中国计算机学会教育专业委员会 推荐
全国高等学校计算机教育研究会 出版
高等学校规划教材

程序设计语言与编译

——语言的设计和实现

（第2版）

龚天富 编著

计算机学科教学计划2001

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

高等学校规划教材

程序设计语言与编译
——语言的设计和实现

（第 2 版）

龚天富　编著

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

内 容 简 介

本书是一本计算机专业的宽口径教材，新版覆盖 CC2001 和 CCC2002 教程中，除自动机外编程语言（PL）模块的全部知识点。内容涉及语言及其编译系统的设计要素、设计思想、设计方法、设计技术和设计风格等知识，全书分为上、下篇。上篇，程序设计语言的设计包括：绪论、数据类型、控制结构、程序语言设计、非过程式程序设计语言和形式语义学简介；下篇，程序设计语言的实现（编译）包括：编译概述、词法分析、自上而下的语法分析、自下而上的语法分析、语义分析和中间代码生成、代码优化和目标代码生成、运行时存储空间的组织。

本书的学习目标是，使读者掌握设计和实现一个程序设计语言的基本思想和方法，具有分析、鉴赏、评价、选择、学习、设计和实现一个语言的基本能力。本书力求简明、通俗，注重可读性，是大学计算机科学和软件工程等专业高级程序设计语言概论及编译技术课程教材，也是软件开发人员的学习参考书。

图书在版编目（CIP）数据

程序设计语言与编译：语言的设计和实现/龚天富编著．—2 版．—北京：电子工业出版社，2003.8
高等学校规划教材
ISBN 7-5053-9007-4

Ⅰ．程… Ⅱ．龚… Ⅲ．程序设计语言学—高等学校—教材 Ⅳ．TP311.1

中国版本图书馆 CIP 数据核字（2003）第 070678 号

责任编辑：童占梅
印 刷 者：北京大中印刷厂
出版发行：电子工业出版社 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：18 字数：446 千字
版 次：2003 年 8 月第 2 版 2003 年 8 月第 1 次印刷
印 数：5 000 册 定价：23.00 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。

新版说明

由中国计算机学会教育专业委员会和全国高等学校计算机教育研究会(简称"两会")组织和推荐,自1996年起电子工业出版社出版了基于CC1991教程的15本系列教材。该系列教材受到高校师生和读者的普遍欢迎和肯定,其中有11本入选1996～2000年全国工科电子类专业规划教材。

几年过去了,计算机学科又有了很大发展。IEEE-CS/ACM联合计算教程专题组,组织世界各国150多位专家,历时3年多,在美、欧、亚召开了一系列会议,在CC1991的基础上,发布了"Computing Curricula 2001-Computer Science Final Report"(简称CC2001)。专家们认为:随着计算机学科技术的飞速发展,使得现有的任何一所学校的计算机专业都很难再像CC1991所提到的那样,能够覆盖计算机学科的所有知识领域。所以,按市场需求将计算机学科划分为4个主要分支:计算机科学、计算机工程、软件工程和信息系统。其中计算机科学是各分支的基础,CC2001正是基于计算机科学制订的。我国"两会"追踪CC2001,经过3年多的工作,最后以中国计算机科学与技术教程2002研究组的名义推出了"China Computing Curricula 2002"(简称CCC2002)。CC2001与CC1991比较有以下几方面的变化:

(1) 将CC1991确定的11个主领域扩展为14个主领域:离散结构、编程基础、算法与复杂性、计算机组织与体系结构、操作系统、网络计算、编程语言、人-机交互、图形学与可视化计算、智能系统、信息管理、职业与社会问题、软件工程、数值计算。对各主领域的名称、核心内容及选学内容都进行了调整和扩充。

