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){
```

```
    result.addAll(Analyzer.LineAnalyse(line));Analyzer.LineAnalyse(line)
}
return result
LineAnalyse 方法
if(当前分析的是 null,//,\n,或者 Length == 0){
    则直接结束
}
else {
    if(字母表含有当前头部的 string){
        while(是字母或者是数字){
            继续取出之后的部分
        }
        得到了一个 String
        if(单词是关键字){
            标记为 关键字
        }
        else{
            标记为 标识符
        }
    }
    else if(数字表含有当前头部的的 string){
        while(是数字或者小数点){
            继续取出之后的部分
        }
        if(数字之后直接追加字母){
            标记错误
            brerak
        }
        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
    }
    else{
        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
            标记 运算符
        }
        else{
            标记 错误
        }
    }
    递归处理之后的 String
    resturn  result.addAll(递归的结果);
}
```

2.算法流程图

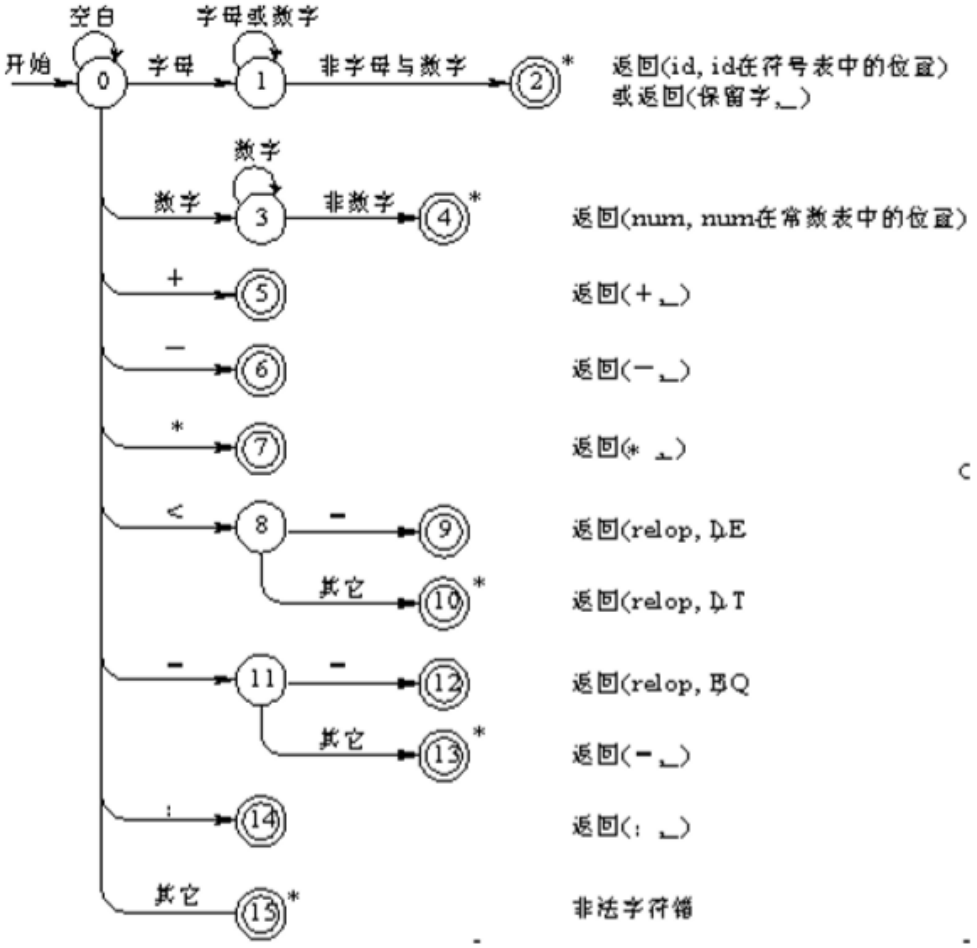

空白
字母或数字
开始
0
字母
1
非字母与数字
2
*
返回(id, id在符号表中的位置)
或返回(保留字,_)
数字
数字
3
非数字
4
*
返回(num, num在常数表中的位置)
+
5
返回(+,_)
-
6
返回(-,_)
*
7
返回(* ,_)
<
8
=
9
返回(relop, LE
其它
10
*
返回(relop, LT
=
11
=
12
返回(relop, EQ
其它
13
*
返回(=,_)
:
14
返回(: ,_)
其它
15
*
非法字符错

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

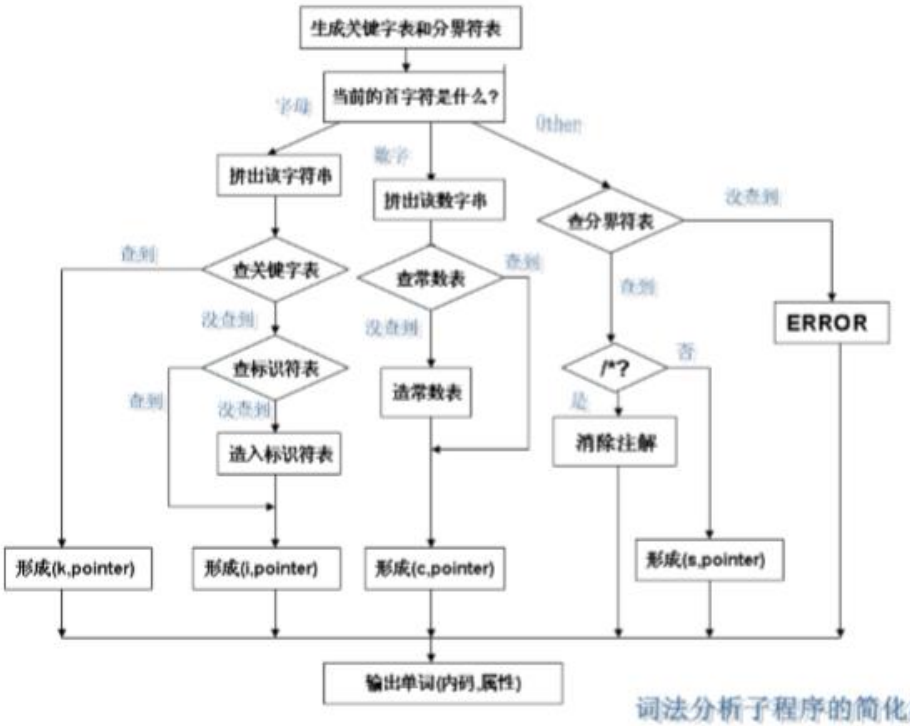
指针	分界符
0	,
1	;
