

编译原理实验报告



实验四、 语法分析实验报告

姓 名： 刘宇诺

班 级： 计算机 20-1

学 号： 20201210207

任课教师： 韩连金

目录

- 一、实验目的3
- 二、实验内容3
- 三、实验环境4
- 四、数据准备4
- 五、实验过程描述5
- 六、实验结果及分析9
- 七、实验心得体会12

一、实验目的

理解语法分析在整个编译过程中的作用,掌握词法分析器生成工具 Flex 与语法分析器生成工具 Bison 的使用方法,并使用 Flex 和 Bison 实现简单的语法分析功能。

二、实验内容

1. 使用 Flex 和 Bison 实现可以对整型数据正确进行带有括号的四则运算的计算器程序。

输入输出示例:

输入: $2*(1+3)/(6-3)$

输出: $= 2$

2. 使用 Flex 和 Bison 实现一个可以分析符合以下规则的表达式的语法分析器。

词法规则:

(1)标识符: $[a-zA-Z][0-9a-zA-Z]^+$

(2)数字: $[0-9]^+$

文法规则:

$\langle \text{表达式} \rangle ::= \langle \text{项} \rangle \{ \langle \text{加法运算符} \rangle \langle \text{项} \rangle \}$

$\langle \text{项} \rangle ::= \langle \text{因子} \rangle \{ \langle \text{乘法运算符} \rangle \langle \text{因子} \rangle \}$

$\langle \text{因子} \rangle ::= \langle \text{标识符} \rangle \mid \langle \text{数字} \rangle \mid '(\langle \text{表达式} \rangle)'$

<加法运算符> ::= + | -

<乘法运算符> ::= * | /

输入输出示例:

输入: $a*2-3/4+b$

$a-*b+c$

输出: syntax correct

syntax error

三、实验环境

Windows 7 系统或更高版本

gcc 5.4.0 或更高版本

winflex & winbison 2.5.4 或更高版本

四、数据准备

data 文件夹下的 calc.txt 文件为实验内容 1 所需要计算的算术表达式, 每行为一个带括号的四则运算; data 文件夹下的 expr.txt 文件为实验容 2 所需要进行分析的表达式, 每行为一个合法或非法的表达式。

五、实验过程描述

第一计算 calc 实验的.l 文件, exp1.l

```
1 %{\n2 #include "exp1.tab.h"\n3 %}\n4\n5\n6 %\n7 [\n] {return END;}\n8 [ ]+ {}-\n9 [0-9]+ {yyval = atoi(yytext); return DIGIT;}\n10 [+]{return ADD;}\n11 [-]{return SUB;}\n12 [*]{return MUL;}\n13 [/]{return DIV;}\n14 [(]{return LP;}\n15 [)]{return RP;}\n16\n17 %\n18\n19 int yywrap() {\n20     return 1;\n21 }
```

第一计算 calc 实验的.y 文件 exp1.y 文件

```
1 %f
2 #include <stdio.h>
3 void yyerror(const char* msg) {printf("ERROR: %s\n", msg);}
4 int yylex();
5 %}
6
7 %token DIGIT LP RP
8 %token END
9 %left ADD SUB
10 %left MUL DIV
11
12 %%
13
14 calc :
15     | calc exp END {printf(" = %d\n", $2);}
16     ;
17
18 exp : item {$$ = $1;}
19     | exp ADD exp {$$ = $1 + $3;}
20     | exp SUB exp {$$ = $1 - $3;}
21     | exp MUL exp {$$ = $1 * $3;}
22     | exp DIV exp {$$ = $1 / $3;}
23     ;
24
25 item : DIGIT {$$ = $1;}
26       | LP exp RP {$$ = $2;}
27       ;
28
29 %%
30 int main() {
31     return yyparse();
32 }
```

exp1.y

第二计算 expr 实验的.l 文件 exp2.l 文件

```
1 %{
2 #include "exp2.tab.h"
3 %}
4
5
6 %%
7 [\n] {return END;}
8 [ ]+ {}
9 [0-9]+ {yyval = atoi(yytext); return DIGIT;}
10 [+] {return ADD;}
11 [-] {return SUB;}
12 [*] {return MUL;}
13 [/] {return DIV;}
14 [(] {return LP;}
15 [)] {return RP;}
16 [a-zA-Z][0-9a-zA-Z]* {return VAR;}
17
18 %%
19
20 int yywrap() {
21     return 1;
22 }
```

exp2.l

"exp2.l" [dos] 22L, 331C written

第二计算 expr 实验的.y 文件 exp2.y 文件

```
1 %f
2 #include <stdio.h>
3 void yyerror(const char* msg) {printf("ERROR: %s\n", msg);}
4 int yylex();
5 %}
6
7 %token DIGIT LP RP
8 %token END VAR
9 %left ADD SUB
10 %left MUL DIV
11
12 %%
13
14 line : exp END { printf("syntax correct\n");}
15 exp : item
16     | exp ADD item
17     | exp SUB item
18     ;
19 item : factor
20     | item MUL factor
21     | item DIV factor
22     ;
23 factor : DIGIT
24        | VAR
25        | LP exp RP
26        ;
27
28 %%
29 int main() {
30     return yyparse();
31 }
```

exp2.y

六、实验结果及分析

实验一：

先通过 flex 和 bison 命令处理源文件

```
1:0:~$ bash - "liuyunuo"
lyn@liuyunuo:~/xjuexp/exp4$ ls
calc.txt  exp1.y  exp2.y
exp1.l    exp2.l  expr.txt
lyn@liuyunuo:~/xjuexp/exp4$ flex exp1.l
lyn@liuyunuo:~/xjuexp/exp4$ bison -d exp1.y
lyn@liuyunuo:~/xjuexp/exp4$ ls
calc.txt  exp1.tab.c  exp1.y  exp2.y  lex.yy.c
exp1.l    exp1.tab.h  exp2.l  expr.txt
lyn@liuyunuo:~/xjuexp/exp4$ |
```