(2) 提出了课程的组织结构和实现策略。课程分为3类:入门课程、核心(必修)课程和附加(选修)课程。入门课程可按多种方式组织,使学生接触到计算机系统的设计、构造和应用,为学生提供实用性的技能训练,同时还应提高学生的兴趣和智慧;核心课程的组织可按传统、压缩、系统或网络等方法进行,特别强调贯彻CC1991提出的3个过程、12个重复概念、职业与社会的关系等方法论思想;此外,还应设置一些介绍热门或前沿技术的附加课程。

(3) 更加强调学生的专业实践,要求把专业实践放在重要位置,并贯穿于教学的全过程。

这次对系列教材的全面修订,力求反映计算机学科发展的最新成就,并力争符合CC2001和CCC2002所提出的要求及高校课程和教学改革的需要。这套教材的对象为本科生、研究生和大专生(通过删减使用)。信息技术领域的从业人员也可使用。

为了保证编审和出版质量,编委会进行了调整,电子工业出版社成立了编辑出版小组。在原教材工作的基础上,编委会对教材大纲逐一进行了认真讨论和评审,其中一些关键性和难度较大的教材还进行了多次讨论和修改。

限于水平和经验,教材中还会存在缺点和不足。希望读者提出中肯的批评和建议。读者可以通过电子工业出版社网站的读者论坛反馈信息并发表意见,我们在此表示衷心的感谢!

教材编委会

2003年5月

教材编委会

新版前言

本书是一本计算机专业的宽口径教材，新版按照 CC2001 和 CCC2002 教程改写，覆盖除自动机外编程语言(PL)模块的全部内容。

一个程序设计语言涉及设计者、实现者和使用者。读者在阅读本书之前，都学习过 1～2 门高级编程语言，已经是语言的使用者。他们在学习中还会有许多不解的问题：为什么在设计软件时要选择使用 C++而不是 FORTRON 或别的语言？为什么所使用的语言的数据结构和控制结构是这样规定的？运行中的错误是如何由编译程序发现的？其底层的异常处理机制是什么？编译器是如何工作的？更进一步，语言的设计者和实现者是如何设计、构造和应用编译系统的？……通过本书的学习，相信读者会得到问题的答案。语言及其编译系统的设计要素、设计思想、设计方法、设计技术和设计风格等知识，无论对于语言的高级使用者还是未来的语言设计和实现者都是必需的。本教材的目标是，使读者掌握设计和实现一个程序设计语言的基本思想和方法，具有分析、鉴赏、评价、选择、学习、设计和实现一个语言的基本能力。

随着计算机技术的快速发展，有越来越多的人认为，编译器的设计和实现是专家的工作领域，并非每个计算机专业的学生都需要设计和实现编译器的知识与能力，有的学校减少了学时，更有的学校砍掉了编译课程，取而代之一些更现代的课程。多年的教学经验告诉我们：编译系统作为系统软件之一，其设计和实现的系统性，能使学生对软件系统的结构形式及系统建立有一个充分的了解。另一方面，语言是软件设计与开发最重要的工具，程序设计语言数以千计，千姿百态，但它们在发展与变化之中体现出很多共性，又各具特点和优势，更有不少的创新，有必要从语言设计层面展现这些共性和语言发展的脉络及全貌，这些也是每个计算机从业人员需要透彻了解的。所以，无论从计算机专业人才的知识结构还是专业素养的培养来看，编译课程都是必不可少的核心课程。基于以上的考虑，我们重新设计了教材的结构和内容，将其分为上、下两篇，上篇为程序设计语言的设计，下篇为程序设计语言的实现(编译)。

本书在改写中力求简明、通俗，注重可读性。上篇中，在第 2 章增加了 C 和 C++语言的相关内容；还特别增加了第 4 章——程序设计语言的设计。通篇用抽象的方法，讨论程序设计语言的共性，并用具体的语言作为实例来说明这些共性。下篇中，所有章节全部改写，删除了一些内容，回避了一些基础理论问题，把这些问题留给专门的课程(如形式语言与自动机)去讨论。本书最后还增加了附录 Java 语言概述。相信这样的安排会适合更多学校的教学要求。本书为教师提供教学参考资料(包括习题参考答案)，请直接与作者联系(E-mail：gtf@uestc.edu.cn)，或通过电子工业出版社的教材服务部(E-mail：eduservice@phei.com.cn)获得教学支持。

