

北京邮电大学

yacc 语法分析程序

编译原理

姓名：陈朴炎 学号：2021211138

2023-11-22

目录

1 概述	2
1.1 问题描述	2
1.2 实现方法	2
2 实验环境	2
3 语法分析程序 lr1.y	3
3.1 语法分析思路	3
3.2 语法程序源代码	5
4 词法分析程序 lr1.l	7
4.1 词法分析思路	7
4.2 词法程序源代码	9
5 实验步骤	10
6 测试	11
6.1 测试用例 1	11
6.2 测试用例 2	12
6.3 测试用例 3	12
6.4 测试用例 4	13
7 附录—完整项目代码	14
7.1 lr1.y	14
7.2 lr1.l	15
7.3 lr1.tab.h	16
7.4 lr1.tab.c	17
7.5 lex.yy.c	55

1 概述

1.1 问题描述

语法分析程序的设计与实现：

编写语法分析程序，实现对算术表达式的语法分析。要求所分析算数表达式由如下的文法产生。

$$E \rightarrow E + T \mid E - T \mid T$$

$$T \rightarrow T * F \mid T / F \mid F$$

$$F \rightarrow (E) \mid \text{num}$$

图 1-1 文法示意图

1.2 实现方法

下述两种方法二选一

方法 3:编写语法分析程序实现自底向上的分析,要求如下

- (1)构造识别该文法所有活前缀的 DFA。
- (2)构造该文法的 LR 分析表。
- (3)编程实现算法 4.3,构造 LR 分析程序。

方法 4:利用 YACC 自动生成语法分析程序,调用 LEX 自动生成的词法分析程序

2 实验环境

在 windows11 下, gcc、flex、bison 版本如下:

gcc (x86_64-posix-seh-rev0, Built by MinGW-W64 project) 8.1.0

flex version 2.5.4
bison (GNU Bison) 2.4.1

3 语法分析程序 lr1.y

3.1 语法分析思路

语法程序是通过 bison 编译的，bison 的程序中有三个部分：

1. 头部
2. 语法规则
3. 主函数和错误处理

在这三个部分中，每个部分中间都用双百分号 %% 隔开。

在头部中，一般是用来给 c 文件添加头文件和定义结构体、全局变量和宏，或者是给语法程序规定终结符号。观察此题中的文法规则，我们可以得到终结符有：=、-、*、/、数字。因此我们可以定义出这些记号：

```
%token T_PLUS T_MINUS T_MUL T_DIV T_LP T_RP T_NUM
```

T_PLUS 表示加号

T_MINUS 表示减号

T_MUL 表示乘号

T_DIV 表示除号

T_LP 表示左括号

T_RP 表示右括号

T_NUM 表示数字

定义好记号流后，就可以开始定义语法规则。

在 bison 中，\$ 会自动匹配上终结符号，通过\$1 可以匹配上左边式子中的

第一个符号, 比如

```
1. term T_MUL factor {
2.     printf("reduced by : T -> T * F\n", $1, $3);
3.     $$ = $1 * $3;
4. }
```

对于 term T_MUL factor 来说, term 是第一个符号, T_MUL 是第二个符号, factor 是第三个符号, 因此可以使用 \$1 来匹配 term, \$3 来匹配 factor。

对于 $E \rightarrow E+T \mid E-T \mid T$, 我们可以规定如下规则:

```
1. expression:
2.     expression T_PLUS term {
3.         printf("reduced by : E -> E + T \n");
4.         $$ = $1 + $3;
5.     }
6. | expression T_MINUS term {
7.     printf("reduced by : E -> E - T \n");
8.     $$ = $1 - $3;
9. }
10. | term {
11.     printf("reduced by : E -> T \n");
12.     $$ = $1;
13. }
14. ;
15.
```

这表示在规约到 $E \rightarrow E + T$ 时, 我们将打印出 $E \rightarrow E + T$, 并将规范到的数值赋给记号。同理, 对于规约到 $E - T$ 或者是 $E \rightarrow T$ 时, 我们也会提示出语法规则的规约过程。

对于 $T \rightarrow T * F \mid T / F \mid F$, 我们可以规定如下规则:

```
1. term:
2.     term T_MUL factor {
3.         printf("reduced by : T -> T * F\n");
4.         $$ = $1 * $3;
5.     }
6. | term T_DIV factor {
7.     printf("reduced by : T -> T / F\n");
```

```

8.      $$ = $1 / $3;
9.    }
10.   | factor      {
11.       printf("reduced by : T -> F\n");
12.       $$ = $1;
13.   }
14.   ;
15.

```

而对于 $F \rightarrow (E) \mid \text{num}$ ，我们可以制定如下规则：

```

1. factor:
2.   T_LP expression T_RP {
3.       printf("reduced by : F -> ( E )\n");
4.       $$ = $2;
5.   }
6.   | T_NUM              {
7.       printf("reduced by : F -> NUM \n");
8.       $$ = $1;
9.   }
10.  ;

```

第二部分结束后，可以在主函数中加入错误处理，如下所示：

```

1.  %%
2.
3.  int main() {
4.      yyparse();
5.      return 0;
6.  }
7.
8.  int yyerror(const char *msg) {
9.      fprintf(stderr, "Error: %s\n", msg);
10.     fprintf(stderr, "Token: %d\n", yylex());
11.     return 1;
12. }

```

3.2 语法程序源代码

```

1. %{
2. #include <stdio.h>
3. #include <stdlib.h>
4. #include <math.h>
5.
6. %}

```

```

7.
8. %token T_PLUS T_MINUS T_MUL T_DIV T_LP T_RP T_NUM
9.
10. %%
11.
12. expression:
13.     expression T_PLUS term {
14.         printf("reduced by : E -> E + T \n");
15.         $$ = $1 + $3;
16.     }
17. | expression T_MINUS term {
18.     printf("reduced by : E -> E - T \n");
19.     $$ = $1 - $3;
20. }
21. | term {
22.     printf("reduced by : E -> T \n");
23.     $$ = $1;
24. }
25. ;
26.
27. term:
28.     term T_MUL factor {
29.         printf("reduced by : T -> T * F\n");
30.         $$ = $1 * $3;
31.     }
32. | term T_DIV factor {
33.         printf("reduced by : T -> T / F\n");
34.         $$ = $1 / $3;
35.     }
36. | factor {
37.         printf("reduced by : T -> F\n");
38.         $$ = $1;
39.     }
40. ;
41.
42. factor:
43.     T_LP expression T_RP {
44.         printf("reduced by : F -> ( E )\n");
45.         $$ = $2;
46.     }
47. | T_NUM {
48.         printf("reduced by : F -> NUM \n");

```

```

49.         $$ = $1;
50.     }
51. ;
52.
53. %%
54.
55. int main() {
56.     yyparse();
57.     return 0;
58. }
59.
60. int yyerror(const char *msg) {
61.     fprintf(stderr, "Error: %s\n", msg);
62.     fprintf(stderr, "Token: %d\n", yylex());
63.     return 1;
64. }

```

4 词法分析程序 lr1.1

4.1 词法分析思路

flex 的程序也是由三部分构成，首先在头部，我们要将 yacc 编译后的头文件添加进去，如下所示

```

1. %{
2. #include "lr1.tab.h" // .tab.h 是 yacc 编译后的头文件
3. #include <stdlib.h>
4. %}

```

之后可以制定语法规则。对于输入的每一个字符，我们会进行比较

如果时加号 "+", 则会返回一个 T_PLUS 的类型; 如果时减号 "-", 会返回一个 T_MINUS 的类型; 如果是一个乘号 "*", 则会返回 T_MUL 的类型; 如果是除号 "/" 则会返回 T_DIV 类型; 如果是左括号 "(" 则会返回 T_LP 类型; 如果是右括号 ")", 就返回 T_RP 类型。

```

1. "+" { return T_PLUS; }
2. "-" { return T_MINUS; }

```



```

3.  "*"    { return T_MUL; }
4.  "/"    { return T_DIV; }
5.  "("    { return T_LP; }
6.  ")"    { return T_RP; }

```

对于数字，我们要判断的是：

1. 浮点数
2. 整数
3. 科学计数法表示的数字

相关的正则表达式和语法规则可以写成如下形式：

```

1.  [0-9]+("[0-9]+([eE][-+]?[0-9]+)?)? {
2.      // sscanf(yytext, "%lf", &yylval.float_val);
3.      yyval = atof(yytext); // 将数字转换，使用科学计数法、浮点数形式
4.      return T_NUM;
5.  }
6.
7.  [0-9]+([eE][-+]?[0-9]+)? {
8.      // sscanf(yytext, "%lf", &yylval.float_val);
9.      yyval = atof(yytext); // 同理，这个是不含小数点的科学计数法
10.     return T_NUM;
11. }

```

这里表示当遇到一个数字时，我会将 `yytext` 的内容转成一个浮点数，并赋值给 `yylval`，然后再返回 `T_NUM` 类型。

同时，对于空格，我们是不管的，只需写入：

```
[ \t\n] ; // 跳过空格
```

而对于其他的记号，文法并不能识别，因此要做一个错误处理，当出现其他文法符号时，应当退出程序，如下

```

1.  . {
2.      fprintf(stderr, "Error: Unrecognized character %s\n", yytext);
3.      exit(EXIT_FAILURE);
4.  }

