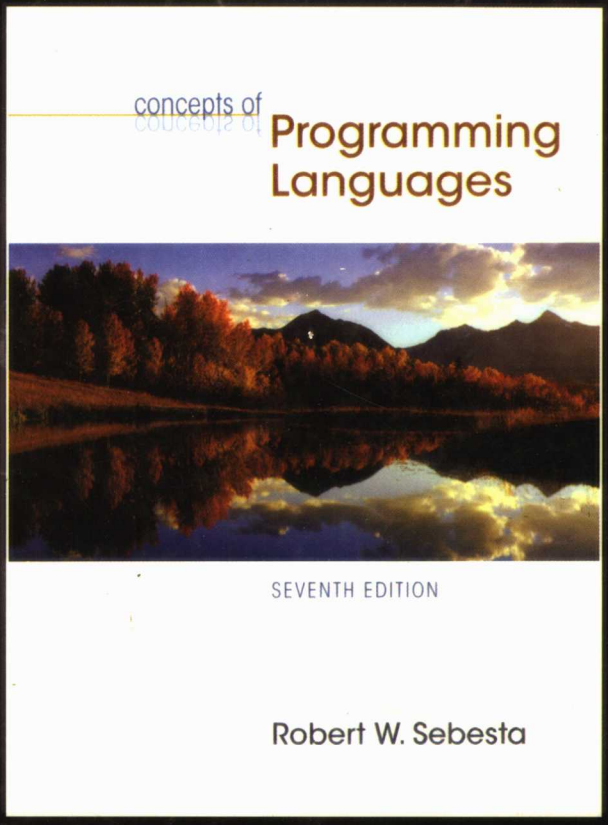
计 算 机 科 学 丛 书

原书第7版

程序设计语言原理

(美) Robert W. Sebesta 著 张勤 王方矩 译



Concepts of Programming Languages

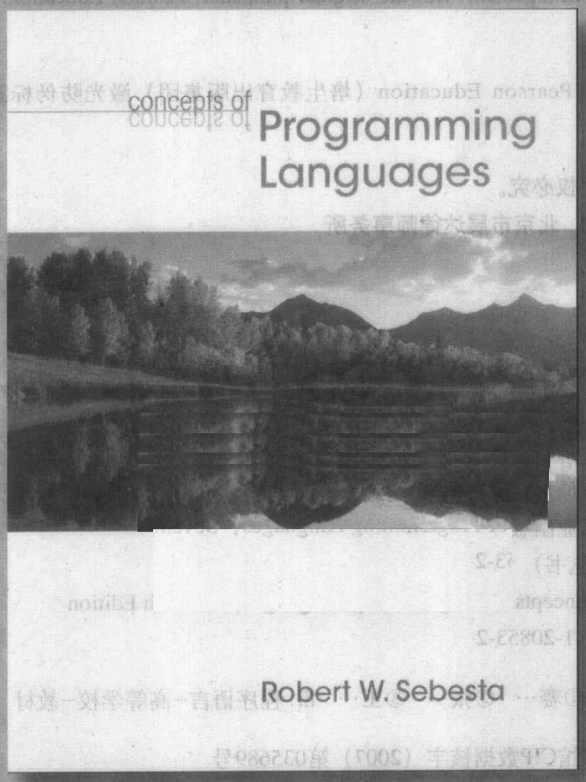
Seventh Edition

机械工业出版社
China Machine Press

计 算 机 科 学 丛

程序设计语言原理

(美) Robert W. Sebesta 著 张勤 王方矩 译



Concepts of Programming Languages

Seventh Edition

机械工业出版社
China Machine Press

本书从为什么学习程序设计语言入手，深入细致地讲解了命令式语言的主要结构及其设计与实现，内容涉及变量、数据类型、表达式和赋值语句、控制语句、子程序、数据抽象机制、支持面向对象程序设计（继承和动态方法绑定）、并发程序单元和异常处理等方面。在最后两章介绍了函数式程序设计语言和逻辑程序设计语言。

本书内容丰富，剖析透彻，被美国和加拿大多所高等院校采用作为教材。本书既可用做高等院校计算机及相关专业本科生程序设计语言课程的教材和参考书，也可供程序设计人员参考。

