



中国矿业大学  
CHINA UNIVERSITY OF MINING AND TECHNOLOGY

# 中国矿业大学计算机学院 2017级本科生课程设计报告

## 实验二 Flex理论与练习2

课程名称	系统软件开发实践
报告时间	2020.02.19
学生姓名	李治远
学    号	07172757
专    业	计算机科学与技术
任课教师	席景科老师

# 目录

<b>1</b>	<b>实验目的</b>	<b>1</b>
<b>2</b>	<b>实验任务</b>	<b>1</b>
<b>3</b>	<b>实验代码</b>	<b>1</b>
3.1	lex2-1.1代码 . . . . .	1
3.2	lex2-2.1代码 . . . . .	2
<b>4</b>	<b>分析程序输出结果</b>	<b>4</b>
4.1	分析lex2-1.1实验结果 . . . . .	4
4.2	分析lex2-2.1实验结果 . . . . .	5
<b>5</b>	<b>在Windows环境下实验</b>	<b>6</b>
5.1	lex2-1.1实验结果 . . . . .	6
5.2	lex2-2.1实验结果 . . . . .	6
<b>6</b>	<b>在ubuntu环境下实验</b>	<b>7</b>
6.1	lex2-1.1实验结果 . . . . .	7
6.2	lex2-2.1实验结果 . . . . .	7
<b>7</b>	<b>实验感想</b>	<b>8</b>

## 实验二 Flex理论与练习1

### 1 实验目的

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。

### 2 实验任务

1. 编写Flex代码『lex2-1.1』，实现对上述C<sub>1</sub>语言的词法分析。要求：输出所有的关键字、专用符号、标识符、整型常数。
2. 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，并另存为『lex2-2.1』：
  - (a) 输出上述标记所在的行号；
  - (b) 忽略注释及其内容，如，注释中的数字/\*123\*/，//123；
  - (c) 增加科学记数法；
  - (d) 十六进制、八进制常数。

### 3 实验代码

#### 3.1 lex2-1.1代码

```
1  %{
2      int line = 1;
3  %}
4
5  KKEYWORDS (else|if|switch|for|int|float|return|void|while)
6  L  [a-zA-Z_]
7  D  [0-9]
8  ID  {L}({L}|{D})*
9  NUM  [1-9]{D}*
10
11  CHARACTER  (\+|\-|\*|\/|<|<=">|>=">|"=="|"!="|=|;|,|\(|\)|\[|\]|{\}|})
12
13  delim [ \t\n]
14  whitespace {delim}+
15
16  A  [/]
17  B  [*]
```

```

18    C [^*/]
19
20    NOTES1 "/*"(.|\n)*"/"
21    NOTES2 "//".*\n
22    NOTES ({NOTES1}|{NOTES2})
23    %%
24    \n {line++;}
25    {whitespace} {}
26    {NOTES} {/*注释*/ line++;}
27    {KWYWORS} {printf("%d\t%20s\t关键字\n", line, yytext);}
28    {CHARACTER} {printf("%d\t%20s\t专用符号\n", line, yytext);}
29
30    {ID} {printf("%d\t%20s\t标识符\n", line, yytext);}
31    {NUM} {printf("%d\t%20s\t整数\n", line, yytext);}
32    . {}
33    %%
34
35    void main(){
36        printf("Line\t%20s\tPs\n", "Actor");
37        printf("\n");
38        yylex();
39    }
40    int yywrap(){
41        return 1;
42    }

```

### 3.2 lex2-2.1代码

此代码基于实验要求1扩展分析。

- 增加专用符号集合；
- 增加关键字集合范围；
- 增加字符串、字符匹配；
- 增加数字类型匹配，整数、八进制、十六进制、科学计数法与浮点类型；

```

1    %{
2        int line = 1;
3    %}
4
5    /*浮点数指数部分*/
6    EXP ([Ee] [-+]?[0-9]+)
7    /*关键字*/

