高等学校“十二五”重点规划教材
信息与自动化系列

编译原理

主编 张晶

哈尔滨工程大学出版社
Harbin Engineering University Press

编译原理

主　编　张　晶
副主编　陈香凝　董德义
　　李　鑫　赵　曦

哈尔滨工程大学出版社

内容简介

本书系统、全面地介绍了编译程序的基本原理、实现技术、方法和工具。在全书的安排上，首先介绍了编译程序的定义、结构、组织以及文法和语言的相关知识；然后以编译程序的结构为主线介绍了词法分析、语法分析、语法制导翻译与中间代码生成、代码优化和目标代码生成。另外，本书还涉及到运行时的存储管理以及符号表，并对词法分析、语法分析程序的生成工具 LEX，YACC 以及面向对象语言的编译进行了简要介绍。在基本概念和基本理论的阐述上做到准确清楚，在内容的组织上做到循序渐进、深入浅出、通俗易懂，精心挑选的习题可以使学习者对自己的掌握情况有较好的把握。

本书可以作为计算机专业的大专生、本科生和研究生的教材，也可以作为计算机科技工作者的参考书。

图书在版编目(CIP)数据

编译原理/张晶主编. —哈尔滨：哈尔滨工程大学出版社，2011.8
ISBN 978-7-5661-0115-0

Ⅰ.①编… Ⅱ.①张… Ⅲ.①编译程序-程序设计 Ⅳ.①TP314

中国版本图书馆 CIP 数据核字(2011)第 162263 号

出版发行	哈尔滨工程大学出版社
社　　址	哈尔滨市南岗区东大直街 124 号
邮政编码	150001
发行电话	0451-82519328
传　　真	0451-82519699
经　　销	新华书店
印　　刷	哈尔滨市石桥印务有限公司
开　　本	787mm×1 092mm　1/16
印　　张	16.25
字　　数	393 千字
版　　次	2011 年 8 月第 1 版
印　　次	2011 年 8 月第 1 次印刷
定　　价	32.00 元

http://press.hrbeu.edu.cn
E-mail:heupress@hrbeu.edu.cn

前　言

编译原理是计算机学科的一门重要专业基础课，是一门理论性、实践性、技术性很强的课程，强调理论与实践的紧密结合。设置本课程的目的在于系统地讲述程序设计语言编译程序构造的基本原理和设计方法，通过本课程的学习，一方面使学生掌握和理解编译系统的结构、工作流程，以及编译程序各组成部分的设计原理和实现技术；另一方面，通过学习编译的理论和方法，提高学生对知识的综合理解能力，使学生更好地理解程序语言的内部机制，培养学生初步掌握设计大型系统软件的方法、技术，以及设计大型软件的能力。

全书共分 14 章，第 1 章主要介绍了编译程序的结构和组织；第 2 章介绍了文法的定义和分类等；第 3 章主要介绍了有穷自动机、正规文法、正规式、正规集以及它们之间的相互转换，此外还介绍了词法分析器的设计等；第 4 章介绍了自顶向下的语法分析，主要包括 LL(1)分析法和递归下降分析法；第 5 章介绍了自底向上分析中的优先分析法，简要介绍了简单优先分析法，详细介绍了算符优先分析法；第 6 章介绍了自底向上语法分析的 LR 分析法，其中主要是对 LR(0)，SLR，LALR 以及 LR(1)文法的详细介绍；第 7 章介绍了语法制导翻译和中间代码生成，主要是对说明语句以及各种可执行语句的翻译；第 8 章介绍了运行时的存储管理，主要是静态、栈式和堆式分配；第 9 章介绍了符号表的作用、组织等；第 10 章介绍了代码优化，包括局部、循环和全局优化等；第 11 章介绍了目标代码生成，其中涉及目标代码的形式以及寄存器的分配等；第 12 章介绍了词法分析程序生成器 LEX；第 13 章介绍了语法分析程序生成工具 YACC；第 14 章介绍了面向对象语言的编译。

