普通高等教育“十一五”国家级规划教材

21世纪高等学校计算机专业核心课程规划教材

Visual C++ 面向对象程序设计教程与实验(第3版)

温秀梅　丁学钧　主编

清华大学出版社

普通高等教育“十一五”国家级规划教材

21世纪高等学校计算机专业核心课程规划教材

Visual C++面向对象程序设计教程与实验（第3版）

温秀梅　丁学钧　主编

清华大学出版社
北京

内 容 简 介

本书将C++面向对象程序设计的思想和方法作为重点，结合例题进行了详细的分析解释，除在每章后附有习题外，还在附录中整合了实验设计。全书结构严谨、通俗易懂，兼有普及与提高的双重功能。

全书由三部分组成。第一部分第1～8章结合实例深入浅出地讲解了C++面向对象程序设计的思想和方法；第二部分第9～12章是关于Visual C++的MFC程序设计；第三部分附录包括重要的实验内容设计及Visual C++ 6.0环境介绍，这是掌握编程语言的重要环节。

本书遵循少而精的原则，力求做到版面清晰、结构紧凑、信息含量高，因此特别适宜作为计算机专业本科教材。同时，还可以作为自学或函授学习的参考书。

图书在版编目(CIP)数据

Visual C++面向对象程序设计教程与实验/温秀梅，丁学钧主编. —3版. —北京：清华大学出版社，2014
21世纪高等学校计算机专业核心课程规划教材
ISBN 978-7-302-33146-9

Ⅰ. ①V…　Ⅱ. ①温…　②丁…　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(2013)第159636号

责任编辑：魏江江
封面设计：杨　兮
责任校对：白　蕾
责任印制：宋　林

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦A座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课 件 下 载：http://www.tup.com.cn，010-62795954
印 刷 者：北京富博印刷有限公司
装 订 者：北京市密云县京文制本装订厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：21.5　　**字　　数**：529千字
版　　次：2005年9月第1版　2013年12月第3版　　**印　　次**：2014年1月第1次印刷
印　　数：34501～36500
定　　价：34.50元

产品编号：052586-01

第三版前言

《Visual C++面向对象程序设计教程与实验(第二版)》教材自2009年出版后,受到了广大读者的好评,并被多所高校选作教材,作者也获得了许多宝贵的意见和建议。为了更好地为广大读者服务,秉承严谨细致的态度,决定对第二版内容予以补充和修改。

修编后的教材根据教学改革、实践教学的需要及教师多年的教学经验,适当修改增补了部分例题和习题,并与第二版在结构安排、编写风格等方面保持一致,使采用第二版的教师可以很容易地过渡到新版教材。

作为教材,使用者可以根据自己的需求,选取相应的内容进行教学。如果前面开设的是"C++语言程序设计"课程,而不是"C语言程序设计"课程,则可以略过第2章和第7章的部分内容。如果课时不足,可将第9章到第12章总体上通过一个实例进行讲解,其余的内容可以让学生自学,也可以在课程的开始阶段布置学生通过实例进行自学。

作为本书的姐妹篇,我们将出版与本教程配套的学习指导和习题解答一书,给出教程的学习指导、所有习题的参考答案及每个实验的参考程序,供教师和学生参考。

本版教材由河北建筑工程学院温秀梅、丁学钧教授任主编,李建华、庞慧、赵巍任副主编,参加编写的还有刘晓群、孟凡兴、狄巨星、宋淑彩、高丽婷、周丽莉、岳杰、司亚超、孙皓月、王庆林,全书最后由温秀梅进行审校并统稿。

本书在编写和出版过程中得到了清华大学出版社的大力支持和帮助,在此表示诚挚的感谢!

