高等学校
电子信息类

规划教材

程序设计语言
编译原理

（第3版）

陈火旺　　刘春林
谭庆平　赵克佳　刘　越　　编著

国防工业出版社
National Defense Industry Press

程序设计语言
编 译 原 理

（第 3 版）

陈火旺　刘春林
谭庆平　赵克佳　刘　越　编著

国防工业出版社
·北京·

图书在版编目(CIP)数据

程序设计语言编译原理/陈火旺等编著. —3 版. —北京:国防工业出版社,2009.6 重印

ISBN 978-7-118-02207-0

Ⅰ.程... Ⅱ.陈... Ⅲ.编译程序-程序设计 Ⅳ. TP314

中国版本图书馆 CIP 数据核字(1999)第 68630 号

※

国防工业出版社出版发行

(北京市海淀区紫竹院南路 23 号 邮政编码 100048)

腾飞印务有限公司印刷

新华书店经售

*

开本 787×1092 1/16 **印张** 25 **字数** 575 千字

2009 年 6 月第 50 次印刷 **印数** 490051—500100 册 **定价** 31.00 元

(本书如有印装错误,我社负责调换)

国防书店:(010)68428422 发行邮购:(010)68414474

发行传真:(010)68411535 发行业务:(010)68472764

出 版 说 明

为做好全国电子信息类专业“九五”教材的规划和出版工作，根据国家教委《关于“九五”期间普通高等教育教材建设与改革的意见》和《普通高等教育“九五”国家级重点教材立项、管理办法》，我们组织各有关高等学校、中等专业学校、出版社，各专业教学指导委员会，在总结前四轮规划教材编审、出版工作的基础上，根据当代电子信息科学技术的发展和面向21世纪教学内容和课程体系改革的要求，编制了《1996—2000年全国电子信息类专业教材编审出版规划》。

本轮规划教材是由个人申报，经各学校、出版社推荐，由各专业教学指导委员会评选，并由我部教材办商各专指委、出版社后，审核确定的。本轮规划教材的编制，注意了将教学改革力度较大、有创新精神、特色风格的教材和质量较高、教学适用性较好、需要修订的教材以及教学急需，尚无正式教材的选题优先列入规划。在重点规划本科、专科和中专教材的同时，选择了一批对学科发展具有重要意义，反映学科前沿的选修课、研究生课教材列入规划，以适应高层次专门人才培养的需要。

限于我们的水平和经验，这批教材的编审、出版工作还可能存在不少缺点和不足，希望使用教材的学校、教师、同学和广大读者积极提出批评和建议，以不断提高教材的编写、出版质量，共同为电子信息类专业教材建设服务。

电子工业部教材办公室

前　言

本教材系按电子工业部的《1996—2000年全国电子信息类专业教材编审出版规划》，由全国高校计算机专业教学指导委员会编审、推荐出版。本教材由国防科技大学陈火旺院士担任主编，主审侯文永教授、赵雄芳教授。

本教材的参考学时数80学时，其主要内容包括词法分析、语法分析、属性文法与语法制导翻译、语义分析与中间代码产生、符号表与运行时存储空间组织、优化与目标代码生成、并行编译技术。本书将编译技术的最新发展，例如属性文法、面向对象语言的编译技术、并行编译技术、编译程序自动构造工具等内容系统地融合到教材中。本书的主要例题和习题均以C、Pascal为语言背景，并在一些重要的章节中增加了必要的例题，以帮助读者理解和自学。使用本教材时应注意，在学这门课之前，学生必须预修计算引论（程序设计方法）和高级语言（PASCAL、C或C++），并且最好具有数据结构和离散数学方面的基本知识。