本书第 2，3，4，5，6，7，10，11 章由张晶编写，第 8，12 章由陈香凝编写，第 9，13 章由董德义编写，第 14 章由李鑫编写，第 1 章由赵曦编写。鉴于编者水平有限，时间仓促，本书稿虽经审慎校阅，仍难免存在疏误，殷切希望广大读者批评指正。

本书参考了国内外一些专著和资料，参考、借鉴了一些专家、学者的研究成果，编者在此对所有这些前辈和同行的引导和帮助表示衷心感谢。

编　者

2011 年 4 月

目　录

第1章 引 论

1.1 什么是编译程序

程序设计语言是人与计算机进行联系的工具，人们可以通过它来控制计算机去按照人的意志进行各种运算和操作。通常，程序设计语言可以划分为两大类：低级语言和高级语言。

低级语言包括机器语言、汇编语言以及其他面向机器的程序设计语言。它对计算机的依赖性强、直观性差，而且编写程序的工作量也很大，只有对相应计算机的结构比较熟悉且经过一定训练的程序人员才能较好地使用它。与低级语言相比，高级语言独立于机器，比较接近自然语言，因而易于学习和掌握。高级语言不论在算法描述的能力上，还是在编写和调试程序的效率上，都远比低级语言优越，用它编写出的程序易读、易理解、易修改和易移植。

高级语言虽然优越，但计算机硬件却只懂得自己的指令系统，即只能直接执行相应机器语言格式的代码程序，而不能直接执行用高级语言或汇编语言编写的程序。因此，要在计算机上实现除机器语言之外的任意程序设计语言，就必须先使此种语言为计算机所理解，也就是必须经过等价变换，使它成为计算机能理解与执行的机器语言的程序，而这种等价变换的工作就是由翻译程序来完成的。

翻译程序是以某一种程序设计语言所编写的程序作为翻译或加工的对象，将它翻译成与之等价的另一种语言的程序。翻译前的程序称为源语言程序，简称源程序，翻译后的程序称为目标语言程序，简称目标程序。汇编程序和反汇编程序都是翻译程序。其中，汇编程序是把汇编语言翻译为机器语言的翻译程序，在这里，源语言就是汇编语言，而目标语言就是机器语言。反汇编程序恰好相反，它是将机器语言翻译为汇编语言的翻译程序，所以，源语言是机器语言，目标语言是汇编语言。如果一个翻译程序的源语言是某种高级语言，其目标语言是相应于某一计算机的汇编语言或机器语言，则称这种翻译程序为编译程序，也就是说，编译程序是从高级语言到汇编语言或机器语言的翻译程序。用图形表示编译程序的功能如图 1－1 所示。

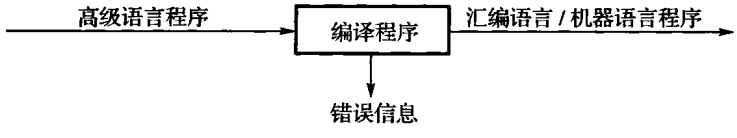

图 1－1 编译程序的功能

编译程序是现代计算机系统的基本组成部分之一，从功能上看，一个编译程序就是一个语言翻译程序，而一个编译程序的重要性体现在它使得多数计算机用户不必考虑与机器有

关的繁琐细节。

按编译方式执行一个用高级语言编写的程序时，通常经过两个阶段，即编译阶段和运行阶段。编译阶段的任务是将源程序变换为目标程序，若目标程序不是机器代码，而是汇编语言程序，则还要由汇编程序将其编译为机器代码；运行阶段的任务就是在目标计算机上执行编译阶段所得到的目标程序。通常，在执行目标程序时还要有一些子程序来配合进行工作，这些子程序组成一个子程序库，称为运行系统。在图 1－2 中，粗略地给出了按编译方式执行程序的工作过程。

