合肥工業大學
HEFEI UNIVERSITY OF TECHNOLOGY

编译原理实验报告

学生姓名 ： 林天岳

学 号 ： 2017217893

班 级 ： 计算机科学与技术 2017-5

完成日期 ： 2019 年 10 月 22 日

实验 1：词法分析设计

1.数据结构及算法描述

```
1.    String alphabet = "ABCDEFGHIGKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";//字母
2.      String number = "0123456789";//数字
3.      String keyword[] = {"auto","break","case","char","const ","continue","default","do ",
4.              "double ","else ","enum ","extern","float","for","goto","if","int",
5.              "long","register","return","short","signed","sizeof","static","struct","switch",
6.              "typedef","unsigned","union","void","volatile","while"};//关键字
7.      String operator[] ={"<<=",">>=","&&","||","<=","|=","*=","^=","==","++","--","/=","-
    =","+=","%=","!=",">=","[","]","!","%","(",")","*","+",",","-","/",";","<","=",">"};//运算
    符
8.      String arithmeticOperator[] = {"++","--","+","-","*","/","%"};//算术运算符
9.      String relationalOperator[] = {"<=","<",">=",">","==","!="};//关系运算符
10.     String logicalOperator[] = {"&&","||","!"};//逻辑运算符
11.     String delimiter[] = {";",",","(",")","[","]"};//分界符
12.     String assignmentOperator[] ={"=","+=","-
    =","*=","/=","%=","<<=",">>=","%=","^=","|="};//赋值运算符
13.
14.     Map<String,String> opS;//<单个运算符,种类名>
15.     Map<String,String[]> KindtoArrary;//<种类名,对应的运算符数组>
16.
17.     List<Result> result = new ArrayList<>();//结果 保存后显示为表格
```

```
1.  GUI 包含一个 Solution
2.  分析时 在 textArea 中输入需要的分析的代码
3.  或者直接打开文件读取到 textArea
4.
5.  分析 则使用 Solution.Solve 返回分析结果
6.  显示在界面上
7.  Solution 包含一个 Analyzer 分析器
8.  调用 Solve 方法  传入 String 数组 返回分析结果
9.  for(String line:传入的 String 数组){
10.     result.addAll(Analyzer.LineAnalyse(line));Analyzer.LineAnalyse(line)
11. }
12. return result
13. LineAnalyse 方法
14. if(当前分析的是 null,//,\n,或者 Length == 0){
15.     则直接结束
16. }
17. else {
18.     if(字母表含有当前头部的 string){
19.         while(是字母或者是数字){
20.             继续取出之后的部分
21.         }
22.         得到了一个 String
23.         if(单词是关键字){
24.             标记为 关键字
25.         }
26.         else{
27.             标记为 标识符
28.         }
29.     }
30.     else if(数字表含有当前头部的的 string){
31.         while(是数字或者小数点){
32.             继续取出之后的部分
33.         }
34.         if(数字之后直接追加字母){
35.             标记错误
36.             brerak
37.         }
38.         标记为 常数//常数的标记使用一个静态方法调用方法返回当前的数目+1 ERROR 使用同样的方法编号
39.     }
40.     else{
41.         if(匹配到了符号){//运算符经过按照长度排序 确保长度较长的先匹配到 比如 ++ 会优先于+匹配
42.             标记 运算符
43.         }
44.         else{
45.             标记 错误
46.         }
47.     }
48.     递归处理之后的 String
49.     resturn  result.addAll(递归的结果);
50. }
```

2.算法流程图

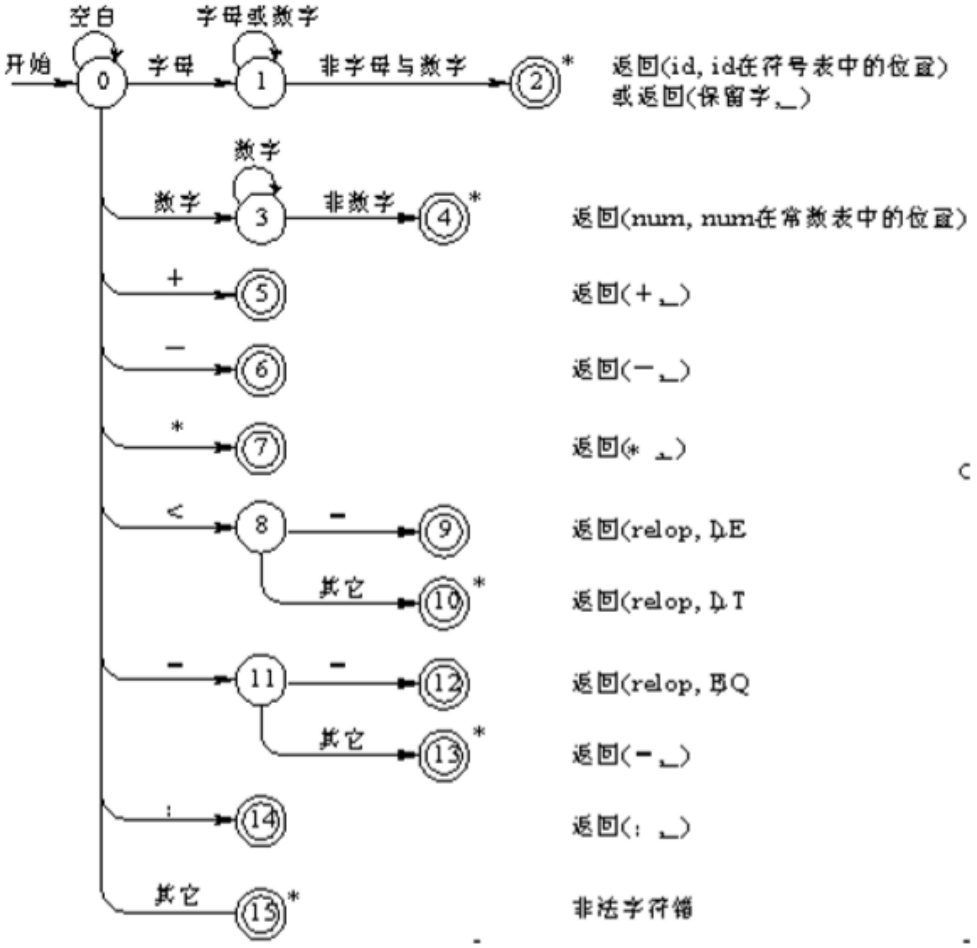

空白
字母或数字
开始
0
字母
1
非字母与数字
2
*
返回(id, id在符号表中的位置)
或返回(保留字,_)
数字
数字
3
非数字
4
*
返回(num, num在常数表中的位置)
+
5
返回(+,_)
-
6
返回(-,_)
*
7
返回(* ,_)
<
8
=
9
返回(relop, LE
其它
10
*
返回(relop, LT
=
11
=
12
返回(relop, EQ
其它
13
*
返回(=,_)
:
14
返回(: ,_)
其它
15
*
非法字符错

表1 关键字表

指针	关键字
0	do
1	end
2	for
3	if
4	printf
5	scanf
6	then
7	while

表2 分界符表

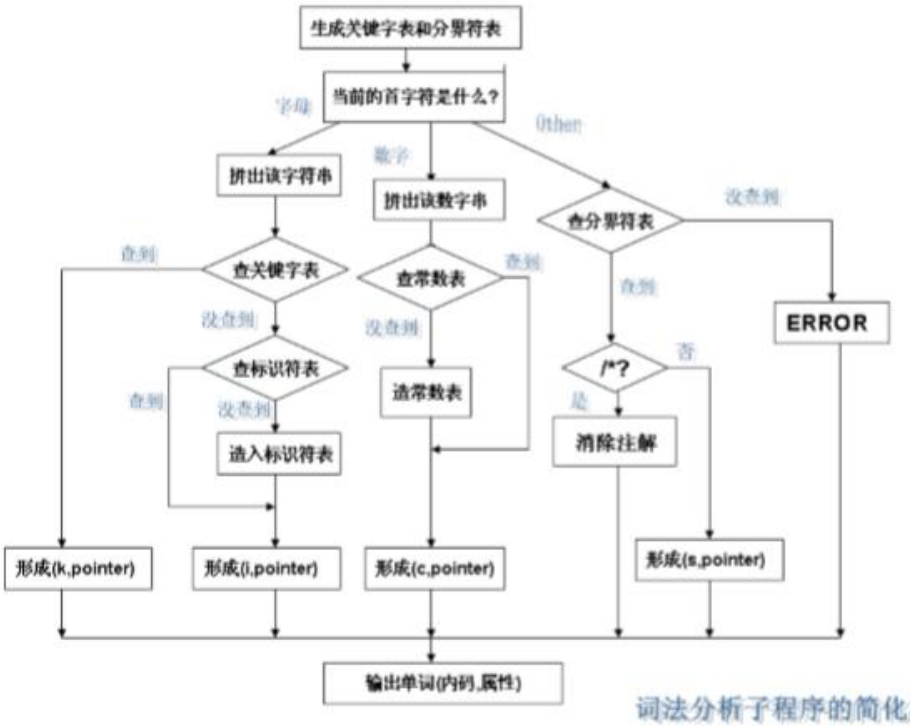
指针	分界符
0	,
1	;
2	(
3	)
4	[
5	]

表3 算术运算符

i值	算术运算符
10H	+
11H	-
20H	*
21H	/

表4 关系运算符

i值	关系运算符
00H	<
01H	<=
02H	=
03H	>
04H	>=
05H	<>



词法分析子程序的简化框图

3.源码及测试结果

Main.java：

```
package 实验一___词法分析设计;


public class Main{
    public static void main(String[] args) {
        Windows windows = new Windows();
    }
}
```

Solution.java

```
package 实验一___词法分析设计;

import java.util.*;

class Solution{
    Analyzer ana = new Analyzer();
    public List<Result> Solve(String[] lines) {
        List<Result> res = new ArrayList<>();
        if(lines==null|| lines.length==0)
            return res;
        List<String> text = new ArrayList<>();
        for(String line:lines){
            line = line.replaceAll("\t"," ");
            if(line.length()>0)
                text.add(line);
        }
        int l = 1;
        for(String str:text){
            if(str.length()>2 && str.substring(0,2).equals("//"))
                continue;
            ana.result.clear();
            res.addAll(ana.LineAnalyse(str+"\n",l,1));
            l++;
        }
        return res;
    }
    public void manullySetKP(String k[] , String p[] ){
        ana.setKP(k,p);
    }
}
class Analyzer{//在这里是用 C 的标准了
    String alphabet = "ABCDEFGHIGKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";//字母
    String number = "0123456789";//数字

    String keyword[] = {"auto","break","case","char","const ","continue","default","do ",
            "double ","else ","enum ","extern","float","for","goto","if","int",
            "long","register","return","short","signed","sizeof","static","struct","switch"
,
            "typedef","unsigned","union","void","volatile","while"};//关键字
    String operator[] ={"<<=",">>=","&&","||","<=","|=","*=","^=","==","++","--","/=","-
="","+=","%=","!=",">=","[","]","!","%","(",")","*","+",",","-","/",";","<","=",">"};//运算
符
    String arithmeticOperator[] = {"++","--","+","-","*","/","%"};//算术运算符
    String relationalOperator[] = {"<=","<",">=",">","==","!="};//关系运算符
    String logicalOperator[] = {"&&","||","!"};//逻辑运算符
    String delimiter[] = {";",",","(",")","[","]"};//分界符
    String assignmentOperator[] ={"=","+=","-
=","*=","/=","%=","<<=",">>=","%=","^=","|="};//赋值运算符

    Map<String,String> opS;//<单个运算符,种类名>
    Map<String,String[]> KindtoArrary;//<种类名,对应的运算符数组>

    List<Result> result = new ArrayList<>();

    public Analyzer(){
        KindtoArrary = new HashMap<>();
        KindtoArrary.put("算术运算符",arithmeticOperator);
        KindtoArrary.put("关系运算符",relationalOperator);
        KindtoArrary.put("逻辑运算符",logicalOperator);
        KindtoArrary.put("分界符",delimiter);
        KindtoArrary.put("赋值运算符",assignmentOperator);
```

```
58.         opS = new HashMap<>();
59.         KindtoArrary.keySet().forEach(KindStr->Arrays.asList(KindtoArrary.get(KindStr)).for
    Each(str->opS.put(str,KindStr)));
60.     }
61.
62.     public void setKP(String k[] , String p[] ){
63.         this.keyword = k;
64.         this.operator = p;
65.     }
66.     public List<Result> LineAnalyse(String line,int L,int C){//当前行 行数 列数
67.         //System.out.print("当前分析 :"+line+" ");
68.         if(line == null || line.length()==0 || line.equals("\n") || (line.length()>=2 && li
    ne.substring(0,2).equals("//"))){
69.             return null;//行空 长度为0 回车 注释 行结束
70.         }
71.         if(line.substring(0,1).equals(" ")){//是空格 跳过当前单词
72.             LineAnalyse(line.substring(1),L,C);
73.             return result;
74.         }
75.         Result res = new Result();
76.         String head = line.substring(0,1);
77.         int i = 0;
78.         if(alphabet.contains(head)){//匹配到字母
79.             while(i!=line.length() &&  (alphabet+number).contains(line.substring(i,i+1))){

80.                 i++;
81.             }
82.             String wordGet = line.substring(0,i);
83.             Boolean ketWordMatch = false;
84.             int count = 0;
85.             for(String str:keyword){
86.                 if(wordGet.equals(str)){//是关键字
87.                     ketWordMatch = true;
88.                     res.setKind("关键字");
89.                     res.setSequence("("+count+","+wordGet+")");
90.                     break;
91.                 }
92.                 count++;
93.             }
94.             if(!ketWordMatch){//是标识符
95.                 res.setKind("标识符");
96.                 res.setSequence("("+DataList.getID(wordGet)+","+ wordGet+")");
97.             }
98.             res.setWord(wordGet);
99.         }
100.        else if(number.contains(head)){//匹配到数字考虑小数，但小数不会以"."开头
101.            while (i!=line.length() && (number+".").contains(line.substring(i,i+1))){
102.                i++;
103.            }
104.            if(alphabet.contains(line.substring(i,i+1))){//数字之后直接追加字母  非法输入
105.                while(i!=line.length() &&  (alphabet+number).contains(line.substring(i,i+1)
    )){
106.                    i++;
107.                }
108.                res.setWord(line.substring(0,i+1));
109.                res.setKind("ERROR");
110.                res.setSequence("ERROR"+DataList.getERROR(line.substring(0,i+1)));
111.            }
112.            else{
113.                String number = line.substring(0,i);
114.                res.setWord(number);
115.                res.setKind("常数");
116.                res.setSequence("("+DataList.getCI(line.substring(0,i))+","+ line.substring
    (0,i)+")");
117.            }
118.
119.        }
120.        else{
121.            Boolean match = false;
122.            for(String str:operator){//用运算符来匹配而不是去匹配运算符号  避免 ++ 匹配出 +*2
123.                if(str.length() > line.length())
124.                    continue;//符号是在尾部 且不会匹配成功 则直接跳过
125.                //System.out.println(str+"匹配"+line.substring(0,str.length()));
126.                if(str.equals( line.substring(0,str.length()) )){//是运算符
127.                    res.setWord(str);
128.                    res.setKind(opS.get(str));
129.                    int count = Arrays.asList(KindtoArrary.get(opS.get(str))).indexOf(str);

130.                    res.setSequence("("+count+","+str+")");
131.                    match = true;
132.                    i+=str.length();
133.                    break;
134.                }
135.            }
136.            if(!match){//没有匹配到
137.                res.setWord(line.substring(0,1));
138.                res.setKind("ERROR");
```

```
                res.setSequence("ERROR"+DataList.getERROR(line.substring(0,1)));
                i++;
            }
        }
        line = line.substring(i);
        res.setLocation("("+L+","+C+")");
        result.add(res);
        LineAnalyse(line,L,++C);
        return result;
    }
}
class Result{
    private String word;//单词
    private String binarySequence;//二原序列
    private String kind;//类型
    private String location;//位置
    public Result(){
        this.word = "Null";
        this.binarySequence = "Null";
        this.kind = "Null";
        this.location = "Null";
    }
    public void setWord(String word){
        this.word = word;
    }
    public void setSequence(String Sequence){
        this.binarySequence = Sequence;
    }
    public void setKind(String kind){
        this.kind = kind;
    }
    public void setLocation(String location){
        this.location = location;
    }
    public String[] toStringArrary(){
        String stringS[] = {"  "+word,"  "+binarySequence,"  "+kind,"  "+location};
        return stringS;
    }
    @Override
    public String toString(){
        String strs[] = {word,binarySequence,kind,location};
        StringBuffer toString = new StringBuffer();
        for(String str:strs){
            str =  String.format("%-20s", str);
            toString.append(str);
        }
        return toString.toString();
    }
}
class DataList{
    static List<String> id = new ArrayList<>(),ci = new ArrayList<>(),ERROR = new ArrayList
<>();//标识符 常数
    public static int getID(String str){//获取标识符位置  存在则返回地址 不存在则存入 返回最后位
置
        if(id.contains(str)){
            return id.indexOf(str);
        }
        else{
            id.add(str);
            return id.size()-1;
        }
    }
    public static int getCI(String str){//获取常数位置
        if(ci.contains(str)){
            return ci.indexOf(str);
        }
        else{
            ci.add(str);
            return ci.size()-1;
        }
    }
    public static int getERROR(String str){//获取错误代码
        if(ERROR.contains(str)){
            return ERROR.indexOf(str);
        }
        else{
            ERROR.add(str);
            return ERROR.size()-1;
        }
    }
}
```

