# 面向对象方法和C++程序设计

黄平牧　肖　波　编著

北京邮电大学出版社
www.buptpress.com

# 面向对象方法和C++程序设计

黄平牧　肖　波　编著

北京邮电大学出版社
·北京·

## 内 容 简 介

本书较全面、细致地介绍面向对象的方法和C++语言。在内容的安排上，被分成面向对象的编程语言以及面向对象的分析和设计两个部分。

在面向对象编程语言部分，主要介绍C++语言的相关知识，包括：对象的封装方法，代码复用技术，多态的应用，泛型编程，异常处理机制，输入输出操作等。

在面向对象的分析和设计部分，首先介绍如何从陈述需求开始，构建三个分析模型（对象模型、动态模型和功能模型），然后阐述怎样对这三个模型进行扩充，以完成面向对象的设计过程。

为了使读者能较好地理解和掌握本书的内容，各章末尾均配备了练习题。本书可供高等院校信息、通信、计算机等专业的师生使用，也可作为广大应用计算机人员的重要参考书。

**图书在版编目(CIP)数据**

面向对象方法和C++程序设计/黄平牧，肖波编著. --北京：北京邮电大学出版社，2010.9

ISBN 978-7-5635-2426-6

Ⅰ.①面… Ⅱ.①黄… ②肖… Ⅲ.①面向对象语言—程序设计 ②C语言—程序设计 Ⅳ.①TP312

中国版本图书馆CIP数据核字(2010)第184834号

---

**书　　名**：面向对象方法和C++程序设计

**作　　者**：黄平牧　肖　波

**责任编辑**：赵玉山

**出版发行**：北京邮电大学出版社

**社　　址**：北京市海淀区西土城路10号(邮编：100876)

**发 行 部**：电话：010-62282185　传真：010-62283578

**E-mail**：publish@bupt.edu.cn

**经　　销**：各地新华书店

**印　　刷**：北京源海印刷有限责任公司

**开　　本**：787 mm×1 092 mm　1/16

**印　　张**：14

**字　　数**：328千字

**印　　数**：1—3 000册

**版　　次**：2010年9月第1版　2010年9月第1次印刷

---

ISBN 978-7-5635-2426-6　　定　价：24.00元

# 前　言

C++语言是目前工程领域方面应用最广泛的编程语言，面向对象的方法是21世纪主流软件设计方法，两者的结合无疑具有强大的生命力。在信息、通信、计算机等学科的工程应用领域，无论是软件开发还是硬件设计，它们都发挥着巨大的作用。

C++程序设计基础是信息工程及相关专业的重要专业基础课，它其实是C程序设计课程的延伸。C程序设计介绍的是结构化程序设计方法和指针的概念，而C++程序设计基础重点介绍面向对象的方法和C++语言对其相关概念的描述及其实现，两门课程在内容上不应出现重叠。本书就是遵循这样的思路编写的。

面向对象方法实际上是一种软件系统的分析、设计和实现方法，所对应的概念依次是面向对象分析、设计和编码，它是一种围绕真实世界的概念来组织模型的全新思维方法，程序设计过程和人的思维方式一致，开发出的软件的可重用性、可维护性好。

面向对象方法和C++语言所包含的内容丰富、概念抽象，学习的难度比较大，本书尽量做到深入浅出。在内容选择上不仅全面、细致，而且能突出重要的知识点，从内容到用例的选择上都经过了精心挑选。撰写本书的目的在于它能帮助读者很好地理解和较全面地掌握面向对象的方法和C++语言，并能用之于实际的软件开发。本书作者长期从事C++程序设计及相关课程的教学工作，在本书的写作过程中注重知识点的把握，内容的衔接和实际的应用。在内容的陈述上，尽量做到详细、全面，又不失简洁。

