



School of Computer Science & Technology
Harbin Institute of Technology



第三章 词法分析

重点： 词法分析器的输入、输出，
用于识别符号的状态转移图的构造

难点： 根据状态转移图实现词法分析器的设计





第3章 词法分析

3.1 词法分析器的功能

3.2 单词的描述

3.3 词法分析程序的自动生成

3.4 本章小结



3.1 词法分析器的功能

- 功能：输入源程序，输出单词符号(token)。
即：把构成源程序的字符串转换成“等价的”
单词(记号)序列
 - 根据词法规则识别及组合单词
 - 对数字常数完成数字字符串到二进制数值的转换
 - 查填符号表
 - 删去空格字符和注释
 - 错误检查



3.1.1 单词的分类与表示 & 3.1.2 词法分析器的输出

一、单词的种类

1. 关键字:也称基本字, begin、end、for、do...
2. 标识符:由用户定义, 表示各种名字
3. 常数:整常数、实常数、布尔常数、字符串常数等
4. 运算符:算术运算符+、-、*、/等; 逻辑运算符not、or与and等; 关系运算符=、<>、>=、<=、>和<等
5. 分界符: , 、 ; 、 (、) ...

二、单词的内部形式

表示单词的种类，可用整数编码或记忆符表示

种别	属性值
----	-----

不同的单词不同的值

种别码是一种自主设计的编码方案，可以将一类单词编为一码，如所有关键字为一码，所有算符为一码；也可以根据单词的特性进行编码，如一个关键字一码，一个算符一个码

属性值指的是单词在计算机内的存放表形式，通常会是一个指针，用于进行单词间的区分



1、种别码--按单词种类分类

单词名称	类别编码	单词值
标识符	1	内部字符串
无符号常数(整)	2	整数值
无符号浮点数	3	数值
布尔常数	4	0 或 1
字符串常数	5	内部字符串
关键字	6	保留字或内部编码
分界符	7	分界符或内部编码
算符	8	算符或内部编码

2、种别码--按单词特性编码

单词名称	类别编码	单词值
标识符	1	内部字符串
无符号常数(整)	2	整数值
无符号浮点数	3	数值
布尔常数	4	0 或 1
字符串常数	5	内部字符串
FOR	6	—
DO	7	—
IF	8	—
ELSE	9	—
THEN	10	—
INT	11	—
CHAR	12	—
FLOAT	13	—
WHILE	14	—
.....	

2、种别码—按单词特性编码

单词名称

类别编码

单词值

.....

.....

.....

+

20

—

—

21

—

*

22

—

/

23

—

=

24

—

25

—

(

26

—

)

27

—

,

28

—

;

29

—

>

30

—

>=

31

—

==

32

—

.....

.....

.....

例3.1 语句if count>7 then result := 100; 的单词符号序列

(9, 0)

(1, 指向count 的符号表入口)

(30, 0)

(11, 7)

(10, 0)

(1, 指向result的符号表入口)

(24, 0)

(11, 100)

(29, 0)

跟实现有关

词法分析后:

1. 数字化表示
2. 对同类单词进行了合并
3. 打破了原来的关系

例3.1 语句if count>7 then result := 3.14; 的单词符号序列

(IF, 0)

(ID, 指向count 的符号表入口)

(>, 0)

(INT, 7)

(THEN, 0)

(ID, 指向result的符号表入口)

(=, 0)

(REAL, 3.14)

(;, 0)

记号表示

关于单词的自身属性值

- 对于关键字、界符、运算符来说，它们的词类编码就可以表示其完整的信息
- 故对于这类单词，其单词自身的属性值通常为空
- 而对于标识符，词类编码所反映的信息不够充分，标识符的具体特性还要通过单词自身的属性进行互相区分。
- 标识符的单词自身的属性常用其在符号表中的入口指针来表示