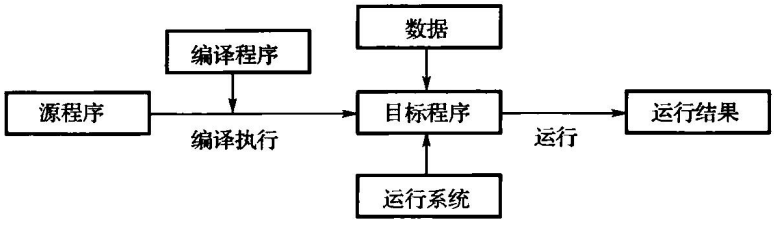

图 1－2　按编译方式执行程序的过程

解释程序也是从高级语言到汇编语言或机器语言的翻译程序，它是将源程序中的语句逐句翻译成可执行代码，一旦具备执行条件，就立即将一段代码执行得到结果。解释程序与编译程序的主要区别是在解释程序的执行过程中不产生目标程序，而编译程序要先把全部程序翻译为目标程序，然后再执行，而且目标程序可以反复执行。相对地，由于解释程序对源程序逐句地翻译执行，所以只能执行一次，若需重新执行，则必须重新解释源程序。在图 1－3 中，粗略地给出了按解释方式执行程序的工作过程。

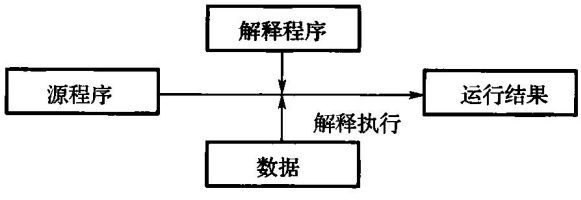

图 1－3　按解释方式执行程序的过程

解释程序的主要优点是便于对源程序进行调试和修改，但其加工和处理过程的速度较慢，效率不如编译程序高。根据编译程序和解释程序的优缺点，可以把编译和解释作某种程度的结合。

1.2　编译程序的基本结构

编译程序的主要功能是将源程序翻译成等价的目标程序，这个过程是一个复杂的整体的过程。从概念上来讲，一个编译程序的整个工作过程是划分成阶段来进行的，每个阶段将源程序的一种表示形式转换成另一种表示形式，各个阶段进行的操作在逻辑上是紧密联结

的,图 1-4 给出了一个编译过程的各个阶段,这是一种典型的划分方法。它由词法分析、语法分析、语义分析、中间代码生成、代码优化和目标代码生成阶段来实现,而且在以上各个阶段中,都涉及到符号表管理和出错管理。

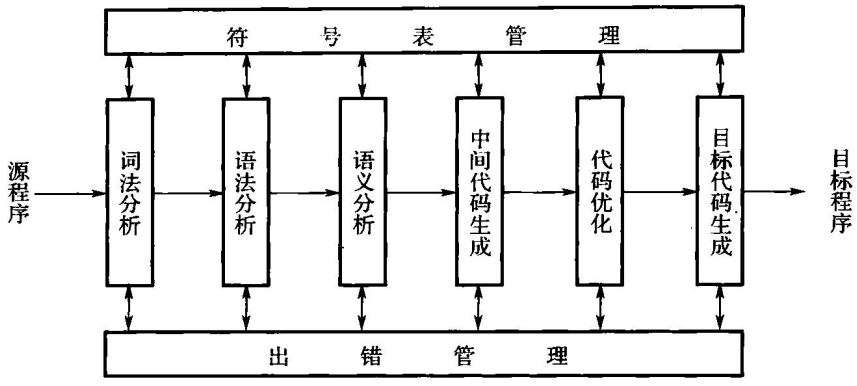

图 1-4 编译程序的逻辑结构示意图

1.2.1 词法分析

词法分析是编译过程的基础,其主要任务是从左到右一个字符一个字符地读入源程序,按照源语言规定的词法规则,对构成源程序的字符流进行扫描和分解,从而识别出一个个单词,并把它们表示成机内单词形式。这里所谓的单词是指逻辑上紧密相连的一组字符,编译程序把它当作源语言的最小单位。例如,标识符、关键字、运算符、常数等都是单词。在对下面的 C 语言赋值语句进行词法分析之后,就可以识别出 8 个单词,它包含标识符“sum,a,b,c”,两个算术运算符“+”,赋值运算符“=”,界符“;”。

```
sum = a + b + c;
```

