高等学校21世纪计算机教材

C++面向对象程序设计教程

陈懿 编著

冶金工业出版社

高等学校21世纪计算机教材

C++面向对象程序设计教程

陈　懿　编著

北　京

冶金工业出版社

内 容 简 介

本书主要介绍了面向对象程序设计语言——C++的基本概念和相关知识。全书共 分12章，内容包括C++概述，C++基本数据类型与表达式，数组，预处理和语句，函数，指针、引用与动态空间管理，结构体、共用体与枚举，类与对象，继承性与派生类，多态性与虚函数，C++的I/O流库，异常处理。

本书在编写过程中注重基础知识，内容丰富，文字叙述简明易懂，各章还配有精心设计的习题。本书可作为高等院校计算机专业的教材，也可作为C++开发人员提高读物和工具书及参考资料。

图书在版编目（CIP）数据

C++面向对象程序设计教程 / 陈懿编著. —北京：
冶金工业出版社，2005.7
ISBN 7-5024-3777-0

I. C… II. 陈… III. C语言—程序设计—教材
IV. TP312

中国版本图书馆CIP数据核字（2005）第065357号

出版人 曹胜利（北京沙滩嵩祝院北巷39号，邮编100009）
责任编辑 戈兰
佛山市新粤中印刷有限公司印刷；冶金工业出版社发行；各地新华书店经销
2005年7月第1版第1次印刷
787mm×1092mm 1/16； 19印张； 436千字； 294页
25.00元
冶金工业出版社发行部 电话：（010）64044283 **传真：**（010）64027893
冶金书店 地址：北京东四西大街46号（100711） **电话：**（010）65289081
（本社图书如有印装质量问题，本社发行部负责退换）

前　言

一、关于本书

面向对象的程序设计是一种围绕真实世界的概念来组织模型的程序设计方法，这种方法把数据及与之相关的处理过程作为一个有机的整体看待，这样的整体称为对象。通过对象这种实体，更容易完整地反映现实问题本质，因而便于对问题进行自然的、符合人们思维习惯的分析，进而建立能够更直观地反映客观世界的模型。根据这样的模型开发出的软件是由一系列对象组成的，通过对象间的消息传递，完成系统的功能。面向对象的程序设计克服了传统设计方法的主要缺点，因而得到日益广泛的应用，是软件工程领域的重大突破。本书以C++这种最典型的面向对象程序设计语言为媒介，全面介绍面向对象程序设计的基本理论、方法与技巧。之所以借助C++语言，是因为C++全面支持面向对象程序设计，能较为全面地反映面向对象方法中的各种概念，同时又应用广泛。

本书较全面较系统地讲述了C++语言的基本概念和编程方法。在学习的过程中，读者应注意把握以下几点：

（1）能够正确地理解C++语言中的面向对象的方法。

（2）基本掌握C++语言中的词法、语法。

（3）能够达到使用C++语言编写简单程序的目的。

二、本书结构

本书共分12章，内容结构安排如下：

第1章：C++概述。主要介绍了计算机程序设计语言的发展、面向对象的方法、面向对象的软件开发、C++语言的起源与特点、C++的基本符号与词法、C++语言程序的结构特点、C++程序的实现。

第2章：C++基本数据类型与表达式。主要介绍了C++数据类型、常量、变量、运算符、表达式。

第3章：数组。主要介绍了一维数组、二维数组和字符数组。

第4章：预处理和语句。主要介绍了预处理功能、语句、选择语句、循环语句以及转向语句。

第5章：函数。主要介绍了函数的定义及其说明、函数的调用、函数的原型声明、函数调用中的数据传递、函数指针、函数重载、变量的作用域与存储类型，还简要介绍了内联函数、递归函数。