```

在第三部分，只需将该此法程序包裹成函数即可

```

1.  int yywrap() {
2.      return 1;
3.  }

```

4.2 词法程序源代码

```

1.  %{
2.  #include "lr1.tab.h" // .tab.h 是 yacc 编译后的头文件
3.  #include <stdlib.h>
4.  %}
5.
6.  %%
7.
8.  "+" { return T_PLUS; }
9.  "-" { return T_MINUS; }
10. "*" { return T_MUL; }
11. "/" { return T_DIV; }
12. "(" { return T_LP; }
13. ")" { return T_RP; }
14.
15. [0-9]+ "." [0-9]+ ([eE] [-+]? [0-9]+)? {
16.     // sscanf(yytext, "%lf", &yyval.float_val);
17.     yyval = atof(yytext); // 将数字转换, 使用科学计数法、浮点数形式
18.     return T_NUM;
19. }
20.
21. [0-9]+ ([eE] [-+]? [0-9]+)? {
22.     // sscanf(yytext, "%lf", &yyval.float_val);
23.     yyval = atof(yytext); // 同理, 这个是不含小数点的科学计数法
24.     return T_NUM;
25. }
26.
27. [ \t\n] ; // 跳过空格
28.
29. . {
30.     fprintf(stderr, "Error: Unrecognized character %s\n", yytext);
31.     exit(EXIT_FAILURE);
32. }
33.
34. %%
35.
36. int yywrap() {
37.     return 1;

```

```
38. }
```

5 实验步骤

步骤 1. 在项目文件夹下打开 cmd

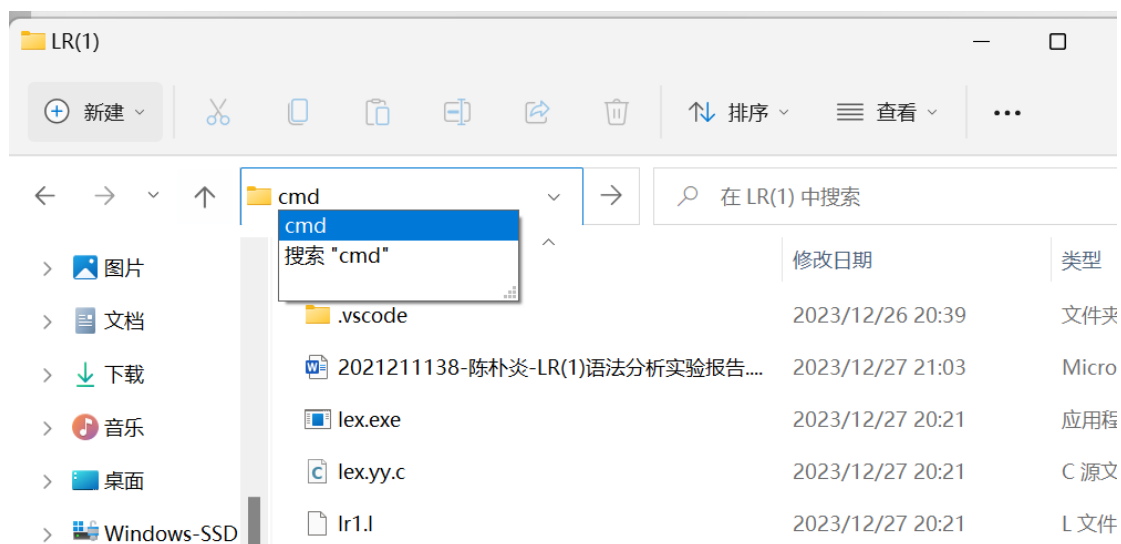


图 5-1 在项目文件夹下打开 cmd

步骤 2. 编译词法程序

```
E:\bupt-homework\compiler_principle\LR(1)>flex lr1.l
```

图 5-2 编译词法程序

输入 flex <词法程序名称> 来编译，没报错就说明可以了。

步骤 3. 编译语法程序

```
E:\bupt-homework\compiler_principle\LR(1)>bison -d lr1.y
```

图 5-3 编译词法程序

输入 bison -d <语法程序名称>来编译，没报错就可以了。

这时候，文件夹中会多出几个文件：

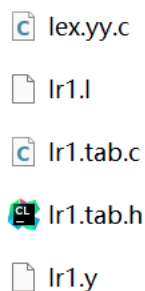


图 5-4 自动生成 C 语言文件

它们是最后要编译的 C 语言源程序。

程序在 lex.yy.c 和 lr1.tab.c 中,通过 gcc 命令编译这两个程序,如下:

```
E:\bupt-homework\compiler_principle\LR(1)>gcc -o lex lex.yy.c lr1.tab.c
lr1.tab.c: In function 'yyparse':
lr1.tab.c:592:16: warning: implicit declaration of function 'yylex' [-Wimplicit-function-declaration]
# define YYLEX yylex ()
                  ^~~~~~
lr1.tab.c:1237:16: note: in expansion of macro 'YYLEX'
yychar = YYLEX;
                  ^~~~~~
lr1.tab.c:1407:7: warning: implicit declaration of function 'yyerror'; did you mean 'yyerrok'? [-Wimplicit-function-declaration]
yyerror (YY_("syntax error"));
          ^~~~~~
          yyerrok
```

图 5-5 编译 C 语言源程序

没有报错,只有警告,说明编译成功,此时会生成一个可执行文件 lex.exe,

在 cmd 中输入 lex, 来执行该文件, 就可以启动实验。

6 测试

6.1 测试用例 1

输入 5*6, 查看语法分析过程, 如下:

```
E:\bupt-homework\compiler_principle\LR(1)>lex
5*6
reduced by : F -> NUM
reduced by : T -> F
reduced by : F -> NUM
reduced by : T -> T * F
reduced by : E -> T
```

图 6-1 执行结果 1

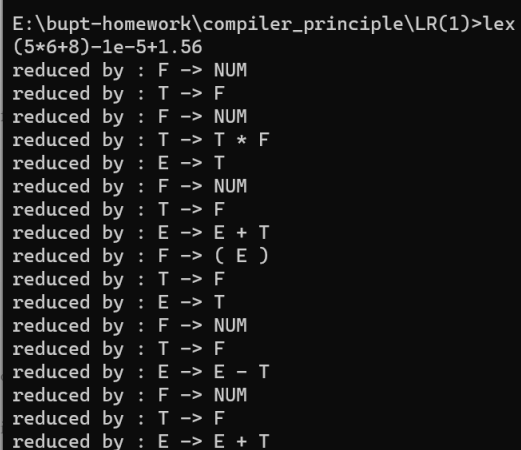
可以看到，语法分析程序规约过程为：

```
5 : F -> NUM
F : T -> F
6 : F -> NUM
T * F : T -> T * F
T : E -> T
```

6.2 测试用例 2

输入：(5*6+8)-1e-5+1.56

执行结果如下：



```
E:\bupt-homework\compiler_principle\LR(1)>lex
(5*6+8)-1e-5+1.56
reduced by : F -> NUM
reduced by : T -> F
reduced by : F -> NUM
reduced by : T -> T * F
reduced by : E -> T
reduced by : F -> NUM
reduced by : T -> F
reduced by : E -> E + T
reduced by : F -> ( E )
reduced by : T -> F
reduced by : E -> T
reduced by : F -> NUM
reduced by : T -> F
reduced by : E -> E - T
reduced by : F -> NUM
reduced by : T -> F
reduced by : E -> E + T
```

图 6-2 执行结果 2

规约过程如上所示。

6.3 测试用例 3

输入：789*(1.5+654)/789

执行结果如下：

```

E:\bupt-homework\compiler_principle\LR(1)>lex
789*(1.5+654)/789
reduced by : F -> NUM
reduced by : T -> F
reduced by : F -> NUM
reduced by : T -> F
reduced by : E -> T
reduced by : F -> NUM
reduced by : T -> F
reduced by : E -> E + T
reduced by : F -> ( E )
reduced by : T -> T * F
reduced by : F -> NUM
reduced by : T -> T / F
reduced by : E -> T

```

图 6-3 执行结果 3

6.4 测试用例 4

输入: $1e-5 + 1e-6 * 1e+10 / 2 - 6$

执行结果如下:

```

E:\bupt-homework\compiler_principle\LR(1)>lex
1e-5 + 1e-6 * 1e+10 / 2 - 6
reduced by : F -> NUM
reduced by : T -> F
reduced by : E -> T
reduced by : F -> NUM
reduced by : T -> F
reduced by : F -> NUM
reduced by : T -> T * F
reduced by : F -> NUM
reduced by : T -> T / F
reduced by : E -> E + T
reduced by : F -> NUM
reduced by : T -> F
reduced by : E -> E - T

