Lex 词法分析器分析各记号

一、实验目的：

熟悉 cygwin 环境的使用，学习使用 lex 写简单的词法分析程序，会在 cygwin 环境下使用 flex 调试 lex 写的程序。

二、实验内容：

读懂 exam1.l 和 exam2.l 两个例子，使用 cygwin 下的 flex 工具将 exam1.l 和 exam2.l 编译并调试通过。并且修改 exam2.l，在其基础上增加如下记号：

- 左右大小括号：{ } ( )
- 将关系算符改写成 C 中的形式
- 分号、赋值号：; =
- 关键字：if else
- 双斜线表示的注释：//
- 算术运算符号：+ - * /
- 将标识符改为可含有下划线，并且可以以下划线开头
- 将注释内容忽略

三、实验要求：

在 cygwin 下用 flex 和 gcc 工具将实验调试通过，并写出测试例测试正确性。

四、具体实现

对 exam2.l 文件修改如下：

1. 新增声明：

```
%{

#include <stdio.h>
#define LT          1
#define LE          2
#define GT          3
#define GE          4
#define EQ          5
#define NE          6

#define IF          16
#define ELSE        17
#define WHILE       18
#define DO          19
#define ID          20
#define NUMBER      21
#define RELOP       22

#define NEWLINE     23
#define ERRORCHAR   24

#define COPY        25
#define SEMICOLON   26
#define BRACKET     27
#define OP          28

%}
```

新增定义识别 if 和 else 的记号名分别对应其大写英文 IF 和 ELSE

- COPY 是赋值符号“=”的记号名；
- SEMICOLON 是分号“;”的记号名；
- BRACKET 是括号的记号名，包括左右大小括号：”(” ”)” ”{” ”}”；
- OP 是运算符号的记号名，包括加减乘除：”+” ”-” ”*” ”/”

2. 正规定义的修改：

```
delim    [ \t \n]
ws       {delim}+
letter_  [A-Za-z_]
digit    [0-9]
id       {letter_}({letter_}|{digit})*
number   {digit}+(\.{digit}+)?(E[+-]?{digit}+)?
```

将下划线”_”加入 letter 的正规定义，使得标识符可含有下划线，并且可以以下划线开头。为了区分，给 letter 重新命名为 letter_，后面的使用到 letter 的正规定义跟随修改。

3. 对注释的处理：（增加识别双斜杠注释和忽略注释内容）

```
%s COMMENT1
%s COMMENT2

%%

<INITIAL>"/*"          {BEGIN COMMENT1;}
<COMMENT1>"*/"         {BEGIN INITIAL;}
<COMMENT1>.|\n         {;}

<INITIAL>"//"          {BEGIN COMMENT2;}
<COMMENT2>\n           {BEGIN INITIAL;}
<COMMENT2>.|\n         {;}
```

- 双斜杠注释以双斜杠”//”开头，回车换行符\n 结尾。
- ECHO 是一个宏，相当于 fprintf(yyout, "%s", yytext)，所以要忽略注释只需要把原文件中的 ECHO 去掉即可，这样就不会输出注释内容。

4. 翻译规则的修改：

```
63  <INITIAL>"<"            {return (RELOP);}
64  <INITIAL>"<="           {return (RELOP);}
65  <INITIAL>"=="           {return (RELOP);}
66  <INITIAL>"!="           {return (RELOP);}
67  <INITIAL>">"            {return (RELOP);}
68  <INITIAL>">="           {return (RELOP);}
69
70  <INITIAL>"="            {return (COPY);}
71  <INITIAL>";"            {return (SEMICOLON);}
72
73  <INITIAL>"("            {return (BRACKET);}
74  <INITIAL>")"            {return (BRACKET);}
75  <INITIAL>"{"            {return (BRACKET);}
76  <INITIAL>"}"            {return (BRACKET);}
77
78  <INITIAL>"+"            {return (OP);}
79  <INITIAL>"-"            {return (OP);}
80  <INITIAL>"*"            {return (OP);}
81  <INITIAL>"/"            {return (OP);}
```

- C 语言中的不等于是”!=”
- 而”=”是赋值符号，”==”才是等于符号。

新增对赋值符号、分号、大小左右括号、运算符号的翻译规则。

5. 输出函数的修改：

```
92  void writeout(int c){
93    switch(c){
94      case ERRORCHAR: fprintf(yyout, "(ERRORCHAR, \"%s\") ", yytext);break;
95      case RELOP: fprintf(yyout, "(RELOP, \"%s\") ", yytext);break;
96      case WHILE: fprintf(yyout, "(WHILE, \"%s\") ", yytext);break;
97      case DO: fprintf(yyout, "(DO, \"%s\") ", yytext);break;
98      case IF: fprintf(yyout, "(IF, \"%s\") ", yytext);break;
99      case ELSE: fprintf(yyout, "(ELSE, \"%s\") ", yytext);break;
100     case NUMBER: fprintf(yyout, "(NUM, \"%s\") ", yytext);break;
101     case ID: fprintf(yyout, "(ID, \"%s\") ", yytext);break;
102     case NEWLINE: fprintf(yyout, "\n");break;
103     case COPY: fprintf(yyout, "(COPY, \"%s\") ", yytext);break;
104     case SEMICOLON: fprintf(yyout, "(SEMICOLON, \"%s\") ", yytext);break;
105     case BRACKET: fprintf(yyout, "(BRACKET, \"%s\") ", yytext);break;
106     case OP: fprintf(yyout, "(OP, \"%s\") ", yytext);break;
107     default:break;
108   }
109   return;
110 }
```

