****

编译原理实验

报告二

**实验名称： Syntax Parser**

**学生姓名： 柳沿河**

**学生学号： 71117230**

东南大学计算机科学与工程学院、软件学院

School of Computer Science & Engineering College of Software Engineering

Southeast University

二 0 一 九 年 十 二 月

1. 目的
   * 了解语法分析的原理
   * 掌握通过文法构造LR项集簇的方法
   * 掌握通过LR项集簇编写语法分析程序的方法
2. 内容描述

编写一个语法分析程序，要求如下：

1. 输入：字符流、CFG
2. 输出
   1. 自顶向下分析：推导序列
   2. 自底向上分析：规约序列
3. 句子类别自定义
4. 考虑错误处理

任选一下三种种实现方式实现：

1. LL(1)分析
2. LR(1)分析
3. 基于Yacc
4. 方法

本次实验采用第二种实现方式，即通过将CFG构造相应的LR(1)分析表来编写语法分析程序。

1. 假设
   * 分析语言：类C语言语法
   * CFG：

S -> id = E ; S

S -> id = E ;

S -> E ; S

S -> E ;

E -> E + T | E - T | T

T -> T \* F | T / F | F

F -> id | number | ( E )

1. 相关FA描述

根据CFG构造的LR(1)项集见：[LR(1)项集.pdf](LR1项集.pdf)

根据LR1项集构造的分析预测表如下：



1. 重要的数据结构描述

* LR1PPT类：预测分析表对象
  + 成员
    - private Map<String, Action[]> actions：存放动作表的成员对象
    - private Map<String, int[]> gotos：存放转换表的成员对象
  + 方法
    - private void buildActionTable()：构造动作表
    - private void buildGotoTable()：构造转换表
    - public Action[] ACTION(String terminalSymbol)：获取读入的终结符对应的所有动作
    - public int[] GOTO(String nonterminalSymbol)：获取读入的非终结符对应的所有状态转换
* Production类：产生式对象
  + 成员
    - private String left：存放产生式左部
    - private String[] right：存放产生式右部
  + 方法
    - get方法：获取产生式的左部和右部
    - 重载toString方法：用于向文件中输出规约
* Action类：动作对象
  + 成员
    - ActionType type：动作类型，有ActionType.Shift, ActionType.Reduce, ActionType.Accept三种类型
    - int stateRule：存放要移进到的状态或规约用的产生式编号
* StackItem类：分析栈中元素对象
  + 成员
    - StackItemType type：类型，有StackItemType.Token, StackItemType.State两种
    - String value：若为token，则存放对应的字面值
    - int state：若为状态，则存放对应的状态编号

1. 核心算法描述

算法：类C语言语法分析算法parse

输入：类C词法分析得到的token序列，CFG，PPT

输出：归约序列

执行过程：

* + 1. 构造用于文件写入的BufferedWriter对象、token序列对象的迭代器对象，并初始化分析栈——添加状态为0的栈元素，通过迭代器获取当前读入的token对象，记为curToken，执行下一步
    2. 根据token的字面值和栈顶的状态，获取curToken对应的动作。若动作为null，则token不合法，报错；若动作为Accept，则提示语法分析成功，结束分析；否则执行下一步
    3. 若动作为Shift，则用curToken构造新的StackItem对象并将其压入栈中，再用该Shift动作对应的要移到的状态构造StackItem对象并将其压入栈中。最后用迭代器读取下一个token赋给curToken，返回2；否则执行下一步
    4. 若动作为Reduce，则用该Reduce动作对应的产生式编号在产生式集合productionSet中获取对应的产生式，将其写入归约文件中。之后，从最右端开始，对于的右部的每一个符号，先将栈顶的状态项弹出，然后将与栈顶元素的字面值进行对比，若两者相等则弹出，进行下一次循环；否则报错。最后用左部的符号和当前栈顶的状态在Goto表中获得的下一个状态，用构造StackItem对象并压入栈中，再用构造StackItem对象并压入栈中。返回2

1. 用例

测试用例如下：

x = 10;

y = 100;