本书是在陈火旺、钱家骅、孙永强三位教授1980年编写的《程序设计语言编译原理》的基础上，结合编译技术的最新研究成果和作者多年的教学经验编写而成的。本书由陈火旺院士确定内容的选取和组织，由刘春林、谭庆平、赵克佳、刘越具体执笔，最后由陈火旺院士定稿。刘春林编写第一、二、四、五、六、七、十章以及十一章部分内容；谭庆平编写第三章；刘越编写第八、九章以及十一章部分内容；赵克佳编写第十二章；翟桂英负责全书文字及图表的录入工作。侯文永教授和赵雄芳教授认真审阅了本书的全部初稿，提出了很多宝贵的意见，在此表示诚挚的感谢。由于编者水平有限，书中难免还存在一些缺点和错误，殷切希望广大读者批评指正。

编　者

目　录

第一章 引 论

1.1 什么叫编译程序

使用过现代计算机的人都知道,多数用户是应用高级语言来实现他们所需要的计算的。现代计算机系统一般都含有不止一个的高级语言编译程序,对有些高级语言甚至配置了几个不同性能的编译程序,供用户按不同需要进行选择。高级语言编译程序是计算机系统软件最重要的组成部分之一,也是用户最直接关心的工具之一。

在计算机上执行一个高级语言程序一般要分为两步:第一步,用一个编译程序把高级语言翻译成机器语言程序;第二步,运行所得的机器语言程序求得计算结果。

通常所说的**翻译程序**是指这样的一个程序,它能够把某一种语言程序(称为**源语言**程序)转换成另一种语言程序(称为**目标语言**程序),而后者与前者在逻辑上是等价的。如果源语言是诸如 FORTRAN、Pascal、C、Ada、Smalltalk 或 Java 这样的"高级语言",而目标语言是诸如汇编语言或机器语言之类的"低级语言",这样的一个翻译程序就称为**编译程序**。

高级语言程序除了像上面所说的先编译后执行外,有时也可"解释"执行。一个源语言的**解释程序**是这样的程序,它以该语言写的源程序作为输入,但不产生目标程序,而是边解释边执行源程序本身。本书将不对解释程序作专门的讨论。实际上,许多编译程序的构造与实现技术同样适用于解释程序。

根据不同的用途和侧重,编译程序还可进一步分类。专门用于帮助程序开发和调试的编译程序称为**诊断编译程序**(Diagnostic Compiler),着重于提高目标代码效率的编译程序叫**优化编译程序**(Optimizing Compiler)。现在很多编译程序同时提供了调试、优化等多种功能,用户可以通过"开关"进行选择。运行编译程序的计算机称宿主机,运行编译程序所产生目标代码的计算机称**目标机**。如果一个编译程序产生不同于其宿主机的机器代码,则称它为**交叉编译程序**(Cross Compiler)。如果不需重写编译程序中与机器无关的部分就能改变目标机,则称该编译程序为**可变目标编译程序**(Retargetable Compiler)。

世界上第一个编译程序——FORTRAN 编译程序是 20 世纪 50 年代中期研制成功的。当时,人们普遍认为设计和实现编译程序是一件十分困难、令人生畏的事情。经过 40 年的努力,编译理论与技术得到迅速发展,现在已形成了一套比较成熟的、系统化的理论与方法,并且开发出了一些好的编译程序的实现语言、环境与工具。在此基础上设计并实现一个编译程序不再是高不可攀的事情。

本书主要介绍设计和构造编译程序的基本原理和方法。我们不想罗列太多细节性的材料,着重讲一些原理性的东西,但将反映一些最新的进展。

1.2 编译过程概述

编译程序的工作,从输入源程序开始到输出目标程序为止的整个过程,是非常复杂的。但就其过程而言,它与人们进行自然语言之间的翻译有许多相近之处。当我们把一种文字翻译为另一种文字,例如把一段英文翻译为中文时,通常需经下列步骤:

(1) 识别出句子中的一个个单词;

(2) 分析句子的语法结构;

(3) 根据句子的含义进行初步翻译;

(4) 对译文进行修饰;

