

**实验报告**



**题目：语法分析程序(LL)的设计与实现**

**班 级： 2022211304**

**学 号： 2022211119**

**姓 名： 赵宇鹏**

**学 院： 计算机学院（国家示范性软件学院）**

**2024年 10 月 20 日**

## 实验题目及要求

#### 实验内容：

编写 LL(1)语法分析程序，实现对算术表达式的语法分析。要求所分析算数 表达式由如下的文法产生**。**

**E→ E+T | E–T | T**

**T→ T\*F | T/F | F**

**F→ (E) | num**

要求在对输入的算术表达式进行分析的过程中，依次输出所采用的产生式。

#### **实验要求：**

(1) 编程实现算法 4.2，为给定文法自动构造预测分析表。

(2) 编程实现算法 4.1，构造 LL(1)预测分析程序。

## **程序设计说明**

#### **1.程序总体结构介绍**

程序由三个类：Symbol, Production以及GrammarAnalyzer构成，分为读取文法，消除左递归，生成FIRST集，生成FOLLOW集，生成分析表以及进行预测六个模块，程序的整体模块结构如下图所示：



1. **程序类成员介绍及说明**

Symbol类：该类主要用于存储文法中的各类符号，其类成员有

string value;

bool isTerminal;

其中value用于储存符号的内容，由于符号可以有多个字符如”E’”，因此value采用string类型，isTerminal用于记录该符号是否为终结符，以此完成对一个符号的属性的存储记录。

Production类：该类用于储存文法中的各类转换规则，类成员有

Symbol left;

vector<vector<Symbol>> right;

left用于存储生成式左侧的符号，在实验中进行了一定程度上的简化，只允许左侧有一个符号，right用于存储对应的生成式右侧的符号，由于一个left可能有多个生成式，所以采用vector进行储存并嵌套一层vector单独储存各个符号。

GrammarAnalyzer类：该类为程序的主体类，具体结构如下

map<string, Symbol> symbols;  // 保存终结符和非终结符

vector<Production> productions;  // 保存文法产生式

map<Symbol, set<Symbol>> first;  // FIRST集

map<Symbol, set<Symbol>> follow;  // FOLLOW集

Symbol startSymbol;  // 开始符

map<pair<Symbol, Symbol>, vector<Symbol>> parsingTable;  // 分析表

symbols表为字符串和字符的映射关系，用来保存终结符和非终结符，first表则为Symbol和set<Symbol>的映射关系，其中set用来储存符号所对应的多个字符，follow与first结构相同不再做解释，parsingTable为pair<Symbol, Symbol>与vector<Symbol>的映射关系，在本程序中没有采用二维数组的结构，而是采用映射关系，对存在生成式的进行插入，不存在相应生成式的在parsingTable中不存在，以方便后续匹配的查找。

类成员函数如下：

void readGrammar()：用于读取文法，按照非终结符，终结符，开始符号以及产生式的顺序进行读取，符号之间以','分隔，产生式内符号以' '进行分隔，对存在多个产生式的符号，不同表达式则通过'|'进行分隔，程序逐行读取，对不符合规则的产生式也会进行报错，并将符号和产生式储存进对应的数据结构中。

void eliminateLeftRecursion()：用于消除左递归，程序会首先申明新的非终结符集合以及表达式集合，然后对原来的表达式进行逐个遍历，检查生成式右侧的第一个字符是否与左侧的非终结符相同，若相同则存在左递归，进行消除左递归处理：对原非终结符，将其加上一个’产生一个新的非终结符，若在非终结符表中已有该字符则说明此非终结符已被命名过了，需要再加上一个’来确保非终结符唯一，将新非终结符插入到新非终结符集合中，然后分别存储非递归部分的产生式和递归部分的产生式，并在相应位置加上新非终结符，将生成式插入到新生成式集合中，若没有左递归，则直接将原有生成式和非终结符插入到新生成式集合中，最后将新非终结符集合和新生成式集合赋值给非终结符集合以及生成式集合，完成消除左递归的操作。

