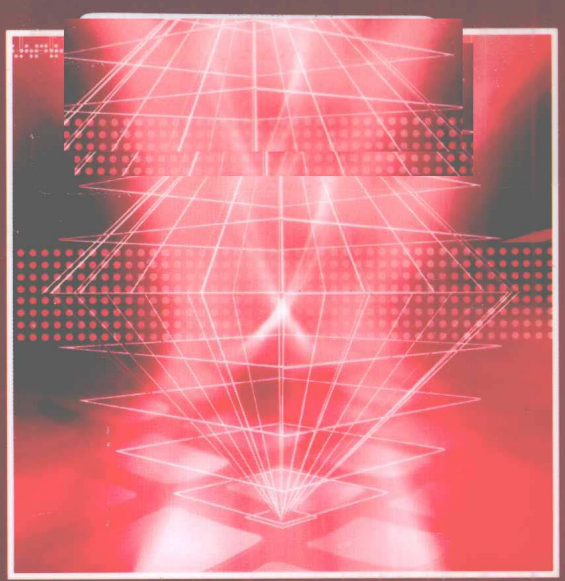
普通高等教育“十一五”国家级规划教材
普通高等教育“十一五”计算机类规划教材

编译原理及实现技术

第2版

刘磊　等编著



免费
电子课件

机械工业出版社
CHINA MACHINE PRESS

普通高等教育“十一五”国家级规划教材
普通高等教育“十一五”计算机类规划教材

编译原理及实现技术

第 2 版

刘 磊 郭德贵
张 晶 张 红 杨 冬 编著

机 械 工 业 出 版 社

编译原理是计算机学科的一门重要专业基础课。本书旨在介绍编译程序设计的基本原理、实现技术、方法和工具，充分考虑了教师便于教学，学生便于自学的问题。在介绍基本原理和实现技术中，注重循序渐进、深入浅出，每一章节都提供了编译程序实现的具体实例，每章末尾给出了丰富的习题以辅助学生更好地掌握编译过程。

本书包含了编译程序设计的基础理论和具体实现技术，主要内容有：形式语言和自动机理论、词法分析、语法分析、语义分析、中间代码生成、中间代码优化和目标代码生成等编译过程。

本书可作为大专院校计算机专业本科生教材，也可作为计算机工程技术人员的参考书。为方便教师教学，本书配有免费电子课件，欢迎选用本书作教材的教师登录 www. cmpedu. com 下载或发邮件到 llm7785@ sina. com 索取。

图书在版编目（CIP）数据

编译原理及实现技术/刘磊等编著．—2 版．—北京：机械工业出版社，2010.7

普通高等教育“十一五”国家级规划教材．普通高等教育“十一五”计算机类规划教材

ISBN 978-7-111-31261-1

Ⅰ.①编… Ⅱ.①刘… Ⅲ.①编译程序－程序设计－高等学校－教材 Ⅳ.①TP314

中国版本图书馆 CIP 数据核字（2010）第 133154 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）

策划编辑：王保家 责任编辑：刘丽敏 版式设计：霍永明

责任校对：李秋荣 封面设计：张 静 责任印制：乔 宇

北京机工印刷厂印刷（三河市南杨庄国丰装订厂装订）

2010 年 8 月第 2 版第 1 次印刷

184mm×260mm · 12 印张 · 293 千字

标准书号：ISBN 978-7-111-31261-1

定价：23.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务

社服务中心：(010)88361066

销 售 一 部：(010)68326294

销 售 二 部：(010)88379649

读者服务部：(010)68993821

网络服务

门户网：http://www. cmpbook. com

教材网：http://www. cmpedu. com

封面无防伪标均为盗版

前　言

编译原理是计算机学科的一门重要专业基础课。学习编译课程，不仅可以掌握编译程序本身的实现技术，而且能够提高对程序设计语言的理解，提高开发大型软件的能力，提高软件程序的设计能力，提高抽象思维能力。

