
词法分析

一、实验内容与要求

(1)实验目的

根据 PL/0 语言的文法规范，编写 PL/0 语言的词法分析程序。

通过设计调试词法分析程序，实现从源程序中分出各种单词的方法；加深对课堂教学的理解；提高词法分析方法的实践能力。

掌握从源程序文件中读取有效字符的方法和产生源程序的内部表示文件的方法。

掌握词法分析的实现方法。

上机调试编出的词法分析程序。

(2)实验内容

已给 PL/0 语言文法，输出单词符号（关键字、专用符号以及其它标记）。

PL/0 语言功能简单、结构清晰、可读性强，而又具备了一般高级程序设计语言的必须部分，因而 PL/0 语言的编译程序能充分体现一个高级语言编译程序实现的基本方法和技术。

1. PL/0 语言文法的 EBNF

<程序>::=<分程序>.

<分程序> ::= [<常量说明>][<变量说明>][<过程说明>]<语句>

<常量说明> ::= CONST<常量定义>{, <常量定义>;}

<常量定义> ::= <标识符>=<无符号整数>

<无符号整数> ::= <数字>{<数字>}

<变量说明> ::= VAR <标识符>{, <标识符>;}

<标识符> ::= <字母>{<字母>|<数字>}

<过程说明> ::= <过程首部><分程序>{; <过程说明> };

<过程首部> ::= PROCEDURE <标识符>;

<语句> ::= <赋值语句>|<条件语句>|<当循环语句>|<过程调用语句>

 |<复合语句>|<读语句>|<写语句>|<空>

<赋值语句> ::= <标识符>:=<表达式>

<复合语句> ::= BEGIN <语句> {;<语句> }END
 <条件语句> ::= <表达式> <关系运算符> <表达式> | ODD<表达式>
 <表达式> ::= [+|-]<项> {<加法运算符> <项>}
 <项> ::= <因子> {<乘法运算符> <因子>}
 <因子> ::= <标识符> | <无符号整数> | '('<表达式>')'
 <加法运算符> ::= +|-
 <乘法运算符> ::= */
 <关系运算符> ::= =|<|<=|>|=
 <条件语句> ::= IF <条件> THEN <语句>
 <过程调用语句> ::= CALL 标识符
 <当循环语句> ::= WHILE <条件> DO <语句>
 <读语句> ::= READ('<标识符>{,<标识符>}')
 <写语句> ::= WRITE('<表达式>{,<表达式>}')
 <字母> ::= a|b|...|X|Y|Z
 <数字> ::= 0|1|...|8|9

2. PL/0 语言的词汇表

序号	类别	单词	编码
1	基本字	begin, call, const, do, end if, odd, procedure, read then, var, while, write	beginsym, callsym, constsym dosym, endsym, ifsym, oddsym proceduresym, readsym, thensym varsym, whilesym, writesym
2	标识符		ident
3	常数		number
4	运算符	+, -, *, /, odd =, <>, <, <=, >, >=, :=	plus, minus, times, slash, oddsym eql, neq, lss, leq, gtr, geq, becomes

5	界符	() , ; .	Lparen, rparen, comma, semicolon period
---	----	-----------	---

(3)实验要求

确定编译中使用的表格、标识符与关键字的区分方法等。

把词法分析器设计成一个独立一遍的过程。

词法分析器的输出形式采用二元式序列，即：

(单词种类, 单词的值)