同样的道理,下面的 if 语句经过词法分析后,就可以识别出 20 个单词,其中包括关键字、界符、标识符、赋值运算符和关系运算符。

```
if (a > b) {t = a; a = b; b = t;}
```

机内单词形式,也称为 TOKEN 字,一种经常使用的方法是采用形如(Class,Value)的长度统一的二元形式(即序偶对)来表示。其中,Class 指示该单词的类别,如关键字、标识符、常数等,而 Value 则指出单词的自身值。

当然,词法分析在识别单词的过程中,同时也滤除源程序中不要的符号,如注释、多余空格、回车字符等。此外,扫描器还进行词法检查,指出源程序中的单词错误,有时还要求把识别出来的各种名字填入符号表,以备后续阶段使用。

1.2.2 语法分析

语法分析是在词法分析的基础上进行的,它的主要任务是根据源语言的语法规则把单词序列分解成各类语法单位,如表达式、语句、程序等,并指出其中的语法错误。语法规则描述了该语言的各种语法成分的组成结构,可以用上下文无关文法或与之等价的 Backus-Naur 范式将一个程序设计语言的语法规则确切地描述出来。通常,语法单位可表示成语法树。

语法分析依据的是程序结构的规则，通过语法分析，可以确定整个输入符号串是否构成一个语法上正确的程序。为了完成这种分析，一般的途径是由语法分析程序试着为其构造一棵完整的语法树，如果这种尝试成功，就表明该输入串在结构上的确是一个合乎语法的程序，否则，源程序中就必然存在语法错误。

应当指出，在实际进行语法分析的过程中，并不一定真正为源程序建立一个树形的数据结构，也不一定需要对整个源程序产生一棵语法树，而是参照建立语法树的思路一步一步地进行语法分析。

1.2.3 语义分析

在编译过程中需要对源程序进行语义分析，因为对任何一种程序设计语言来说，它都具有两方面的特征，即语法特征和语义特征。语法特征用来定义语言各语法成分的形式或结构，语义特征则用来规定各语法成分的含义和功能，即规定它们的属性或在执行时应进行的运算或操作。

在进行语义分析的过程中需要进行相应的语义检查，以保证源程序在语义上的正确性。通常，所需检查的项目是非常繁杂的。例如，在说明语句中是否有矛盾的类型说明；在过程调用中，实在参数与形式参数是否在个数、次序、种属等方面按相应语言的规定进行对应等。类型检查是一个重要部分，它会检查每个算符是否具有语言规范允许的运算对象，当不符合语言规范时，编译程序应报告错误。例如，有的编译程序要对实数用作数组下标的情况报告错误。

1.2.4 中间代码生成

中间代码生成阶段的任务是在语法分析和语义分析的基础上，根据语法成分的语义对其进行翻译，并且把它译成在语义上等价的中间代码语言。

中间代码生成对于编译程序来说并不是必不可少的阶段，通常为了处理上的方便，特别是为了便于代码的优化处理，在语义分析后不直接产生机器语言或汇编语言形式的目标代码，而是生成一种介于源语言和目标语言之间的中间语言代码。一般来说，这种中间表示应有两个重要的性质：一个是容易生成；另一个是容易把它翻译为目标代码。目前，常见的中间代码形式有逆波兰式、三元式和四元式等。

例如，对于 C 语言表达式 a + b * c - d + f/e 来说，它的逆波兰式可以写成 abc * + d - fe/ + 。当然，如果采用四元式作为中间代码，则可以表示为：

```
( * ,b ,c ,T1)
( + ,a ,T1,T2)
( - ,T2,d ,T3)
(/,f , e ,T4)
( + ,T3,T4,T5)
```