```

图 6-4 执行结果 4

7 附录——完整项目代码

7.1 lr1.y

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

%}

%token T_PLUS T_MINUS T_MUL T_DIV T_LP T_RP T_NUM

%%

expression:
    expression T_PLUS term {
        printf("reduced by : E -> E + T \n");
        $$ = $1 + $3;
    }
| expression T_MINUS term {
    printf("reduced by : E -> E - T \n");
    $$ = $1 - $3;
}
| term {
    printf("reduced by : E -> T \n");
    $$ = $1;
}
;

term:
    term T_MUL factor {
        printf("reduced by : T -> T * F\n");
        $$ = $1 * $3;
    }
| term T_DIV factor {
    printf("reduced by : T -> T / F\n");
    $$ = $1 / $3;
}
| factor {
    printf("reduced by : T -> F\n");
    $$ = $1;
}
```

```

    }
;

factor:
    T_LP expression T_RP {
        printf("reduced by : F -> ( E )\n");
        $$ = $2;
    }
| T_NUM {
    printf("reduced by : F -> NUM \n");
    $$ = $1;
}
;

%%

int main() {
    yyparse();
    return 0;
}

int yyerror(const char *msg) {
    fprintf(stderr, "Error: %s\n", msg);
    fprintf(stderr, "Token: %d\n", yylex());
    return 1;
}

```

7.2 lr1.l

```

%{
#include "lr1.tab.h" // .tab.h 是 yacc 编译后的头文件
#include <stdlib.h>
}%

%%

"+" { return T_PLUS; }
"-" { return T_MINUS; }
"*" { return T_MUL; }
"/" { return T_DIV; }
"(" { return T_LP; }
")" { return T_RP; }

```



```

[0-9]+"."[0-9]+([eE][+-]?[0-9]+)? {
    // sscanf(yytext, "%lf", &yylval.float_val);
    yylval = atof(yytext); // 将数字转换，使用科学计数法、浮点数形式
    return T_NUM;
}

[0-9]+([eE][+-]?[0-9]+)? {
    // sscanf(yytext, "%lf", &yylval.float_val);
    yylval = atof(yytext); // 同理，这个是不含小数点的科学计数法
    return T_NUM;
}

[ \t\n] ; // 跳过空格

. {
    fprintf(stderr, "Error: Unrecognized character %s\n", yytext);
    exit(EXIT_FAILURE);
}

%%

int yywrap() {
    return 1;
}

```

7.3 lr1.tab.h

```

#ifndef YYTOKENTYPE
# define YYTOKENTYPE
    /* Put the tokens into the symbol table, so that GDB and other
    debuggers
        know about them. */
    enum yytokentype {
        T_PLUS = 258,
        T_MINUS = 259,
        T_MUL = 260,
        T_DIV = 261,
        T_LP = 262,
        T_RP = 263,
        T_NUM = 264
    };

```

```

#endif

#if ! defined YYSTYPE && ! defined YYSTYPE_IS_DECLARED
typedef int YYSTYPE;
# define YYSTYPE_IS_TRIVIAL 1
# define yystype YYSTYPE /* obsolescent; will be withdrawn */
# define YYSTYPE_IS_DECLARED 1
#endif

extern YYSTYPE yylval;

```

7.4 lr1.tab.c

```

/* Identify Bison output. */
#define YYBISON 1

/* Bison version. */
#define YYBISON_VERSION "2.4.1"

/* Skeleton name. */
#define YYSKELETON_NAME "yacc.c"

/* Pure parsers. */
#define YYPURE 0

/* Push parsers. */
#define YYPUSH 0

/* Pull parsers. */
#define YYPULL 1

/* Using locations. */
#define YYLSP_NEEDED 0

/* Copy the first part of user declarations. */

```

```

/* Line 189 of yacc.c */
#line 1 "lr1.y"

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/* Line 189 of yacc.c */
#line 81 "lr1.tab.c"

/* Enabling traces. */
#ifndef YYDEBUG
# define YYDEBUG 0
#endif

/* Enabling verbose error messages. */
#ifdef YYERROR_VERBOSE
# undef YYERROR_VERBOSE
# define YYERROR_VERBOSE 1
#else
# define YYERROR_VERBOSE 0
#endif

/* Enabling the token table. */
#ifndef YYTOKEN_TABLE
# define YYTOKEN_TABLE 0
#endif

/* Tokens. */
#ifndef YYTOKENTYPE
# define YYTOKENTYPE
    /* Put the tokens into the symbol table, so that GDB and other
    debuggers
        know about them. */
    enum yytokentype {
        T_PLUS = 258,
        T_MINUS = 259,
        T_MUL = 260,
        T_DIV = 261,

```

```

    T_LP = 262,
    T_RP = 263,
    T_NUM = 264
};
#endif

#if ! defined YYSTYPE && ! defined YYSTYPE_IS_DECLARED
typedef int YYSTYPE;
# define YYSTYPE_IS_TRIVIAL 1
# define yystype YYSTYPE /* obsolescent; will be withdrawn */
# define YYSTYPE_IS_DECLARED 1
#endif

/* Copy the second part of user declarations. */

/* Line 264 of yacc.c */
#line 132 "lr1.tab.c"

#ifdef short
# undef short
#endif

#ifdef YYTYPE_UINT8
typedef YYTYPE_UINT8 yytype_uint8;
#else
typedef unsigned char yytype_uint8;
#endif

#ifdef YYTYPE_INT8
typedef YYTYPE_INT8 yytype_int8;
#elif (defined __STDC__ || defined __C99_FUNC__ \
      || defined __cplusplus || defined _MSC_VER)
typedef signed char yytype_int8;
#else
typedef short int yytype_int8;
#endif

#ifdef YYTYPE_UINT16
typedef YYTYPE_UINT16 yytype_uint16;

```

```

#else
typedef unsigned short int yytype_uint16;
#endif

#ifdef YYSTYPE_INT16
typedef YYSTYPE_INT16 yytype_int16;
#else
typedef short int yytype_int16;
#endif

#ifndef YYSIZE_T
# ifdef __SIZE_TYPE__
#  define YYSIZE_T __SIZE_TYPE__
# elif defined size_t
#  define YYSIZE_T size_t
# elif ! defined YYSIZE_T && (defined __STDC__ || defined __C99_FUNC__
\
    || defined __cplusplus || defined _MSC_VER)
#  include <stddef.h> /* INFRINGES ON USER NAME SPACE */
#  define YYSIZE_T size_t
# else
#  define YYSIZE_T unsigned int
# endif
#endif

#define YYSIZE_MAXIMUM ((YYSIZE_T) -1)

#ifndef YY_
# if YYENABLE_NLS
#  if ENABLE_NLS
#   include <libintl.h> /* INFRINGES ON USER NAME SPACE */
#   define YY_(msgid) dgettext ("bison-runtime", msgid)
#  endif
# endif
# ifndef YY_
#  define YY_(msgid) msgid
# endif
#endif

/* Suppress unused-variable warnings by "using" E. */
#if ! defined lint || defined __GNUC__
# define YYUSE(e) ((void) (e))

```

```

#else
# define YYUSE(e) /* empty */
#endif

/* Identity function, used to suppress warnings about constant
conditions. */
#ifndef lint
# define YYID(n) (n)
#else
#if (defined __STDC__ || defined __C99_FUNC__ \
    || defined __cplusplus || defined _MSC_VER)
static int
YYID (int yyi)
#else
static int
YYID (yyi)
    int yyi;
#endif
#endif
{
    return yyi;
}
#endif

#if ! defined yyoverflow || YYERROR_VERBOSE

/* The parser invokes alloca or malloc; define the necessary
symbols. */

# ifdef YYSTACK_USE_ALLOCA
#   if YYSTACK_USE_ALLOCA
#     ifdef __GNUC__
#       define YYSTACK_ALLOC __builtin_alloca
#     elif defined __BUILTIN_VA_ARG_INCR
#       include <alloca.h> /* INFRINGES ON USER NAME SPACE */
#     elif defined _AIX
#       define YYSTACK_ALLOC __alloca
#     elif defined _MSC_VER
#       include <malloc.h> /* INFRINGES ON USER NAME SPACE */
#       define alloca _alloca
#     else
#       define YYSTACK_ALLOC alloca

```

```

#   if ! defined _ALLOCA_H && ! defined _STDLIB_H && (defined __STDC__
|| defined __C99_FUNC__ \
    || defined __cplusplus || defined _MSC_VER)
#   include <stdlib.h> /* INFRINGES ON USER NAME SPACE */
#   ifndef _STDLIB_H
#       define _STDLIB_H 1
#   endif
#   endif
#   endif
#   endif
#   endif

# ifdef YYSTACK_ALLOC
    /* Pacify GCC's 'empty if-body' warning. */
#   define YYSTACK_FREE(Ptr) do { /* empty */; } while (YYID (0))
#   ifndef YYSTACK_ALLOC_MAXIMUM
        /* The OS might guarantee only one guard page at the bottom of the
        stack,
           and a page size can be as small as 4096 bytes.  So we cannot
        safely
           invoke alloca (N) if N exceeds 4096.  Use a slightly smaller
        number
           to allow for a few compiler-allocated temporary stack slots. */
#   define YYSTACK_ALLOC_MAXIMUM 4032 /* reasonable circa 2006 */
#   endif
#   else
#   define YYSTACK_ALLOC YYMALLOC
#   define YYSTACK_FREE YYFREE
#   ifndef YYSTACK_ALLOC_MAXIMUM
#   define YYSTACK_ALLOC_MAXIMUM YYSIZE_MAXIMUM
#   endif
#   if (defined __cplusplus && ! defined _STDLIB_H \
        && ! ((defined YMALLOC || defined malloc) \
        && (defined YYFREE || defined free)))
#   include <stdlib.h> /* INFRINGES ON USER NAME SPACE */
#   ifndef _STDLIB_H
#       define _STDLIB_H 1
#   endif
#   endif
#   ifndef YMALLOC
#   define YMALLOC malloc

```

