**中国矿业大学计算机学院**

**2018 级本科生课程设计报告**

课程名称 系统软件开发实践

实验名称 实验3-2 高级计算器

报告时间 2021年3月20日

学生姓名 丁旭行

学 号 14184501

专 业 计算机科学与技术

任课教师 徐东红

目 录

[**1 实验目的 1**](#_Toc67738027)

[**2 实验内容 1**](#_Toc67738028)

[**3 实验要求 1**](#_Toc67738029)

[**4 高级计算器功能 1**](#_Toc67738030)

[**5 代码文件分析 2**](#_Toc67738031)

[**5.1 fb3-2.h文件分析 2**](#_Toc67738032)

[**5.2 fb3-2.y文件分析 3**](#_Toc67738033)

[**5.3 fb3-2.l文件分析 3**](#_Toc67738034)

[**5.4 fb3-2funcs.c文件分析 4**](#_Toc67738035)

[**6 节点分析 6**](#_Toc67738036)

[**6.1 比较运算节点 6**](#_Toc67738037)

[**6.2 符号节点 7**](#_Toc67738038)

[**6.3 形式参数列表节点 7**](#_Toc67738039)

[**6.4 语句列表节点 7**](#_Toc67738040)

[**6.5 内置函数节点 8**](#_Toc67738041)

[**6.6 用户自定义函数节点 8**](#_Toc67738042)

[**6.7 控制语句节点 8**](#_Toc67738043)

[**6.8 引用节点 9**](#_Toc67738044)

[**6.9 赋值节点 9**](#_Toc67738045)

[**7 实验步骤 9**](#_Toc67738046)

[**7.1 Windows环境 9**](#_Toc67738047)

[**7.2 Linux环境 12**](#_Toc67738048)

[**8 分析抽象语法树构建过程 14**](#_Toc67738049)

[**9 实验总结 17**](#_Toc67738050)

# 1 实验目的

使用Flex和Bison开发一个具有全部功能的桌面计算器。

# 2 实验内容

使用Flex和Bison开发一个具有全部功能的桌面计算器，能够支持变量，过程，循环和条件表达式，使它成为一个虽然短小，但是具有现实意义的编译器。重点学习抽象语法树的用法，它具有强大而简单的数据结构来表示分析结果。

# 3 实验要求

(1) 阅读《Flex&Bison》第三章，学习抽象语法树”Abstract Syntax Trees”；

(2) 阅读fb3-2.y、fb3-2.l、fb3-2funcs.c、fb3-2.h；

(3) 增加新的内置函数pow(a,n)计算一个数a的n次方；

(4) 撰写实验报告，结合实验结果，如sqrt(10)和sq(10)，给出抽象语法树的构建过程；

(5) 提交报告和实验代码。

# 4 高级计算器功能

* 变量命名；
* 实现赋值功能；
* 实现比较表达式(大于、小于、等于等等)；
* 实现if/then/else和do/while的流程控制；
* 用户可以自定义函数；
* 简单的错误恢复机制。

(注：此处实现功能均是在基本计算器功能基础之上追加的功能)

# 5 代码文件分析

## 5.1 fb3-2.h文件分析

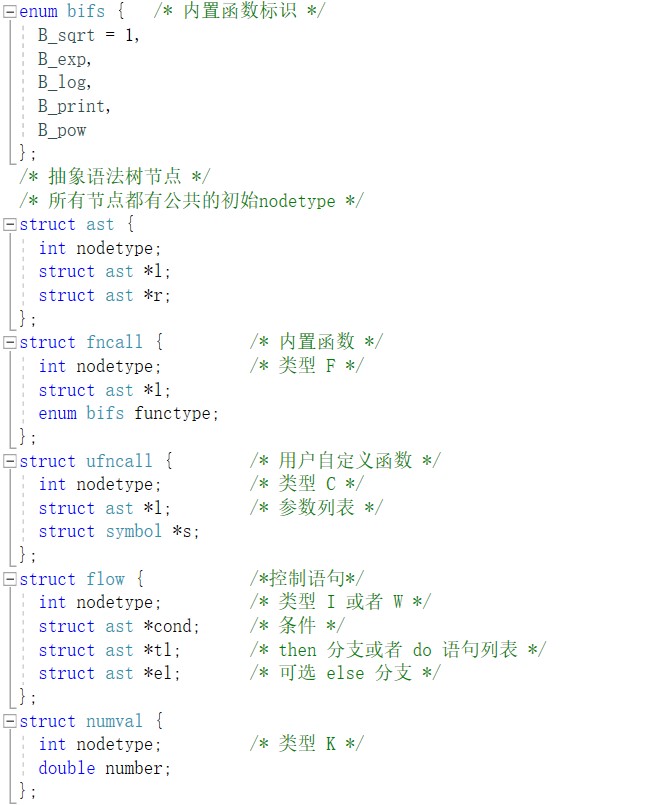
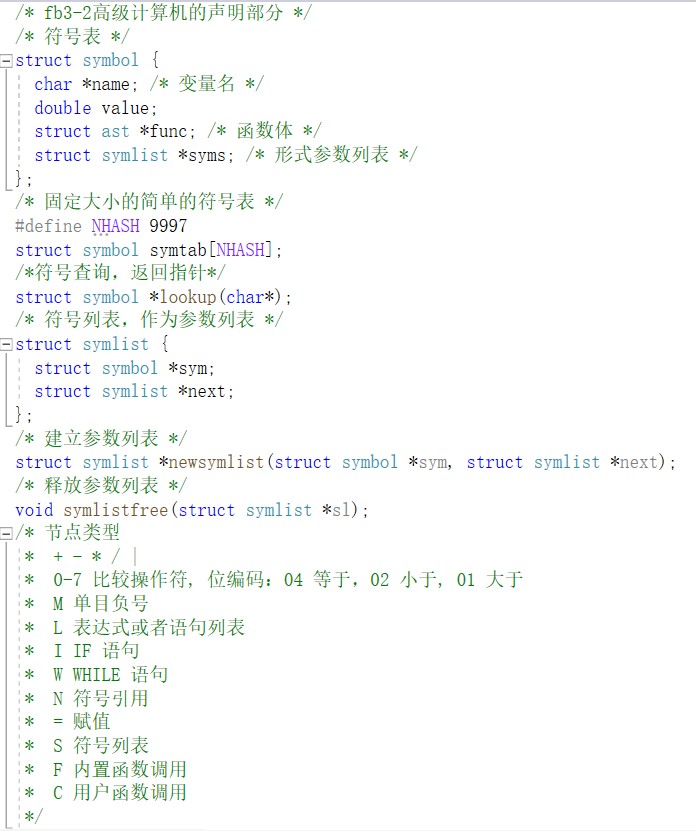


