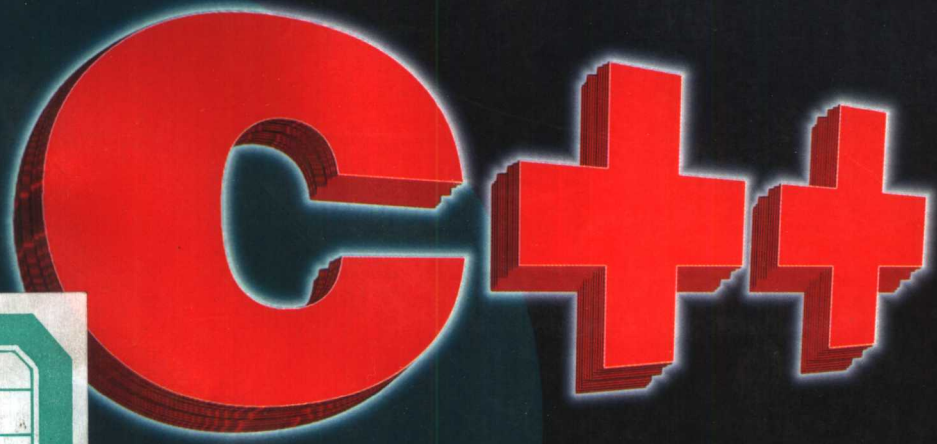
C++ 面向对象程序设计简明教程

廉师友　编著



西安电子科技大学出版社

C++ 面向对象
程序设计简明教程

廉师友　编著

西安电子科技大学出版社

1998

内 容 简 介

本书简明、系统地介绍了 C++ 面向对象程序设计的基本原理、基本方法和技术。书中第 1 章概述了面向对象程序设计的基本概念、主要特点、发展和应用情况、语言谱系以及面向对象技术的基本内容。第 2 章介绍了 C++ 对 C 语言在非面向对象方面的扩充。第 3 至 7 章详述了 C++ 提供的面向对象程序设计的各种语言设施，包括：类、对象、消息、派生类、继承、多态、模板、流库等。第 8 章介绍了面向对象程序设计的一般步骤、方法和技巧，并给出了程序实例。

本书语言通俗、简明实用，可作为高等院校计算机及相关专业面向对象程序设计课程的教材，也可作为其他人员自学面向对象程序设计的教材或参考书。对于那些已有 C 语言基础的读者，利用本书对 C++ 则可无师自通。

C++面向对象程序设计简明教程

廉师友　编著
责任编辑　陈宇光
出版发行　西安电子科技大学出版社
（西安市太白南路 2 号）
邮　编　710071
电　话　（029）8227828
经　销　新华书店
印　刷　陕西省富平印刷有限责任公司
版　次　1998 年 11 月第 1 版
1998 年 11 月第 1 次印刷
开　本　787 毫米 × 1092 毫米　1/16　印张　10.875
字　数　251 千字
印　数　1~4 000 册
定　价　11.00 元

ISBN 7-5606-0659-8/ TP·0331

前　言

90 年代以来，面向对象程序设计(Object-Oriented Programming, 简称 OOP)异军突起，迅速地在全世界流行，并一跃而成为程序设计的主流技术。近年来推出的新一代程序设计语言、软件开发工具与环境以及操作系统，几乎都是面向对象的。这种形势给人们的一个直接印象是：不学习面向对象程序设计，将跟不上计算机发展的步伐；不懂得面向对象程序设计，将面临沦为计算机外行的危险。

OOP 为什么会受到如此青睐？OOP 为什么会发展得这样迅速？这是因为 OOP 是不同于传统程序设计的一种新的程序设计范型。OOP 中的对象、类、继承、多态等概念和技术，对降低软件的复杂性、改善其重用性和可维护性，提高软件的生产效率，有着十分重要的意义。因此，OOP 被普遍认为是程序设计方法学的一场实质性革命，OOP 技术也成为被普遍看好的一门新技术。另一方面，OOP 非常适合于新一代图形界面的操作系统和计算机网络环境。事实上，这两种环境下的软件开发已普遍采用 OOP。

