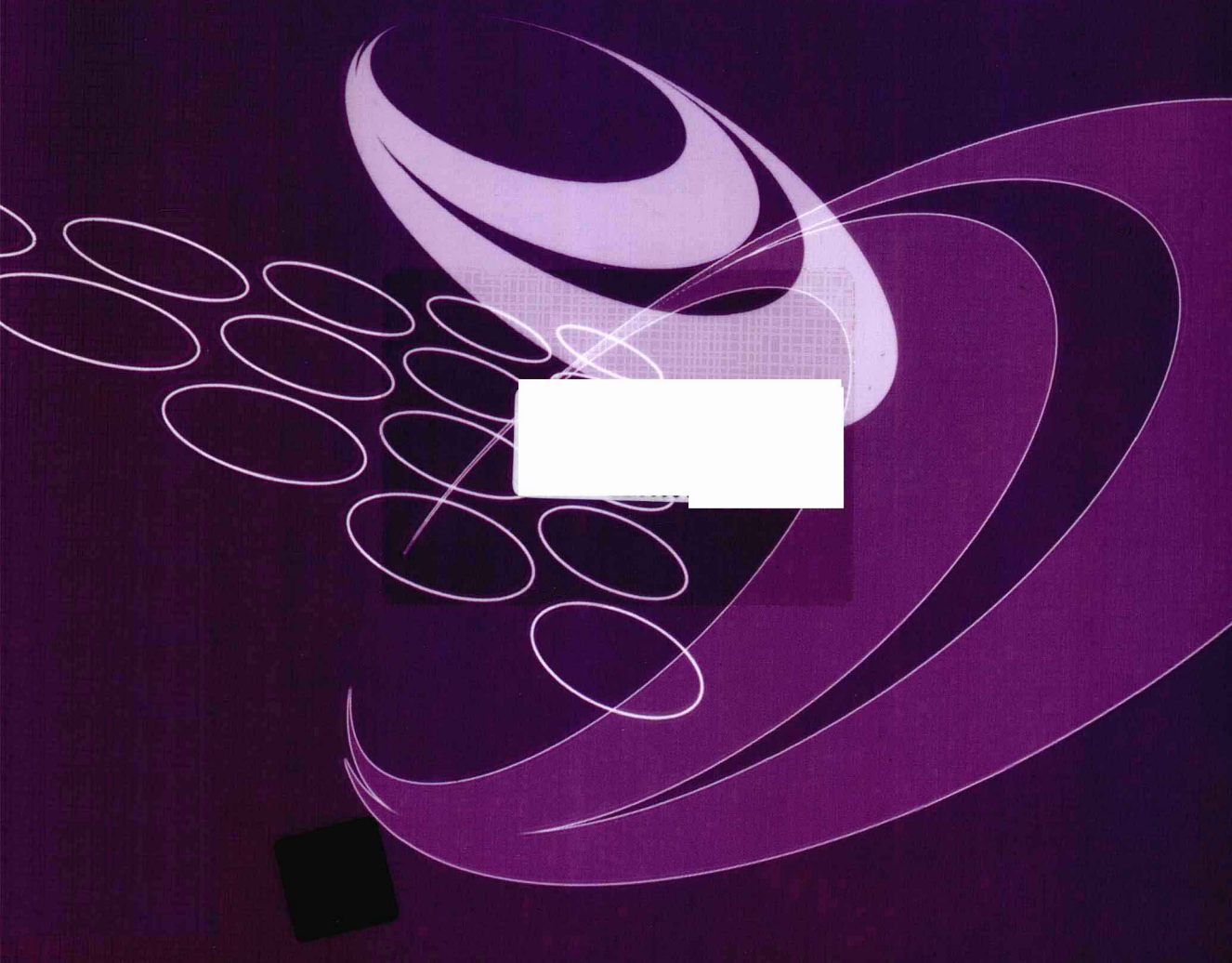
普通高等院校
计算机专业（本科）实用教程系列

编译原理实用教程

（第2版）

温敬和　编著



清华大学出版社

普通高等院校计算机专业(本科)实用教程系列

编译原理实用教程

(第2版)

温敬和　编著

清华大学出版社
北　京

内 容 简 介

全书共分7章，主要介绍编译程序的基本原理和实现方法。内容包括：词法分析，形式语言和自动机的基本概念，语法分析，符号表和静态内存分配，语法制导翻译和中间代码产生，目标代码生成。本书还介绍了作者本人的一些工作成果，如LR分析法在词法分析器自动构造中的应用，语法制导翻译在汇编程序自动构造中的应用。为了方便读者学习，各章都安排了一定数量的习题，并配有习题答案。