在本书的内容安排上，第1章绪论，较全面地概述了面向对象方法的历史背景和相关概念；第2章C++语言基础，介绍了一些与C语言所不同的新规定、新概念、新运算符，以及在函数方面的新变化；第3章类和对象，介绍类的概念和类中重要的成员以及友元；第4章为C++中的重载多态，主要内容是运算符重载和运算过程中的类型转换；第5章是C++中重要的代码复用技术——继承的概念和应用；第6章为运行时多态的实现机制——虚函数；第7章介绍了参数多态——模板的概念，并在此基础上介绍泛型编程，一种通过模板来实现同一算法源代码，并将其用于不同数据类型的软件重用方法；第8章异常处理，介绍C++中的异常处理机制try-catch结构以及异常的组织和处理；第9章输入输出流，介绍C++中如何实现对标准输入输出设备的操作，还有对磁盘文件的读写以及内存的操作；第10章是面向对象的分析和设计，首先介绍面向对象的分析思路，即如何从陈述需求开始，构建对象模型、动态模型和功能模型，然后阐述怎样对这三个模型进行扩充，以完成面向对象的设计过程。

在编写本书时，力图体现以下特点：

**1. 实用性强**

书中的很多例题都结合实际问题，使读者容易理解。同时，本书的练习中有许多实用性的题目，方便读者对所学内容加深理解。

**2. 易懂**

所写的内容都经过作者的理解、消化和吸收，尤其是所有的C++编程例子都经过了上机调试运行。在内容讲解方面深入浅出，条理清楚，并配有大量的图表。读者若使用计算机，结合例子实际操作，对内容的学习、理解以及掌握都会有事半功倍的效果。

本书共有10章。第7章由肖波编写，其余各章及练习题由黄平牧编写，并由黄平牧负责全书主编。书中的所有C++代码都可通过出版社网站www.buptpress.com下载。

本书的写作得到了同事及学生的广泛支持和帮助，还得到了郭军教授、林家儒教授的大力支持，并提出很多有益的建议，在此一并表示感谢。

尤其需要说明的是，本书的编写适逢北京邮电大学“信息工程”国家第二类特色专业建设之际，得到了专业建设项目的支持(“信息工程”国家第二类特色专业建设点，项目编号：TS2423)，在此表示衷心的感谢。

由于作者水平有限，书中难免有错误和缺点。在此欢迎广大读者和同行专家提出宝贵的意见和建议，对书中错误疏漏之处批评指正，可直接将意见发送至pmhuang@bupt.edu.cn，作者将非常感谢。

作　者

2010年8月于北京邮电大学

# 目　录

# 第1章 绪 论

任何事物都有一个从低级到高级，由量变到质变的发展过程。软件产业也符合这一规律，从计算机产生那一天开始，软件设计就出现了，由早期寥寥几行、几十行的代码发展到现在动辄成百万、上千万行语句组成的程序，软件设计理念发生了很大的变化。作为特别适合于开发大规模程序的软件设计方法——面向对象方法应运而生，表现出巨大的生命力，并已成为现在主流的软件设计方法。本章试图对这种方法进行较全面、浅显的概括性介绍，努力使读者有一个较清晰和全面的理解，其中包括面向对象方法产生的历史背景、面向对象的基本概念、面向对象的软件开发等。

## 1.1 软件业历史和程序设计方法

### 1.1.1 软件业历史

软件发展的历史是与计算机系统的发展紧密相关的，更好的硬件性能、更小的体积以及更低的成本将推动更复杂软件系统的形成。相对于计算机系统的发展，软件业的历史大致可分为三个阶段。

(1) 第一个阶段是第一台计算机出现(1946 年)到 20 世纪 60 年代中期：个体化的设计体制。

这一时期经历了从电子管计算机到晶体管计算机的变革，但对计算机软件的研究和发展不够重视，基本上没有系统化的软件开发方法存在。此时的软件开发不过是根据应用的需要写出能够运行的程序。由于硬件价格昂贵，运算速度低，内存容量少，所以当时的程序员非常强调“程序设计技巧”，把缩短每一微秒 CPU 时间和节省每一个二进制存储单元，作为程序设计的重要目标。软件是为具体应用而专门设计的，大多数软件的开发者和使用者是同一个人(或同一组人)，这是一种个体化的软件环境。除了程序清单之外，一般没有其他文档资料保存下来。