图1 fb3-2.h文件1 图2 fb3-2.h文件2



图3 fb3-2.h文件3

## 5.2 fb3-2.y文件分析

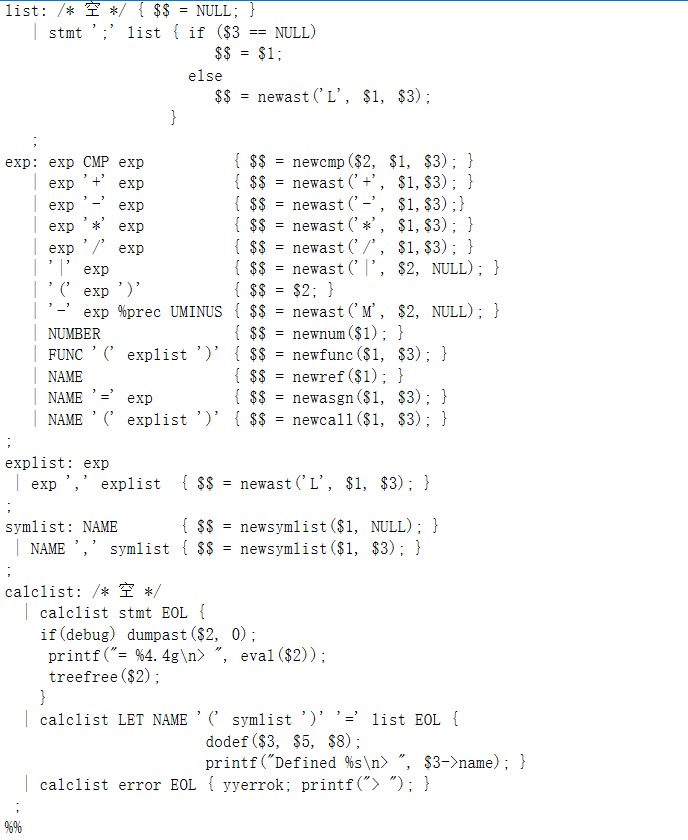
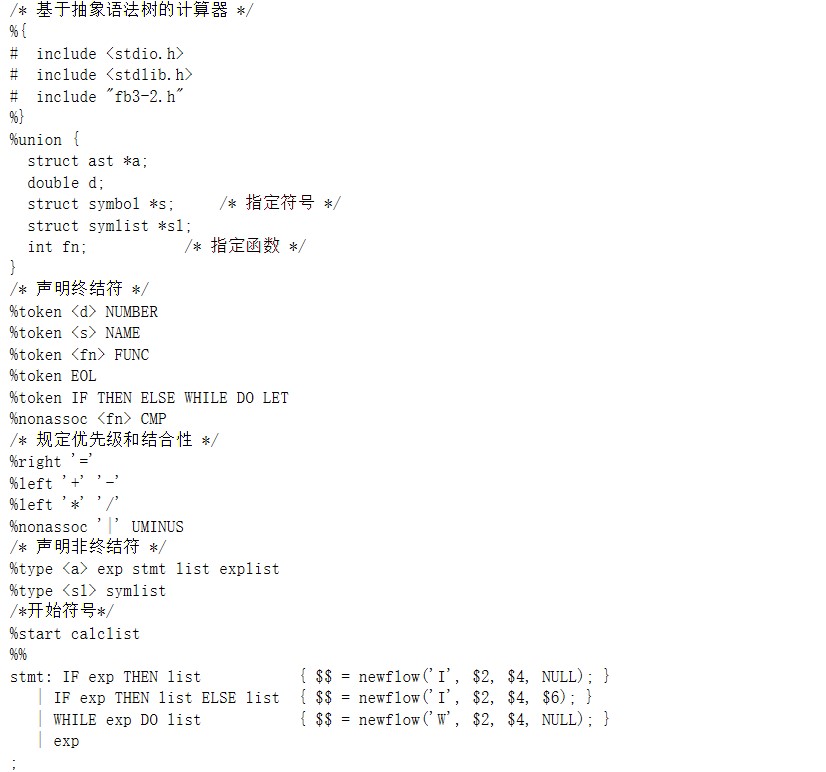


图4 fb3-2.y文件1 图5 fb3-2.y文件2

## 5.3 fb3-2.l文件分析

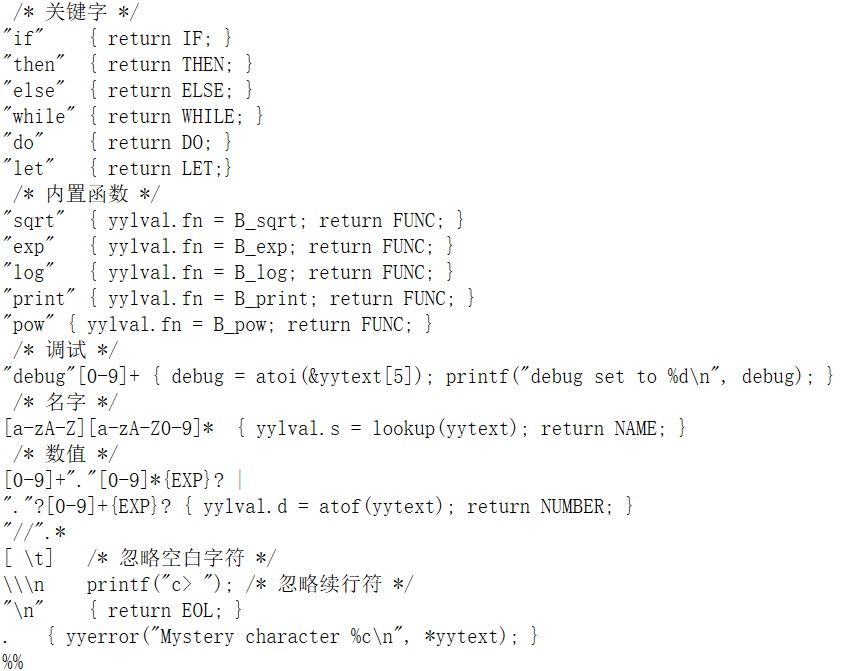
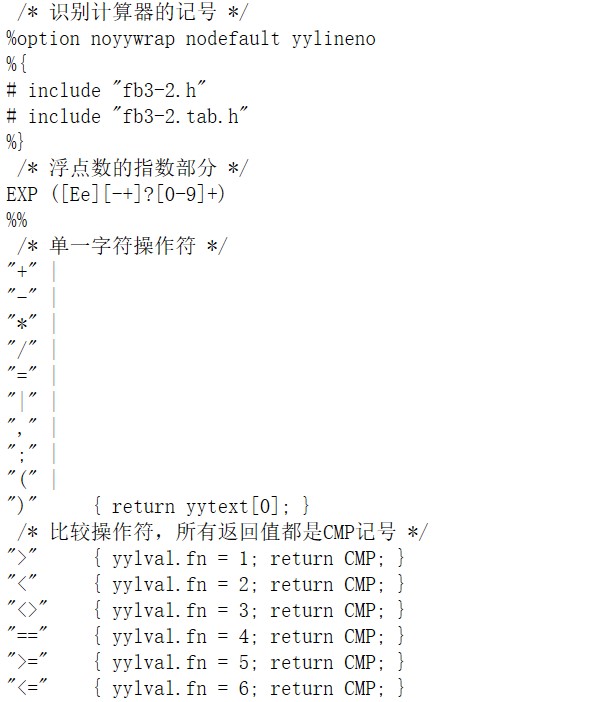