第6章：指针、引用与动态空间管理。主要介绍了指针、指针与数组、引用和动态存储空间管理。

第7章：结构体、共用体与枚举。主要介绍了结构体类型定义、结构体变量声明与初始化、结构体变量使用方式、结构体与函数、链表、共用体、枚举类型、类型名的重定义。

第8章：类与对象。主要介绍了类的定义、对象的定义、对象的初始化、成员函数的

特性、静态成员、友元、类的作用域、局部类和嵌套类、对象的生存期。

第 9 章：继承性与派生类。主要介绍了基类和派生类、单继承、多继承、虚基类，最后还介绍了一个应用实例。

第 10 章：多态性与虚函数。主要介绍了运算符重载、静态联编和动态联编、虚函数、纯虚函数和抽象类、虚析构函数、模板。

第 11 章：C++的 I/O 流库。主要介绍了屏幕输出、键盘输入、插入符和提取符的重载、格式化输入和输出、磁盘文件的输入和输出、字符串流、流错误的处理。

第 12 章：异常处理。主要介绍了异常处理的概念、异常处理的基本思想、异常处理的实现、异常处理的规则、异常事件的多路捕获、异常处理机制、使用异常处理的方法。

三、本书特点

本书是编者综合国内外优秀教材，结合大量国内外的最新资料，再融入丰富的 C++编程经验编写而成的。

该书通俗易懂、适于自学；由浅入深、便于理解；概念明确、语言简洁；例题丰富、内容全面；重点突出、难点详解。

四、本书适用对象

本书适用面广，既可以作为 C++初学者的学习参考书，也可以作为高等院校相关专业的教材。对于具有一定 C++语言知识的读者，本书也不失为实用性很强的参考资料。

编者在编写本书的过程中，参考了大量的文章和书籍，吸取了很多朋友宝贵的建议，在此向对本书编写过程中做出帮助的同行表示衷心的感谢。

由于编者水平有限、编写时间仓促，书中错漏之处在所难免，敬请广大读者批评。

虽然经过严格的审核、精细的编辑，本书上在质量上有了一定的保障，但我们是目标是力求尽善尽美，欢迎广大读者和专家对我们的工作提出宝贵的建议，联系方法如下 ：

电子邮件：service@cnbook.net

网址：www.cnbook.net

本书电子教案和源代码可在本网站免费下载，此外，该网站还有一些其他相关书书籍的介绍，可以方便读者选购参考。

编　者

2005 年 5 月

目　录

第 1 章　C++概述

教学目标：

（1）计算机程序设计语言的发展。

（2）面向对象的方法。

（3）面向对象的软件开发。

（4）C++语言的起源与特点。

（5）C++的基本符号与词法。

（6）C++语言程序的结构特点。

（7）C++程序的实现。

随着计算机应用技术的发展，C++语言在各个领域的应用越来越广泛。几乎各类计算机都支持 C++语言的开发环境，这给 C++的普及与应用奠定了基础。

本章重点介绍 C++语言的起源与特点、C++的基本符号与词法、C++语言程序的结构特点。

1.1　计算机程序设计语言的发展

语言是一套具有语法、词法规则的系统。语言是思维的工具，思维是通过语言来表达的。计算机程序设计语言是计算机可以识别的语言，用于描述解决问题的方法，供计算机阅读和执行。

1.1.1　机器语言与汇编语言

自从 1946 年 2 月世界上第一台数字计算机 ENIAC 诞生以来，在这短暂的 50 多年间，计算机科学得到了迅猛发展，计算机及其应用已渗透到社会的各个领域，有力地推动了整个社会的信息化发展，目前，计算机已经成为信息化社会中必不可少的工具。

计算机系统包括硬件和软件。计算机之所以有如此强大的功能，不仅因为它具有强大的硬件系统，而且依赖于软件系统。软件包括了计算机运行所需的各种程序及相关的文档资料。计算机的工作是用程序来控制的，离开程序，计算机将一事无成。程序是指令的集合。软件工程师将解决问题的方法、步骤编写为由一条条指令组成的程序，然后输入到计算机的存储设备中。计算机执行这一指令序列，便可完成预定的任务。