(2) 第二个阶段是从 20 世纪 60 年代中期到 20 世纪 80 年代：出现“软件作坊”、“软件危机”，产生新兴学科“软件工程”。

这个时期硬件经历了从晶体管计算机到集成电路、大规模和超大规模集成电路计算机的变革，CPU 的运算速度和内存容量都有了很大提高，为计算机在众多领域中的应用

提供了潜在的可能性。为了更有效地利用硬件的能力，需要有规模大并且复杂的软件。

这个时期的早期阶段(60年代中期～70年代)出现了“软件作坊”。许多用户不再自己开发软件，而是去“软件作坊”购买。软件的开发基本上沿用早期形成的开发方法。但随着计算机系统数量的不断增加，计算机软件库开始膨胀，并出现了一些无法回避的问题，比如：在程序运行中发现错误时必须设法改正，用户有新要求时必须相应修改程序，买进新硬件或操作系统的新版本时也需改变程序以适应新环境等。上述种种软件维护工作，变得复杂而困难。软件的开发虽有一些工具的支持，例如编译连接器，但基本上还是依赖开发人员的个人能力，没有可遵循的原理、原则和方法，也缺乏有效的管理。软件可靠性、可维护性较差，“软件危机”开始出现。所谓“软件危机”就是在计算机软件开发和维护过程中所遇到的一系列问题，如：(1)不能正确估计软件开发成本和进度；(2)软件产品不可靠；(3)交付的软件不易升级；(4)软件的生产率低下。

1968年，北大西洋公约组织的计算机科学家在联邦德国召开国际会议讨论“软件危机”的问题，并在这次会议上正式提出并使用了“软件工程”这个名词，一门新兴的工程学科就此出现了。它倡导以工程的原理、原则和方法进行软件开发，以解决软件开发中存在的一系列问题。20世纪60年代末到80年代初，围绕软件项目，开展了有关开发模型、支持工具以及开发方法的研究，提出了软件开发的“瀑布”模型、结构化程序设计方法和结构化编程语言(如Pascal、C等)。

(3) 第三阶段是20世纪80年代以后：面向对象方法成为主流的软件开发方法。

这一阶段，高性能低成本的微处理机大量涌现，计算机的应用得到了普及和深化。围绕软件工程的开发过程，开展了有关软件生产技术，特别是软件复用技术和软件生产管理的研究和实践，提出了具有广泛应用前景的面向对象方法，相应的开发语言也陆续得到应用，如UML(Unified Modeling Language)、C++、Java等，并大力开展计算机辅助软件工程(CASE，Computer Aided Software Engineering)的研究与发展，各类CASE产品也相继出现，如IBM公司的Rational Rose、北京大学的JBOO等。其中UML是面向对象建模语言，C++、Java是面向对象编程语言，而Rational Rose和JBOO是使用UML语言的建模工具。

### 1.1.2 程序设计方法

有人认为结构化方法和面向对象方法是完全对立的，其实不然。无论从发展历史看，还是从其内在的意义看，面向对象方法都是结构化方法的发展。为了更好地理解面向对象方法，我们就从结构化方法的特点谈起。

结构化设计是一种面向数据流的设计，这点是它与面向对象方法的最大差异。从最初画数据流图开始，到最后由顺序、选择(分支)和循环结构组成源程序，始终是以数据的流向为线索的。在这里，算法被做成功能模块(函数或过程)可以被调用，输入的数据经过算法的加工变成输出的数据，整个程序就是一个对数据顺序加工的过程。这种结构对数据的流向表达得很清晰，但对事件的并发性以及数据间的层次关系并没有充分的体现，不太容易理解。