图6 fb3-2.l文件1 图7 fb3-2.l文件2

## 5.4 fb3-2funcs.c文件分析

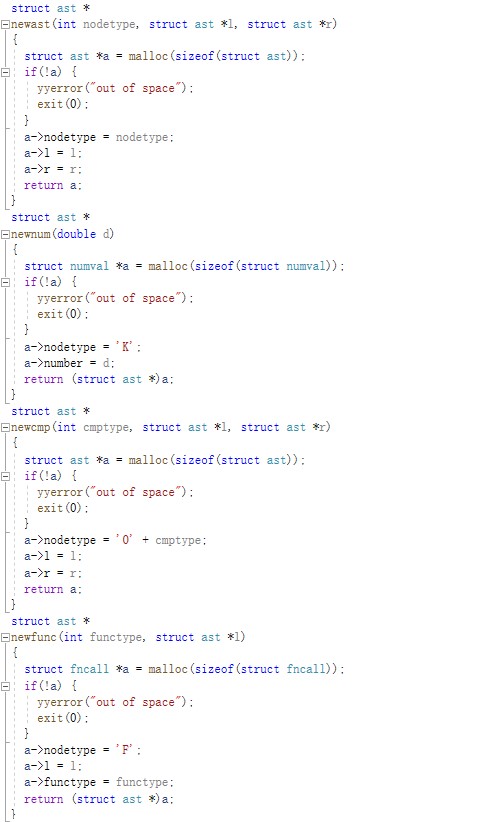


图8 fb3-2funcs.c文件1 图9 fb3-2funcs.c文件2

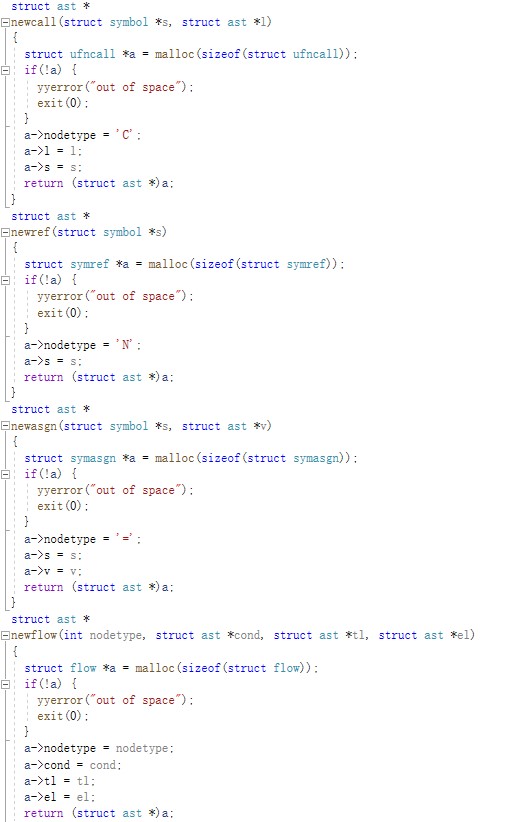


图10 fb3-2funcs.c文件3 图11 fb3-2funcs.c文件4

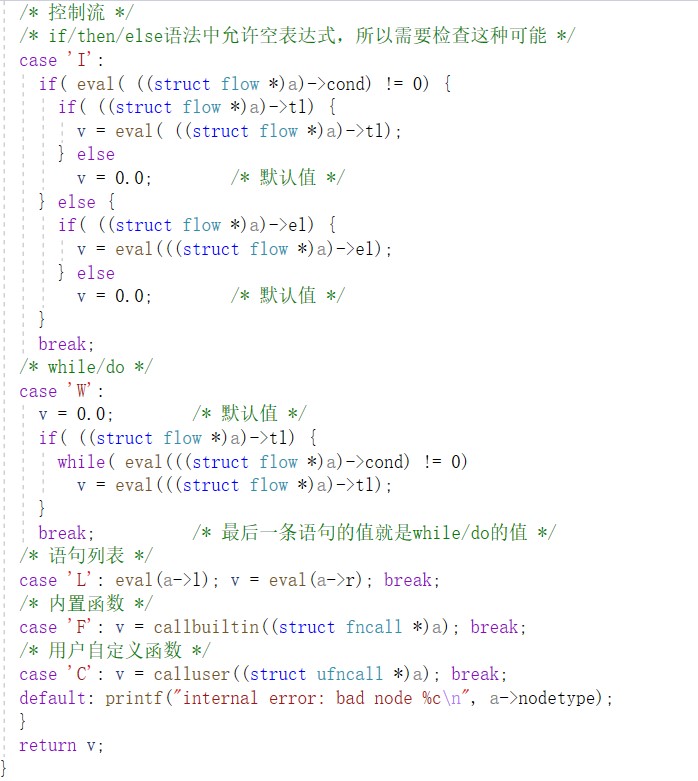
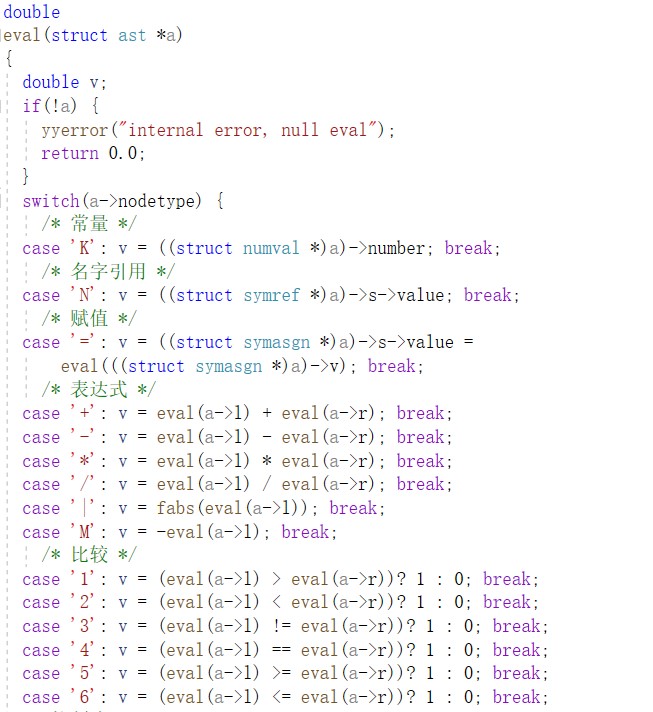


