



编译原理实验报告

|  |  |
| --- | --- |
| 专 业 班 级 | 计算机科学与技术20-2班 |
| 学生姓名及学号 | 2020214231 何煦 |
| 任 课 教 师 | 李宏芒老师 |
| 实验指导教师 | 李宏芒老师 |
| 实验地点 | 科教楼C304 |
| 2021 ~2022 学年第二学期 | |

本次编译原理实验Github网址：

https://github.com/TENETX/bianyiyuanliTest

## 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. 可读取txt文件内的字符段，结合分析表进行分析，且有报错处理。
2. 可读取txt文件内的分析表，创建分析表，且有报错处理。
3. 可手动输入属性与字符串创建分析列表。
4. 统计行数和列数用于错误单词的定位。
5. 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。
6. 如果发现错误则报告出错。
7. 根据需要是否填写标识符表供以后各阶段使用。
8. 识别注释。
9. 判别浮点数和整数。

还有其他功能，具体就不再赘述。

## 三、程序结构描述

本程序总共分为两个py文件：



**图1. 项目整体介绍**

以下为main.py的函数介绍：

|  |  |  |  |
| --- | --- | --- | --- |
| main.py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 |
| judge(word)  返回：int | 需判断的分割出来的一个字串 | 属性列表对应的序号 | 判定函数，即判定该字串的属性值并返回一个整型序号值 |
| createbiao(filename)  返回：无 | 分析表文件的文件名 | 无 | 读取txt文件并且创建分析表供下一步分析使用 |
| main(filename)  返回：无 | 字符段文件的文件名 | 无 | 读取txt文件中的字符串同时调用上面的两个函数进行分析，最后得到结果。 |

以下为main.py的变量介绍：

|  |  |  |
| --- | --- | --- |
| 变量名 | 变量类型 | 变量功能 |
| k | list | 关键字列表 |
| s | list | 分界符列表 |
| aop | list | 算数运算符列表 |
| rop | list | 关系运算符列表 |
| comeout | list | 输出列表 |
| result | list | 结果列表 |

main.py主要负责文件读取，初步创建分析列表，以及内码的属性分析和结果储存。

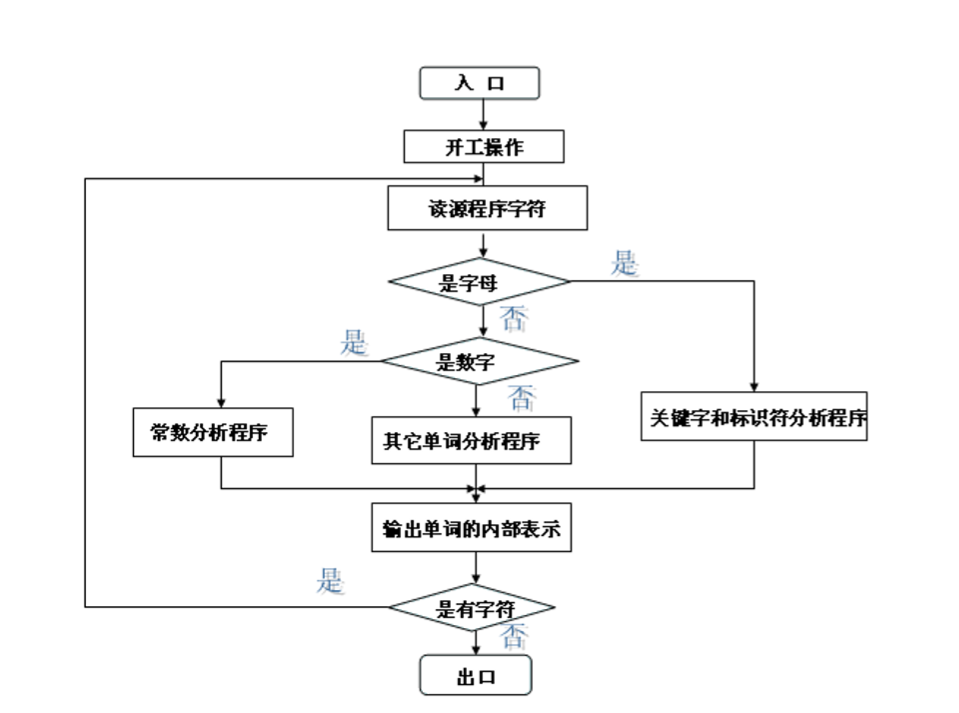
以下为api.py关键的几个函数介绍：

|  |  |  |  |
| --- | --- | --- | --- |
| api.py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 |
| go()  返回：无 | 无 | 无 | 整个程序的启动函数 |
| renewbiao(self)  返回：无 | 整个界面类 | 无 | 将main.py里面result列表里的结果进行显示 |

