2023

编译原理实验报告

词法分析程序设计与实现

班级：2021211307　　姓名：陈朴炎　　学号：2021211138

目录

1 概述

1.1 问题描述

设计并实验一个词法分析程序，要求实现如下功能。

（1）可以识别出用 C 语言编写的源程序中的每个单词符号,并以记号的形式输出每个单词符号。

（2）可以识别并跳过源程序中的注释。

（3）可以统计源程序中的语句行数、各类单词的个数、以及字符总数，并输出统计结果。

（4）检查源程序中存在的词法错误,并报告错误所在的位置。

（5）对源程序中出现的错误进行适当的恢复，使词法分析可以继续进行,对源程序进行一次扫描,即可检查并报告源程序中存在的所有词法错误。

1.2 实现方法

方法 1：采用 C/C++作为实现语言，手工编写词法分析程序。

方法 2：编写 LEX 源程序，利用 LEX 编译程序自动生成词法分析程序。

两种方法选择一种，本次实验我选择的是第二种，编写一个 lex 源程序，并让 lex 自动生成词法分析程序。

2 实验环境

2.1 lex 与 yacc 安装，gcc 安装

对于 Unix 和 Linux 来说，这些都是标配，不需要额外配置，而对于 Windows 来说，我们需要额外配置环境。我们需要使用 flex 来代替 lex，用 bison 来代替 yacc，这两者完全可以提供我们需要的功能， flex 还是 lex 的加强版，它们可以在 windows 上运行，且是免费的。此外，我们还需要使用 gcc 来将 flex 与 bison 翻译成的 c 文件编译为可执行的 exe 文件。

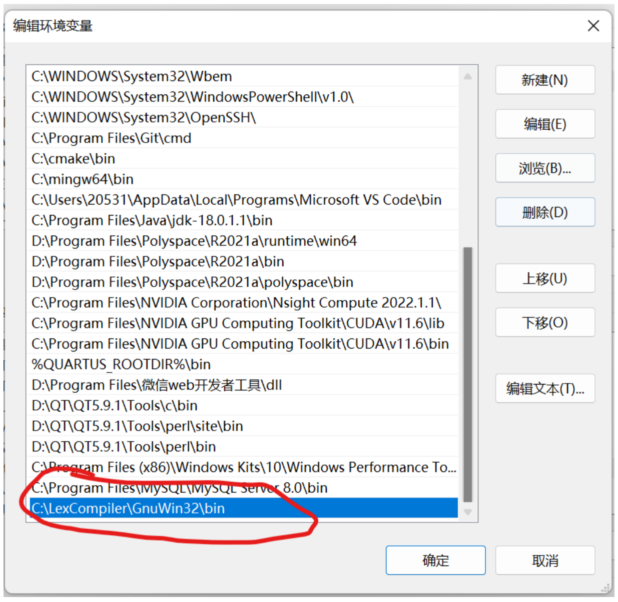
从 GCC 官网 http://www.mingw.org 获取 gcc。

从 https://gnuwin32.sourceforge.net/packages/bison.htm 获取 bison。

从 https://gnuwin32.sourceforge.net/packages/flex.htm 获取 flwx。

2.2 配置环境变量

在获取完这些软件后并不是就可以开始写程序了。在 Windows 中，我们需要在 cmd 中运行这些编译程序,因此我们需要将上述三个软件配置到我们系统环境变量 Path 中。比如说我找到了下载好的 bison.exe 文件和下载好的 flex.exe 文件，它们都在 C:\LexCompiler\GnuWin32\bin 里面，我就要将这个路径放到系统环境变量中，如图 2-1。



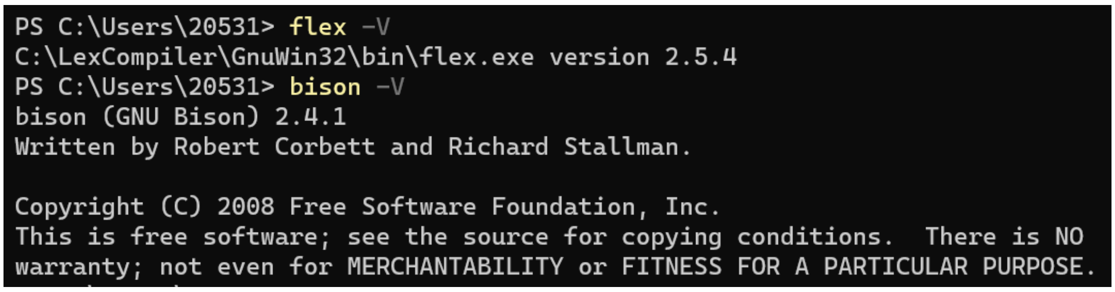
```
C:\WINDOWS\System32\Wbem
C:\WINDOWS\System32\WindowsPowerShell\v1.0\
C:\WINDOWS\System32\OpenSSH\
C:\Program Files\Git\cmd
C:\cmake\bin
C:\mingw64\bin
C:\Users\20531\AppData\Local\Programs\Microsoft VS Code\bin
C:\Program Files\Java\jdk-18.0.1.1\bin
D:\Program Files\Polyspace\R2021a\runtime\win64
D:\Program Files\Polyspace\R2021a\bin
D:\Program Files\Polyspace\R2021a\polyspace\bin
C:\Program Files\NVIDIA Corporation\Nsight Compute 2022.1.1\
C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.6\lib
C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.6\bin
%QUARTUS_ROOTDIR%\bin
D:\Program Files\微信web开发者工具\dll
D:\QT\QT5.9.1\Tools\c\bin
D:\QT\QT5.9.1\Tools\perl\site\bin
D:\QT\QT5.9.1\Tools\perl\bin
C:\Program Files (x86)\Windows Kits\10\Windows Performance To...
C:\Program Files\MySQL\MySQL Server 8.0\bin
C:\LexCompiler\GnuWin32\bin
```

图 2-1 添加路径到系统 Path

之后点击确定。同理，也要将包含有 gcc.exe 的路径配置到环境变量中，通常来说都是添加/bin 目录。

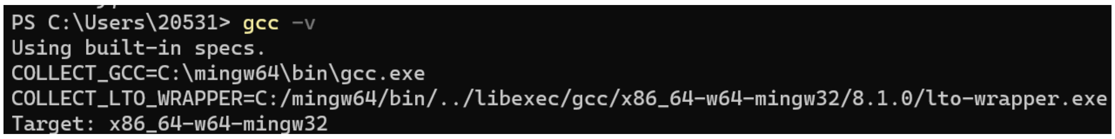
2.3 检测是否安装配置成功

在 Windows 下，以管理员权限打开 cmd，输入 flex -V 检查 flex 版本，输入 bison -V 检查 bison 版本，输入 gcc -v 检查 gcc 版本，以此来检查是否安装成果。如图 2-2、2-3 所示，其中 flex 和 bison 后面跟的是大写的 V，gcc 后面跟的是小写的 v。



```
PS C:\Users\20531> flex -V
C:\LexCompiler\GnuWin32\bin\flex.exe version 2.5.4
PS C:\Users\20531> bison -V
bison (GNU Bison) 2.4.1
Written by Robert Corbett and Richard Stallman.

Copyright (C) 2008 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

图 2-2 检查 flex、bison



```
PS C:\Users\20531> gcc -v
Using built-in specs.
COLLECT_GCC=C:\mingw64\bin\gcc.exe
COLLECT_LTO_WRAPPER=C:/mingw64/bin/../libexec/gcc/x86_64-w64-mingw32/8.1.0/lto-wrapper.exe
Target: x86_64-w64-mingw32
```

图 2-3 检查 gcc

到此为止，环境配置就完成了，接下来就是真正的实验了。

3 文法及状态转换图

3.1 语言说明

C 语言中有一下记号和单词：

1、标识符：以字母或下划线开头的、后跟字母或数字组成的符号串

2、关键字：标识符集合的子集，C 语言定义的关键字有 32 个，分别是：auto|break|case|char|const|continue|default|do|double|else|enum|extern|float|for|goto|if|inline|int|long|return|short|signed|sizeof|static|struct|switch|typedef|union|unsigned|void|volatile|while

3、数字：C 语言中的数字可以是整数、浮点数，也可以是指数形式。

4、关系运算符：关系运算符有：<、<=、==、>=、>、!= 、||、&&、!等。

5、算数运算符有：+、-、*、/、++、--、%、>>、<<等。

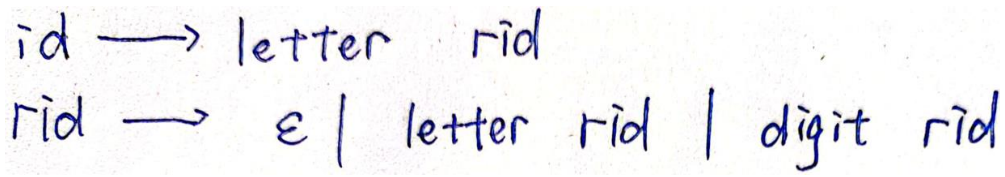
6、标点符号：（ ） [ ] { } : , ; . 等

7、赋值符号：=、+=、-=、*=、/=、%=、>>=、<<=等

8、注释标记：以/*开始，以*/结尾。或者以//开始，以换行符\n 结尾

3.2 记号的正规文法

1、标识符的文法

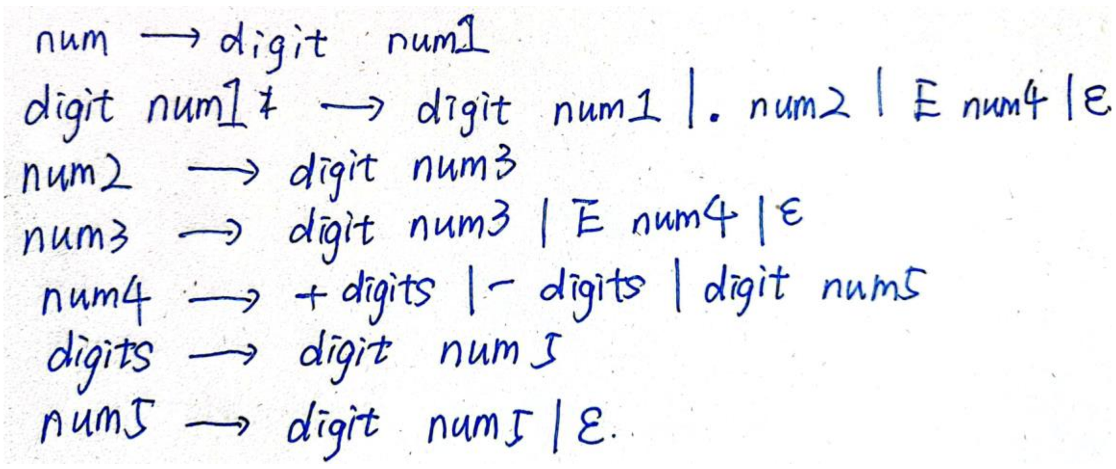


id → letter rid

rid → ε | letter rid | digit rid

图 3-1 标识符的文法产生式

2、数字的文法



num → digit num1

digit num1 ~~7~~ → digit num1 | . num2 | E num4 | ε

num2 → digit num3

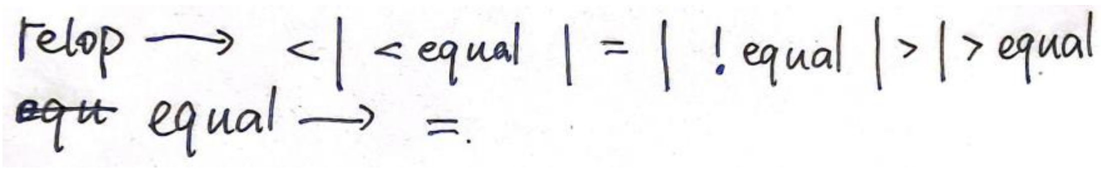
num3 → digit num3 | E num4 | ε

num4 → + digits | - digits | digit num5

digits → digit num5

num5 → digit num5 | ε.

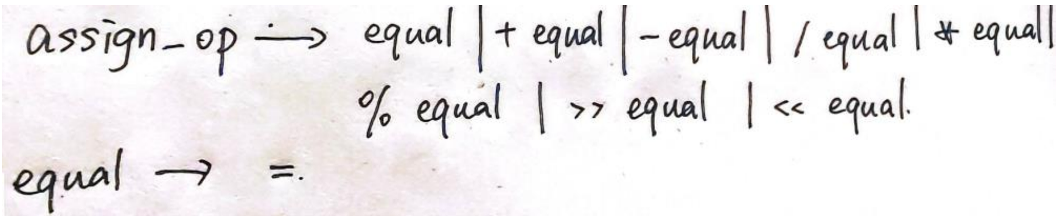
图 3-2 数字的文法产生式



relop → < | < equal | = | ! equal | > | > equal

~~equ~~ equal → =.

图 3-3 关系运算符的文法产生式

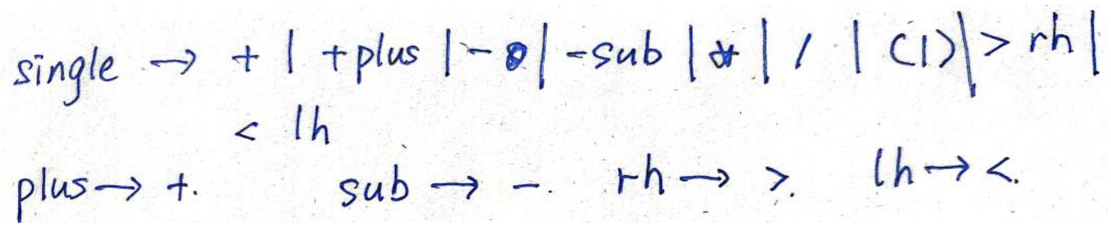


assign_op → equal | + equal | - equal | / equal | * equal |

% equal | >> equal | << equal.

equal → =.

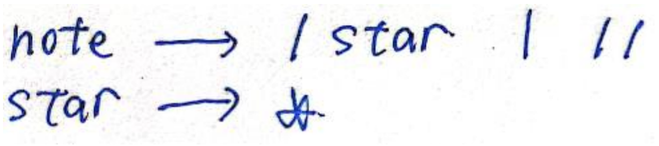
图 3-4 赋值运算符的文法产生式



single → + | +plus | -0 | -sub | * | / | (1) | > rh | < lh

plus → +. sub → -. rh → >. lh → <.

图 3-5 算术运算符的文法产生式



note → / star | //

star → *

图 3-6 注释的文法产生式

3.3 状态转换图

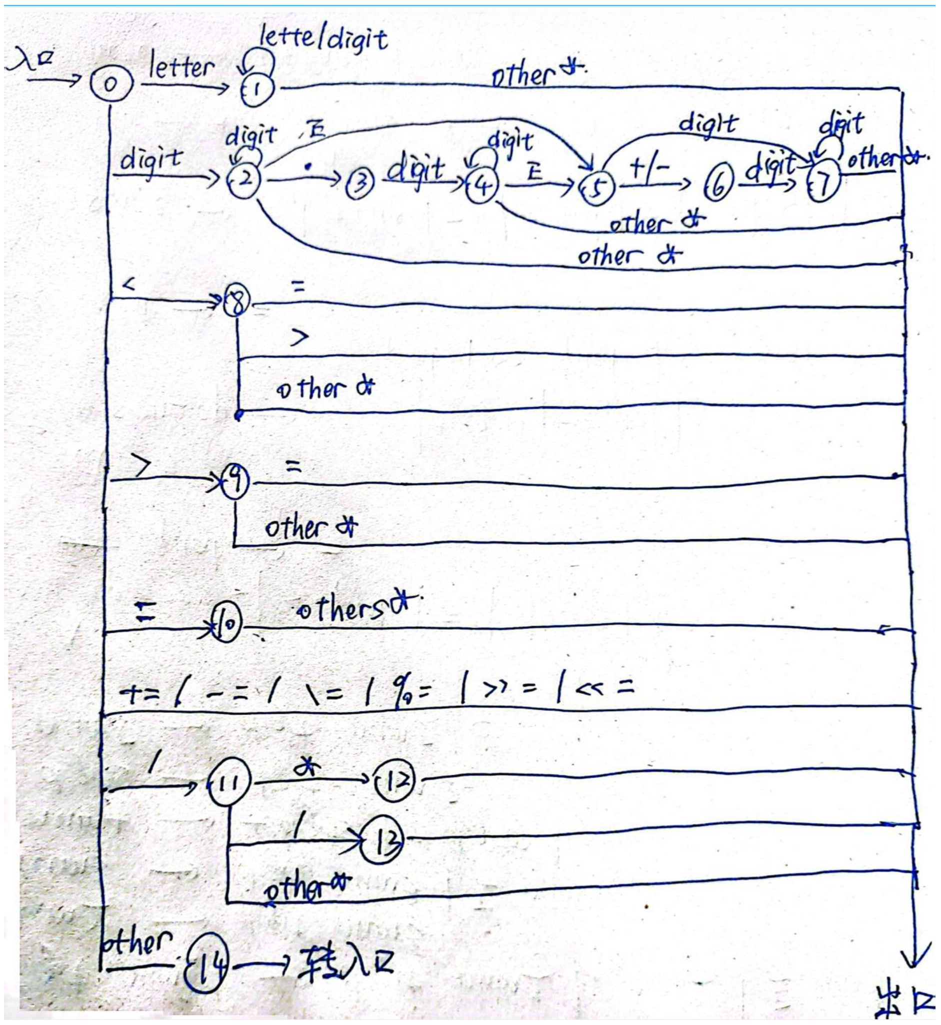


图 3-7 状态转换图

其中，状态 0 是初始状态，若此时读入的字符串是字母，则转到状态 1，进入标识符识别过程；如果读入的字符是数字，则转换到状态 2，进入数字识别过程；如果读到的字符是< 或 > 或 ! 则转入状态 8、9，识别关系运算符……若

读入的字符是/，转换到 11，再读入下一个字符，如果读入的是*，则转换到状态 12，若读入的是//，则转换到状态 13。如果是其他的接受不了的字符，则转到错误处理状态。

4 程序设计

4.1 定义输出文件中的全局变量及函数