本书版权登记号：图字：01-2005-4511

图书在版编目（CIP）数据

程序设计语言原理（原书第7版）/（美）赛巴斯塔（Sebesta, R. W.）著；张勤，王方矩译. -北京：机械工业出版社，2007.6

（计算机科学丛书）

书名原文：Concepts of Programming Languages，Seventh Edition

ISBN 978-7-111-20853-2

Ⅰ.程… Ⅱ.①赛… ②张… ③王… Ⅲ.程序语言-高等学校-教材 Ⅳ.TP312

中国版本图书馆CIP数据核字（2007）第035689号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）

责任编辑：王 璐

北京牛山世兴印刷厂印刷·新华书店北京发行所发行

2007年6月第1版第1次印刷

184mm×260mm · 32印张

定价：65.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：（010）68326294

出版者的话

文艺复兴以降，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的传统，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭橥了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短、从业人员较少的现状下，美国等发达国家在其计算机科学发展的几十年间积淀的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章图文信息有限公司较早意识到“出版要为教育服务”。自1998年开始，华章公司就将工作重点放在了遴选、移译国外优秀教材上。经过几年的不懈努力，我们与Prentice Hall，Addison-Wesley，McGraw-Hill，Morgan Kaufmann等世界著名出版公司建立了良好的合作关系，从它们现有的数百种教材中甄选出Tanenbaum，Stroustrup，Kernighan，Jim Gray等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及庋藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力襄助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作，而原书的作者也相当关注其作品在中国的传播，有的还专程为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍，为进一步推广与发展打下了坚实的基础。

随着学科建设的初步完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都步入一个新的阶段。为此，华章公司将加大引进教材的力度，在“华章教育”的总规划之下出版三个系列的计算机教材：除“计算机科学丛书”之外，对影印版的教材，则单独开辟出“经典原版书库”；同时，引进全美通行的教学辅导书“Schaum's Outlines”系列组成“全美经典学习指导系列”。为了保证这三套丛书的权威性，同时也为了更好地为学校和老师们服务，华章公司聘请了中国科学院、北京大学、清华大学、国防科技大学、复旦大学、上海交通大学、南京大学、浙江大学、中国科技大学、哈尔滨工业大学、西安交通大学、中国人民大学、北京航空航天大学、北京邮电大学、中山大学、解放军理工大学、郑州大学、湖北工学院、中国国家信息安全测评认证中心等国内重点大学和科研机构在计算机的各个领域的著名学者组成“专家指导委员会”，为我们提供选题意见和出版监督。

这三套丛书是响应教育部提出的使用外版教材的号召，为国内高校的计算机及相关专业的教学度身订造的。其中许多教材均已为M. I. T.，Stanford，U.C. Berkeley，C. M. U. 等世界名牌大学所采用。不仅涵盖了程序设计、数据结构、操作系统、计算机体系结构、数据库、编译

原理、软件工程、图形学、通信与网络、离散数学等国内大学计算机专业普遍开设的核心课程，而且各具特色——有的出自语言设计者之手、有的历经三十年而不衰、有的已被全世界的几百所高校采用。在这些圆熟通博的名师大作的指引之下，读者必将在计算机科学的宫殿中由登堂而入室。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证，但我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。教材的出版只是我们的后续服务的起点。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

电子邮件：hzjsj@hzbook.com
联系电话：(010) 68995264
联系地址：北京市西城区百万庄南街1号
邮政编码：100037

专家指导委员会

（按姓氏笔画顺序）

尤晋元	王　珊	冯博琴	史忠植	史美林
石教英	吕　建	孙玉芳	吴世忠	吴时霖
张立昂	李伟琴	李师贤	李建中	杨冬青
邵维忠	陆丽娜	陆鑫达	陈向群	周伯生
周克定	周傲英	孟小峰	岳丽华	范　明
郑国梁	施伯乐	钟玉琢	唐世渭	袁崇义
高传善	梅　宏	程　旭	程时端	谢希仁
裘宗燕	戴　葵			