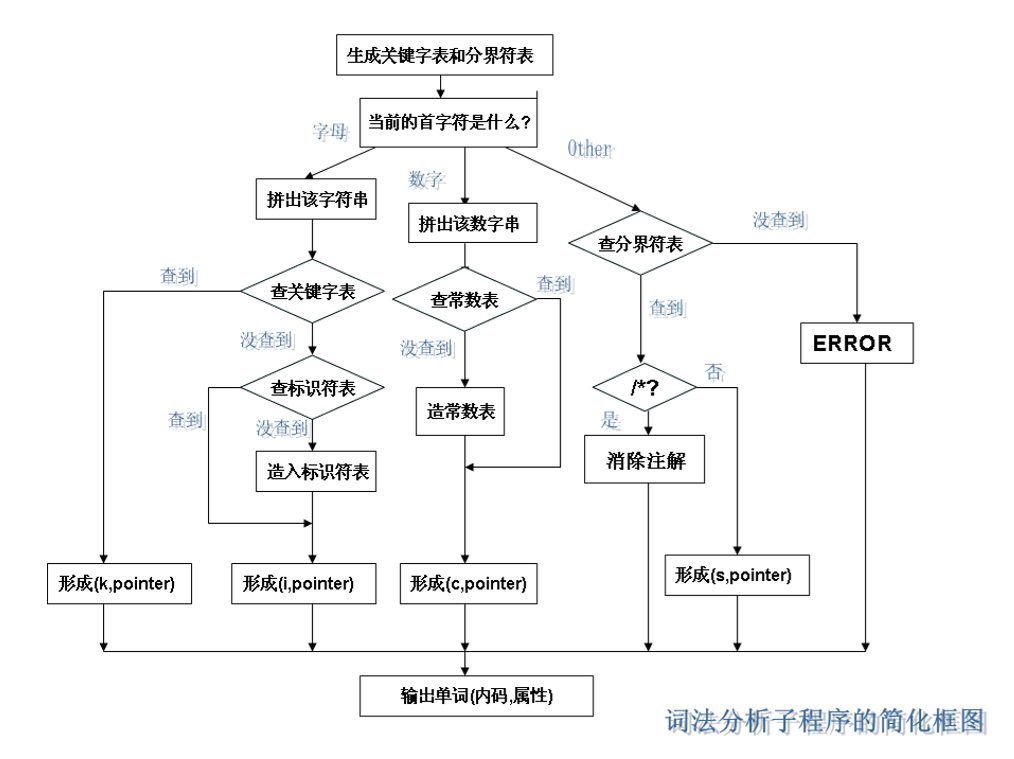
api.py主要负责pyqt界面的设计，整个程序的启动，结果可视化等功能。

总的来说，main.py里面放的是算法API函数，api.py里面放的是前端设计和API调用与处理函数。

## 四、详细的算法描述



**图2. 分析程序流程图**



**图3. 词法分析子程序的简化框图**

**补充：**

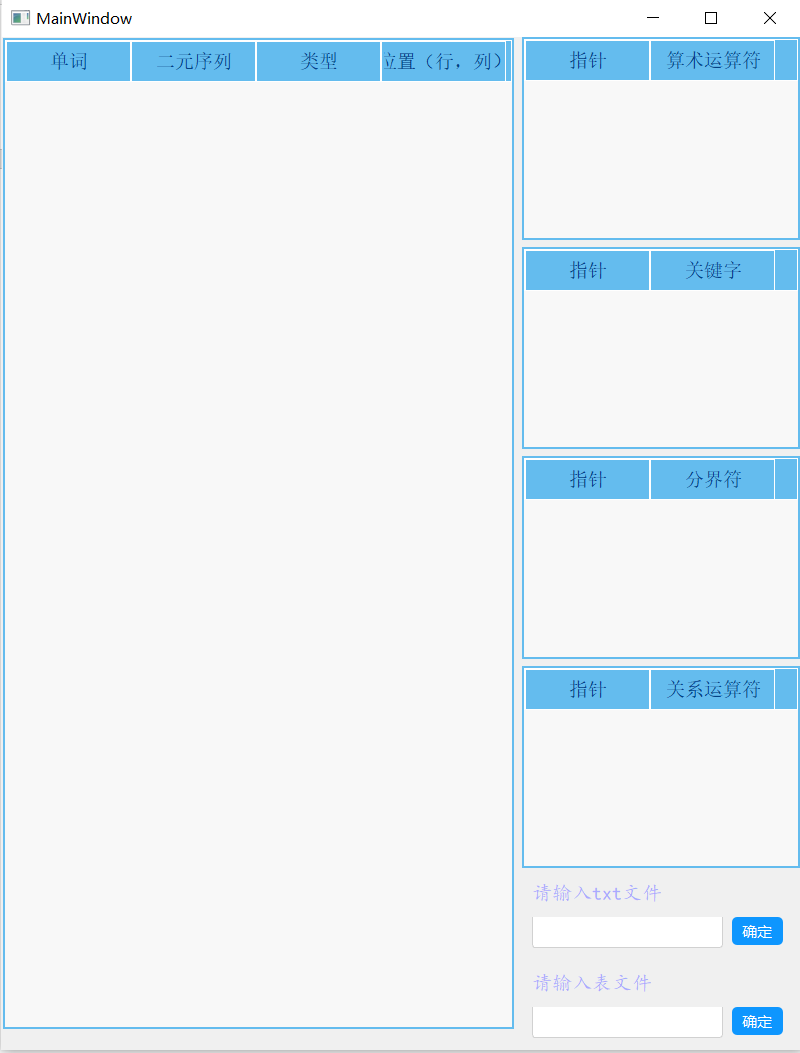
本程序识别过程如下：

1. 首先将各个字串分开，然后分开进行处理。
2. 先利用python封装的包初步进行识别，这些包可以判断一些数字和字母。
3. 对于一些复杂的，如数字与字母结合、下划线等等，一位一位进行判断。
4. 对于“//”，我们认为他是注释。
5. 对于浮点数，我们判断数字串里面有多少个小数点，一个就是浮点数，多个就报错。
6. 对于“++”，“--”，我们判断是否存在连续的加号或减号。