本书附录B中的“课程实习指导”向读者提供了一个较为完整的、切实可用的“编译原理”课程实习方案，并附有参考程序，可供有关教师选用或参考。

本书可作为本科院校计算机专业“编译原理”课程的教材，也可供有关教师、研究生以及从事计算机软件设计和开发人员参考。

图书在版编目（CIP）数据

编译原理实用教程/温敬和编著. —2版. —北京：清华大学出版社，2013.4
普通高等院校计算机专业（本科）实用教程系列
ISBN 978-7-302-31243-7

Ⅰ. ①编… Ⅱ. ①温… 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（2013）第002291号

责任编辑：闫红梅　薛　阳
封面设计：张　昱
责任校对：焦丽丽
责任印制：李红英

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦A座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn，010-62795954

印 刷 者：北京富博印刷有限公司
装 订 者：北京市密云县京文制本装订厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：14　　**字　　数**：341千字
版　　次：2005年4月第1版　2013年4月第2版　　**印　　次**：2013年4月第1次印刷
印　　数：1～3000
定　　价：25.00元

产品编号：048097-01

前　言

1982年2月本人毕业于上海交通大学，2010年1月退休，在上海第二工业大学工作了近三十年。在此期间，主要从事“编译程序”和“算法”这两门学科的教学和科研。2005年4月清华大学出版社出版了由本人编著的《编译原理实用教程》，该书至今仍用于我校和国内其他普通高等院校“编译原理”课程的教学。该书从脱稿至今已近十年，先后共印刷了1万册左右。虽然印刷的数量不大，但是90%是外校师生所使用的，说明书的质量得到了同行的认可。

在第2版中，书的章节基本没有变化，仅删除了原书中的5.10.3小节(5.10.3 LR分析控制程序的修改)，增加了6.11节(6.11 自上而下分析制导翻译概述)。做出上述调整，主要考虑用于词法分析的LR分析控制程序修改不大，一是增加了token数组，用于记录构成单词的字符。在执行移进操作时，除完成规定动作外，还应将当前字符移入token数组；二是把“出错”理解为找到单词尾。对于熟悉LR分析控制程序工作原理的读者，在理解上不会有困难。在后继章节中，对于用于词法分析的LR分析控制程序有详细介绍，没有必要单独列出。为了完整，在6.11节简略讨论了自上而下分析制导翻译技术。原书中的附录A和附录B合并为新书的附录A。原书的附录C删除，改为下载文件。原书的附录D改为新书的附录B。

在第2版中，各章节的知识点没有变化，增加了算法伪代码描述，对原书各章节中的所有源程序都做了比较大的修改。在原书中，算法除文字简单描述外，基本用源程序表达，这样对算法的描述和理解有可能受到语言细节的束缚。在本书中，增加了算法伪代码描述，这样可避免语言的限制，更容易表达算法的基本思想。考虑有些读者编程经验不足，源程序仍保留了下来，但在编排上做了改进，使其更容易阅读和理解。在本书中出现的源程序，除附录B中两个程序外，都可以从清华大学出版社指定网站下载。另外，由本人编写的“编译原理”课程电子教案和试卷集锦可以从“中国高等学校教学资源网”下载。

在写第1版时，主要考虑程序的正确性。在再版中，力求使程序写得更简洁、更易理解，并且注意前后统一。例如，本书介绍了三个词法分析器，它们是Lex1、Lex2和Lex3。三个词法分析器都是由预处理程序和扫描器(单词识别程序)两个部分构成。预处理程序是同一个，差异在于如何实现扫描器。Lex1是利用状态转换图来实现的，Lex2是利用确定有限自动机来实现的，而Lex3是利用LR分析法来实现的。扫描器的程序结构大同小异，读者只需关注单词识别时所使用的技术和方法。

借此机会，向清华大学出版社表示感谢。是清华大学出版社向本人提供了机会，使我能够在退休之后，继续为高等学校计算机教育尽自己微薄之力。继2011年6月的“算法设计与分析”出版之后，这是本人主编的第2本教科书。

