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

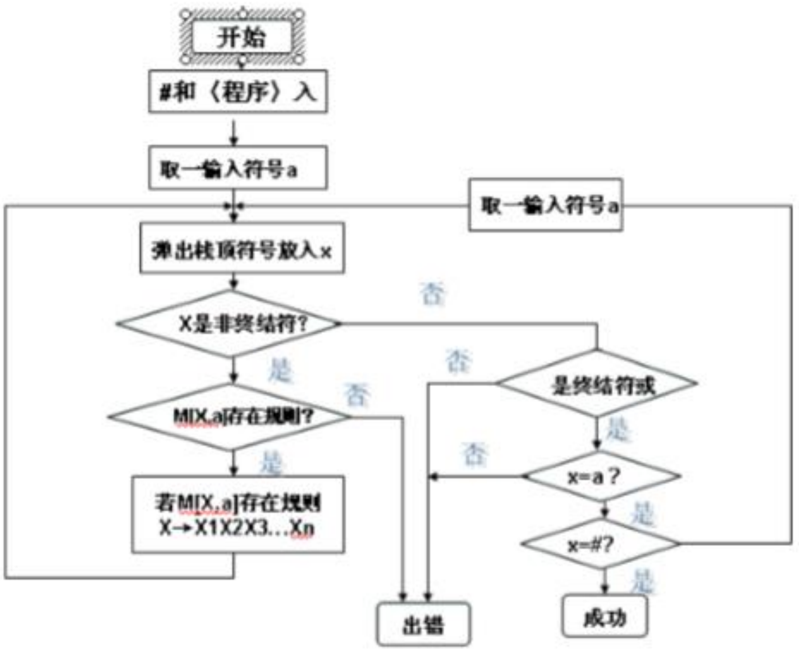
实验 2：LL(1)分析法

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) {
            setSelect(strings,0);   //创建 select 集  key = 产生式 value  = [] Set
        }
        AnaTable.clear();//清空
        terminal.add("#");//在终结符内加入#   以达到
        for (String s0 : select.keySet()) {
            for (String s1 : select.get(s0)) {
                AnaTable.put(s0.substring(0,1)+s1,s0.substring(1));
            }
        }
        this.AnaTable =AnaTable;
        return null;
    }
    Vector<String[]> Solve(String text){
        Vector<String[]> procesList = new Vector();
        int textLength = text.length();
        MyStack AnaStack = new MyStack();//分析栈
        MyStack inputString = new MyStack();//输入串
        AnaStack.push("#","E","S");
        inputString.push( new StringBuffer(text).reverse().toString().split("") );
        Boolean flag = true;
        Boolean matched = true;
        int linenumber = 0;
        while (flag){
            String[] INFO = new String[5];
            INFO[0] = String.valueOf(linenumber++);
            INFO[1] = AnaStack.toString();
            StringBuffer inputsb= new StringBuffer(inputString.toString());

            for(int sblength = inputsb.length() ; sblength < textLength+1 ; sblength++ ){
                inputsb.append(" ");
            }
            INFO[2] = inputsb.reverse().toString();
            x = AnaStack.getTop();//获取分析栈顶
            a = inputString.getTop();//第一个符号读到 a
            if(M(x,a)!=null){//存在对应的文法
                if(M(x,a)[0].equals("ε")){//为空
                    INFO[3] = x+" -> ε";
                    INFO[4] = "POP";
                    AnaStack.pop();
                }
                else {
                     AnaStack.push(M(AnaStack.pop(), a));
                     StringBuffer sb = new StringBuffer();
                     for(String string:M(x,a)){
                         sb.append(string);
                     }
                     INFO[3] = x+" -> " + sb.reverse();
                     INFO[4] = "POP,PUSH("+sb.reverse()+")";
                     //System.out.println("存在文法["+x+","+a+"] -> ["+sb + "]  STACK:" + AnaStack);
                 }
             }
            else if( !x.equals("#") && x.equals(a) ){//匹配到了
                //System.out.println("匹配到了"+x+" "+a);
                INFO[4] = "POP,GETNEXT(i)";
                inputString.pop();
                AnaStack.pop();
            }
            else if(x.equals("#") && x.equals(a)){//结束了
                flag=false;
            }
            else{//报错
                flag=false;
                matched=false;
            }
            for(int i = 0 ; i < 5 ; i++ ){
                if(INFO[i]!=null)
                    INFO[i] = " "+INFO[i];
                else
                    INFO[i] = " ";
            }

