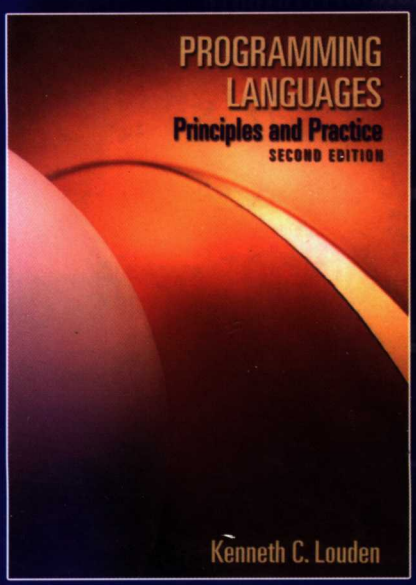
国外计算机科学教材系列

程序设计语言

——原理与实践（第二版）

Programming Languages

Principles and Practice, Second Edition



[美] Kenneth C. Louden 著
黄林鹏 毛宏燕 黄晓琴 等译
黄林鹏 审校

THOMSON

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

国外计算机科学教材系列

程序设计语言

——原理与实践

（第二版）

Programming Languages

Principles and Practice

Second Edition

[美] Kenneth C. Louden 著

黄林鹏 毛宏燕 黄晓琴 等译

黄林鹏 审校

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内容简介

本书介绍了程序设计语言的一般概念，包括程序设计语言的语法和语义，涉及命令式语言、面向对象语言、函数式语言、逻辑式语言和并行语言等多种范例，分析了各种语言的设计原理和内在机制，讨论了语言的理论基础和实现时必须考虑的问题。

本书可用于计算机及其相关专业学生的双语教材，软件与理论专业研究生相关课程的参考书，也可供计算机专业人员参考。

981-240-663-8

版权贸易合同登记号　图字：01-2002-3799

图书在版编目（CIP）数据

程序设计语言——原理与实践：第二版/（美）劳登（Louden, K. C.）著；黄林鹏等译.
-北京：电子工业出版社，2004.4
（国外计算机科学教材系列）
书名原文：Programming Languages: Principles and Practice, Second Edition
ISBN 7-5053-9787-7

Ⅰ.程… Ⅱ.①劳… ②黄… Ⅲ.程序语言-教材 Ⅳ.TP312

中国版本图书馆CIP数据核字（2004）第023476号

责任编辑：李秦华　特约编辑：王崧
印　刷：北京兴华印刷厂
出版发行：电子工业出版社
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：33.75　字数：864千字
印　次：2004年4月第1次印刷
定　价：48.00元

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换；若书店售缺，请与本社发行部联系。联系电话：（010）68279077。质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。

出 版 说 明

21世纪初的5至10年是我国国民经济和社会发展的重要时期，也是信息产业快速发展的关键时期。在我国加入WTO后的今天，培养一支适应国际化竞争的一流IT人才队伍是我国高等教育的重要任务之一。信息科学和技术方面人才的优劣与多寡，是我国面对国际竞争时成败的关键因素。

当前，正值我国高等教育特别是信息科学领域的教育调整、变革的重大时期，为使我国教育体制与国际化接轨，有条件的高等院校正在为某些信息学科和技术课程使用国外优秀教材和优秀原版教材，以使我国在计算机教学上尽快赶上国际先进水平。

电子工业出版社秉承多年来引进国外优秀图书的经验，翻译出版了“国外计算机科学教材系列”丛书，这套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。这些教材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。同时，我们也适当引进了一些优秀英文原版教材，本着翻译版本和英文原版并重的原则，对重点图书既提供英文原版又提供相应的翻译版本。

在图书选题上，我们大都选择国外著名出版公司出版的高校教材，如Pearson Education培生教育出版集团、麦格劳－希尔教育出版集团、麻省理工学院出版社、剑桥大学出版社等。撰写教材的许多作者都是蜚声世界的教授、学者，如道格拉斯·科默（Douglas E. Comer）、威廉·斯托林斯（William Stallings）、哈维·戴特尔（Harvey M. Deitel）、尤利斯·布莱克（Uyless Black）等。

为确保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系列教材的选题、翻译和审校工作。他们中既有讲授同类教材的骨干教师、博士，也有积累了几十年教学经验的老教授和博士生导师。

在该系列教材的选题、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，包括对所选教材进行全面论证；选择编辑时力求达到专业对口；对排版、印制质量进行严格把关。对于英文教材中出现的错误，我们通过与作者联络和网上下载勘误表等方式，逐一进行了修订。

此外，我们还将与国外著名出版公司合作，提供一些教材的教学支持资料，希望能为授课老师提供帮助。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。