编译程序是计算机系统软件的重要组成部分，其基本原理和实现技术也适用于一般软件的设计和实现，而且在软件工程、软件自动化、程序分析等领域有着广泛的应用。通常把编译程序视为高级语言到机器语言的转换程序，而这种转换不是结构上的变换，而是基于语言语义的等价变换，因此，编译程序设计的难度和复杂性是很高的。同时，编译原理也是一门对实践性要求较高的课程。本书充分考虑了便于教师教学，便于学生自学的问题，循序渐进地介绍了编译程序设计的基本原理、主要实现技术、基本设计方法和一些自动构造工具，深入浅出地介绍了完整的编译程序构造和实现过程，使学生能够掌握编译的整体结构。

本书共分10章。第1章介绍了编译程序的基础知识。第2章作为编译程序的理论基础，简单介绍了形式语言、有限自动机理论和正则表达式等基础知识。第3章以正则表达式、有限自动机为工具，讨论了词法分析程序的设计与实现，并简要介绍了词法分析器生成器LEX的基本原理和使用方法。第4章介绍了自顶向下的语法分析方法的基本思想，并讨论了递归下降语法分析方法和LL（1）语法分析方法的实现技术。第5章介绍了自底向上语法分析方法的基本思想，并详细讨论了LR类语法分析的基本原理和实现方法，同时简单介绍了流行的语法分析器生成器YACC、Bison等工具。第6章专门介绍语义分析，包括标识符、类型、值的内部表示及其构造，符号表的构造及其管理。第7章介绍了中间代码生成，包括常用中间代码结构、表达式的中间代码、下标变量的中间代码以及语句的中间代码。第8章介绍了中间代码优化的基本方法，重点讨论了常量表达式优化、公共表达式优化和循环不变式优化。第9章介绍编译程序运行时的存储空间组织与存储分配技术，重点讨论了运行时的存储结构、存储分配、过程活动记录以及变量访问环境等。第10章介绍了目标代码生成的基本技术，重点讨论了中间代码到目标代码的翻译。

本书是根据作者多年的教学经验和科研经历编写而成的，由刘磊教授确定内容的选取和组织，最后由刘磊教授定稿。张红与杨冬编写第1、2、10章；郭德贵编写第3～5、7～8章；张晶编写第6、9章；刘磊参与了各章的编写工作。

本书是在第1版的基础上修订而来。在修订过程中我们尽量保持原书的风格，同时又增加了一些例题和习题，以帮助读者理解编译概念和原理。

由于编者水平有限，书中难免还存在一些缺点和不妥之处，殷切希望广大读者批评指正。

编　者

目　录

第1章 编译引论

1.1 程序设计语言和编译程序

众所周知，计算机系统的工作是由事先设计好的程序来控制的。人们首先按自己的需要把让计算机做的工作编写成计算机程序，并把程序送入计算机，然后启动计算机执行程序，从而完成人们设想的要计算机完成的工作。

在计算机发展的初期，人们直接使用机器语言编写程序。机器语言由能被计算机直接执行的机器指令组成，每条机器指令是一串二进制代码，用机器语言编写出来的程序是一串二进制代码序列。这种机器语言程序能被计算机直接识别和执行，但是它很不直观，可读性很差，编写过程费时费力，非常容易出错，而且出错后难于调试和修改。基于上述原因，人们引入了汇编语言。

汇编语言是符号化了的机器语言，也就是引入一些助记符表示机器指令中的操作码和存储地址。汇编语言程序的语句基本上与机器指令对应，却不能被机器硬件直接识别和执行。汇编语言大大提高了编程的速度和准确度，人们至今仍在使用它，在代码需要极快的运行速度和极高的简洁度时尤为如此。但是，汇编语言也有很多缺点：编写起来仍然很不容易；阅读和理解很难；而且严格依赖于特定的计算机，程序不便于移植。因此，人们又进一步引进了高级语言。

高级语言不依赖于具体的计算机，它以较接近于自然语言的方式来描述程序的操作。显然，这种程序十分好读，编写起来也很容易，而且还具有通用性，可以在不同机器上运行，便于移植。

尽管用汇编语言和高级语言编写程序比用机器语言方便得多，但是机器硬件只能识别机器语言，因此必须有这样的一个程序，它把用汇编语言或高级语言编写的程序转换成等价的机器语言程序。我们把这种执行转换功能的程序统称为翻译程序（Translator），其中汇编语言的翻译程序称为汇编程序（Assembler），高级语言的翻译程序称为编译程序（Compiler），也称为编译器。