            procesList.add(INFO);
        }
        if(matched){
            //System.out.println("匹配成功");
        }
        else{
            //System.out.println("匹配失败");
            String ss[] ={"ERROR","ERROR","ERROR","ERROR","ERROR"};
            procesList.add(ss);
        }

//      for (String s : Grammer) {
//          System.out.print(s+" ");
//      }
        return procesList;
    }
    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");
        }
```

```java
        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

```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();
                }


            }
        }
```

```java
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 !", JOptionPane.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){
```

```java
                        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();
```

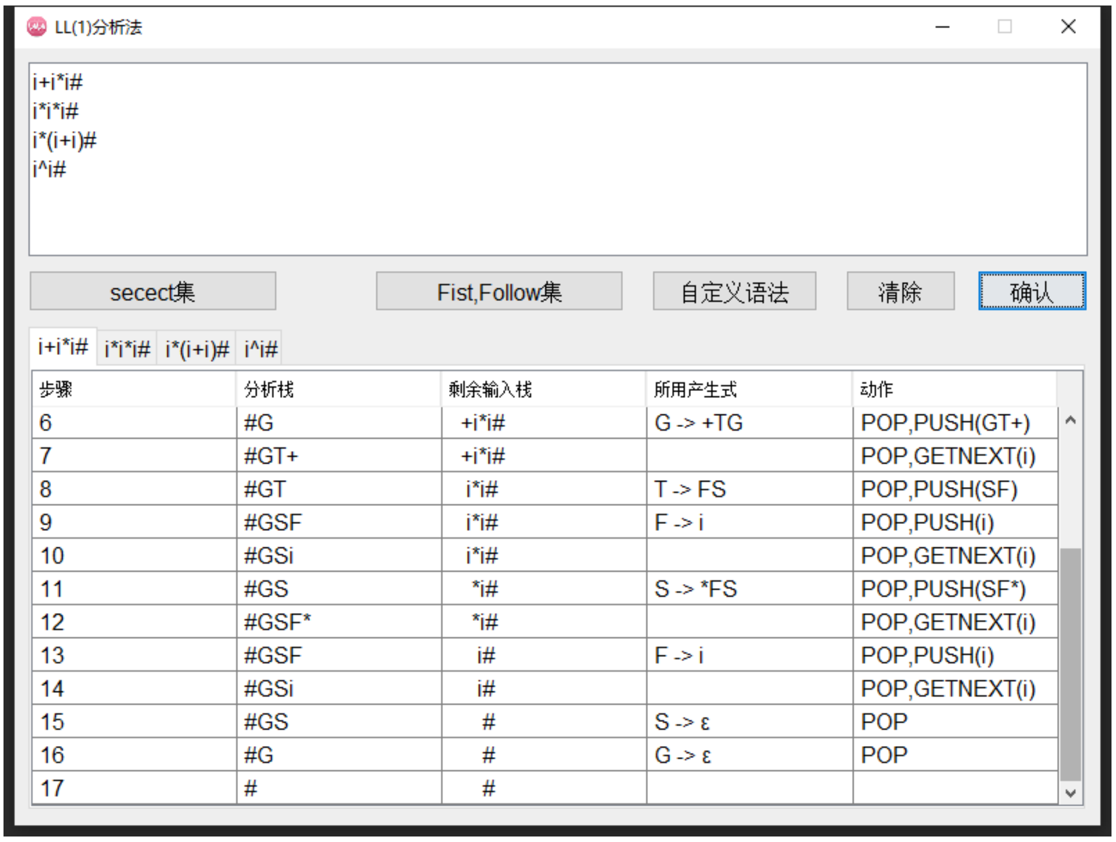
```

                    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);
```

```java
        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();
//                      }
```

```java


                        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);

                    }




                }
            }


            JButton select;
            select = new JButton("secect 集");
            select.setBounds(10,160,180,30);
            select.addActionListener(new selectListen());
            add(select);
```

