可移植编译程序及其分析说明

PASCAL

杨德元 编

清华大学出版社

PASCAL可移植编译程序P_4及其分析说明

杨德元 编

清华大学出版社

一九八二年

内 容 简 介

PASCAL 可移植的编译程序 P4，是用它自己处理的 PASCAL 语言的子集编写的，仅四千行左右，但适宜于自展，是在各种类型计算机上实现 PASCAL 的有效工具。

本书共三部分：第一部分是编者为编译程序文本作了简要分析说明，包括：概述、编译思想、说明的处理、代码生成、错误校正、存贮组织等六节；第二、第三部分是编译程序的文本及目标程序的汇编和解释程序文本。这些文件是学习、研究和实现 PASCAL 语言的重要资料，对用户很有参考价值，也是学习编译课程的有效的教学资料。

本书适用系统软件和应用软件设计人员、PASCAL 语言的用户、计算中心软件维护人员及有关专业师生教学参考书。

PASCAL 可移植的编译程序 P4
及 其 分 析 说 明

杨 德 元 编

☆

清 华 大 学 出 版 社 出 版
北京 海淀 清华园
武汉地质学院北京研究生部印刷厂排印
北 京 市 海 淀 区 印 刷 厂 印 装
新华书店北京发行所发行·各地新华书店经售

☆

开本：787×1092 1/16 印张：15 字数：358 千字
1982年6月第一版 1982年6月第一次印刷
印 数：1—25,000
统一书号：15235·32 定 价：1.90元

编 者 的 话

Pascal 语言的可移植编译程序 P4 是用它自己处理的 Pascal 语言的子集编写的。仅四千行左右，适宜于自展，是在大、中、小和微型计算机上实现 Pascal 的有效工具。P4 文本是由安曼（U. Ammann）、诺里（K. Nori）和雅科布（C. Jocobi）等人所编写，并于1977年1月修改后提供给用戶。大大地促进了 Pascal 语言的广泛实现和应用。

为了适应编译课程的教学需要，编者专门为 P4 文本写了分析说明，希望能有助于一般的程序设计人员易于掌握它，从而了解编释程序的全貌。当然，本书的出版如能对 Pascal 语言在我国的广泛实现有所促进，那更是编者所热切期望的。

感谢英国曼彻斯特大学计算机系威尔逊博士（I. Wilson）对编者在掌握和理解可移植编译程序 P4 文本方面所给予的帮助。清华大学计算机工程与科学系软件教研组蒋国南同志全面、仔细地多次审改本书。吕映芝同志阅读了全部手稿，提出了许多宝贵的意见和建议。在此一并表示衷心的感谢。

书中不当之处，希望读者批评指正。

1981年11月

目　　录

序　　言

Pascal 语言的 P 编译程序是为标准 Pascal 的一个子集编写的可移植的编译程序。它是由 U.Ammann 等人写于 1973 年，最后的文本完成于 1977 年1月，它为在各类计算机上实现 Pascal 提供了有效的工具。这个编译程序是用它自己所处理的子集编写的，篇幅在四千行左右。它所产生的目标程序是一台假想的栈计算机（stack computer，以下简称为 SC）的汇编语言形式的代码，可在不同的计算机上解释执行。本书首先为编译程序文本提供简要的分析说明，希望能有助于读者对 Pascal 的编译思想、实现技巧有所具体了解，从而促进程序设计语言 Pascal 在我国大、中、小以及微型计算机上的实现。

在简明扼要的分析说明之后，附有 Pascal 的P 编译程序的全部文本，目标程序的汇编和解释程序文本。这些文件是学习，研究 Pascal 及其实现的重要资料。特别是对于要实现 Pascal 的单位很有参考价值。另有一份用 SC 汇编语言描述的 P编译程序，全文约18,000行，由于篇幅所限未能列入本书。

另外，P 编译程序是用 Pascal 语言写成的，易于阅读、理解。因此本书也是计算机工程与科学系的大学生和研究生学习编译程序结构和原理的有效教学资料。

第 一 部 分

可移植的编译程序 P4 的分析说明

(一) 概 述

目前 Pascal 语言已经作为有效的高级语言之一得到广泛使用。从 Pascal 最早在 CDC 6000 系列上的实现，至今无论是大型或中、小型计算机，以至微型计算机都纷纷配备了 Pascal 语言的不同文本的编译程序，受到广大用户，特别是计算机科学与工程领域内的学生、研究生、教师、设计人员、研究人员的欢迎，而且已经开始进入到更为广阔的工业、商业应用领域。短暂的历史，已经充分显示出 Pascal 是一个富有生命力的语言。

由于 Pascal 与 Fortran、PL/1、BLISS 相比更为独立于机器；而和 Fortran、Algol 60 相比，它又是相当有效的。然而比 PL/1 和 Algol 68 显然简单得多。因此，不仅用户欢迎，而且使它能成为一种能实现自编译的可移植的语言。

