**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3-1 基本计算器

报告时间 2021年3月17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**](#_Toc67733233)

[**2 实验内容 1**](#_Toc67733234)

[**3 实验要求 1**](#_Toc67733235)

[**4 fb3-1.h文件分析 1**](#_Toc67733236)

[**5 fb3-1.l文件分析 2**](#_Toc67733237)

[**6 fb3-1.y文件分析 2**](#_Toc67733238)

[**7 节点分析 3**](#_Toc67733239)

[**7.1 普通节点 3**](#_Toc67733240)

[**7.2 数值节点 4**](#_Toc67733241)

[**8 实验步骤 4**](#_Toc67733242)

[**8.1 Windows环境 4**](#_Toc67733243)

[**8.2 Linux环境 6**](#_Toc67733244)

[**9 分析抽象语法树构建过程 8**](#_Toc67733245)

[**10 实验总结 9**](#_Toc67733246)

# 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fb3-1.y、fb3-1.l、fb3-1funcs.c、fb3-1.h文件；

(3) 撰写实验报告，结合实验结果，如(1+2)-(2\*6)、1+2-3\*2/5，给出抽象语法树的构建过程；

(4) 提交报告和实验代码。

# 4 fb3-1.h文件分析

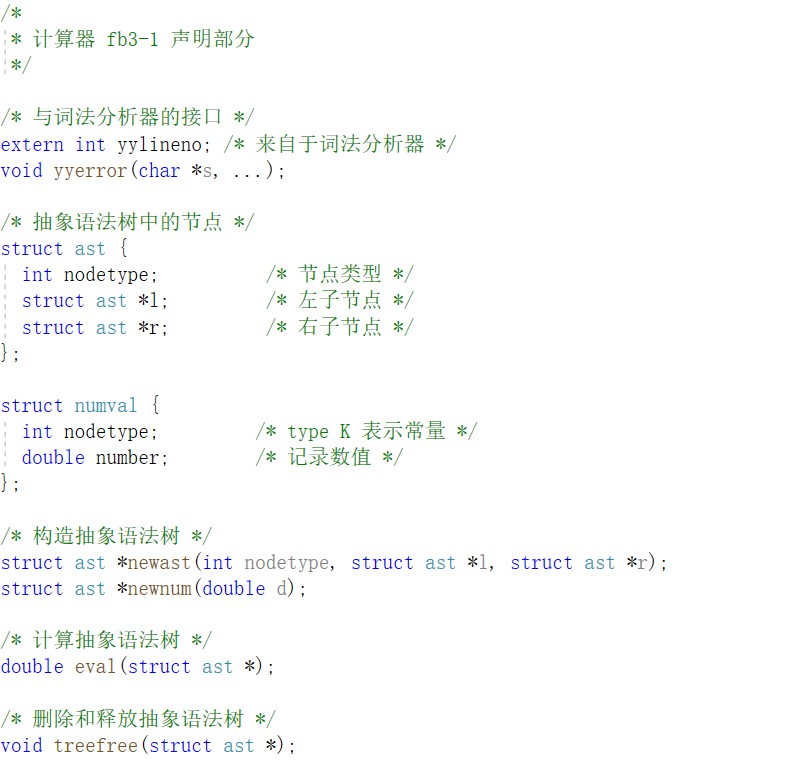


图1 fb3-1.h文件

# 5 fb3-1.l文件分析

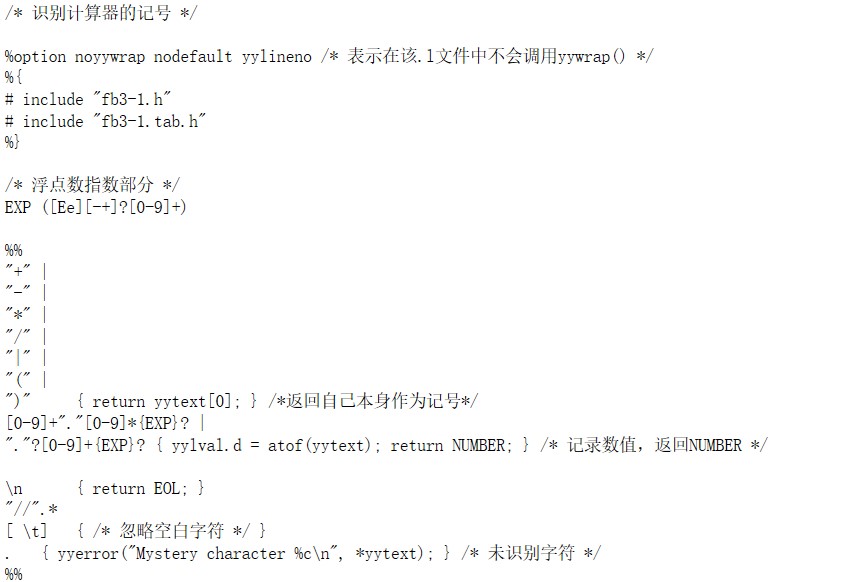


图2 fb3-1.l文件

# 6 fb3-1.y文件分析



图3 fb3-1.y文件1

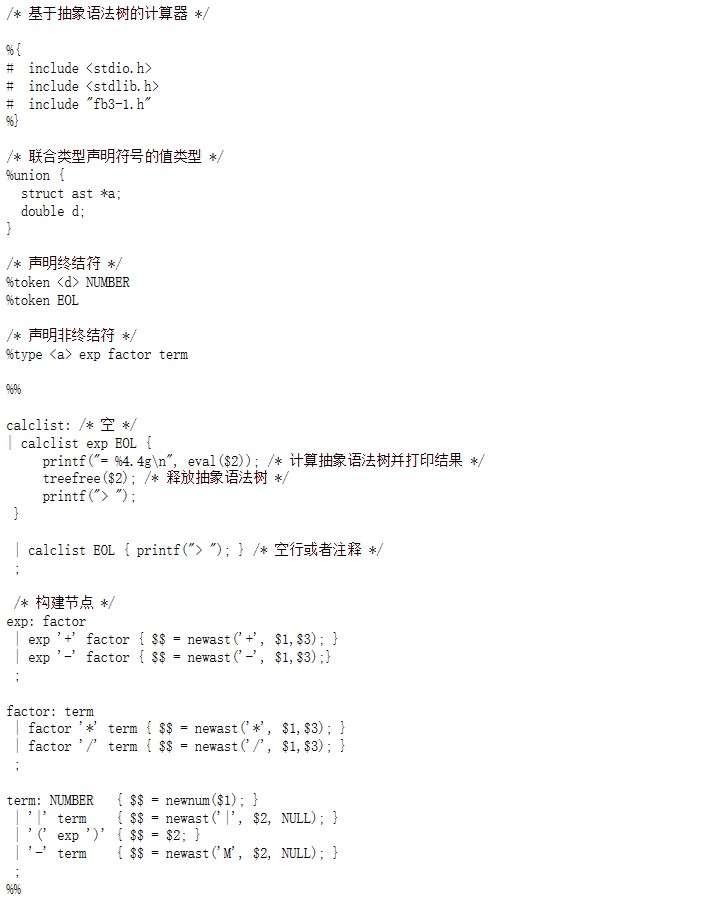


图4 fb3-1.y文件2

# 7 节点分析

## 7.1 普通节点

普通节点包含有自身节点类型值以及其左右子节点：

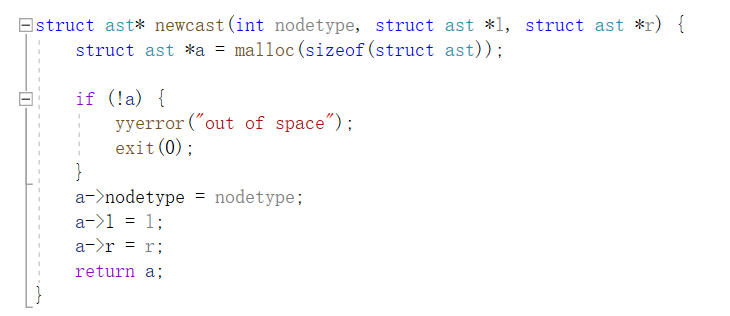


图5 构建普通节点

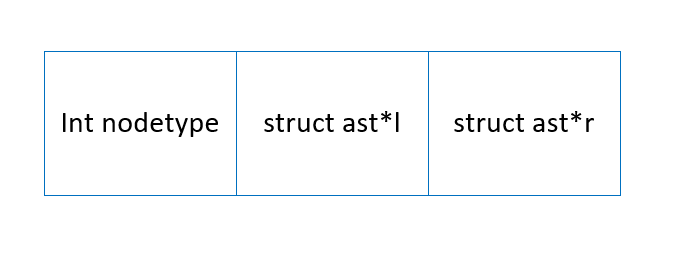


