



编译原理实验报告

|  |  |
| --- | --- |
| 专 业 班 级 | 计算机科学与技术19-4班 |
| 学生姓名及学号 | 2019218117 曹辰鹏 |
| 任 课 教 师 | 李宏芒老师 |
| 实验指导教师 | 李宏芒老师 |
| 实验地点 | 计算机实验楼第五机房 |
| 2020 ~2021 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. 可手动输入多行字符并读入进行分析
2. 统计行数和列数用于错误单词的定位
3. 删除空格类字符，包括回车、制表符空格
4. 按拼写单词（关键字、标识符、常数、运算符、关系运算符、分界符号），并用（内码，属性）二元式表示
5. 如果发现错误则报告出错
6. 根据需要是否填写标识符表供以后各阶段使用
7. 识别注释
8. 判别浮点数和整数

**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 |
| bool judgeK(string s) | s为字符串类型，表示输入的字符串 | 返回判断的bool值，判断为真返回true，否则返回false | 判断字符串是不是关键字 |
| bool judgeS(string S) | S为字符串类型，表示输入的字符串 | 返回判断的bool值，判断为真返回true，否则返回false | 判断字符串是不是分界符 |
| bool judgeB(string words) | words为字符串类型，表示输入的字符串 | 返回判断的bool值，判断为真返回true，否则返回false | 判断是不是标识符 |
| bool judgeR(string words) | words为字符串类型，表示输入的字符串 | 返回判断的bool值，判断为真返回true，否则返回false | 判断是不是关系运算符 |
| bool judgeC(string words ) | words为字符串类型，表示输入的字符串 | 返回判断的bool值，判断为真返回true，否则返回false | 判断是不是常数 |
| bool judgeStr(string words) | words为字符串类型，表示输入的字符串 | 返回判断的bool值，判断为真返回true，否则返回false | 判断输入的语句是不是一条合法的语句 |
| int main() | 无 | 0 | 对所有的模块进行综合，得到输入的字符串并进行解析 |

bool judgeStr(string words)

bool judgeC(string words )

bool judgeR(string words)

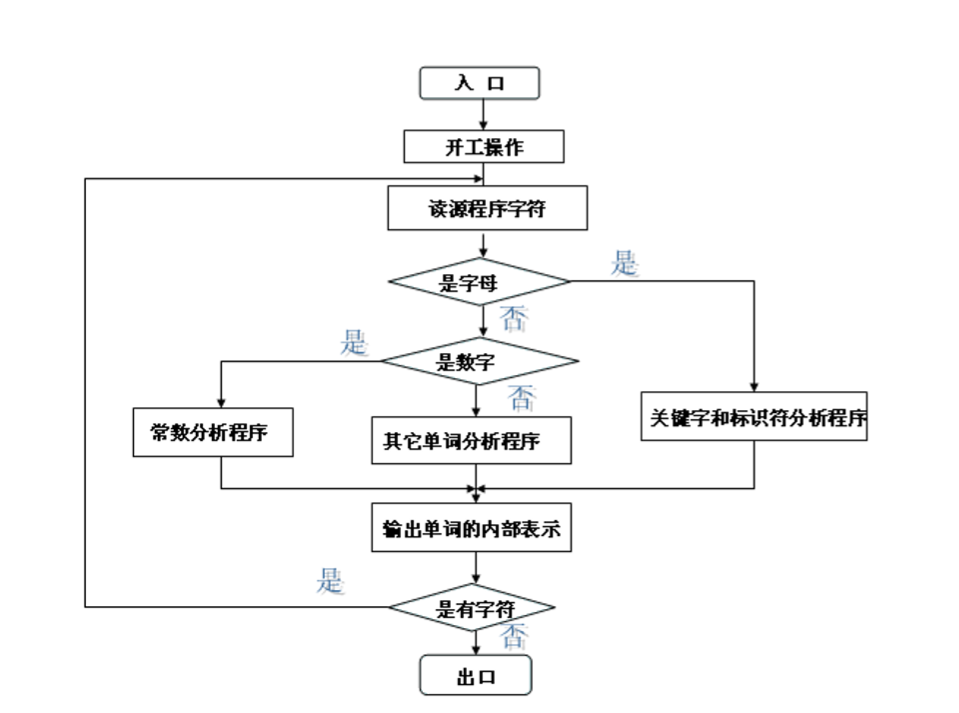
主函数main()

