徐宏喆 李 文 董丽丽

西安交通大学出版社
XI'AN JIAOTONG UNIVERSITY PRESS

徐宏喆 李 文 董丽丽

西安交通大学出版社
XI'AN JIAOTONG UNIVERSITY PRESS

内容简介

本书是一本介绍面向对象程序设计内容及原理的教材，主要用于本科生学习面向对象程序设计课程及实验。本书以C++语言为基础，Microsoft Visual C++ 6.0(简称为VC++6.0)和Code::Blocks为实验环境，系统地阐述了面向对象程序设计的特点和思想，旨在使读者迅速迈入面向对象程序设计的大门，同时掌握C++程序设计的基本技能和面向对象程序设计的概念与方法，并能编写出具有良好风格的程序。

本书共分为8章及4个附录。第1章总体介绍面向对象程序设计和C++语言；第2章通过与传统程序设计的比较，介绍面向对象程序设计的概念和特性；第3章至第8章，详细阐述了C++支持的面向对象程序设计的基本方法，包括C++语言基础、类、对象、派生与继承、多态性、I/O流等；最后，在附录中介绍相应的开发环境，并安排综合与系统的训练，作为知识性的扩充与编程能力的提高。

图书在版编目(CIP)数据

C++面向对象程序设计基础/徐宏喆，李文，董丽丽编. —西安：西安交通大学出版社，2014.8

ISBN 978-7-5605-6449-4

Ⅰ.①C… Ⅱ.①徐…②李…③董… Ⅲ.①C语言-程序设计 Ⅳ.①TP312

中国版本图书馆CIP数据核字(2014)第144316号

书　　名　C++面向对象程序设计基础
编　　者　徐宏喆　李　文　董丽丽
责任编辑　屈晓燕

出版发行　西安交通大学出版社
　　(西安市兴庆南路10号　邮政编码710049)
网　　址　http://www.xjtupress.com
电　　话　(029)82668357　82667874(发行中心)
　　(029)82668315　82669096(总编办)
传　　真　(029)82668280
印　　刷　陕西奇彩印务有限责任公司

开　　本　787mm×1092mm　1/16　印张　22　字数　532千字
版次印次　2014年11月第1版　2014年11月第1次印刷
书　　号　ISBN 978-7-5605-6449-4/TP·625
定　　价　40.00元

读者购书、书店添货、如发现印装质量问题，请与本社发行中心联系、调换。
订购热线：(029)82665248　(029)82665249
投稿热线：(029)82664954
读者信箱：jdlgy@yahoo.cn

前　言

面向对象程序设计是不同于面向过程程序设计的一种新的程序设计范型，它对降低软件的复杂性，改善其通用性和维护性，提高软件的开发效率，有着十分重要的意义。

C++语言作为同时支持面向对象程序设计和面向过程程序设计的混合型语言，既保留了C语言灵活高效的特点，同时大量引入了面向对象的思想，增加了许多面向对象所独有的语法结构，如类、继承、多态等。C++语言中灵活的变量说明，新的输入/输出流，引用的使用，都大大方便了编程操作，即使在面向过程程序设计过程中，使用C++语言也会带来更高的效率和安全性。

本书共分为8章及4个附录。书中以准确的语言和丰富的示例程序，系统地介绍了C++语言的各种语法结构及特性。第1章总体介绍面向对象程序设计和C++语言；第2章介绍面向对象程序设计的概念和特性；第3章介绍C++语言的基础语法结构，包括数据类型、表达式、语句和函数，同时还介绍了几种C++语言中特有的语法结构，如引用等。第4章介绍面向对象程序设计最基础的概念——类和对象，是本书的核心和基础；第5章介绍静态成员与友元的概念；第6章介绍面向对象程序设计非常重要的概念——派生类与继承，也是本书的一个核心和难点；第7章介绍多态的编程思想及C++中提供的多态机制；第8章介绍C++语言全新的I/O流。附录1是VC++6.0开发环境的简介；附录2是Code::blocks开发环境的简介；附录3中列出了C++语言中运算符优先级关系；附录4是综合训练，旨在提高读者的实际程序设计能力。

本书各章都有综合训练、小结、练习题和提示。综合训练包含了本章的核心语法点，通过对实际题目的分析、编程，逐步向读者介绍使用C++语言编程的方法和经验，力求让读者能够掌握一些实际编程的方法和技巧；小结是对本章所有内容的回顾，可以帮助读者复习；练习题是章节内容的补充，读者可以通过完成练习题来巩固各章的内容；提示贯穿章节始终，介绍实际编程的一些技巧。