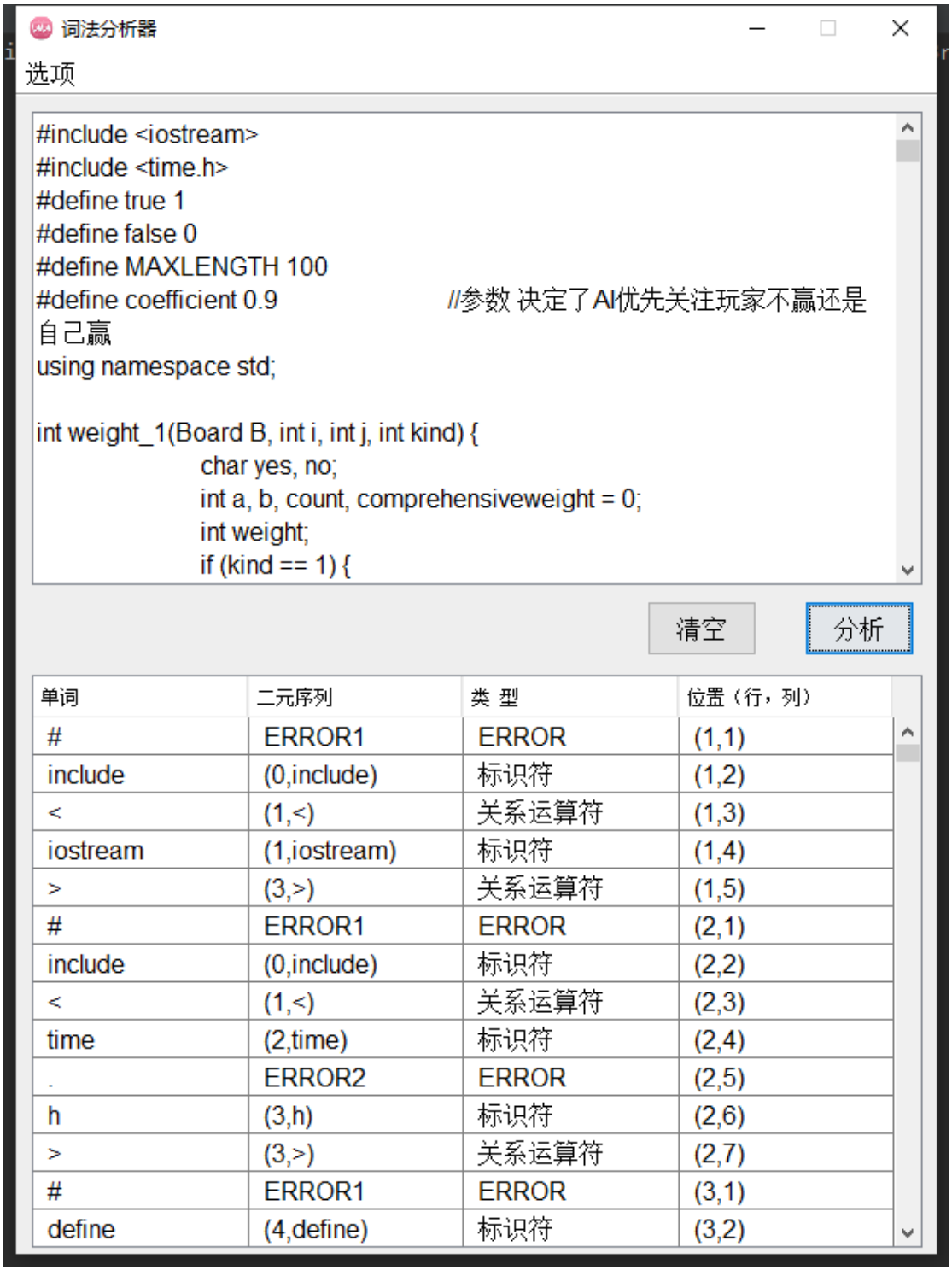
GUI.java

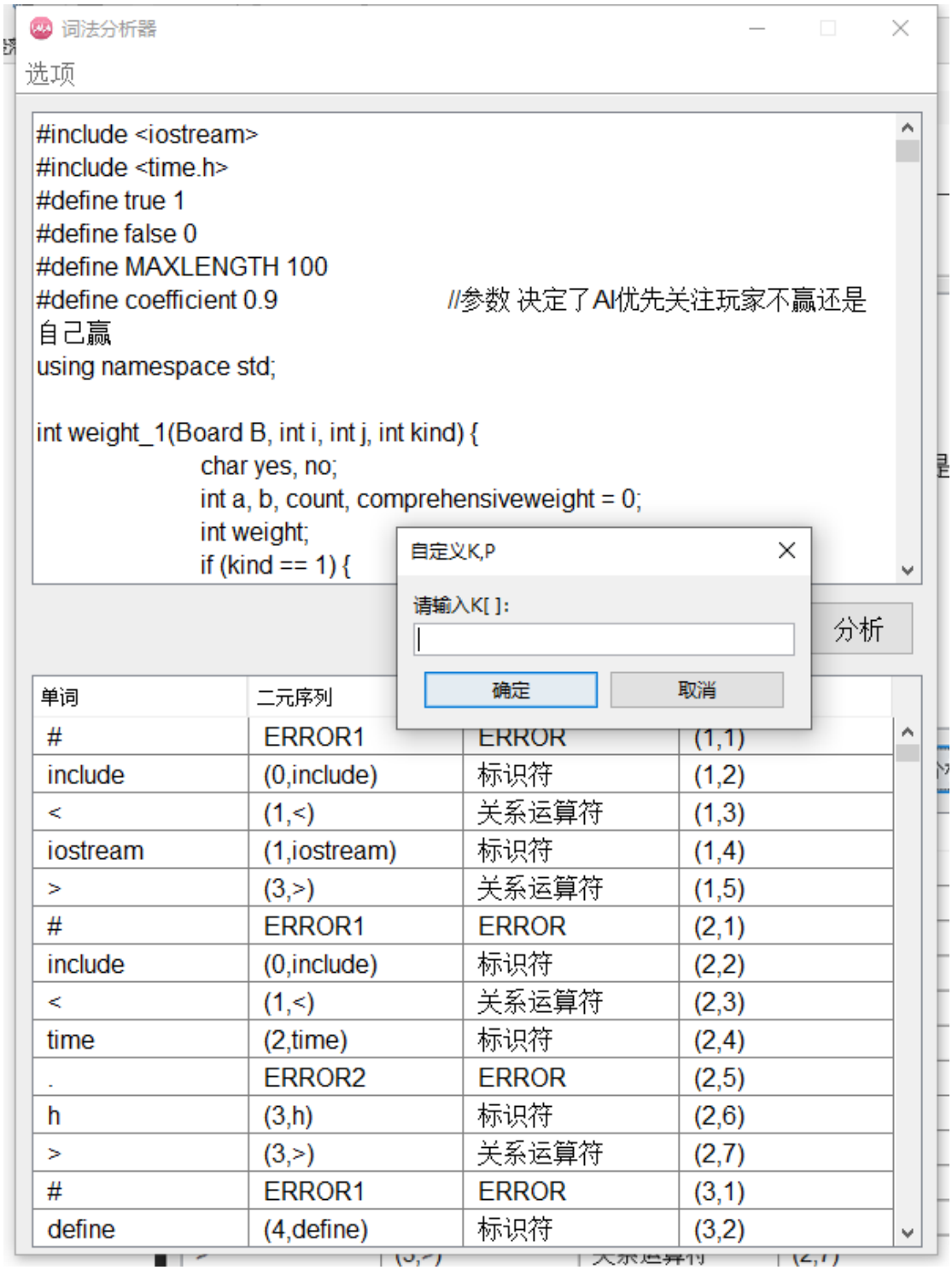
```java
package 实验一___词法分析设计;

import javax.swing.*;
import javax.swing.table.AbstractTableModel;
import java.awt.*;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.List;
import java.util.Vector;

class Windows extends JFrame{
    JMenuBar bar;
    JMenu menu;
    JMenuItem file;
    JMenuItem manuallySet;
    JMenuItem exit;
    JTextArea TA;
    JButton clear;
    JButton analyse;
    String[] text;
    JTable table;
    Vector<String[]> vecRes = new Vector<>();
    TableDataModel tableDataModel;
    JScrollPane restablescrollPane;
    Solution sl = new Solution();
    public Windows(){
        try{
            setIconImage(new ImageIcon("bilibili.PNG").getImage());
            Font f = new Font("Yahei Consolas Hybrid",Font.PLAIN,16);
            String   names[]={ "MenuBar","Menu","MenuItem", "TextArea", "Button", "ScrollPane", "Table"};
            for (String item : names) {
                UIManager.put(item+ ".font",f);
            }
            UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");

        }catch(Exception e){}
        init();

        setSize(600,800);//初始大小
        setLocation(640,100);//初始位置
        setVisible(true);//是否可视
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);//X 退出
    }
    public void init(){
        setTitle("词法分析器");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);
        setResizable(false);
        setLayout(null);
        setBounds(10, 10, 300, 400);
        initMenu();//初始化菜单
        initTextArea();//初始化输入文本
        initButton();//初始化按钮
        initResultTable();//初始化结果区域
    }
    private void initMenu(){
        class fileListen implements ActionListener{
            @Override
            public void actionPerformed(ActionEvent e){
                JFileChooser fileChooser = new JFileChooser("D:\\工作\\programs");
                fileChooser.setFileSelectionMode(JFileChooser.FILES_ONLY);
                fileChooser.showOpenDialog(null);
                File file = fileChooser.getSelectedFile();
                if(file!=null){
                    try{
                        BufferedReader read = new BufferedReader(new FileReader( file ));
                        Object[] lines = read.lines().toArray();
                        StringBuffer bufferTA = new StringBuffer();
                        for(Object line:lines){
                            bufferTA.append(line.toString()+"\n");
                        }
                        TA.setText(bufferTA.toString());
                    }
                    catch (IOException err){
                    }
                }
            }
        }
        class manuallySet implements ActionListener {
            @Override
            public void actionPerformed(ActionEvent e) {
                String [] k ={};
```

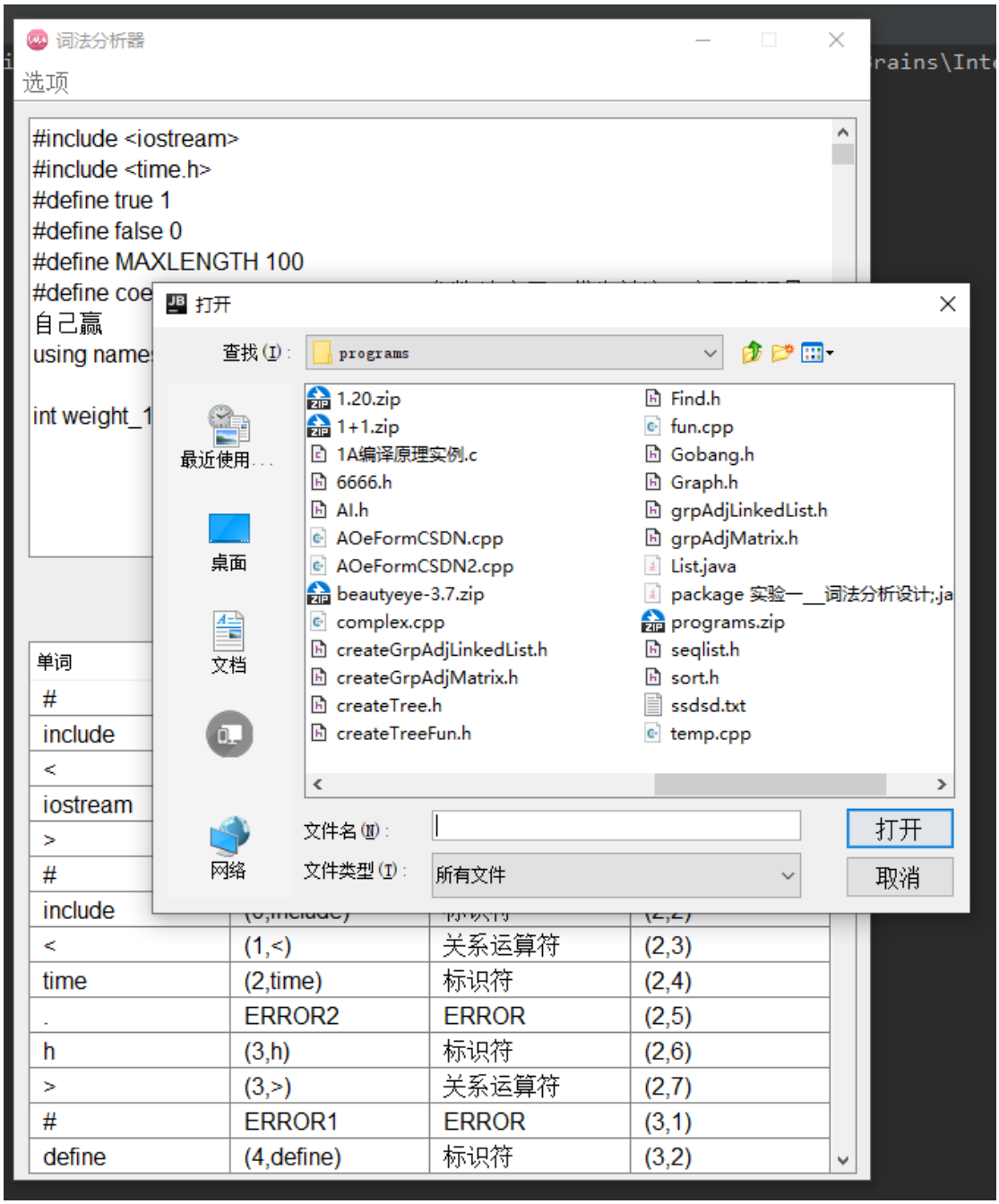
```
                String [] p ={};
                String kString,pString;
                Boolean changed = true;
                do{
                    kString = JOptionPane.showInputDialog(null,"请输入 K[ ]: \n","自定义
    K,P",JOptionPane.PLAIN_MESSAGE);
                    if(kString==null){
                        changed = false;
                        break;
                    }
                }while (kString.length()<2);
                if(changed){
                    boolean allchanged = true;
                    do{
                        pString = JOptionPane.showInputDialog(null,"请输入 P[ ]: \n","自定义
    K,P",JOptionPane.PLAIN_MESSAGE);
                        if(pString == null){
                            allchanged = false;
                            break;
                        }
                    }while (pString.length()<2);
                    if(allchanged){
                        kString = kString.substring(1,kString.length()-1);
                        pString = pString.substring(1,pString.length()-1);
                        k = kString.split(" ");
                        p = pString.split(" ");
                        if(k.length<2 || p.length<2){//如果输入不规范 警告 不修改 kp
                            JOptionPane.showMessageDialog(null, "格式输入错误
    ", "Error !", JOptionPane.ERROR_MESSAGE);
                        }
                        else
                            sl.manullySetKP(k,p);
                    }
                }
            }
        }
        class exitListen implements ActionListener {
            @Override
            public void actionPerformed(ActionEvent e) {
                dispose();
            }
        }

        bar = new JMenuBar();
        setJMenuBar(bar);

        menu = new JMenu("选项");
        bar.add(menu);

        file = new JMenuItem("选择文件");
        file.addActionListener(new fileListen());//读取文件到 TA 里

        manuallySet = new JMenuItem("手动设定");
        manuallySet.addActionListener(new manuallySet());

        exit = new JMenuItem("退出");
        exit.addActionListener(new exitListen());

        menu.add(file);
        menu.add(manuallySet);
        menu.add(exit);
    }
    private void initTextArea(){
        TA = new JTextArea();
        JScrollPane SP = new JScrollPane(TA);
        TA.setLineWrap(true); // 设置自动换行
        SP.setBounds(10, 10, 565, 300);
        add(SP);
    }
    private void initButton(){
        class clearListen implements ActionListener{
            @Override
            public void actionPerformed(ActionEvent e){
                TA.setText("");
                vecRes.clear();
                table.validate();
                table.updateUI();
                restablescrollPane.updateUI();
            }
        }
        class analyseListen implements ActionListener{
            @Override
            public void actionPerformed(ActionEvent e){
                text = TA.getText().split("\n");//这样分割后的 String 没有\n
                //for(String str:text) System.out.println(str);
                vecRes.clear();
                List<Result> resS = sl.Solve(text);
```

```
                for(Result result:resS){
                    vecRes.add(result.toStringArrary());
                }
                //vecRes.forEach(Strings -> {for(String str:Strings) System.out.print(str+" ");System.out.println();});
                table.validate();
                table.updateUI();
                restablescrollPane.updateUI();
            }
        }
        clear = new JButton("清空");
        clear.addActionListener(new clearListen());

        analyse = new JButton("分析");
        analyse.addActionListener(new analyseListen());

        clear.setBounds(400,320,70,35);
        analyse.setBounds(500,320,70,35);
        add(clear);
        add(analyse);
    }
    private void initResultTable(){
        tableDataModel = new TableDataModel(vecRes);
        table = new JTable(tableDataModel);
        table.setVisible(true);
        table.setPreferredScrollableViewportSize(new Dimension(550, 100));
        table.setRowHeight(24);
        restablescrollPane = new JScrollPane(table);
        restablescrollPane.setBounds(10, 367, 565, 363);
        add(restablescrollPane);
        pack();
    }
}

class TableDataModel extends AbstractTableModel{
    private Vector<String[]> TableData;//用来存放表格数据的线性表
    private Vector<String> TableTitle;//表格的 列标题
    public TableDataModel(Vector data){
        String Names[] = {"单词","二元序列","类 型","位置（行，列）"};
        Vector Namessss = new Vector();
        for(String str:Names){
            Namessss.add(str);
        }
        TableTitle = Namessss;
        TableData = data;
    }

    @Override
    public int getRowCount(){
        return TableData.size();
    }
    public int getColumnCount(){
        return TableTitle.size();
    }
    @Override
    public String getColumnName(int colum){
        return TableTitle.get(colum);
    }
    public Object getValueAt(int rowIndex, int columnIndex){
        String LineTemp[] = this.TableData.get(rowIndex);
        return LineTemp[columnIndex];
    }
    @Override
    public boolean isCellEditable(int rowIndex, int columnIndex){//不允许编辑
        return false;
    }
}
```

