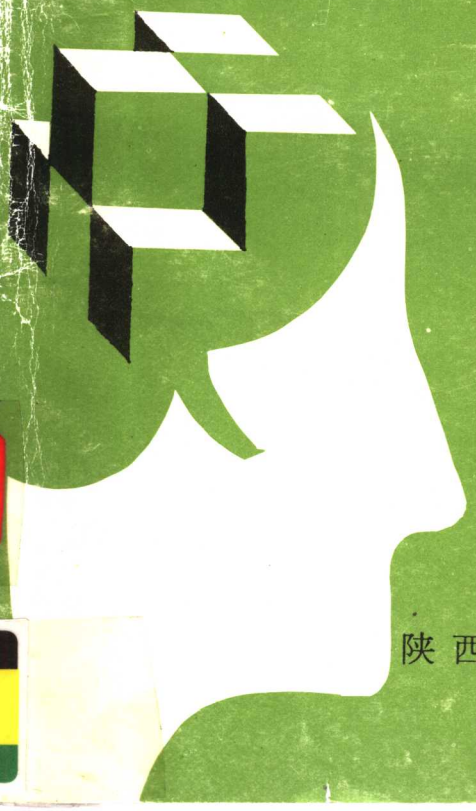
面向对象的方法学与C++语言



王斌君 编著
卢安国
郝克刚 审

陕西师范大学出版社

面向对象的方法学与C++语言

王诚君　卢安国 编著
郝克刚　　审

陕西师范大学出版社

(陕)新登字 008 号

面向对象的方法学与 C++语言

王斌君　卢安国　编著

郝克刚　审

陕西师范大学出版社出版发行
(西安市陕西师大 120 信箱　邮政编码 710062)
新华书店经销　西安市未央区东方彩印厂印刷
开本 787×1168　1/16　印张 16　字数 389 千
1995 年 2 月第 1 版　　1995 年 2 月第 1 次印刷
印数:1—5000
ISBN 7-5613-1214-8/G・915
定　价:12.80 元

前　言

面向对象（Object－Orientation，简称OO）是近几年来在计算机界引用较多的一个重要概念。从80年代后期OO方法被重新认识并取得了较大的进展，用OO方法进行程序设计、软件开发已成为一种时尚。这种方法从根本上改变了人们以往设计软件的思维方式，它力图将问题空间的结构和概念直接映射到软件的解空间，从而使人类认识问题的过程与软件开发的过程保持一致，因此被认为是最有希望解决软件对问题空间的直接表示问题和控制复杂性问题的有效方法，面向对象的方法被称为是一场“软件革命”。

其实，面向对象并非什么新生事物，它最早起源于人们对面向对象语言的研究。早在60年代末，挪威的科学家已成功地开发了用于仿真系统的Simula67语言，该语言首次引入了类、子类等概念，这已形成了面向对象语言的雏形。70年代中后期Xerox公司的PARC（Palo Alto Research Central）研究中心研制成功了Smalltalk语言，以后又相继推出了商用产品Smalltalk80，它已具有面向对象语言的所有特征。然而，Smalltalk仅仅在研究领域和学院的圈子中被人们所重视，并没有被广大的软件人员所接受，除了技术上的原因和开发出的产品效率上的因素外，主要是人们并没有了解面向对象的语言最终能给软件开发带来什么益处。早期从事Smalltalk软件开发的人员经常抱怨他们设计出的软件并非所希望的那样好，其主要原因是还没有形成一套系统的、全面的面向对象的方法论和世界观，并用此贯穿到整个软件开发的全过程，因此，面向对象语言所提供的良好机制并没能充分地发挥其作用。

进入90年代，软件要处理的问题日趋庞大和复杂，原有软件已不足以处理诸如图形、图像、声音等多媒体数据，而且随着计算机的推广和普及，要求软件具有更好的灵活性和易用性，这一矛盾随着硬件性能的高速发展更加紧张，这些都迫切需要一种新软件方法的问世。面向对象正是在这种背景下被人们重新认识并加以完善，这次人们从方法论的高度全面地探讨了软件的开发过程，完善了面向对象的体系，形成了面向对象的软件开发方法学。