(5) 写出最后的译文。

类似地,编译程序的工作过程一般也可以划分为五个阶段:词法分析、语法分析、语义分析与中间代码产生、优化、目标代码生成。

第一阶段,**词法分析**。词法分析的任务是:输入源程序,对构成源程序的字符串进行扫描和分解,识别出一个个的单词(亦称单词符号或简称符号),如基本字(begin、end、if、for、while 等),标识符、常数、算符和界符(标点符号、左右括号等等)。例如,对于 Pascal 的循环语句

```
for I: = 1 to 100 do
```

词法分析的结果是识别出如下的单词符号:

基本字	for
标识符	I
赋值号	: =
整常数	1
基本字	to
整常数	100
基本字	do

这些单词是组成上述 Pascal 语句的基本符号。单词符号是语言的基本组成成分,是人们理解和编写程序的基本要素。识别和理解这些要素无疑也是翻译的基础。如同将英文翻译成中文的情形一样,如果你对英语单词不理解,那就谈不上进行正确的翻译。在词法分析阶段的工作中所依循的是语言的词法规则(或称构词规则)。描述词法规则的有效工具是正规式和有限自动机。

第二阶段,**语法分析**。语法分析的任务是:在词法分析的基础上,根据语言的语法规则,把单词符号串分解成各类语法单位(语法范畴),如“短语”、“子句”、“句子”(“语句”)、“程序段”和“程序”等。通过语法分析,确定整个输入串是否构成语法上正确的“程序”。语法分析所依循的是语言的语法规则。语法规则通常用上下文无关文法描述。词法分析是一种线性分析,而语法分析是一种层次结构分析。例如,在很多语言中,符号串

```
Z: = X + 0.618 * Y
```

代表一个“赋值语句”,而其中的 X + 0.618 * Y 代表一个“算术表达式”。因而,语法分析的任务就是识别 X + 0.618 * Y 为算术表达式,同时,识别上述整个符号串属于赋值语句

这个范畴。

第三阶段，**语义分析与中间代码产生**。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，并进行初步翻译(产生中间代码)。这一阶段通常包括两个方面的工作。首先，对每种语法范畴进行静态语义检查，例如，变量是否定义、类型是否正确等等。如果语义正确，则进行另一方面工作，即进行中间代码的翻译。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。

“翻译”仅仅在这里才开始涉及到。所谓“中间代码”是一种含义明确、便于处理的记号系统，它通常独立于具体的硬件。这种记号系统或者与现代计算机的指令形式有某种程度的接近，或者能够比较容易地把它变换成现代计算机的机器指令。例如，许多编译程序采用了一种与“三地址指令”非常近似的“四元式”作为中间代码。这种四元式的形式是：

算符	左操作数	右操作数	结果

它的意义是：对“左、右操作数”进行某种运算(由“算符”指明)，把运算所得的值作为“结果”保留下来。在采用四元式作为中间代码的情形下，中间代码产生的任务就是按语言的语义规则把各类语法范畴翻译成四元式序列。例如，下面的赋值句

$$Z:=(X+0.418)*Y/W$$

可被翻译为如下的四元式序列：

序号	算符	左操作数	右操作数	结果
(1)	+	X	0.418	T_1
(2)	*	T_1	Y	T_2
(3)	/	T_2	W	Z

其中，T_1 和 T_2 是编译期间引进的临时工作变量；第一个四元式意味着把 X 的值加上 0.418 存放于 T_1 中；第二个四元式指将 T_1 的值和 Y 的值相乘存于 T_2 中；第三个四元式指将 T_2 的值除以 Y 的值留结果于 Z 中。

一般而言，中间代码是一种独立于具体硬件的记号系统。常用的中间代码，除了四元式之外，还有三元式、间接三元式、逆波兰记号和树形表示等等。

第四阶段，**优化**。优化的任务在于对前段产生的中间代码进行加工变换，以期在最后阶段能产生出更为高效(省时间和空间)的目标代码。优化的主要方面有：公共子表达式的提取、循环优化、删除无用代码等等。有时，为了便于“并行运算”，还可以对代码进行并行化处理。优化所依循的原则是程序的等价变换规则。例如，如果我们把程序片断

```
for K:=1  to  100  do
  begin
    M:=I+10*K;
    N:=J+10*K
  end
```