可移植的编译程序（以下简称为 P 编译程序）是在 Pascal 语言诞生初期就开始建立的，并在大量实践的基础上进行了修改完善。作为一个程序来说 P编译程序可以在一个假想的 SC 上运行。由于 SC 不依赖于任何具体的机器，而且很简单，所以 P编译程序的文本只需要花费不大的代价就能加以移植。多年来大量事实说明，许多计算机上的 Pascal 的编译程序，大都源出于此。

Pascal 编译程序是按照预先确定的逐步求精的次序组织的：

1. 语法分析。
2. 语法分析并具有上下文无关的错误校正。
3. 生成各种编译表。
4. 处理内容有意义的错误。
5. 地址的计算与分配。
6. 代码生成。

依据这些步骤，编译程序的实现者在 CDC 6000 系列和 SC 上反复多次，在CDC 6000 系列上最早建立了具有代码优化功能的 Pascal 标准文本的编译程序。这一过程也说明了，可以用具有大量共同的子结构的编译程序的几个文本来为不同的机器生成目标代码，这种考虑是可行的，这就为建立易于移植的编译程序提供了基础。

为便于迅速和广泛的实现 Pascal，U.Ammann 和 K.Jensen 在 1973 年初，就已准备了一组有关实现 Pascal 的完整的文件资料，包括：

1. 源语言形式的 P编译程序文本。
2. SC 汇编语言形式的 P编译程序文本。

3. 对 SC 汇编语言的汇编和解释程序，它也是 Pascal 源语言形式的文本。

4. 标准的 Pascal 文本。

5. SC 的文本*。

这里 P编译程序所处理的语言是标准 Pascal 的子集，作了如下的限制：

1. 过程和函数不能用作参数。

2. 不能用 GOTO 语句转出过程体或函数体。

3. 仅有预先定义的字符文件。

4. 无紧缩 (Packing)。

此外，还有一处改变，即用 'Mark' 和 'Release' 两个过程来代替标准文本中的 'Dispose' 过程。

至此，不难看出，利用 U.Ammann 等人所提供的上述几个文件，就有可能移植 P编译程序。迅速实现的方法之一，就是为 SC 的汇编语言编写一个解释程序，然后用它来运行 SC 汇编语言所写的 P编译程序。（即上述文件资料 2）。另外一种快速方法，自然是将文件 2 翻译成为所要实现的机器语言的形式。

按照上述的基本思路，下面提出实现 Pascal 的三种方案。这是最基本的思路，自然还会有很多其它巧妙的方案。

1. 如果使用 Pascal 的主要目的是教学，而且编译和执行的大都是学生实习的小程序，那么实现的最简单方法就是为 SC 写一个有效的汇编和解释程序。

采用这种方法，唯一的困难就是要妥善处理从栈中提取和存贮数据的问题。特别是当不同类型的数据，例如：整型、实型、字符、集合、指针等，要求有不同数量的存贮单元时，就格外需要有效地解决这一问题。在编写了汇编和解释程序之后，就可以运行文件 2 用来编译短小的程序。编译程序的输出可以用汇编和解释程序来处理，使被编译了的程序得以解释执行。应当看到，尽管是解释执行，然而总的效率仍然可以与大型语言的商品性编译程序相比拟。

实现这一方案，大约需要 54 K 字节以存放 SC 汇编语言形式的 P 编译程序文本，20K～30K 字节用来存放 P编译程序的数据（以编译学生练习题为例），以及存放汇编和解释程序。

2. 适宜于自展的方法。系统程序设计人员可将 SC 汇编语言形式的 P编译程序转换成为某一机器的汇编语言。毫无疑义，这将增加存贮量，但由于实现者可以拟定 P编译程序的代码生成模式，和把文件资料 2 转换成为汇编语言的确切方案，因此有可能实现有效的细微优化 (peophole optimization)，这种优化方法也还是相当有效的。完成上述工作就可以对文件 1 直接加工，以生成所实现机器的代码。

3. 如果主要是受存贮器容量的限制，最合适的方法也许是采取解释和机器执行之间适当的混合。有可能实现的技术就是 J.R.Bell 所提出的“穿线式编码”(threaded code)。与第一种方案相比较，在速度方面可能有较大的提高，而只需增加少量的存贮。采用这一方案，可以同时设计适合于特定机器结构的中间语言，和支持该语言运行的程序。然后，改造文件 2 成为这一语言的形式。这样 Pascal 在所实现的机器上就是有效的了。进而再对文件 1 进行修改，实现自展。

K.V.Nori 等人指出，实现这一切是要花费代价的。少数人常常从一些论文中得出相反的结论，似乎只要花费少量代价就能得到可移植性和实现自展。为此，也需要对可移植能力