上海第二工业大学计算机与信息学院教师王娜参与了本书各章的编写(包括习题答案)，上海第二工业大学成人与继续教育学院教师杨坤参与了各章源程序的编写。

温敬和

2012年秋

目　录

第1章　编译系统概述

在科学技术高速发展的今天，信息技术的使用已广泛普及，计算机已经作为必不可少的工具融入人们的日常工作和生活。你可能正在使用浏览器观看奥运会最新报道，或者正在使用文字编辑器书写家信。如果你是一个计算机专业的学生，可能正在考虑如何使用某一程序设计语言（例如C语言）来实现类似于计算器这样的应用程序。当然在动手之前，需要认真学习这种程序设计语言的使用方法，然后使用类似于文字编辑器这样的软件来书写程序，就好像书写家信一样。所不同的是，家信是按照自然语言来书写的，用文字描述要告诉对方的事情；而程序是按照程序设计语言的规定来书写的，它也是用单词描述用户的想法，而告知的对方是计算机，该计算机称为“目标计算机”。所写的程序称为“源程序”，书写源程序所使用的语言称为“源语言”。源程序通常存放在文本文件中，文本文件是由字符构成的，所以文本文件又称为ASCII码文件，文件的扩展名通常为txt。我们所书写的程序不能直接交给计算机执行，需使用一种称作“翻译程序”的工具软件，将源程序翻译成计算机可识别的机器指令程序。计算机的机器指令称为“目标语言”，由机器指令构成的程序称为“目标程序”。机器指令就是二进制数，存放二进制数的文件称为二进制文件，文件的扩展名通常为exe。此时，可将存放在exe文件中的机器指令程序交给计算机执行。

现在我们知道了翻译程序的作用，翻译程序是这样一个程序，它能够把某一种语言程序（称为源语言程序，简称源程序）改造成另一种语言程序（称为目标语言程序，简称目标程序）。源语言是诸如FORTRAN、ALGOL、Pascal和C这样的高级语言，而目标语言是诸如汇编语言或机器语言这样的低级语言。翻译程序通常称为“编译程序”，有些翻译程序在翻译过程中并不产生完整的目标程序，而是翻译一句，解释执行一句，这样的翻译程序称作“解释程序”。如何将源程序翻译成目标程序呢？这就是本书讨论的主题，在回答这个问题之前，先简单回顾一下程序设计语言的发展史。

最早的计算机程序是用机器语言编写的，或者说是用二进制数编写的。假设要计算算术表达式3＊16＋2的值，实现该计算的机器语言程序如下所示（该计算机的系统结构详见本书第7章）：

```
0    2203                    //十六进制
1    8210
2    2602
3    6101
4    1000
5    f000
```

从上述二进制机器语言程序可知，使用机器语言编写程序是极其困难的，这项工作需由专业人士完成。它的难度不仅仅在于二进制数的直观性差，还因为在指令中存在绝对地址，在机器语言程序中增加或减少一条指令，将会引起指令修改的连锁反应。除此之外，编程者还要协调内存的使用。

随后出现了汇编语言,在汇编语句中,二进制数被符号取代。由于符号的引入,提高了程序的可理解性。性能较好的汇编语言,可用符号名来表示存储地址和汇编语句序号,这样避免了在汇编语句中绝对地址的出现,在一定程度上降低了程序编写和修改的难度。上述机器语言程序可用汇编语言(该计算机的汇编语言详见本书第 7 章)改写如下:

```
0    Load R0,3
1    Mul R0,10                          //10₁₆=16
2    Load R1,2
3    Add R0,R1
4    Write R0
5    Halt
```

尽管在汇编语言中增加了伪指令和宏功能,但是汇编语句和机器指令基本上是一对一的,所以汇编语言的编程效率没有质的提高,和机器语言一样,汇编语言依附于目标计算机。当然计算机是不能识别用汇编语言编制的程序的,需要用汇编程序将其译成机器指令,可以说汇编程序是编译程序的雏形。

1954 年 FORTRAN Ⅰ语言问世,标志着计算机高级语言的诞生。高级语言接近于英语,相当于工程语言,它完全独立于具体计算机。高级语言的出现极大地提高了编程效率,大幅度地降低了编程难度。上述汇编语言程序若用高级语言(C/C++ 语言)书写,可简单表示为:

```
1    void main()
2    {
3        cout<<3*16+2;
4    }
```

编译过程恰恰和程序设计语言发展过程相反,它是将高级语言程序变换成汇编语言程序或者机器语言程序。在高级语言中存在各种语句,并且允许语句嵌套使用,面向过程的高级语言进一步发展为面向对象的高级语言,可想而知,将高级语言程序翻译成目标语言程序的过程是相当复杂的。整个 20 世纪 50 年代,编译程序的编写一直被认为是一个难题,第一个 FORTRAN 语言编译程序整整花了 18 人年才得以实现。

