

# 中国矿业大学计算机学院

## 2019 级本科生课程设计报告

课程名称 系统软件开发实践

报告时间 2022.2.25

学生姓名 王杰永

学    号 03190886

专    业 计算机科学与技术

任课教师 张博

## 成绩考核

编号	课程教学目标	占比	得分
1	<b>目标 1:</b> 针对编译器中词法分析器软件要求，能够分析系统需求，并采用 FLEX 脚本语言描述单词结构。	15%	
2	<b>目标 2:</b> 针对编译器中语法分析器软件要求，能够分析系统需求，并采用 Bison 脚本语言描述语法结构。	15%	
3	<b>目标 3:</b> 针对计算器需求描述，采用 Flex/Bison 设计实现高级解释器，进行系统设计，形成结构化设计方案。	30%	
4	<b>目标 4:</b> 针对编译器软件前端与后端的需求描述，采用软件工程进行系统分析、设计和实现，形成工程方案。	30%	
5	<b>目标 5:</b> 培养独立解决问题的能力，理解并遵守计算机职业道德和规范，具有良好的法律意识、社会公德和社会责任感。	10%	
总成绩			
指导教师		评阅日期	

## Flex 实验二

1 实验内容.....	1
2 Flex 源码分析 .....	1
2.1 lex2-1.l 源码分析 .....	1
2.2 lex2-2.l 源码分析 .....	2
3 实验结果.....	3
3.1 lex2-1 实验结果 .....	4
3.2 lex2-2 实验结果 .....	5
4 实验总结.....	6
4.1 你在编程过程中遇到了哪些难题? .....	6
4.2 你对你的程序的评价? .....	6
4.3 你的收获有哪些? .....	7

## 1 实验内容

1) 阅读《Flex/Bison.pdf》第一章，第二章，掌握 Flex 基础知识。

2) 利用 Flex 实现用于 C 语言子集 C1 的词法分析器。其中，子集 C1 的要求如下：

a. 语言的关键字：else if switch for int float return void while。所有的关键字都是保留字，并且必须是小写。

b. 专用符号：+ - \* / < <= > >= == != = ; , ( ) [ ] { }

c. 标识符（ID）和 整型常数（NUM）通过下列正则表达式定义：

• L = [a-zA-Z\_]

• D = [0-9]

• ID = {L} ({L} | {D})\*

• NUM = [1-9] {D}\*

d. 空格由空白、换行符和制表符组成。

e. 注释用通常的 C 语言符号/\* ... \*/ 或 //。

3) 在实现以上基本功能的基础上，参考《ANSI C grammar (Lex).pdf》，实现以下功能：

- 输出上述标记所在的行号、列号；
- 忽略注释及其内容，如，注释中的数字/\*123\*/ , //123
- 增加科学记数法；
- 十六进制、八进制常数

## 2 Flex 源码分析

### 2.1 lex2-1.l 源码分析

lex2-1 实现了 C 语言子集 C1 的词法分析功能，对应实验内容 2。

由于实验内容 2 已经给出了标识符 ID 与整型常量 NUM 的正则表达式定义，因此只需给出当词法分析器成功匹配到一个串后执行的动作即可。主要代码如下。

```
1. \n {
2.     currentLine++;
3. }
4. {NUMBER} {
5.     printf("line %d:\tnumber\t%s\n", currentLine, yytext);
```

```

6. }
7. {KEYWORD} {
8.     printf("line %d:\tKeyWord\t%s\n", currentLine, yytext);
9. }
10. {ID} {
11.     printf("line %d:\tID\t%s\n", currentLine, yytext);
12. }
13. {SYMBOL} {
14.     printf("line %d:\tSymbol\t%s\n", currentLine, yytext);
15. }
16. . {
17.
18. }

```

为了更直观的显示，为每个单词额外显示行号。只需要在匹配到换行符\n后使记录行数的全局变量currentLine自增1，就可以实现行数的计算。

对于其余的标识符ID、数字NUMBER、关键字KEYWORD和符号SYMBOL，在匹配到后输出相关内容。

最后，当词法分析器扫描到不识别的字符时，选择了跳过继续扫描的动作。上述代码的行16~行18实现了该功能。

## 2.2 lex2-2.1 源码分析

这一部分要求在上述基础上，额外增加行列号、忽略注释、科学计数法、各个进制数的功能。对应实验内容3。

为了方便表示，首先定义了两个正则表达式，分别表示字母和数字。

$$letter [a-zA-Z\_]\tag{1}$$

$$digit [0-9]\tag{2}$$

### ◆ 十进制数

$$DEC\_NUMBER [+ -]? (0|([1-9]\{digit\}^*))(\.\{digit\}^+)?\tag{3}$$

[+ -]?表示数字前的符号是可选的。符号后由两部分组成。最后一部分(\.\{digit\}^+)?表示小数点与至少一位的数字组合也是可选的，即要么是整型数要么是浮点数。前一部分则是十进制整数的定义。