图12 fb3-2funcs.c文件5 图13 fb3-2funcs.c文件6

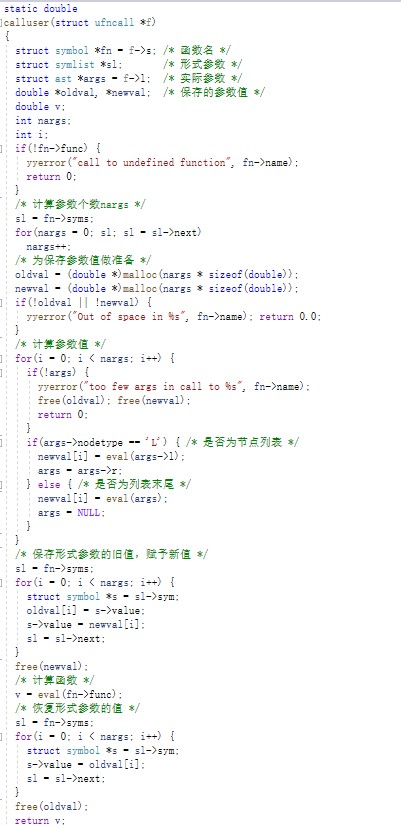
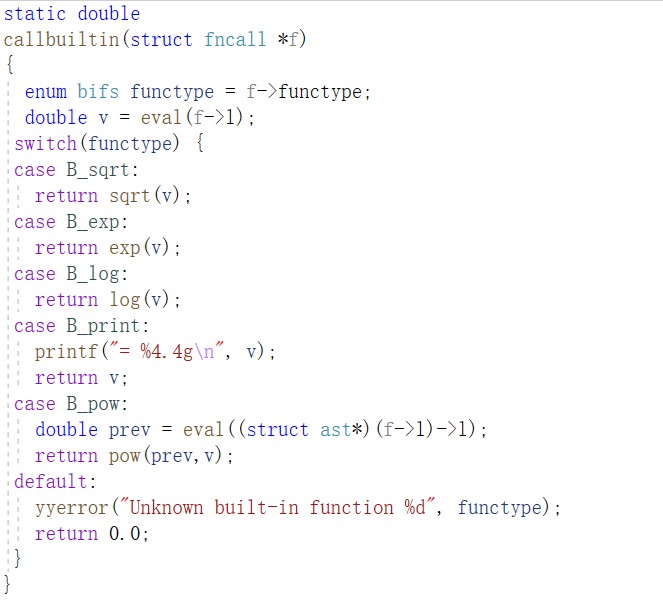


图14 fb3-2funcs.c文件7 图15 fb3-2funcs.c文件8

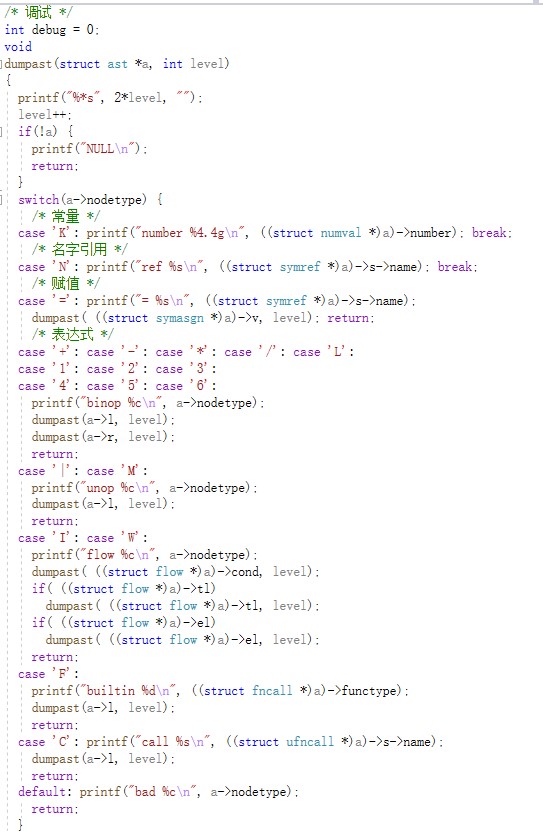
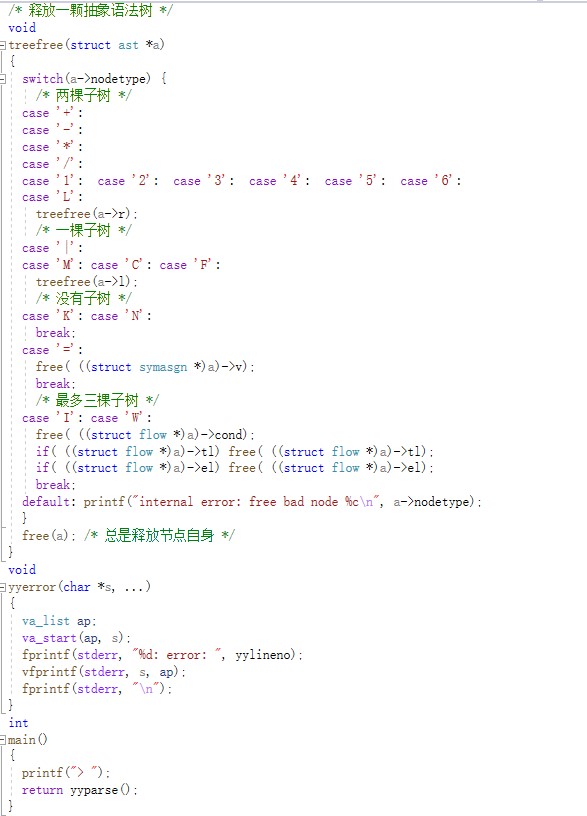


图16 fb3-2funcs.c文件9 图17 fb3-2funcs.c文件10

# 6 节点分析

关于普通节点ast和数值节点numval的分析与实验五报告内分析相同在此不再花费篇幅赘述。

## 6.1 比较运算节点

比较运算节点与普通节点相似，但其类型为内置的比较运算符的编号值，其左右子节点指向其参与比较的两个表达式：

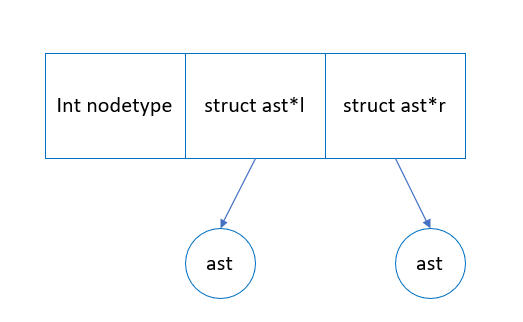


图18 比较运算节点类型

## 6.2 符号节点

词法分析器对于用户输入的标识符会在其符号表中寻找是否已存在同名符号，若为存在则为其新建一个符号节点，否则直接引用已存在符号节点。符号节点包含标识符名称、数值、参数列表节点和普通节点：