的中间代码：

序号	OP	ARG1	ARG2	RESULT	注　　解
(1)	: =	1		K	K: = 1
(2)	j<	100	K	(9)	若 100 < K 转至第(9)个四元式
(3)	*	10	K	T_1	T_1: = 10 * K; T_1 为临时变量
(4)	+	I	T_1	M	M: = I + T_1
(5)	*	10	K	T_2	T_2: = 10 * k; T_2 为临时变量
(6)	+	J	T_2	N	N: = J + T_2
(7)	+	K	1	K	K: = K + 1
(8)	j			(2)	转至第(2)个四元式
(9)					

转换成如下的等价代码：

序号	OP	ARG1	ARG2	RESULT	注　　解
(1)	: =	I		M	M: = I
(2)	: =	J		N	N: = J
(3)	: =	1		K	K: = 1
(4)	j<	100	K	(9)	if(100 < k)goto(9)
(5)	+	M	10	M	M: = M + 10
(6)	+	N	10	N	N: = N + 10
(7)	+	K	1	K	K: = K + 1
(8)	j			(4)	goto (4)
(9)					

那么，最终所得的目标程序的执行效率就肯定会提高很多。因为，对于前者，在循环中需做 300 次加法和 200 次乘法；对于后者，在循环中只需做 300 次加法。尤其是，在多数硬件中，乘法的时间比加法的时间要长得多。

第五阶段，**目标代码生成**。这一阶段的任务是：把中间代码(或经优化处理之后)变换成特定机器上的低级语言代码。这阶段实现了最后的翻译，它的工作有赖于硬件系统结构和机器指令含义。这阶段工作非常复杂，涉及到硬件系统功能部件的运用，机器指令的选择，各种数据类型变量的存储空间分配，以及寄存器和后援寄存器的调度，等等。如何产生出足以充分发挥硬件效率的目标代码是一件非常不容易的事情。

目标代码的形式可以是绝对指令代码或可重定位的指令代码或汇编指令代码。如目标代码是绝对指令代码，则这种目标代码可立即执行。如果目标代码是汇编指令代码，则需汇编器汇编之后才能运行。必须指出，现代多数实用编译程序所产生的目标代码都是一种可重定位的指令代码。这种目标代码在运行前必须借助于一个连接装配程序把各个

目标模块(包括系统提供的库模块)连接在一起,确定程序变量(或常数)在主存中的位置,装入内存中指定的起始地址,使之成为一个可以运行的绝对指令代码程序。

上述编译过程的五个阶段是一种典型的分法。事实上,并非所有编译程序都分成这五阶段。有些编译程序对优化没有什么要求,优化阶段就可省去。在某些情况下,为了加快编译速度,中间代码产生阶段也可以去掉。有些最简单的编译程序是在语法分析的同时产生目标代码。但是,多数实用编译程序的工作过程大致都像上面所说的那五个阶段。