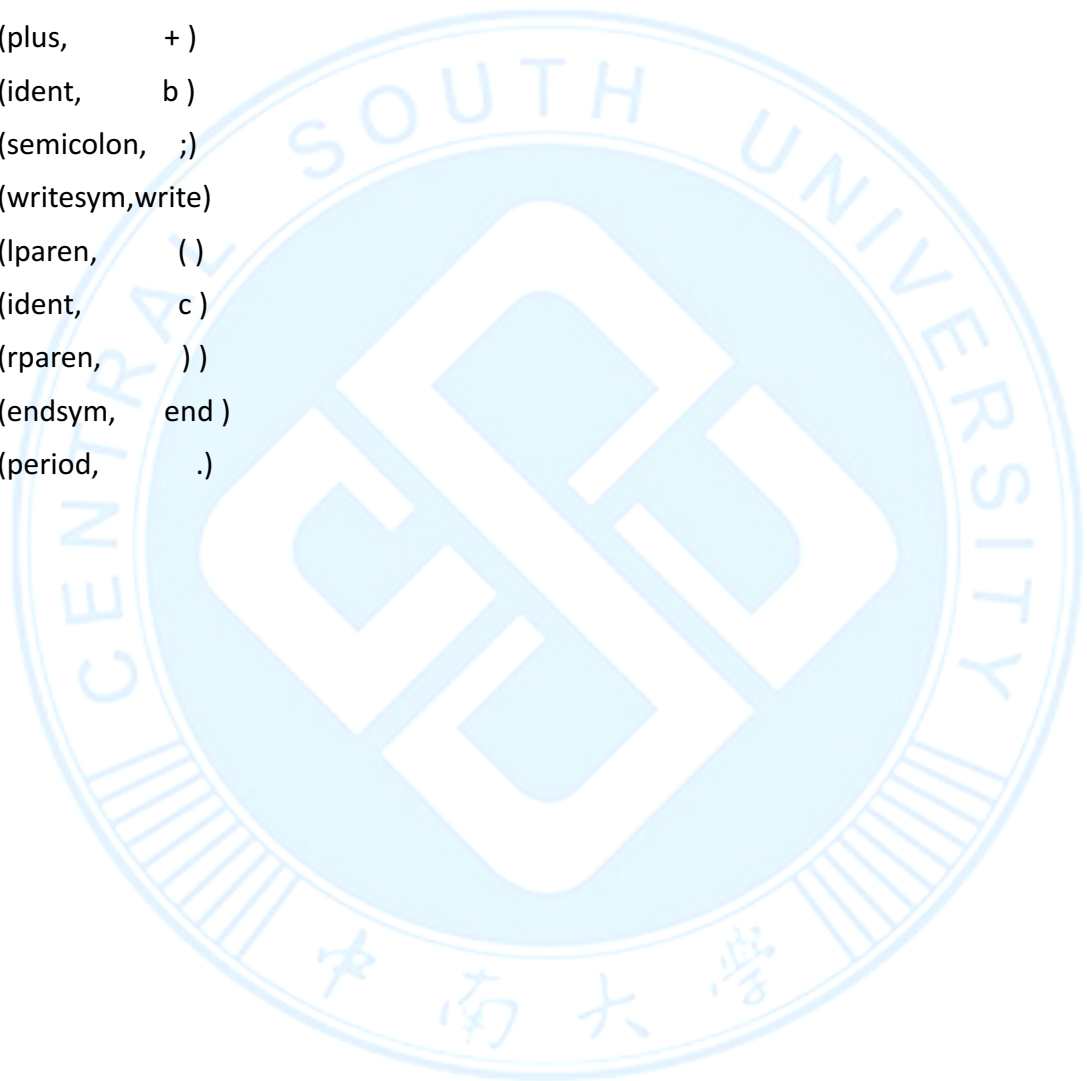
输入：PL/O 源程序。例：

```
const a=10;
var b,c;
begin
read(b);
c:=a+b;
write(c)
end.
```

输出：

```
(constsym,const)
(ident , a)
(eql , =)
(number, 10)
(semicolon, ;)
(varsym, var )
(ident, b)
(comma, ,)
(ident, c)
(semicolon, ;)
(beginsym,begin)
(readsym, read )
```

(lparen, ()
(ident, b)
(rparen,))
(semicolon, ;)
(ident, c)
(becomes, :=)
(ident, a)
(plus, +)
(ident, b)
(semicolon, ;)
(writesym,write)
(lparen, ()
(ident, c)
(rparen,))
(endsym, end)
(period, .)



二、实验分析与设计

编译简单的说，就是把源程序转化为另一种形式的程序,而其中关键的部分就是理解源程序所要表达的意思，才能转化为另一种源程序。

编译器也一样，它的输入是语言的源文件（一般可以是文本文件）对于输入的文件，首先要分离出这个输入文件的每个元素（关键字、变量、符号、、）

然后根据语言的文法，分析这些元素的组合是否合法，以及这些组合所表达的意思。

词法分析，也就是把输入的符号串整理成特定的词素。

词法分析器的功能输入源程序，按照构词规则分解成一系列单词符号。单词是语言中具有独立意义的最小单位，包括关键字、标识符、运算符、界符和常量等

(1) 关键字 是由程序语言定义的具有固定意义的标识符。例如，Pascal 中的 begin, end, if, while 都是保留字。这些字通常不用作一般标识符。

(2) 标识符 用来表示各种名字，如变量名，数组名，过程名等等。

(3) 常数 常数的类型一般有整型、实型、布尔型、文字型等。

(4) 运算符 如+、-、*、/等等。

(5) 界符 如逗号、分号、括号、等等。

输出：

词法分析器所输出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)