## 五、测试方法和测试结果

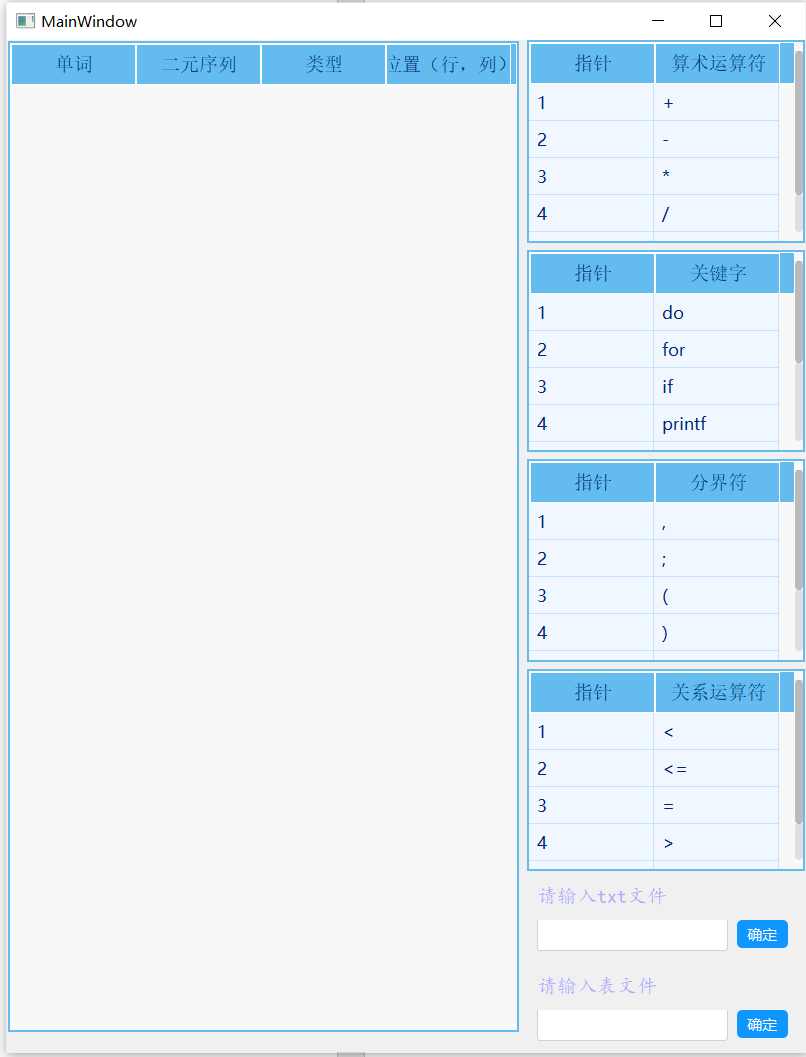
本次使用VsCode + python3.8开发。

Python api.py即可



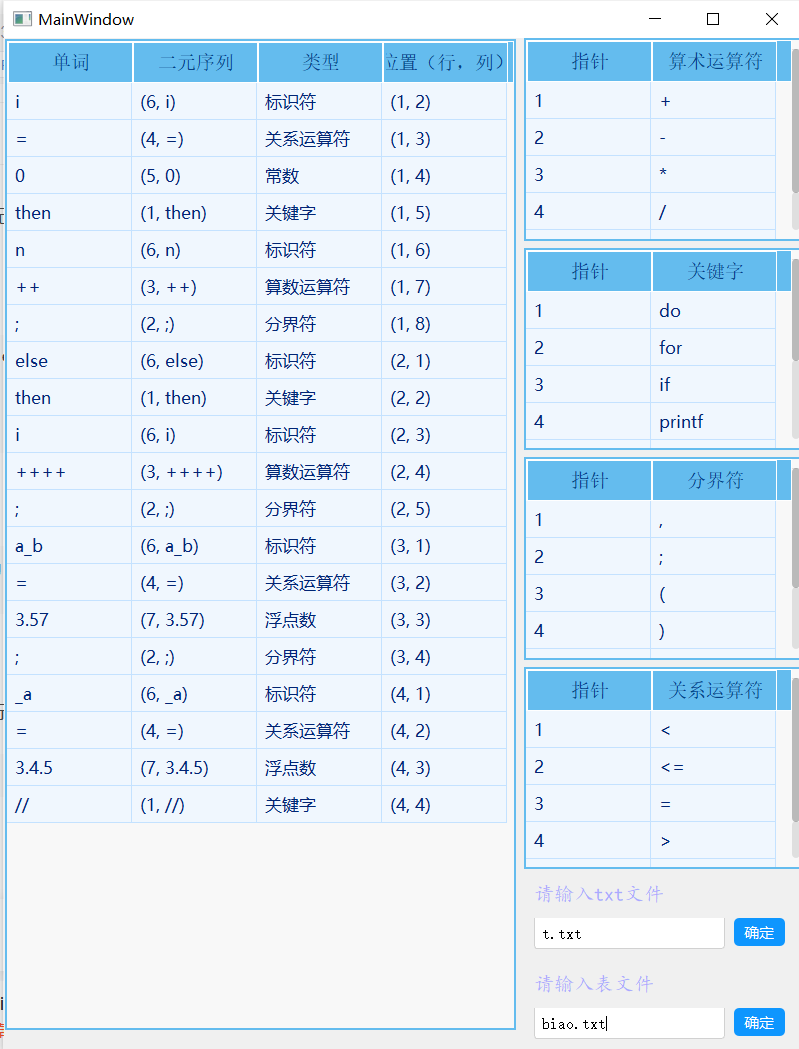
**图4. 程序主界面**

先在下方输入需要读取的表文件，初始化分析表：



**图5. 加载分析表**

随后输入字符文件：



**图6. 分析结果图**

即可得到分析结果。

## 六、关键源码

以下给出除前端设计外与算法相关代码

**1.main.py：**

|  |
| --- |
| import os  # 关键字  k = ["do", "for", "if", "printf", "scanf", "then", "while", "end"]  # 分界符  s = [",", ";", "(", ")", "[", "]", "//"]  # 算数运算符  aop = ["+", "-", "\*", "/", "++", "--"]  # 关系运算符  rop = ["<", "<=", "=", ">", ">=", "<>"]  # 输出  comeout = ["Error", "关键字", "分界符", "算数运算符", "关系运算符", "常数", "标识符", "浮点数"]  # 结果  result = []  def judge(word):  # 常数  if word.isdigit():  return 5  if word[0].isdigit():  for i in range(len(word)):  if word[i].isdigit() is False:  if word[i].isdigit() is False and (word[i + 1].isdigit() is False or i == len(word)):  return 6  else:  return 7  if word[0] == '-':  Flag = True  for i in word:  if i.isdigit() is False and i != '.' and Flag is False:  return 0  if i == '.':  Flag = False  return 5  if word[0] == '/' and word[1] == '/':  return 1  if word[0] == '+' and word[1] == '+':  return 3  if word[0].isalpha() or word[0] == "\_":  for i in k:  if word == i:  return 1  return 6  for i in s:  if word == i:  return 2  for i in aop:  if word == i:  return 3  for i in rop:  if word == i:  return 4  return 0  def createbiao(filename):  if not os.path.exists(filename):  print("文件打开失败！")  return  else:  b = open(filename)  global k, s, aop, rop  k = []  s = []  aop = []  rop = []  choice = 0  for line in b:  for j in comeout:  if j == b:  choice = comeout.index(j)  continue  if choice == 1:  k = line.split(" ")  elif choice == 2:  s = line.split(" ")  elif choice == 3:  aop = line.split(" ")  elif choice == 4:  rop = line.split(" ")  def main(filename):  global result  row = 1  col = 1  getstring = ""  result = []  if not os.path.exists(filename):  print("文件打开失败！")  