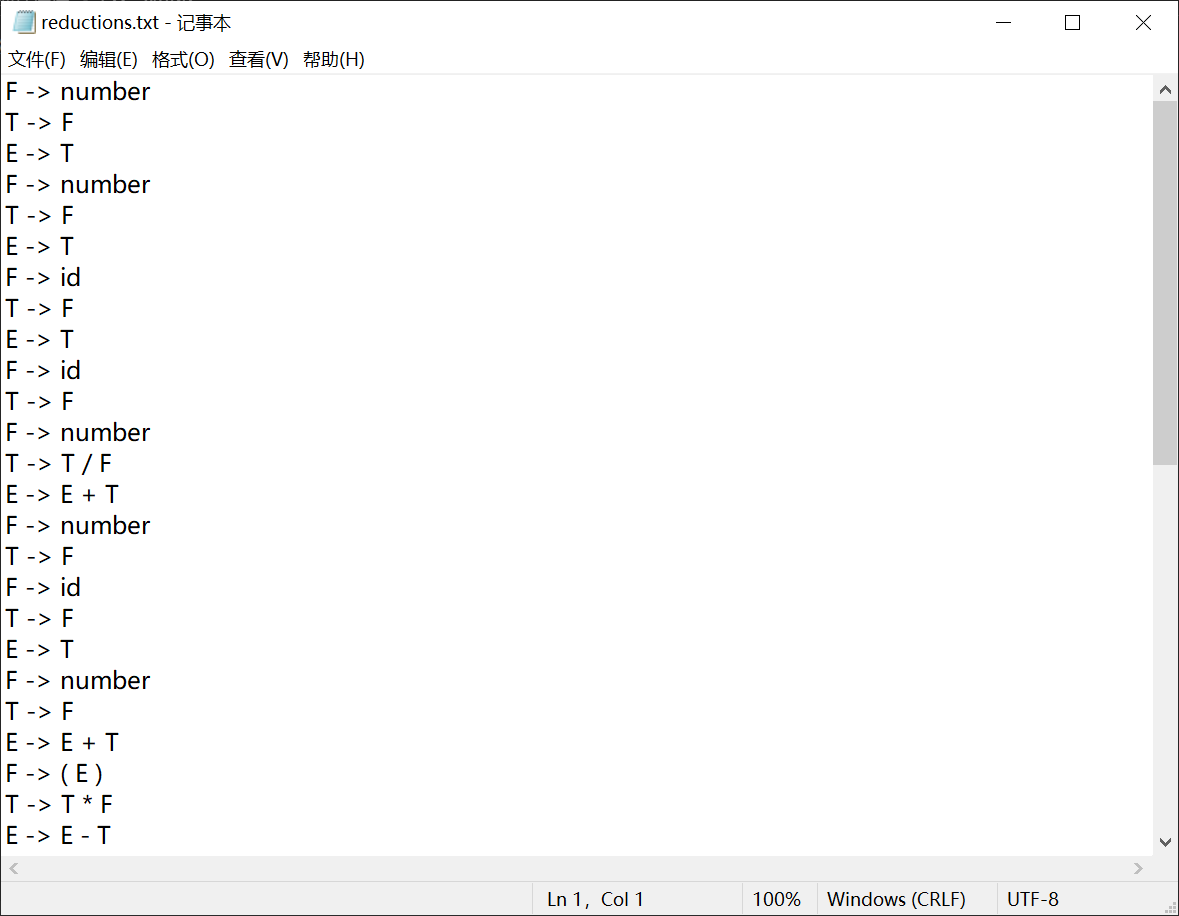
t = x + y / 3 - 2 \* ( x + 6 ) ;

t + 3 - x\*5 / (y-1);

运行程序方法：

* java SyntaxParser test.txt tokens.txt reductions.txt
* IDE中运行测试类LR1ParserTest

运行结果如下：



1. 发生的问题和相关的解决方案

* 无法顺利结束语法分析：在从词法分析器获取的token序列最后添加字面值为$的token对象
* 分析算法可读性较差，结构化不高：构造Action类和StackItem类，并用枚举变量表示动作类型和栈元素类型

1. 感受和评论

刚开始进行本次实验时，尝试采用了比较简单的LL(1)分析法，但为了熟悉更为强大的LR(1)分析法，最终还是采用了LR(1)分析法。该方法分析能力强大，符合机器的运行方式，但同时代价非常高，需要构造非常多的LR(1)项目集，人工实现方式几乎无法分析稍微复杂一点的CFG。经过权衡，最终去除了一些含有关键字的句子，保留了常用的四则运算和变量的定义语句，即使如此还是有40个LR(1)项集。构造过程要求十分的仔细，稍有不慎就会造成错误。尽管最终完成了这次试验，但花费的时间和精力较大，在以后的学习实践中会尝试Yacc来构造分析表和分析器。