**LL(1)语法分析程序实验报告**

**1 实验题目**

编写LL(1)语法分析程序

**2 实验内容**

对给定的消除左递归的文法G'：

|  |  |
| --- | --- |
| **编号** | **产生式** |
| 1 | E -> TA |
| 2 | A -> +TA |
| 3 | A -> -TA |
| 4 | A -> ε |
| 5 | T -> FB |
| 6 | B -> \*FB |
| 7 | B -> /FB |
| 8 | B -> ε |
| 9 | F -> (E) |
| 10 | F -> num |

构造LL(1)预测分析程序

**3 构造LL(1)分析表**

**3.1 构造 FIRST，FOLLOW 集**

FIRST(A) 集合是非终结符号A的所有可能推导出的开头终结符或组成的集合。称FIRST(A)为A的开始符号集或首符号集。

对于大部分文法而言，存在一个产生式存在多个候选式的情况，而选择哪一个候选式是不确定的，所以这就产生了回溯。回溯需要消耗大量的计算、存储空间，所以我们需要消除回溯。而消除回溯的其中一种方法叫作“预测”，即根据栈顶非终结符去预测后面的候选式，那预测方法就是求第一个非终结符，来判断是否和读头匹配，以达到预测的效果。

FOLLOW(A) 集合是所有紧跟A之后的终结符或 $ 所组成的集合（$ 是句尾的标志），称 FOLLOW(A)是A的随符集。

当某一非终结符的产生式中含有空产生式时，它的非空产生式右部的开始符号集两两不相交，并与 在推导过程中紧跟该非终结符右部可能出现的终结符集也不相交，则仍可构造确定的自顶向下分析。因 此，引入了文法符号的后跟符号集合 FOLLOW。

FIRST, FOLLOW 集合的构造结果如下表

|  |  |  |
| --- | --- | --- |
|  | **FIRST** | **FOLLOW** |
| **E** | **( num** | **$ )** |
| **A** | **+ - ε** | **$ )** |
| **T** | **( num** | **+ - $ )** |
| **B** | **\* / ε** | **+ - $ )** |
| **F** | **( num** | **\* / + - $ )** |

**3.2 分析表**

设M[A][a]是一个二维数组，其中行A表示的是栈顶符号，a 表示的读头下的符号（A为非终结符，a为终结符），它们存放的是当前状态下所使用的候选式（或存放出错标志，指出A不该面临a的输入），称该数组M为文法的LL(1)分析表。

为了消除回溯，我们进行了FIRST集合和FOLLOW集合的求解。它们两个组合，达到了预测候选式的目的。为了使计算机比较好处理，把它们的预测结果统计成一张二维表。

构造 LL(1) 分析表需要经过如下步骤：

1. 对任意终结符号 a ∈ FIST(A)，将 A → a 填入 M[A,a]；

2. 如果存在 ϵ ∈ FIRST(A)，则对任意终结符号 a ∈ FOLLOW(A)，将 A → a 填入 M[A,a]；

3. 将所有没有定义的 M[A,b] 设置为出错。

构造好的 LL(1) 分析表如下表

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **(** | **)** | **/** | **+** | **-** | **\*** | **num** | **$** |
| **E** | E -> TA |  |  |  |  |  | E -> TA |  |
| **A** |  | A -> ε |  | A -> +TA | A -> -TA |  |  | A -> ε |
| **T** | T -> FB |  |  |  |  |  | T -> FB |  |
| **B** |  | B -> ε | B -> /FB | B -> ε | B -> ε | B -> \*FB |  | B -> ε |
| **F** | F -> (E) |  |  |  |  |  | F -> num |  |

预测分析表M中不含多重定义的表项，说明此文法为LL(1)文法

**3.2 分析过程详解**

预测分析程序的总控程序在任何时候都是按STACK栈顶符号X和当前的输入符号 a 行事的。对于任何 (X,a)，总控程序每次都执行下述三种可能的动作之一：

1. 若 X = a = ‘$’，则宣布分析成功，停止分析过程。

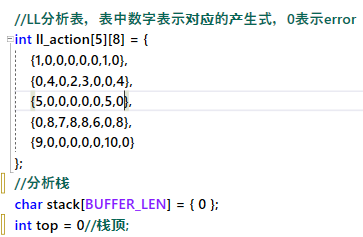
2. 若 X = a != ‘$’，则把X 从STACK 栈顶弹出，让a指向下一个输入符号。 3. 若 X 是一个非终结符，则查看分析表 M。

4. 若 M[X,a] 中存放着关于 X 的一个产生式，那么，先把 X 弹出 STACK 栈顶，然后把产生式的右部 符号串按反序一一推进 STACK 栈（若右部符号为 ϵ，则意味着不推什么东西进栈）。