2	(
3	)
4	[
5	]

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[] ){
```

```
28.         ana.setKP(k,p);
29.     }
30. }
31. class Analyzer{//在这里是用 C 的标准了
32.     String alphabet = "ABCDEFGHIGKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"
    ;//字母
33.     String number = "0123456789";//数字
34.
35.     String keyword[] = {"auto","break","case","char","const ","continue","de
    fault","do ",
36.             "double ","else ","enum ","extern","float","for","goto","if","in
    t",
37.             "long","register","return","short","signed","sizeof","static","s
    truct","switch",
38.             "typedef","unsigned","union","void","volatile","while"};//关键
    字
39.     String operator[] ={"<<=",">>=","&&","||","<=","|=","*=","^=","==","++",
    "--","/=","-=","+=","%=","!=",">=","[","]","!","%","(",")","*","+",",","-
    ","/",";","<","=",">"};//运算符
40.     String arithmeticOperator[] = {"++","--","+","-","*","/","%"};//算术运算
    符
41.     String relationalOperator[] = {"<=","<",">=",">","==","!="};//关系运算
    符
42.     String logicalOperator[] = {"&&","||","!"};//逻辑运算符
43.     String delimiter[] = {";",",","(",")","[","]"};//分界符
44.     String assignmentOperator[] ={"=","+=","-
    =","*=","/=","%=","<<=",">>=","%=","^=","|="};//赋值运算符
45.