* SC并不是专为可移植的P编译程序设计的，它最初是用于可移植的BCPL语言的目标代码的机器。

有一个比较确切的，统一的认识。可移植性是关于一个程序能够从某一环境移植到另一环境的难易程度的度量，但这种移植不会对希望保留的特性有任何损失。也就是被移植系统的效率和可靠性不会受到什么影响。

在这里之所以要强调可移植性的问题，那是想提醒读者注意，本文所介绍之主题—Pascal 的 P 编译程序，其 P 字不是 Pascal 的缩写，而是 Portable（可移植的）一字的字头。P4 是指本书所介绍的是P 编译程序的第四个版本。

（二） 编 译 思 想

采用何种策略实现一种语言的编译，这是一个困难的但必须确定的问题。由于编译程序本身的性质，决定了它不能和机器无关，最终它总是要以某种语言来表达，为某种机器生成代码，以及在某个操作系统下运行。尽管如此，根据可移植性的要求，应当对数据和运算的基本单位和结构的约定最少。

假如我们设计一个很清晰的机器，其中没有对数据和运算的基本单位与结构的表示作出约定，那么这个设计就完全可以依据 Pascal 的逻辑考虑为基础。然而这方面的处理是相当地困难。另外一种选择，是对数据和运算的表示作出完全的约定。例如，Bcpl语言的Ocode。但这一办法过于死板，不能有效地解决对于不同系统的适应问题。因而都没有采用。

P 编译程序采取了折衷的办法。SC 的所有的基本运算都是出自 Pascal 的逻辑要求，也有一些额外的运算，那是由于程序的线性化和以树结构方式存取数据所需要的。其次集合在 SC 中是作为基本单元出现的。这种选择是按照编译程序和解释程序的简单性和有效性来考虑的。

由于 Pascal 语言是符合于 LL(1) 型特征的，而且可以利用语法图和 BNF 公式同时来定义语法，因此采用递归下降方法来实现对语法正确的程序进行语法分析工作是比较简单的、对语法图或 BNF 公式所定义的每一个语法实体都可以建立一个相应的实现语法分析的过程。此外，还要有一个扫描符号的辅助过程，同时它也实现词法分析与转换。这个扫描程序实际上是自底向上的，它将输入字符归纳为终极符序列。图 1 是分析与扫描的简要说明。

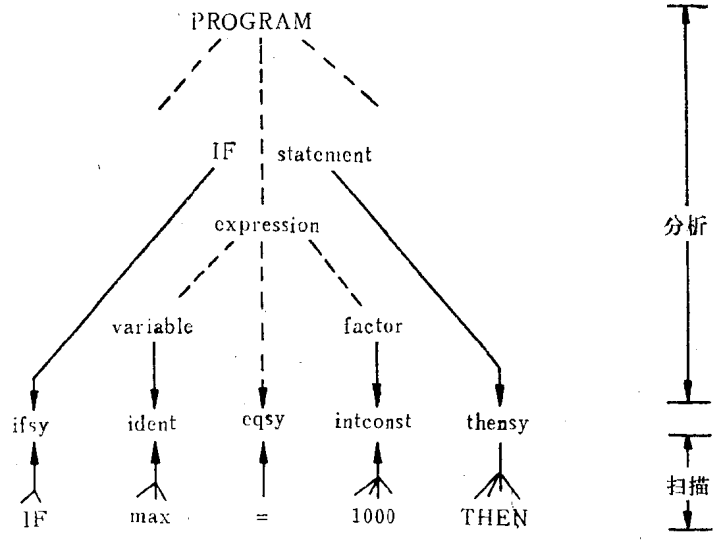

图 1 分 析 与 扫 描

扫描过程是顺次从输入文件中读取各种合法的单词符号，并转换成为相应的内部表示，以作为语法分析的基础。它的基本流程如图2所示，任何一个单词符号的分析结果均置于sy和op中。