关于单词的自身属性值

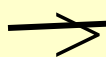
对于常数，其**单词自身的属性**常用其在常数表中的入口指针来表示

a = b + c * d

为例，假设按方案
分析后的结果为：

Token字

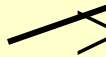
<1,



<24,

>

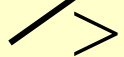
<1,



<20,

>

<1,



<22,

>

<1,



符号表

a 1

b 1

c 1

d 1

3.1.3 源程序的输入缓冲与预处理

■ 超前搜索和回退

- 双字符运算符 (`**`, `/*`, `:=`, ...)
- `D0 90 k=1, 10`
- `D0 90 k=1. 10`

■ 缓冲区

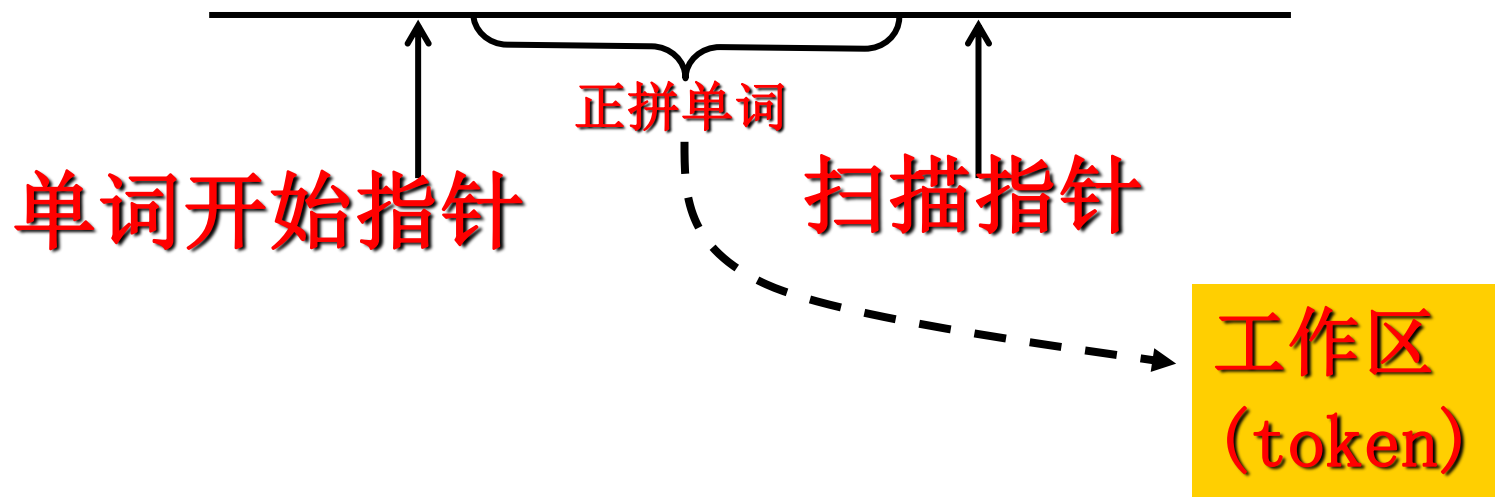
- 假定源程序存储在磁盘上，这样每读一个字符就需要访问一次磁盘，效率显然是很低的。

■ 空白字符的剔除

- 剔除源程序中的无用符号、空格、换行、注释等

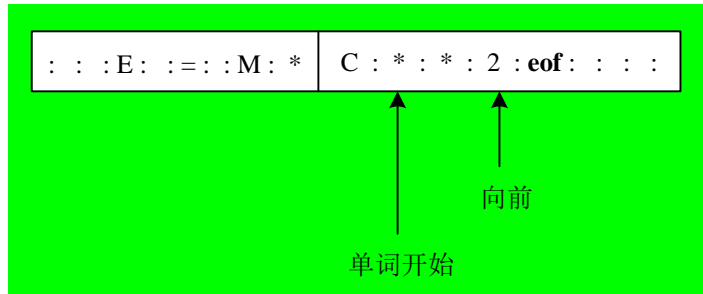
3.1.3 源程序的输入缓冲与预处理(续)

- 输入缓冲区

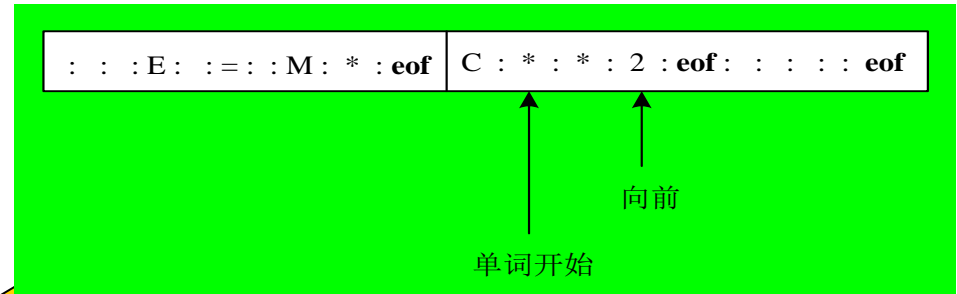


3.1.3 源程序的输入缓冲与预处理(续)

双缓冲区问题__超前扫描导致的效率问题



```
if forward在缓冲区第一部分末尾 then begin
  重装缓冲区第二部分;
  forward := forward + 1
end
else if forward在缓冲区第二部分末尾 then begin
  重装缓冲区第一部分;
  将forward移到缓冲区第一部分开始
end
else forward := forward + 1;
```



```
forward := forward + 1;
if forward ↑ = eof then begin
  if forward在第一部分末尾 then begin
    重装第二部分;
    forward := forward + 1
  end
  else if forward在第二部分末尾 then begin
    重装第一部分;
    将forward 移到第一部分开始
  end
  else /* eof 在表示输入结束 */
    终止词法分析
  end
```