本书的出版得到许多同事的帮助，杭诚方、陈文宇和王晓斌为资料的收集和整理做了大量的工作，并为本书的编写提出了许多有价值的建议，在此对他们表示衷心的感谢。特别要感谢向重伦教授，他在百忙之中抽出时间认真审阅了全稿，并提出了许多宝贵意见。还要感谢侯文永教授，他是我前一版的合作者，由于他的前期工作，为我此次改版打下了良好的基础。由于水平有限，时间仓促，难免出现谬误，恳请读者不吝赐教。

作　者

2003 年 4 月 4 日于成都

目　录

上篇　程序设计语言的设计

下篇　程序设计语言的实现(编译)

上篇　程序设计语言的设计

第1章　绪　　论

本章将讨论程序设计语言中的一些重要概念，为深入了解程序设计语言打下基础。同时还要介绍学习本书的目的和方法。最后一节简单介绍程序设计语言的发展历史。

1.1　引言

语言是人们交流思想的工具。人类在长期的历史发展过程中，为了交流思想、表达感情和交换信息，逐步形成了语言。这类语言，如汉语和英语，通常称为自然语言(Natural Language)。另一方面，人们为了某种用途，又创造出各种不同的语言，如旗语和哑语，这类语言通常称为人工语言(Artificial Language)。专门用于计算机的各种人工语言称为程序设计语言(Programmig Language)，本书将讨论这类语言的设计(Design)和实现(Implementation)。

1946年出现了第一台电子数字计算机(Electronic Digital Computer)，它一问世就成为强有力的计算工具。只要针对预定的任务(问题)，告诉计算机“做什么”和“怎么做”，计算机就可以自动地进行计算，对给定的问题求解。为此，人们需要将有关的信息告诉计算机，同时也要求计算机将计算结果告诉人们。这样，人与计算机之间就要进行通信(Communication)，既然要通信，就需要信息的载体。人们设计出词汇量少、语法简单、意义明确的语言作为载体，这样的载体通常称为程序设计语言。这类语言有别于人类在长期交往中形成的自然语言，它是由人设计创造的，故称为人工语言。

每当设计出一种类型的计算机，就随之产生一种该机器能理解并能直接执行的程序设计语言，这种语言称为机器语言(Machine Language)。用机器语言编写的程序由二进制代码组成，计算机可以直接执行。对人来说，机器语言程序既难编写，又难读懂。为了提高程序的可写性(Writability)和可读性(Readability)，人们将机器语言符号化，于是产生了汇编语言(Assemble Language)。机器语言和汇编语言都是与机器有关的语言(Machine-dependent Language)，通常称为低级语言(Low-level Language)。其他与机器无关的程序设计语言(Machine-independent Language)，通常称为高级语言(High-level Language)。由于计算机只理解机器语言，可直接执行用机器语言编写的程序，而用汇编语言和高级语言编写的程序，机器不能直接执行，必须将它们翻译成功能完全等价的机器语言程序才能执行。这个翻译工作是自动进行的，由一个特殊的程序来完成。翻译汇编语言的程序称为汇编程序(Assembler)，又称为汇编器；翻译高级语言的程序称为编译程序(Compiler)，又称为编译器。编写一个高级语言

的编译程序的工作，通常称为对这个语言的实现。

每种高级语言都有一个不大的字汇表(Vocabulary)及构造良好的语法(Syntax)规则和语义(Semantics)解释。规定这些基本属性，便于实现高级语言到低级语言的机器翻译。

高级语言较接近于数学语言和自然语言，它具有直观、自然和易于理解的优点。用高级语言编写的程序易读、易写、易交流、易出版和易存档。由于易理解，使程序员容易编出正确的程序，以便验证程序的正确性，发现错误后也容易修改。因此，用高级语言开发软件的成本比用低级语言低得多。今天，绝大多数的软件都是用高级语言开发的，因此，高级语言是软件开发的最重要工具。