自顶向下逐步求精是结构化设计方法的核心。结构化方法的过程就是一个从抽象到

具体的过程。对于一个比较庞大而复杂的软件系统，从提出设计目标到产生源程序之间有着很大的距离。要保证设计过程顺利进行，设计结果满足要求，并且稳定可靠、易于维护，设计活动必须遵循正确、固定的步骤和规律来进行，使设计过程规范化（如采用“瀑布模型”开发软件）。

自顶向下逐步求精的过程就是划分模块、向下分解，再把模块划分成子模块的过程。模块的划分是基于功能的划分，即先将总任务分解为若干子任务，再将子任务分解为更小的子功能，直到子功能简单得不能再分为止。在每一步由上一层到下一层的求精过程都应确保系统功能的正确性。换句话说，如果实现了下一层的全部子功能，也就实现了上一层的功能。但是，这种方法不能满足设计开发复杂的软件系统，因为功能一旦有变就会使开发的软件系统产生较大的变化，甚至推倒重来。一般说来，软件规模在三四万行以内时，结构化方法是可以满足需要的，规模超过这个尺度时，软件的开发和维护就会变得越来越难控制。

软件系统的复杂性与问题本身的复杂程度密切相关。由于问题的复杂性，用户很难一开始就准确地描述自己的需求。在开发过程中，需求经常会改变，这不仅是由需求本身的复杂性所导致，也是由于用户和软件开发者相互不了解对方的工作领域而造成的。大型的软件系统通常处在不断完善的过程中，软件必须随着需求的变化而不断进行改进。

上述事实要求软件要有好的可重用性和可进化性。理想的方式就是软件系统能像硬件系统一样，不同的系统可以由标准部件的不同组合来构成，而这些部件的接口又是标准且通用的。这种思想就是面向对象方法中的封装机制，通过该机制可以创建软件组件，并使软件的重用达到较高的层次。

结构化方法中也存在着软件重用，只不过是函数或算法的重用，这种重用相对来说是低层次、低效率的。从另一方面来看，对于结构化方法来说，上一层到下一层求精过程的正确性是整个过程得以顺利进行并得出正确结果的必要条件。假如在设计的开始阶段就出现了偏差，那么到了后期再对初期的错误进行修正几乎是不可能的，这需要将所有程序推倒重来。因此结构化方法也不能很好地适应系统的进化。

面向对象的方法将数据及对数据的操作封装在一起，以对外隐蔽的方式把每个组件之间的耦合度降到最低，通过标准接口相互通信，使程序的易维护性和可进化性大大提高。

软件产品还要求程序结构和数据结构具有良好的安全性，这样才能保证整个系统的可靠性。结构化的程序中对数据的访问缺乏安全性的限制机制，任何操作都有可能造成对数据的破坏。而面向对象的程序结构在安全性保障方面更有效，如作为类的属性的变量都有其被访问的范围，超出这个范围的程序代码将被禁止访问该变量。

面向对象方法对复杂数据具有良好的描述能力，它不仅能定义数据的结构，而且还能定义作用在其上的操作，这样从程序的外部来看，程序处理逻辑清晰简单，比结构化程序具有更好的可读性。为使读者能更好地了解两种程序设计方法的不同思路，请看下面的实例。

**例 1.1** 为图书馆的出纳台编写一个管理程序。

**解** 程序的需求如下所述：学校图书馆(library)有数十万本藏书(books)和数千位读

者(readers)。每本书具有书号(id)、书名(name)、作者(author)、专业(speciality)和借阅人(borrower)等登记项,其中各书的书号不重复。每位读者具有读者号(id)、姓名(name)、专业(speciality)和所借图书(holding)等登记项,读者都是本校的教师(teachers)和学生(students)。出纳台的功能有:

(1) 注册(join)新读者;

