<<程序设计语言概念>>

图书基本信息

书名：<<程序设计语言概念>>

13位ISBN编号：9787302229568

10位ISBN编号：7302229562

出版时间：2011-1

出版时间：清华大学出版社

作者：Robert W.Sebesta

页数：563

译者：徐明星,邬晓钧

前言

程序设计语言是计算机教学的基础课程。
大部分的高等院校程序设计语言教材旨在讲解语法形式，讲解如何写出让编译器接受的语句，以及这些语句是如何被编译器理解的。
学生很容易将程序设计语言当作一门特殊的外语来学习，将编程当作是句型练习，编程过程就是语言翻译的过程一即将头脑中的“ 内部言语 ”翻译成符合某种程序设计语言语法的语句。
当面对以练习语法为主要目的习题时，这样的翻译过程容易完成，而当面对复杂的综合性习题或者要编程实现算法时，学生往往就会觉得难以下手。
导致这种情况的根本原因是学生的思维没有上升到计算机解题的思维。
程序设计语言是一种人造语言，它与汉语、英语、法语等自然语言存在着很大的不同。
程序设计语言是在一定的设计思想指导下，权衡多种因素，精心为计算机定义的。
程序设计语言的语法和语义只是外在的表现，程序设计语言的本质是解决应用领域问题的方法与工具。
语言既是思维的表达，也是思维的工具。
如果局限于程序设计语言定义的语法和语义，不能运用语言工具来辅助自己的思维，必然会限制用程序设计语言来解决实际问题的能力。
本书并不是介绍某一种具体的程序设计语言所蕴涵的计算机解题原理，而是直接解释说明一般性原理，并介绍这些原理在不同程序设计语言中的实现，比较不同语言实现的优劣。
这是一种从本质到现象的论述方式，它使读者更关注于程序设计语言原理本身，为今后深入理解和掌握具体的程序设计语言，选择更合适的程序设计语言来解决具体问题，乃至于创造实现新的程序设计语言都打下扎实的基础。

<<程序设计语言概念>>

内容概要

《世界著名计算机教材精选：程序设计语言概念（第9版）》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）、并发和异常处理等方面。
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。
《世界著名计算机教材精选：程序设计语言概念（第9版）》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。
《世界著名计算机教材精选：程序设计语言概念（第9版）》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。