编译程序的输入对象称为源程序（Source Program），它是使用高级语言编写的程序；输出对象称为目标程序（Object Program），目标程序可以是机器语言程序、汇编语言程序或用户自定义某种中间语言程序。

编译程序支持的源程序的执行分为两个阶段：编译阶段和运行阶段。编译阶段对整个源程序进行分析，翻译成等价的目标程序，翻译的同时做语法检查和语义检查，凡是有错误的源程序均指出其错误；运行阶段在运行子程序的支持下执行目标程序。运行子程序是为了支持目标程序的运行而开发的程序。例如，有系统提供的标准函数和其他目标程序所调用的程序等。其过程如图 1.1 所示。

高级语言程序的执行也可以采用另一种方式，即并不把源程序先翻译成目标程序然后再

执行目标程序，而是一个语句一个语句地读入源程序，边翻译边执行，在翻译过程中不产生目标程序。这种处理程序称为解释程序（Interpreter）。解释程序的工作结果是源程序的执行结果而不是目标程序。其功能如图 1.2 所示。

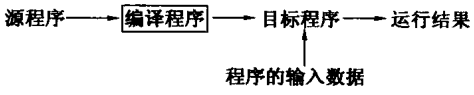


图 1.1　高级程序设计语言的编译实现方式

图 1.2　高级程序设计语言的解释实现方式

从理论上说，任何程序设计语言都可以被解释或被编译，在实际应用中，解释方式特别适合于交互式语言；而且解释方式允许程序执行时改变自身，比如调试程序。这种情形编译程序不易胜任，因为它需要动态编译，而解释程序可以毫无困难的胜任；此外，解释程序不依赖于目标机，因为它不生成目标代码，可移植性优于编译程序。但是和编译程序相比，解释程序开销大，运行速度慢得多，因此，大多数高级语言主要采用编译方式。解释方式和编译方式的最根本区别在于：在解释方式下，并不生成目标代码程序，而是直接执行源程序本身。

实现高级语言的方式除了上文提到的编译方式和解释方式，还有一种转换方式。假如要实现 L 语言，现在已有 L′语言的编译程序，就可以先把用 L 语言编写的程序转换成等价的 L′语言的程序，再利用 L′语言的编译程序实现 L 语言，其功能如图 1.3 所示。在实际应用中，这种转换方式使用得较少。

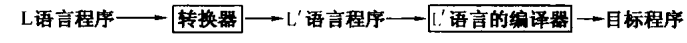


图 1.3　高级程序设计语言的转换实现方式

1.2　编译程序的结构

1.2.1　编译程序的构成

编译程序是比较庞大的系统软件，它的处理过程复杂，不同的编译程序实现的方法千差万别，构造原理各异，但编译程序的基本逻辑功能及必须完成的处理任务是非常相似的。一般编译程序要完成以下工作：词法分析、语法分析、语义分析、中间代码生成、中间代码优化、目标代码生成、表格管理和出错处理。它们之间的关系如图 1.4 所示。其中除了中间代码部分外，其他所有工作都是任何编译程序完成的工作，但完成的方式可能不同，有的可能把几个工作合起来交错完成。不过从原理的角度说，图 1.4 所示的结构是比较理想的逻辑结构，它具有一般性。

各部分的主要任务：

1. 词法分析阶段

词法分析的任务是扫描源程序的 ASCII 码序列，识别出一个个具有独立意义的最小语法单位，即单词，如关键字（if、then、for、while 等）、标识符、常数、运算符和界符（标点符号、左右括号等），并把每个单词的 ASCII 码序列替换为所谓的“TOKEN”形式。其中 TOKEN 是单词的内部表示，它的结构没有统一的规定，依赖于具体语言和具体编译器，也

依赖于词法分析器的具体功能，但一般包括单词属性标识和单词内码两个部分。在词法分析阶段还要检查括号类配对等词法错误并去掉源程序中的注释。词法分析阶段不依赖于语言的语法定义。