return  else:  allstring = open(filename)  for line in allstring:  for i in line:  if i == ' ' or i == '\n':  num = judge(getstring)  if num:  second = "({0}, {1})".format(str(num), getstring)  else:  second = "Error"  go = [getstring, second, comeout[num], "({0}, {1})".format(str(row), str(col))]  result.append(go)  col += 1  getstring = ""  else:  getstring += i  col = 1  row += 1  allstring.close() |

**2.api.py：**

## 七、实验总结

试验由自己从零做起，此次试验体验到了从编译器角度怎样逐步分析高级语言的分析过程，并在一定程度上理解到编译过程的繁琐过程以及诸多考虑事项。

整个程序对于行列的分析以及其判别是浑然一体的，同时进行，并用二位数组进行数据的存储，使用stl提高工作效率。

## 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：**

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

**文法2：**

1. E→TE′
2. E′→+TE′ | ε
3. T→FT′
4. T′→\*FT′ | ε
5. F→(E) | i
6. **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 |
| gramma\_analysis(string str) | 对于输入串的赋值 | 无 | 构造函数，对部分变量进行赋值及对整个过程的函数调用 |
| bool judge(vector<string> str1, string str2) | 在string类型数组，string类型字符串str2 | 若存在则返回true若查找失败则返回false | 判断数组str1中是否存在str2 |
| bool judge\_nt(string str3) | 字符串str3 | 若str3是终结符则返回true若是非终结符则返回false | 判断字符串str3是终结符或非终结符 |
| void sol\_vn\_vt() | 无 | 无 | 对产生式进行分析，并提取所有的非终结符与非终结符并将其置于两个字符串向量中 |
| void sol\_first() | 无 | 无 | 根据文法进行分析对其所有非终结符first集合进行求解 |
| void sol\_follow() | 无 | 无 | 根据文法进行分析对其所有非终结符follow集合进行求解 |
| void sol\_analysis\_table() | 无 | 无 | 根据first，follow集合以及文法产生式进行ll(1)文法分析表构造 |
| string str\_add(string str1, string str2) | 两个string类型的字符串 | string类型字符串 | 将str1中与str2不重复的元素加入str2并返回 |
| void OnBnClickedButton2() | 无 | 无 | 该函数位于lexical\_analysisDlg.cpp为主对话框中的一个分析函数，对输入串根据分析表进行分析并在对话框中输出 |
| void OnBnClickedButton3() | 无 | 无 | button3与button4分别为first follow 集合表与分析表的对话框显示调用，其初始化由其初始化函数完成，在此不做过多描述 |

注：由于该程序基于MFC，对话框形式进行输入输出并可视化显示，有部分函数为内嵌函数，部分MFC的函数自行修改并进行变量的外部连接，不过多在原有框架函数上的描述，只进行主要分析的ll1.h的主要分析描述。

void OnBnClickedButton2()

bool judge(vector<string> str1, string str2)

gramma\_analysis(string str)

bool judge\_nt(string str3)

void sol\_vn\_vt()

void sol\_first()

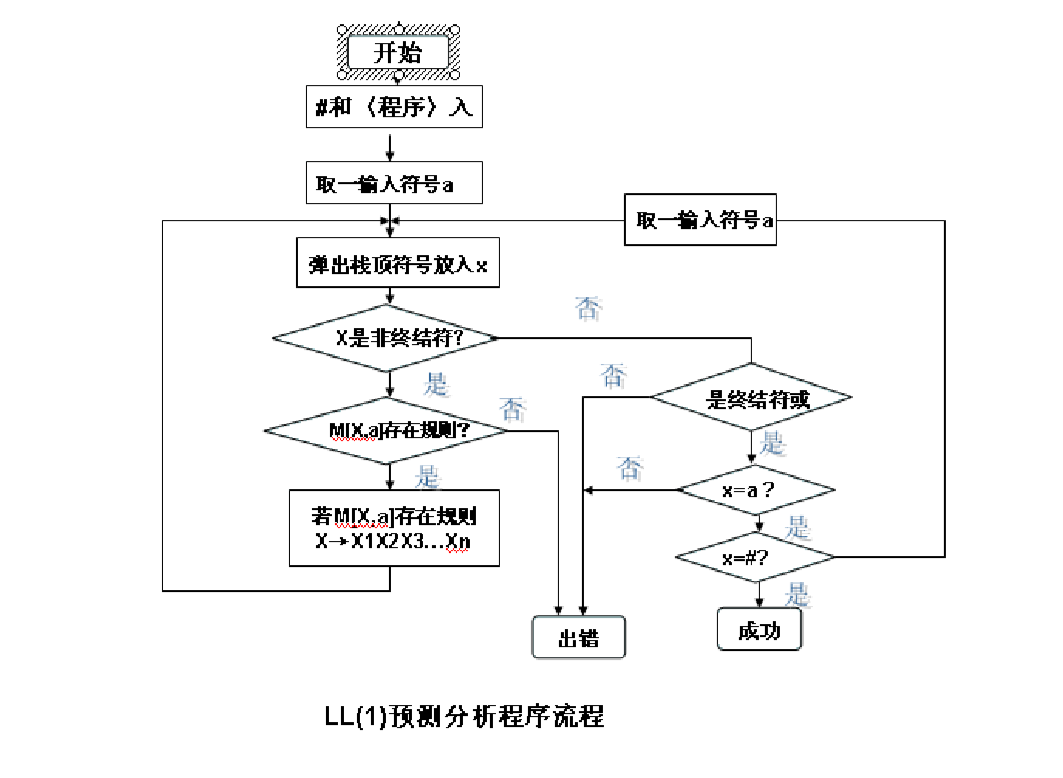
void sol\_follow()

string str\_add(string str1, string str2)

void sol\_analysis\_table()