本书是编者集多年对普通高等院校本科生、研究生C++面向对象程序设计的教学经验，并参阅大量中外资料的基础上完成的。为了使读者能够更好地掌握面向对象程序设计的思想，编者根据多年在教学与科研工作中的心得体会，通过大量生动明晰的实例、丰富的图例、生动的语言，一步步揭示和阐述面向对象程序设计的概念与思想，并运用软件工程的方法、示例，力求使本书具有通俗性和实用性。

本书从初学者的视角入手，紧紧围绕着如何进行面向对象程序设计而展开，由浅入深地使读者掌握面向对象程序设计的思想与方法。大量生动、形象、丰富的实例与每章、每小节都会出现的建议、提示等内容都是本书的一大特色。

在本书的编写过程中，朱晓光、韩俊强、赵嘉麒、魏中强、吴夏、武晓周、姚智海、薛岁庆等人完成了大量校正、录入工作。在此对他们的工作表示感谢！

对于本书存在的疏漏和不足之处，希望广大读者批评指正。

编者

目　录

第1章　绪论 …… (1)
1.1　软件开发方法 …… (2)
1.1.1　面向过程的开发方法 …… (2)
1.1.2　面向对象的开发方法 …… (3)
1.2　面向对象的概念和面向对象程序设计 …… (3)
1.2.1　面向对象方法的概念 …… (3)
1.2.2　面向对象的程序设计 …… (4)
1.2.3　面向对象开发技术的优点 …… (4)
1.3　面向对象语言(C++)及开发工具 …… (4)
思考与练习题 …… (5)
第2章　面向对象的程序设计 …… (6)
2.1　对象与类 …… (6)
2.1.1　对象与类的概念 …… (6)
2.1.2　对象的交互 …… (8)
2.2　数据的抽象与封装 …… (9)
2.2.1　现实世界中的抽象与封装 …… (9)
2.2.2　程序设计中的抽象与封装 …… (9)
2.3　继承 …… (10)
2.3.1　继承的概念 …… (10)
2.3.2　继承与封装的关系 …… (11)
2.4　多态性 …… (11)
2.4.1　多态的概念 …… (11)
2.4.2　重载 …… (12)
2.5　本章小结 …… (12)
思考与练习题 …… (12)
第3章　C++语言基础 …… (14)
3.1　C++语言基础 …… (14)
3.1.1　C++编程简介 …… (14)
3.1.2　基本数据类型 …… (17)
3.1.3　表达式 …… (21)
3.1.4　基本语句 …… (24)
3.1.5　函数 …… (30)
3.1.6　const 修饰符 …… (32)
3.1.7　动态内存分配运算符 new 和 delete …… (33)

3.1.8　作用域运算符::　(35)
3.1.9　引用　(36)
3.2　综合训练　(39)
训练 1　(39)
训练 2　(40)
3.3　本章小结　(42)
思考与练习题　(42)
第4章　类和对象　(47)
4.1　类和对象的基本概念　(47)
4.1.1　从结构体到类　(47)
4.1.2　类的定义　(49)
4.1.3　类中的数据成员　(51)
4.1.4　类中的成员函数　(51)
4.1.5　类中的成员访问　(53)
4.1.6　类对象　(54)
4.1.7　类的作用域　(58)
4.1.8　小结与建议　(60)
4.2　构造函数与析构函数　(60)
4.2.1　构造函数的必要性　(60)
4.2.2　构造函数　(61)
4.2.3　析构函数　(62)
4.2.4　带参数的构造函数　(65)
4.2.5　默认参数的构造函数　(67)
4.2.6　重载构造函数　(68)
4.2.7　拷贝构造函数　(70)
4.2.8　构造函数与析构函数调用的顺序　(75)
4.2.9　小结与建议　(76)
4.3　对象数组与对象指针　(76)
4.3.1　对象数组　(76)
4.3.2　对象指针　(80)
4.3.3　this 指针　(83)
4.3.4　小结与建议　(85)
4.4　向函数传递对象　(85)
4.5　综合训练　(87)
训练 1　(87)
训练 2　(90)
训练 3　(94)
训练 4　(97)
4.6　本章小结　(100)