单词种别通常用整数编码。标识符一般统归为一种。常数则宜按类型（整、实、布尔等）分种。关键字可将其全体视为一种。运算符可采用一符一种的方法。界符一般用一符一种的方法。对于每个单词符号，除了给出了种别编码之外，还应给出有关单词符号的属性信息。单词符号的属性是指单词符号的特性或特征。

示例：

while(i>=j) i--

经词法分析器处理后，它将被转为如下的单词符号序列：

<while, _>

<(, _>

<id, 指向 i 的符号表项的指针>

<>=, _>

<id, 指向 j 的符号表项的指针>

<), _>

<id, 指向 i 的符号表项的指针>

<--, _>

<;, _>

本次实验把词法分析分析器作为一个独立子程序

词法分析是编译过程中的一个阶段，在语法分析前进行。词法分析作为一遍，可以简化设计，改进编译效率，增加编译系统的可移植性。也可以和语法分析结合在一起作为一遍，由语法分析程序调用词法分析程序来获得当前单词供语法分析使用。

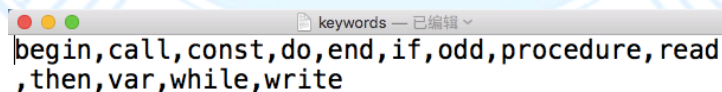
本次实验采用的词法规则比较简单，大体来说可以分为以下几类：

- (1) 保留字
- (2) 标识符
- (3) 整数
- (4) 关系运算符、运算符

```
//标识符表
HashMap<Integer, String> identTable = new HashMap<Integer, String>();
//常量表
HashMap<Integer, String> numberTable = new HashMap<Integer, String>();
//保留字表
HashMap<String, String> keyWords = new HashMap<String, String>();
```

首先应该给保留字单独创建一个表，表中对应保留字的类别名称和名字：

使用 Java 语言中的 HashMap 保留这个表，首先从文件中利用正则表达式将关键字提取出来：



```
begin,call,const,do,end,if,odd,procedure,read
,then,var,while,write
```

然后，把文件中的关键字以及对应的类别放入 Map 中：

```
void createMap() {  
    try {  
        FileReader fileReader = new FileReader("keywords.txt");  
        BufferedReader bufferedReader = new BufferedReader(fileReader);  
        String flush = bufferedReader.readLine();  
        String[] arrys = flush.split(",");  
        for (int i = 0; i < arrys.length; i++)  
            keyWords.put(arrys[i], arrys[i] + "sym");  
    } catch (Exception e) {  
        System.err.println("keywords.txt Not Found");  
        e.printStackTrace();  
    }  
}
```

然后设置几个全局变量，用于标识缓冲区里的字符串（等待归类）和正在读入的字符：

```
// 当前读入的字符  
char ch;  
// 当前指针位置  
int index = 0;  
// 存放单词符号  
String strToken = "";
```

然后构造词法分析的自动机：

```

void onFA(String str)
{
    getBC(str);

    if(index>=str.length())
    { //避免读入的序列最后是空格而导致异常
        return;
    }

    PrintStream ps = null;
    try {
        ps = new PrintStream(new FileOutputStream("Lex.txt",true));
    } catch (FileNotFoundException e) {
        System.err.println("Lex.txt Not Found");
        e.printStackTrace();
    }

    if(isLetter())
    { chargeIdent(str);
    } else if (isDigit()) {
        chargeNumber(str);
    } else if (ch=='+') {
        System.out.println("( "+"plus"+" , "+ch+" )");
        ps.println("plus"+" , "+ch);
    } else if (ch=='-') {
        System.out.println("( "+"minus"+" , "+ch+" )");
        ps.println("minus"+" , "+ch);
    } else if (ch=='*') {

```

依次按顺序判断类型。

读取字符有两种方式，一种是一个一个字符地读入，剩下一种是忽略空格的读入，主要适用于开始词法分析自动机时，应该忽略空格，但一旦开始一个单词的识别，空格就被当作分隔符而不能忽略。

```

// 从输入串中获得字符
void getChar(String str) {
    ch = str.charAt(index);
    index++;
}
// 自动屏蔽空格
// 从index开始，返回搜索的下一个index
void getBC(String str) {
    for (int i = index; i < str.length(); i++) {
        if (str.charAt(i) != ' ') {
            ch = str.charAt(i);
            index = i+1;
            return;
        }
    }
    index = str.length();
}

