新世纪高等学校计算机系列教材

面向对象程序设计与C++语言

（第二版）

◎ 杨庚　王汝传　叶晓国　编著

人民邮电出版社
POSTS & TELECOM PRESS

新世纪高等学校计算机系列教材

面向对象程序设计与C++语言

（第二版）

杨 庚　王汝传　叶晓国　编著

人民邮电出版社

图书在版编目（CIP）数据

面向对象程序设计与C++语言 / 杨庚，王汝传，叶晓国编著．—2版．
—北京：人民邮电出版社，2006.11
（新世纪高等学校计算机系列教材）
ISBN 7-115-15194-6

Ⅰ．面…　Ⅱ．①杨…②王…③叶…　Ⅲ．①面向对象语言－程序设计
－高等学校－教材②C++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（2006）第101816号

内 容 提 要

本书系统地介绍了面向对象技术及C++语言的相关知识。内容包括面向对象技术的概念和特征、C++语言基础、类和对象、派生与继承、虚函数与多态性、模板、运算符重载、输入/输出流库、异常处理等。

本书注重基本概念，从实际应用出发，突出重点，叙述清楚，深入浅出，论述详尽，使读者既能深刻领会面向对象程序设计的思想，了解面向对象程序设计的特征，又能掌握C++语言的编程与应用。

本书可作为高等学校计算机及相关专业面向对象程序设计的教材，也可作为其他专业师生和科技工作者的参考用书。

新世纪高等学校计算机系列教材
面向对象程序设计与C++语言（第二版）

◆ 编　　著　杨庚　王汝传　叶晓国
责任编辑　邹文波

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街14号
邮编　100061　　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京艺辉印刷有限公司印刷
新华书店总店北京发行所经销

◆ 开本：787×1092　1/16
印张：16.25
字数：390千字　　2006年11月第2版
印数：18 001－21 000册　　2006年11月北京第1次印刷

ISBN 7-115-15194-6/TP・5662

定价：25.00元

读者服务热线：(010)67170985　印装质量热线：(010)67129223

第二版前言

面向对象技术是近年来迅速发展的一个研究领域，它对信息科学、软件工程、人工智能、系统工程和认知科学等学科都有重要的影响，尤其在计算机科学与技术的各个方面（如数据库技术、操作系统设计、多媒体技术和计算机网络等）影响深远。面向对象技术的出现被认为是程序设计方法学方面的一场实质性革命，它与传统的结构化程序设计相比，具有许多优点。这种方法从根本上改变了人们以往设计软件的思维方式，从而使程序设计者摆脱具体的数据格式和过程的束缚，将精力集中于要处理对象的设计和研究上，极大地减少了软件开发的繁杂性，提高了软件开发的效率。

C++是一种面向对象程序设计语言，但不是纯面向对象语言，而是混合型面向对象语言，它在过程语言中增加了面向对象的结构。这种特性使得 C++语言保持与 C 语言的兼容，使许多 C 语言代码不经修改就可以为 C++语言使用。用 C++语言编写的程序可读性更好，代码结构更合理，可以直接在程序中映射问题空间的结构。C++可以为面向对象技术提供全面支持，它是实现面向对象技术比较通行和适用的手段，要想理解和掌握 C++语言，离不开面向对象技术的指导，而介绍面向对象技术也必须结合 C++语言的运用。

随着计算机科学与技术的飞速发展和计算机应用的普及，广大科技工作者和高等学校的学生迫切需要了解和掌握面向对象技术，学会用 C++语言编写程序、开发软件。本书就是为了适应这一形势需要，在总结作者从事面向对象技术和 C++语言研究、应用和教学与实践经验的基础上，参考国内外有关资料编写而成的。其目的是为广大科技工作者及高等院校相关专业教师和学生系统地学习、掌握面向对象技术和 C++语言提供一本实用教材，使读者能迅速了解并掌握面向对象程序设计的概念和方法，深刻理解面向对象程序设计的本质，并能将其应用于实际编程之中。在该书的再版过程中，对内容进行了再次审核，体现了教育部 2003 年《关于进一步加强高校计算机基础教学的几点意见》的思想。

本书共分 10 章。第 1 章、第 2 章主要介绍面向对象技术的基本概念和特征。第 3 章至第 10 章详细描述了用 C++语言进行面向对象程序设计的思相和方法，包括 C++语言的基本语句、函数、类和对象、派生与继承、虚函数与多态性、模板、运算符重载、I/O 流库、异常处理等。每章有大量练习题，供教学选用，以便巩固所学内容、提高编程能力。

