实验报告

一、实验目标

借助 flex 工具实现一个词法分析器,将 SysY 代码中的单词和符号进行分类,然后按照单词符号出现顺序依次输出:原始单词符号、种类、出现在源程序的位置(行数和列数)。其中单词的符号共分为 K、I、C、O、D、T 共 6 类。还需要对注释进行处理,包括单行注释和多行注释。

二、代码说明及思路

首先,在声明部分,除了需要的头文件之外,还声明了两个全局变量,其中 num_lines 用于统计当前行数, num_cols 用于统计当前列数。

```
%{
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int num_lines = 1;
int num_cols = 1;
%}
```

1. K 类, 关键字

关键字包括 int、main、if、else 等等,直接在识别规则处以"int"的形式要求完全匹配即可。动作中的输出按照要求的格式,种类为 K (已匹配的部分存放在 yytext 中); 此外动作中还要记得将 num_cols 加上已匹配内容的长度(yyleng),以统计当前所在列。(这一操作在以后的每次识别中都要进行,此后不再赘述)

"int" {printf("%s: K, (%d, %d)\n", yytext, num_lines, num_cols); num_cols += yyleng;} "main" {printf("%s: K, (%d, %d)\n", yytext, num_lines, num_cols); num_cols += yyleng;} continue" {printf("%s: K, (%d, %d)\n", yytext, num_lines, num_cols); num_cols += yyleng;} 'const" {printf("%s: K, (%d, %d)\n", yytext, num_lines, num_cols); num_cols += yyleng;} 'if" {printf("%s: K, (%d, %d)\n", yytext, num_lines, num_cols); num_cols += yyleng;} 'else" {printf("%s: K, (%d, %d)\n", yytext, num_lines, num_cols); num_cols += yyleng;} return" {printf("%s: K, (%d, %d)\n", yytext, num_lines, num_cols); num_cols += yyleng;} 'void" {printf("%s: K, (%d, %d)\n", yytext, num_lines, num_cols); num_cols += yyleng;} while" {printf("%s: K, (%d, %d)\n", yytext, num_lines, num_cols); num_cols += yyleng;} enum" {printf("%s: K, (%d, %d)\n", yytext, num_lines, num_cols); num_cols += yyleng;} 'switch" {printf("%s: K, (%d, %d)\n", yytext, num_lines, num_cols); num_cols += yyleng;} 'case" {printf("%s: K, (%d, %d)\n", yytext, num_lines, num_cols); num_cols += yyleng;} 'for" {printf("%s: K, (%d, %d)\n", yytext, num_lines, num_cols); num_cols += yyleng;} 'sizeof" {printf("%s: K, (%d, %d)\n", yytext, num_lines, num_cols); num_cols += yyleng;} 'static" {printf("%s: K, (%d, %d)\n", yytext, num_lines, num_cols); num_cols += yyleng;} 'typedef" {printf("%s: K, (%d, %d)\n", yytext, num_lines, num_cols); num_cols += yyleng;} 'break" {printf("%s: K, (%d, %d)\n", yytext, num_lines, num_cols); num_cols += yyleng;} 'do" {printf("%s: K, (%d, %d)\n", yytext, num lines, num cols); num cols += yyleng;} 'struct" {printf("%s: K, (%d, %d)\n", yytext, num_lines, num_cols); num_cols += yyleng;} 'signed" {printf("%s: K, (%d, %d)\n", yytext, num_lines, num_cols); num_cols += yyleng;} 'unsigned" {printf("%s: K, (%d, %d)\n", yytext, num_lines, num_cols); num_cols += yyleng;} "default" {printf("%s: K, (%d, %d)\n", yytext, num_lines, num_cols); num_cols += yyleng;}