面向对象的软件开发（OOSD）是一种采用统一的面向对象的观点，通过面向对象的分析（OOA）、面向对象的设计（OOD）和面向对象的程序设计（OOP）将现实世界的问题空间直接、平滑地过渡到软件空间的软件开发过程，使得软件开发自然、容易，增强了软件的正确性和可维护性。因此，建立面向对象的观点，掌握面向对象的方法就显得非常重要。

C＋＋是一种非常成功的面向对象的程序设计语言，它是在广为流传的C语言的基础上，引入了对象、类、继承性、多态性和模板等面向对象的成份而形成的一种混合型面向对象的程序设计语言。C＋＋与纯面向对象语言相比，它所显示出的高效性使其实用性很强，且由于它包容了C语言的全部特征，保持了C语言的风格，因此容易被C的用户所接受，比较容易推广和应用。

作者认为：只有以面向对象的方法为指导进行面向对象的程序设计，才能充分展示出

面向对象程序设计语言的优越性，而现有介绍C++的教科书大部分都只重视C++语言本身所具有的面向对象的语法特点，而没有从整体上用OO方法作指导，这本书的主要目的就是使读者学会系统地使用OO方法进行面向对象的程序设计。

本书分为三大部分，共九章内容：

第一部分包括二章内容，论述面向对象的方法学。第一章论述面向对象方法的形成、基本概念、机制和原理；第二章结合作者近几年来对面向对象方面潜心研究的心得和体会，提出了一种面向对象的立体模型以及以该对象模型为中心的软件开发过程。

第二部分包括六章内容，论述C++程序设计语言。除了从语法的角度对其进行论述外，将着重体现这些面向对象的语法现象在面向对象的程序设计过程中如何正确表达对象、类、继承、多态等概念。

第三部分包括一章内容，面向对象的软件开发在很大程度上是建立在已有类库的基础上进行的，本部分以C++本身带的流库为例，说明如何分析和使用已有的类库。

本书的初稿已被作为本科生的教科书，使用过程中经过了进一步的完善，书中大量的例子是结合作者的实际工作精心挑选出来的，并在BORLAND C++ 3.1下调试通过。目录中带*号的内容较难，但它确能给我们的设计带来很多益处和启示，第一次阅读时可以跳过，并不影响本书的系统性。

本书第一部分和第二部分由王斌君编写，第三部分由卢安国编写。

本书在写作过程中，得到了葛玮、高岭、党华锐等老师以及刘衡刚、杨卫东、杨扬等研究生的热情帮助，王西民同志对书中的插图做了精心的绘制，在此表示衷心的感谢。另外，本书参考了国内外大量的文献资料，为便于读者进一步阅读，在第二章之后列出了面向对象方法学的参考资料，第九章之后列出了C++语言的参考资料。

由于作者水平有限，并且面向对象方法学和C++语言本身也在不断地发展和完善之中，书中的不足和错误在所难免，诚请广大读者批评指正。

编　者

1994年9月于西北大学

目　录

第一部分　面向对象的方法学

第一章　什么是面向对象

§1.1 为什么要用面向对象 …… (3)
一、人机交互的用户界面与软件的重用 …… (3)
二、多媒体技术与软件的复杂性 …… (5)
§1.2 面向对象的由来及发展 …… (7)
一、面向对象的程序设计语言的形成 …… (7)
二、程序设计语言本身所经历的变革 …… (8)
三、面向对象的方法学的形成 …… (11)
§1.3 面向对象的基本概念、机制和原理 …… (13)
一、对象(object) …… (13)
二、对象类 …… (15)
三、消息 …… (17)
四、继承性 …… (17)
五、多态性 …… (19)
六、动态编联 …… (21)

第二章　面向对象的软件开发

§2.1 面向对象的立体模型 …… (23)
一、对象的信息模型 …… (24)
二、对象的动态模型 …… (29)
三、对象的协作模型 …… (32)
§2.2 面向对象的软件开发过程 …… (35)
一、面向对象的分析(OOA) …… (35)
二、面向对象的设计(OOD) …… (38)
三、面向对象的实现(OOP) …… (39)
参考文献 1 …… (40)

第二部分　面向对象的程序设计语言 C++

第三章　一个更好的 C

§3.1 Hello, World! …… (44)
§3.2 标识符 …… (46)
一、作用域 …… (47)
二、类型 …… (48)
三、内存对象 …… (49)