```

#   if ! defined malloc && ! defined _STDLIB_H && (defined __STDC__ ||
defined __C99_FUNC__ \
    || defined __cplusplus || defined _MSC_VER)
void *malloc (YYSIZE_T); /* INFRINGES ON USER NAME SPACE */
#   endif
#   endif
#   ifndef YYFREE
#   define YYFREE free
#   if ! defined free && ! defined _STDLIB_H && (defined __STDC__ ||
defined __C99_FUNC__ \
    || defined __cplusplus || defined _MSC_VER)
void free (void *); /* INFRINGES ON USER NAME SPACE */
#   endif
#   endif
#   endif
#endif /* ! defined yyoverflow || YYERROR_VERBOSE */

#if (! defined yyoverflow \
    && (! defined __cplusplus \
    || (defined YYSTYPE_IS_TRIVIAL && YYSTYPE_IS_TRIVIAL)))

/* A type that is properly aligned for any stack member.  */
union yyalloc
{
    yytype_int16 yyss_alloc;
    YYSTYPE yyvs_alloc;
};

/* The size of the maximum gap between one aligned stack and the
next.  */
# define YYSTACK_GAP_MAXIMUM (sizeof (union yyalloc) - 1)

/* The size of an array large to enough to hold all stacks, each with
N elements.  */
# define YYSTACK_BYTES(N) \
    ((N) * (sizeof (yytype_int16) + sizeof (YYSTYPE)) \
     + YYSTACK_GAP_MAXIMUM)

/* Copy COUNT objects from FROM to TO.  The source and destination do
not overlap.  */
# ifndef YYCOPY
#   if defined __GNUC__ && 1 < __GNUC__

```



```

# define YYCOPY(To, From, Count) \
    __builtin_memcpy (To, From, (Count) * sizeof (*(From)))
# else
# define YYCOPY(To, From, Count) \
    do \
    { \
        YYSIZE_T yyi; \
        for (yyi = 0; yyi < (Count); yyi++) \
            (To)[yyi] = (From)[yyi]; \
    } \
    while (YYID (0))
# endif
# endif

/* Relocate STACK from its old location to the new one.  The
   local variables YYSIZE and Yystacksize give the old and new number of
   elements in the stack, and YYPTR gives the new location of the
   stack.  Advance YYPTR to a properly aligned location for the next
   stack.  */
# define YYSTACK_RELOCATE(Stack_alloc, Stack) \
    do \
    { \
        YYSIZE_T yynewbytes; \
        YYCOPY (&yyptr->Stack_alloc, Stack, yysize); \
        Stack = &yyptr->Stack_alloc; \
        yynewbytes = yystacksize * sizeof (*Stack) + YYSTACK_GAP_MAXIMUM; \
        yyptr += yynewbytes / sizeof (*yyptr); \
    } \
    while (YYID (0))

# endif

/* YYFINAL -- State number of the termination state.  */
#define YYFINAL 7
/* YYLAST -- Last index in YYTABLE.  */
#define YYLAST 14

/* YYNTOKENS -- Number of terminals.  */
#define YYNTOKENS 10
/* YYNNTS -- Number of nonterminals.  */
#define YYNNTS 4
/* YYNRULES -- Number of rules.  */

```

```

#define YYNRULES 9
/* YYNRULES -- Number of states. */
#define YYNSTATES 17

/* YYTRANSLATE(YYLEX) -- Bison symbol number corresponding to YYLEX. */
#define YYUNDEFTOK 2
#define YYMAXUTOK 264

#define YYTRANSLATE(YYX) \
    ((unsigned int) (YYX) <= YYMAXUTOK ? yytranslate[YYX] : YYUNDEFTOK)

/* YYTRANSLATE[YYLEX] -- Bison symbol number corresponding to YYLEX. */
static const yytype_uint8 yytranslate[] =
{
    0,    2,    2,    2,    2,    2,    2,    2,    2,    2,
    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
    2,    2,    2,    2,    2,    2,    2,    2,    2,    2,
    2,    2,    2,    2,    2,    2,    1,    2,    3,    4,
    5,    6,    7,    8,    9
};

```

```

#if YYDEBUG
/* YYPRHS[YYN] -- Index of the first RHS symbol of rule number YYN in
   YYRHS.  */
static const yytype_uint8 yyprhs[] =
{
    0,    0,    3,    7,   11,   13,   17,   21,   23,   27
};

/* YYRHS -- A '-1'-separated list of the rules' RHS.  */
static const yytype_int8 yyrhs[] =
{
    11,    0,   -1,   11,    3,   12,   -1,   11,    4,   12,
   -1,   12,   -1,   12,    5,   13,   -1,   12,    6,   13,
   -1,   13,   -1,    7,   11,    8,   -1,    9,   -1
};

/* YYRLINE[YYN] -- source line where rule number YYN was defined.  */
static const yytype_uint8 yyrline[] =
{
    0,   13,   13,   17,   21,   28,   32,   36,   43,   47
};
#endif

#if YYDEBUG || YYERROR_VERBOSE || YYTOKEN_TABLE
/* YYTNAME[SYMBOL-NUM] -- String name of the symbol SYMBOL-NUM.
   First, the terminals, then, starting at YYNTOKENS, nonterminals.  */
static const char *const yytname[] =
{
    "$end", "error", "$undefined", "T_PLUS", "T_MINUS", "T_MUL", "T_DIV",
    "T_LP", "T_RP", "T_NUM", "$accept", "expression", "term", "factor", 0
};
#endif

# ifdef YYPRINT
/* YYTOKNUM[YYLEX-NUM] -- Internal token number corresponding to
   token YYLEX-NUM.  */
static const yytype_uint16 yytoknum[] =
{
    0,  256,  257,  258,  259,  260,  261,  262,  263,  264
};
# endif

```

```

/* YYR1[YYN] -- Symbol number of symbol that rule YYN derives. */
static const yytype_uint8 yyr1[] =
{
    0,    10,    11,    11,    11,    12,    12,    12,    13,    13
};

/* YYR2[YYN] -- Number of symbols composing right hand side of rule
YYN. */
static const yytype_uint8 yyr2[] =
{
    0,    2,    3,    3,    1,    3,    3,    1,    3,    1
};

/* YYDEFACCT[STATE-NAME] -- Default rule to reduce with in state
STATE-NAME when YYTABLE doesn't specify something else to do. Zero
means the default is an error. */
static const yytype_uint8 yydefact[] =
{
    0,    0,    9,    0,    4,    7,    0,    1,    0,    0,
    0,    0,    8,    2,    3,    5,    6
};

/* YYDEFGOTO[NTERM-NAME]. */
static const yytype_int8 yydefgoto[] =
{
    -1,    3,    4,    5
};

/* YYPACT[STATE-NAME] -- Index in YYTABLE of the portion describing
STATE-NAME. */
#define YYPACT_NINF -6
static const yytype_int8 yypact[] =
{
    -5,    -5,    -6,    3,    4,    -6,    -3,    -6,    -5,    -5,
    -5,    -5,    -6,    4,    4,    -6,    -6
};

/* YYPGOTO[NTERM-NAME]. */
static const yytype_int8 yypgoto[] =
{
    -6,    7,    5,    1
};

```

```

};

/* YYTABLE[YYPACT[STATE-NUM]].  What to do in state STATE-NUM.  If
   positive, shift that token.  If negative, reduce the rule which
   number is the opposite.  If zero, do what YYDEFAC says.
   If YYTABLE_NINF, syntax error.  */
#define YYTABLE_NINF -1
static const yytype_uint8 yytable[] =
{
    8,    9,    1,    7,    2,   12,    8,    9,    6,   10,
   11,   15,   16,   13,   14
};

static const yytype_uint8 yycheck[] =
{
    3,    4,    7,    0,    9,    8,    3,    4,    1,    5,
    6,   10,   11,    8,    9
};

/* YYSTOS[STATE-NUM] -- The (internal number of the) accessing
   symbol of state STATE-NUM.  */
static const yytype_uint8 yyustos[] =
{
    0,    7,    9,   11,   12,   13,   11,    0,    3,    4,
    5,    6,    8,   12,   12,   13,   13
};

#define yyerrok    (yyerrstatus = 0)
#define yyclearin (yychar = YYEMPTY)
#define YYEMPTY   (-2)
#define YYEOF     0

#define YYACCEPT   goto yyacceptlab
#define YYABORT    goto yyabortlab
#define YYERROR    goto yyerrorlab

/* Like YYERROR except do call yyerror.  This remains here temporarily
   to ease the transition to the new meaning of YYERROR, for GCC.
   Once GCC version 2 has supplanted version 1, this can go.  */

#define YYFAIL     goto yyerrlab

```

```

#define YYRECOVERING()  (!yyerrstatus)