1.3 编译程序的结构

1.3.1 编译程序总框

上述编译过程的五个阶段是编译程序工作时的动态特征。编译程序的结构可以按照这五阶段的任务分模块进行设计。图 1.1 给出了编译程序总框。

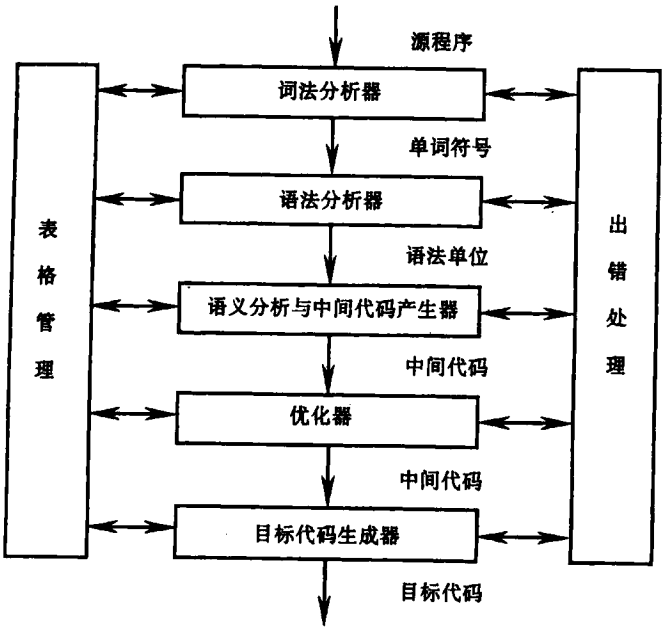

图 1.1 编译程序总框

词法分析器,又称**扫描器**,输入源程序,进行词法分析,输出单词符号。

语法分析器,简称**分析器**,对单词符号串进行语法分析(根据语法规则进行推导或归约),识别出各类语法单位,最终判断输入串是否构成语法上正确的"程序"。

语义分析与中间代码产生器,按照语义规则对语法分析器归约出(或推导出)的语法单位进行语义分析并把它们翻译成一定形式的中间代码。

有的编译程序在识别出各类语法单位后,构造并输出一棵表示语法结构的语法树,然后,根据语法树进行语义分析和中间代码产生。还有许多编译程序在识别出语法单位后并不真正构造语法树,而是调用相应的语义子程序。在这种编译程序中,扫描器、分析器和中间代码产生器三者并非是截然分开的,而是相互穿插的。

优化器,对中间代码进行优化处理。

目标代码生成器,把中间代码翻译成目标程序。

除了上述五个功能模块外,一个完整的编译程序还应包括“表格管理”和“出错处理”两部分。

1.3.2 表格与表格管理

编译程序在工作过程中需要保持一系列的表格,以登记源程序的各类信息和编译各阶段的进展状况。合理地设计和使用表格是编译程序构造的一个重要问题。在编译程序使用的表格中,最重要的是**符号表**。它用来登记源程序中出现的每个名字以及名字的各种属性。例如,一个名字是常量名、变量名,还是过程名等等;如果是变量名,它的类型是什么、所占内存是多大、地址是什么等等。通常,编译程序在处理到名字的定义性出现时,要把名字的各种属性填入到符号表中;当处理到名字的使用性出现时,要对名字的属性进行查证。

当扫描器识别出一个名字(标识符)后,它把该名字填入到符号表中。但这时不能完全确定名字的属性,它的各种属性要在后续的各阶段才能填入。例如,名字的类型等要在语义分析时才能确定,而名字的地址可能要到目标代码生成才能确定。

由此可见,编译各阶段都涉及到构造、查找或更新有关的表格。

1.3.3 出错处理

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。如果源程序有错误,编译程序应设法发现错误,把有关错误信息报告给用户。这部分工作是由专门的一组程序(叫做**出错处理程序**)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地点,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,以便进一步发现其它可能的错误。如果不仅能够发现错误,而且还能自动校正错误,那当然就更好了。但是,自动校正错误的代价是非常高的。

编译过程的每一阶段都可能检测出错误,其中,绝大多数错误可以在编译的前三阶段检测出来。源程序中的错误通常分为语法错误和语义错误两大类。语法错误是指源程序中不符合语法(或词法)规则的错误,它们可在词法分析或语法分析时检测出来。例如,词法分析阶段能够检测出“非法字符”之类的错误;语法分析阶段能够检测出诸如“括号不匹配”、“缺少 ;”之类的错误。语义错误是指源程序中不符合语义规则的错误,这些错误一般在语义分析时检测出来,有的语义错误要在运行时才能检测出来。语义错误通常包括:说明错误、作用域错误、类型不一致等等。关于错误检测和处理方法,我们将穿插在有关章节介绍。

1.3.4 遍

前面介绍的编译过程的五个阶段仅仅是逻辑功能上的一种划分。具体实现时,受不同源语言、设计要求、使用对象和计算机条件(如主存容量)的限制,往往将编译程序组织为若干**遍**(Pass)。所谓“遍”就是对源程序或源程序的中间结果从头到尾扫描一次,并作