四、const 类型说明 …… (49)
五、volatile 类型说明 …… (51)
§3.3 引用 …… (51)
§3.4 函数原形 …… (57)
§3.5 内置函数 …… (57)
§3.6 带缺省参数的函数 …… (59)
§3.7 函数名重载 …… (60)
§3.8 new 和 delete …… (62)
第四章 类和对象
§4.1 类和对象的说明 …… (66)
§4.2 构造函数和析构函数 …… (77)
§4.3 类的自引用 …… (84)
§4.4 友元 …… (86)
§4.5 类和数组 …… (90)
一、类对象数组 …… (90)
二、类对象指针数组 …… (90)
三、类对象数据成员数组 …… (91)
四、类数据成员指针数组 …… (91)
五、类成员函数指针数组 …… (92)
§4.6 静态成员 …… (93)
一、静态的数据成员 …… (93)
二、静态的成员函数 …… (94)
§4.7 类的对象成员 …… (95)
§4.8 const 与 volatile 成员函数 …… (98)
§4.9 其它 …… (99)
第五章 运算符函数及重载
§5.1 运算符的重载 …… (103)
一、运算符重载的语法 …… (104)
二、用成员函数重载运算符 …… (105)
三、用友元函数重载运算符 …… (108)
* §5.2 重载++和-- …… (109)
* §5.3 重载= …… (111)
* §5.4 重载[] …… (113)
* §5.5 重载() …… (115)
* §5.6 重载 type …… (116)
* §5.7 重载 new 和 delete …… (118)
第六章 继承性:派生的类
§6.1 单继承 …… (124)
一、派生类的构造函数和析构函数 …… (128)
二、基类的存取限定符 …… (130)

1. 公有派生 …… (130)
2. 私有派生 …… (131)
3. 访问声明 …… (134)
§6.2 多重继承 …… (136)
一、虚基类 …… (139)
二、二义性 …… (142)
§6.3 赋值兼容规则 …… (143)
第七章 多态性:虚函数
§7.1 虚函数 …… (153)
一、函数覆盖 …… (158)
二、空的虚函数 …… (160)
三、抽象类 …… (161)
四、虚析构函数 …… (163)
§7.2 多态性的例子 …… (165)
* §7.3 多态性的进一步讨论 …… (176)
第八章 模板
§8.1 函数模板 …… (185)
一、函数模板的概念 …… (185)
二、重载函数模板 …… (189)
§8.2 类模板 …… (192)
一、类模板的概念 …… (193)
二、类模板的友元 …… (196)
三、模板的例子 …… (199)

第三部分 类库

第九章 流库
* §9.1 流库的类层次结构 …… (208)
一、streambuf 类层次结构 …… (209)
1. 类 streambuf …… (210)
2. 类 filebnf …… (213)
二、ios 类层次结构 …… (215)
1. 类 ios …… (215)
2. 类 istream …… (218)
3. 类 ostream …… (220)
4. 类 iostream …… (222)
5. 类 istream－withassign …… (222)
6. 类 ostream－withassign …… (223)
7. 类 iostream－withassign …… (223)
8. 类 fstreambase …… (224)
9. 类 ifstream …… (224)
10. 类 ofstream …… (226)

11. 类 fstream …… (227)
12. 类 strstreambase …… (228)
13. 类 istrstream …… (228)
14. 类 ostrstream …… (229)
15. 类 strstream …… (231)
§ 9.2 重载操作符>>和<< …… (232)
§ 9.3 操作函数 …… (235)
一、预定义的操作函数 …… (235)
二、用户定义的无参操作函数 …… (237)
三、用户定义的有参操作函数 …… (240)
§ 9.4 对类库的扩充 …… (241)
参考文献 2 …… (245)

第一部分

面向对象的方法学

目前，我们所采用的软件开发的过程与人类认识一个系统的过程不一致；软件的成份和结构不能直接影射问题空间。

计算机的软件一直被以上两大问题所困扰。解决的途径只有靠完善软件本身，使软件空间与问题空间的结构保持一致，以此改善软件的开发过程，使得分析问题、设计问题的解以及解的实现采用统一的模型，即软件开发的过程是一个不断完善、加细模型的过程，而不是模型不断转化的过程。