译 者 序

本书是一本在美国、加拿大得到广泛使用的大学教材，适用于计算机科学或计算机工程专业本科二、三年级开设的程序设计语言课程。本书已升级至第7版，多次再版的事实足以说明其受欢迎的程度。一本书的市场价值往往能够反映出它在技术上的价值，本书就是如此。

计算机科学的各个方面都离不开程序设计语言。计算机工作者一生中必然会接触好几种语言：当一种新的语言问世并被广泛接受时，你需要学习这种语言以更新技能；当接手一个新项目时，你必须为这个项目选择一种最合适的实现语言；甚至你可能会为它专门设计并实现一种新的语言。本书并不教授如何使用一种语言，而是讨论程序设计语言的结构与特性、这些结构与特性在不同语言中的设计与实现以及这些结构与特性带给语言的优点与缺点。掌握了这些知识，会让读者在学习新的语言时有一种“似曾相识”的感觉，并很快掌握该语言中的许多特性。当读者需要选择一种语言时，可以根据各种语言的适用性及它们的优缺点做到知“语”善用；当其需要构造一种新语言时，也会知道应该从何处着手来选择出最优的设计实现方案。另外，本书所提供的关于语言“内部”结构的设计与实现的知识，正是大多数介绍语言使用方法的书籍所欠缺的。这种知识能够让读者将一种语言的优点充分地发挥出来，并且避免该语言本身的缺点可能带来的种种问题。

本书的主要内容可分为四部分。第一部分包括第1章和第2章，第二部分包括第3章和第4章，第三部分从第5章到第14章，第四部分包括第15章和第16章。

在第一部分中，第1章介绍一些预备性的基础知识。第2章是对程序设计语言的历史进行有趣而引人入胜的回顾，这一章介绍了一种在第二次世界大战期间由德国开发的功能十分齐全、但鲜为人知的语言——Plankalkül语言，还介绍了Fortran语言的巨大成功及其深远的影响，还包括ALGOL、COBOL、Pascal、Ada、C、C++、Java、C#等语言，所有重要的语言几乎都涉及了，本章为程序设计语言描绘了一个完整的家谱，在这里，各种语言的来龙去脉变成了一幅清晰的画卷。

在第二部分中，第3章是关于语法和语义的描述，语法和语义是语言中的两大要素，这一章讨论对程序设计语言的语法和语义进行形式描述的方法。第4章简明地介绍了程序设计语言的编译原理，这一章是专门为不开设编译技术课程的学校而编写的。然而实际上，即使你打算去啃一本编译技术的大部头书，或者是准备学习一门编译课程，这一章也很值得读一读。它会让你事先获得编译技术的概貌，而不至于一头扎入大量的技术细节当中，导致“不识庐山真面目”。

第三部分集中了本书精华的部分。这一部分详细地剖析语言的各个组成部分设计的实现。程序设计语言的主要组成部分包括变量、类型、表达式、赋值语句、控制结构、子程序、并发、异常处理等。对于语言的每一个组成部分，本书首先是讨论其必要性，接着分析设计中必须解决的问题，然后列出了各种不同的解决方法，并评价了各种方法的优缺点，讨论了一些重要语言中具体的设计实现。在读完这一部分之后，一门语言将不再是一个黑盒子，而像是在硬件高手面前打开了盖子的一台计算机。各个部分的特征、优劣以及对整体性能的影响都变得清清楚楚，这些与语言有关的知识对于编写可靠高效的程序将是极有帮助的。例如，如果一位程序员了解他所使用的语言中调用子程序的代价，他就能根据对程序速度和程序模块化的特定要求来

决定是否使用递归子程序。同样，如果一位程序员熟悉他所使用的语言中的变量引用环境，他的程序中由于变量引用错误而产生的“疑难病症”就会少得多。有时，这些知识还能帮助你判断出编译器内部的错误，使你从盲目地挑剔那些无辜程序的思维中解放出来。