目前,我们已经系统地掌握了高级语言的编译理论和技术,编译理论和技术是计算机科学中发展得最为迅速的一个分支,已经形成一套比较成熟的理论和方法,这些理论和方法指导人们如何设计和构造编译程序。本课程的目的是介绍设计和构造编译程序的基本原理和基本方法,是一门引论性课程。本书仅仅包括了一些较为常用的编译技术和方法,但是力求介绍一些新的编译技术和方法。

典型的编译程序工作过程是:输入源程序,对它进行加工处理,最后输出目标程序。由于整个加工处理过程极其复杂,因此最好把整个翻译过程看成由一系列不同阶段所组成,每一个阶段完成一个特定任务。整个过程如图 1.1 所示,编译程序的结构基本上是按照这个流程来设计的。

现以一个算术表达式 3+abc*128 的翻译为例来说明编译过程。

1. 词法分析

执行词法分析的程序称为词法分析器,词法分析依据的是语言的构词规则。词法分析

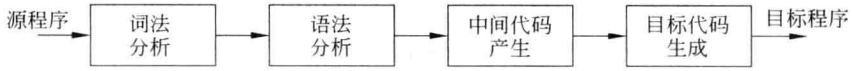


图　1.1

器从文件读入源程序，由字符拼接单词。每当识别出一个单词，词法分析器就输出这个单词的内部码。

表达式 3＋abc＊128 的单词内部码为：

```
('x',"3") ('+',) ('i',"abc") ('*',) ('x',"128")
```

单词的内部码由两部分组成，一是单词的种别，二是单词的值。单词的种别通常用整数码表示，为直观，可用字符表示，字符相当于整数。并不是所有的单词都具有值，只有当一个种别可能有多个单词时，才用单词的值予以区别。在上例中，整数的单词种别用字符'x'表示，因为在源程序中可能存在多个整数，所以还需给出它的值，单词值这里用字符串"3"表示，当然也可以用数值表示。同理，标识符 abc 用('i',"abc")表示，其中'i'是标识符的单词种别，而"abc"是它的值。为了形式上统一，用字符串"NUL"表示单词没有值。在上例中，('＋',)改用('＋', "NUL")表示，('＊',)改用('＊', "NUL")表示。

单词的种别在语法分析时使用，单词的值在语义分析（中间代码产生）时使用。所以在语法分析时，表达式 3＋abc＊128 的语法结构应表示为：

```
x+i*x
```

2. 语法分析

执行语法分析的程序叫做语法分析器。语法分析的任务是：根据语言的语法规则，将词法分析器所提供的单词种别分成各类语法范畴。

例如，当接受符号串 x＋i＊x 时，语法分析器最终应识别出这是一个算术表达式。识别过程相当于建立一棵语法树，如图 1.2 所示。

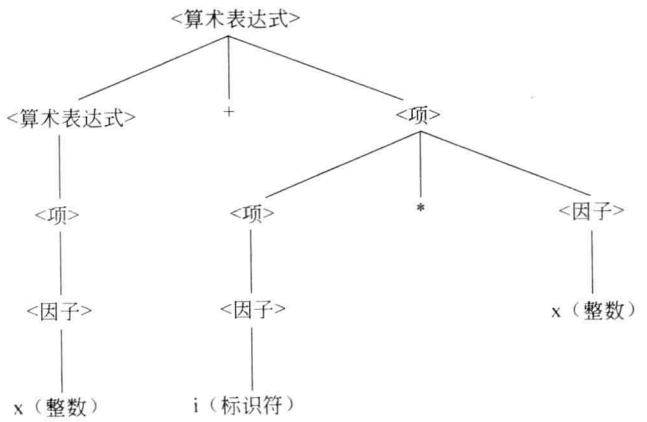


图　1.2

3. 中间代码产生

中间代码产生有时称为语义分析，执行中间代码产生的程序称为中间代码产生器。它

的任务是：按照语法分析器所识别出的语法范畴产生相应的中间代码，并建立符号表、常数表等各种表格。把语法范畴翻译成中间代码，依据的是语言的语义规则。中间代码非常接近于机器指令，但和具体机器无关，常用的中间代码有三元式和四元式。

表达式 3＋abc＊128 可译成如下四元式：

(1) (＊,&abc,&128,&T1)

(2) (＋,&3,&T1,&T2)