46.     Map<String,String> opS;//<单个运算符,种类名>
47.     Map<String,String[]> KindtoArrary;//<种类名,对应的运算符数组>
48.
49.     List<Result> result = new ArrayList<>();
50.
51.     public Analyzer(){
52.         KindtoArrary = new HashMap<>();
53.         KindtoArrary.put("算术运算符",arithmeticOperator);
54.         KindtoArrary.put("关系运算符",relationalOperator);
55.         KindtoArrary.put("逻辑运算符",logicalOperator);
56.         KindtoArrary.put("分界符",delimiter);
57.         KindtoArrary.put("赋值运算符",assignmentOperator);
58.         opS = new HashMap<>();
59.         KindtoArrary.keySet().forEach(KindStr->Arrays.asList(KindtoArrary.ge
    t(KindStr)).forEach(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 && line.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;
```

```
                }
            }
            if(!match){//没有匹配到
                res.setWord(line.substring(0,1));
                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;
```

```
176.     }
177.     @Override
178.     public String toString(){
179.         String strs[] = {word,binarySequence,kind,location};
180.         StringBuffer toString = new StringBuffer();
181.         for(String str:strs){
182.             str =  String.format("%-20s", str);
183.             toString.append(str);
184.         }
185.         return toString.toString();
186.     }
187. }
188. class DataList{
189.     static List<String> id = new ArrayList<>(),ci = new ArrayList<>(),ERROR
     = new ArrayList<>();//标识符 常数
190.     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
191.         if(id.contains(str)){
192.             return id.indexOf(str);
193.         }
194.         else{
195.             id.add(str);
196.             return id.size()-1;
197.         }
198.     }
199.     public static int getCI(String str){//获取常数位置
200.         if(ci.contains(str)){
201.             return ci.indexOf(str);
202.         }
203.         else{
204.             ci.add(str);
205.             return ci.size()-1;
206.         }
207.     }
208.     public static int getERROR(String str){//获取错误代码
209.         if(ERROR.contains(str)){
210.             return ERROR.indexOf(str);
211.         }
212.         else{
213.             ERROR.add(str);
214.             return ERROR.size()-1;
215.         }
216.     }
217. }
```

GUI.java

```
1.  package 实验一___词法分析设计;
2.
3.  import javax.swing.*;
4.  import javax.swing.table.AbstractTableModel;
5.  import java.awt.*;
6.  import java.io.BufferedReader;
7.  import java.io.File;
8.  import java.io.FileReader;
9.  import java.io.IOException;
10. import java.awt.event.ActionEvent;
11. import java.awt.event.ActionListener;
12. import java.util.List;
13. import java.util.Vector;
14.
15. class Windows extends JFrame{
16.     JMenuBar bar;
17.     JMenu menu;
18.     JMenuItem file;
19.     JMenuItem manuallySet;
20.     JMenuItem exit;
21.     JTextArea TA;
22.     JButton clear;
23.     JButton analyse;
24.     String[] text;
25.     JTable table;
26.     Vector<String[]> vecRes = new Vector<>();
27.     TableDataModel tableDataModel;
28.     JScrollPane restablescrollPane;
29.     Solution sl = new Solution();
30.     public Windows(){
31.         try{
32.             setIconImage(new ImageIcon("bilibili.PNG").getImage());
33.             Font f = new Font("Yahei Consolas Hybrid",Font.PLAIN,16);
34.             String   names[]={ "MenuBar","Menu","MenuItem", "TextArea", "But
    ton", "ScrollPane", "Table"};
35.             for (String item : names) {
36.                 UIManager.put(item+ ".font",f);
37.             }
38.             UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.Window
    sLookAndFeel");
39.         }catch(Exception e){}
40.         init();
```

```
41.