所谓指令，就是计算机可以识别的命令。虽然在人类社会中，各民族都有丰富的语言用来表达思想、交流感情、记录信息，但计算机却不能识别它们。计算机所能识别的指令形式，只能是简单的“0”和“1”的组合。一台计算机硬件系统能够识别的所有指令的集合，称为它的指令系统。

由计算机硬件系统可以识别的二进制指令组成的语言称为机器语言。毫无疑问，虽然计算机语言便于计算机识别，但对于人类来说却是晦涩难懂的，更难以记忆。可是在计算机发展的初期，软件工程师们只能用机器语言来编写程序。这一阶段，在人类的自然语言和计算机编程语言之间存在着巨大的鸿沟，软件开发的难度大、周期长，开发出的软件功

能却很简单，界面也不友好。

不久，出现了汇编语言，它将机器指令映射为一些可以被人读懂的助记符，如 ADD、SUB 等。此时编程语言与人类自然语言间的鸿沟略有缩小，但仍与人类的思维相差甚远。因为它的抽象层次太低，程序员需要考虑大量的机器细节。

尽管如此，从机器语言到汇编语言，仍是一大进步。这意味着人与计算机的硬件系统不必非得使用同一种语言。程序员可以使用较适合人类思维的语言，而计算机硬件系统仍只识别机器指令。那么两种语言间的沟通如何实现呢？这就需要一种翻译工具（软件）。汇编语言的翻译软件称为汇编程序，它可以将程序员写的助记符直接转换为机器指令，然后再由计算机去识别和执行。

1.1.2 高级语言

高级语言的出现是计算机编程语言的一大进步。它屏蔽了机器的细节，提高了语言的抽象层次，程序中可以采用具有一定含义的数据和容易理解的执行语句。这使得在书写程序时可以联系到程序所描述的具体事物。

20 世纪 60 年代末开始出现的结构化编程语言进一步提高了计算机语言的层次。结构化数据、结构化语句、数据抽象、过程抽象等概念使程序更便于体现客观事物的结构和逻辑含义，这使得编程语言与人类的自然语言更接近。但是二者之间仍有不少差距，主要问题是程序中的数据和操作分离，不能够有效地组成与自然界中的具体事物紧密对应的程序成分。目前应用比较广泛的几种高级语言有 FORTRAN、BASIC、PASCAL、C 等。当然本书介绍的 C++语言也是高级语言，但它与其他面向过程的高级语言有着根本的不同。

1.1.3 面向对象的语言

面向对象的编程语言与以往各种编程语言的根本不同点在于，它设计的出发点就是为了能更直接地描述客观世界中存在的事物（即对象）以及它们之间的关系。

开发一个软件是为了解决某些问题，这些问题所涉及的业务范围称为该软件的问题域。面向对象的编程语言将客观事物看作具有属性和行为（或称服务）的对象，通过抽象找出同一类对象的共同属性（静态特征）和行为（动态特征），形成类。通过类的继承与多态可以很方便地实现代码重用，大大缩短了软件开发周期，并使得软件风格统一。因此，面向对象的编程语言使程序能够比较直接地反映问题域的本来面目，软件开发人员能够利用人类认识事物所采用的习惯思维方法来进行软件开发。

面向对象的程序设计语言经历了一个很长的发展阶段。例如，LISP 家族的面向对象的语言、Simula67 语言、Smalltalk 语言以及 CLU、Ada、Modula-2 等语言，或多或少地都引入了面向对象的概念，其中 Smalltalk 是第一个真正的面向对象的程序语言。

然而，应用最广的面向对象程序语言是在 C 语言基础上扩充出来的 C++语言。由于 C++对 C 兼容，而 C 语言又早已被广大程序员所熟知，所以 C++语言也理所当然地成为应用最广泛的面向对象程序语言。