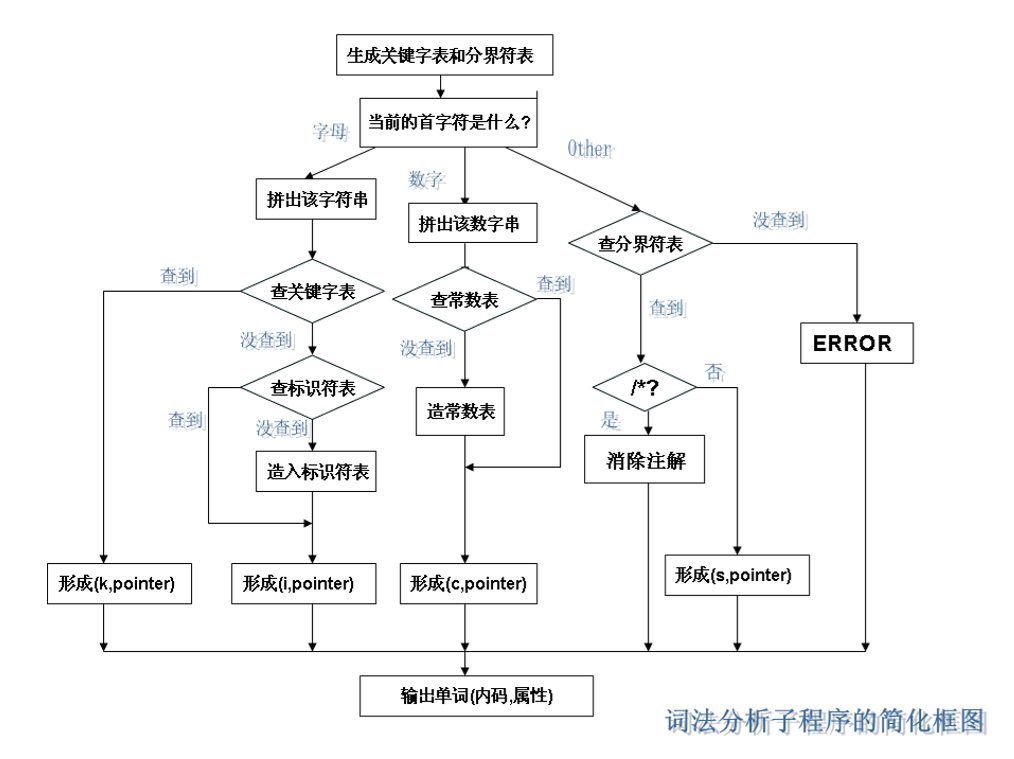
bool judgeS(string S)

bool judgeK(string s)

bool judgeB(string words)

**四、详细的算法描述**



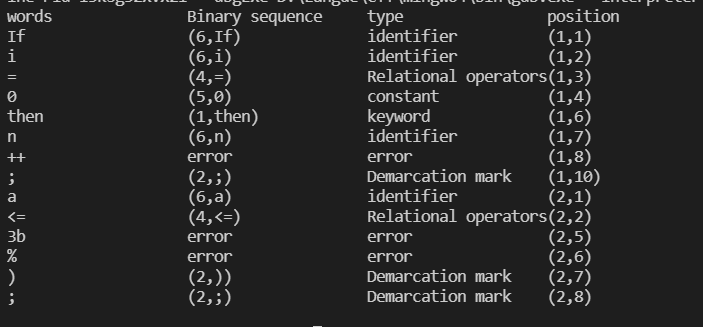


**补充：**

1. 对数字读取若其中有一位"."进行判别为浮点数，若有若干个则判别为"Error"。
2. 若字母后接数字判别为标识符，若数字后面接字母不符合变量命名规则判别为"Error"。
3. 程序大致和这个图2相同，只不过扔出字符串这一步进行了改进。输入的语法都是根据空格进行区分的。所以我在程序中，先写了一个划分程序，根据文件中每一行的内容，按照空字符（空格，\t,\n等）进行划分，然后得到一个存储字符串的vector。如果vector的字符串内容判断不出来是什么类型，就会再次按照图2进行划分。