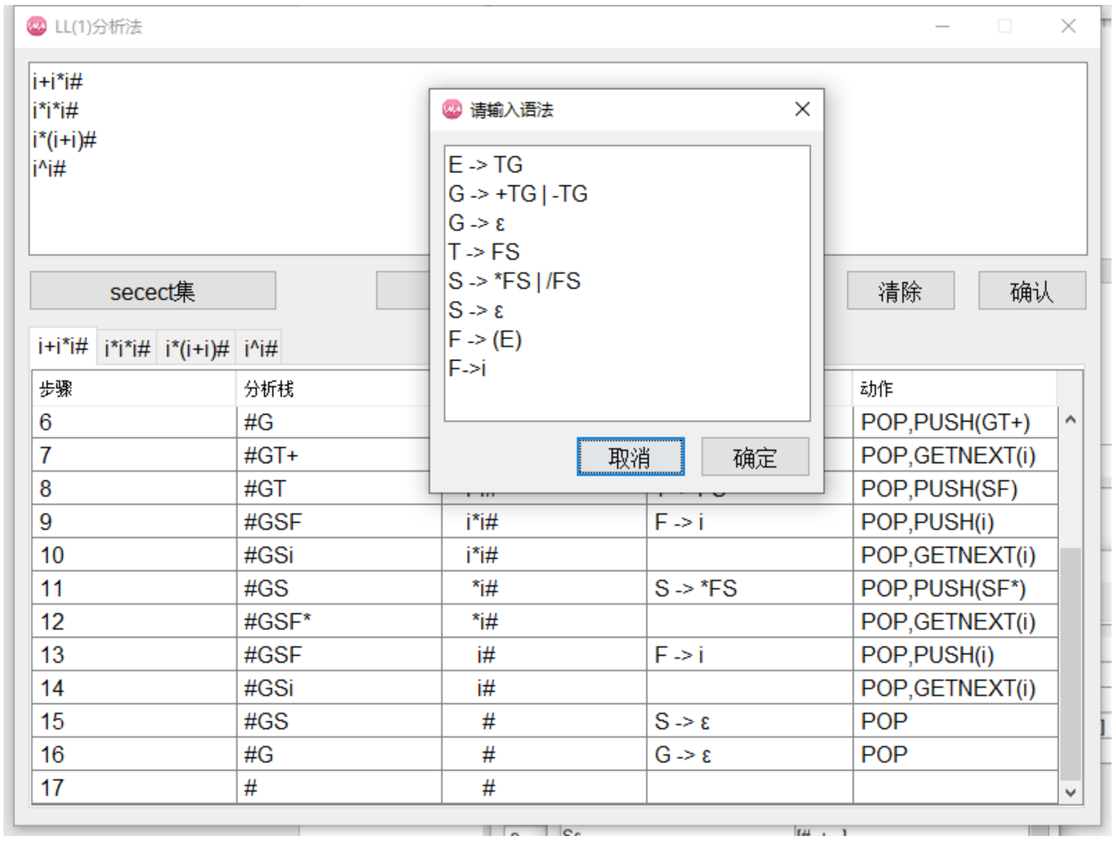
```java

        add(clear);
        add(confirm);
        add(setG);
        add(FF);
    }
    void initText(){
        textArea = new JTextArea("i+i*i#\ni*i*i#\ni*(i+i)#\ni^i#");
        JScrollPane textAreaRollPane = new JScrollPane(textArea);
        textAreaRollPane.setBounds(10,10,765,140);
        add(textAreaRollPane);
    }
    void initResult(){
        tabbedPane = new JTabbedPane();
        tabbedPane.setBounds(10, 200, 765, 350);
        add(tabbedPane);
    }
    void addTable(String title,Vector vec ){
        TableDataModel tableDataModel = new TableDataModel(vec);
        JTable table = new JTable(tableDataModel);
        table.setVisible(true);
        table.setPreferredScrollableViewportSize(new Dimension(550, 100));

        table.setRowHeight(24);
        JScrollPane tablePane = new JScrollPane(table);
        tablePane.setBounds(10, 200, 765, 350);
        tabbedPane.addTab(title,tablePane);
    }
}

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[] = {"步骤","分析栈","剩余输入栈","所用产生式","动作"};
        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：



