编译程序构造方法

丘玉圃
刘椿年 编著
刘建丽

科学出版社

编译程序构造方法

丘玉圃　刘椿年　刘建丽 编著

科学出版社

1991

(京)新登字 092 号

内 容 简 介

本书是机械电子工业部计算机教材编审委员会组织出版的一本教材，讲述编译程序的构造方法。

与现有的各种编译技术的教材相比，本书具有独特的风格：在讲述编译程序各功能模块的构造方法时，除讲授一般的原则外，均结合本书所给出的实例语言和实例编译程序进行讨论。全书共十一章，包括词法分析，语言与文法，自顶向下的语法分析方法，作用域分析，类型分析，代码生成和编译程序自动构造技术等，最后一章给出了一个供学生编制编译程序的大型课程设计。每章末均附有相应的习题。

本书为理工科院校计算机编译技术课程的教材，也可供有关专业的广大科技人员阅读、参考。

编译程序构造方法

丘玉圃 刘椿年 刘建丽 编著

责任编辑 那莉莉

科学出版社出版

北京东黄城根北街 16 号

邮政编码：100707

北京宏伟胶印厂印刷

新华书店北京发行所发行 各地新华书店经售

*

1991 年 11 月第 一 版 开本：787×1092 1/16

1991 年 11 月第一次印刷 印张：22 1/4

印数：0001—3 000 字数：513 000

ISBN 7-03-002598-9/TP·196

定价：14.00 元

前　言

本书是根据机械电子工业部的工科电子类专业教材 1986—1990 年出版规划，由计算机教材编审委员会推荐出版的一本教材。

与现有的各种编译技术的教材相比，本书具有独特的风格。编者认为，当前实用的编译程序构造方法主要有两种：递归下降方法和基于 LR 分析法的自动构造方法。前一种方法简便易学，对环境要求低（只需任一种有递归功能的高级语言），应用面广（大中小型编译程序均可适用），因此，本书侧重于这种方法。为使学生对编译的全过程有一个清晰、具体的了解，本书选取 Pascal 的一个适中的子集 Pascal-T 作为实例语言，用 Pascal 写出此实例语言的五趟扫描的编译程序。在讲述编译程序的各个功能模块的构造方法时，除讲授一般原则外，均结合这个实例语言和实例编译程序进行讨论。通过对实例编译程序进行修改与扩充的若干练习，使学生真正掌握编译程序构造方法，并具有实际动手的能力，这是本书的目标。

本书共十一章。第一章为概述；第二章为词法分析；第三章介绍语言与文法的一般概念；第四章详细讨论递归下降语法分析方法；第五、六两章为语义分析，分别介绍作用域分析和类型分析的任务和方法；第七章为存贮分配；第八、九两章为代码生成，分别介绍理想机代码的生成及其向 Intel 8088 汇编代码的转换，并简略介绍代码的优化；第十章为编译程序自动构造技术；第十一章给出编译程序构造的一个大型课程设计。除了第二章 2.10 节介绍词法程序自动构造工具 Lex，第十章介绍语法分析程序自动构造工具 Yacc 外，其余内容均结合实例编译程序讲授。

本书可作为计算机各专业的编译技术课的教材。此课程的参考学时数为 64 学时。使用本书必须同时使用 Pascal-T 编译程序，该编译程序的源码包括在教材中，并带有软盘一起发行。学生需熟练地掌握 Pascal 语言，教师应充分利用教材和 Pascal-T 编译程序努力使学生学会写编译程序。如时间不够或软件工具缺乏，可略去第二章 2.10 节和第十章的内容。本书及所附的 Pascal-T 编译程序对计算机软件和计算机应用的广大科技人员也有很大的参考价值。