2.1 类. 标识符

标识符以字母或下划线开头,后接任意个字母数字下划线。采用辅助定义IDENTIFIER、识别时输出为 I 类。

|IDENTIFIER [_A-Za-z][_A-Za-z0-9]*
|{IDENTIFIER} {printf("%s: I, (%d, %d)\n", yytext, num_lines, num_cols); num_cols += yyleng;}
| 3. C 类,常量

常量有三种表示方法: 十进制, 八进制, 十六进制。十进制以非 0 的数开头,

后接任意数字; 八进制以 0 开头, 后接 0-7 的任意数字; 十六进制以 0x 或 0X 开头, 后接任意个 0-F。分别进行辅助定义, 并汇总为 INTCONST。

DECIMAL 0|[1-9][0-9]*
OCTALCONS 0[0-7]+
HEXCONS 0[xX][0-9a-fA-F]+
INTCONST {DECIMAL}|{OCTALCONS}|{HEXCONS}

{INTCONST} {printf("%s: C, (%d, %d)\n", yytext, num_lines, num_cols); num_cols += yyleng;}

4. O, 算符

算符用辅助定义 OPERATOR 表示,将可能的算符全部列出即可。

 $\mathsf{OPERATOR} \ ([\setminus + \setminus - \setminus \times \setminus \cdot \times \setminus - \setminus - \setminus \times \setminus]] | " + = " | " - = " | " + = " | " - = " | " < = " | " < = " | " = = " | " + + " | " - - " | " & & " | " | | ")$

{OPERATOR} {printf("%s: O, (%d, %d)\n", yytext, num_lines, num_cols); num_cols += yyleng;}

5. D, 界符

界符同上、辅助定义为 BORDER

BORDER [\(\)\{\}\[\]\,\;\"\"]

{BORDER} {printf("%s: D, (%d, %d)\n", yytext, num_lines, num_cols); num_cols += yyleng;}

6. T, 其他

标识符不能以数字开头, 若以数字开头则应识别为 T。还有在识别规则的的最后, 用 . 匹配除换行符之外的任意字符, 已匹配可能出现的其他情况。

[0-9][_A-Za-z0-9]* {printf("%s: T, (%d, %d)\n", yytext, num_lines, num_cols); num_cols += yyleng;} {printf("%s: T, (%d, %d)\n", yytext, num_lines, num_cols); num_cols += yyleng;}

7. 注释

注释不进行输出, 但要统计行数

对于单行注释, 先匹配//, 中间为除换行符之外的任意字符, 最后匹配换行符\n。识别到单行注释时行数要加一。

NOTE_S \/\/(.)*\n

```
{NOTE_S}* {++num_lines;num_cols = 1;}
```

对于多行注释,为了统计行数,采用 BEGIN 条件激活规则,识别到/*后即进入 COMMENT 部分,然后匹配到换行符就增加行数,其他字符只增加列数,直至匹配到*/为止,结束该部分。

```
commentbegin "/*"
commentend "*/"
%x COMMENT
```

```
{commentbegin} {num_cols += yyleng; BEGIN COMMENT;}
<COMMENT>. {num_cols += yyleng;}
<COMMENT>\n {++num_lines; num_cols = 1;}
<COMMENT>{commentend} {num_cols += yyleng; BEGIN INITIAL;}
```

最后的用户子程序部分:

```
%%
int yywrap(){
    return 1;
}
int main(){
    yylex();
    return 0;
}
```

三、实验过程

.l 源程序写好后,用 flex 运行生成 lex.yy.c 文件,再用 gcc 编译运行即可。图中所示为输入样例 6 的输出结果,经检验符合要求。

```
lixinyu@LAPTOP-OGLG3R4D: $ flex analyze.1
lixinyu@LAPTOP-OGLG3R4D: $ gcc -o test lex.yy.c
lixinyu@LAPTOP-OGLG3R4D: $ ./test<6.txt
int: K, (1, 1)
main: K, (1, 23)
(: D, (1, 27)
): D, (1, 28)
{: D, (1, 30)
int: K, (9, 5)
a: I, (9, 9)
;: D, (9, 10)
}: D, (10, 1)
lixinyu@LAPTOP-OGLG3R4D: $
```