```
i+i*i#
i*i*i#
i*(i+i)#
i^i#
```

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	#	#		

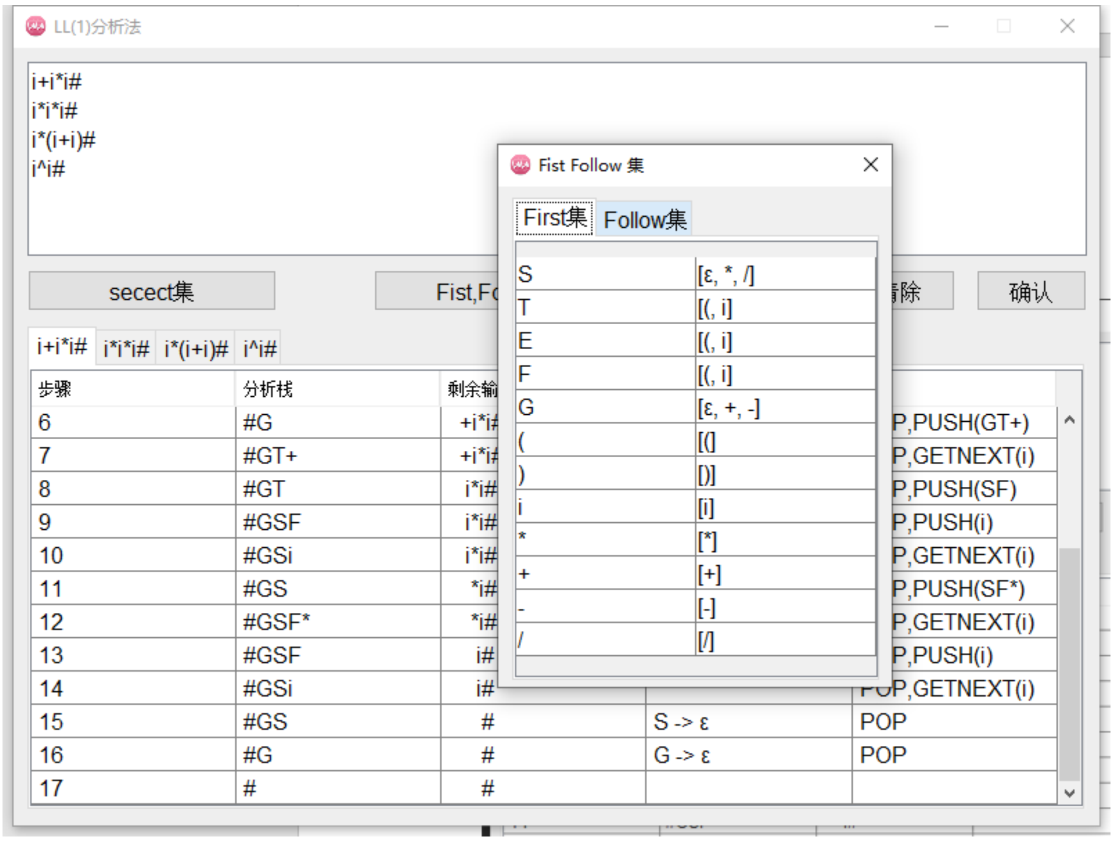


```
i+i*i#
i*i*i#
i*(i+i)#
i^i#
```

请输入语法

```
E -> TG
G -> +TG | -TG
G -> ε
T -> FS
S -> *FS | /FS
S -> ε
F -> (E)
F->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	#	#		