由于高级语言独立于机器，用高级语言编写的程序很容易从一种机器应用到另一种机器上，因而具有较好的可移植性(Portability)。

高级语言至今还没有完全取代低级语言，在一些场合还必须使用机器语言或汇编语言，例如编译程序的目标程序和各种子程序，以及实时应用系统中要求快速执行的代码段等。但是，随着功能强大且具有高级语言和汇编语言特性的 C 语言的出现，使应用汇编语言的人越来越少。

人们在进行科学研究的过程中，总是对具体现象和事物进行观察、分析和综合，以发现它们的重要性质和特征，建立相应的模型。这种通过观察、分析和综合建立模型的过程称为抽象(Abstract)。利用抽象模型，人们可以把注意力集中在有关的性质和特征上，忽略那些不相干的因素。本书在后面的讨论中，大量使用抽象的方法阐述程序设计语言的概念和结构，然后以各种语言中的具体实例来说明这些概念和结构，从而教会读者如何去设计一个程序设计语言。事实上，程序设计语言中处处都使用了抽象概念，例如变量(Variable)是存储单元(Memory Cell，Memory Location)的抽象；子程序(Subroutine 或 Subprogram)是一段多处重复执行的程序段的抽象等。

在此，我们讨论的对象是高级语言，接下来利用抽象的方法讨论高级语言具有的共性概念和结构及它们的属性。为了叙述简洁，在不引起混淆的情况下，以下将高级语言简称为语言。

一种语言涉及设计者、实现者和使用者，有了设计者和实现者，才可能有使用者。读者在中学或进入大学后，已经使用过这种或那种程序设计语言，也就是说，已经是使用者。本书的目标是引导读者成为语言的设计者和实现者。由于教学时数的限制，本书仅给出入门知识和技术，读者如果要真正设计或实现一个语言，尚需查阅相关的文献资料，建议感兴趣的读者阅读参考文献[59]。通过本书的学习，读者可以提高鉴赏和评价语言(或语言设计方案)的能力；了解语言的重要概念、功能和限制，以便具有为某个目的选择一种恰当语言的能力；具有设计一种语言或扩充现有语言的能力；初步具有实现一个语言的能力。最终使读者能够鉴赏、分析、选择、设计和实现程序设计语言。

1.2 强制式语言

通常的高级语言又称为强制式语言(Imperative Language)，本书主要讨论强制式语言的设计和实现。

1.2.1 程序设计语言的分类

语言的分类没有一个统一的标准，通常按不同的尺度有不同的分类方法和结果。例如，按语言设计的理论基础来分类，可分为4类强制式，其基础是冯·诺依曼(Von Neumann)模型，函数式语言(Functional Language)的基础是数学函数；逻辑式语言(Logic Programming Language)的基础是数理逻辑谓词演算；对象式语言(Object-oriented Language)的基础是抽象数据类型(Abstract Data Type)。

人们习惯上按语言的发展进程来对语言进行分类。

1. 第一代语言

第一代语言(First-generation Language)通常称为机器语言，它与机器孪生。实际上，它完全依赖于机器的指令系统(Instruction System)，以二进制代码表示。这类语言程序既难编写又难读懂。

2. 第二代语言

第二代语言(Second-generation Language)通常称为汇编语言，它将机器语言符号化，用符号来代表机器语言的某些属性。例如用符号名来代表机器语言的地址码。这样可以帮助程序员记忆，摆脱使用二进制代码的烦恼，提高了程序的可写性和可读性。

不同的机器有不同的机器语言和汇编语言，通常人们又把它们称为与机器有关的语言。

3. 第三代语言

第三代语言(Third-generation Language)指通常的高级语言，这类语言的设计基础与冯·诺依曼体系结构有关。高级语言程序按语句顺序执行，因此又称为面向语句的语言(Sentence-oriented Language)。通常，每条语句对应机器的一个命令，因此又称命令式语言(Order Language)。用这类语言编写的程序，实际上是描述对问题求解的计算过程，因此也有人称它为过程语言(Procedure Language)。