```


首先是判断是否是标识符的自动机：

```
int chargeIdent(String str) {
    PrintStream ps = null;
    try {
        ps = new PrintStream(new FileOutputStream("Lex.txt",true));
    } catch (FileNotFoundException e) {
        System.err.println("Lex.txt Not Found");
        e.printStackTrace();
    }
    while (isDigit() || isLetter()) {
        strToken+=ch;// 将当前字符放入缓冲区中
        getChar(str);
    }
    index--;//指针向前一位（多读了一个）
    String result = Reserve();
    if(result.equals("notFound"))
    {
        int key = identTable.size();
        identTable.put(key+1, strToken);//在常数表中添加
        System.out.println(" (" + "ident" + " , " + strToken + " )");
        ps.println("ident" + " , " + strToken);
    }
    else {
        System.out.println(" (" + result + " , " + strToken + " )");
        ps.println(result + " , " + strToken);
    }
}

strToken="";// 清空缓冲区
ps.close();
return index;
```

标识符的特征是以字母开头，中间部分是数字或字母
在识别完标识符后，需要判断是否是关键字：

```
// 从保留字表中查询缓冲区内的单词是否为保留字
String Reserve() {
    String string = strToken;
    if (keyWords.containsKey(string))
        return keyWords.get(string);
    else
        return "notFound";
}
```

类似的判断是否为数字的自动机：

```

int chargeNumber(String str) {
    PrintStream ps = null;
    try {
        ps = new PrintStream(new FileOutputStream("Lex.txt",true));
    } catch (FileNotFoundException e) {
        System.err.println("Lex.txt Not Found");
        e.printStackTrace();
    }
    while (isDigit()) {
        strToken+=ch; // 将当前字符放入缓冲区中
        getChar(str);
    }
    index--;
    int key = numberTable.size();
    numberTable.put(key+1, strToken); // 在常数表中添加
    System.out.println(" ( "+number+" , "+strToken+" )");
    ps.println("number"+", "+strToken);
    strToken=""; // 清空缓冲区
    ps.close();
    return index;
}

```

如果不是标识符也不是常数，就进入符号的判别，如果都不符合或者中间有格式错误，会报错：

```

void error()
{
    System.err.println("Error: wrong char "+ch);
    System.exit(0);
}

```

将词法分析作为单独的一遍，应该一次性读入整个源文件，循环执行词法分析自动机知道指针指到程序末尾：

```

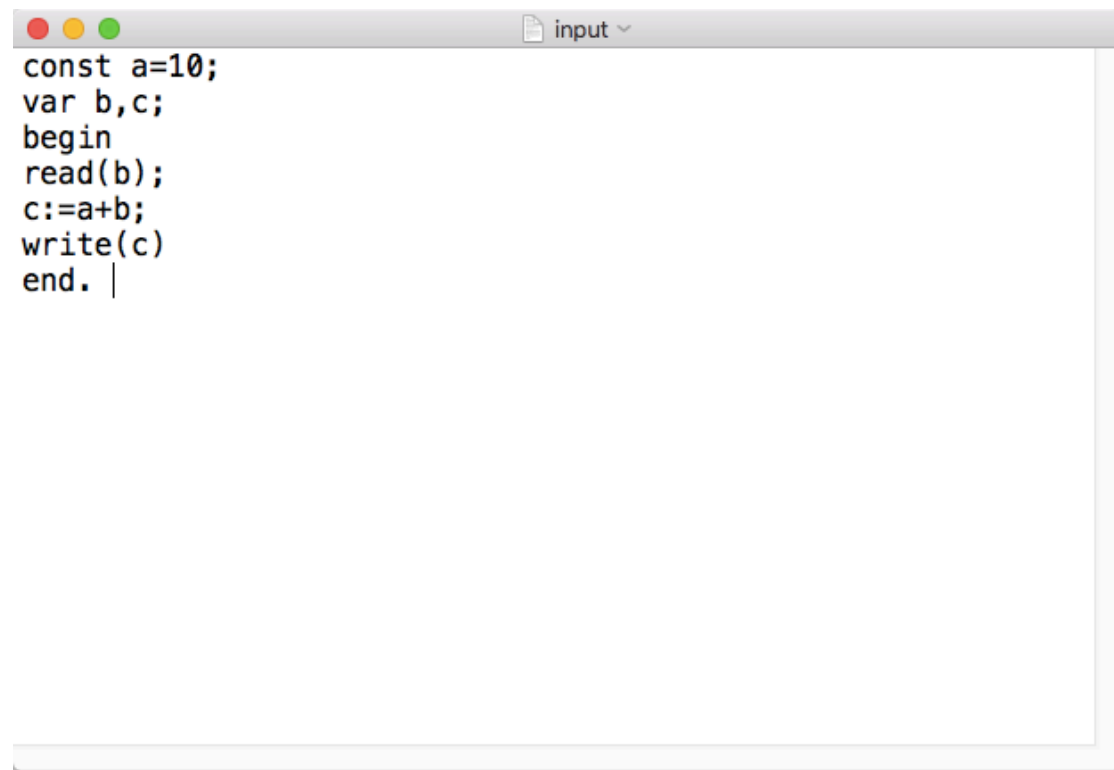
public void begin() {
    createMap();
    String str = loadFile();

    while (index<str.length()) {
        onFA(str);
    }
}

