



编译原理实验报告

|  |  |
| --- | --- |
| 专 业 班 级 | 计算机创新实验18-1班 |
| 学生姓名及学号 | 2018213106 刘嘉伟 |
| 任 课 教 师 | 李宏芒老师 |
| 实验指导教师 | 李宏芒老师 |
| 实验地点 | 家 |
| 2019 ~2020 学年第二学期 | |

## 实验一 词法分析设计

**一、实验目的**

通过本实验的编程实践，使学生了解词法分析的任务，掌握词法分析程序设计的原理和构造方法，使学生对编译的基本概念、原理和方法有完整的和清楚的理解，并能正确地、熟练地运用。

**二、功能描述**

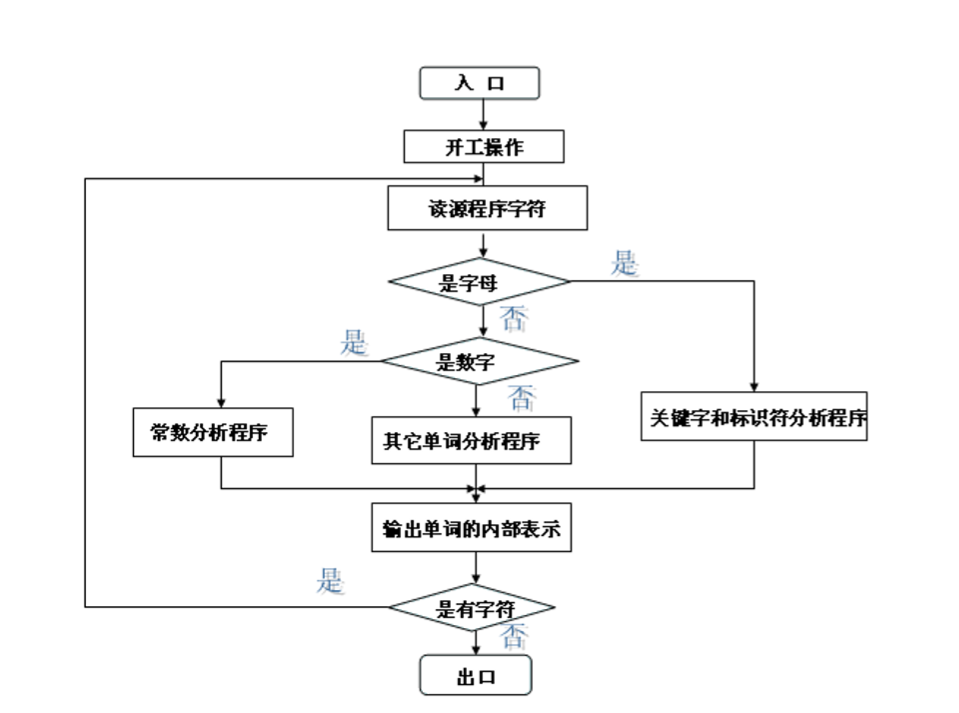
1. 不同的文本包含不同种类的字符，在相应的文本文件中添加关键字或单运算符，即可拓展该分析器的功能
2. 能自动搜索出错误的位置并进行标记
3. 消除所有形式的注释（//和/\*\*/）
4. 清除所有的空格和换行
5. 能一键清空所有输入与输出，以便重新分析
6. 能识别浮点数，出现两个点能报错
7. 内部保存每个变量的地址，能供编译器后续部分使用
8. 以表格的形式显示输出结果