思考与练习题 …… (100)
第5章　静态成员与友元 …… (109)
5.1　静态成员 …… (109)
5.1.1　静态成员的必要性 …… (109)
5.1.2　静态数据成员 …… (110)
5.1.3　静态成员函数 …… (111)
5.1.4　静态成员的使用 …… (112)
5.2　友元 …… (115)
5.2.1　需要友元的原因 …… (115)
5.2.2　友元函数 …… (116)
5.2.3　友元成员 …… (121)
5.2.4　友元类 …… (123)
5.2.5　友元的使用 …… (124)
5.3　类对象作为成员 …… (126)
5.4　综合训练 …… (130)
训练1 …… (130)
训练2 …… (131)
5.5　本章小结 …… (132)
思考与练习题 …… (132)
第6章　派生类与继承 …… (137)
6.1　派生类的概念 …… (137)
6.1.1　继承的意义 …… (138)
6.1.2　派生类的声明 …… (138)
6.1.3　保护继承与私有继承 …… (142)
6.1.4　继承的访问控制 …… (143)
6.1.5　保护成员的作用 …… (148)
6.2　派生类的构造函数和析构函数 …… (148)
6.2.1　派生类构造函数和析构函数的执行顺序 …… (149)
6.2.2　派生类构造函数和析构函数的构造规则 …… (151)
6.3　多重继承 …… (156)
6.3.1　多继承的过程 …… (156)
6.3.2　继承的二义性 …… (159)
6.3.3　虚拟继承 …… (161)
6.3.4　多继承构造函数和析构函数的执行顺序 …… (167)
6.4　综合训练 …… (170)
训练1 …… (170)
训练2 …… (173)
训练3 …… (174)
6.5　本章小结 …… (176)

思考与练习题 …… (176)
第7章 多态性 …… (181)
7.1 多态的思考方式 …… (182)
7.2 函数重载 …… (184)
7.3 运算符重载 …… (188)
7.3.1 为什么需要运算符重载 …… (188)
7.3.2 如何进行运算符重载 …… (188)
7.3.3 运算符函数作为成员函数 …… (191)
7.3.4 运算符函数作为友元函数 …… (194)
7.3.5 重载方式的选择 …… (197)
7.3.6 增量运算符的重载 …… (198)
7.3.7 转换运算符重载 …… (200)
7.3.8 赋值运算符重载 …… (204)
7.4 函数及运算符重载的限制 …… (207)
7.4.1 函数重载限制 …… (207)
7.4.2 运算符重载限制 …… (208)
7.4.3 重载限制与二义性 …… (209)
7.5 模板 …… (211)
7.5.1 模板概念的引入 …… (211)
7.5.2 使用模板的原因 …… (211)
7.5.3 函数模板 …… (213)
7.5.4 类模板 …… (219)
7.5.5 模板高级应用入门 …… (223)
7.6 函数重载、函数模板和类型转换 …… (228)
7.7 虚函数 …… (230)
7.7.1 覆写 …… (230)
7.7.2 指向派生类的基类指针与引用 …… (233)
7.7.3 虚函数的定义与运用 …… (237)
7.7.4 纯虚函数与抽象类 …… (243)
7.8 编译时多态与运行时多态 …… (251)
7.9 综合训练 …… (252)
训练1 …… (252)
训练2 …… (253)
训练3 …… (255)
训练4 …… (259)
7.10 本章小结 …… (267)
思考与练习题 …… (267)
第8章 I/O流 …… (273)
8.1 C++的流和流类库 …… (273)

8.1.1　C++的流 …… (273)
8.1.2　流类库 …… (274)
8.2　输入输出流及其格式控制 …… (275)
8.2.1　屏幕输出操作 …… (275)
8.2.2　键盘输入操作 …… (277)
8.2.3　输入输出格式控制 …… (280)
8.3　文件流类 …… (283)
8.3.1　文件的打开和关闭操作 …… (283)
8.3.2　文本文件的读写操作 …… (285)
8.3.3　二进制文件的读写操作 …… (286)
8.4　综合训练 …… (287)
8.5　本章小结 …… (289)
思考与练习题 …… (290)
附录1　VC++开发环境简介 …… (291)
1　用 Visual C++ 6.0 创建 C++源程序 …… (291)
2　Visual C++ 6.0 MFC 特点介绍 …… (296)
3　用 Visual C++ 6.0 创建 MFC 源程序的例子 …… (297)
3.1　建立程序框架 …… (297)
3.2　程序建立示例 …… (304)
附录2　Code::Blocks 开发环境简介 …… (306)
附录3　运算符优先级 …… (310)
附录4　综合练习 …… (311)
综合练习1　象棋类 …… (311)
综合练习2　职工档案管理系统 …… (313)
综合练习3　较完整的日期类 …… (317)
综合练习4　矩阵类 …… (322)
综合练习5　电话簿管理程序 …… (329)
参考文献 …… (340)