1.2 面向对象的方法

程序设计语言是编写程序的工具，因此程序设计语言的发展恰好反应了程序设计方法

的演变过程。这里首先初步介绍一下面向对象方法的基本概念和基本思想，待大家学习完本书之后，相信会对面向对象的方法有一个深入、完整的认识。

1.2.1 面向对象方法的由来

在面向对象的方法出现以前，程序员都是采用面向过程的程序设计方法。早期的计算机是用于数学计算的工具，例如用于计算炮弹的飞行轨迹。为了完成计算，就必须设计出一个计算方法或解决问题的过程。因此，软件设计的主要工作就是设计求解问题的过程。

随着计算机硬件系统的高速发展，计算机的性能越来越强，用途也更加广泛，不再仅限于数学计算。由于所处理的问题日益复杂，程序也就越来越复杂和庞大。20 世纪 60 年代产生的结构化程序设计思想，为使用面向过程的方法解决复杂问题提供了有力的手段。因而，在 20 世纪 70 年代到 80 年代，结构化程序设计方法成为了所有软件开发设计领域及每个程序员都采用的方法。结构化程序设计的思路是：自顶向下、逐步求精；其程序结构是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；每一模块内部均是由顺序、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。结构化程序设计由于采用了模块分解与功能抽象以及自顶向下、分而治之的思想，从而有效地将一个较复杂的程序系统设计任务分解成许多易于控制和处理的子任务，便于开发和维护。

虽然结构化程序设计方法具有很多的优点，但它仍是一种面向过程的程序设计方法。它把数据和处理数据的过程分离为相互独立的实体，当数据结构改变时，所有相关的处理过程都要进行相应的修改，每一种相对于老问题的新方法都要带来额外的开销，程序的可重用性差。另外，由于图形用户界面的应用，使得软件使用起来越来越方便，但开发起来却越来越困难。如果仍使用面向过程的方法，软件的开发和维护都将很困难。

那么，什么是面向对象的方法呢？首先，它将数据及对数据的操作方法放在一起，作为一个相互依存、不可分离的整体——对象。对同类型抽象出其共性，形成类。类中的大多数数据，只能用本类的方法进行处理。类通过一个简单的外部接口与外界发生关系，对象与对象之间通过消息进行通信。这样，程序模块间的关系更为简单，程序模块的独立性、数据安全性就有了良好的保障。另外，通过后续章节中介绍的继承与多态性，还可以大大提高程序的可重用性，使得软件的开发和维护都更为方便。

面向对象的方法有如此的优点，然而对于初学程序设计的人来说，是否容易理解、容易掌握呢？回答是肯定的。面向对象方法的出现，实际上是程序设计方法发展的一个返璞归真的过程。软件开发从本质上来讲，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。

面向对象方法所强调的基本原则，就是直接面对客观存在的事物进行软件开发，将人们在日常生活中习惯的思维方式和表达方式应用在软件开发中，使软件开发从过分专业化的方法、规则和技巧中回到客观世界，回到人们通常的思维方式。

1.2.2 面向对象的基本概念

下面简单介绍一下面向对象方法中的几个基本概念。虽然不能期望通过几句话就完全理解这些概念，但在本书的后续章节中，会不断帮助读者加深对这些概念的理解，以达到

熟练运用。

1. 对象

从一般意义上讲，对象是现实世界中一个实际存在的事物，它可以是有形的（比如一辆汽车），也可以是无形的（比如一项计划）。对象是构成世界的一个独立单位。它具有自己的静态特征（可以用某种数据来描述）和动态特征（对象所表现的行为或具有的功能）。

面向对象方法中的对象，是系统中用来描述客观事物的一个实体，它是用来构成系统的一个基本单位。对象由一组属性和一组行为构成。属性是用来描述对象静态特征的数据项，行为是用来描述对象动态特征的操作序列。