作者简介

作者：（美国）塞巴斯塔（Robert W.Sebesta） 译者：徐明星 邬晓钧

<<程序设计语言概念>>

书籍目录

第1章 预备知识1.1 学习程序设计语言原理的原因1.2 程序设计领域1.2.1 科学应用1.2.2 商务应用1.2.3 人工智能1.2.4 系统程序设计1.2.5 网络软件1.3 语言评价标准1.3.1 可读性1.3.2 可写性1.3.3 可靠性1.3.4 代价1.4 影响语言设计的因素1.4.1 计算机体系结构1.4.2 程序设计方法学1.5 程序设计语言的分类1.6 语言设计中的权衡1.7 实现方法1.7.1 编译1.7.2 完全解释1.7.3 混合实现系统1.7.4 预处理器1.8 编程环境小结复习题习题第2章 主要程序设计语言的发展2.1 Zuse的Plankalkul语言2.1.1 历史背景2.1.2 语言概述2.2 最少硬件的程序设计：伪代码2.2.1 Short Code语言2.2.2 Speedcoding系统2.2.3 UNIVAC“编译系统”2.2.4.相关工作2.3 IBM704.计算机与Fortran语言2.3.1 历史背景2.3.2 设计过程2.3.3 Fortran I概述2.3.4 Fortran II2.3.5 Fortran IV、77、90、95和20032.3.6 评价2.4 函数式程序设计：LISP语言2.4.1 人工智能的起源和表处理2.4.2 LISP语言的设计过程2.4.3 语言概述2.4.4 评价2.4.5 LISP的两种后代语言2.4.6 相关语言2.5 迈向成熟的第一步：ALGOL602.5.1 历史背景2.5.2 早期设计过程2.5.3 ALGOL58概述2.5.4 对ALGOL58报告的响应2.5.5 ALGOL60的设计过程2.5.6 ALGOL60概述2.5.7 评价2.6 商务记录计算机化：COBOL语言2.6.1 历史背景2.6.2 FLOW-MATIC语言2.6.3 COBOL语言的设计过程2.6.4 评价2.7 分时处理的开始：BASIC语言2.7.1 设计过程2.7.2 语言概述2.7.3 评价2.8 满足所有人的需要：PL／I2.8.1 历史背景2.8.2 设计过程2.8.3 语言概述2.8.4 评价2.9 两种早期的动态语言：APL和SNOBOL2.9.1 APL语言的起源与特点2.9.2 SNOBOL语言的起源与特点2.10 数据抽象的开始SIMULA672.10.1 设计过程2.10.2 语言概述2.11 正交设计：ALGOL682.11.1 设计过程2.11.2 语言概述2.11.3 评价2.12 ALGOL系列语言的早期后代语言2.12.1 为简单性而设计：Pascal语言2.12.2 可移植的系统语言：C语言2.13 基于逻辑的程序设计：Prolog语言2.13.1 设计过程2.13.2 语言概述2.13.3 评价2.14 历史上规模最大的设计工作：Ada语言2.14.1 历史背景2.14.2 设计过程2.14.3 语言概述2.14.4 评价2.14.5 Ada952.15 面向对象的程序设计：Smalltalk2.15.1 设计过程2.15.2 语言概述2.15.3 评价2.16 结合命令式和面向对象的特性：C++2.16.1 设计过程2.16.2 语言概述2.16.3 评价2.16.4 一种相关语言：Eiffe2.16.5 另一种相关语言：Delphi2.17 基于命令式的面向对象语言：Java2.17.1 设计过程2.17.2 语言概述2.17.3 评价2.18 脚本语言2.18.1 Perl的起源与特点2.18.2 JavaScript的起源与特点2.18.3 PHP的起源与特点2.18.4 Python的起源与特点2.18.5 Ruby的起源与特点2.18.6 Lua的起源与特点2.19 一种新千年的基于C的语言：C#2.19.1 设计过程2.19.2 语言概述2.19.3 评价2.20 标记与程序设计混合的语言2.20.1 XSLT2.20.2.ISP小结文献注释复习题习题程序设计练习访谈：用户设计与语言设计第3章 描述语法和语义3.1 概述3.2 描述语法的普遍问题3.2.1 语言识别器3.2.2 语言生成器3.3 描述语法的形式化方法3.3.1 巴科斯.诺尔范式和上下文无关文法3.3.2 扩展的BNF3.3.3 文法与识别器3.4 属性文法3.4.1 静态语义3.4.2 基本概念3.4.3 属性文法定义3.4.4 本质属性3.4.5 属性文法的例子3.4.6 计算属性值3.4.7 评价3.5 描述程序的意义动态语义3.5.1 操作语义3.5.2 指称语义3.5.3 公理语义小结文献注释复习题习题第4章 词法分析和语法分析4.1 概述4.2 词法分析4.3 语法分析问题4.3.1 语法分析概述4.3.2 自项向下的语法分析器4.3.3 自底向上的语法分析器4.3.4 语法分析的复杂度4.4 递归下降的语法分析4.4.1 递归下降的语法分析过程4.4.2 LL文法类4.5 自底向上的语法分析4.5.1 自底向上语法分析器的分析问题4.5.2 移进-归约算法4.5.3 LR语法分析器小结复习题习题程序设计练习第5章 名字、绑定和作用域5.1 引言5.2 名字5.2.1 设计问题5.2.2 名字形式5.2.3 特殊字5.3 变量5.3.1 名字5.3.2 地址5.3.3 类型5.3.4 数值5.4 绑定的概念5.4.1 属性与变量绑定5.4.2 绑定类型5.4.3 存储绑定和生存期5.5 作用域5.5.1 静态作用域5.5.2 块5.5.3 声明的次序5.5.4 全局作用域5.5.5 静态作用域评估5.5.6 动态作用域5.5.7 动态作用域评估5.6 作用域和生存期5.7 引用环境5.8 命名常量小结复习题问题集编程题访谈：脚本语言以及其他灵活解决方案的例子第6章 数据类型6.1 引言6.2 基本数据类型6.2.1 数值类型6.2.2 布尔类型6.2.3 字符类型6.3 字符串类型6.3.1 设计问题6.3.2 字符串及其操作6.3.3 字符串长度的设计选项6.3.4 评估6.3.5 字符串类型的实现6.4 用户定义的序数类型6.4.1 枚举类型6.4.2 子界类型6.4.3 用户定义的有序类型的实现6.5 数组类型6.5.1 设计问题6.5.2 数组和索引6.5.3 下标的绑定和数组的种类6.5.4 数组初始化6.5.5 数组操作6.5.6 矩形数组和不规则数组6.5.7 切片6.5.8 评估6.5.9 数组类型的实现6.6 关联数组6.6.1 结构和操作6.6.2 关联数组的实现6.7 记录类型6.7.1 记录的定义6.7.2 纪录域引用6.7.3 记录操作6.7.4 评估6.7.5 录类型的实现6.8 联合类型6.8.1 设计问题6.8.2 判别式联合与自由联合6.8.3 Ada的联合类型6.8.4 评估6.8.5 联合类型的实现6.9 指针和引用类型6.9.1 设计问题6.9.2 指针操作6.9.3 指针的相关问题6.9.4 Ada语言中的指针6.9.5 C和C++中的指针6.9.6 引用类型…

