普通高等教育“十五”国家级规划教材

C++程序设计

吴乃陵　况迎辉　李海文　编著

高等教育出版社

普通高等教育“十五”国家级规划教材

C++程序设计

吴乃陵　况迎辉　李海文　编著

高等教育出版社

内容提要

本书是普通高等教育"十五"国家级规划教材，是教育部21世纪初高等理工科教育教学改革项目"电子与电气信息类专业人才培养改革成果的整合与深化"的研究成果。

本书结合理工科专业程序设计课程教学方法的改革，直接讲授面向对象的C++程序设计，并突出学生编程能力的培养。本书体现了最新的C++国际标准ISO14882的规范，内容包括软件概念、基本控制结构、函数、类与对象(封装、继承与多态)、指针与数组、模板与基本数据结构、异常处理和标准模板库。本书同时配有《C++程序设计实践教程》，电子教案和部分程序源码可从网上下载。

本书适用于高等学校理工科各专业的C++程序设计课程，特别是电子与电气信息类等对程序设计要求较高的专业，也可供程序设计爱好者和工程技术人员参考使用。

图书在版编目（CIP）数据

C++程序设计 / 吴乃陵，况迎辉，李海文编著. —北京：高等教育出版社，2003.8
普通高等教育"十五"国家级规划教材
ISBN 7-04-012301-0

Ⅰ.C... Ⅱ.①吴... ②况... ③李... Ⅲ.C语言—程序设计—高等学校—教材 Ⅳ.TP312

中国版本图书馆CIP数据核字（2003）第056896号

出版发行	高等教育出版社	购书热线	010-64054588
社　　址	北京市西城区德外大街4号	免费咨询	800-810-0598
邮政编码	100011	网　　址	http://www.hep.edu.cn
总　　机	010-82028899		http://www.hep.com.cn
经　　销	新华书店北京发行所		
印　　刷	北京市鑫霸印务有限公司		
开　　本	787×1092　1/16	版　　次	2003年8月第1版
印　　张	28.75	印　　次	2003年8月第1次印刷
字　　数	590 000	定　　价	29.50元

前　言

C++程序设计语言是所有程序设计语言中最有活力和应用最广的一种。C++程序设计的教学自20世纪90年代中期开始，逐步成为各高等学校，尤其是理工科专业的程序设计课程的主流。

这本书是教学改革的产物，而且力度较大，力图在教学理念、教学内容、编排顺序和教学方法上有所创新和突破，实现跨越式发展。本书是作为大学理工科第一门程序设计语言课的教材编写的，编写难度更大。但是正因为改革力度大，教材的特点也非常明显。

第一个特点是突出算法描述，强调编程能力的培养。程序设计的教学目的是培养大学生的编程能力，所以我们认为授课的重点应是程序设计而不是语法。在计算机语言层次，人与计算机的意识活动的交流是通过程序设计这个环节来完成的。1976年N.Wirth出版了一本题名为Algorithms + Data Structure = Programs的著作，明确提出算法和数据结构是程序的两个要素，也就是说程序设计主要包括两方面的内容：行为特性的设计和结构特性的设计。行为特性的设计是指完整地描述问题求解的全过程，并精确地定义每个解题步骤，这一过程即是算法的设计；而结构特性的设计是指在问题求解的过程中，计算机所处理的数据及数据之间联系的表示方法。也就是说，授课的重点应是算法和数据结构，而不是语法，语法只是为描述算法服务的。两者并重似乎是个好主意。但是，第一学时有限，目前学时一压再压，而要学的知识一涨再涨，两者并重则学时远远不足；第二学生精力和时间有限，他们还有很多其他课程同时在学，双重点的效果不如单重点。从现有的教材看，也都是侧重一个方面。

第二个特点是突出面向对象。C++虽然是面向对象的语言，但它是从面向过程的C语言发展而来的，并非如Java那样是纯面向对象的语言。从面向过程过渡到面向对象，两者相互分离，这样学生很难在接受面向过程的思想后再顺利接受相对复杂的面向对象的思想。处理好面向过程和面向对象的关系是非常重要的。我们力图把面向对象与面向过程有机地结合起来。面向过程突出了算法，学生易接受，C++不应该不讲面向过程的部分，但必须尽早引入面向对象的概念，这本教材在介绍完函数后，就进入面向对象的教学。学生刚进入编程能力培养的实质阶段，学习的就是面向对象的程序设计。这样，大学生才可能顺利接受面向对象的思想。