### ◆ 科学计数法

$$SCI\_NUMBER [-]? ([1-9](\.\{digit\}^+)?|(0.\{digit\}^+))e[-]? \{digit\}^+\tag{4}$$

$([1-9](\backslash.\{digit\}+)?|(0.\{digit\}+))$ 该处可分为两部分，中间使用符号|隔开。前一部分表示非零数字开头的，因此小数部分可选；后一部分是以 0 开头的，因此小数部分必选。

#### ◆ 二进制、八进制以及十六进制

C 语言的二进制数以 0b 或 0B 开头，八进制以 0 开头，十六进制以 0x 或 0X 开头，其正则表达式如下。

$$BIN\_NUMBER [-]? 0[Bb][1][01]^* \quad (5)$$

$$OCT\_NUMBER [-]? 0[1-7][0-7]^* \quad (6)$$

$$HEX\_NUMBER [-]? 0[xX][1-9a-fA-F][0-9a-fA-F]^* \quad (7)$$

#### ◆ 行号与列号

列号使用如下规则实现：当匹配到任意串时使列号增加串的长度；当匹配到换行符时使列号重新设置为 1；特别注意的是，匹配到制表符\t 时则需要使列号加 8。

行号只需要在匹配到换行符时增加一。

#### ◆ 单行注释

单行注释的匹配模式与动作如下。

```
1. SINGLE_LINE_COMMENT "//" [^\n]^*
2. {SINGLE_LINE_COMMENT} {
3.     /*单行注释不需要做动作，因为接下来必定出现的换行符会被匹配，从而更新 line 与
       column*/
4. }
```

当匹配到//时，我们认为遇到了单行注释。忽略其后的所有非换行符以外的内容。同时不需要增加额外动作。原因是单行注释不匹配换行符，结束后的换行符会将行号加一列号重置为一。

