国外计算机科学教材系列

# 程序设计语言：设计与实现（第四版）

# Programming Languages
## Design and Implementation
Fourth Edition

[美] Terrence W. Pratt
Marvin V. Zelkowitz 著

傅育熙 张冬茉 黄林鹏 译

傅育熙 审校

PH PTR

電子工業出版社
Publishing House of Electronics Industry
URL:http://www.phei.com.cn

国外计算机科学教材系列

# 程序设计语言：设计与实现

## （第四版）

Programming Languages

Design and Implementation

Fourth Edition

[美] Terrence W. Pratt
Marvin V. Zelkowitz 著

傅育熙 张冬茉 黄林鹏 译

傅育熙 审校

電子工業出版社

**Publishing House of Electronics Industry**

北京·BEIJING

## 内容简介

本书在一定广度和深度上介绍了程序语言的发展历史和基本概念，讲述了程序语言语法、语义和编译实现之间的关系，介绍了数据类型、顺序控制、子程序、封装、继承等概念以及其实现技术，涉及函数式语言、逻辑式语言、命令式语言和面向对象的语言，包括排版、并行、分布式和网络程序语言等，分析了13种不同语言的编程实例。

本书适合于所有对程序语言感兴趣的读者，可用于作为大专院校计算机系本科生教材或教学参考书。



**图书在版编目(CIP)数据**

程序设计语言：设计与实现(第四版)/(美)普拉特(Pratt, T. W.)著；傅育熙等译.
—北京：电子工业出版社，2001.6
(国外计算机科学教材系列)
书名原文：Programming Languages: Design and Implementation Fourth Edition
ISBN 7-5053-6730-7

Ⅰ.程… Ⅱ.①普… ②傅… Ⅲ.程序语言-教材 Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2001)第041740号

丛 书 名：国外计算机科学教材系列
书　　名：**程序设计语言：设计与实现(第四版)**
原 书 名：Programming Languages: Design and Implementation Fourth Edition
著　　者：[美]Terrence W. Pratt　Marvin V. Zelkowitz
译　　者：傅育熙　张冬茉　黄林鹏
审 校 者：傅育熙
责任编辑：梁卫红　赵宏英
排版制作：电子工业出版社计算机排版室监制
印 刷 者：北京东光印刷厂
出版发行：电子工业出版社　URL：http://www.phei.com.cn
　　北京市海淀区万寿路173信箱　邮编100036
经　　销：各地新华书店
开　　本：787×1092　1/16　印张：29.75　字数：761.6千字
版　　次：2001年6月第1版　2001年6月第1次印刷
书　　号：ISBN 7-5053-6730-7 / TP·3762
定　　价：43.00元
版权贸易合同登记号　图字：01-2000-3472

# 出 版 说 明

随着21世纪的到来，计算机技术的发展更加迅猛，在各行各业的应用更加广泛，越来越多的高等院校增设了有关计算机科学的课程内容，或对现有计算机课程设置进行了适当调整，以紧跟前沿技术。在这个教学体系和学科结构变革的大环境下，对适合不同院系、不同专业、不同层次的教材的需求量与日俱增。此时，如果能够借鉴、学习国外一流大学的先进教学体系，引进具有先进性、实用性和权威性的国外一流大学计算机教材，汲取其精华，必能更好地促进中国高等院校教学的全面改革。

美国Prentice Hall出版公司是享誉世界的高校教材出版商，自1913年成立以来，一直致力于教材的出版，所出版的计算机教材为美国众多大学采用，其中有不少是专业领域中的经典名著，已翻译成多种文字在世界各地的大学中使用，成为全人类的共同财富。许多蜚声世界的教授、学者都是该公司的资深作者，如道格拉斯·科默（Douglas E. Comer）、威廉·斯大林（William Stallings）等。早在1997年，电子工业出版社就从Prentice Hall引进了一套计算机英文版专业教材，并将其翻译出版，同时定名为《国外计算机科学教材系列》（下称：第一轮教材）。截至2000年12月，该系列教材已出版23种，深受读者欢迎，被许多大学选为高年级学生和研究生教材或参考书。