本书第 1 章、第 9 章由王汝传老师编写，第 10 章由叶晓国老师编写，其余各章由杨庚老师编写，全书由杨庚老师统稿。南京邮电大学博士生导师王绍棣教授、南京航空航天大学博士生导师秦小麟教授在百忙之中认真细致地审阅了全部书稿，提出了许多有益的修改意见。南京邮电大学教务处和教材科将本书列为校十五规划建设教材。此外，书中还引用了其他同行的工作成果，在此向他们一并表示衷心感谢。

由于作者水平有限，书中难免存在错误与不妥之处，敬请读者批评指正。

编　者

2006 年 7 月

目　录

第1章　面向对象技术概述

这一章主要介绍面向对象技术的概念、基本特征、面向对象程序设计语言的发展现状等内容。

1.1　面向对象技术概念

面向对象技术概念来源于程序设计，已逐步成为目前软件开发领域的主流技术。但它现在显然不仅局限于程序设计方面，而已经发展成为软件开发领域的一种方法论，是一种全新的设计和构造软件的思维方法，它使计算机解决问题的方式更加类似于人类的思维方式，更能直接地描述客观世界。

众所周知，计算机的发明和使用，对人类社会的进步和发展起了巨大的作用，是现代文明的重要组成部分，是信息革命的基础。然而，计算机硬件技术和软件技术的发展却是不平衡的，这极大地限制了计算机系统的发展和应用。从1946年第一台电子计算机问世以来，计算机的硬件已经历了四代的变化，即电子管时代、晶体管时代、集成电路时代以及大规模集成电路时代，计算机的硬件性能取得了长足的发展，速度、容量等成倍增长，而价格却成倍下降，而且，计算机的硬件水平还在突飞猛进地发展着。但这几十年来，计算机软件的性能却基本上没有发生根本性的变化，尽管各高等院校和研究机构在软件开发工具和技术方面取得了较大的进展，但这些进展并没有推广到软件开发者的日常工作中，当今软件开发过程的研究与实际应用需求是不相适应的，程序员实际上仍在采用较原始的方式进行工作，他们编出来的程序代码缺乏一般性、可读性和可扩充性。由于软件产品不能适应诸如硬件、操作系统的改进，特别是不能适应用户要求的改变或提高等不断变化的环境，因此，软件的维护不仅占据了相当大的比重，而且非常困难，软件的生命周期很少能达到10年以上，这一问题的实质在于软件技术的发展远远落后于硬件技术的发展。

另外，随着计算机硬件性能的提高和价格的下降，计算机得到了大规模的推广、普及和应用，软件的应用范围越来越广，软件的规模越来越大，要解决的问题越来越复杂，人们对软件功能的要求越来越多，并且对软件性能的要求越来越高，因此，传统的软件开发技术、方法和工具不足以满足这些日益增长的要求，特别是组织大型的软件开发，这将使软件人员陷入困境。

大家都知道，用计算机解决问题时需要用程序设计语言对问题的求解加以描述，实质上，软件是问题求解的一种表达形式。显然，如果软件能够直接地表现求解问题的方法，则软件不仅易于被人理解，而且易于维护和修改，从而可提高软件的可靠性和可维护性。此外，如果能按人们通常的思维方式来建立问题域的模型，则可以提高公共问题域中的软件模块化和

重用化的可能性。面向对象技术的基本原则是：按人们通常的思维方式建立问题域的模型，设计出尽可能自然地表现求解方法的软件。

和人们认识世界的规律一样，面向对象技术认为：客观世界是由许多各种各样的对象组成的，每种对象都有各自的内部状态和运动规律，不同对象间的相互作用和联系就构成了各种不同的系统，构成了我们所面对的客观世界。可见，对象是组成一个系统的基本逻辑单元，是一个有组织形式的含有信息的实体。它既可以表示一个抽象的概念，也可以表示一个具体的模块，当然也就既可以表示软件，也可以表示硬件。于是，面向对象的技术既提供了一个分析、设计和实现系统的统一方法，又提供了描述、设计、实现硬件和软件系统的统一框架。

