例如，为了让 Student 类中的对象能够响应输入运算，在 Student 类中必须给出成员函数 void input()的定义，也就是要给出这个函数的实现代码。

7. 重载

在解决问题时，经常会遇到一些函数，它们的功能相同，但参数类型不同或参数个数不相等。例如，求一个数的立方或求最大值问题，参数类型可能是整型，也可能是实型；可能是求两个参数的最大值，也可能是求三个参数的最大值。但很多程序设计语言要求函数名必须唯一，因此就需要定义不同函数名，使得程序员需要记忆很多不同的名字，增加了程序员的负担。针对这类问题，C++提供了重载机制，即允许具有相同或相似功能的函数使用同一函数名，从而减少了程序员记忆多个函数名字的负担。C++提供的重载包括函数重载和运算符重载。函数重载是指同一作用域内的若干个参数特征不同的函数可以使用相同的函数名字，运算符重载是指同一个运算符可以施加于不同类型的操作数上。也就是说，相同名字的函数或运算符在不同的场合可以表现出不同的行为。

1.3.3 面向对象程序设计的特征

面向对象程序设计是一种把面向对象的思想应用于软件开发过程中，指导开发活动的系统方法，是建立在“对象”概念基础上的方法学。对象是由数据和操作组成的封装体，与客观实体有直接对应关系，一个对象类定义了具有相似性质的一组对象。面向对象程序设计具有抽象性、封装性、继承性和多态性等特征。

抽象：从事物中舍弃个别的、非本质的特征，而抽取共同的、本质特征的思维方式。

封装：将数据和代码捆绑到一起，避免了外界的干扰和不确定性。对象的某些数据和代码可以是私有的，不能被外界访问，以此实现对数据和代码不同级别的访问权限。

继承：让某个类型的对象获得另一个类型的对象的特征。通过继承可以实现代码的重用：从已存在的类派生出的一个新类将自动具有原来那个类的特性，同时，它还可以拥有自己的新特性。

多态：一般类和特殊类可以有相同格式的属性或操作，但这些属性或操作具有不同的含义，即具有不同的数据类型或表现出不同的行为。

1. 抽象

类的定义中明确指出类是一组具有内部状态和运动规律对象的抽象，抽象是一种从一般的观点看待事物的方法。它要求我们集中于事物的本质特征（内部状态和运动规律），而非具体细节或具体实现。面向对象鼓励我们用抽象的观点来看待现实世界，也就是说，现实世界是一组抽象的对象——类组成的。

抽象就是从众多的事物中抽取出共同的、本质性的特征，舍弃其非本质的特征。例如，苹果、香蕉、酥梨、葡萄、桃子等，它们共同的特征就是水果。得出水果概念的过程，就是一个抽象的过程。共同特征是指那些能把一类事物与其他类事物分开来的特征，这些具有区分作用的特征又称为本质特征。而共同特征是相对的，是指从某一片面来看是共同的。例如，对于汽车和大米，从买卖的角度看都是商品，都有价格，这是它们的共同特征，而从其他方面比较时，它们则是不同的。所以在抽象时，哪些是共同特征，决定于从什么角度进行抽象，抽象的角度取决于分析问题的目的。抽象的目的主要是降低复杂度，以得到问题域中较简单的概念，好让人们能够控制其过程或以宏观的角度了解许多特定的事态。

抽象包含两个方面：一方面是过程抽象，另一方面是数据抽象。过程抽象就是针对对象的行为特征，如鸟会飞、会跳等，这些方面可以抽象为方法，即过程，写成类时都是鸟的方法。数据抽象就是针对对象的属性，如建立一个鸟这样的类，鸟会有以下特征——两个翅膀，两只脚，有羽毛等，写成类时这些都应是鸟的属性。

例如，用面向对象程序设计方法设计学生信息管理系统，由于管理的对象是学生，分析的重点应该是学生，通过分析学生信息管理系统的各种功能、操作和学生的主要属性（学号、姓名、班级、年龄、各科成绩等），找出其共性，将学生作为一个整体对待，并抽象成一个类（Student）。将学生群体抽象为一个类的过程如图 1-1 所示。在该抽象过程中，首先有高低、胖瘦、俊丑、学习好坏等各不相同的学生 1、学生 2……但他们都属于学生，都具有学号、姓名、班级、年龄、性别、成绩等属性（数据），还有输入学号、修改班级、打印各科成绩等行为（方法）。因此，可以把这些属性和方法封装起来而形成类。有了类后就可以建立类的实例，即类对应的对象。在此基础上还可以派生出其他类。