42.         setSize(600,800);//初始大小
43.         setLocation(640,100);//初始位置
44.         setVisible(true);//是否可视
45.         setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);//X 退出
46.     }
47.     public void init(){
48.         setTitle("词法分析器");
49.         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
50.         setVisible(true);
51.         setResizable(false);
52.         setLayout(null);
53.         setBounds(10, 10, 300, 400);
54.         initMenu();//初始化菜单
55.         initTextArea();//初始化输入文本
56.         initButton();//初始化按钮
57.         initResultTable();//初始化结果区域
58.     }
59.     private void initMenu(){
60.         class fileListen implements ActionListener{
61.             @Override
62.             public void actionPerformed(ActionEvent e){
63.                 JFileChooser fileChooser = new JFileChooser("D:\\工作
    \\programs");
64.                 fileChooser.setFileSelectionMode(JFileChooser.FILES_ONLY);
65.                 fileChooser.showOpenDialog(null);
66.                 File file = fileChooser.getSelectedFile();
67.                 if(file!=null){
68.                     try{
69.                         BufferedReader read = new BufferedReader(new FileRea
    der( file ));
70.                         Object[] lines = read.lines().toArray();
71.                         StringBuffer bufferTA = new StringBuffer();
72.                         for(Object line:lines){
73.                             bufferTA.append(line.toString()+"\n");
74.                         }
75.                         TA.setText(bufferTA.toString());
76.                     }
77.                     catch (IOException err){
78.                     }
79.                 }
80.             }
81.         }
82.         class manuallySet implements ActionListener {
```

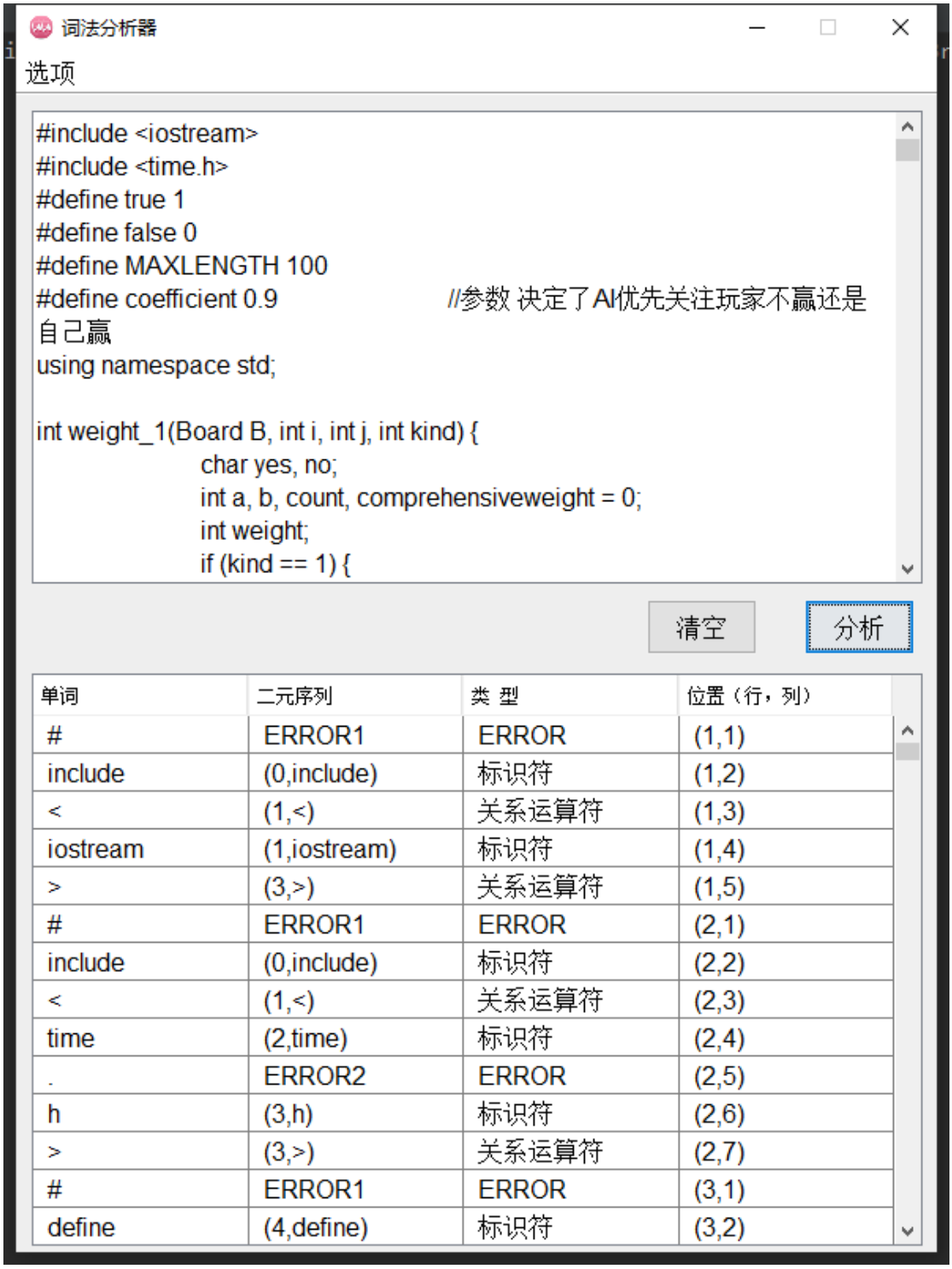
```
83.             @Override
84.             public void actionPerformed(ActionEvent e) {
85.                 String [] k ={};
86.                 String [] p ={};
87.                 String kString,pString;
88.                 Boolean changed = true;
89.                 do{