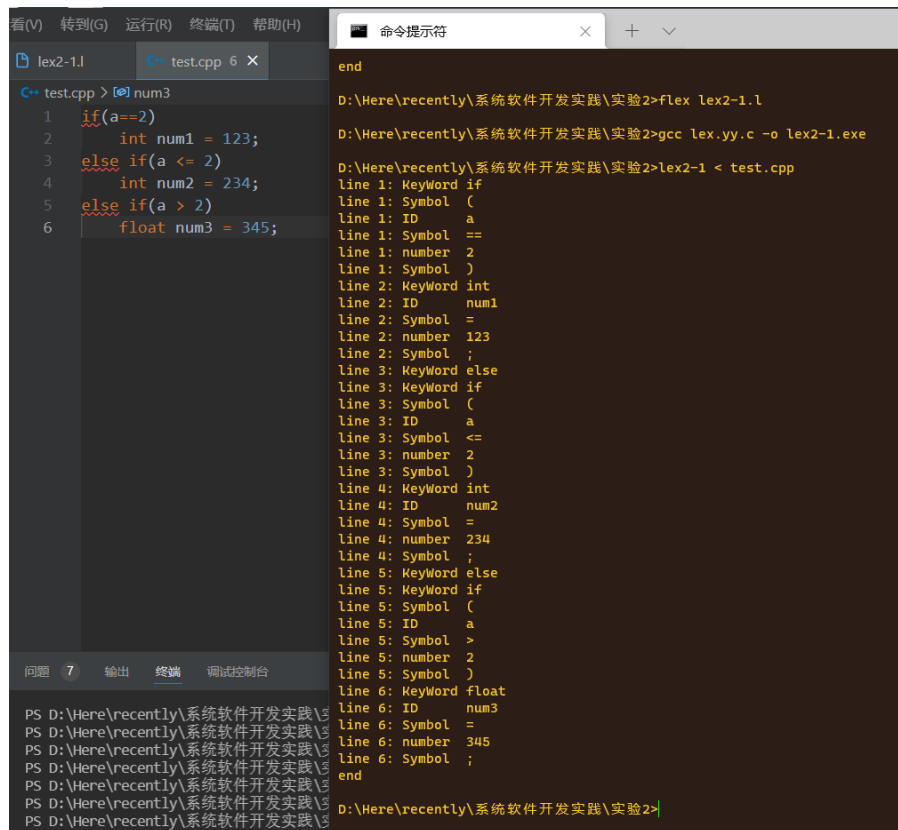
#### ◆ 多行注释

当匹配到/\*时认为开始了多行注释，忽略其后的所有字符直至遇到连续的\*/串，此时多行注释结束。

## 3 实验结果

### 3.1 lex2-1 实验结果

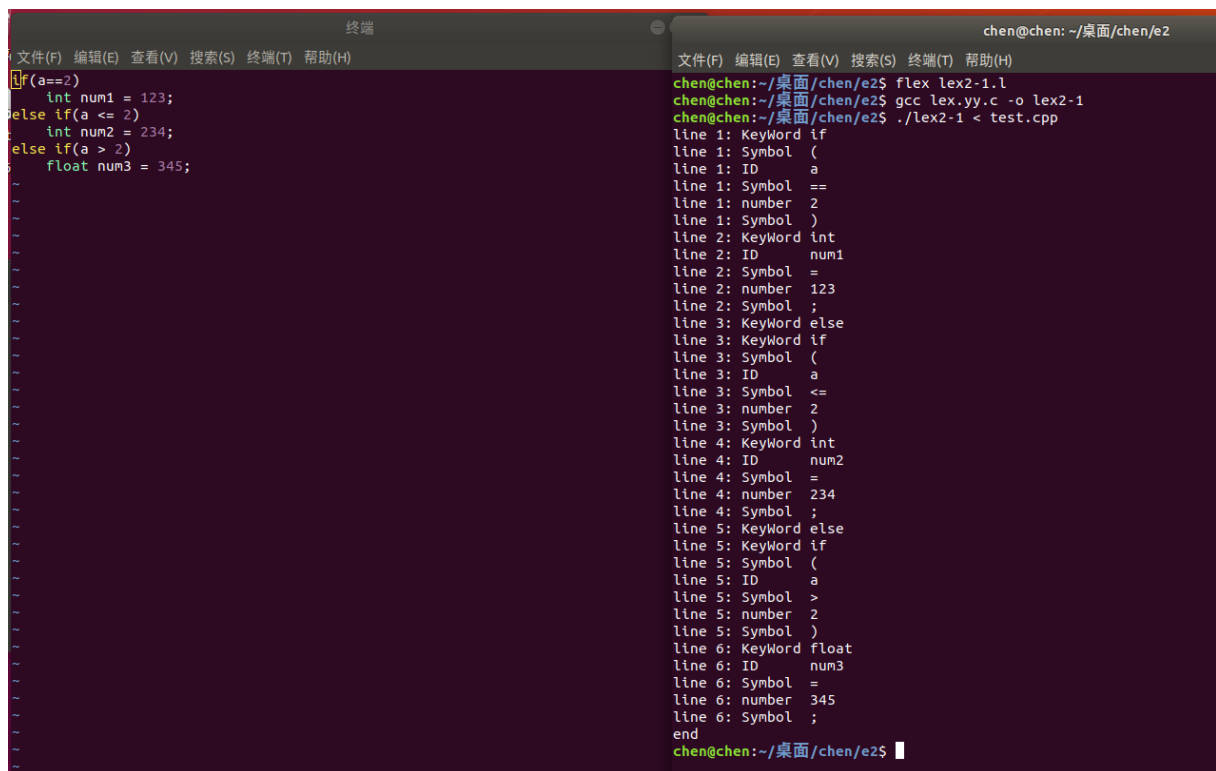
自主编写测试文件，在 windows 和 Linux 系统下的测试结果如下。



```
lex2-1.1 test.cpp 6 X
C++ test.cpp > num3
1 if(a==2)
2     int num1 = 123;
3 else if(a <= 2)
4     int num2 = 234;
5 else if(a > 2)
6     float num3 = 345;

问题 7 输出 终端 调试控制台
PS D:\Here\recently\系统软件开发实践\实验2> gcc lex.yy.c -o lex2-1.exe
PS D:\Here\recently\系统软件开发实践\实验2> flex lex2-1.1
PS D:\Here\recently\系统软件开发实践\实验2> gcc lex.yy.c -o lex2-1.exe
PS D:\Here\recently\系统软件开发实践\实验2> lex2-1 < test.cpp
line 1: Keyword if
line 1: Symbol (
line 1: ID a
line 1: Symbol ==
line 1: number 2
line 1: Symbol )
line 2: Keyword int
line 2: ID num1
line 2: Symbol =
line 2: number 123
line 2: Symbol ;
line 3: Keyword else
line 3: Keyword if
line 3: Symbol (
line 3: ID a
line 3: Symbol <=
line 3: number 2
line 3: Symbol )
line 4: Keyword int
line 4: ID num2
line 4: Symbol =
line 4: number 234
line 4: Symbol ;
line 5: Keyword else
line 5: Keyword if
line 5: Symbol (
line 5: ID a
line 5: Symbol >
line 5: number 2
line 5: Symbol )
line 6: Keyword float
line 6: ID num3
line 6: Symbol =
line 6: number 345
line 6: Symbol ;
end
PS D:\Here\recently\系统软件开发实践\实验2>
```