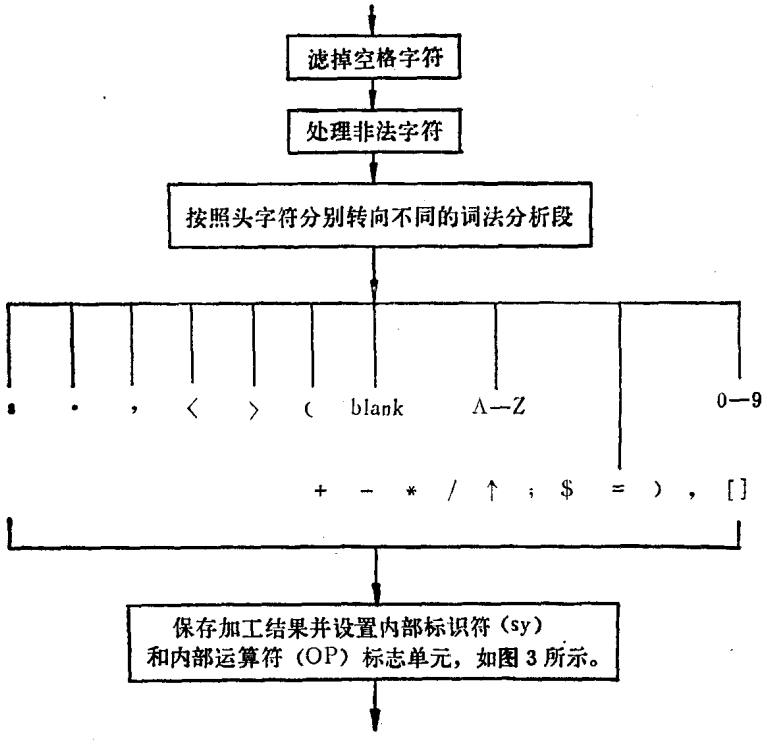

图 2　扫描与词法分析流程示意图

扫描过程（insymbol）利用 sy、op、id、val 以及 lgth 5 个全程量与语法分析程序通讯。其中 val 是记录型变量，它被用来存放整型或实型数和字符串。在表1中，列出了单词符号和扫描加工的结果信息。这里是一个简表，列出部分单词符号。

表 1　单词符号与加工结果简表

首字符	单词符号	语　义	加工结果sy	加工结果op	其　　它
字　母	保留字	保留字	保留字后接sy	大部分是noop	少数是inop，andop
字　母	标识符	标识符	ident	noop	规格化为8字符长，存放在id中
数　字	整　数	整　数	inconstsy	noop	数值在 ival 中
数　字	实　数	实　数	realconstsy	noop	字符在 rval 中
〃	〃字符序列〃	串常数	stringconstsy	noop	串在 sval 中，长度在 lgth 中
:	:	标　号	colon	noop	
:	:=	赋值号	becomes	noop	
.	..	冒　号	colon	noop	例如：〔1..5〕

<	<=	关系符	relop	leop	
.	.	结束标志	period	noop	
<	<	关系符	relop	leop	
<	<>	关系符	relop	neop	
>	>=	关系符	relop	geop	
>	>	关系符	relop	gtop	
(	(	左括号	lparent	noop	
其 它*			ssy[ch]	sop[ch]	例如：ssy[+]=addop, sop[+]=plus
空			othersy	noop	

* 系指 +, −, *, /, =,), [,], ;, ,, :, ↑, $ 等符号

除去扫描符号和词法分析这一必需的辅助过程，全部编译程序的实现，从初始化开始，经过对程序标题部分的处理，然后顺次加工标号说明，常量说明，类型说明，变量说明，过程说明（可能又要递归进入新层次），接着是对程序实体生成目标代码。这一切工作由分布在不同层次上的近百个过程（功能大小，程序繁易当然是不同的）来完成。

（三） 说 明 的 处 理

Pascal 语言所描述的程序由首部和分程序组成。分程序又包含说明和语句两个部分。说明部分定义所有局部于该程序的对象；它由标号说明、常量说明、类型说明、变量说明和过程与函数说明五个部分构成。说明的处理是编译程序工作的重要阶段。建立名字表是本阶段处理的核心，名字表给代码生成提供了关键信息。由于 Pascal 许可多种构造的数据类型，因此名字表结构要稍为复杂一些。在此，利用 Pascal 所具有的记录结构来描述名字表及其属性是较为方便的。每一登记项及其属性主要由两个记录结构来描述：标识符基本状况记录和结构的构造属性记录。标识符基本状况记录（简称标识符记录）的定义是：

```
identifier = PACKED RECORD
        name : alpha; llink, rlink : ctp; idtype : stp; next : ctp;
        CASE  klass : idclass OF
            konst : (values : valu);
            vars  : (vkind : idkind; vlev : levrange; vaddr : addrrange);
            field : (fldaddr : addrrange);
            proc,
            func  : (CASE pfdeckind : declkind OF
                        standard : (key : 1..15);
```

```
                declared : (pflev : levrange;
                            pfname : integer;
                            CASE pfkind : idkind OF
                            actual : (forwdecl,
                                      extern : boolean)))
        END;
```

结构的构造属性记录（简称结构记录）的定义如下：

```
structure = PACKED RECORD
        marked : boolean;
        size : addrrange;
        CASE form : structform OF
            scalar : (CASE scalkind : declkind OF
                        declared : (fconst : ctp));
            subrange : (rangetype : stp; min, max : valu));
            pointer : (eltype : stp);
            power : (elset : stp);
            arrays : (aeltype, inxtype : stp);
            records : (fstfld : ctp; recvar : stp);
            files : (filtype : stp);
            tagfld : (tagfieldp : ctp; fstvar : stp);
            variant : (nxtvar, subvar : stp; varval : valu)
        END;
```