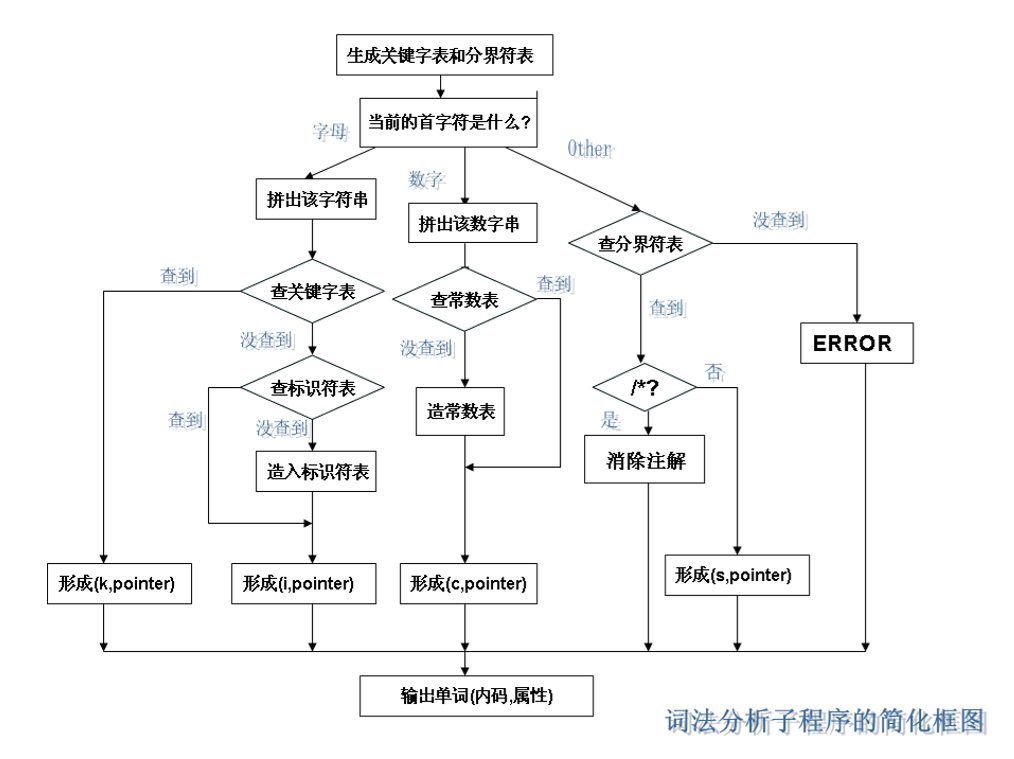
注：原本只显示Error无法查看Error是什么，现在能显示Error的单词

**三、程序结构描述**

|  |  |  |  |
| --- | --- | --- | --- |
| 函数名称及调用格式返回类型 | 参数含义 | 返回值描述 | 函数功能 |
| init init() | 无 | 无 | 初始化所有数据结构，并读取文本文件中的关键字等 |
| void analysis(); | 无 | 无 | 主分析程序，将输入的文本进行词法分析 |
| void insert(String name, inttype, introw, intcol  ) | 分析出的单词name,类型type,所在的行row和列col | 无 | 将每个分析出的结果保存在一个result表中，读取时四个一组读取 |
| publicString[][] createTable() | 无 | 返回构造输出表表所用的的二维数组 | 为后续的可视化做准备，JTable的构造需要一个二维字符数组 |

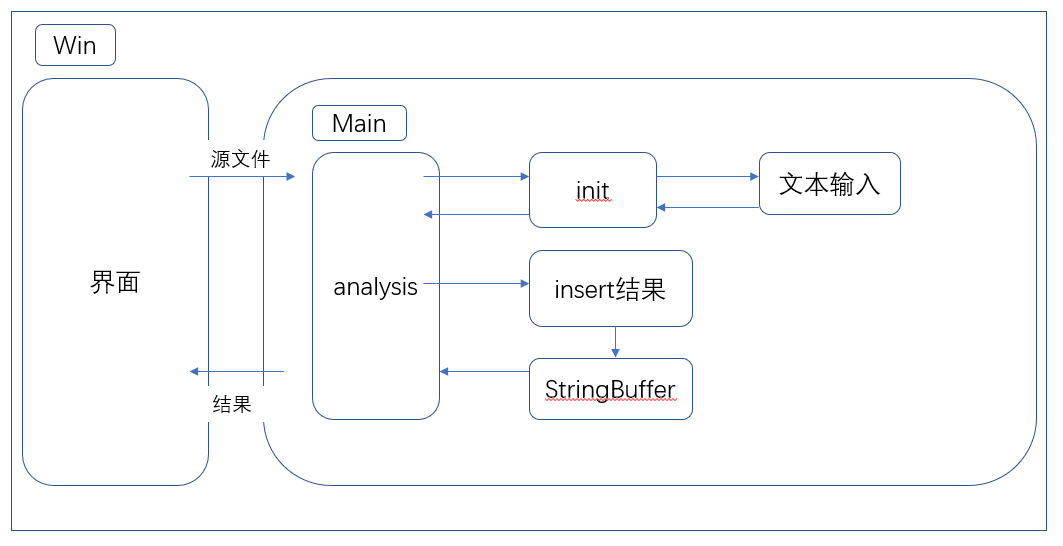
**四、算法流程示意图**





**五、详细的算法描述**

**算法模块调用**

****

1) 标识符的识别

如果没到行尾或空格，并且开头的字符是字母，并且之后是字母或数字，则一直向后读取。

|  |
| --- |
| **while** (cur < sentence.**length** && sentence[cur] != **' '**) { *//接受代码* **if** (Character.*isAlphabetic*(sentence[cur]) || Character.*isDigit*(sentence[cur])) {  tmp += sentence[cur++];  } **else break**; } |

2) 数字的识别

如果开头是数字，则之后只能读取到数字或小数点，且小数点只能读取一次。

|  |
| --- |
| **while** (cur < sentence.**length** && sentence[cur] != **' '**) { *//接受代码* **if** (Character.*isDigit*(sentence[cur]) || sentence[cur] == **'.'**){  **if**(sentence[cur]==**'.'**){  **if**(!hasDot)hasDot = **true**;  **else** {  flag = **false**;  **break**;  }  }  tmp += sentence[cur++];  }  **else if**(Character.*isAlphabetic*(sentence[cur])) {  flag = **false**;  **break**;  }**else break**; } |