4年过去了，已出版的教材中多数已经有了后续版本。因此，我们开始设计新一轮教材（第二轮教材）的出版，成立了由我国计算机界著名专家和教授组成的“教材出版委员会”，并结合第一轮教材的使用情况和师生反馈意见，组织了第二轮《国外计算机科学教材系列》出版工作。

第二轮教材的出版原则为：

1. 引进Prentice Hall出版公司2000年和2001年推出的新版教材，作为替换版本。
2. 在著名高校教授的建议下，除了从Prentice Hall新选了一些教材之外，还从McGraw-Hill和Addison Wesley Longman等著名专业教材出版社、麻省理工学院出版社和剑桥大学出版社等著名大学出版社引进了一些经典教材，作为增补版本。
3. 对于第一轮中无新版本的优秀教材，我们将其作为沿用版本，直接进入第二轮使用。
4. 对于第一轮中翻译质量较好且无新版本的教材，我们将其进行了修订，也作为沿用版本，进入第二轮使用。

这次推出的教材覆盖学科范围广、领域宽、层次多，既有本科专业课程教材，也有研究生课程教材，以适应不同院系、不同专业、不同层次的师生对教材的需求。广大师生可自由选择和自由组合使用。

按照计划，本轮教材规划出版37种，其中替换版本8种，新增版本14种，沿用版本15种。教材内容涉及的学科方向包括网络与通信、操作系统、计算机组织与结构、算法与数据结构、数据库与信息处理、编程语言、图形图像与多媒体、软件工程等。本轮教材计划于2001年7月前全部出版。教材的使用年限平均为3年。我们还将陆续推出一些教材的参考课件，希望能为授课老师提供帮助。

为了保证本轮教材的选题质量和翻译质量，我们约请了清华大学、北京大学、北京航空航天大学、复旦大学、上海交通大学、南京大学、浙江大学、哈尔滨工业大学、华中科技大学、西安交通

大学、国防科学技术大学、解放军理工大学等著名高校的教授和骨干教师参与了本轮教材的选题、翻译和审校工作。他们中既有讲授同类教材的骨干教师和博士，也有积累了几十年教学经验的教授和博士生导师。

在本轮教材的选题、翻译和编辑加工过程中，为提高教材质量，我们做了大量细致的工作，包括：

1. 对于新选题和新版本进行了全面论证。
2. 对于沿用版本，认真审查了前一版本教材，修改了其中的印刷错误。
3. 对于译者和编辑的选择，达到了专业对口。
4. 对于从英文原书中发现的错误，我们通过与作者联络、从网上下载勘误表等方式，一一做了修改。
5. 对于翻译、审校、编辑、排版、印刷质量进行了严格的审查把关。

通过这些工作，保证了本轮教材的质量较前一轮有明显的提高。相信读者一定能够从字里行间体会到我们的这些努力。

今后，我们将继续加强与各高校教师的密切联系，为广大师生引进更多的国外优秀教材和参考书，为我国计算机科学教学体系与国际教学体系的接轨做出努力。

由于我们对国际计算机科学、我国高校计算机教育的发展存在认识上的不足，在选题、翻译、出版等方面的工作中还有许多有待提高之处，恳请广大师生和读者提出批评和建议。

电子工业出版社
2001 年春

# 教材出版委员会

# 译 者 序

《程序设计语言:设计与实现(第四版)》是一本关于程序设计语言理论和实践的经典著作,是任何一位想从事程序设计语言研究、了解程序设计语言工作机理的计算机科学工作者的必读之书,也是一本程序设计语言发展史和程序设计方法学的优秀教材和教学参考书。

该书从软件开发和计算机体系结构两方面讲述了程序语言的基本概念,从实现效率和程序正确性角度探讨了程序语言的发展方向。

与第三版相比,本书第四版在结构上作了较大的调整,如将程序设计语言范例作为附录处理,并在叙述一些重要的概念时穿插有关的程序语言的历史和基本特性介绍。本书第四版还增加了一些新的内容,如分布式计算、网络程序设计和排版处理语言等。

从国内外不同计算机院系程序语言教学的情况看,基本上可分为两类,有提倡讲授各种不同具体语言的;有认为不需要讲授具体语言,而应开设程序语言概论课程,对于具体语言学生可按需要自行学习的。但不管如何,了解主流程序设计语言的基本概念和实现技术,对于提高软件开发者的素质和理解程序设计语言的发展趋势是有帮助的。