电子工业出版社

教材出版委员会

译 者 序

给定一个字符集，字符集上所有字符串的集合的任何一个子集是一种语言。

如果该字符集仅包含一个字符，该字符集上语言的个数就不是可数的。那么哪些是计算机科学感兴趣的语言呢？或者说计算机程序设计语言应该是什么样的呢？

在构造语法分析器的时候有没有想过这样的问题：使用BNF来刻画一种程序设计语言的语法时，实际上它描述的是该语言所有合法字符串的集合，那么该集合又是什么呢？如果不知道该集合是什么，又凭什么来说明语法分析器是正确的呢？

程序设计语言除了语法外，还有两个相关的概念是语言的语义和语用。通常描述语言语义的方法有指称语义、操作语义和公理语义。那么不同的方法有什么区别和适用范围呢？是不是可以互相转换？对于一种语言若使用一种方法刻画，转换后是不是与使用另一种方法刻画的结果相等呢？另外，在构造编译器的时候又如何保证语言经转换后语义不变呢？

除了命令式语言外，为什么还有函数式语言、逻辑式语言和面向对象语言，它们在表达能力和实现上有什么不同？为什么有数千种语言，只有一种语言不是更好吗？难道这是上天的旨意（计算机科学工作者将不同的程序设计语言汇集并记成巴别塔）？所有这一切问题的答案，都可以在本书中找到。

本书作者Kenneth C. Louden是美国圣何塞大学计算机系教授，在计算机程序设计语言和编译原理方面贡献卓著。本书第一版被许多学校选为教材，国内一些程序设计语言的教材也是根据该书第一版的体系编写的。另外本书作者所著的“Compiler Construction: Principles and Practice”一书也是计算机专业的重要参考书，可以和本书一起参照学习。

本书基本涵盖了ACM/IEEE-CS 2001教程所推荐的所有程序设计语言的知识点，包括命令式语言、函数式语言、逻辑式语言、面向对象语言和并行语言，涉及抽象数据类型、类型系统、形式语义等。本书是一本中等程度的程序设计语言原理的书籍，对于扩大知识面，了解不同范例语言的内在原理和实现技术有一定的帮助。

本书可以作为计算机专业高年级程序设计语言概论的教材，也可作为计算机专业研究生程序语言原理的教材。

本书的翻译工作由黄林鹏负责并与一些学生合作。其中第2章、第3章、第7章和第9章由毛宏燕初译，第4章、第6章和第8章由黄晓琴初译，第5章和第13章由谢谨奎初译，第1章和第14章由王佳锃初译，欧佳凡、周蕴凌、严正、赵毅斌、陆奇伟等同学也参与本书的翻译并测试了本书的所有用例和习题。本书最后由黄林鹏修改、整理和定稿。所出现的问题由黄林鹏负责。请将你的意见寄至lphuang@sjtu.edu.cn。

感谢上海交通大学孙永强教授对本书的翻译疑点提供了不少有益的帮助，感谢译者所在单位的领导，他们给译者提供了宽松的工作环境，使本书的翻译能顺利完成。

前　言

本书介绍程序设计语言的一般性原理和现代程序设计语言的具体细节，涉及一些最新的函数式语言和面向对象语言。与许多介绍性的教材不同，本书包括语言实现技术，语言语义基础和大量的习题，可作为编译原理和程序设计语言理论研究的参考资料。作为高年级本科生程序设计语言概论的教材，本书涵盖了2001 ACM/IEEE-CS联合拟定的与程序设计语言相关的知识点以及1978 ACM教程CS8科目的内容。

本书新版本的目标是将详细而精确的语言资料和自本书1993年第一版出版以来程序设计语言在流行趋势和使用上的变化结合在一起，修改并扩充所涵盖的领域，改进并提高例子和习题的实用性，并尽量保留原有的结构。

学生并不需要掌握任何一种特定的语言，然而，具有使用一种语言的经验对于本书的学习是有益的，当然，学生还应该学过数据结构和离散数学等课程。

本书使用的主要语言有C，C++，Java，Ada，ML，Haskell，Scheme和Prolog，其他语言也略有提及。

本书概览和组织

大体上说，虽然不是刻意如此，但每章的内容是相互独立的。书中的交叉引用使读者或教师可以快速参阅由于教学需要被略过的章节。

第1章概述了后面章节要学习的概念，以典型语言和简单例子介绍了不同的语言范例。

第2章和第3章回顾了程序设计语言的历史并介绍了语言设计原理。第3章的内容可以看成是全书的引子，提出的观点在本书其他章节详细讨论。

第4章讨论语法问题，包括BNF，EBNF和语法图。我们将递归定义（如BNF）看成一组等式的解，这种思想将贯穿全章。其中有一节讨论了递归下降语法分析法和语法分析工具的使用。

