编译原理实验报告

语法分析程序的设计与实现

姓　　名	李劼
班　　级	07409班
学　　号	071202
班内序号	1

一、实验内容

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

二、实验要求

在对输入表达式进行分析的过程中，输出所采用的产生式。

方法 1：编写递归调用程序实现自顶向下的分析。

方法 2：编写 LL(1)语法分析程序，要求如下

(1) 编写实现算法 4.2，为给定文法自动构造预测分析表。

(2) 编写实现算法 4.1，构造 LL(1)预测分析表。

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) 构造识别所有活前缀的 DFA。

(2) 构造 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 自动生成的自发分析程序。

三、实验环境

```
Linux Kernel 2.6.31-ARCH
flex 2.5.35
GNU bison 2.4.1
vim 7.2
```

四、实验代码

语法分析 ***yacc.y*** 文件

```
01   /* yacc 源代码 */
02  %{
03      #include "symtab.h"
04      #include <stdio.h>
05      #include <string.h>
06
07      extern FILE * yyout;
```

```
%}

%union {
    double val;
    struct symtab * symp; /* 符号表指针 */
}

%token <val> NUM
%token <symp> NAME

%type <val> FACT
%type <val> EXPR
%type <val> TERM

%%

LINE : LINE EXPR '\n'
     | EXPR '\n'
     | LINE '\n'
     | '\n'
     ;
/*
STAT : NAME '=' EXPR { $1->val = $3; yprintf("id = expr"); }
     | EXPR { yprintf(""); }
     ;
*/

EXPR : EXPR '+' TERM { $$ = $1 + $3; yprintf("E -> E + T"); }
     | EXPR '-' TERM { $$ = $1 - $3; yprintf("E -> E - T"); }
     | TERM { $$ = $1; yprintf("E -> T"); }
     ;

TERM : TERM '*' FACT  { $$ = $1 * $3; yprintf("T -> T * F"); }
     | TERM '/' FACT
         {
            if ($3 == 0.0)
                yyerror("Error, divide by zero.");
            else
                $$ = $1 / $3;
            yprintf("T -> T / F");
         }
     | FACT { $$ = $1; yprintf("T -> F"); }
     ;

FACT : NAME { $$ = $1->val; yprintf("F -> id"); }
     | '(' EXPR ')' { $$ = $2; yprintf("F -> (E)"); }
     | NUM { yprintf("F -> num"); }
     ;
%%

#include "lex.yy.c"

```

```
60
61  int yprintf(char * s) {
62      fprintf(yyout, "%s\n", s);
63  }
64  struct symtab * symlook(char * s) /* 在符号表里检索一个符号 */
65  {
66      char * p;
67      struct symtab * sp;
68      for (sp = symtab; sp < &symtab[SYM_NR]; ++sp) {
69          if (sp->name && !strcmp(sp->name, s))
70              return sp;
71
72          if (!sp->name) { /* 新符号，插入符号表 */
73              sp->name = strdup(s);
74              return sp;
75          }
76      }
77      yyerror("Too many symbols.");
78      exit(1);
79  }
80
81
82  int main(int argc, char * argv[])
83  {
84      yyin = stdin;
85      yyout = stdout;
86      while (!feof(yyin)) {
87          yyparse();
88      }
89
90      return 0;
91  }
92
```

词法分析 ***lex.l*** 文件

```
01  %{
02      #include "y.tab.h"
03      #include "symtab.h"
04      #include <stdlib.h>
05      extern FILE * yyout;
06  %}
07
08  digit   [0-9]
09  letter  [_a-zA-Z]
10  integer {digit}+
11  real    ({digit}+)|({digit}*(\.{digit}+)([Ee][+\-]?{digit}+)?)
12  id      {letter}({letter}|{digit})*
13
14  %%
```

```
15
16  [ \t]+  {}
17  [\n]    { return '\n'; }
18  "+"     { return '+'; }
19  "-"     { return '-'; }
20  "*"     { return '*'; }
21  "/"     { return '/'; }
22  "="     { return '='; }
23  "("     { return '('; }
24  ")"     { return ')'; }
25
26  {integer}   {yylval.val = atoi(yytext); return NUM;}
27  {real}      {yylval.val = atof(yytext); return NUM;}
28  {id}        {yylval.symp = symlook(yytext); return NAME;}
29
30  .       { yyerror("syntax error."); return 0; }
31
32  %%
33
34  int yywrap()
35  {
36      return 1;
37  }
38
39  int yyerror(char * s)
40  {
41      fprintf(stderr, "%s\n", s);
42      return 0;
43  }
```

注：lex.l 中 include 的 y.tab.h 为 yacc 生成的提供给 lex 做词法分析的头文件，包含在 yacc.y 中%token 定义的类型的声明。

symtab.h 符号表等 *lex/yacc* 公用数据定义

```
01  #ifndef _EXP2_SYMTAB_H_JACKAL_
02  #define _EXP2_SYMTAB_H_JACKAL_
03
04  #define SYM_NR 100
05
06  //enum type{INT, REAL};
07
08  struct symtab {
09      char * name;
10      double val;
11  }symtab[SYM_NR];
12
13  int yywarp(void);
14  int yyerror(char *s);
15  struct symtab * symlook(char * s);
16
17  #endif
```

五、运行测试

编译运行

```
$ yacc -d yacc.y
$ lex lex.l
$ gcc y.tab.c -o exp2
$ ./exp2
```

输入***:a***

```
  F -> id
  T -> F
  E -> T
```

输入***:(a+b)*c***

```
  F -> id
  T -> F
  E -> T
  F -> id
  T -> F
  E -> E + T
  F -> (E)
  T -> F
  F -> id
  T -> T * F
  E -> T
```

输入***:id/(1+2)***

```
  F -> id
  T -> F
  F -> num
  T -> F
  E -> T
  F -> num
  T -> F
  E -> E + T
  F -> (E)
  T -> T / F
  E -> T
```

六、实验总结

通过本次实验，学会使用 lex＋yacc 自动生成词法分析器＋语法分析器。进一步理解了移进 - 规约的 LR 语法分析方法。