实验一 PL/0 语言词法分析器

Fan Xinyan

2019.3.29

一、实验目的

利用 flex 工具生成一个 PL/0 语言的词法分析程序，对 PL/0 语言的源程序进行扫描，识别出单词符号的类别，输出各种符号的信息。

二、实验要求

（一）基本要求

对于 PL/0 源程序，将程序中的单词分为六类，然后按照单词符号出现的顺序依次输出各个单词符号的种类和出现在源程序的位置（行数和列数）。

PL/0 语言中单词的种类说明见表 1。

表 1 单词符号种类

单词种类	类别表示	例子
关键词	K 类	var、while、do 等
标识符	I 类	a、b、test 等长度小于 11 的变量名、过程名
常量	C 类	1、2、3 等长度小于 15 的整数
算符	O 类	+、-、：=等运算符
界符	D 类	； 、. 、, 等符号
其他（非法符、空白符等）	T 类	3a、3@a、a_3_4 等非法符号 \t\r\v 和空格等空白符号

（二）扩展要求

实现 for 循环语句。能够识别 for 语句的关键词。

本实验中设计的 for 语句见代码 1.

代码 1 – for 语句

```
1  for i := 1, i <= 10 do
2      begin
3          a := a + 1;
4          i := i + 1;
5      end;
```

三、实验环境

词法分析生成工具：flex

编程语言：C

操作系统：macOS

四、实验内容

（一）PL/0 源程序分析

对每一类单词符号进行分析后，总结如下：

1、K 类

基本单词有 13 个 :if、then、while、do、read、write、call、procedure、begin、end、const、var、odd；扩展 for 语句后添加关键词：for。

2、I 类

长度小于 1 的变量名、过程名。以字母开头的字母和数字的组合。

3、C 类

长度小于 15 的整数。

4、O 类

涉及到的运算符有："+"、"-"、"*"、"/"、"#"、"<"、">"、"="、"<>"、"!="、">="、"<="、":="

5、D 类

涉及到的界符有："("、")"、","、":"、";"、"."

6、T 类

（1）空白字符：包括空格、制表符、换行符

（2）非法单词：定义非法符号有?!@#¥&|$\·_。非法单词的组合类型有：包含非法符号的字符串、数字开头的数字字母组合、浮点数。

（二）辅助定义和识别规则的设计

1、不同类型单词符号的正规式表达（辅助定义）

不同类型单词符号的正规式的设计方案见表 2.

表 2 单词符号的正规式表达

表示	正规式	含义
digit	[0-9]	单个数字
letter	[a-zA-Z]	单个字母
identifier	{letter}({letter}|{digit})*	I 类字符
number	{digit}+	C 类字符
wrong_1	([^\t\n\f\v\r]*)([_?!$&#@\`]+)({letter}|{digit})*	含非法符号的字符串，T 类字符
wrong_2	({digit}+){letter}({letter}|{digit})*	数字开头的数字字母组合，T 类字符
wrong_3	({digit}+)([.]+)({digit}+)	浮点数，T 类字符
whitespace	[\t\v]	空白符，T 类字符
newline	[\n\r]	换行符，用于更新行列号

2、识别规则

当识别到符合标识符规则的单词后执行 analyse 程序。识别规则见代码 2.

代码 2 – 识别规则

```
1  /*识别 K 类字符*/
2  "procedure" |"call" |"begin" |"end" |"var" |"const" |"if" |"then"|"while"   |"do"
3  |"read" |"write" |"odd" |"for"     {analyse(1);}
4  /*识别 I 类字符*/
5  {identifier}   {analyse(2);}
6  /*识别 C 类字符*/
7  {number}    {analyse(3);}
8  /*识别 O 类字符*/
9  "+"|"-"|"*"|"/"|"#"|"<"|">"|"="|"<>"|"!="|">="|"<="|":="{analyse(4);}
10 /*识别 D 类字符*/
```

```
11  "("|")"|","|":"|";"    |"."        {analyse(5);}
12  /*识别 T 类字符*/
13  {whitespace}|{wrong_1}|{wrong_2}|{wrong_3} {analyse(6);}
14  /*识别换行符，并进行更新行号和列号操作*/
15  {newline}    {rownum+=1; colnum=1;}
```

