**语法分析器**

### 一、背景

完成CMM语言的语法分析器,生成语法树和中间代码

姓名:袁浩

学号:2013302580115

班级:软工三班

日期:2015-11-18

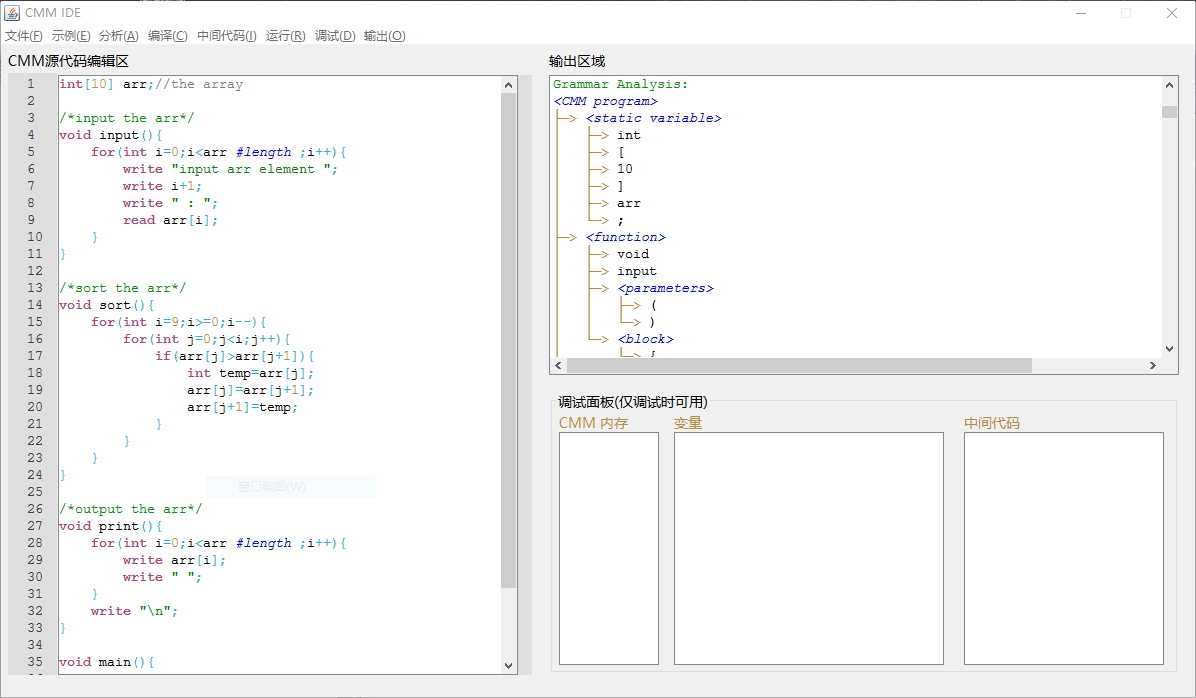
平台:Windows 10 th2 build 10586

开发工具:JDK8,Eclipse MARS.1,Intellij IDEA 15 Ultimate

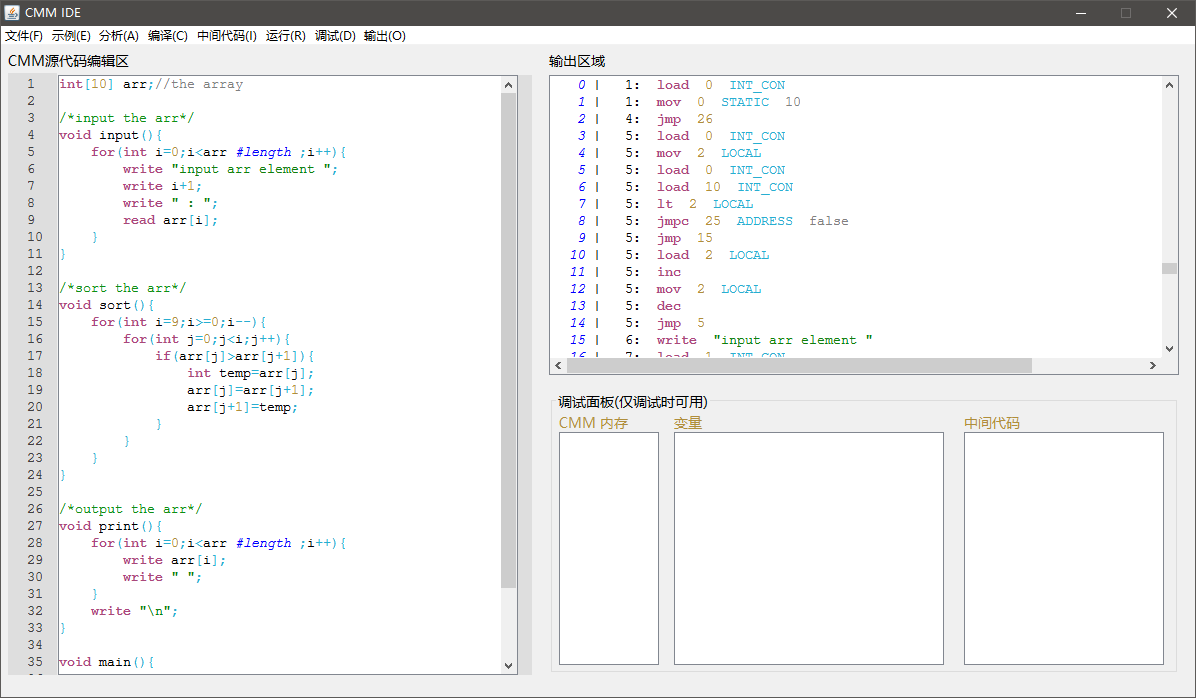
### 二、运行效果

运行效果:

菜单->分析->语法分析(Ctrl+G)



菜单->编译->编译/生成中间代码(F7)



### 三、功能实现描述

实现CMM语言的语法分析器,生成语法树和中间代码,夹杂着语义分析和错误处理

1. 使用的方法

自顶向下的分析方法.

1. CMM语法和文法

我的CMM语法不太一样,请参见CMM语法说明.txt和cmm.jj

1. 主要的算法和数据结构

语法分析程序主要存在于com.yuanhonglong.analysis.GrammarAnalysis中,语法分析器需要使用到词法分析器,主要的分析结构是函数,变量和语句.

辅助的类有:

Symbol,符号

Function,函数

SymbolTable,符号表

CMMVariable,变量

VariableType,变量数据类型

ReturnVariable,函数返回变量

ReturnType,函数返回变量类型

VariableAddress,变量名称列表,用于调试的时候记录变量的地址和名称映射

InternalCode,中间代码

CMMOperator,中间代码操作数类型

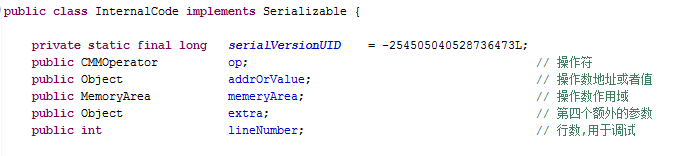
GrammarError,语法错误

GrammarTree,语法树

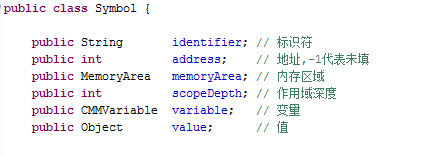
MemoryArea,内存区域

主要的数据结构有

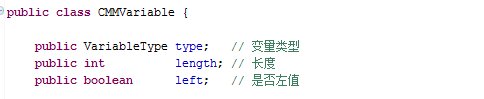
中间代码结构



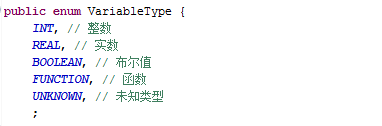
符号结构



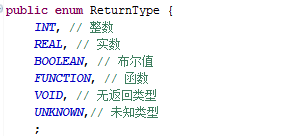
变量结构



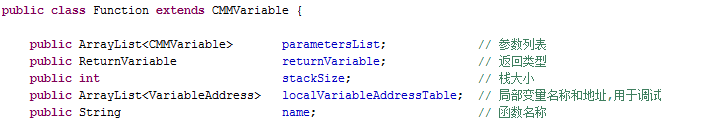
变量类型



返回类型



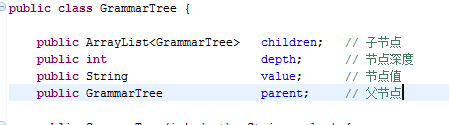
函数结构



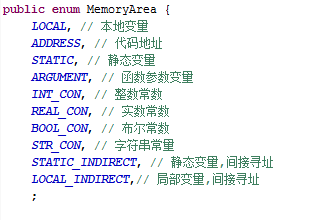
操作符类型:



语法树结构



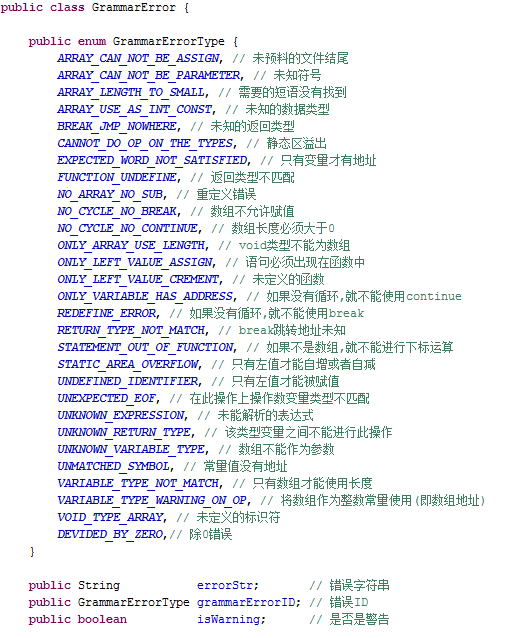
内存区域



语法分析主程序结构



语法错误与警告

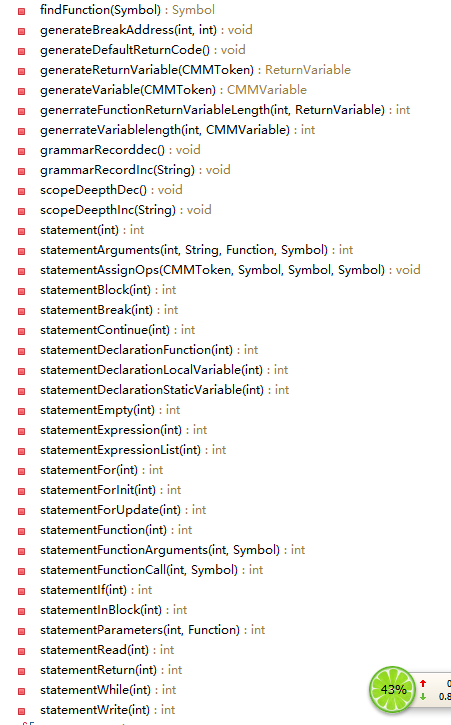


1. 分析过程

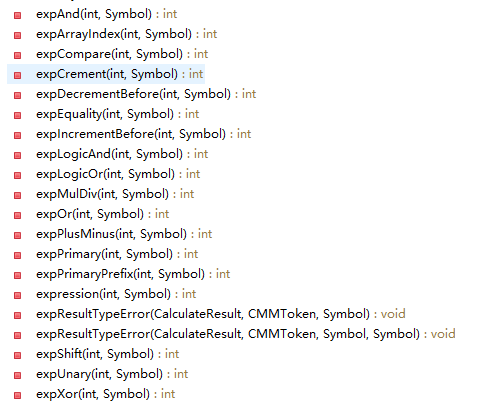
分析过程从GrammarAnalysis的startAnalysis函数开始

每一次读取一个短语,根据自顶向下分析方法,根据每个文法产生式的开始符号集合和后跟符号集合还有各个语法结构的结束符号集合(比如分号和右大括号),判断进入哪一个语法分析子程序,主要的语法分析子程序有:函数,块,语句,表达式等等

