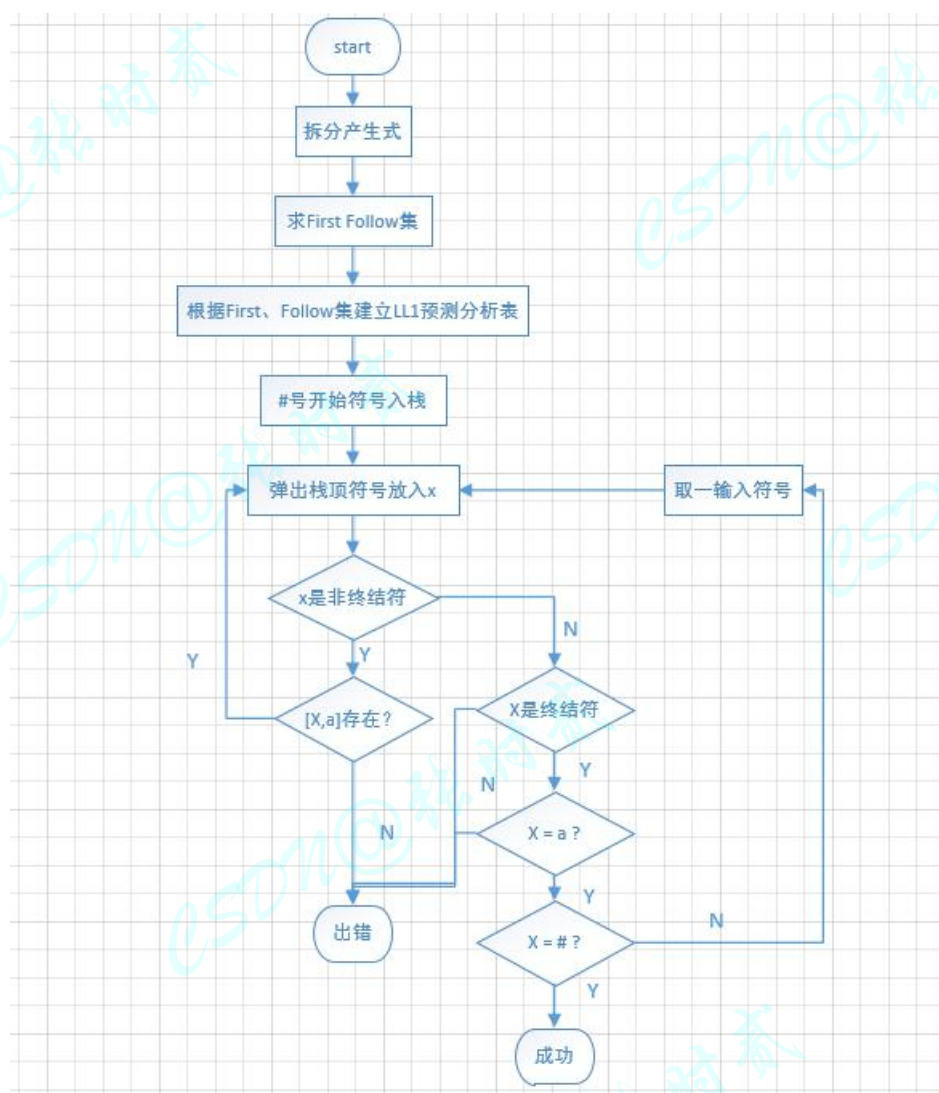


实验三：LL(1)分析法

实验环境	PC 机+Win 2003+VS2017	严禁抄袭 仅供参考 Blog:zhangshier.vip								
<p>一. 实验项目要求</p> <p>根据某一文法编制调试 LL（1）分析程序，以便对任意输入的符号串进行分析。本次实验的目的主要是加深对预测分析 LL（1）分析法的理解。</p> <p>对下列文法，用 LL（1）分析法对任意输入的符号串进行分析：</p> <p>(1) $E \rightarrow TG$</p> <p>(2) $G \rightarrow +TG \mid -TG$</p> <p>(3) $G \rightarrow \varepsilon$</p> <p>(4) $T \rightarrow FS$</p> <p>(5) $S \rightarrow *FS \mid /FS$</p> <p>(6) $S \rightarrow \varepsilon$</p> <p>(7) $F \rightarrow (E)$</p> <p>(8) $F \rightarrow i$</p> <p>输出的格式如下：</p> <p>(1)LL（1）分析程序，编制人：姓名，学号，班级</p> <p>(2)输入一以#结束的符号串(包括+—*/（）i#)：在此位置输入符号串</p> <p>(3)输出过程如下：</p> <table><tr><td>步骤</td><td>分析栈</td><td>剩余输入串</td><td>所用产生式</td></tr><tr><td>1</td><td>E</td><td>i+i*i#</td><td>$E \rightarrow TG$</td></tr></table> <p>(4)输入符号串为非法符号串(或者为合法符号串)</p> <p>备注：</p> <p>(1)在“所用产生式”一列中如果对应有推导则写出所用产生式；如果为匹配终结符则写明匹配的终结符；如分析异常出错则写为“分析出错”；若成功结束则写为“分析成功”。</p> <p>(2)在此位置输入符号串为用户自行输入的符号串。</p> <p>(3)上述描述的输出过程只是其中一部分的。</p> <p>注意：</p> <p>1.表达式中允许使用运算符（+—*/）、分割符（括号）、字符 i，结束符#；</p> <p>2.如果遇到错误的表达式，应输出错误提示信息（该信息越详细越好）；</p> <p>二. 理论分析或算法分析（含实验项目要求的分析、数学或逻辑推导等）</p> <p>1.模块设计：将程序分成合理的多个模块（函数），每个模块做具体的同一事情。</p> <div><div><p>结构体type记录</p><p>大写字符origin</p><p>产生式右边字符array[]</p><p>字符个数length</p><p>type C[][]预测分析表</p></div><div><p>数组</p><p>分析栈A[]</p><p>剩余串B[]</p><p>终结符v1[]</p><p>非终结符v2[]</p></div><div><p>函数</p><p>输出分析栈</p><p>void print()</p><p>输出剩余串</p><p>void print1()</p></div><div><p>do...while</p><p>循环遍历对符号串分析</p></div></div>			步骤	分析栈	剩余输入串	所用产生式	1	E	i+i*i#	$E \rightarrow TG$