（三）核心代码设计

1、程序总体流程

程序执行流程图如图 1 所示。

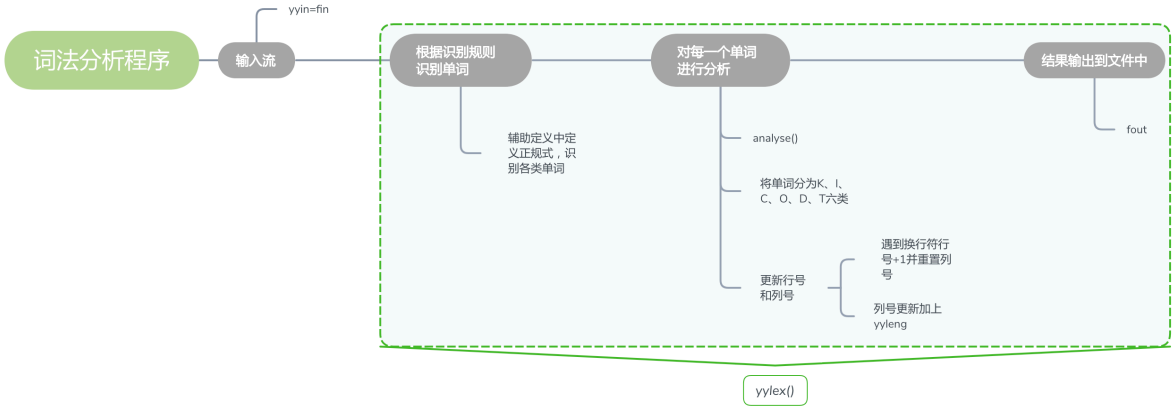


图 1 程序流程图

2、数据结构

设计结构体 token，里边存放每识别出的单词的类型和名称。见代码 3.

代码 3 – 数据结构

```
1  struct token{
2      char *type;/*单词类型*/
3      char *name;/*单词字符串*/
4  }entity[1024];
```

3、识别单词后的分析程序 void analyse(int flag)

该分析程序中，flag 为单词类别对应的序号，并将每一个单词存入一个 token 结构体中，设置其类型并输出到文件，最终更新列号。分析程序见代码 4.

代码 4 – 分析函数

```
1  void analyse(int flag){
2       if (flag == 1) { entity[cnt].type = "K";}/*K 类单词*/
3      else if (flag == 2) {
4            if(yyleng >= 11) { entity[cnt].type = "T";}/*标识符长度需<=10*/
5            else {entity[cnt].type = "I";}}/*I 类单词*/
6      else if (flag == 3){
7            if (yyleng >= 15) { entity[cnt].type = "T";}/*数字长度需<=10*/
8            else {entity[cnt].type = "C";}}/*C 类单词*/
9      else if (flag == 4) { entity[cnt].type = "O";}/*O 类单词*/
10     else if (flag == 5) { entity[cnt].type = "D";}/*D 类单词*/
11     else {entity[cnt].type = "T";}/*T 类单词*/
12      entity[cnt].name = yytext;/*获取单词字符串*/
13      fprintf(fout,  "\"%s\"  :\t%s,\t<%2d,%2d>  \n",  entity[cnt].name,
14 entity[cnt].type, rownum, colnum);/*输出到文件*/
15      colnum += yyleng;/*更新列号，column+字符串长度*/
16      cnt += 1;
17 }
```

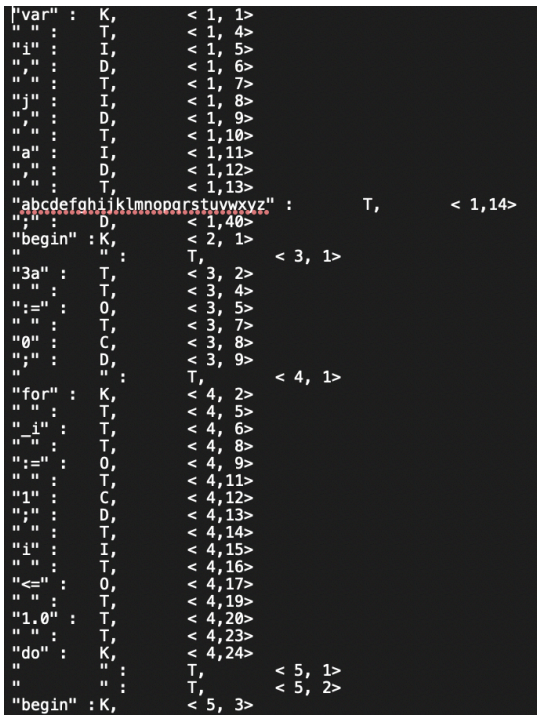
五、实验结果

（一）测试用例

```
var i, j, a, abcdefghijklmnopqrstuvwxyz;
begin
        3a := 0;
        for _i := 1; i <= 1.0 do
                begin
                        1_a := a@3 + 1_;
                        i := i + 123456789012345678901;
                end;
        write(a);
end.
```

其中设计了 for 语句的格式，且包含一些非法单词，如：3a，_i，1.0，1_a，a@3，1_，还有长度不合法的数字和标识符。

（二）输出结果



由结果可以看出，程序能够正确识别出 K、I、C、O、D 这五类符号，而且能够正确给出每个单词的行号和列号。对于非法单词，包括长度过长的标识符和数字以及含有非法符号或不合理命名的单词，都能够识别出来，将其划分为 T 类。

六、实验总结

通过本次实验，熟练掌握了 flex 工具的使用方法，能够用它生成一个 PL/0 语言的词法分析程序，对 PL/0 语言的源程序进行扫描，正确识别出单词符号的类别，输出各种符号的类别和位置信息。为进一步的语法分析和语义分析做好了基础工作。