<<程序设计语言概念>>

…第7章 表达式与赋值语句第8章 语句辑控制结构第9章 子程序第10章 实现子程序第11章 抽象数据类型与封装结构第12章 面向对象程序设计的支持第13章 描述语法和语义第14章 异常处理和事件数理第15章 函数式程序设计语言第16章 逻辑程序设计语言参考文献

章节摘录

插图：增加表达思想的能力。
一般认为，人们思考问题的深度受到他们思考时所使用语言的表达能力的影响。
那些对自然语言理解肤浅的人，思维复杂度也有限，特别是在抽象的深度上。
换言之，人们难以将口头或书面无法表达的东西概念化。
程序员在开发软件的过程中同样受到这一限制。
他们开发软件所用的语言对他们所用的控制结构、数据结构和抽象层次也造成限制，从而也同样限制了他们能够构造的算法形式。
了解更多的程序设计语言的特性能够在软件开发时减少这些限制。
程序员学会新的语言结构后，能够提升软件开发时思维过程的层次。
可能有人认为，了解其他语言的功能对一个被要求使用不具有这一功能的语言进行开发的程序员没有帮助。
然而这种看法并不成立，因为通常来说语言的结构能够被不直接支持这些结构的其他语言模拟出来。
例如，一个了解Perl语言（Wall等，2000）中关联数组的结构和用法的C语言程序员，可能会用C语言设计出模拟关联数组的结构。
换句话说，对程序设计语言概念的学习能够使程序员对重要的语言特性与结构有充分的理解，鼓励程序员去使用它们，甚至在所用的语言不直接支持这种特性或结构的情况下。
扩充选择合适语言的背景知识。
许多专业的程序员没有受过多少正规的计算机科学教育，而是通过自学或单位内部培训。
这类培训通常只教授与公司当前项目直接相关的一两种语言。
其他许多程序员在很久以前受过正规的培训，他们那时所学的语言已经不再使用，现在程序设计语言中许多特性当时知道的人并不多。

编辑推荐

《程序设计语言概念(第9版)》：世界著名计算机教材精选

<<程序设计语言概念>>

版权说明