90.                     kString = JOptionPane.showInputDialog(null,"请输入 K[ ]: \n","自定义 K,P",JOptionPane.PLAIN_MESSAGE);
91.                     if(kString==null){
92.                         changed = false;
93.                         break;
94.                     }
95.                 }while (kString.length()<2);
96.                 if(changed){
97.                     boolean allchanged = true;
98.                     do{
99.                         pString = JOptionPane.showInputDialog(null,"请输入 P[ ]: \n","自定义 K,P",JOptionPane.PLAIN_MESSAGE);
100.                        if(pString == null){
101.                            allchanged = false;
102.                            break;
103.                        }
104.                    }while (pString.length()<2);
105.                    if(allchanged){
106.                        kString = kString.substring(1,kString.length()-1);
107.                        pString = pString.substring(1,pString.length()-1);
108.                        k = kString.split(" ");
109.                        p = pString.split(" ");
110.                        if(k.length<2 || p.length<2){//如果输入不规范 警告 不修改 kp
111.                            JOptionPane.showMessageDialog(null, "格式输入错误", "Error !", JOptionPane.ERROR_MESSAGE);
112.                        }
113.                        else
114.                            sl.manullySetKP(k,p);
115.                    }
116.                }
117.            }
118.        }
119.        class exitListen implements ActionListener {
120.            @Override
```

```
121.             public void actionPerformed(ActionEvent e) {
122.                 dispose();
123.             }
124.         }
125.
126.         bar = new JMenuBar();
127.         setJMenuBar(bar);
128.
129.         menu = new JMenu("选项");
130.         bar.add(menu);
131.
132.         file = new JMenuItem("选择文件");
133.         file.addActionListener(new fileListen());//读取文件到 TA 里
134.
135.         manuallySet = new JMenuItem("手动设定");
136.         manuallySet.addActionListener(new manuallySet());
137.
138.         exit = new JMenuItem("退出");
139.         exit.addActionListener(new exitListen());
140.