由于作者水平有限,书中难免有疏漏和错误之处,恳请广大专家和读者批评指正。

编　者

2013年10月

第二版前言

《Visual C++面向对象程序设计教程与实验》教材自2005年出版后，受到读者好评，并被多所高校选作教材。根据师生反馈意见，我们及时修编原书，形成了本版教材，该教材被列为普通高等教育“十一五”国家级规划教材。

修编后的教材保持了与原书内容、风格一致，使采用原书的教师可以自然地过渡到新版教材。这次修编主要就以下几个方面进行了改进：

根据教学改革、实践教学的需要和教师多年的教学经验，适当修改、增补了第3章、第5章、第6章和第7章部分语言和问题描述的表达方式；增加了一套模拟考试题，并增加两套模拟考试题的答案；修改了部分例题的源程序，对部分例题进行了详细分析；更正了原书中的错误。

作为教材，使用者可以根据自己的需求，选取相应的内容进行教学。如果前面开设的是“C++语言程序设计”课程，而不是“C语言程序设计”课程，则可以略过第2章不讲，第7章的部分内容也可以不讲。如果课时不够，第9章到第12章总体通过一个实例讲解，其余的内容可以让学生自学，也可以在课程的开始先布置让学生自己上机通过实例学习。

本版教材由温秀梅、丁学钧任主编，刘建臣、高丽婷、赵巍任副主编。参加编写的有丁学钧（第1～2章）、温秀梅（第3～10章、附录F）、刘建臣（第11章）、高丽婷（第12章）、李建华（附录A、B）、宋淑彩（附录C），赵巍（附录D）、祁爱华（附录E）、全书最后由温秀梅、丁学钧、刘建臣进行审校并统稿。

由于作者水平有限，书中难免有疏漏和错误之处，恳请广大专家和读者指正。

编　者

2009年1月

第一版前言

作为一种计算机语言，C++有很多优点。它既可以进行过程化程序设计，也可以进行面向对象程序设计，很多复杂的算法和设计可以比较容易地用C++面向对象的思想来实现。

在编写本书之前，作者已在高校从事了多年的“C++语言程序设计”、“面向对象程序设计”教学及科研工作，对于该语言的概念、功能及应用有着较深入的理解和丰富的实践经验。在教学过程中，我们发现很多教材在讲解C++语言时既包括结构化程序设计又包括面向对象程序设计，而在面向对象程序设计部分讲得不透彻，不适合计算机专业的学生学习。故组织编写了这本教材，旨在通过本教材在内容安排、教学深度及实验要求等方面满足计算机专业本科生“面向对象程序设计”课程的教学要求。

作为一本教材，本书具有如下特点：

(1) 本书在结构上将C++面向对象程序设计的思想和方法作为重点，并结合例题进行了详细的分析解释，除在每章后附有习题外，还在附录中整合了实验设计。使全书结构严谨、通俗易懂，兼有普及与提高的双重功能。