**五、给出软件的测试方法和测试结果**

先创建一个文件yuan.txt,在这个文件里面输入需要识别的语句，然后再运行程序，可得到结果。



注：程序特别对++号记性了特殊处理，当一个+号后面有+号时，我们选择作为一个整体输出。

**六、实验总结**

1. 这个实验时我从零开始做的，因此对这个程序花费了一些时间。但是我们其实可以发现，这个实验整个就是一个对输入字符串的分析程序，只不过是要求从文件中读入字符串，然后根据输入得到不同放入分析程序中，得到最后的输出结果。
2. 整个实验最重要的部分就是分析程序，分析程序对输入进行了一定程度的改进。分析程序的输入都是一段一段的字符串。也就是从文件得到的字符串，然后按照空串进行划分得到的字符串。这样就能减少一些对字符串整体的判断，如果一个一个判断字符串的字符的话，可能有点麻烦。而且这样也方便了后序程序的编写。方便分析。

## 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. **写出 LL（1）分析法的思想及写出符合 LL（1）分析法的文法**

对文法G的句子进行确定的自顶向下语法分析的充分必要条件是，G的任意两个具有相同左部的产生式A—>α|β 满足下列条件：

（1）如果α、β均不能推导出ε，则 FIRST(α) ∩ FIRST(β) = ∅。

（2）α 和 β 至多有一个能推导出 ε。

（3）如果 β \*═> ε，则 FIRST(α) ∩ FOLLOW(A) = ∅。

将满足上述条件的文法称为LL(1)文法。

**First构造：**

对每一文法符号X∈VT∪VN构造FIRST(X)，连续使用下面的规则，直至每个集合FIRST不再增大为止：

1. 若X∈VT，则FIRST(X)＝{X}。

2. 若X∈VN，且有产生式X→a…，则把a加入到FIRST(X)中；若X→ε也是一条产生式，则把ε也加到FIRST(X)中。

**Follow构造：**

对于文法G的每个非终结符A构造FOLLOW(A)的办法是，连续使用下面的规则，直至每个FOLLOW不再增大为止：

1. 对于文法的开始符号S，置＃于FOLLOW(S)中；

2. 若A→αBβ是一个产生式，则把FIRST(β)\{ε}加至FOLLOW(B)中；

3. 若A→αB是一个产生式，或A→αBβ是一个产生式而β→ε (即ε∈FIRST(β))，则把FOLLOW(A)加至FOLLOW(B)中。

**文法：**

1. E->TG
2. G->+TG|—TG
3. G->ε
4. T->FS
5. S->\*FS|/FS
6. S->ε
7. F->(E)
8. F->i

**注：** 由于ε在程序中实在是不好输入，故所有的ε都在程序中用@表示。

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 |
| def splitG(gramma): | 输入的文法 | 消除->的文法的列表 | 对输入的文法进行解析，消除->，并且文法中出现|的语句拆分成两个语句。 |
| def getFirst(v): | 输入文法字符 | 该字符对应的FIRST集（内容是字典） | 得到输入字符对应的FIRST集 |
| def getFollow(curVn): | 文法字符 | 该字符对应的FOLLOW集（内容是字典） | 得到输入字符的FOLLOW集 |
| def getLastFirst(cur): | 文法生成式 | 这个式子对应的First集 | 德奥式子对应的FIRST集 |
| def getSelect(): | 无 | 无 | 得到select表，数据存储到全局变量M中 |
| def getResult(sentence): | 需要进行分析的句子 | 分析过程中每一步对应的栈，输入字符串，输出内容 | 根据我们构造的SELECT表对输入字符串进行逐步解析 |
| class Stack(object): | 无 | 无 | 解析需要用到的栈 |
| def main(s): | 输入字符串 | 分析过程中每一步对应的栈，输入字符串，输出内容以及select表 | 对上边所有模块进行一个综合，得到最后的结果 |

注：该程序进行到最后，为了实现可视化，将写的这些模块全部封装成一个main函数,根据输入字符串得到最后的输出。之后又写了一段程序，用于进行可视化GUI的编程。在GUI 的实践中绑定mian函数。GUI输出最后的结果

**Main()**

GUI

def splitG(gramma):

def main(s):

def getFirst(v):

    def getFollow(curVn):

    def getLastFirst(cur):