第1章 绪 论

软件开发过程是一种高密集度的脑力劳动，开发方需要投入大量的人力、物力和财力。由于早期的面向过程软件开发模式及技术不能适应软件发展的需要，致使大量质量低劣的软件产品涌向市场，有的甚至在开发过程中就夭折了。国内外在开发一些大型软件系统时，遇到了许多困难，有的系统最终彻底失败了；有的系统则比原计划推迟了很多年，而且费用严重超过了预算；有的系统无法达到用户当初的期望；有的系统则无法进行修改维护，这些在软件开发过程中出现的情况都称之为“软件危机”。出现软件危机的原因是多方面的，如软件需求变化频繁、开发方法落后等。人们尝试从不同角度、不同层次来解决软件危机，如严格确定软件需求、采用新的开发模型、采用计算机辅助工具等。

面向对象程序设计就是在这样的环境中产生的。在面向对象程序设计语言(下文简称为面向对象语言)产生之后，面向对象程序设计逐步成为软件开发的主流，其中所蕴涵的面向对象的思想不断向开发过程的上游和下游发展，形成现在的面向对象分析、面向对象设计、面向对象测试等软件开发方法，并一起逐步发展为面向对象软件开发方法。本章包含的主要内容如图 1.1 所示。

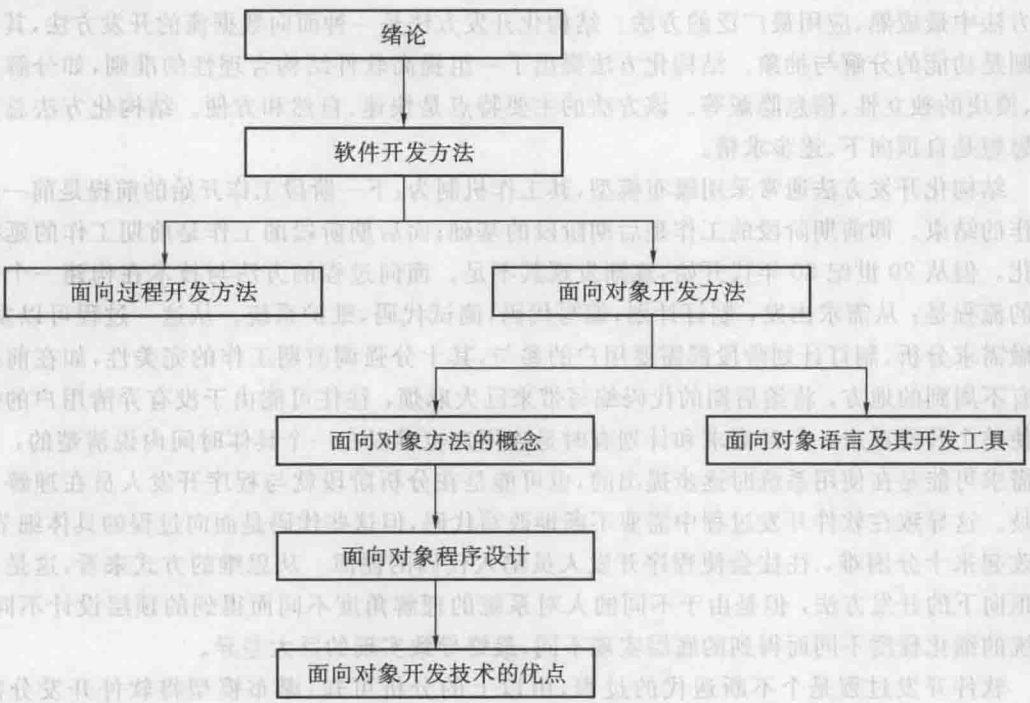

图 1.1 绪论的知识导图

1.1 软件开发方法

关于“软件危机”的一些具体案例。

案例1:IBM公司的OS/360,共约100万条指令,花费了5000个人年;经费达数亿美元,结果却令人沮丧,错误达2000个以上,系统根本无法正常运行。OS/360系统的负责人Brooks这样描述开发过程的困难和混乱:“像巨兽在泥潭中作垂死挣扎,挣扎得越猛,泥浆就沾得越多,最后没有一个野兽能够逃脱淹没在泥潭中的命运。”

案例2:1962年6月,美国飞往金星的第一个空间探测器(水手Ⅰ号),因计算机导航程序的一条语句出错,致使空间探测器偏离航线无法取得成功。

案例3:阿波罗8号由于太空飞船的一个计算机软件错误,造成存储器的一部分信息丢失;阿波罗14号在飞行的10天中,出现了18个软件错误。

以上案例表明,在软件开发的过程中一定要像其他行业那样,需要一定的方法来指导开发过程,这就是软件开发方法。软件开发方法很多,一般归结为面向过程的开发方法和面向对象的开发方法。