第三个特点是内容新。教学内容的选定也是教学改革的重要方面。计算机技术的发展速度可以说是最快的，许多计算机的教材刚出版，内容就已经有点陈旧了。虽然程序设计作为计算机基础课程，变化没有那么快，但用最新发展的知识教学生，也是最基本的要求。1998年通过了C++国际标准ISO14882，它将相对稳定5年以上，其中一个较大的改动就是

把模板引入标准库，使用模板类来代替传统的 C++ 中定义的类，实现通用的与数据类型无关的算法。STL(标准模板库)用模板技术实现了常用数据结构及其算法的通用性。为了让大学生能掌握面向对象 C++ 编程中最新最实用的这一部分，相应的基础知识的教学是不可少的，与 STL 对应，内容包括：顺序表、链表、栈、队、二叉树和二叉排序树，常用的排序与查找方法。按照"与时俱进"的思想，这些与数据结构有关的算法的描述也全部采用模板进行。这样的处理方法也与后续课程(例如，用面向对象方法和 C++ 描述的数据结构课程)衔接得更好。

继承与多态也是面向对象程序设计实现通用性的另一有力手段。在 C++ 程序设计教学中往往成为纯语法的介绍，在我们的教学改革中，在语法之外还介绍用继承与多态实现算法的通用性，并与模板做对比。

第四个特点是基础教学与实践教学相结合。Windows 平台提供了一个很好的程序框架，但是初学者如果注意力被吸引到程序框架，那么很难学好 C++ 程序设计的基础知识。所以，在基础教学中还是采用 Windows 平台下 VC++ 的控制台方式。同时在基础教学中，逐步介绍 VC++ 的 MFC 编程的理论知识。而在实践部分比较全面地学习 Windows 编程。这安排在课程设计中，由学生在教师指导下通过自学独立完成。我们教学改革的具体目标之一是要在有限的学时内让学生掌握 Windows 环境下面向对象的软件开发技术。

第五个特点是普遍使用图解法。许多复杂事物的相互关系和发展过程用图来说明可以一目了然。所谓"一图抵千言"，无论是描述算法、分析程序、理解内存的分配与释放和软件编制的全过程，图解法都是必不可少的工具。教学实践证明：学生掌握了图解法后编程能力大大提高。

第六个特点是强调培养学生从客观事物中抽象出类来的能力。学以致用，是学习的最直接的基本目的，不是采用了面向对象程序设计语言或工具平台就能实现面向对象的程序设计。掌握了怎样从客观事物中抽象出类来的方法，能够从客观事物中抽象出合理的、可供实用的类来，才有可能实现面向对象的程序设计。为了达到这一目的，在本书中，我们刚进入面向对象的程序设计教学就开始进行从客观事物中抽象出类来的方法的讲授。

本教材包括配套的 C++ 程序设计电子教案和《C++ 程序设计实践教程》。实践内容包括 MFC 编程、课程设计等。

因本教材跟踪 C++ 国际标准 ISO14882，有关数据结构的内容较多，如果后续课程中包括用面向对象方法和 C++ 描述的数据结构，两门课相互衔接较好，本教材中有关内容学习后，学生在学习数据结构课程时会倍感轻松，而且有学时可以多学一些数据结构的新知识点。

本教材推荐授课学时为 64 学时，上机实验 32 学时，课外上机实验 32 学时。建议授课学时安排：第一章 2 学时，第二章 4 学时，第三章 6 学时，第四章 6 学时，第五章 7 学时，第六章 9 学时，第七章 8 学时，第八章 8 学时，第九章 6 学时，第十章 4 学时，第十一章 4 学时。书中的内容比较丰富，可以满足多数学生的需求。

