编译原理 Lab2

1. 实验目的

   自行定义文法，运用语法分析方法对输入语句进行语法分析并输出结果，加深对语 法分析过程的理解。

2. 内容描述

   此程序用 Java 编写。程序读取一个文本文件，运用上次实验写的词法分析程序得到 TOKEN 序列，再对其进行语法分析。

   这里使用的是 LL(1)方法进行自顶向下分析，最 后输出产生式序列。

   定义的具体文法有：

   一般赋值语句，如 a=1*(2+3);（为减小复杂度，只有+和*，且*优先级高）

   If-else 语句，if(con){stmt}else{stmt}

   While 语句，while(con){stmt}

   条件语句，a==1||(b==2||c==3)（为减小复杂度，只有||）

3. 实验方法

   1）首先自定义要分析的文法

   2) 对文法进行预处理（消除左递归、二义性）

   3) 手动构造预测分析表

   4) 基于分析表编写代码

   5) 代码中具体的实现：根据输入队列和状态栈的第一个元素进行分析进行终极符匹配或非终极符产生子项，循环至处理完输入队列

   首先将获取的 TOKEN 序列放入队列，状态栈压入初始非终结符。读取栈顶元素和队列的第一个元素进行分析，若栈顶元素是非终结符，则调用 generate()结合队列元素查表运用产生式，将新产生的元素压栈；若栈顶元素是终结符，则与队列元素匹配，成功则都弹出。循环至输出元素均处理完。过程中有查表失败或终结符不匹配则输出错误。

