编译原理PA1A实验报告

张钰晖 计55 2015011372 yuhui-zh15@mails.tsinghua.edu.cn

任务描述

本阶段需要编码实现Decaf语言编译器的词法分析和语法分析部分，同时生成抽象语法树。

本阶段需要借助词法分析工具JFlex和语法分析工具BYACC。

文件说明

在本阶段，以下文件非常重要，主要需要修改以下文件。

文件名	含义	说明
Lexer.l	JFlex程序	使用正则表达式新增对关键字的识别
Parser.y	BYACC程序	使用上下文无关语法新增语法规则和归约动作
Tree.java	抽象语法树各种节点	新增抽象语法树节点所需的类

- 词法分析阶段：修改Lexer.l实现。
  - Lexer.l主要用正则表达式和一些规则匹配出代码中的关键字
- 语法分析阶段：修改Parser.l和Tree.java实现。
  - Parsel.l主要可以理解为上下文无关语法，其每个非终结符均为SemValue类，SemValue类是Tree各种类的整合体（类似指针）。
  - Tree.java文件主要包含了抽象语法树各种节点。

单词符号说明

- 关键字：语言保留字，如"int", "void"...
- 标识符：字母开头，后跟字母数字下划线，如"my_var123"...
- 常量：整数、布尔、字符串，如"123"...
- 算符和界符：单字符和双字符，如"+", "-"...
- 注释：单行注释，"//"开头

实验内容及实现

在Decaf语言的基础上新增以下特性，主要修改列于表格之中。

1. 整复数类型的支持

本阶段需要增加整复数类型，增加识别复数常量虚部功能，增加获取复数实部、虚部及强制转换复数表达式，增加复数打印语句。

文件名	修改
Lexer.l	新增关键字"complex","@", "$", "#","PrintComp"
	新增正则表达式识别复数虚部
BaseLexer.java	新增函数imgConst()识别复数虚部
Parser.y	新增终结符"COMPLEX","@", "$", "#","PRINTCOMP"
	新增规则Type ::= COMPLEX
	新增规则Expr ::= @Expr | $Expr | #Expr
	新增规则Stmt ::= PrintCompStmt
	新增规则PrintCompStmt ::= PRINTCOMP(Expr+)
	新增算符优先级"@", "$", "#"
Tree.java	修改类Unary，对运算符"@", "$", "#"支持
	修改类Literal，对复数虚部常量支持
	新增类PrintComp，对复数打印语句支持

2. case表达式的支持

本阶段需要支持case表达式，语法为case(表达式) {常量1:表达式1,..., default:表达式}。

文件名	修改
Lexer.l	新增关键字"case","default",":"
Parser.y	新增终结符"CASE","DEFAULT",":"
	新增规则Expr ::= CASE (Expr) {ACaseExprList DefaultExpr}
	新增规则ACaseExprList ::= ACaseExprList ACaseExpr | 空
	新增规则ACaseExpr ::= Constant: Expr;
	新增规则DefaultExpr ::= DEFAULT: Expr;
Tree.java	新增类CaseExpr，对case表达式支持
	新增类ACaseExpr，对case表达式单条语句支持
	新增类DefaultExpr，对case表达式default语句支持

3. super表达式的支持

本阶段需要支持super表达式，类似this表达式。

文件名	修改
Lexer.l	新增关键字"super"
Parser.y	新增终结符"SUPER"
	新增规则Expr ::= SUPER
Tree.java	新增类SuperExpr，对super表达式支持

4. 对象复制的支持

本阶段需要支持深复制dcopy()和浅复制scopy()表达式。

文件名	修改
Lexer.l	新增关键字"dcopy","scopy"
Parser.y	新增终结符"DCOPY","SCOPY"
	新增规则Expr ::= DCOPY(Expr) | SCOPY(Expr)
Tree.java	新增类DcopyExpr和ScopyExpr，支持对象复制

5. 串行循环卫士的支持

本阶段需要支持串行循环卫士语句，语法为do E1:S1|||E2:S2... od

文件名	修改
Lexer.l	新增关键字"do","od","|||"
Parser.y	新增终结符"DO","OD","|||"
	新增规则Stmt ::= DoStmt;
	新增规则DoStmt ::= DO DoBranchList DoSubStmt OD
	新增规则DoBranchList ::= DoBranchList DoBranch | 空
	新增规则DoBranch ::= DoSubStmt |||
	新增规则DoSubStmt ::= Expr : Stmt
Tree.java	新增类DoStmt，对串行循环卫士支持
	新增类DoSubStmt，对串行循环卫士子语句支持

技巧心得

1. 特殊产生式的支持

部分产生式可能需要递归，这时可以采用下列实现方式，递归尽量使用左递归，避免状态栈溢出。

产生式	实现
A*	B ::= B A | 空
A+	B ::= B A | A
A?	B ::= A | 空

2. 消除冲突

通过指定新增算法的优先级和结合性来消除冲突，优先级由声明结合性的位置决定，越往下运算符优先级越高。

结合性	实现
左结合	%left
右结合	%right
无结合	%nonassoc

总结

通过本次实验，笔者对词法分析和语法分析的认识有了质的提高。尽管亲自动手用代码实现这些分析过程开始是痛苦的，但是写完之后有一种豁然开朗之感。尽管这只是众多实验中微小的开头，但笔者在实践之中真正感受到了编译的神奇之处。