实验四、利用YACC生成能进行整数和实数运算的计算器

3180103468 胡絮燕

实验目的

- 了解YACC属性处理的基本方法。

实验要求

- 生成如下文法表示的表达式对应的计算器
- 要求能连续处理若干个数学表达式，直到输入结束或文件结束。

```
exp->exp + exp|exp – exp
    |exp * exp|exp / exp
    |exp ^ exp|- exp
    |(exp)|NUM
```

代码设计

calc.l

```
%{
#include "y.tab.h"
%}

%%

[0-9]+"."[0-9]+  { /* match float number first */
    yylval.value.flag = 1;
    yylval.value.float_value = atof(yytext);
    return NUMBER;
    }

[0-9]+   {
    yylval.value.flag = 0;
    yylval.value.int_value = atoi(yytext);
    return NUMBER;
    }

[ \t]    ;   /* skip blank chars */
\n       { return yytext[0]; }
.        { return yytext[0]; }

%%
```

设计思路

- 先匹配浮点数，再匹配整数

calc.y

```
%{
#include <stdio.h>
#include <ctype.h>
#include <math.h>
%}

%token NUMBER

%union
{
    struct number {
        int int_value;
            float float_value;
            int flag; /* 0 for int, 1 for float */
    } value;
}

%type <value> exp NUMBER
%left '+' '-'
%left '*' '/'
%left NEG
%right '^'


%%

input   :
        | input command
        ;

command : exp '\n' {
        if($1.flag == 0) printf("%d\n", $1.int_value);
        else printf("%.1f\n", $1.float_value);
        }
        ;

exp     : NUMBER { $$ = $1; }
        | exp '+' exp {
        if($1.flag == 1 && $3.flag == 1) {
            $$.flag = 1;
            $$.float_value = $1.float_value + $3.float_value;
        } else if ($1.flag == 1 && $3.flag == 0) {
            $$.flag = 1;
            $$.float_value = $1.float_value + $3.int_value;
        } else if ($1.flag == 0 && $3.flag == 1) {
            $$.flag = 1;
            $$.float_value = $1.int_value + $3.float_value;
        } else {
            $$.flag = 0;
            $$.int_value = $1.int_value + $3.int_value;
        }
    }
        | exp '-' exp {
```

```
        if($1.flag == 1 && $3.flag == 1) {
            $$.flag = 1;
            $$.float_value = $1.float_value - $3.float_value;
        } else if ($1.flag == 1 && $3.flag == 0) {
            $$.flag = 1;
            $$.float_value = $1.float_value - $3.int_value;
        } else if ($1.flag == 0 && $3.flag == 1) {
            $$.flag = 1;
            $$.float_value = $1.int_value - $3.float_value;
        } else {
            $$.flag = 0;
            $$.int_value = $1.int_value - $3.int_value;
        }
    }
        | exp '*' exp {
        if($1.flag == 1 && $3.flag == 1) {
            $$.flag = 1;
            $$.float_value = $1.float_value * $3.float_value;
        } else if ($1.flag == 1 && $3.flag == 0) {
            $$.flag = 1;
            $$.float_value = $1.float_value * $3.int_value;
        } else if ($1.flag == 0 && $3.flag == 1) {
            $$.flag = 1;
            $$.float_value = $1.int_value * $3.float_value;
        } else {
            $$.flag = 0;
            $$.int_value = $1.int_value * $3.int_value;
        }
    }
        | exp '/' exp {
        if($1.flag == 1 && $3.flag == 1) {
            $$.flag = 1;
            $$.float_value = $1.float_value / $3.float_value;
        } else if ($1.flag == 1 && $3.flag == 0) {
            $$.flag = 1;
            $$.float_value = $1.float_value / $3.int_value;
        } else if ($1.flag == 0 && $3.flag == 1) {
            $$.flag = 1;
            $$.float_value = $1.int_value / $3.float_value;
        } else {
            $$.flag = 0;
            $$.int_value = $1.int_value / $3.int_value;
        }
    }
        | '-' exp %prec NEG {
        if($2.flag == 1) {
            $$.flag = 1;
            $$.float_value = -$2.float_value;
        } else {
            $$.flag = 0;
            $$.int_value = -$2.int_value;
        }
    }
    | exp '^' exp {
        if($1.flag == 1 && $3.flag == 1) {
            $$.flag = 1;
            $$.float_value = pow($1.float_value, $3.float_value);
        } else if ($1.flag == 1 && $3.flag == 0) {
```

```
            $$.flag = 1;
            $$.float_value = pow($1.float_value, $3.int_value);
        } else if ($1.flag == 0 && $3.flag == 1) {
            $$.flag = 1;
            $$.float_value = pow($1.int_value, $3.float_value);
        } else {
            $$.flag = 0;
            $$.int_value = pow($1.int_value, $3.int_value);
        }
    }
    | '(' exp ')' { $$ = $2; }
    ;

%%

int main()
{
    yyparse();
}

int yyerror(char *s)
{
    fprintf(stderr, "%s\n",s);
    return 0;
}
```

设计思路

- 自定义数据类型`struct number`，在yacc中利用`%union`来声明联合，利用`%type`改变`YYSTYPE`的类型。
- 优先级的设计为，幂>负数>乘/除>加/减，在yacc中，可以利用`%left` `%right`来声明左结合还是右结合，调整声明的顺序可以改变优先级。由于`-`出现了两次，我们需要利用`%prec`来将其重命名为`NEG`。这里需要注意的是幂运算是右结合的。

run.sh

```
flex -l calc.l
yacc -d calc.y
gcc -o calc y.tab.c lex.yy.c -ll -lm
./calc < test
```

test

```
3+(4*5)
3+(4.2*2)
3.2+(1/2)
3.2+(1.0/2)
3+(1/2)
2+2^3^2/8
-5*6+7
32/-2+6
-2^2
(-2)^2
```

运行结果

```
hhubibi@hhubibi-virtual-machine:~/Documents/compiler$ bash run.sh
calc.y:18.19-24: warning: POSIX yacc reserves %type to nonterminals [-Wyacc]
   18 | %type <value> exp NUMBER
      |                   ^~~~~~
y.tab.c: In function ‘yyparse’:
y.tab.c:1243:16: warning: implicit declaration of function ‘yylex’ [-Wimplicit-f
unction-declaration]
 1243 |       yychar = yylex ();
      |                ^~~~~
y.tab.c:1505:7: warning: implicit declaration of function ‘yyerror’; did you mea
n ‘yyerrok’? [-Wimplicit-function-declaration]
 1505 |       yyerror (YY_("syntax error"));
      |       ^~~~~~~
      |       yyerrok
23
11.4
3.2
3.7
3
66
-23
-10
-4
4
hhubibi@hhubibi-virtual-machine:~/Documents/compiler$
```

- `3.2+(1/2)=3.2` 这与C语言计算表达式一致，因为我在扫描表达式时，选择了运算符两边都是整型输出才为整型的策略，`3.2+(1.0/2)=3.7`，该结果正确
- `2+2^3^2/8=2+2^9/2^3=2+2^6=66` 结果也正确
- `-2^2=-4` 和 `(-2)^2=4` 结果均正确