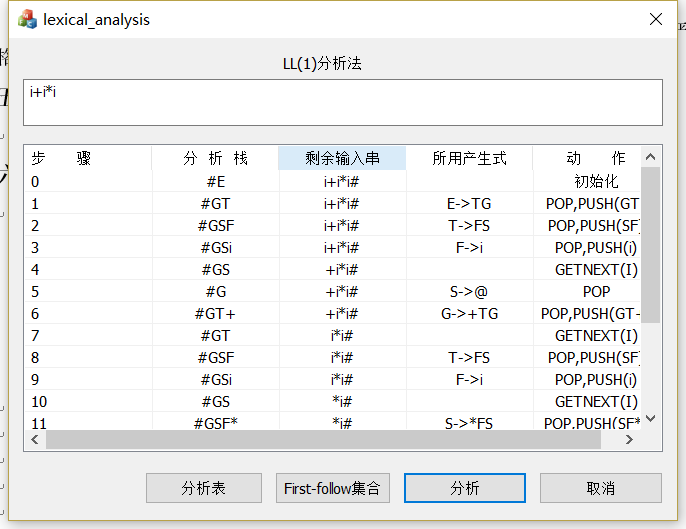
注：由上注释描述，在此框架内的OnBnClickedButton3()、OnBnClickedButton4()、OnInitDialog()等函数不在做过多描述。上述左侧函数箭头为调用顺序，右侧函数与左侧函数的箭头为调用关系。另外两个子对话框的初始化分别在C\_show\_f与C\_showtable中进行。

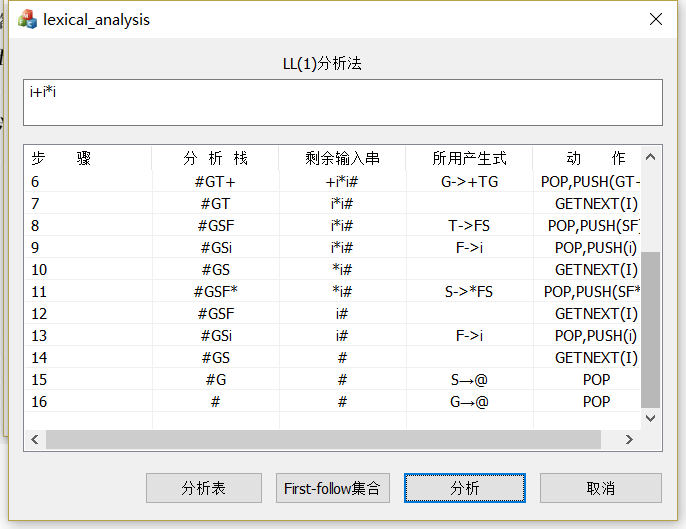
1. **详细的算法描述**



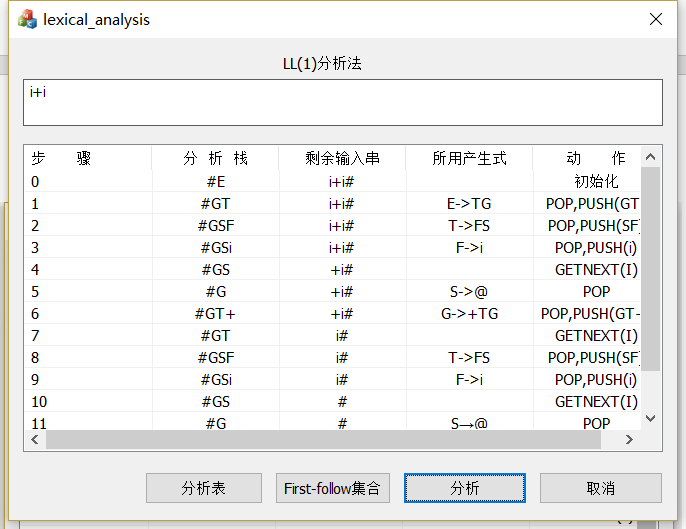
由文法构造分析表的流程由四、中给出顺序，算法实现在源码中给出，并严格依据教学概念进行。