```
%{
#include<math.h>
#include<stdlib.h>
#include<stdio.h>

void jumpMultiComment(void);
void jumpSingleComment(void);
void printCompilerResult(void);
void addCharNum(int length);
int lineCount = 1;
int keywordNum = 0;
int idNum = 0;
int errorNum = 0;
int intNum = 0;
int floatNum = 0;
int charNum = 0;
%}
```

全局变量说明：

1、lineCount 用于记录 C 语言源程序的行数
2、keywordNum 用于记录 C 语言源程序的关键字个数
3、idNum 用于记录 C 语言源程序的标识符个数
4、errorNum 用于记录 C 语言源程序的语法错误个数
5、intNum 用于记录 C 语言源程序的整型个数
6、floatNum 用于记录 C 语言源程序的浮点型个数
7、charNum 用于记录 C 语言源程序的所有字符的总个数

函数说明：

1、jumpMultiComment 函数用来跳过以/**/包裹起来的多行注释，在函数中，会记录每个字符个数，以及行数，并添加到 charNum 和 lineCount 中。
2、jumpSingleComment 函数用来跳过以//开头的单行注释，并会将注释中的所有字符个数添加到 charNum 中，，并将 lineCount+1
3、printCompilerResult 函数用来输出词法分析程序的执行结果，它将打印出源程序的行数、关键字个数、标识符个数、异常个数、整型个数、浮点型个数、所有的字符个数
4、addCharNum 函数用来将读取到的字符串里的字符个数添加到 charNum 中

4.2 定义数字、标识符及关键字

```
DIGIT [0-9]
ID [a-zA-Z_][a-zA-Z0-9_]*
KEYWORD
("auto"|"break"|"case"|"char"|"const"|"continue"|"default"|"do"|"doub
le"|"else"|"enum"|"extern"|"float"|"for"|"goto"|"if"|"inline"|"int"|"
long"|"return"|"short"|"signed"|"sizeof"|"static"|"struct"|"switch"|"
typedef"|"union"|"unsigned"|"void"|"volatile"|"while")
```

4.3 定义翻译规则

```
%%
"/*"            {
    charNum+=2;
    jumpMultiComment();
}
"//"      {
    charNum += 2;
    jumpSingleComment();
}
{DIGIT}+|{DIGIT}+[eE][+-]{DIGIT}+             {
    printf("IntNum:    %s\n",yytext);
    intNum++;
    addCharNum(yyleng);
}
({DIGIT}+[.]{DIGIT}+)|({DIGIT}+[.]{DIGIT}+[eE][+-]{DIGIT}+)       {
    printf("FloatNum:    %s\n",yytext);
    floatNum++;
    addCharNum(yyleng);
}
{KEYWORD}     {
    printf("KEYWORD: %s\n", yytext);
    keywordNum++;
    addCharNum(yyleng);
}
{ID}             {
    printf("Identifier: %s\n",yytext);
    idNum++;
    addCharNum(yyleng);
}
[a-zA-Z_]?\"(\\.|[^\\"\n])*\"     { printf("const_string: %s\n",yytext); addCharNum(yyleng);}
[a-zA-Z_]?'(\\.|[^\\'\n])+' { printf("const_char: %s\n",yytext); addCharNum(yyleng);}
```

```
">>="            { printf("RIGHT_ASSIGN: %s\n",yytext); addCharNum(yyleng);}
"<<="            { printf("LEFT_ASSIGN: %s\n",yytext); addCharNum(yyleng);}
"+="             { printf("ADD_ASSIGN: %s\n",yytext); addCharNum(yyleng);}
"-="             { printf("SUB_ASSIGN: %s\n",yytext); addCharNum(yyleng);}
"*="             { printf("MUL_ASSIGN: %s\n",yytext); addCharNum(yyleng);}
"/="             { printf("DIV_ASSIGN: %s\n",yytext); addCharNum(yyleng);}
"%="             { printf("MOD_ASSIGN: %s\n",yytext); addCharNum(yyleng);}
"&="             { printf("AND_ASSIGN: %s\n",yytext); addCharNum(yyleng);}
"^="             { printf("XOR_ASSIGN: %s\n",yytext); addCharNum(yyleng);}
"|="             { printf("OR_ASSIGN: %s\n",yytext); addCharNum(yyleng);}
">>"             { printf("RIGHT_OP: %s\n",yytext); addCharNum(yyleng);}
"<<"             { printf("LEFT_OP: %s\n",yytext); addCharNum(yyleng);}
"++"             { printf("INC_OP: %s\n",yytext); addCharNum(yyleng);}
"--"             { printf("DEC_OP: %s\n",yytext); addCharNum(yyleng);}
"->"             { printf("PTR_OP: %s\n",yytext); addCharNum(yyleng);}
"&&"             { printf("AND_OP: %s\n",yytext); addCharNum(yyleng);}
"||"             { printf("OR_OP: %s\n",yytext); addCharNum(yyleng);}
"<="             { printf("LE_OP: %s\n",yytext); addCharNum(yyleng);}
">="             { printf("GE_OP: %s\n",yytext); addCharNum(yyleng);}
"=>"             { printf("ARROW: %s\n",yytext); addCharNum(yyleng);}
"=="             { printf("EQ_OP: %s\n",yytext); addCharNum(yyleng);}
"!="             { printf("NE_OP: %s\n",yytext); addCharNum(yyleng);}
";"         { printf("'SEMICOLON': %s\n",yytext); addCharNum(yyleng);}
("{"|"<%")       { printf("'L_BRACE': %s\n",yytext); addCharNum(yyleng);}
("}"|"%>")       { printf("'R_BRACE': %s\n",yytext); addCharNum(yyleng);}
","         { printf("COMMA: %s\n",yytext); addCharNum(yyleng);}
":"         { printf("COLON: %s\n",yytext); addCharNum(yyleng);}
"="         { printf("ASSIGN: %s\n",yytext); addCharNum(yyleng);}
"("         { printf("L_PAREN: %s\n",yytext); addCharNum(yyleng);}
")"         { printf("R_PAREN: %s\n",yytext); addCharNum(yyleng);}
("["|"<:")      { printf("L_SQUARE: %s\n",yytext); addCharNum(yyleng);}
("]"|":>")      { printf("R_SQUARE: %s\n",yytext); addCharNum(yyleng);}
"."         { printf("DOT: %s\n",yytext); addCharNum(yyleng);}
"&"         { printf("BIT_AND: %s\n",yytext); addCharNum(yyleng);}
"!"         { printf("NOT: %s\n",yytext); addCharNum(yyleng);}
"~"         { printf("BIT_NOT: %s\n",yytext); addCharNum(yyleng);}
"-"         { printf("SUB: %s\n",yytext); addCharNum(yyleng);}
"+"         { printf("ADD: %s\n",yytext); addCharNum(yyleng);}
"*"         { printf("STAR: %s\n",yytext); addCharNum(yyleng);}
"/"         { printf("DIV: %s\n",yytext); addCharNum(yyleng);}
"%"         { printf("MOD: %s\n",yytext); addCharNum(yyleng);}
```

```
"<"          { printf("LESS: %s\n",yytext); addCharNum(yyleng);}
">"          { printf("GREATER: %s\n",yytext); addCharNum(yyleng);}
"^"          { printf("BIT_XOR: %s\n",yytext); addCharNum(yyleng);}
"|"          { printf("BIT_OR: %s\n",yytext); addCharNum(yyleng);}
"?"          { printf("QUESTION: %s\n",yytext); addCharNum(yyleng);}
"#"          { printf("WELL: %s\n", yytext); addCharNum(yyleng);}
"\n"         { lineCount++; addCharNum(yyleng);}
[ \t\v\f]    { addCharNum(yyleng);}
.                    {
    printf("ERROR: line %d, %s\n",lineCount,yytext);
    errorNum++;
    addCharNum(yyleng);
}
%%
```

对于每个匹配到的字符，都会打印出相应的记号，如“<”小于号就会打印出 LESS：< 并且，会将字符串长度添加到 charNum 中，保证总字符数能更新。

对于没有匹配到的字符串，会进行相应的进行 error 操作，打印出出错的字符串，并将 error 数添加，再添加字符总数。

4.4 辅助函数定义

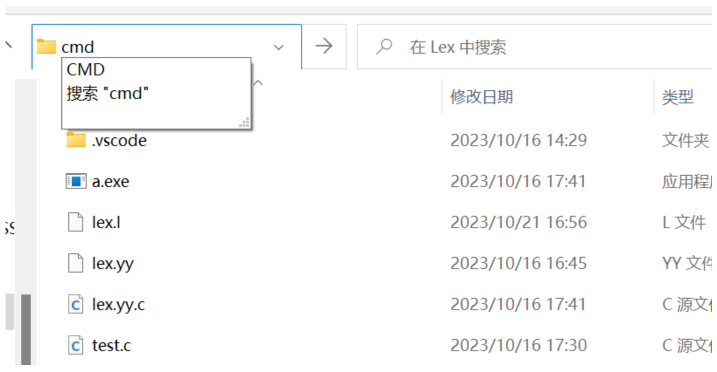
```
%%

int main(int argc,char **argv){
    if(argc>1) yyin=fopen(argv[1],"r");
    else printf("error:\n command: lexC filename");
    yylex();
    return 0;
}
int yywrap(){
    printCompilerResult();
    return 1;
}

void jumpMultiComment(void){
    char c, prev = 0;

    while ((c = input()) != 0)      /* (EOF maps to 0) */
    {
        if(c == '\n'){
            lineCount++;
        }
        if (c == '/' && prev == '*')
```

```
            return;
        prev = c;
    }
}
void jumpSingleComment(void){
    char c;
    while((c = input()) != 0){
        charNum++;
        if(c == '\n'){
            lineCount++;
            return;
        }
    }
}
void printCompilerResult(void){
    printf("===================================================\n");
    printf("         Compiler finished, the result are:        \n");
    printf("All char number: %d\n", charNum);
    printf("Line count: %d\n", lineCount);
    printf("Ketword number: %d\n", keywordNum);
    printf("Identifier number: %d\n", idNum);
    printf("Integer number: %d\n", intNum);
    printf("Floate number: %d\n", floatNum);
    printf("Error number: %d\n", errorNum);
}
void addCharNum(int length){
    charNum += length;
}
```

辅助函数的功能大部分都在 4.1 中介绍了。

5 编译过程

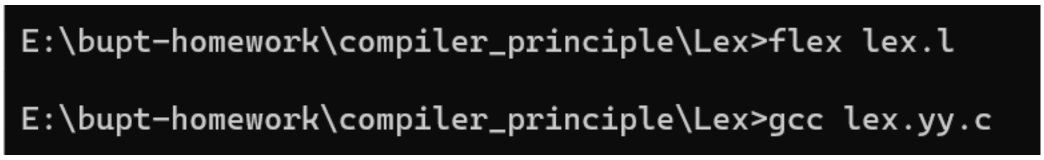
写好 lex 程序，在项目文件夹下打开 cmd，如图 5-1.