#define YYBACKUP(Token, Value)      \
do                                  \
{                                  \
    if (yychar == YYEMPTY && yylen == 1)      \
    {                                  \
        yychar = (Token);                  \
        yylval = (Value);                  \
        yytoken = YYTRANSLATE (yychar);      \
        YYPOPSTACK (1);                    \
        goto yybackup;                      \
    }                                  \
    else                                  \
    {                                  \
        yyerror (YY_("syntax error: cannot back up")); \
        YYERROR;                          \
    }                                  \
} while (YYID (0))

#define YYTERROR 1
#define YYERRCODE 256

/* YYLLOC_DEFAULT -- Set CURRENT to span from RHS[1] to RHS[N].
   If N is 0, then set CURRENT to the empty location which ends
   the previous symbol: RHS[0] (always defined).  */

#define YYRHSLOC(Rhs, K) ((Rhs)[K])
#ifndef YYLLOC_DEFAULT
# define YYLLOC_DEFAULT(Current, Rhs, N)      \
do                                  \
    if (YYID (N))                      \
    {                                  \
        (Current).first_line   = YYRHSLOC (Rhs, 1).first_line; \
        (Current).first_column = YYRHSLOC (Rhs, 1).first_column; \
        (Current).last_line    = YYRHSLOC (Rhs, N).last_line; \
        (Current).last_column  = YYRHSLOC (Rhs, N).last_column; \
    }                                  \
    else                                  \
    {                                  \
        (Current).first_line   = (Current).last_line   = \
        YYRHSLOC (Rhs, 0).last_line; \
    }

```

```

        (Current).first_column = (Current).last_column = \
            YYRHSLOC (Rhs, 0).last_column; \
    } \
    while (YYID (0))
#endif

/* YY_LOCATION_PRINT -- Print the location on the stream.
   This macro was not mandated originally: define only if we know
   we won't break user code: when these are the locations we know. */

#ifndef YY_LOCATION_PRINT
# if YYLTYPE_IS_TRIVIAL
#  define YY_LOCATION_PRINT(File, Loc) \
      fprintf (File, "%d.%d-%d.%d", \
                (Loc).first_line, (Loc).first_column, \
                (Loc).last_line, (Loc).last_column)
# else
#  define YY_LOCATION_PRINT(File, Loc) ((void) 0)
# endif
#endif

/* YYLEX -- calling `yylex' with the right arguments. */

#ifdef YYLEX_PARAM
# define YYLEX yylex (YYLEX_PARAM)
#else
# define YYLEX yylex ()
#endif

/* Enable debugging if requested. */
#ifdef YYDEBUG

# ifndef YYFPRINTF
#  include <stdio.h> /* INFRINGES ON USER NAME SPACE */
#  define YYFPRINTF fprintf
# endif

# define YYDPRINTF(Args) \
do { \
    if (yydebug) \
        YYFPRINTF Args; \

```

```

} while (YYID (0))

# define YY_SYMBOL_PRINT(Title, Type, Value, Location) \
do { \
    if (yydebug) \
    { \
        YYFPRINTF (stderr, "%s ", Title); \
        yy_symbol_print (stderr, \
            Type, Value); \
        YYFPRINTF (stderr, "\n"); \
    } \
} while (YYID (0))

/*-----
| Print this symbol on YYOUTPUT.  |
`-----*/

/*ARGSUSED*/
#if (defined __STDC__ || defined __C99_FUNC__ \
    || defined __cplusplus || defined _MSC_VER)
static void
yy_symbol_value_print (FILE *yyoutput, int yytype, YYSTYPE const *
const yyvaluep)
#else
static void
yy_symbol_value_print (yyoutput, yytype, yyvaluep)
    FILE *yyoutput;
    int yytype;
    YYSTYPE const * const yyvaluep;
#endif
{
    if (!yyvaluep)
        return;
# ifdef YYPRINT
    if (yytype < YYNTOKENS)
        YYPRINT (yyoutput, yytoknum[yytype], *yyvaluep);
# else
    YYUSE (yyoutput);
# endif
    switch (yytype)
    {
        default:

```



```

        break;
    }
}

/*-----
| Print this symbol on YYOUTPUT. |
`-----*/

#if (defined __STDC__ || defined __C99_FUNC__ \
    || defined __cplusplus || defined _MSC_VER)
static void
yy_symbol_print (FILE *yyoutput, int yytype, YYSTYPE const * const
yyvaluep)
#else
static void
yy_symbol_print (yyoutput, yytype, yyvaluep)
    FILE *yyoutput;
    int yytype;
    YYSTYPE const * const yyvaluep;
#endif
{
    if (yytype < YYTOKENS)
        YYFPRINTF (yyoutput, "token %s (", yytname[yytype]);
    else
        YYFPRINTF (yyoutput, "nterm %s (", yytname[yytype]);

    yy_symbol_value_print (yyoutput, yytype, yyvaluep);
    YYFPRINTF (yyoutput, ")");
}

/*-----
| yy_stack_print -- Print the state stack from its BOTTOM up to its |
| TOP (included). |
`-----*/

#if (defined __STDC__ || defined __C99_FUNC__ \
    || defined __cplusplus || defined _MSC_VER)
static void
yy_stack_print (yytype_int16 *yybottom, yytype_int16 *yytop)
#else
static void
yy_stack_print (yybottom, yytop)

```

```

    yytype_int16 *yybottom;
    yytype_int16 *yytop;
#endif
{
    YYFPRINTF (stderr, "Stack now");
    for (; yybottom <= yytop; yybottom++)
    {
        int yybot = *yybottom;
        YYFPRINTF (stderr, " %d", yybot);
    }
    YYFPRINTF (stderr, "\n");
}

# define YY_STACK_PRINT(Bottom, Top) \
do { \
    if (yydebug) \
        yy_stack_print ((Bottom), (Top)); \
} while (YYID (0))

/*-----
| Report that the YYRULE is going to be reduced. |
`-----*/

#if (defined __STDC__ || defined __C99_FUNC__ \
    || defined __cplusplus || defined _MSC_VER)
static void
yy_reduce_print (YYSTYPE *yyvsp, int yyrule)
#else
static void
yy_reduce_print (yyvsp, yyrule)
    YYSTYPE *yyvsp;
    int yyrule;
#endif
{
    int yynrhs = yyr2[yyrule];
    int yyi;
    unsigned long int yylno = yyrline[yyrule];
    YYFPRINTF (stderr, "Reducing stack by rule %d (line %lu):\n",
        yyrule - 1, yylno);
    /* The symbols being reduced. */
    for (yyi = 0; yyi < yynrhs; yyi++)
    {

```

```

        YYFPRINTF (stderr, "    $%d = ", yyi + 1);
        yy_symbol_print (stderr, yyrhs[yyprhs[yyrule] + yyi],
            &(yyvsp[(yyi + 1) - (yynrhs)])
            );
        YYFPRINTF (stderr, "\n");
    }
}

# define YY_REDUCE_PRINT(Rule)    \
do {                               \
    if (yydebug)                  \
        yy_reduce_print (yyvsp, Rule); \
} while (YYID (0))

/* Nonzero means print parse trace.  It is left uninitialized so that
   multiple parsers can coexist.  */
int yydebug;
#else /* !YYDEBUG */
# define YYDPRINTF(Args)
# define YY_SYMBOL_PRINT(Title, Type, Value, Location)
# define YY_STACK_PRINT(Bottom, Top)
# define YY_REDUCE_PRINT(Rule)
#endif /* !YYDEBUG */

/* YYINITDEPTH -- initial size of the parser's stacks.  */
#ifndef YYINITDEPTH
# define YYINITDEPTH 200
#endif

/* YYMAXDEPTH -- maximum size the stacks can grow to (effective only
   if the built-in stack extension method is used).

   Do not make this value too large; the results are undefined if
   YYSTACK_ALLOC_MAXIMUM < YYSTACK_BYTES (YYMAXDEPTH)
   evaluated with infinite-precision integer arithmetic.  */

#ifndef YYMAXDEPTH
# define YYMAXDEPTH 10000
#endif

#if YYERROR_VERBOSE

```

```

# ifndef yystrlen
#   if defined __GLIBC__ && defined _STRING_H
#     define yystrlen strlen
#   else
/* Return the length of YYSTR.  */
#if (defined __STDC__ || defined __C99_FUNC__ \
    || defined __cplusplus || defined _MSC_VER)
static YYSIZE_T
yystrlen (const char *yystr)
#else
static YYSIZE_T
yystrlen (yystr)
    const char *yystr;
#endif
{
    YYSIZE_T yylen;
    for (yylen = 0; yystr[yylen]; yylen++)
        continue;
    return yylen;
}
#   endif
# endif

# ifndef yystrcpy
#   if defined __GLIBC__ && defined _STRING_H && defined _GNU_SOURCE
#     define yystrcpy strcpy
#   else
/* Copy YYSRC to YYDEST, returning the address of the terminating '\0'
in
YYDEST.  */
#if (defined __STDC__ || defined __C99_FUNC__ \
    || defined __cplusplus || defined _MSC_VER)
static char *
yystrcpy (char *yydest, const char *yysrc)
#else
static char *
yystrcpy (yydest, yysrc)
    char *yydest;
    const char *yysrc;
#endif
{

```