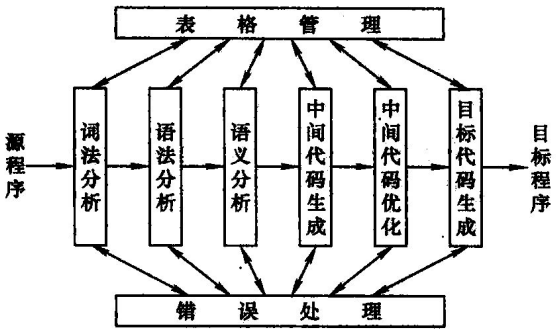


图 1.4 编译器的功能结构图

2. 语法分析阶段

语法分析的任务是根据程序设计语言的语法规则，把词法分析的结果分解成各种语法单位，同时检查程序中的语法错误。语法分析的扫描对象有两种可能：一种是将词法分析程序作为独立的一遍运行，扫描整个源程序的 ASCII 码序列，将之转换为TOKEN序列，输出到一个中间文件，该文件作为语法分析程序的扫描对象继续编译的过程；更一般的情况是将词法分析程序设计成一个子程序，每当语法分析程序需要读取单词时，则调用该子程序。这种设计方案中，词法分析程序和语法分析程序处于同一遍，可以省去中间文件。

3. 语义分析阶段

这一阶段的任务是对语法分析所识别出的各类语法范畴，分析其含义，并进行静态语义检查。例如，变量是否定义、类型是否匹配等。这一阶段所依循的是语言的语义规则。通常使用属性文法描述语义规则。

4. 中间代码生成

在进行了上述的语法分析和语义分析阶段的工作后，有些编译程序将源程序变成一种内部表示形式，这种内部表示形式叫做中间代码。使用中间代码的主要好处是便于移植、便于修改、便于优化。这种中间代码的形式有很多种，常见的有后缀式（栈式）中间代码、三地址中间代码（三元式和四元式）、图结构中间代码（树，DAG）。其中，后缀式中间代码是最早使用的一种中间代码，现在很少使用，目前使用的主要是后两种。

5. 中间代码优化

此阶段的任务是对前阶段产生的中间代码在不改变源程序语义的前提下进行加工变换，使生成的代码更为高效，缩短运行时间或节省存储空间。主要的优化方式包括常量表达式优化、公共子表达式优化、不变表达式的循环外提和削减运算强度等。

6. 目标代码生成

这一阶段的任务是把中间代码变换成特定机器上的机器指令代码或汇编指令代码。这是编译的最后阶段，因为目标语言的关系而十分依赖于硬件系统。如何充分利用寄存器、合理选择指令、生成尽可能短而有效的目标代码，都与目标机的结构有关。

生成的目标代码如果是汇编指令代码，则需经由汇编程序处理后才能执行；生成的目标代码如果是绝对指令代码，则可直接投入运行；如果是可重定位的指令代码，那么目标代码只是一个代码模块，必须由连接装配程序将输入/输出模块、标准函数等系统模块与目标代码模块连接在一起，才能形成一个绝对指令代码程序以供执行。大多数现代实用的编译程序生成的目标代码都是这种可重定位的指令代码。

7. 表格管理

编译程序在对源程序的分析过程中，需要创建和管理一系列的表格，以登记源程序的各类信息和编译各阶段的进展情况。例如，编译程序需要知道变量的类型，数组的大小，函数的参数个数和类型等，这些信息一般可以从源程序中得到。随着编译过程的进行，需要不断地建表、查表和填表，或修改表中的某些数据，或从表中查找相关信息，这些活动贯穿整个编译过程的始终。因此，合理地设计和使用表格是编译程序构造中的一个非常重要的问题。不少编译程序都设立一些专门子程序（称为表格管理程序），它们专门负责管理表格。

8. 错误处理

一个编译程序不仅应能对书写正确的程序进行翻译，而且应能对出现在源程序中的错误进行处理。错误包括词法错误、语法错误、静态语义错误、动态语义错误，其中动态语义错误只能在运行目标程序时才能发现。在编译程序的各个阶段都要有错误处理部分，词法错误和语法错误都集中一次完成检查，而语义检查则分散在以后的各个阶段在完成别的工作时顺便完成。