概括起来，人类在认识世界的过程中，无论是抽象思维还是形象思维都采用从特殊到一般的归纳方法或从一般到特殊的演绎方法。并且，这种认识过程是一个交叉往返、不断重复、逐步深化的过程。人类对抽象思维的研究较为深刻，如形式逻辑、数理逻辑等均已建立了一套较完整的从归纳到演绎的理论和方法体系。而形象思维是在相似基础上进行的，它按照事物之间的相似性进行分类，分类后对每个类再进行详细的解剖和分析，然后再综合优化、形成对世界的进一步的认识。

这就要求我们所建立的软件能够反映人类认识世界的方法学，而这些基本概念和机制正是面向对象的方法学所追求的。面向对象的方法学认为：客观世界是由对象组成的，任何事物都可以是对象，每个对象都有自己的运动规律和内部状态。通过归纳法将系统中具有相同或相似行为的对象抽象为“类”，类是对这些相似行为的对象的统一描述，每个对象都是某个类的实例对象。对象之间相互作用、相互协作构成整个客观世界。类之间也存在着共同的特性，抽象出来称为基类，基类体现了各相关类的一般特征，也可以认为各相关类是通过继承基类的特征并增加新的内容而形成的派生类，类的这种继承层次充分地反映了现实世界中的分类结构。在这个层次结构中，可进一步进行推理，按照演绎的方法发现新的类，从而进一步地完善我们对系统的了解，同时，进一步对对象类进行研究，发现每个类都通过一定的结构来维护其表达的知识和内部的状态，并通过定义一组“方法”来说明其功能。对象外部只能根据其接口中提供的“方法”来访问对象，对象间的联系是通过传递消息相互依赖、相互协作，消息就是通知某个对象完成一定的操作，而该对象是如何完成这一消息要求的细节将被封装在该对象类定义的内部，对外是信息隐蔽的。

综上所述，面向对象的方法学比较自然地模拟了人类认识问题的方法。并且，面向对象的程序设计语言中支持对象、类、继承、多态等概念，所以，对象既可以表示一个抽象的概念，也可以表示一个具体的模块；既可以表示问题域中的概念，也可以表示软件解空

间的概念。由此建立的软件模型与问题结构是一致的，并随着人们对问题认识的深化，使该模型得到逐步完善。

第一章

什么是面向对象

§1.1 为什么要用面向对象

目前的软件开发技术和方法远远地落后于硬件系统的发展水平，软件很难满足应用系统的需求，软件从诞生到现在虽然经历了几次变革，但总的来说，并没有突破程序的执行逻辑，发展是非常缓慢的，这些都孕育着一种全新软件的问世。

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到了不可磨灭的贡献，是现代文明的重要组成部分，是信息革命和产业革命的基础。然而，计算机系统本身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进展，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。然而这几十年来，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，当今的软件开发过程仍然很落后，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，而且非常的困难，软件的生命周期很少能达到十年以上。这一问题的实质在于软件远远落后于硬件的发展水平。目前，软件与硬件之间的差距最少也有两代处理器之多，而且这种差距还呈不断扩大之趋势。

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、人们对软件的要求越来越多且性能越来越高，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。

下面我们以几种典型的软件开发领域为例说明目前软件所存在的问题，它们都表明需要有新的软件及其软件开发方法的支持。

一、人机交互的用户界面与软件的重用

人机交互能力是衡量计算机能力的主要因素之一[12]。

人机交互能力是指人们在使用计算机软件时，人与计算机之间传递信息的交互能力。用户通过用户接口向计算机系统提供数据和命令等各种输入信息，操纵计算机完成预定的计

算、处理、控制等任务。同时，计算机通过用户接口及时地把处理的结果、提示信息、出错信息等显示给用户，这种人机交互活动大量地存在于计算机运行的整个过程之中。

从计算机诞生至今，人们为改善人与计算机的交互能力一直进行着不懈地努力。

计算机发展的初期，是使用机器语言的时代，人们只能通过 0 和 1 这两种数值符号作为媒体承载信息，即用纸带和卡片的有孔和无孔来表示信息，纸带机和卡片机是主要的输入输出设备。0 和 1 组成的代码很不直观、不方便，输入输出的内容很难理解，而且容易出错，出了错也不容易发现。因此，计算机应用只限于极少数从事计算机的专业人员。

