编译原理第二次实验报告

**计45 谭思楠 2013011720**

编译原理的第二（1B）次实验主要练习手写LL（1）递归下降文法分析器。

# 沿用之前的部分

在本次实验中，我的词法分析器沿用上一次实验的设计，AST节点类也基本沿用上一次的设计。因此这些相关代码只需要直接平行迁移即可。

# 语法分析的改动

## 添加运二目运算符“<<”

在手写语法分析器的情况下，二目左结合运算符处理比三目右结合运算符麻烦，因此我们先讨论“<<”运算符。

“<<”是二目的左结合运算符，根据形式语言的相关经验，典型的处理运算符优先级的文法如下（Expr1为可能含低优先级运算符的表达式，Expr2为只含高优先级运算符）：

这个文法规则的主要问题是含左递归，消去左递归即可得到：

实际上所谓的Expr1Right在助教给的手写分析器中一般被叫做ExprT1，op1叫做Oper1。

这个改动有一个很大的问题，是会造成语义的改变。我们期待的是左结合运算符，但是左结合运算符必然会导致左递归，变换完的文法则必然是右结合文法。

为了解决这个问题，助教例子中的办法是，将右结合文法递归分析的结果在低优先级运算符节点展开成一个列表，然后重新按左结合文法构造一个文法树。

助教的代码中，为我们预留了一个Expr1的位置，我们就可以延续上述的思路，实现二目左结合运算符“<<”。

## 添加三目运算符“?:”

三目运算符“?:”的实现实际上比二目运算符简单，这是因为三目运算符是右结合的，它固有地不存在左结合的问题。三目运算符的语法规则实现为（记可能含有三目运算符的表达式为Ternary）：

Expr原来解析为之前添加的Expr1，现在解析为可能含有更低优先级运算符的Ternary。由于三目运算符的右结合特性，其“?”前面的内容必然是不含有三目运算符的（否则那个?应该属于本层次的三目运算符）。而之后的两个表达式则含有或不含有三目运算符的。

上面的表达式不含左递归，唯一的问题是有左公因子，可以变换为：

（事实上对于手写文法分析起来说，可以通过额外的条件语句或来处理左递归。变换是可选的。）

## 添加switch-case语句

Switch-case语句的实现相对较为复杂，但是并没有什么本质性的困难。

首先我们定义Case块和Default块的文法：

两条规则需要的Lookahead很清晰。StmtList的解析助教也为我们实现了。

接下来，我们需要构造CaseList，为了避免问题，我们直接采用右递归的形式。

只需要效仿助教的StmtList的构造，将树状结构展平成Case的ArrayList即可。

最后使用它们构造完整的Switch语句：

实际上手写文法分析器，也可以直接在解析到OptionalDefault的位置时添加条件判断语句，减少一次递归，也无需额外的文法树节点。

最后需要注意将对于lookahead时发现SWITCH情况添加到StmtListParse和StmtParse之中，确保相关语句能被正确解析。

## 添加repeat-until和continue语句

repeat-until和continue语句的添加非常简单。其文法如下：

其构造非常简单，使用类似之前的方法即可。

最后需要注意将对于lookahead时发现REPEAT和CONTINUE情况添加到StmtListParse和StmtParse之中，确保相关语句能被正确解析。