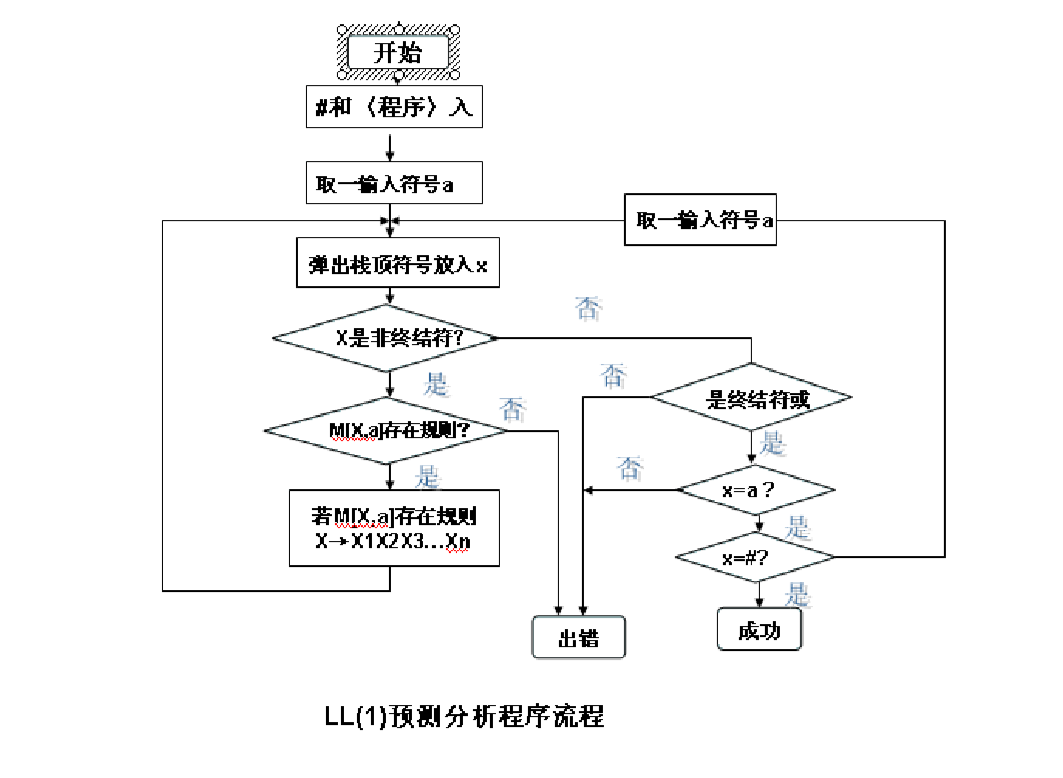
def getSelect():

    def getResult(sentence):

注：上图左边是main函数的执行过程，这个是按照图中所示按照顺序进行的。右边的图是整体进行的

1. **详细的算法描述**

LL(1)分析



FIRST集：

因为存储生成式的是一个列表，不过列表中的生成式已经将->消去，并且将字符分开。所以每个列表元素的第一个元素就是指生成式的左边，其他的元素指的就是生成式的右部。按照给出的FIRST集生成的方法，遍历每一个生成式，如果生成式的左边是我们需要求解的字符，就看该生成式的又不第一个，如果是终结符，直接放入对应的字典中，如果不是，如果第一个字符还没有FIRST集就，用递归得到第一个字符的FIRST集，有的话，直接给出，最后就可得到输入字符的FIRST集。

FOLLOW集：  
遍历所有的生成式右部，找到输入字符的位置，得到右边的字符。如果右边的字符是一个终结符，那就将该终结符放入到对应的FOLLOW集中，如果不是，判断右边字符是否能推导出空串，如果能，就将递归求出当前FOLLOW集之后，将空串去掉，不能推导出空串，就直接递归求出FOLLOW集。

每条语句的first集：

求出这个目的是可以根据求出的结果，再结合生成式的左部，方便得到SELECT集。直接遍历每一个生成式的右部，得到第一个字符的FIRST集，如果其能推导出空串，则找到下一个字符，继续球出FIRST集。

SELECT集：

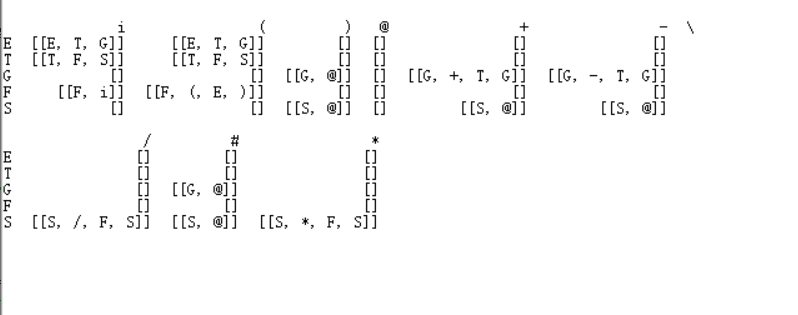
根据求出的每个生成式的FIRST集，再根据每条生成式的右部，既可以将SELECT表填充上（是一个二维列表，但是每一个元素依旧是列表，用来存储生成式）。

