# 实验：词法分析器

**周炎亮 2018202196 信息学院**

## 一、实验目的

用lex语言实现PL/0编译器的词法分析部分

•理解编译器的工作机制，掌握编译器的构造方法

•掌握词法分析器的生成工具LEX的用法

•掌握语法分析器的生成工具YACC的用法

## 二、实验/代码思路

因为词法分析器只需识别不同类别的符号，因此只需要从左到右、从上到下对整个代码段进行正则匹配，并归到相应类中即可。

### 扩展语法的EBNF范式

扩展0：<字符串类型>::="<任意字符>{<任意字符>}"

扩展1：<浮点类型>::=[+-]<数字>{<数字>}\.<数字>{<数字>}[E|e[+-]<数字>{<数字>}]

扩展2：<整型数组>::=<标识符>\(<无符号整数>\:<无符号整数>\)

扩展3：<for型循环语句>::=for<赋值语句>to<整数>step<整数>

扩展4：<repeat型循环语句>::=repeat<语句>until<条件>

### 宏定义

因为yylex返回的是int类型的值，故需要在一开始对不同类进行宏定义以便返回以及输出的识别。

```
%{
#include<stdio.h>
int num_lines=1,num_chars=1;
#define KEY 1
#define IDENTIFIER 2
#define CONSTANT 3
#define OPERATOR 4
#define DELIMITER 5
#define OTHER 6
%}
```

### 正规定义

**K类（关键字）**

因为关键字数量有限且无固定正则匹配规则，故只需将PL/0中的13个保留字全部抄入并以|分隔即可。

而对于**扩展0**和**扩展1**，为了与PL/0中原本定义整型的var区分，将字符串类型以string定义，将浮点类型以float定义，故将string和float也加入关键字列表中。

对于**扩展3**，定义for循环的语句规则为for i = n1 to n2 step m，故将三个关键字for、to、step加入。

对于**扩展4**，定义repeat为repeat<语句>until<条件>，将repeat和until加入。

```
key
if|then|while|do|read|write|call|begin|end|const|var|procedure|odd|for|to|step|r
epeat|until|string|float
```

**I类（标识符）**

符合语法的标识符以字母、数字和下划线组成，不以数字开头且不能是关键字。但具体标识符是否符合规范可放到语法部分处理，在本代码中代码段满足"以字母、数字和下划线组成"的条件即被视为标识符。

因此标识符的正则表达式为：

```
letter [A-Za-z]
digit [0-9]
id ({letter}|{digit}|_)*
```

对于**扩展2**的数组识别，PL/0语法定义其为形如array(i:j)，但如果将其整个识别会对之后的语法分析产生阻碍，故最后将其分开识别，array视为标识符，(、:和)视为界符，而i和j则视为常量。

**C类(常量)**

对于PL/0的整型，其为由0-9数字组成的串，且前面可能会有+-号，故整型的正则表达式为：

```
integer [+-]?{digit}+
```

而对于**扩展1**中定义的字符串，定义其两边被双引号包裹且不含换行符，故在识别时的正则表达式为：

```
string \"[^\n^\"]+\"
```

对于**扩展2**中的浮点数，在整型的基础上，其后会有小数点以及又一0-9组成的数串。且加以扩展，使其可以支持科学计数法，故其正则表达式为：

```
float [+-]?{digit}+(\.{digit}+)?((E|e)[+-]?{digit}+)?
```

**O类（算符）**

因为算符数量有限，故同关键字，直接将其放入即可，但对有些算符需进行转义处理

```
operator \+|\-|\*|\/|\\|\<|\>|"<="|">="|"=="|"!="|":="|"="
```

**D类（界符）**

同算符

```
delimiter ":"|"."|","|";"|\[|\]|\"|\'|\{|\}|\(|\)
```

**T类（其他）**

因为上面五类已基本囊括了正规代码中的所有规则，因此属于本类的可视为类似于非法字符之类的非法代码段。而其正则匹配也只需是不属于之前提到过的几类即可，同时也除去空格换行符一类的字符

```
other [^ \n\r{key}{id}{integer}{float}{string}{operator}{delimiter}]
```

## 翻译部分

因为在本题中还需输出被匹配的代码段首字符的行号和列号，因次需要定义分别记录行号和列号的变量num_lines和num_chars。此外还需对没有意义的空格和换行符进行处理：当匹配到tab键(\t)时，num_chars+4；当匹配到空格时，num_chars+1；当匹配到换行符(\n\r)时，num_lines+1并重置num_chars为1。每次匹配到代码段时，num_chars加上该代码段的长度yyleng，在输出时只需减去yyleng即可。