为了实现用面向对象技术进行系统分析的目的，必须建立直接表示组成问题域的事物以及这些事物间相互联系的概念，还必须建立适应人们一般思维方式的描述范式（Paradigm）。在面向对象技术中，对象（Object）和传递消息（Message Passing）分别是表现事物及事物间相互联系的概念；类（Class）和继承（Inheritance）是适应人们一般思维方式的描述范式；服务（Service）是允许作用于该类对象上的各种操作。这种对象、类、消息和服务的程序设计范式的基本点在于对象的封装性（Encapsulation）和继承性。通过封装能将对象的定义和对象的实现分开，通过继承能体现类与类之间的关系以及由此带来的动态聚束（Dynamic Binding）和实体的多态性（Polymorphism），从而构成了面向对象技术的基本特征。

1.2 结构化程序设计与面向对象程序设计的区别

20世纪90年代以来，面向对象程序设计（Object Oriented Programming，OOP）异军突起，迅速地在全世界流行，并一跃成为程序设计的主流技术。面向对象程序设计是软件系统设计与实现的新方法，面向对象程序设计和传统结构化程序设计（Structure Programming，SP）比较起来，有许多优越性。那么，什么是面向对象程序设计？在对它给出解释之前，需首先讨论一下结构化程序设计。

结构化程序设计是20世纪60年代诞生的，在20世纪70年代到20世纪80年代已遍及全球，成为软件开发设计领域和程序员通常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路是：自顶向下、逐步求精；其程序结构是按功能划分成若干个基本模块，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；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，自顶向下、分而治之的手段，从而有效地将一个较复杂的程序系统的设计任务分成许多易于控制和处理的子任务，这些子任务都是可独立编程的子程序模块。这些子程序都有一个清晰的接口，使用起来非常方便。

结构化程序设计方法虽然具有很多的优点，但它仍是一种面向数据/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的格式。对于不同的数据格式即使要做同样的处理或对相同的数据格式要做不同的处理都需编写不同的程序，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了错误的程序模块的可能性。因此，使数据与程序始终保持相容，已经成为程序员的一个沉重负担。上述这些问题，结构化程序设计方法本身是解决不了的，它需要借助于面向对象程序设计方法才能给予解决。

面向对象程序设计既吸取了结构化程序设计的优点，又考虑了现实世界与面向对象解空间的映射关系，它所追求的目标是将现实世界的问题求解尽可能简单化。

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类的重用性。

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意义。

1.3　面向对象程序设计的作用

面向对象技术的应用为提高软件开发效率起着重要的作用，面向对象技术主要优点体现在解决软件开发过程中维护复杂性和提高生产率以及大型程序设计等方面。面向对象技术通过下面的方法充分发挥其优势：

- 编写可重用代码；
- 编写可维护代码；
- 修改代码模块；
- 共享代码。

当高质量的代码可重复使用时，复杂性就得以降低，生产率则得到提高。面向对象提供的机制，特别是继承，非常鼓励代码的重新使用。程序员不是拷贝及修改模块，而是利用精心测试过的代码的类库来开发软件。框架不久也将为诸如图形用户界面这样的复杂领域所用。继承能很大程度地提高开发人员建立、扩充和维护系统的能力。

面向对象代表了开发与使用软件在方法上的本质改变。软件可重用意味着类可被容易地组合与修改以建立新的应用程序。将数据与服务封装在一起则改变了程序设计过程的整个性质。

传统的软件开发方法将数据与函数区分开发考虑，由于数据结构经常需要修改，必须重新考虑函数与数据的一致性，从而使数据结构与函数协调一致需要花去大量的资源，并使错误产生的机会大大提高。面向对象技术将数据与服务封装在一起，简化了此过程，方便了维护，并减少了程序设计过程中出错的可能性。

当然，为了得到这些益处，开发人员必须在其分析问题及将问题转换成程序的方式上有很大的改变。面向对象的方法与传统方法显然是有很大区别的，虽然许多人都认为面向对象的程序更易于编写，但其中的概念仍比传统方法中的概念要抽象，而且起初也较难掌握，但随着对面向对象程序设计思想的不断领会，就会逐步掌握并应用自如。

1.3.1　复杂性的维护

传统的分析与设计方法将应用分解成对用户有意义的实体及关系，而面向对象的方法则使此分解过程扩展到了实现阶段，由于应用领域中的对象与软件领域中的对象有着直接的对应，所以设计与实现面向对象的应用程序较为容易。