1. **给出软件的测试方法和测试结果。**

可根据不同输入串做出不同分析并清空列表后输出：



SELECT输出：



注：中间第二位的箭头省略了。

1. **实验总结**
2. 此次试验全程由个人完成，我尝试这消除左递归，但是最后只是消除了直接左递归，没有消除间接左递归。为了方便后续写程序的方便，我把数据结构设计成每一个生成式都对应与一个列表，但是生成式的->消去。这样方便了后续程序的遍历。
3. 从文法分析到分析表的构造，中间涉及非终结符终结符的提取，first集合以及follow集合的构造，完全按照定义一步步进行，将生成的结果放入到对应的字典中，方便后来的查询取出。
4. 本实验的程序框架是很清晰的，目的就是分析输入字符串，想要得到结果，就要求SELECT表。而要得到select表，就要求出每个生成式的FIRST集，所以要求出每个非终结字符的FIRST集和FOLLOW集。整体框架也就出来了，整体来说还是非常清晰的。
5. 在写程序的时候，用到了大量的全局变量，使得程序的耦合性有了一定的影响。下次，再写这样的程序就要考虑尽量少用全局变量，就算可能会增加空间复杂性也要防止耦合性出现影响。这样再写类似的程序的时候就可以直接调用之前写好的程序，不用再重新写一遍。
6. 该程序使用了tkinter框架进行对话框形式的设计，由于较长时间没有上手对此生疏，所以临时学习了一下。最后直接把按钮的事件绑定成运行main函数,得到结果输出到text上就行。自己设计的GUI还是和想象中的差别比较大的，不过好在该实现的功能都实现了。

## 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. **描述 LR(1)语法分析程序的设计思想**

**1.提取所有有效识别活前缀的式子：**

形式上我们说一个LR(1)项目[A→α·β, a]对于活前缀γ是有效的，如果存在规范推导其中，

1) γ＝δα；

2) a是ω的 第一个符号，或者a为#而ω为ε。

[A→α·Bβ, a]对活前缀γ＝δα是有效的，则对于每个形如B→ξ的产生式， 对任何b∈FIRST(βa)，[B→·ξ, b]对γ也是有效的。

**2.项目集I 的闭包CLOSURE(I)构造：**

1. I的任何项目都属于CLOSURE(I)。
2. 若项目[A→α·Bβ, a]属于CLOSURE(I)，B→ξ 是一个产生式，那么，对于FIRST(βa) 中的每个终结符b，如果[B→·ξ, b]原来不在CLOSURE(I)中，则把它加进去。
3. 重复执行步骤2，直至CLOSURE(I)不再增大为止。

**3.GO构造：**

令I是一个项目集，X是一个文法符号，函数GO(I，X)定义为：GO(I，X)＝CLOSURE(J)其中J＝{任何形如[A→αX·β, a]的项目| [A→α·Xβ, a]∈I}

**4.分析表构造：**

令每个Ik的下标k为分析表的状态，令含有[S′→·S, #]的Ik的k为分析器的初态。

1. 若项目[A→α·aβ, b]属于Ik且GO(Ik, a)＝Ij， a为终结符，则置ACTION[k, a]为 “sj”。
2. 若项目[A→α·，a]属于Ik，则置ACTION[k, a]为 “rj”；其中假定A→α为文法G′的第j个产生式。
3. 若项目[S′→S·, #]属于Ik，则置ACTION[k, #]为 “acc”。
4. 若GO(Ik，A)＝Ij，则置GOTO[k, A]=j。
5. 分析表中凡不能用规则1至4填入信息的空白栏均填上“出错标志”。

注：在程序表中，为了程序方便，并没有将ACTION表和GO表单独列出来，而是综合成一个表。但是我们依旧是可以看到最后几列是关于GO表的。能够很明显地区别出来，只不过是没标注罢了。