```

char *yyd = yydest;
const char *yys = yysrc;

while ((*yyd++ = *yys++) != '\0')
    continue;

return yyd - 1;
}
# endif
# endif

# ifndef yytnamerr
/* Copy to YYRES the contents of YYSTR after stripping away unnecessary
quotes and backslashes, so that it's suitable for yyerror.  The
heuristic is that double-quoting is unnecessary unless the string
contains an apostrophe, a comma, or backslash (other than
backslash-backslash).  YYSTR is taken from yytname.  If YYRES is
null, do not copy; instead, return the length of what the result
would have been.  */
static YYSIZE_T
yytnamerr (char *yyres, const char *yystr)
{
    if (*yystr == '"')
    {
        YYSIZE_T yyn = 0;
        char const *yyp = yystr;

        for (;;)
            switch (*++yyp)
            {
                case '\\':
                case ',':
                    goto do_not_strip_quotes;

                case '\\\\':
                    if (*++yyp != '\\')
                        goto do_not_strip_quotes;
                    /* Fall through.  */
                default:
                    if (yyres)
                        yyres[yyn] = *yyp;
                    yyn++;
            }
    }
}

```

```

        break;

    case '':
        if (yyres)
            yyres[yyn] = '\0';
        return yyn;
    }
    do_not_strip_quotes: ;
}

if (! yyres)
    return yystrlen (yystr);

return yystpcpy (yyres, yystr) - yyres;
}
# endif

/* Copy into YYRESULT an error message about the unexpected token
YYCHAR while in state YYSTATE. Return the number of bytes copied,
including the terminating null byte. If YYRESULT is null, do not
copy anything; just return the number of bytes that would be
copied. As a special case, return 0 if an ordinary "syntax error"
message will do. Return YYSIZE_MAXIMUM if overflow occurs during
size calculation. */
static YYSIZE_T
yysyntax_error (char *yyresult, int yystate, int yychar)
{
    int yyn = yypact[yystate];

    if (! (YYPACT_NINF < yyn && yyn <= YYLAST))
        return 0;
    else
        {
            int yytype = YYTRANSLATE (yychar);
            YYSIZE_T yysize0 = yytnamerr (0, yytname[yytype]);
            YYSIZE_T yysize = yysize0;
            YYSIZE_T yysize1;
            int yysize_overflow = 0;
            enum { YYERROR_VERBOSE_ARGS_MAXIMUM = 5 };
            char const *yyarg[YYERROR_VERBOSE_ARGS_MAXIMUM];
            int yyx;

```

```

# if 0
    /* This is so xgettext sees the translatable formats that are
    constructed on the fly. */
    YY_("syntax error, unexpected %s");
    YY_("syntax error, unexpected %s, expecting %s");
    YY_("syntax error, unexpected %s, expecting %s or %s");
    YY_("syntax error, unexpected %s, expecting %s or %s or %s");
    YY_("syntax error, unexpected %s, expecting %s or %s or %s
or %s");
# endif

char *yyfmt;
char const *yyf;
static char const yyunexpected[] = "syntax error, unexpected %s";
static char const yyexpecting[] = ", expecting %s";
static char const yyor[] = " or %s";
char yyformat[sizeof yyunexpected
+ sizeof yyexpecting - 1
+ ((YYERROR_VERBOSE_ARGS_MAXIMUM - 2)
* (sizeof yyor - 1))];
char const *yyprefix = yyexpecting;

/* Start YYX at -YYN if negative to avoid negative indexes in
YYCHECK. */
int yxbegin = yyn < 0 ? -yyn : 0;

/* Stay within bounds of both yycheck and yytnam. */
int yychecklim = YYLAST - yyn + 1;
int yxend = yychecklim < YYNTOKENS ? yychecklim : YYNTOKENS;
int yycount = 1;

yyarg[0] = yytnam[yytype];
yyfmt = yystrcpy (yyformat, yyunexpected);

for (yyx = yxbegin; yyx < yxend; ++yyx)
if (yycheck[yyx + yyn] == yyx && yyx != YYTERROR)
{
    if (yycount == YYERROR_VERBOSE_ARGS_MAXIMUM)
    {
        yycount = 1;
        yysize = yysize0;
        yyformat[sizeof yyunexpected - 1] = '\0';
        break;
    }
}

```

```

    }
    yyarg[yycount++] = yytname[yyx];
    yysize1 = yysize + yytnamerr (0, yytname[yyx]);
    yysize_overflow |= (ysize1 < yysize);
    yysize = yysize1;
    yyfmt = yystpncpy (yyfmt, yyprefix);
    yyprefix = yyor;
}

yyf = YY_(yyformat);
ysize1 = yysize + yystrlen (yyf);
ysize_overflow |= (ysize1 < yysize);
ysize = yysize1;

if (ysize_overflow)
return YYSIZE_MAXIMUM;

if (yyresult)
{
/* Avoid sprintf, as that infringes on the user's name space.
   Don't have undefined behavior even if the translation
   produced a string with the wrong number of "%s"s. */
char *yyp = yyresult;
int yyi = 0;
while ((*yyp = *yyf) != '\0')
{
    if (*yyp == '%' && yyf[1] == 's' && yyi < yycount)
    {
yyp += yytnamerr (yyp, yyarg[yyi++]);
yyf += 2;
    }
    else
    {
yyp++;
yyf++;
    }
}
return yysize;
}
}
#endif /* YYERROR_VERBOSE */

```



```

/*-----
| Release the memory associated to this symbol. |
`-----*/

/*ARGSUSED*/
#if (defined __STDC__ || defined __C99_FUNC__ \
    || defined __cplusplus || defined _MSC_VER)
static void
yydestruct (const char *yymsg, int yytype, YYSTYPE *yyvaluep)
#else
static void
yydestruct (yymsg, yytype, yyvaluep)
    const char *yymsg;
    int yytype;
    YYSTYPE *yyvaluep;
#endif
{
    YYUSE (yyvaluep);

    if (!yymsg)
        yymsg = "Deleting";
    YY_SYMBOL_PRINT (yymsg, yytype, yyvaluep, yylocationp);

    switch (yytype)
    {

        default:
            break;
    }
}

/* Prevent warnings from -Wmissing-prototypes. */
#ifdef YYPARSE_PARAM
#if defined __STDC__ || defined __cplusplus
int yyparse (void *YYPARSE_PARAM);
#else
int yyparse ();
#endif
#else /* ! YYPARSE_PARAM */
#if defined __STDC__ || defined __cplusplus
int yyparse (void);

```

```

#else
int yyparse ();
#endif
#endif /* ! YYPARSE_PARAM */

/* The lookahead symbol. */
int yychar;

/* The semantic value of the lookahead symbol. */
YYSTYPE yylval;

/* Number of syntax errors so far. */
int yynerrs;

/*-----.
| yyparse or yypush_parse. |
`-----*/

#ifdef YYPARSE_PARAM
#if (defined __STDC__ || defined __C99_FUNC__ \
    || defined __cplusplus || defined _MSC_VER)
int
yyparse (void *YYPARSE_PARAM)
#else
int
yyparse (YYPARSE_PARAM)
    void *YYPARSE_PARAM;
#endif
#else /* ! YYPARSE_PARAM */
#if (defined __STDC__ || defined __C99_FUNC__ \
    || defined __cplusplus || defined _MSC_VER)
int
yyparse (void)
#else
int
yyparse ()
#endif
#endif
{

```

```

int yystate;
/* Number of tokens to shift before error messages enabled. */
int yyerrstatus;

/* The stacks and their tools:
   `yyss': related to states.
   `yyvs': related to semantic values.

   Refer to the stacks thru separate pointers, to allow yyoverflow
   to reallocate them elsewhere.  */

/* The state stack.  */
yytype_int16 yyssa[YYINITDEPTH];
yytype_int16 *yyss;
yytype_int16 *yyssp;

/* The semantic value stack.  */
YYSTYPE yyvsa[YYINITDEPTH];
YYSTYPE *yyvs;
YYSTYPE *yyvsp;

YYSIZE_T yystacksize;

int yyn;
int yyresult;
/* Lookahead token as an internal (translated) token number.  */
int yytoken;
/* The variables used to return semantic value and location from the
   action routines.  */
YYSTYPE yyval;

#if YYERROR_VERBOSE
/* Buffer for error messages, and its allocated size.  */
char yymsgbuf[128];
char *yymsg = yymsgbuf;
YYSIZE_T yymsg_alloc = sizeof yymsgbuf;
#endif

#define YYPOPSTACK(N)   (yyvsp -= (N), yyssp -= (N))

/* The number of symbols on the RHS of the reduced rule.

```

```

    Keep to zero when no symbol should be popped. */
int yylen = 0;

yytoken = 0;
yyss = yyssa;
yyvs = yyvsa;
yystacksize = YYINITDEPTH;

YYDPRINTF ((stderr, "Starting parse\n"));

ystate = 0;
yyerrstatus = 0;
ynerrs = 0;
yychar = YYEMPTY; /* Cause a token to be read. */