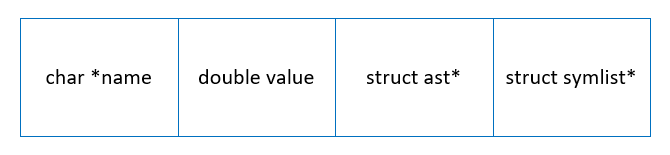


图19 符号节点类型

## 6.3 形式参数列表节点

形式参数列表节点包含一个符号节点指针，其指向第一个形式参数，还包含一个形式参数列表节点的指针，其指向下一个形式参数列表：

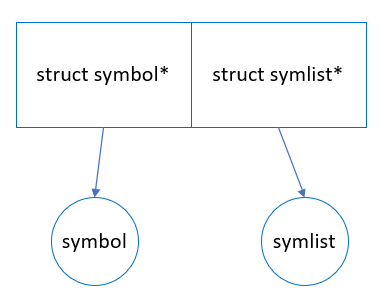


图20 形式参数列表节点类型

## 6.4 语句列表节点

语句列表节点和普通节点相似，只是其节点类型为’L’，左右子节点指向相邻的两条语句：

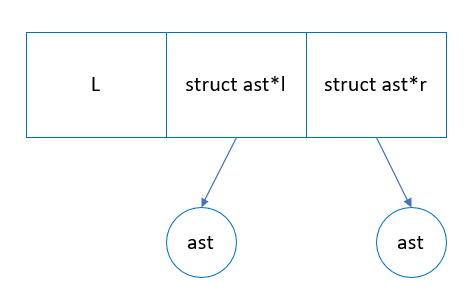


图21 语句列表节点类型

## 6.5 内置函数节点

内置函数节点类型为’F’，内含指明函数类型的整型数和指向参数的指针：

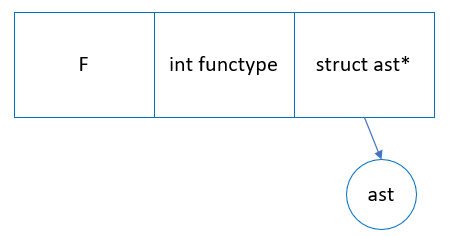


图22 内置函数节点类型

## 6.6 用户自定义函数节点

用户自定义函数节点的节点类型为’C’，内含指向符号节点的指针来寻找函数名和指向参数的指针：

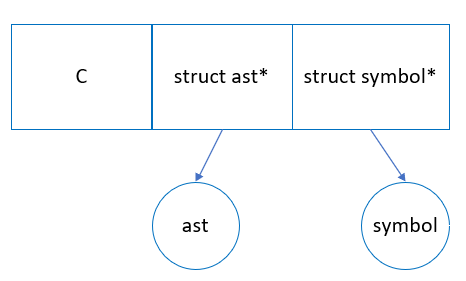


图23 用户自定义函数节点类型

## 6.7 控制语句节点

控制语句节点的节点类型可以为’I’或者’W’，分别对应的if\_then\_else语句和while\_do语句，其包含三个指向子节点的指针，分别指向条件、分支操作1、分支操作2，当然分支操作2只有在else情况下不为空：

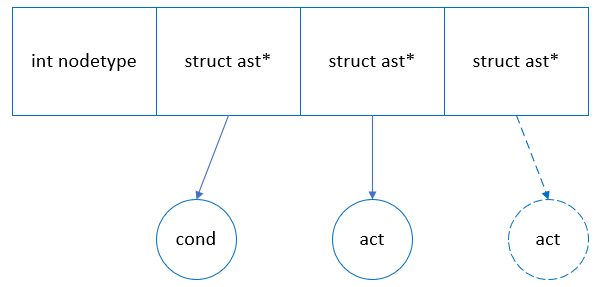


图24 控制语句节点类型

## 6.8 引用节点

引用节点类型为’N’，内含指向一个符号节点的指针，即是对其的引用：

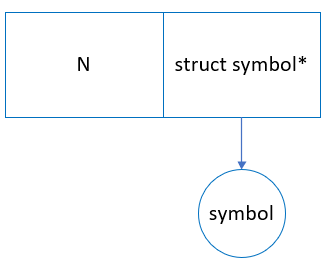


图25 引用节点类型

## 6.9 赋值节点

赋值节点类型为’=’，内含指向一个符号节点的指针，即是对其的赋值，还有一个指向普通节点的指针，其结果值赋值给符号节点代表的符号：

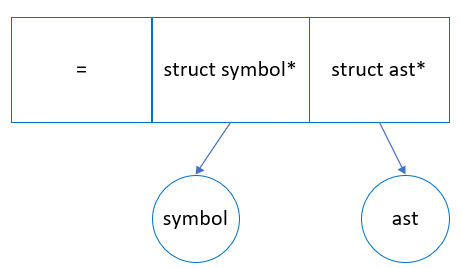


图26 赋值节点类型

# 7 实验步骤

## 7.1 Windows环境

(1) 打开Developer Command Prompt for VS 2019命令窗口，依次输入e:、cd E:\Flex\GnuWin32\bin进入安装目录下，再依次输入bison -d fb3-2.y、flex -ofb3-2.lex.c fb3-2.l，生成fb3-2.tab.h、fb3-2.tab.c和fb3-2.lex.c文件。