图 1 Windows 系统下 lex2-1 实验结果



```
chen@chen: ~/桌面/chen/e2
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
if(a==2)
    int num1 = 123;
else if(a <= 2)
    int num2 = 234;
else if(a > 2)
    float num3 = 345;

chen@chen:~/桌面/chen/e2$ flex lex2-1.1
chen@chen:~/桌面/chen/e2$ gcc lex.yy.c -o lex2-1
chen@chen:~/桌面/chen/e2$ ./lex2-1 < test.cpp
line 1: Keyword if
line 1: Symbol (
line 1: ID a
line 1: Symbol ==
line 1: number 2
line 1: Symbol )
line 2: Keyword int
line 2: ID num1
line 2: Symbol =
line 2: number 123
line 2: Symbol ;
line 3: Keyword else
line 3: Keyword if
line 3: Symbol (
line 3: ID a
line 3: Symbol <=
line 3: number 2
line 3: Symbol )
line 4: Keyword int
line 4: ID num2
line 4: Symbol =
line 4: number 234
line 4: Symbol ;
line 5: Keyword else
line 5: Keyword if
line 5: Symbol (
line 5: ID a
line 5: Symbol >
line 5: number 2
line 5: Symbol )
line 6: Keyword float
line 6: ID num3
line 6: Symbol =
line 6: number 345
line 6: Symbol ;
end
chen@chen:~/桌面/chen/e2$
```

图 2 Linux 系统下 lex2-1 实验结果

### 3.2 lex2-2 实验结果

lex2-2 实现了不同进制以及科学计数法的词法规则，自行编写数字测试文件进行测试。

```
D:\Here\recently\系统软件开发实践\实验2>lex2-2 < test.cpp
line 1 column 1: String "十进制数"
line 2 column 1: Dec_Number +1.8
line 3 column 1: Dec_Number -1.8
line 4 column 1: Dec_Number +0.1
line 5 column 1: Dec_Number 0.1
line 6 column 1: Dec_Number 0.05
line 7 column 1: Dec_Number -0.007
line 8 column 1: String "科学计数法"
line 9 column 1: Sci_Number 1e5
line 10 column 1: Sci_Number 1e-5
line 11 column 1: Sci_Number 0.4e7
line 12 column 1: Sci_Number 0.01e-3
line 13 column 1: Sci_Number 1.33333e7
line 14 column 1: Sci_Number -1e5
line 15 column 1: Sci_Number -1e-5
line 16 column 1: Sci_Number -0.4e7
line 17 column 1: Sci_Number -0.01e-3
line 18 column 1: Sci_Number -1.33333e7
line 19 column 1: String "二进制"
line 20 column 1: Bin_Number 0b1110
line 21 column 1: Bin_Number 0B10
line 22 column 1: Bin_Number -0B101
line 23 column 1: String "八进制"
line 24 column 1: Oct_Number 0765
line 25 column 1: Oct_Number -07102
line 26 column 1: String "十六进制"
line 27 column 1: Hex_Number 0x8A33FA9F
line 28 column 1: Hex_Number -0Xfffff
line 29 column 1: Hex_Number 0XFa101De
end
```