1.1.1 面向过程的开发方法

面向过程的开发方法包含分析、设计、实现、确认(测试)、演化(维护)等活动。典型的面向过程的开发方法有:Jackson方法、结构化开发方法等。其中结构化开发方法是现有的软件开发方法中最成熟、应用最广泛的方法。结构化开发方法是一种面向数据流的开发方法,其基本原则是功能的分解与抽象。结构化方法提出了一组提高软件结构合理性的准则,如分解和抽象、模块的独立性、信息隐蔽等。该方法的主要特点是快速、自然和方便。结构化方法总的指导思想是自顶向下、逐步求精。

结构化开发方法通常采用瀑布模型,其工作机制为:下一阶段工作开始的前提是前一阶段工作的结束。即前期阶段的工作是后期阶段的基础;而后期阶段的工作是前期工作的延续和深化。但从20世纪80年代开始,逐渐发现其不足。面向过程的方法与技术在构建一个系统时的流程是:从需求出发,制订计划,编写代码,测试代码,维护系统。从这一过程可以发现,在做需求分析、制订计划阶段都需要用户的参与,其十分强调前期工作的完美性,如在前期阶段有不周到的地方,将给后期的代码编写带来巨大麻烦,往往可能由于没有弄清用户的需求而使整个开发重来。但是需求和计划有时是连用户也难以在一个具体时间内说清楚的,用户的需求可能是在使用系统时逐步提出的,也可能是在分析阶段就与程序开发人员在理解上有分歧。这导致在软件开发过程中需要不断地改写代码,但这些代码是面向过程的具体细节的,修改起来十分困难,往往会使程序开发人员陷入代码的泥潭。从思维的方式来看,这是一种自顶向下的开发方法,但是由于不同的人对系统的理解角度不同而得到的顶层设计不同,对系统的细化程度不同而得到的底层实现不同,最终导致实现的巨大差异。

软件开发过程是个不断迭代的过程,由以上的分析可知:瀑布模型将软件开发分割为独立的几个阶段,不能从本质上反映软件开发过程本身的规律。此外,过分强调复审,并不能完全避免较为频繁的变动。尽管如此,瀑布模型仍然是开发软件产品的一个行之有效的工程模型。

1.1.2 面向对象的开发方法

面向对象方法的出现在一定程度上弥补了面向过程方法的不足。面向对象技术是一种设计和构造软件的技术，它使计算机解决问题的方式更符合人类的思维方式，并且能更直观地描述客观世界。面向对象方法通过增加代码的可重用性、可扩充性和程序自动生成功能来提高编程效率，很大程度上减少了软件维护的开销。20世纪80年代末以来，随着面向对象技术成为研究的热点，出现了数十种支持软件开发的面向对象方法。

随着Windows操作系统和Internet的普及，面向对象程序设计技术得到越来越广泛的应用，面向对象方法和技术已成为计算机领域的主流技术。面向对象方法与技术起源于面向对象的编程语言(OOPL)。20世纪80年代大批OOPL的出现和不断提高标志着OO(面向对象)技术开始走向繁荣和实用。20世纪80年代后期到90年代相继出现了一大批关于面向对象的分析与设计的论文和专著。进入20世纪90年代以来，在学术界面向对象的方法与技术已成为最受关注的研究热点，越来越多的学术会议和学术期刊把面向对象列为主要议题之一，并且每年都有许多关于面向对象的专著出版。在产业界，几乎所有新的软件开发，都全面地或部分地采用面向对象技术。

面向对象的方法与技术和面向过程的方法都在一定程度上缓解了“软件危机”现象，但是前者弥补了后者在软件开发中遇到的种种问题。将面向对象的方法与技术运用在需求分析阶段，开发人员可以从一个用户清楚的需求出发，构造实现这种需求的对象，规定其操作，由多个对象抽象为类，由类构造超类，由类派生出实例，一步步逐步构造系统。由于对象与其操作是封装的，所以在对某个对象修改时只涉及该对象、该类的细节，不影响整个系统。这是一种自底向上的开发方法，用户看到的是与系统更加贴近的内容。

提示：为了克服“软件危机”，提高软件的开发效率，人们不断地从实践经验和理论中探索软件工程的新途径。上面提到的面向过程和面向对象开发方法并不是相互排斥的，而是相互补充、相互促进的。在实际的软件开发过程中，要根据实际情况来选择适当的开发方法，这样才能确保开发出成功的软件。

1.2 面向对象的概念和面向对象程序设计

面向对象方法是一种运用对象、类、继承、封装、聚合、消息传送、多态等概念来构造系统的软件开发方法。其基本思想是从现实世界中客观存在的事物(即对象)出发来构造系统，并在系统构造中尽可能运用人类的自然思维方式。