void FirstSets()：本函数主要思路与课本上FIRST的生成思路相同，对每个文法符号，若其为终结符，则其FIRST集的内容为它本身，若其不是终结符，那么就查找以它为左侧的产生式，查看生成式右侧，若右侧首个字符为终结符，则把该终结符加入到该符号的FIRST集中，若右侧首字符为非终结符Y，则把FIRST(Y)中的所有非空元素加入到该字符的FIRST集中，若从右侧首字符开始往后i-1个字符都为非终结符且都可以推导到空，则把第i个字符的FIRST集中的所有非空元素加入到该字符的FIRST集中，程序在执行开始时设置一个changed标签，开始时赋值为false，当有符号的FIRST集发生变化则赋值为true，不断循环，直到最后changed的值在循环末尾仍然为false，说明所有FIRST集不再改变，FIRST集生成完成。

void FollowSets()：本函数用于生成非终结符的FOLLOW集，对于文法开始符号S，置$于FOLLOW(S)中，如果有产生式A->α**B**β，则把FIRST(β)中的所有非ε元素加入到FOLLOW(B)中，若有产生式A->αB，或有产生式A->α**B**β，但是ε∈FIRST(β)，则把FOLLOW(A)中的所有元素加入到FOLLOW(B)中，以此循环往复，同样设立一个changed的flag，循环至changed的值为flase不再改变，以此FOLLOW集生成完成。

void generateParsingTable()：本函数为对课本上算法4.2的实现功能，对文法的每一个产生式A->α，对每个终结符号a∈FIRST(α)将A->α放入分析表的表项M[A,a]中，如果ε∈FIRST(α)，那么就对每个b∈FOLLOW(A)则把A->α放入到表项M[A,b]中，然后再实现分析表的匹配错误处理，为分析表加上同步信息，对非终结符E,遍历FOLLOW(E)，若对应表项为空，则填入同步信息synch，就可以得到带有同步信息的预测分析表。

void parsing():语句匹配函数，为课本上算法4.1的实现，首先读取要分析的句子，以$结尾，初始化，将$压入栈底，然后将文法开始符号S压入栈顶，将句子w$放入输入缓冲区中，并置向前指针ip指向w$的第一个符号，然后进行如下分析，如果X是终结符号或$，则进行判断，若X等于a，则从栈顶弹出X，ip向前移一个位置，否则错误，匹配失败，若X不是终结符和$，则查找分析表相应位置处的值，若有产生式则依次把产生式反向压入栈并输出产生式，若为空则进行报错并跳过该符号，向前移动向前指针，若为synch，则进行报错并弹出X，当X等于a等于$时，则说明匹配完成，分析成功，停止分析。

## **测试说明**

**测试一：**

**输入：**

E,T,F

+,\*,(,),id

E

E -> E + T | T

T -> T \* F | F

F -> ( E ) | id

.

id + id \* id $

本输入为课本上文法4.3的输入样例，以方便验证比较程序运行的结果正确与否并检验程序是否能够正常运行，

其消除左递归后得到如下式子：

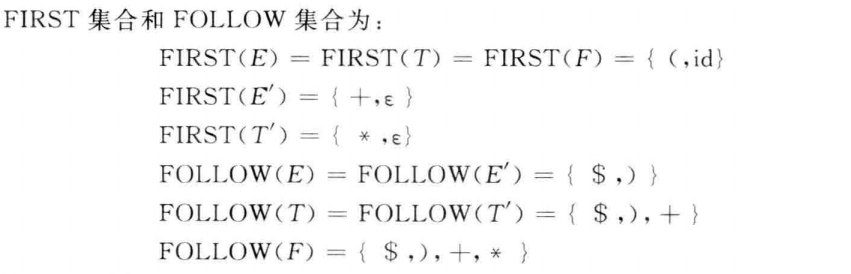
E -> T E’