第5章、第6章和第7章讨论了程序设计语言的核心：声明、分配、计值，作为语义函数的符号表和运行环境、数据类型和类型检查、过程调用和参数传递、异常和异常处理等。

第9章讨论了模块和抽象数据类型，包含语言机制和代数规范等。

第10章、第11章和第12章介绍了不同的语言范例。其中第10章讨论了面向对象语言，这里以Java语言为例讨论面向对象语言的概念，也介绍了C++和Smalltalk等语言的特性。第11章讨了论函数式语言，分别介绍了Scheme语言、ML语言和Haskell语言等。另外还涉及λ演算和递归函数理论。第12章介绍逻辑式程序设计语言，讨论了Prolog和等式语言（如OBJ等）。

第13章介绍程序设计语言的形式语义：操作语义、指称语义和公理语义，涉及不同风格和具体的使用方法。

第14章处理在程序设计语义中引进并行性的主要方式：子例程、线程、信号量、管程和消息传递等，主要以Java和Ada为例。在最后的一节，还介绍了一些最新的在Prolog和LISP中引入并行性的努力。

如何使用本书

10多年来，在圣何塞州立大学，本书一直作为计算机科学专业课程CS 152的教材，该课程面向高年级本科生和研究生。在教学中，笔者设计了两种不同的体系，分别称为“原理”性课程和“范例”性课程。作为一个学期的教学任务，教学内容可按如下方式组织：

- “原理”性课程：讲解第1章、第4章、第5章、第6章、第7章和第8章等章节，如果时间充裕，可以再包括第2章和第3章的内容。
- “范例”性课程：讲解第1章、第10章、第11章、第12章、第13章和第14章等章节，如果时间充裕，可以再包括第2章和第3章以及其他章节。

如果有两个学期的话，建议讲解内容覆盖本书大部分章节。

每章后面习题的一些答案可以在www.brookscole.com或本书作者的网站www.cs.sjsu.edu/faculty/louden上找到。大多数程序设计练习（不是特别长）使用本书讨论的语言编写。概念性的练习则包括对基本概念理解进行考核的简答题和需要思考才能给出答案的论述题。对于困难的题目，笔者将给读者一些启发。翻阅答案还可以得到关于程序设计语言的一些补充知识，将答案看成正文知识的延伸，因此有时还提供超越求解特定问题的知识。一些练习需要读者查阅语言参考手册以了解未在本书讨论的内容。本书将尝试通过添加行号的方法来增加例子的可读性，许多例子还包含了完整的程序，这些程序可以执行并能演示所描述的行为。本书所有例子和其他一些例子（限于篇幅，不在正文中出现），可在www.brookscole.com或本书作者的网站www.cs.sjsu.edu/faculty/louden上找到。除了许多有用的资料外，上述网站还包括许多本书所讨论的语言的免费编译器的下载地址，其中有一些被用来测试本书的例子。

第一版和第二版的差别

第一版使用了一些广为人知的命令式语言如C，Pascal，Ada，Module-2，FORTRAN和一些不那么有名的其他范例的程序设计语言，如Scheme，ML，Miranda，C++，Eiffel，SmallTalk，Prolog等。目前版本最大的变化是原来的Pascal和Module-2被C，C++和Java所替代。对于Module-2，除了第9章关于ADT的历史提及外，基本没有涉及，但本书还是使用了一些Pascal的例子。本书也使用了一些Ada的资料用于介绍那些未在C，C++和Java中出现的特性（如子范围、数组和片断、数据类型的名字等价等）。对于面向对象语言，新的版本使用了Java语言以代替原来的Simula语言，并删去了关于Eiffel语言的章节。在讨论函数式语言的第11章，增加ML和Haskell语言的篇幅，添加了ML语言的例子。最后在讨论并行程序设计语言的第14章，使用Java的线程作为并发的例子。另外一些重要的改进包括：

- 关于控制，本书将过程和环境分开，这样第7章将讨论控制表达式和语句，而新的第8章将讨论控制过程和环境，本书将原来第5章结束处的表达式的语义移到第7章的开始处。这是因为在许多情况下，表达式计值过程中隐藏着一些显式和隐式的控制过程，这可以和其他控制机制放在一起讨论。
- 本书将重载和第5章的符号表放在一起，因为标识符重载的解决基本上是符号表的任务。
- 本书将参数多态和类型检查放在一起，并对Hindley-Milner多态类型检查进行了深入的讨论。参数多态在第9章讨论Ada包和第10章讨论C++模板时将再次出现。