(2) 本书没有涉及面向过程的程序设计内容，只在第2章中讲解了C++在结构化程序设计方面对C的扩充，因此学生应在学习了相关的基础知识之后再使用本教材。

(3) 本书以现代教育理念为指导，在讲授方式上注意结合应用开发实例，注重培养学生理解面向对象程序设计思想，以提高分析问题和解决实际问题的能力。

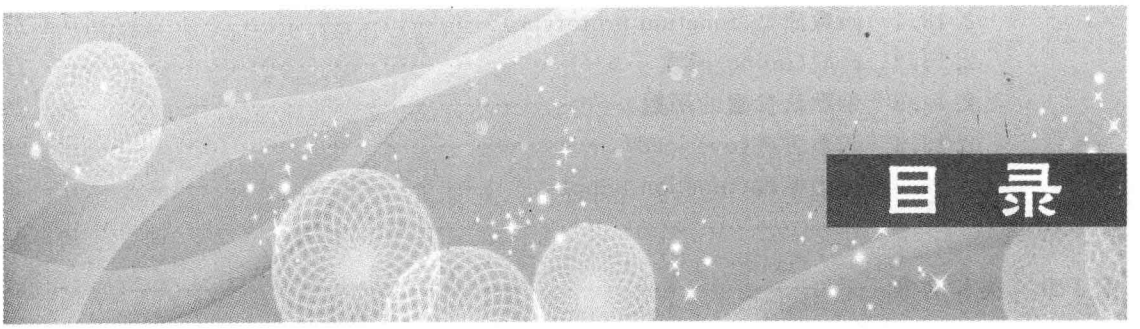
(4) 本书中的所有程序都是在VC 6.0环境下编译调试通过的。

本书由温秀梅、丁学钧主编并统稿，孟凡兴、刘建臣任副主编。参加编写的有丁学钧(第1～2章)，温秀梅(第3～11章、附录E)，孟凡兴(第12章)，李建华(附录A、B)，宋淑彩(附录C)，周丽莉(附录D)，刘建臣担任本书的审校工作。参加本书部分内容编写工作的还有赵巍、徐晓君、岳杰、庞慧、董颢霞、王庆林、司亚超、刘海龙等。在本书的大纲讨论和分工编写过程中，我们始终互相帮助，彼此鼓励，是一次非常难忘的经历。

在此还要特别感谢我们的学生梁金龙，他为本书做了很多前期工作。

由于时间仓促，加之水平有限，书中难免有疏漏和错误之处，恳请广大读者和专家指正。

编者



目 录

第1章

绪　论

面向对象程序设计是软件系统设计与实现的新方法，这种方法是通过增加软件的可扩充性和可重用性来提高程序员的生产能力，控制软件的复杂性，降低软件维护的开销。因此，它的应用使软件开发的难度和费用大幅度降低，已为世界软件产业带来了革命性的突破。

1.1　面向对象方法的起源

“对象”一词在现实生活中经常会遇到，它表示现实世界中的某个具体的事物。社会的进步和计算机科学的发展是相互促进的，随着计算机的普及和应用，人们越来越希望能更直接地与计算机进行交互，而不需要经过专门学习和长时间训练后才能使用它。这使得软件设计人员的负担越来越重，软件的实现越来越复杂，并且对计算机领域自身的发展也提出了新的要求。利用传统的程序设计思想无法满足这一要求时，人们就开始寻求一种能帮助人类解决问题的自然方法，这就是“面向对象”技术。

20 世纪 50 年代的程序都是用指令代码或汇编语言编写的，这种程序的设计相当复杂，编制和调试一个稍大一点的程序常常要花费很长时间，培养一个熟练的程序员更需经过长期训练和实践，这种局面严重影响了计算机的普及和应用。

20 世纪 60 年代高级语言的出现大大简化了程序设计，缩短了软件开发周期，显示出了强大的生命力。此后，编制程序已不再是专业软件人员才能做的事了，一般工程技术人员花较短的时间学习后，也可以使用计算机解题。这个时期，随着计算机日益广泛地渗透到各个学科和技术领域，一系列不同风格、为不同目标服务的程序设计语言发展起来了，其中较为著名的有 FORTRAN、COBOL、ALGOL、LISP、PASCAL 等十几种语言。高级语言的蓬勃兴起，使得编译原理和形式语言理论日趋完善，这是该时期的主要特征。但是就整个程序设计方法而言，并无实质性的改进。

自 20 世纪 60 年代末到 20 世纪 70 年代初，出现了大型软件系统，如操作系统、数据库，这给程序设计带来了新的问题。大型系统的研制需要花费大量的资金和人力，可是研制出来的产品却可靠性差、错误多、不易维护和修改。一个大型操作系统的研制有时需要每年几千人的工作量，而研制出的系统又常常会隐藏着几百甚至几千个错误。当时，人们称这种现象为“软件危机”。

为了克服 20 世纪 60 年代出现的软件危机，1968 年北约组织提出“软件工程”的概念。对程序设计语言的认识从强调表达能力为重点转向以结构化和简明性为重点，将程序从语句序列转向相互作用的模块集合。1969 年，E. W. Dijkstra 首先提出了结构化程序设计的概念，他强调从程序结构和风格上来研究程序设计。在软件工程迫切需要改进的背景下，20 世纪70 年代结构化语言获得蓬勃发展并得到广泛应用。使用结构化程序设计方法可显著地减少软件的复杂性，提高软件的可靠性、可测试性和可维护性。经过几年的探索和实践，结构化程序设计

