高等学校计算机科学与技术教材

C++语言程序设计

李 强 编著
骆耀祖 主审

清华大学出版社
http://www.tup.tsinghua.edu.cn
北方交通大学出版社
http://press.njtu.edu.cn

高等学校计算机科学与技术教材

C++语言程序设计

李　强　编著

骆耀祖　主审

清 华 大 学 出 版 社

北方交通大学出版社

·北京·

内 容 简 介

本书全面系统地介绍了C++程序设计语言的主要概念、语法及程序设计技巧等方面的内容。在内容的安排上循序渐进，突出重点，深入浅出。从C++语言的基本数据类型与基本控制结构入手，逐渐过渡到函数、类与对象、继承、多态、输入输出流等复杂的C++机制，最后介绍面向对象的应用程序设计技术。全书通俗易懂，行文流畅。在内容上始终贯穿培养学生进行面向对象的程序设计的思想。本书提供了丰富的典型例题，并且每一章都有一定数量的练习题，便于读者掌握基本知识及检验学习效果。

本书可作为高等院校计算机专业和高等院校理工科专业C++程序设计课程的教材，也可以作为C++语言的培训教材和工程技术人员的自学参考书。

图书在版编目（CIP）数据

C++语言程序设计/李强编著. —北京：北方交通大学出版社，2003.10
（高等学校计算机科学与技术教材）
ISBN 7-81082-194-6

Ⅰ.C… Ⅱ.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076549号

责任编辑：谭文芳
印 刷 者：北京市黄坎印刷厂
出版发行：北方交通大学出版社　　邮编：100044　　电话：010-51686045，62237564
　　清 华 大 学 出 版 社　　邮编：100084
经　　销：各地新华书店
开　　本：787×1092　1/16　　印张：19　　字数：483千字
版　　次：2003年10月第1版　　2003年10月第1次印刷
印　　数：5 000册　　定价：25.00元

前　言

本书全面系统地讲述了C++语言的基础知识、基本语法和程序设计技术，详尽地介绍了面向对象程序设计技术的基本特征：类和对象、封装、继承性和多态性等方面的内容。

C++语言是一个既可以进行结构化程序设计，也可以进行面向对象程序设计的语言。它具有表达自由、使用灵活的特点，目前应用得很广泛。但该语言有些概念很抽象，比较难学，在程序设计过程中也比较容易出错。

作者集多年C++语言的教学经验和体会，精心编写了本书。本书偏重于应用，以面向对象的程序设计思想作为主线贯穿全书。全书在内容上合理编排，循序渐进、分散难点、突出重点，在文字上通俗易懂，每章还配有一定数量的练习题。

读者通过本书的学习，能够正确地理解面向对象的基本概念和思想，掌握基本的C++面向对象的程序设计方法。

全书主要由以下三部分组成。

一、C++语言基础部分。由第1，2，3，4，5，6，7章构成。主要介绍C++作为一门程序设计语言的基本要素和基本内容，如基本数据类型与基本输入输出、流程控制语句、函数、指针、结构和联合等。这一部分是程序设计基础部分。

二、面向对象的C++语言程序设计部分。由第8，9，10，11，12，13章构成。这一部分是面向对象的C++语言的核心部分，主要突出C++语言面向对象的特征，如封装、继承、多态的概念及代码重用的思想。主要内容有：类和对象、类的继承与派生、多态性和虚函数、输入输出流类、模板等。这部分是整个教材的核心部分。

三、VC++应用程序设计部分。由第14，15章构成，主要介绍C++语言的程序设计应用，使学生建立起Windows环境下用面向对象的思想进行程序设计的概念，为今后进一步地学习相关面向对象的程序设计语言建立基础。

本书所有例题程序都在Visual C++ 6.0版本的编译系统下运行通过，读者也可以使用其他版本的编译器（如Borland C++编译器）编译。但个别程序可能需要稍加修改，以适应具体版本的特殊约定。

