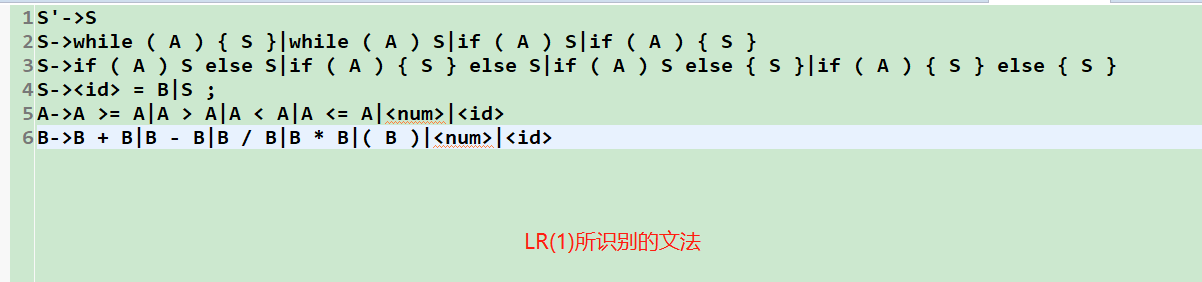
**课程设计 基于LR(1)的语法分析器**

1. **编写LR(1)语法分析器的目的**
2. 理解语法分析器的基本功能。语法分析的任务是：以词法分析分析器生成的单词序列作为输入，根据语言的文法规则，检查所给的单词符号序列是否是该语言文法的一个句子。是则输出改单词符号序列的语法树，否则输出错误。
3. 构造 LR (1) 分析程序，利用它进行语法分析，判断给出的符号串是否为该文法识别的句子，了解 LR（K）分析方法是严格的从左向右扫描，和自底向上的语法分析方法。
4. 通过一个简易的语法分析器的实现，进一步加深对《编译原理》课程中语法分析理论的理解，增强动手程序设计能力。本次试验采用的是 LR(1)分析法，加深对它的理解。

**二、语法分析器的需求**

1. 对与单行注释与多行注释可以进行屏蔽
2. 对于待识别的语言序列,如果其符合文法，则输出其对应的语法树，否则则输出错误
3. 对于运行过程中创建的预测分析表、LR1项目集簇，分析过程，token序列，要输出为文件。
4. 要实现文件读入待分析单词序列
5. 运行环境：硬件windows 软件 eclipse JDK版本1.8
6. 实现语言：java
7. 待分析语言：C语言子集文法的语法
8. **语法分析器的设计**
9. **语法分析器的文法规则**