1. 软件开发的灵活性

一旦对象被定义，类库被扩充，程序设计过程就变得越来越容易。通过继承机制进行子

类化的过程使得程序设计变成仅对子类与超类或父类的差异进行编程的过程。

继承机制不仅影响到程序构造的效率与质量，而且还影响到程序员任务的分配。由于继承使得对对象的增删不必通过大量修改应用程序的逻辑结构即可进行，所以大型项目可被划分成若干个开发组。虽然在编程小组中对任务的分工已不是一个新概念，但面向对象的程序设计方法却有利于各程序组的独立工作——对每个对象及其服务的实现完全可以独立进行。具体工作过程如下。

（1）一组开发人员先就一组抽象类及其相应的服务取得一致意见，然后将此类集作为其应用程序框架。抽象类说明了开发组必须实现的行为。

（2）接下来，开发人员可独立工作以创建能对某组消息进行处理并使之能方便地结合进整个应用程序的新的子类。开发组的某些成员可以实现协调消息发送的代码，另一些成员则可定义及实现新的子类，从而可使开发组发挥出最佳效益。

对每个对象的建立与修改都不必考虑其他组的成员是如何对他们的系统进行编码的。因此，工作可根据由公共父类所提供的一致性而在各个不同的路径上进行。面向对象的开发工具还方便了对各独立成分以及整个系统的调试。这些图形丰富的开发环境含有诸如浏览器这样的新工具，从而可使调试更快也更有效率。此外，面向对象的语言还消除了许多常见的编码错误。总之，不断出现的开发环境伴随着强壮的类库将大大减少创建、调试及集成至大型系统的代码量。

面向对象系统提供了实际实现所需要的性能与灵活性。编程可用标准商用语言（如 C 语言）的扩展来完成，并且面向对象的技术在某种程度上还能与过程语言一起使用。

面向对象的程序设计还扩展了可被编程的应用领域，因为它解除了对预定义数据类型的约束。面向对象的程序设计可接受复杂的数据结构。新的数据类型能不通过修改现有代码就被加入。此外，面向对象的环境还能封装不同的应用程序以提供一个面向对象的桥梁。

2. 可重用性

面向对象的技术可使程序员不必反复地编写类似的程序。通过用新的对象替换旧的元素或对象，或直接在应用程序中加入新的对象，程序员就能修改一程序的功能。一般指令（消息）不需修改，因为专门的实现细节（服务与数据）驻留在对象之中，每个对象都知道如何执行其自己的动作，这一概念与面向结构化程序设计完全不同。在结构化编程中，操作及规则都是作用到独立的数据集上的，程序员将注意力集中在语言问题；而在面向对象环境中，重要的问题是建立在各种环境下可被使用的类库或对象集。类库可提供高质量的应用程序建立模块。类不仅提供了模块化与信息隐蔽，而且也提供了为继承与多态性所进一步促进的可重用性。

可重用软件的传统技术并没有被面向对象技术取代。事实上，传统技术常常含有面向对象的成分。例如，程序骨架的概念就为抽象类的概念所蕴含。通常，传统的程序员是通过拷贝与编辑而重用代码的，而面向对象的程序员则可通过创建一些子类并重载其某些方法而完成同样的功能。

1.3.2 生产率的提高

1. 可扩充性与可维护性

修改与扩充一个面向对象应用程序是较容易的，可以增加新的对象类型而不需改变已有

的结构。继承特性使一个新的类可以从旧的类派生而来，类中的服务是易于修改的，因为它们被放在唯一的地方，而不是分散在程序中或潜在地在程序中多次重复。因此，使用面向对象技术时，就不需要在整个过程体中搜索、替代函数和变量了。

面向对象特征对于软件维护也是特别有用的。模块化使得对程序的修改变得更加有效。多态性减少了过程个数，因而也减少了维护人员需要理解的程序规模。类继承性使得可以建立一个程序的新版本而不影响旧版本。继承机制把程序修改作为子类归档，这些子类表示对超类的修改历史。

创建一个子类，即通过仅描述其本身与其父类之间的差别而定义一个新类的面向对象技术，使确定一个程序与其旧版本有何不同变得更加容易。一个子类集表示对一个超类的修改历史。继承机制减少了人为错误，因为对一个类的修改将会自动传播到它的所有子类。