其中，stp = ↑structure；ctp = ↑identifier；其余各个元素的类型定义请看本书第二部分编译程序文本中的类型说明部分。这里不一一列举了。

此外，为了记录和区分不同层次的标号表和名字表，本编译程序采用了一个变量display，它是以记录作为元素的一维数组型变量，我们称它为显示表。显示表的格式和内容如下：

```
display : ARRAY [disprange] OF
            PACKED RECORD
                fname : ctp; flabel : lbp;
                CASE occur : where OF
                    crec : (clev : levrange;
                            cdspl : addrrange);
                    vrec : (vdspl : addrrange)
            END;
```

其中，where = (blck, crec, vrec, rec)，因此标志域occur（出现位置）可以取上述枚举量中的任何一个，它用以区分显示表是在何种情况下建立的，并在相应的域中保存有关信息。

blck　它标志编译进入对程序、过程或函数的处理时，这时标识符是通常的变量标识符。

rec　处理记录类型定义时，它标志本显示表登记项所指向的名字表是由记录中各个域元素标识符所组成。

crec　进入处理 WITH 语句时，当该记录型变量不是传送地址形式参数时，设立此值。

vrec　进入处理 WITH 语句时，当该记录型变量是传送地址型形式参数，设置此值。

供编译处理域元素的计算用。

数组 display 的下标是标识符 top，它的值就代表对应的层次数。本编译程序开始运行时，首先把35个标准名字登记到名字表中，它们构成为第 0 层名字表。这些标准名字分别属于 types（如 integer, real, …），konst（如 false, nil, …），vars（如 input, output, …），proc（如 get, read, …），func（如 abs, eof, …）等不同种类。它们是系统赋予特定含义的。用户的变量名字实际上是从 top = 1 开始。显示表中登记项的元素 fname 的内容给出该层名字表第一个名字的位置，而元素 flabel 的内容则指向该层标号表的起始位置（如果存在有局部标号）。同一层内全部名字的标识符记录利用左链和右链联结在一起形成二叉树结构。而同一类型的名字之间又通过顺序链（next）进行联结。图 3 示意性地表达名字表和标号表的总体结构。

