|  |
| --- |
| 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](#_Toc154605885)

[1.1 问题描述 2](#_Toc154605886)

[1.2 实现方法 2](#_Toc154605887)

[2 实验环境 2](#_Toc154605888)

[3 语法分析程序lr1.y 3](#_Toc154605889)

[3.1 语法分析思路 3](#_Toc154605890)

[3.2 语法程序源代码 5](#_Toc154605891)

[4 词法分析程序 lr1.l 7](#_Toc154605892)

[4.1 词法分析思路 7](#_Toc154605893)

[4.2 词法程序源代码 9](#_Toc154605894)

[5 实验步骤 10](#_Toc154605895)

[6 测试 11](#_Toc154605896)

[6.1 测试用例1 11](#_Toc154605897)

[6.2 测试用例2 12](#_Toc154605898)

[6.3 测试用例3 12](#_Toc154605899)

[6.4 测试用例4 13](#_Toc154605900)

[7 附录——完整项目代码 14](#_Toc154605901)

[7.1 lr1.y 14](#_Toc154605902)

[7.2 lr1.l 15](#_Toc154605903)

[7.3 lr1.tab.h 16](#_Toc154605904)

[7.4 lr1.tab.c 17](#_Toc154605905)

[7.5 lex.yy.c 55](#_Toc154605906)

# 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。

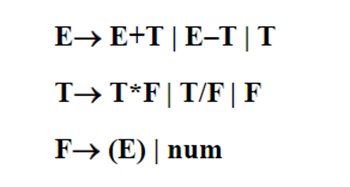


图1-1 文法示意图

## 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 -> E+T | E-T | 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. ;

这表示在规约到E -> E + T时，我们将打印出E -> 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 -> T时，我们也会提示出语法程序的规约过程。

对于T -> T\*F | T/F | 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. ;

而对于F -> (E) | 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. %%
3. **int** main() {
4. yyparse();
5. **return** 0;
6. }
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>
6. %}
8. %token T\_PLUS T\_MINUS T\_MUL T\_DIV T\_LP T\_RP T\_NUM
10. %%
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. $$ = $1;
24. }
25. ;
27. term:
28. term T\_MUL factor   {
29. printf("reduced by : T -> T \* F\n");
30. $$ = $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. ;
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. ;
53. %%
55. **int** main() {
56. yyparse();
57. **return** 0;
58. }
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.l

## 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. yylval = atof(yytext);  // 将数字转换，使用科学计数法、浮点数形式
4. **return** T\_NUM;
5. }
7. [0-9]+([eE][-+]?[0-9]+)? {
8. // sscanf(yytext, "%lf", &yylval.float\_val);
9. yylval = 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. %}
6. %%
8. "+"    { **return** T\_PLUS; }
9. "-"    { **return** T\_MINUS; }
10. "\*"    { **return** T\_MUL; }
11. "/"    { **return** T\_DIV; }
12. "("    { **return** T\_LP; }
13. ")"    { **return** T\_RP; }
15. [0-9]+"."[0-9]+([eE][-+]?[0-9]+)? {
16. // sscanf(yytext, "%lf", &yylval.float\_val);
17. yylval = atof(yytext);  // 将数字转换，使用科学计数法、浮点数形式
18. **return** T\_NUM;
19. }
21. [0-9]+([eE][-+]?[0-9]+)? {
22. // sscanf(yytext, "%lf", &yylval.float\_val);
23. yylval = atof(yytext);  // 同理，这个是不含小数点的科学计数法
24. **return** T\_NUM;
25. }
27. [ \t\n]  ;  // 跳过空格
29. . {
30. fprintf(stderr, "Error: Unrecognized character %s\n", yytext);
31. exit(EXIT\_FAILURE);
32. }
34. %%
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