/* Initialize stack pointers.
   Waste one element of value and location stack
   so that they stay on the same level as the state stack.
   The wasted elements are never initialized. */
yyssp = yyss;
yyvsp = yyvs;

goto yysetstate;

/*-----
| yynewstate -- Push a new state, which is found in yystate. |
`-----*/
yynewstate:
    /* In all cases, when you get here, the value and location stacks
       have just been pushed. So pushing a state here evens the
       stacks. */
    yyssp++;

yysetstate:
    *yyssp = yystate;

    if (yyss + yystacksize - 1 <= yyssp)
    {
        /* Get the current used size of the three stacks, in elements. */
        YYSIZE_T yysize = yyssp - yyss + 1;

#ifdef yyoverflow

```

```

    {
/* Give user a chance to reallocate the stack. Use copies of
these so that the &'s don't force the real ones into
memory. */
YYSTYPE *yyvs1 = yyvs;
yytype_int16 *yyss1 = yyss;

/* Each stack pointer address is followed by the size of the
data in use in that stack, in bytes. This used to be a
conditional around just the two extra args, but that might
be undefined if yyoverflow is a macro. */
yyoverflow (YY_("memory exhausted"),
            &yyss1, yysize * sizeof (*yyssp),
            &yyvs1, yysize * sizeof (*yyvsp),
            &yystacksize);

yyss = yyss1;
yyvs = yyvs1;
    }
#else /* no yyoverflow */
# ifndef YYSTACK_RELOCATE
    goto yyexhaustedlab;
# else
    /* Extend the stack our own way. */
    if (YYMAXDEPTH <= yystacksize)
        goto yyexhaustedlab;
    yystacksize *= 2;
    if (YYMAXDEPTH < yystacksize)
        yystacksize = YYMAXDEPTH;

    {
yytype_int16 *yyss1 = yyss;
union yyalloc *yyptr =
    (union yyalloc *) YYSTACK_ALLOC (YYSTACK_BYTES (yystacksize));
if (! yyptr)
    goto yyexhaustedlab;
YYSTACK_RELOCATE (yyss_alloc, yyss);
YYSTACK_RELOCATE (yyvs_alloc, yyvs);
# undef YYSTACK_RELOCATE
if (yyss1 != yyssa)
    YYSTACK_FREE (yyss1);
    }
#endif

```

```

# endif
#endif /* no yyoverflow */

    yyssp = yyss + yysize - 1;
    yyvsp = yyvs + yysize - 1;

    YYDPRINTF ((stderr, "Stack size increased to %lu\n",
        (unsigned long int) yystacksize));

    if (yyss + yystacksize - 1 <= yyssp)
YYABORT;
}

YYDPRINTF ((stderr, "Entering state %d\n", yystate));

if (yystate == YYFINAL)
    YYACCEPT;

goto yybackup;

/*-----
| yybackup. |
`-----*/
yybackup:

    /* Do appropriate processing given the current state.  Read a
       lookahead token if we need one and don't already have one.  */

    /* First try to decide what to do without reference to lookahead
token.  */
    yyn = yypact[yystate];
    if (yyn == YYPACT_NINF)
        goto yydefault;

    /* Not known => get a lookahead token if don't already have one.  */

    /* YYCHAR is either YYEMPTY or YYEOF or a valid lookahead symbol.  */
    if (yychar == YYEMPTY)
    {
        YYDPRINTF ((stderr, "Reading a token: "));
        yychar = YYLEX;
    }

```

```

if (yychar <= YYEOF)
{
    yychar = yytoken = YYEOF;
    YYDPRINTF ((stderr, "Now at end of input.\n"));
}
else
{
    yytoken = YYTRANSLATE (yychar);
    YY_SYMBOL_PRINT ("Next token is", yytoken, &yyval, &yyloc);
}

/* If the proper action on seeing token YYTOKEN is to reduce or to
   detect an error, take that action. */
 yyn += yytoken;
if (yyn < 0 || YYLAST < yyn || yycheck[yyn] != yytoken)
    goto yydefault;
 yyn = yytable[yyn];
if (yyn <= 0)
{
    if (yyn == 0 || yyn == YYTABLE_NINF)
goto yyerrlab;
    yyn = -yyn;
    goto yyreduce;
}

/* Count tokens shifted since error; after three, turn off error
   status. */
if (yyerrstatus)
    yyerrstatus--;

/* Shift the lookahead token. */
YY_SYMBOL_PRINT ("Shifting", yytoken, &yyval, &yyloc);

/* Discard the shifted token. */
yychar = YYEMPTY;

yystate = yyn;
*++yyvsp = yyval;

goto yynewstate;

```

```

/*-----.
| yydefault -- do the default action for the current state. |
`-----*/
yydefault:
    yyn = yydefact[yystate];
    if (yyn == 0)
        goto yyerrlab;
    goto yyreduce;

/*-----.
| yyreduce -- Do a reduction. |
`-----*/
yyreduce:
    /* yyn is the number of a rule to reduce with. */
    yylen = yyr2[yyn];

    /* If YYLEN is nonzero, implement the default value of the action:
       `$$ = $1'.

       Otherwise, the following line sets YYVAL to garbage.
       This behavior is undocumented and Bison
       users should not rely upon it. Assigning to YYVAL
       unconditionally makes the parser a bit smaller, and it avoids a
       GCC warning that YYVAL may be used uninitialized. */
    yyval = yyvsp[1-yylen];

    YY_REDUCE_PRINT (yyn);
    switch (yyn)
    {
        case 2:

/* Line 1455 of yacc.c */
#line 13 "lr1.y"
        {
            printf("reduced by : E -> E + T \n");
            (yyval) = (yyvsp[(1) - (3)]) + (yyvsp[(3) - (3)]);
        }
        break;

        case 3:

```



```

/* Line 1455 of yacc.c */
#line 17 "lr1.y"
{
    printf("reduced by : E -> E - T \n");
    (yyval) = (yyvsp[(1) - (3)]) - (yyvsp[(3) - (3)]);
    ;}
    break;

case 4:

/* Line 1455 of yacc.c */
#line 21 "lr1.y"
{
    printf("reduced by : E -> T \n");
    (yyval) = (yyvsp[(1) - (1)]);
    ;}
    break;

case 5:

/* Line 1455 of yacc.c */
#line 28 "lr1.y"
{
    printf("reduced by : T -> T * F\n");
    (yyval) = (yyvsp[(1) - (3)]) * (yyvsp[(3) - (3)]);
    ;}
    break;

case 6:

/* Line 1455 of yacc.c */
#line 32 "lr1.y"
{
    printf("reduced by : T -> T / F\n");
    (yyval) = (yyvsp[(1) - (3)]) / (yyvsp[(3) - (3)]);
    ;}
    break;

case 7:

/* Line 1455 of yacc.c */
#line 36 "lr1.y"

```

```

    {
        printf("reduced by : T -> F\n");
        (yyval) = (yyvsp[(1) - (1)]);
    }
    break;

case 8:

/* Line 1455 of yacc.c */
#line 43 "lr1.y"
    {
        printf("reduced by : F -> ( E )\n");
        (yyval) = (yyvsp[(2) - (3)]);
    }
    break;

case 9:

/* Line 1455 of yacc.c */
#line 47 "lr1.y"
    {
        printf("reduced by : F -> NUM \n");
        (yyval) = (yyvsp[(1) - (1)]);
    }
    break;

/* Line 1455 of yacc.c */
#line 1397 "lr1.tab.c"
    default: break;
}
YY_SYMBOL_PRINT ("-> $$ =", yyr1[yyn], &yyval, &yyloc);

YYPOPSTACK (yylen);
yylen = 0;
YY_STACK_PRINT (yyss, yyssp);

*++yyvsp = yyval;

/* Now `shift' the result of the reduction. Determine what state
   that goes to, based on the state we popped back to and the rule
   number reduced by. */

```

```

yyn = yyr1[yyn];

yystate = yypgoto[yyn - YYNTOKENS] + *yyssp;
if (0 <= yystate && yystate <= YYLAST && yycheck[yystate] == *yyssp)
    yystate = yytable[yystate];
else
    yystate = yydefgoto[yyn - YYNTOKENS];

goto yynewstate;

/*-----.
| yyerrlab -- here on detecting error |
`-----*/
yyerrlab:
    /* If not already recovering from an error, report this error. */
    if (!yyerrstatus)
    {
        ++yynerrs;
#ifdef YYERROR_VERBOSE
        yyerror (YY_("syntax error"));
#else
        {
            YYSIZE_T yysize = yysyntax_error (0, yystate, yychar);
            if (yymsg_alloc < yysize && yymsg_alloc < YYSTACK_ALLOC_MAXIMUM)
            {
                YYSIZE_T yyalloc = 2 * yysize;
                if (! (yysize <= yyalloc && yyalloc <= YYSTACK_ALLOC_MAXIMUM))
                    yyalloc = YYSTACK_ALLOC_MAXIMUM;
                if (yymsg != yymsgbuf)
                    YYSTACK_FREE (yymsg);
                yymsg = (char *) YYSTACK_ALLOC (yyalloc);
                if (yymsg)
                    yymsg_alloc = yyalloc;
                else
                {
                    yymsg = yymsgbuf;
                    yymsg_alloc = sizeof yymsgbuf;
                }
            }

            if (0 < yysize && yysize <= yymsg_alloc)

```

```

    {
        (void) yysyntax_error (yymsg, yystate, yychar);
        yyerror (yymsg);
    }
else
    {
        yyerror (YY_("syntax error"));
        if (ysize != 0)
            goto yyexhaustedlab;
    }
}
#endif
}

if (yyerrstatus == 3)
{
    /* If just tried and failed to reuse lookahead token after an
    error, discard it.  */

    if (yychar <= YYEOF)
    {
        /* Return failure if at end of input.  */
        if (yychar == YYEOF)
            YYABORT;
    }
    else
    {
        yydestruct ("Error: discarding",
                    yytoken, &yyval);
        yychar = YYEMPTY;
    }
}