这类语言书写自然，具有更好的可读性、可写性和可修改性(Modifiability)，读者使用过的语言大多是这种高级语言。高级语言程序就是要告诉计算机“做什么”和“怎么做”。

4. 第四代语言

第四代语言(Fourth-generation Language)是说明性语言(Declaretion Language)，它只需要告诉(说明)计算机“做什么”，不必告诉计算机“怎么做”；也就是说不需要描述计算过程，系统就能自动完成所需要做的工作。所以，这类语言又称为超高级语言或甚高级语言(Very-high-level Language)，典型的例子是SQL语言。

5. 新一代语言

另一类不同风格的语言，如函数式和逻辑式语言，它们的理论基础和程序设计风格均不同于高级程序设计语言(参看第5章)。它们不适合称为第五代或第六代语言，因此，语言学家把它们称为新一代程序设计语言。

1.2.2 冯·诺依曼体系结构

当今的计算机模型是由数学家冯·诺依曼提出来的，我们称为冯·诺依曼模型(Von

Neumann Model)或冯·诺依曼机(Von Neumann Machine)。直到今天,几乎所有的计算机都是沿用这一模型设计的。1978 年,巴科斯(Buckus)在获得图灵奖的颁奖大会上发表演说,批判了冯·诺依曼的体系结构和程序设计风格,称这种结构和风格影响了计算机系统的执行效率,提出了函数式程序设计风格,并发表了 FP 和 FFP 语言。今天,人们越来越多地强调使用并行体系结构和并行程序设计,以提高计算机的执行效率。

下面讨论冯·诺依曼体系结构和它对高级语言的影响。

冯·诺依曼机的概念基于以下思想:一个存储器(用来存放指令和数据),一个控制器和一个处理器(控制器负责从存储器中逐条取出指令,处理器通过算术或逻辑操作来处理数据),最后的处理结果还必须送回存储器中。我们可以把这些特点归结为以下 4 个方面。

(1) 数据或指令以二进制形式存储(数据和指令在外形上没有什么区别,但每位二进制数字有不同的含义)。

(2) "存储程序"方式工作(事先编好程序,执行之前先将程序存放到存储器某个可知的地方)。

(3) 程序顺序执行(可强行改变执行顺序)。

(4) 存储器的内容可以被修改(存储器的某个单元一旦放入新的数据,则该单元原来的数据立即消失,且被新数据代替)。

冯·诺依曼体系结构的作用体现在命令式语言的下述 3 大特性上。

(1) 变量　存储器由大量存储单元组成,数据就存放在这些单元中,汇编语言通过对存储单元的命名来访问数据。在命令式语言中,存储单元及其名称由变量的概念来代替。变量代表一个(或一组)已命名的存储单元,存储单元可存放变量的值(Value),其值可以被修改;也正是这种修改,产生了副作用(Side Effect)问题(参看 3.3.1 节)。

(2) 赋值　使用存储单元概念的另一个后果是每个计算结果都必须存储,即赋值于某个存储单元,从而改变该单元的值。

(3) 重复　语句按顺序执行,指令存储在有限的存储器中;要完成任何复杂的计算,有效的办法就是重复执行某些指令序列。

1.2.3　绑定和绑定时间

一个对象(或事物)与其各种属性建立起某种联系的过程称为绑定(Binding)。这种联系的建立,实际上就是建立了某种约束。绑定这个词是由英文 Binding 音译过来的,过去也曾翻译成"联编"、"汇集"、"拼接"或"约束"等。现在之所以选定"绑定"这个词,除了它能形象地表达上述过程外,它还与英文读音一致。