1.2.1 面向对象方法的概念

在面向对象方法中有两个重要的和最为基础的概念——类与对象。在面向对象的程序设计中，“对象”是系统中的基本运行实体，是有特殊属性(数据)和行为方式(方法)的实体。即对象由两个部分构成：一组包含数据的属性和对属性中包含的数据进行操作的方法。也可以说，“对象”是将某些数据代码和对该数据的操作代码封装起来的模块，是有特殊属性(数据)和行为方式(方法)的逻辑实体。对象包含了数据和方法，每个对象就是一个微小的程序。

类是对具有公共方法和一般特殊性的一组基本相同对象的描述。一个类实质上定义

的是一种对象类型，由数据和方法构成，其描述了属于该类型的所有对象的性质。对象是在执行过程中由其所属的类动态生成的，一个类可以生成不同的对象。在面向对象的程序设计中，对象是构成程序的基本单位，每个对象都应该属于某一类，对象也可称为类的一个实例。

1.2.2 面向对象的程序设计

面向对象的程序设计就是用一种面向对象的编程语言（如C++）把软件系统书写出来。在面向对象编程中，程序被看作是相互协作的对象集合，对象间的通信是通过消息来实现的，每个对象都是某个类的实例，所有的类构成一个通过继承关系相联系的层次结构。面向对象的编程方法有以下4个基本特征。

(1)抽象。抽象就是忽略一个主题中与当前目标无关的那些方面，以便充分地注意与当前目标有关的方面。

(2)继承。继承是一种连接类的层次模型，并且允许和鼓励类的重用，其提供了一种明确表述共性的方法。这种特征体现了一般和特殊的关系。继承性很好地解决了软件的可重用性问题。

(3)封装。封装是对象和类的重要特性。封装把过程和数据封闭起来，只对外界提供访问开放数据的接口。封装保证了模块具有较好的独立性，使得程序维护修改较为容易。

(4)多态。多态是指允许不同类的对象对同一消息作出响应。多态性语言具有灵活、抽象、行为共享、代码共享的优势，解决了应用程序的函数同名问题。

1.2.3 面向对象开发技术的优点

面向对象开发技术的优点如下：

(1)与人类的思维习惯类似。面向对象技术对问题空间进行自然分割，以更接近人类思维的方式建立问题域模型，以便对客观实体进行结构模拟和行为模拟，从而使设计出的软件尽可能直接地描述现实世界。

(2)可重用性好。应用程序更易于维护、更新和升级。继承和封装使得应用程序的修改带来的影响更加局部化。

(3)具有良好的稳定性。运用面向对象技术，软件开发时间短，效率高，所开发的程序更稳定。由于面向对象编程的可重用性，可以在应用程序中大量采用成熟的类库，从而缩短了开发时间。

面向对象技术有诸多的优点，但是如果没有很好的软件开发工具支持的话，则就会增加开发的难度。为此，本书采用当前比较流行的C++开发工具——VC++和Code::blocks（部分代码只能在VC++6.0下运行）作为本书所有例子的开发工具。

1.3 面向对象语言(C++)及开发工具

C++语言是从C语言中孕育出来的。C语言是一种面向过程的语言，以其灵活和高效而著称。Bjarne Stroustrup对C语言的内核进行了必要的修改，使其满足了面向对象模型的要求。于是一种新的语言——C++语言应运而生。

C++的一个设计目标就是通过支持面向对象来增强C语言的功能。C++只是在C语言的基础上添加了一些面向对象概念的新语法规则，并不影响原有C语言所有的语法规则，所以C++是完全兼容C的，原来用C编程的程序可以不做修改或者做很少的修改就可以在C++环境中正确地执行。因此，也可以把C++认为是一个混合型的语言，它既支持结构化的编程(C++包含C的全部语法规则)，同时也支持面向对象的编程(C++添加了适应面向对象的语法)。

VC++是目前较为流行的C++集成开发环境(IDE)，是由Microsoft公司开发的。该开发环境除了提供标准的C++语言的库函数以外，还提供了MFC(微软基础类库)，方便用户创建一些高级特性的类，使开发人员在一定程度上避免了诸如写任何一个类都要从头开始写的繁琐过程。

Code::Blocks是一个开源的C/C++的集成开发环境。Code::Blocks由纯粹的C++语言开发完成，开发环境具有较快的响应速度并具有跨平台性，支持在不同的操作系统下运行。Code::Blocks是目前进行C/C++程序开发的主流开发环境之一。

本书内的所有源代码均可在VC++6.0和Code::Blocks下运行(但部分代码只能在VC++6.0下运行)。

在VC++和Code::blocks里面创建一个类是很轻松的一件事情，因为指定了类名以后，就会自动生成一些代码。除了自动生成代码以外，开发环境还提供了一个良好的编译、连接、执行以及调试环境，在开发环境中书写完代码以后，可以马上点击工具栏中相应的按钮执行相应的功能。值得称道的是，VC++和code::Blocks为开发人员提供了基于图形界面的调试环境，开发人员通过设定相应的断点，并跟踪所有变量的变化，就可以为程序排错，为编写出优秀的代码提供了便利的条件。关于VC++和Code::Blocks的使用方法参见附录1和附录2。

思考与练习题

1. 软件开发方法包括哪些?
2. 面向过程的开发方法和面向对象的开发方法的优缺点有哪些?
3. 试列出面向对象的编程方法的基本特征。
4. 面向对象开发技术有哪些优点?
5. 什么是VC++?