图6 普通节点类型

## 7.2 数值节点

数值节点的节点类型为’K’，自身有double类型的数值：

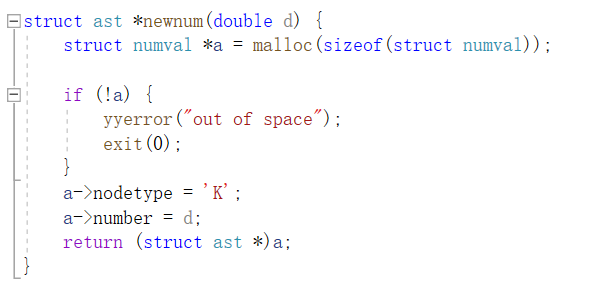


图7 构建数值节点

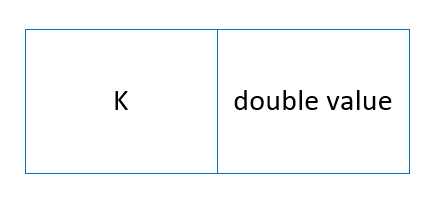


图8 数值节点类型

# 8 实验步骤

## 8.1 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 fb3-1.y、flex -ofb3-1.lex.c fb3-1.l，生成fb3-1.tab.h、fb3-1.tab.c和fb3-1.lex.c文件。

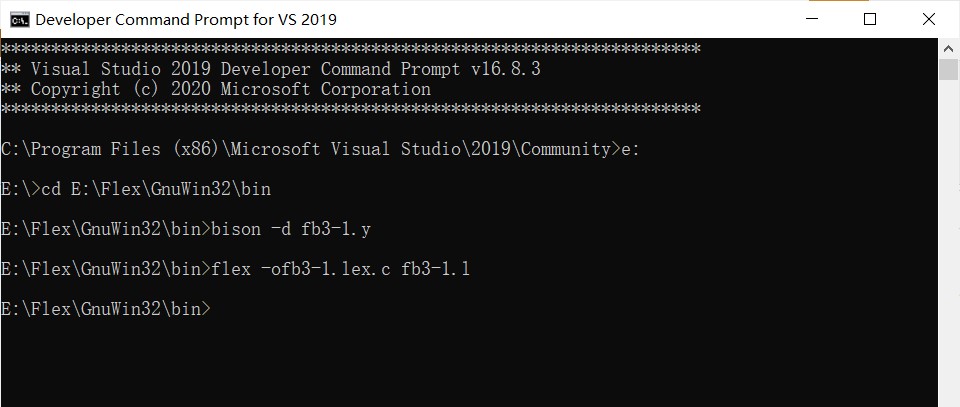


图9 输入命令

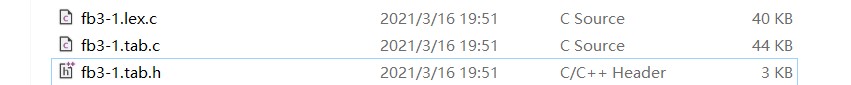


图10 生成文件

(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”fb3-1.tab.exe” fb3-1.tab.c fb3-1.lex.c fb3-1funcs.c -lm生成其对应的fb3-1.tab.exe文件。