大小问题

128Byte*2|1024Byte*2|4096Byte*2

■ 问题：如何设计和实现扫描器？



3.1.4 词法分析阶段的错误处理

1. 非法字符检查
2. 关键字拼写错误检查
3. 不封闭错误检查
4. 重复说明检查
5. 错误恢复与续编译

紧急方式恢复(panic-mode recovery)

反复删掉剩余输入最前面的字符，直到词法分析器能发现一个正确的单词为止。

3.1.5 词法分析器的位置

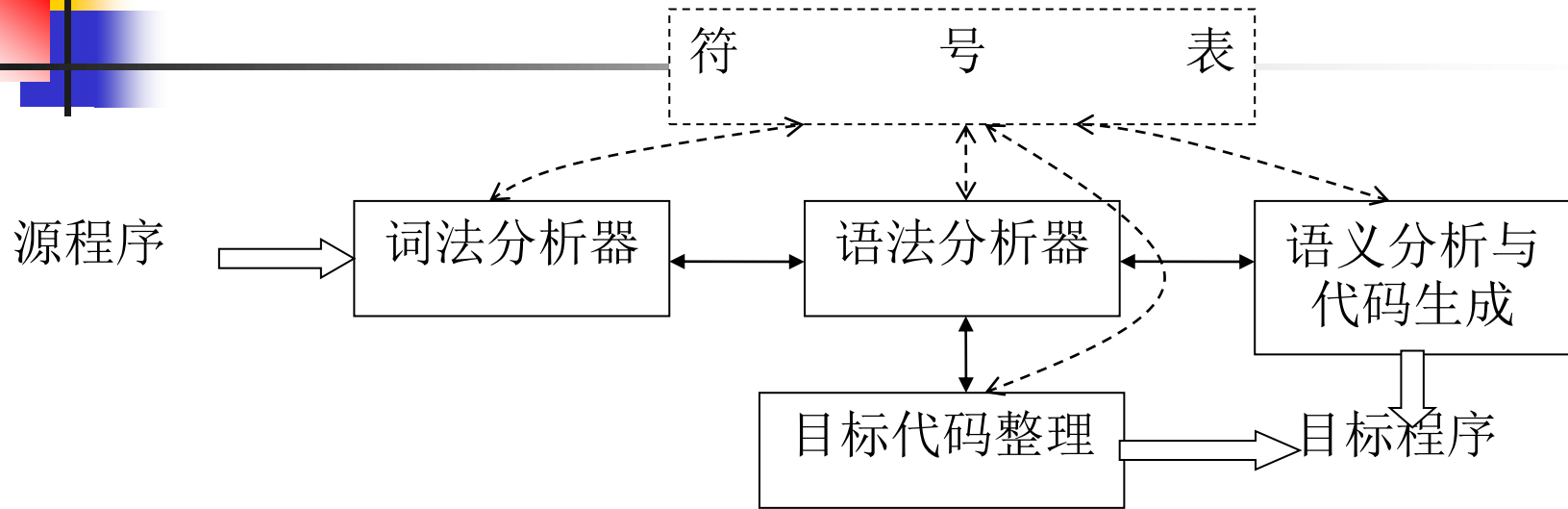