```
cmd
CMD
搜索 "cmd"
在 Lex 中搜索

              修改日期            类型
.vscode       2023/10/16 14:29   文件夹
a.exe         2023/10/16 17:41   应用程
lex.l         2023/10/21 16:56   L 文件
lex.yy        2023/10/16 16:45   YY 文件
lex.yy.c      2023/10/16 17:41   C 源文
test.c        2023/10/16 17:30   C 源文
```

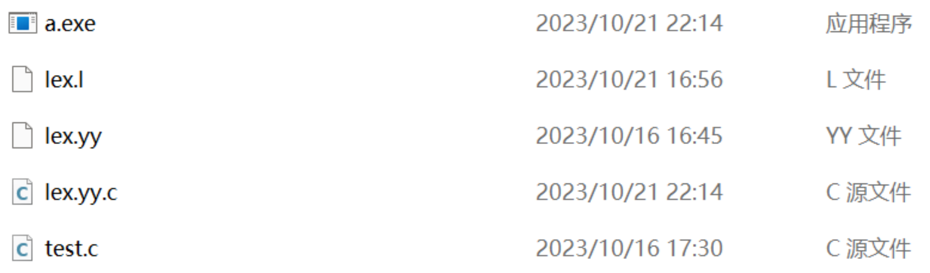
图 5-1 打开 cmd

如图5-2，输入flex <lex文件名> 回车，等待文件生成，再输入gcc lex.yy.c，等待可执行文件生成



```
E:\bupt-homework\compiler_principle\Lex>flex lex.l

E:\bupt-homework\compiler_principle\Lex>gcc lex.yy.c
```

图 5-2 flex 执行命令

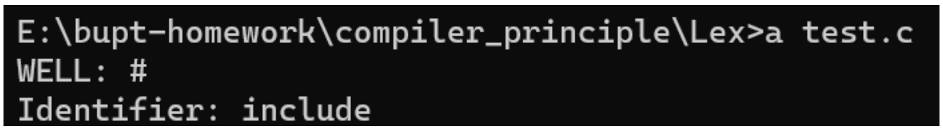


a.exe	2023/10/21 22:14	应用程序
lex.l	2023/10/21 16:56	L 文件
lex.yy	2023/10/16 16:45	YY 文件
lex.yy.c	2023/10/21 22:14	C 源文件
test.c	2023/10/16 17:30	C 源文件

图 5-3 生成结果

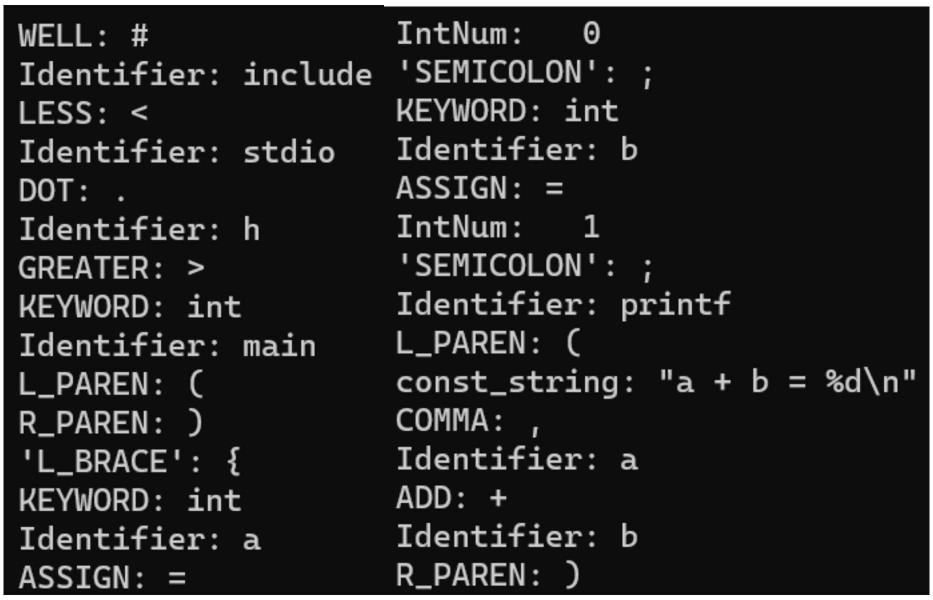
之后生成出 lex.yy.c 以及 a.exe 文件，其中 a.exe 是可执行文件。但是 a.exe 文件并不能直接打开，而是需要在命令行中输入 a 或者 a.exe 回车，才能执行文件。

我再实现时，将命令修改为了 a <测试文件名>，这样在打开可执行文件时就能一起将测试文件传入进去了。



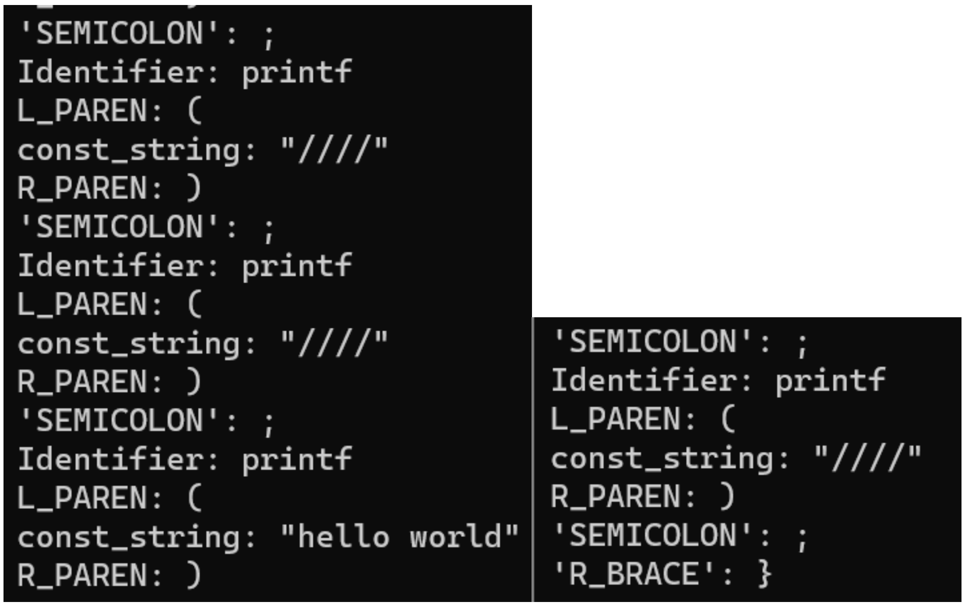
```
E:\bupt-homework\compiler_principle\Lex>a test.c
WELL: #
Identifier: include
```

图 5-4 执行用例



```
WELL: #                 IntNum:   0
Identifier: include     'SEMICOLON': ;
LESS: <                 KEYWORD: int
Identifier: stdio       Identifier: b
DOT: .                  ASSIGN: =
Identifier: h           IntNum:   1
GREATER: >              'SEMICOLON': ;
KEYWORD: int            Identifier: printf
Identifier: main        L_PAREN: (
L_PAREN: (              const_string: "a + b = %d\n"
R_PAREN: )              COMMA: ,
'L_BRACE': {            Identifier: a
KEYWORD: int            ADD: +
Identifier: a           Identifier: b
ASSIGN: =               R_PAREN: )
```

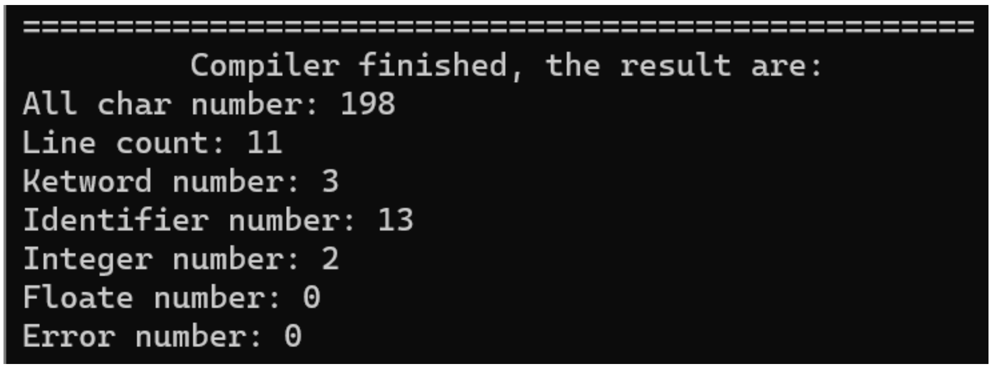
图 5-5 记号输出结果 1



```
'SEMICOLON': ;
Identifier: printf
L_PAREN: (
const_string: "////"
R_PAREN: )
'SEMICOLON': ;
Identifier: printf
L_PAREN: (
const_string: "////"
R_PAREN: )
'SEMICOLON': ;
Identifier: printf
L_PAREN: (
const_string: "hello world"
R_PAREN: )
```

```
'SEMICOLON': ;
Identifier: printf
L_PAREN: (
const_string: "////"
R_PAREN: )
'SEMICOLON': ;
'R_BRACE': }
```

图 5-6 记号输出结果 2



```
==================================================
         Compiler finished, the result are:
All char number: 198
Line count: 11
Ketword number: 3
Identifier number: 13
Integer number: 2
Floate number: 0
Error number: 0
```

图 5-7 执行分析结果

6 测试用例及结果

6.1 测试用例 1

```
#include <stdio.h>
int main(){
    int a = 0;
    int b = 1;
    printf("a + b = %d\n", a+b);
    printf("////");
    printf("////");
    printf("hello world"); // hello world
    printf("////");

}
```

执行结果：

```
E:\bupt-homework\compiler_principle\Lex>a test.c
```

```
WELL: #
Identifier: include
LESS: <
Identifier: stdio
DOT: .
Identifier: h
GREATER: >
KEYWORD: int
Identifier: main
L_PAREN: (
R_PAREN: )
'L_BRACE': {
KEYWORD: int
Identifier: a
ASSIGN: =
IntNum:   0
'SEMICOLON': ;
KEYWORD: int
Identifier: b
ASSIGN: =
IntNum:   1
'SEMICOLON': ;
Identifier: printf
L_PAREN: (
const_string: "a + b = %d\n"
COMMA: ,
Identifier: a
ADD: +
Identifier: b
R_PAREN: )
'SEMICOLON': ;
Identifier: printf
L_PAREN: (
const_string: "////"
R_PAREN: )
'SEMICOLON': ;
Identifier: printf
L_PAREN: (
const_string: "////"
R_PAREN: )
'SEMICOLON': ;
Identifier: printf
L_PAREN: (
const_string: "hello world"
R_PAREN: )
'SEMICOLON': ;
Identifier: printf
L_PAREN: (
const_string: "////"
R_PAREN: )
'SEMICOLON': ;
'R_BRACE': }
===================================================
        Compiler finished, the result are:
All char number: 198
Line count: 11
Ketword number: 3
Identifier number: 13
Integer number: 2
Floate number: 0
Error number: 0
```

6.2 测试用例 2

```
int main(int argc, char **argv)
{
  yylex()
  return 0;
}
int yywrap()
{
    return 1;
}
```

执行结果

```
E:\bupt-homework\compiler_principle\Lex>a test2.c
KEYWORD: int
Identifier: main
L_PAREN: (
KEYWORD: int
Identifier: argc
COMMA: ,
KEYWORD: char
STAR: *
STAR: *
Identifier: argv
R_PAREN: )
'L_BRACE': {
Identifier: yylex
L_PAREN: (
R_PAREN: )
KEYWORD: return
IntNum:   0
'SEMICOLON': ;
'R_BRACE': }
KEYWORD: int
Identifier: yywrap
L_PAREN: (
R_PAREN: )
'L_BRACE': {
KEYWORD: return
IntNum:   1
'SEMICOLON': ;
'R_BRACE': }
===================================================
        Compiler finished, the result are:
All char number: 85
Line count: 9
Ketword number: 6
Identifier number: 5
Integer number: 2
Floate number: 0
Error number: 0
```

6.3 测试用例 3

```
/*
1
2
3
4
5
6
7
8
*/
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "C/C++ Runner: Debug Session",
      "type": "cppdbg",
      "request": "launch",
      "args": [],
      "stopAtEntry": false,
      "externalConsole": true,
      "cwd": "e:/bupt-homework/compiler_principle/Lex",
      "program": "e:/bupt-
homework/compiler_principle/Lex/build/Debug/outDebug",
      "MIMode": "gdb",
      "miDebuggerPath": "gdb",
      "setupCommands": [
        {
          "description": "Enable pretty-printing for gdb",
          "text": "-enable-pretty-printing",
          "ignoreFailures": true
        }
      ]
    }
  ]
}
```

执行结果

```
E:\bupt-homework\compiler_principle\Lex>a test3.c
'L_BRACE': {
const_string: "version"
COLON: :
const_string: "0.2.0"
COMMA: ,
const_string: "configurations"
COLON: :
L_SQUARE: [
'L_BRACE': {
const_string: "name"
COLON: :
const_string: "C/C++ Runner: Debug Session"
COMMA: ,
const_string: "type"
COLON: :
const_string: "cppdbg"
COMMA: ,
const_string: "request"
COLON: :
const_string: "launch"
COMMA: ,
const_string: "args"
COLON: :
L_SQUARE: [
R_SQUARE: ]
COMMA: ,
const_string: "stopAtEntry"
COLON: :
Identifier: false
COMMA: ,
const_string: "externalConsole"
COLON: :
Identifier: true
COMMA: ,
const_string: "cwd"
COLON: :
const_string: "e:/bupt-homework/compiler_principle/Lex"
COMMA: ,
const_string: "program"
COLON: :
const_string: "e:/bupt-
homework/compiler_principle/Lex/build/Debug/outDebug"
COMMA: ,
const_string: "MIMode"
COLON: :
const_string: "gdb"
COMMA: ,
const_string: "miDebuggerPath"
COLON: :
const_string: "gdb"
COMMA: ,
const_string: "setupCommands"
COLON: :
L_SQUARE: [
'L_BRACE': {
const_string: "description"
COLON: :
const_string: "Enable pretty-printing for gdb"
COMMA: ,
const_string: "text"
COLON: :
const_string: "-enable-pretty-printing"
COMMA: ,
const_string: "ignoreFailures"
COLON: :
Identifier: true
'R_BRACE': }
R_SQUARE: ]
'R_BRACE': }
R_SQUARE: ]
'R_BRACE': }
===================================================
        Compiler finished, the result are:
All char number: 622
Line count: 36
Ketword number: 0
Identifier number: 3
Integer number: 0
Floate number: 0
Error number: 0
```

7 生成的 C 语言文件