50 年代到 70 年代，出现了高级程序设计语言，开始用文字作为信息的载体，人们可以用文字编写源程序，输入计算机，计算机的应用也随之扩大到具有一般文化程度的科技人员。这时的输入输出设备主要是打字机、键盘和显示终端。但使用英文文字同计算机交往，对于人员的文化水平较低、特别是非英语国家，仍然是件困难的事情。80 年代，人们开始致力于研究将声音、图形和图像等作为新的信息媒体输入输出计算机，从而使计算机的应用更加直观、容易。1984 年 Apple 公司的 Macintosh 个人计算机，首先引进了“位映射”的图形机理，用户接口开始使用 Mouse 驱动的窗口、菜单和图符（Window、Icon、Menu and Point device. 简称 WIMP）技术，受到了广大用户的欢迎。这些新技术的出现使得文化水平较低的大众，包括儿童在内都能方便地使用计算机。

自从 60 年代初出现了分时操作系统及交互式终端以来，虽然软、硬件技术得到了迅猛发展，但人们却不知如何利用这些技术来构造一个良好的人一机界面。滥用已有的技术已经成为当时用户界面设计中的一大问题，而与此形成鲜明对照的是，用户界面在软、硬件设计中所占的比例却超过了计算机系统的功能部分，同时，用户界面的开发费用也大大地增加了。据统计，目前采用图形用户界面（GUI）的典型软件系统中，约有 75%的程序代码与用户界面有关，而只有 25%的代码才用于计算。因此，在很大程度上，一个计算机系统的成功与否不仅仅取决于计算机存储容量的大小或处理机速度的快慢，而关键取决于人机界面的设计质量。

人们在开发软件系统时，要将很大的精力用于该软件与用户进行交互作用的设计上，而这种设计包含着很大的重复性，不同的应用程序，用户接口部分的程序逻辑和处理方式是独立于该程序的计算的，它们之间极其相似，特别是同一应用领域的不同开发项目之间，这种接口的设计就更加相似了。让软件开发人员花费大量的时间与精力，去重复开发一些很类似、但又不通用的用户接口，显然存在着巨大的浪费。目前，用户接口管理系统（UIMS）的主要设计任务是如何处理和表示图形的输入、输出，这些工作大多数都基于标准图形包提供的子程序，如：MOTIF、X 窗口、PM 系统、MS—WINDOW 等。然而，即使用它们来设计用户界面中的 WIMP 各成份，仍需要做大量的工作，无法达到完美的重用。

另外，一个成功的用户界面的设计，不仅是计算机专家的任务，而且还要有美学知识以及对用户心理方面的研究等等，只有这样才能设计出美观大方、易学、易用、易维护的用户界面，这些都要求用户界面中的成份是可重用的。

实际上，在计算机软件系统中，重用现象比比皆是，例如：数组、队列、哈希表、栈等在软件系统的设计中是经常使用的软件成份。然而，按软件生命周期进行的软件开发以及采用传统的结构化技术却无法重用这些软件成份，每次只能从头做起。

软件工程追求的目标之一就是提高软件生产率，而软件生产率的提高必须从软件开发

成本和软件维护两方面进行考虑。近几年来，人们对软件自动化进行研究，力图减少开发的费用，然而，由于技术上的原因，进展缓慢；而采用 VDM 等方法来开发软件，力图减少维护的费用，但由于这种方法太形式化，很难推广，受到了很大的限制。目前，人们将这一问题的解决寄希望于软件的重用。

所谓重用是指同一事物不做任何修改或稍作修改就可多次重复使用，如：用旧部件装配一台新的设备，借助以往的知识和经验来解决所谓的新问题等。

重用可分为三个层次：

（1）知识的重用：记载在文件、资料和人们头脑中的知识的重用。如：CAD 系统的总体结构、人机交互方式等。

（2）方法和标准的重用：如：结构化程序设计的方法、自顶向下的软件开发方法等。

（3）软件成份的重用：如：栈、队列等可重用模块等。

这里，我们重点是指第三种情况。软件工作者应尽可能地利用与开发任务有关的、成熟的软件成份库来开发一个新的应用系统。据统计，在开发一个新的软件时，有 40%～60% 的代码是重复以前的工作，有的甚至高达 85%。显然，有效地利用软件重用技术必将提高软件的生产率。