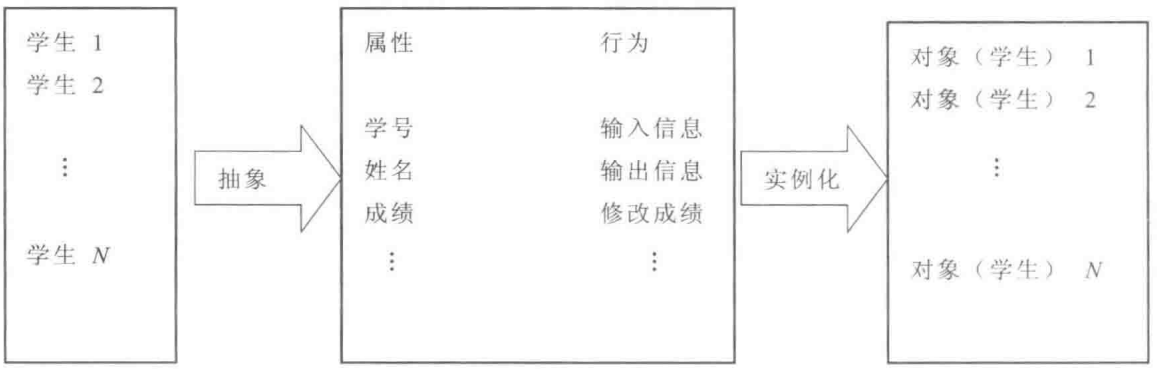


图 1-1 抽象过程示意图

2. 封装

对象间的相互联系和相互作用过程主要通过消息机制得以实现。对象之间并不需要过多地了解对方内部的具体状态或运动规律。面向对象的类是封装良好的模块，类定义将其说明（用户可见的外部接口）与实现（用户不可见的内部实现）显式地分开，其内部实现按其具体定义的作用域提供保护。对象结构示意图如图 1-2 所示。

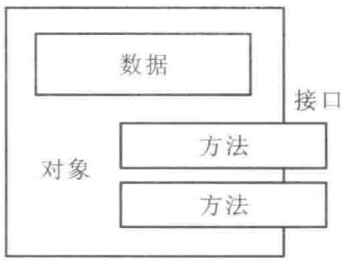


图 1-2 对象结构示意图

封装是一种信息隐蔽技术，是对象的重要特性。封装使数据和加工该数据的方法（函数）封装为一个整体，以实现独立性很强的模块，使得用户只能见到对象的外特性（对象能接受哪些消息，具有哪些处理能力），而对象的内特性（保存内部状态的私有数据和实现加工能力的算法）对用户是隐蔽的。封装的目的在于把对象的设计者和对象的使用者分开，使用者不必知晓行为实现的细节，只需用设计者提供的消息来访问该对象。就像电视机、录音机、洗衣机等，从其外形来看，各种电子或机械部件被封装在盒子内部。使用这些电器的人并不需要知道电器内部有哪些部件，它们是如何组装的，它们的工作原理又如何。使用者只需要会使用电器提供的几个外部按钮（对应于对象的外部接口），就可以实现自己所需要的功能。将电器部件封装在盒子内部，既可以避免各种人为的损坏，也便于维护和管理。在此，类是封装的最基本单位，在类中定义的接收对方消息的方法可称为类的接口。封装防止了程序相互依赖性而带来的变动影响。

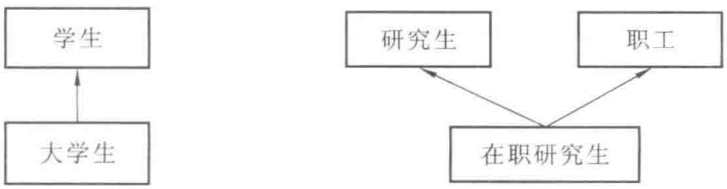
通过封装，我们很好地实现了细节对外界的隐藏，从而达到数据说明与操作实现分离的目的，使用者只需要知道它的说明即可使用它。

3. 继承

继承是类不同抽象级别之间的关系。类的定义主要有 2 种办法归纳和演绎。由一些特殊类归纳出来的一般类称为这些特殊类的父类，特殊类称为一般类的子类，同样父类可演绎出子类，父类是子类更高级别的抽象。子类可以继承父类的所有内部状态和运动规律。广义地说，继承是指能够直接获得已有的性质与特征，而不必重复定义它们，且继承具有传递性。在 OO 软件技术中，继承是子类自动地共享基类中定义的数据和方法的机制。继承性使得相似的对象可以共享程序代码和数据结构，从而大大减少了程序中的冗余信息。

在计算机软件开发中采用继承性，提供了类的规范的等级结构；通过类的继承关系，使公共的特性能够共享，提高了软件的重用性。

一个类通过继承不仅可以直接继承其他类的全部特性，同时还可对继承来的特性进行修改并新增子类所特有的特性。继承分为单继承（一个子类只有一个父类）和多继承（一个类有多个父类），如图 1-3 所示。当允许一个类只能继承一个类时，类的继承就是单继承，比如 Java 语言中，一个类继承另一个类时只能是单继承，而 C++语言中就允许一个类继承多个类。



(a) 单继承　　(b) 多继承

图 1-3 单继承和多继承