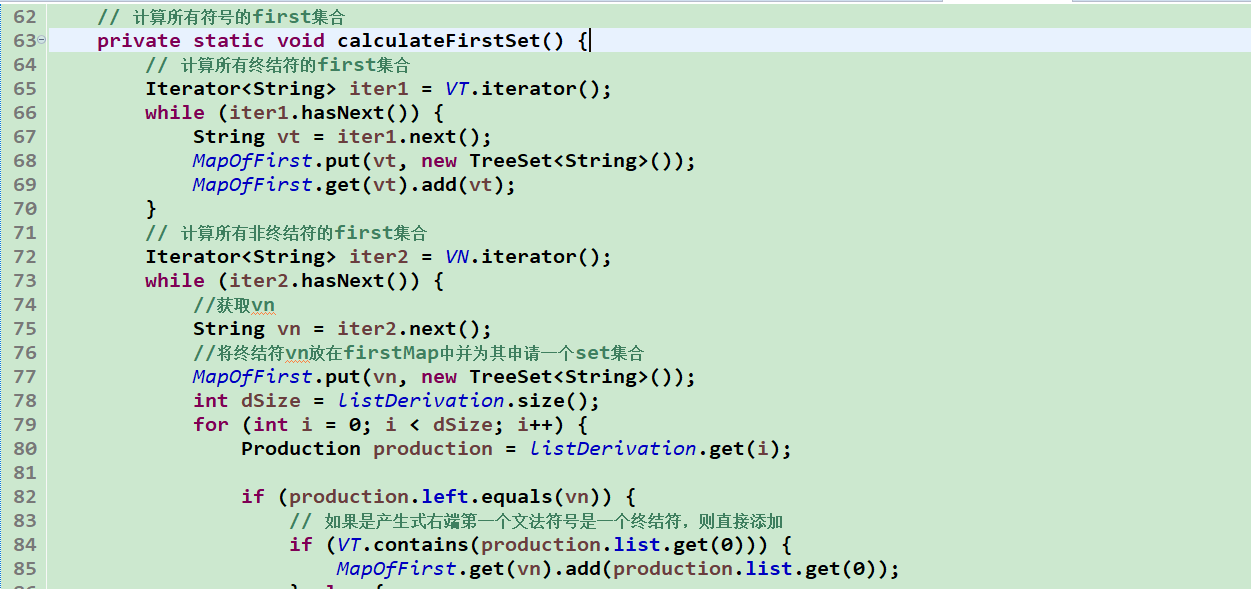
**注意：文法中的<num>代表数字，<id> 代表标识符**

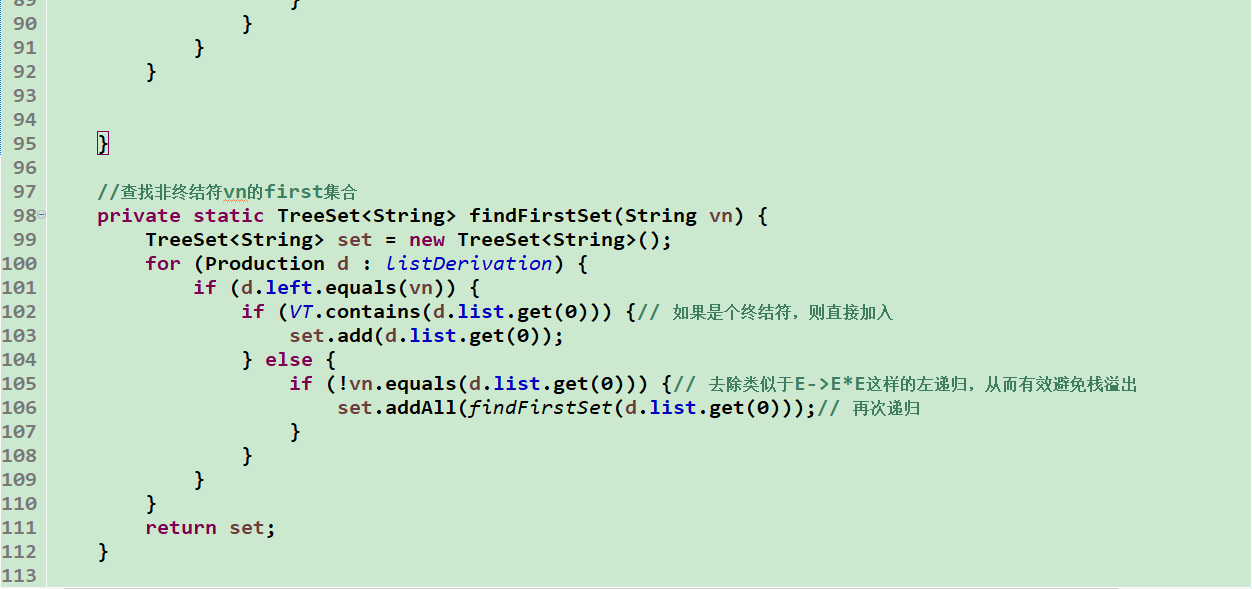


1. **分析过程的输出格式**

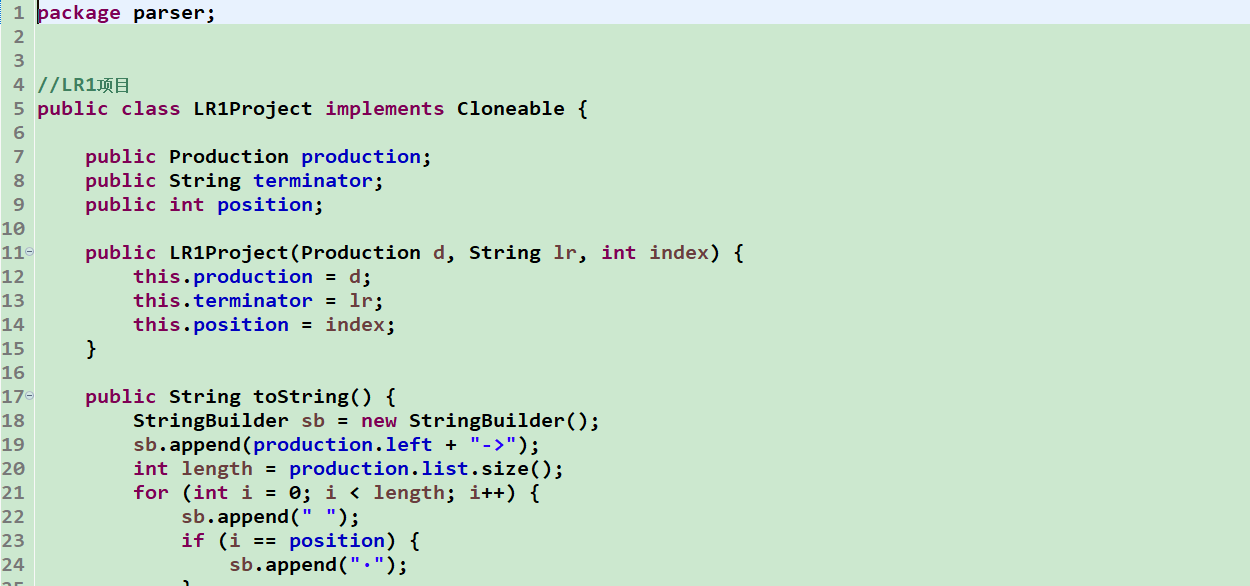


1. **递归构建first集合**