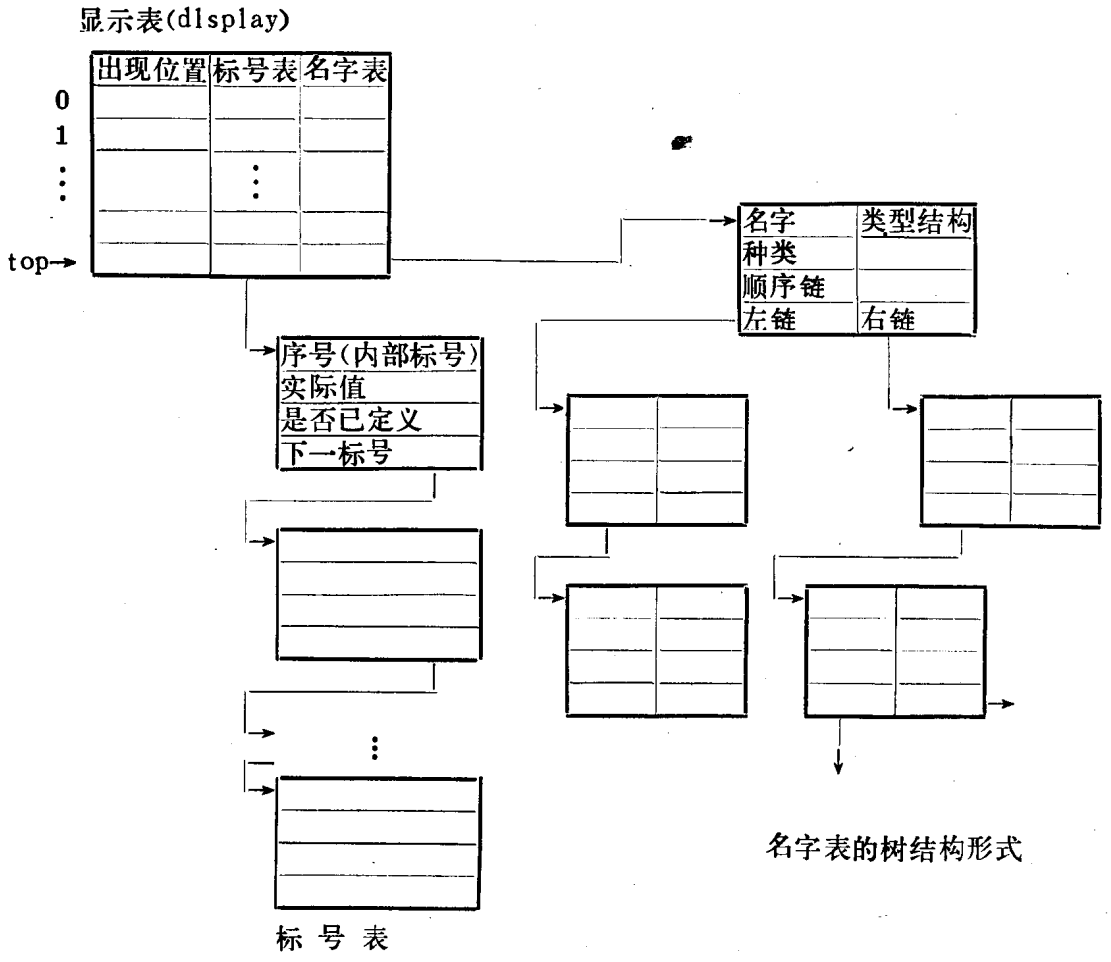

图 3　名字表和标号表结构示意

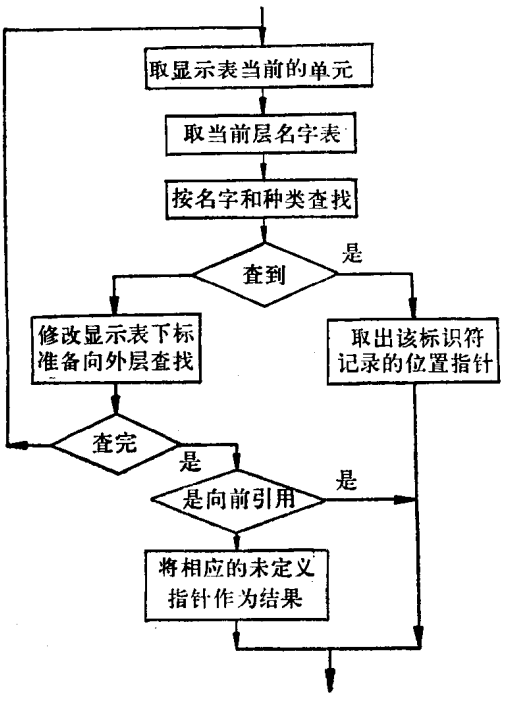

图 4　从名字表中查找标识符

对于名字表中的一个登记项的标识符基本状况记录和结构记录之间的关系，以及它们之中各个字段的含义，将分别结合各项具体内容加以说明。

由于名字表是以二叉树结构方式组织的，因此它的建立和查找方法都比较简单。图 4 和图 5 分别给出从表中查找特定的名字，和把名字登录到表中的算法框图。

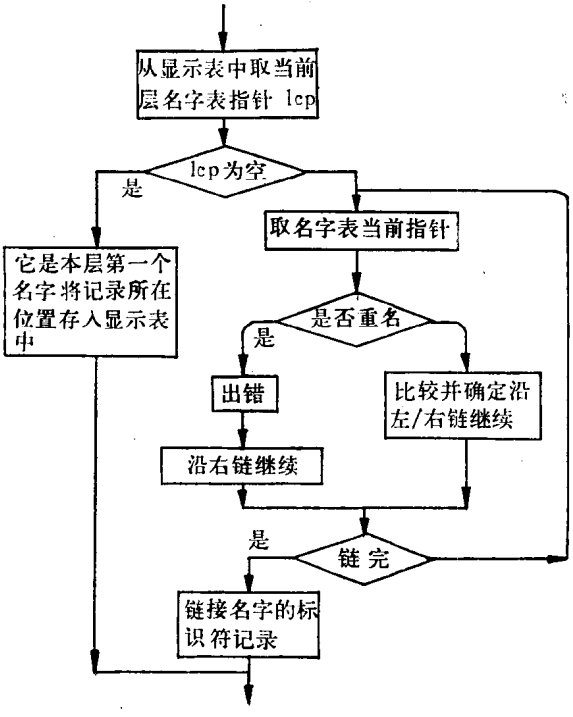

图 5　标识符填入名字表

下面分别逐项介绍各类说明的处理，大体上按：语法描述，算法示意，处理结果等三部分内容来叙述。

标　号　说　明

语法描述：见图6。

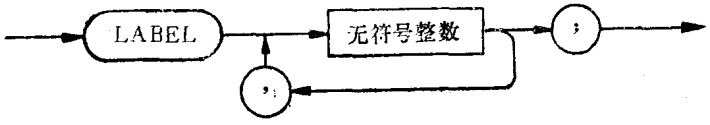

图 6　标号说明的语法描述

算法框图：见图7。

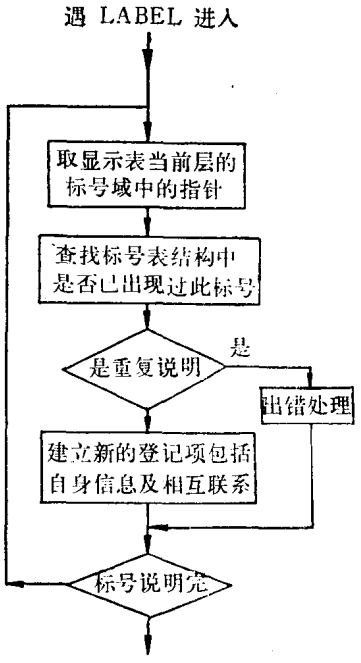

图 7　标号说明的处理

加工结果　P4 的标号限定为无符号整数，而且又规定不允许由内层转向外层所定义的标号，因此编译时只需在当前层中处理，算法简单，加工所得之标号表结构也很清晰（如图3所示）。

常　量　说　明