的应用确实取得了成效，用结构化程序设计的方法编写出来的程序不仅结构良好、易写易读，而且易于验证其正确性。

进入20世纪80年代，由于一系列高技术的研究，如第五代计算机、计算机辅助制造(CAM)和知识工程等领域的研究，都迫切要求大型的软件系统作支撑。它们所用的数据类型也超出了常规的结构化数据类型的范畴，并且提出了对图像、声音、规则等非结构化信息的管理。为了满足这些应用领域的需要，就迫切要求软件模块具有更强的独立自治性，以便于大型软件的管理、维护和重用。由于结构化语言的数据类型较为简单，采用过程调用机制也不够灵活，独立性较差，所以不能胜任对非结构化数据的定义与管理。

为了适应高技术发展的需要，消除结构化程序设计语言的局限，自20世纪80年代以来，出现了面向对象程序设计流派，研制出了多种面向对象程序设计语言(Object Oriented Programming Language，OOPL)，如Ada、Smalltalk、C++语言和当前使用在Internet上与平台无关的Java语言等。

由于OOPL的对象、类具有高度的抽象性，所以能很好地表达复杂的数据类型，并且OOPL也允许程序员灵活地定义自己所需要的数据类型。类本身具有完整的封装性，可以使用它作为编程中的模块单元，满足模块独立自治的要求。另外，类的继承性和多态性功能更有助于简化大型软件和大量重复定义的模块，从而增强了模块的可重用性，提高了软件的可靠性，缩短了软件的开发周期。

1.2 面向对象是软件方法学的返璞归真

客观世界是由许多具体事物、抽象概念、规则等组成的，人们将任何感兴趣或要加以研究的事、物、概念统称为对象(object)。每个对象都有各自的内部状态和运动规律，不同对象之间通过消息传递进行相互作用和联系就构成了各种不同的系统。面向对象的方法正是以对象作为基本元素的一种分析问题和解决问题的方法。

传统的结构化方法强调的是功能抽象和模块化，每个模块都是一个过程，结构化方法处理问题是以过程为中心的。对象包含数据和对数据的操作，是对数据和功能的抽象和统一。而面向对象强调的是功能抽象和数据抽象，用对象来描述事物和过程，面向对象方法处理问题的过程是对一系列相关对象的操纵，即发送消息到目标对象，由目标对象执行相应的操作。因此面向对象方法是以对象为中心的，这种以对象为中心的方法更自然、更直接地反映现实世界的问题空间，从而具有独特的抽象性、封装性、继承性和多态性的特点，更好地适应了复杂大系统不断发展与变化的要求。

采用对象的观点看待所要解决的问题，并将其抽象为应用系统是极其自然与简单的，因为它符合人类的思维习惯，使得应用系统更容易理解。同时，由于应用系统是由相互独立的对象构成的，系统的修改可以局部化，因此系统维护更加容易。

软件开发从本质上讲就是对软件所要处理的问题域进行正确的认识，并把这种认识正确地描述出来。既然如此，那就应该直接面对问题域中客观存在的事物来进行软件开发，这就是面向对象。另一方面，人类在认识世界的过程中形成的普遍有效的思维方法，在软件开发中也是适用的。在软件开发中尽量采用人们日常生活中习惯的思维方式和表达方式，这就是面向对象方法所强调的基本原则。软件开发从过分专业化的方法、规则和技巧中脱离出来，并重新回到了客观世界，回到了人们的日常思维当中，所以说面向对象方法是软件方法学的返璞归真。

1.3　结构化程序设计与面向对象程序设计

要想真正了解面向对象程序设计，首先需要回顾一下结构化程序设计的含义。

1. 结构化程序设计

结构化程序设计是 20 世纪 60 年代诞生的，在 70 年代到 80 年代已遍及全球，成为软件开发设计所有领域及每个程序员都采用的程序设计方法，它的产生和发展形成了现代软件工程的基础。