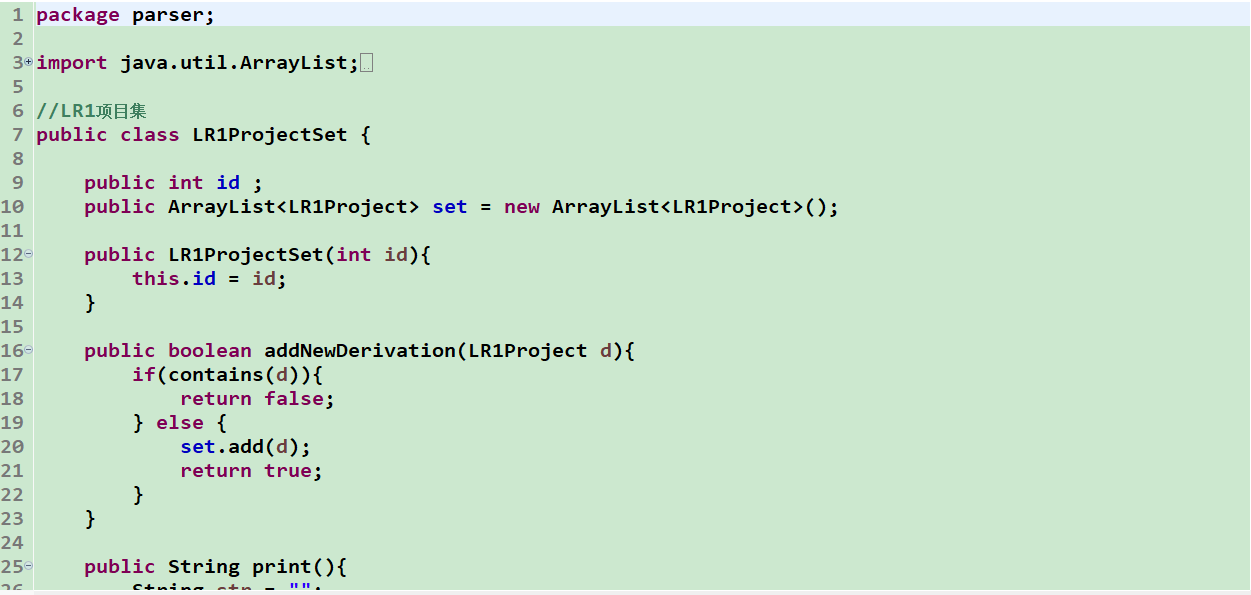




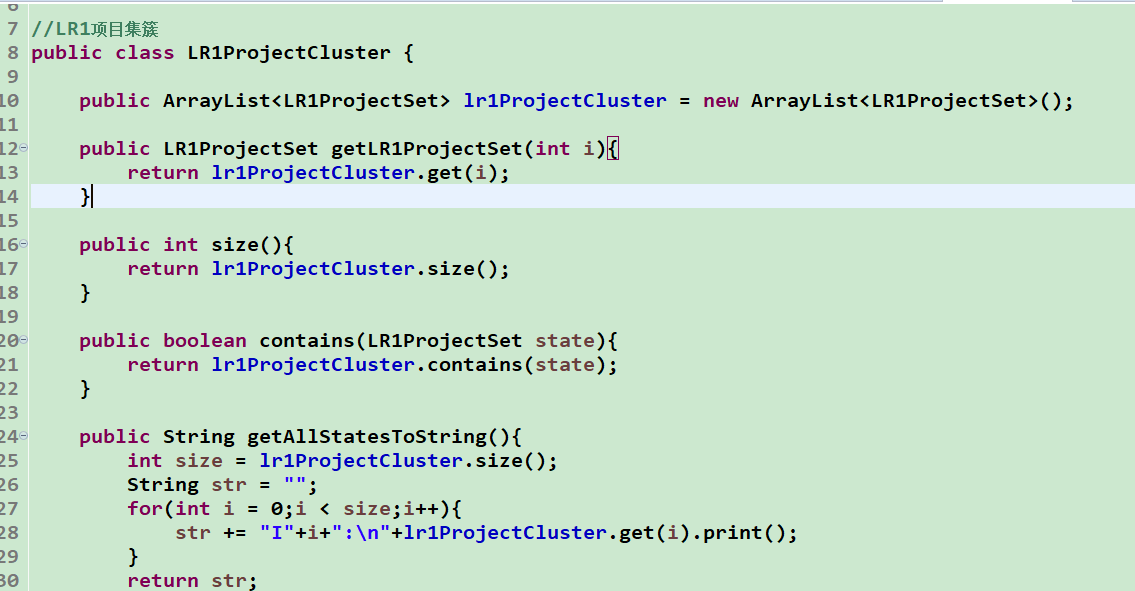
1. **构建LR1项目集簇**



LR(1)项目

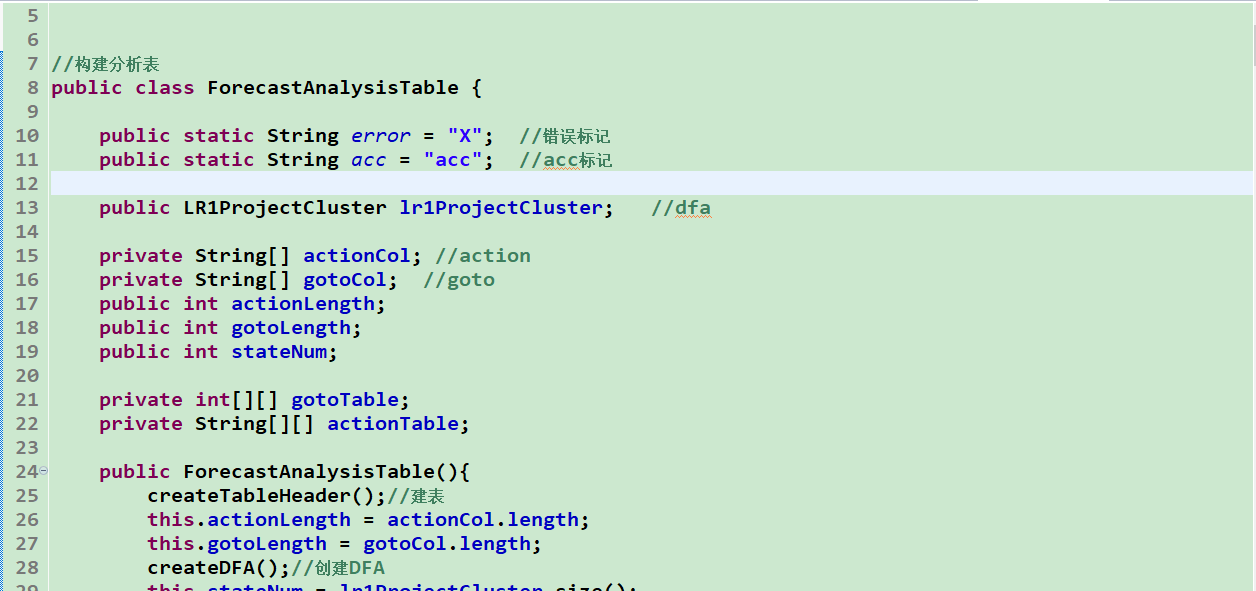


LR(1)项目集

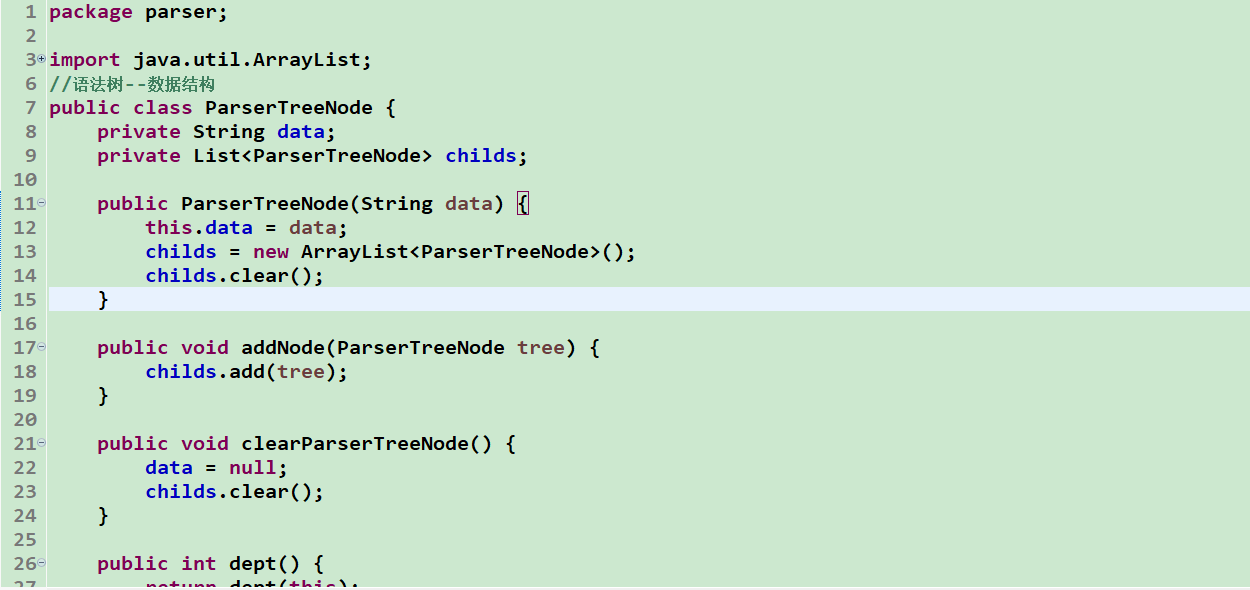