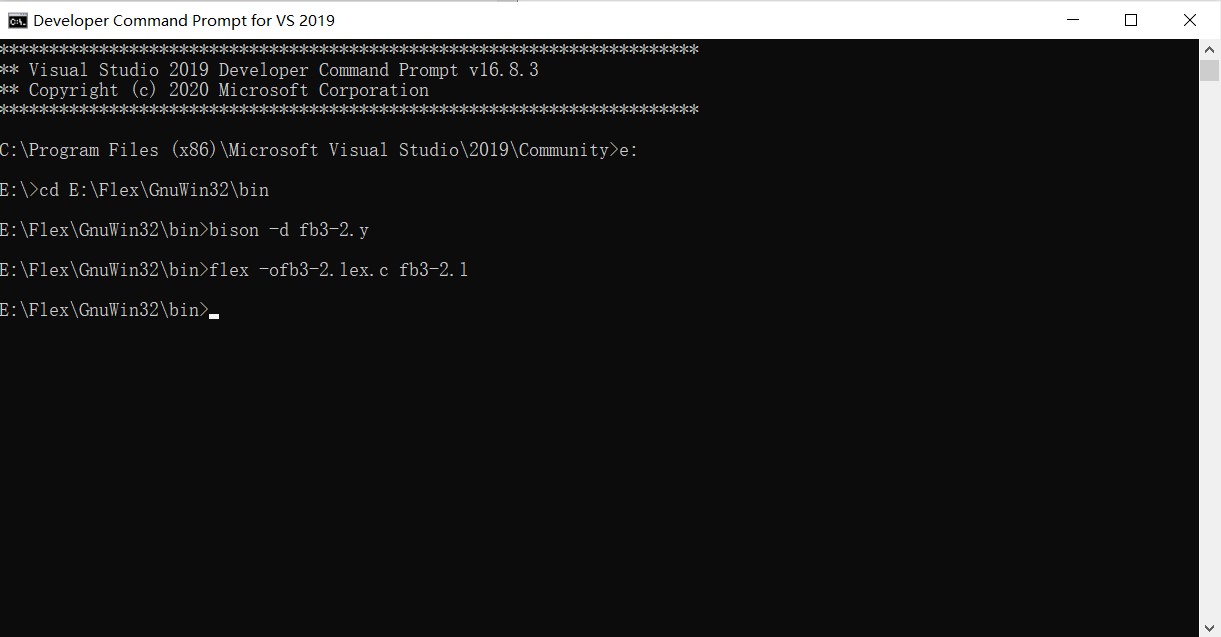


图27 输入命令

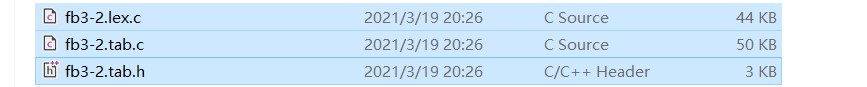


图28 生成文件

(2) 打开Cygwin命令窗口，输入cd E:/Flex/GnuWin32/bin进入所在目录，再输入命令gcc -o”fb3-2.tab.exe” fb3-2.tab.c fb3-2.lex.c fb3-2funcs.c -lm生成其对应的fb3-2.tab.exe文件。

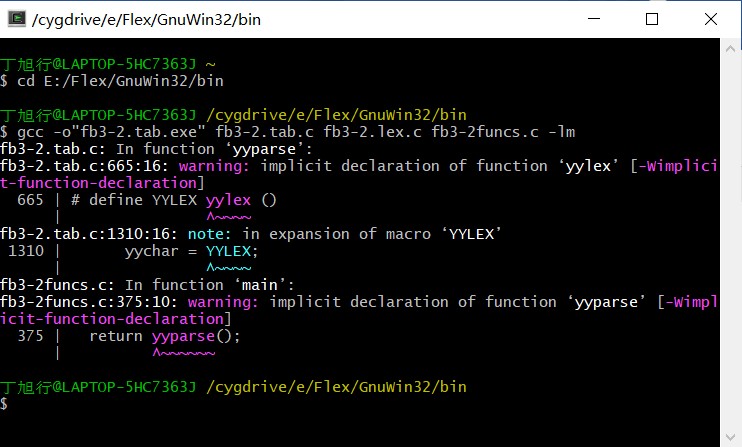


图29 输入命令

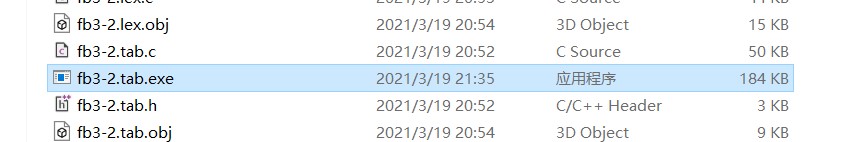


图30 生成文件

(3) 再回到命令窗口输入fb3-2.tab.exe运行fb3-2.tab.exe文件，输入计算式示例，回车后执行计算任务并输出计算结果。

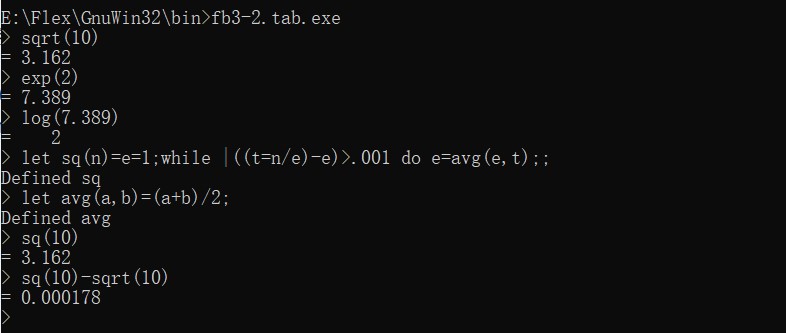


图31 运行程序及结果显示

(4) 为了新增一个新的内置函数pow(a,n)来计算一个数a的n次方，则需要增加一些代码。在fb3-2.h声明文件内增加pow(a,n)内置函数标识；在fb3-2.l文件中增加对字符串“pow”的识别，返回FUNC；在fb3-2funcs.c文件中对于内置函数的计算函数中增加对于pow(a,n)函数的处理代码。

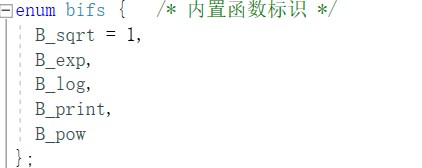


图32 fb3-2.h声明文件改动

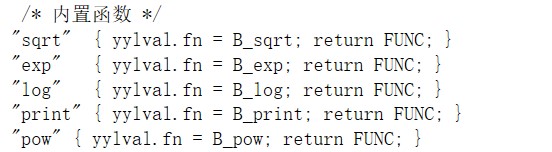


图33 fb3-2.l文件代码改动

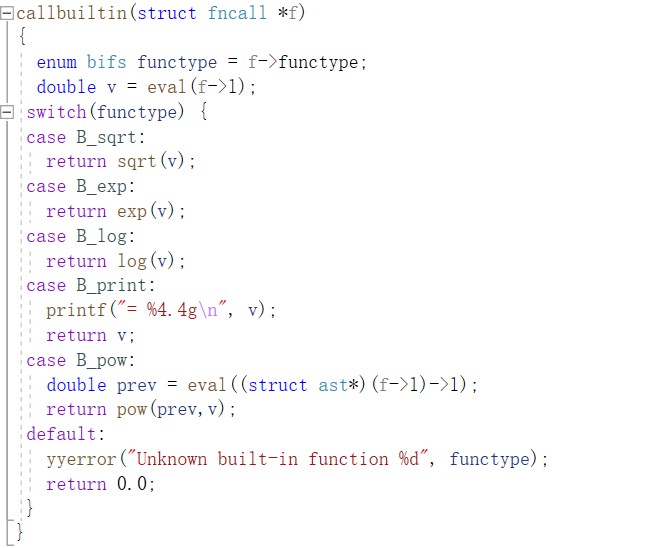


图34 fb3-2funcs.c文件代码改动

(5) 代码修改后，重复上述步骤重新生成exe文件并运行测试pow(a,n)函数。

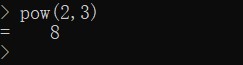


图35 测试内置函数

## 7.2 Linux环境

(1) 打开命令窗口，输入cd Desktop命令进入桌面目录，再依次输入bison -d fb3-2.y、flex -ofb3-2.lex.c fb3-2.l命令生成fb3-2.tab.c、fb3-2.tab.h和fb3-2.lex.c文件。