3) 分隔符识别

只需要查找分隔符的Map即可。

|  |
| --- |
| **if** (**sepSet**.containsKey(c + **""**)) |

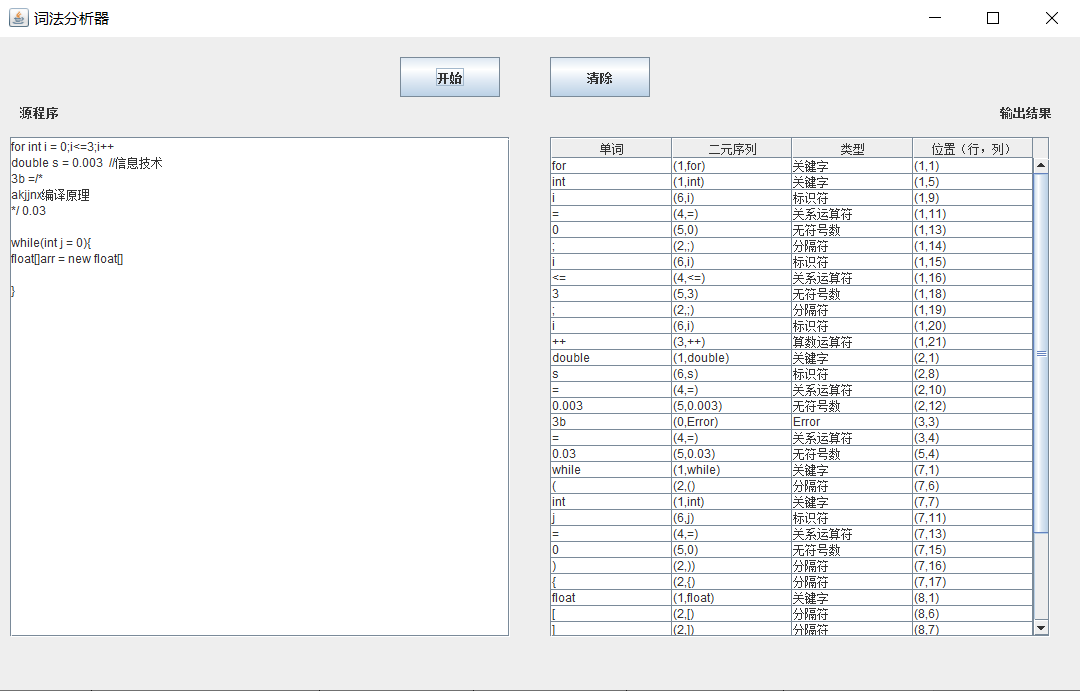
4) 运算符识别

单目运算只需要查找Map即可，双目需要手动添加条件，如果当前字符是/则需特殊处理，如果下一个是/则跳过本行，如果是\*则一直向下寻找知道出现\*/，将所有的都跳过。