图3.4 以语法分析器为中心

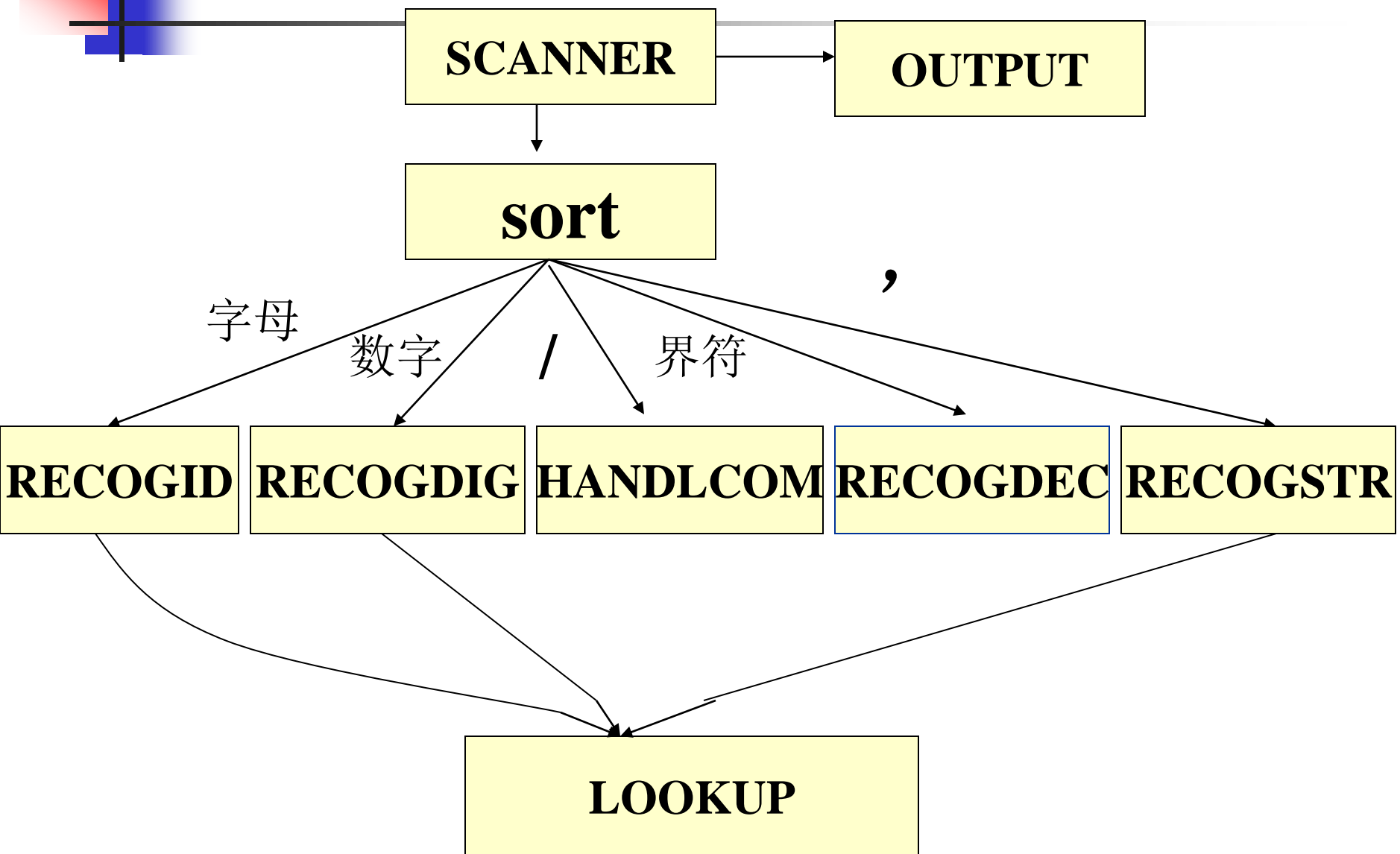
- 以语法分析器为中心的优点：
 - 简化编译器的设计。
 - 提高编译器的效率。
 - 增强编译器的可移植性。



3.2 词法分析阶段的设计与实现

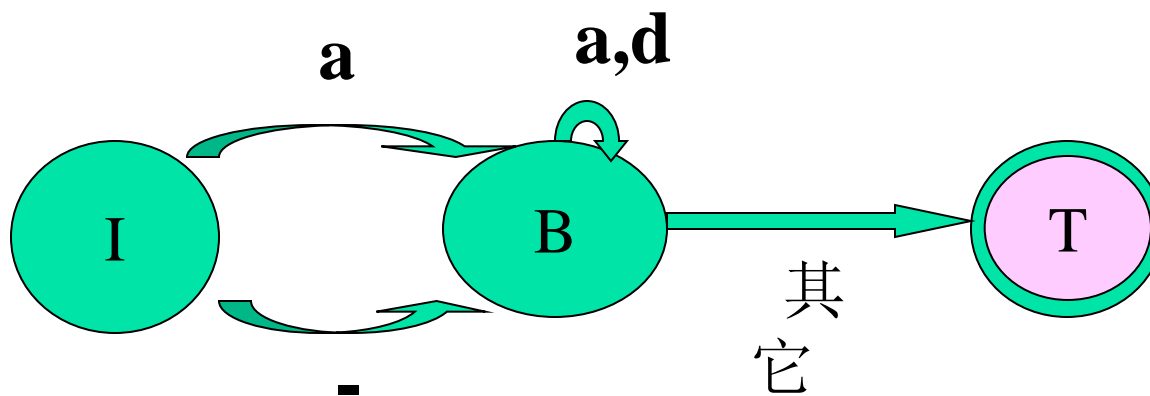
- 为了构造词法分析器，要研究构词法、每种词类的结构模式以及识别它的数学模型——有穷自动机。
- 一. 构造识别单词的DFA
- 二. 编写词法分析程序

