编译原理 PA1-B 作业报告

黄家晖 2014011330

PA1-B 的要求是构造 Decaf 语言的 LL(1) 文法，利用自顶向下的方法对语法进行解析，从而代替 PA1-A 中使用 Byacc 产生的 Praser.y 文件。

1 新增加的数据结构和函数

1. 在 Tree 中增加了一些数据结构，增加的部分和 PA1-A 相同，不再赘述；
2. 在 Paser 中增加了 CASE, CONTINUE, DEFAULT, REPEAT, SWITCH, UNTIL, PCLONE 的常量定义，使得文法分析器能够引用正确的语义值；
3. 修改了 StmtListParse 和 StmtParse 函数，为其增加了相应的 lookahead token 以及对于 token 需要进行的规约操作；
4. 增加了 OperCEParse 和 OperPCloneParse 函数，负责识别条件运算符 “?” 和 “«” 运算符；
5. 增加了 ExprCE(T)Parse 和 ExprPClone(T)Parse 函数，能够对消除左递归的算符树进行解析；
6. 增加了 ContinueStmtParse，SwitchStmtParse，CaseStmtListParse，CaseStmtParse，DefaultStmtParse，RepeatStmtParse 函数，实现对 Case 语句和 Repeat 语句的解析。

2 遇到的问题和解决方法

我认为此次实验除了某些地方的细节需要注意之外，比较复杂的地方仍然在于三元操作符`?:`的解析。课堂上老师并没有对 LL(1) 文法的结合性与优先级做出说明，实际上 LL(1) 文法在自顶向下分析的时候语义树都是使用最左推导得出的左结合。但是经过学习原有 `Parser.y`源代码可以得知，可以通过语法规则的层次划分（即化为 Expr1, Expr2, ... 优先级层层递增）

来解决优先级问题，每层中通过抽象语法树建立时的逻辑确定相同优先级运算符的左右结合性。

具体地，在解决?:问题的时候，可以将`? Expr :`看成一个运算符赋给其优先级和结合性，既可以轻松解决问题，这种思路参考了以下网站:

http://stackoverflow.com/questions/13681293/how-can-i-incorporate-ternary-operators-into-a-precedence-climbing-algorithm