还需指出的是，OOP 现在还引出了一门崭新的技术——面向对象技术（OOT）。除 OOP 外，OOT 还包括：OOA（面向对象系统分析）、OOD（面向对象系统设计）和 OODB （面向对象数据库）等一系列软件技术。事实上，OOT 已在软件开发、人机界面、操作系统、知识工程甚至硬件结构等众多领域展现出强大的生命力，扮演着越来越重要的角色。

正是在此种情况下，近年来许多院校纷纷将面向对象程序设计及面向对象技术正式列入教学计划，作为计算机专业学生的必修课；许多有识之士也都纷纷把目光转向面向对象程序设计。

C++ 语言是在 C 语言的基础上扩充了面向对象机制而形成的一种面向对象程序设计语言。C++ 除继承了 C 语言的全部优点和功能外，还支持面向对象程序设计，而且主要是用于面向对象程序设计。由于与 C 语言的关系，再加上它的面向对象特征，C++ 现在已成为一种十分流行的面向对象程序设计语言。因此，C++ 也就自然成为介绍面向对象程序设计的首选语言。

学习 C++ 不仅可深刻理解和领会面向对象程序设计的特点和风格，掌握其方法和要领，而且可使读者掌握一种十分实用的程序设计语言。同时，学会了 C++ 面向对象程序设计，对别的面向对象语言和工具环境也就不难触类而旁通了。另外，学习 C++，还可使读者得到一个意外的收获，那就是再学当前流行的网络程序设计语言 JAVA 就很容易了。因为 JAVA 的语法格式与 C++ 基本一样，而且也是面向对象的。

市面上目前关于 C++ 的书籍并不少见，但其中以大部头和跟踪新版本的居多，所以，真正要找一本合适的教材用书却还不太容易。

鉴于以上情况，我们编写了这本《C++面向对象程序设计简明教程》，旨在使读者迅速迈入面向对象程序设计的大门，并真正掌握基于 C++ 的面向对象程序设计技术。

本书第 1 章概述了面向对象程序设计的基本概念、主要特点、发展和应用情况、语言谱系以及面向对象技术的基本内容。第 2 章介绍了 C++ 对 C 语言在非面向对象方面的扩

充，为后面的编程作好了准备。第 3 至 7 章详述了 C++ 提供的面向对象程序设计的各种语言设施，包括：类、对象、消息、派生类、继承、多态、模板、流库等。第 8 章介绍了面向对象程序设计的一般步骤、方法和技巧，并给出了程序实例。

本书的突出特点是简明易懂并在写法上采用理例结合的方法。书中以大家容易理解的“学生类”入手，结合编程中常用的数据结构（如堆栈、队列、链表、串等）和最具面向对象特色的图形类（如点、圆、矩形等），循序渐进、全面系统地介绍了面向对象程序设计的基本概念、基本原理、基本方法和实际应用。特别是在第 8 章中，通过精选的两个程序实例，综合介绍了面向对象程序设计的思路、方法和技巧，同时也展示了面向对象程序的特点和风格。另外，本书各章之末还有程序举例、小结和习题，便于读者学习、练习和复习。

本书层次分明、简明实用、理例结合、图文并茂，可作为高等院校计算机及有关专业面向对象程序设计课程的教材，也可作为其他人员自学面向对象程序设计的教材或参考书。对于那些已有 C 语言基础的读者，利用本书对 C++ 则可无师自通。

以本书作为教材，教学时数以 30 学时为宜。

作者

1998 年 5 月

目　录

第 1 章　面向对象程序设计概述

90 年代以来，面向对象程序设计(Object-Oriented Programming, 简称 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。近年来推出的新一代程序设计语言、软件开发工具与环境以及操作系统几乎都是面向对象的。