5. 若 M[X,a] 中为空，则出错并停止分析。

**4 LL(1)分析程序的实现**

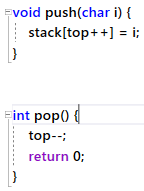
**4.1 定义全局变量**



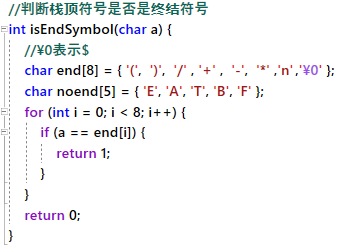
**4.2 主要函数说明**

**注：由于编译器字体原因，字符’\’都将显示为’¥’**

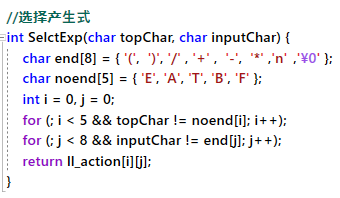
定义入栈出栈操作



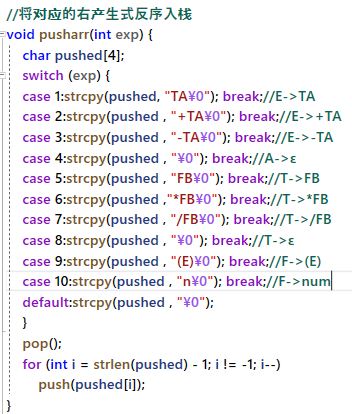
isEndSymbol函数用于判断当前栈顶符号是终结符号还是非终结符号，以便我们确定使用哪种分析动作



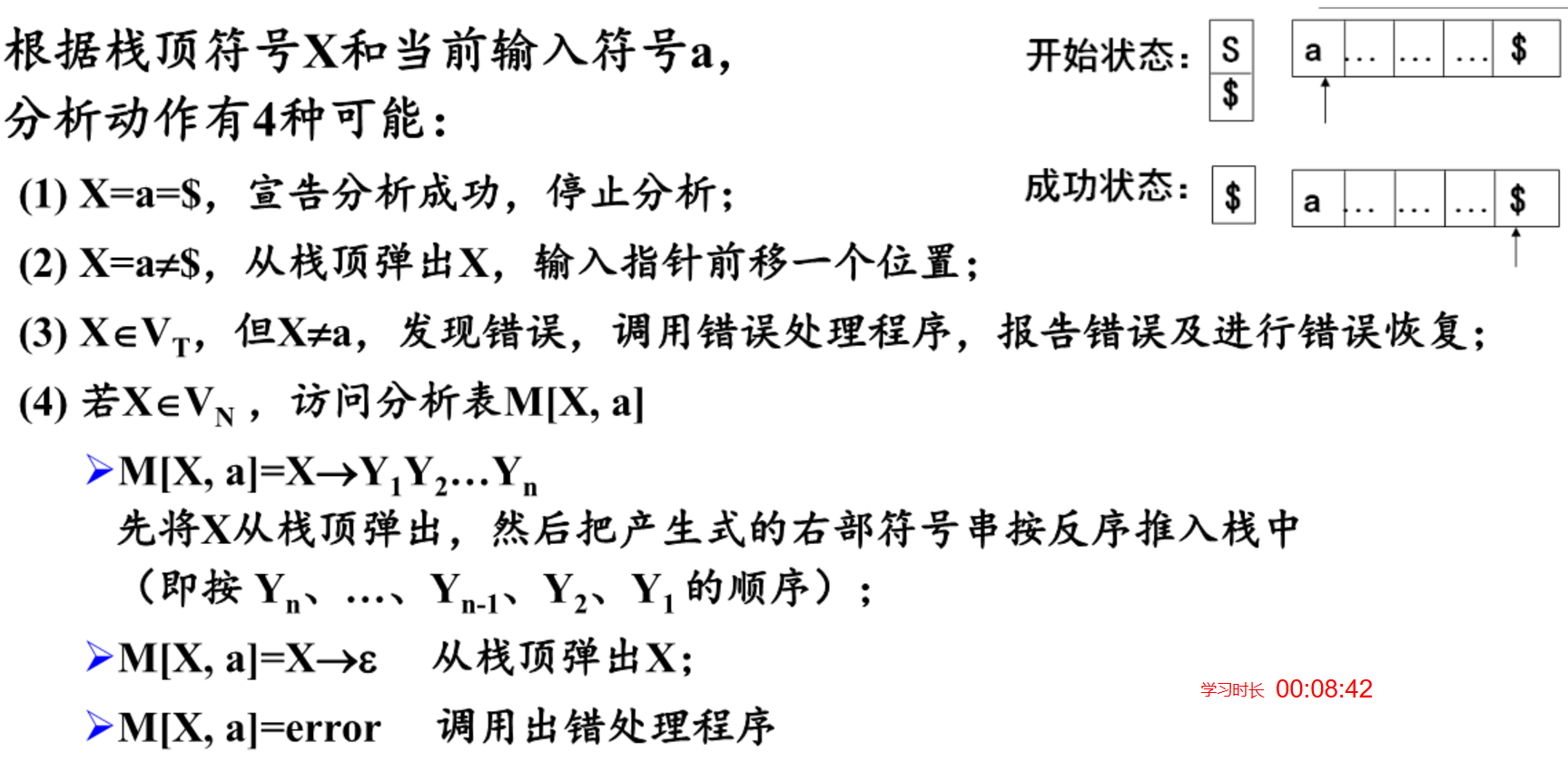
SelcExp函数根据栈顶非终结符和输入符号确定选择分析表中的哪个产生式



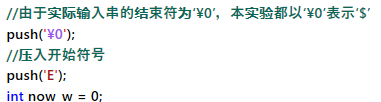
pusharr函数将分析栈顶符号出栈，并将选择的产生式右式反序入栈（X->εz相当于入栈串为空串）



LL函数为本程序主体函数，其程序逻辑都对应下图预测分析控制程序运行逻辑



初始化分析栈



栈顶符号为终结符号时执行分析动作(1) （2）（3）



栈顶符号为非终结符号时执行分析动作（4）



**5 词法分析程序测试**

见文件夹中测试文件及结果

结果均正确