如果源程序有错误，编译程序应设法发现错误，把相关错误信息报告给用户。这部分工作是由专门的错误处理程序完成的。一个好的编译程序应能最大限度地发现源程序中的各种错误，准确地指出错误的性质和发生错误的位置，并且能将错误所造成的影响限制在尽可能小的范围内，使得源程序的其余部分能继续被编译下去，以便进一步发现其他可能的错误。

1.2.2 遍

前面介绍的编译过程的各个阶段仅仅是逻辑功能上的一种划分。在具体实现上，受不同源语言、设计要求和计算机硬件条件的限制，往往将编译程序组织成若干遍（Pass）。所谓“遍”就是对源程序或源程序的中间表示形式从头到尾扫描一次，并作加工处理，生成新的中间结果或目标程序。既可以将编译过程中的几个不同阶段合为一遍，也可以把一个阶段的工作分为若干遍。例如，词法分析这一阶段可以作为单独的一遍，但更多的时候是把词法分析程序作为语法分析程序的子程序来加以调用，将词法分析阶段和语法分析阶段合并为一遍。

1.2.3 编译程序的前端和后端

概念上，有时把编译程序划分为编译前端和编译后端。前端主要由与源语言有关但与目标机无关的那些部分组成。编译前端通常包括词法分析、语法分析、语义分析、中间代码生成，与目标机无关的中间代码优化部分也可包含在前端，当然，前端也包括相应部分的错误处理。

编译后端包括与目标机有关的中间代码优化部分和目标代码生成等。一般来说，这些部分与源语言无关而仅仅依赖于中间语言。很明显，编译后端是面向目标语言的，而编译前端则不是，它几乎独立于目标语言。

一种典型的做法是取已有的编译程序的前端，改写其后端以生成不同目标机上相同源语言的编译程序。如果编译程序的后端是经过精心设计的，这样的改写无需太大的工作量，可以很方便地将程序设计语言移植到不同体系结构的计算机上。

1.3 编译程序和程序设计环境

除了编译程序之外，程序设计开发人员通常还需要一些其他的工具，如编辑器、连接程序、装入程序、调试工具和项目管理程序等。编译程序和这些程序一起，构成了一个完整的程序设计环境。

1. 编辑器

编译器通常接受由任何生成标准文件（例如 ASCII 文件）的编辑器（Editor）所编写的源程序，而且编辑器和编译器及其他程序通常捆绑在一起，构成交互开发环境。有些编辑器还根据源语言的特点，具有语法制导的结构化编辑功能。除了一般文本编辑器的功能外，还可对正在编辑的文本进行分析、提示，能自动地提供关键字和与其匹配的关键字，如 if 和 then，begin 和 end 的配对。这样可以减少程序中的语法错误，提高效率和质量。

2. 预处理器

预处理器（Preprocessor）是翻译工作开始之前由编译器调用的独立程序，它所做的工作包括删除源程序中的注释、执行宏替换以及包含文件的嵌入等。

3. 连接程序

连接程序（Linker）负责将分别在不同的目标文件中编译或汇编的代码集中到一个可执行文件中，并将目标和标注库函数的代码以及计算机的操作系统提供的资源连接在一起。连接程序严重依赖于操作系统和处理器。

4. 装入程序

装入程序（Loader）用来把程序加载到内存储器中，以便执行。由于用户的程序经汇编或编译后生成的目标代码通常采用相对地址的形式，它的起始地址是不确定的。这样的代码被称为可重定位的。装入程序可处理所有与指定的基地址或起始地址有关的可重定位的地址，它使得可执行代码更加灵活。

5. 调试程序

调试程序（Debugger）是可在被编译了的程序中判定执行错误的程序，它也经常与编译器一起放在集成开发环境（IDE）中。运行一个带有调试程序的程序与直接执行不同。这是因为调试程序保存着大量的源代码信息（例如行数和变量名等）。它还可以在预先指定的位置（称为断点）暂停执行，并提供有关已调用的函数和变量的当前值等信息。