其中，&abc 表示标识符 abc 在符号表中的入口。

符号表用于记录源程序中出现的标识符，符号表的结构示意如表 1.1 所示。一个标识符往往具有一系列的语义值，它包括标识符的名称、标识符的种属、标识符的类型、标识符值的存放地址等。每个标识符在符号表中都有一项记录，用于记录标识符的这些信息。在四元式中填写的是标识符在符号表中的记录地址，通常称为符号表入口，借助“&”来表示。

表 1.1

	内存地址	符号名	种属	类型	…	…
&abc ⟶	未分配	abc	简单变量	整型	…	…
	未分配	T1	简单变量	整型	…	…
	未分配	T2	简单变量	整型	…	…

T1 和 T2 是在翻译过程中由编译程序引进的临时变量，它们在符号表中的入口用 &T1 和 &T2 表示。

&128 表示整数 128 在常数表中的地址，常数表用于记录在源程序中出现的常数。假定每个整数在常数表中占 2 个字节，每个实数在常数表中占 4 个字节，常数表的结构示意如表 1.2 所示。

表 1.2

	常数的二进制值
	3
&abc ⟶	128

4. 目标代码生成

执行目标代码生成的程序称为目标代码生成器。这个阶段的任务是：根据中间代码和表格信息，确定各类数据在内存中的位置，选择合适的指令代码，将中间代码翻译成汇编语言或机器指令，这部分工作和计算机硬件有关。

假设目标机器的指令格式如下所示。

(1) 直接地址寻址：

op Ri,M　　(Ri)op(M)→Ri

(2) 寄存器寻址：

op Ri,Rj　　(Ri)op(Rj)→Ri

(3) 直接数寻址：

op Ri,C　　(Ri)opC→Ri

其中，R 表示寄存器(i 和 j 表示寄存器号)，M 表示内存地址(可用符号表示)，C 表示常数。

上述算术表达式最终形成的汇编语言程序示意如下：

```
0    Load R0,abc
1    Mul R0,80                          //80₁₆=128
2    Store R0,T1
3    Load R0,3
4    Add R0,T1
5    Store R0,T2
```

在编译的每个阶段都可能发现源程序中的错误，最简单的处理是：终止编译程序工作，指出错误的地点和错误的类型。目前有些高级语言的翻译程序同时具有解释和编译两种功能，在调试时采用解释方式，在调试过程中若发现错误，翻译程序立即暂停工作，并用特殊记号指示当前错误的位置，供程序员修改源程序参考；当调试完成后，可利用翻译程序的编译功能，将源程序译成机器码程序。机器码程序通常存放在扩展名为 exe 的文件中，它可在操作系统环境下脱离翻译程序直接运行。

由于在编译程序的内部引入了中间代码，这样可将编译程序分为两个相互独立的部分。词法分析器、语法分析器和中间代码产生器称为编译程序的前端，它们依赖于被编译的源语言，和目标机器无关；目标代码生成器称为编译程序的后端，和源语言无关，仅仅和目标机器有关。为一个源语言构造好前端后，若要在某一个特定计算机上构造该源语言的编译程序，只要构造这个目标机器的后端即可。相反，已构造了一个高质量的后端，若要在同一台目标机器上为另一源语言构造编译程序，只要构造该源语言的前端即可。

习　　题

名词解释

(1) 源语言　(2) 源程序　(3) 目标语言　(4) 目标程序
(5) 翻译程序　(6) 编译程序　(7) 解释程序　(8) 汇编程序
(9) 词法分析　(10) 语法分析　(11) 中间代码产生　(12) 目标代码生成
(13) 符号表　(14) 常数表　(15) 编译程序前端　(16) 编译程序后端
(17) 文本文件　(18) 二进制文件

(习题答案略，可参考本章内容)

第2章 词法分析

人们理解一篇文章是在单词级别上来思考的。同样，编译程序也是在单词级别上来分析和翻译源程序的。因此，词法分析是编译的第一步。

2.1 词法分析器的设计考虑及手工构造

从操作系统的角度来看，源程序是由字符构成的，以文本文件的形式存储于外存；而从编译程序的角度来看，源程序是由程序设计语言的单词构成的，单词是程序设计语言不可分割的最小单位。词法分析的任务是：从文件读入源程序，由字符拼接单词。每当识别出一个单词，就产生这个单词的内部码。这一章节将讨论词法分析器的手工构造。