OOP 为什么会受到如此青睐？OOP 为什么会发展得这样迅速？本章将从整体上对 OOP 的基本概念、主要特点、发展和应用概况等作以简要介绍。

1.1　什么是面向对象程序设计

1.1.1　新的程序设计范型

程序设计范型(Paradigm)是指设计程序的规范、模型和风格，它是一类程序设计语言的基础。一种程序设计范型体现了一类语言的主要特征，这些特征能用以支持应用领域所希望的设计风格。不同的范型有不同的程序设计技术和方法学。

流行最广泛的程序设计范型是过程程序设计范型。基于过程范型的语言称为过程性语言。如 C、FORTRAN、PASCAL、Ada 等都是典型的过程性语言。过程程序设计范型的主要特征是，程序由过程定义和过程调用组成，即程序 = 过程 + 调用。除过程程序设计范型外，还有许多其他程序设计范型。如：模块程序设计范型（典型语言是 Modula）、函数程序设计范型（典型语言是 LISP）、逻辑程序设计范型（典型语言是 PROLOG）、进程程序设计范型、类型程序设计范型、事件程序设计范型、数据流程序设计范型等。

面向对象程序设计(OOP)是一种新的程序设计范型。这种范型的主要特点是：

程序 = 对象 + 消息

面向对象程序的基本元素是对象。面向对象程序的结构特点是，程序一般由类的定义和类的使用两部分组成。程序中的一切操作都是通过向对象发送相应的消息来实现的。对象接收到消息后，启动有关方法完成相应的操作。一个程序中涉及到的类，可以由本程序自己定义（即本程序的设计者自己定义），也可以使用现成的类（包括所用语言系统为用户提供的类库中的类和他人已定义好的类）。面向对象程序设计范型的突出特点之一就是可以使用现成的类，而尽量使用现成的类，也正是面向对象程序设计范型所倡导的程序设计风格。

1.1.2 面向对象程序设计的基本概念

1. 对象

对象(Object)也称客体，在自然语言中其外延十分广泛，可泛指任何事物，包括具体实体（客观世界中的万事万物）和抽象概念（主观世界中的万事万物）。

我们知道，对象总具有一些属性、状态和行为。例如，每一个人都有姓名、性别、年龄、身高、体重、职业等属性，都有工作、学习、吃饭、睡觉等行为。所以，对象一般可表示为：属性（状态）+行为。在面向对象程序设计中，对象被表示成：数据+操作，其中操作又被称为方法。这就是说，面向对象程序设计中的对象是指由一组数据和作用在其上的一组方法组成的实体。

2. 类

程序设计中，往往会涉及很多对象。如果我们把程序中所涉及的所有对象逐个进行描述的话，程序就会变得很冗长，甚至无法进行程序设计。不过，我们总是可以根据属性和行为将对象分门别类，使同类对象具有相似的属性和行为。为此，面向对象程序设计中引入了类的概念。

OOP 中的类是具有某些共性的对象的抽象模型，即一个类就表示一个概念。例如，“学生”就是一个类，它是由千千万万个具体的学生抽象而来的一般概念。同理，书、桌子、教室、计算机、人等等都是类。

类在 OOP 中被表示为由一组数据项和作用在其上的一组操作所构成的整体。“学生类”可由学号、姓名、学分等数据项和对这些数据的录入、修改和查询等操作组成。类中的数据项一般称为实例变量或数据成员，类中的操作一般称为方法或成员函数。

一个类中的实例变量(数据)一般只能由本类中定义的方法(操作)来操纵，外界是不能直接访问的。外界要访问类中的数据，只能通过类中的方法去实现。而且，类只向外界提供方法的界面，而隐蔽其实现细节。所以，类本身就是一种抽象数据类型，它实现了信息隐蔽，是实例变量和方法的封装体。

在 OOP 中，总是先定义类，再用类生成其对象。一个类所生成的对象称为该类的实例（instance）。所以，类也就是其实例的模板。一个类的所有对象都是由类生成的，反过来，类的所有对象都共享类中的全体实例变量和方法。这样，在逻辑上每一个对象都包含数据和操作两部分，但在物理上同类的所有对象都共享类中的实例变量和方法。