第四部分介绍两种“另类”的语言：以LISP为代表的函数式语言以及逻辑程序设计语言Prolog语言，它们都是人工智能语言，而此前所讨论的语言都属于命令式语言。命令式语言的程序向机器发出一条条可执行的命令语句。然而，在函数式和逻辑程序设计语言中，程序设计却遵循一种完全不同的思维方式，连算法设计都大相径庭。在这些语言里，没有我们熟悉的、必不可少的赋值语句、循环语句等。使用这些语言时，程序设计不再能够先画框图，然后将每一个框图翻译成一条或几条命令语句。函数式程序设计通过函数调用来解决问题，逻辑程序设计则是通过逻辑推理来解决问题。在这两章中，作者举了几个很好的例子，让你很容易转换思维，接受这两种不同的方式；另外还给出了十分巧妙的算法来解决传统的问题，它们会帮助你拓宽解决问题的思路。也许在下一个项目中，这两种程序设计的知识能够使你在解决问题时另辟蹊径，获取出人意料的成功。

本书还收录了与一批著名的计算机科学家进行的有趣而生动的访谈，其中还有C++、Java、PHP、Perl等语言的作者，他们在程序设计语言上的卓越贡献对计算机事业的发展产生了巨大的影响。从这些访谈之中，我们可以了解到这些科学家们丰富的成长经历、独特的思维方式、对问题的深刻见解，以及他们对未来的科学展望。与这些科学家们进行的这些访谈，向读者们打开了另一扇奇妙的窗子，其形式生动活泼、妙趣横生；希望这些内容能够激励读者，尤其是年轻的读者，走上一条成功的道路。

这是一本适用面很广的书，既可用作大学程序设计语言课程的教材，也可用作自学语言的读物，毕业多年的经验丰富的计算机工作者也可以读一读来更新知识。

译 者

2007年1月

前　言

第7版的变化

本书的目的、总体结构以及写作方式与前面的6个版本保持了一致。其主要目的是介绍当代程序设计语言的主要结构，并为读者提供对已有的程序设计语言和未来的程序设计语言进行客观评估所必需的工具；还有一个目的是通过提供对程序设计语言的深入讨论，以及通过表述一种描述语法的形式化方法，为读者学习编译器的设计做准备。

本书第7版采用一些较新语言的素材取代了较老的程序设计语言内容，从而保持内容上的新颖。例如，第1章与第2章中都增加了以XSLT和JSP为例讨论标记与程序设计混合式语言的内容，还增加了使用公理语义进行程序验证的内容，并包括了一种新的验证方式。第4章通过增加一个新的语法分析程序和包括一套使用递归下降算法的完整语法分析，增强了关于递归下降语法分析的内容。本书多处加入了介绍Java 5.0语言版本中最有趣特征的材料，包括该语言版本中新的迭代语句、枚举类以及它的通用性。最后，为了增强本书表述的清晰程度，书中的大部分章节都进行了细微的修正。

全书概貌

本书通过讨论各种语言结构的设计问题，讨论最常用语言中这些结构的设计选择，并客观比较各种设计选择，来描述程序设计语言的基本概念。

要对程序设计语言进行认真研究，需要讨论以下相关课题，其中包括描述程序设计语言的语法和语义的形式方法，该课题被概括进本书的第3章。本书还考虑了各种语言结构的实现技术，第4章讨论词法分析和语法分析，第10章讨论子程序链接的实现。其他一些语言结构的实现问题，也在本书的各个不同部分进行讨论。

下面简略说明第7版包括的内容。

章节概述

第1章从学习程序设计语言的目的开始，讨论用于评估程序设计语言及语言结构的标准，以及影响语言设计的主要因素，设计中常用到的权衡方法及其基本的实现。

第2章概述书中讨论的大部分重要语言的演化过程。虽然没有完整地描述某一种语言，但对于每一种语言的起源、发展和它的贡献都进行了分析。这种历史回顾十分有价值：它为人们理解当代语言设计的实践和理论基础提供了必要的背景，也为进一步学习语言的设计以及评估提供了动力。除此之外，因为书中的其他章节都不依赖第2章，因而可以将这一章作为完全独立的部分来阅读。

第3章讨论描述程序设计语言语法的主要形式方法，即巴科斯-诺尔范式（BNF）。接着是关于属性文法的描述，该文法描述了语言的静态语义及语法。然后讲解语义描述这一有难度的主题，这里包括对三种最常用语义描述方法的简略介绍，即操作语义、公理语义和指称语义。