LL(1)分析法

```
i+i*i#
i*i*i#
i*(i+i)#
i^i#
```

secect集 | Fist,F | 除 | 确认

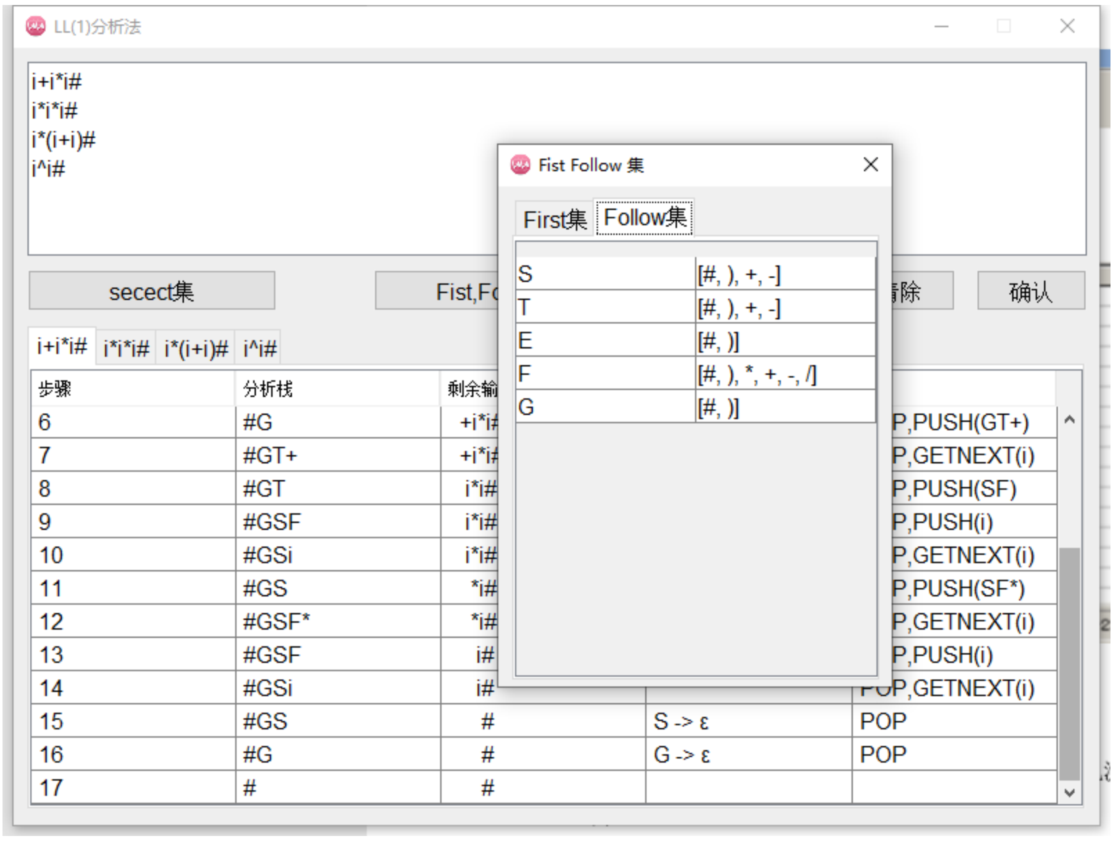
i+i*i# | i*i*i# | i*(i+i)# | i^i#

步骤	分析栈	剩余输		
6	#G	+i*i		P,PUSH(GT+)
7	#GT+	+i*i		P,GETNEXT(i)
8	#GT	i*i#		P,PUSH(SF)
9	#GSF	i*i#		P,PUSH(i)
10	#GSi	i*i#		P,GETNEXT(i)
11	#GS	*i#		P,PUSH(SF*)
12	#GSF*	*i#		P,GETNEXT(i)
13	#GSF	i#		P,PUSH(i)
14	#GSi	i#		POP,GETNEXT(i)
15	#GS	#	S -> ε	POP
16	#G	#	G -> ε	POP
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 | Fist,F | 除 | 确认