考虑到不同学校和不同系的生源不同，各专业相关课程配置不同，大纲安排学时不同，

授课内容可有筛选。建议电子、电气(含计算机)类专业和对计算机要求较高的其他专业基本按教材内容进行教学。如因学时不同、生源不同需删减部分教学内容,建议删减的内容依次为:二叉树、标准模板类库、栈与队列、双链表、部分排序方法、部分流类库内容以及有关Windows平台下的MFC编程基本概念的内容。万不得已,异常处理亦可不讲(但因它是面向对象程序设计的重要组成部分,而且考虑到它对开发软件的重要性,并且不会再在其他课程中讲解,所以只要有可能,就不要删去)。这样不会破坏面向对象C++程序设计教学的系统性。未讲的内容,可以作为学有余力的同学的自学内容,做到因才施教。

采用这套教材的困难之处不在学生,而在教师。学生是一张白纸,按新的教学体系学习学生感觉很自然;而教师是从面向过程到面向对象走过来的,改用新体系就有困难,有很多地方不习惯,甚至还有一些东西是未接触过的;我们在编写教材的过程中就深有体会,仅为了理顺新体系就大改了书稿多次。所以我们提供配套的C++程序设计电子教案及其他电子文档,并公开放在高等教育出版社网站和东南大学网站上,供任课教师自由下载使用,并欢迎教师与读者将自己的或修改的教案及看法回贴到网上,供大家交流、讨论和使用,群策群力,共同把新的C++程序设计课程建设好。

本项改革只是一个初步的成果,在今后的教学实践中还将会与时俱进,不断发展。我们并不排斥面向对象C++程序设计其他的教学方法。"仁者见仁,智者见智",百花齐放才是春。目的是把C++程序设计的教学搞好。世界因其多样性而生气勃勃,教学也因其多样性而富于魅力。

本项教学改革由朱敏教授领导。本书第一到第四章由况迎辉编写,第五到第十一章由吴乃陵编写,第五到第十一章面中向对象程序设计实践指导内容及实践教材由李海文编写。全书由吴乃陵统稿。讨论本书编写大纲的有吴乃陵、李海文、何洁月、王磊和朱敏。张文兰、周其伟也为本书做了不少工作,在此表示感谢。

本教材为普通高等教育"十五"国家级规划教材。

在本书出版之际,我们感谢南开大学的刘璟教授,他在百忙之中认真审阅了本书并提出宝贵意见。我们按他的意见做了相应修改。

因我们水平与能力有限,时间紧迫,再加上本教材改革力度较大,难免有许多错误和不足,欢迎使用和阅读本教材的教师和同学批评指正。作者的E-mail地址分别是:

wunailing@mail.edu.cn;

kuangyh@seu.edu.cn;

lihw628@seu.edu.cn;

编者

2003年5月于东南大学

目　录

第一章　软件设计概论

软件设计是一个将人类思维转化为计算机思维的过程，通过这个过程计算机获得一定程度的独立加工甚至思维能力，从而将人类思维推向更高层次。关于软件设计概念和方法的研究是随着计算机应用的深入和所处理问题的复杂化而不断加深的。本章将就软件设计的相关概念和程序设计方法的演化发展作一简要介绍。

1.1　软件与软件危机

一个计算机系统由硬件系统和软件系统组成。关于软件，过去人们狭义地认为就是程序，软件设计就是程序设计，软件是程序员个人劳动的成果。然而在经历了“软件危机”之后，人们对软件的概念有了新的认识。所谓“软件危机”，是于20世纪60年代末提出的。在大型程序设计中，人们发现投入大量人力和时间设计出来的软件，其成本、效率、质量等方面却呈现失控状态，尤其是软件的维护异常困难，发现和修改错误以及功能扩充往往需要大量的重复性投入。产生软件危机的原因主要有3个：

(1) 软件开发者与用户之间对于问题的理解不一致。一方面开发者不熟悉用户问题的领域，或没有理解用户的需求，导致设计的软件产品与用户要求不一致；另一方面，用户也难以提出准确、完整、无二义性的软件需求描述。

(2) 软件产品不同于其他物理产品，软件开发过程更多地体现为设计人员个人的思考过程。在提出软件工程概念之前，这种思考过程一般不完整表现在书面上，因此无法对思考过程进行科学规范及质量管理，软件开发进度也无法控制。