第4章介绍词法分析和语法分析。这一章是为那些没有设置编译器设计课程的学校准备的。与第2章相似，这一章也是独立的，可以不依赖书中的其他章节独立学习。

第5章到第14章详细地描述命令式语言主要结构的设计问题。对其中每一种结构，作者列举和评估了几种范例语言的设计选择。具体讲，第5章包括变量的多种特性，第6章包括数据类型；而第7章则解释表达式及赋值语句，第8章描述控制语句，第9章和第10章讨论子程序及其实现问题，第11章研究数据抽象的机制，第12章提供了关于支持面向对象程序设计语言特征（继承和动态方法绑定）的深入讨论，第13章讨论并发程序单元，第14章介绍异常处理以及事件处理。

最后两章（第15章和第16章）描述两种最重要的、不同的程序设计范型：函数式程序设计及逻辑程序设计。第15章介绍Scheme语言，包括它的一些基本功能、特殊形式、函数形式，并给出用Scheme语言编写的简单函数示例。接下来，简略地介绍 COMMON LISP、ML和 Haskell语言，以说明不同形式的函数式语言。第16章介绍逻辑程序设计以及逻辑程序设计语言：Prolog。

写给教师

在位于科罗拉多州斯普林的科罗拉多大学的初级程序设计语言课堂上，我们是这样来使用本书的：我们通常会详细地讲授第1章与第3章。由于第2章没有很难的技术内容，我们只花费很少的课时来讲解，而且如我们在前面提到的，后面所有章节的内容都不依赖于第2章，所以这一章的内容完全可以自学完成。此外，我们单独设置编译器设计课程，所以第4章也不在本课程里讲授。

对于具有丰富的C++、Java和C#语言程序设计经验的学生，第5章到第9章是相对容易的。第10章到第14章则有些挑战性，因而需要进行比较详细地讲授。

对大多数的低年级学生而言，第15章和第16章是全新的。理想情况下，应该对那些学习这两章内容的学生提供 Scheme 和 Prolog的语言处理器。书中提供了充足的资料，指导学生写出简单的程序。

本科生的课程中可以不讲解最后两章的全部内容，但在研究生的课程中则可以讨论这两章的所有内容，此时可以跳过前面几章关于命令式语言的内容。

教辅资料

本书的所有读者都可以从网址www.aw.com/cssupport得到下列的教辅资料：

- 一套教学幻灯片，片中把每一章都作为一个独立文件。
- 这里也提供书中所有图片的PowerPoint幻灯片，使读者可以自己动手制作教学课件。
- 为了便于增强课堂教学的效果、配合课程中动手实验、帮助远程教学中学习的学生，我们在网站www.aw.com/sebesta放置了以下资源与大家共享：
  - 提供几种语言的小型学习手册（大约100页的辅导材料）。我们希望借助这套手册，给予学生足够的信息来完成书中章节里有关各种语言的练习，使学生们知道如何在其他的语言中编写程序。在作者提供的相关网站上包括C++、C、Java以及Smalltalk 语言的手册。
  - 实行自我评分。借助于Addison-Wesley软件引擎，学生们可以通过完成多项选择以及填空练习来检测自己对于学完章节的理解程度。

在此声明练习题的答案只提供给教师。请采用本书作为教材的教师按书后的教学支持说明表中提供的联系方式联络Pearson公司北京办事处索取相关教辅资料。

可用的语言处理器

书中讨论的一些程序设计语言的处理器，以及一些关于程序设计语言的信息，能够通过下列网址获得：

C#	microsoft.com
Java	java.sun.com
Haskell	haskell.org
Scheme	www.cs.rice.edu/CS/PLT/packages/drscheme/
Perl	www.perl.com