3. 消息

消息就是要求对象进行某种活动（操作）的信息。在面向对象程序中，要求某对象作某操作时，就向该对象发送一个相应的消息；当对象接收到发向它的消息时，就调用有关的方法，执行相应的操作。

消息及其传递机制是 OOP 中一个重要特色。在面向对象程序中对象的一切活动，只能通过消息去驱动。消息传递也是对象间进行通讯的唯一方式。

由以上所述可以看出，类和消息传递机制使得对象成了数据和操作的统一体和封装体，对象的状态只能由自身的行为来改变，而外界是无法对它施加任何直接作用的。外界

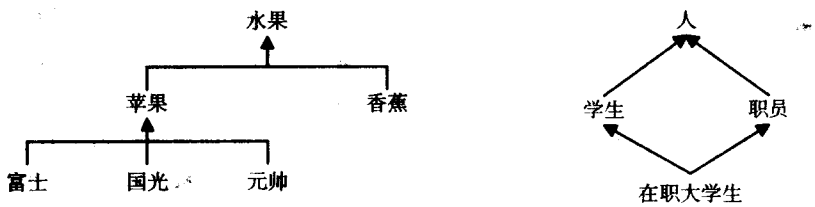
要想改变对象的属性和状态，可通过也只能通过向对象发送消息来间接地实现。所以，对象是一种被封装起来的实体。对象似乎变成了独立自主的东西。

4. 方法

方法是对对象（的属性或状态）的各种操作。方法包括界面和实现两部分。方法的界面也就是消息的模式，它给出了方法的调用协议。方法体则是实现某种操作的具体过程，也就是一段程序。这就是说，消息与方法的关系是：对象根据接收到的消息调用相应的方法；反过来，有了方法，对象才能响应相应的消息。所以，消息模式与方法界面应该是一致的。同时，只要方法界面保持不变，方法体的改动不会影响方法的调用。也就是说，使用方法的程序是不需要修改的。

5. 继承

对类进行考察可以看出，不仅实体对象能够聚成类，抽象的类也还能再进一步聚类成更高层的类，即父类（父类也称为超类或基类）；反之一个类也可以（或可能）再分成一些子类（子类也称为派生类或导出类）。这就是说，一个类向上可以有父类，向下可能有子类。这样，某一范围的事物就可以构成具有层次结构的类集——类树或类网（一般称为类格），例如图 1-1 所示。这种类集一般称为类库。



(a) 树结构　　(b) 格结构

图 1-1　类层次结构

由类与子类的关系可知，父类具有的属性或行为（数据与操作），其子类也一定具有。这就是说，父类把其属性和行为遗传给了子类。反过来，子类也可以从父类那里继承属性和行为。也就是说，凡父类中的数据和操作也都是子类的数据和操作。例如，图 1－1(a)中的苹果可以继承水果的属性（如味甜）。既然如此，在子类的定义中，就不需要再重复父类中已有的数据项和方法，而只需给出子类所独有的数据项（属性）和方法（行为）。

1.2　为什么需要面向对象程序设计

要回答为什么需要面向对象程序设计，让我们先看看当前的软件开发对程序设计有什么需求。

1.2.1 软件开发对程序设计的需求

当前的软件开发对程序设计至少有以下需求：

1. 提高生产能力

众所周知，电子计算机自 1946 年诞生以来，其硬件的生产能力不断提高，而且越来越强，致使硬件更新越来越快、价格却不断下降。但相比之下，软件的生产能力还比较低下，开发周期长、效率低、费用不断上升，以至出现了所谓的“软件危机”。

硬件生产之所以效率高，一个重要原因是，其生产方式已从当初的分立元件一级的设计，发展到今天的芯片（超大规模集成电路）一级的设计。这就是说，硬件有大粒度的构件，而且这些构件有很好的重用性。于是，也就便于实现生产过程的工程化和自动化，生产效率自然也就提高了。