**六、程序用法及实验结果**

用法：在左侧的输入区输入待分析的源程序，点击开始，右侧即可出现分析结果，如需重新分析别的文本，点击清除即可把文本框清空，并清空右侧的输出结果。

实验结果：

****

**注：原本没有使用表格输出，输出不对其，现在添加了表格输出**

**七、实验总结**

1）第一个实验比较简单，只需要按照书上的if-else语句分别进行判断就能实现大部分功能

2）让我重新学习了Java-GUI的知识，以及模块化设计方式

3）这个实验让我们大致了解我们平时所使用的编译器检查错误的基本流程

4）还有一些不足之处，如可以用一个文件的多行来表示各类字符，而不需要四个文件分别保存，非常混乱。

## 实验二 LL(1)分析法

1. **实验目的**

通过完成预测分析法的语法分析程序，了解预测分析法和递归子程序法的区

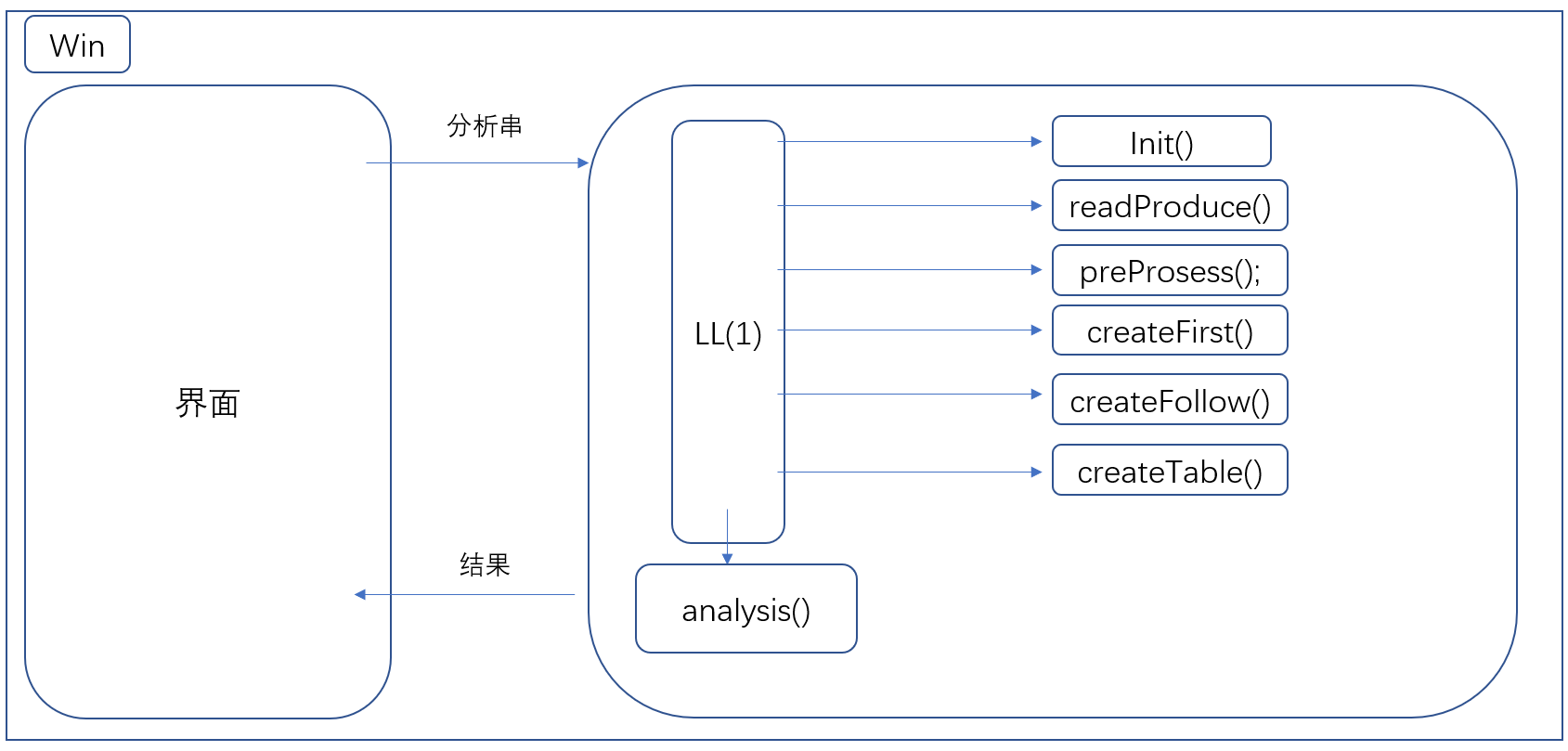
别和联系。使学生了解语法分析的功能，掌握语法分析程序设计的原理和构造方

法，训练学生掌握开发应用程序的基本方法。有利于提高学生的专业素质，为培

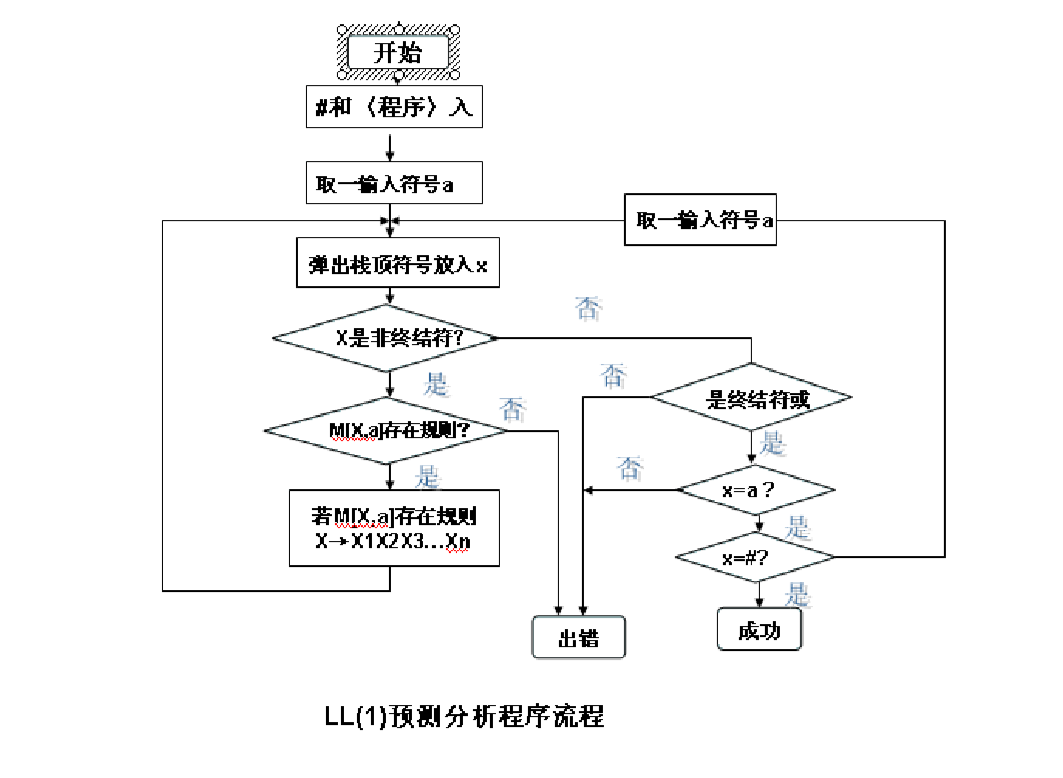
养适应社会多方面需要的能力。

1. **完成的功能**
2. 实现了消除左递归
3. 自动构造first及follow集合
4. 自动构造LL（1）分析预测表
5. 自动对输入串进行分析，并及时报错停止分析
6. 界面化分析表与分析结果
7. 在文件中修改源产生式，不需要在程序中输入
8. **程序结构描述**