运行结果：


词法分析器
选项
#include <iostream>
#include <time.h>
#define true 1
#define false 0
#define MAXLENGTH 100
#define coefficient 0.9 //参数 决定了AI优先关注玩家不赢还是
自己赢
using namespace std;
int weight_1(Board B, int i, int j, int kind) {
char yes, no;
int a, b, count, comprehensiveweight = 0;
int weight;
if (kind == 1) {
清空
分析
单词 二元序列 类 型 位置（行，列）
ERROR1 ERROR (1,1)
include (0,include) 标识符 (1,2)
< (1,<) 关系运算符 (1,3)
iostream (1,iostream) 标识符 (1,4)
> (3,>) 关系运算符 (1,5)
ERROR1 ERROR (2,1)
include (0,include) 标识符 (2,2)
< (1,<) 关系运算符 (2,3)
time (2,time) 标识符 (2,4)
. ERROR2 ERROR (2,5)
h (3,h) 标识符 (2,6)
> (3,>) 关系运算符 (2,7)
ERROR1 ERROR (3,1)
define (4,define) 标识符 (3,2)


词法分析器
选项
#include <iostream>
#include <time.h>
#define true 1
#define false 0
#define MAXLENGTH 100
#define coefficient 0.9 //参数 决定了AI优先关注玩家不赢还是
自己赢
using namespace std;
int weight_1(Board B, int i, int j, int kind) {
char yes, no;
int a, b, count, comprehensiveweight = 0;
int weight;
if (kind == 1) {
自定义K,P
请输入K[ ]:
确定
取消
分析
单词 二元序列
ERROR1 ERROR (1,1)
include (0,include) 标识符 (1,2)
< (1,<) 关系运算符 (1,3)
iostream (1,iostream) 标识符 (1,4)
> (3,>) 关系运算符 (1,5)
ERROR1 ERROR (2,1)
include (0,include) 标识符 (2,2)
< (1,<) 关系运算符 (2,3)
time (2,time) 标识符 (2,4)
. ERROR2 ERROR (2,5)
h (3,h) 标识符 (2,6)
> (3,>) 关系运算符 (2,7)
ERROR1 ERROR (3,1)
define (4,define) 标识符 (3,2)



<	(1,<)	关系运算符	(2,3)
time	(2,time)	标识符	(2,4)
.	ERROR2	ERROR	(2,5)
h	(3,h)	标识符	(2,6)
>	(3,>)	关系运算符	(2,7)
#	ERROR1	ERROR	(3,1)
define	(4,define)	标识符	(3,2)

4.实验收获

本次试验算法部分较为简单，核心部分为递归行分析中的使用每个符号去匹配字符串的头部，根据匹配结果得出分析结果，然后将剩余的部分递归处理。大部分时间都用于学习设计界面 UI，初步掌握了 UI 的设计方法，有了一套自己的设计思路。

实验 2：LL(1)分析法

1.数据结构及算法描述

```
Set<String> noTerminal = new HashSet<>();//非终结符
Set<String> terminal = new HashSet<>();//终结符
Map<String,Set<String>> First = new HashMap<>();//First 集
Map<String,Set<String>> Follow  = new HashMap<>();//Follow 集
Map<String,Set<String>> select = new HashMap<>();//产生式的 select 集
List<String[]> Symbol_Gram = new ArrayList<>();// { { 符号 , 存在的文法  } *n }
//使用 set 保存数据,确保无重复元素 在计算的时候无序手动排除重复元素
```

```
传入文法 G

G 按照\n 和"->"以及"\\|"分割为单元

if(成功){
    保存并更新语法
}
else{
    弹出语法错误警告
}
if(存在左递归){
    弹出左递归警告
}

计算 First 集(){
    for(String symbol:终结符){
        First(symbol) = [symbol]
    }//终结符的 First 集是本身
    while(First 集的大小还在变化){
        for(String 左边->右边 :所有文法){
            取出右边下标为 0 的符号
            if(当前符号为非终结符){//当前符号的 First 集除了空都加入到左边符号的 First 集中
                First(左边).addAll(First(当前符号).except("ε"));//
                if(当前符号的 First 集合含有空){
                    看向下一个符号 //即下标+1 递归处理
                }
            }
            else if(当前符号是终结符 或者 "ε"){
                把当前符号加入到左边符号的 First 集中
            }
            else if(是"\0"){
                停止
            }
            else{
                停止
            }
        }
    }
}

计算 Follow 集(){
    在开始符号的 Follow 集中加入"#"
    for(String 当前符号:非终结符){
        for(语法 当前语法:所有的含有当前计算 Follow 集符号的语法){
            String 紧跟符号 = 当前语法中,当前符号之后的一个符号
            if(紧跟符号为终结符){
                把紧跟符号加到当前符号的 Follow 集中
            }
            else if(紧跟符号为非终结符){
                把紧跟符号的 First 集-"ε"加入到当前符号的 Follow 集中
                if(当前符号可以的 First 集含有空)
                    看向下一个符号//也是递归求解
            }
            else if(当前符号是"\0"){
                把空加入到当前符号的 Follow 集
            }
            else{
                报错 停止
            }
        }
    }
}

计算 Select 集(){
```

```
    for(String 当前文法(左边->右边) :所有文法){
        String 当前符号 = 右边的第一个符号
        if(当前符号是终结符){
            把当前符号加入 Select(当前文法)
        }
        else if(当前符号是"ε"或者是"\0"){
            把 Follow(左边)加入到 Select(当前文法)
        }
        else if(当前符号是非终结符){
            把 First(当前符号).except("ε")加入到 Select(当前文法)中
            if(当前符号的 First 集含有"ε)
                看向下一个符号 递归求解
        }
        else{
            报错 停止
        }
    }
}

计算 M 表(){
    for(String 当前文法:select 集){
        for(String 当前符号:Select(当前文法)){
            M(当前文法 的 左边,当前符号) = 当前文法
        }
    }
}

分析过程(String 输入的内容){
    初始化分析栈
    初始化输入栈
    while(结束标记为未结束){
        if(存在文法){
            if(M(x,a) == "ε"){
                分析栈出栈
            }
            else{
                分析栈.push(M(分析栈.pop(),a))
            }
        }
        else if(匹配到了 且没有结束){
            输入栈.pop()
            分析栈.pop()
        }
        else if(匹配成功 是#){
            结束标记修改为结束
        }
        else{
            报错
            结束标记修改为结束
        }
    }
                    }
```

2.算法流程图

LL(1)预测分析程序流程

3.源码及测试结果

Main.java：

```java
package 实验二___LL1分析法;

public class Main {
    public static void main(String[] args) {
        Windows windows = new Windows();
    }
}
```

Solution.java

```java
package 实验二___LL1分析法;
import java.util.*;
import java.util.List;
import java.util.stream.Collectors;

class Solution {
    Map<String,String> AnaTable = new HashMap<>();
    Map<String,Set<String>> select = new HashMap<>();//产生式的select集
    List<String[]> Symbol_Gram = new ArrayList<>();// { { 符号 , 存在的文法  } *n }
    String x;
    String a;
    Set<String> noTerminal = new HashSet<>();
    Set<String> terminal = new HashSet<>();
    Set<String> allG = new HashSet<>();
    Map<String,Set<String>> First = new HashMap<>();
    Map<String,Set<String>> Follow  = new HashMap<>();
    String GStart = "null";
    Solution(){
        setG("E -> TG \n" +
                "G -> +TG | -TG \n" +
                "G -> ε \n" +
                "T -> FS \n" +
                "S -> *FS | /FS \n" +
                "S -> ε \n" +
                "F -> (E) \n" +
                "F->i \n");
    }
    String setG(String G){
            Map<String,String> AnaTable = new HashMap<>();
            Symbol_Gram.clear();
            G = G.replaceAll(" ","");
            String []Gs = G.split("\n");
            if(Gs.length<1){
                return "输入错误";
            }
            GStart =   Gs[0].substring(0,1) ;
            for (String gLine : Gs) {
                if( gLine.split("->").length!=2){
                    return "格式错误";
                }
                String split0 = gLine.split("->")[0];
                for (String str : gLine.split("->")[1].split("\\|")) {
                    String [] SingleG = {split0,str};
                    Symbol_Gram.add(SingleG);
                }
            }
            GStart =   Gs[0].substring(0,1) ;
            MyStack stack = new MyStack();
            for (String[] strings : Symbol_Gram) {
                if(strings[0].equals(strings[1].substring(0,1))){
                    return "左递归";
                }
            }
        getFF();//计算First Follow集
        for (String[] strings : Symbol_Gram) {
            select.put(strings[0]+strings[1],new HashSet<>());
        }
        //例如   s->ab   strings[0]  ->  strings[1]
        for (String[] strings : Symbol_Gram) {
```

```
60.             setSelect(strings,0);   //创建 select 集  key = 产生式 value  = [] Set
61.         }
62.         AnaTable.clear();//清空
63.         terminal.add("#");//在终结符内加入#   以达到
64.         for (String s0 : select.keySet()) {
65.             for (String s1 : select.get(s0)) {
66.                 AnaTable.put(s0.substring(0,1)+s1,s0.substring(1));
67.             }
68.         }
69.         this.AnaTable =AnaTable;
70.         return null;
71.     }
72.     Vector<String[]> Solve(String text){
73.         Vector<String[]> procesList = new Vector();
74.         int textLength = text.length();
75.         MyStack AnaStack = new MyStack();//分析栈
76.         MyStack inputString = new MyStack();//输入串
77.         AnaStack.push("#","E","S");
78.         inputString.push( new StringBuffer(text).reverse().toString().split("") );
79.         Boolean flag = true;
80.         Boolean matched = true;
81.         int linenumber = 0;
82.         while (flag){
83.             String[] INFO = new String[5];
84.             INFO[0] = String.valueOf(linenumber++);
85.             INFO[1] = AnaStack.toString();
86.             StringBuffer inputsb= new StringBuffer(inputString.toString());
87.             for(int sblength = inputsb.length() ; sblength < textLength+1 ; sblength++ ){
88.                 inputsb.append(" ");
89.             }
90.             INFO[2] = inputsb.reverse().toString();
91.             x = AnaStack.getTop();//获取分析栈顶
92.             a = inputString.getTop();//第一个符号读到 a
93.             if(M(x,a)!=null){//存在对应的文法
94.                 if(M(x,a)[0].equals("ε")){//为空
95.                     INFO[3] = x+" -> ε";
96.                     INFO[4] = "POP";
97.                     AnaStack.pop();
98.                 }
99.                 else {
100.                    AnaStack.push(M(AnaStack.pop(), a));
101.                    StringBuffer sb = new StringBuffer();
102.                    for(String string:M(x,a)){
103.                        sb.append(string);
104.                    }
105.                    INFO[3] = x+" -> " + sb.reverse();
106.                    INFO[4] = "POP,PUSH("+sb.reverse()+")";
107.                    //System.out.println("存在文法
    ["+x+","+a+"] -> ["+sb + "]  STACK:" + AnaStack);
108.                }
109.            }
110.            else if( !x.equals("#") && x.equals(a) ){//匹配到了
111.                //System.out.println("匹配到了"+x+" "+a);
112.                INFO[4] = "POP,GETNEXT(i)";
113.                inputString.pop();
114.                AnaStack.pop();
115.            }
116.            else if(x.equals("#") && x.equals(a)){//结束了
117.                flag=false;
118.            }
119.            else{//报错
120.                flag=false;
121.                matched=false;
122.            }
123.            for(int i = 0 ; i < 5 ; i++ ){
124.                if(INFO[i]!=null)
125.                    INFO[i] = " "+INFO[i];
126.                else
127.                    INFO[i] = " ";
128.            }
129.
130.            procesList.add(INFO);
131.        }
132.        if(matched){
133.            //System.out.println("匹配成功");
134.        }
135.        else{
136.            //System.out.println("匹配失败");
137.            String ss[] ={"ERROR","ERROR","ERROR","ERROR","ERROR"};
138.            procesList.add(ss);
139.        }
140.
141. //      for (String s : Grammer) {
142. //          System.out.print(s+" ");
143. //      }
144.        return procesList;
145.    }
```

```
    String[] M(String Line ,String column){
        //倒序 分割
        //System.out.println("查询 " + Line + "<->" + column);
        if(AnaTable.get(Line+column) == null) {
            if ( AnaTable.get(Line+"#")!=null) {
//              System.out.println("返回空");
                String ss[] = {"ε"};
                return ss;
            }
            else {
                return null;
            }
        }
        else{
            return new StringBuffer(AnaTable.get(Line+column)).reverse().toString().split(" ");
        }
    }
    Map[] getFF(){
        noTerminal.clear();
        First.clear();
        Follow.clear();
        allG.clear();
        terminal.clear();
        for (String[] strings : Symbol_Gram) {
            noTerminal.add(strings[0]);
        }//非终结符
        for (String[] strings : Symbol_Gram) {
            allG.add(strings[0]);
            for (String s : strings[1].split("")) {
                allG.add(s);
            }
        }//所有符
        allG.remove("ε");
        terminal.addAll(allG.stream().filter(S->  !noTerminal.contains(S)).collect(Collectors.toSet()));//终结符 = 所有符号 - 非终结符
        //System.out.println(Grammer+"\n"+EndG);
        for (String s1 : allG) {
            First.put(s1,new HashSet<>());
            Follow.put(s1,new HashSet<>());
        }//终结符的 First 集是本身终结符的 First 集是本身
        for (String s1 : terminal) {
            First.get(s1).add(s1);
        }
        int FirstSize = 0;
        do{
            FirstSize = 0;
            for (String s1 : First.keySet()) {
                FirstSize+=First.get(s1).size();
            }
            for (String[] strings : Symbol_Gram) {
                String lam = strings[1];
                String G = strings[0];
                setFirst(lam,G);
            }
            for (String s1 : First.keySet()) {
                FirstSize-=First.get(s1).size();
            }
        }while (FirstSize != 0);
        Follow.get(GStart).add("#");//文法开始符号 Follow 加入#

        int  FollowSize = 0;
        do{
            FollowSize = 0;
            for (String s1 : Follow.keySet()) {
                FollowSize+=Follow.get(s1).size();
            }
            for (String[] strings : Symbol_Gram) {
                String lam = strings[1];
                String G = strings[0];
                for (String s2 : noTerminal) {
                    setFollow(lam,G,s2);
                }
            }


            for (String s1 : Follow.keySet()) {
                FollowSize-=Follow.get(s1).size();
            }
        }while (FollowSize != 0);
        for (String s1 : terminal) {
            Follow.remove(s1);
        }
        Map[] res = new Map[2];
        res[0] = First;
        res[1] = Follow;
        return res;
```

```
    }
    private void setFirst(String lam,String G){
        String first = lam.substring(0,1);
        if(terminal.contains(first)){//终结符
            First.get(G).add(first);
        }
        else if(first.equals("ε")){//符号空
            First.get(G).add("ε");
        }
        else if(noTerminal.contains(first)){//非终结符
            First.get(G).addAll(First.get(first).stream().filter(S->!S.equals("ε")).collect(Collectors.toSet()));
            if(M(first,"ε")!=null){//是否可以推出空
                setFirst(lam.substring(1),G);//扫描下一个
            }
        }
        else {
        }
    }
    private void setFollow(String lam,String G,String sym){//产生式  ->左边的符号  当先所求的非终结符
        if(!lam.contains(sym)){
            return;
        }
        int index = lam.indexOf(sym);//位置
        if(index == lam.length()-1){// 是\0
            Follow.get(sym).add("#");
            Follow.get(sym).addAll(Follow.get(G));//把产生式左边的 FOLLOW 加入到其的 FOLLOW 集中
            return;
        }
        else if(index < lam.length()-1){
            String next = lam.substring(index+1,index+2);//右边的符号
            if(terminal.contains(next)){//是终结符
                Follow.get(sym).add(next);
            }
            else if(noTerminal.contains(next)){//非终结符
                Follow.get(sym).addAll(First.get(next).stream().filter(S->!S.equals("ε")).collect(Collectors.toSet()));//把他的 Fist 集-ε 加入到当前分析的 Follow 集中
                if(M(next,"ε")!=null){//检查可否推出空
                    //扫描下一个符号
                    StringBuffer changedLam = new StringBuffer(lam) ;
                    changedLam.deleteCharAt(index+1);//删除 达到左移的效果\
                    setFollow(changedLam.toString(),G,sym);
                }
            }


        }

    }
    private void setSelect(String[] strings,int index){
        if(index == strings[1].length()){//是空
            select.get(strings[0]+strings[1]).addAll(Follow.get(strings[0]));
        }
        else {
            String firstSym = strings[1].substring(index,index+1);
            if(terminal.contains(firstSym)){//如果是终结符
                select.get(strings[0]+strings[1]).add(strings[1].substring(0,1));
            }
            else if(firstSym.equals("ε")){//是空
                select.get(strings[0]+strings[1]).addAll(Follow.get(strings[0]));
            }
            else if(noTerminal.contains(firstSym)){//是非终结符
                select.get(strings[0]+strings[1]).addAll(First.get(firstSym).stream().filter(S->!S.equals("ε")).collect(Collectors.toSet()));
                if(M(firstSym,"ε") != null){//可以推空  则扫描下一个
                    setSelect(strings,index+1);
                }
            }
        }
    }
}
class MyStack{
    List<String> s;
    MyStack(){
        s = new LinkedList<>();
    }
    void push(String value){
        s.add(value);
    }
    void push(String...values){
        for(String value:values){
            push(value);
        }
    }
    String pop(){
```

```
        return s.remove(s.size()-1);
    }
    String getTop(){
        return s.get(s.size()-1);
    }
    @Override
    public String toString(){
        StringBuffer sb = new StringBuffer();
        for(String value:s){
            sb.append(value);
        }
        return sb.toString();
    }
    public Boolean isEmpty(){
        return s.size()==0;
    }
}
```