2.1.1 单词类型及二元式编码

标准 ASCII 码共有 128 个字符，其中 94 个为图形字符，ASCII 码范围为 33("!")～126("～")，程序设计语言所使用的字符是它的子集，标准 FORTRAN 语言仅使用了 47 个图形字符。除图形字符和文件结束控制符($1A_H$)外，源程序文件中还有 4 个控制字符，它们是空格(20_H)、回车($0D_H$)、换行($0A_H$)和制表符(09_H)，这 4 个控制字符用于对文本显示格式的控制，并不是程序的组成部分。这里要注意的是，回车和换行在源程序文件中是用两个字符来表示的，它们是 $0D_H$ 和 $0A_H$，而用高级语言(例如 C 语言)将其读入内存后，回车和换行在内存中用一个字符(换行 $0A_H$)表示。这是在用高级语言编写词法分析器时，常被人忽略而导致错误的原因。

任何程序设计语言的单词都可将其分为 5 种类型，它们是基本字、标识符、常数、运算符和界符。

(1) 基本字：例如 real、integer 等。

(2) 标识符：通常是以字母开始的数字字母串，用于定义简单变量、标号等。

(3) 常数：例如整数(123)、实数(123.456)等。

(4) 运算符：例如＋、＊、/等。

(5) 界符：例如 ；、(、)等。

其中，基本字、运算符和界符对于某一程序设计语言来说是确定的，而源程序中的标识符和常数的个数是不确定的，随源程序而异。经词法分析后，单词用二元式(code，val)来表示，code 表示单词的种别，val 表示单词的值。单词种别通常用整数码表示，为了直观，在本书中单词种别采用单个字符表示，且尽可能采用与原单词最接近的形式(注意字符内部码是一个整数)。单词的种别表示了单词的语法特性，在语法分析时使用；单词的值表示了单词的语义特性，在语义分析时使用。

一种语言的单词如何划分种别？怎样编码？主要取决于后续处理的方便。通常将标识符归为一种，常数按类型分种，基本字、运算符和界符采用一符一种。若一个种别含有多个单词，对于此类单词除给出种别外，还需给出它的值。在本书中，单词的值用字符串来表示。如果一个种别仅包含一个单词，那么种别就可代表该单词，无须给出单词值。但是，为了后续处理的方便，在本书中无用的单词值用字符串“NUL”表示。设有某一程序设计语言，它的部分单词二元式编码如表 2.1 所示。

表 2.1

单　词	二元式编码	举例说明
integer	('a',"NUL")	
real	('c',"NUL")	
begin	('{',"NUL")	
end	('}',"NUL")	
标识符	('i',标识符名)	变量 r 的单词二元式为('i',"r")
无符号整数	('x',字符串形式数字)	整数 2 的单词二元式为('x',"2")
无符号实数	('y',字符串形式数字)	实数 3.14 的单词二元式为('y',"3.14")
=	('=',"NUL")	
*	('*',"NUL")	
+	('+',"NUL")	
(	('(',"NUL")	
)	(')',"NUL")	
,	(',',"NUL")	
;	(';',"NUL")	

用该程序设计语言编制的计算圆柱体表面积的源程序(输入输出略)如下所示：

```
1    begin
2        real r,h,s;
3        s=2*3.14*r*(r+h)
4    end
```

经词法分析，源程序的单词二元式序列为：

('{',"NUL") ('c',"NUL") ('i',"r") (',',"NUL") ('i',"h") (',',"NUL") ('i',"s") (';',"NUL") ('i',"s") ('=',"NUL") ('x',"2") ('*',"NUL") ('y',"3.14") ('*',"NUL") ('i',"r") ('*',"NUL") ('(',"NUL") ('i',"r") ('+',"NUL") ('i',"h") (')',"NUL") ('}',"NUL")

注：在实际处理中，左右括号和逗号是没有的，数据用空格、Tab 或换行分隔。

2.1.2 源程序的输入及预处理

源程序通常以文件形式存于外存,首先要将其读入内存才可进行词法分析。早期编译程序是用汇编语言,甚至是用机器语言编写的,计算机的硬件配置远不能和今天相比,只能在内存设置长度有限的缓冲区,分段读入源程序进行处理。在编制程序时,必须考虑由于分段读入产生的问题。例如源程序中由多个字符构成的单词,有可能被缓冲区边界所打断(留作习题)。目前计算机所使用的内存已超过若干年前硬盘容量,计算机内存足以容纳全部源程序,故源程序可一次全部读入内存进行处理。