语法描述：见图8。

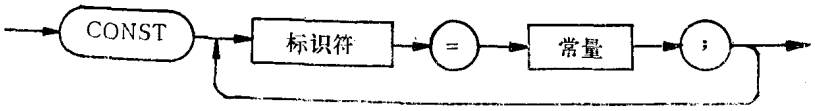

图 8　常量说明的语法描述

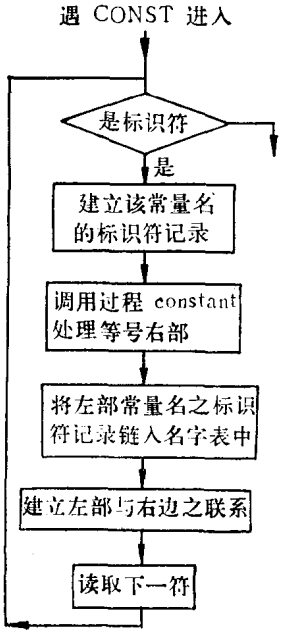

图 9　常量说明的处理

算法示意：见图9。

其中，过程调用 constant，它是用以区分右部的字符串常量、无或有符号整数或实数、以及标识符常量，在此标识符前允许有符号。本过程提供所处理的常量之类型结构记录指针和值记录作为结果，以便建立与左部之联系。

加工结果：如图10所示，左部常量标识符全部进入名字表，通过类型指针实现与右部的类型结构记录相联系。而右部之数值（如果是标识符，最后也是取其值）则存放在标识符记录的值域中。

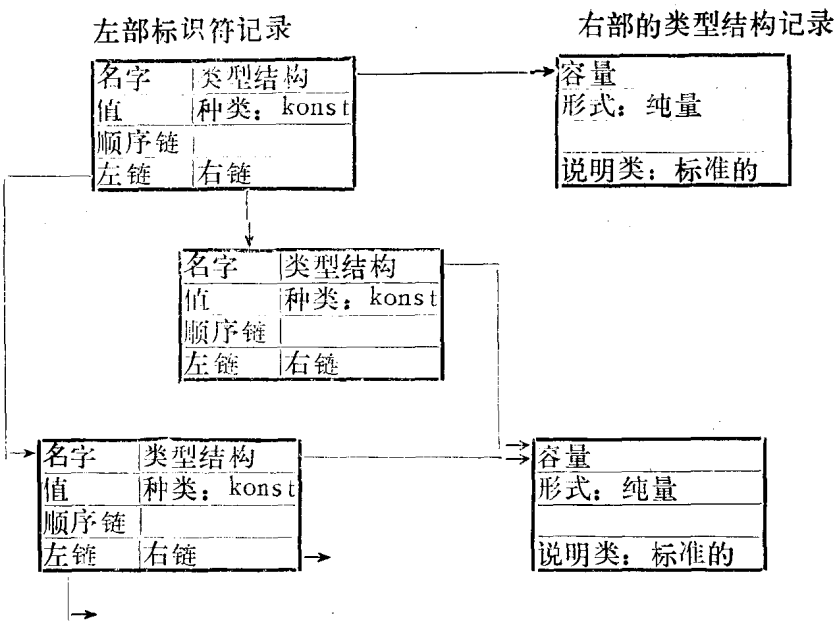

图 10 常 量 说 明 结 构

类 型 说 明

前已提及，具有多种数据结构是 Pascal 语言的一个重要特点。除去整型、实型、字符型之外，它还采用特定的语法——类型说明——来定义多种数据结构。从类型说明的语法图（见图11）可以看出，它与常量说明形式上类似，因而分析思路也大体类似，只不过右部处理更为复杂而已。结果结构的基本格式也类似，它包含出现在名字表中的标识符记录，这时是类型标识符，和一个类型结构指针，它指向所定义的数据结构，而这一结构要远比常量说明复杂。