重用的基础是系统能提供可重用的“标准部件”，例如，一个硬件维护人员整天与诸如 8255 芯片、内存条、CPU 等各种集成电路片子打交道，但他并不一定了解每一个芯片是如何制造出来的，也不必要为这些片子出了毛病而担心，因为市场上有的是可供选择的各种集成电路片子。而软件的情况则不同，软件设计者不用关心他们所使用的对象（可重用模块）只是近几年的事，由于种种原因，软件设计者习惯于什么都从头开始。关于算法和数据结构方面的书不少，但能帮助供程序员挑选的诸如栈、队列、二叉树等软件插件却没有。目前，我们所采用的自顶向下软件开发技术，逐步求精的设计结果想期望重用以前的大量工作，显然是困难的，也是有限的。因而，需要有一种新的软件开发技术，使得在设计阶段就能采用一种基于已有成份的自底向上的软件开发，以达到最大限度的重用。

面向对象的方法对产生可重用的软件设计具有极大的优越性，它支持由底向上的软件开发过程。在面向对象的软件开发方法中，软件系统是由对象组成的，而对象则是一个能完整地反映问题域空间中本质的实体。面向对象的程序设计方法使得程序人员摆脱了具体的数据格式与过程，而集中精力去研究所要处理的对象。由于数据抽象、信息隐蔽等机制使得对象的内部实现与外部使用相分离，从而构成了一个很好的可重用的软件成份，即使对象内部做了修改，也不会引起外部（即系统）的变化。另一方面，这种方法在创建和组合可重用软件成份的基础上，也有很大的灵活性，它可通过继承已有对象的性质（数据和操作）产生新的对象。

重用性是解决软件危机、提高软件开发效率和质量、降低成本的重要手段之一。

二、多媒体技术与软件的复杂性

目前，多媒体技术已成为一股世界性的潮流，它使人们能够采用更加丰富多样的手段进行交往，它使人们利用电子通信技术进行相互交流变得无限接近，就如同互相见面那样直接、方便。

简单地说，多媒体是以电脑做为中心，利用文字、图形、图像、动画、声音、视屏等

不同的媒体信息，在不同的界面上组合及流通，使得此媒体信息可在电脑上存取、转换、编辑、同步化等，以达到电脑与使用者双向的交互式及多样化环境。

多媒体技术从1990年刚刚萌牙到现在仅仅三四年的时间，已有了飞速的发展，并具有广阔的应用前景。

多媒体的出现极大地推动了计算机的应用，使其进一步向广度和深度发展，从而使软件变得日趋复杂，这是因为，问题域本身存在着固有的复杂性[2]，所以反映问题域的软件空间的解不可能不复杂。软件的这种复杂性主要来源于它所处理的数据对象的复杂性，由于在程序中无法直接表达这类数据对象，只好用复杂的程序逻辑来完成对这些数据对象处理的要求，使得软件开发过程以及程序的逻辑更加复杂化。图1.1说明了软件所处理的对象这些年来所发生的变化。

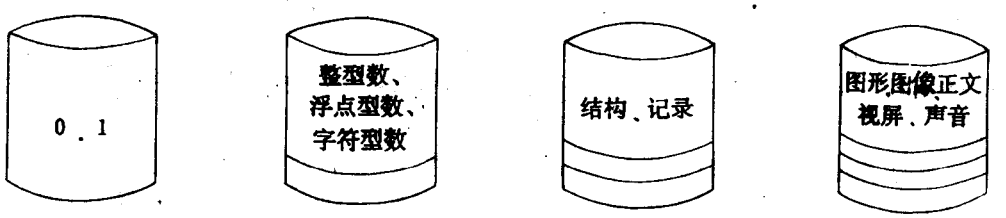


图1.1 数据类型的变化

可见，在传统的程序设计语言中，由系统内部所提供的整型、浮点型等数据类型的表达能力远远不够，并且编程已超过面向记录，而要操作更加复杂的数据类型。许多应用领域，如计算机集成制造系统（CIMS）、计算机辅助设计（CAD）、计算机辅助软件工程（CASE）和计算机辅助出版（CAP）等等都需要仿真真实世界的表示，操作与各种数据类型复杂的相互关系，这些对现今的程序设计和软件系统结构的能力都提出了新的要求。