本书所配软盘（共 6 片，由科学出版社负责发行）如下：

Microsoft Pascal 编译　　二片

Microsoft MASM 宏汇编程序　　二片

用Microsoft Pascal语言书写的Pascal－ T编译程序源程序和目标程序

用Microsoft Pascal语言书写的Pascal－ T解释程序

QPAS.BAT和PAST.BAT批命令文卷

Pascal－ T编译程序的可执行文卷QPAST.EXE

Pascal－ T解释程序的可执行文卷QPAST.EXE

QPAST1.DAT和QPAST2.DAT数据文卷

} 二片

本书第二章 2.10 节和第十章为刘椿年编写，第七章和每章的习题为刘建丽编写，其余章节均为丘玉圃编写，Pascal-T 编译程序是丘玉圃专为教学而设计和开发的。编者特别感谢林行良教授，他仔细审阅本书，提出了许多宝贵的意见；感谢郭福顺教授，他关心本书的出版；最后，还感谢冯仁平、阚胜国和冯强三位同学，是他们认真耐心地把全部书稿录入计算机。由于编者水平有限，书中难免存在一些缺点和错误，殷切希望广大读者批评指正。

1989 年 9 月于北京

目　录

第一章　编译程序概述 …… (1)
1.1　编译程序与翻译程序 …… (1)
1.2　编译程序的结构 …… (2)
1.3　编译程序的书写语言与 T 形图 …… (4)
1.4　Pascal-T 语言及其编译程序 …… (7)
习题 …… (12)
程序清单 1　Pascal-T 编译程序的主程序 …… (13)
第二章　词法分析 …… (32)
2.1　词法分析的任务 …… (32)
2.2　取字符与写符号 …… (34)
2.3　扫描 …… (36)
2.4　标识符与保留字 …… (37)
2.5　数 …… (41)
2.6　字符串 …… (42)
2.7　注解 …… (43)
2.8　差错处理 …… (44)
2.9　其它语言的特有问题 …… (44)
2.10　词法分析程序的自动构造 …… (45)
习题 …… (51)
程序清单 2　Pascal-T 编译程序的词法分析程序 …… (53)
第三章　文法与语言 …… (63)
3.1　形式语言 …… (63)
3.2　文法的分类 …… (66)
3.3　上下文无关文法的二义性 …… (68)
3.4　扩展的 BNF …… (71)
3.5　Pascal-T 语言的语法 …… (73)
习题 …… (74)
第四章　自顶向下的语法分析方法 …… (76)
4.1　自顶向下分析的基本思想 …… (76)
4.2　LL(1)分析方法 …… (80)
4.3　递归下降分析程序的构造 …… (80)
4.4　验证 Pascal-T 满足 LL(1)条件 …… (86)
4.5　Pascal-T 语言的分析程序 …… (93)
4.6　语法差错修正 …… (99)

4.7 Pascal-T 语言带语法差错修正的分析程序 …… (101)
习题 …… (105)
程序清单 3 Pascal-T 编译程序的语法分析程序 …… (107)
第五章 作用域分析 …… (125)
5.1 标识符的作用域与程序体 …… (125)
5.2 标识符表 …… (128)
5.3 程序体表与层次表 …… (129)
5.4 Pascal-T 语言的语义分析程序 …… (133)
程序清单 4 Pascal-T 编译程序的语义分析程序 …… (138)
第六章 类型分析 …… (166)
6.1 标识符表的完整形式 …… (166)
6.2 类型结构表 …… (167)
6.3 查标识符表算法的完整形式 …… (169)
6.4 常量定义 …… (170)
6.5 类型定义 …… (173)
6.6 变量定义与形式参数定义 …… (181)
6.7 语义差错修正 …… (185)
6.8 表达式 …… (186)
6.9 语句 …… (196)
6.10 过程与函数 …… (199)
习题 …… (207)
第七章 存贮管理 …… (210)
7.1 静态存贮管理 …… (210)
7.2 栈式存贮管理 …… (212)
7.3 堆式存贮管理 …… (215)
习题 …… (216)
第八章 代码生成 …… (217)
8.1 理想计算机概念 …… (217)
8.2 Pascal-T 计算机及其解释程序 …… (217)
8.3 Pascal-T 语言的代码生成程序 …… (225)
8.4 表达式和变量的目标 …… (226)
8.5 语句的目标 …… (230)
8.6 过程和函数的目标 …… (235)
习题 …… (243)
程序清单 5 Pascal-T 编译程序的代码生成程序 …… (244)
程序清单 6 Pascal-T 解释程序 …… (257)
第九章 汇编代码生成 …… (267)
9.1 Intel 8088 处理器简介 …… (267)
9.2 Pascal-T 语言的汇编代码生成程序 …… (268)