之后，只需对相应代码段进行正则匹配的编写即可。又因为PL/0中标识符长度不能大于10，因此若标识符对应的yyleng大于10，则会出现报错提示，且对整型（<10）和float（<15）也做了对应报错处理。又因为**非法字符**已全部归位T类，因此可用输出的T类作为判断非法字符的标准，不再特别处理。

```
"\n"|"\r" {num_lines++;num_chars=1;}
[ ] {num_chars++;}
"\t" {num_chars+=4;}
{key} {num_chars+=yyleng;return (KEY);}
{id} {num_chars+=yyleng;if(yyleng>10) fprintf(yyout,"Warning: identifier length
is longer than 10.\n");return (IDENTIFIER);}
{integer} {num_chars+=yyleng;if(yyleng>10) fprintf(yyout,"Warning: integer
length is longer than 10.\n");return (CONSTANT);}
{float} {num_chars+=yyleng;if(yyleng>15) fprintf(yyout,"Warning: float length is
longer than 15.\n");return (CONSTANT);}
{string} {num_chars+=yyleng;return (CONSTANT);}
{operator} {num_chars+=yyleng;return (OPERATOR);}
{delimiter} {num_chars+=yyleng;return (DELIMITER);}
{other} {num_chars+=yyleng;return(OTHER);}
```

## 辅助函数

对于main函数，规定用户必须要有一个输入文件，可以有一个输出文件名（若没有则直接输出至屏幕）。

为了方便输出，定义了一个writeout函数，根据yylex的返回值对yytext进行判断，并按格式输出。

```
int yywrap (){
  return 1;
}
void writeout(int c)
{
    switch(c)
    {
        case KEY:
            fprintf(yyout,"%s : K, (%d, %d)\n",yytext,num_lines,num_chars-
yyleng);break;
        case IDENTIFIER:
            fprintf(yyout,"%s : I, (%d, %d)\n",yytext,num_lines,num_chars-
yyleng);break;
        case CONSTANT:
            fprintf(yyout,"%s : C, (%d, %d)\n",yytext,num_lines,num_chars-
yyleng);break;
        case OPERATOR:
            fprintf(yyout,"%s : O, (%d, %d)\n",yytext,num_lines,num_chars-
yyleng);break;
        case DELIMITER:
            fprintf(yyout,"%s : D, (%d, %d)\n",yytext,num_lines,num_chars-
yyleng);break;
```

```
        case OTHER:
            fprintf(yyout,"%s : T, (%d, %d)\n",yytext,num_lines,num_chars-
yyleng);break;
    }
}
int main(int argc, char *argv[])
{
    FILE * fIn;         //PL0文件的指针
    switch(argc)
    {
    /*case 1:           //打开缺省文件
        fIn=fopen("test.frag","r");
        if(fIn == NULL){
            printf("default file is not found\n");
            exit(1);
        }
        else    yyin = fIn;
        break;      */
    case 2:             //打开指定文件
        if ((fIn = fopen(argv[1],"r")) == NULL) {
                printf("File %s is not found.\n",argv[1]);
                exit(1);
            }
            else    yyin=fIn;
        break;
    case 3:
        if ((fIn = fopen(argv[1],"r")) == NULL) {       //打开输入文件
                printf("File %s is not found.\n",argv[1]);
                exit(1);
            }
            else    yyin=fIn;
        yyout=fopen(argv[2], "w");      //打开输出文件
        break;
    default:
        printf("useage:flex [filename]\n");
        exit(1);
    }
    int c;
    while(c=yylex())
        writeout(c);

    fclose(fIn);
    if(argc==3) fclose(yyout);
    return 0;
}
```