E’ -> + T E’ | ε

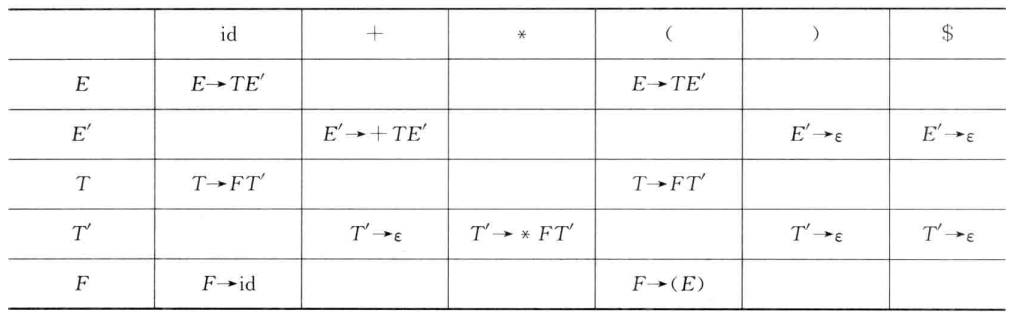
T - > F T’

T’ -> \* F T’ | ε

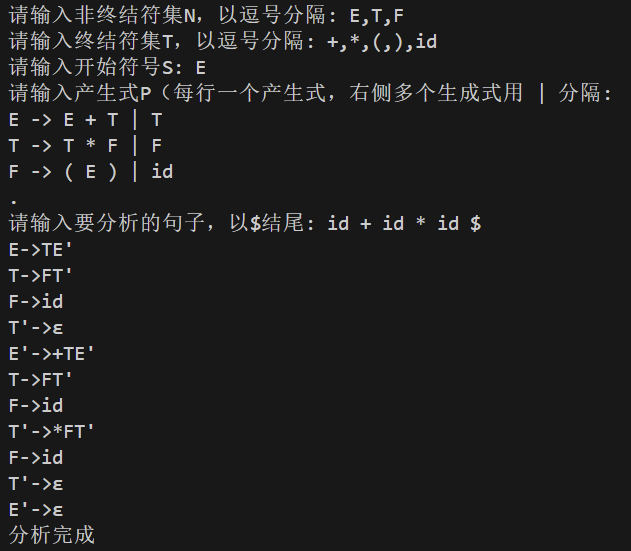
F -> ( E ) | id



预测分析表如下



**输出：**



**分析：**

输出结果与课本上输出结果相同，程序各部分运行正常，对匹配到的生成式则会把栈顶符号先弹出，再以倒序把生成式右侧符号一一压入栈中，遇到终结符若匹配成功则直接弹出，最后栈中只剩$符时匹配结束，程序运行结果符合预期。