GUI.java

```
package 实验二___LL1分析法;

import javax.swing.*;
import javax.swing.table.AbstractTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.*;

class Windows extends JFrame {
    JButton clear,confirm,setG,FF;
    JTextArea textArea;
    JTabbedPane tabbedPane;
    Solution sol;
    Windows(){
        setVisible(false);
        try{
            setIconImage(new ImageIcon("bilibili.PNG").getImage());
            Font f = new Font("Yahei Consolas Hybrid",Font.PLAIN,16);
            String   names[]={ "MenuBar","Menu","MenuItem", "TextArea", "Button", "ScrollPane", "Table","TabbedPane"};
            for (String item : names) {
                UIManager.put(item+ ".font",f);
            }
            UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");

        }catch(Exception e){}
        init();
        setSize(800,600);//初始大小
        setLocation(300,200);//初始位置
        setVisible(true);//是否可视
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);//X退出
    }
    void init(){
        setTitle("LL(1)分析法");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);
        setResizable(false);
        setLayout(null);
        sol = new Solution();
        initButton();
        initText();
        initResult();

    }
    void initButton(){
        class clearListen implements ActionListener{
            @Override
            public void actionPerformed(ActionEvent e){
                textArea.setText("");
                tabbedPane.removeAll();
                tabbedPane.updateUI();
            }
        }
        class confirmListen implements ActionListener{
            @Override
            public void actionPerformed(ActionEvent e){
                tabbedPane.removeAll();
                for(String text:textArea.getText().split("\n")){
```

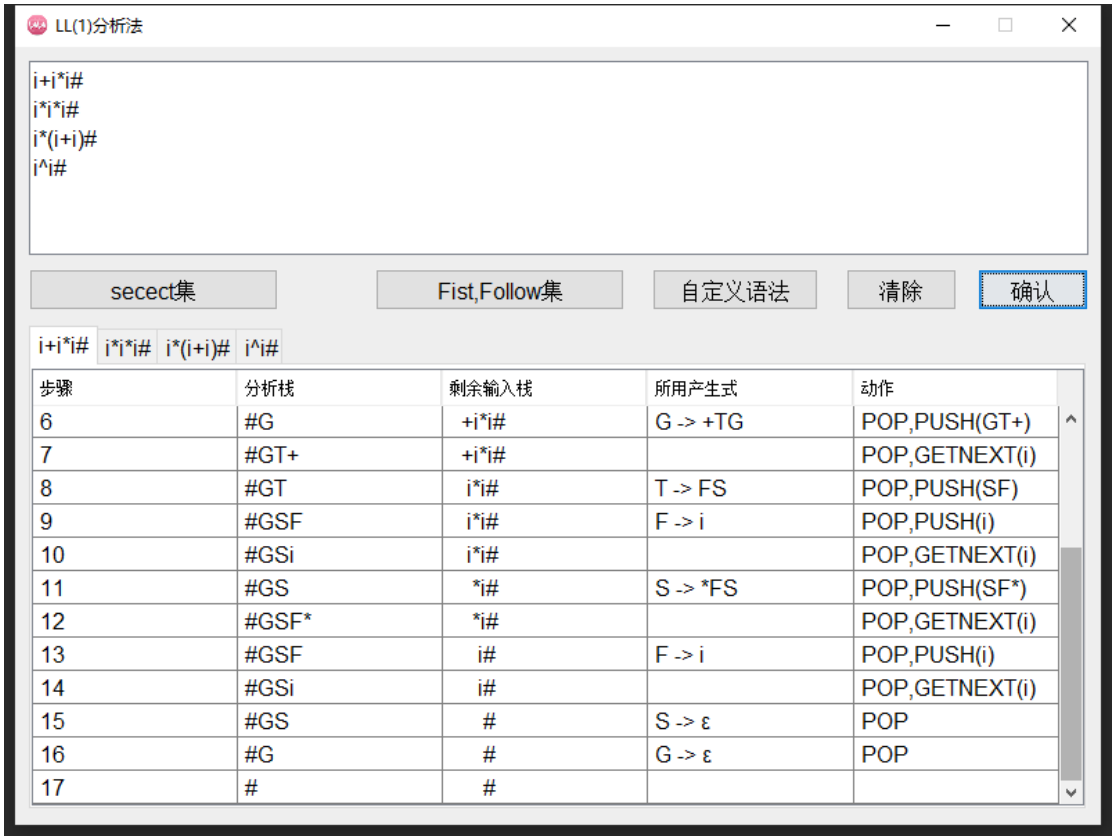
```
                    if(text.length()<2 || !text.substring(text.length()-1).equals("#") ){
                        JOptionPane.showMessageDialog(null, "格式输入错误
    ", "Error !", JOptionPane.ERROR_MESSAGE);
                        break;
                    }
                    addTable(text,sol.Solve(text));
                    tabbedPane.updateUI();
                }


            }
        }
        class MyDialog extends JDialog implements ActionListener{
            JTextArea input;
            JButton confirm,cancel;
            String title;
            MyDialog(Windows f){
                setLayout(null);
                setResizable(false);
                setIconImage(new ImageIcon("bilibili.PNG").getImage());
                setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
                setTitle("请输入语法");
                input=new JTextArea("E -> TG \n" +
                        "G -> +TG | -TG \n" +
                        "G -> ε \n" +
                        "T -> FS \n" +
                        "S -> *FS | /FS \n" +
                        "S -> ε \n" +
                        "F -> (E) \n" +
                        "F->i \n");
                JScrollPane jScrollPane = new JScrollPane(input);
                jScrollPane.setBounds(10,10,265,200);
                add(jScrollPane);

                class confirmListener implements ActionListener{
                    @Override
                    public void actionPerformed(ActionEvent e){
                        String getInput = input.getText();
                        String setRes = sol.setG(getInput);
                        if(setRes!=null){
                            JOptionPane.showMessageDialog(null, setRes, "Error !", JOptionP
    ane.ERROR_MESSAGE);
                        }
                        else {
                            setVisible(false);
                        }
                    }
                }
                confirm=new JButton("确定");
                confirm.addActionListener(new confirmListener());
                confirm.setBounds(195,220,80,30);
                add(confirm);

                class cancelListener implements ActionListener{
                    @Override
                    public void actionPerformed(ActionEvent e){
                        input.setText("E -> TG \n" +
                                "G -> +TG | -TG \n" +
                                "G -> ε \n" +
                                "T -> FS \n" +
                                "S -> *FS | /FS \n" +
                                "S -> ε \n" +
                                "F -> (E) \n" +
                                "F->i \n");
                        setVisible(false);
                    }
                }
                cancel=new JButton("取消");
                cancel.addActionListener(new cancelListener());
                cancel.setBounds(105,220,80,30);
                add(cancel);


                setBounds(600,260,300,300);
                setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
            }
            public void actionPerformed(ActionEvent e){
                setVisible(true);
            }
        }
        class FFListen extends JDialog implements ActionListener{
            JTabbedPane jTabbedPane;
            FFListen(){
                jTabbedPane = new JTabbedPane();
                setLayout(null);
                setResizable(false);
                setIconImage(new ImageIcon("bilibili.PNG").getImage());
```

```
                setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
                setTitle("Fist Follow 集");
                setBounds(650,300,300,400);
                jTabbedPane.setBounds(10,7,267,350);
                add(jTabbedPane);

            }
            @Override
            public void actionPerformed(ActionEvent e){
                setVisible(true);
                jTabbedPane.removeAll();
                JTable First,Follow;
                Map<String,Set<String>>[]  res =   sol.getFF();

                Object[][] FirstData = new Object[res[0].size()][2];
                int index = 0;
                for (String s : res[0].keySet()) {
                    FirstData[index][0] = s;
                    FirstData[index][1] = res[0].get(s).toString();
                    index++;
                }
                Object[] columnNames = {"", ""};
                First = new JTable(FirstData, columnNames);
                First.setRowHeight(24);
                First.getTableHeader().setVisible(false);
                JScrollPane FirstScrollable = new JScrollPane(First);
                FirstScrollable.setBorder(null);
                jTabbedPane.addTab("First 集",FirstScrollable);

                Object[][] FollowData = new Object[res[1].size()][2];
                index = 0;
                for (String s : res[1].keySet()) {
                    FollowData[index][0] = s;
                    FollowData[index][1] = res[1].get(s).toString();
                    index++;
                }
                Follow = new JTable(FollowData, columnNames);
                Follow.setRowHeight(24);
                Follow.getTableHeader().setVisible(false);
                JScrollPane FollowScrollable = new JScrollPane(Follow);
                FollowScrollable.setBorder(null);
                jTabbedPane.addTab("Follow 集",FollowScrollable);

            }
        }


        clear = new JButton("清除");
        clear.setBounds(600,160,80,30);
        clear.addActionListener(new clearListen());

        confirm = new JButton("确认");
        confirm.setBounds(695,160,80,30);
        confirm.addActionListener(new confirmListen());

        setG = new JButton("自定义语法");
        setG.setBounds(460,160,120,30);
        setG.addActionListener(new MyDialog(this));

        FF = new JButton("Fist,Follow 集");
        FF.setBounds(260,160,180,30);
        FF.addActionListener(new FFListen());

        class selectListen extends JDialog implements ActionListener{
            JTabbedPane jTabbedPane;
            selectListen(){
                jTabbedPane = new JTabbedPane();
                setLayout(null);
                setResizable(false);
                setIconImage(new ImageIcon("bilibili.PNG").getImage());
                setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
                setTitle("select 集");
                setBounds(650,300,500,300);
                jTabbedPane.setBounds(10,7,467,250);
                add(jTabbedPane);

            }
            @Override
            public void actionPerformed(ActionEvent e){
                setVisible(true);
                jTabbedPane.removeAll();
                JTable First,Follow;
                Map<String,Set<String>>  res =   sol.select;

                Object[][] FirstData = new Object[res.size()][2];
                int index = 0;
                for (String s : res.keySet()) {
```

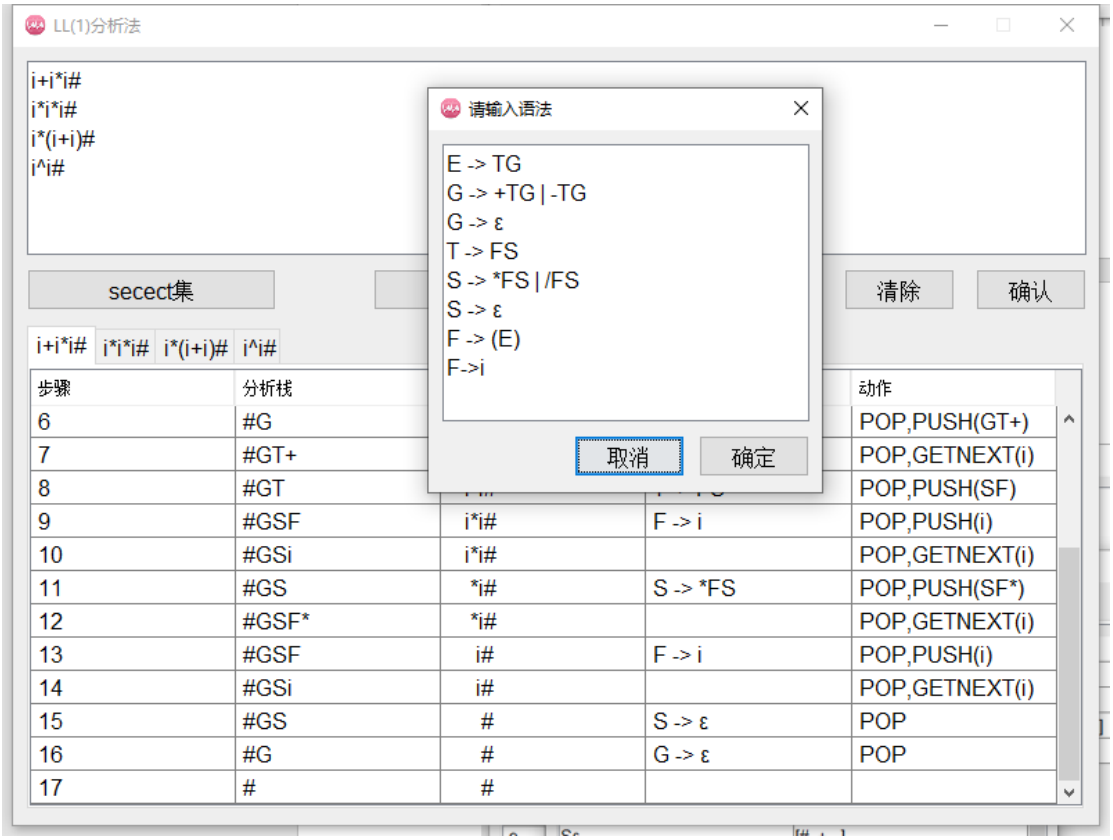
```
                    FirstData[index][0] = s;
                    FirstData[index][1] = res.get(s).toString();
                    index++;
                }
                Object[] columnNames = {"", ""};
                First = new JTable(FirstData, columnNames);
                First.setRowHeight(24);
                First.getTableHeader().setVisible(false);
                JScrollPane FirstScrollable = new JScrollPane(First);
                FirstScrollable.setBorder(null);
                jTabbedPane.addTab("select 集",FirstScrollable);

                {
                    Map<String,Integer> Grammap = new HashMap<>();
                    Map<String,Integer>  endGmap = new HashMap<>();


                    int a = 0;
                    for (String s1 : sol.terminal) {
                        //System.out.print("     "+s1);
                        endGmap.put(s1,a);
                        a++;
                    }
                    int b = 0;
                    for (String s : sol.noTerminal) {
                        Grammap.put(s,b);
                        b++;
                    }
                    String[][] map = new String[Grammap.size()][endGmap.size()];
                    //System.out.println();
                    for (String s1 : sol.noTerminal) {
                        //System.out.print(s1+":  ");
                        for (String s2 : sol.terminal) {
                            String resss = sol.AnaTable.get(s1+s2);
                            if(resss == null){
                                //System.out.print("      ");
                                map[Grammap.get(s1)][endGmap.get(s2)] = " ";
                            }
                            else {
                                //System.out.print(resss+"    ");
                                map[Grammap.get(s1)][endGmap.get(s2)] = resss;
                            }
                        }
                        //System.out.println();
                    }
                    //System.out.println("====================");


//                    for (String[] strings : map) {
//                        for (String string : strings) {
//                            System.out.print(string+" ");
//                        }
//                        System.out.println();
//                    }


                    String [] colum = new String[sol.noTerminal.size()+1];
                    int counter = 0;
                    for (String s : sol.noTerminal) {
                        colum[counter] = s;
                        counter++;
                    }
                    String[][] data = new String[map.length][map[0].length+1];
                    for (int j = 0; j < data.length; j++ ) {
                        data[j][0] = colum[j];
                        for (int i = 1; i < data[0].length; i++) {
                            data[j][i] = map[j][i-1];
                        }
                    }
                    String[] name = new String[sol.terminal.size()+1];
                    int i = 1;
                    name[0] = " ";
                    for (String s : sol.terminal) {
                        name[i] = s;
                        i++;
                    }
                    JTable  mmm = new JTable(data, name);
                    mmm.setRowHeight(30);
                    JScrollPane secsa = new JScrollPane(mmm);
                    secsa.setBorder(null);
                    jTabbedPane.addTab("分析表",secsa);

                }
```