## 三、实验结果截图

对于给定的测试文件column.pl0：

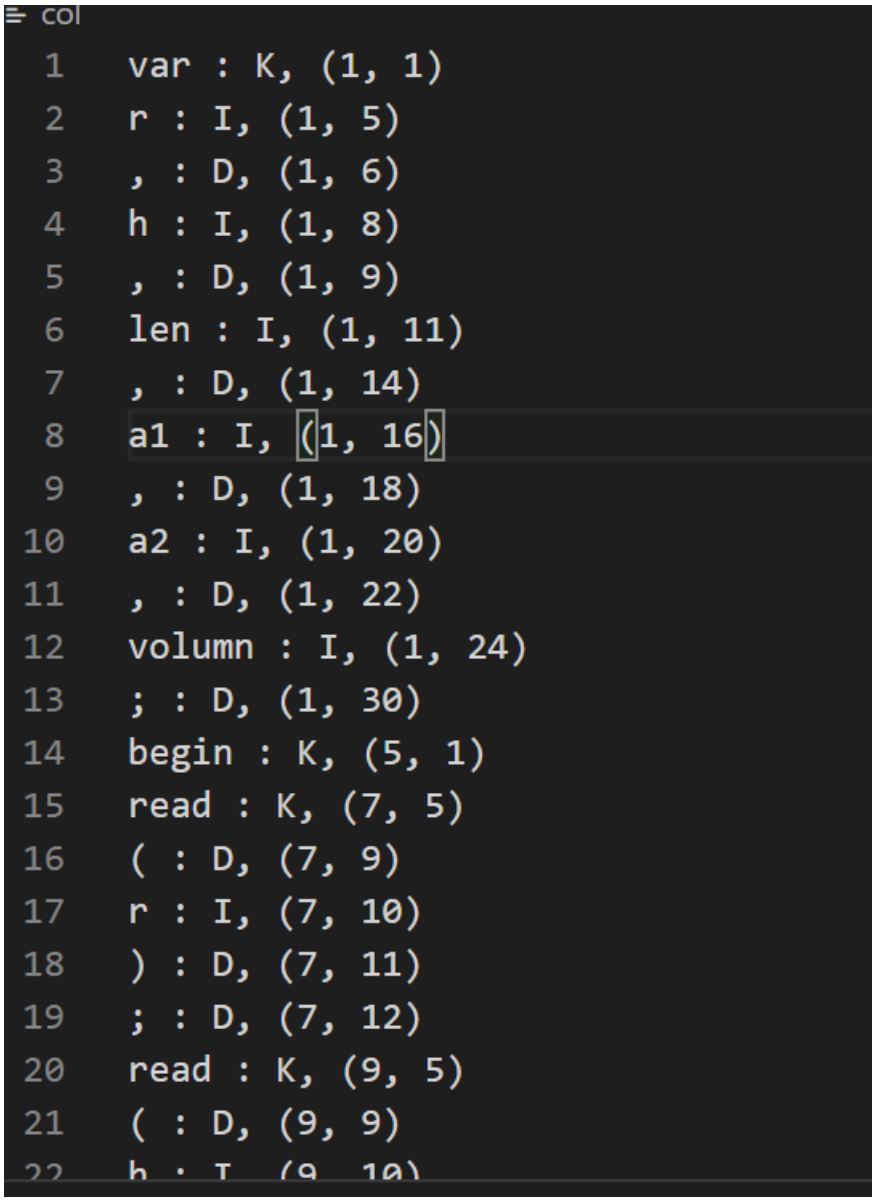
```
var r, h, len, a1, a2, volumn;

begin
    read(r);
    read(h);

    len := 2 * 3 * r;
```

```
    a1 := 3 * r * r;
    a2 := a1 + a1 + len * h;
    volumn := a1 * h;

    write(len);
    write(a1);
    write(a2);
    write(volumn);
end.
```




对于扩展语法部分：

```
var i,j,a(1:10);
string b;
float c;

for i = 1 to 10
    b = b + "aaa";
    c = c + 0.5;
j = 1;
repeat
    a(j) = j
    j = j + 1
```

```
until j = 10;
var asd1234567890asd = -12345678901;

@
00hhh
#$%^
_asdf
0.01e-1
```

```
53  + : O, (21, 11)
54  1 : I, (21, 13)
55  until : K, (23, 1)
56  j : I, (23, 7)
57  = : O, (23, 9)
58  10 : I, (23, 11)
59  ; : D, (23, 13)
60  var : K, (25, 1)
61  Warning: identifier length is longer than 10.
62  asd1234567890asd : I, (25, 5)
63  = : O, (25, 22)
64  Warning: integer length is longer than 10.
65  -12345678901 : C, (25, 24)
66  ; : D, (25, 36)
67  @ : T, (29, 1)
68  00hhh : I, (31, 1)
69  # : T, (33, 1)
70  $ : T, (33, 2)
71  % : T, (33, 3)
72  ^ : T, (33, 4)
73  _asdf : I, (35, 1)
74  0.01e-1 : C, (37, 1)
```