```

```

8      KKEYWORDS (void|signed|unsigned|short|long|int|float|double|char
9                  |enum|struct|union|typedef|const|volatile|auto|static|extern
10                 |register|sizeof|goto|return|break|continue|if|else|switch|case
11                 |default|do|while|for)
12
13      L [a-zA-Z_]
14      D [0-9]
15      ID {L}({L}|{D})*
16      /*整数*/
17      NUM 0|[1-9]{D}*
18      /*专用符号*/
19      CHARACTER (#|\+|-|\*|\/|<|\"<=\"|>|\">=\"|\"==\"
20                 |\"!=\"|=|;|,|\\(|\\)|\\[|\\]|\\{|\\}|!|'|\\\")
21      delim [ \\t\\n]
22      whitespace {delim}+
23      /*注释*/
24      A [/]
25      B [*]
26      C [^*/]
27      NOTES1 \"/*\"(.|\\n)*\"*/\"
28      NOTES2 \"//\".*\\n
29      NOTES ({NOTES1}|{NOTES2})
30      /*科学计数法*/
31      SCIENTIFIC [0-9]*\\.?[0-9]*{EXP}?
32      /*八进制*/
33      OCTAL 0[0-7]*
34      /*十六进制*/
35      HEX 0[Xx][0-9a-fA-F]+
36      /*字符串、字符*/
37      STRING \".*\"
38      CHAR \'.*\'
39
40      %%
41      \\n {line++;}
42      {whitespace} {}
43      {NOTES} {/*注释*/ line++;}
44      {NUM} {printf(\"%d\\t%20s\\t整数\\n\", line, yytext);}
45      {HEX} {printf(\"%d\\t%20s\\t十六进制\\n\", line, yytext);}
46      {OCTAL} {printf(\"%d\\t%20s\\t八进制\\n\", line, yytext);}
47      {SCIENTIFIC} {printf(\"%d\\t%20s\\t科学计数法\\n\", line, yytext);}
48      {STRING} {printf(\"%d\\t%20s\\t字符串\\n\", line, yytext);}
49      {CHAR} {printf(\"%d\\t%20s\\t字符\\n\", line, yytext);}

```

```

50     {KWYWORS} {printf("%d\t%20s\t关键字\n", line, yytext);}
51     {CHARACTER} {printf("%d\t%20s\t专用符号\n", line, yytext);}
52     {ID} {printf("%d\t%20s\t标识符\n", line, yytext);}
53     %%
54
55     void main(){
56         printf("Line\t%20s\tPs\n", "Actor");
57         printf("\n");
58         yylex();
59     }
60     int yywrap(){
61         return 1;
62     }

```

## 4 分析程序输出结果

### 4.1 分析lex2-1.1实验结果

2-1.cpp测试代码如下所示。

```

1     #include <iostream>
2     using namespace std
3     int main()
4     {
5         // comment1
6         cout<<"Hello! "<<endl;
7         /*
8         comment2
9         123 456 int
10        */
11        cout<<"Welcome to c++! "<<endl;
12        return 0;
13    }

```

第1行, #符号不在子集 $C_1$ 中, 不匹配, 不输出; include 匹配标识符, 输出; < 符号匹配专用符号, 输出; iostream 匹配标识符, 输出; > 符号匹配专用符号, 输出。

第2行, using 匹配标识符, 输出; namespace 匹配标识符, 输出; std 匹配标识符输出。

第3行, int 匹配关键字, 输出; main 匹配标识符, 输出; ( 匹配专用符号, 输出; ) 匹配专用符号, 输出。

第4行, { 匹配专用符号, 输出。

第5行, `cout` 匹配标识符, 输出; `<` 两次匹配专用符号, 输出; `"` 不在集合, 不匹配, 不输出; `Hello` 匹配标识符, 输出; `!` 不在集合, 不匹配, 不输出; `"` 不在集合, 不匹配, 不输出; `<` 两次匹配专用符号, 输出; `endl` 匹配标识符, 输出。; 匹配专用符号输出。

第6行, `cout` 匹配标识符, 输出; `<` 两次匹配专用符号, 输出; `"` 不在集合, 不匹配, 不输出; `Welcome` 匹配标识符, 输出; `to` 匹配标识符, 输出; `c` 匹配标识符, 输出; `+` 两次匹配专用符号, 输出; `!` 不在集合, 不匹配, 不输出; `"` 不在集合, 不匹配, 不输出; `<` 两次匹配专用符号, 输出; `endl` 匹配标识符, 输出。; 匹配专用符号输出。