(3) 人的智力在面对越来越复杂的问题时，处理问题的效率会越来越低。因此，在没有找到控制问题复杂性的有效方法时，开发软件所需的时间和费用将随问题复杂性的增长而急剧增加。

1.2 软件工程

1. 软件工程的提出和软件的定义

随着计算机应用领域的扩大以及问题复杂程度的增加，对软件的规模、可靠性和可维护性提出了越来越高的要求，软件危机的出现迫使人们重新认识软件的概念和软件开发过程。借鉴机械、建筑等行业从手工方式发展成为完整的工程科学的过程，软件产品也应和其他产品一样，由市场分析人员、管理人员、设计人员、测试人员甚至用户共同协作完成，即大型软件的开发也应向"工程化"方向发展。1968 年在北大西洋公约组织(NATO)的学术会议上第一次提出"软件工程"的概念，此后又逐步提出了"软件生存期"的概念，经过对软件工程思想系统的归纳和整理，1983 年 IEEE(电气和电子工程师协会)给出了软件的定义：软件是程序、方法、规则、相关文档以及在计算机上运行所必需的数据的集合。而软件工程是开发、运行、维护软件的系统方法。

2. 软件生存期

对一个较大的工程项目，把项目开发过程分成若干阶段，分别制定各阶段的实施时间、任务要求和质量要求，对整个工程的进展与质量进行分段管理，显然比整体开发更容易获得高质量的开发结果。例如开发一台机械设备，从开始研制到废弃不用为止，一般要经过分析用户要求、设计、制造、测试、运行(同时不断维护)等几个阶段。在各阶段中，分析和设计人员需要产生一整套图纸和资料，如可行性报告、零件图、装配图、使用手册、维护手册等。这台设备的生存期就是从开始研制到废弃不用为止之间的这段时间。

类似地，对于软件产品，从开始研制到废弃不用为止的整个期间称为软件的生存期。软件生存期也可划分为以下 5 个阶段：需求分析、设计、编程、测试和运行维护。每个阶段都有明确的任务和要求，必须产生一定规格文档资料；下一阶段在上期交付的文档的基础上开展工作。生存期的前 4 个阶段总称为开发期，最后一个阶段称为运行期。

3. 软件的质量标准

衡量软件质量的指标有很多，通常从正确性、健壮性、可维护性、可用性、可重用性和效率等方面进行综合评价。

(1) 正确性。软件的正确性是指软件系统在正常条件下能够正确工作，完成规定功能。这是软件的首要指标。

(2) 健壮性。软件的健壮性是指在意外情况下，如输入数据不合理或某部分硬件出现故障，软件系统仍能适当地工作，并对意外情况进行适当处理，而不至于导致错误结果甚至

系统的瘫痪或死机。例如，要求设计程序，根据输入的三边 a、b、c 的长度来判别三角形类型。现有如下设计思想：若 a、b、c 中只有两个量相等，则为等腰三角形；若三个量均相等，则为等边三角形；否则，为一般三角形。按照这样的思想设计程序，当输入分别为(3,2,3)、(2,2,2)和(3,4,6)时，程序输出分别为：等腰三角形、等边三角形和一般三角形，结果是正确的；但如果输入为(-2,-2,-2)时，程序输出为：等边三角形。这个结果显然是错误的。这是由于程序对不合理数据不能进行适当处理，因此说这个程序的健壮性不好。正确性与健壮性合称可靠性。

(3) 可维护性。软件的维护包括发现并改正软件的错误，以及由于软件运行环境发生变化或软件功能扩充而对软件进行的改动。软件维护的含义十分广泛，一个软件投入运行后经常会遇到需要维护的情况。而软件的修改必须基于对原来软件设计情况全面、正确、细致的了解，同时修改后的软件系统还必须经过测试、验证后才能重新投入使用。因此在软件的生存期中，无论从时间上还是从费用上，软件维护都占有很高的比例。