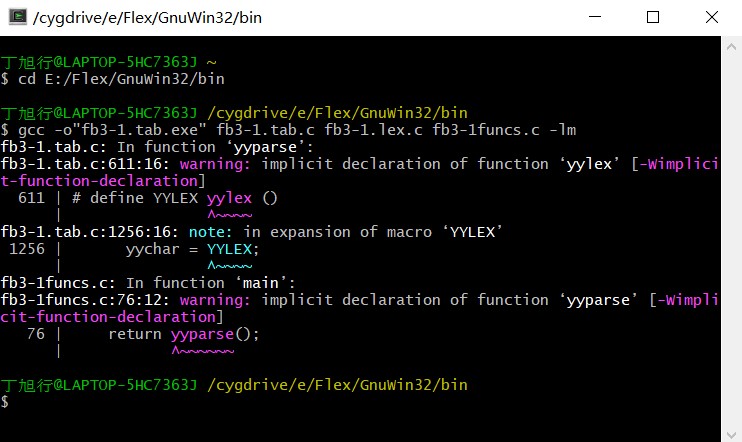


图11 输入命令

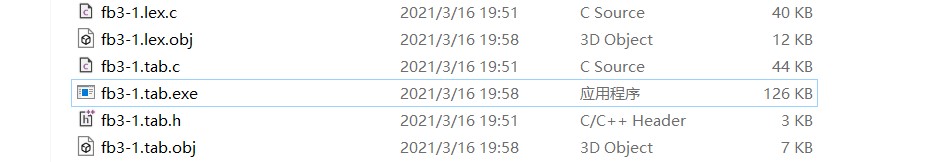


图12 生成文件

(3) 再回到命令窗口输入fb3-1.tab.exe运行fb3-1.tab.exe文件，输入计算式示例，回车后输出计算结果。

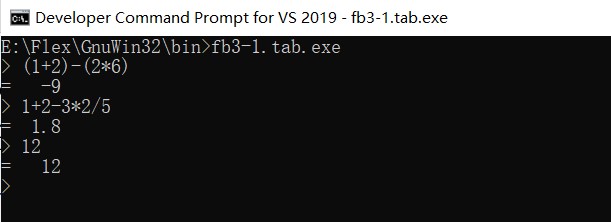


图13 运行程序及结果显示

## 8.2 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 fb3-1.y、flex -ofb3-1.lex.c fb3-1.l命令生成fb3-1.tab.c、fb3-1.tab.h和fb3-1.lex.c文件。