本书第1章、第2章、第4章、第7章、第8章由傅育熙同志翻译;第3章、第5章、第6章由张冬茉同志翻译;第9章、第10章、第11章、第12章和附录由黄林鹏同志翻译。全书最后由傅育熙同志审校。本书的部分内容曾在译者所在院系的本科生课程《程序设计语言概论》和研究生课程《程序设计方法学》中讲授,李渊洁、侯忆铭、易朝阳和肖连兵等同学提出了不少宝贵意见,在此表示谢意。同时还要感谢电子工业出版社的编辑给予的帮助和支持。

由于种种原因,书中错误和不妥之处在所难免,恳请读者批评指正。

译 者

# 前　言

本书是《程序设计语言:设计与实现》的第四版,它沿袭以前各版中以运行各种语言编写的程序所必需的软硬件体系结构为基础的传统,以帮助程序设计人员开发既高效又正确的软件。新版继续采用这一做法,并且加强了有关基础理论和形式模型的内容,这些内容是语言创建的基础。

在计算机界,程序语言设计,即语言的"诞生"、"成熟"和最终的"消亡",仍是一个十分活跃的研究分支。本书第四版将介绍21世纪最重要的各种程序语言。为反映WWW——当前程序设计领域的新热点,本版在上一版的基础上添加了Postscript、Java、HTML等语言。相反,Pascal、FORTRAN和Ada等语言的讨论在这一版中不再作为重点,并且,随着时间的推移这些内容在下一版本中可能不再出现。

在马里兰大学有一门与本书结构一致的课程,已持续教授了25年。这门课通常假定学生已学过C、Java或C++,课程的重点是学习Smalltalk、ML、Prolog和LISP,并就C++语言的实现做进一步的讨论。C++语言的学习可以加深同学对添加了面向对象特性的过程式语言的认识,而对LISP、Prolog和ML语言的讨论则拓宽了同学们在程序设计语言范例的知识面。当然也可以将其中的一到两种语言用FORTRAN、Ada或Pascal替代。

本书假定读者至少熟悉一种过程式语言,如C、C++、Java或FORTRAN。对于那些预备在大学低年级使用本书的院校,或准备使用本书作为讨论语言设计的框架性的必备资料的单位,本书第1章和第2章提供了为理解后续章节所必需的素材,其中第1章是程序设计语言的概括性介绍,第2章则介绍了执行给定程序的基础硬件体系结构。

本书的主题是程序语言的设计与实现,第3章、第5章~第12章构成了本书的基础,其中第3章描述了程序设计语言的语法模型和编译原理,第5章描述了基本数据类型,第6章描述了数据结构和封装,第7章描述了继承,第8章讨论了语句,第9章讨论了子程序调用,第12章则讨论了网络程序设计,所有这些都是语言设计的中心论题。

另外本书第4章涉及语言的语义,包括程序验证、指称语义和lambda(λ)演算。在大学二三年级的课程中可以略去。与本书第三版相比,本书增加了一个附录,概括了13种不同程序语言的细节。

本书包括了1991年ACM/IEEE Computer Society Joint Curriculum Task Force就程序设计语言专题领域推荐的12个知识点[TUCKER et al.,1991]。

编译器的编写曾是计算机学科中的核心课程,但现在越来越多的人认为并非每一个计算机专业的学生都需要具备设计编译器的能力,这项工作应属于编译器的设计专家。去除了这项课程后,可以在课程表中填充如下内容,如软件工程、数据库工程或其他一些关于计算机科学技术实际应用的课程。然而,我们认为编译器的设计技术应是一个优秀的程序设计人员所必须具备一个的背景知识,因此本书的一个重点就是关注各种程序设计语言是如何进行编译的,第3章将提供一个比较完整的语法分析技术的总结。

本书的第12章着重于讨论用FORTRAN、Ada、C、Java、Pascal、ML、LISP、Perl、Posrscript、Prolog、C++和Smalltalk所编写的程序示例,其他的例子以HTML、PL/I、SNOBOL4、