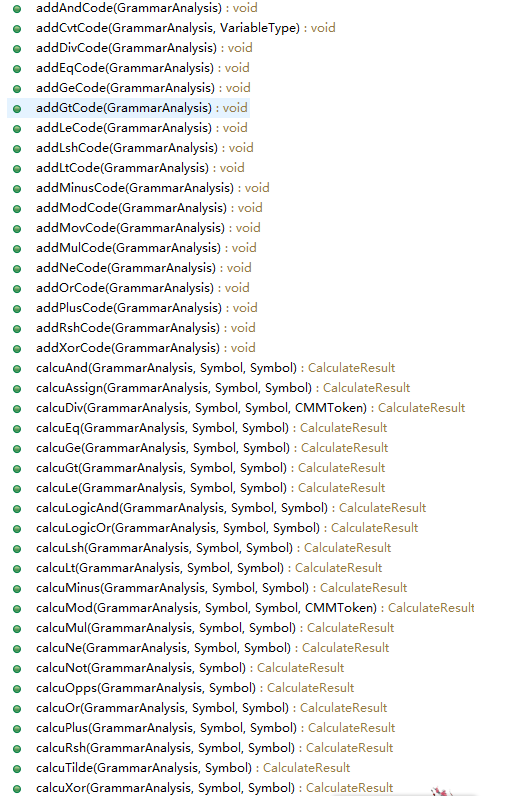
以下是各个子程序



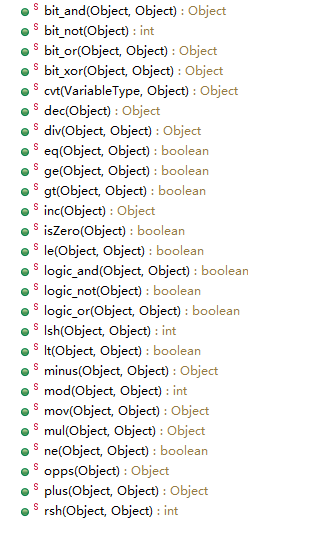
还有表达式



表达式的计算还涉及到数据类型和左值右值以及常量变量的区别,这些语义分析的内容夹杂在语法分析中,并且抽离到了Symbol类中(因为GrammarAnalysis类太大了),如下:



真正进行表达式计算的类实际上是Calculator类



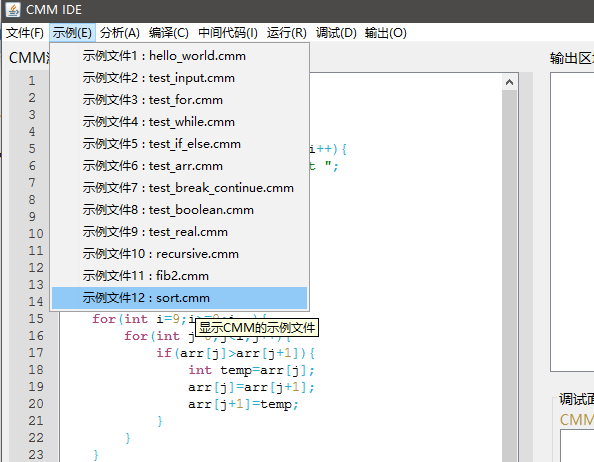
1. 调试数据和结果

在提供的CMMParser.jar文件中已经包含了一些例子,比如sort.cmm是一个用cmm写的排序程序,fib2.cmm是一个用cmm写的计算斐波那契数列的程序.