在高级语言发展的早期，这些工具都是独立的、缺乏整体性。随着程序设计语言的发展，这些工具往往被集成在一起，构成了所谓的集成开发环境（IDE），集编辑、编译、调试、连接、运行等功能于一体。在这种集成开发环境中，编译程序起到核心作用。

1.4 编译程序的实现

编译程序是一个相当复杂的系统程序，通常有上万甚至几万条指令。随着编译技术的发展，编译程序的开发周期也在逐渐缩短，但仍然需要很多人年，而且工作很艰巨，正确性也不易保证。

要实现一个编译程序，通常需要做到：

1）对源语言的语法和语义要有准确无误的理解，否则难以保证编译程序的正确性。

2）对目标语言和编译技术也要有很好的了解，否则会生成质量不高的目标代码。

3）确定对编译程序的要求，如搞不搞优化，搞优化搞到哪一级等。

4）根据编译程序的规模，确定编译程序的扫描次数、每次扫描的具体任务和所要采用的技术。

5）设计各遍扫描程序的算法并加以实现。

一般开发编译程序有如下几种可能途径：

1. 转换法（预处理法）

假如要实现 L 语言的编译器，现在有 L′语言的编译器，那么可以把 L 语言程序转换成 L′语言的程序，再利用 L′语言的编译器实现 L 语言，这种方法通常用于语言的扩充。如对于 C ++语言，可以把 C ++程序转换成 C 程序，再应用 C 语言的编译器进行编译，而不用重新设计和实现 C ++编译器。常见的宏定义和宏扩展都属于这种情形。

2. 移植法

假设在 A 机器上已有 L 语言的编译程序，想在 B 机器上开发一个 L 语言的编译程序。这里有两种实现方法：

（1）实现方法一

最直接的办法就是将 A 机的代码直接转换成 B 机代码。

（2）实现方法二

假设 A 机和 B 机上都有高级程序设计语言 W 的编译程序，并且 A 机上的 L 语言编译程序是用 W 语言写的，可以修改 L 编译程序的后端，即把从中间代码生成 A 机目标代码部分改为生成 B 机的目标代码。这种在 A 机上产生 B 机目标代码的编译程序称为交叉编译程序（Cross Compiler）。

3. 自展法

实现思想：先用目标机的汇编语言或机器语言书写源语言的一个子集的编译程序，然后再用这个子集作为书写语言，实现源语言的编译程序。通常这个过程会分成若干步，像滚雪球一样直到生成预计源语言的编译程序为止。这样的实现方式称为自展技术。使用自展技术开发编译器要求这种高级语言必须是能够编译自身的。

4. 工具法

20 世纪 70 年代随着诸多种类的高级程序设计语言的出现和软件开发自动化技术的提高，编译程序的构造工具陆续诞生，如 20 世纪 70 年代 Bell 试验室推出的 LEX、YACC 至今还在广泛使用。其中 LEX 是词法分析器的自动生成工具，YACC 是语法分析器的自动生成工具。然而，这些工具大都是用于编译器的前端，即与目标机有关的代码生成和代码优化部分由于对语义和目标机形式化描述方面还存在困难，虽有不少生成工具被研制，但还没有广泛应用。

5. 自动生成法

如果能根据对编译程序的描述，由计算机自动生成编译程序是最理想的方法，但需要对语言的语法、语义有较好的形式化描述工具，才能自动生成高质量的编译程序。目前，语法分析的自动生成工具比较成熟，如前面提到的 YACC 等。但是整个编译程序的自动生成技术还不是很成熟，虽然有基于属性文法的编译程序自动生成器和基于指称语义的编译程序自动

生成器，但产生目标程序的效率很低，离实用尚有一段距离，因此，要想真正地实现自动化，必须建立形式化描述理论。

习 题 1

1. 典型的编译程序在逻辑功能上由哪几部分组成？
2. 实现编译程序的主要方法有哪些？
3. 将用户使用高级语言编写的程序翻译为可直接执行的机器语言程序有哪几种主要的方式？
4. 编译方式和解释方式的根本区别是什么？