软件的可维护性指的是软件容易维护的程度。一般而言，软件的可读性好，容易理解，维护起来也就比较容易。因此可读性是可维护性的基础。

(4) 可用性。软件的可用性是指软件是否容易被用户接受的程度。包括是否容易学习、容易操作、容易准备数据、容易理解和输出结果等。

(5) 可重用性。软件可重用性是指软件中的模块或由一组模块组成的部件在多种应用场合下是否可重新使用的程度。可重用性好对提高软件产品的质量和开发效率有十分重大的意义。

(6) 效率。软件的效率是指软件系统能否有效地使用计算机资源，包括执行时间和占用内存情况。一般而言，执行时间和占用内存量之间往往是矛盾的，需要根据具体情况和要求进行取舍。另外，追求效率和追求可维护性通常也是相互矛盾的。为了节省运行时间和存储空间，往往需要使用复杂的技术，使程序变得复杂难以理解。而追求可靠性通常也要以一定的时间作为代价。在目前硬件价格下降而人工费用上升的情况下，可取的策略是，当效率、可维护性和可靠性发生冲突时，一般是牺牲一些效率来保证较好的可靠性和可维护性。

1.3 程序设计方法

程序设计方法经历了由传统的面向过程的设计到目前被日益广泛接受的面向对象的设计。

1.3.1 传统的结构化程序设计

传统的程序设计方法可以归结为“程序 = 算法 + 数据结构”，将程序定义为处理数据的

一系列过程。这种设计方法的着眼点是面向过程的，特点是将数据与程序分开存储，即数据与数据处理分离。20 世纪 60 年代，随着软件工程概念的提出，结构化程序设计方法逐渐产生并发展起来。

1976 年 N.Wirth 出版名为《Algorithms + Data Structure = Programs》的著作，明确提出算法和数据结构是程序的两个要素，即程序设计主要包括两方面的内容：行为特性的设计和结构特性的设计。行为特性的设计是指完整地描述问题求解的全过程，并精确地定义每个解题步骤，这一过程即是算法的设计；而结构特性的设计是指在问题求解的过程中，计算机所处理的数据、及数据之间联系的表示方法。

结构化程序设计（Structured Programming）的核心是算法设计，基本思想是采用自顶向下、逐步细化的设计方法和单入单出的控制结构。自顶向下、逐步细化指的是将一个复杂任务按照功能进行拆分，形成由若干模块组成的树状层次结构，逐层细化到便于理解和描述的程度。各模块尽可能相对独立。而单入单出的控制结构，指的是每个模块内部均用顺序、选择、循环三种基本结构来描述。举一个较为简单的例子：要求读入一组整数，统计其中正整数和负整数的个数。该任务的顶层模块可设计为：

模块 1：读入数据；

模块 2：统计正数、负数的个数；

模块 3：输出结果；

其中模块 2 可继续细化为：

正整数个数为 0；

负整数个数为 0；

取第一个数；

重复执行以下 3 个步骤直到统计完所有数据：

2.1：如果该数大于 0，正整数个数加 1；

2.2：如果该数小于 0，负整数个数加 1；

2.3：取下一个数；

结构化程序设计将任务划分为模块，对各个模块进行独立的设计和测试，这为处理复杂问题提供了有利手段，所以一度成为程序设计的主流方法。然而到 20 世纪 80 年代末，这种设计方法开始逐渐暴露出缺陷。主要表现在：

（1）难以适应大型软件的设计。结构化程序设计方法在一定程度上为解决复杂问题和大型软件设计提供了便利，但由于数据与数据处理相对独立，在大型多文件软件系统中，随着数据量的增大，程序变得越来越难以理解，多个文件之间的数据沟通也变得困难，还容易产生意想不到的结果，即所谓副作用。

（2）程序可重用性差。基于可重用性的思想是建立一些具有已知特性的部件，从而使

软件设计能模仿硬件组合的方法，在需要时将已知部件插入到程序中，而不是任何问题都要重新设计。但结构化程序设计方法缺乏具备这种能力的工具，既使对于老问题，处理方法的改变或数据类型的改变都将导致重新设计，这种额外开销与可重用性相对，称为重复投入。