LR(1)项目集簇

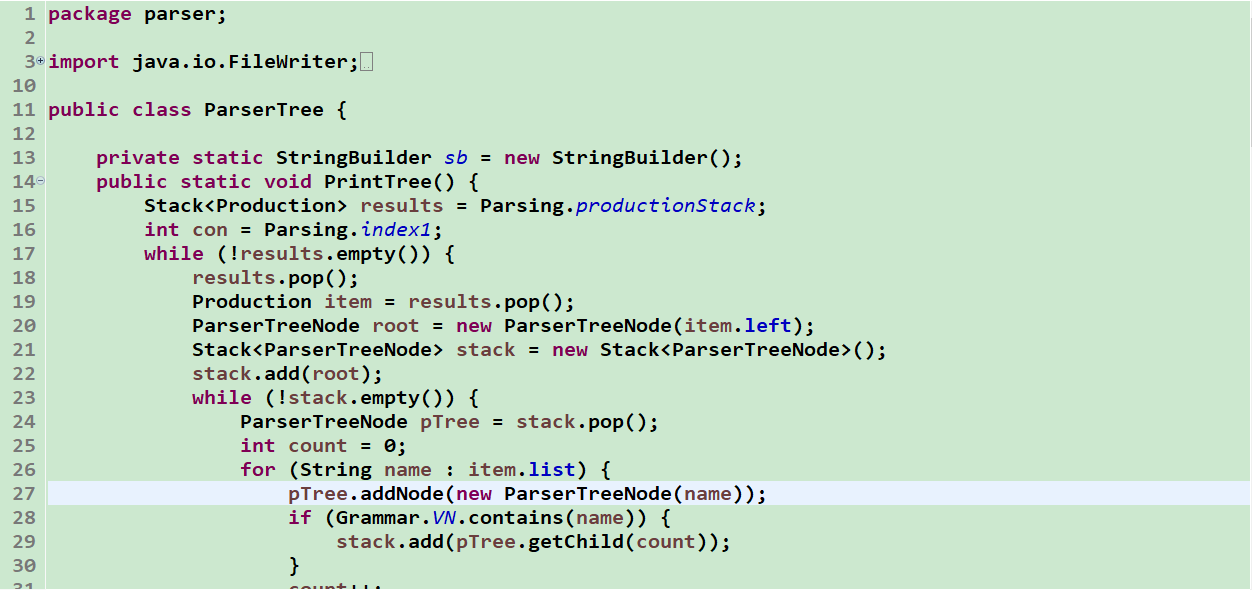
1. **预测分析表**



1. **语法树**



语法树节点



语法树

1. **语法分析程序的设计想法**

①根据读入的文法 G 初始化辅助的数据结构，包括终结符集，非终结符集，非终结符的 first集（用于产生 LR(1)分析表中的预测符）等。并构造出 LR(1)状态机（项目集簇）。