中间代码的产生与语义分析紧密相连，对于采用语法制导翻译的编译程序，通常的做法是将产生中间代码的工作交给语义过程来完成，即每当一个语义过程被调用而对相应的语法结构进行语义分析时，它就根据此语法结构的语义，并结合在分析时所获得的语义信息，产生相应的中间代码，再把后者放到中间代码的序列中去。这里所说的语法制导翻译的方法是把编译程序的语法分析和语义分析有机地组织起来，穿插地进行。

1.2.5 代码优化

代码优化不是编译程序的必要阶段，通常为了得到质量较高的目标代码，在中间代码生成和目标代码生成两个阶段之间设置代码优化。代码优化的主要任务是对中间代码进行改造、变换，目的是使生成的目标代码更为高效，即省时间和省空间。编译程序所产生的目标代码质量的高低，主要取决于这一阶段里代码优化程序功能的强弱。

代码优化所涉及的范围很广，如果从与具体计算机的关系上看，可以分为与机器有关的优化和与机器无关的优化。与机器有关的优化主要涉及如何分配寄存器，这种优化是在生成目标代码时进行的。与机器无关的优化是对中间代码程序的优化，常用的优化措施有公共子表达式的删除、合并已知量、循环优化等。其中，有些优化措施效果很明显，例如循环中参与运算的运算量，如其值并不随着循环而发生变化，这类运算称为循环不变运算，它完全不必每次循环都计算一次，可以将它们提到循环前计算一次即可。

不同的编译器完成不同程度的优化，能完成大多数优化的编译器叫做“优化编译器”，但是编译的相当一部分时间消耗在优化上。简单的优化也可以使目标程序的运行时间大大缩短，而编译速度并没有降低太多。

应当指出，对于一个进行优化处理工作的编译程序，虽然它在工作时可得到质量较高的目标程序，然而却是以增加编译程序本身的时空复杂度和可靠性为代价的。另外，也有这样一些优化项目，它们在时间效率和空间效率上是互相矛盾的。因此，在设计一个编译程序时，究竟应考虑哪些优化项目以及各种优化项目进行到何种程度，应权衡利弊，根据具体情况而定。

1.2.6 目标代码生成

目标代码生成是编译程序的最后阶段，如果编译程序采用了中间代码，那么目标代码生成阶段的任务就是将中间代码或优化后的中间代码转换为等价的目标代码，即机器指令或汇编指令。如果对源程序的编译不设置中间代码生成阶段，则在语法和语义分析之后将直接产生目标代码。目标代码依赖于具体计算机的硬件系统结构和指令系统，所以目标代码生成涉及到如何选择并充分利用寄存器，合理选择指令和生成尽可能短而有效的目标代码，等等。

由于目标程序总是按某一具体计算机的机器语言或汇编语言来产生，因此在设计目标代码生成程序时，首先要确定源语言的各种语法成分的目标代码结构，然后要针对不同的情况，制定从中间代码到目标代码的翻译策略或算法，并据此编写出目标代码生成程序。在制定此种策略或算法时，总的要求是所生成的目标代码要有较高的执行效率。所以，应该使所生成的目标代码尽可能短，并充分发挥计算机可用资源的效率，即尽量使用执行速度快的指令，充分利用计算机的寄存器，以节省访问内存所用的时间，等等。

通常目标代码可采用如下三种形式之一：汇编语言形式的目标程序、绝对形式或相对形式的机器码程序。如果生成的目标代码是汇编语言程序，则须经由汇编程序汇编后，产生机器代码才能执行；如果生成的目标代码是具有绝对地址的机器指令代码，则可以直接投入运行；如果是具有相对地址的机器指令代码，则必须经连接装配程序将它们和另外一些运行子程序连接装配后才能投入运行。

1.2.7 出错管理

程序人员在编写程序时,存在错误是难免的,编译的各个阶段都可能遇到错误,例如,词法分析可能发现字符拼写错误,又如,两个标识符进行算术运算,一个为变量名,另一个为函数名,虽然语法上允许,但语义上不允许。诸如此类的各种错误,都应在相应的阶段进行处理。一个仅能处理绝对正确源程序的编译程序并无实用价值,较完善的编译程序应具有广泛的程序查错能力,并能准确地报告源程序中错误出现的位置及错误的种类,以便程序人员修改。