9.3 汇编代码生成的例子 …… (268)
9.4 目标代码优化 …… (271)
习题 …… (274)
程序清单 7 Pascal-T 编译程序的汇编代码生成程序 …… (275)
第十章 基于自底向上语法分析技术的编译程序自动构造 …… (285)
10.1 LR 分析方法 …… (285)
10.2 Yacc 简介 …… (288)
10.3 用 Yacc 构造 PL/ 0 编译程序 …… (290)
习题 …… (301)
第十一章 编译程序构造课题 …… (302)
11.1 PL 语言的程序示例 …… (302)
11.2 PL 语言的描述 …… (303)
11.3 PL 语言的语法 …… (305)
11.4 课程设计的步骤和要求 …… (306)
11.5 PL 语言的解释程序 …… (307)
附录一 一个用 Pascal-T 语言编写的源程序及其编译过程的例子 …… (313)
附录二 一组用 Pascal-T 语言编写的用户源程序的实例 …… (336)
附录三 Pascal-T 编译程序(软盘)的使用说明 …… (344)
参考文献 …… (346)

第一章　编译程序概述

编译程序是翻译程序中的一种，它把诸如 Pascal，FORTRAN 等高级语言编写的程序翻译成机器语言的程序。本章先介绍什么是编译程序，编译程序的一般结构；探讨应该用什么语言来书写编译程序；最后，给出本书的一个样本语言 Pascal-T。

1.1　编译程序与翻译程序

1956 年，在 IBM 704 计算机上构造的第一个 FORTRAN 编译程序使人们从繁重的机器语言程序设计中解放出来，它的出现使程序设计跨进了一个新阶段。今天，人们与计算机打交道几乎都不使用机器语言。例如，我们可通过 FORTRAN，Pascal，C，PROLOG，dBASE Ⅲ等语言与机器打交道。

翻译程序是把一种语言编写的程序翻译成另一种语言的程序。前一种语言称为源语言，而后一种语言称为目标语言。也就是说，翻译程序或翻译器是这样的一个程序，它把源语言的指令序列翻译为目标语言的指令序列(图 1.1)。

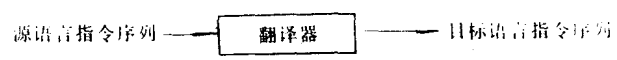

图 1.1

根据源语言与目标语言的不同种类，有各种不同种类的翻译器：

1.若源语言是汇编语言，目标语言是机器语言，这样的翻译器称为汇编程序(assembler)。通常地，汇编程序把汇编语言的指令一对一地翻译成机器指令。如果有某些汇编指令被翻译成若干条机器指令，这样的汇编程序称为宏汇编程序(macroassembler)。汇编语言或宏汇编语言都属低级语言。

2.若源语言是高级语言，如 Pascal，FORTRAN 语言等，目标语言是可执行的机器代码①，或汇编代码，这样的翻译器称为编译器或编译程序(compiler)。通常一条高级语言语句要翻译为多条机器指令或汇编指令。

3.若源语言是比目标语言大的一种语言，则我们可通过一个翻译程序把源语言超出目标语言的成分翻译成目标语言的表示。例如，FORTRAN Ⅳ中没有 IF-THEN-ELSE 成分，我们通过一个翻译程序把这种成分翻译成 IF 语句与 GOTO 语句。又如，在 C 语言中的宏替换与包含(include)文卷等结构也可作类似的处理。这种翻译器称为预处理器(preprocessor)。更一般地，把一种高级语言翻译成另一种高级语言的翻译器也属于预处理器。