**测试二：**

**输入：**

E,T,F

+,\*,(,),id

E

E -> E + T | T

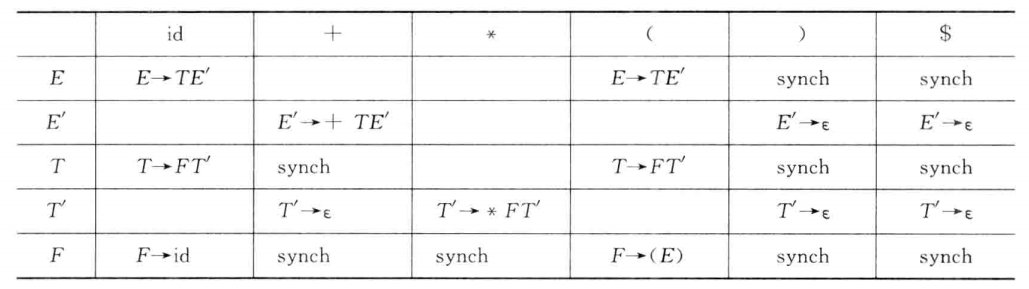
T -> T \* F | F

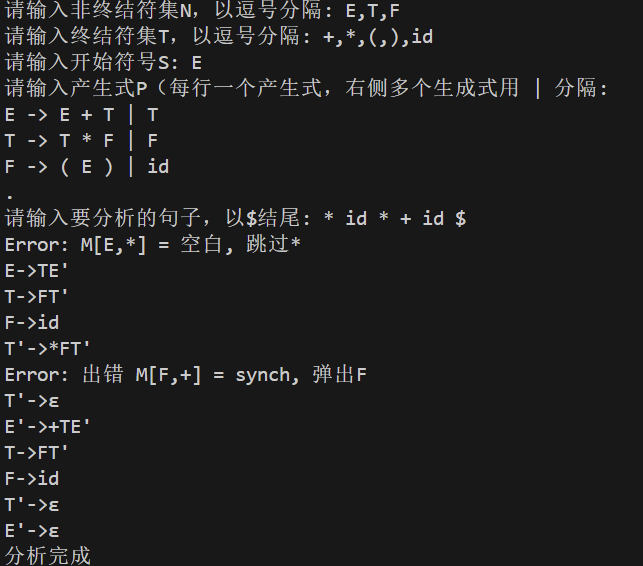
F -> ( E ) | id

.

\* id \* + id $

测试二的输入在测试一的输入上进行了一些小改动，以检验带同步信息的分析表的排错功能。分析表具体内容如下：



**输出：**

**分析**：

程序输出与课本上输出相同，能够正确进行错误处理，在遇到为空的表项时发生错误向前指针前移跳过该终结符，遇到表项为synch时，弹出栈内元素，分析成功。

**测试三：**

**输入：**

**E,T,F**

**+,-,\*,/,(,),num**

**E**

**E -> E + T | E - T | T**

**T -> T \* F | T / F | F**

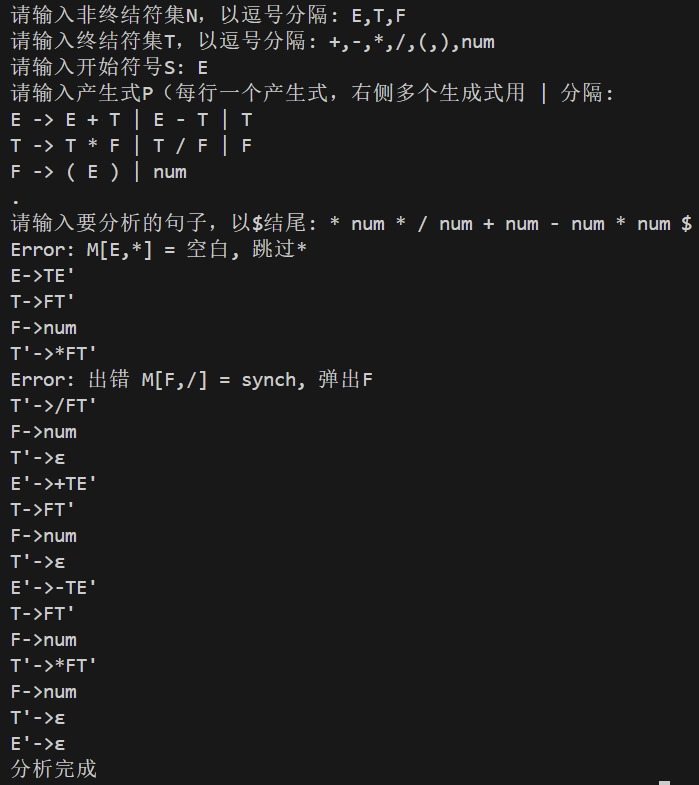
**F -> ( E ) | num**

**.**

**\* num \* / num + num - num \* num $**

本测试样例主要是检验程序对题目所给要求文法识别的能力。

**输出：**



**分析：**

程序正确对开始的\*以及后续的\*/进行了检验和报错，进行了正确的错误处理，并对句子剩余部分进行了正确的分析，符合要求。

1. **实验总结**

本次实验主要是对LL(1)程序进行实现，需要消除左递归，建立FIRST表，FOLLOW表以及对应的预测分析表，将理论与实践相结合，让我对在课堂上没有理解的知识有了更多的了解，加深了我对LL(1)语法分析程序的了解和运用，并能够熟练计算FIRST集和FOLLOW集并画出相应的分析表，收获颇丰。