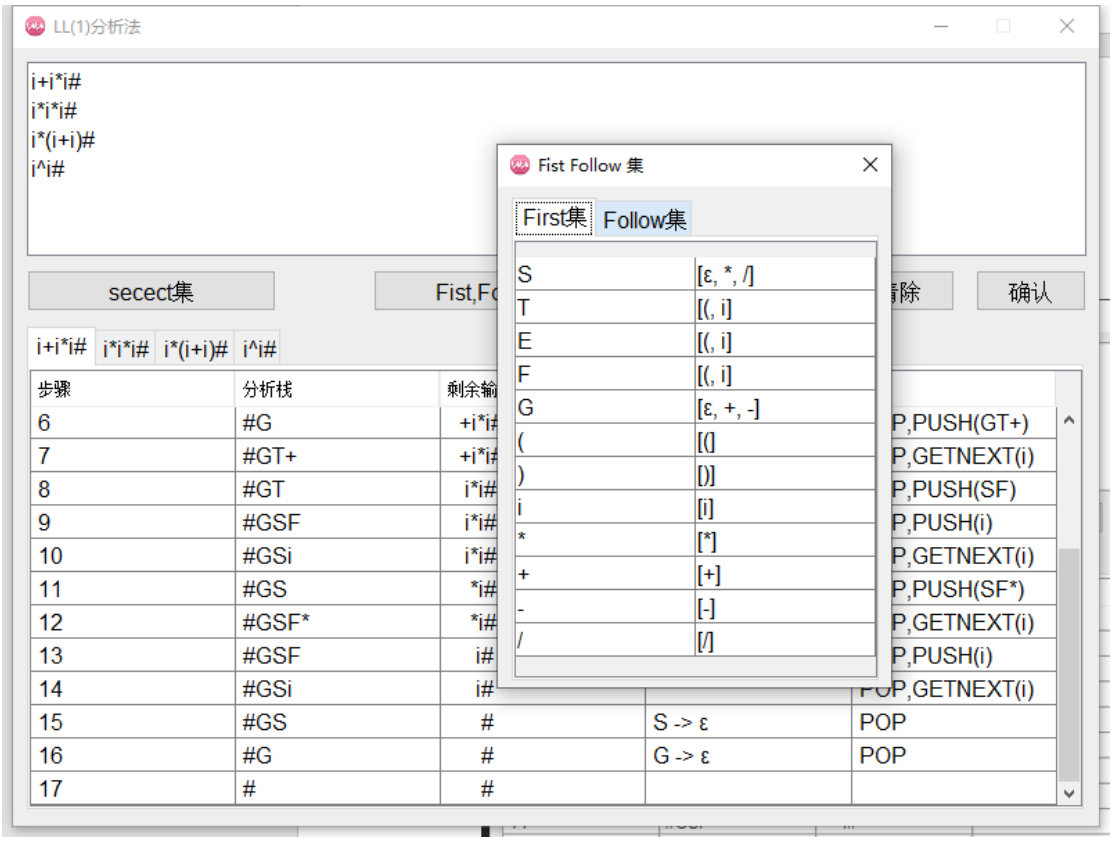
```
317.
318.             }
319.         }
320.
321.
322.         JButton select;
323.         select = new JButton("secect集");
324.         select.setBounds(10,160,180,30);
325.         select.addActionListener(new selectListen());
326.         add(select);
327.
328.         add(clear);
329.         add(confirm);
330.         add(setG);
331.         add(FF);
332.     }
333.     void initText(){
334.         textArea = new JTextArea("i+i*i#\ni*i*i#\ni*(i+i)#\ni^i#");
335.         JScrollPane textAreaRollPane = new JScrollPane(textArea);
336.         textAreaRollPane.setBounds(10,10,765,140);
337.         add(textAreaRollPane);
338.     }
339.     void initResult(){
340.         tabbedPane = new JTabbedPane();
341.         tabbedPane.setBounds(10, 200, 765, 350);
342.         add(tabbedPane);
343.     }
344.     void addTable(String title,Vector vec ){
345.         TableDataModel tableDataModel = new TableDataModel(vec);
346.         JTable table = new JTable(tableDataModel);
347.         table.setVisible(true);
348.         table.setPreferredScrollableViewportSize(new Dimension(550, 100));
349.         table.setRowHeight(24);
350.         JScrollPane tablePane = new JScrollPane(table);
351.         tablePane.setBounds(10, 200, 765, 350);
352.         tabbedPane.addTab(title,tablePane);
353.     }
354. }
355.
356. class TableDataModel extends AbstractTableModel {
357.     private Vector<String[]> TableData;//用来存放表格数据的线性表
358.     private Vector<String> TableTitle;//表格的 列标题
359.     public TableDataModel(Vector data){
360.         String Names[] = {"步骤","分析栈","剩余输入栈","所用产生式","动作"};
361.         Vector Namessss = new Vector();
362.         for(String str:Names){
363.             Namessss.add(str);
364.         }
365.         TableTitle = Namessss;
366.         TableData = data;
367.     }
368.
369.     @Override
370.     public int getRowCount(){
371.         return TableData.size();
372.     }
373.     public int getColumnCount(){
374.         return TableTitle.size();
375.     }
376.     @Override
377.     public String getColumnName(int colum){
378.         return TableTitle.get(colum);
379.     }
380.     public Object getValueAt(int rowIndex, int columnIndex){
381.         String LineTemp[] = this.TableData.get(rowIndex);
382.         return LineTemp[columnIndex];
383.     }
384.     @Override
385.     public boolean isCellEditable(int rowIndex, int columnIndex){//不允许编辑
386.         return false;
387.     }
388. }
```

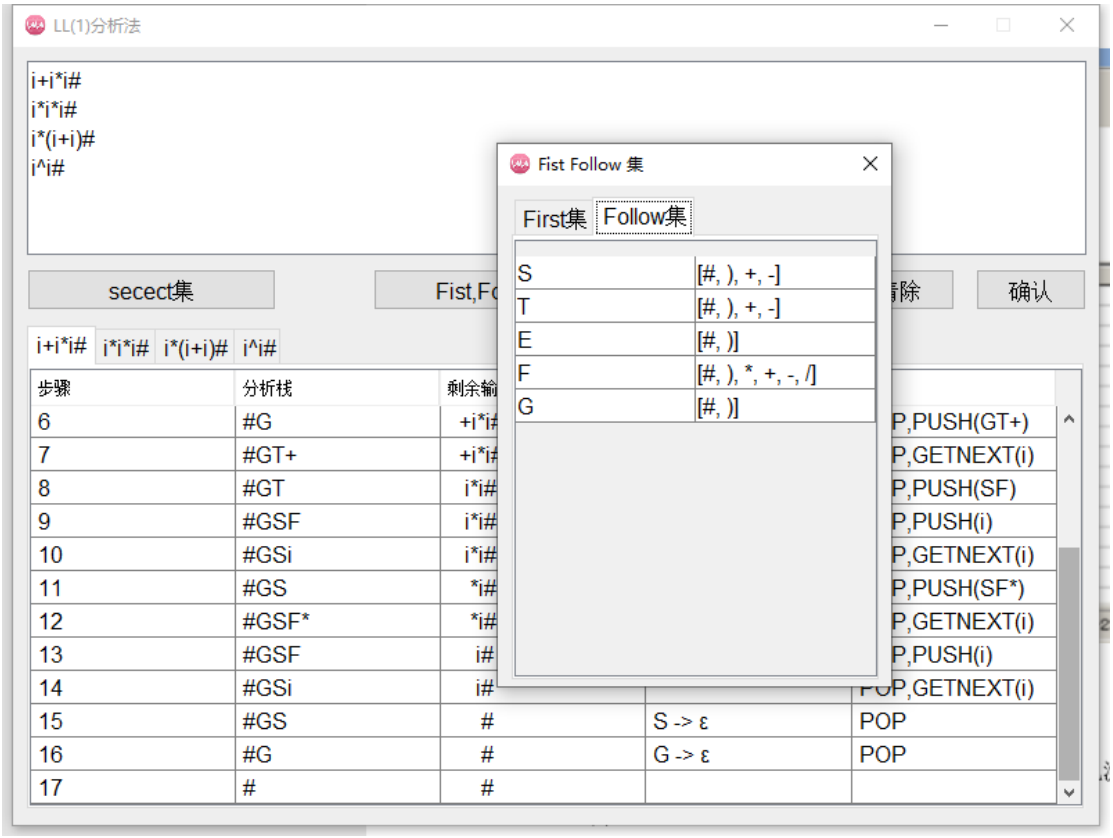
运行结果：

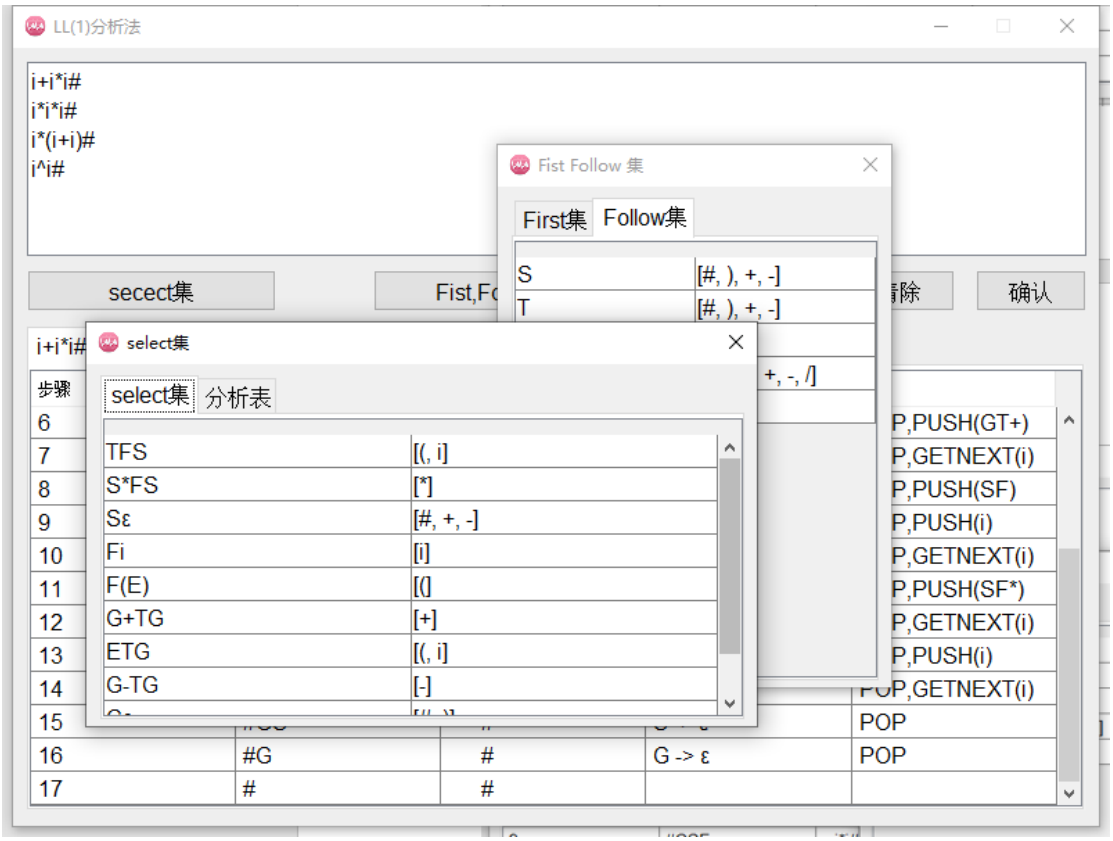


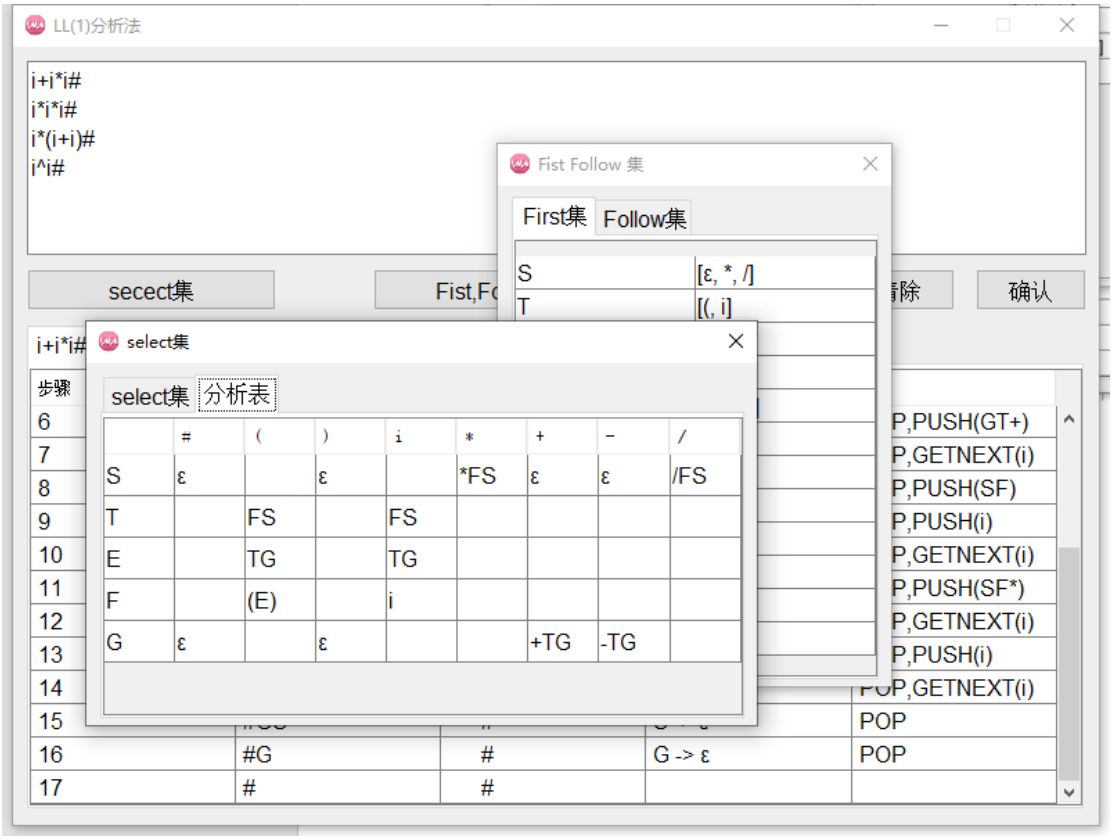
步骤	分析栈	剩余输入栈	所用产生式	动作
6	#G	+i*i#	G -> +TG	POP,PUSH(GT+)
7	#GT+	+i*i#		POP,GETNEXT(i)
8	#GT	i*i#	T -> FS	POP,PUSH(SF)
9	#GSF	i*i#	F -> i	POP,PUSH(i)
10	#GSi	i*i#		POP,GETNEXT(i)
11	#GS	*i#	S -> *FS	POP,PUSH(SF*)
12	#GSF*	*i#		POP,GETNEXT(i)
13	#GSF	i#	F -> i	POP,PUSH(i)
14	#GSi	i#		POP,GETNEXT(i)
15	#GS	#	S -> ε	POP
16	#G	#	G -> ε	POP
17	#	#		