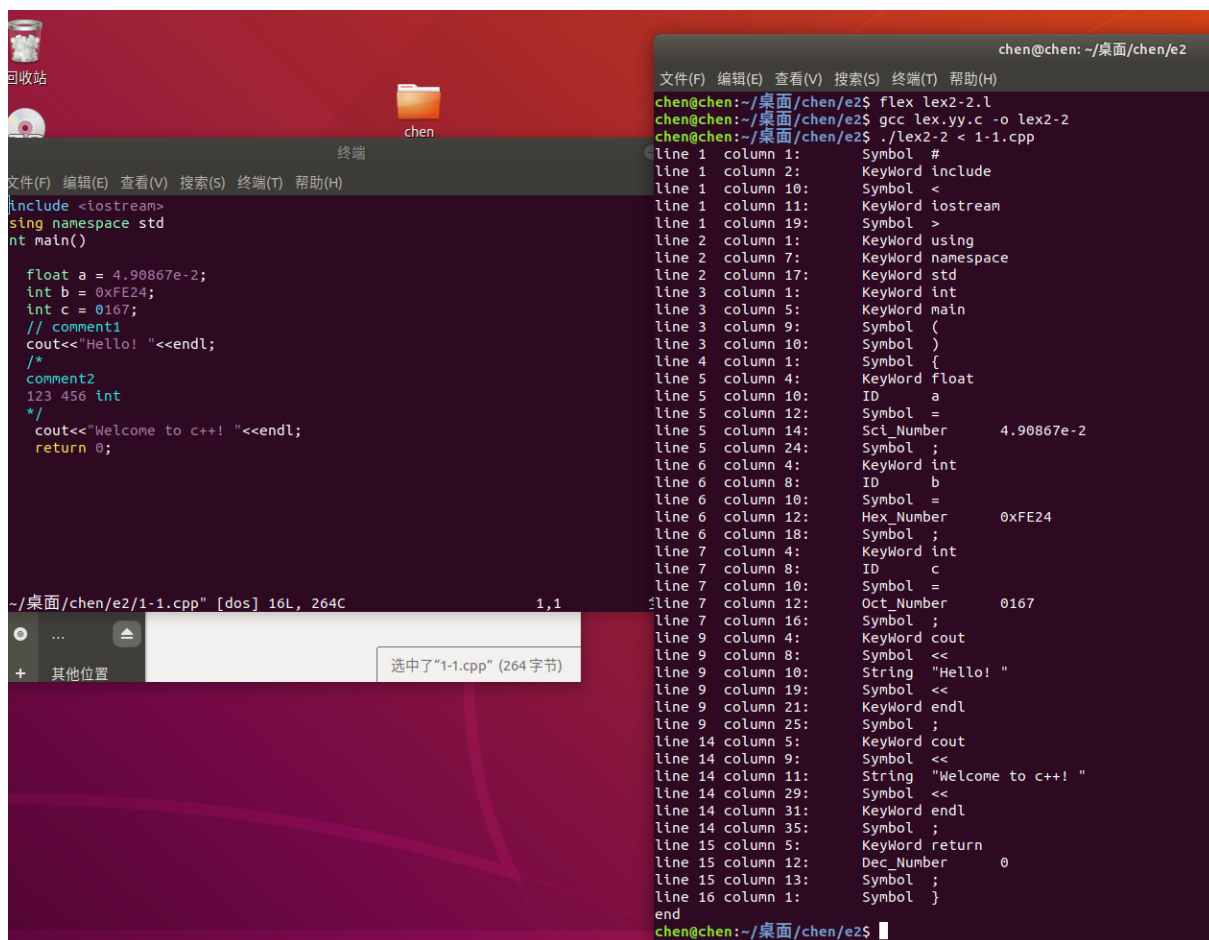
图 3 各类数字测试结果

使用资料中的测试文件 1-1.cpp，在 Windows 与 Linux 系统下的测试结果如下图。

```
D:\Here\recently\系统软件开发实践\实验2>lex2-2 < 1-1.cpp
line 1 column 1: Symbol #
line 1 column 2: Keyword include
line 1 column 10: Symbol <
line 1 column 11: Keyword iostream
line 1 column 19: Symbol >
line 2 column 1: Keyword using
line 2 column 7: Keyword namespace
line 2 column 17: Keyword std
line 3 column 1: Keyword int
line 3 column 5: Keyword main
line 3 column 9: Symbol (
line 3 column 10: Symbol )
line 4 column 1: Symbol {
line 5 column 4: Keyword float
line 5 column 10: ID a
line 5 column 12: Symbol =
line 5 column 14: Sci_Number 4.90867e-2
line 5 column 24: Symbol ;
line 6 column 4: Keyword int
line 6 column 8: ID b
line 6 column 10: Symbol =
line 6 column 12: Hex_Number 0xFE24
line 6 column 18: Symbol ;
line 7 column 4: Keyword int
line 7 column 8: ID c
line 7 column 10: Symbol =
line 7 column 12: Oct_Number 0167
line 7 column 16: Symbol ;
line 9 column 4: Keyword cout
line 9 column 8: Symbol <<
line 9 column 10: String "Hello! "
line 9 column 19: Symbol <<
line 9 column 21: Keyword endl
line 9 column 25: Symbol ;
line 14 column 5: Keyword cout
line 14 column 9: Symbol <<
line 14 column 11: String "Welcome to c++! "
line 14 column 29: Symbol <<
line 14 column 31: Keyword endl
line 14 column 35: Symbol ;
line 15 column 5: Keyword return
line 15 column 12: Dec_Number 0
line 15 column 13: Symbol ;
line 16 column 1: Symbol }
end
```