APL、BASIC 和 COBOL 等语言给出。给出大量的用不同语言编写的例子，目的是方便课程指导者从中进行合理的选择。

本书的所有例子，除了那些最简单的外，都在一个合理的翻译器上测试通过。然而，正如 1.3.3 节所指出的那样，在该翻译器上程序可以正确运行并不能保证该翻译器能按照语言标准来处理程序。当然，由于各种原因，一些简单的程序也难免包含错误，在此我们对所有可能出现的问题预先致歉。

综上所述，出版本书第四版的目的如下：

- 提供现代编程语言中一些代表性范例的概览；
- 重点介绍某些具有显著特征的语言，通过该语言编写的程序详细地展现这些特征；
- 详细并且深入地探讨每一种语言的实现，使程序员了解源程序及其实现间的关系；
- 提供足够的形式理论以说明程序设计与计算机科学研究的关系；
- 提供大量的习题和参考书目录使学生有机会拓宽在一些重要问题上的知识。

我们十分感谢本书第三版读者所提出的有益的建议和评论，感谢选修马里兰大学 CMSC 330 课程的数百名同学对于如何改善本书内容的反馈意见。

## 第四版的改动

对于那些熟悉第三版的读者来说，第四版做了下述变更：

1. 增加了关于 WWW 的一章（第 12 章），其中 Java 被作为一种主要的程序设计语言进行讨论，HTML 和 Postscript 语言也被加入本书，使本书偏离了传统的以 FORTRAN 语言数值处理为中心的编译观念。
2. 关于面向对象设计的内容被移到本书的前面以强调其在软件设计过程中的重要性。另外的一些改动是一些次要章节的移动，目的是更好地组织材料使叙述更一致。
3. 我们发现第三版第二部分对语言的详细讨论没有所期望的那么有用。因此将 13 种语言的简短历史添加到相对能体现这些语言主要特性的章节，而将第三版第二部分中语言的概述作为附录处理。除此之外，由于去除了一些过时的内容，本书的篇幅并没有增加。

# 目　录

# 第1章　程序语言设计问题

任何描述算法及数据结构的符号都可以构成一种程序语言，但是本书中我们的兴趣主要集中于那些在计算机上实现的语言。所谓语言的"实现"将在以下两章中加以讨论。本书的其余章节将详细讨论一种语言各组成部分的设计与实现。其目的是研究语言的特征，这些特征独立于任何特定语言，并从广泛的常用语言中给出实例。

本书中，我们将通过十二种具体语言及它们的变种的设计来阐述这些概念的应用；这十二种语言是：Ada、C、C++、FORTRAN、Java、LISP、ML、Pascal、Perl、Postscript、Prolog 和 Smalltalk。另外，我们也将对在本领域有影响的其他语言进行简单的总结。它们包括：APL、BASIC、COBOL、Forth、PL/I 和 SNOBOL4。在正式开始程序语言学习之前，我们来探讨一下为何计算机程序员要进行这一学习。

## 1.1　为什么学习程序语言

现在已经有数百种不同的程序语言被设计出并实现了。甚至早在 1969 年，萨米特[SAMMET 1969]就列举了 120 种广泛使用的语言，之后，又有许多新的语言产生。然而，绝大多数程序员从不冒险使用多种语言，很多人仅使用一两种语言来完成编程。事实上，程序员工作的电脑上安装使用的仅仅是必需的特定语言，如 Java、C、Ada 或 FORTRAN。那么，学习各种各样不同而且有可能永远用不上的语言能使你得到什么呢？