- 本书重写了第9章关于ADT和模块的内容，强调了模块的概念并将其加入标题。这是有挑战性的，因为除了并发机制之外，只有ADT和模块与程序设计语言的其他特性有较大的区别。本章使用ML和Ada作为主要例子，也涉及C++的命名空间和Java的包。这里暂不使用类来讨论ADT和模块概念，类在面向对象的程序设计一章中讨论。
- 除了第2章有过简短介绍外，本书没有提及脚本语言，如，Perl，JavaScript和Tcl等。尽管这类语言在Web应用中的使用日益普及，学生们对它们的兴趣也与日俱增，但笔者认为它们是专用语言。当然，并不反对教师在教学中使用这些语言的例子来说明程序设计语言的特征。
- 本书没有介绍“可视化”语言和组件的概念，如Visual Basic或JavaBean等。笔者的观点是这些“语言”最好是在GUI或软件工程的课程中学习。同样，也只是在语法分析的章节中提到XML，SGML和HTML等语言。

致谢

要感谢的人实在太多，他们在过去的数年间，通过电子邮件传递给笔者本书的评论、建议和修改意见。特别感谢如下人士对本书第二版的建议和评论：亚利桑那州立大学的Leonard M. Faltz，圣伊丽莎白学院的Jesse Yu，墨西哥ITESM的Ariel Ortiz Ramírez，印第安纳州立大学的James J. Ball，Grinnell学院的Samuel A. Rebelsky，以及衣阿华大学的Arthur Fleck。

感谢Tel-Aviv大学的Eran Yahav，他阅读了本书第14章并就并发的内容给予了评论，感谢得克萨斯大学的Hamilton Richards，他阅读了本书第1章和第11章，并就函数式语言部分进行了评论。

感谢圣何塞州立大学选修CS 152课程的全体同学，他们对本书以及本书前一版所做的贡献；感谢我的同事，Michael Beeson，Cay Horstmann以及Vinh Phat，他们阅读并评论了本书第一版的一些章节。另外要感谢的人还有美国水利学院的Ray Fanselau，俄亥俄大学的Larry Irwin，加州理工学院的Zane C. Motteler，伊利诺伊Urbana大学的Tony P. Ng，宾州Shippensburg大学的Rick Ruth和北得克萨斯大学的Ryan Stansifer。

对本书的不足之处和出现的错误由我本人承担，对本书的评论和错误报告请寄至louden@cs.sjsu.edu，非常感谢。

特别感谢Kallie Swanson，Brooks/Cole公司的计算机书籍编辑，她耐心对待本书似乎无尽的修改，并给我鼓励。特别感谢第一个说服我开始本书写作的人——Majorie Schlaikjer。

最后，永远感谢我的妻子Margreth，我的儿子Andrew和Robin，感谢他（她）们对我的爱和支持，没有他们，就没有本书。

目　录

第1章 引 言

本章要点：

- 什么是程序设计语言
- 程序设计语言中的抽象
- 计算范例
- 语言定义
- 语言翻译
- 语言设计

通信方式会影响思维习惯，反过来思维习惯也会影响通信方式。类似地，我们同计算机的交互方式也会影响我们对计算机的认识，反之亦然。在过去几十年里，计算机科学家已经在程序设计语言的设计和使用上积累了大量的经验。尽管还有很多语言设计问题没有被完全理解，但语言设计的基本原理和概念已经成为计算机科学的基本知识。对程序设计者和计算机科学家来说，学习程序设计语言的基本原理就如同学习C语言和Java语言一样重要。没有这些知识，我们将无法深入了解程序设计语言和语言设计是如何影响我们对计算机和计算的认识。

本书的目的是介绍适用于所有程序设计语言的基本原理和概念，而不是拘泥于某一种特别的语言。本书将使用C，Java，C++，Ada，ML，LISP，FORTRAN，Pascal和Prolog等语言作为实例进行说明，读者虽不需要了解所有这些语言，但要求有使用某一种语言的经验，并且知道数据结构、算法和计算的一些基本概念。

在本章中，我们将介绍程序设计语言的一些基本符号和概念，另外，我们也会简短地介绍一下语言的编译器，然而，本书并不准备对编译技术进行详细的讨论。

1.1 什么是程序设计语言

程序设计语言通常被定义为“指挥计算机工作的一种符号记法”。

然而，这个定义并不恰当。20世纪40年代之前，编程就是改变硬件电路的连接方式，即通过设置各种开关来改变计算机内部电路的连接，从而达到控制计算机工作的目的。虽然这可以有效地指挥计算机进行工作，但很难说它是一种程序设计语言。

20世纪40年代，计算机的设计出现了巨大的改进。冯·诺依曼指出不应该通过“开关和线路连接”方式来使计算机完成某种任务，而应该使用一系列存储为数据的指令来控制计算机工作。很快，程序设计者就认识到使用符号来表示指令代码和数据存储地址将带来极大的方便，由此诞生了汇编语言，其指令代码形式为：

```
LDA #2
STA X
```

但汇编语言对机器具有很强的依赖性，其抽象层次低、编程和阅读困难，所以通常不被认为