我们知道，高级语言的出现为我们提供了能够直接表达整数、浮点数、字符串等日常概念的能力，并且在程序设计语言的内部定义了可施加在这些类型上的相关操作，如整型数的值可以是

$$\cdots -2, -1, 0, 1, 2, \cdots$$

可作用在其上的操作有加、减、乘、除等。因此，在程序中能对它们直接进行操作，从而大大简化了程序设计的处理逻辑，使我们能用形如：a+b这样的形式自然、简单地表达对问题的处理。尽管现代的高级程序设计语言中都可用结构去表示诸如记录这样较复杂的概念，但它不是一个抽象的数据类型，我们无法在其上定义必要的操作，也就是说，没有提供对定义的结构进行操作的手段，例如，用两个浮点数定义一个结构来表达复数，但在程序中却无法直接表达对复数进行各种运算的逻辑。

面向对象的程序设计语言为我们描述这种数据的复杂性提供了良好的机制，OO中的类不仅反映了所描述对象的结构(即它所包含的数据)，而且能定义一组作用在其上的操作，使其成为一个真正的类型，从而大大拓展了系统中原有的数据类型，简化了系统的处理逻辑。不仅如此，面向对象的方法通过分类、抽象、封装、信息隐蔽以及对问题空间的直接影射等技术，为我们解决复杂性问题提供了强有力的手段。

人们从软件工程的发展过程中发现，已有的软件是一种非常可贵的资源，应作为一种可重用的成份加以利用，而传统的软件开发的出发点却是从零开始，使得已有的软件没有得到应有的重视；人们从程序设计语言的发展过程中又发现，推动软件发展的重要因素是抽象级别的提高。而传统的软件开发不能有效地、全面地支持这两个概念。所以，采用一种新的软件开发风范就成了一种必然的选择。

另外，以对象为中心的解题方法，采用消息传递机制作为对象之间相互通讯的唯一方式，对象本身具有很强的自治性和独立性。接受消息的对象将按其自己的方法独立地负责并处理所接受到的消息，而与发送消息的对象无关。发送消息的对象可将同一条消息同时发送至多个接受消息的对象中，并允许这些接受同一条消息的对象按照自身的适当方法加以响应，这就使 OO 方法很自然地与分布式并行程序、多机系统、网络通讯等领域取得一致。从而强有力的支持大而复杂系统的开发与研制。

§1.2 面向对象方法的由来及发展

一、面向对象的程序设计语言的形成

面向对象的概念和方法是从面向对象的程序设计语言（OOPL）发展、演变而来的。

60 年代末，由挪威计算机中心的科学家 Ole Dahl 和 Kristren Nyuard 等人设计出来的 Simula 语言是用于仿真的程序设计语言，其中引入的类概念和继承性机制等已初步形成了面向对象语言的雏形，被认为是面向对象语言的先驱。

具有里程碑意义的是 Xerox Palo Alto 研究中心(PARC)在设计一个叫做 Dynabook 的系统时所开发的一种 Smalltalk 语言。从 70 年代初推出了 Smalltalk72 后，又经过了不懈的努力，使其逐渐发展、完善，终于在 1981 年推出了商用化的 Smalltalk80。它已全面地具备了面向对象语言的特征，如：对象、类、继承性、多态性、模板等概念和机制。正是由于 Smalltalk80 的出现，才使得面向对象的方法和原理有了很大的进展。

实际上，许多程序设计语言都对面向对象程序设计语言的形成和发展起到了积极的作用。当今的面向对象语言，从 LISP 人工智能语言中吸收了动态约束的概念、从 Algol 算法语言中引入了重载的概念，而对象的概念是在抽象的数据类型基础上扩充定义的，这得益于 CLU 语言、Modula—2 语言和 Ada 语言等等。这样就形成了如图 1.2 所示的面向对象的语言谱系。