第7行, `return` 匹配标识符, 输出; `0` 不匹配定义中的数字, 不输出; `;` 匹配专用符号, 输出。

第8行, `}` 匹配专用符号, 输出。

注: 此分析输出仅仅基于实验要求  $C_1$  集合, 是常规C语言的一个子集。所有的空格、换行符与注释均不匹配, 不输出。

## 4.2 分析lex2-2.1实验结果

```

1  #include <iostream>
2  using namespace std
3  int main(){
4      int a 123 023 0x23ff 34830.34E+4
5      char 'a'
6      // comment1
7      /*
8      comment2
9      123 456 int
10     */
11
12     cout<<"Welcome to c++! " endl;
13     return 0;
14 }
```

第1行, `#`匹配专用符号, 输出; `include` 匹配标识符, 输出; `<` 符号匹配专用符号, 输出; `iostream` 匹配标识符, 输出; `>` 符号匹配专用符号, 输出。

第2行, `using` 匹配标识符, 输出; `namespace` 匹配标识符, 输出; `std` 匹配标识符输出。

第3行, `int` 匹配关键字, 输出; `main` 匹配标识符, 输出; `(` 匹配专用符号, 输出; `)` 匹配专用符号, 输出; `{` 匹配专用符号, 输出。

第4行, `int` 匹配关键字, 输出; `a` 匹配标识符, 输出; `123` 匹配整数, 输出; `023` 匹配八进制输出; `0x23ff` 匹配十六进制, 输出; `34830.34E+4` 匹配科学计数, 输出。

第5行, char 匹配标识符, 输出; 'a' 匹配字符, 输出。

第6行, cout 匹配标识符, 输出; < 两次匹配专用符号, 输出; "Welcome to c++! " 匹配字符串, 输出; endl 匹配标识符, 输出。; 匹配专用符号输出。

第7行, return 匹配标识符, 输出; 0 匹配整数, 输出; ; 匹配专用符号, 输出。

第8行, } 匹配专用符号, 输出。

## 5 在Windows环境下实验

### 5.1 lex2-1.1实验结果

在Windows环境下lex2-1.1实验编译结果如图1所示, 实验结果如图3所示。

```
C:\Users\yuan\Documents\课程\系统软件开发实践\代码\ex2>flex lex2-1.1
C:\Users\yuan\Documents\课程\系统软件开发实践\代码\ex2>cl lex.yy.c
用于 x86 的 Microsoft (R) C/C++ 优化编译器 19.24.28316 版
版权所有 (C) Microsoft Corporation。保留所有权利。

lex.yy.c
Microsoft (R) Incremental Linker Version 14.24.28316.0
Copyright (C) Microsoft Corporation. All rights reserved.

/out:lex.yy.exe
lex.yy.obj
```

图 1: lex2-1.1编译结果

### 5.2 lex2-2.1实验结果

在Windows环境下lex2-2.1实验编译结果如图2所示, 实验结果如图4所示。

```
C:\Users\yuan\Documents\课程\系统软件开发实践\代码\ex2>flex lex2-2.1
C:\Users\yuan\Documents\课程\系统软件开发实践\代码\ex2>cl lex.yy.c
用于 x86 的 Microsoft (R) C/C++ 优化编译器 19.24.28316 版
版权所有 (C) Microsoft Corporation。保留所有权利。

lex.yy.c
lex.yy.c(1): warning C4819: 该文件包含不能在当前代码页 (936) 中表示的字符。请将该文件保存为
Microsoft (R) Incremental Linker Version 14.24.28316.0
Copyright (C) Microsoft Corporation. All rights reserved.

/out:lex.yy.exe
lex.yy.obj
```

图 2: lex2-2.1编译结果



Line	Actor	Ps
1	include	Identifier
1	<	Special symbol
1	iostream	Identifier
1	>	Special symbol
2	using	Identifier
2	namespace	Identifier
2	std	Identifier
3	int	Keyword
3	main	Identifier
3	(	Special symbol
3	)	Special symbol
4	{	Special symbol
5	cout	Identifier
5	<	Special symbol
5	<	Special symbol
5	Hello	Identifier
5	<	Special symbol
5	<	Special symbol
5	endl	Identifier
5	;	Special symbol
6	cout	Identifier
6	<	Special symbol
6	<	Special symbol
6	Welcome	Identifier
6	to	Identifier
6	c	Identifier
6	+	Special symbol
6	+	Special symbol
6	<	Special symbol
6	<	Special symbol
6	endl	Identifier
6	;	Special symbol
7	return	Keyword
7	;	Special symbol
8	}	Special symbol

