编译原理

北方工业大学信息学院
School of Information Science and Technology,
North China University of Technology
束劼
shujie@ncut.edu.cn
瀚学楼1122，88801615

第三章 词法分析

第三章 词法分析

- 本章目录

第三章 词法分析

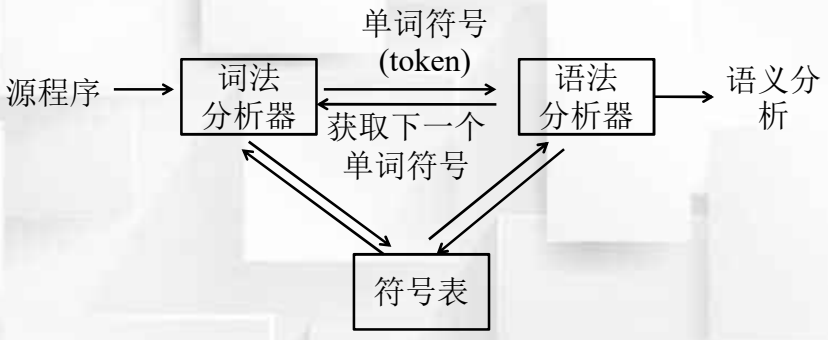
- 大纲要求

1. 掌握：词法分析器的设计与实现方法，基于状态转换图的词法分析器的构造算法。
2. 理解：状态转化图的作用与画法。
3. 了解：对于词法分析器的要求；正规文法与有限自动机的等价性，正规式与有限自动机的等价性；词法分析器的自动产生工具LEX的基本作用。

第三章 前言

词法分析器与语法分析器的交互

- 词法分析器与语法分析器之间的交互



3.1 对于词法分析器的要求

3.1 对于词法分析器的要求

- 3.1.1 词法分析器的功能和输出形式
- 3.1.2 词法分析器作为一个独立子程序

3.1.1 词法分析器的功能和输出形式

3.1.1 词法分析器的功能和输出形式

- 3.1.1 词法分析器的**功能和输出形式**
  - 词法分析器的功能是接收输入源程序，输出单词符号。
  - **单词符号(token)**分为五种：**关键字(keyword)**；**标识符(identifier)**；**常数(constants)**；**运算符(operators)**；**界符(punctuation)**。
  - 词法分析器所输出的单词符号常常表示成如下的二元式：

    〈单词种别，单词符号的属性值〉

3.1.1 词法分析器的功能和输出形式

- 词法分析器的**功能**
  - ① 识别单词。
  - ② 去掉注释和空白(空白<blank>、换行<newline>、标签<tab>，以及其他用于间隔输入标记的字符)。
  - ③ 关联编译程序产生的错误信息与对应的源程序。
    比如，词法分析器会跟踪换行符，如果有多个换行符连续出现，词法分析器会在每个行号后面跟上一个错误信息。

3.1.1 词法分析器的功能和输出形式

- 词法分析器的**两个过程**
  - ① 扫描过程
    只是扫描，不对输入做标记(tokenization)，例如删除注释以及把多个连续的空白符变为1个。
  - ② 词法分析过程
    比扫描更复杂的处理过程，需要对扫描后的内容生成标记。

3.1.1 词法分析器的功能和输出形式

- 词法分析器扫描**单词种别**
  - 本书假定关键字、运算符和界符都是一符一种，标识符单列一种，常数按类型分种。
- 单词符号的**属性信息**(Attributes for Tokens)
  - 属性信息(值)是指单词符号的特性或特征值。本书仅给出标识符、常量的属性信息，即存放它们的符号表表项的指针。

13

3.1.1 词法分析器的功能和输出形式

C程序例子

While (i>=j) i--; 经词法分析器处理后的结果为：

① <**While**,->
② < **(**,->
③ <id,指向**i**的符号表项的指针>
④ < **>=**,->
⑤ <id,指向**j**的符号表项的指针>
⑥ < **)**,->
⑦ < id,指向**i**的符号表项的指针>
⑧ <**--** , ->
⑨ <**;** ,->

14

3.1.1 词法分析器的功能和输出形式

- 单词种别通常用整数编码表示。
  - 若一个种别只有一个单词符号，则种别编码就代表该单词符号。假定关键字、运算符和界符都是一符一种。
  - 若一个种别有多个单词符号，则对于每个单词符号，给出种别编码和自身的属性值。

15

3.1.1 词法分析器的功能和输出形式

FORTRAN 程序例子

IF (5.EQ.M) GOTO 100

输出单词符号：

①逻辑**IF** 〈34，-〉
②左括号**(** 〈2，-〉
③整常数**5** 〈20，‘**5**’的二进制〉
④等号**=** 〈6，-〉
⑤标识符**M** 〈26，‘**M**’〉
⑥右括号**)** 〈16，-〉
⑦**GOTO** 〈30，-〉
⑧标号**100** 〈19，‘**100**’的二进制〉