步骤	分析栈	剩余输入串	所用产生式							
1	E	i+i*i#	$E \rightarrow TG$							

2.写出（画出）设计方案：模块关系简图、流程图、全局变量、函数接口等。



3.程序编写

- (1) 定义部分：定义常量、变量、数据结构。
- (2) 初始化：设立 LL(1) 分析表、初始化变量空间（包括堆栈、结构体、数组、临时变量等）；
- (3) 控制部分：从键盘输入一个表达式符号串；
- (4) 利用 LL(1) 分析算法进行表达式处理：根据 LL(1) 分析表对表达式符号串进行堆栈（或其他）操作，输出分析结果，如果遇到错误则显示错误信息。

三. 实现方法（含实现思路、程序流程图、实验电路图和源程序列表等）
调试程序，输入符号串 $i+i*#$ ， $i+*#$ 输出并判断是否非法

四、实验结果分析（含执行结果验证、输出显示信息、图形、调试过程中所遇的问题及处理方法等，如果有引用的参考文献，安排在本节最后列出）

分析成功测试：i+i*#

```
Microsoft Visual Studio 调试控制台
LL(1)分析程序，编制人：张帆，192062116，1920542班
输入：以#结束的符号串(包括+ - * / ( ) i #)：i+i*#
步骤：分析栈 剩余字符 所用产生式
0 #E A[++top] = i+i*# 则不进栈*/ E->TG
1 #GT if (A[top] == i+i*# 为空则不进栈*/ T->FS
2 #GSF top--; i+i*# F->i
3 #GSI i+i*# i匹配 S->
4 #GS /*出错处理*/ +i*# G->+TG
5 #G +i*# +匹配
6 #GT+ print(); +i*# T->FS
7 #GT printl(); i+i*# F->i
8 #GSF printf("%c出错\n", x); /*输出出错非终结符*/ i匹配
9 #GS cout << "分析失败" << endl; S->*FS
10 #GSF *i*# *i*# *匹配
11 #GSF *i*# *i*# F->i
12 #GSF *i*# *i*# i匹配
13 #GSF *i*# *i*# S->
14 #GSF *i*# *i*# G->
15 #GSF *i*# *i*#
16 #GSF *i*# *i*#
分析成功
```

分析失败测试：i+*#

```
Microsoft Visual Studio 调试控制台
LL(1)分析程序，编制人：张帆，192062116，1920542班
输入：以#结束的符号串(包括+ - * / ( ) i #)：i+*#
步骤：分析栈 剩余字符 所用产生式
0 #E A[top] = i+*# 则不进栈*/ E->TG
1 #GT if (A[top] == i+*# 为空则不进栈*/ T->FS
2 #GSF top--; i+*# F->i
3 #GSI i+*# i匹配 S->
4 #GS /*出错处理*/ +*# G->+TG
5 #G +*# +匹配
6 #GT+ print(); +*# T->FS
7 #GT printl(); i+*# F->i
8 #GSF printf("%c出错\n", x); /*输出出错非终结符*/ F出错
分析失败
cout << "分析失败" << endl;
return 1;
```