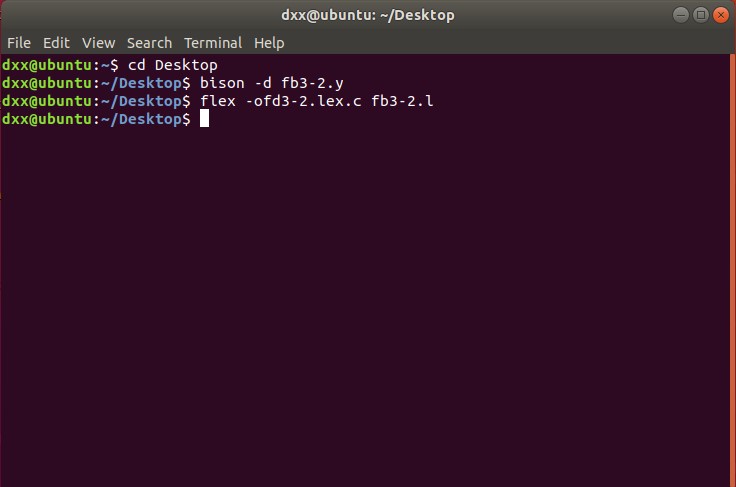


图36 输入命令

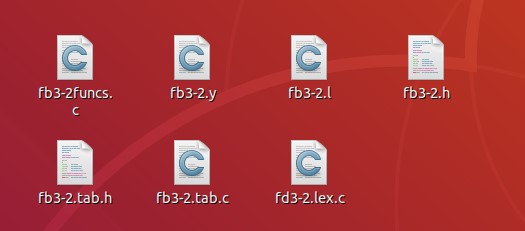


图37 生成文件

(2) 继续输入命令cc fb3-2.tab.c fb3-2.lex.c fb3-2funcs.c -lm进行编译工作，生成out文件。

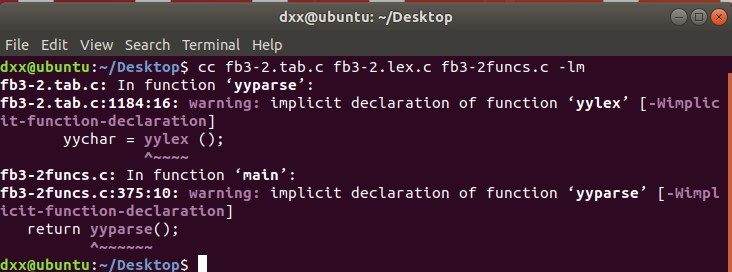


图38 输入命令

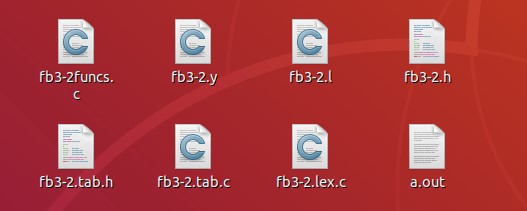


图39 生成文件

(3) 在命令窗口输入./a.out，调用生成文件，输入计算式示例，回车后执行计算任务并输出计算结果。

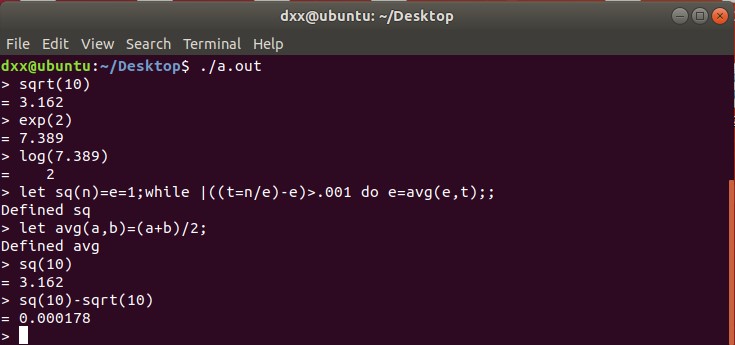


图40 运行程序及结果显示

(4) 为了新增一个新的内置函数pow(a,n)来计算一个数a的n次方，其具体过程与Windows环境下一致，在此不再赘述，下图为测试结果：

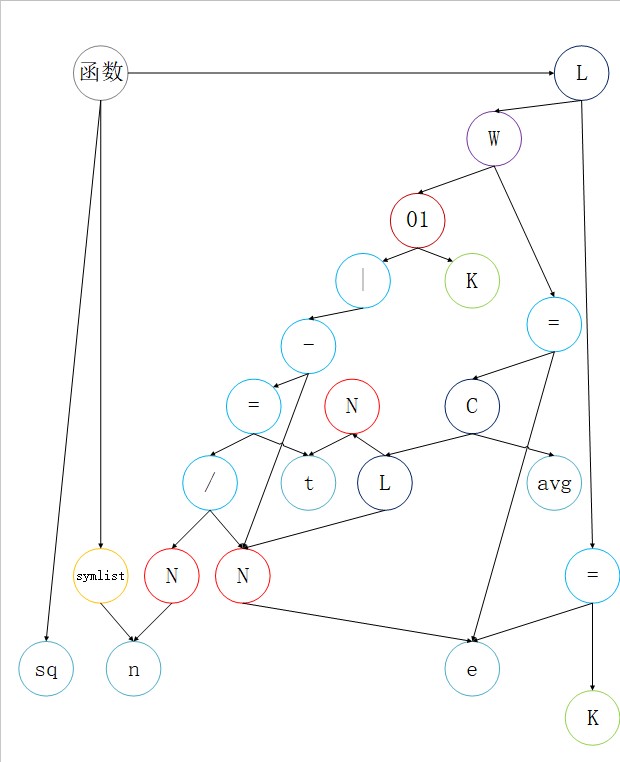


图41 测试内置函数

# 8 分析抽象语法树构建过程

根据运行结果，现以“let sq(n)=e=1;while |((t=n/e)-e)>.001 do e=avg(e,t);;”为例分析其抽象语法树的构建过程：

1. 读入‘let’，语法分析器接受词法分析器传来的‘LET’，并移进。
2. 读入‘sq’，语法分析器接受词法分析器传来的‘NAME’，构建其符号节点，并移进。
3. 读入‘(’，语法分析器接受词法分析器传来的‘(’，并移进。
4. 读入’n’，语法分析器接受词法分析器传来的’NAME’， 构建其符号节点，并归约为symlist，构建节点类型为‘S’的参数列表节点。
5. 读入‘)’，语法分析器接受词法分析器传来的‘)’，并移进。
6. 读入‘=’，语法分析器接受词法分析器传来的‘=’，并移进。
7. 读入’e’，语法分析器接受词法分析器传来的’NAME’，构建其符号节点，并移进。
8. 读入‘=’，语法分析器接受词法分析器传来的‘=’，并移进。
9. 读入1，语法分析器接受词法分析器传来的NUMBER，并归约为exp，构建节点类型为‘K’的子节点，此时NAME ‘=’ exp归约为exp，构建节点类型为’=’的子节点，(7)中符号节点成为其符号节点，(9)中’K’类型节点为其值节点。最后exp归约为stmt。
10. 读入‘;’，语法分析器接受词法分析器传来的‘;’，并移进。
11. 读入‘while’，语法分析器接受词法分析器传来的‘WHILE’，并移进。
12. 读入‘|’，语法分析器接受词法分析器传来的‘|’，并移进。
13. 读入‘(’，语法分析器接受词法分析器传来的‘(’，并移进。
14. 读入‘(’，语法分析器接受词法分析器传来的‘(’，并移进。
15. 读入’t’，语法分析器接受词法分析器传来的’NAME’，构建其符号节点，并移进。
16. 读入‘=’，语法分析器接受词法分析器传来的‘=’，并移进。
17. 读入’n’，语法分析器接受词法分析器传来的’NAME’， 构建其符号节点，并归约为exp，构建类型为’N’的符号引用节点，(17)中符号节点为其符号节点。
18. 读入‘/’，语法分析器接受词法分析器传来的‘/’，并移进。
19. 读入’e’，语法分析器接受词法分析器传来的’NAME’， 构建其符号节点，并归约为exp，构建类型为’N’的符号引用节点，(19)中符号节点为其符号节点。之后exp ‘/’ exp归约为exp，构建类型为’/’的子节点，(17)(19)中的‘N‘类型节点为其子节点。最后NAME ‘=’ exp归约为exp，构建类型为’=’的子节点，(15)中符号节点成为其符号节点，(19)中’/’类型节点为其值节点。
20. 读入‘)’，语法分析器接受词法分析器传来的‘)’，并移进。之后’(’ exp ‘)’归约为exp。
21. 读入‘-’，语法分析器接受词法分析器传来的‘-’，并移进。
22. 读入’e’，语法分析器接受词法分析器传来的’NAME’，构建其符号节点，并移进。之后归约为exp，构建类型为‘N‘的符号引用节点，(22)中符号节点为其符号节点。最后exp ’-‘ exp归约为exp，构建类型为’-‘的子节点，(19)中的’=‘节点和(22)中的’N’节点为其子节点。