16

3.1.1 词法分析器的功能和输出形式

- 词法分析器的功能

① 按照语言的定义规则，逐个地读入源程序的符号，识别出对语言有意义的符号串，即单词符号；

② 分析单词符号的属性，并把单词符号及其属性填写在符号表中；

③ 同时把源程序改造成等价的计算机内部表示单词符号，以便编译的后续阶段使用。

④ 还要对源程序进行预处理工作，包括：删除源程序中的空格、制表符、换行、注释等不影响程序语法、语义的结构。

⑤ 对数字常数完成数字字符串到二进制数值的转换。

3.1.2 词法分析器作为一个独立的子程序

3.1.2 词法分析器作为一个独立子程序

- 3.1.2 词法分析器作为一个独立子程序

  是否应当将词法分析器作为独立的一遍呢？

  - 如果作为一遍，把源程序翻译成一连串的单词符号，并保留在内存中，当语法分析开始时，再一一读入；

    是否有必要如此？

  - 更恰当的是，将其处理为一个子程序。

  作为独立阶段的优点：结构简洁、清晰和条理化，有利于集中考虑词法分析一些枝节问题。

3.2 词法分析器的设计

3.2 词法分析器的设计

3.2.1 输入、预处理

3.2.2 超前搜索

3.2.3 状态转换图

3.2.4 状态转换图的实现

3.2.1 输入、预处理

3.2.1 输入、预处理

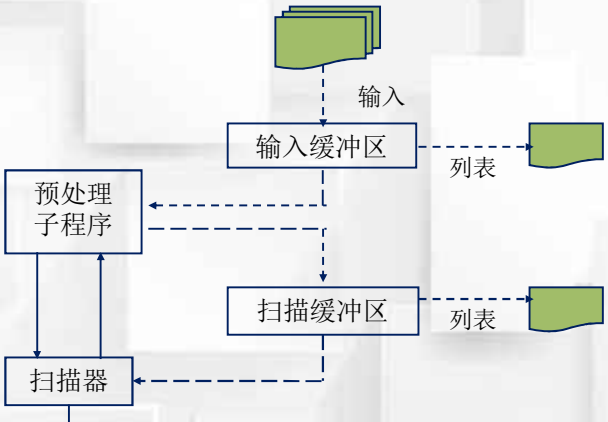


图3.2 词法分析器

3.2.1 输入、预处理

- 输入
  - 源程序。
- 输入缓冲区
  - 存放输入串，词法分析的工作可以直接在这个缓冲区进行。

假如遇到空格、回车、换行等符号，怎么办？

3.2.1 输入、预处理

- 预处理子程序
  - 对输入串进行预处理，其主要工作是去掉注释行，合并空白符等。
- 扫描缓冲区
  - 存放预处理好的符号串。
- 分析器
  - 不断地从扫描缓冲区读入字符串，并进行词法分析工作。

3.2.1 输入、预处理

- 分析器
  - 不断地从扫描缓冲区读入字符串，并进行识别。

如何扫描？逐个字符扫描？还是多个字符扫描？

  - 单个字符：-，=，<等等
  - 双字符：>=，==，<=等等

如何识别双字符的符号？

3.2.1 输入、预处理

- 分析器

  实际问题：不论缓冲区多大都不能保证单词不被它的边界打断，那如何设置缓冲区？

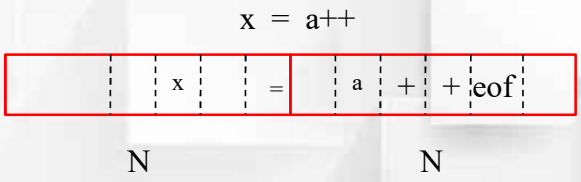
  解决方案：把缓冲区分为相同的两个半区，每半区可容N个字符。

  分为**基本缓冲区**和**补充缓冲区**，如果基本缓冲区不够，则要求输入串一定在补充缓冲区内结束，所以高级语言的符号串长度有限制；

3.2.1 输入、预处理

- 分析器
  - 双缓冲器(two-buffer scheme)