4.若源语言属于低层语言，目标语言属于高层语言，这种把低层语言的目标代码重新

①用某种语言形式表示的数据和程序经常被称为该语言的"代码"。

恢复成高层语言的源代码的翻译器，当低层语言是机器语言，高层语言是汇编语言时称为反汇编器(disassembler)；当低层语言是低级语言，高层语言是高级语言时称为反编译器(decompiler)。反汇编器在许多机器上已十分成功地作为产品实现，但反编译器的实现却是一个十分困难的课题。

1.2 编译程序的结构

编译程序的功能是把用高级语言编写的程序翻译成等价的机器指令程序或汇编语言程序(图 1.2)。

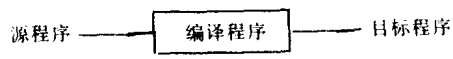

图 1.2

编译程序是一个高度复杂的程序，从功能上看，它的典型结构如图 1.3 所示。

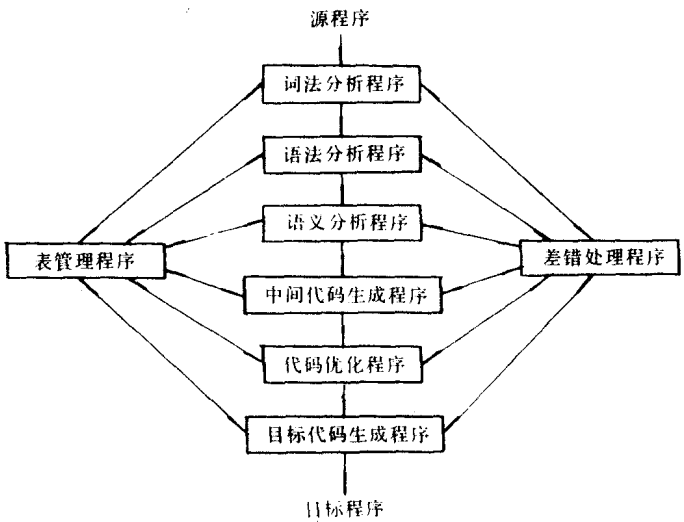

图 1.3 编译程序的典型结构

对于一个源程序，一般要经过一系列的阶段，例如词法分析、语法分析、语义分析等阶段，才能把它从一种表示翻译成另一种表示，最后翻译成目标程序。在每一阶段中，要调用表管理程序和差错处理程序。对于实际的编译程序，上述有些阶段可合在一起进行。例如，当词法分析识别出一个符号(token)时，控制就转到语法分析程序从语法上检查这个符号并作相应的语义处理。

在编译过程中，要从头至尾扫视源程序或其内部表示。每扫视一遍称为一趟扫描。如果经过一趟扫描就能生成目标代码，这样的编译程序称为一趟扫描的编译程序，否则称为多趟扫描的编译程序。

图 1.4 是一个不带代码优化的五趟扫描的编译程序的实现方案。

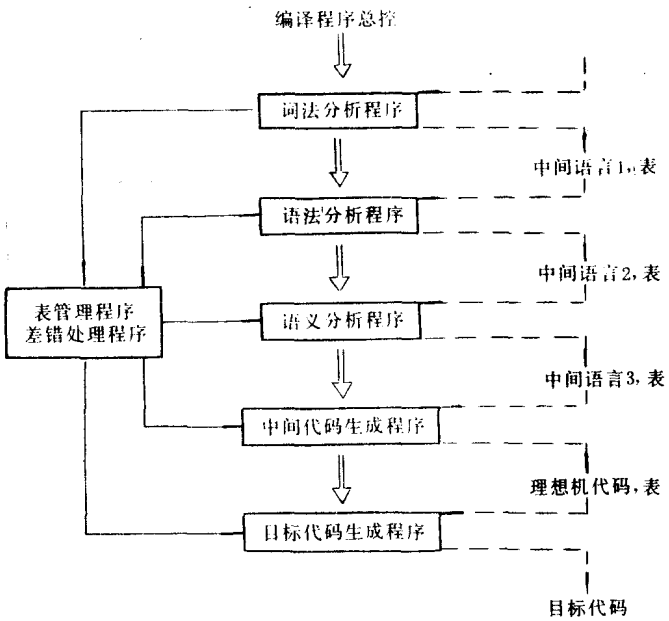

图 1.4　五趟扫描的编译程序的实现方案

这个五趟扫描的编译程序是这样工作的。首先,编译程序的主程序调用词法分析程序,词法分析程序把源程序作为输入,将它转换为一种内部表示,称为中间语言 1,并得到有关的一些表;然后,主程序调用语法分析程序,语法分析程序把中间语言 1 作为输入,进行语法分析,并转换为中间语言 2……;最后,主程序调用目标代码生成程序,把中间代码(可看作为一种理想机代码)作为输入,并转换为目标代码。