出错管理程序负责发现源程序中可能出现的错误,并把错误报告给用户,指出错误的性质和发生错误的位置。为了最大限度地发现源程序中的错误,应将错误所造成的影响限制在尽可能小的范围之内,使得编译程序能继续编译源程序的余下部分,以便能查出尽可能多的错误让用户纠正。当然,发现错误后能自动校正错误最好,只是在大部分情况下,校正的代价太大。

1.2.8 符号表管理

在编译过程中,源程序的相关信息保留在各种表格中,这就涉及到相应表格的构造。对于各种不同用途的表格,通常统称为符号表。除了对表格进行构造,在编译的各个阶段还要查找或更新相关的表格,这些任务都是由编译程序中的符号表管理程序来完成的。

编译器的一项重要工作就是记录源程序中使用的标识符和收集每个标识符的各种属性,符号表会为每个标识符保存一个记录的数据结构,记录的域是标识符的属性,所以可以在记录中存储和读取标识符的相关信息。如词法分析程序发现源程序中的标识符时,就会把该标识符填入符号表。通常词法分析期间不能确定一个标识符的各种属性,标识符的信息是在不同的阶段填入符号表的,而且是以不同的方式使用这些信息。例如,语义分析和中间代码生成需要知道标识符的类型,才能检查出源程序是否以合法的方式使用它们。

在编译时根据不同的编译方法,还将保持一些专用表格,如 LL 分析表、LR 分析表等,这些表格会配合相应的编译算法进行工作。由于在编译的各个阶段都必须进行频繁的造表和查表工作,而且这些工作将占用相当一部分编译时间,所以合理地组织编译程序中的各种表格,并恰当地选用相应的造表和查表算法,是提高编译程序工作效率的有效途径之一。

1.3 编译程序的组织

上一节介绍了根据一个编译程序应具有的功能,把编译程序划分为八个组成部分,即词法分析、语法分析、语义分析、中间代码生成、代码优化、目标代码生成、符号表管理和出错管理,这种划分是编译程序的逻辑组织方式。有时也把编译的过程分为前端和后端两个部分,前端主要依赖于源语言,它由几乎独立于目标机器的阶段或阶段的一部分组成。通常这些阶段包括词法分析、语法分析、语义分析和中间代码生成,也包括与前端每个阶段相关的符号表管理和出错管理,某些与目标机无关的优化工作也可以在前端完成。后端是编译器中依赖于目标机器的部分,它一般独立于源语言而与中间语言有关,后端包括代码优化中涉及目标机的部分、代码生成和与这些阶段相关的出错处理和符号表管理。

把编译过程分为前端和后端的这种组织方式便于编译程序的移植，若要将编译程序移植到不同类型的机器，只需修改编译程序的后端即可。也就是说，某一编译程序的前端加上相应不同的后端可以为不同的机器构成同一个源语言的编译程序。另外，不同语言编译的前端生成同一种中间语言，再使用一个共同的后端，则可以为同一个机器生成几个语言的编译程序。

编译过程还可以采用“分遍”的形式，即编译过程可以由一遍或多遍来完成。上一节给出了编译程序的逻辑结构，它描述了各个阶段的逻辑关系，但不一定是执行时间上的先后顺序。事实上，可以按不同的执行流程来组织各部分的工作。这在很大程度上依赖于编译过程中对源程序扫描的遍数，以及如何划分各遍扫描所进行的工作。

在编译过程中，对源程序或其等价的内部表示从头到尾扫视，并进行相应的加工处理，进而完成规定任务的过程称为编译的一遍，也称“趟”。有很多不同的方式可以把编译器的阶段组成遍，一个编译过程可由一遍、两遍或多遍完成，每一遍扫视可完成上述一个或多个阶段的工作。例如，一遍可以只完成词法分析工作（显然，编译的各个阶段都可以单独作为一遍）；一遍也可以完成词法分析和语法分析两个阶段的工作；如果将词法分析作为一个子程序，语法分析程序每当需要单词时便调用词法分析程序获得一个单词，每当按语法规则发现形成一个短语时，便调用相应的语义子程序做翻译工作，生成中间代码，则可以将词法分析、语法分析、语义分析以及中间代码生成组成一遍，这样整个编译过程就可以用两遍或三遍来完成。当然，也可以一遍完成整个编译工作。在图 1－5 中，给出了以语法分析程序为中心的一遍的编译程序。