```
/* A lexical scanner generated by flex */

/* Scanner skeleton version:
 * $Header: /home/daffy/u0/vern/flex/RCS/flex.skl,v 2.91 96/09/10
16:58:48 vern Exp $
 */

#define FLEX_SCANNER
#define YY_FLEX_MAJOR_VERSION 2
#define YY_FLEX_MINOR_VERSION 5

#include <stdio.h>


/* cfront 1.2 defines "c_plusplus" instead of "__cplusplus" */
#ifdef c_plusplus
#ifndef __cplusplus
#define __cplusplus
#endif
```

```
#endif


#ifdef __cplusplus

#include <stdlib.h>
#include <unistd.h>

/* Use prototypes in function declarations. */
#define YY_USE_PROTOS

/* The "const" storage-class-modifier is valid. */
#define YY_USE_CONST

#else   /* ! __cplusplus */

#if __STDC__

#define YY_USE_PROTOS
#define YY_USE_CONST

#endif  /* __STDC__ */
#endif  /* ! __cplusplus */

#ifdef __TURBOC__
 #pragma warn -rch
 #pragma warn -use
#include <io.h>
#include <stdlib.h>
#define YY_USE_CONST
#define YY_USE_PROTOS
#endif

#ifdef YY_USE_CONST
#define yyconst const
#else
#define yyconst
#endif


#ifdef YY_USE_PROTOS
#define YY_PROTO(proto) proto
#else
```

```
#define YY_PROTO(proto) ()
#endif

/* Returned upon end-of-file. */
#define YY_NULL 0

/* Promotes a possibly negative, possibly signed char to an unsigned
 * integer for use as an array index.  If the signed char is negative,
 * we want to instead treat it as an 8-bit unsigned char, hence the
 * double cast.
 */
#define YY_SC_TO_UI(c) ((unsigned int) (unsigned char) c)

/* Enter a start condition.  This macro really ought to take a
parameter,
 * but we do it the disgusting crufty way forced on us by the ()-less
 * definition of BEGIN.
 */
#define BEGIN yy_start = 1 + 2 *

/* Translate the current start state into a value that can be later
handed
 * to BEGIN to return to the state.  The YYSTATE alias is for lex
 * compatibility.
 */
#define YY_START ((yy_start - 1) / 2)
#define YYSTATE YY_START

/* Action number for EOF rule of a given start state. */
#define YY_STATE_EOF(state) (YY_END_OF_BUFFER + state + 1)

/* Special action meaning "start processing a new file". */
#define YY_NEW_FILE yyrestart( yyin )

#define YY_END_OF_BUFFER_CHAR 0

/* Size of default input buffer. */
#define YY_BUF_SIZE 16384

typedef struct yy_buffer_state *YY_BUFFER_STATE;

extern int yyleng;
extern FILE *yyin, *yyout;

#define EOB_ACT_CONTINUE_SCAN 0
#define EOB_ACT_END_OF_FILE 1
#define EOB_ACT_LAST_MATCH 2

/* The funky do-while in the following #define is used to turn the
definition
 * int a single C statement (which needs a semi-colon terminator).  This
 * avoids problems with code like:
 *
 *  if ( condition_holds )
 *      yyless( 5 );
 *  else
 *      do_something_else();
 *
 * Prior to using the do-while the compiler would get upset at the
 * "else" because it interpreted the "if" statement as being all
 * done when it reached the ';' after the yyless() call.
 */

/* Return all but the first 'n' matched characters back to the input
stream. */

#define yyless(n) \
    do \
        { \
        /* Undo effects of setting up yytext. */ \
        *yy_cp = yy_hold_char; \
        YY_RESTORE_YY_MORE_OFFSET \
        yy_c_buf_p = yy_cp = yy_bp + n - YY_MORE_ADJ; \
        YY_DO_BEFORE_ACTION; /* set up yytext again */ \
        } \
    while ( 0 )

#define unput(c) yyunput( c, yytext_ptr )

/* The following is because we cannot portably get our hands on size_t
 * (without autoconf's help, which isn't available because we want
 * flex-generated scanners to compile on their own).
 */
typedef unsigned int yy_size_t;
```

```
struct yy_buffer_state
    {
    FILE *yy_input_file;

    char *yy_ch_buf;        /* input buffer */
    char *yy_buf_pos;       /* current position in input buffer */

    /* Size of input buffer in bytes, not including room for EOB
     * characters.
     */
    yy_size_t yy_buf_size;

    /* Number of characters read into yy_ch_buf, not including EOB
     * characters.
     */
    int yy_n_chars;

    /* Whether we "own" the buffer - i.e., we know we created it,
     * and can realloc() it to grow it, and should free() it to
     * delete it.
     */
    int yy_is_our_buffer;

    /* Whether this is an "interactive" input source; if so, and
     * if we're using stdio for input, then we want to use getc()
     * instead of fread(), to make sure we stop fetching input after
     * each newline.
     */
    int yy_is_interactive;

    /* Whether we're considered to be at the beginning of a line.
     * If so, '^' rules will be active on the next match, otherwise
     * not.
     */
    int yy_at_bol;

    /* Whether to try to fill the input buffer when we reach the
     * end of it.
     */
    int yy_fill_buffer;
    int yy_buffer_status;
#define YY_BUFFER_NEW 0
#define YY_BUFFER_NORMAL 1
    /* When an EOF's been seen but there's still some text to process
     * then we mark the buffer as YY_EOF_PENDING, to indicate that we
     * shouldn't try reading from the input source any more.  We might
     * still have a bunch of tokens to match, though, because of
     * possible backing-up.
     *
     * When we actually see the EOF, we change the status to "new"
     * (via yyrestart()), so that the user can continue scanning by
     * just pointing yyin at a new input file.
     */
#define YY_BUFFER_EOF_PENDING 2
    };

static YY_BUFFER_STATE yy_current_buffer = 0;

/* We provide macros for accessing buffer states in case in the
 * future we want to put the buffer states in a more general
 * "scanner state".
 */
#define YY_CURRENT_BUFFER yy_current_buffer


/* yy_hold_char holds the character lost when yytext is formed. */
static char yy_hold_char;

static int yy_n_chars;      /* number of characters read into yy_ch_buf
*/


int yyleng;

/* Points to current character in buffer. */
static char *yy_c_buf_p = (char *) 0;
static int yy_init = 1;     /* whether we need to initialize */
static int yy_start = 0;    /* start state number */

/* Flag which is used to allow yywrap()'s to do buffer switches
 * instead of setting up a fresh yyin.  A bit of a hack ...
 */
static int yy_did_buffer_switch_on_eof;
```

```c
void yyrestart YY_PROTO(( FILE *input_file ));

void yy_switch_to_buffer YY_PROTO(( YY_BUFFER_STATE new_buffer ));
void yy_load_buffer_state YY_PROTO(( void ));
YY_BUFFER_STATE yy_create_buffer YY_PROTO(( FILE *file, int size ));
void yy_delete_buffer YY_PROTO(( YY_BUFFER_STATE b ));
void yy_init_buffer YY_PROTO(( YY_BUFFER_STATE b, FILE *file ));
void yy_flush_buffer YY_PROTO(( YY_BUFFER_STATE b ));
#define YY_FLUSH_BUFFER yy_flush_buffer( yy_current_buffer )

YY_BUFFER_STATE yy_scan_buffer YY_PROTO(( char *base, yy_size_t
size ));
YY_BUFFER_STATE yy_scan_string YY_PROTO(( yyconst char *yy_str ));
YY_BUFFER_STATE yy_scan_bytes YY_PROTO(( yyconst char *bytes, int
len ));

static void *yy_flex_alloc YY_PROTO(( yy_size_t ));
static void *yy_flex_realloc YY_PROTO(( void *, yy_size_t ));
static void yy_flex_free YY_PROTO(( void * ));

#define yy_new_buffer yy_create_buffer

#define yy_set_interactive(is_interactive) \
    { \
    if ( ! yy_current_buffer ) \
        yy_current_buffer = yy_create_buffer( yyin, YY_BUF_SIZE ); \
    yy_current_buffer->yy_is_interactive = is_interactive; \
    }

#define yy_set_bol(at_bol) \
    { \
    if ( ! yy_current_buffer ) \
        yy_current_buffer = yy_create_buffer( yyin, YY_BUF_SIZE ); \
    yy_current_buffer->yy_at_bol = at_bol; \
    }

#define YY_AT_BOL() (yy_current_buffer->yy_at_bol)

typedef unsigned char YY_CHAR;
FILE *yyin = (FILE *) 0, *yyout = (FILE *) 0;
typedef int yy_state_type;
```

```
extern char *yytext;
#define yytext_ptr yytext

static yy_state_type yy_get_previous_state YY_PROTO(( void ));
static yy_state_type yy_try_NUL_trans YY_PROTO(( yy_state_type
current_state ));
static int yy_get_next_buffer YY_PROTO(( void ));
static void yy_fatal_error YY_PROTO(( yyconst char msg[] ));

/* Done after the current pattern has been matched and before the
 * corresponding action - sets up yytext.
 */
#define YY_DO_BEFORE_ACTION \
    yytext_ptr = yy_bp; \
    yyleng = (int) (yy_cp - yy_bp); \
    yy_hold_char = *yy_cp; \
    *yy_cp = '\0'; \
    yy_c_buf_p = yy_cp;

#define YY_NUM_RULES 59
#define YY_END_OF_BUFFER 60
static yyconst short int yy_accept[192] =
    {   0,
        0,    0,   60,   58,   57,   56,   43,   58,   55,   49,
       42,   58,   37,   38,   47,   46,   34,   45,   41,   48,
        3,   35,   31,   50,   36,   51,   54,    6,   39,   40,
       52,    6,    6,    6,    6,    6,    6,    6,    6,    6,
        6,    6,    6,    6,    6,    6,   32,   53,   33,   44,
       30,    0,    7,    0,   15,   33,   24,   16,    0,    0,
       13,   21,   11,   22,   12,   23,    1,    2,   14,    0,
        3,    0,   40,   32,   39,   20,   26,   29,   28,   27,
       19,    0,    0,    6,   17,    6,    6,    6,    6,    6,
        6,    5,    6,    6,    6,    6,    6,    6,    5,    6,

        6,    6,    6,    6,    6,    6,    6,    6,    6,    6,
       18,   25,    8,    4,    0,   10,    9,    6,    6,    6,
        6,    6,    6,    6,    6,    6,    6,    6,    6,    6,
        6,    6,    6,    6,    6,    6,    6,    6,    6,    6,
        6,    6,    6,    6,    0,    3,    6,    6,    6,    6,
        6,    6,    6,    6,    6,    6,    6,    6,    6,    6,
        6,    6,    6,    6,    6,    6,    0,    6,    6,    6,
        6,    6,    6,    6,    6,    6,    6,    6,    6,    6,
        6,    4,    6,    6,    6,    6,    6,    6,    6,    6,
        0

    } ;

static yyconst int yy_ec[256] =
    {   0,
        1,    1,    1,    1,    1,    1,    1,    1,    2,    3,
        2,    2,    1,    1,    1,    1,    1,    1,    1,    1,
        1,    1,    1,    1,    1,    1,    1,    1,    1,    1,
        1,    2,    4,    5,    6,    1,    7,    8,    9,   10,
       11,   12,   13,   14,   15,   16,   17,   18,   18,   18,
       18,   18,   18,   18,   18,   18,   18,   19,   20,   21,
       22,   23,   24,    1,   25,   25,   25,   25,   26,   25,
       25,   25,   25,   25,   25,   25,   25,   25,   25,   25,
       25,   25,   25,   25,   25,   25,   25,   25,   25,   25,
       27,   28,   29,   30,   25,    1,   31,   32,   33,   34,

       35,   36,   37,   38,   39,   25,   40,   41,   42,   43,
       44,   45,   25,   46,   47,   48,   49,   50,   51,   52,
       53,   54,   55,   56,   57,   58,    1,    1,    1,    1,
        1,    1,    1,    1,    1,    1,    1,    1,    1,    1,
        1,    1,    1,    1,    1,    1,    1,    1,    1,    1,
        1,    1,    1,    1,    1,    1,    1,    1,    1,    1,
        1,    1,    1,    1,    1,    1,    1,    1,    1,    1,
        1,    1,    1,    1,    1,    1,    1,    1,    1,    1,
        1,    1,    1,    1,    1,    1,    1,    1,    1,    1,
        1,    1,    1,    1,    1,    1,    1,    1,    1,    1,

        1,    1,    1,    1,    1,    1,    1,    1,    1,    1,
        1,    1,    1,    1,    1,    1,    1,    1,    1,    1,
        1,    1,    1,    1,    1,    1,    1,    1,    1,    1,
        1,    1,    1,    1,    1,    1,    1,    1,    1,    1,
        1,    1,    1,    1,    1,    1,    1,    1,    1,    1,
        1,    1,    1,    1,    1
    } ;

static yyconst int yy_meta[59] =
    {   0,
        1,    1,    2,    1,    3,    1,    1,    1,    4,    1,
        1,    1,    1,    1,    1,    1,    1,    3,    1,    1,
        1,    1,    1,    1,    3,    3,    1,    1,    1,    1,
        3,    3,    3,    3,    3,    3,    3,    3,    3,    3,
        3,    3,    3,    3,    3,    3,    3,    3,    3,    3,
        3,    3,    3,    3,    1,    1,    1,    1
    } ;

static yyconst short int yy_base[195] =
    {   0,
        0,    0,  247,  518,  518,  518,  217,   54,  518,   38,
       54,  202,  518,  518,  198,   50,  518,   51,  518,   53,
       51,  186,  518,   71,   57,   61,  518,   59,  518,  518,
      182,   76,   82,   89,   91,   96,   90,   97,  102,   99,
      105,  108,  107,  110,  113,  127,  518,   67,  518,  518,
      518,  116,  518,    0,  518,  518,  518,  518,  121,  149,
      518,  518,  518,  518,  518,  518,  518,  518,  518,  124,
      136,  148,  518,  518,  518,  107,  518,  518,  518,  518,
      102,  145,   81,  146,  518,  159,  161,  167,  169,  170,
      172,  175,  178,  177,  180,  183,  185,  186,  188,  194,

