```
%{
#include "stdio.h"
#include "math.h"
#include "string.h"
#include "Node.h"
extern char *yytext;
extern FILE *yyin;
void display(struct Exp *,int);
%}

%union {
	int type_int;
	char type_id[32];
	struct Exp *pExp;
};

%type	<pExp> line exp		/*指定 line 和 exp 的语义值是结点指针 pExp*/
%token <type_int> INT		/*指定 INT 的语义值是 type_int，有词法分析得到的数值*/
%token <type_id> ID		/*指定 ID 的语义值是 type_id，有词法分析得到的标识符字符串*/

%token LP RP	PLUS MINUS STAR DIV ASSIGNOP		/*用 bison 对该文件编译时，带参数-d，生成的 exp.tab.h 中
给这些单词进行编码, 可在 lex.l 中包含 exp.tab.h 使用这些单词种类码*/

%left PLUS MINUS
%left STAR DIV
%left UMINUS

%%
input:
	| input line
	;
line : '\n'	{ ;}
	| exp '\n' { display($1,0);}		/*显示语法树*/
	| error '\n' { printf("exp error!\n");}		/*一旦有语法错误，跳过这行*/
	;
exp	: INT {$$=(PEXP)malloc(sizeof(struct Exp)); $$->kind=INT_NODE;$$->type_int=$1;}
	| ID	{$$=(PEXP)malloc(sizeof(struct Exp)); $$->kind=ID_NODE;strcpy($$->type_id,$1);}
	| exp PLUS exp {$$=(PEXP)malloc(sizeof(struct Exp)); $$->kind=PLUS_NODE; $$->ptr.pExp1=$1;$$->ptr.pExp2=$3;}
	| exp MINUS exp {$$=(PEXP)malloc(sizeof(struct Exp)); $$->kind=MINUS_NODE; $$->ptr.pExp1=$1;$$->ptr.pExp2=$3;}
```

```
      | exp STAR exp {$$=(PEXP)malloc(sizeof(struct Exp)); $$->kind=STAR_NODE;
$$->ptr.pExp1=$1;$$->ptr.pExp2=$3;}
      | exp DIV exp {$$=(PEXP)malloc(sizeof(struct Exp)); $$->kind=DIV_NODE;
$$->ptr.pExp1=$1;$$->ptr.pExp2=$3;}
      | LP exp RP    {$$=(PEXP)$2;}
      | MINUS exp %prec UMINUS {$$=(PEXP)malloc(sizeof(struct Exp));
$$->kind=UMINUS_NODE;  $$->ptr.pExp1=$2;}
      ;
      /*以上 exp 的规则的语义动作生成抽象语法树*/
%%

int main(int argc, char *argv[]){
      yyin=fopen(argv[1],"r");
      if (!yyin) return;
      yyparse();
      return 0;
      }

yyerror(char *s){
    printf("%s    %s \n",s,yytext);

 }

/*
命令序列:
flex lex.l
bison -d -v exp.y
gcc -o exp   exp.tab.c lex.yy.c display.c -Lfl -Ly
*/
```