然而，软件的生产中还缺乏大粒度、可重用的构件。因而其生产方式至今基本上还处于手工作业阶段，程序设计基本上还是从语句一级开始。虽然也运用了一些好的程序设计技术（如结构化程序设计），但程序的重用性问题仍然没有很好地解决。从而致使软件生产的工程化和自动化屡屡受阻。诚然，高级语言一般都附有子程序库供用户程序调用，但这些子程序之间无继承关系，因而它们的重用实际是很有限的。用程序语言学中的术语讲，就是子程序的抽象级别太低。

复杂性也是影响软件生产效率的重要方面。软件的复杂度越高，就越难以实现工程化生产。传统程序的特点是数据与其操作相分离，而且对同一数据的操作往往分散在程序的各处。这样，如果一个或多个数据的结构发生了变化，那么这种变化将波及程序的很多部分甚至遍及整个程序，致使许多函数和过程必须重写，严重时会导致整个软件结构的崩溃。这就是说，传统程序的复杂性控制是一个很棘手的问题。随着软件规模的增大，复杂性问题也就日趋严重。而且，这种复杂性也是程序难以重用的一个重要原因。

维护是软件生命期中的最后一个环节，也是非常重要的一个环节。如果一个软件难以维护，则它的生命期就将是短暂的。传统程序那种数据与操作相分离的结构，恰恰不利于程序维护。

总之，要提高软件生产的效率，就必须很好地解决软件的重用性、复杂性和可维护性问题。但这些问题，仅靠传统的程序设计语言是难以解决的。

2. 扩大处理范围

我们知道，当代计算机的应用领域已从数值计算扩展到了人类社会的各个方面，所处理的数据已从简单数字和字符发展为具有多种格式的多媒体数据，如文本、图形、图像、影像、声音等，描述的问题从单纯的计算问题到仿真复杂的自然现象和社会现象。于是，计算机处理的数据量与数据类型激增，程序的规模日益庞大，复杂度不断增加。这些都要求程序设计语言有更大的处理能力。然而，面对这些庞大的数据量和多样的数据格式，传统程序设计方法几乎已无法应付了。

3. 面向新的环境

从程序的运行方式和运行环境看，并行处理、分布式、网络和多机系统等，将是或已经是程序运行的主流方式和主流环境。所以，今后的程序设计必须面向这些新的环境。

这些环境的一个共同的特点是都具有一些有独立处理能力的节点，节点之间有通讯机制，即以消息传递进行联络。显然，要开发能适应这些新环境的软件系统，传统的程序设计技术实在难以胜任。

由上所述我们看到，当前的软件开发迫切需要一种新的程序设计范型的支持。那么，面向对象程序设计是否能担当此任呢？我们再分析一下面向对象程序设计的特点。

1.2.2　面向对象程序设计的优点

由 OOP 的特点可以看出，面向程序设计具有如下优点：

1．可控制程序的复杂性

OOP 把有关数据及其上的所有操作集中在一个个类中，这样，在程序中任何要访问这些数据的地方都只需简单地调用而不需要再重新编码，这就有效地控制了程序的复杂性。

2. 可增强程序的模块性

类是一种抽象数据类型，所以，类作为一个程序模块，要比通常的子程序的独立性强得多。

3. 可提高程序的重用性

类是含有数据和程序的独立模块，它完全可以作为一个大粒度的程序构件，供同类程序直接使用；特别是，父类与子类之间的继承关系，也构成了程序重用的重要方式。

4. 可改善程序的可维护性

由于对对象的操作只能通过消息传递来实现，所以，只要消息模式即对应的方法界面不变，方法体的任何修改不会导致发送消息的程序的修改。这显然对程序的维护带来了方便。另外，类的信息隐蔽和封装机制使得外界对其中数据和子程序（方法）的非法操作成为不可能。这也就大大减少了程序错误率。