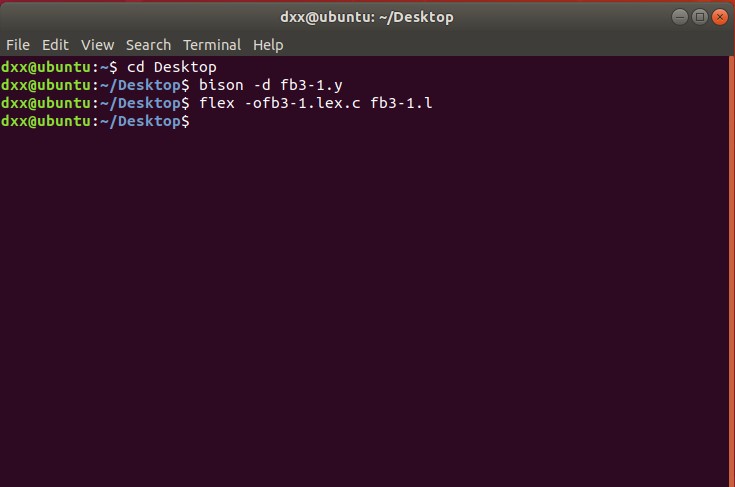


图14 输入命令

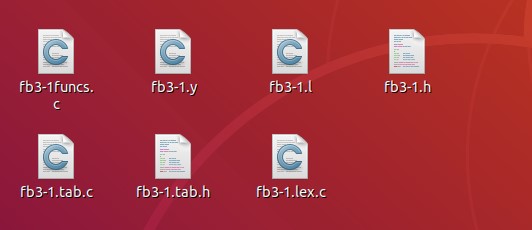


图15 生成文件

(2) 继续输入命令cc fb3-1.tab.c fb3-1.lex.c fb3-1funcs.c -lm进行编译工作，生成out文件。

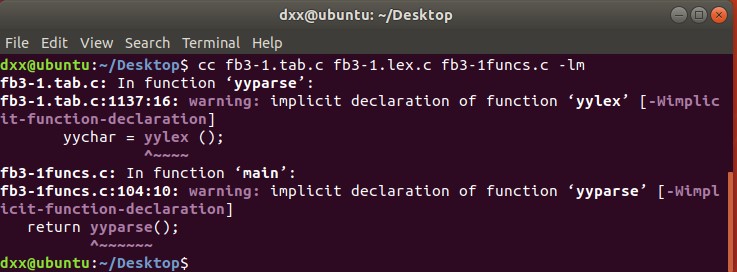


图16 输入命令

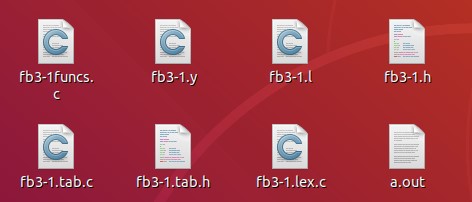


图17 生成文件

(3) 在命令窗口输入./a.out，调用生成文件，输入计算式示例，回车后输出计算结果。

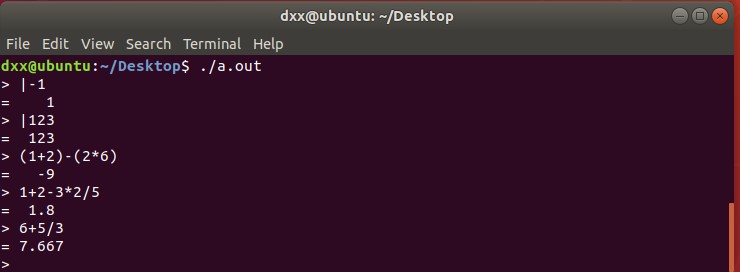


图18 运行程序及结果显示

# 9 分析抽象语法树构建过程

根据运行结果，现以“(1+2)-(2\*6)”为例分析其抽象语法树的构建过程：

(1) 读入‘(’，语法分析器接受词法分析器传来的‘(’，并移进。

(2) 读入1，语法分析器接受词法分析器传来的NUMBER，并归约为term，构建节点类型为‘K’的子节点，进而归约为factor，最后归约为exp。

(3) 读入‘+’，语法分析器接受词法分析器传来的‘+’，并移进。

(4) 读入2，语法分析器接受词法分析器传来的NUMBER，并归约为term，构建节点类型为‘K’的子节点，进而归约为factor。最后exp ‘+’ factor归约为exp，构建节点类型为‘+’的子节点，(2)(4)中的‘K’子节点成为其左右子节点。

(5) 读入‘)’，语法分析器接受词法分析器传来的‘)’，‘(’ exp ‘)’归约为term，进而归约为factor，最后归约为exp。

(6) 读入‘-’，语法分析器接受词法分析器传来的‘-’，并移进。

(7) 读入‘(’，语法分析器接受词法分析器传来的‘(’，并移进。

(8) 读入2，语法分析器接受词法分析器传来的NUMBER，并归约为term，构建节点类型为‘K’的子节点，进而归约为factor。

(9) 读入‘\*’，语法分析器接受词法分析器传来的‘\*’，并移进。

(10) 读入6，语法分析器接受词法分析器传来的NUMBER，并归约为term，构建节点类型为‘K’的子节点。最后factor ‘\*’ term归约为factor，构建节点类型为‘\*’的子节点，(8)(10)中的‘K’子节点成为其左右子节点，之后factor归约为exp。

(11) 读入‘)’，语法分析器接受词法分析器传来的‘)’，‘(’ exp ‘)’归约为term，进而归约为factor。之后exp ‘-’ factor归约为exp，构建节点类型为‘-’的子节点，(4)(10)中的‘+’子节点和‘\*’子节点成为其左右子节点。

(12) 读入‘\n’，语法分析器接受词法分析器传来的‘EOL’，最后calclist exp EOL归约为calclist开始符号，开始计算结果并打印。

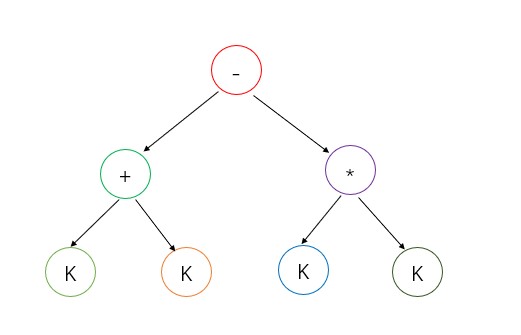


图19 抽象语法树

# 10 实验总结

在本次实验中使用Flex和Bison开发了一个具有基本功能的桌面计算器，支持四则运算，取绝对值，括号和负号，使它成为一个虽然短小但是具有现实意义的编译器。通过阅读《Flex&Bison》第三章，学习了抽象语法树的用法，掌握了抽象语法树的构造过程。但是，在实验的过程中还是遇到了一个小问题，VS的cl编译器无法执行-lm命令连接数字库，所以我改用了Cygwin的gcc编译器，最后成功生成了exe文件。由于有了上次实验阅读大段代码的经验，本次的代码理解起来更加的轻松，也为以后的实验打下了基础。