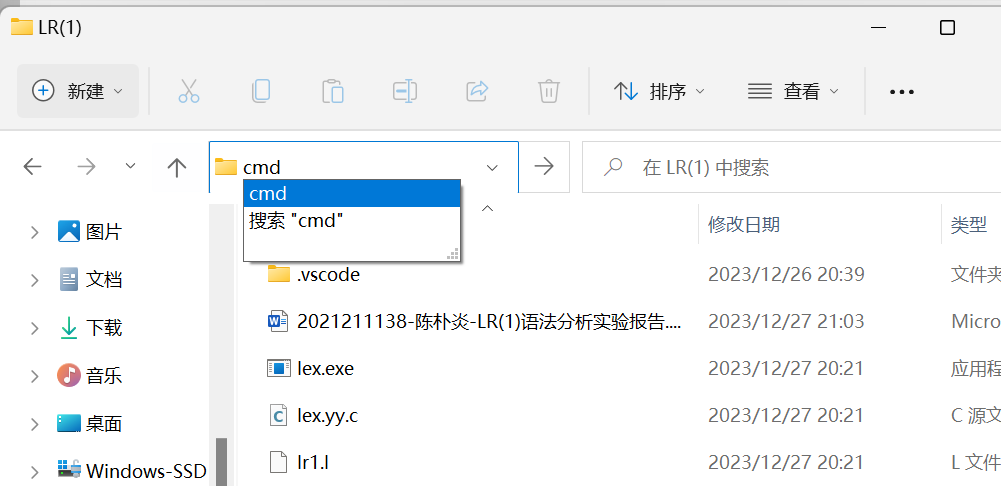


图5-1 在项目文件夹下打开cmd

步骤2. 编译词法程序

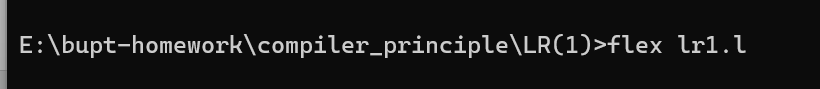


图5-2 编译词法程序

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

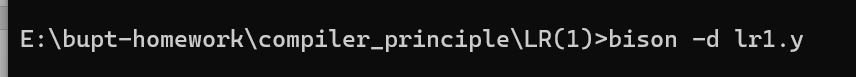


图5-3 编译词法程序

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：

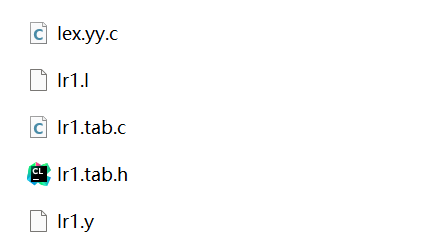


图5-4 自动生成C语言文件

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：

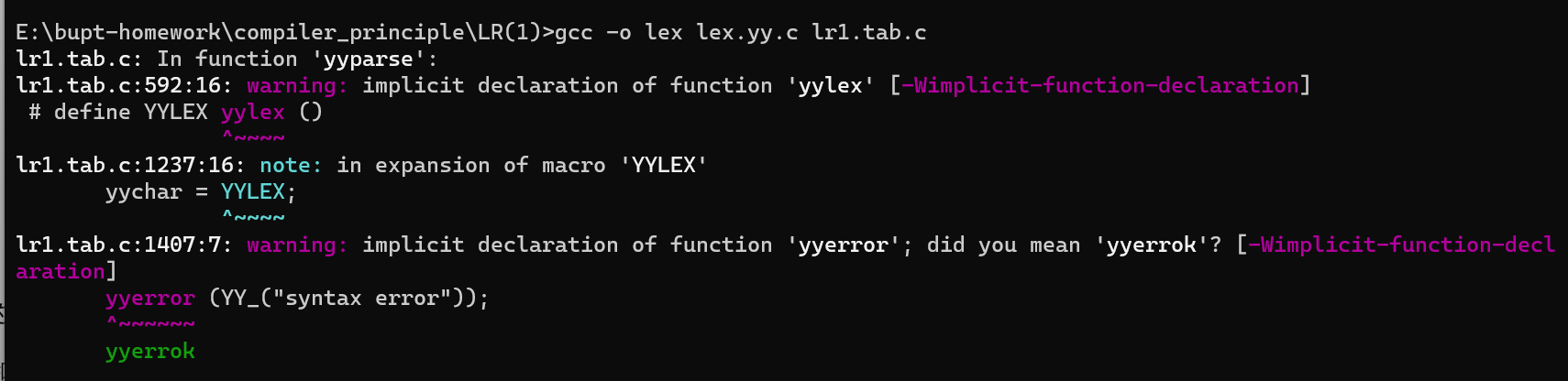


图5-5 编译C语言源程序

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：