作为C++语言程序设计课程的教材，建议本教材课堂教学学时为54～72学时，上机学时为30个学时左右。依据因材施教的原则，如果学生已经有了一定的计算机语言方面的知识和一定的程序设计基础，任课老师在教学过程中可以适当压缩第一部分（C++语言基础部分）的学时，而增加第二部分（面向对象的C++语言程序设计部分）的学时。本教材第三部分（VC++应用程序设计）应当以学生自学为主、教师引导为辅的教学原则组织教学，培养学生在掌握了面向对象程序设计技术的基础上，理解程序设计应用环境（如Windows的消息驱动机制），迅速掌握面向对象的开发语言工具的能力。

本书由李强编著，骆耀祖高工主审。

目　录

第 1 部分　C++语言基础

第 2 部分 面向对象的 C++语言程序设计

第 3 部分 Visual C++应用程序设计

第 1 部分　C++语言基础

第 1 章　概　　述

1.1　计算机程序设计方法及程序设计语言的发展

计算机是工业化、批量生产的产物，为保证通用化，其硬件功能相对固定，用户需要设计程序改变计算机系统的状态以适合人们的需求。计算机实际上是带有一套指令系统的机器，开发者直接面对的是这套指令系统，利用这套指令系统，开发者可以设计程序，驱动计算机工作，使功能相对固定的机器灵活地满足用户的需要。因此，程序设计是计算机应用的核心。

计算机是信息处理的工具，所谓计算机程序就是对客观世界的问题抽象后，建立合理的模型，最后由计算机程序设计人员利用合适的程序设计语言设计而成的、按人们的要求驱动计算机进行信息处理的指令序列，该指令序列包括数据和操作两部分内容。如图 1-1 所示。

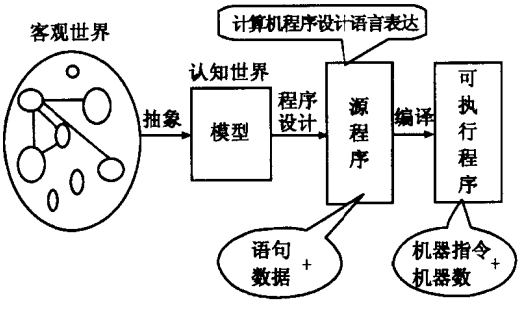

图 1-1　计算机程序设计语言在程序设计中的作用

程序设计语言在程序设计中具有很重要的作用，它既要符合计算机驱动的要求，又要合理地表达、描述客观世界。计算机从本质上来说是具有一定智能构成的机器，它要求对它的驱动准确、并符合其工作规律（比如：顺序执行、顺序存储）；而全面描述客观事物，又要求程序设计语言符合人类思维习惯。因此，寻找合理的计算机程序设计方法和开发高效率的程序设计语言是计算机行业的永恒的课题。从第一台计算机诞生以来，程序设计方法和程序设计语言就在不断发展。

1.1.1　早期发展

在计算机发展的初期，由于计算机硬件条件的限制，运算速度低，存储空间有限，软件的规模也小，这时程序员追求的目标是计算机系统资源的高利用率。这一阶段没有固定的程序设计方法，程序设计是程序员本人很个性化的艺术，强调的是技巧，不太讲究所编写程序的结构，

所开发的程序其他人难以理解，程序的修改也比较困难，计算机的应用领域也主要在科学计算方面。此时，系统软件还不完善，所以软件的设计中还没有系统软件和应用软件的界限，用户所开发的程序往往要兼顾系统管理的一些功能，软件设计有很明显的面向系统硬件平台的特性，无可移植性。软件设计工具主要以机器语言和汇编语言为主。

所谓机器语言，是指由计算机硬件系统可以直接识别的二进制编码组成的语言。机器语言面向硬件系统平台，只适用于开发功能简单、规模较小的软件。用机器语言开发软件，存在通用性差、修改、测试都比较困难的问题。

汇编语言是用比较好理解的英文助记符号代表机器语言的一种软件开发工具，如 mov，jmp，add 等，程序员用汇编语言设计源程序，然后可以利用计算机汇编程序完成对所设计程序的翻译，即将助记符号翻译成二进制机器码，如图 1-2 所示。