2. 类

把众多的事物归纳、划分成一些类，是人类在认识客观世界时经常采用的思维方法。分类所依据的原则是抽象，即忽略事物的非本质特征，只注意那些与当前目标有关的本质特征，从而找出事物的共性，把具有共同性质的事物划分为一类，得出一个抽象的概念。例如，石头、树木、汽车、房屋等都是人们在长期的生产和生活实践中抽象出来的概念。

面向对象方法中的“类”，是具有相同属性和服务的一组对象的集合。它为属于该类的全部对象提供了抽象的描述，其内部包括属性和行为两个主要部分。类与对象的关系尤如模具与铸件之间的关系，一个属于某类的对象称为该类的一个实例。

3. 封装

封装是面向对象方法的一个重要原则，就是把对象的属性和服务结合成一个独立的系统单位，并尽可能隐蔽对象的内部细节。这里有两个含义：第一个含义是把对象的全部属性和全部服务结合在一起，形成一个不可分割的独立单位。第二个含义也称作“信息隐蔽”，即尽可能隐蔽对象的内部细节，对外形成一个边界（或者说一道屏障），只保留有限的对外接口使之与外部发生联系。

4. 继承

继承是面向对象技术能够提高软件开发效率的重要原因之一，其定义是：特殊类的对象拥有其一般类的全部属性与服务，称作特殊类对一般类的继承。

继承具有重要的实际意义，它简化了人们对事物的认识和描述。比如在认识了轮船的特征之后，再考虑客轮时，因为知道客轮也是轮船，于是可以认为它理所当然地具有轮船的全部一般特征，从而只需要把精力用于发现和描述客轮独有的那些特征。

继承对于实现软件复用有着重要的意义，其实特殊类继承一般类本身就是软件复用。而且不仅于此，如果将开发好的类作为构件放到构件库中，在开发新系统时便可以直接使用或继承使用。

5. 多态性

多态性是指在一般类中定义的属性或行为，被特殊类继承之后，可以具有不同的数据类型或表现出不同的行为。这使得同一属性或行为在一般类及其各个特殊类中具有不同的语义。例如，可以定义一个一般类“几何图形”，它具有“绘图”行为，但这个行为并不具有具体含义，也就是说并不确定执行时到底画一个什么样的图（因为此时尚不知道“几何图形”到底是一个什么图形，“绘图”行为当然也就无从实现）。然后再定义一些特殊类，如“椭圆”和“多边形”，它们都继承一般类“几何图形”，因此也就自动具有了“绘图”

行为。接下来，可以在特殊类中根据具体需要重新定义“绘图”，使之分别实现画椭圆和多边形的功能。进而，还可以定义“矩形”类继承“多边形”类，在其中使用“绘图”实现绘制矩形的功能。这就是面向对象方法中的多态性。

1.3 面向对象的软件开发

在整个软件开发过程中，编写程序只是相对较小的一个部分。软件开发的真正决定性因素来自前期概念问题的提出，而非后期的实现问题。只有识别、理解和正确表达了应用问题的内在实质，才能做出好的设计，然后才是具体的编程实现。

早期的软件开发所面临的问题比较简单，从认清要解决的问题到编程实现并不是太难的事。随着计算机应用领域的扩展，计算机所处理的问题日益复杂，软件系统的规模和复杂度空前扩大，以至于软件的复杂性和其中包含的错误已达到软件人员无法控制的程度，这就是20世纪60年代初期的“软件危机”。软件危机的出现，促进了软件工程学的形成与发展。

在学习面向对象的程序设计时，首先应该对软件开发和维护的全过程有一个初步了解。因此，在这里先简要介绍一下什么是面向对象的软件工程。面向对象的软件工程在软件工程领域有着广泛的应用。它包括面向对象的分析（OOA）、面向对象的设计（OOD）、面向对象的编程（OOP）、面向对象的测试（OOT）和面向对象的软件维护（OOSM）等主要内容。