第 2 章　面向对象的程序设计

面向对象程序设计方法(Objected Oriented Programming,OOP)的主要出发点是弥补面向过程程序设计方法中的一些缺点。OOP 把数据看作程序开发中的基本元素,并且不允许其在系统中自由流动。它将数据和操作这些数据的函数紧密地连接在一起,并保护数据不会被外界的函数意外地改变。OOP 将问题分解为一系列"实体"——这些"实体"被称为对象(object),围绕这些实体建立数据和函数。本章包含的主要内容如图 2.1 所示。

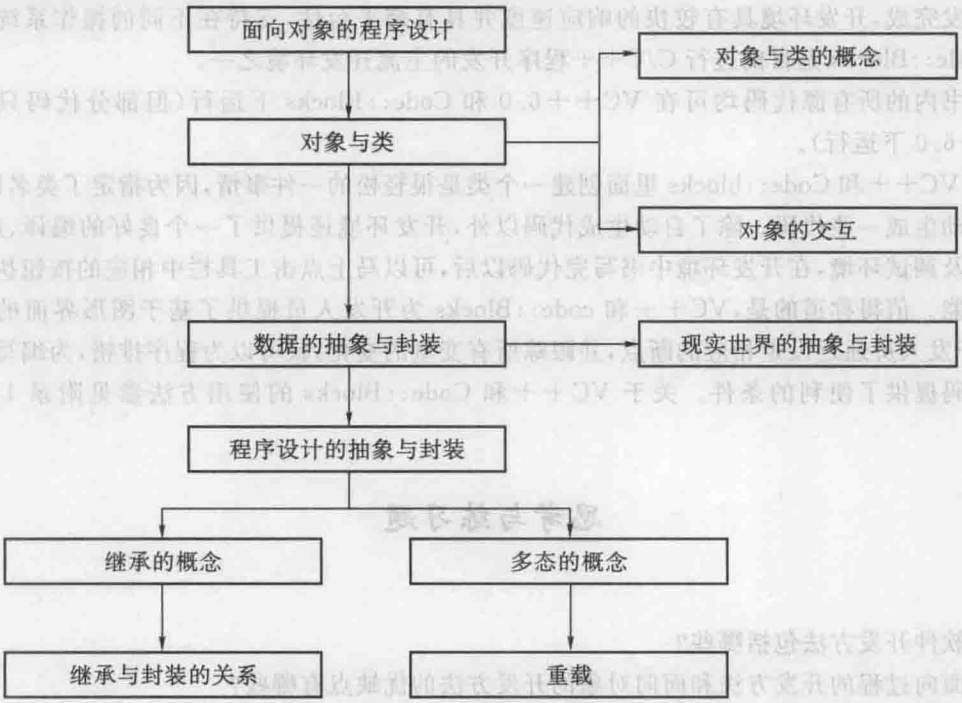

图 2.1　面向对象程序设计的知识导图

2.1　对象与类

2.1.1　对象与类的概念

面向对象不仅是一种编程的方法,同时也是一种看待世界和分析问题的思想,并且还是对实际问题建模的方法和工具。面向对象方法的思考方式与人的思维方式类似,即面向对象思想使用人类习惯的思维方式来描述需要解决的问题。人类在认识事物的时候最基本的单元——实体,对应面向对象方法中的对象,是面向对象方法中最基本的一个概念。

对象是客观世界中的一个实体。世界和宇宙就是由许多不同的对象构成的，一个人、一辆车都是一个对象。需要注意的是，对象是一个具体存在的实体，不是抽象的概念，例如一辆黄色的奔驰轿车是一个对象，但“车”这个抽象的概念并不是一个对象（其为一个类，这将在之后介绍类的概念时提到），只有具体化的真实存在的实体才可以被叫做对象。

对象包含以下几个内容。

(1)对象的名字。客观世界中没有两个实体是完全一样的，同样地，当用对象来表示这些实体的时候，也要用不同的对象名字加以区分。

(2)对象的属性。属性是对实体某一方面的描述，属性表示了对象包含的数据。实体之间的区别就在于属性的不同，这种不同表现为两种方式，一种是属性的种类不同，如学生具有学号这个属性，老师就没有；第二种不同是属性的内容不同，例如两个学生都有学号这个属性，但学号不同。