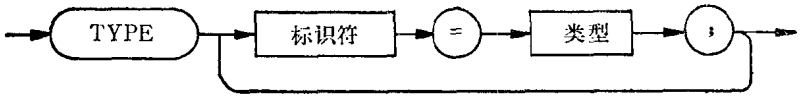

图 11 类型说明的语法描述

类型说明的语法如图11所示，下面重点介绍的是说明的等号右端部分类型的处理，它是由名为 typ 的过程来实现的。

过程 typ 的功能是分析各种类型的数据结构，如：记录、集合，……等等，并建立起它们相应的结构记录，它就是被说明的类型标识符的属性，供建立变量的标识符名字表时用作相应的属性。

本编译程序许可的数据类型有（1）非构造型，（2）构造型两大类。其中，非构造型又可分为标准纯量型和用户定义型两种。构造型包括有数组，集合，记录，文件等。

类型说明右端‘类型’的分析由过程 typ 来处理，它可分为两大部分。一部分用以处理非构造型（其中系统定义的标准纯量型，均由编译系统在开工初始化时建立其结构记录）。

另一部分用以处理构造型。为有助于了解 typ 过程，给出了它的总框图（见图12）。然后，再分别描述各种数据结构的加工。

需要注意的是，在 typ 中对非构造型处理是通过调用过程 simpletype 来完成的。在图12中给出了子界型和枚举型的处理，仅仅是为了完整和简明。

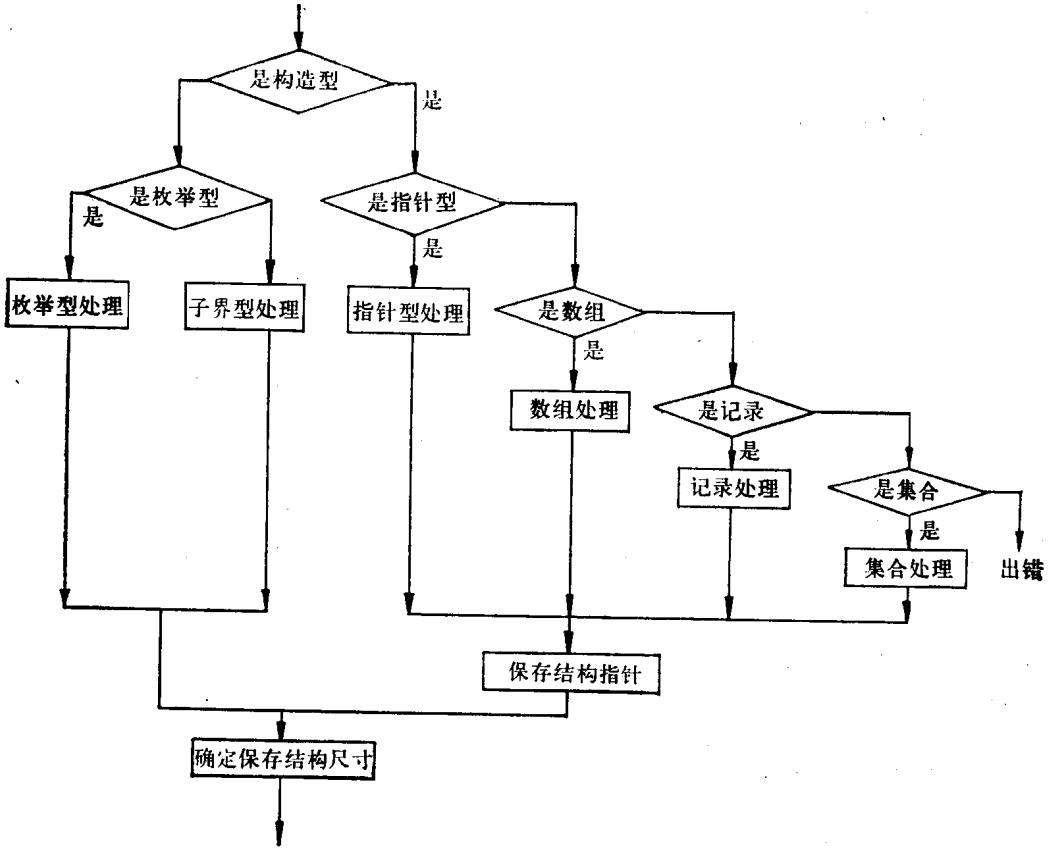

图 12　过程 typ 总框图

1. 枚 举 型

语法描述：见图13。

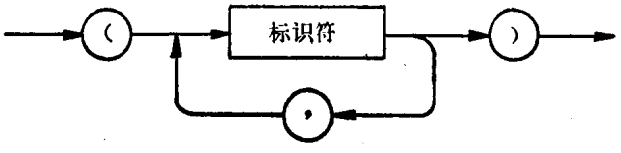

图 13　枚举型的语法描述