1. **给出软件的测试方法和测试结果。**

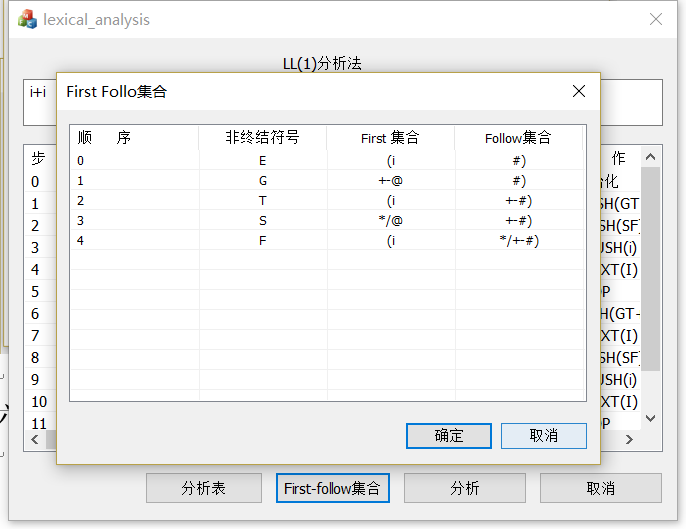




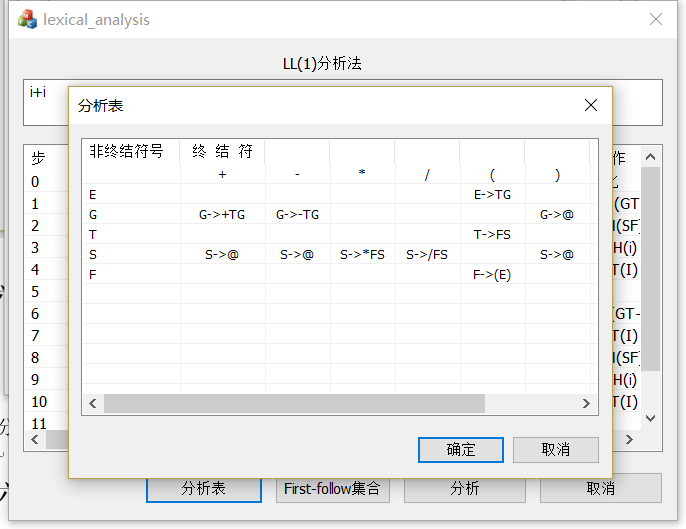
可根据不同输入串做出不同分析并清空列表后输出：



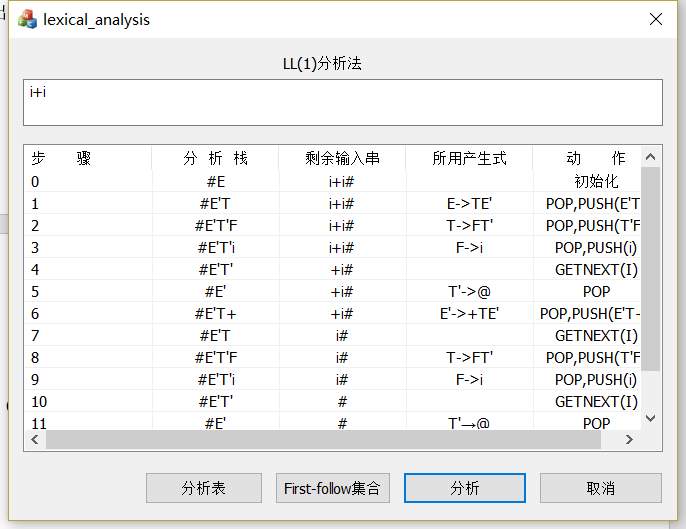
First、follow输出：



分析表输出：



不同文法测试只给出个别示例在此不过多描述：



1. **实验总结**
2. 此次试验全程由个人完成，由于文法为满足课堂所讲左递归消除后的符号"'"的判别及使用，该单引号符号与文法中非终结符进行连用，判断及提取难度较大，每次判断由后至前或先判断第二位再判断第一位进行，每次分析若碰到该符号需要多进行一步的判断并多一步分析，工作量较大。
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，由于工作量较大，使用了部分库函数进行辅助。
4. 该程序实现时由理论到实践的步骤较多较为复杂，进行了较多次迭代才完成所有功能，下次进行较为复杂的逻辑设计时，应该先用纸笔进行逻辑的完善再进行代码编写会提高较大速率。
5. 该程序使用了MFC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，在变量的外部连接以及控件的声明初始化调用方面花费较多时间，对此未来学习更新的框架以及效率更高的方式。

## 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”。
6. **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 |
| lr1\_analysis() | 无 | 无 | 构造函数，对文法进行分析，调用该类中的部分函数进行分析并构造分析表 |
| bool judge(vector<string> str1, string str2) | 在string类型数组，string类型字符串str2 | 若存在则返回true若查找失败则返回false | 判断数组str1中是否存在str2 |
| bool judge\_nt(string str3) | 字符串str3 | 若str3是终结符则返回true若是非终结符则返回false | 判断字符串str3是终结符或非终结符 |
| void sol\_vn\_vt() | 无 | 无 | 对产生式进行分析，并提取所有的非终结符与非终结符并将其置于两个字符串向量中 |
| void sol\_first() | 无 | 无 | 根据文法进行分析对其所有非终结符first集合进行求解 |
| void sol\_follow() | 无 | 无 | 根据文法进行分析对其所有非终结符follow集合进行求解 |
| string str\_add(string str1, string str2) | 两个string类型的字符串 | string类型字符串 | 将str1中与str2不重复的元素加入str2并返回 |
| void sol\_tuoguang() | 无 | 无 | 对文法分析后并进行将文法进行拓广，并将文法按顺序将其所有项目写入二维数组 |
| void sol\_closure() | 无 | 无 | 对拓广后的文法提取所有有效识别活前缀的的式子作为项目集，对所有项目集进行闭包构造。 |
| void action\_goto() | 无 | 无 | 对项目集闭包构造后，根据族集构造器DFA，将构造好的DFA按照定义构造分析表，此程序中将Action与Goto放置于一个数据表中 |

注：所有变量的赋值更改全部为类内函数完成，因为其为类内数据成员，故不需特别声明，同试验二，OnBnClickedButton1()等函数不做过累述。

void OnBnClickedButton1()

bool judge(vector<string> str1, string str2)

lr1\_analysis()

bool judge\_nt(string str3)

void sol\_vn\_vt()

void sol\_first()

void sol\_follow()

string str\_add(string str1, string str2)

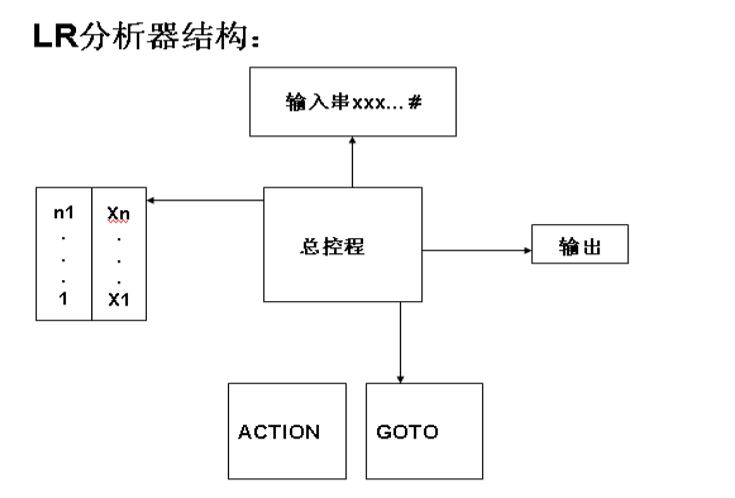
void sol\_tuoguang()

void sol\_closure()

void action\_goto()