1.3.2 面向对象的程序设计

面向过程程序设计缺点的根源在于数据与数据处理分离，而面向对象程序设计(Object Oriented Programming，OOP)方法正是克服这个缺点，同时吸纳结构化设计思想的合理部分而发展起来的，这两种设计思想并非对立关系。面向对象设计思想模拟自然界认识和处理事物的方法，将数据和对数据的操作方法放在一起，形成一个相对独立的整体——对象(Object)，对同类型对象抽象出共性，形成类(Class)。任何一个类中的数据都只能用本类自有的方法进行处理，并通过简单的接口与外部联系。对象之间通过消息(Message)进行通信。以下将就面向对象程序设计中的基本概念、面向对象软件开发方法和面向对象程序设计的特点作一简单介绍。

1. 基本概念

(1) 对象

在自然界中，对象是一个常见概念，人们通常将所面对的事物看做具有某些属性和行为(或操作)的对象。例如手表是一个对象，它具有的属性包括表针、旋钮及复杂的机械结构等，行为包括调节旋钮这样的操作，其中调节旋钮提供了对外的接口。在手表之外，只能通过这个接口对对象进行操作，而不能直接调整其内部的机械零件。在“面向对象”程序设计中，同样使用对象的概念描述问题和事物，但更加抽象，含义的范围也更加广泛，可包括有形实体、图形、甚至抽象概念。如程序设计中的窗口、单击鼠标这样的事件等，都可以抽象成为一个对象。对象是组成程序系统的基本单位。

(2) 类

人们在认识事物时，通常是分类进行的。通过抽象的方法，从一些对象中概括出此类对象共有的静态特征(属性)和动态特征(行为)，形成类。类是一个抽象的概念，用来描述这类对象所共有的、本质的属性和行为。任何一个对象都是这个类的一个具体实现，称为实例(Instance)。同类对象具有相同的属性和行为。类和对象之间的关系正如手表和一块具体的手表之间的关系。手表只是个概念，这个概念描述了所有手表共有的属性(包括表针、旋钮、内部结构)和行为(调节旋钮)，而一块具体的手表则是一个实在的对象。两块手表可能外形不同，但都具有手表所共有的属性和行为。

面向对象程序设计也使用分类抽象的方法，通过对一类对象的抽象形成“类”。在程序设计中，“类”表现为一种用户定义的数据类型，用这种数据类型描述该类所具有的属性和行

为，其中属性用数据进行描述，而行为用一系列函数描述，也称操作。例如定义一个矩形类，它的数据包括矩形的顶点坐标，而方法可包含以下函数：移动矩形位置、扩大、缩小等。用这个矩形类可以定义两个矩形对象，这两个矩形对象就是同类对象，它们都用顶点坐标来描述，都可以进行移动、扩大和缩小等操作。

(3) 消息(Message)

自然界是由各种各样的对象组成的，这些对象之间通过信息传递产生相互作用，构成富有生机的世界。人们把对象之间产生相互作用所传递的信息称做消息。比如汽车和人是两个对象，人启动汽车，就是向汽车发送消息，转动方向盘，也是发送消息，其中转动的角度是消息中的参数。汽车接收到消息后，按照消息及其参数执行相应的操作。

在面向对象设计的程序中，对象之间的相互作用也是通过消息机制实现的。

2. 面向对象的软件开发方法

在面向对象程序设计发展的早期，软件业界主要集中于研究面向对象的编程(OOP)，但对于大型软件的开发过程，编程只是其中一个很小的部分。面向对象方法的根本合理性在于它符合客观世界的组成方式和大脑的思维方式，因此面向对象的思想方法应贯穿软件开发的全过程，这就是面向对象的软件工程。

面向对象的软件工程同样遵循分层抽象、逐步细化的原则，软件开发过程包括面向对象分析(Object Oriented Analysis，OOA)、面向对象设计(Object Oriented Design，OOD)、面向对象编程(OOP)、面向对象测试(Object Oriented Test，OOT)、面向对象维护(Object Oriented Soft Maintenance，OOSM)5 个阶段。

分析阶段的主要任务是按照面向对象的概念和方法，从问题中识别出有意义的对象，以及对象的属性、行为和对象间的通信，进而抽象出类结构，最终将它们描述出来，形成一个需求模型。由于系统的复杂性，这个模型一般只能反映用户对系统主体部分的需求。