LL(1)分析法
i+i*i#
i*i*i#
i*(i+i)#
i^i#
secect集
清除
确认
i+i*i#
i*i*i#
i*(i+i)#
i^i#
步骤
分析栈
动作
6 #G POP,PUSH(GT+)
7 #GT+ POP,GETNEXT(i)
8 #GT POP,PUSH(SF)
9 #GSF i*i# F -> i POP,PUSH(i)
10 #GSi i*i# POP,GETNEXT(i)
11 #GS *i# S -> *FS POP,PUSH(SF*)
12 #GSF* *i# POP,GETNEXT(i)
13 #GSF i# F -> i POP,PUSH(i)
14 #GSi i# POP,GETNEXT(i)
15 #GS # S -> ε POP
16 #G # G -> ε POP
17 # #
请输入语法
E -> TG
G -> +TG | -TG
G -> ε
T -> FS
S -> *FS | /FS
S -> ε
F -> (E)
F->i
取消
确定


LL(1)分析法
i+i*i#
i*i*i#
i*(i+i)#
i^i#
secect集
确认
i+i*i#
i*i*i#
i*(i+i)#
i^i#
步骤
分析栈
17 # #
Fist Follow 集
First集
Follow集
S [ε, *, /]
T [(, i]
E [(, i]
F [(, i]
G [ε, +, -]
([(]
) [)]
i [i]
* [*]
+ [+]
- [-]
/ [/]


LL(1)分析法
i+i*i#
i*i*i#
i*(i+i)#
i^i#
secect集
确认
i+i*i#
i*i*i#
i*(i+i)#
i^i#
步骤
分析栈
剩余输
6 #G +i*i#
7 #GT+ +i*i#
8 #GT i*i#
9 #GSF i*i#
10 #GSi i*i#
11 #GS *i#
12 #GSF* *i#
13 #GSF i#
14 #GSi i#
15 #GS # S -> ε POP
16 #G # G -> ε POP
17 # #
Fist Follow 集
First集
Follow集
S [#,), +, -]
T [#,), +, -]
E [#,)]
F [#,), *, +, -, /]
G [#,)]
P,PUSH(GT+)
P,GETNEXT(i)
P,PUSH(SF)
P,PUSH(i)
P,GETNEXT(i)
P,PUSH(SF*)
P,GETNEXT(i)
P,PUSH(i)
POP,GETNEXT(i)


LL(1)分析法
i+i*i#
i*i*i#
i*(i+i)#
i^i#
secect集
确认
Fist Follow 集
First集
Follow集
S [#,), +, -]
T [#,), +, -]
select集
select集
分析表
TFS [(, i]
S*FS [*]
Sε [#, +, -]
Fi [i]
F(E) [(]
G+TG [+]
ETG [(, i]
G-TG [-]
步骤
6
7
8
9
10
11
12
13
14
15 POP
16 #G # G -> ε POP
17 # #
P,PUSH(GT+)
P,GETNEXT(i)
P,PUSH(SF)
P,PUSH(i)
P,GETNEXT(i)
P,PUSH(SF*)
P,GETNEXT(i)
P,PUSH(i)
POP,GETNEXT(i)



	#	(	)	i	*	+	-	/
S	ε		ε		*FS	ε	ε	/FS
T		FS		FS				
E		TG		TG				
F		(E)		i				
G	ε		ε			+TG	-TG	

4.实验收获

本次实验设计的程序层次分明，程序界面与处理程序低耦合高内聚，分析器使用文法作为参数，对传入的字符串进行分析，并返回分析结果，此外通过调用类中的方法可以返回对应的 First 集 Follow 集等，便于显示。LL1 对分析法有了更好的理解和掌握。

实验三　LR(1)分析法

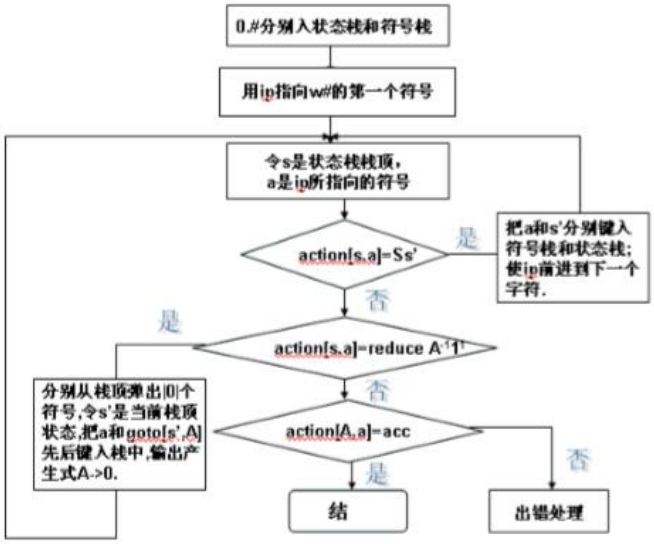
9. 数据结构及算法描述

```
a)   public String grammarText;
b)   public List<String[]> grammar = new ArrayList<>();//语法格式:(T->SF)= {T,SF}
c)   public Set<String> nonTerminal = new HashSet<>();//非终结符
d)   public Set<String>  terminal = new HashSet<>();//终结符
e)   public Set<String> allSymbol = new HashSet<>();//全部符号
f)   public Map<String,Set<String>> First = new HashMap<>();//First 集
g)   public List<project> projectList = new ArrayList<>();//所有项目
h)   public List<projectSet> cProjectSets = new ArrayList<>();//项目集 C
i)   public Map<point,String> ActionTable = new HashMap<>();//action 表
j)   public Map<point,String> GOTO = new HashMap<>();//goto 表
k)   public String[][] ActionAndGoTo;//Action 和 GOTO 表
l)   public String startSymbol = "S`";//文法开始符号


1.  初始化文法
2.  读取文法,计算 First 集(使用实验 2 的算法即可)
3.  初始化项目 List(){
4.      加入(S`->.S),#和(S`->S.,#)//开始文法 特殊情况 手动添加
5.      for(文法 A->BC:所有文法){
6.          for( index : 所有可以插入点的位置){
7.              for(文法 当前文法:所有文法){
8.              if(当前文法的右边含有 A){
9.                  把(A->BC,index,First(点后部的字符串))加入到项目集中
10.             }
11.         }
12.         }
13.     }
14. }
15. 初始化项目集 C(){
16.     初始化栈
17.     创建一个项目集
18.     把开始文法的项目放入其中
19.     计算它的闭包
20.     把这个项目放入栈中
21.     while(栈非空){
22.         当前项目集 = 栈.pop()
23.         for(String Symbol : 每个符号){
24.             创建一个新的项目集
25.             for(项目 : 当前项目集中的所有项目){
26.                 if(是形如 A->...•X... 的项目){
27.                     点向后移动一位创建新的项目,加入到新的项目集中
28.                 }
29.             }
30.             计算闭包(新的项目集)
31.             if(如果新的项目集为空){
32.                 标记当前项目集通过 Symbol 跳转到-1
33.             }
34.             else if(新的项目集已经存在){
35.                 标记当前项目集通过 Symbol 跳转到 项目集 List.indexOf(新的项目集)
36.             }
37.             else{
38.                 把新的项目集加入到项目集 List 中
39.                 标记当前项目集通过 Symbol 跳转到 项目集 List.indexOf(新的项目集)
40.                 栈.push(新的项目集)
41.             }
42.         }
43.     }
44. }
45. 计算闭包(项目集){
46.     新建栈
47.     项目集.forEach(栈::push)
48.     while(栈非空){
49.         当前项目 = 栈.pop()
50.         项目.add(当前项目);
51.         点后符号 = 当前项目的点后的第一个符号
52.         if(点后符号是非终结符){
53.             for(当前产生式:每个左边是 B 的产生式){
54.                 for(symbol:First(B 之后的部分,当前项目的展望符)){
55.                     new 新项目(当前产生式,0,symbol);
56.                     if(项目原本不存在项目集中){
57.                         栈.push(新项目)
58.                     }
59.                 }
60.             }
61.         }
```

```
    }
}
Go(项目集,符号){
    if(项目集通过(符号跳转) != -1){
        return 项目集通过(符号跳转)的项目集
    }
    else
        return 空集
}
创建 Action 和 Goto 表(){
    for(当前项目集:所有项目集){
        for( 当前项目 : 所有项目 ){
            当前项目 刑如 [A->...•a...,b]
            if(a 是终结符){
                ActionTable(当前项目.index,a) = S + Go(当前项目集,a).index
            }
            if(a 是""){
                ActionTable(当前项目.index,b) = R + 当前项目集.indexOf 语法
            }
        }
    }
    ActionTable(初始项目.index,#) = acc
    for(当前项目集:所有项目集){
        for(当前符号:所有符号){
            Goto(当前项目集,当前符号) = Go(当前项目集,当前符号)
        }
    }
    其他位置标记为 err
}
主控函数(){
    初始化输入串栈
    初始化符号栈
    初始化状态栈
    while(结束标记不为结束){
        String i =  状态栈顶
        String a = 输入串栈顶
        String action = ActionTable.(i,a)
        if(action() == null 或者 er ){
            报错
            标记结束
        }
        else if(action == acc ){
            成功
            标记结束
        }
        else if(action == Si){
            i 入状态栈
            a 到文法符号栈
        }
        else if(action == Ri){
            用 index 产生式规约
            符号栈中取出文法的右侧的长度的字符
            再压栈文法左边的符号
            状态栈.push( GOTO ( 状态栈.top() , 符号栈.top() ) )
        }
        else{
            报错
            标记结束
        }
    }
}
```

2.算法流程图

LR分析器结构：



3.源码及测试结果

Main.java：

```
1.  package 实验三_LR1 分析法;
2.
```

```java
public class Main {
    public static void main(String[] args) {
        new Thread(() -> new Windows()).start();
    }
}
```