一个程序往往要涉及若干实体,如变量、子程序和语句等。实体具有某些特性,这些特性称为实体的属性(Attribute)。变量的属性有名字(Name)、类型(Type)和保留其值的存储区等。子程序的属性有名字、某些类型的形参(Formal Parameter)和某种参数传递方式的约定等。语句的属性是与之相关的一系列动作。在处理实体之前,必须将实体与相关的属性联系起来(即绑定)。每个实体的绑定信息来源于所谓的描述符(Descriptor)。描述符实际上是各种形式的表格的统称(抽象),用来存放实体的属性。例如,程序员用类型说明语句来描述变量的类型属性,编译时将它存放在符号表(Symbol Table)中;程序员用数组说明语句来描述一个数组的属性,编译时将这些属性存放在一个专门设计的表格中(参见 7.3 节),这个表格称为数

组描述符，又称内情向量(Dope Vector)。

对于计算机科学来说，绑定是一个随处遇到且重复使用的重要概念，借助于它可以阐明许多其他概念。把对象(实体)与它的某个属性联系起来的时刻称为绑定时间(Binding Time)。一旦把某种属性与一个实体绑定，这种约束关系就一直存在下去，直到对这一实体的另一次绑定实现，该属性的约束才会改变。

某些属性可能在语言定义时绑定，例如，FORTRAN 语言中的 INTEGER 类型，在语言定义的说明中就绑定了，它由语言编译器来确定这个类型所包含的值的集合。Pascal 语言中允许重新定义 integer 类型，因此 integer 类型在编译时才能绑定一个具体表示。若一个绑定在运行之前(即编译时)完成，且在运行时不会改变，则称为静态绑定(Static Binding)。若一个绑定在运行时完成(此后可能在运行过程中被改变)，则称为动态绑定(Dynamic Binding)。

今后讲到的许多特性，有的是在编译时所具有的，有的是在运行时所具有的，凡是在编译时确定的特性均称为静态的(Static)；凡是在运行时确定的特性均称为动态的(Dynamic)。

1.2.4 变量

强制式语言最重要的概念之一是变量，它是一个抽象概念，是对存储单元的抽象。如前所述，冯·诺依曼机基于存储单元组成的主存储器(Main Memory)概念，它的每个存储单元用地址来标识，可以对它进行读或写操作。写指修改存储单元的值，即以一个新值代替原来的值。语言中引入变量的概念，实质上是对一个(或若干个)存储单元的抽象，赋值(Assignment)语句则是对修改存储单元内容的抽象。

变量用名字来标识，此外它还有 4 个属性：作用域(Scope)、生存期(Lifetime)、值和类型。变量可以不具有名字，这类变量称为匿名变量(Anomymous Variable)。下面将讨论上述 4 个属性，以及它们在不同语言中所采用的绑定策略。

1. 变量的作用域

变量的作用域是指可访问该变量的程序范围。在作用域内，变量是可控制的(Manipulable)。变量可以被静态地或动态地绑定于某个作用范围。在作用域内变量是可见的(Visible)，在作用域外变量是不可见的(Invisible)。按照程序的语法结构定义变量的作用域的方法，称为静态作用域绑定(Static Scope Binding)。这时，对变量的每次引用都静态地绑定于一个实际(隐式或显式)的变量说明。大多数传统语言采用静态作用域绑定规则。有的语言在程序执行中动态地定义变量的作用域，这称为动态作用域绑定(Dynamic Scope Binding)。每个变量说明延伸其作用域到它后面的所有指令(语句)，直到遇到一个同名变量的新说明为止。APL，LISP 和 SNOBL 4 语言是采用动态作用域规则的语言。

动态作用域规则很容易实现，但掌握这类语言的程序设计比较困难，实现的有效性也偏低。对于动态作用域语言，给定变量绑定于特定说明之后程序执行到的某个特定点，因为其不能静态确定，所以程序很难读懂。

2. 变量的生存期

一个存储区绑定于一个变量的时间区间称为变量的生存期。这个存储区用来保存变量的值。我们将使用术语“数据对象(Data Object)”，或简称“对象(Object)”来同时表示存储区和它保存的值。