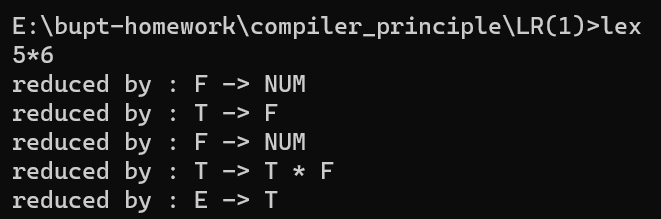


图6-1 执行结果1

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：



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：

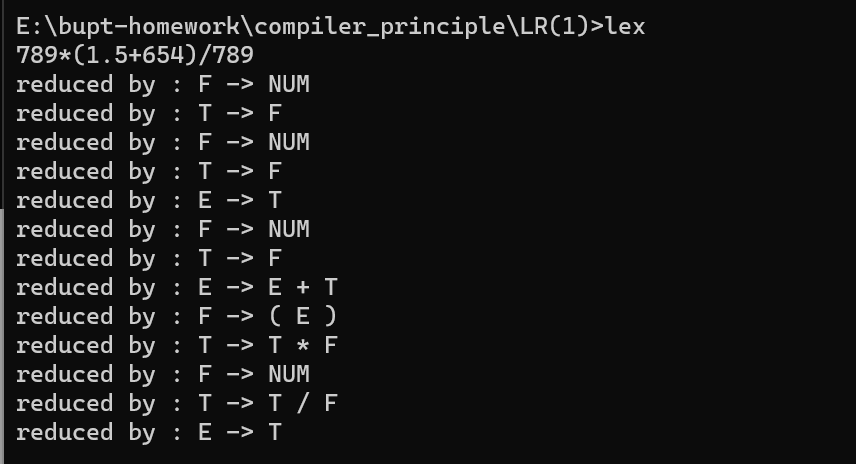


图6-3 执行结果3

## 6.4 测试用例4

输入：1e-5 + 1e-6 \* 1e+10 / 2 – 6

执行结果如下：

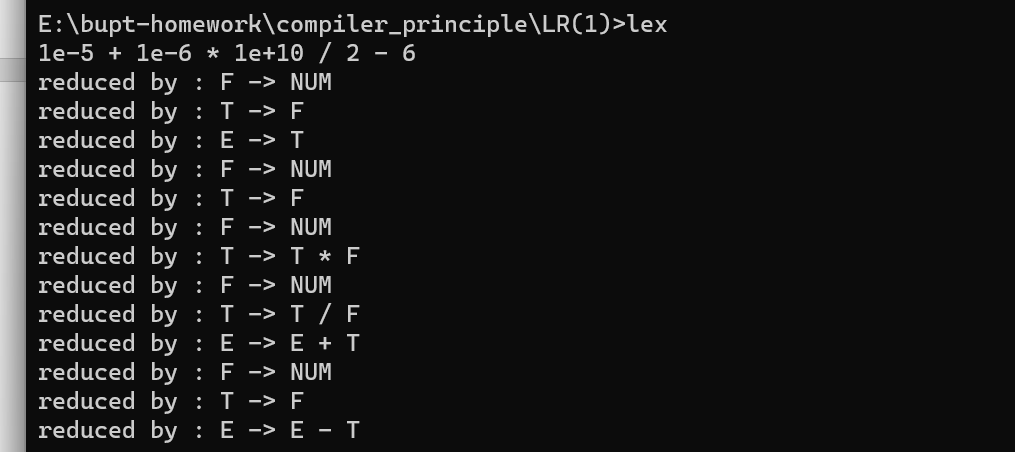


图6-4 执行结果4

# 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 YYTYPE\_INT16

typedef YYTYPE\_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

{

  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 YYMALLOC || defined malloc) \

