编译原理 PA1 作业报告

黄家晖 2014011330

PA1 的要求是修改原有编译器的框架，为编译器增加新的语法特性，最终输出正确的抽象语法分析树。本次的作业涉及到了框架代码的阅读和 JFlex/byacc 工具的使用，整体较为简单。

1 新增加的数据结构和函数

1. 在 Tree 中增加 Ternary 语法节点，代表三元运算符，实现 condition expression 的功能，包含三个子节点：left、middle 和 right。
2. 在 Tree 中修改 Binary 语法节点，加入 `<<`二元运算符，与其他二元运算符并列。
3. 在 Tree 中增加 Case 语法节点，代表 `switch`语法块中的一个 `case`语句，其中包含了一个 Expr 节点（代表 `case`之后的常量）以及一些 Tree 节点（代表目标命中之后执行的语句列表)。
4. 在 Tree 中增加 Switch 语法节点，代表整个 `switch`语句块，其中包含了一些 Case 语法节点以及一些 Tree 节点（代表 `default`情况下需要执行的语句列表)。
5. 在 Tree 中增加 Repeat 语法节点，代表 `repeat...until`语句块，其子节点包括了一个 Expr 节点（代表 `until`之后的逻辑语句）和一些 Tree 节点（代表 repeat 之后需要执行的语句列表)。这里需要注意 repeat 的规则与 C 和 JAVA 的风格不太一致，根据 PA1 给出的参考语法，reapeat 之后可以是不加大括号的连续语句，这与 if 是不同的。
6. 在 Tree 中增加 Continue 语法节点，实现同 Break 语法节点，是叶子节点。
7. 修改了 SemValue 语义值类，添加了成员变量 `caselist`和 `casedef`，为了实现 Lexer 和 Parser 的交互添加了一些临时变量，用于存储 case 语句。
8. 为 Tree 和 SemValue 增加相应的 `visit()`和 `toString()`函数，使得程序风格统一，使用 diagnose 调试时更方便。

2 遇到的问题和解决方法

实现三元操作符?:的时候，在简单添加了 parser 生成式之后，byacc 报告 Shift/Reduce 冲突，同时程序不能够正确识别形如 `a?b:c?d:e`的表达式。为此，需要告知 byacc 三元表达式的右结合性，具体来说需要在适当位置加入`%right ':' '?'`语句。

为了更清楚地了解 byacc 解决 Shift（入栈）/Reduce（规约）冲突的办法，我在网上进行了相关搜索：

http://stackoverflow.com/questions/17904706/why-does-my-ternary-operator-cause-a-shift-reduce-conflict

上述网站中清楚地阐明了解决冲突的办法：即当找到符合条件的产生式之后，为了确定是否进行规约，需要比较产生式和下一个符号的优先级，而产生式的优先级与产生式右端的最后一个终结符相同，当定义%left 或是%right 之后（部分情况针对特定表达式需要加入%prec），实际上就能根据结合性和定义位置确定每个产生式和终结符的优先级，冲突得以解决。