面向对象技术还被成功地结合到传统体系中。如 C++这样的混合体系在程序设计开发工具中增加了面向对象功能，这使得程序员可以在自己熟悉的背景中利用新技术。

2. *方便用户编程*

面向对象程序设计技术使程序设计工作变得更加容易。事实上，用户虽然不知道应用程序内部细节，但是用户可以结合面向对象技术模型和类库中的知识，建立和扩充应用程序。

1.3.3　设计大型应用程序

大型程序的复杂性主要表现在两个方面：一方面，由于大型程序必须由多人合作完成，如何划分任务、估计和分配资源、掌握每个程序员的进度、控制及检查每个阶段的设计标准等，构成了进行大型程序设计时管理的复杂性，也是大型程序设计与单人能够完成的小型程序设计的一个重要区别；另一方面，大型程序具有大量的系统状态，要正确地处理它的中间状态、组织系统的程序逻辑和验证系统的正确性都是比较困难的。

对于大型程序，正确性是程序的首要目标，程序的任一微小错误都可能造成重大的损失，并且大型程序设计的周期长，需要多人合作，不可避免地要出现一些错误。

但是，对于一个大型程序，仅仅满足正确性是远远不够的，易维护性、可读性和可重用性都是非常重要的，这是因为，要想在短期内重新开发一个大型程序是不可能的。易维护性能使程序在出现错误时，得到及时的更正；可读性又可以保证程序易于理解、便于使用和调试，从而使程序功能得以充分的发挥；可重用性可以使一个程序中开发的通用模块能够在其他程序中再次使用，从而节省开发费用，简化程序的部分复杂性。

在开发一个大型系统时，应对整个任务进行清晰的、严格的划分，使每个程序员清晰地了解自己要做的工作以及与他人的接口，使每个程序员可以独立地设计调试自己负责的模块，以使各模块能顺利地应用到整个系统中去。

大型程序在设计时采用模块化的方法，将一个问题分解成若干个小问题，而每一个小问题又可以再分解成更小的问题，每个小问题都可以是一个独立的模块，这些模块都有一个清晰的抽象接口，它只说明做什么，不必说明如何去做，同时还说明模块之间的关系。这些模块构成了一种层次结构，在设计时采用自顶向下，分而治之的办法。

面向对象技术提供了一种有效的模块分解方法，进一步发展了基于数据抽象的模块化设计，并且在数据抽象和抽象数据类型之上又引入了动态连接和继承性等机制，使其更好地支持大型程序设计。

1.4 面向对象程序设计语言

1.4.1 程序设计语言发展概况

自计算机系统不断普及应用以来，与其密切相关的计算机语言也经历了由“机器语言”到“汇编语言”，再进一步到“高级语言”的发展历程。由于高级语言克服了机器语言与汇编语言的缺点，使人们采用类似于自然语言的方式同计算机通信，且标准化的语言文本又可独立于具体的计算机，因此，高级语言的出现与不断向前发展对提高计算机软件产品的生产率、提高其质量都起着重大的作用。高级语言的发展具体可分为 3 个阶段。

1. 基础语言的发展阶段

基础语言是目前人们最为熟悉、应用最为广泛的语言。 基础语言的共同特点是具有很好的语言表达能力，可用于科学计算和商业事务处理中，如 Fortran、Cobol、Algol、Basic 均属于基础语言，其中，Fortran 语言是 20 世纪 50 年代发展起来的。随着软件规模的不断扩大，这些以语句序列作为程序的基础语言就显得不便于管理和维护，迫切要求采用结构化的程序设计语言。

2. 结构化程序设计语言的发展阶段

为了克服 20 世纪 60 年代出现的软件危机，1968 年由北约组织提出了“软件工程”的概念。对程序设计语言的认识从强调以表达能力为重点转向以结构化和简明性为重点，即增强软件的工程化，将程序从语句序列转向将程序作为相互作用的模块集合。正是基于软件工程化的迫切要求，在 20 世纪 70 年代结构化语言获得了蓬勃发展并得到广泛应用。其中，Pascal、C 语言都属于这类语言的代表。这类语言的共同特点是都支持结构化的程序设计原理。按照这一原理，程序中的任何逻辑问题均可用“顺序”、“选择”和“重复”3 种基本结构加以描述。

结构化语言的编程方法就是确定所需要的过程和采取能找到的最好算法。其中过程设计是关键，也就是将要设计的软件过程用有限个基本结构表示出来；其次，还需要有好的算法来执行所要求的计算。结构化语言支持过程设计是通过将参数传送给函数，并由函数返回值来实现的。采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；使用为数不多的基本结构可使过程描述清晰、易读易懂；更宝贵的特性是使过程的构造可由粗到细、随着设计的深化逐步求精、逐步分解与细化。正是由于这一系列的特色和优点，使结构化语言从 20 世纪 70 年代以来获得越来越广泛的应用，取得了辉煌的成果，以致计算机应用人员将 20 世纪 70 年代誉为结构化语言的年代。