图 3: lex2-1.1实验结果

Line	Actor	Ps
1	#	Special symbol
1	include	Identifier
1	<	Special symbol
1	iostream	Identifier
1	>	Special symbol
2	using	Identifier
2	namespace	Identifier
2	std	Identifier
3	int	Keyword
3	main	Identifier
3	(	Special symbol
3	)	Special symbol
3	{	Special symbol
4	int	Keyword
4	a	Identifier
4	123	Integer
4	023	Octal
4	0x23ff	Hex
4	34830.34E+4	Scientific notation
5	char	Keyword
5	'a'	Char
6	cout	Identifier
6	<	Special symbol
6	<	Special symbol
6	"Welcome to c++!"	String
6	endl	Identifier
6	;	Special symbol
7	return	Keyword
7	0	Integer
7	;	Special symbol
8	}	Special symbol

图 4: lex2-2.1实验结果

## 6 在ubuntu环境下实验

### 6.1 lex2-1.1实验结果

在ubuntu环境下lex2-1.1实验编译结果如图5所示，实验结果如图7所示。

```
yuan@ubuntu: ~/桌面/System S
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
yuan@ubuntu:~/桌面/System Software/ex2$ flex lex2-1.1
yuan@ubuntu:~/桌面/System Software/ex2$ cc lex.yy.c
```

图 5: lex2-1.1编译结果

### 6.2 lex2-2.1实验结果

在ubuntu环境下lex2-2.1实验编译结果如图6所示，实验结果如图8所示。

```
yuan@ubuntu: ~/桌面/Syste
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
yuan@ubuntu:~/桌面/System Software/ex2$ flex lex2-2.1
yuan@ubuntu:~/桌面/System Software/ex2$ cc lex.yy.c
```

图 6: lex2-2.1编译结果

```
yuan@ubuntu: ~/桌面/System Software/
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
yuan@ubuntu:~/桌面/System Software/ex2$ ./a.o
Line      Actor      Ps
1          include 标识符
1          <        专用符号
1          iostream 标识符
1          >        专用符号
2          using   标识符
2          namespace 标识符
2          std      标识符
3          int      关键字
3          main     标识符
3          (        专用符号
3          )        专用符号
4          {        专用符号
5          cout     标识符
5          <        专用符号
5          <        专用符号
5          Hello    标识符
5          <        专用符号
5          <        专用符号
5          endl     标识符
5          ;        专用符号
6          cout     标识符
6          <        专用符号
6          <        专用符号
6          Welcome  标识符
6          to       标识符
6          c        标识符
6          +        专用符号
6          +        专用符号
6          <        专用符号
6          <        专用符号
6          endl     标识符
6          ;        专用符号
7          return   关键字
7          ;        专用符号
8          }        专用符号
yuan@ubuntu:~/桌面/System Software/ex2$
```

图 7: lex2-1.1实验结果

```
yuan@ubuntu: ~/桌面/System Softw
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
Line      Actor      Ps
1          #        专用符号
1          include 标识符
1          <        专用符号
1          iostream 标识符
1          >        专用符号
2          using   标识符
2          namespace 标识符
2          std      标识符
3          int      关键字
3          main     标识符
3          (        专用符号
3          )        专用符号
3          {        专用符号
4          int      关键字
4          a        标识符
4          123       整数
4          023       八进制
4          0x23ff    十六进制
4          34830.34E+4 科学计数法
5          char     关键字
5          'a'      字符
6          cout     标识符
6          <        专用符号
6          <        专用符号
6          "Welcome to c++! " 字符串
6          endl     标识符
6          ;        专用符号
7          return   关键字
7          0        整数
7          ;        专用符号
8          }        专用符号
yuan@ubuntu:~/桌面/System Software/ex2$
```

图 8: lex2-2.1实验结果

7 实验感想

由于在编译原理课程中有过类似的实验，所以这次实验没有遇到较大的问题，只是遇到了编码错误问题，在 ubuntu 中编写好的代码在 windows 环境中，中文会出现乱码。在 windows 环境下重新编写，任然出现此问题，所以将所有中文输出改为英文，解决了此问题。

虽然我完成了这次实验，但是程序分析的效果还没有达到预期的效果，比如说注释没有算在代码行数中，这在实际的编译过程中是不可能，因为这样会造成出错位置的行数不对，无法快速找到错误的代码。

经过这次实验，我对正则表达式的书写更加清晰，在编写 flex 代码也更加熟练，对于 flex 如何分析代码程序也更加理解。