    两个长度为N的缓冲器，N通常为120个字符，**两个缓冲器交替使用**。

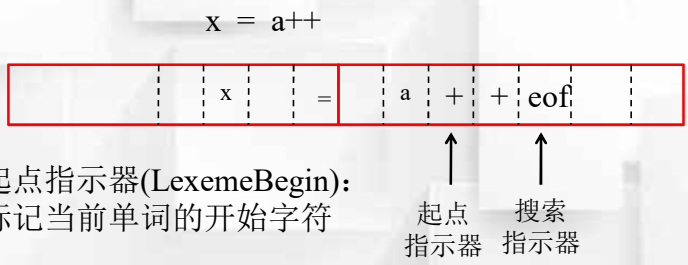


如果少于N个字符在缓冲器中，用特殊字符eof表示末尾

3.2.2 单词符号的识别：超前搜索

3.2.2 单词符号的识别：超前搜索

- 超前搜索（搜索指示器）Pointer Forward



起点指示器(LexemeBegin)：
标记当前单词的开始字符

搜索指示器(Pointer forward)：往前扫描，直到确认某个单词符号(token)，并把指示器指到单词的右端

3.2.2 单词符号的识别:超前搜索

- 超前搜索（搜索指示器）Pointer Forward
  - 由于符号串需要结合后面的符号明确语义，所以需要向前读取多个符号后，判断其含义，这种向前读取符号的机制称为超前搜索。
- 超前搜索应用：
  - 关键字识别
  - 标识符的识别
  - 常数的识别
  - 运算符和界符识别

3.2.2 单词符号的识别:超前搜索

- 超前搜索（搜索指示器）Pointer Forward
  - 关键字识别

    FORTRAN中的关键字不加特殊保护，关键字和用户自定义的标识符或标号之间没有特殊的界符作间隔。

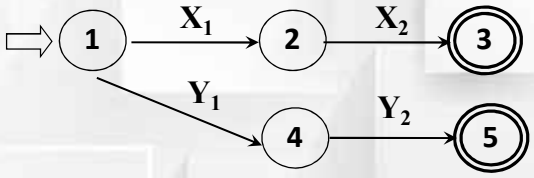
```
DO99k = 1,10      /*循环语句，DO是关键字
DO99k = 1.10   /*赋值语句，DO99k是标识符
```

看到DO不一定是循环Do语句，向前搜索找到“=”，再向前搜索找到界符“，”或“.”才能确定是赋值，还是Do循环。

3.2.3 状态转换图

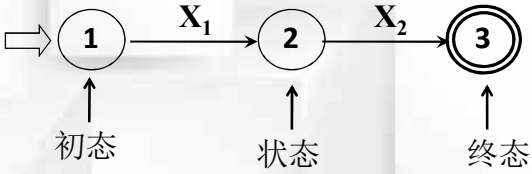
3.2.3 状态转换图

- 状态转换图 Trasition Diagrams
  - 转换图是一个有向图。在状态转换图中，结点代表状态，用圆圈表示。状态之间用箭弧连结。箭弧上的标记（字符）代表在射出结点（即箭弧开始结点）状态下可能出现的输入字符或字符类。



3.2.3 状态转换图

- 状态转换图



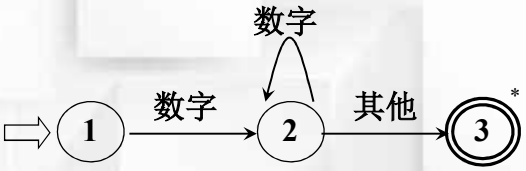
结点：状态用○表示；终态用◎表示。

一张转换图只包含有限个状态，其中有一个为初态，至少要有一个终态。

35

3.2.3 状态转换图

- 状态转换图



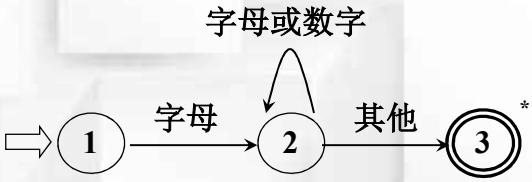
识别整常数的状态转换图

一个状态转换图可用于识别(或接受)一定的字符串

36

3.2.3 状态转换图

- 状态转换图



识别标识符的状态转换图

3.2.3 状态转换图

- 状态转换图

一个综合例子：构造一个识别某个简单语言的所有单词符号的转换图。在符号表中直接用整数编码表示不便于记忆，因此用特殊符号表示，即**助忆符**。这些符号都以**$**开始。这些符号允许自定义。

例如：

IF	$IF
DO	$DO
END	$END
=	$ASSIGN
+	$PLUS

3.2.3 状态转换图

- 符号转换图　　DIM是定义变量 例如：Dim b as integer

单词符号	种别编码	助忆符	内码值
DIM	1	$DIM	-
IF	2	$IF	-
DO	3	$DO	-
STOP	4	$STOP	-
END	5	$END	-
标识符	6	$ID	内部字符串
常数（整）	7	$INT	标准二进制形式
=	8	$ASSIGN	
+	9	$PLUS	-
*	10	$STAR	-
* *	11	$POWER	-
,	12	$COMMA	-
（	13	$LPAR	-
）	14	$RPAR	-