```

四，调试结果

输入一段程序：

A screenshot of a code editor window with a title bar containing three colored circles (red, yellow, green) and a dropdown menu labeled 'input'. The editor contains the following Pascal code:

```
const a=10;  
var b,c;  
begin  
  read(b);  
  c:=a+b;  
  write(c)  
end. |
```

运行得到结果，可以看到结果正确无误：



Problems Console

<terminated> LA_FA [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0

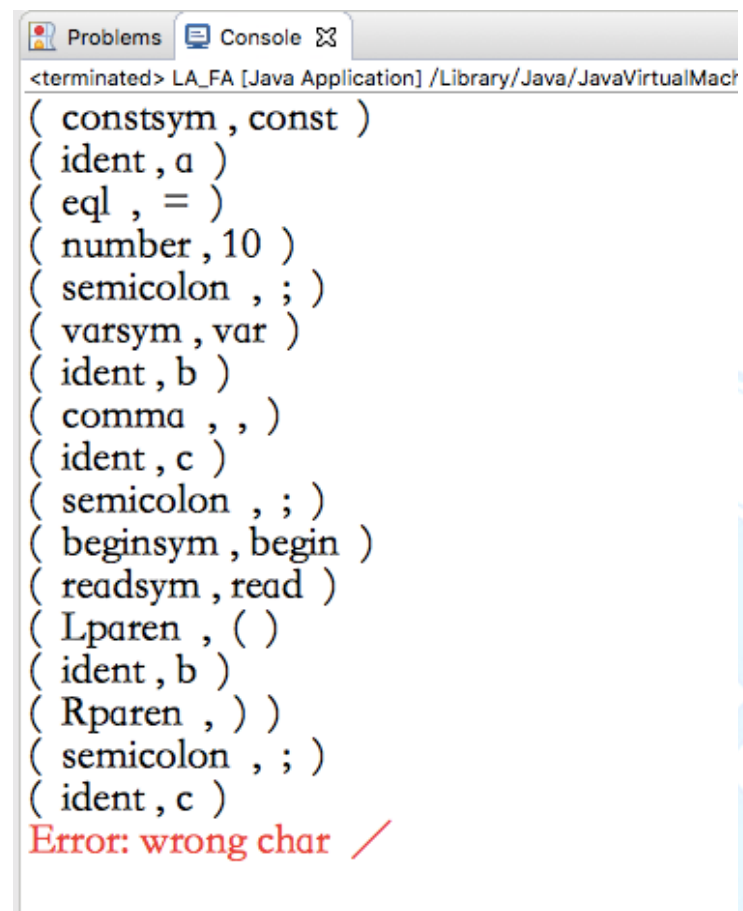
```
( constsym , const )
( ident , a )
( eql , = )
( number , 10 )
( semicolon , ; )
( varsym , var )
( ident , b )
( comma , , )
( ident , c )
( semicolon , ; )
( beginsym , begin )
( readsym , read )
( Lparen , ( )
( ident , b )
( Rparen , ) )
( semicolon , ; )
( ident , c )
( becomes , := )
( ident , a )
( plus , + )
( ident , b )
( semicolon , ; )
( writesym , write )
( Lparen , ( )
( ident , c )
( Rparen , ) )
( endsym , end )
( period , . )
```

异常错误演示：输入非法符号

in

```
const a=10;
var b,c;
begin
read(
c: /=a+b;
write(c)
end.
```

词法分析器会报错:



```
Problems Console
<terminated> LA_FA [Java Application] /Library/Java/JavaVirtualMach
( constsym , const )
( ident , a )
( eql , = )
( number , 10 )
( semicolon , ; )
( varsym , var )
( ident , b )
( comma , , )
( ident , c )
( semicolon , ; )
( beginsym , begin )
( readsym , read )
( Lparen , ( )
( ident , b )
( Rparen , ) )
( semicolon , ; )
( ident , c )
Error: wrong char /
```