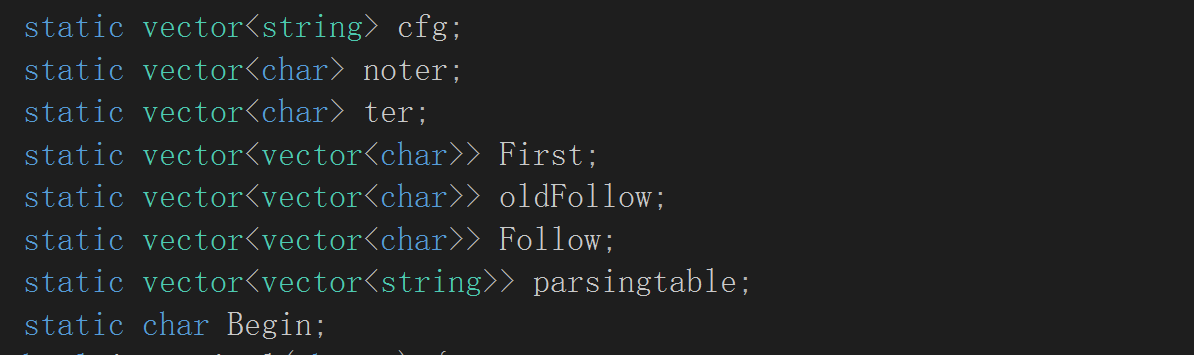
**编译原理：LL（1）文法实现**

——111162 燕飞扬 20161003441

**1、程序规则：**

1. 空字符用$代替，输入时注意替换
2. @表示终止符
3. 大写字母表示非终结符，其他字符为终结符或空字符

**2、数据结构：**上下文无关文法直接用string数组储存，后面要操作产生式的左部和右部时直接遍历字符串即可。



**3、实现思路：**

1. 遍历每个产生式，提取出非终结符和终结符分别存储在noter数组和ter数组里。
2. 求first集（$为空字符）

实现思路根据上课ppt上的过程，用递归实现即可

具体实现思路：

如果产生式右部第一个字符为终结符，则将其计入左部first集

 如果产生式右部第一个字符为非终结符执行以下步骤

求该非终结符的first集

将该非终结符的非$first集计入左部的first集

若存在$，则将指向产生式的指针右移

若不存在$，则停止遍历该产生式，进入下一个产生式

若已经到达产生式的最右部的非终结符，则将$加入左部的first集

处理数组中重复的first集中的终结符

1. 求follow集（@为终止符）

实现思路根据上课ppt的过程，由于求follow集会出现循环，不能用递归实现，我采用填表的方法进行实现

具体实现思路（开始时Follow集都为空）：

1. 如果S为开始符，则将终止符@加入到Follow（S）;
2. 对于每一个产生式S->…B A1 … An

把First(A1)-{$}加入到Follow(B).如果First(A1)不包含$，结束

把First(A2)-{$}加入到Follow(B).如果First(A2)不包含$，结束

…

把First(An)-{$}加入到Follow(B).如果First(An)不包含$，结束

把Follow(S)加入到Follow(B)

1. 重复2过程直到Follow集不再改变（计算一遍Follow集用Follow集和oldFollow集比较，如果相等，则说明Follow集不再改变，否则把Follow集赋值给oldFollow集，再重复计算）
2. 求LL（1） Parsing table

直接根据ppt上构造转换表的算法实现即可

具体实现：

对每个产生式A->a，做:

对每个属于First(a)的终结符b,做T[A,b]=a;

如果a->\*$（空字符），对每个Follow(A)中的终结符b,做T[A,b]=a

1. 用栈验证输入字符串(@为终止符)

假设S为开始符，初始化栈stack=<S,@>，输入字符串s后面追加一个@，用一个指针i指向字符串第一个字符

重复：

将栈顶元素与指针i指向的字符比较，如果相同，则action为terminal，栈顶元素出栈，指针后移i++；

如果不相同，查parsingtable表，得到action，假设查表结果为一个字符串l，栈顶元素出栈，字符串l逆序入栈，如果l为空字符，则不入栈。

如果栈顶元素和指针当前字符都为终止符@，则action为accept

如果指针指向的字符不是终结符或终止符，则action为not accept

直到栈为空。

**4、遇到的问题及解决方案**

问题一：递归求Follow集会出现环

解决方案：Follow集应该开始初始化为空，每次要用Follow集时用当前Follow集的值而不是递归，重复求Follow集直到Follow集不变结束。

问题二：验证字符串时，查表得到的字符串入栈顺序出错

解决方案：应该逆序入栈

**5、总结**

通过这次实习，我对LL（1）文法的过程认识更加清晰。每一步该做什么，怎么做，有什么细节需要注意我都也了一个深入的了解。First集、Follow集、parsingtable表、栈验证输入字符串，这几个关键步骤的思路我理解的更加透彻。总之，LL（1）文法的实现让我对它的认识得到了一个质的飞跃。

**测试结果截图：**