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 |
| def splitG(gramma): | 输入的文法 | 消除->的文法的列表 | 对输入的文法进行解析，消除->，并且文法中出现|的语句拆分成两个语句。 |
| def getFirst(v): | 输入文法字符 | 该字符对应的FIRST集（内容是字典） | 得到输入字符对应的FIRST集 |
| def getLastFirst(cur): | 文法生成式 | 这个式子对应的First集 | 得到式子对应的FIRST集（注意和实验二的有所不同，这个函数是将输入的语句看做是生成式的右部，然后求出结果） |
| def judgeAandB(v1,v2): | 两个列表 | Bool | 判断两个list是否有重合 |
| def judgeAinB(v1,v2): | V1表示的生成式列表，v2表示的是闭包列表 | 在闭包，返回True和对应的位置，否则返回Faslse,-1 | 判断生成式v1是不是在闭包v2中 |
| class Stack(object): | 无 | 无 | 解析需要用到的栈 |
| def getSet(v1,v2): | list | List | 得到两个列表的并集（二维及以上不适用） |
| def getCLOSURE(b): | 生成式列表 | 生成式对应的闭包列表 | 对输入的生成式求出其闭包。 |
| def deletePoin(V): | 生成式 | 将点的位置后移一位的生成式 | 将生成式的点后移一位 |
| def judgeAandB\_B(A,B): | 生成式A，  闭包B | A在B中返回True，否则返回False | 断生成式是否在闭包内，包括展望 |
| class obj(): | 无 | 无 | 每一个闭包对应的属性：  num:表示第几号闭包  queue：每个闭包的表达式  surper：用来记录指向这一个状态的上一状态标号和遇到的字符。  type:表示这个闭包是状态还是该规约了 |
| def judgeAinB\_DFA(A,B): | 闭包A，项目集B | 如果A在B中，返回True，否则返回False | 求闭包是否在项目集中 |
| def getSet2(A,B): | 二位列表A,B | 两个列表的并集 | 求二维数组的并集 |
| def getDFA(): | 无 | 项目集列表 | 根据第一个式子得到存储项目集的列表 |
| def getM(p): | 项目集列表，每一个元素是obj类型 | 分析列表dataFrame类型 | 根据输入的项目集得到分析表 |
| def main(s\_get): | 输入字符串 | 分析过程中，各个栈的状态，字符串类型 | 对输入字符串进行LR（1）分析。 |

注：这里的有一部分函数是和实验二相同的。项目集是一个列表，不过每一个元素是obj类型。每个obj类型就是用来存储每个闭包的状态的。其实还有一个函数，但是这个函数是用来判断一个字符是否在一个列表中的，因为过于简单，没有列出来。

def splitG(gramma):

def getFirst(v):

def getLastFirst(cur):

class Stack(object):

def getCLOSURE(b):

class obj():

def getDFA():

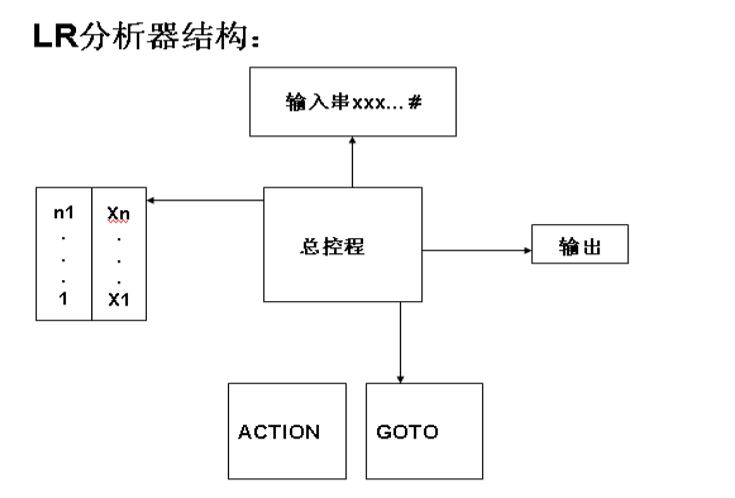
def getM(p):

def main(s\_get):

GUI显示

注：没有列出其他函数是因为，其他函数基本就是分布在这些函数中的，他们是用来完成特定的功能，耦合性很低，只不过是为了增加程序可读性才封装成函数。而且现在列举的函数和结构已经将整个程序的框架勾勒出来了。