Solution.java

```java
package 实验三_LR1 分析法;


import java.util.*;
import java.util.stream.Collectors;

class Solution {
    public String grammarText;
    public List<String[]> grammar = new ArrayList<>();//(T->SF)= {T,SF} forAll
    public Set<String> nonTerminal = new HashSet<>();//非终结符
    public Set<String>  terminal = new HashSet<>();//终结符
    public Set<String> allSymbol = new HashSet<>();//全部符号
    public Map<String,Set<String>> First = new HashMap<>();
    public List<project> projectList = new ArrayList<>();//所有项目
    public List<projectSet> cProjectSets = new ArrayList<>();//项目集 C
    public Map<point,String> ActionTable = new HashMap<>();//action 表
    public Map<point,String> GOTO = new HashMap<>();//goto 表
    public String[][] ActionAndGoTo;//Action 和 GOTO 表
    public String startSymbol = "S`";
    class project{//项目  A→α·Bβ,a   grammar: A→αBβ  index: location of(.)  extSymbol: a
        private int indexOfGrammar;
        private int indexOfNode;
        private String extSymbol;
        project(int indexOfGrammar,int index,String extSymbol){
            this.indexOfGrammar = indexOfGrammar;
            this.indexOfNode = index;
            this.extSymbol = extSymbol;
        }
        public String[] getGrammar() {//获取产生式
            return grammar.get(indexOfGrammar);
        }
        public int getIndex() {//获取所用产生式的编号
            return indexOfNode;
        }
        public String getHead(){//获取产生式左边
            return this.getGrammar()[0];
        }
        public String getExtSymbol() {
            return extSymbol;
        }
        public String getRight(){//获取产生式右边点后面的部分
            return this.getGrammar()[1].substring(indexOfNode);
        }
        public String getFirstSymbolAfterNode(){//获取右侧字符串的首字符
            if(this.getRight().length()<1){
                return "";
            }
            else {
                return this.getRight().substring(0,1);
            }
        }
        public String getRestStringAfterFirst(){//获取右侧首个字符的之后的字符串
            if(this.getRight().length()<2){
                return "";
            }
            else {
                return this.getRight().substring(1);
            }
        }
        @Override
        public String toString() {
            StringBuffer str = new StringBuffer(grammar.get(indexOfGrammar)[1]);
            str.insert(indexOfNode,".");
            str.insert(0,"["+grammar.get(indexOfGrammar)[0]+"->");
            str.append(","+extSymbol+"]");
            return str.toString();
        }
        @Override
        public boolean equals(Object obj) {
            if(!obj.getClass().equals(this.getClass())){
                return false;
            }
            project cmp = ((project)obj);
```

```
74.             if(  (cmp.indexOfGrammar == this.indexOfGrammar)  &&  (cmp.indexOfNode==this.in
    dexOfNode)  && (cmp.extSymbol.equals(this.extSymbol)) ){
75.                 return true;
76.             }
77.             return false;
78.         }
79.         @Override
80.         public int hashCode() {
81.             String hash = indexOfGrammar+","+indexOfNode+","+extSymbol;
82.             return hash.hashCode();
83.         }
84.     }//项目
85.     class projectSet {//项目集(闭包)  重写了 equals 和 hashcode 再加上是 set 存储 判断是不是生成了
    重复的对象
86.         Set<project> projects;//集合
87.         Set<Integer> indexOfProjects;//
88.         Map<String,Integer> sons;//孩子 S  即 通过 String 可以跳转到 Integer 下标的另一个集合
89.         public projectSet(){
90.             projects = new HashSet<>();
91.             indexOfProjects = new HashSet<>();
92.             sons = new HashMap<>();
93.         }
94.         public boolean add(project project){
95.             if(projectList.contains(project)){
96.                 indexOfProjects.add(projectList.indexOf(project));
97.                 return projects.add(project);
98.             }
99.             else {
100.                return false;
101.            }
102.        }
103.        public Set<project> getSet() {
104.            return projects;
105.        }
106.        private String toCompare(){
107.            StringBuffer sb = new StringBuffer();
108.            projects.forEach(S->sb.append(S.toString()));
109.            return sb.toString();//比较用  set 相同即可认为是相同的集合
110.        }
111.        @Override
112.        public String toString() {
113.            return cProjectSets.indexOf(this)+"";
114.        }
115.        @Override
116.        public boolean equals(Object obj) {
117.            if(!obj.getClass().equals(this.getClass())){
118.                return false;
119.            }
120.            return this.toCompare().equals(((projectSet)obj).toCompare()) ;
121.        }
122.        @Override
123.        public int hashCode() {
124.            return this.toCompare().hashCode();
125.        }
126.    }
127.    public Solution(String text){
128.        grammarText = text;
129.        setGrammar(text);
130.        setFirst();
131.        setProjectList();
132.        setCanonicalCollection();
133.        setActionAndGOTOTable();
134.        System.out.println("跳转表");
135.        for (projectSet projectSet : cProjectSets) {
136.            System.out.println(cProjectSets.indexOf(projectSet));
137.            projectSet.getSet().forEach(System.out::print);
138.            System.out.println("\n");
139.            projectSet.sons.keySet().forEach(S-> System.out.print("["+S+"=>"+projectSet.son
    s.get(S)+"]"+"  "));
140.            System.out.println("\n\n");
141.        }
142.    }
143.    private void setGrammar(String text){
144.        for (String s : text.replaceAll(" ","").split("\n")) {
145.            for (String s1 : s.split("->")[1].split("\\|")) {
146.                String [] gram = {s.split("->")[0],s1};
147.                grammar.add(gram);
148.            }
149.        }
150.        for (String[] strings : grammar) {
151.            System.out.println(strings[0]+"->"+strings[1]);
152.        }//读取文法
153.        System.out.println();
154.        for (String[] strings : grammar) {
155.            nonTerminal.add(strings[0]);
156.            terminal.addAll(Arrays.asList(strings[1].split("")));
157.        }
```

```
        nonTerminal.forEach(S->terminal.remove(S));

        allSymbol.addAll(terminal);
        allSymbol.addAll(nonTerminal);
        System.out.println("非终结符"+nonTerminal);
        System.out.println("终结符"+terminal);
        System.out.println();
    }
    private void setFirst(){
        nonTerminal.forEach(S->First.put(S,new HashSet<>()));
        terminal.forEach(S->First.put(S,new HashSet<>()));
        terminal.forEach(S->First.get(S).add(S));//终结符的 First 集是本身
        int FirstSize = 0;
        do{
            FirstSize = 0;
            for (String s1 : First.keySet()) {
                FirstSize+=First.get(s1).size();
            }//记录原本大小
            for (String[] strings : grammar) {
                String lam = strings[1];
                String G = strings[0];
                setSingleFirst(lam,G);//计算 First 集过程
            }
            for (String s1 : First.keySet()) {
                FirstSize-=First.get(s1).size();
            }//计算修改后大小
        }while (FirstSize != 0);//如果大小不在变化 则停下
        System.out.println("First");
        for (String s : First.keySet()) {
            System.out.println(s+":"+First.get(s));
        }
    }
    private void setSingleFirst(String lam,String G){
        String first = lam.substring(0,1);
        if(terminal.contains(first)){//终结符
            First.get(G).add(first);
        }
        else if(first.equals("ε")){//符号空
            First.get(G).add("ε");
        }
        else if(nonTerminal.contains(first)){//非终结符
            First.get(G).addAll(First.get(first).stream().filter(S->!S.equals("ε")).collect
(Collectors.toSet()));
            if(First.get(first).contains("ε")){//是否可以推出空
                setSingleFirst(lam.substring(1),G);//扫描下一个
            }
        }
        else {
            System.out.println("ERROR");
        }
    }
    private void setProjectList(){
        projectList.add(new project(0,0,"#"));
        projectList.add(new project(0,1,"#"));//开始符号 特殊  手动添加
        for (int indexOfGrammar = 0; indexOfGrammar < grammar.size(); indexOfGrammar++) {
            for (int indexOfNode = 0; indexOfNode <= grammar.get(indexOfGrammar)[1].length(
) ; indexOfNode++) {
                String A = grammar.get(indexOfGrammar)[0];//A->BC  A
                Set<String> a = new HashSet<>();
                for (String[] strings : grammar) {
                    if(strings[1].contains(A)){
                        int index = strings[1].indexOf(A)+1;
                        String sub = strings[1].substring(index);
                        a.addAll(First(sub));
                    }
                }
                for (String s : a) {
                    projectList.add( new project(indexOfGrammar,indexOfNode,s) );
                }
            }
        }
        System.out.println("项目 s");
        for (int i = 0; i < projectList.size(); i++) {
            System.out.println(i+" : " +projectList.get(i));
        }
    }//读取项目
    private projectSet extendSingleClosure(projectSet closure) {
        Stack<project> stack = new Stack<>();
        closure.getSet().forEach(stack::push);
        while(!stack.empty()){
            project top = stack.pop();
            String B = top.getFirstSymbolAfterNode();
            closure.projects.add(top);
            if(nonTerminal.contains(B)){//如果是 A->...•B...
                List<String[]> GsHeadIsB =  grammar.stream().filter(G->G[0].equals(B)).coll
ect(Collectors.toList());//每个左边是 B 的产生式
                String beta = top.getRestStringAfterFirst();//获得 B 之后的部分
```

```
                String a = top.getExtSymbol();//产生式之后的符号
                for (String[] strings : GsHeadIsB) {
                    for (String b : First(beta, a)) {
                        project newProject = new project(grammar.indexOf(strings),0,b);
                        if(!closure.getSet().contains(newProject)){
                            stack.push(newProject);
                        }
                    }
                }
            }
        }

        return closure;
    }//传入一个非空的项目集 将其扩充到不改变大小为止  返回此项目集
    private void setCanonicalCollection(){
        System.out.println("创建集合 C:");
        projectSet StartI = new projectSet();
        for (project project : projectList) {
            if(project.getGrammar()[0].equals(startSymbol) && project.getIndex()==0){
                StartI.add(project);
            }
        }
        extendSingleClosure(StartI);
        Stack<projectSet> stack = new Stack();
        stack.push(StartI);
        cProjectSets.add(StartI);
        while (!stack.empty()){//DFS 顺序去创建
            projectSet current = stack.pop();
            //System.out.println(">>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>");
            //System.out.print(current+" : ");
            //System.out.println(current.getSet().toString());
            for (String s : allSymbol) {
                List<project> temp = new ArrayList<>();
                for (project project : current.getSet()) {
                    if(project.getFirstSymbolAfterNode().equals(s)){//对于每个形
如 A->...•X... 的项目
                        //if(s.equals("L"))
                        //System.out.println("使用 : "+project  +  "  s = "+ s);
                        project newProject = new project(project.indexOfGrammar,project.ind
exOfNode+1,project.getExtSymbol());
                        //if(s.equals("L"))
                        //System.out.println("得到了 : "+newProject);
                        temp.add(newProject);
                    }
                }
                //System.out.println("最终 list : "+temp);

                projectSet nextI = new projectSet();
                nextI.projects.addAll(temp);
                extendSingleClosure(nextI);
                //System.out.println("最终 set "+nextI.getSet());

                if(nextI.getSet().size()==0){
                    current.sons.put(s,-1);
                    continue;
                }
                if(cProjectSets.contains(nextI)){
                    current.sons.put(s,cProjectSets.indexOf(nextI));
                    continue;
                }
                stack.push(nextI);
                Integer index = cProjectSets.size();
                cProjectSets.add(nextI);
                current.sons.put(s,index);
            }
            //System.out.println(">>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>");
        }
    }//设定项目集 C
    private projectSet GO(projectSet I,String X ){//如果 sons 可以得出 则直接返回结果 结果应当是
扩充过的  如果不能得出 则去计算 返回的也是计算过闭包的

        if(I.sons.containsKey(X) && I.sons.get(X)!=-1){
            return cProjectSets.get(I.sons.get(X));
        }
       return new projectSet();
    }
    private void setActionAndGOTOTable(){
        for (projectSet projectSet : cProjectSets) {
            for (project project : projectSet.getSet()) {
                String a = project.getFirstSymbolAfterNode();
                if(terminal.contains(a)){//项目 [A->...•a...,b] a 是终结符
                    ActionTable.put(new point(projectSet.toString(),a),"s"+GO(projectSet,a)
.toString());
                }
            }
        }//<1>
        for (projectSet projectSet : cProjectSets) {
```

```
            for (project project : projectSet.getSet()) {
                if(project.getFirstSymbolAfterNode().equals("")){
                    ActionTable.put(new point(projectSet.toString(),project.getExtSymbol())
,"r"+project.indexOfGrammar);
                }
            }
        }//<2>
        for (project project : projectList) {
            if(project.getHead().equals(startSymbol) && project.getFirstSymbolAfterNode().e
quals("")){
                for (projectSet projectSet : cProjectSets) {
                    if(projectSet.getSet().contains(project)){
                        int k = cProjectSets.indexOf(projectSet);
                        ActionTable.put(new point(String.valueOf(k),"#"),"acc");
                    }
                }
            }
        }//<3>
        for (String A : nonTerminal) {
            for (int k = 0; k < cProjectSets.size(); k++) {
                int j = cProjectSets.indexOf(GO(cProjectSets.get(k),A));
                if(j!=-1){
                    GOTO.put(new point(String.valueOf(k),A),String.valueOf(j));
                }
            }
        }//<4>
        int len = cProjectSets.size();//C 集的 SIZE
        for (int i = 0; i < len ; i++) {
            for (String s : nonTerminal) {
                if(!ActionTable.containsKey(new point(String.valueOf(i),s))){
                    ActionTable.put(new point(String.valueOf(i),s),"err");
                }
            }
            for (String s : terminal) {
                if(!ActionTable.containsKey(new point(String.valueOf(i),s))){
                    ActionTable.put(new point(String.valueOf(i),s),"err");
                }
            }
            if(!ActionTable.containsKey(new point(String.valueOf(i),"#"))){
                ActionTable.put(new point(String.valueOf(i),"#"),"err");
            }
        }//空位置打上 err
        Map<point,String> adder = new HashMap<>();//Action 和 GOTO 合并为一个 方便显示
        adder.putAll(ActionTable);
        adder.putAll(GOTO);
        List<String> tableSymbol = new ArrayList<>();
        tableSymbol.addAll(terminal);
        tableSymbol.add("#");//加上#
        tableSymbol.addAll(nonTerminal);
        tableSymbol.remove(startSymbol);//删掉 S`
        ActionAndGoTo = new String[len][tableSymbol.size()+1];
        for (int i = 0; i < len ; i++) {
            ActionAndGoTo[i][0] = String.valueOf(i);
            for (int j = 0; j < tableSymbol.size() ; j++) {
                ActionAndGoTo[i][j+1] = adder.get(new point(String.valueOf(i),tableSymbol.g
et(j)));
            }
        }
        System.out.print("\t");
        for (String s : tableSymbol) {
            System.out.print(s+"\t");
        }
        System.out.println();
        for (String[] strings : ActionAndGoTo) {
            for (String string : strings) {
                System.out.print(string+"\t");
            }
            System.out.println();
        }
    }//创建 Action 和 GOTO 表
    public String[][] getActionAndGoTo() {//返回分析表
        return ActionAndGoTo;
    }
    public String[] getHeader(){
        List<String> tableSymbol = new ArrayList<>();
        tableSymbol.addAll(terminal);
        tableSymbol.add("#");//加上#
        tableSymbol.addAll(nonTerminal);
        tableSymbol.remove(startSymbol);//删掉 S`
        String[] header = new String[tableSymbol.size()+1];
        header[0] = "";
        for (int i = 0; i < tableSymbol.size(); i++) {
            header[i+1] = tableSymbol.get(i);
        }
        return header;
    }
    public Vector<String[]> analyse(String text){
```

```
409.         Vector<String[]> processRecord = new Vector<>();
410.         MyStack inputStack = new MyStack();//输入串
411.         MyStack symbolStack = new MyStack();//符号栈
412.         MyStack statusStack = new MyStack();//状态栈
413.         inputStack.push("#");
414.         inputStack.push(new StringBuffer(text).reverse().toString().split(""));
415.         symbolStack.push("#");
416.         statusStack.push("0");
417.         Boolean iFlag = true;
418.         int count = 0;
419.         System.out.println("分析开始====================");
420.         while (iFlag){
421.             String[] currentStep = new String[5];
422.             currentStep[1] = statusStack.toString()+"\t";
423.             currentStep[2] = symbolStack.toString()+"\t";
424.             currentStep[3] = new StringBuffer(inputStack.toString()).reverse()+"\t";
425.             processRecord.add(currentStep);
426.             String i =  statusStack.getTop();//状态栈
427.             String a = inputStack.getTop();//输入串
428.             System.out.println("状态栈顶:["+i+"]");
429.             System.out.println("输入栈顶:["+a+"]");
430.             String action = ActionTable.get(new point(i,a));
431.             if(action == null){
432.                 System.out.println("nullActErr");
433.                 currentStep[4] = "nullActErr";
434.                 iFlag = false;
435.             }
436.             else if(action.equals("err")){
437.                 System.out.println("EqualsErr");
438.                 currentStep[4] = "EqualsError";
439.                 iFlag = false;
440.             }
441.             else if(action.equals("acc")){
442.                 System.out.println("成功!");
443.                 currentStep[4] = "成功";
444.                 iFlag = false;
445.             }
446.             else if(action.substring(0,1).equals("s")){//状态入栈
447.                 Integer j = Integer.valueOf(action.substring(1));
448.                 System.out.println("当前 S"+j);
449.                 statusStack.push(String.valueOf(j));//j 入状态栈
450.                 symbolStack.push(inputStack.pop());//a 到文法符号栈
451.                 currentStep[4] = "Action["+i+","+a+"]="+action+",状态"+a+"入栈";
452.             }
453.             else if(action.substring(0,1).equals("r")){//规约 然后 GOTO 入栈
454.                 Integer index = Integer.valueOf(action.substring(1));//用 index 产生式规约
455.                 for (int times = 0; times < grammar.get(index)[1].length(); times++) {
456.                     symbolStack.pop();
457.                     statusStack.pop();
458.                 }//符号栈中取出文法的右侧的长度的字符
459.                 symbolStack.push(grammar.get(index)[0]);//再压栈文法左边的符号
460.
461.                 String nextSta = GOTO.get(new point(statusStack.getTop(),symbolStack.getTop
    ()));
462.                 currentStep[4] = action+":"+grammar.get(index)[0]+"->"+grammar.get(index)[1
    ]+"归约,GOTO("+statusStack.getTop()+","+statusStack.getTop()+")="+nextSta+"入栈";
463.                 statusStack.push(nextSta);
464.             }
465.             else {
466.                 System.out.println("err");
467.                 currentStep[4] = "ERROR";
468.                 iFlag = false;
469.             }
470.             currentStep[0] = count+"\t";
471.             count++;
472.
473.
474.             System.out.println("================================");
475.         }
476.         return processRecord;
477.     }
478.     Set<String> First(String beta,String a){
479.         Set<String> res = new HashSet<>();
480.         if(beta.length() == 0){
481.             res.add(a);
482.             return res;
483.         }
484.         else {
485.             res.addAll(First(beta));
486.             if(First(beta).contains("ε")){
487.                 res.add(a);
488.                 res.remove("ε");
489.             }
490.         }
491.         return res;
492.     }
493.     Set<String> First(String s){//单个字串的 First 集
```

```
494.         Set<String> res = new HashSet<>();
495.         if(s.length() == 0){
496.             res.add("#");
497.             return res;
498.         }
499.         for (String symbol : s.split("")) {
500.             if(First.containsKey(symbol)){
501.                 res.addAll(First.get(symbol));
502.                 if(!First.get(symbol).contains("ε")){//如果 当前的字 可以推出空 看向字串的下一个字
503.                     break;
504.                 }
505.             }
506.         }
507.         Boolean elicitNull = true;
508.         for (String symbol : s.split("")){
509.             if(First.containsKey(symbol)  && !First.get(symbol).contains("ε")){
510.                 elicitNull = false;
511.                 break;
512.             }
513.         }
514.         if(!elicitNull){
515.             res.remove("ε");
516.         }//只有 所有的字 都能推出空 这个字串才可以推出空
517.         return res;
518.     }
519. }
520.
521. class point {
522.     String head, tail;
523.
524.     point(String head, String tail) {
525.         this.head = head;
526.         this.tail = tail;
527.     }
528.
529.     public String getHead() {
530.         return head;
531.     }
532.
533.     public String getTail() {
534.         return tail;
535.     }
536.
537.     @Override
538.     public int hashCode() {
539.         return (head + "->" + tail).hashCode();
540.     }
541.
542.     @Override
543.     public boolean equals(Object o) {
544.         if (o.getClass() != this.getClass()) {
545.             return false;
546.         } else {
547.             return ((point) o).getHead().equals(this.head) && ((point) o).getTail().equals(this.tail);
548.         }
549.     }
550.
551. }
552.
553. class MyStack {
554.     List<String> s;
555.
556.     MyStack() {
557.         s = new LinkedList<>();
558.     }
559.
560.     void push(String value) {
561.         s.add(value);
562.     }
563.
564.     void push(String... values) {
565.         for (String value : values) {
566.             push(value);
567.         }
568.     }
569.
570.     String pop() {
571.         return s.remove(s.size() - 1);
572.     }
573.
574.     String getTop() {
575.         return s.get(s.size() - 1);
576.     }
577.
578.     @Override
```

```
    public String toString() {
        StringBuffer sb = new StringBuffer();
        for (String value : s) {
            sb.append(value);
        }
        return sb.toString();
    }

    public Boolean isEmpty() {
        return s.size() == 0;
    }
}
```

GUI.java

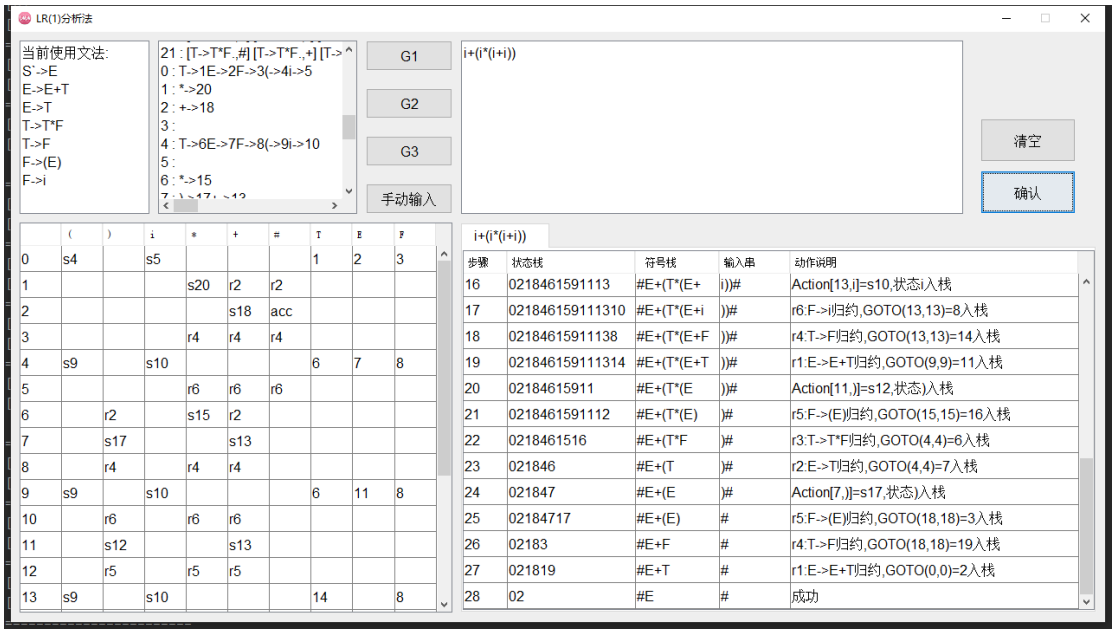
```
package 实验三_LR1 分析法;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Vector;
import javax.swing.table.JTableHeader;
import javax.swing.table.TableColumn;
import java.util.*;

class Windows extends JFrame {
    Solution sol;
    JButton clear,confirm;
    JTextArea grammarTextArea,inputTextArea,projectArea;
    JScrollPane tablePane;
    JTabbedPane resultTable;
    JScrollPane projectList;

    Windows(){
        setVisible(false);
        try{
            setIconImage(new ImageIcon("bilibili.PNG").getImage());
            Font f = new Font("Yahei Consolas Hybrid",Font.PLAIN,16);
            String   names[]={ "MenuBar","Menu","MenuItem", "TextArea", "Button", "ScrollPane", "Table","TabbedPane"};
            for (String item : names) {
                UIManager.put(item+ ".font",f);
            }
            UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");

        }catch(Exception e){}
        sol = new Solution("S`->E\n"+
                "E->E+T\n" +
                "E->T\n" +
                "T->T*F\n" +
                "T->F\n" +
                "F->(E)\n" +
                "F->i");
        init();
        setSize(1280,720);//初始大小
        setLocation(100,80);//初始位置
        setVisible(true);//是否可视
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);//X 退出
    }
    void init(){
        setTitle("LR(1)分析法");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setVisible(true);
        setResizable(false);
        setLayout(null);
        initGrammarText();
        initActTable();
        initButton();
        initInputArea();
        initResultTable();
        initProjectPane();
    }
    void initGrammarText(){
        grammarTextArea = new JTextArea("当前使用文法:\n"+sol.grammarText);
        grammarTextArea.setEditable(false);
        JScrollPane textAreaRollPane = new JScrollPane(grammarTextArea);
        textAreaRollPane.setBounds(10,10,150,200);
        add(textAreaRollPane);
```