③根据构造出的状态机（项目集），生成分析表的 GOTO 和 ACTION 部分，构造出 LR(1)分析表。

④根据 LR(1)分析表，对输入的字符串进行分析得到并输出分析结果，如果识别的序列为合法则会输出其对应的语法树。

⑤如果输入的文法有二义性规约冲突，或者输入的字符串不合法会有错误提示。

**四、语法分析器的全部代码**

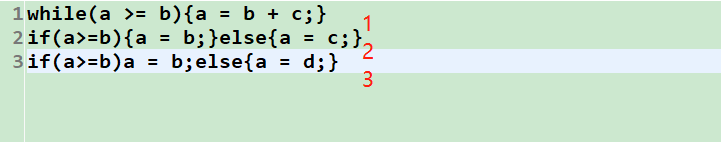
由于代码太多我就给出了树形目录



**五、语法分析器的测试与结论**

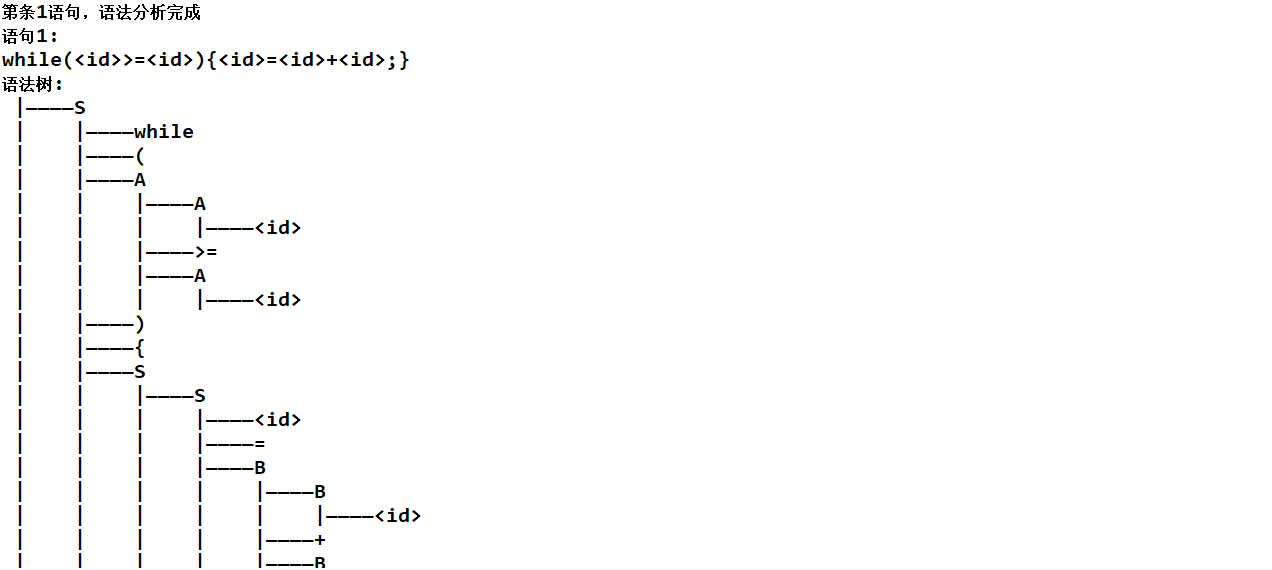
**（一）测试**

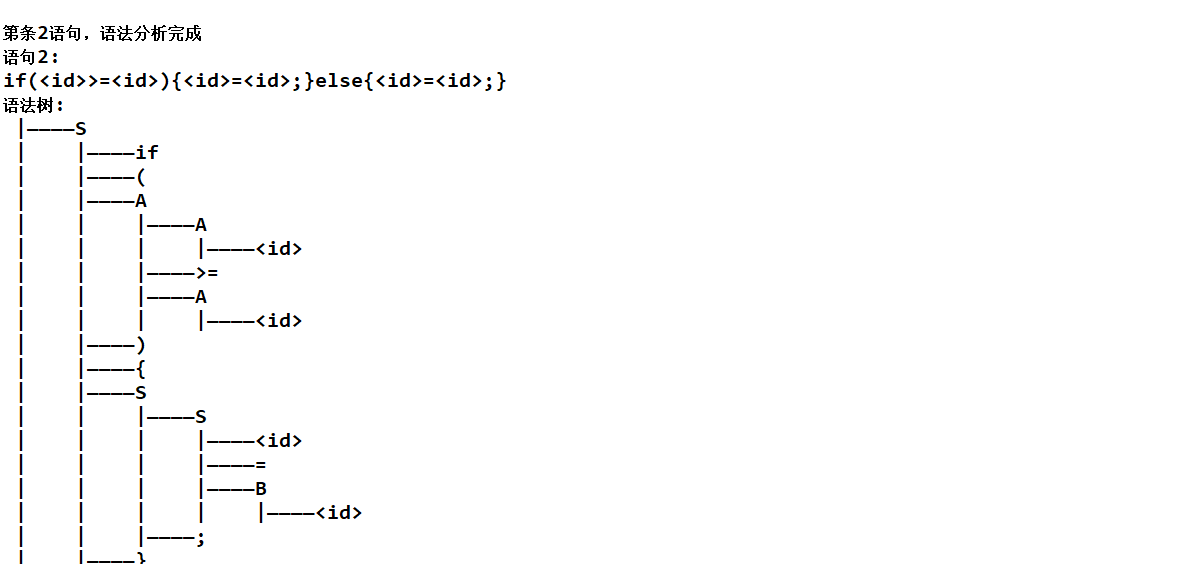
**1.测试文件：**

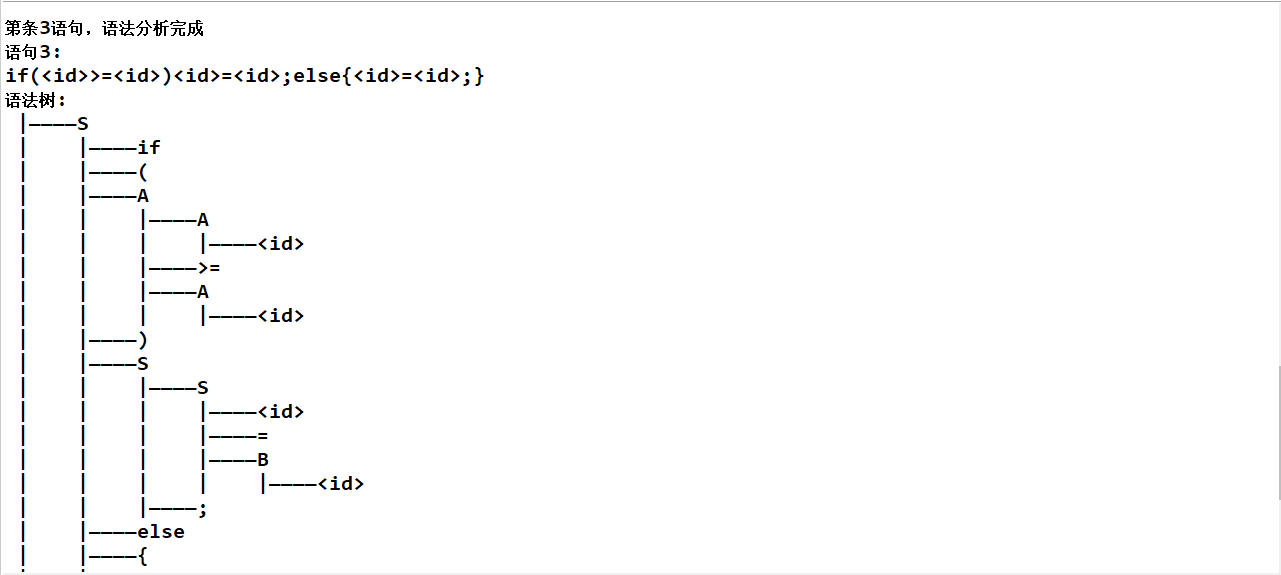