4. 假设

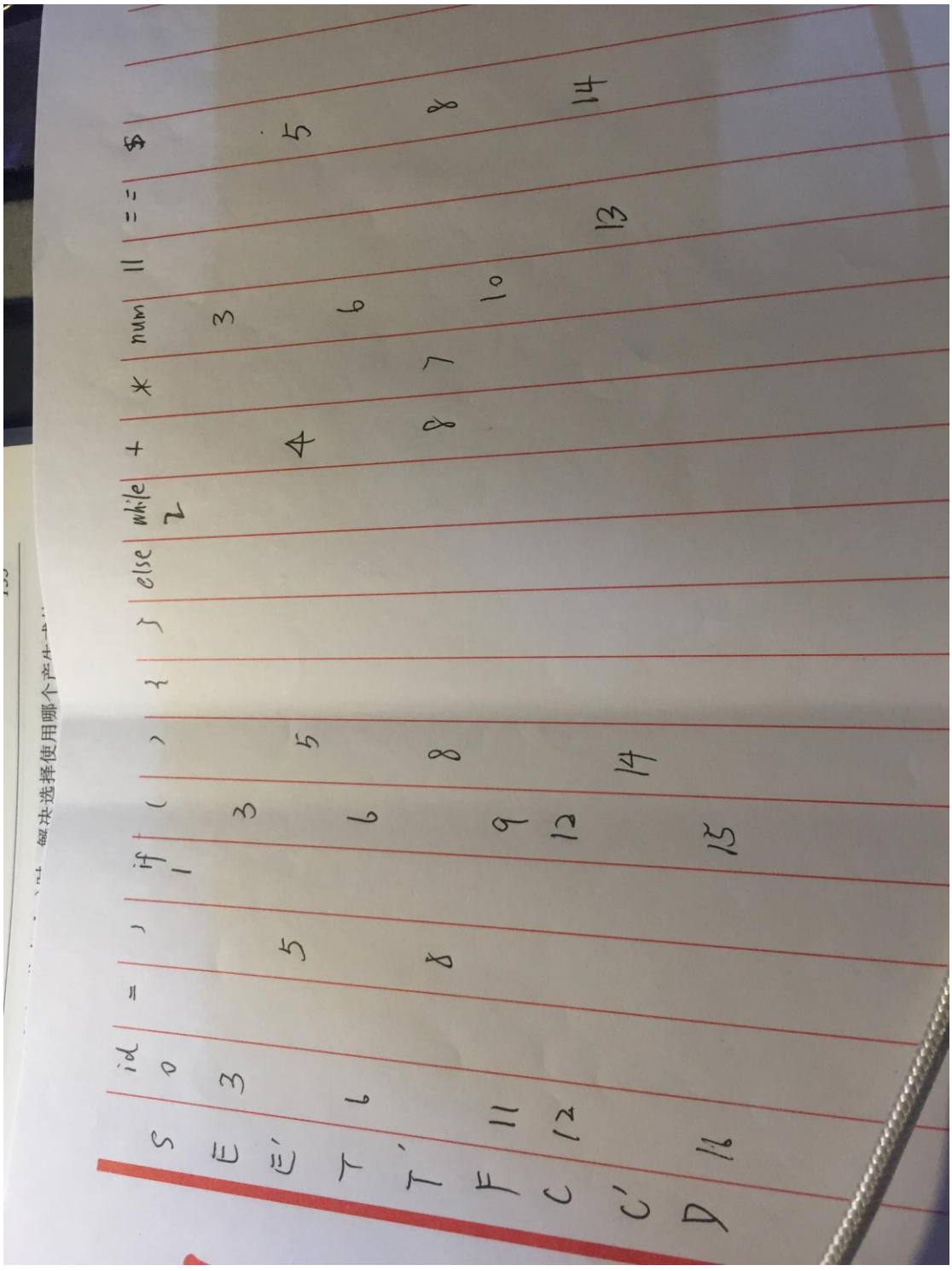
   程序中所有变量都显示为 id，所有常数都显示为 num。

自定义的文法如下：

0. S -> id=E;
1. S -> if(C){S}else{S}
2. S -> while(C){S}
3. E -> TE'
4. E' -> +TE'
5. E' -> ε
6. T -> FT'
7. T' -> *FT'
8. T' -> ε

9. F -> (E)
10. F -> num
11. F -> id
12. C -> DC'
13. C' -> ||DC'
14. C' -> ε
15. D -> (C)
16. D -> id==num

预测分析表



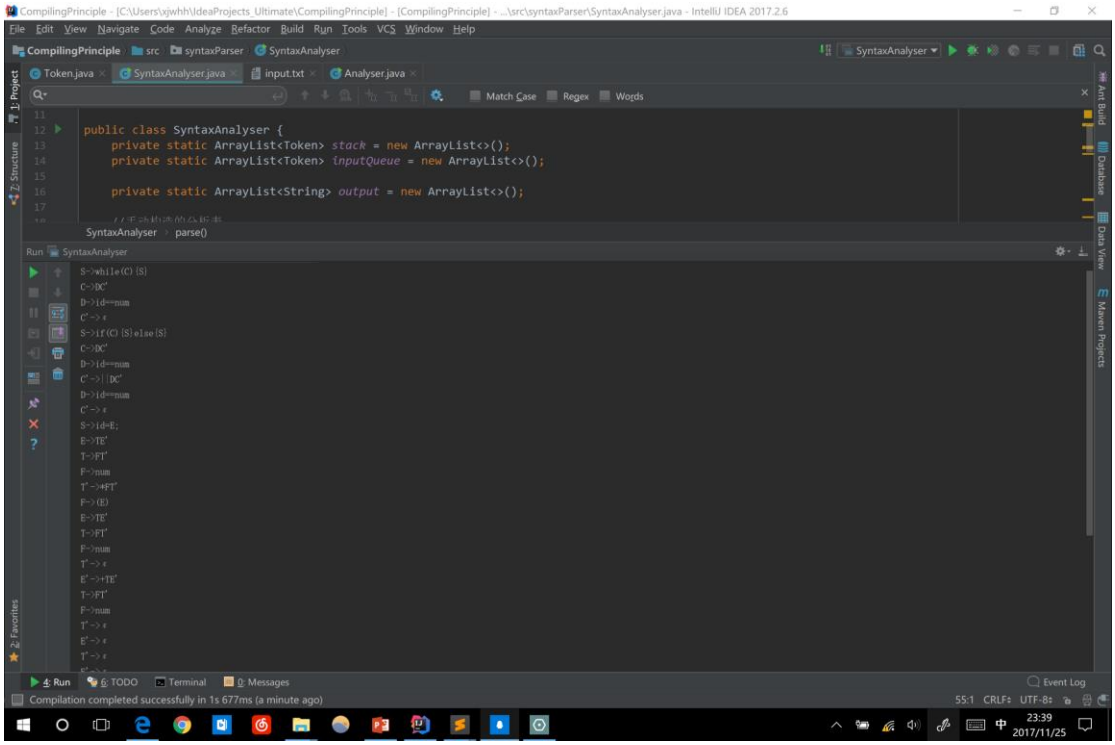
5. 重要数据结构描述
   inputToken:ArrayList<String>：输入缓冲区
   stack:ArrayList<String>：栈
   PPT:int[ ][ ]：分析表
   generations::ArrayList<String>：自定义产生式

6. 代码算法描述

parse()：判断栈顶元素并进行相应操作，非终结符调用 reduce，终结符则与输入缓冲区第一个元素比较，相等则都弹出，否则报错

reduce()：栈顶元素为非终结符，查表，若有对应条目则弹出非终结符并将产生式压栈，否则报错

7. 示例运行



8. 产生的问题和解决方法
   1) 调用的是上次的实验一产生的代码产生 token 序列，为了实现 id 和 num 做了一些特殊化处理
   2) 发现了上次代码中的一个 bug，单纯返回+之前没有将指针前移
9. 感受

   对于语法分析的状态和变化有了更好的理解