结构化程序设计的设计思想是：自顶向下、逐步求精；其程序结构按功能划分为若干个基本模块，这些模块形成一个树状结构；各模块之间的关系尽可能简单，在功能上相对独立；每一模块内部均由顺序、选择和循环三种基本结构组成；其模块化实现的具体方法是使用子程序、过程或函数。

结构化程序设计由于采用了模块分解和功能抽象、自顶向下、分而治之的手段，从而有效地将一个复杂的软件系统的设计任务分成许多容易控制和处理的子任务，这些子任务都是可独立编程实验的子程序模块。每一个子程序都有一个清晰的界面，使用起来非常方便。

结构化程序设计方法虽然具有许多的优点，但它仍是一种面向过程的设计方法，它把数据和过程分离为相互独立的实体，程序员在编程时必须时刻考虑所要处理的数据格式。对于不同的数据格式即使做同样的处理或对相同的数据格式做不同的处理都需要编写不同的程序。因此结构化程序的可重用性不好；另一方面，当数据和过程相互独立时，总存在着用错误的数据调用正确的程序模块或用正确的数据调用错误的程序模块的可能性。因此，要使数据和程序始终保持相容，已成为程序员的一个沉重负担，并且随着软件系统的规模越来越大，程序的复杂性越来越难以控制。上述这些问题，结构化程序设计方法本身是解决不了的，需要借助于下面要讨论的面向对象程序设计方法给予解决。

程序设计的任务是描述问题并解决问题，在结构化程序设计中可以用下面的式子表示程序：

程序 = 数据结构 + 算法 + 程序设计语言 + 语言环境

图 1.1 所示为结构化程序设计中程序的结构。

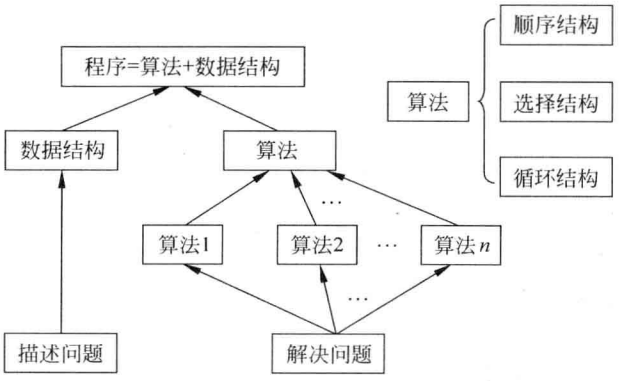


图 1.1　结构化程序设计中程序的结构

2. 面向对象程序设计——程序设计的新思维

面向对象程序设计既吸取了结构化程序设计的一切优点，又考虑了现实世界与面向对象

空间的映射关系，它所追求的目标是将现实世界问题的求解尽可能简单化。

面向对象程序设计将数据及对数据的操作放在一起，作为一个相互依存、不可分割的整体来处理，它采用了数据抽象和信息隐藏技术。它将对象及对对象的操作抽象成一种新的数据类型——类，并且考虑不同对象之间的联系和对象所在类的可重用性。

面向对象程序设计优于传统的结构化程序设计，其优越性表现在，它有希望解决软件工程的两个主要的问题——软件复杂性控制和软件生产率的提高，此外它还符合人类的思维习惯，能够自然地表现现实世界的实体和问题，它对软件开发过程具有重要的意义。

面向对象程序设计能支持的软件开发策略有：

（1）编写可重用代码。

（2）编写可维护代码。

（3）共享代码。

（4）精减已有代码。

有了高质量的可重用代码就能有效地降低软件的复杂度、提高开发效率。面向对象方法，尤其是它的继承性，是一种代码重用的有效途径。开发者在设计软件时可以利用一些已经精心设计好并且经过测试的代码，这些可重用的代码以类的形式被组织存放在程序设计环境的类库中。类库中的这些类的存在，使以后的程序设计过程变得简单，程序复杂性不断降低、正确性不断提高，程序越来越容易理解、修改和扩充。