      193,  196,  201,  206,  207,  212,  213,  224,  241,  247,
      518,  518,  518,  214,   85,  518,  518,  232,  250,  252,
      257,  259,  260,  265,  270,  268,  279,  280,  285,  287,
      290,  293,  295,  303,  304,  310,  311,  313,  312,  319,
      325,  214,  326,  327,  154,   79,  335,  328,  340,  343,
      345,  350,  357,  360,  362,  365,  372,  373,  375,  378,
      385,  386,  380,  388,  393,  395,   70,  401,  407,  410,
      417,  419,  422,  424,  427,  429,  430,  437,  438,  444,
      446,   53,  447,  452,  459,  462,  461,  471,  474,  472,
      518,  508,  512,  513

    } ;

static yyconst short int yy_def[195] =
    {   0,
      191,    1,  191,  191,  191,  191,  191,  192,  191,  191,
      191,  193,  191,  191,  191,  191,  191,  191,  191,  191,
      191,  191,  191,  191,  191,  191,  191,  194,  191,  191,
      191,  194,  194,  194,  194,  194,  194,  194,  194,  194,
      194,  194,  194,  194,  194,  194,  191,  191,  191,  191,
      191,  192,  191,  192,  191,  191,  191,  191,  193,  193,
      191,  191,  191,  191,  191,  191,  191,  191,  191,  191,
      191,  191,  191,  191,  191,  191,  191,  191,  191,  191,
      191,  192,  193,  194,  191,  194,  194,  194,  194,  194,
      194,  194,  194,  194,  194,  194,  194,  194,  194,  194,

      194,  194,  194,  194,  194,  194,  194,  194,  194,  194,
      191,  191,  191,  191,  191,  191,  191,  194,  194,  194,
      194,  194,  194,  194,  194,  194,  194,  194,  194,  194,
      194,  194,  194,  194,  194,  194,  194,  194,  194,  194,
      194,  194,  194,  194,  191,  191,  194,  194,  194,  194,
      194,  194,  194,  194,  194,  194,  194,  194,  194,  194,
      194,  194,  194,  194,  194,  194,  191,  194,  194,  194,
      194,  194,  194,  194,  194,  194,  194,  194,  194,  194,
      194,  191,  194,  194,  194,  194,  194,  194,  194,  194,
        0,  191,  191,  191

    } ;

static yyconst short int yy_nxt[577] =
    {   0,
        4,    5,    6,    7,    8,    9,   10,   11,   12,   13,
       14,   15,   16,   17,   18,   19,   20,   21,   22,   23,
       24,   25,   26,   27,   28,   28,   29,    4,   30,   31,
       32,   33,   34,   35,   36,   37,   38,   28,   39,   28,
       40,   28,   28,   28,   28,   41,   42,   43,   44,   45,
       46,   28,   28,   28,   47,   48,   49,   50,   53,   55,
       56,   57,   62,   82,   67,   64,   70,   83,   71,   68,
      182,   63,   65,   66,   69,   58,   72,   74,   78,   79,
       82,   54,   80,   81,   83,   72,   82,  182,  111,   75,
       83,   76,   77,   82,   82,   82,  146,   83,   83,   83,

       82,   82,  146,   82,   83,   83,   82,   83,   60,   82,
       83,   82,   82,   83,   82,   83,   83,   82,   83,   88,
       53,   83,  112,  117,   86,   91,   89,   87,  116,  113,
       96,   82,   90,   97,   92,   83,   93,   99,   94,  102,
       98,  114,  101,   54,  100,  103,  104,   95,   60,   53,
      191,   70,  108,   71,  191,  105,  109,   59,  106,  107,
      115,   72,  115,  191,  110,  191,  167,  191,  167,  191,
       72,  191,   54,  191,  191,  191,  191,  191,  191,  191,
      191,  191,  191,  191,  191,  191,  191,  191,  191,  191,
      191,  191,  191,  191,  191,  119,  191,  191,  191,  121,

      191,  191,  191,   85,  191,  191,  118,  123,   73,  191,
      191,  191,  122,  120,  191,  191,  191,  191,  191,   61,
      191,  191,  191,  124,  125,  126,  128,  127,  191,   60,
       99,  114,  191,  129,  130,  131,  191,  136,   51,  145,
      191,   99,  134,  132,  133,  191,  191,   99,  145,  191,
      138,  191,  137,  191,  191,  191,  191,  139,  191,  135,
      191,  191,  140,  191,  191,  191,  191,  191,  191,  191,
      141,  191,  191,  191,  191,   99,  191,  191,  191,  142,
      147,  143,  191,  191,  191,  144,   99,  191,  191,  191,
      150,  191,  191,  191,  191,  191,  151,  191,  191,  191,

      191,  191,   99,  191,   99,  148,  149,  191,  191,   99,
      153,  191,  191,  152,  191,  191,  191,  191,  191,  191,
      191,  191,  191,  191,  191,  154,   99,  191,   99,  191,
      191,  191,  191,  191,  191,  191,  191,  191,  158,  191,
      156,  155,  191,  191,  191,  157,  162,  191,  191,  191,
      191,  191,  191,  191,  191,  191,  165,  159,  191,  160,
      161,  191,  163,  164,  191,  191,  191,  166,  191,  191,
      191,  191,  191,  191,   99,   99,  191,  191,  168,  191,
      191,  191,  191,  191,  191,  170,  191,  191,  191,  191,
      191,  169,  191,  191,  191,  171,  191,  191,  191,  191,

      191,  191,  172,  191,   99,  191,  174,  173,  191,  191,
      177,  191,   99,  176,  191,  191,  175,  178,  191,  179,
      191,  191,   99,  191,  180,  191,  191,  191,  191,   99,
      191,  191,  191,  191,  191,  191,  191,  191,  191,  191,
      181,  191,  191,  183,   99,  191,  191,  184,  191,  191,
      191,  191,  191,   99,  191,  191,  191,   99,  191,   99,
      191,   99,   99,  191,   99,  191,  191,  191,  191,  191,
      191,  191,  185,  191,   99,  191,  191,   99,  191,  191,
      191,  191,  191,  191,  187,  191,  186,  191,  191,  191,
      191,  191,  191,  191,   99,  188,  189,  191,  191,   99,

      191,  190,  191,  191,  191,   99,   99,   99,   52,  191,
       52,   52,   59,  191,   59,   84,   84,    3,  191,  191,
      191,  191,  191,  191,  191,  191,  191,  191,  191,  191,
      191,  191,  191,  191,  191,  191,  191,  191,  191,  191,
      191,  191,  191,  191,  191,  191,  191,  191,  191,  191,
      191,  191,  191,  191,  191,  191,  191,  191,  191,  191,
      191,  191,  191,  191,  191,  191,  191,  191,  191,  191,
      191,  191,  191,  191,  191,  191
    } ;

static yyconst short int yy_chk[577] =
    {   0,
        1,    1,    1,    1,    1,    1,    1,    1,    1,    1,
        1,    1,    1,    1,    1,    1,    1,    1,    1,    1,
        1,    1,    1,    1,    1,    1,    1,    1,    1,    1,
        1,    1,    1,    1,    1,    1,    1,    1,    1,    1,
        1,    1,    1,    1,    1,    1,    1,    1,    1,    1,
        1,    1,    1,    1,    1,    1,    1,    1,    8,   10,
       10,   11,   16,   28,   20,   18,   21,   28,   21,   20,
      182,   16,   18,   18,   20,   11,   21,   24,   25,   25,
       32,    8,   26,   26,   32,   21,   33,  167,   48,   24,
       33,   24,   24,   34,   37,   35,  146,   34,   37,   35,

       36,   38,  115,   40,   36,   38,   39,   40,   83,   41,
       39,   43,   42,   41,   44,   43,   42,   45,   44,   34,
       52,   45,   48,   81,   32,   35,   34,   33,   76,   59,
       37,   46,   34,   37,   35,   46,   36,   39,   36,   41,
       38,   70,   40,   52,   39,   42,   42,   36,   59,   82,
       84,   71,   44,   71,   84,   42,   45,   60,   42,   43,
       72,   71,   72,   86,   46,   87,  145,   86,  145,   87,
       71,   88,   82,   89,   90,   88,   91,   89,   90,   92,
       91,   94,   93,   92,   95,   94,   93,   96,   95,   97,
       98,   96,   99,   97,   98,   87,   99,  101,  100,   89,

      102,  101,  100,   31,  102,  103,   86,   91,   22,  103,
      104,  105,   90,   88,  104,  105,  106,  107,  142,   15,
      106,  107,  142,   92,   93,   94,   96,   95,  108,   12,
       97,  114,  108,   98,  100,  101,  118,  105,    7,  114,
      118,  100,  104,  102,  103,  109,    3,  142,  114,  109,
      106,  110,  105,    0,  119,  110,  120,  107,  119,  104,
      120,  121,  108,  122,  123,  121,    0,  122,  123,  124,
      108,    0,  126,  124,  125,  118,  126,    0,  125,  109,
      119,  109,    0,  127,  128,  110,  120,  127,  128,  129,
      123,  130,    0,  129,  131,  130,  124,  132,  131,  133,

        0,  132,  121,  133,  125,  122,  122,  134,  135,  126,
      128,  134,  135,  127,  136,  137,  139,  138,  136,  137,
      139,  138,    0,  140,    0,  130,  131,  140,  129,  141,
      143,  144,  148,  141,  143,  144,  148,    0,  135,  147,
      133,  132,    0,  147,  149,  134,  139,  150,  149,  151,
        0,  150,    0,  151,  152,    0,  143,  136,  152,  137,
      138,  153,  140,  141,  154,  153,  155,  144,  154,  156,
      155,    0,    0,  156,  147,  148,  157,  158,  149,  159,
      157,  158,  160,  159,  163,  151,  160,    0,  163,  161,
      162,  150,  164,  161,  162,  152,  164,  165,    0,  166,

        0,  165,  154,  166,  153,  168,  157,  155,    0,  168,
      160,  169,  156,  159,  170,  169,  158,  161,  170,  162,
        0,  171,  163,  172,  164,  171,  173,  172,  174,  166,
      173,  175,  174,  176,  177,  175,    0,  176,  177,    0,
      165,  178,  179,  168,  170,  178,  179,  169,  180,    0,
      181,  183,  180,  172,  181,  183,  184,  174,    0,  171,
      184,  176,  175,  185,  173,  187,  186,  185,    0,  187,
      186,    0,  179,    0,  178,  188,  190,  177,  189,  188,
      190,    0,  189,    0,  181,    0,  180,    0,    0,    0,
        0,    0,    0,    0,  185,  183,  186,    0,    0,  184,

        0,  187,    0,    0,    0,  188,  190,  189,  192,    0,
      192,  192,  193,    0,  193,  194,  194,  191,  191,  191,
      191,  191,  191,  191,  191,  191,  191,  191,  191,  191,
      191,  191,  191,  191,  191,  191,  191,  191,  191,  191,
      191,  191,  191,  191,  191,  191,  191,  191,  191,  191,
      191,  191,  191,  191,  191,  191,  191,  191,  191,  191,
      191,  191,  191,  191,  191,  191,  191,  191,  191,  191,
      191,  191,  191,  191,  191,  191
    } ;

static yy_state_type yy_last_accepting_state;
static char *yy_last_accepting_cpos;

/* The intent behind this definition is that it'll catch
 * any uses of REJECT which flex missed.
 */
#define REJECT reject_used_but_not_detected
#define yymore() yymore_used_but_not_detected
#define YY_MORE_ADJ 0
#define YY_RESTORE_YY_MORE_OFFSET
char *yytext;
#line 1 "lex.l"
#define INITIAL 0
#line 2 "lex.l"
#include<math.h>
#include<stdlib.h>
#include<stdio.h>

void jumpMultiComment(void);
void jumpSingleComment(void);
void printCompilerResult(void);
void addCharNum(int length);
int lineCount = 1;
int keywordNum = 0;
int idNum = 0;
int errorNum = 0;
int intNum = 0;
int floatNum = 0;
int charNum = 0;
#line 571 "lex.yy.c"

/* Macros after this point can all be overridden by user definitions in
 * section 1.
 */

#ifndef YY_SKIP_YYWRAP
#ifdef __cplusplus
extern "C" int yywrap YY_PROTO(( void ));
#else
extern int yywrap YY_PROTO(( void ));
#endif
#endif

#ifndef YY_NO_UNPUT
static void yyunput YY_PROTO(( int c, char *buf_ptr ));
#endif

#ifndef yytext_ptr
static void yy_flex_strncpy YY_PROTO(( char *, yyconst char *, int ));
#endif

#ifdef YY_NEED_STRLEN
static int yy_flex_strlen YY_PROTO(( yyconst char * ));
#endif

#ifndef YY_NO_INPUT
#ifdef __cplusplus
static int yyinput YY_PROTO(( void ));
#else
static int input YY_PROTO(( void ));
#endif
```

```
#endif

#if YY_STACK_USED
static int yy_start_stack_ptr = 0;
static int yy_start_stack_depth = 0;
static int *yy_start_stack = 0;
#ifndef YY_NO_PUSH_STATE
static void yy_push_state YY_PROTO(( int new_state ));
#endif
#ifndef YY_NO_POP_STATE
static void yy_pop_state YY_PROTO(( void ));
#endif
#ifndef YY_NO_TOP_STATE
static int yy_top_state YY_PROTO(( void ));
#endif

#else
#define YY_NO_PUSH_STATE 1
#define YY_NO_POP_STATE 1
#define YY_NO_TOP_STATE 1
#endif