对于一趟扫描的编译程序,程序本身紧凑,编译速度快;对于多趟扫描的编译程序,它的功能块清晰,占用内存贮器少(因为各趟扫描可以重叠使用内存)。这是它们各自的优点。

编译程序设计成一趟扫描或多趟扫描,是基于:

1.语言的定义;

2.目标代码是否优化;

3.宿主机存贮器的规模。

对于本书中为教学设计的 Pascal-T 编译程序,其前四趟扫描很容易合并成一趟扫描。但是,为便于教学,我们还是把它设计成五趟扫描。

最后,我们给出一个好的编译程序的重要指标:

1.任何符合语言定义的源程序都能被正确地翻译成目标程序。

2.任何荒唐的程序都不致引起编译程序的崩溃(即非正常结束)。

3.任何非法的结构都能被发现和标志出来,且不应产生过多的错觉(连锁反应)。

4.编译程序可读、可维护。

5.编译程序模块化与结构化。

对于 Pascal-T 教学编译程序,尽管可改进之处可能永远存在,但是,为了使它在教学上起典范作用,我们一直以上述五项为主要目标。

1.3 编译程序的书写语言与T形图

在某台机器上运行的编译程序理应用该机器的代码来书写。但是,用机器语言书写的编译程序可靠性差,难以进行维护。在70年代前,几乎所有的编译程序都用机器语言或汇编语言来书写。在80年代,几乎所有的编译程序都是用高级语言来编写的。我们来探讨其中的原因。

一个编译程序涉及三种语言:

1. 源语言。
2. 目标语言。
3. 编译程序的书写语言。

我们可以用一个T形图来表示一个编译程序(图1.5)。T形图的左上角写源语言,右上角写目标语言,而下角写书写语言。

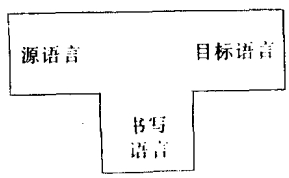

图1.5 T形图

例如,一个用PDP-11代码书写的、产生PDP-11目标代码的PL/1语言的编译程序可表示成图1.6。

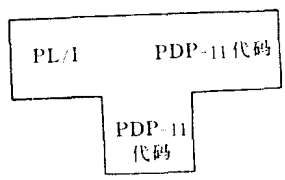

图1.6

这是一个能在PDP-11上直接运行且产生同一机器代码的PL/1编译程序。一般地,一个编译程序可产生某一个机器的代码。例如,图1.7的T形图表示一个能在IBM PC上直接运行而产生PDP-11代码的Pascal编译程序。

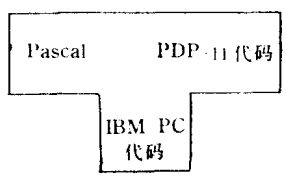

图1.7

这种在一个机器上运行产生另一个机器的目标代码的编译程序称为交叉编译程序(cross-compiler)。它在 IBM PC 上编译 Pascal 源程序,在 PDP-11 上运行目标程序。

如果在 *A* 机器上已经有一个用 *A* 代码书写的、产生 *A* 代码的 Pascal 编译程序,如图 1.8(a)所示,我们要得到一个用 *A* 代码书写的、产生 *A* 代码的另一种语言,例如是 FORTRAN 语言的编译程序,如图 1.8(b),则我们只要用 Pascal 语言书写一个产生 *A* 代码的 FORTRAN 编译程序(图 1.9)即可。