用同样方法测试了其他样例,输出均符合要求。受篇幅限制,在此只展示样例 1 的结果。

```
int: K, (1, 1)
main: K, (1, 5)
(: D, (1, 9)
                                                                                                 2: C,
]: D,
                                  =: 0, (3, 15)
010: C, (3, 16)
;: D, (3, 19)
                                                                                                                     16)
                                                                                                             (8,
                                                                                                             (8,
                                                                                                                     17)
                                 int: K, (4, 5)
test: I, (4, 9)
=: 0, (4, 13)
                                                                                                                    (9,
): D, (1, 10)
                                                                                                 while: K,
                                                                                                                            5)
1: D, (1, 10)
{: D, (1, 11)
int: K, (2, 5)
b: I, (2, 9)
[: D, (2, 10)
10: C, (2, 11)
]: D, (2, 13)
[: D, (2, 14)
10: C, (2, 15)
]: D (2, 17)
                                                                                                 (: D,
                                                                                                             (9,
                                                                                                                    10)
                                                                                                 a: I,
                                                                                                             (9,
                                                                                                                    11)
                                                                                                                    12)
                                  21474836472147483647: C, (4, 14) >: 0,
                                                                                                             (9,
                                                                                                                    13)
                                                                                                             (9,
                                  ;: D, (4, 34)
                                 int: K, (5, 5)
test: I, (5, 9)
=: 0, (5, 13)
                                                                                                 ): D,
                                                                                                             (9, 14)
                                                                                                 {: D,
                                                                                                             (9, 15)
                                                                                                             (10, 9)
(10, 10)
                                                                                                 a: I,
                                 -: 0, (3, 15)

0xabcde: C, (5, 14)

:: D, (5, 21)

int: K, (6, 5)

666bbb666: T, (6, 9)
                                                                                                 =: 0,
]: D, (2, 17)
                                                                                                             (10, 11)
                                                                                                 a: I,
            (2, 18)
(2, 19)
                                                                                                             (10, 12)
[: D,
                                                                                                 -: 0,
                                                                                                 1: C,
10: C,
                                                                                                             (10, 13)
            (2, 19)
(2, 21)
(2, 22)
(2, 23)
(2, 24)
(3, 5)
                                                                                                ;: D, (10, 14)
if: K, (11, 9)
(: D, (11, 11)
a: I, (11, 12)
==: 0, (11, 13)
]: D,
                                 ;: D, (6, 18)
int: K, (7, 5)
aaalllaaa: I,
, : D,
a: I,
;: D,
                                  ;: D, (7, 18)
b: I,
                                  a: I,
                                             (8, 5)
[: D,
                                                                                                 5: C, (11, 15)
             (3,
                   6)
                                 =: 0,
                                             (8, 6)
2: C,
]: D,
                   7)
8)
             (3,
                                  b: I,
                                                   7)
8)
                                                                                                 ): D, (11, 16)
                                              (8,
                                                                                                break: K, (12, ;: D, (12, 18) }: D, (13, 5)
             (3,
                                  [: D,
                                                                                                                              13)
                                              (8,
[: D,
             (3,
                    9)
                                  2: C,
                                              (8, 9)
2: C,
]: D,
[: D,
             (3,
                    10)
                                  ]: D,
[: D,
                                              (8, 10)
                                                                                                 return: K, (14 a: I, (14, 12)
             (3,
                    11)
                                                                                                                      (14, 5)
                                              (8, 11)
                    12)
                                 2: C,
]: D,
             (3,
                                              (8, 12)
                                                                                                 ;: D,
}: D,
 2: C,
             (3,
                    13)
                                                                                                             (14, 13)
                                              (8,
                                                     13)
             (3,
                    14)
                                                                                                             (15, 1)
                                       D,
                                              (8,
                                                     14)
```