**2.测试结果:**

①控制台结果截图：

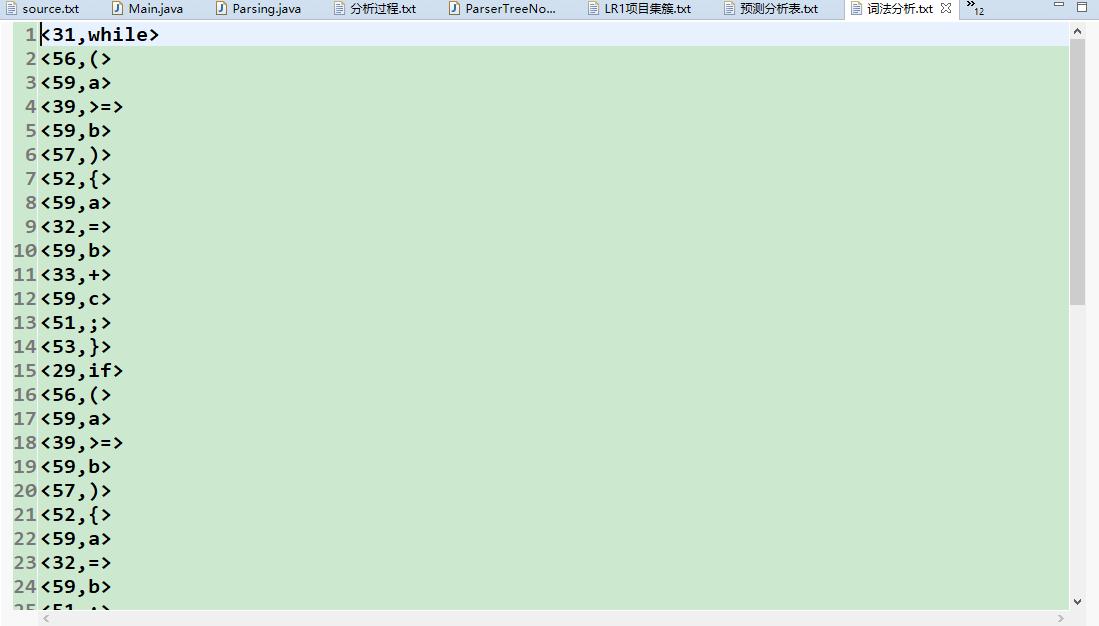






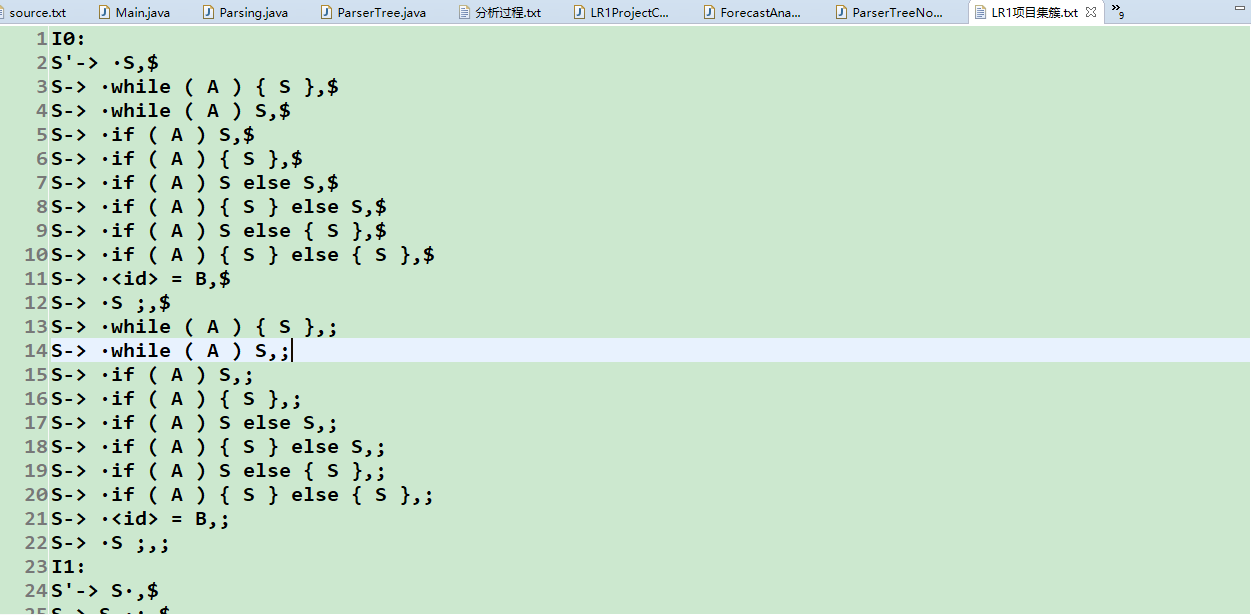
②结果文档截图：

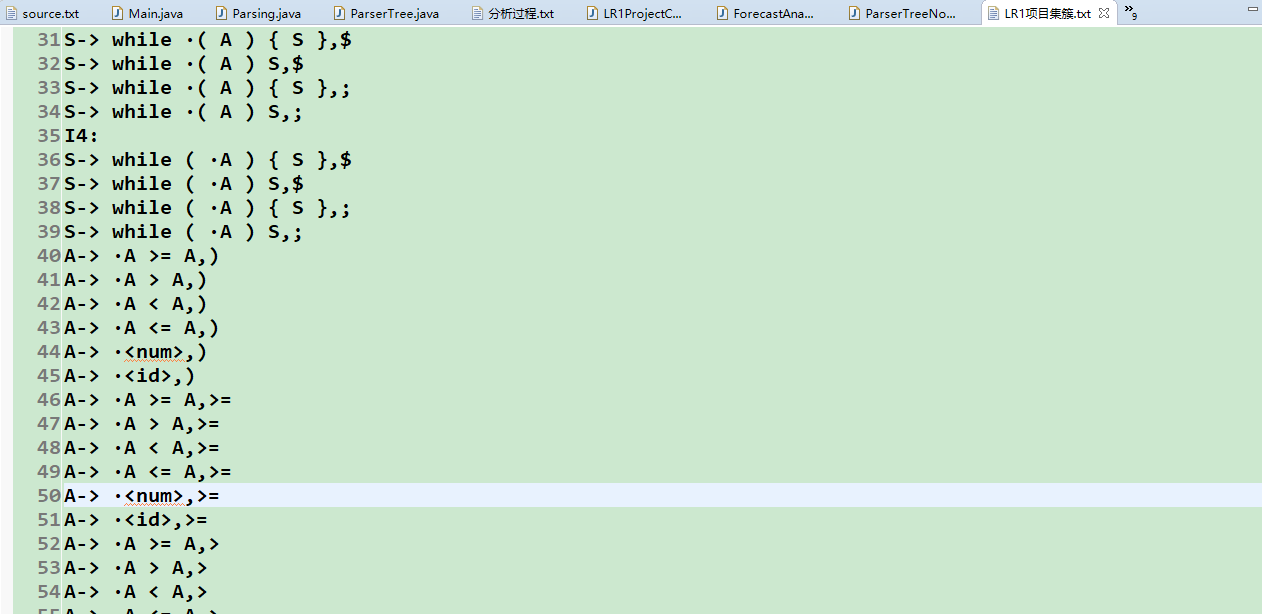
**词法分析：**

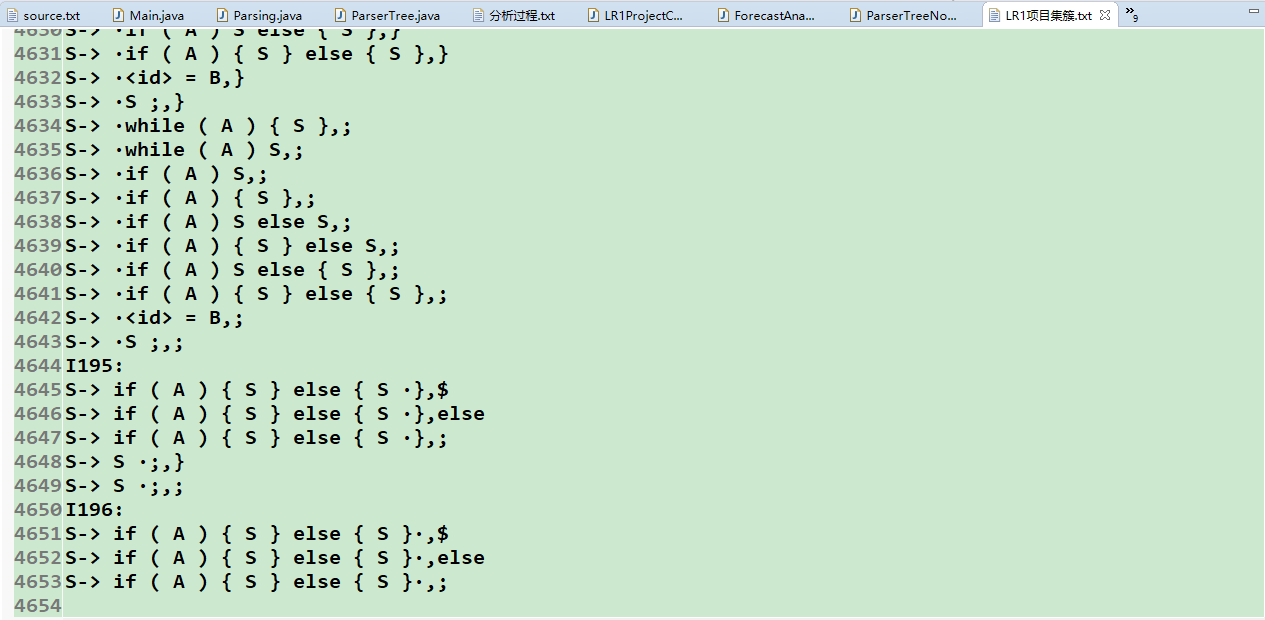




**LR1项目集簇：**

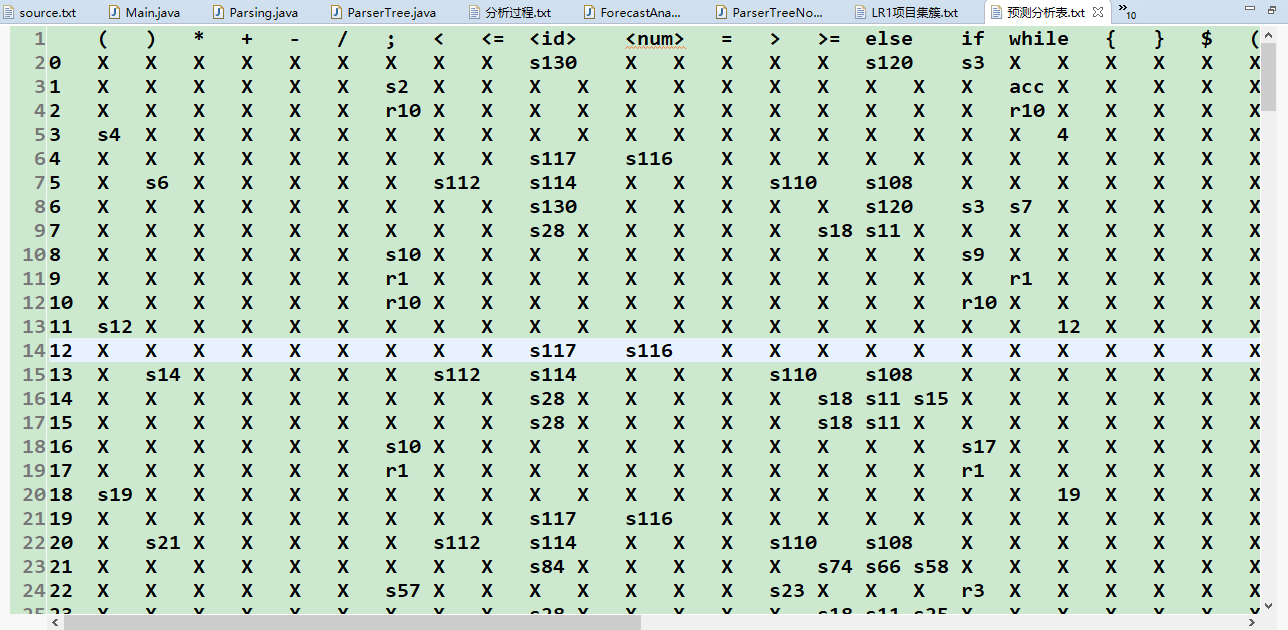


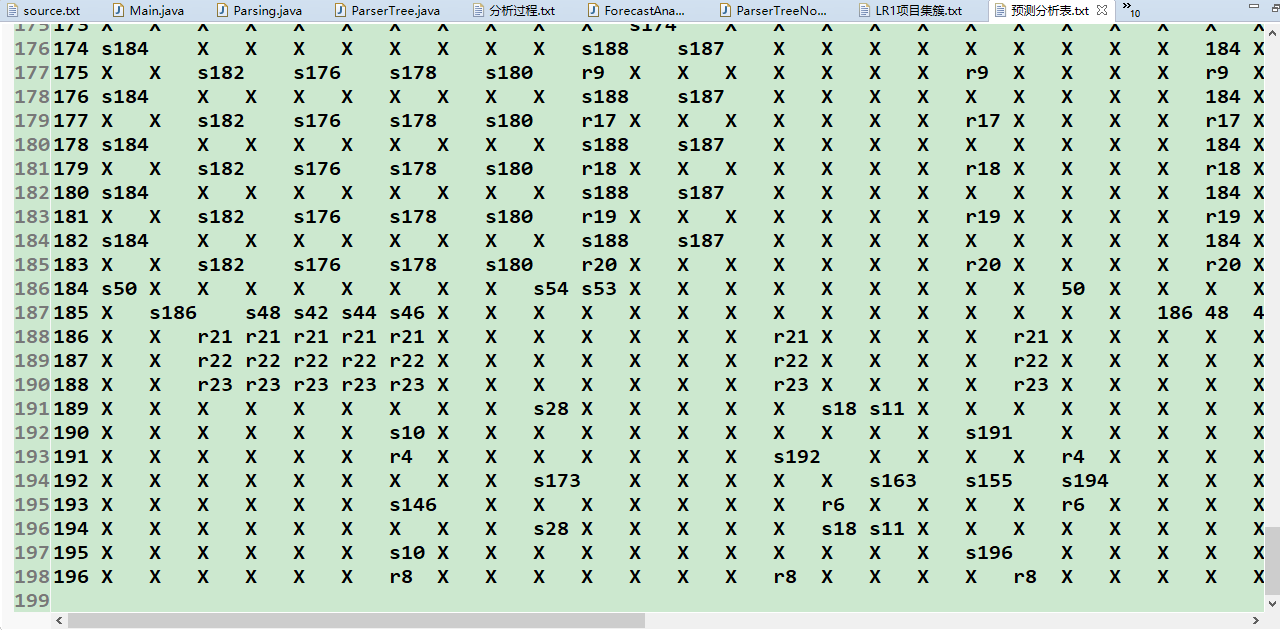




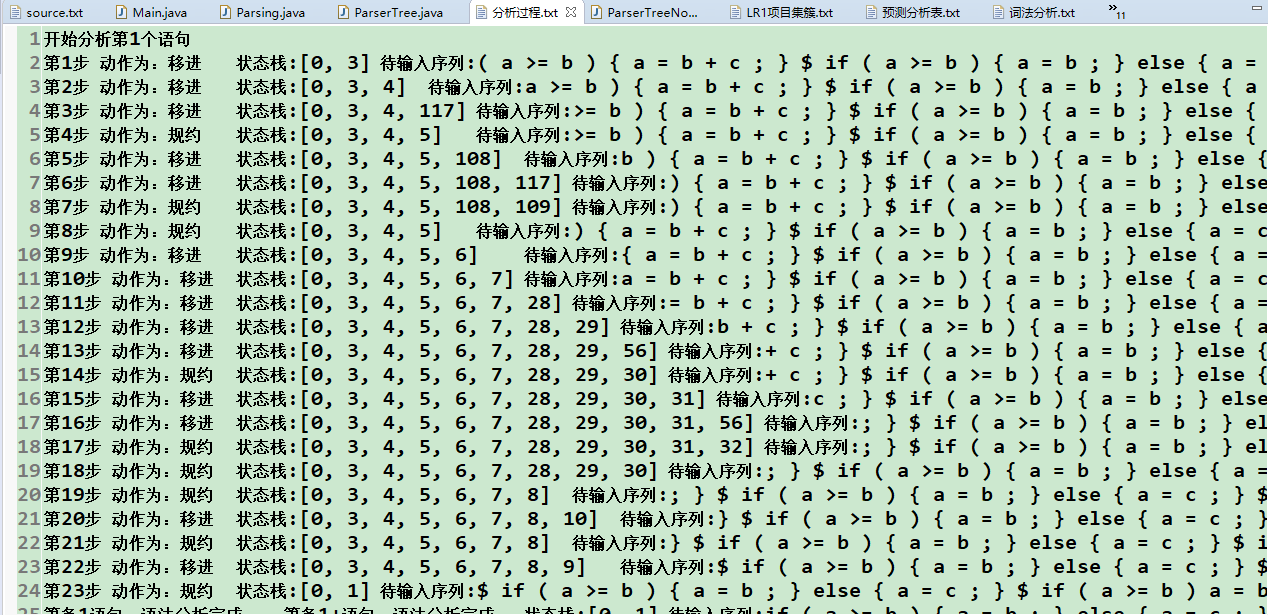
这个项目集太多了，就先放这么些，具体可以看文件