#ifdef YY_MALLOC_DECL
YY_MALLOC_DECL
#else
#if __STDC__
#ifndef __cplusplus
#include <stdlib.h>
#endif
#else
/* Just try to get by without declaring the routines.  This will fail
 * miserably on non-ANSI systems for which sizeof(size_t) !=
sizeof(int)
 * or sizeof(void*) != sizeof(int).
 */
#endif
#endif

/* Amount of stuff to slurp up with each read. */
#ifndef YY_READ_BUF_SIZE
#define YY_READ_BUF_SIZE 8192
#endif
```

```
/* Copy whatever the last rule matched to the standard output. */

#ifndef ECHO
/* This used to be an fputs(), but since the string might contain
NUL's,
 * we now use fwrite().
 */
#define ECHO (void) fwrite( yytext, yyleng, 1, yyout )
#endif

/* Gets input and stuffs it into "buf".  number of characters read, or
YY_NULL,
 * is returned in "result".
 */
#ifndef YY_INPUT
#define YY_INPUT(buf,result,max_size) \
    if ( yy_current_buffer->yy_is_interactive ) \
        { \
        int c = '*', n; \
        for ( n = 0; n < max_size && \
                 (c = getc( yyin )) != EOF && c != '\n'; ++n ) \
            buf[n] = (char) c; \
        if ( c == '\n' ) \
            buf[n++] = (char) c; \
        if ( c == EOF && ferror( yyin ) ) \
            YY_FATAL_ERROR( "input in flex scanner failed" ); \
        result = n; \
        } \
    else if ( ((result = fread( buf, 1, max_size, yyin )) == 0) \
          && ferror( yyin ) ) \
        YY_FATAL_ERROR( "input in flex scanner failed" );
#endif

/* No semi-colon after return; correct usage is to write
"yyterminate();" -
 * we don't want an extra ';' after the "return" because that will
cause
 * some compilers to complain about unreachable statements.
 */
#ifndef yyterminate
#define yyterminate() return YY_NULL
```

```
#endif

/* Number of entries by which start-condition stack grows. */
#ifndef YY_START_STACK_INCR
#define YY_START_STACK_INCR 25
#endif

/* Report a fatal error. */
#ifndef YY_FATAL_ERROR
#define YY_FATAL_ERROR(msg) yy_fatal_error( msg )
#endif

/* Default declaration of generated scanner - a define so the user can
 * easily add parameters.
 */
#ifndef YY_DECL
#define YY_DECL int yylex YY_PROTO(( void ))
#endif

/* Code executed at the beginning of each rule, after yytext and yyleng
 * have been set up.
 */
#ifndef YY_USER_ACTION
#define YY_USER_ACTION
#endif

/* Code executed at the end of each rule. */
#ifndef YY_BREAK
#define YY_BREAK break;
#endif

#define YY_RULE_SETUP \
    YY_USER_ACTION

YY_DECL
    {
    register yy_state_type yy_current_state;
    register char *yy_cp, *yy_bp;
    register int yy_act;

#line 23 "lex.l"
```

```c
#line 724 "lex.yy.c"

    if ( yy_init )
        {
        yy_init = 0;

#ifdef YY_USER_INIT
        YY_USER_INIT;
#endif

        if ( ! yy_start )
            yy_start = 1;   /* first start state */

        if ( ! yyin )
            yyin = stdin;

        if ( ! yyout )
            yyout = stdout;

        if ( ! yy_current_buffer )
            yy_current_buffer =
                yy_create_buffer( yyin, YY_BUF_SIZE );

        yy_load_buffer_state();
        }

    while ( 1 )     /* loops until end-of-file is reached */
        {
        yy_cp = yy_c_buf_p;

        /* Support of yytext. */
        *yy_cp = yy_hold_char;

        /* yy_bp points to the position in yy_ch_buf of the start of
         * the current run.
         */
        yy_bp = yy_cp;

        yy_current_state = yy_start;
yy_match:
        do
            {
```

```
            register YY_CHAR yy_c = yy_ec[YY_SC_TO_UI(*yy_cp)];
            if ( yy_accept[yy_current_state] )
                {
                yy_last_accepting_state = yy_current_state;
                yy_last_accepting_cpos = yy_cp;
                }
            while ( yy_chk[yy_base[yy_current_state] + yy_c] !=
yy_current_state )
                {
                yy_current_state = (int) yy_def[yy_current_state];
                if ( yy_current_state >= 192 )
                    yy_c = yy_meta[(unsigned int) yy_c];
                }
            yy_current_state = yy_nxt[yy_base[yy_current_state] +
(unsigned int) yy_c];
            ++yy_cp;
            }
        while ( yy_base[yy_current_state] != 518 );

yy_find_action:
        yy_act = yy_accept[yy_current_state];
        if ( yy_act == 0 )
            { /* have to back up */
            yy_cp = yy_last_accepting_cpos;
            yy_current_state = yy_last_accepting_state;
            yy_act = yy_accept[yy_current_state];
            }

        YY_DO_BEFORE_ACTION;


do_action:  /* This label is used only to access EOF actions. */


        switch ( yy_act )
    { /* beginning of action switch */
            case 0: /* must back up */
            /* undo the effects of YY_DO_BEFORE_ACTION */
            *yy_cp = yy_hold_char;
            yy_cp = yy_last_accepting_cpos;
            yy_current_state = yy_last_accepting_state;
            goto yy_find_action;
```

```
case 1:
YY_RULE_SETUP
#line 24 "lex.l"
{
    charNum+=2;
    jumpMultiComment();
}
    YY_BREAK
case 2:
YY_RULE_SETUP
#line 28 "lex.l"
{
    charNum += 2;
    jumpSingleComment();
}
    YY_BREAK
case 3:
YY_RULE_SETUP
#line 32 "lex.l"
{
    printf("IntNum:   %s\n",yytext);
    intNum++;
    addCharNum(yyleng);
}
    YY_BREAK
case 4:
YY_RULE_SETUP
#line 37 "lex.l"
{
    printf("FloatNum:   %s\n",yytext);
    floatNum++;
    addCharNum(yyleng);
}
    YY_BREAK
case 5:
YY_RULE_SETUP
#line 42 "lex.l"
{
    printf("KEYWORD: %s\n", yytext);
    keywordNum++;
    addCharNum(yyleng);
}
```

```
    YY_BREAK
case 6:
YY_RULE_SETUP
#line 47 "lex.l"
{
    printf("Identifier: %s\n",yytext);
    idNum++;
    addCharNum(yyleng);
}
    YY_BREAK
case 7:
YY_RULE_SETUP
#line 52 "lex.l"
{ printf("const_string: %s\n",yytext); addCharNum(yyleng);}
    YY_BREAK
case 8:
YY_RULE_SETUP
#line 53 "lex.l"
{ printf("const_char: %s\n",yytext); addCharNum(yyleng);}
    YY_BREAK
case 9:
YY_RULE_SETUP
#line 55 "lex.l"
{ printf("RIGHT_ASSIGN: %s\n",yytext); addCharNum(yyleng);}
    YY_BREAK
case 10:
YY_RULE_SETUP
#line 56 "lex.l"
{ printf("LEFT_ASSIGN: %s\n",yytext); addCharNum(yyleng);}
    YY_BREAK
case 11:
YY_RULE_SETUP
#line 57 "lex.l"
{ printf("ADD_ASSIGN: %s\n",yytext); addCharNum(yyleng);}
    YY_BREAK
case 12:
YY_RULE_SETUP
#line 58 "lex.l"
{ printf("SUB_ASSIGN: %s\n",yytext); addCharNum(yyleng);}
    YY_BREAK
case 13:
YY_RULE_SETUP
```

```
#line 59 "lex.l"
{ printf("MUL_ASSIGN: %s\n",yytext); addCharNum(yyleng);}
	YY_BREAK
case 14:
YY_RULE_SETUP
#line 60 "lex.l"
{ printf("DIV_ASSIGN: %s\n",yytext); addCharNum(yyleng);}
	YY_BREAK
case 15:
YY_RULE_SETUP
#line 61 "lex.l"
{ printf("MOD_ASSIGN: %s\n",yytext); addCharNum(yyleng);}
	YY_BREAK
case 16:
YY_RULE_SETUP
#line 62 "lex.l"
{ printf("AND_ASSIGN: %s\n",yytext); addCharNum(yyleng);}
	YY_BREAK
case 17:
YY_RULE_SETUP
#line 63 "lex.l"
{ printf("XOR_ASSIGN: %s\n",yytext); addCharNum(yyleng);}
	YY_BREAK
case 18:
YY_RULE_SETUP
#line 64 "lex.l"
{ printf("OR_ASSIGN: %s\n",yytext); addCharNum(yyleng);}
	YY_BREAK
case 19:
YY_RULE_SETUP
#line 65 "lex.l"
{ printf("RIGHT_OP: %s\n",yytext); addCharNum(yyleng);}
	YY_BREAK
case 20:
YY_RULE_SETUP
#line 66 "lex.l"
{ printf("LEFT_OP: %s\n",yytext); addCharNum(yyleng);}
	YY_BREAK
case 21:
YY_RULE_SETUP
#line 67 "lex.l"
{ printf("INC_OP: %s\n",yytext); addCharNum(yyleng);}
```

```
    YY_BREAK
case 22:
YY_RULE_SETUP
#line 68 "lex.l"
{ printf("DEC_OP: %s\n",yytext); addCharNum(yyleng);}
    YY_BREAK
case 23:
YY_RULE_SETUP
#line 69 "lex.l"
{ printf("PTR_OP: %s\n",yytext); addCharNum(yyleng);}
    YY_BREAK
case 24:
YY_RULE_SETUP
#line 70 "lex.l"
{ printf("AND_OP: %s\n",yytext); addCharNum(yyleng);}
    YY_BREAK
case 25:
YY_RULE_SETUP
#line 71 "lex.l"
{ printf("OR_OP: %s\n",yytext); addCharNum(yyleng);}
    YY_BREAK
case 26:
YY_RULE_SETUP
#line 72 "lex.l"
{ printf("LE_OP: %s\n",yytext); addCharNum(yyleng);}
    YY_BREAK
case 27:
YY_RULE_SETUP
#line 73 "lex.l"
{ printf("GE_OP: %s\n",yytext); addCharNum(yyleng);}
    YY_BREAK
case 28:
YY_RULE_SETUP
#line 74 "lex.l"
{ printf("ARROW: %s\n",yytext); addCharNum(yyleng);}
    YY_BREAK
case 29:
YY_RULE_SETUP
#line 75 "lex.l"
{ printf("EQ_OP: %s\n",yytext); addCharNum(yyleng);}
    YY_BREAK
case 30:
```

```
YY_RULE_SETUP
#line 76 "lex.l"
{ printf("NE_OP: %s\n",yytext); addCharNum(yyleng);}
    YY_BREAK
case 31:
YY_RULE_SETUP
#line 77 "lex.l"
{ printf("'SEMICOLON': %s\n",yytext); addCharNum(yyleng);}
    YY_BREAK
case 32:
YY_RULE_SETUP
#line 78 "lex.l"
{ printf("'L_BRACE': %s\n",yytext); addCharNum(yyleng);}
    YY_BREAK
case 33:
YY_RULE_SETUP
#line 79 "lex.l"
{ printf("'R_BRACE': %s\n",yytext); addCharNum(yyleng);}
    YY_BREAK
case 34:
YY_RULE_SETUP
#line 80 "lex.l"
{ printf("COMMA: %s\n",yytext); addCharNum(yyleng);}
    YY_BREAK
case 35:
YY_RULE_SETUP
#line 81 "lex.l"
{ printf("COLON: %s\n",yytext); addCharNum(yyleng);}
    YY_BREAK
case 36:
YY_RULE_SETUP
#line 82 "lex.l"
{ printf("ASSIGN: %s\n",yytext); addCharNum(yyleng);}
    YY_BREAK
case 37:
YY_RULE_SETUP
#line 83 "lex.l"
{ printf("L_PAREN: %s\n",yytext); addCharNum(yyleng);}
    YY_BREAK
case 38:
YY_RULE_SETUP
#line 84 "lex.l"
```

```
{ printf("R_PAREN: %s\n",yytext); addCharNum(yyleng);}
	YY_BREAK
case 39:
YY_RULE_SETUP
#line 85 "lex.l"
{ printf("L_SQUARE: %s\n",yytext); addCharNum(yyleng);}
	YY_BREAK
case 40:
YY_RULE_SETUP
#line 86 "lex.l"
{ printf("R_SQUARE: %s\n",yytext); addCharNum(yyleng);}
	YY_BREAK
case 41:
YY_RULE_SETUP
#line 87 "lex.l"
{ printf("DOT: %s\n",yytext); addCharNum(yyleng);}
	YY_BREAK
case 42:
YY_RULE_SETUP
#line 88 "lex.l"
{ printf("BIT_AND: %s\n",yytext); addCharNum(yyleng);}
	YY_BREAK
case 43:
YY_RULE_SETUP
#line 89 "lex.l"
{ printf("NOT: %s\n",yytext); addCharNum(yyleng);}
	YY_BREAK
case 44:
YY_RULE_SETUP
#line 90 "lex.l"
{ printf("BIT_NOT: %s\n",yytext); addCharNum(yyleng);}
	YY_BREAK
case 45:
YY_RULE_SETUP
#line 91 "lex.l"
{ printf("SUB: %s\n",yytext); addCharNum(yyleng);}
	YY_BREAK
case 46:
YY_RULE_SETUP
#line 92 "lex.l"
{ printf("ADD: %s\n",yytext); addCharNum(yyleng);}
	YY_BREAK
```