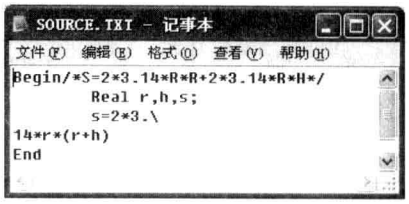


图 2.1

设源程序如图 2.1 所示,其中"\"为续行符。

源程序读入后,输入缓冲区的内容如下所示。

B	e	g	i	n	/	*	S	=	2	*	3	.	1	4	*	R	*	R	+	2	*	3

.	1	4	*	R	*	H	*	/	\n	\t	R	e	a	l		r	,	h	,	s	;	\n

\t	s	=	2	*	3	.	\	\n	1	4	*	r	*	(	r	+	h	)	\n	E	n	d

\n	\0	\0	…	\0

目前使用的程序设计语言大都采用自由格式书写,允许在单词之间存在多余的空格、换行和制表符(Tab)。源程序通常还带有注释,注释的存在旨在使程序容易阅读,不是程序的必要组成部分。有些语言还提供续行功能(例如 C 语言中的续行符"\"),当一个单词过长(例如字符串常数),可分多行列出。对于 FORTRAN 和 COBOL 之类的语言,源程序还受到书写格式的限制。词法分析可在输入缓冲区上直接进行,但从程序设计的角度来讲,若把输入串预处理一下,则单词识别就比较容易,故词法分析器通常由预处理程序和扫描器(单词识别程序)两部分组成。预处理主要工作包括以下内容。

(1) 删除注释。

(2) 删除续行符,包括后续换行符($0A_H$)。

(3) Tab 作用相当于多个空格,换行符、Tab 和空格具有界符作用,预处理时通常予以保留。在后面的分析中可以看到,它们的存在反而给单词识别带来方便。为了简化判断,可在预处理时,将换行符和 Tab 统一替换为空格。

(4) 大多数语言(除 C 语言)不区分大小写,可在预处理时,将大写字母变换成小写字母,或相反,以方便后续处理。

(5) 对于受书写格式限制的语言(例如 FORTRAN 和 COBOL 语言),还应识别标号区,给出语句标号。识别续行标志,把相继行连接在一起,给出语句结束符。

上述源程序经预处理后,扫描缓冲区中的内容如下所示:

b	e	g	i	n			r	e	a	l		r	,	h	,	s	;			s	=	2

*	3	.	1	4	*	r	*	(	r	+	h	)		e	n	d		\0	…	\0	\0

如果用高级语言编写预处理程序，输入和预处理可同时进行，无须输入缓冲区，将读入后经预处理的源程序直接送入扫描缓冲区。

算法 2.1 给出了一个用伪代码书写的预处理程序。它的作用为：从文件 Source.txt 读入源程序，去除源程序中注释和续行符，将 Tab 和换行符替换为空格，并将大写字母变换成小写字母。将经预处理的源程序存入扫描缓冲区 Buf，供扫描器识别单词。由于算法需要，在源程序尾部添加字符“#”，这是一个特殊的字符，以示源程序的结束。源程序中的注释用/*…*/标记，不允许嵌套使用，这和大多数高级语言规定相一致。

算法 2.1 Pretreatment

```
输入：源程序文件 Source.txt。
输出：扫描缓冲区 Buf[1..n]。
1    c0←'$';c1←Source.txt 的第一个字符      //c0 为前一个字符,c1 为当前字符
2    in_comment←false                       //状态标志,false 表示当前字符未处于注释中
3    i←1
4    while not eof("Source.txt") do
5        switch in_comment do
6        case false:                        //当前字符未处于注释中
7            if c0c1="/*" then              //进入注释,去除已存入扫描缓冲区的字符'/'
8                in_comment←true;i←i-1
9            else
10               if c0c1="\换行符" then
11                   i←i-1                  //去除已存入扫描缓冲区的续行符'\'
12               else
13                   if c1 是大写字母 then c1←c1+32
14                   if (c1=制表符) or (c1=换行符) then c1←空格
15                   Buf[i]←c1;i←i+1        //将字符存入扫描缓冲区
16               end if
17           end if
18       case true:                         //当前字符处于注释中,丢弃该字符
19           if c0c1="*/" then in_comment←false     //离开注释
20       end switch
21       c0←c1;c1←Source.txt 的下一个字符 //当前字符成为前一个字符
22   end while
23   Buf[i]←'#'
```