只要你能穿透对语言特征的肤浅认识并深入到设计概念的基础及语言实现的实效上来，那么将有许多理由进行这一学习。总结如下：

1. *增强设计高效算法的能力*　许多语言都有这样的特点：当程序员能正确使用它们时，将十分有益；但如果运用有误，则会浪费大量的机器时间或将程序员引入耗时的逻辑错误中。即使是一位运用某一程序语言已有多年经验的程序员也不能保证他已经理解了该语言的全部特征。递归就是一个典型的例子，当正确地运用这个简单的编程技巧时，它能够使简短而高效的算法直接实现。但如果不能正确使用，就会产生天文数字般增长的执行时间。对那些对递归引起的设计问题及实现上的难度一无所知的程序员而言，他们可能会尽量避开这种有些神秘的结构。但是，如果有了递归的原理及其实现的基本知识，程序员就能理解在特定的语言中使用递归运算所要付出的代价；从而，他就能够决定在特殊编程环境中递归的使用是否可靠。文献中常常介绍一些新的编程方法。要最好地使用面向对象编程、逻辑编程或并行编程之类的概念，就需要理解实现这些概念的语言。
2. *提高对现存程序语言的运用能力*　通过理解一种语言如何实现那些特征，将极大地增强写出高效程序的能力。例如，理解了在我们所使用的语言中数据结构（如数组、串、

表或记录)是如何创建和操作的,知道了递归的实现细节或了解了对象类的构造方法,将会使我们编写出包含这些结构的高效的程序。

3. *增加有用程序结构的词汇表* 在思考时,语言既提供了帮助,但同时又是一种限制。人们用自然语言来表达思想,但自然语言同时也帮助人们组织其思想,从某种程度上说,用不能以语言直接表达的方式进行思考是困难的。同理,单一的程序语言也会有类似的限制。为了找到能够解决问题的数据与程序结构,程序员往往只会考虑那些他非常熟悉而且可以立即表达出来的结构。程序员通过学习广泛的程序语言以及这些语言的结构和这些结构的实现方式,就能增加他的程序"词汇量"。对这些实现技术的理解尤为重要,因为如果要使用在某种程序语言中未能直接提供的结构,程序员需要根据这种语言提供的基本元素来自己实现这些结构。例如,子程序控制结构作为协同程序在许多语言中十分有用,但是极少有语言直接提供协同程序特征。一个 C 或 FORTRAN 程序员可能准备设计一个使用协同程序结构的程序;如果他熟悉协同程序概念及其实现,就能够用 C 或 FORTRAN 程序来实现它们。
4. *可以对程序语言有更好的选择* 了解了各种不同的程序语言,程序员就能够针对特定的项目选择正确的语言,以减少编程量。其中需要大量数学计算的应用程序可以用 C、FORTRAN 或 Ada 语言轻松地设计出。而开发决策领域的应用程序,如人工智能,用 LISP、ML 或 Prolog 语言编程将更轻松些。因特网上的应用更适合用 Perl 和 Java 来设计。如果对每种语言的强项和弱点有所了解,则程序员对编程将有广泛的选择。
5. *使得学习新语言更加容易* 一个语言学家通过对自然语言结构的深刻认识,在学习一门新外语时,往往比连对自己母语结构也几乎一无所知的勤奋的初学者要轻松而且快速得多。类似地,对各种程序语言结构和实现技术有全面认识的程序员在学习一种新的程序语言时也会容易得多。
6. *使得设计出一种新语言更加容易* 极少有程序员曾把自己当做语言设计师,然而任何应用程序也就是程序语言的一种形式。一些大型程序,如文本编辑器、操作系统或一个图形包的用户界面设计者往往必须考虑许多与一般程序语言相同的问题。许多新的程序语言都以 C 或 Pascal 作为实现原型。如果程序员对一般程序语言的各种构造与实现方法都很熟悉,程序设计的方面将会简化。

很明显,学习程序语言远非粗略地浏览它们的特征那么简单。事实上,许多特征的相似性都是假的。同一种特征在两种不同的语言中可能是以截然不同的方式实现的,这样在两个版本中其使用代价会有很大差异。例如,几乎每一种语言均会提供添加操作作为基本实现,但这些添加操作在 C、COBOL 或 Smalltalk 中可能会因其重要性的不同而有所不同。

在本书中,讨论了许多语言结构,同时还讨论了如何在普通计算机上使用一种或多种设计实现这些结构。但是,我们并不试图让读者理解所有的实现途径。要在读者自己的计算机上实现相同的语言或结构,当采用的实现技术不同或计算机硬件基础与这里假定的常规结构不同时,其实现和结构细节可能根本不同。

## 1.2 程序语言简史

自从 20 世纪 50 年代最早的高级语言出现以来,程序语言的设计和实现方法也在不断地