23. 读入‘)’，语法分析器接受词法分析器传来的‘)’，并移进。之后’(’ exp ‘)’归约为exp。’|‘ exp归约为exp构建类型为’|’的子节点，(22)中的’-‘节点为其子节点。
24. 读入‘>’，语法分析器接受词法分析器传来的CMP，并移进。
25. 读入.001，语法分析器接受词法分析器传来的NUMBER，并归约为exp，构建节点类型为‘K’的子节点。exp CMP exp归约为exp构建类型为‘01‘的子节点，(23)中的’|‘节点(25)中的’K‘节点为其子节点。
26. 读入‘do’，语法分析器接受词法分析器传来的‘DO’，并移进。
27. 读入’e’，语法分析器接受词法分析器传来的’NAME’，构建其符号节点，并移进。
28. 读入‘=’，语法分析器接受词法分析器传来的‘=’，并移进。
29. 读入’avg’，语法分析器接受词法分析器传来的’NAME’，构建其符号节点，并移进。
30. 读入‘(’，语法分析器接受词法分析器传来的‘(’，并移进。
31. 读入’e’，语法分析器接受词法分析器传来的’NAME’，构建其符号节点，并移进，再归约为exp，构建节点类型为‘N’的符号引用节点。
32. 读入‘,’，语法分析器接受词法分析器传来的‘,’，并移进。
33. 读入’t’，语法分析器接受词法分析器传来的’NAME’，构建其符号节点，并移进，再归约为exp，构建节点类型为‘N’的符号引用节点，最后归约为explist。之后exp ‘,‘ explist归约为explist构建类型为’L‘的子节点，(31)(33)中的’N‘类型节点为其子节点。
34. 读入‘)’，语法分析器接受词法分析器传来的‘)’，并移进。NAME ’(‘ explist ‘)’归约为exp，构建类型为‘C’的节点，(29)中符号节点(33)中‘L‘节点为其子节点。最后NAME ‘=’ exp归约为exp构建类型为’=‘的节点，进而归约为stmt。
35. 读入‘;’，语法分析器接受词法分析器传来的‘;’，并移进。stmt ‘;’ list归约为list最后WHILE exp DO list规约为stmt构建类型为‘W’的节点。
36. 读入‘;’，语法分析器接受词法分析器传来的‘;’，并移进。stmt ‘;’ list归约为list，最后stmt ’;‘ list规约为list构建类型为’L‘的节点。
37. 读入‘\n’，语法分析器接受词法分析器传来的‘EOL’，最后calclist LET NAME ’(’ symlist ’)‘ ’=‘ list EOL归约为calclist开始符号，定义一个函数给到(2)中的符号节点。



图表 42 抽象语法树1

根据运行结果，现以“sqrt(10)”为例分析其抽象语法树的构建过程：

1. 读入‘sqrt’，语法分析器接受词法分析器传来的‘FUNC’，并移进。
2. 读入‘(’，语法分析器接受词法分析器传来的‘(’，并移进。
3. 读入10，语法分析器接受词法分析器传来的NUMBER，并归约为exp，构建节点类型为‘K’的子节点，进而归约为explist。
4. 读入‘)’，语法分析器接受词法分析器传来的‘)’，FUNC ‘(’ explist ‘)’归约为exp，并构建节点类型为‘F’的子节点，(3)中的‘K’类型子节点成为其左子节点，最后再归约为stmt。
5. 读入‘\n’，语法分析器接受词法分析器传来的‘EOL’，最后calclist stmt EOL归约为calclist开始符号，开始计算结果并打印结果数值。

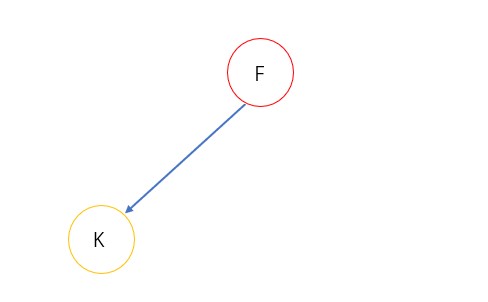


图43 抽象语法树2

# 9 实验总结

在本次实验中使用Flex和Bison在基本计算器的基础上添加变量命名、实现赋值、比较运算、流程控制、内置函数、用户自定义函数和简单错误恢复机制功能，使其成为一个具有全部功能的桌面计算器，虽然短小但是极具现实意义。通过阅读《Flex&Bison》第三章，加深学习抽象语法树的用法，熟练地掌握了抽象语法树的构造过程。由于有了上次实验的经验，本次对于代码的阅读以及为增加新内置函数pow(a,n)而修改增加代码变得简单了起来。但是，在实验的过程中还是遇到了一些编译错误的问题，但通过搜索相关资料，这些问题都得以解决。通过本次实验开发了一个具有全部功能的高级计算器，深刻理解了抽象语法树的用法以及Flex和Bison的代码语法，也激起了我的学习兴趣，相信以后的实验会更加顺利。