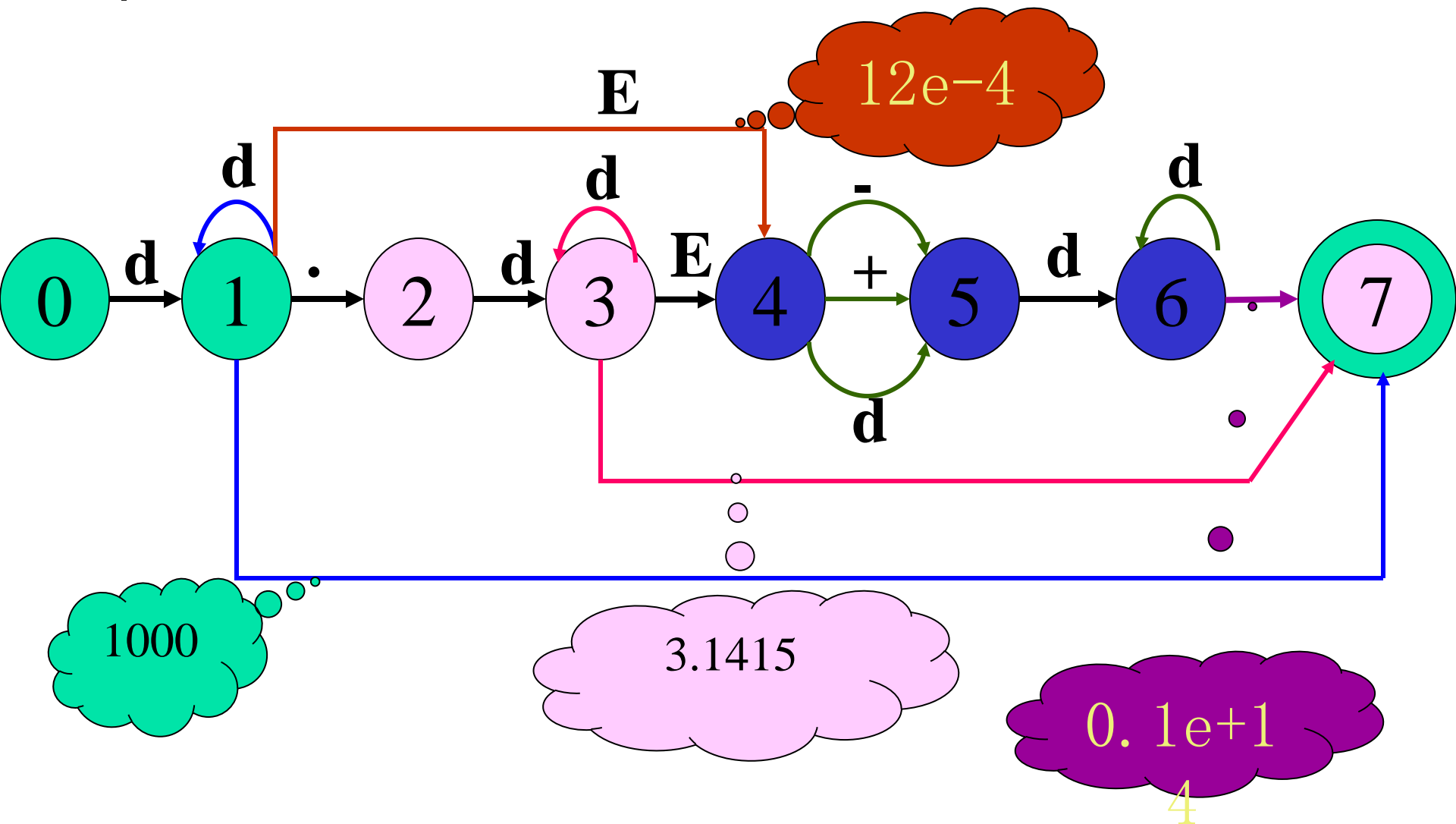
词法分析程序的设计框图



一. 识别单词的DFA

C语言的标识符







二.编写词法分析程序

- 根据画出的识别单词的状态转换图,构造词法分析程序,每个状态对应一段程序,完成到达此状态的工作;

program SCANNER;

Begin initiate符号表, 字符串表, 行, 列计数器;

Open 源文件, TOKEN文件

Repeat

 FIRSTCH(CH);

 if CH!=EOL then

 call SORT(CH)

 else RDLINE;

until CH=EOF;

把符号表, 字符串表做成文件;

close源文件, TOKEN文件;

call OUTPUTR;

模块0:

扫描器主控

单词分类模块 (SORT) 输入:

CH 内含单词首符;

```
procedure SORT (CH);
```

```
{   case CH of ‘字母’ :
```

```
        ‘字母’ :      call RECOGID (CH, TOKEN);
```

```
        ‘ / ’ :                               call
```

```
HANDLECOM (CH, TOKEN);
```

```
        ‘数字’ :      call RECOGDIG (CH, TOKEN);
```

```
        ‘ , ’ :                               call
```

```
RECOGSTR (CH, TOKEN);
```

```
        otherwise call RECOGDEL (CH, TOKEN);
```

```
    end case;
```

```
    write TOKEN into TOKEN文件;
```

```
procedure RECOGID(CH, TOKEN);
```

```
{  WORD:= '  ';
```

```
  WORD:=WORD||CH;
```

```
Repeat {
```

```
    call GETCH(CH);
```

```
    if CH是字母或数字 then
```

```
        WORD:=WORD||CH;    } until CH!=字母或数字;
```

```
if CH是非法字符 then
```

```
    call PRINTERR('非法字符' )
```

```
else 列计数-1;
```

```
if WORD 是关键字
```

```
    then TOKEN:=(关键字词类编码, _)
```

```
else {  call LOOPUP(WORD, '标识符', ENTRY)
```

```
        TOKEN:=(标识符字词类编码, ENTRY)};
```

```
Return    };
```

识别标识符;

输入: CH中含标识符的首字母;

输出: TOKEN(二元式形式);

```
procedure HANDLECOM(TOKEN);
```

```
{  call GETCH(CH);
```

处理注解 (HANDLECOM);

```
if CH!='*' then
```

输入: '/' ;进入该模块之前已扫描了一个字符
'/'

```
{  列计数-1;
```

```
  TOKEN=('/'的词类编码, _的TOKEN字或空TOKEN字;
```

```
  return  };
```

```
TOKEN='-1' ;
```

```
GETCH(CH);
```

```
while 列计数<=行长-1  do
```

```
{  CH1:=CH;
```

```
    call GETCH(CH);
```

```
    if CH1='*' and CH='/' then  
TOKEN:=' ' ; }
```

```
if TOKEN!=' ' then call PRINTERR('注解未完' );
```

```
TOKEN:=' ' ;    return  }
```

识别界限符 (RECOGDEL)

输入：CH内含单界限符；

输出：各种界符的TOKEN字；

```
procedure RECOGDEL (CH, TOKEN) ;
```

```
{ case CH of
```

```
    ‘+’:  TOKEN:=(‘+’的词类编码, _);
```

```
    ‘)’ :  TOKEN:=(‘)’的词类编码, _);
```

```
    ‘<’ :  { call GETCH (CH);
```

```
            if CH=‘=’ then TOKEN:=(‘<=’的词类编码, _)
```

```
            else if CH=‘>’ then TOKEN:=(‘<>’的词类编码, _)
```

```
            else {列计数-1; TOKEN:=(‘<’的词类编  
码, _)}
```

```
        }
```

```
    .....
```

```
endcase;
```

```
return }
```