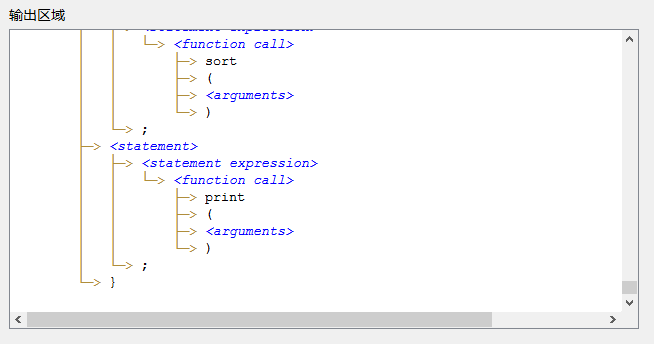
具体使用方法是

菜单->示例

比如我们可以选择sort.cmm



菜单->分析->语法分析结果如下

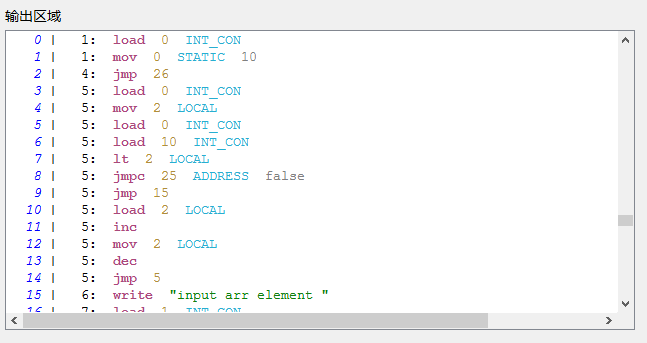


由于分析结果比较复杂,需要拖动滚动条才可以看到,所以请亲自尝试.

当然也可以选择

菜单->编译->编译/生成中间代码

结果为:



同样请亲自尝试

1. 一些关键的语法结构
2. for语句

for语句右四个部分构成

for-init,condition,for-update和statement

for-init代表for语句初始化

condition就是一个表达式,是循环继续的条件

for-update是循环更新语句

statement是for语句的主体

for语句最后会翻译成这样

for-init

condition

for-update

statement

1. if-else语句

为了让else语句匹配最近的if语句,在if语句解析完之后,立即查看后面是不是else语句,如果是,就立即解析

1. 函数声明

函数可以先声明再使用最后定义,声明的函数依然要加入符号表,只是这时候函数的地址和帧都是空的,如果用到了这个函数,那么最后需要返填

1. 前置++,--与后置++,--
2. 短路运算符&&,||
3. break语句

由于解析break语句的时候还没有到达循环结尾的地方,所以break语句的跳转地址也是需要返填的

### 四、总结

语法分析难度不小,因为在语法分析和设计中间代码的时候必须要考虑周全,否则最后会发现中间代码完全跑不起来,主要的难点如下

1. 函数

一旦决定支持函数,就必须自己设计函数栈,控制帧指针和栈顶指针以及返回地址,要考虑静态变量和局部变量,要考虑返回值,函数调用,形参和实参的处理

1. 表达式

支持的操作越多,操作数类型越多,表达式处理起来就越复杂,我的CMM语言几乎支持java中除了三元表达式和某种特殊的移位运算之外的所有运算符.

优先级与java相同

1. 寻址方式

由于数组的原因,必须支持间接寻址的方式,我的中间代码支持操作数立即寻址,直接寻址,间接寻址三种寻址方式.

1. 数据类型

数据类型越多则越复杂,我在int,real类型的基础上又支持了boolean类型

1. 流程控制

我支持了for语句,if-else,while

另外支持了break,continue和return

### 五、使用说明（可选）

运行需要JDK1.7及以上版本

请参见使用说明.txt

### 六、关于运行和调试

这一次实验还不需要运行和调试,但是相关的功能我已经实现,有兴趣的话请尝试一下.具体运行和调试的功能涉及到很多方面,下一次报告的时候再详述.

### 七、参考资料

界面的一些逻辑参考了eclipse

语言参考了java