(3)对象的操作。对象的操作指的是对象能够进行的行为。例如“一辆黄色的奔驰轿车”这个对象就具有行驶操作。

从对象的内容可以看出，数据（属性）和操作被紧密地联系在一起，这就克服了结构化设计中数据和操作分离的弱点。同时这种方式也比较符合人的认知过程，人在看到小汽车的时候很自然地会联想到它的功能（如小汽车具有行驶功能）。客观实体的属性和操作在人脑中本来就是被放在一起处理的，使用符合对象认识问题过程的方法，有利于程序设计者描述复杂的现实问题。

下面是一个对象的例子。

对象名称：小明

对象属性：

学历：大学

年龄：21

专业：历史系

对象操作：

上课

吃饭

人在认识世界的各个对象时，并不是把每个对象都当做一个孤立的个体，如一只黄猫和一只白猫是两个不同的对象，人们在认识它们的时候会认为它们都是猫。也就是说，人在认识对象的时候往往能看到对象间的共性，通过这些共性人会对对象进行分类，这反映在面向对象方法中就是“类”的概念。

类可以说是对象的模型，用一个模型便能建立许多类似的对象。这种关系就类似于月饼和月饼模，一旦制好了月饼模，就可以成批地制作相同的月饼了。

下面是个类的例子。

类名称：学生

类属性：

学历

年龄

专业

对象操作：

上课

吃饭

对这个类的例子和前面对象的例子进行比较。

(1)名称并不指某个实体，例如上例中的类名称“学生”是指一个范围的人，而不是某一个人；而对象是具体的，对象的名称各不相同。

(2)属性表示一个范围，其值是不确定的，如上个例子中的属性年龄，只表示一个范围，一般取1～120；而对象也有年龄属性，但对象的年龄属性是一个确定的值21。

(3)类和对象都有操作，且二者的操作是相同的。

从上面的比较可以看出，类与对象的关系就是抽象与具体的关系。类是一个框架，是对具有同类型属性和操作的对象组的抽象；对象则是类的实例化，给类这个框架中的属性填上一组确定的值就可以得到一个对象。

2.1.2　对象的交互

在面向对象设计中，所有解决问题的函数都被放进了一个个类(和对象)中。与结构化方法相比，面向对象方法同样将问题分解成多个子问题，只是和结构化的分解方法不同，面向对象是把功能函数按照其所属的对象分类。也就是说，哪个对象实现这个功能，这个功能的函数就分给哪个对象(在面向对象中函数被称为方法)。这样的分解使得功能函数不再按实现功能的结构来排列，完成一个功能可能要涉及多个对象中的方法，一个对象往往不能解决问题，需要几个对象协作来完成。

结构化编程中，主程序掌握着执行的顺序，因为所有的函数都是被主函数逐级调用返回的，函数是按照功能分模块排放的，解决一个问题所用的函数都放在一起，呈现分层的关系，控制流程简单。但面向对象中，方法是按照对象排放的，当一个方法想调用其他对象中的方法时，这两个对象就要发生交互，所以面向对象的控制流程相对复杂，这种复杂关系正是对客观世界中复杂的实体关系的真实反映。这就好比军队和企业，军队的管理制度像结构化设计方法，上下级明确，下级绝对服从上级，完成上级指派的任务，最上层的领导很容易控制全局；企业的管理制度则更像面向对象的设计方法，各个部门之间大多属于平行的关系，如果一个项目要跨几个部门，就需要几个部门进行交互，没有哪个部门有绝对的主导权，每个部门都只做好自己应做的部分，再把需要别的部门做的任务传递给相应的部门并等待回应。

对象之间的交互是通过消息传递机制进行的。传递只是抽象的叫法，并不是真的传递一个完整的消息，对别的对象中方法的调用也是消息传递。这里的消息发送并不强调传递形式，消息传递只是一种通知，告诉其他对象要使用其中的方法。这种程序执行控制的方法和现实中解决一个问题的方法类似。在生活中，当遇到不能做的事情的时候也会发消息给其他人来帮助解决。例如，去买火车票，只是给窗口工作人员发送了一个口头消息，消息包括了车票的信息，然后就等待工作人员来完成登记、打印车票等一系列任务。

面向对象的设计靠消息传递来推动程序的运行，这种方法间的耦合程度显然低于结构化设计中的函数调用。这样的执行控制形式是由对象的特性和封装性(下一节将提到)决定的。面向对象方法是使用信息发送的方式完成对象交互，将分散在各个对象中的方法联合起来解决问题。