(2) 查询(acquire)某书下落;

(3) 借出(lend)某书给某人;

(4) 收回(receive)某人归还的某书。

按照图书借阅规则(rules),教师可同时借本专业书5本和其他书3本,借期最长3个月;学生可同时借本专业书3本和其他书1本,借期最长1个月。

按结构化方法的分析思路,该程序包含4个功能:注册新读者、查询图书、借书和还书,分别对应程序中的4个函数:join(注册新读者)、acquire(查询图书)、lend(借书)、receive(还书)。主要有两类数据:读者(Readers)和图书(Books),如图1.1所示。

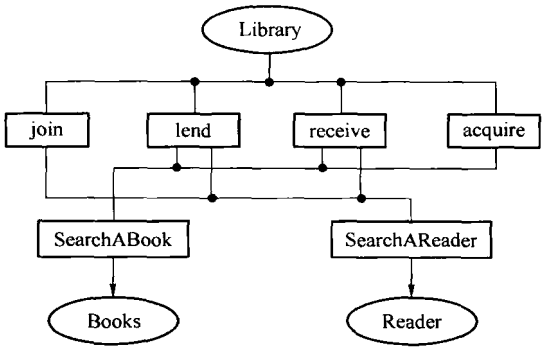


图1.1 功能优先分解的抽象系统[3](结构化方法)

按面向对象方法的分析思路,整个活动(注册新读者、查询书、借书和还书)将在三个实体(读者、书库和出纳台)之间进行,如图1.2所示。

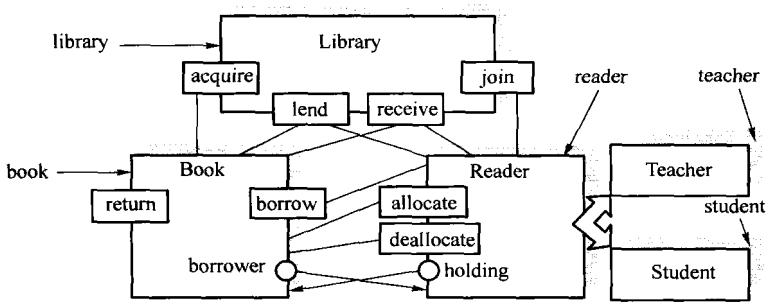


图1.2 对象优先分解的抽象系统[3](面向对象方法)

比较一下图1.1和图1.2,可以看出:结构化方法的首要问题是设计过程,也就是功能分解,需要实现的功能为:注册新读者(join),借书(lend),还书(receive)和查询图书

(acquire)。数据和处理数据的过程是分离的。而面向对象方法的首要任务是确定所需要的对象类:出纳台(Library)、书库(Book)和读者(Reader),读者又被分为教师(Teacher)和学生(Student)两类。数据和对数据的操作方法结合在一起形成对象,并引入继承等概念,使程序设计过程与人的思维方式一致。

## 1.2 面向对象的基本概念

### 1.2.1 什么是面向对象

所谓面向对象,就是以对象的观点来分析现实世界中的问题。从普通人认识世界的观点出发,把事物归类、综合,提取共性并加以描述。在面向对象的系统中,世界被看成是独立对象的集合,对象之间通过"消息"相互通信。对象被描述为数据(又称为属性)以及基于这些数据的行为(又称为方法)的复合体。

面向对象实际上是一种软件系统的分析、设计和实现方法。文献[6]给出的面向对象的定义是:面向对象是一种运用对象、类、继承、封装、聚合、消息传送和多态性等概念来构造系统的软件开发方法。它是一种围绕真实世界的概念来组织模型的全新思维方法,其基本思想是:对问题空间进行自然分割,以更接近人类的思维方式,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,使设计出的软件尽可能直接地描述现实世界,构造出模块化的、可重用的、维护性好的软件,并能够控制软件的复杂性和降低开发维护费用。在面向对象方法中,对象是核心概念,所有面向对象的技术都是建立在这个概念上的。

