

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。

```
1  exp->exp + exp|exp - exp
2      |exp * exp|exp / exp
3      |exp ^ exp|- exp
4      |(exp)|NUM
```

代码设计

calc.l

```
1  %{
2  #include "y.tab.h"
3  %}
4
5  %%
6
7  [0-9]+ "." [0-9]+ { /* match float number first */
8      yylval.value.flag = 1;
9      yylval.value.float_value = atof(yytext);
10     return NUMBER;
11 }
12
13 [0-9]+ {
14     yylval.value.flag = 0;
15     yylval.value.int_value = atoi(yytext);
16     return NUMBER;
17 }
18
19 [ \t] ; /* skip blank chars */
20 \n { return yytext[0]; }
21 . { return yytext[0]; }
22
23 %%
```

设计思路

- 先匹配浮点数，再匹配整数

calc.y

```
1  %{
2  #include <stdio.h>
3  #include <ctype.h>
4  #include <math.h>
5  %}
6
7  %token NUMBER
8
9  %union
10 {
11     struct number {
12         int int_value;
13         float float_value;
14         int flag; /* 0 for int, 1 for float */
15     } value;
16 }
17
18 %type <value> exp NUMBER
19 %left '+' '-'
20 %left '*' '/'
21 %left NEG
22 %right '^'
23
24
25 %%
26
27 input  :
28         | input command
29         ;
30
31 command : exp '\n' {
32     if($1.flag == 0) printf("%d\n", $1.int_value);
33     else printf("%.1f\n", $1.float_value);
34 }
35 ;
36
37 exp    : NUMBER { $$ = $1; }
38         | exp '+' exp {
39     if($1.flag == 1 && $3.flag == 1) {
40         $$ .flag = 1;
41         $$ .float_value = $1.float_value + $3.float_value;
42     } else if ($1.flag == 1 && $3.flag == 0) {
43         $$ .flag = 1;
44         $$ .float_value = $1.float_value + $3.int_value;
45     } else if ($1.flag == 0 && $3.flag == 1) {
46         $$ .flag = 1;
47         $$ .float_value = $1.int_value + $3.float_value;
48     } else {
49         $$ .flag = 0;
50         $$ .int_value = $1.int_value + $3.int_value;
51     }
52 }
53         | exp '-' exp {
```

```

54     if($1.flag == 1 && $3.flag == 1) {
55         $$flag = 1;
56         $$float_value = $1.float_value - $3.float_value;
57     } else if ($1.flag == 1 && $3.flag == 0) {
58         $$flag = 1;
59         $$float_value = $1.float_value - $3.int_value;
60     } else if ($1.flag == 0 && $3.flag == 1) {
61         $$flag = 1;
62         $$float_value = $1.int_value - $3.float_value;
63     } else {
64         $$flag = 0;
65         $$int_value = $1.int_value - $3.int_value;
66     }
67 }
68 | exp '*' exp {
69     if($1.flag == 1 && $3.flag == 1) {
70         $$flag = 1;
71         $$float_value = $1.float_value * $3.float_value;
72     } else if ($1.flag == 1 && $3.flag == 0) {
73         $$flag = 1;
74         $$float_value = $1.float_value * $3.int_value;
75     } else if ($1.flag == 0 && $3.flag == 1) {
76         $$flag = 1;
77         $$float_value = $1.int_value * $3.float_value;
78     } else {
79         $$flag = 0;
80         $$int_value = $1.int_value * $3.int_value;
81     }
82 }
83 | exp '/' exp {
84     if($1.flag == 1 && $3.flag == 1) {
85         $$flag = 1;
86         $$float_value = $1.float_value / $3.float_value;
87     } else if ($1.flag == 1 && $3.flag == 0) {
88         $$flag = 1;
89         $$float_value = $1.float_value / $3.int_value;
90     } else if ($1.flag == 0 && $3.flag == 1) {
91         $$flag = 1;
92         $$float_value = $1.int_value / $3.float_value;
93     } else {
94         $$flag = 0;
95         $$int_value = $1.int_value / $3.int_value;
96     }
97 }
98 | '-' exp %prec NEG {
99     if($2.flag == 1) {
100         $$flag = 1;
101         $$float_value = -$2.float_value;
102     } else {
103         $$flag = 0;
104         $$int_value = -$2.int_value;
105     }
106 }
107 | exp '^' exp {
108     if($1.flag == 1 && $3.flag == 1) {
109         $$flag = 1;
110         $$float_value = pow($1.float_value, $3.float_value);
111     } else if ($1.flag == 1 && $3.flag == 0) {

```

```

112         $$ .flag = 1;
113         $$ .float_value = pow($1.float_value, $3.int_value);
114     } else if ($1.flag == 0 && $3.flag == 1) {
115         $$ .flag = 1;
116         $$ .float_value = pow($1.int_value, $3.float_value);
117     } else {
118         $$ .flag = 0;
119         $$ .int_value = pow($1.int_value, $3.int_value);
120     }
121 }
122 | '(' exp ')' { $$ = $2; }
123 ;
124
125 %%
126
127 int main()
128 {
129     yyparse();
130 }
131
132 int yyerror(char *s)
133 {
134     fprintf(stderr, "%s\n", s);
135     return 0;
136 }

```

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

```

1 flex -l calc.l
2 yacc -d calc.y
3 gcc -o calc y.tab.c lex.yy.c -ll -lm
4 ./calc < test

```

test

```

1 3+(4*5)
2 3+(4.2*2)
3 3.2+(1/2)
4 3.2+(1.0/2)
5 3+(1/2)
6 2+2^3^2/8
7 -5*6+7
8 32/-2+6
9 -2^2
10 (-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