利用 gcc, 将 lex.yy.c、exp1.tab.h 和 exp1.tab.c 文件一起进行编译, 生成可执行文件 test1

```
lyn@liuyunuo:~/xjuexp/exp4$
lyn@liuyunuo:~/xjuexp/exp4$ gcc lex.yy.c exp1.tab.h exp1.tab.c -o test1
lyn@liuyunuo:~/xjuexp/exp4$ ls
calc.txt  exp1.tab.c  exp1.y  exp2.y  lex.yy.c
exp1.l    exp1.tab.h  exp2.l  expr.txt  test1
lyn@liuyunuo:~/xjuexp/exp4$ |
```

利用 calc.txt 文件中数据对 test1 进行测试, 测试结果与预期相同。

```
1:0:./test1 - "liuyunuo"
lyn@liuyunuo:~/xjuexp/exp4$
lyn@liuyunuo:~/xjuexp/exp4$
lyn@liuyunuo:~/xjuexp/exp4$ ./test1
1+2*3-4+6/4
= 4
23*(7-5)/(3+3)
= 7
64/(24-16)+99+(25-24)
= 108
20/4+5*(76-22)/9
= 35
6*(5+4)-22+(22+36)
= 90

1 1+2*3-4+6/4
2 23*(7-5)/(3+3)
3 64/(24-16)+99+(25-24)
4 20/4+5*(76-22)/9
5 6*(5+4)-22+(22+36)
~
~
```

实验二:

通过 flex 和 bison 处理源文件, 获得 lex.yy.c、exp2.tab.c、
和 exp2.tab.h

```
1:0:~ - "liuyunuo"
lyn@liuyunuo:~/xjuexp/exp4$ ls
calc.txt  exp1.y  exp2.y  test1
exp1.l    exp2.l  expr.txt
lyn@liuyunuo:~/xjuexp/exp4$ flex exp2.l
lyn@liuyunuo:~/xjuexp/exp4$ bison -d exp2.y
lyn@liuyunuo:~/xjuexp/exp4$ ls
calc.txt  exp1.y  exp2.tab.c  exp2.y  lex.yy.c
exp1.l    exp2.l  exp2.tab.h  expr.txt  test1
lyn@liuyunuo:~/xjuexp/exp4$ |
```

使用 gcc 将生成的三个文件一起进行编译生成 test2 文件

```
yn@liuyunuo:~/xjuexp/exp4$  
yn@liuyunuo:~/xjuexp/exp4$ gcc lex.yy.c exp2.tab.  
exp2.tab.c -o test2  
yn@liuyunuo:~/xjuexp/exp4$ ls  
alc.txt  exp2.l      exp2.y      test1  
xp1.l    exp2.tab.c  expr.txt    test2  
xp1.y    exp2.tab.h  lex.yy.c  
yn@liuyunuo:~/xjuexp/exp4$
```

用 expr.txt 中的数据测试 test2。

测试结果和预期相同

第二个+*一起用，错误；第四个//两个除号，错误；第五个
3ac 数字开头的标识符，错误。

```
1:0: bash - "liuyunuo"  
lyn@liuyunuo:~/xjuexp/exp4$  
lyn@liuyunuo:~/xjuexp/exp4$  
lyn@liuyunuo:~/xjuexp/exp4$ ./test2  
(x+r)*8-(a+b)/7  
syntax correct  
^C  
lyn@liuyunuo:~/xjuexp/exp4$ ./test2  
a+b-255  
ERROR: syntax error  
lyn@liuyunuo:~/xjuexp/exp4$ ./test2  
3*day-year+month/(second+531)  
syntax correct  
^C  
lyn@liuyunuo:~/xjuexp/exp4$ ./test2  
5//a-c+25-4  
ERROR: syntax error  
lyn@liuyunuo:~/xjuexp/exp4$ ./test2  
myList * (size + 5 - 3ac)  
ERROR: syntax error  
lyn@liuyunuo:~/xjuexp/exp4$ |  
  
1 (x+r)*8-(a+b)/7  
2 a+b-255  
3 3*day-year+month/(second+531)  
4 5//a-c+25-4  
5 myList * (size + 5 - 3ac)  
~
```

七、实验心得体会

学习了语法分析的过程，利用 linux 系统安装了 flex 和 bison 以及 gcc 工具进行语法分析实验。通过实验进行实操练习，更加了解了语法分析器的生成过程，也锻炼了自己的动手能力。在一个有规定的文本文件中，通过 BNF 范式对语法进行定义，Bison 读取文件后，输出有相应语法分析功能的 C 语言程序。然后将 Flex 和 Bison 分别生成的具有词法和语法分析功能的 C 语言源程序一起进行编译后又，最终获得语法分析器。