/* Else will try to reuse lookahead token after shifting the error
token.  */
goto yyerrlab1;

/*-----
| yyerrorlab -- error raised explicitly by YYERROR.  |
`-----*/

```

```

yyerrorlab:

/* Pacify compilers like GCC when the user code never invokes
   YYERROR and the label yyerrorlab therefore never appears in user
   code.  */
if (/*CONSTCOND*/ 0)
    goto yyerrorlab;

/* Do not reclaim the symbols of the rule which action triggered
   this YYERROR.  */
YYPOPSTACK (yylen);
yylen = 0;
YY_STACK_PRINT (yyss, yyssp);
yystate = *yyssp;
goto yyerrlab1;

/*-----
| yyerrlab1 -- common code for both syntax error and YYERROR.  |
`-----*/
yyerrlab1:
    yyerrstatus = 3; /* Each real token shifted decrements this.  */

    for (;;)
    {
        yyn = yypact[yystate];
        if (yyn != YYPACT_NINF)
        {
            yyn += YYTERROR;
            if (0 <= yyn && yyn <= YYLAST && yycheck[yyn] == YYTERROR)
            {
                yyn = yytable[yyn];
                if (0 < yyn)
                    break;
            }
        }
    }

    /* Pop the current state because it cannot handle the error
token.  */
    if (yyssp == yyss)
        YYABORT;

```

```

    yydestruct ("Error: popping",
               yystos[yystate], yyvsp);
    YYPOPSTACK (1);
    yystate = *yyssp;
    YY_STACK_PRINT (yyss, yyssp);
}

*++yyvsp = yylval;

/* Shift the error token. */
YY_SYMBOL_PRINT ("Shifting", yystos[yyn], yyvsp, yylsp);

yystate = yyn;
goto yynewstate;

/*-----
| yyacceptlab -- YYACCEPT comes here. |
`-----*/
yyacceptlab:
    yyresult = 0;
    goto yyreturn;

/*-----
| yyabortlab -- YYABORT comes here. |
`-----*/
yyabortlab:
    yyresult = 1;
    goto yyreturn;

#if !defined(yyoverflow) || YYERROR_VERBOSE
/*-----
| yyexhaustedlab -- memory exhaustion comes here. |
`-----*/
yyexhaustedlab:
    yyerror (YY_("memory exhausted"));
    yyresult = 2;
    /* Fall through. */
#endif

yyreturn:
    if (yychar != YYEMPTY)

```

```

        yydestruct ("Cleanup: discarding lookahead",
        yytoken, &yylval);
/* Do not reclaim the symbols of the rule which action triggered
   this YYABORT or YYACCEPT. */
YYPPOPSTACK (yylen);
YY_STACK_PRINT (yyss, yyssp);
while (yyssp != yyss)
{
    yydestruct ("Cleanup: popping",
    yystos[*yyssp], yyvsp);
    YYPOPSTACK (1);
}
#ifdef yyoverflow
    if (yyss != yyssa)
        YYSTACK_FREE (yyss);
#endif
#ifdef YYERROR_VERBOSE
    if (yymsg != yymsgbuf)
        YYSTACK_FREE (yymsg);
#endif
/* Make sure YYID is used. */
return YYID (yyresult);
}

/* Line 1675 of yacc.c */
#line 53 "lr1.y"

int main() {
    yyparse();
    return 0;
}

int yyerror(const char *msg) {
    fprintf(stderr, "Error: %s\n", msg);
    fprintf(stderr, "Token: %d\n", yylex());
    return 1;
}

```

7.5 lex.yy.c

```
/* 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 */

#define YY_USE_PROTOS
#define YY_USE_CONST

#endif /* __STDC__ */

#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 yless(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;

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 11
#define YY_END_OF_BUFFER 12

```

```

static yyconst short int yy_accept[24] =
{
    0,
    0,    0,   12,   10,    9,    9,    5,    6,    3,    1,
    2,    4,    8,    0,    8,    0,    7,    0,    8,    0,
    0,    7,    0
} ;

static yyconst int yy_ec[256] =
{
    0,
    1,    1,    1,    1,    1,    1,    1,    2,    3,
    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,
    1,    1,    1,    1,    1,    1,    1,    1,    1,    1,
    1,    2,    1,    1,    1,    1,    1,    1,    1,    4,
    5,    6,    7,    1,    8,    9,   10,   11,   11,   11,
  11,   11,   11,   11,   11,   11,   11,    1,    1,    1,
    1,    1,    1,    1,    1,    1,    1,    1,   12,    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,

   12,    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[13] =
{
    0,
    1,    1,    1,    1,    1,    1,    2,    2,    1,    1,

```

```

        2,    1
    } ;

static yyconst short int yy_base[26] =
    {
        0,
        0,    0,  27,  28,  28,  28,  28,  28,  28,  28,
        28,  28,   4,  15,   0,  14,   6,  13,  12,  11,
        10,   9,  28,  17,  12
    } ;

static yyconst short int yy_def[26] =
    {
        0,
        23,   1,  23,  23,  23,  23,  23,  23,  23,  23,
        23,  23,  23,  23,  13,  24,  23,  23,  23,  25,
        23,  23,   0,  23,  23
    } ;

static yyconst short int yy_nxt[41] =
    {
        0,
        4,   5,   6,   7,   8,   9,  10,  11,   4,  12,
        13,   4,  14,  21,  15,  16,  17,  20,  18,  22,
        22,  22,  19,  19,  19,  17,  23,   3,  23,  23,
        23,  23,  23,  23,  23,  23,  23,  23,  23,  23
    } ;

static yyconst short int yy_chk[41] =
    {
        0,
        1,   1,   1,   1,   1,   1,   1,   1,   1,   1,
        1,   1,  13,  25,  13,  13,  17,  17,  24,  22,
        21,  20,  19,  18,  16,  14,   3,  23,  23,  23,
        23,  23,  23,  23,  23,  23,  23,  23,  23,  23
    } ;

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 "lr1.1"
#define INITIAL 0
#line 2 "lr1.1"
#include "lr1.tab.h" // .tab.h 是 yacc 编译后的头文件
#include <stdlib.h>
#line 379 "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 6 "lr1.1"

#line 533 "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 >= 24 )
            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] != 28 );

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 8 "lr1.1"

```

```

{ return T_PLUS; }
    YY_BREAK
case 2:
YY_RULE_SETUP
#line 9 "lr1.1"
{ return T_MINUS; }
    YY_BREAK
case 3:
YY_RULE_SETUP
#line 10 "lr1.1"
{ return T_MUL; }
    YY_BREAK
case 4:
YY_RULE_SETUP
#line 11 "lr1.1"
{ return T_DIV; }
    YY_BREAK
case 5:
YY_RULE_SETUP
#line 12 "lr1.1"
{ return T_LP; }
    YY_BREAK
case 6:
YY_RULE_SETUP
#line 13 "lr1.1"
{ return T_RP; }
    YY_BREAK
case 7:
YY_RULE_SETUP
#line 15 "lr1.1"
{
    // sscanf(yytext, "%lf", &yylval.float_val);
    yylval = atof(yytext); // 将数字转换，使用科学计数法、浮点数形式
    return T_NUM;
}
    YY_BREAK
case 8:
YY_RULE_SETUP
#line 21 "lr1.1"
{
    // sscanf(yytext, "%lf", &yylval.float_val);
    yylval = atof(yytext); // 同理，这个是不含小数点的科学计数法

```

```

        return T_NUM;
    }

    YY_BREAK
case 9:
YY_RULE_SETUP
#line 27 "lr1.l"
; // 跳过空格
    YY_BREAK
case 10:
YY_RULE_SETUP
#line 29 "lr1.l"
{
    fprintf(stderr, "Error: Unrecognized character %s\n", yytext);
    exit(EXIT_FAILURE);
}
    YY_BREAK
case 11:
YY_RULE_SETUP
#line 34 "lr1.l"
ECHO;
    YY_BREAK
#line 682 "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 >= 24 )
                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 >= 24 )
            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 == 23);

    return yy_is_jam ? 0 : yy_current_state;
}

#ifdef 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
#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 EOF 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
            }
        }
    }

```

```

#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;

#ifdef YY_ALWAYS_INTERACTIVE
    b->yy_is_interactive = 1;
#else
#ifdef 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();
}

#ifdef 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
#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

#ifdef 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

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

#ifdef 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 34 "lr1.1"

int yywrap() {
    return 1;
}

```