在面向对象程序设计中可以用下面的式子表示程序：

程序 = 对象 + 对象 + … + 对象
对象 = 算法 + 数据结构 + 程序设计语言 + 语言环境

图 1.2 所示为面向对象程序设计中程序的结构。

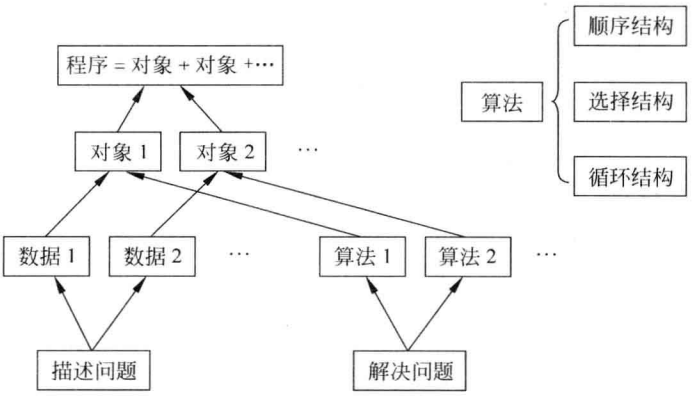


图 1.2　面向对象程序设计中程序的结构

1.4　面向对象的基本概念和面向对象系统的特性

1.4.1　面向对象的基本概念

1. 对象(object)

对象是现实世界中一个实际存在的事物，它可以是有形的(如一支粉笔、一块橡皮等)，也可以是无形的或无法整体触及的抽象事件(如一项计划、一场球赛、一次借书行为等)。对象是

构成世界的一个独立单位，它具有自己的静态特征和动态特征。静态特征可以用某种数据来描述，动态特征即对象所表现的行为或对象所具有的功能。

在面向对象系统中，对象是系统中用来描述客观事物的一个实体，它是构成系统的一个基本单位。一个对象由一组属性和对这组属性进行操作的一组服务构成。属性和服务是构成对象的两个主要因素，属性是一组数据结构的集合，表示对象的一种状态，对象的状态只供对象自身使用，用来描述静态特性。而服务是用来描述对象动态特征(行为)的一个操作序列，是对象一组功能的体现，包括自操作和它操作。自操作是对象对其内部数据(属性)进行的操作，它操作是对其他对象进行的操作。

一个对象可以包含多个属性和多个服务，对象的属性值只能由这个对象的服务存取和修改。对象是其自身所具有的状态特征及可以对这些状态施加的操作结合在一起所构成的独立实体。对象具有如下的特性：

(1) 具有唯一标识名，可以区别于其他对象。

(2) 具有一个状态，由与其相关联的属性值集合所表征。

(3) 具有一组操作方法即服务，每个操作决定对象的一种行为。

(4) 一个对象的成员仍可以是一个对象。

(5) 模块独立性。从逻辑上看，一个对象是一个独立存在的模块，模块内部状态不因外界的干扰而改变，也不会涉及其他模块；模块间的依赖性极小或几乎没有；各模块可独立地被系统组合选用，也可被程序员重用，不必担心影响其他模块。

(6) 动态连接性。客观世界中的对象之间是有联系的，在面向对象程序设计中，通过消息机制，把对象之间动态连接在一起，使整个机体运转起来，便称为对象的连接性。

(7) 易维护性。由于对象的修改、完善功能及其实现的细节都被局限于该对象的内部，不会涉及外部，这就使得对象和整个系统变得非常容易维护。

对象从形式上看是系统程序员、应用程序员或用户所定义的抽象数据类型的变量，当用户定义了一个对象，就创造出了具有丰富内涵的抽象数据类型的实例。

2. 类(class)

