**中国矿业大学计算机学院**

**系统软件开发实践报告**

课程名称 系统软件开发实践

报告时间 2022.3.15

学生姓名 王杰永

学 号 03190886

专 业 计算机科学与技术

任课教师 张博

成绩考核

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 编号 | 课程教学目标 | | | 占比 | | 得分 |
| 1 | **目标1：**针对编译器中词法分析器软件要求，能够分析系统需求，并采用FLEX脚本语言描述单词结构。 | | | 15% | |  |
| 2 | **目标2：**针对编译器中语法分析器软件要求，能够分析系统需求，并采用Bison脚本语言描述语法结构。 | | | 15% | |  |
| 3 | **目标3：**针对计算器需求描述，采用Flex/Bison设计实现高级解释器，进行系统设计，形成结构化设计方案。 | | | 30% | |  |
| 4 | **目标4：**针对编译器软件前端与后端的需求描述，采用软件工程进行系统分析、设计和实现，形成工程方案。 | | | 30% | |  |
| 5 | **目标5：**培养独立解决问题的能力,理解并遵守计算机职业道德和规范，具有良好的法律意识、社会公德和社会责任感。 | | | 10% | |  |
| 总成绩 | | | | | |  |
| 指导教师 | |  | 评阅日期 | |  | |

目录

[三、 综合实验3 1](#_Toc98335996)

[1 实验目的 1](#_Toc98335997)

[2 实验内容 1](#_Toc98335998)

[3 实验步骤 1](#_Toc98335999)

[3.1 文法修改 1](#_Toc98336000)

[3.2 移进/归约冲突的解决 2](#_Toc98336001)

[4 实验结果 5](#_Toc98336002)

[4.1 Windows系统下的实验结果 5](#_Toc98336003)

[4.2 Linux系统下的实验结果 5](#_Toc98336004)

[5 实验总结 6](#_Toc98336005)

1. 综合实验3
   1. 实验目的

阅读《flex&Bison》第三章。使用 flex 和 Bison 开发一个具有全部功能的计算器，包括如下功能：

1) 支持变量；

2) 实现复制功能；

3) 实现比较表达式（大于小于等）；

4) 实现 if/then/else 和 do/while 流程控制；

5) 用户可以自定义函数；

6) 简单的错误恢复机制。

* 1. 实验内容

10)阅读《Flex&Bison》第三章 P79习题1，学习抽象语法树

11)修改fb3-2的相关代码，实现特定函数，并保存为fb3-3

12)撰写实验报告，结合实验结果，给出抽象语法树的构建过程

* 1. 实验步骤

本次实验的重点与前两次综合实验略有不同，前两次实验考察词法、语法分析过程并给出源代码；而本次实验则侧重于编译前端的最后一部分——语法制导翻译。

实验中，我们需要修改特定文法或是语法制导翻译模式，使计算器支持特定的语法。

* + 1. 文法修改

实验要求我们修改计算器自定义函数的语句let，使之支持**大括号**作为函数体作用范围的区域；同时在if/else等选择判断分支语句处，也要支持**大括号**的使用。

首先观察let的语法分析段，如下：

1. calclist: */\* nothing \*/*
2. | calclist stmt EOL {
3. if(debug) dumpast($2, 0);
4. printf("= %4.4g**\n**> ", eval($2));
5. treefree($2);
6. }
7. | calclist LET NAME '(' symlist ')' '=' list EOL {
8. dodef($3, $5, $8);
9. printf("Defined %s**\n**> ", $3->name); }
11. | calclist error EOL { yyerrok; printf("> "); }
12. ;

行7给出了let的产生式。为了使let后支持大括号，我们去掉行7中的'='，并修改dodef($3, $5, $7)。接下来对文法的修改思路就很清晰了——修改非终结符，使之支持大括号结构。

参考资料中的《ANSI C grammar (Yacc).pdf》，我们对list产生式修改为如下结构：

1. list: '{' list '}' {$$ = $2;}
2. | '{' list stmt '}' {$$ = newast('L', $2, $3);}
3. | stmt {$$ = $1;}
4. | exp {$$ = $1;}