根据算法 2.1，用 C/C++ 语言编程如下：

```
1    #include <fstream.h>
2    void pretreatment(char filename[],char Buf[])
3    {
4        ifstream cinf(filename,ios::in);
```

```
    char c0='$',c1;                          //c0 为前一个字符,c1 为当前字符
    bool in_comment=false;                   //状态标志,false 表示当前字符未处于注释中
    cout<<"<源程序>"<<endl;
    int i=0;
    void *p=cinf.read(&c1,sizeof(char));     //从文件读第一个字符(包括控制字符)
     while(p){
        cout<<c1;                            //输出读入字符
        switch(in_comment){
        case false:                          //当前字符未处于注释中
            if(c0=='/' && c1=='*')           //进入注释,去除已存入扫描缓冲区的字符'/'
                in_comment=true,i--;
            else
                if(c0=='\\' && c1=='\n')     //去除已存入扫描缓冲区的续行符'\'
                    i--;
                else{
                    if(c1>='A' && c1<='Z')   //大写转换成小写
                        c1+=32;
                    if(c1=='\t' || c1=='\n')
                        c1=' ';
                    Buf[i++]=c1;             //将字符存入扫描缓冲区
                }
            break;
        case true:                           //当前字符处于注释中,丢弃该字符
            if(c0=='*' && c1=='/')           //离开注释
                in_comment=false;
        }//end of switch
        c0=c1;                               //当前字符成为前一个字符
        p=cinf.read(&c1,sizeof(char));       //从文件读下一个字符(包括控制字符)
    }//end of while
    Buf[i]='#';
}
void main()
{
    char Buf[4048]={'\0'};                   //扫描缓冲区
    pretreatment("source.txt",Buf);
    cout<<"<预处理结果>"<<endl;
    cout<<Buf<<endl;
}
```

程序运行结果如图 2.2 所示。

2.1.3 基本字的识别和超前搜索

有些语言(例如 FORTRAN 语言)对基本字不加保护,用户可以把它们用作普通标识符,这就使得基本字的识别相当困难。观察下面三个 FORTRAN 语言语句:

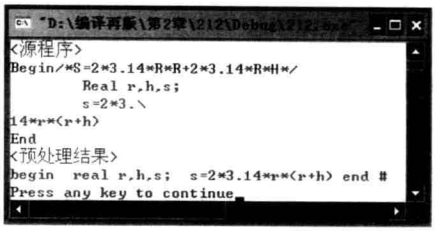


图 2.2

(1) IF(5.EQ.M)GOTO55。

逻辑 IF。当 M 等于 5,转移至标号为 55 的语句,否则顺序执行。

(2) IF(5)=55。

IF 为数组名,IF(5)为下标变量。

(3) IF(X+Y)110,120,130。

算术 IF。当 X+Y<0,转移至标号为 110 的语句;当 X+Y 等于 0,转移至标号为 120 的语句;当 X+Y>0,转移至标号为 130 的语句。

显然仅根据 IF 无法判断其为何种单词,可能是基本字,也可能是标识符。解决办法是超前搜索,一直扫描到右括号后面的字符。若该字符为字母 G 或 g,则为逻辑 IF;若为数字,则为算术 IF;若为=,则为标识符。

超前搜索导致词法分析器实现困难。为了降低词法分析器的复杂性,避免超前搜索,在实际实现中,大多数语言的编译程序对用户采取了以下两条限制措施:

(1) 所有基本字均为保留字,用户不能使用它们作为标识符。例如语句 IF(5)=55,由于 IF 为基本字,不能用作标识符,编译程序会指示该语句错误。

(2) 将空格、Tab 和换行符视为界符。在基本字、用户定义的标识符和常数之间,若没有运算符或界符,则至少用一个空格(或 Tab、换行符)加以分隔。这样空格、Tab 和换行符不再是没有意义的了,这就是为什么在词法分析预处理中将空格、Tab 和换行符保留下来的原因。例如 Pascal 语言语句:

```
FOR I:=1 TO 10 DO X:=X+1
```

不能写成:

```
FORI:=1TO10DOX:=X+1
```

对于后者,FORI、TO10DOX 和 X 被认为是标识符。

采用上述两条限制措施,对用户来讲是完全可以接受的,并且已成为程序员进行程序设计的惯例。词法分析器对于所有单词的识别,最多只要向前看一个字符就足够了。

2.1.4 状态转换图和词法分析器的手工构造

在本书中,词法分析器是以过程(函数)形式书写的,返回值是单词二元式。词法分析器可作为语法分析器的一个子程序,语法分析器每调用一次,词法分析器就回送一个单词二元式。也可编制一个程序,在程序中组织一个循环,反复调用词法分析器,将源程序改造成单