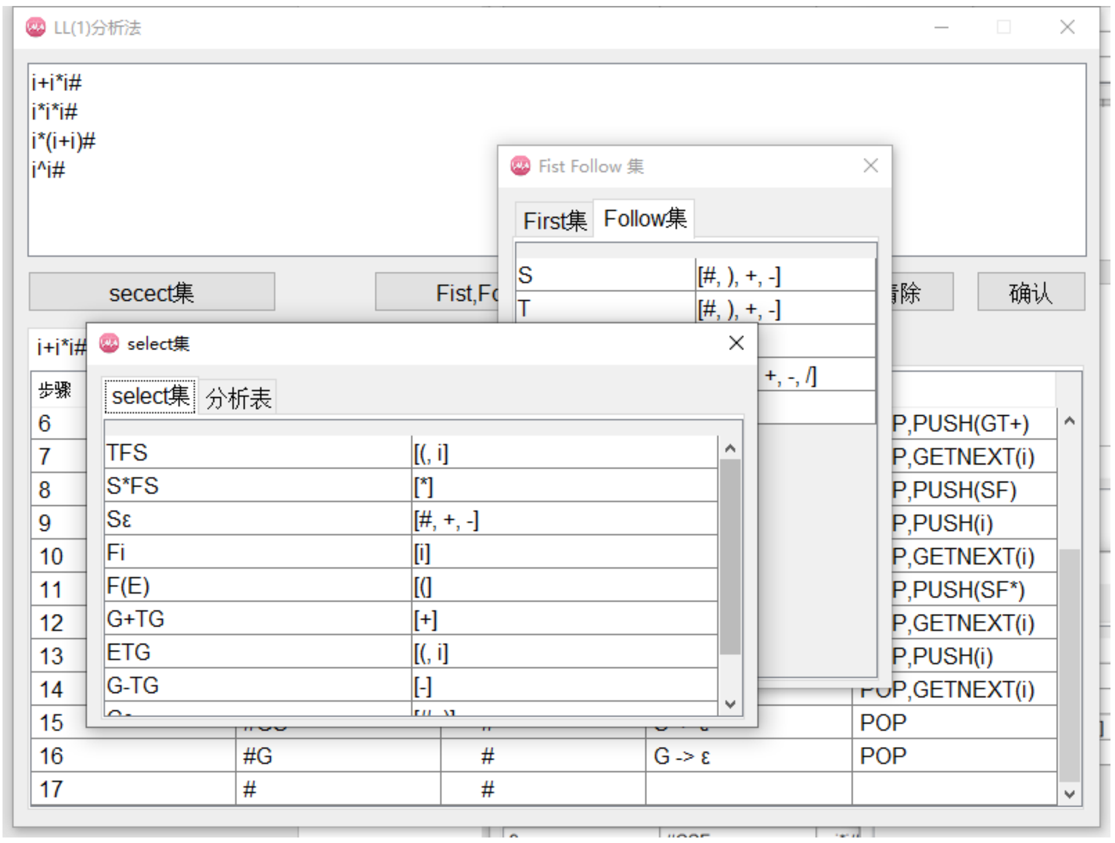
i+i*i# | i*i*i# | i*(i+i)# | i^i#

步骤	分析栈	剩余输		
6	#G	+i*i		P,PUSH(GT+)
7	#GT+	+i*i		P,GETNEXT(i)
8	#GT	i*i#		P,PUSH(SF)
9	#GSF	i*i#		P,PUSH(i)
10	#GSi	i*i#		P,GETNEXT(i)
11	#GS	*i#		P,PUSH(SF*)
12	#GSF*	*i#		P,GETNEXT(i)
13	#GSF	i#		P,PUSH(i)
14	#GSi	i#		POP,GETNEXT(i)
15	#GS	#	S -> ε	POP
16	#G	#	G -> ε	POP
17	#	#		

Fist Follow 集

First集 | Follow集

S	[#,), +, -]
T	[#,), +, -]
E	[#,)]
F	[#,), *, +, -, /]
G	[#,)]