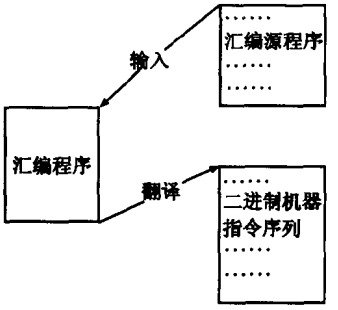

图 1-2　汇编语言源程序的编译

汇编语言的出现使人们摆脱了记忆机器语言的负担，借助于汇编语言，可以使用符合人们记忆表达习惯的语言设计计算机程序。汇编语言为人们设计程序时摆脱机器语言的束缚，使用适合于人类思维的工具进行计算机程序设计开辟了一条途径。

但汇编语言程序设计表达方式与人的正常思维方式仍然有很大的差距，程序员仍然摆脱不了顺序存储、顺序执行的机器束缚，在程序设计时依然要处理硬件工作的所有细节问题，所设计的程序的可读性、可修改性很差。程序某一部分的修改，可以导致整个程序重新编写。这一时期为 20 世纪 40 年代后期至 50 年代中期，软件规模不大，软件设计工作是少数计算机专家们特有的才能。

这时候，人们对客观事物的思想表达与计算机软件之间没有直接的联系，思想的表达必须经过具有专门计算机研究工作经验、具备汇编程序设计技巧的程序员的加工，纳入具体计算机的顺序执行、顺序存储的硬件工作执行范围内，才能够成为软件在计算机上运行。程序与描述客观事物的思想是分离的。

1.1.2　结构化程序设计时期

20 世纪 50 至 60 年代，随着计算机硬件技术的发展，运算速度和存储空间都得到了增长，系统软件也逐渐从应用软件中分离开。所谓系统软件是指管理计算机系统本身的专门软件（如操作系统等）它支持用户对计算机的运行管理，对用户屏蔽计算机运行的底层细节，用户通过通用的人机接口就可以调度计算机进行正常工作。用户也可以在系统软件的基础上开发应用软件，这就为程序设计语言脱离具体的硬件环境，进一步摆脱硬件工作机制的束缚提供了条件。

在这一时期，各种计算机高级语言相继出现，如 FORTRAN、COBOL 和 BASIC 语言等。

高级语言的出现是计算机程序设计语言技术发展的一大进步，它借助于系统软件屏蔽了计算机硬件的细节，提高了语言的抽象层次，程序可以由具有一定逻辑含义、便于理解的语句构成，这使得所设计的程序和程序所描述的具体事物之间可以从逻辑上直接联系起来。

随着软件的规模和复杂程度不断提高，人们也在不断探索新的程序设计方法。20 世纪 70 年代初期出现了结构化程序设计语言：C 语言和 Pascal 语言。

结构化程序设计的方法是自顶向下、逐步求精的程序设计技术，自顶向下是一种分解问题的技术，逐步求精是指结构化程序的连续分解，最终成为三种基本控制结构（顺序、选择、循环）的组合。其结果是将一个程序最终由若干个过程组成，每个过程完成一个确定的功能。

Pascal 语言是根据结构化程序设计方法开发出来的计算机高级程序设计语言，其特点是它能提炼出程序设计的共同特征并能将这些特征编译成高效的代码。Pascal 语言是结构化程序设计的有力工具，至今仍然是描述数据结构及其算法的强有力的语言，在计算机教育界广受关注。

C 语言是 1972 年美国贝尔实验室的丹尼思·里奇（Dennis Ritchie）开发的，是在肯·汤普生（K.Thompson）的 B 语言的基础上重新设计的语言。其最初的目标是为了描述和实现 UNIX 操作系统提供一种工作语言。UNIX 操作系统的 90%的代码是由 C 语言编写的。该语言最初是在 DEC 公司的 PDP-II 小型计算机上，在 UNIX 操作系统环境上实现的，随着 UNIX 操作系统的成功和广泛的应用，C 语言成为一种普遍使用的计算机语言。C 语言具有灵活方便、目标代码效率高、可移植性好的特点，一度成为事实上的语言标准，美国国家标准局于 1987 年制定了 C 语言标准，称为 ANSI C。