网站上提供了JavaScript脚本语言、PHP供读者参考。

致谢

许多学校为此书的再版提供了宝贵的支持，在此向这些学校表示感谢：

* Liang Cheng，利哈尔大学
* Amer Diwan，科罗拉多大学
* Nigel Gwee，路易斯安那州立大学
* John V. Harrison，拉斯维加斯内华达大学
* Leon Jololian，新泽西城市大学
* K.N. King，佐治亚州立大学
* Donald Kraft，路易斯安那州立大学
* Simon H. Lin，加利福尼亚州立大学Northridge分校
* Meilu Lu，加利福尼亚州立大学萨克拉门托分校
* Amar Raheja，加利福尼亚科技大学
* Hossein Saiedian，堪萨斯大学
* Raghvinder Sangwan，佩恩州立大学
* Young Park，布雷德利大学
* Steve J. Phelps，加利福尼亚州立大学富勒顿分校
* Yang Wang，西南密苏里州立大学
* Franck Xia，罗拉密苏里大学
* Salih Yurttas，得克萨斯州A & M大学

还有许多其他人也对本书前几个版本提出过宝贵意见，我也向他们致以深深的谢意：Vicki Allan、Henry Bauer、Carter Bays、Manuel E. Bermudez、Peter Brouwer、Margaret Burnett、Paosheng Chang、John Crenshaw、Charles Dana、Barbara Ann Griem、Mary Lou Haag、Eileen Head、Ralph C. Hilzer、Eric Joanis、Hikyoo Koh、Donald Kraft、Jiang B. Liu、Meiliu Lu、Jon Mauney、Bruce R. Maxim、Robert McCoard、Dennis L Mumaugh、Michael G. Murphy、Andrew Oldroyd、Rebecca Parsons、Jeffery Popyack、Steven Rapkin、Hamilton Richard、Tom Sager、Joseph Schell、Sibylle Schupp、Mary Louise Soffa、Neelam Soundarajan、Ryan Stansifer、Steve Stevenson、Virginia Teller、Yang Wang、John M. Weiss和Salih Yurttas。

编辑Matt Goldstein，项目编辑Katherine Harutunian和Addison-Wesley出版社的生产主管Pat Mahtani以及在Argosy的Daniel Rausch和Edalin Michael，我也感谢他们对本书出版所做的努力。

最后，我要感谢我的孩子Jake和Darcie，因为写作本书的第7版本占用了我大量时间，因此冷落了他们，他们对此毫无怨言并一直支持我。

Robert W. Sebesta

目　录

第1章 基本概念

我们在深入学习程序设计语言原理之前，需要考虑几个基本的概念。首先我们将讲解，为什么计算机科学专业的学生以及专业软件开发人员需要学习语言的设计和评估的一般原理。这种讨论对于那些认为只要有一、两种程序设计语言的工作经验就足够了的计算机科学人员来说，是十分有价值的。接下来，我们将简略描述程序设计的主要范畴。然后，由于本书将评价各种程序设计语言的构成和特性，我们将会列出用于判别程序设计语言优劣的标准。紧接着，我们将讨论两个影响程序设计语言的重要因素，即计算机体系结构以及程序设计方法学。之后，我们将给出程序设计语言的不同分类。再接着，我们将阐述在语言设计中必须考虑的几种主要权衡取舍方法。

由于本书也叙述程序设计语言的实现，因而我们在这一章中概括了最常用的程序设计语言实现的方法。最后，我们将简略地描述几个程序设计环境的例子并且讨论这些环境因素对于软件产品的影响。

1.1 学习程序设计语言原理的缘由

学生们会很自然地问，他们如何能够从程序设计语言原理的学习中得益。毕竟在计算机科学领域里，还有其他大量的题材值得花费时间认真学习。下面，是我们认为计算机科学人员通过学习程序设计语言原理能够获得的种种益处。

- **增进表达思想的能力。**人们普遍认为，我们思维的深度受我们用来进行思想交流的语言以及这种语言的表达能力的影响。对自然语言而言，那些只掌握有限语言的人，其思维的复杂程度，特别是其抽象思维的深度，必然受到局限。换言之，人们对不能口头或笔头描述的事物结构必定很难将其概念化。程序员在开发软件的过程中也同样受制于这个法则，用来开发软件的程序设计语言制约着程序员能够应用的控制结构、数据结构和数据抽象的类型，因而也制约着他们所能够设计的算法。