```
case 47:
YY_RULE_SETUP
#line 93 "lex.l"
{ printf("MUL: %s\n",yytext); addCharNum(yyleng);}
    YY_BREAK
case 48:
YY_RULE_SETUP
#line 94 "lex.l"
{ printf("DIV: %s\n",yytext); addCharNum(yyleng);}
    YY_BREAK
case 49:
YY_RULE_SETUP
#line 95 "lex.l"
{ printf("MOD: %s\n",yytext); addCharNum(yyleng);}
    YY_BREAK
case 50:
YY_RULE_SETUP
#line 96 "lex.l"
{ printf("LESS: %s\n",yytext); addCharNum(yyleng);}
    YY_BREAK
case 51:
YY_RULE_SETUP
#line 97 "lex.l"
{ printf("GREATER: %s\n",yytext); addCharNum(yyleng);}
    YY_BREAK
case 52:
YY_RULE_SETUP
#line 98 "lex.l"
{ printf("BIT_XOR: %s\n",yytext); addCharNum(yyleng);}
    YY_BREAK
case 53:
YY_RULE_SETUP
#line 99 "lex.l"
{ printf("BIT_OR: %s\n",yytext); addCharNum(yyleng);}
    YY_BREAK
case 54:
YY_RULE_SETUP
#line 100 "lex.l"
{ printf("QUESTION: %s\n",yytext); addCharNum(yyleng);}
    YY_BREAK
case 55:
YY_RULE_SETUP
```

```
#line 101 "lex.l"
{ printf("WELL: %s\n", yytext); addCharNum(yyleng);}
    YY_BREAK
case 56:
YY_RULE_SETUP
#line 102 "lex.l"
{ lineCount++; addCharNum(yyleng);}
    YY_BREAK
case 57:
YY_RULE_SETUP
#line 103 "lex.l"
{ addCharNum(yyleng);}
    YY_BREAK
case 58:
YY_RULE_SETUP
#line 104 "lex.l"
{
    printf("ERROR: line %d, %s\n",lineCount,yytext);
    errorNum++;
    addCharNum(yyleng);
}
    YY_BREAK
case 59:
YY_RULE_SETUP
#line 109 "lex.l"
ECHO;
    YY_BREAK
#line 1128 "lex.yy.c"
case YY_STATE_EOF(INITIAL):
    yyterminate();

    case YY_END_OF_BUFFER:
        {
        /* Amount of text matched not including the EOB char. */
        int yy_amount_of_matched_text = (int) (yy_cp - yytext_ptr) - 1;

        /* Undo the effects of YY_DO_BEFORE_ACTION. */
        *yy_cp = yy_hold_char;
        YY_RESTORE_YY_MORE_OFFSET

        if ( yy_current_buffer->yy_buffer_status == YY_BUFFER_NEW )
            {
```

```
            /* We're scanning a new file or input source.  It's
             * possible that this happened because the user
             * just pointed yyin at a new source and called
             * yylex().  If so, then we have to assure
             * consistency between yy_current_buffer and our
             * globals.  Here is the right place to do so, because
             * this is the first action (other than possibly a
             * back-up) that will match for the new input source.
             */
            yy_n_chars = yy_current_buffer->yy_n_chars;
            yy_current_buffer->yy_input_file = yyin;
            yy_current_buffer->yy_buffer_status = YY_BUFFER_NORMAL;
            }

        /* Note that here we test for yy_c_buf_p "<=" to the position
         * of the first EOB in the buffer, since yy_c_buf_p will
         * already have been incremented past the NUL character
         * (since all states make transitions on EOB to the
         * end-of-buffer state).  Contrast this with the test
         * in input().
         */
        if ( yy_c_buf_p <= &yy_current_buffer->yy_ch_buf[yy_n_chars] )
            { /* This was really a NUL. */
            yy_state_type yy_next_state;

            yy_c_buf_p = yytext_ptr + yy_amount_of_matched_text;

            yy_current_state = yy_get_previous_state();

            /* Okay, we're now positioned to make the NUL
             * transition.  We couldn't have
             * yy_get_previous_state() go ahead and do it
             * for us because it doesn't know how to deal
             * with the possibility of jamming (and we don't
             * want to build jamming into it because then it
             * will run more slowly).
             */

            yy_next_state = yy_try_NUL_trans( yy_current_state );

            yy_bp = yytext_ptr + YY_MORE_ADJ;
            if ( yy_next_state )
                {
                /* Consume the NUL. */
                yy_cp = ++yy_c_buf_p;
                yy_current_state = yy_next_state;
                goto yy_match;
                }

            else
                {
                yy_cp = yy_c_buf_p;
                goto yy_find_action;
                }
            }

        else switch ( yy_get_next_buffer() )
            {
            case EOB_ACT_END_OF_FILE:
                {
                yy_did_buffer_switch_on_eof = 0;

                if ( yywrap() )
                    {
                    /* Note: because we've taken care in
                     * yy_get_next_buffer() to have set up
                     * yytext, we can now set up
                     * yy_c_buf_p so that if some total
                     * hoser (like flex itself) wants to
                     * call the scanner after we return the
                     * YY_NULL, it'll still work - another
                     * YY_NULL will get returned.
                     */
                    yy_c_buf_p = yytext_ptr + YY_MORE_ADJ;

                    yy_act = YY_STATE_EOF(YY_START);
                    goto do_action;
                    }

                else
                    {
                    if ( ! yy_did_buffer_switch_on_eof )
                        YY_NEW_FILE;
                    }
                break;
                }

            case EOB_ACT_CONTINUE_SCAN:
                yy_c_buf_p =
                    yytext_ptr + yy_amount_of_matched_text;

                yy_current_state = yy_get_previous_state();

                yy_cp = yy_c_buf_p;
                yy_bp = yytext_ptr + YY_MORE_ADJ;
                goto yy_match;

            case EOB_ACT_LAST_MATCH:
                yy_c_buf_p =
                &yy_current_buffer->yy_ch_buf[yy_n_chars];

                yy_current_state = yy_get_previous_state();

                yy_cp = yy_c_buf_p;
                yy_bp = yytext_ptr + YY_MORE_ADJ;
                goto yy_find_action;
            }
        break;
        }

    default:
        YY_FATAL_ERROR(
            "fatal flex scanner internal error--no action found" );
    } /* end of action switch */
        } /* end of scanning one token */
    } /* end of yylex */


 /* yy_get_next_buffer - try to read in a new buffer
  *
  * Returns a code representing an action:
  *  EOB_ACT_LAST_MATCH -
  *  EOB_ACT_CONTINUE_SCAN - continue scanning from current position
  *  EOB_ACT_END_OF_FILE - end of file
  */
```

```
static int yy_get_next_buffer()
    {
    register char *dest = yy_current_buffer->yy_ch_buf;
    register char *source = yytext_ptr;
    register int number_to_move, i;
    int ret_val;

    if ( yy_c_buf_p > &yy_current_buffer->yy_ch_buf[yy_n_chars + 1] )
        YY_FATAL_ERROR(
        "fatal flex scanner internal error--end of buffer missed" );

    if ( yy_current_buffer->yy_fill_buffer == 0 )
        { /* Don't try to fill the buffer, so this is an EOF. */
        if ( yy_c_buf_p - yytext_ptr - YY_MORE_ADJ == 1 )
            {
            /* We matched a single character, the EOB, so
             * treat this as a final EOF.
             */
            return EOB_ACT_END_OF_FILE;
            }

        else
            {
            /* We matched some text prior to the EOB, first
             * process it.
             */
            return EOB_ACT_LAST_MATCH;
            }
        }

    /* Try to read more data. */

    /* First move last chars to start of buffer. */
    number_to_move = (int) (yy_c_buf_p - yytext_ptr) - 1;

    for ( i = 0; i < number_to_move; ++i )
        *(dest++) = *(source++);

    if ( yy_current_buffer->yy_buffer_status == YY_BUFFER_EOF_PENDING )
        /* don't do the read, it's not guaranteed to return an EOF,
         * just force an EOF
         */
        yy_current_buffer->yy_n_chars = yy_n_chars = 0;

    else
        {
        int num_to_read =
            yy_current_buffer->yy_buf_size - number_to_move - 1;

        while ( num_to_read <= 0 )
            { /* Not enough room in the buffer - grow it. */
#ifdef YY_USES_REJECT
            YY_FATAL_ERROR(
"input buffer overflow, can't enlarge buffer because scanner uses
REJECT" );
#else

            /* just a shorter name for the current buffer */
            YY_BUFFER_STATE b = yy_current_buffer;

            int yy_c_buf_p_offset =
                (int) (yy_c_buf_p - b->yy_ch_buf);

            if ( b->yy_is_our_buffer )
                {
                int new_size = b->yy_buf_size * 2;

                if ( new_size <= 0 )
                    b->yy_buf_size += b->yy_buf_size / 8;
                else
                    b->yy_buf_size *= 2;

                b->yy_ch_buf = (char *)
                    /* Include room in for 2 EOB chars. */
                    yy_flex_realloc( (void *) b->yy_ch_buf,
                             b->yy_buf_size + 2 );
                }
            else
                /* Can't grow it, we don't own it. */
                b->yy_ch_buf = 0;

            if ( ! b->yy_ch_buf )
                YY_FATAL_ERROR(
                "fatal error - scanner input buffer overflow" );
            yy_c_buf_p = &b->yy_ch_buf[yy_c_buf_p_offset];

            num_to_read = yy_current_buffer->yy_buf_size -
                        number_to_move - 1;
#endif
            }

        if ( num_to_read > YY_READ_BUF_SIZE )
            num_to_read = YY_READ_BUF_SIZE;

        /* Read in more data. */
        YY_INPUT( (&yy_current_buffer->yy_ch_buf[number_to_move]),
            yy_n_chars, num_to_read );

        yy_current_buffer->yy_n_chars = yy_n_chars;
        }

    if ( yy_n_chars == 0 )
        {
        if ( number_to_move == YY_MORE_ADJ )
            {
            ret_val = EOB_ACT_END_OF_FILE;
            yyrestart( yyin );
            }

        else
            {
            ret_val = EOB_ACT_LAST_MATCH;
            yy_current_buffer->yy_buffer_status =
                YY_BUFFER_EOF_PENDING;
            }
        }

    else
        ret_val = EOB_ACT_CONTINUE_SCAN;

    yy_n_chars += number_to_move;
    yy_current_buffer->yy_ch_buf[yy_n_chars] = YY_END_OF_BUFFER_CHAR;
    yy_current_buffer->yy_ch_buf[yy_n_chars + 1] =
YY_END_OF_BUFFER_CHAR;
```

```
    yytext_ptr = &yy_current_buffer->yy_ch_buf[0];

    return ret_val;
    }


/* yy_get_previous_state - get the state just before the EOB char was
reached */

static yy_state_type yy_get_previous_state()
    {
    register yy_state_type yy_current_state;
    register char *yy_cp;

    yy_current_state = yy_start;

    for ( yy_cp = yytext_ptr + YY_MORE_ADJ; yy_cp < yy_c_buf_p;
++yy_cp )
        {
        register YY_CHAR yy_c = (*yy_cp ? yy_ec[YY_SC_TO_UI(*yy_cp)] :
1);
        if ( yy_accept[yy_current_state] )
            {
            yy_last_accepting_state = yy_current_state;
            yy_last_accepting_cpos = yy_cp;
            }
        while ( yy_chk[yy_base[yy_current_state] + yy_c] !=
yy_current_state )
            {
            yy_current_state = (int) yy_def[yy_current_state];
            if ( yy_current_state >= 192 )
                yy_c = yy_meta[(unsigned int) yy_c];
            }
        yy_current_state = yy_nxt[yy_base[yy_current_state] + (unsigned
int) yy_c];
        }

    return yy_current_state;
    }


/* yy_try_NUL_trans - try to make a transition on the NUL character
 *
 * synopsis
 *  next_state = yy_try_NUL_trans( current_state );
 */

#ifdef YY_USE_PROTOS
static yy_state_type yy_try_NUL_trans( yy_state_type yy_current_state )
#else
static yy_state_type yy_try_NUL_trans( yy_current_state )
yy_state_type yy_current_state;
#endif
    {
    register int yy_is_jam;
    register char *yy_cp = yy_c_buf_p;

    register YY_CHAR yy_c = 1;
    if ( yy_accept[yy_current_state] )
        {
        yy_last_accepting_state = yy_current_state;
        yy_last_accepting_cpos = yy_cp;
        }
    while ( yy_chk[yy_base[yy_current_state] + yy_c] !=
yy_current_state )
        {
        yy_current_state = (int) yy_def[yy_current_state];
        if ( yy_current_state >= 192 )
            yy_c = yy_meta[(unsigned int) yy_c];
        }
    yy_current_state = yy_nxt[yy_base[yy_current_state] + (unsigned int)
yy_c];
    yy_is_jam = (yy_current_state == 191);

    return yy_is_jam ? 0 : yy_current_state;
    }


#ifndef YY_NO_UNPUT
#ifdef YY_USE_PROTOS
static void yyunput( int c, register char *yy_bp )
#else
static void yyunput( c, yy_bp )
int c;
register char *yy_bp;
#endif
```

```
    {
    register char *yy_cp = yy_c_buf_p;

    /* undo effects of setting up yytext */
    *yy_cp = yy_hold_char;

    if ( yy_cp < yy_current_buffer->yy_ch_buf + 2 )
        { /* need to shift things up to make room */
        /* +2 for EOB chars. */
        register int number_to_move = yy_n_chars + 2;
        register char *dest = &yy_current_buffer->yy_ch_buf[
                    yy_current_buffer->yy_buf_size + 2];
        register char *source =
                &yy_current_buffer->yy_ch_buf[number_to_move];

        while ( source > yy_current_buffer->yy_ch_buf )
            *--dest = *--source;

        yy_cp += (int) (dest - source);
        yy_bp += (int) (dest - source);
        yy_current_buffer->yy_n_chars =
            yy_n_chars = yy_current_buffer->yy_buf_size;

        if ( yy_cp < yy_current_buffer->yy_ch_buf + 2 )
            YY_FATAL_ERROR( "flex scanner push-back overflow" );
        }

    *--yy_cp = (char) c;


    yytext_ptr = yy_bp;
    yy_hold_char = *yy_cp;
    yy_c_buf_p = yy_cp;
    }
#endif  /* ifndef YY_NO_UNPUT */


#ifdef __cplusplus
static int yyinput()
#else
static int input()
#endif
    {
```

```
    int c;