1.1.3 面向对象的程序设计

时至今日，结构化程序设计的方法仍然在软件业有极大的市场，几乎每一种程序设计语言都支持结构化程序设计的机制，包括 C++语言在内。

面向对象程序设计是建立在结构化程序设计的基础上的。与以往的程序设计方法不同的是，其程序设计是围绕着被操作数据而进行的，而不是围绕着程序本身。因此，面向对象的程序设计的出发点就是为了方便地描述客观世界的事物（即对象）及事物之间的关系。

打一个通俗的比喻，制作课桌，使用面向结构的设计方法，先要能够制作桌子腿、抽屉、台面，最后才能够制作课桌，其制作过程与人们正常思维的过程是相反的；而使用面向对象的设计方法，先进行课桌整体功能的设计、然后再考虑组成课桌的其他部分的制作方法，其制作过程与人类的思维方式相同。

如果说，汇编语言的出现，使人们的程序设计语言与实际的机器语言脱离开来，人们可以使用符合人类记忆习惯的符号进行程序设计；而面向过程的高级程序设计语言又使人们摆脱了在程序设计过程中具体硬件平台的限制，摆脱了程序设计中很多涉及计算机硬件的麻烦、枯燥的细节处理，只关心程序设计过程中的数据合理的描述和算法本身的实现，将软件设计从进行高深研究的实验室的专门人员手中解放出来，成为一个较为大众化的专业技能。因此，面向对象的程序设计语言的出现又给人们提供了符合人类思维习惯、符合人类社会化协作多人工作方式的开发计算机软件的工具，将软件设计变成一个社会化的职业工作，将软件生产变成一种工

业产业。

另外，现代计算机已经从最初的科学计算、字符处理向多媒体应用方面发展，软件的规模和复杂性在不断增加。比如，以个人微机为例，十年前在字符界面 DOS 操作系统上进行的软件开发需要一个人在几个月内完成的工作量，在 Windows 图形界面上需要多人去协作开发，开发周期可以持续数年。软件规模和复杂性的增长也在迫使人们寻找新的开发工具，按新的软件开发组织方式进行软件生产，形成一种社会工业产业。在这种背景下，面向对象的程序设计方法和工具应运而生，这是软件产业化发展的必然趋势，面向对象的程序设计方法还在不断发展，它会不断给人们提供更加高效、快捷的软件开发工具。

C++语言是由 AT&T 贝尔实验室的 Bjarne Stroustrup 博士在 1980 年设计的，它是在 C 语言的基础上为支持面向对象的程序设计而研制的程序设计语言，为了支持面向对象的程序设计，该语言引入了类的机制，因此面世时称为带类的 C，1983 年正式命名为 C++程序设计语言。1989 年开始其标准化工作，1994 年制定 ANSI C++标准草案。目前，已经有比较成熟的 C++语言编译器产品，如 Microsoft 公司的 Visual C++和 Borland 公司的 Borland C++。C++语言得到广泛的应用，其软件工具还在不断发展、完善。