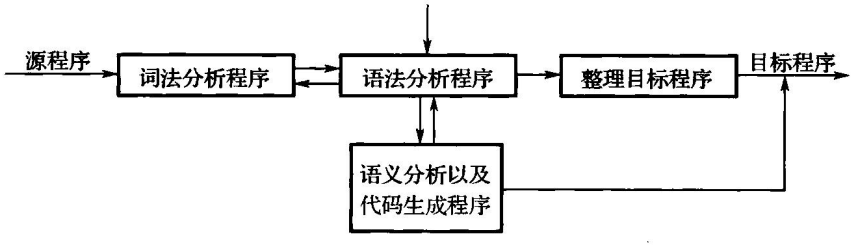

图 1－5　以语法分析程序为中心的编译程序逻辑结构

对于多遍的编译程序，第一遍的输入是用户书写的源程序，最后一遍的输出是目标语言程序，其余都是上一遍的输出为下一遍的输入。

编译过程分为多遍来完成，最初是由于内存小且比较贵，将编译分为多遍，每遍只完成一部分工作，可以使编译程序的结构比较清晰。当然，并不是遍数越多越好，它也有时间开销。因为遍数多就意味着增加读写中间文件的次数，这样就会消耗较多的时间，所以在内存允许的情况下，减少编译的遍数可以加快编译速度。当然，一遍虽然速度快，但有空间开销。

在设计一个编译程序时，如何确定扫描遍数，如何组织各遍中的工作，主要取决于源语言的具体情况及编译程序运行的具体环境，如语言的结构、计算机各种软硬件的配置，以及对编译程序本身运行效率的要求，等等。

习　题

1.1　何谓翻译程序？为什么高级程序设计语言需要翻译程序？

1.2　何谓编译程序，何谓解释程序，它们之间的区别是什么？

1.3　通常一个编译程序由哪几个部分组成，各部分的主要任务是什么？

1.4　简述编译程序的组织方式。

第2章　文法和语言

为了使编译工作有效地进行，首先应确切地描述或定义一种程序设计语言。人们把用一组数学符号和规则来描述语言的方式称为形式描述，而把所用的数学符号和规则称为形式语言。程序语言是形式化的语言，文法则是程序语言的生成系统。本章将初步介绍形式语言中的一些基本概念和知识。

2.1　字母表和符号串

程序设计语言是由程序所组成的集合，程序又是由一些基本符号所组成的，每个程序都是一个“基本符号”串，程序设计语言可以看成是在基本符号集上定义的、按一定规则构成的一切基本符号串组成的集合。为了给出语言的形式定义，首先给出一些基本概念和术语。

(1)字母表　字母表是由若干个元素组成的有穷非空集合，习惯上用Σ或其他大写字母表示。不同的语言可以有不同的字母表，字母表中的元素称为符号，所以字母表也称为符号集。例如，汉语的字母表有汉字、数字及标点符号等，$\Sigma=\{a,b,c\}$是一个含有a，b和c三个元素的字母表。

(2)符号串　字母表中的符号所组成的任何有穷序列称为符号串，简称串。一个字母表上的全部符号串所组成的集合是一个无穷集合。例如，0，1，00，100和111等都是字母表$\Sigma=\{0,1\}$上的符号串。

符号串x中所含符号的个数称为符号串x的长度，通常用$|x|$表示。例如，符号串10101的长度为5，abc的长度为3，它们分别记为$|10101|=5$和$|abc|=3$。符号串中比较特殊的一个串是空符号串，空符号串是不包含任何符号的符号串，简称空串，通常用ε表示。空串的长度为0，即$|\varepsilon|=0$。对任一字母表Σ，ε都是Σ上的字符串。