5. 能对现实世界的分类系统进行自然的描述

由于 OOP 中的类与现实世界中的类型是一致的，所以，用类来描述现实世界中的类型和分类系统就是很方便、很自然的事情了。用类来直接描述现实世界的类型，可使计算机系统的描述和处理对象从数据扩展到现实世界和思维世界的各种事物，这实际上也就大大扩展了计算机系统的描述和处理范围。

6. 能很好地适应新的硬件环境

面向对象程序设计中的对象、消息传递等思想和机制，与分布式、并行处理、多机系统及网络等硬件环境也恰好相吻合。

由面向对象程序设计的上述优点，我们看到：面向对象程序设计是很有希望能满足当前软件开发的迫切需求。这也就是我们需要面向对象程序设计的理由。事实上，面向对象程序设计正是在软件开发需求的刺激下发展起来的。

1.3 从面向对象程序设计到面向对象技术

1.3.1 OOP、OOA、OOD 和 OODB

正像由结构化程序设计引出了结构化（系统）分析和结构化（系统）设计一样，面向对象程序设计（OOP）也引出了面向对象系统分析（OOA）和面向对象系统设计（OOD）。80 年代，面向对象的程序设计语言趋于成熟，作为一种新的程序设计范型开始为人们所关注，并且为更多的人理解和接受。于是，人们就把程序设计中这一新思想、新方法扩展到整个软件系统开发的全过程中，探索面向对象的系统分析和面向对象的系统设计。就目前的情况来看，虽然 OOA 和 OOD 还没有传统的分析和设计方法那样成熟，但它已表现出了许多独特的优点，如对客观世界描述的自然性、系统开发诸环节的和谐性等，特别是面向对象技术与称为快速原型法的系统开发方法正好相辅相成，简直可以改变传统的软件开发的程序和周期等概念。事实上，一种新的软件开发方法论——面向对象的方法论，已经崭露头角。

将 OOP 与数据库技术相结合，产生了面向对象数据库（OODB）。OODB 与传统的数据库（DB）相比，可以说已经有了质的差别。传统 DB 仅是纯数据的“仓库”，而 OODB 中不仅有数据，还有作用于这些数据上的操作。从结构上看，OODB 的数据模型已是对象模型。从抽象程度来看，OODB 中实际上已存放着知识（因为一个类就是一个概念，例如“学生类”就描述了“学生”这个概念）。

值得一提的是，OOP 更适合于并发程序设计，特别是多机系统、网络和分布式环境下的并发程序设计。在这几种环境下的并发程序中，面向对象技术的对象和消息传递等机制恰好与系统本身的机制不谋而合，从而可相得益彰。

1.3.2 面向对象思想与计算机硬件体系结构

面向对象的思想也影响到计算机硬件的体系结构，现在已在研究能直接支持对象概念的面向对象计算机。这样的计算机将会更适合于面向对象程序设计，更充分地发挥面向对象技术的威力。另外，人们还发现，正在研制与发展中的多机系统、神经网络计算机与面向对象计算机有惊人的相似之处。所以，可以预料，将面向对象技术与神经网络技术、分布式及计算机网络技术相结合，将会制造出新一代计算机硬件系统。

总之，面向对象的思想和方法已从程序设计扩展到整个计算机科学技术领域，甚至更大的范围，如信息科学、系统科学、智能科学、认知科学和工程领域等，而形成了一门面向对象技术，或面向对象方法学。反过来，也可以说，面向对象技术已成为计算机学科中各分支领域的交汇点。

1.3.3 面向对象技术的应用

面向对象技术的应用实际上已十分广泛。仅从计算机学科本身来看，除了软件开发和程序设计以外，面向对象技术至少还可应用于以下方面：

· 操作系统设计——用面向对象技术设计新一代操作系统，特别是分布式操作系统。

事实上，这方面现已取得了不少成果。例如，操作系统 V、Amoeba、Nexus、Eden　Argus、Clouds 等，都是一些著名的面向对象的分布式操作系统。

· 人机交互界面设计——面向对象技术广泛应用于图形用户界面（GUI）、多媒体界面与系统、可视化程序设计等。由于对象的概念很容易描述诸如窗口、图形、图像、图符、按钮等实体，所以面向对象技术在这一领域的应用十分活跃。事实上，面向对象技术已成为人机交互界面设计中的主流技术。