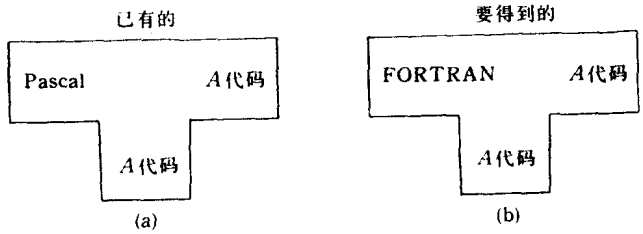

图 1.8

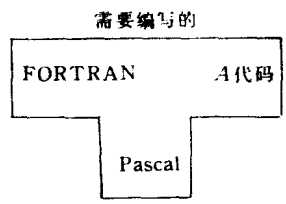

图 1.9

这个编译程序是一个 Pascal 程序。由于 *A* 机器上已有产生 *A* 代码的 Pascal 编译程序,因此可编译这个特殊的 Pascal 程序,得到的是具同样功能(把 FORTRAN 源程序翻译成等价的 *A* 代码)用 *A* 代码表示的目标程序,这就是所要的编译程序。

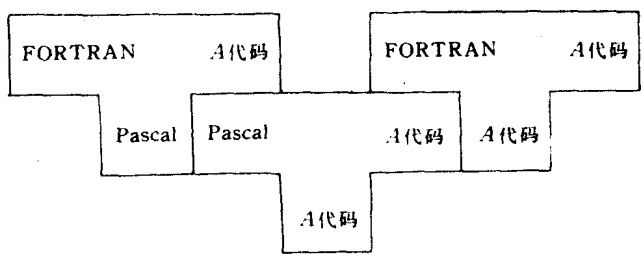

图 1.10

把已有的编译程序的 T 形图写在中间的下方(图 1.10),把要得到的编译程序的 T 形图写在右上方,且后者的书写语言与前者的目标语言对齐,那么左上方就是需要编写的编译程序的 T 形图。它的书写语言应是 Pascal 语言,而源语言与目标语言应分别与要得到的编译程序的源语言与目标语言相同。这便是 T 形图的运算。显然,用 Pascal 语言编写一个 FORTRAN 编译程序比用 *A* 代码编写要容易得多。可见,只要 *A* 机器上有了第一个可书写编译程序的高级语言的编译程序,原则上其它的高级语言编译程序都可用该高

级语言来书写。

如何获得 *A* 机器上第一个可书写编译程序(例如 Pascal 的编译程序)的高级语言?是否一定要用机器语言来编写呢?有以下两种简便的途径。

1. 自展。可先在 *A* 机器上用机器语言或汇编语言编写该高级语言的子集,例如 Small Pascal 的编译程序(图 1.11 中的 a)。然后再用 Small Pascal 编写 Middle Pascal 的编译程序(图 1.11 中的 b)。经 Small Pascal 的编译程序编译后,得 Middle Pascal 编译程序(图 1.11 中的 b′)。最后再用 Middle Pascal 编写全 Pascal 编译程序(图 1.11 中的 c)。经 Middle Pascal 编译程序编译后,便得到所要的第一个编译程序(图 1.11 中的 c′)。这种方式就像滚雪球那样扩大语言的编译程序。

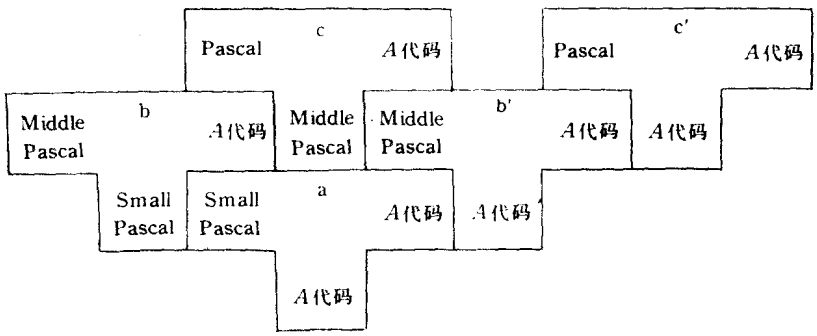

图 1.11

利用自展的方法使我们用机器语言或汇编语言编写高级语言的编译程序的工作量减少到最低限度。上面我们仅给出自展的基本思想,而在实际上可能要作多次的自编译才能得到可靠的编译程序。