计算机程序设计语言和软件开发方法的发展历程如图 1-3 所示。

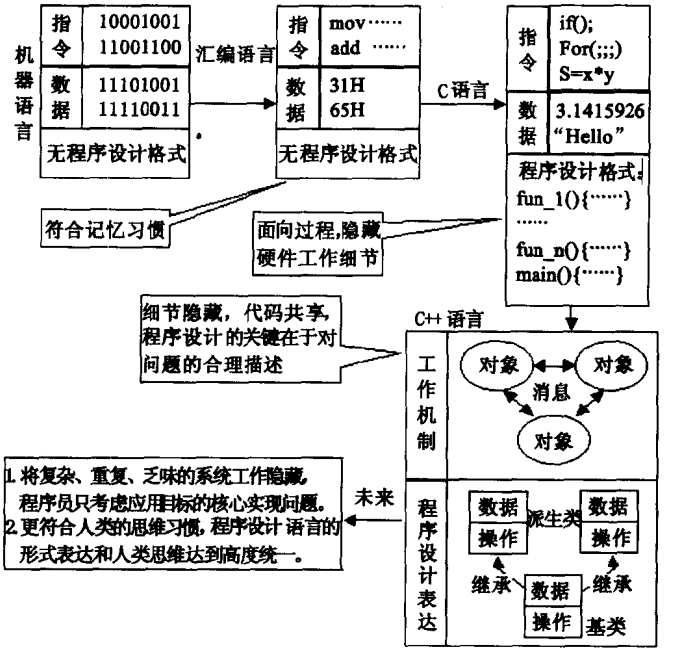

图 1-3　计算机语言和计算机软件开发方法的发展历程

1.2　面向对象程序设计的特点

从程序设计方法的发展可看到，面向对象的程序设计方法注重于对问题的分析和在分析基础上的合理设计。本节主要介绍面向对象程序设计技术中的一些主要概念。

1. 抽象（Data Abstraction）

抽象是指将具体事物一般化的过程，即对具有特定属性及行为特征的对象提取其共性的过程。在抽象过程中，需要研究目标程序要解决的问题，以及组成该问题的概念性实体，找出有利于解决问题的一些特性，从而得到对该问题的抽象结果。

在面向对象的程序设计中，往往需要进行不同级别的抽象过程。对高级抽象，可能会将一个问题抽象为一些特定对象的交互行为；而对低级抽象，这些特定的对象常常可抽象分解为其他更小对象的交互行为。假如要设计一个学校的办公管理系统，可以在高级别上将该办公系统看成是学校的一些不同的系、部门之间的信息交流，在低级别上，又可以看做不同的教研室、科室之间工作的相互配合。

抽象的目的在于代码重用，由于每一层的抽象都会带来不同的可重用代码，因此面对问题经常需要进行不同级别的抽象。

抽象是面向对象程序设计的一个基本特征，类就是对一些问题和概念进行抽象的工具，类从客观世界的一组事物中抽取其共同的属性和行为，对象则是类的实例化，具体化。

在面向对象的程序设计方法中，用对象来描述具体的事物，对象是构成应用系统的基本单位，它由类而来，包括属性和行为，属性就是描述对象特征的数据，行为则是对属性的操作。类和对象的关系如图 1-4 所示。

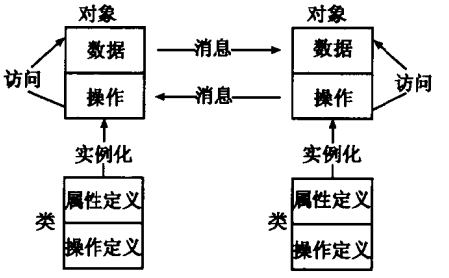

图 1-4 类的实例化及对象之间的交互方式

2. 封装（Encapsulation）

对象（Object）是面向对象程序设计的重要性质之一，从具体的程序设计工具的使用上来讲，对象就是既含有数据（叫做对象的属性——Attribute）又含有对数据操作的代码（Method）的一个逻辑实体。使用对象将数据和操作进行封装，这样，在程序中对数据的存取只能通过对该对象本身的操作来进行，程序的其他部分不能直接访问作用于此对象的数据，因此封装又称为数据隐藏。在面向对象的程序设计中，对象之间的交互只能通过消息来进行，如图 1-4 所示。

从图 1-4 可见，在面向对象程序中，类是对象的抽象，对象是类的实例化。通过定义类来尽可能地将一些关键数据、函数隐藏起来，这样，只有通过用户定义的接口才能访问类中的数据，避免了在程序开发过程中数据的不恰当使用，大大增加了程序设计过程的可靠性。这一点在需要多人协作、共同开发大型软件系统时，具有很重要的意义。

3. 继承（Inheritance）

继承是面向对象程序设计的又一个重要特性，在其他类型的程序设计中，数据类型之间没