3. 面向对象程序设计语言的发展阶段

进入 20 世纪 80 年代后，尤其是近几年，一系列高技术，如第五代计算机、CAD/CAM/CIMS、CASE、知识工程等项目的研究与实现都迫切要求大型的软件系统作为支撑；它们所使用的数据类型也越出了常规的结构化数据类型的范畴，提出对图形、语言、规则等非结构化的信息的管理。为了适应这些应用领域的需要，强烈要求软件模块应具有更强的独立自治性，以便于大型软件的管理、扩充与重用。结构化语言的数据类型较为简单，不能胜任对非结构化数据的定义与管理，采用过程调用机制也不够灵活，独立性较差，在规模庞大而复杂

的软件面前，它已显得力不从心。

为了适应高技术发展的需要，为了消除结构化程序设计语言的局限性，自 20 世纪 70 年代以来研制出了各种不同的面向对象程序设计语言，现在公认的第一个面向对象程序设计语言是 Smalltalk。它是由美国的 Xerox 公司于 20 世纪 70 年代初研制的，该语言第一次使用了“面向对象”的概念和程序风格。

面向对象语言的出现并非偶然，它是程序设计语言发展的必然结果。事实上，面向对象语言 Smalltalk 中类和继承的概念是源于 20 世纪 60 年代开发的 Simula 语言，它的动态联编（聚束）概念和交互式开发环境的思想则来自于 20 世纪 50 年代诞生的 LISP 语言，其信息隐藏与封装机制则可以看作是 20 世纪 70 年代出现的 CLU 语言、Modula-2 语言及 Ada 语言数据抽象机制的进一步发展（因为类就是一种抽象数据类型）。

Smalltalk 的问世，标志着面向对象程序设计语言的正式诞生。20 世纪 80 年代以来，面向对象语言得到飞速发展，形形色色的面向对象语言如雨后春笋般地出现。这时候，面向对象语言朝着两个方向发展：一个方向是纯面向对象，如继 Smalltalk 之后，又出现了 Eiffel、SELF 等语言；另一个方向是混合型面向对象语言，如将过程型与面向对象结合产生了诸如 C++、Objective C、Object Pascal、Object Assembler、Object Logo 等一大批语言，将函数型（LISP）与面向对象结合产生了诸如 LOOPS、Flavors、CLOS 等语言，将逻辑型（PROLOG）与面向对象结合产生了诸如 SPOO 等语言。

当然新推出的程序设计语言和软件平台几乎都是面向对象的或基于对象的。例如我们熟知的 Borland C++、Visual C++、Visual Basic、PowerBuilder、Windows 95、Windows NT 等，这些语言和软件平台把 OOP 的概念和技术与数据库、多媒体、网络等技术融为一体，成为新一代的软件开发工具与环境。它们的出现标志着 OOP 已全面进入软件开发的主战场，成为软件开发的主力军。

1.4.2　面向对象程序设计语言简介

随着面向对象技术的发展，各种各样面向对象的程序设计语言不断出现，下面简要介绍几种面向对象程序设计语言。

1. Smalltalk

Smalltalk 是第一个流行的面向对象语言，其起源可追溯到 20 世纪 60 年代后期，那时 Alan Kay 是犹他大学的研究生。他认识到可以利用 Simula 语言中的概念来扩展他在图形方面的工作，到 Xerox 公司后，Alan Kay 将这些早期的思想发展为基于图形的、交互式个人工作站的概念。

Smalltalk 自问世后，经过不断的改进，到 1981 年推出了 Smalltalk 80，它是目前最流行的版本。

Smalltalk 被认为是最纯正、最具代表性的面向对象程序设计语言。Smalltalk 中的一切元素都是对象，如数字、符号、串、表达式、程序等。类也是对象，类是元类的对象。该语言从本身的实现和程序设计环境到所支持的程序设计风格，都是面向对象的。它仍然在面向对象程序设计乃至面向对象技术中扮演着不可取代的重要角色。

2. C++

C++是 AT&T Bell 实验室的 Bjarne Stroustrup 于 20 世纪 80 年代早期提出的，是迄今为止