1.3.1 分析

在分析阶段，要从问题的陈述着手，建立了一个说明系统重要特性的真实情况模型。为理解问题，系统分析员需要与客户一起工作。系统分析阶段应该扼要、精确地抽象出系统必须做什么，而不是关心如何去实现。

面向对象的系统分析，直接用问题域中客观存在的事物建立模型中的对象，无论是对单个事物还是对事物之间的关系，都保留它们的原貌，不做转换，也不打破原有界限而重新组合，因此能够很好地映射客观事物。

1.3.2 设计

在设计阶段，是针对系统的一个具体实现运用面向对象的方法。其中包括两方面的工作，一是把OOA模型直接搬到OOD，作为OOD的一部分；二是针对具体实现中的人-机界面、数据存储、任务管理等因素补充一些与实现有关的部分。

1.3.3 编程

编程是面向对象的软件开发最终落实的重要阶段。在OOA和OOD理论出现之前。程序员要写一个好的面向对象的程序，首先要学会运用面向对象的方法来认识问题域，所以OOP被看作一门比较高深的技术。现在，OOP的工作比较简单了。认识问题域与设计系统成分的工作已经在OOA和OOD阶段完成，OOP工作就是用一种面向对象的编程语言把OOD模型中的每个成分书写出来。

尽管如此，学习面向对象的程序设计仍然要注重学习基本的思考过程，而不能仅仅学

习程序的实现技巧。因此，虽然本书面向的是初学编程的读者，介绍的主要是C++语言和面向对象的程序设计方法，但仍然用了一定的篇幅通过例题介绍设计思路。

1.3.4 测试

测试的任务是发现软件中的错误，任何一个软件产品在交付使用之前都要经过严格的测试。在面向对象的软件测试中继续运用面向对象的概念与原则来组织测试，以对象的类作为基本测试单位，可以更准确地发现程序错误，提高测试效率。

1.3.5 软件维护

无论经过怎样的严格测试，软件中通常还是会存在错误。因此软件在使用的过程中需要不断地维护。

使用面向对象的方法开发的软件，其程序与问题域是一致的，软件工程各个阶段的表示是一致的，从而减少了维护人员理解软件的难度。无论是发现了程序中的错误而追溯到问题域，还是因需求发生变化而追踪到程序，道路都是比较平坦的；而且对象的封装性使一个对象的修改对其他对象的影响很少。因此，运用面向对象的方法可以大大提高软件维护的效率。

读者在初学程序设计的时候，教科书中的例题都比较简单，从这些简单的例题中读者很难体会到软件工程的作用，而且题目本身往往已经对需要解决的问题做了清楚准确的描述。但尽管如此，也不应该直接开始编程，而应该首先进行对象设计。当然，本书主要的目的是介绍编程方法。建议读者在熟练掌握了C++语言编程技术后，另外专门学习面向对象的软件工程。

1.4 C++语言的起源与特点

1.4.1 C++语言的起源

C++是美国贝尔实验室的Bjarne Stroustrup博士在C语言的基础上，于1980年开发出来的一种程序设计语言。C++语言弥补了C语言的一些不足，增加了面向对象的特征。最初，Bjarne Stroustrup博士把这种语言叫做“含类的C”，1983年才正式取名为C++。

C++语言继承了C语言原有的精髓，例如高效性、灵活性，增加了对开发大型软件非常有效的面向对象的特征，弥补了C语言不支持代码重用、不适宜开发大型软件的不足，成为一种既可用于表现过程模型，又可用于表现对象模型的优秀的程序设计语言。

1.4.2 C++语言的特点

1. C++对C语言的改进

（1）C++语言增加了一些运算符，使其功能有所增强。

例如：::、new、delete和->*。

（2）C++语言对变量的声明更加灵活，在程序中可以根据需要随时声明变量。

（3）C++语言的类型多采用强制转换，取消了对函数的缺省类型，并规定函数必须用原型说明，增加了安全性。