设计阶段从需求模型出发，分别进行类的设计和应用程序的设计。类的设计需要应用分层抽象的方法，而应用程序的设计是根据已设计好的类来构造满足要求的应用程序。

20 世纪 80、90 年代，面向对象的系统分析与设计成为研究热点，并推出了专门用于面向对象系统开发的工具——统一建模语言(Unified Modeling Language，UML)，现在 UML 已发展成为信息技术的国际标准。

编程阶段实现由设计表示到面向对象程序设计语言描述的转换。

测试的任务在于发现并改正程序中的错误。在面向对象程序设计中，类是程序的基本单元，因此也是测试的基本单元。

经过测试后的程序进入运行维护期，即投入使用。

3. "面向对象"程序设计的特点

面向对象程序设计中，对象是程序的基本单元。从类和对象的概念及面向对象设计方

法所提供的支持看，这种设计方法具有以下几个特点及相应的优点。

(1) 封装性(Ecapsulation)

对象是一个封装体，在其中封装了该对象所具有的属性和操作。对象作为独立的基本单元，实现了将数据和数据处理相结合的思想。此外，封装特性还体现在可以限制对象中数据和操作的访问权限，从而将属性"隐藏"在对象内部，对外只呈现一定的外部特性和功能。手表是一个典型的例子，大量的零件和动作被封装在外壳中，并被隐藏起来，提供给人们的只是读表盘和旋旋钮。同样，可以将一个对象中的数据隐藏起来，而方法定义为开放，那么在这个类之外不能直接引用其中的数据，只能通过方法达到间接使用数据的目的。就好比不能直接去拨驱动表针的内部结构，只能通过旋旋钮实现一样。

封装性增加了对象的独立性，C++通过建立数据类型——类，来支持封装和数据隐藏。一个定义完好的类一旦建立，就可看成完全的封装体，作为一个整体单元使用。用户不需要知道这个类是如何工作的，而只需要知道如何使用就行。另一方面，封装增加了数据的可靠性，保护类中的数据不被类以外的程序随意使用。这两个优点十分有利于程序的调试和维护。

(2) 继承(Inheritance)和派生性

以汽车为例，如果已经定义了汽车类，现在需要定义小汽车。通常人们不会重复描述属于汽车的那些共有特征，而是在继承汽车类特性的基础上，描述出属于小汽车的新特征。可以称小汽车继承了汽车，也可以称是由汽车派生出来的。面向对象程序设计提供了类似的机制。当定义了一个类后，若需定义一个新类，这个新类与原来类相比，只是增加或修改了部分属性和操作，这样，在定义新类时只需说明新类继承原来的类，然后描述出新类所特有的属性和操作即可。称原来类为基类，由它派生出来的类称为子类或派生类。由基类派生出子类，子类还可继续派生它的子类，如此下去，可以形成树状派生关系，称为派生树或继承树，如图 1.1 所示(注意箭头指向基类)。

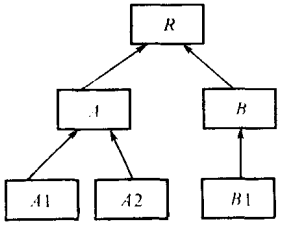

图 1.1 类的继承关系

继承性可以简化人们对问题的认识和描述，同时还可以在开发新程序和修改原程序时最大限度利用已有的程序，提高了程序的可重用性，从而提高了程序修改、扩充和设计的效率。

(3) 多态性(Polymorphism)

多态性是指同样一个消息，被不同对象接收时，产生不同的结果。系统提供的这种机制主要用在具有继承关系的类体系中。一个类体系中的不同对象可以用不同方式响应同一消息，并产生不同结果，即实现"同一接口，多种方法"。例如，定义了中学生类，再由中学生派生出大学生类。对于"计算平均成绩"这样一个消息，对中学生对象，计算的是语文、数学、英语等课程；而对大学生对象，则计算高等数学、英语、线性代数等课程。

继承和多态性的组合，可以很容易地生成一系列虽然类似但独一无二的对象。继承性