2. 移植。设在 *B* 机器上已有一个可运行的 Pascal 编译程序(图 1.12 中的 a)。只要我们编写一个用 Pascal 语言书写的,产生 *A* 机器代码的 Pascal 编译程序(图 1.12 中的 b),按如下 T 形图的运算去做,就能得到 *A* 机器上所要的 Pascal 编译程序(图 1.12 中的 c)。

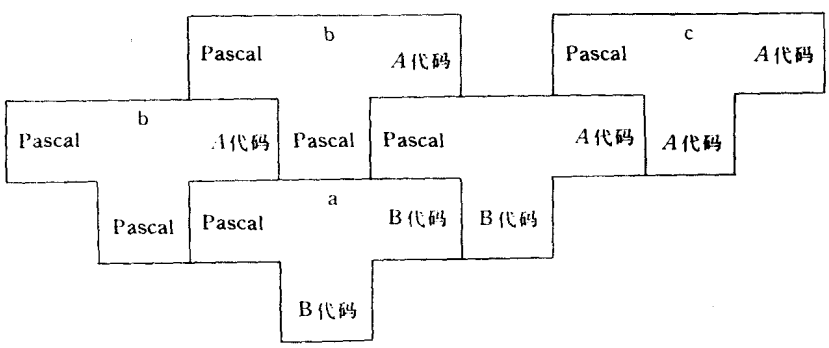

图 1.12

从前面的讨论可以看出,用高级语言来书写编译程序,生产的周期短,可靠性高,易修改、扩充与维护,并且易于移植。有人或许会提出这样的疑问,用高级语言书写的编译程

序的目标代码会长些，运行会慢些，但是，当今的计算机的存贮器越来越便宜，运行速度越来越快，程序正确性是主要矛盾。因此，我们应学会用高级语言编写编译程序。

1.4 Pascal-T 语言及其编译程序

学习编译程序的构造，最好有一个样品。要选择一个语言，为它设计与构造编译程序。这个语言不应该太小，否则不能反映编译程序的各个方面；这个语言也不应该太大，否则将陷入无穷的细节中。Pascal 语言是 70 年代以来最有影响的语言，我们选取它一个大小适宜的子集作为教学语言(teaching language)，称为 Pascal-T 语言，并用 Pascal 语言为它编写编译程序。因此，读者应熟练掌握 Pascal 语言。

1.4.1 Pascal-T 语言的规模

Pascal-T 语言是 Pascal 语言的一个子集。

在 Pascal-T 语言中，简单类型仅有 integer，Boolean，char 和子域类型，没有 real 型和枚举类型。构造类型仅有数组类型和记录类型，没有文卷类型和集合类型，并且没有记录变体和紧缩类型；没有指针类型。

Pascal-T 的语句包括赋值，过程，空，复合，if，while 和 for 语句，没有 goto，case，repeat 和 with 语句。

Pascal-T 没有标号说明部分；形式参数仅有赋值参数和变量参数，没有过程参数和函数参数，当然也没有可调数组参数；过程与函数不能向前引用；write 语句的参数仅是表达式，不能出现

表达式：表达式　　或　　表达式：表达式：表达式

的形式。

从 Pascal 语言的语法规则中，删去那些 Pascal-T 不包含的成分，就可得到 Pascal-T 语言的语法规则，我们把它留给读者去做。

1.4.2 Pascal-T 语言编译程序总体结构

Pascal-T 编译程序是一个为教学而设计的编译程序。为了使读者能清晰地看到从源程序到目标程序的整个翻译过程，有利于编译程序构造方法的教学，我们把它设计成五趟扫描。1.2 节中那个不带代码优化的五趟扫描的编译程序实现方案(图 1.4)就是 Pascal-T 的实现方案。