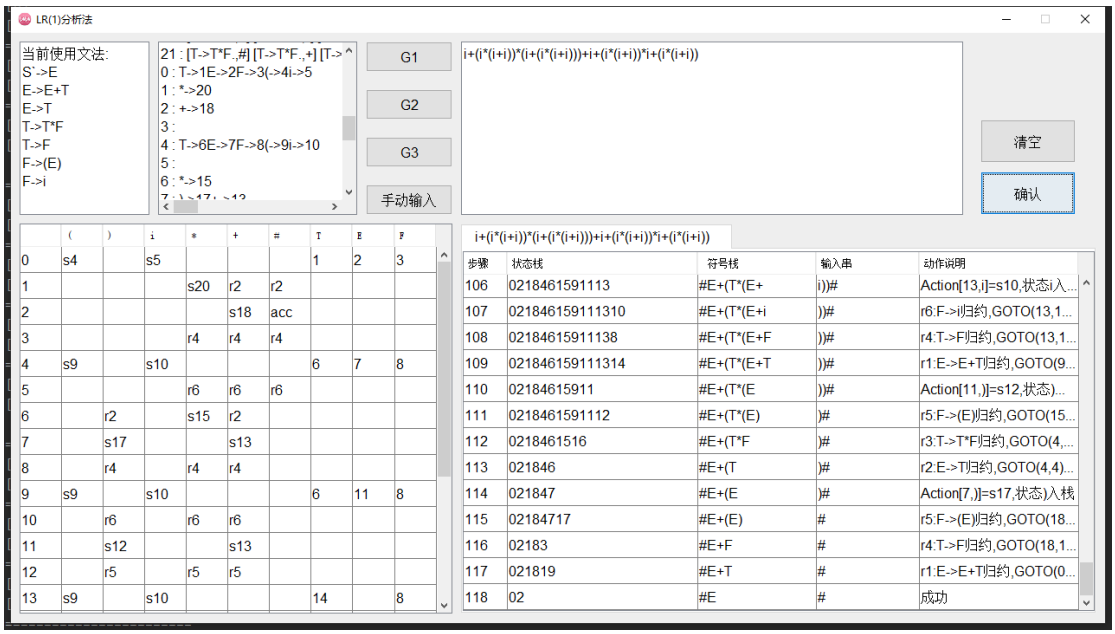
```
    }
    private void initActTable(){
        tablePane = new JScrollPane();
        tablePane.setBounds(10,220,500,450);
        add(tablePane);
        updateActTable();
    }
    private void initButton(){
        clear = new JButton("清空");
        clear.setBounds(1120,100,110,50);
        add(clear);
        clear.addActionListener(actionEvent -> {
            inputTextArea.setText("");
            resultTable.removeAll();
        });
        confirm = new JButton("确认");
        confirm.setBounds(1120,160,110,50);
        add(confirm);
        confirm.addActionListener(actionEvent -> {
            resultTable.removeAll();
            for (String inputText : inputTextArea.getText().split("\n")) {
                Vector<String[]> result =  sol.analyse(inputText);
                String[] head = {"步骤 ","状态栈"," 符号栈 ","输入串 ","动作说明 "};
                String[][] data = new String[result.size()][5];
                int i = 0;
                for (String[] strings : result) {
                    data[i] = strings;
                    i++;
                }
                JTable singleResult = new JTable(data,head);
                FitTableColumns(singleResult);
                singleResult.setRowHeight(30);
                JScrollPane resultTablePane = new JScrollPane(singleResult);
                resultTable.addTab("  "+inputText+"     ",resultTablePane);
            }
        });
        JButton G1 = new JButton("G1");
        G1.setBounds(410,10,100,35);
        add(G1);
        G1.addActionListener(actionEvent -> {
            updateGrammar("S`->E\nE->E+T\nE->T\nT->T*F\nT->F\nF->(E)\nF->i");
        });
        JButton G2 = new JButton("G2");
        G2.setBounds(410,65,100,35);
        add(G2);
        G2.addActionListener(actionEvent -> {
            updateGrammar("S`->E\nE->E+T | T\nT->T*F | F\nF->P↑F | P\nP->(E) | i\n");
        });
        JButton G3 = new JButton("G3");
        G3.setBounds(410,120,100,35);
        add(G3);
        G3.addActionListener(actionEvent -> {
            updateGrammar("S`->S\nS->aAd\nS->bAc\nS->aec\nS->bed\nA->e");
        });
        JButton more = new JButton("手动输入");
        more.setBounds(410,175,100,35);
        add(more);
        class MyDialog extends JDialog implements ActionListener{
            JTextArea input;
            JButton confirm,cancel;
            String title;
            MyDialog(){
                setLayout(null);
                setResizable(false);
                setIconImage(new ImageIcon("bilibili.PNG").getImage());
                setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
                setTitle("输入语法");
                input=new JTextArea();
                JScrollPane jScrollPane = new JScrollPane(input);
                jScrollPane.setBounds(10,10,265,200);
                add(jScrollPane);
                class confirmListener implements ActionListener{
                    @Override
                    public void actionPerformed(ActionEvent e){
                        updateGrammar(input.getText());
                        setVisible(false);
                    }
                }
                confirm=new JButton("确定");
                confirm.addActionListener(new confirmListener());
                confirm.setBounds(195,220,80,30);
                add(confirm);
                class cancelListener implements ActionListener{
                    @Override
                    public void actionPerformed(ActionEvent e){
                        setVisible(false);
                    }
```

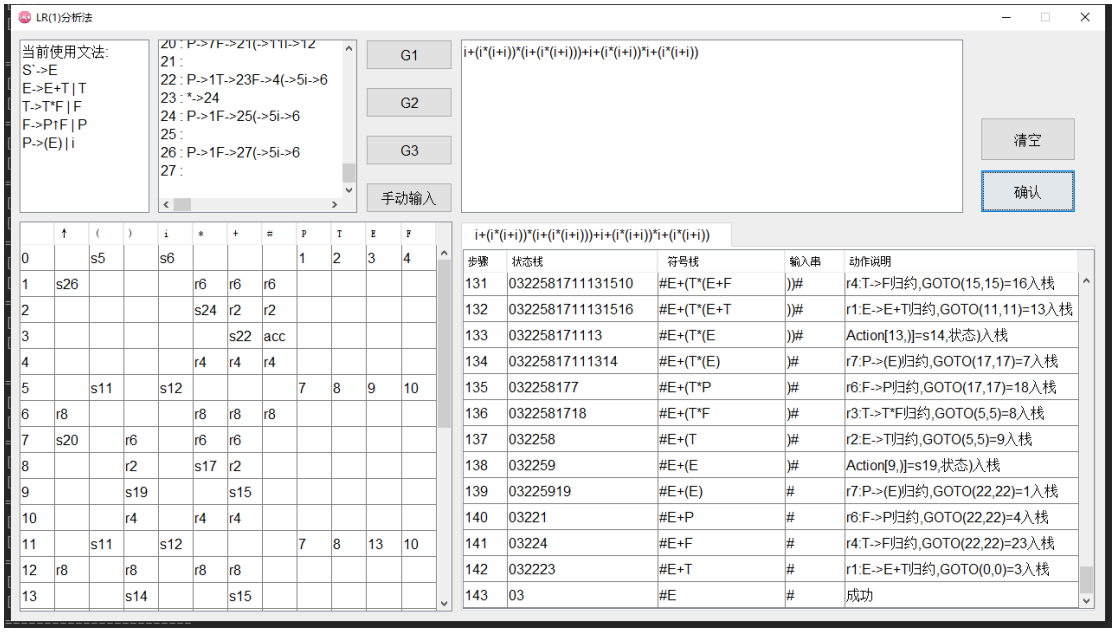
```
                }
                cancel=new JButton("取消");
                cancel.addActionListener(new cancelListener());
                cancel.setBounds(105,220,80,30);
                add(cancel);
                setBounds(600,260,300,300);
                setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
            }
            public void actionPerformed(ActionEvent e){
                setVisible(true);
            }
        }
        more.addActionListener(new MyDialog());
    }
    private void initInputArea(){
        inputTextArea = new JTextArea();
        inputTextArea.setLineWrap(true);
        JScrollPane inputAreaPane = new JScrollPane(inputTextArea);
        inputAreaPane.setBounds(520,10,580,200);
        add(inputAreaPane);
    }
    private void initResultTable(){
        resultTable = new JTabbedPane();
        resultTable.setBounds(520,220,735,450);
        add(resultTable);
    }
    private void initProjectPane(){
        projectArea = new JTextArea();
        projectList = new JScrollPane(projectArea);
        projectList.setBounds(170,10,230,200);
        add(projectList);
        updateProjectPane();

    }
    private void updateProjectPane(){
        StringBuffer sb = new StringBuffer();
        int count = 0;
        for (Solution.projectSet projectSet : sol.cProjectSets) {
            sb.append( (count++ )+" : ");
            for (Solution.project project : projectSet.getSet()) {
                sb.append(project+" ");
            }
            sb.append("\n");
        }
           count = 0;
        for (Solution.projectSet projectSet : sol.cProjectSets) {
            sb.append( (count++ )+" : ");
            for (String s : sol.allSymbol) {
                int index = projectSet.sons.get(s);
                if(index !=-1){
                    sb.append(s+"->"+index);
                }
            }
            sb.append("\n");
        }
        projectArea.setText(sb.toString());
        projectList.updateUI();
    }
    private void updateActTable(){
        String[] head =sol.getHeader();
        String [][] data = sol.getActionAndGoTo();
        for (int i = 0; i < data.length; i++) {
            for (int j = 0; j < data[i].length; j++) {
                if(data[i][j].equals("err")){
                    data[i][j] = "";
                }
            }
        }
        JTable actGoTable;
        actGoTable = new JTable(data,head);
        actGoTable.setRowHeight(30);
        tablePane.setViewportView(actGoTable);
        tablePane.updateUI();
    }//更新 ActGo 表
    public void updateGrammar(String s){
        sol = new Solution(s);
        grammarTextArea.setText("当前使用文法:\n"+sol.grammarText);
        updateActTable();
        updateProjectPane();
    }//更新语法
    public void FitTableColumns(JTable myTable) {
        JTableHeader header = myTable.getTableHeader();
        int rowCount = myTable.getRowCount();
        Enumeration columns = myTable.getColumnModel().getColumns();
        while (columns.hasMoreElements()) {
            TableColumn column = (TableColumn) columns.nextElement();
            int col = header.getColumnModel().getColumnIndex(column.getIdentifier());
```

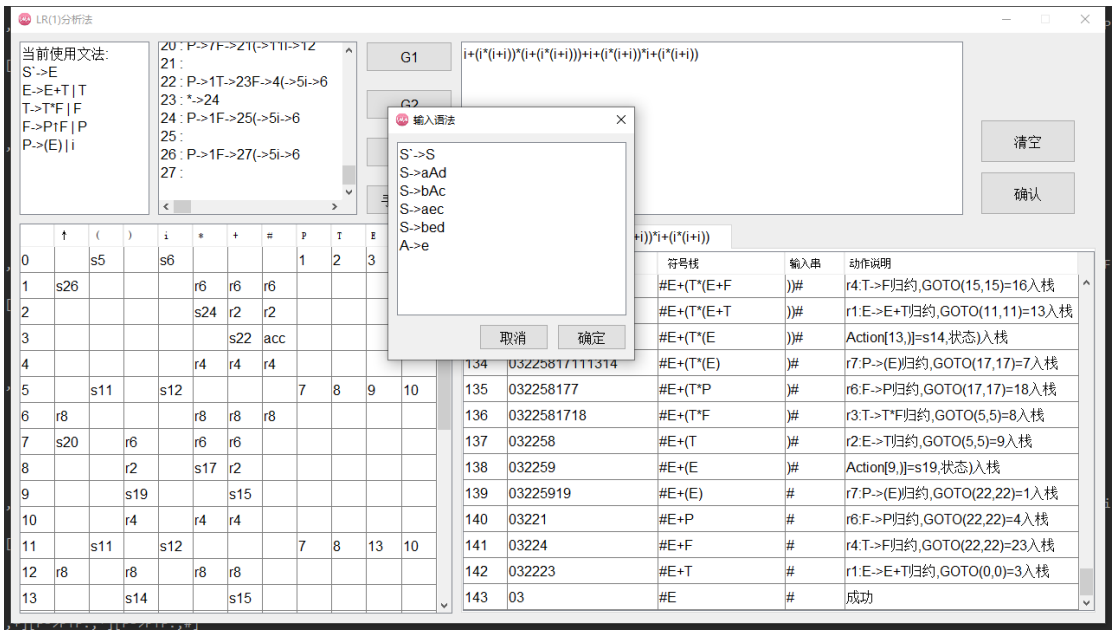
```
            int width = (int) myTable.getTableHeader().getDefaultRenderer()
                    .getTableCellRendererComponent(myTable, column.getIdentifier()
                            , false, false, -1, col).getPreferredSize().getWidth();
            for (int row = 0; row < rowCount; row++) {
                int preferedWidth = (int) myTable.getCellRenderer(row, col).getTableCellRen
dererComponent(myTable,
                        myTable.getValueAt(row, col), false, false, row, col).getPreferredS
ize().getWidth();
                width = Math.max(width, preferedWidth);
            }
            header.setResizingColumn(column);
            column.setWidth(width + myTable.getIntercellSpacing().width+10);
        }
    }
}
```

运行结果：






LR(1)分析法
当前使用文法:
S`->E
E->E+T | T
T->T*F | F
F->P↑F | P
P->(E) | i
20 : P->7F->21(->11T->12
21 :
22 : P->1T->23F->4(->5i->6
23 : *->24
24 : P->1F->25(->5i->6
25 :
26 : P->1F->27(->5i->6
27 :
G1
G2
G3
手动输入
i+(i*(i+i))*(i+(i*(i+i)))+i+(i*(i+i))*i+(i*(i+i))
清空
确认
步骤 状态栈 符号栈 输入串 动作说明
131 0322581711131510 #E+(T*(E+F))# r4:T->F归约,GOTO(15,15)=16入栈
132 0322581711131516 #E+(T*(E+T))# r1:E->E+T归约,GOTO(11,11)=13入栈
133 032258171113 #E+(T*(E))# Action[13,)]=s14,状态)入栈
134 03225817111314 #E+(T*(E))# r7:P->(E)归约,GOTO(17,17)=7入栈
135 032258177 #E+(T*P)# r6:F->P归约,GOTO(17,17)=18入栈
136 0322581718 #E+(T*F)# r3:T->T*F归约,GOTO(5,5)=8入栈
137 032258 #E+(T)# r2:E->T归约,GOTO(5,5)=9入栈
138 032259 #E+(E)# Action[9,)]=s19,状态)入栈
139 03225919 #E+(E) # r7:P->(E)归约,GOTO(22,22)=1入栈
140 03221 #E+P # r6:F->P归约,GOTO(22,22)=4入栈
141 03224 #E+F # r4:T->F归约,GOTO(22,22)=23入栈
142 032223 #E+T # r1:E->E+T归约,GOTO(0,0)=3入栈
143 03 #E # 成功


输入语法
S`->S
S->aAd
S->bAc
S->aec
S->bed
A->e
取消
确定

4.实验收获

相比较与前几次的实验，本次实验的算法相对复杂抽象，但是当理解算法并设计了合适的数据结构去存储后，各个部分功能明确，总体设计起来也就比较容易。项目集和项目都使用了重写了hashCode()和 equals()方法并使用 set 存储来更加方便的判断重复，在计算项目集闭包和展望符跳转的时候更加便利。此外，相比较于上次实验，本次分析器直接将主控程序与语法绑定在一起，在修改语法时就不需要额外的更新语法的方法，只需创建一个新的分析器即可，更新界面上的语法相关信息也更加方便。