第 2 章　形式语言与自动机理论基础

自然语言（Natural Language）是人类日常交流所使用的语言，比如汉语、英语和法语等，这类语言不是人为设计而是自然进化的语言，往往过于复杂，难以描述。形式语言（Formal Language）是为了特定应用而人为设计的语言。例如，数学家用的数字和运算符号、化学家用的分子式等。人与计算机打交道的程序设计语言也是一种形式语言，是专门设计用来表达计算过程的形式语言，具有语法严格、结构正规以及便于计算机处理等特点。自从 1956 年语言学家 Noam Chomsky 建立了形式语言的描述以来，形式语言学理论发展迅速，对计算机科学的发展有着深刻的影响。

程序设计语言等形式语言和自然语言也存在着共性，即语言的核心均由语法和语义两部分构成。所谓一种程序设计语言的语法是指一组规则，用它可以形成和产生一个合适的程序。语法是语言的形式，语义是语言的内容。通常将程序设计语言的语义分为两类：静态语义和动态语义。静态语义是一系列限定规则，用于确定哪些合乎语法的程序是合适的；动态语义也称作运行语义或执行语义，表明程序要做什么，要计算什么。

上下文无关文法、有限自动机和正则表达式是描述形式语言语法的工具，本章将介绍这些形式语言理论的基础知识。

阐明语义要比阐明语法难得多，在第 3 章对其做简单介绍。

2.1　基本概念

下面针对形式语言的基本概念进行简单的介绍。

定义 2.1　字母表

字母表（Alphabet）是元素的非空有穷集合，字母表中的元素称为该字母表的字母（Letter），也可称为符号（Symbol），或者字符（Character）。值得注意的是，字母表具有非空性和有穷性，它包括了语言中允许出现的全部符号。

字母表有时也称为符号表，通常用 Σ 表示。

定义 2.2　符号串

由字母表中的符号组成的任意有穷序列称为符号串。符号串还可以称为字符串、字或句子。

例如 0、1、00、11、01、10、100、101 及 110 均是字母表 $\Sigma = \{0, 1\}$ 上的符号串。

在符号串中，符号的顺序是很重要的。在上例中，01 和 10 就不是同一个符号串，101 和 110 也不是同一个符号串。可以使用字母来表示符号串，如 x = 010，表示字符串 x 是由 0、1、0 三个符号，并按此顺序组成的符号串。

如果某符号串 x 中有 m 个符号，则称其长度为 m，表示为 $|x| = m$。例如，符号串 01 的长度为 2，表示为 $|01| = 2$；符号串 1011 的长度为 4，表示为 $|1011| = 4$。

符号串中可以不包含任何符号，此时称其为空符号串或空串，用字母 ε 表示，其长度为

0，即 $|\varepsilon|=0$。ε 是一个特殊的字符串，对任一字母表 Σ，ε 均是其上的符号串。空字符串有时也用 λ 来表示。值得注意的是集合 $\{\varepsilon\}$ 并不等于空集 $\varnothing$，前者是包含一个元素 ε 的非空集合，而后者是不包含任何元素的空集。

只有当两个符号串长度相等且相应位置的符号均相同时，这两个符号串才是相等的。

下面介绍一些关于符号串的运算。

定义 2.3 符号串的连接

设 x 和 y 均是字母表 Σ 上的符号串，x 和 y 的连接 xy 是把 y 的所有符号顺序地接在 x 的符号之后所得到的符号串。例如，设 x = abc，y = de，则它们的连接 xy = abcde。容易看出 $|xy|=|x|+|y|=5$。由于 ε 是不包含任何符号的字符串，显然有 $\varepsilon x = x\varepsilon = x$。但连接运算一般不满足交换律。

定义 2.4 符号串的幂

设 x 是字母表 Σ 上的符号串，把 x 自身连接 n 次得到的符号串 z，即 z = xx…xx（n 个 x），称作符号串 x 的 n 次幂，记作 $z=x^n$。根据定义有

$x^0=\varepsilon$

$x^1=x$

$x^2=xx$

$x^3=x^2x=xx^2=xxx$

…

$x^n=x^{n-1}x=xx^{n-1}=xx\cdots xx$（n 个 x）

例 2.1 设 x = 001，则有 $x^0=\varepsilon$，$x^2=001001$，$x^3=001001001$。

定义 2.5 前缀和后缀

设 x、y、z 是某一字母表上的符号串，x = yz，则 y 是 x 的前缀，z 是 x 的后缀，特别是当 $z\neq\varepsilon$ 时，y 是 x 的真前缀；$y\neq\varepsilon$ 时，z 是 x 的真后缀。