3.3 词法分析程序的自动生成

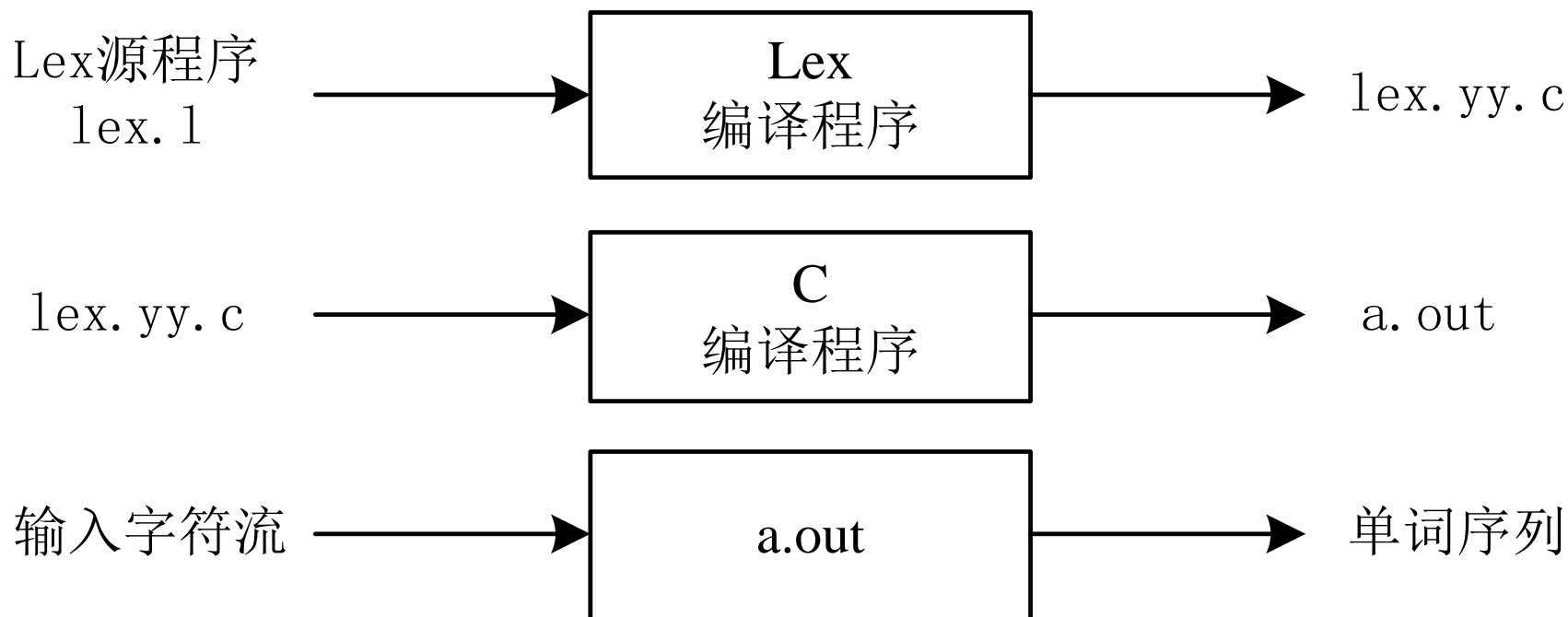


图3. 23 利用Lex建立词法分析程序的过程

3.3.1 Lex源程序

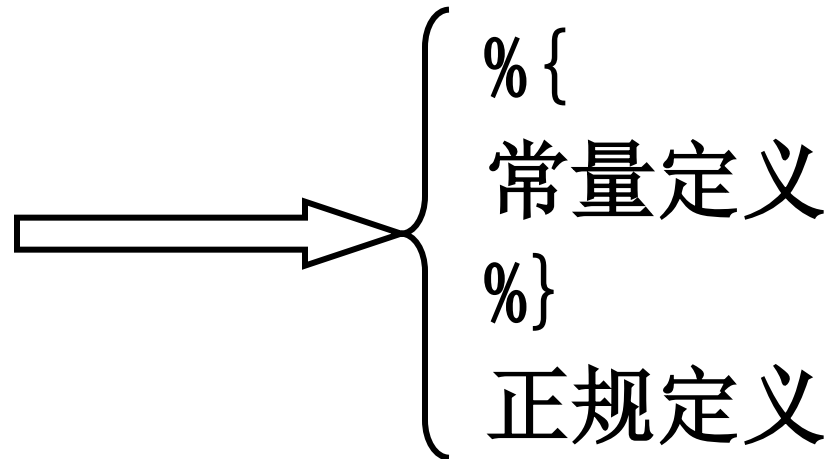
声明部分
(正规定义式)

%%

识别规则部分
(识别规则)

%%

辅助过程部分





3.3.1 Lex源程序

1、正规定义式

letter \rightarrow **A|B|C|...|Z|a|b|c|...|z**

digit \rightarrow **0|1|2|...|9**

identifier \rightarrow **letter(letter|digit)***

integer \rightarrow **digit(digit)***

2、识别规则

正规式	动作描述
-----	------

token₁	{action₁}
--------------------------	-----------------------------

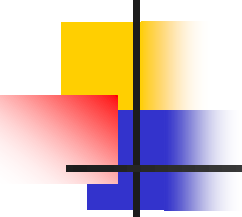
token₂	{action₂}
--------------------------	-----------------------------

.....

token_n	{action_n}
--------------------------	-----------------------------



• %{		• delim	[\t\n]
• #include <stdio.h>		• ws	[delim]+
• #include "y.tab.h"		• letter	[a-zA-Z]
• #define ID	1	• digit	[0-9]
• #define INT	2	• id	{letter}({letter} {digit})*
• #define EXP	3	• number	{digit}+
• #define MULTI	4	• %%	
• #define COLON	5	• {ws}	;
• #define EQ	6	• begin	return(BEGIN);
• #define NE	7	• end	return(END);
• #define LE	8	• if	return(IF);
• #define GE	9	• then	return(THEN);
• #define LT	10	• else	return(ELSE);
• #define GT	11	• do	return(DO);
• #define PLUS	12	• program	return(PROGRAM);
• #define MINUS	13	• {id}	{yyval = install_id(); return(ID);}
• #define RDIV	14	• {number}	{yyval = install_num();
• #define COMMA	15		return(INT);}
• #define SEMIC	16		
• #define RELOP	17		
• #define ASSGIN	18		
• int line_no = 1; %}	2013/3/6 Wednesday		