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。

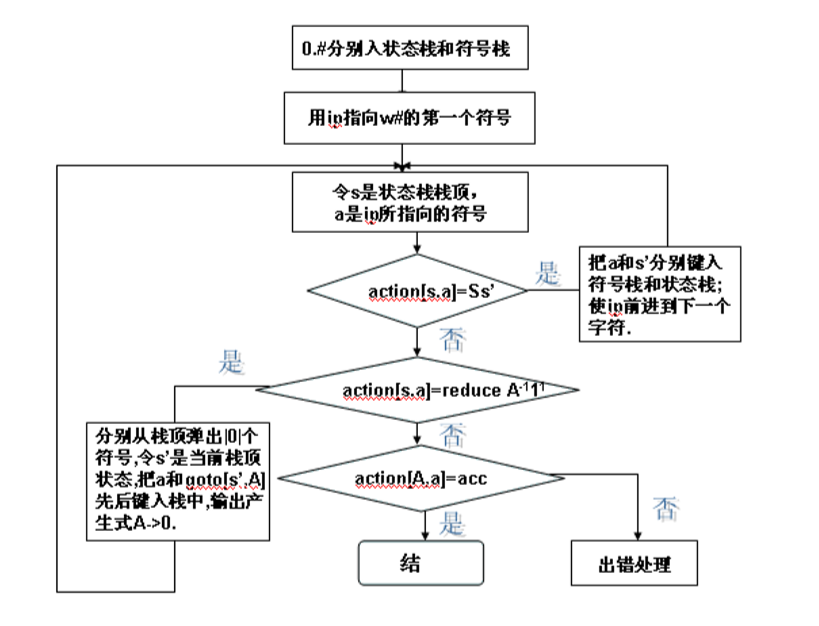
(3)接受 acc:

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。



FIRST集的求法和实验二相同。

求闭包：

根据生数学表示求。大概就是找到输入语句点的位置，然后根据当前位置和后边第一位是否是非终结符来进入不同的状态。如果点的位置已经是最后或者后边第一位是终结符，则退出，否则，就要开始求出新的生成语句。新的生成语句根据点的后边第一位遍历所有生成式得到。然后根据上一个生成式求出当前生成式的展望式，判断生成式是不是已经在闭包中，在的话，就将生成式的展望式合并。

求项目集：

根据第一个式子生成闭包，然后封装成obj，之后，将该闭包中遇到不同字符生成的不同生成式作为下一个状态的首个生成式，生成对应的闭包，然后封装，并将当前状态和遇到的字符封装成列表放入到super中。对新生成的状态进行上述操作，直到所有的状态已经全部生成完毕。注意:一个状态可能遇到同一个字符生成不同的状态。这时候就要将这些不同的状态全部合并成一个状态。

生成表：

遍历每一个项目集的元素，将对应的super遍历，然后根据当前状态生成对应的生成表。当当前的状态是S的时候，正常生成生成表，如果是R就要将闭包中的生成式拿出来找到是第几个生成式，在对应的生成表位置放入。