    *yy_c_buf_p = yy_hold_char;

    if ( *yy_c_buf_p == YY_END_OF_BUFFER_CHAR )
        {
        /* yy_c_buf_p now points to the character we want to return.
         * If this occurs *before* the EOB characters, then it's a
         * valid NUL; if not, then we've hit the end of the buffer.
         */
        if ( yy_c_buf_p < &yy_current_buffer->yy_ch_buf[yy_n_chars] )
            /* This was really a NUL. */
            *yy_c_buf_p = '\0';

        else
            { /* need more input */
            int offset = yy_c_buf_p - yytext_ptr;
            ++yy_c_buf_p;

            switch ( yy_get_next_buffer() )
                {
                case EOB_ACT_LAST_MATCH:
                    /* This happens because yy_g_n_b()
                     * sees that we've accumulated a
                     * token and flags that we need to
                     * try matching the token before
                     * proceeding.  But for input(),
                     * there's no matching to consider.
                     * So convert the EOB_ACT_LAST_MATCH
                     * to EOB_ACT_END_OF_FILE.
                     */

                    /* Reset buffer status. */
                    yyrestart( yyin );

                    /* fall through */

                case EOB_ACT_END_OF_FILE:
                    {
                    if ( yywrap() )
                        return EOF;
                    if ( ! yy_did_buffer_switch_on_eof )
                        YY_NEW_FILE;
#ifdef __cplusplus
                    return yyinput();
#else
                    return input();
#endif
                    }

                case EOB_ACT_CONTINUE_SCAN:
                    yy_c_buf_p = yytext_ptr + offset;
                    break;
                }
            }
        }

    c = *(unsigned char *) yy_c_buf_p;  /* cast for 8-bit char's */
    *yy_c_buf_p = '\0'; /* preserve yytext */
    yy_hold_char = *++yy_c_buf_p;


    return c;
    }


#ifdef YY_USE_PROTOS
void yyrestart( FILE *input_file )
#else
void yyrestart( input_file )
FILE *input_file;
#endif
    {
    if ( ! yy_current_buffer )
        yy_current_buffer = yy_create_buffer( yyin, YY_BUF_SIZE );

    yy_init_buffer( yy_current_buffer, input_file );
    yy_load_buffer_state();
    }


#ifdef YY_USE_PROTOS
void yy_switch_to_buffer( YY_BUFFER_STATE new_buffer )
#else
```

```
void yy_switch_to_buffer( new_buffer )
YY_BUFFER_STATE new_buffer;
#endif
    {
    if ( yy_current_buffer == new_buffer )
        return;

    if ( yy_current_buffer )
        {
        /* Flush out information for old buffer. */
        *yy_c_buf_p = yy_hold_char;
        yy_current_buffer->yy_buf_pos = yy_c_buf_p;
        yy_current_buffer->yy_n_chars = yy_n_chars;
        }

    yy_current_buffer = new_buffer;
    yy_load_buffer_state();

    /* We don't actually know whether we did this switch during
     * EOF (yywrap()) processing, but the only time this flag
     * is looked at is after yywrap() is called, so it's safe
     * to go ahead and always set it.
     */
    yy_did_buffer_switch_on_eof = 1;
    }


#ifdef YY_USE_PROTOS
void yy_load_buffer_state( void )
#else
void yy_load_buffer_state()
#endif
    {
    yy_n_chars = yy_current_buffer->yy_n_chars;
    yytext_ptr = yy_c_buf_p = yy_current_buffer->yy_buf_pos;
    yyin = yy_current_buffer->yy_input_file;
    yy_hold_char = *yy_c_buf_p;
    }


#ifdef YY_USE_PROTOS
YY_BUFFER_STATE yy_create_buffer( FILE *file, int size )
#else
```

```
YY_BUFFER_STATE yy_create_buffer( file, size )
FILE *file;
int size;
#endif
    {
    YY_BUFFER_STATE b;

    b = (YY_BUFFER_STATE) yy_flex_alloc( sizeof( struct
yy_buffer_state ) );
    if ( ! b )
        YY_FATAL_ERROR( "out of dynamic memory in yy_create_buffer()" );

    b->yy_buf_size = size;

    /* yy_ch_buf has to be 2 characters longer than the size given
because
     * we need to put in 2 end-of-buffer characters.
     */
    b->yy_ch_buf = (char *) yy_flex_alloc( b->yy_buf_size + 2 );
    if ( ! b->yy_ch_buf )
        YY_FATAL_ERROR( "out of dynamic memory in yy_create_buffer()" );

    b->yy_is_our_buffer = 1;

    yy_init_buffer( b, file );

    return b;
    }


#ifdef YY_USE_PROTOS
void yy_delete_buffer( YY_BUFFER_STATE b )
#else
void yy_delete_buffer( b )
YY_BUFFER_STATE b;
#endif
    {
    if ( ! b )
        return;

    if ( b == yy_current_buffer )
        yy_current_buffer = (YY_BUFFER_STATE) 0;
    if ( b->yy_is_our_buffer )
        yy_flex_free( (void *) b->yy_ch_buf );

    yy_flex_free( (void *) b );
    }


#ifndef YY_ALWAYS_INTERACTIVE
#ifndef YY_NEVER_INTERACTIVE
extern int isatty YY_PROTO(( int ));
#endif
#endif

#ifdef YY_USE_PROTOS
void yy_init_buffer( YY_BUFFER_STATE b, FILE *file )
#else
void yy_init_buffer( b, file )
YY_BUFFER_STATE b;
FILE *file;
#endif


    {
    yy_flush_buffer( b );

    b->yy_input_file = file;
    b->yy_fill_buffer = 1;

#if YY_ALWAYS_INTERACTIVE
    b->yy_is_interactive = 1;
#else
#if YY_NEVER_INTERACTIVE
    b->yy_is_interactive = 0;
#else
    b->yy_is_interactive = file ? (isatty( fileno(file) ) > 0) : 0;
#endif
#endif
    }


#ifdef YY_USE_PROTOS
void yy_flush_buffer( YY_BUFFER_STATE b )
#else
```

```
void yy_flush_buffer( b )
YY_BUFFER_STATE b;
#endif

    {
    if ( ! b )
        return;

    b->yy_n_chars = 0;

    /* We always need two end-of-buffer characters.  The first causes
     * a transition to the end-of-buffer state.  The second causes
     * a jam in that state.
     */
    b->yy_ch_buf[0] = YY_END_OF_BUFFER_CHAR;
    b->yy_ch_buf[1] = YY_END_OF_BUFFER_CHAR;

    b->yy_buf_pos = &b->yy_ch_buf[0];

    b->yy_at_bol = 1;
    b->yy_buffer_status = YY_BUFFER_NEW;

    if ( b == yy_current_buffer )
        yy_load_buffer_state();
    }


#ifndef YY_NO_SCAN_BUFFER
#ifdef YY_USE_PROTOS
YY_BUFFER_STATE yy_scan_buffer( char *base, yy_size_t size )
#else
YY_BUFFER_STATE yy_scan_buffer( base, size )
char *base;
yy_size_t size;
#endif
    {
    YY_BUFFER_STATE b;

    if ( size < 2 ||
         base[size-2] != YY_END_OF_BUFFER_CHAR ||
         base[size-1] != YY_END_OF_BUFFER_CHAR )
        /* They forgot to leave room for the EOB's. */
        return 0;
    b = (YY_BUFFER_STATE) yy_flex_alloc( sizeof( struct
yy_buffer_state ) );
    if ( ! b )
        YY_FATAL_ERROR( "out of dynamic memory in yy_scan_buffer()" );

    b->yy_buf_size = size - 2;  /* "- 2" to take care of EOB's */
    b->yy_buf_pos = b->yy_ch_buf = base;
    b->yy_is_our_buffer = 0;
    b->yy_input_file = 0;
    b->yy_n_chars = b->yy_buf_size;
    b->yy_is_interactive = 0;
    b->yy_at_bol = 1;
    b->yy_fill_buffer = 0;
    b->yy_buffer_status = YY_BUFFER_NEW;

    yy_switch_to_buffer( b );

    return b;
    }
#endif


#ifndef YY_NO_SCAN_STRING
#ifdef YY_USE_PROTOS
YY_BUFFER_STATE yy_scan_string( yyconst char *yy_str )
#else
YY_BUFFER_STATE yy_scan_string( yy_str )
yyconst char *yy_str;
#endif
    {
    int len;
    for ( len = 0; yy_str[len]; ++len )
        ;

    return yy_scan_bytes( yy_str, len );
    }
#endif


#ifndef YY_NO_SCAN_BYTES
#ifdef YY_USE_PROTOS
YY_BUFFER_STATE yy_scan_bytes( yyconst char *bytes, int len )
```

```
#else
YY_BUFFER_STATE yy_scan_bytes( bytes, len )
yyconst char *bytes;
int len;
#endif
    {
    YY_BUFFER_STATE b;
    char *buf;
    yy_size_t n;
    int i;

    /* Get memory for full buffer, including space for trailing EOB's.
*/
    n = len + 2;
    buf = (char *) yy_flex_alloc( n );
    if ( ! buf )
        YY_FATAL_ERROR( "out of dynamic memory in yy_scan_bytes()" );

    for ( i = 0; i < len; ++i )
        buf[i] = bytes[i];

    buf[len] = buf[len+1] = YY_END_OF_BUFFER_CHAR;

    b = yy_scan_buffer( buf, n );
    if ( ! b )
        YY_FATAL_ERROR( "bad buffer in yy_scan_bytes()" );

    /* It's okay to grow etc. this buffer, and we should throw it
     * away when we're done.
     */
    b->yy_is_our_buffer = 1;

    return b;
    }
#endif


#ifndef YY_NO_PUSH_STATE
#ifdef YY_USE_PROTOS
static void yy_push_state( int new_state )
#else
static void yy_push_state( new_state )
int new_state;
```

```
#endif
    {
    if ( yy_start_stack_ptr >= yy_start_stack_depth )
        {
        yy_size_t new_size;

        yy_start_stack_depth += YY_START_STACK_INCR;
        new_size = yy_start_stack_depth * sizeof( int );

        if ( ! yy_start_stack )
            yy_start_stack = (int *) yy_flex_alloc( new_size );

        else
            yy_start_stack = (int *) yy_flex_realloc(
                    (void *) yy_start_stack, new_size );

        if ( ! yy_start_stack )
            YY_FATAL_ERROR(
            "out of memory expanding start-condition stack" );
        }

    yy_start_stack[yy_start_stack_ptr++] = YY_START;

    BEGIN(new_state);
    }
#endif


#ifndef YY_NO_POP_STATE
static void yy_pop_state()
    {
    if ( --yy_start_stack_ptr < 0 )
        YY_FATAL_ERROR( "start-condition stack underflow" );

    BEGIN(yy_start_stack[yy_start_stack_ptr]);
    }
#endif


#ifndef YY_NO_TOP_STATE
static int yy_top_state()
    {
    return yy_start_stack[yy_start_stack_ptr - 1];
    }
#endif

#ifndef YY_EXIT_FAILURE
#define YY_EXIT_FAILURE 2
#endif

#ifdef YY_USE_PROTOS
static void yy_fatal_error( yyconst char msg[] )
#else
static void yy_fatal_error( msg )
char msg[];
#endif
    {
    (void) fprintf( stderr, "%s\n", msg );
    exit( YY_EXIT_FAILURE );
    }



/* Redefine yyless() so it works in section 3 code. */

#undef yyless
#define yyless(n) \
    do \
        { \
        /* Undo effects of setting up yytext. */ \
        yytext[yyleng] = yy_hold_char; \
        yy_c_buf_p = yytext + n; \
        yy_hold_char = *yy_c_buf_p; \
        *yy_c_buf_p = '\0'; \
        yyleng = n; \
        } \
    while ( 0 )


/* Internal utility routines. */

#ifndef yytext_ptr
#ifdef YY_USE_PROTOS
static void yy_flex_strncpy( char *s1, yyconst char *s2, int n )
#else
static void yy_flex_strncpy( s1, s2, n )
```

```
char *s1;
yyconst char *s2;
int n;
#endif
    {
    register int i;
    for ( i = 0; i < n; ++i )
        s1[i] = s2[i];
    }
#endif

#ifdef YY_NEED_STRLEN
#ifdef YY_USE_PROTOS
static int yy_flex_strlen( yyconst char *s )
#else
static int yy_flex_strlen( s )
yyconst char *s;
#endif
    {
    register int n;
    for ( n = 0; s[n]; ++n )
        ;

    return n;
    }
#endif


#ifdef YY_USE_PROTOS
static void *yy_flex_alloc( yy_size_t size )
#else
static void *yy_flex_alloc( size )
yy_size_t size;
#endif
    {
    return (void *) malloc( size );
    }

#ifdef YY_USE_PROTOS
static void *yy_flex_realloc( void *ptr, yy_size_t size )
#else
static void *yy_flex_realloc( ptr, size )
void *ptr;
```

```
yy_size_t size;
#endif
    {
    /* The cast to (char *) in the following accommodates both
     * implementations that use char* generic pointers, and those
     * that use void* generic pointers.  It works with the latter
     * because both ANSI C and C++ allow castless assignment from
     * any pointer type to void*, and deal with argument conversions
     * as though doing an assignment.
     */
    return (void *) realloc( (char *) ptr, size );
    }

#ifdef YY_USE_PROTOS
static void yy_flex_free( void *ptr )
#else
static void yy_flex_free( ptr )
void *ptr;
#endif
    {
    free( ptr );
    }

#if YY_MAIN
int main()
    {
    yylex();
    return 0;
    }
#endif
#line 109 "lex.l"


int main(int argc,char **argv){
    if(argc>1) yyin=fopen(argv[1],"r");
    else printf("error:\n command: lexC filename");
    yylex();
    return 0;
}
int yywrap(){
    printCompilerResult();
    return 1;
}
```

```c
void jumpMultiComment(void){
    char c, prev = 0;

    while ((c = input()) != 0)      /* (EOF maps to 0) */
    {
        if(c == '\n'){
            lineCount++;
        }
        if (c == '/' && prev == '*')
            return;
        prev = c;
    }
}
void jumpSingleComment(void){
    char c;
    while((c = input()) != 0){
        charNum++;
        if(c == '\n'){
            lineCount++;
            return;
        }
    }
}
void printCompilerResult(void){
    printf("====================================================\n");
    printf("         Compiler finished, the result are:        \n");
    printf("All char number: %d\n", charNum);
    printf("Line count: %d\n", lineCount);
    printf("Ketword number: %d\n", keywordNum);
    printf("Identifier number: %d\n", idNum);
    printf("Integer number: %d\n", intNum);
    printf("Floate number: %d\n", floatNum);
    printf("Error number: %d\n", errorNum);
}
void addCharNum(int length){
    charNum += length;
}
```