最后，由于计算器本身不支持{}这样的终结符，我们在词法分析中，增加左右大括号的词法匹配模式：

1. "{" |
2. "}" |
3. { return yytext[0]; }

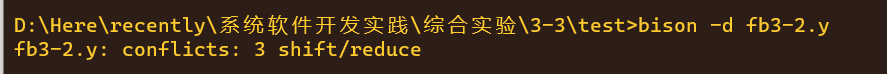
至此，我们的计算器在文法上已经支持大括号结构了。编译结果如下：

图1 第一次编译结果

经过我们修改的文法，存在3个移进/归约冲突。我们使用bison -v命令，生成日志文件，查看具体的移进/归约冲突发生的代码段。

* + 1. 移进/归约冲突的解决

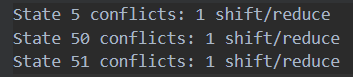
在使用bison -v指令后，得到fb3-2.output日志文件。从中可以看出，识别活前缀的DFA在状态5、50、51发成了移进/归约冲突。

图2 移进/归约冲突的产生位置

* + - * 状态5冲突的解决

日志文件中，对状态5的描述如下。

1. state 5
3. 19 exp: NAME .
4. 20 | NAME . '=' exp
5. 21 | NAME . '(' explist ')'
7. '=' shift, and go to state 16
8. '(' shift, and go to state 17
10. '(' [reduce using rule 19 (exp)]
11. $default reduce using rule 19 (exp)

当自底向上的语法分析归约出文法符号NAME时，如果输入串的下一个符号是'('，此时识别活前缀的DFA既可以移进'('转为状态17，又可以使用产生式归约。即移进/归约冲突。

我们将NAME的优先级降低，优先移进归约左括号后的表达式，从而解决冲突。

在Yacc文件的声明区，加入，并将左括号的改为左结合性后，将产生式更改为，即可解决冲突。

* + - * 状态50冲突的解决

日志文件中，对状态50的描述如下：

1. state 50
3. 1 stmt: IF exp THEN list .
4. 2 | IF exp THEN list . ELSE list
6. ELSE shift, and go to state 57
8. ELSE [reduce using rule 1 (stmt)]
9. $default reduce using rule 1 (stmt)

当符号栈中已经存在了归约出的IF exp THEN list，且输入串的下一位是ELSE时，此时识DFA既可以移进ELSE转为状态57，又可以使用产生式归约。即移进/归约冲突。

根据ELSE与最近的IF匹配的原则，我们选择降低归约操作的优先级，优先移进操作即可解决冲突。

我们将上述产生式修改为，并为符号ELSE 加入左结合性，冲突解决。

* + - * 状态51冲突的解决

日志文件中，对状态50的描述如下：

1. state 51
3. 4 stmt: exp . ';'
4. 8 list: exp .
5. 9 exp: exp . CMP exp
6. 10 | exp . '+' exp
7. 11 | exp . '-' exp
8. 12 | exp . '\*' exp
9. 13 | exp . '/' exp
11. CMP shift, and go to state 26
12. '+' shift, and go to state 27
13. '-' shift, and go to state 28
14. '\*' shift, and go to state 29
15. '/' shift, and go to state 30
16. ';' shift, and go to state 31
18. '-' [reduce using rule 8 (list)]
19. $default reduce using rule 8 (list)