|  |  |  |  |
| --- | --- | --- | --- |
| 函数名称及调用格式返回类型 | 参数含义 | 返回值描述 | 函数功能 |
| LL1() | 无 | 无 | 构造函数，程序的总控函数，包含初始化数据结构，构造first与follow表还有预测分析表 |
| void init() | 初始化数据结构 | 无 | 所有数据结构进行初始化 |
| **void** readProduce() | 无 | 无 | 读取文本中的产生式 |
| **void** initFiestAndFollow() | 无 | 无 | 初始化first与follow集的键，也就是添加所有非终结符进去 |
| **void** createFirst() | 无 | 无 | 构造first集合 |
| **void** createFirst(**char** c) | 需要构造的非终结符c | 无 | 非终结符c的firstfirst集构造 |
| Set<Character> getFirst(**char** c) | 需要获取的非终结符c | 非终结符c的first集 | 获取非终结符c的first集 |
| **void** createFollow() | 无 | 无 | 构造follow集 |
| void preProsess() | 无 | 无 | 预处理消除左递归 |
| **void** createPredictTable() | 无 | 无 | 构造预测分析表 |
| **LinkedList<String[]> analysis(String s)** | 待分析串s | 每一行的分析结果 | 输入分析串进行分析并返回分析结果 |

****

1. **详细的算法描述**



由文法构造分析表的流程由四、中给出顺序，算法实现在源码中给出，并严格依据教学概念进行。

1. **程序用法及实验结果**

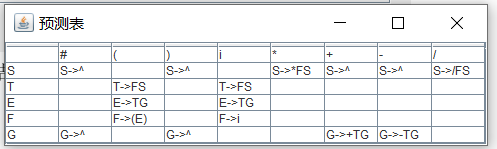
程序用法：在最顶部的输入框中输入待分析的文本，点击分析，就能在下方显示分析结果，分析之后点击分析预测表，就能查看分析表的结果。



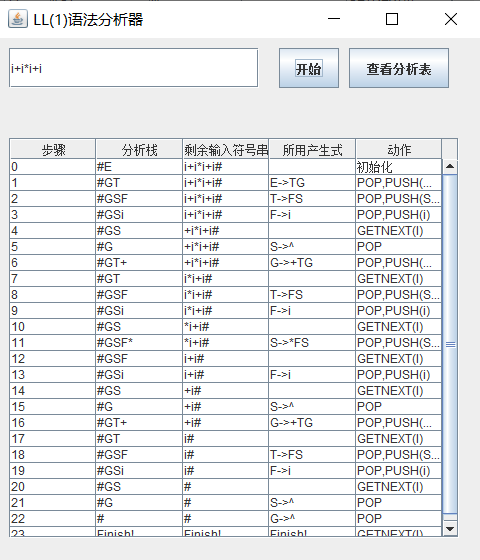
分析结果：



分析表结果：

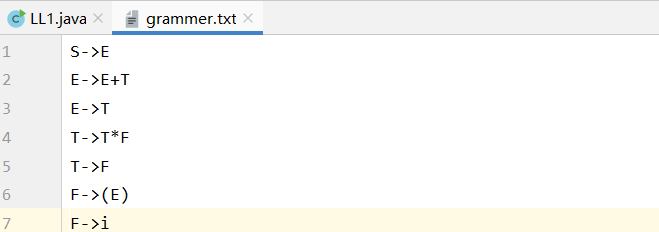


注：验收的时候输出的不是表格，没有对其，现在添加了表格输出，更加美观，还可视化了预测分析表，更直接，不需要在控制台进行打印。

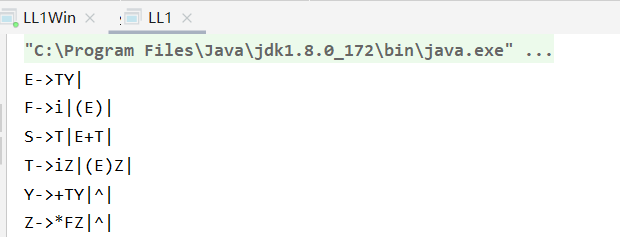


左递归消除演示：

产生式



消除左递归：



1. **实验总结**
2. 实验二有点困难，所有的函数都是自己一行一行打出来的，看着书上的方法，然后化为代码实践，很多地方需要转变一些方法才能做出开。
3. 我所写的消除左递归没有进行化简，也就是没用的产生式没有去掉，待改进。
4. 实验的很多地方都是通过一直循环来实现的，如follow集求法，我的思路是设置一个flag表示是否变化过，然后一直循环直到flag=false，就能求出follow集，这个方法很实用，在很多地方都用得上。
5. 表格显示是后来加上去的，添加了一个类，将表格处理好然后添加进去。
6. 可以进一步优化，可视化first集和follow集。