图 4 Windows 系统下 lex2-2 实验结果





```
chen@chen: ~/桌面/chen/e2
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
chen@chen:~/桌面/chen/e2$ flex lex2-2.1
chen@chen:~/桌面/chen/e2$ gcc lex.yy.c -o lex2-2
chen@chen:~/桌面/chen/e2$ ./lex2-2 < 1-1.cpp

line 1 column 1: Symbol #
line 1 column 2: Keyword include
line 1 column 10: Symbol <
line 1 column 11: Keyword iostream
line 1 column 19: Symbol >
line 2 column 1: Keyword using
line 2 column 7: Keyword namespace
line 2 column 17: Keyword std
line 3 column 1: Keyword int
line 3 column 5: Keyword main
line 3 column 9: Symbol (
line 3 column 10: Symbol )
line 4 column 1: Symbol {
line 5 column 4: Keyword float
line 5 column 10: ID a
line 5 column 12: Symbol =
line 5 column 14: Sci_Number 4.90867e-2
line 5 column 24: Symbol ;
line 6 column 4: Keyword int
line 6 column 8: ID b
line 6 column 10: Symbol =
line 6 column 12: Hex_Number 0xFE24
line 6 column 18: Symbol ;
line 7 column 4: Keyword int
line 7 column 8: ID c
line 7 column 10: Symbol =
line 7 column 12: Oct_Number 0167
line 7 column 16: Symbol ;
line 9 column 4: Keyword cout
line 9 column 8: Symbol <<
line 9 column 10: String "Hello! "
line 9 column 19: Symbol <<
line 9 column 21: Keyword endl
line 9 column 25: Symbol ;
line 14 column 5: Keyword cout
line 14 column 9: Symbol <<
line 14 column 11: String "Welcome to c++! "
line 14 column 29: Symbol <<
line 14 column 31: Keyword endl
line 14 column 35: Symbol ;
line 15 column 5: Keyword return
line 15 column 12: Dec_Number 0
line 15 column 13: Symbol ;
line 16 column 1: Symbol }
end
chen@chen:~/桌面/chen/e2$
```

图 5 Linux 系统下 lex2-2 实验结果

## 4 实验总结

### 4.1 你在编程过程中遇到了哪些难题？

Flex 的.l 文件语法规则不熟练。在完成多行注释时，起初想要使用正则表达式完全匹配符号/\*与符号\*/以及中间的任意字符，但是这样的编码思路总会与其他匹配规则存在冲突。最后参考了《ANSI C grammar (Lex).pdf》中多行注释的实现方案——使用正则匹配注释的开始符号/\*，匹配后的动作则使用 C 语法去逐一判断是否遇到了注释结束，同时这样的编码也使多行注释中行号的处理更灵活一些。

十进制浮点数以及科学计数法的正则表达式书写上遇到了很大的问题。原因是自己对正则表达式的语法及规则不是很熟悉。最后从一些正则表达式的生成网站上找到了相关表达式并进行少量修改。

### 4.2 你对你的程序的评价？

本次实验完成的语法分析器规则较为简单，且还未加入相关的语法分析。不过实现

了科学计数法、浮点数以及字符串的正则匹配相较于上一次实验明显进步了许多。未来还有更大的拓展空间。

#### 4.3 你的收获有哪些？

通过本次实验，我对正则表达式的语法规则更加了解，加深了对词法分析过程的理解。相比于之前编译技术课程中从零开始实现词法分析，借助 Flex 工具，可以通过更简单的语法生成词法分析程序，且具有良好的可拓展性。体会到了使用现代工具完成开发的简便之处。受益匪浅。