(3)符号串的前缀、后缀　设x是一个符号串，x的前缀就是对符号串x自左向右截取若干个($0\sim|x|$个)连续字符得到的符号串。同理，x的后缀就是对符号串x自右向左截取若干个($0\sim|x|$个)连续字符所得的符号串。例如，若x=abc，则x的前缀为ε，a，ab及abc；x的后缀为ε，c，bc及abc。

(4)符号串的子串　去掉一个前缀和一个后缀之后，所余下的部分称为该符号串的子串。例如，符号串abc的子串有ε，a，b，c，ab，bc及abc。可见，一字符串的任何前缀和后缀都是该字符串的子串，但一字符串的子串不一定是该字符串的前缀或后缀。

(5)符号串的连接和方幂　设x和y是两个符号串，则xy被称为符号串x和y的连接，即自左向右取出y的每一个符号并依次置于x的尾部。例如，若x=abc，y=123，则xy=abc123，yx=123abc，xx=abcabc。显然，由ε的定义有$\varepsilon x=x\varepsilon=x$。

符号串与其自身的$n-1(n>0)$次连接称为该符号串的n次方幂，设x为一符号串，则有

$x^0=\varepsilon$

$x^1=x$

$$x^2 = xx$$
$$x^3 = x^2x = xx^2 = xxx$$
$$\vdots$$
$$x^n = x^{n-1}x = xx^{n-1} = \underbrace{x\ x\cdots x}_{n\text{个}}$$

例如,若 x=01,则 $x^0=\varepsilon$,$x^1=01$,$x^2=0101$,$x^3=010101$。

(6)符号串集合　若集合 A 中的一切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。

(7)符号串集合的积　设 A,B 为两个符号串的集合,则 A 与 B 的乘积定义如下:

$$AB = \{xy \mid x \in A \text{ 且 } y \in B\}$$

例如,若有符号串集合 A 和 B,分别为 A={ab,cd},B={0,1},则 AB={ab0,ab1,cd0,cd1}。

需要注意的是,$\{\varepsilon\}A = A\{\varepsilon\} = A$,$\varnothing A = A\varnothing = A$,其中∅为空集;$\varepsilon \notin \varnothing$,即空串不属于空集。

(8)符号串集合的和　设 A,B 为两个符号串的集合,则 A 与 B 的和记为 A∪B(或 A+B),且定义为:

$$A \cup B = \{x \mid x \in A \text{ 或 } x \in B\}$$

即 A∪B 中的符号串或者属于符号串 A,或者属于符号串 B。例如,若有符号串集合A={ab,cd},B={0,1},则 A∪B={ab,cd,0,1}。

对于符号串集合 A,显然有 A∪∅=∅∪A=A。

(9)符号串集合的方幂　符号串集合 A 的方幂运算可定义为:

$$A^0 = \{\varepsilon\}$$
$$A^1 = A$$
$$A^2 = AA$$
$$A^3 = A^2A = AA^2$$
$$\vdots$$
$$A^n = A^{n-1}A = AA^{n-1}(n>0)$$

(10)符号串集合的闭包　设 A 为符号串集合,则 A 的闭包定义为:

$$A^* = A^0 \cup A^1 \cup \cdots \cup A^n \cdots$$

A 的正闭包定义为

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

例如,若 A={ab,01},则有:

$$A^0 = \{\varepsilon\}$$
$$A^1 = \{ab,01\}$$
$$A^2 = \{abab,ab01,01ab,0101\}$$
$$\vdots$$
$$A^+ = \{ab,01,abab,ab01,01ab,0101,\cdots\}$$
$$A^* = \{\varepsilon,ab,01,abab,ab01,01ab,0101,\cdots\}$$

显然

$$A^* = \{\varepsilon\} \cup A^+ = A^+ \cup \{\varepsilon\} = A^0 \cup A^+ = A^+ \cup A^0$$