## 实验三 LR(1)分析法

1. **实验目的**

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。

1. **算法功能实现**
2. 自动构造first与follow集合
3. 自动生成有限自动机
4. 通过自动构造预测分析表
5. 对输入串进行自动分析并可视化结果
6. **实验设计思想及算法**
7. first集的计算方法可以拿实验二的first集构造实验，注意要消除左递归
8. dfa的构造

|  |
| --- |
| **public void** createDFA() {  **int** condition = 0;  **dfa**.put(condition, **new** HashMap<>());  Map<String, Set<Character>> map = **new** HashMap<>();  Set<Character> set = **new** HashSet<>();  set.add(**'#'**);  map.put(**".E"**, set);  **dfa**.get(condition).put(**'S'**, map);  closure(**dfa**.get(condition));  **while** (**dfa**.containsKey(condition)) {  getNextCondition(condition);  condition++;  } } |

先将状态加入到状态0中，然后一直获取下一个状态，并进行闭包，知道没有新的状态产生为止，这个构造的过程才结束。

1. 状态比较

|  |
| --- |
| **for** (**char** c : tmpDFA.keySet()) {  Map<Character, Map<String, Set<Character>>> map = tmpDFA.get(c);  closure(map);  **int** id = cmpCondition(map);  **if** (id == -1) {  **dfa**.put(**i**, map);  tranMap.put(c, **i**++);  } **else** {  tranMap.put(c, id);  } } |

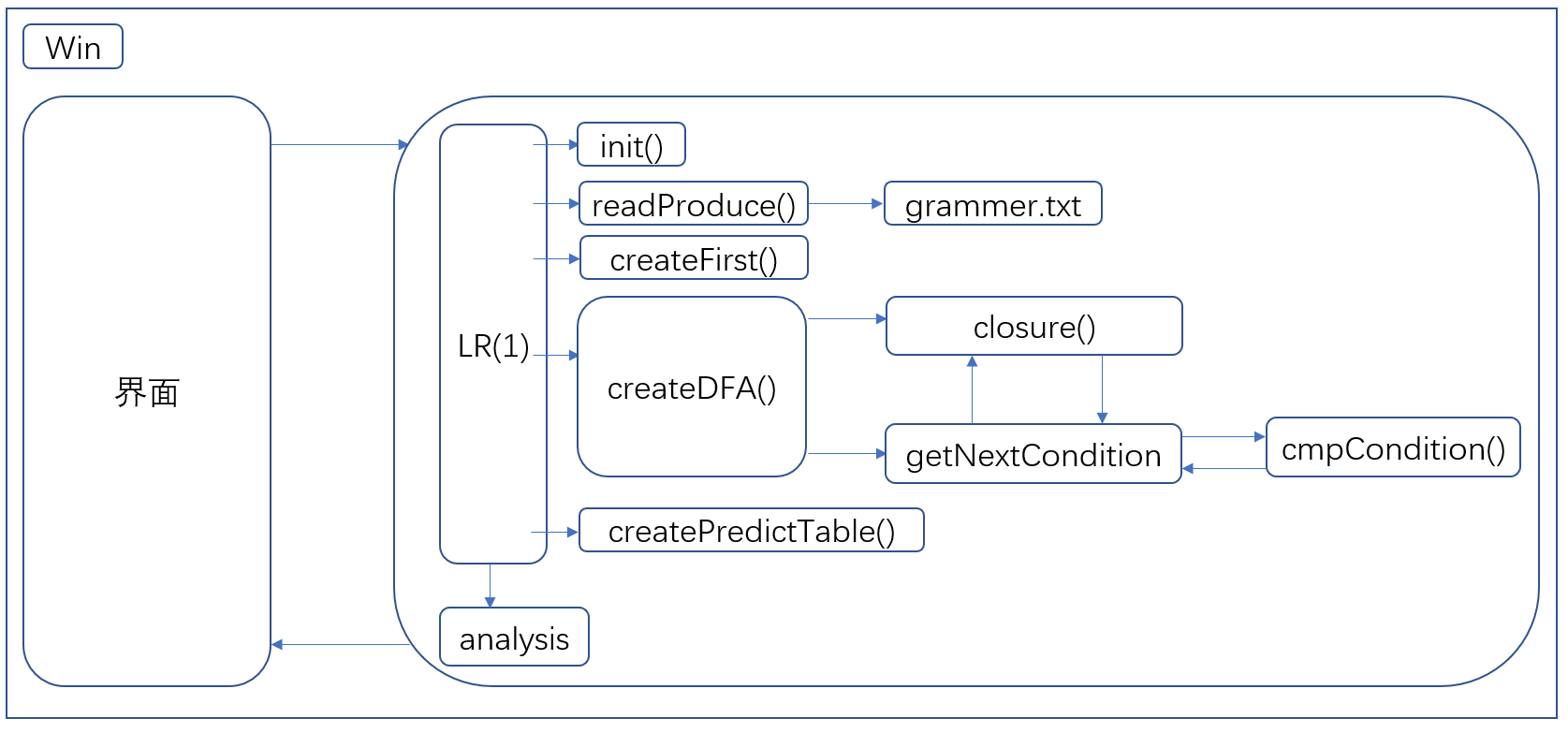
通过一个cmp的比较函数来进行比较，如果返回的是正数，也就是一个现有的状态编号，那么就表示状态已经存在，否则创建一个新的状态，并将状态的转换加入到转换表中，状态加入状态机中。