在面向对象系统中，并不是将各个具体的对象都进行描述，而是忽略其非本质的特性，找出其共性，将对象划分成不同的类，这一过程称为抽象过程。类是对象的抽象及描述，是具有共同属性和操作的多个对象的相似特性的统一描述体。在类的描述中，每个类要有一个名字标识，用以表示一组对象的共同特征。类的每个对象都是该类的实例。类提供了完整的解决特定问题的能力，因为类描述了数据结构(对象属性)、算法(服务、方法)和外部接口(消息协议)，是一种用户自定义的数据类型。

3. 消息(message)

消息是面向对象系统中实现对象间的通信和请求任务的操作，是要求某个对象执行其中某个功能操作的规格说明。发送消息的对象称为发送者，接受消息的对象称为接收者。对象间的联系，只能通过消息来进行。对象在接收到消息时才被激活。

消息具有三个性质：

(1) 同一对象可接收不同形式的多个消息，产生不同的响应。

(2) 相同形式的消息可以发送给不同对象，所做出的响应可以是截然不同的。

(3) 消息的发送可以不考虑具体的接收者，对象可以响应消息，也可以对消息不予理会，对消息的响应并不是必需的。

对象之间传送的消息一般由三部分组成：接收对象名、调用操作名和必要的参数。

在面向对象程序设计中，消息分为两类：公有消息和私有消息。假设有一批消息发向同一个对象，其中一部分消息是由其他对象直接向它发送的，称为公有(public)消息；另一部分消息是它向自己发送的，称为私有(private)消息。

4. 方法(method)

在面向对象程序设计中，要求某一对象完成某一操作时，就向对象发送一个相应的消息，当对象接收到发向它的消息时，就调用有关的方法，执行相应的操作。方法就是对象所能执行的操作。方法包括界面和方法体两部分。方法的界面就是消息的模式，它给出了方法的调用协议；方法体则是实现这种操作的一系列计算步骤，也就是一段程序。消息和方法的关系是：对象根据接收到的消息，调用相应的方法；反过来，有了方法，对象才能响应相应的消息。所以消息模式与方法界面应该是一致的。同时，只要方法界面保持不变，方法体的改动不会影响方法的调用方式。在C++语言中方法是通过函数来实现的，称为成员函数。

1.4.2 面向对象系统的特性

1. 抽象性(abstract)

面向对象的方法鼓励程序员以抽象的观点看待程序，即程序是由一组对象组成的。程序员可以将一组对象的共同特征进一步抽象出来，从而形成“类”的概念。抽象是一种从一般的观点看待事物的方法，它要求程序员集中于事物的本质特征，而不是具体细节或具体实现。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：从特殊到一般的归纳法和从一般到特殊的演绎法。在归纳的过程中，人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”；在演绎的过程中，人们又把同类的事物，根据不同的特征分成不同的小类，这就是“分类”。对于一个具体的类，它有许多具体的个体，面向对象方法称这些个体为“对象”。

2. 封装性(encapsulation)

所谓数据封装就是指一组数据和与这组数据有关的操作集合组装在一起，形成一个能动的实体，也就是对象。数据封装就是给数据提供了与外界联系的标准接口，无论是谁，只有通过这些接口并使用规范的方式，才能访问这些数据。数据封装是软件工程发展的必然产物，它使得程序员在设计程序时可以专注于自己的对象，同时也切断了不同模块之间数据的非法使用途径，从而减少了出错的可能性。

3. 继承性(inheritance)

从已有的对象类型出发建立一种新的对象类型，使它继承原对象的特点和功能，这种思想是面向对象设计方法的主要贡献。继承是对许多问题中分层特性的一种自然描述，因而也是类的具体化和实现重用的一种手段，它所表达的就是一种不同类之间的继承关系。它使得某类对象可以继承另外一类对象的特征和能力。继承所具有的作用有两个方面：一方面可以减少冗余代码，另一方面可以通过协调性来减少接口和界面。

从继承源上划分继承可分为单一继承(单继承)和多重继承(多继承)，子类对单个直接父类的继承称为单一继承，子类对多于一个直接父类的继承称为多重继承。父类也称为基类或超类，子类也称为派生类。如图1.3所示为单一继承举例，图1.4所示为多重继承举例。