39

3.2.3 状态转换图

- 给综合例子设定一些限制
  - ① Keyword 关键字（如IF、WHILE等）都是“保留字”。所谓保留字的意思是，用户不得使用它们作为自己定义的标识符。

    保留字识别有两种方法
  - ② 由于把关键字作为保留字，故可以把关键字作为一类特殊标识符来处理。也就是说，对于关键字不专设对应的转换图。但把它们(及其种别编码)预先安排在一张表格中(此表叫做保留字表)。当转换图识别出一个标识符时，就去查这张表，确定它是否为一个关键字。

40

3.2.3 状态转换图

- 给综合例子设定一些限制

  ③ 关键字、标识符和常数之间没有确定的运算符或界符作间隔，则必须至少用一个空白符作间隔(此时，空白符不再是完全没有意义的)。

  例如，一个条件语句应写为

  IF i>0 i=1;

  - 而绝对不要写成

    IFi>0 i=1;

  因为对于后者，我们的分析无条件地将IFi看成一个标识符。

3.2.3 状态转换图

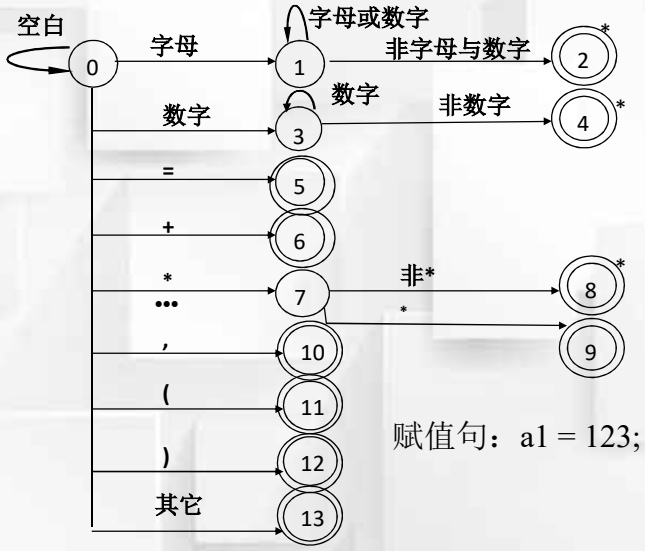
- 状态转换图

  C语言的界符：

  大括符 {和}
  尖括符 <和>
  圆括符 (和)
  方括符 [和]
  注释符 /*和*/
  双引号 “
  单引号 '

3.2.3 状态转换图

对简单语言进行词法分析的状态转换图



赋值句：a1 = 123;

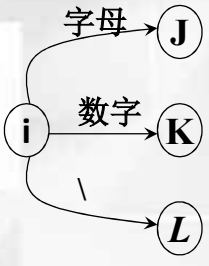
3.2.4 状态转换图的实现

3.2.4 状态转换图的实现

- 状态转换图的实现主要思想
  - 让每个状态结点对应一小段程序。
  - 对不含回路的分支结点，可以对应一个switch或一组if语句。
  - 对含回路的状态结点，可以对应一个while语句或if语句。
  - 终态结点对应一个return(code,value)语句。

3.2.4 状态转换图的实现

- 状态转换图的实现举例
  - 对不含回路的分叉结，可用一个Switch 语句或一组If-Then-Else语句实现



引进一组全局变量和过程如下：

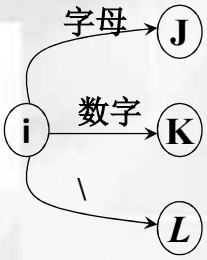
ch 是字符变量，存放最新读进的源程序字符

GetChar 子程序过程，将下一输入字符读到**ch**中，搜索指示器前移一字符位置

IsLetter( ) or IsDigit( ) 布尔函数过程，分别判断**ch**中的字符是否为字母或数字

3.2.4 状态转换图的实现

- 状态转换图的实现举例
  - 对不含回路的分叉结，可用一个Switch 语句或一组If-Then-Else语句实现

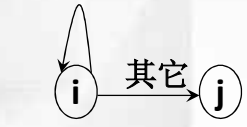



```
GetChar( );
if (IsLetter( )) {…状态J的对应程序段…;}
else if (IsDigit( )) {…状态K的对应程序段…;}
else if (ch='\') {…状态L的对应程序段…;}
else {…错误处理…;}
```

47

3.2.4 状态转换图的实现

- 状态转换图的实现举例
  - 对含回路的状态结，可对应一段由While结构或If语句构成的程序。



退出While循环

```
GetChar( );
While (IsLetter( ) or IsDigit( ))
    GetChar( );
…状态j的对应程序段…
```

48

第三章 小结

第三章 小结

3.1 对于词法分析器的要求

3.2 词法分析器的设计

Coursework

3.1 给定右线性文法G：

S→0S｜1S｜1A｜0B

A→1C｜1

B→0C｜0

C→0C｜1C｜0｜1

请给出文法G对应的状态转换图。