注：分析表构造的逐步分析过程由OnBnClickedButton1()完成在表格控件输出。

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。

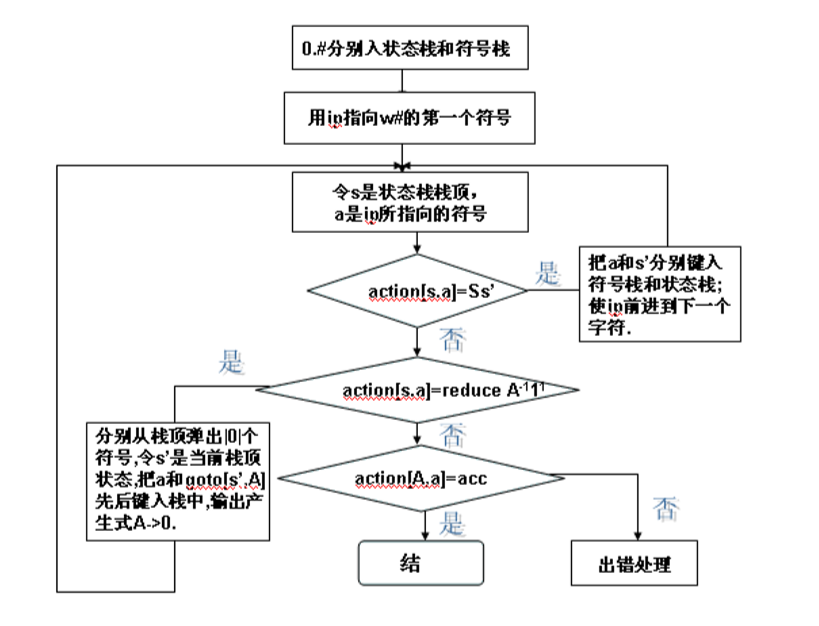
(3)接受 acc:

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。

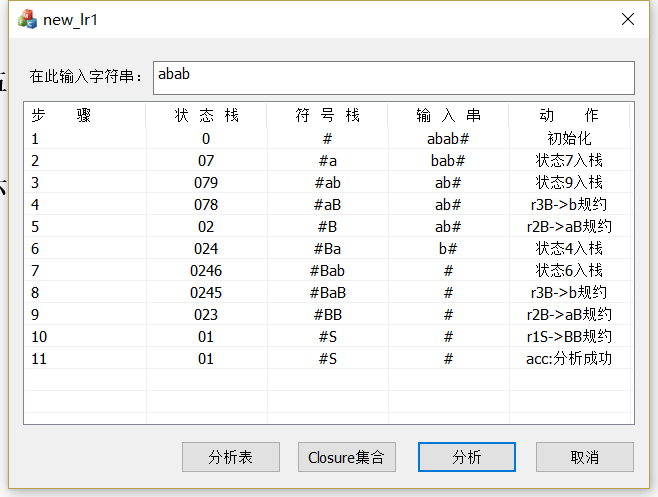
(4)报错:

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

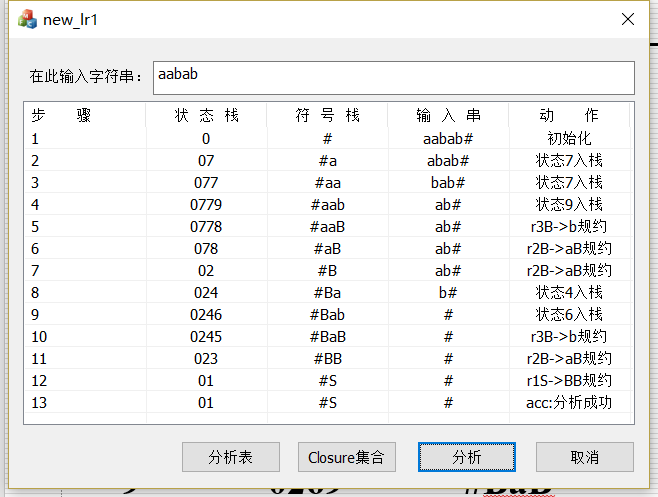
入端不是该文法能接受的符号串。



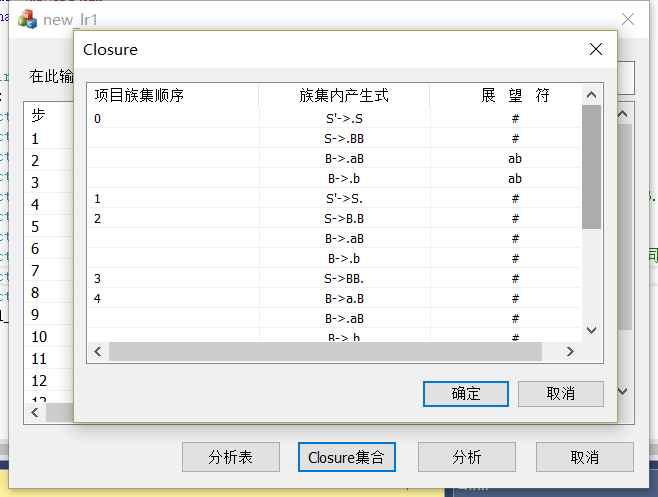
1. **给出软件的测试方法和测试结果。**

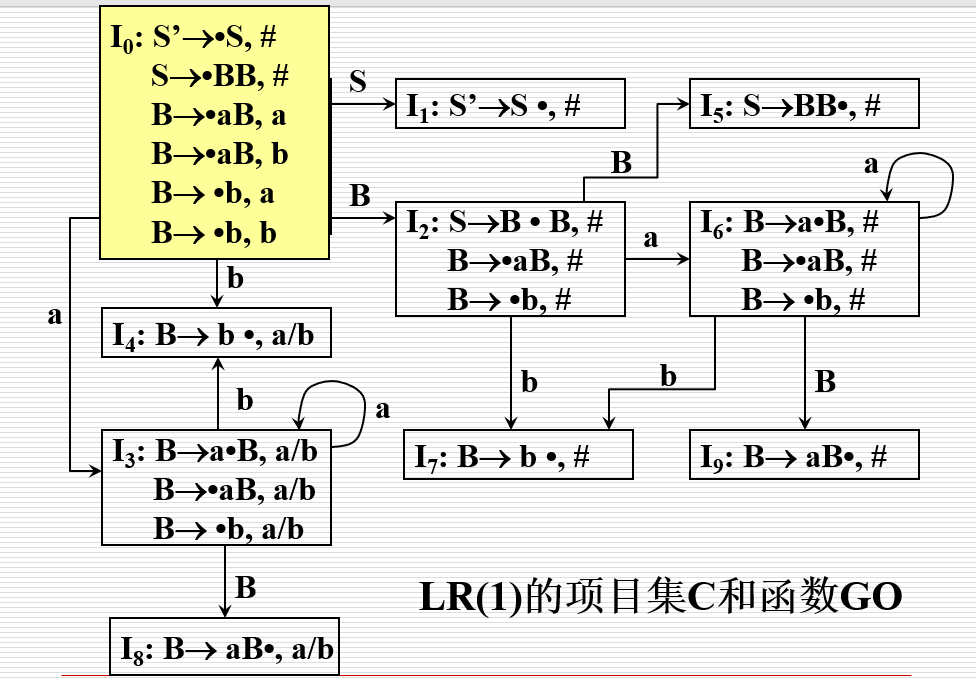


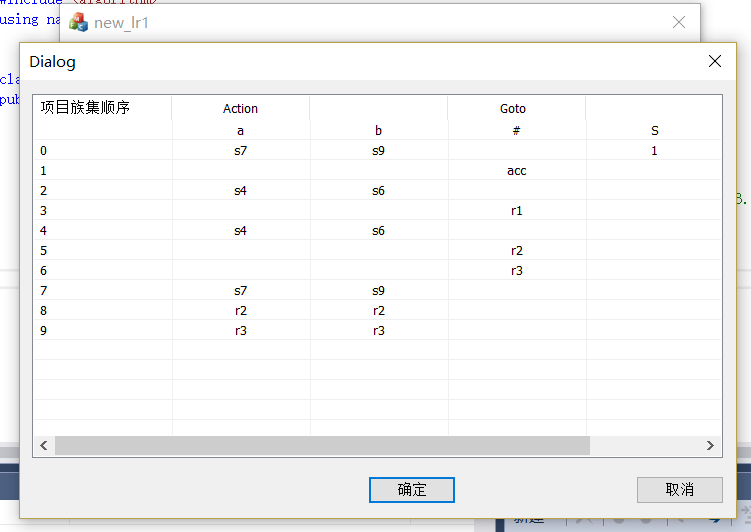


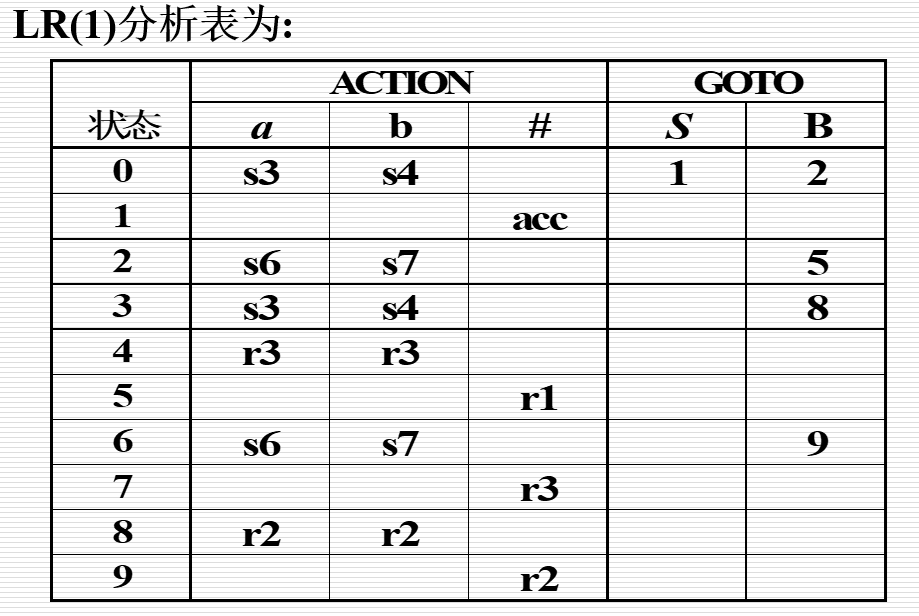












注：由于分析过程的程序化状态顺序略有不同但都一一对应。

1. **实验总结**
2. 此次试验由个人完成，对于first和follow集合求取由试验进行修改后直接复用。
3. 由于前期的设计失误，导致vector<vector <string>> tuoguang的数据与函数中的冲突，又浪费时间从新构造了vector<vector <string>> tg2弥补失误，设计千算万算还是失算了。
4. 关于first follow集合求取，文法递归的改变需要从新设计，工作量较大。
5. 本次试验所有求取严格按照课本定义进行，构造较为复杂，使用了大量库函数后代码量还是比较大，若增加迭代次数可减少行数，故不累述。
6. 此次试验由于涉及步骤较多，所以变量较多，各种数据类型的转化较多，尤其是涉及到MFC特有的变量类型。
7. 两个子对话框由C\_closure与C\_table完成。