1. **给出软件的测试方法和测试结果。**

**给定的文法：**

E->E+T

E->T

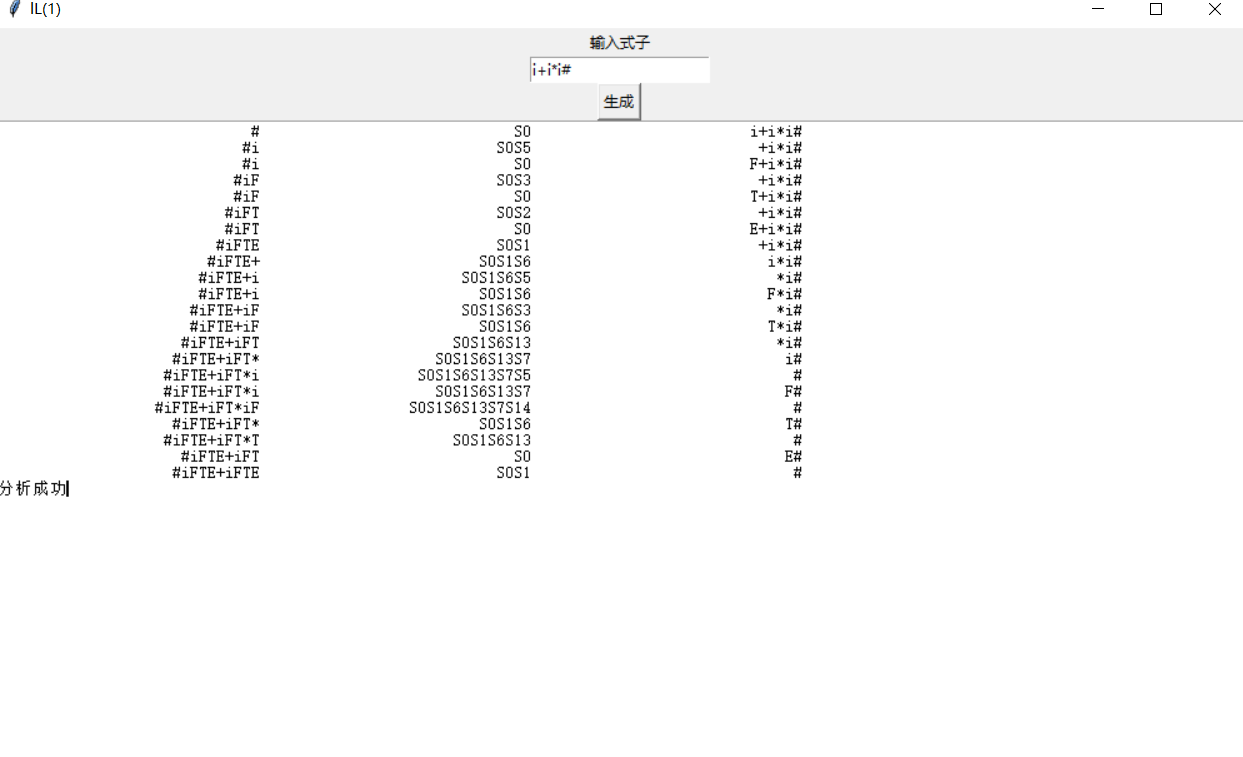
T->T\*F

T->F

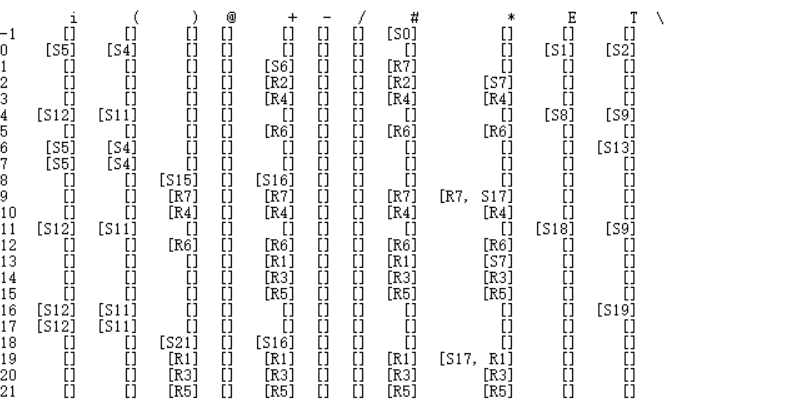
F->(E)

F->i

**输入字符串:i+i\*i#**



**生成表：**





1. **实验总结**
2. 对于first和follow集合求取函数以及getLastFirst()函数由试验进行修改后直接复用。
3. 在实验过程中，每一步都尽量做到独立，求闭包的程序和求项目集的程序互不干扰，只是简单的调用问题。
4. 在求闭包的时候，刚开始没有想到有相同的式子，所以导致闭包一直出错。后来发现问题之后，又发现新的生成式语句可能由于展望式不同又放到了闭包中。然后就改进，将相同生成式的展望式也求出并集。在求并集的时候会发现使用了set这个本来我以为会导致并集的顺序会打乱。但是后来发现我多虑了，结果显示并不会打乱原有的顺序，只是在本来的基础上在末尾加上。
5. 在求项目集的时候发现一个问题，当当前的状态遇到同一个字符的时候，可能会生成相同闭包，刚开始把这个当做新的状态处理，然后就有了问题。当遍历到当前状态的时候，不知道该向哪一个状态转换。所以，生成相同的状态要进行合并。
6. 在生成表时，我并没有对R状态的项目进行处理，导致出现规约不知道该用什么规约，规约的字符有多少个。后来发现后及时改正完美生成生成表
7. 最后的分析程序就比较简单了，根据数学表达式写就行了，只不过需要注意的是每次规约之后，都要将规约的字符放入到输入字符串中，还要得到当前状态栈栈顶的状态数，便于下一次循环转换状态。最后我们只要将每次的各个栈的状态用字符串存储，输出即可。
8. 该程序使用了tkinter框架进行对话框形式的设计，用的是实验二的代码。只要将往TEXT输入的字符串改一下就行了。