了解多种程序设计语言的特性，能够使程序员在软件开发的过程中减少这种受限性。程序员可以通过学习新的程序设计语言的结构来拓宽软件开发的思路。

有人或许会提出不同意见，学习其他程序设计语言所具备的功能，并不能帮助一个正用某种不具备这些功能的程序设计语言工作的编程人员。然而，这种论点事实上不成立。因为一些程序设计语言的结构，常常可以使用其他不直接支持这些结构的语言来构造。例如，一个学会了Perl（Wall et al.，2000）语言中关联数组结构及使用的C程序员，就可能用C 设计出模拟关联数组的结构来。

学习程序设计语言的原理，将培养评价语言特性的能力，并将鼓励程序人员运用这些语言特性。

许多的语言特性可以在其他不具备这些特性的语言中构造出来的事实，丝毫不能削弱设计具有一系列最佳特性语言的重要性。一种已经嵌入某种语言的特性，总是优于对该特性的仿制，仿制的特性往往不够优雅、方便，而且在一种并不支持该特性的语言中也不够安全。

- **增强选择适用语言的能力。**许多专业软件开发人员只受过有限的正规计算机科学教育，

往往是通过自学，或者通过公司或单位的培训课程。这样的培训往往只教会他们一种或者是两种与当前工作项目直接相关的语言。另外的许多程序人员是在很早以前受过正规的训练。他们当时学习的语言已经不再使用，而许多当今在程序设计语言中已经广泛使用的特性那时还没有出现。这样就导致许多程序员当需要为新的项目选择程序设计语言时，他们宁可继续使用自己最为熟悉的那种语言，尽管这种语言已经很不适宜于新的项目。如果这些程序员熟悉其他语言，尤其是熟悉这些语言中的特殊特性，他们便能够更好地选择更适当的语言。

- **增强学习新语言的能力。**计算机程序设计仍旧是一个相对年轻的学科，其设计方法学、软件开发工具，以及程序设计语言，都还处于不断演变的状态之中。这使得软件开发成为一种激动人心的专业，但这同时也意味着不断学习成为关键所在。学习一种新的程序设计语言可能是一个漫长而艰难的过程。这对于那些只习惯于一种或者两种语言，并且从未认真钻研过程序设计语言基本原理的人来说尤其如此。一旦透彻地掌握了程序设计语言的基本概念，就不难理解这些基本原理是如何地融入我们所学习的语言的设计之中。

例如，理解面向对象程序设计原理的程序人员，较之从未运用这些原理的程序人员，能够更为轻松地学习Java 语言（Arnold et al.，2000）。人们在使用自然语言的时候也会发生同样的现象。你对母语的语法理解得越好，你会发现学习第二种语言越容易。更为甚之，学习第二种语言还能够帮助你更好地理解母语。

3 最后，对于进行实际工作的程序设计人员来说，关键是通晓程序设计语言的术语以及基本概念。这样，就能阅读和理解程序设计语言手册，以及语言和编译器的说明书和其他文件。

- **更好地理解实现的重要性。**在学习程序设计语言原理时，影响这些原理的实现方法是十分有趣也是十分必要的课题。常常，人们对程序设计语言实现方法的理解，会使得他们容易理解一种语言为什么要这样设计。而这种理解能够进一步催生更加明智地、按照设计目的来运用语言的能力。通过理解程序设计语言组成结构的种种选择，并且明了这些选择所拥有的结果，我们就能成长为更加优秀的程序设计人员。

往往只有懂得有关的程序设计语言实现细节的程序员才可能发现和改正某些类型的程序错误。理解语言实现问题的另一个好处是，它会使我们明了计算机是怎样执行各种程序设计语言结构的。这种能力进一步帮助我们理解在众多的结构中，应该选择相对高效的一种来编写程序。例如，不懂得递归调用是如何实现的程序员，常常不知道递归算法比等价的迭代算法要慢得多。