· 系统模拟——面向对象与模拟更有历史渊源。最早的面向对象程序设计语言 Simula 就是为模拟而设计的。离散事件模拟是当前主要的模拟模型，这种模型是事件驱动的，而事件发生的主要标志是消息的到达。系统中持有状态的实体和使状态改变的操作正好可用对象来描述。除了顺序模拟外，并行模拟或分布模拟更要采用面向对象技术。

· 人工智能、知识工程——面向对象技术可广泛用于人工智能、知识工程。如用于知识表示和知识库，得到面向对象知识表示和面向对象知识库。又如面向对象程序设计所具有的模块性和数据隐藏、多态性等特点，允许异构型知识结构和推理机制的无缝集成。还有，对象所具有的良好的通讯能力是实现分布式、多代理智能系统的自然选择。所以，面向对象技术在人工智能、知识工程中也将发挥重要作用。

1.4　面向对象程序设计语言

1.4.1　发展概况

现在公认第一个真正的面向对象程序设计语言是 Smalltalk。它是由美国的 Xerox 公司于 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，开创了面向对象程序设计的新范型。

实际上，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 60 年代开发的 Simula 语言，它的动态联编（聚束）的概念和交互式开发环境的思想则来自于 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 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。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、Mod Pascal 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 SPOOL、Orient 84K 等语言。此外，还有一批面向对象的并发程序设计语言也相继出现，如 ABCL、POOL、PROCOL 等。

当前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 BC++、VC++、VB、Power Builder、Windows 95、Windows NT 等。此外，还有 MIT

的基于 UNIX 的 X-Window 和 APPLE 的 Macintosh，以及现在流行的网上编程语言 JAVA 等。就是传统的过程型、函数型和逻辑型编程语言也都在向面向对象方面靠拢，或者与面向对象语言结合。如 Object Pascal、OBJ2、Vulcan、Ada 等，事实上我们将要学习的 C++就是一种过程与面向对象相结合的语言。

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 基于基本特性的分类

从程序语言学的角度考察，一般认为，面向对象程序设计语言应该有 7 个基本特性：对象、类、继承性、信息隐蔽、强类型化、并发性和持久性。按基本特性对面向对象程序设计语言进行分类，则可分为以下几种类型：

(1) 基于对象的语言；
(2) 基于类的语言；
(3) 面向对象的语言；
(4) 面向对象的数据抽象语言；
(5) 面向对象强类型化语言；
(6) 面向对象强类型化支持并发性和持久性的语言。

这 6 种语言的关系如图 1-2 所示。

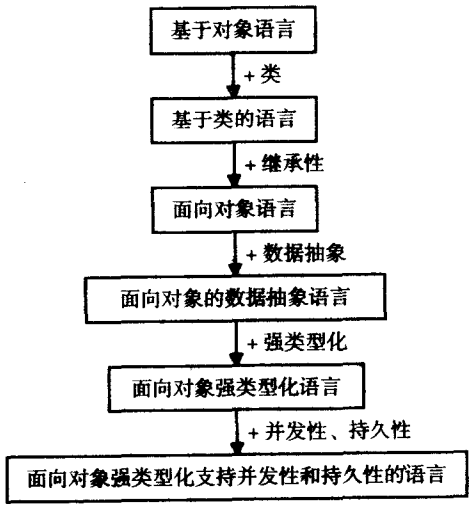


图 1-2 各类面向对象语言的关系

按上述分类，我们下面要着重介绍的 Smalltalk 语言和 C++语言，分别属于上面第四类语言和第五类语言。