LL(1)分析法

```
i+i*i#
i*i*i#
i*(i+i)#
i^i#
```

secect集 | Fist,F | 除 | 确认

Fist Follow 集 — First集 | Follow集

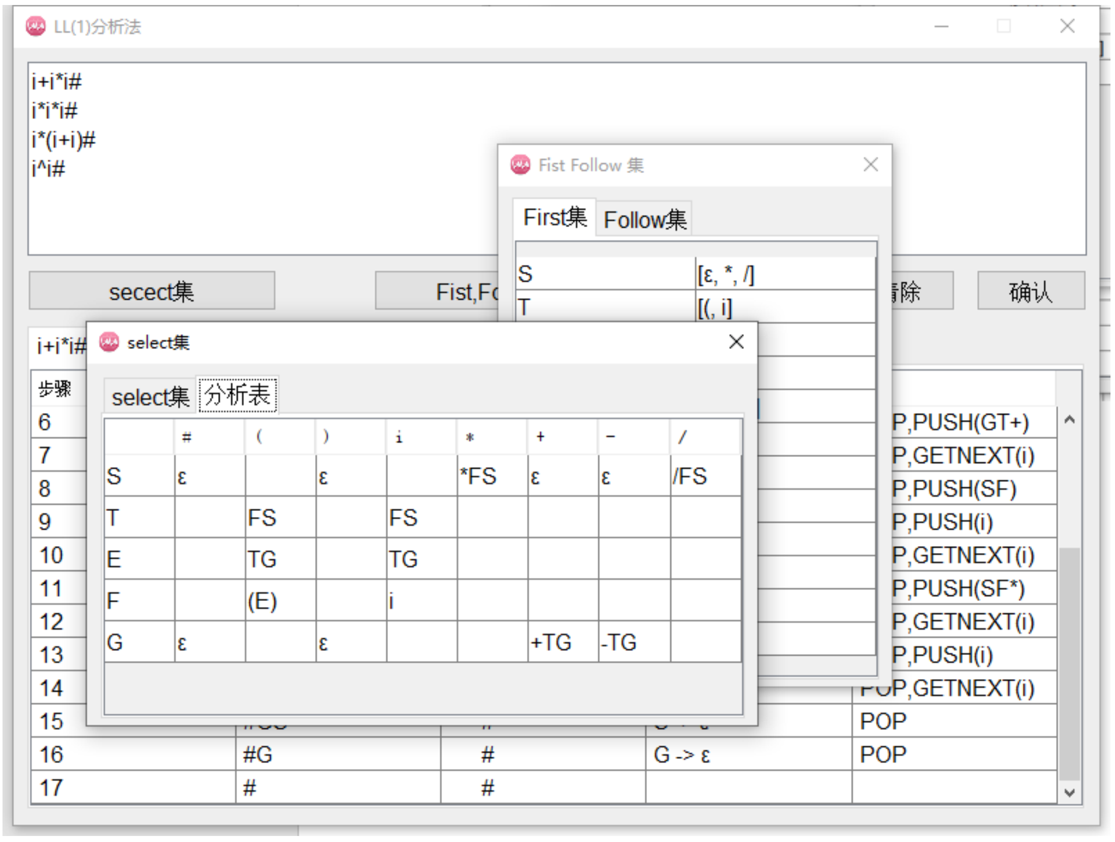
S	[#,), +, -]
T	[#,), +, -]
	+, -, /]

select集 — select集 | 分析表

TFS	[(, i]
S*FS	[*]
Sε	[#, +, -]
Fi	[i]
F(E)	[(]
G+TG	[+]
ETG	[(, i]
G-TG	[-]

i+i*i#

步骤				
6				P,PUSH(GT+)
7				P,GETNEXT(i)
8				P,PUSH(SF)
9				P,PUSH(i)
10				P,GETNEXT(i)
11				P,PUSH(SF*)
12				P,GETNEXT(i)
13				P,PUSH(i)
14				P,GETNEXT(i)
15				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 | Fist,F | 除 | 确认

Fist Follow 集 — First集 | Follow集

S	[ε, *, /]
T	[(, i]

select集 — select集 | 分析表

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

i+i*i#

步骤				
6				P,PUSH(GT+)
7				P,GETNEXT(i)
8				P,PUSH(SF)
9				P,PUSH(i)
10				P,GETNEXT(i)
11				P,PUSH(SF*)
12				P,GETNEXT(i)
13				P,PUSH(i)
14				P,GETNEXT(i)
15				POP
16	#G	#	G -> ε	POP
17	#	#		

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。