当符号栈中已经存在了归约出的exp，且输入串的下一位是'-'时，此时识DFA既可以移进'-'转为状态28，又可以使用产生式归约。即移进/归约冲突。

解决冲突的办法仍然是降低归约的优先级，为'-'增加左结合性即可。

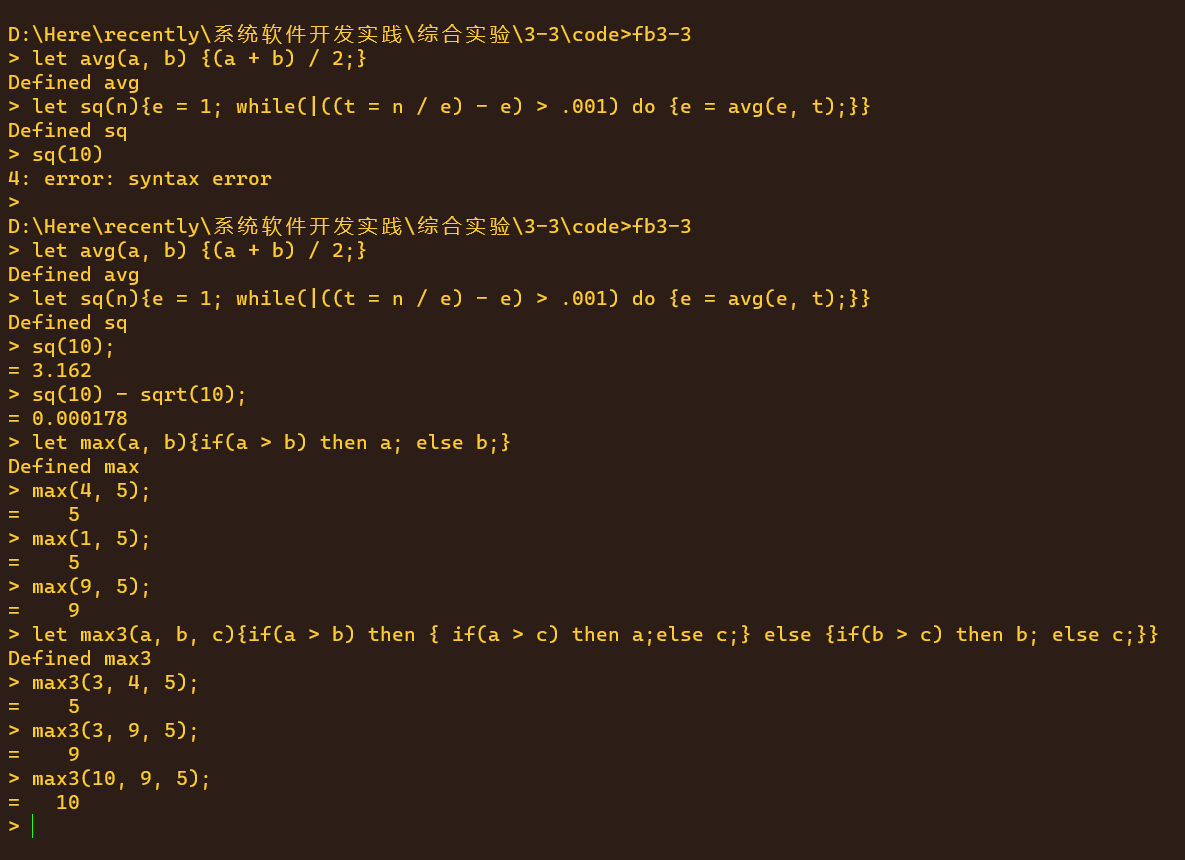
* 1. 实验结果
     1. Windows系统下的实验结果

图3 windows系统下实验结果

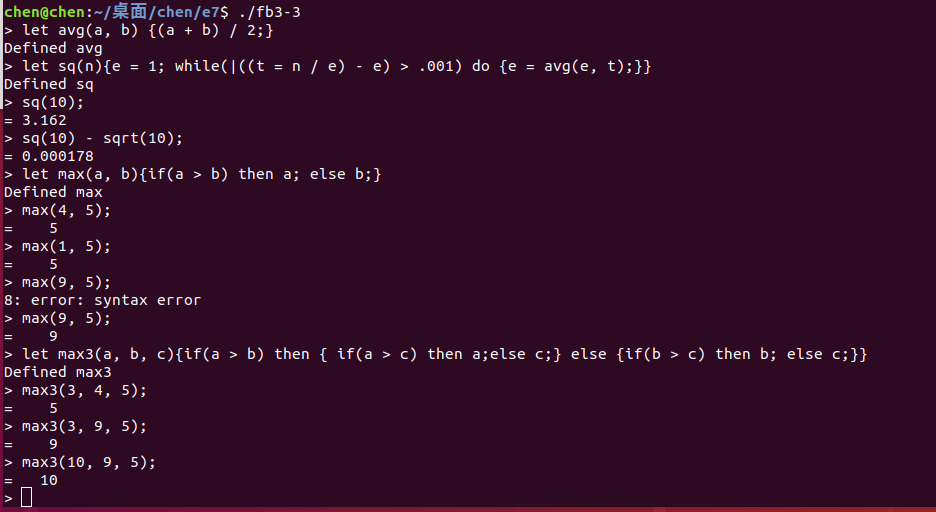
* + 1. Linux系统下的实验结果

图4 Linux系统下实验结果

* 1. 实验总结

本次实验总体上较为简单，根据编译原理课程上所学的语法制导翻译相关内容，以及参考书《ANSI C grammar (Yacc).pdf》，可以比较轻松的完成文法的修改；通过Yacc的冲突发生时的.output日志文件，可以快速定位移进/归约冲突的位置，并给出合理的解决方案。通过本次实验，对Flex与Bison的使用更加熟练，加深了对计算器源码的理解。受益匪浅。