80 年代以后，面向对象的语言如雨后春笋，发展非常迅猛，目前已形成了两类面向对象语言，一类是所谓的纯面向对象的语言，如：Smalltalk、Elffel 等；另一类是混合型的面向对象语言，即在已有的过程语言中增加面向对象的成份而形成的面向对象的语言，如：C++、CLOS（Common Lisp Object System）以及各种面向对象的 Pascal 等。一般来讲，纯面向对象语言着重于方法的研究和快速原型，而混合型面向对象的语言则强调运行速度和使传统的程序员易于接受面向对象的思想。较成熟的面向对象的程序设计语言如 Smalltalk、C++等还提供了强有力的类库和丰富的开发工具集。

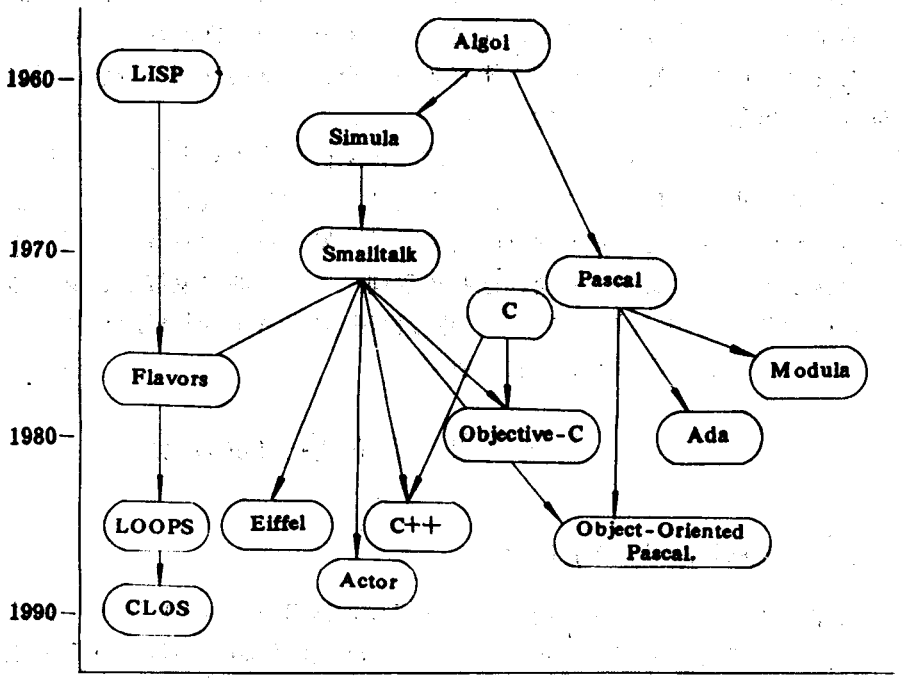


图 1.2 面向对象的语言谱系

可以看出，尽管面向对象的程序设计语言从诞生至今，已有很长的发展时间了，但像 Smalltalk 这样典型的纯面向对象语言也只在研究领域和高等院校受到重视，而它并没有得到推广和应用，这除了面向对象的语言本身在技术方面还不够成熟以外，更重要的是人们没有掌握一套使用这一技术的方法。早期用 Smalltalk 开发软件的人员经常抱怨，用它开发出的软件并没有像 Smalltalk 声称的那样好。这是因为，学习一种新的系统开发方法，要比学习一种新的语言困难得多；学习一种新的系统开发方法，往往需要从根本上改变人们头脑中的思维方式。

二、程序设计语言本身所经历的变革

回顾历史，可以展望未来。

计算机程序设计语言的发展是较缓慢的，尽管它不等于软件的全部，但它在很大程度上决定着软件的水平。

总的来讲，从 40 年代末到 90 年代初，计算机程序设计语言经历了从机器语言到汇编语言，再到高级语言的发展变化过程；经历了一个从面向机器到面向过程、从低级到高级、从简单到复杂的过程；是人们用通用、自然、易于理解的语句，如 a+b，x=a 等方式代替了由 0、1 代码组成的专用于某种计算机的机器语言的过程。尽管高级语言更加通用化，摆脱了原始的机器代码，但它们的编程模式却是相同的，即产生一个动作序列。而近几年逐步发展成熟的面向对象的语言，第一次从根本上改变了半个世纪以来的编程模式，从而带来了一场软件设计领域的革命。