**预测分析表**

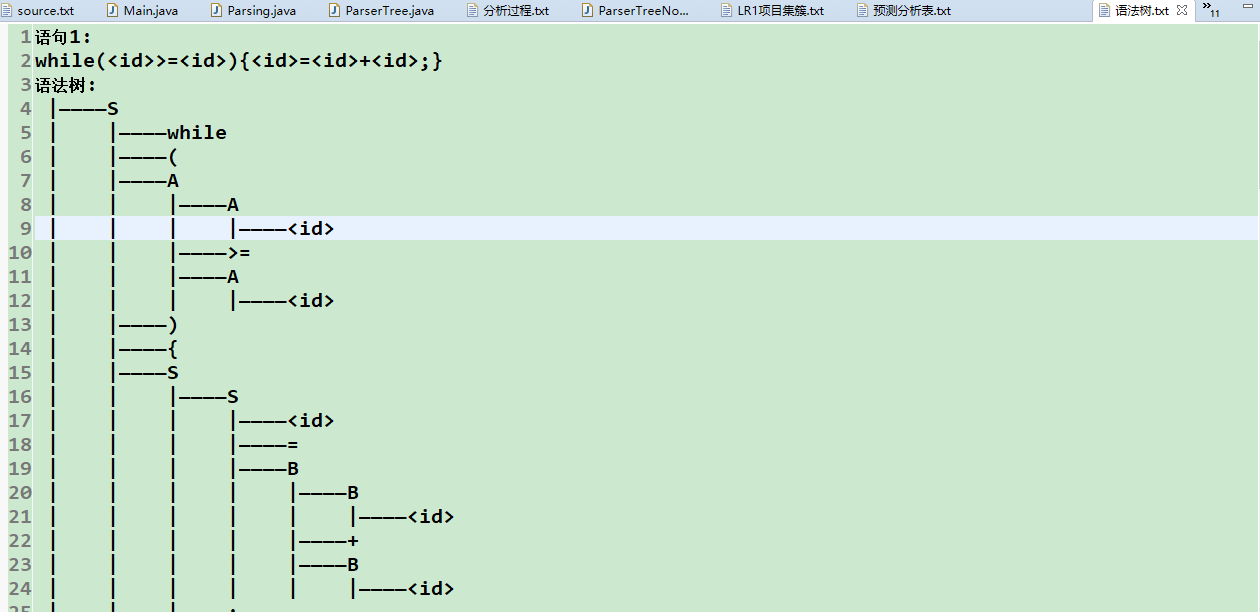


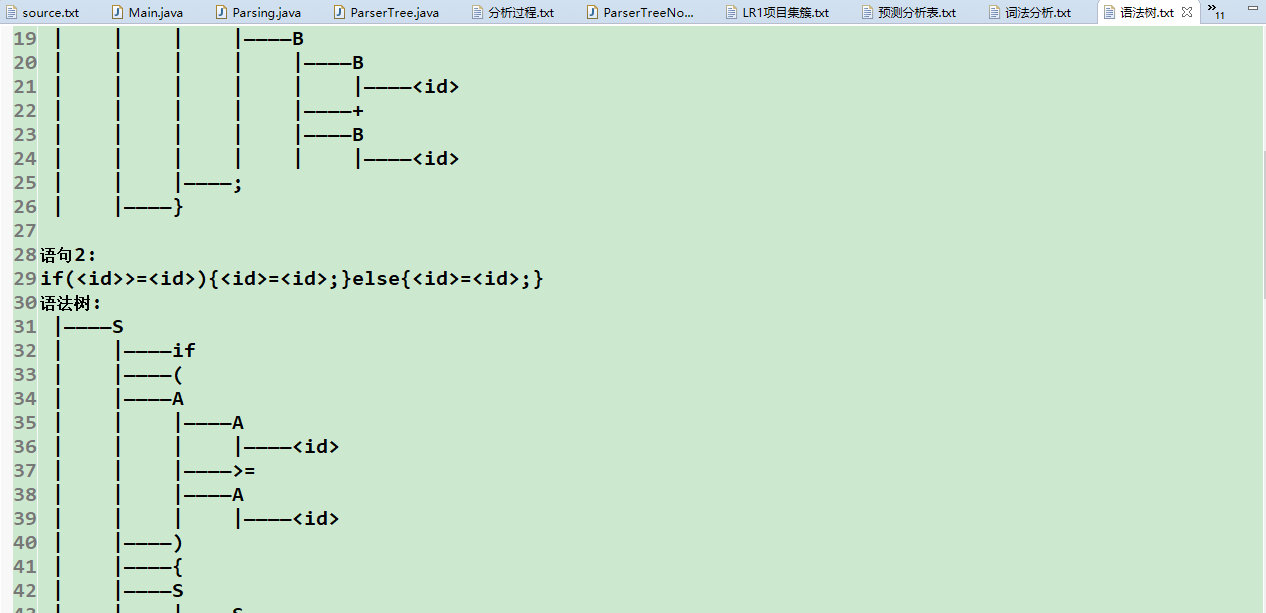


**分析过程：**



**语法树文件：**





**（二）结论**

通过进行测试以及与结果对照，可以说明这个基于LR(1)的语法分析程序运行正确，可以对符合语法的单词序列进行语法分析。

**六、总结**

1.刚刚开始对自动机的自动构建无从下手，后面通过自己建立数据结构一步步解决问题，成功构建了自动机。

2..通过一个实际的语法分析器的具体实现，考虑了很多进行语法分析的过程中的细节，巩固了对课堂上所讲解的相关编译理论的掌握，加深了对语法分析过程的理解，增强了动手进行程序设计的能力，为以后自行实现一个完整的编译器和研究高级程序设计语言的内部原理打下了扎实的基础。

3. LR (1) 构造的 DFA 考虑当前输入的字符，所以构造的 DFA 中存在的冲突比较少，也就是说它接受的语言就比较多。构造分析表时，不用额外的约束，因为它在构造 dfa 时就已经考虑了当前的输入字符。

1. **体会**

1.语法分析中的first集合以及LR1项目集簇的递归求解,是一个考验代码逻辑的事情。

2.编程过程中不要太过追求代码的精简，一段结构清晰、代码整洁（指的是代码清晰易懂），显然会更赏心悦目，同时也方便别人去阅读，所以建议大家在写代码的时候掌握一个度，合理的书写代码，不多不少，刚刚好！

3.逻辑一定要清晰，高级程序设计语言在编译运行的过程中，是按照代码的顺序执行，没有判断力，所以我们的逻辑性就会显得至关重要，因为一定程度上编出的程序就代表了你的智商。

4.时间是检验真理的唯一标准,对于程序员，我认为读万卷书的人是比不过行万里路的人的，代码的编写能力是在日常练习堆积起来的。