4、 其余的很多求自动机的代码就不一一讲解，每个函数都有七八十行，可以看源代码，我都写了注释。

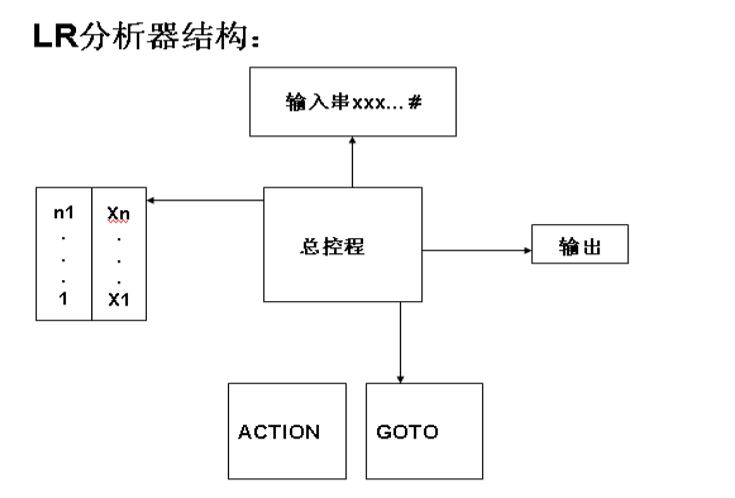
1. **程序结构描述：**

|  |  |  |  |
| --- | --- | --- | --- |
| 函数名称及调用格式返回类型 | 参数含义 | 返回值描述 | 函数功能 |
| LR1() | 无 | 无 | 构造函数，总控程序，将所有模块顺序调用 |
| **void** init() | 无 | 无 | 初始化数据结构 |
| **void** readProduce() | 无 | 无 | 读取产生式 |
| **void** createFirst() | 无 | 无 | 构建first集合 |
| **void** createFirst(**char** c) | 待构建的非终结符c | 无 | 构建字符c对应的first集合 |
| Set<Character> getFirst(**char** c) | 待获取first集的字符c | First集合 | 获取c对应的first集 |
| **void** createDFA() | 无 | 无 | 构造确定性自动机 |
| **void** getNextCondition(**int** n) | 对状态n进行拓展，获取下一个状态 | 无 | 在一个状态闭包后，需要对.之后的字符进行读取，以形成下一个状态 |
| **int** cmpCondition(Map，Set) | Map是原有状态集合，Set是新生成的状态 | 该状态已经存在则返回状态号，否则返回-1 | 对闭包后形成的新状态判断是否已经存在，如果已经存在则返回该状态号 |
| closure(Map) | 状态的Map | 无 | 对状态进行求闭包 |
| **void** createPredictTable() | 无 | 无 | 构造预测分析表 |
| LinkedList<String[]> analysis(String s) | 无 | 待分析文本s | 输入分析串进行分析并返回分析结果 |

1. **模块调用**

****

1. **详细的算法描述**



(1)移进：

table[i，a]= Sj：状态 j 移入到状态栈，把a移入到文法符号栈，其中 i,j 表示状态号。

(2)归约：

table[i，a]=rk：当在栈顶形成句柄时，则归约为相应的非终结符 A，即文法中有 A- B 的产生式，若 B 的长度为 R(即|B|=R)，则从状态栈和文法符号栈中自顶向下去掉 R 个符号，即栈指针 SP 减去 R，并把 A 移入文法符号栈内，j=GOTO[i,A]移进状态栈，其中 i 为修改指针后的栈顶状态。

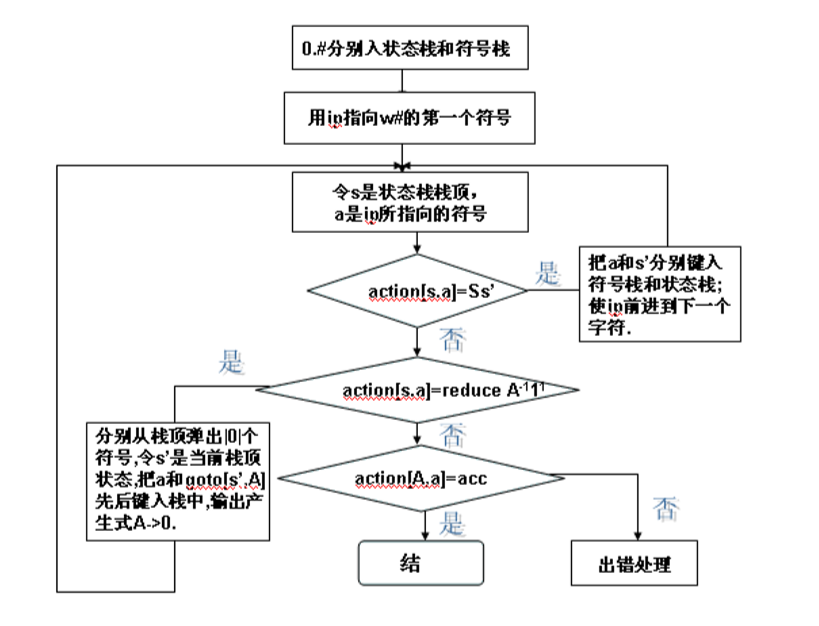
(3)接受 acc:

当归约到文法符号栈中只剩文法的开始符号 S 时，并且输入符号串已结束即当前输入符是'#'，则为分析成功,当table中该项为acc时分析成功接受。

(4)报错:

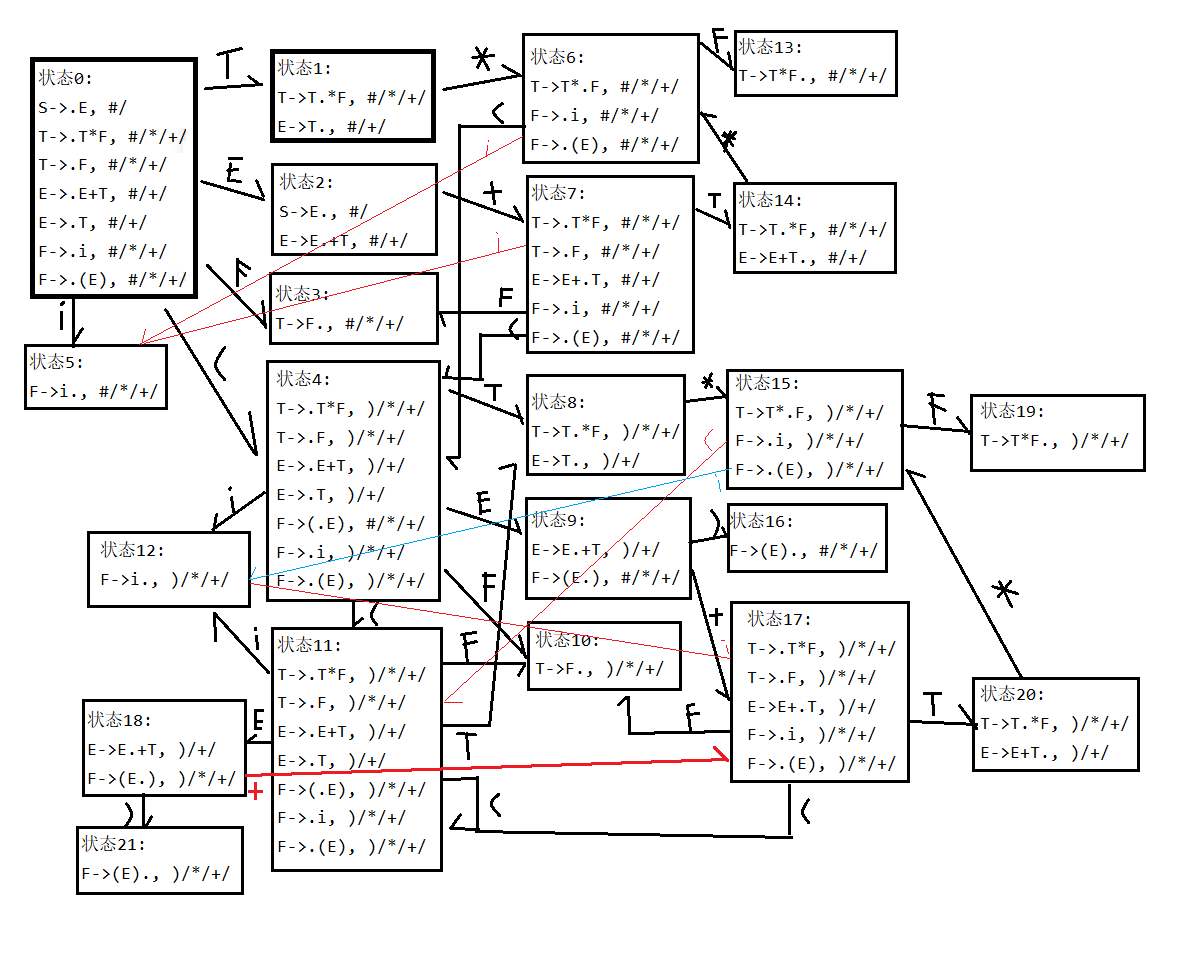
当遇到状态栈顶为某一状态下出现不该遇到的文法符号时，则报错，说明输

入端不是该文法能接受的符号串。



1. **给出软件的测试方法和测试结果。**

通过程序自动生成的状态，我手动画出的状态转换图，结果验证，自动机的状态及转换关系没有问题



分析器：



使用：在输入框中输入待分析串，点开始，下面就能显示出分析结果

注：原本没有用表格输出，没有对其，现在添加表格输出，更美观

实验结果：



与实验指导书上的分析过程一致。



1. **实验总结**
2. 所有函数都是自己手动实验，花了两天时间才写出来，非常锻炼代码的编写能力。
3. 模块化的思想非常重要，一个函数就实现一个函数的特定功能，使代码的逻辑非常清晰。
4. 没有可视化预测分析表，只实现了在控制台打印，实现原理与实验二是一样的，但是由于时间关系没有去完善该功能，有待改进
5. 在求状态机的过程中遇到很多bug,都是由于很微小的错误造成，也让我对Java语言的一些机制特性更加了解。
6. 加强了图形化界面的编程能力。