### 1.2.2 对象和对象的模型化

**1. 对象**

对象是一种事物,是一种可以看得到、摸得到或感觉得到的实体[7],如汽车、空气、人等。这里的对象虽然是现实世界的,但却可以把它应用到计算机领域,作为我们考虑问题的出发点。可将现实世界中的对象与计算机软件操作中的抽象对象相对应,用现实世界中的实体用语(如汽车、空气、人等)来描述和分析问题。

**2. 对象的模型化**

(1) 认识属性(Attribute)

对象内凡是可以描述自身状态、性质的数据名称的总和称为属性。例如一个阀门应当有流量、材质、大小、压力等数据名称。又如骰子应有点数、形状、颜色等反映其性状的数据名称。

(2) 认识方法(Method)

对象在外力作用下产生的可以改变其部分或全部属性值的动作行为的总和称为方法。如骰子的投掷。

(3) 封装(Encapsulation)

当对象含有完整的属性和与之相对应的方法时，称为封装。这样的对象可以用一种被称为“田田圈”的图形表示，如图 1.3 所示。由此可以得出两个推论：

- 从对象的外面不能直接访问对象的属性，而要通过对象的方法才能访问它；
- 对象的方法可以接收来自对象以外的信息。

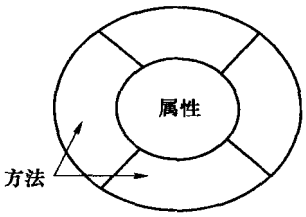


图 1.3 对象的“田田圈”表示[4]

封装是面向对象方法的一个重要原则，它把属性和方法结合在一个对象里，对象的属性(内部信息)对外界隐藏，只能通过对象提供的方法(接口)进行访问。对于外界来说，只能知晓对象的外部行为而无法了解对象的内部实现细节，这样可以保证对象属性数据的安全性。

封装有两层含义：

(1) 结合性。把对象的全部属性和方法结合起来，形成一个独立的不可分割的单位。

(2) 信息隐藏性。尽可能隐蔽对象的内部细节，对外形成一个边界，只保留有限的对外接口与外部发生联系。

封装的基本单位是对象，如集成电路。封装的目的在于将对象的使用者和对象的设计者分开，使用者不必知道行为(方法)实际的实现细节，只需调用设计者提供的接口来访问该对象。把定义模块和实现模块分开，可以大大提高软件的可维护性、可修改性。

### 1.2.3 对象的抽象与类

在一个由对象构成的系统中会大量存在基本数据结构相同，仅属性的具体数值不同的一类对象，将这类对象进行抽象就形成了类的概念，类是对对象进行封装的工具。

类具有以下几个特点：

- 类是对象的属性和方法的抽象结构描述；
- 当提供具体的属性数值后便可使用类生成具体的对象；
- 对象内的属性和方法可以设立访问权限。

在 UML 语言中，为了能够明确地将具体的对象与其抽象形式——类相区别，而采用了图 1.4 所示的符号。

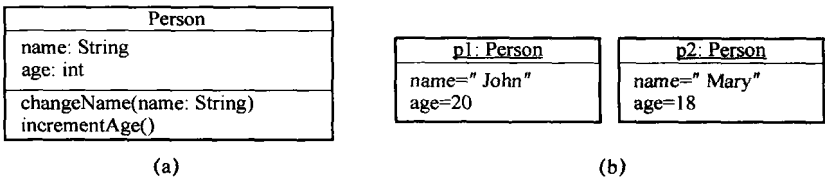


图 1.4 UML 中对象和类的表示

图 1.4(a)表示类 Person，图 1.4(b)是类 Person 生成的两个人 p1(“John”)和 p2(“Mary”)。两者之间的主要差别是在对属性的描述上：类中的属性只给出了属性的名称和类型，而对象中的属性则必须给出确定的数值。