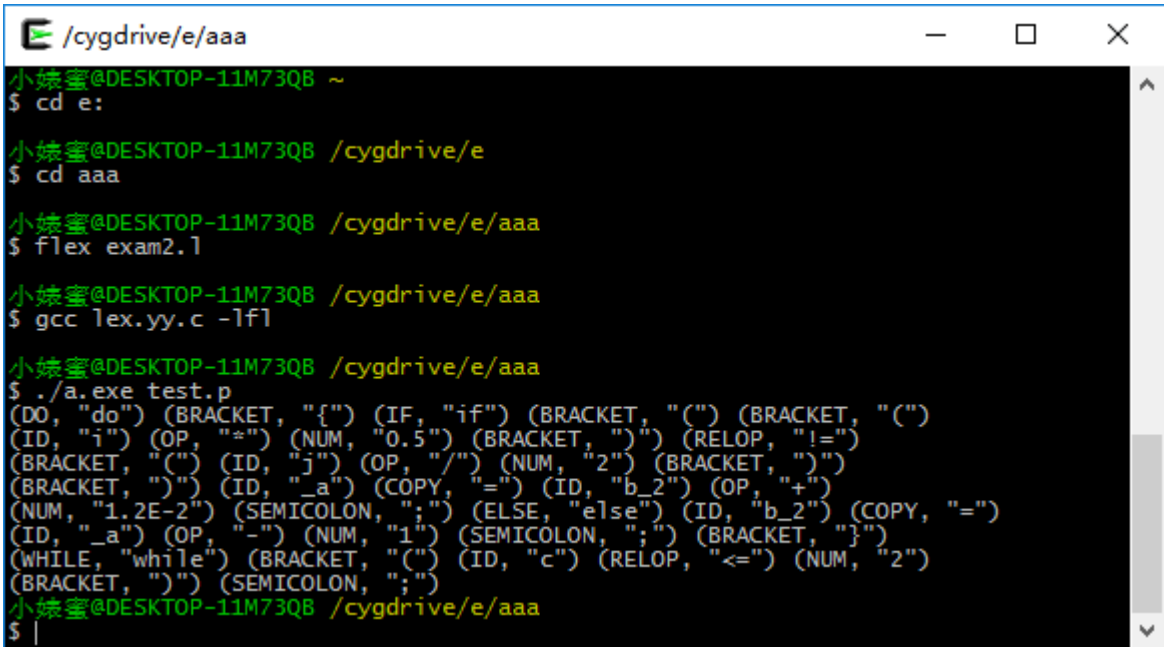
6. 测试文件 test.p 内容如下：

```
1
2  //这是第一条注释，双斜杠的
3
4  do {
5    if((i * 0.5) != (j / 2))
6      _a = b_2 + 1.2E-2;
7    else
8      b_2 = _a - 1;
9  }while (c <= 2);
10
11 /*
12 这是
13 第二条注释
14 斜杠星号的
15 */
```

测试文件中包含：

- ✓ do, while, if, else 关键字；
- ✓ ( ) { } 大小左右括号；
- ✓ + - * / 运算符号；
- ✓ 整数、浮点数、科学表示法数字；
- ✓ “!=”, “<=”比较运算符；
- ✓ 分号”;”和赋值”=”符号；
- ✓ 普通标识符，下划线开头标识符，含有下划线的标识符；
- ✓ 两种注释方法；

7. 在 cygwin 下用 flex 编译器对新的 test2.l 进行编译，生成 lex.yy.c 文件，用 gcc 编译器对该 C 源程序进行编译得到 a.exe 文件，用测试文件 test.p 进行测试，如图所示：



经核对，结果正确。

五、 心得与体会

1. 基本了解和掌握了 flex 词法分析器生成工具的使用，lex 的语法规则和组织方式。熟悉了 cygwin 环境的使用，并能在 cygwin 下使用 flex 编译器调试 lex 程序，利用 gcc 编译器调试 lex.yy.c 文件，以及执行测试文件（.p）。
2. 状态的定义和使用，有时候通过状态可有效的解决一些复杂问题，比如对注释的操作等。不同的状态之间可以使用“BEGIN”进行切换，在不同的状态中对同一正规式表示的句子可以执行不同的动作。
3. 动作可以是一个空语句，相当于过滤掉某些输入。
4. 动作中有返回语句时，返回值应当事先在定义段的第一部分用 C 语言的语法进行定义，以便于后面的操作和使用。

5. 在词法规则段，lex 总是尽可能匹配较长的句子。当发生冲突时，在词法规则段中首先出现的正规式将被匹配。
6. 掌握了一些具体的 lex 元字符的使用方法，可以简化正规定义。
7. 当想要将元字符作为正文字符使用时，可以使用转义字符\或“”。
8. 理解了用 C 语言写的辅助函数对于词法分析的重要作用。这些辅助函数一方面可以作为语义动作的一部分，另一方面可以为词法分析提供依据。而且可以将 main 函数作为辅助函数放在这里，当单独使用词法分析器的时候可以在 main 函数里实现对词法分析器的调度和使用。