例 2.2 设 x = abc，则 ε、a、ab 及 abc 都是 x 的前缀，其中 ε、a 和 ab 为 x 的真前缀；而 abc、bc、c 及 ε 都是 x 的后缀，其中 bc、c 和 ε 为 x 的真后缀。

定义 2.6 子字符串

一个非空字符串 x，删去它的一个前缀和一个后缀后所得到的字符串称为 x 的子字符串，简称子串。如果删去的前缀和后缀不同时为 ε，则称该子串为真子串。

例如，设 x = abc，其子串为 abc、ab、bc、a、b、c、ε，真子串为 ab、bc、a、b、c、ε。值得注意的是 ac 并不是 x 的子串。

定义 2.7 符号串集合

若集合 A 中的所有元素都是某字母表上的符号串，则称 A 为该字母表上的符号串集合。

下面定义几个符号串的集合运算。

定义 2.8 符号串集合的乘积

设 A、B 是两个符号串集合，AB 表示 A 与 B 的乘积，具体定义为

$$AB=\{xy\mid(x\in A)\wedge(y\in B)\}$$

运算结果仍然是符号串的集合。

例 2.3 设 A = {a,bc}，B = {de,f}，则

$$AB=\{ade,\ af,\ bcde,\ bcf\}$$

注意有 $\{\varepsilon\}A=A\{\varepsilon\}=A$，$\varnothing A=A\varnothing=\varnothing$，其中，$\varnothing$为空集。符号串集合的乘积一般不满足交换律。

定义 2.9 符号串集合的幂

设 A 是符号串集合，则称 A^i 是符号串集合 A 的幂。其中，i 是非负整数。具体定义如下：

$A^0=\{\varepsilon\}$

$A^1=A$

$A^2=AA$

…

$A^n=AA\cdots A$（n 个 A）

定义 2.10 符号串集合的正闭包

设 A 是符号串集合，则称 A^+ 为符号串集合 A 的正闭包。其具体定义如下：

$$A^+=A^1\cup A^2\cup A^3\cdots$$

定义 2.11 符号串集合的星闭包

设 A 是符号串集合，则称 A^* 为符号串集合 A 的星闭包。其具体定义如下：

$$A^*=A^0\cup A^1\cup A^2\cup A^3\cdots$$

星闭包又称自反闭包或 Kleene 闭包。

由定义显然有 $A^+=AA^*$。

例 2.4 设 $A=\{ab, cd\}$，则

$A^+=\{ab, cd, abab, abcd, cdab, cdcd, ababab, ababcd, \cdots\}$

$A^*=\{\varepsilon, ab, cd, abab, abcd, cdab, cdcd, ababab, ababcd, \cdots\}$

2.2 文法

给定字母表∑，则∑上任一字符串集合，是∑上的一个语言，记为 L。该语言的每一个字符串，是语言 L 的一个语句或句子。由有限个字符串组成的集合为一有穷语言，反之为无穷语言。

例如，正确的英文句子的集合是定义在 26 个英文字母表上的语言，按照 C 语言的语法正确构造出来的 C 程序集合是定义在 C 字符集上的语言。还有一些抽象的语言，如空集∅和只包含一个空串的集合 $\{\varepsilon\}$ 也都是符合此定义的语言。

文法是描述语言的语法结构的形式规则（亦称文法规则、产生式或重写规则等）。文法规则必须是准确且是可理解的。

2.2.1 文法的定义

定义 2.12 文法

一个文法（Grammar）G 是一个四元组：$G=(V_N, V_T, S, P)$，其中：

V_T：一个非空有限的终极符号集合，它的每个元素称为终极符号或终极符，一般用小写字母表示。从语法分析的角度看，终极符号是一个语言不可再分的基本符号。

V_N：一个非空有限的非终极符号集合，它的每个元素称为非终极符号或非终极符，一