       && (defined YYFREE || defined free)))

#   include <stdlib.h> /\* INFRINGES ON USER NAME SPACE \*/

#   ifndef \_STDLIB\_H

#    define \_STDLIB\_H 1

#   endif

#  endif

#  ifndef YYMALLOC

#   define YYMALLOC 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,     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

};

/\* YYDEFACT[STATE-NAME] -- Default rule to reduce with in state

   STATE-NUM 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-NUM].  \*/

static const yytype\_int8 yydefgoto[] =

{

      -1,     3,     4,     5

};

/\* YYPACT[STATE-NUM] -- Index in YYTABLE of the portion describing

   STATE-NUM.  \*/

#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-NUM].  \*/

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 YYDEFACT 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 yystos[] =

{

       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.  \*/

#if 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 < YYNTOKENS)

    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 yystpcpy

#  if defined \_\_GLIBC\_\_ && defined \_STRING\_H && defined \_GNU\_SOURCE

#   define yystpcpy stpcpy

#  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 \*

yystpcpy (char \*yydest, const char \*yysrc)

#else

static char \*

yystpcpy (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 yyxbegin = yyn < 0 ? -yyn : 0;

      /\* Stay within bounds of both yycheck and yytname.  \*/

      int yychecklim = YYLAST - yyn + 1;

      int yyxend = yychecklim < YYNTOKENS ? yychecklim : YYNTOKENS;

      int yycount = 1;

      yyarg[0] = yytname[yytype];

      yyfmt = yystpcpy (yyformat, yyunexpected);

      for (yyx = yyxbegin; yyx < yyxend; ++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 |= (yysize1 < yysize);

      yysize = yysize1;

      yyfmt = yystpcpy (yyfmt, yyprefix);

      yyprefix = yyor;

    }

      yyf = YY\_(yyformat);

      yysize1 = yysize + yystrlen (yyf);

      yysize\_overflow |= (yysize1 < yysize);

      yysize = yysize1;

      if (yysize\_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"));

  yystate = 0;

  yyerrstatus = 0;

  yynerrs = 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 /\* 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, &yylval, &yylloc);

    }

  /\* 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, &yylval, &yylloc);

  /\* Discard the shifted token.  \*/

  yychar = YYEMPTY;

  yystate = yyn;

  \*++yyvsp = yylval;

  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;

#if ! 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 (yysize != 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, &yylval);

    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.  \*/

  YYPOPSTACK (yylen);

  YY\_STACK\_PRINT (yyss, yyssp);

  while (yyssp != yyss)

    {

      yydestruct ("Cleanup: popping",

      yystos[\*yyssp], yyvsp);

      YYPOPSTACK (1);

    }

#ifndef yyoverflow

  if (yyss != yyssa)

    YYSTACK\_FREE (yyss);

#endif

#if 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 \*/

#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;

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,    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,    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.l"

#define INITIAL 0

#line 2 "lr1.l"

#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.l"

#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.l"

{ return T\_PLUS; }

    YY\_BREAK

case 2:

YY\_RULE\_SETUP

#line 9 "lr1.l"

{ return T\_MINUS; }

    YY\_BREAK

case 3:

YY\_RULE\_SETUP

#line 10 "lr1.l"

{ return T\_MUL; }

    YY\_BREAK

case 4:

YY\_RULE\_SETUP

#line 11 "lr1.l"

{ return T\_DIV; }

    YY\_BREAK

case 5:

YY\_RULE\_SETUP

#line 12 "lr1.l"

{ return T\_LP; }

    YY\_BREAK

case 6:

YY\_RULE\_SETUP

#line 13 "lr1.l"

{ return T\_RP; }

    YY\_BREAK

case 7:

YY\_RULE\_SETUP

#line 15 "lr1.l"

{

    // 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.l"

{

    // 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;

    }

#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 34 "lr1.l"

int yywrap() {

    return 1;

}