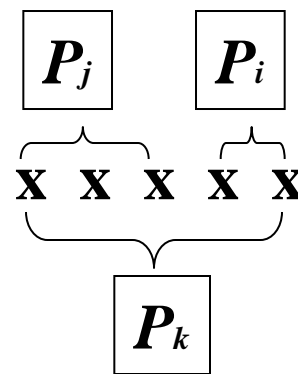
■	"<"	{yyval =LT; return(RELOP);}
■	"<="	{yyval =LE; return(RELOP);}
■	"="	{yyval =EQ; return(RELOP);}
■	">"	{yyval =GT; return(RELOP);}
■	">="	{yyval =GE; return(RELOP);}
■	"<>"	{yyval =NE; return(RELOP);}
■	"+"	return(PLUS);
■	"-"	return(MINUS);
■	"*"	return(MULTI);
■	"/"	return(RDIV);
■	"**"	return(EXP);
■	":"	return(COLON);
■	":="	return(ASSGIN);
■	","	return(COMMA);
■	";"	return(SEMIC);
■	\n	line_no++;
■	.	{ fprintf (stderr, "'%c' (0%o): illegal charcter at
	line	%d\n", yytext[0], yytext[0], line_no); }
■	%%	
■	install_id()	
■	{.....}	
■	install_num()	
■	{.....}	

如:begin:=

LEX二义性问题的两条原则

1.最长匹配原则

在识别单词过程中，有一字符串
根据最长匹配原则，应识别为这是
一个符合 P_k 规则的单词，
而不是 P_j 和 P_i 规则的单词。



2.优先匹配原则

如果有一字符串有两条规则可以同时匹配时，那么用规则
序列中位于前面的规则相匹配，所以排列在最前面的规则优先
权最高。

3.3.2 Lex的实现原理

Lex的功能是根据Lex源程序构造一个词法分析程序，该词法分析器实质上是一个有穷自动机。

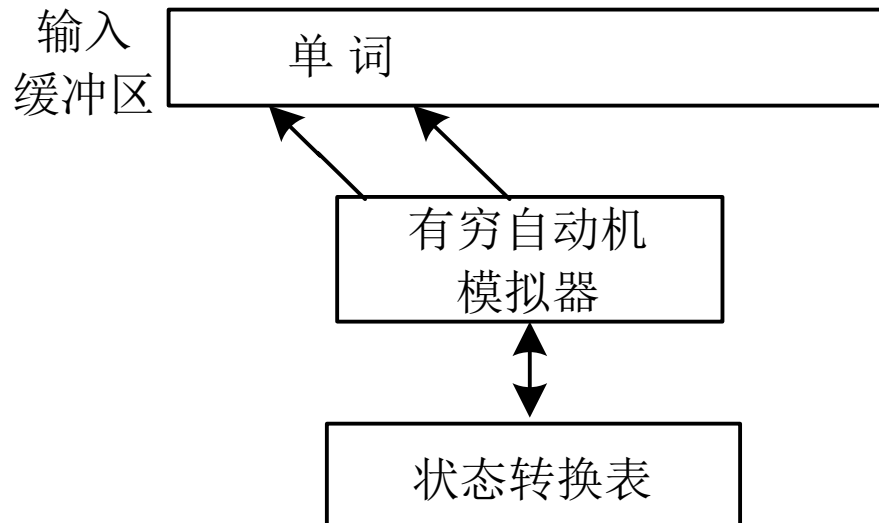


图 3.24 Lex生成的词法分析器结构

Lex的功能是根据Lex源程序生成状态转换矩阵和控制程序



三点说明

- 1) 以上是Lex的构造原理，虽然是原理性的，但据此就不难将Lex构造出来。
- 2) 所构造出来的Lex是一个通用的工具，用它它可以生成各种语言的词法分析程序，只需要根据不同的语言书写不同的LEX源文件就可以了。
- 3) Lex不但能自动生成词法分析器，而且也可以产生多种模式识别器及文本编辑程序等



本章小结

- 词法分析器接收表示源程序的“平滑字符流”，输出与之等价的单词序列；
- 单词被分成多个种类，并被表示成(种别，属性值)的二元组形式；
- 为了提高效率，词法分析器使用缓冲技术，而且在将字符流读入缓冲区时，是经过剔除注解、无用空白符等预处理后的结果；



本章小结

- 单词的识别相当于正则语言的识别；
- 词法的等价描述形式有正则文法、有穷状态自动机、正则表达式，其中有穷状态自动机可以用状态转换图表示；
- 实现词法分析器时状态转换图是一个很好的设计工具，根据该图，容易构造出相应的分析程序；
- 使用恰当的形式化描述，可以实现词法分析器的自动生成，Lex就是一种自动生成工具。