- **促进整个计算科学的发展。**最后，从计算科学的整体上来看，可以证明学习程序设计语言原理的必要性。虽然我们通常可以确定某种程序设计语言得以广泛应用的原因，但许多人在回顾中都认为，最广泛流行的语言在当时并不总是最好的语言。在某些情况下一种语言被广泛使用，部分原因是由于选择语言的决策人员对程序设计语言的一般原理不够熟悉。

例如，许多人认为，20世纪60年代初期，如果ALGOL 60（Backus et al.，1962）取代了Fortran (INCITS/ISO/IEC, 1997) 的话，情形会好得多，因为ALGOL 60语言更优雅，它的控制语句也比Fortran 的好得多。之所以没能如此，是由于当时许多程序员以及软件开发的管理人员没有清晰理解ALGOL 60的设计原理。他们感觉ALGOL 60的说明书太难读（的确如此），又更难懂。他们不能够欣赏它的模块结构、递归结构以及出色的控制语句，因而不能体会到ALGOL 60 超越Fortran的好处。

当然，如我们将在第2章里介绍的，还有其他许多的原因也使得ALGOL 60没能为人们所接受。然而可以肯定的是，当时的开发人员没有领悟到这种语言的种种优点，这起了决定性

的作用。

一般而言，如果选择语言的决策人员具有全面的知识，会选择好的语言取代差的语言。 4

1.2 程序设计应用领域

今天计算机已经大量地应用于现代社会的各个领域，从控制核能发电厂到提供移动电话中的电子游戏。由于计算机应用领域的千差万别，人们开发了用于不同目的的程序设计语言。在这一节中，我们将简略地讨论几个计算机应用领域，以及与其相关的程序设计语言。

1.2.1 科学应用

20世纪40年代出现的第一台数字计算机曾经用于科学应用，并且事实上是为科学应用而发明的。科学计算的特点是，数据结构简单，但具有大量浮点数的算术运算。其中最常用的数据结构是数组和矩阵；而最常用的控制结构是计数循环和选择。早期为科学计算而发明的高级语言就是为了满足这种需要而设计的。那个时候高级语言的对手是汇编语言，因而高效率是其设计的基本要点。第一种科学应用语言是Fortran。ALGOL 60及其绝大多数的后代语言，尽管也是为其他相关的用途而设计的，但主要是以科学计算为主要目的。对于一些以计算效率为基本要点的科学应用而言，如在20世纪50年代和60年代较为普遍的那些科学应用，后来的程序设计语言没有哪一种在效率方面明显优于Fortran。

1.2.2 商务应用

计算机在商务上的应用始于20世纪50年代。人们为此开发了专门用途的计算机，随之以特定的程序设计语言。第一种成功的高级商务语言是初版出现于1960年的COBOL （ISO/IEC，2002）。至今这种语言仍然是这一应用中最为广泛使用的程序设计语言。商务语言的特征有：帮助产生详尽报表的机制，能以精确的方式描述与存储十进制数以及字符数据，以及能够进行十进制算术运算的能力。

随着个人微型计算机的出现，产生了商务尤其是小型商务运用计算机的新方式。两种可以用于微型计算机的特殊工具，电子表格系统和数据库系统，是为商务应用而开发，而当今已被广泛用于家庭及商务。

商务应用语言中，除了COBOL的发展与演化以外，其他开发十分有限。因而本书仅仅包括了对于COBOL语言结构的有限讨论。 5

1.2.3 人工智能

人工智能（AI）是计算机应用的一个广泛领域，它的特征是使用符号运算而非数值运算。符号运算指的是，我们处理的是由名字而非数字组成的符号。对于符号运算，采用数据链表结构比用数组结构更为方便。这种类型的程序设计，有时需要比其他程序设计领域拥有更大的灵活性。例如在一些人工智能的应用中，能够在程序运行的过程中创立并执行程序段的能力将带来方便。

第一种为人工智能应用而开发并被广泛运用的程序设计语言是产生于1959年的函数式语言LISP（McCarthy et al.，1965）。大多数1990年以前的人工智能应用程序，都是用LISP 语言或者与其相近的一种程序设计语言编写的。然而在20世纪70年代的初期，出现了实施于一些人工智能应用的另外一种方法，即使用Prolog（Clocksin and Mellish，2003）语言的逻辑程序设计。更