根据实现方案，在 Pascal-T 编译程序中，应说明如下的文卷变量：

```
PAS:text;  {Pascal-T 源程序}
IL1:ILFileType;  { 中间语言 1}
IL2:ILFileType;  { 中间语言 2}
IL3:ILFileType;  { 中间语言 3}
PTCode:PTCFileType;  {PTC 代码文卷}
ASM:text;  { 汇编代码文卷}
SFile:file of char;  { 字符串文卷}
```

```
        DSP:text;   { 显示文卷}
```

其中 ILFileType 和 PTCFileType 分别是中间语言(intermediate language)文卷类型和 PTC 代码文卷类型,它们在类型定义部分定义为

```
ILFileType = file of cardinal;
PTCFileType = file of integer;
```

其中

```
cardinal = 0:maxint;
```

因为编译程序中所有内部表示都是非负整数,所以我们引入 cardinal(基数)来描述这种类型。

下面给出 Pascal-T 语言编译程序的总体结构:

```
Program PasCompiler(input, output);
    常量定义部分;
    类型定义部分;
    变量说明部分;
    公用子程序(第一组),包含下列六个过程说明:
            Initialization          OpenFiles
            WriteSTabToSFile        WriteTabToDsp
            PassHead                PassFinal
    公用子程序(第二组),包含下列四个过程说明:
            GetSymbol               PutSymbol
            error                   TabOverfolw
    Procedure LexicalAnalysis          { 词法分析}
        (var PAS :text; var IL1 :ILFileType; var DSP :text);
    ......
    Procedure SyntaxAnalysis           { 语法分析}
        (var IL1, IL2 :ILFileType; var DSP :text );
    ......
    Procedure SemanticAnalysis         { 语义分析}
        (var IL2, IL3 :ILFileType; var DSP :text );
    ......
    Procedure CodeGeneration           { 理想机代码生成}
        (var IL3 :ILFileType; var PTCode :PTCFileType;
                                  var DSP :text);
    ......
    Procedure AsmCodeGeneration        { 汇编代码生成}
        (var PTCode :ICFileType; var ASM :text);
    ......
begin { 主程序}
    Initialization;                    { 初始化}
```

```
OpenFiles;                          { 打开文卷}
显示 Pascal-T 编译程序的版本号;
{ 第一趟扫描 :词法分析}
LexicalAnalysis(PAS,IL1,DSP);
{ 第二趟扫描 :语法分析}
SyntaxAnalysis(IL1,IL2,DSP);
if ErrCount = 0
then begin
        { 第三趟扫描 :语义分析}
        SemanticAnalysis(IL2,IL3,DSP);
        if ErrCount = 0
        then begin
                { 第四趟扫描 :代码生成}
                CodeGeneration(IL3,PTCode,DSP);
                 if overflow = [ ]
                 then { 第五趟扫描 :汇编代码生成}
                      AsmCodeGeneration(PTCode,ASM)
                 else 显示代码生成错
             end
        else 显示语义分析错
    end
else 显示词法分析或语法分析错
end { PasCompiler }。
```

在每趟扫描之前,要 reset 或 rewrite 有关文卷,还要调用 PassHead。在每趟扫描之后,要调用 PassFinal,还要关闭(close)或删除(discard)有关文卷以及输出有关的表格。

1.4.3 第一组公用子程序

第一组公用子程序仅被主程序调用。

1. Initialization.在 Pascal-T 编译程序中,为了程序的可读性,定义如下八个枚举类型:

```
symbol      OperandForm     OperatingCode
IdClass     StandFuncs      StandProcs      VarKind     StructForm
```

由于前三个枚举类型的值要在各趟之间进行传递,而输出在中间语言文卷的是枚举值相应的序号,因此就需要从序号恢复相应的枚举值。另一方面,为了使输出在 DSP 文卷的中间语言和各种编译用到的表格具有可读性,应显示枚举值的拼法(spelling),因此在变量说明部分的最后,说明了如下十一个数组变量:

```
sym         OpFm        OperatCd
sp          OpFmSp      OpCdSp      IdClassSp
StandpSp    StandfSp    KindSp      StFormSp
```