141.         menu.add(file);
142.         menu.add(manuallySet);
143.         menu.add(exit);
144.     }
145.     private void initTextArea(){
146.         TA = new JTextArea();
147.         JScrollPane SP = new JScrollPane(TA);
148.         TA.setLineWrap(true); // 设置自动换行
149.         SP.setBounds(10, 10, 565, 300);
150.         add(SP);
151.     }
152.     private void initButton(){
153.         class clearListen implements ActionListener{
154.             @Override
155.             public void actionPerformed(ActionEvent e){
156.                 TA.setText("");
157.                 vecRes.clear();
158.                 table.validate();
159.                 table.updateUI();
160.                 restablescrollPane.updateUI();
161.             }
162.         }
163.         class analyseListen implements ActionListener{
164.             @Override
```

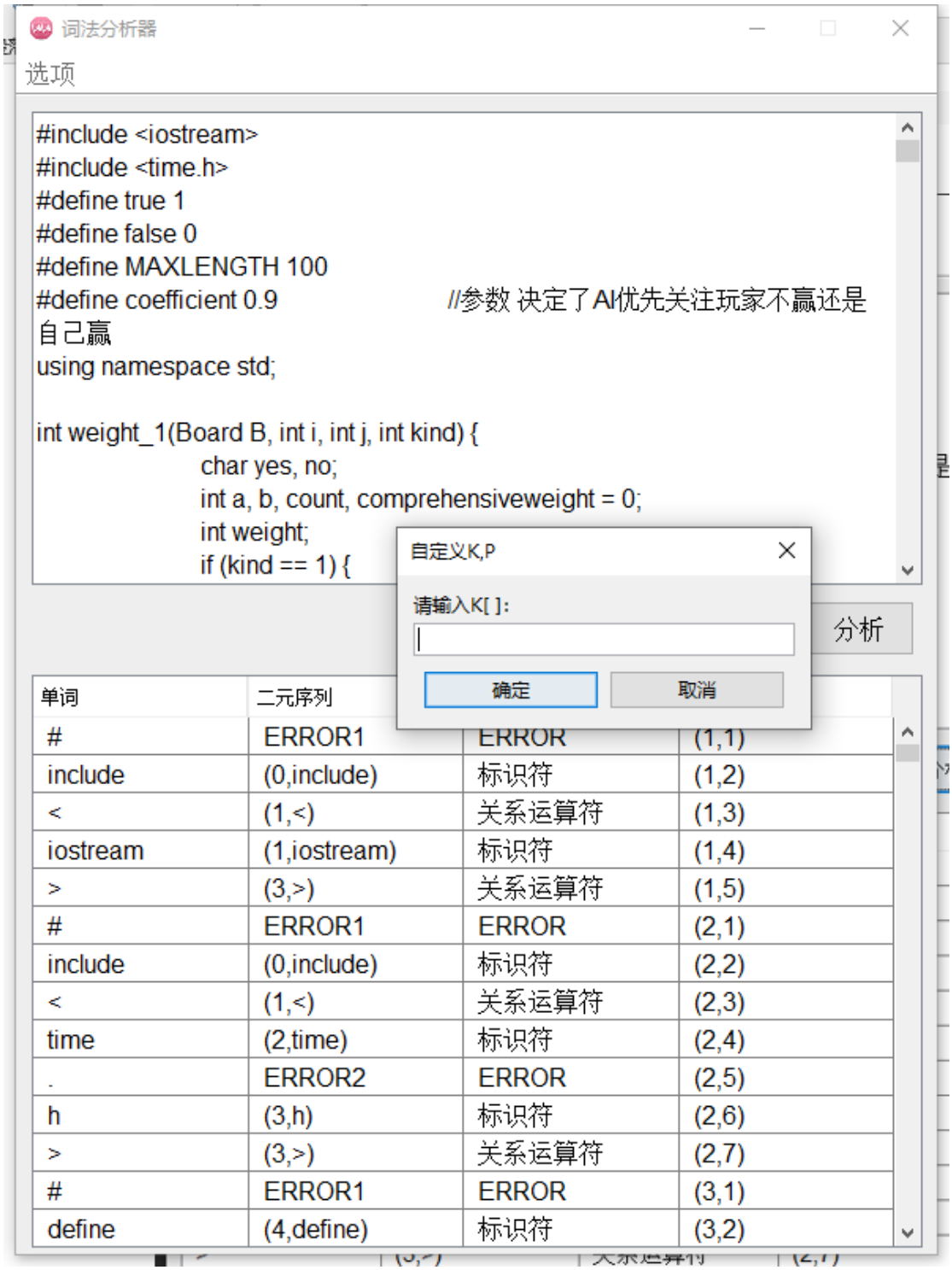
```
165.             public void actionPerformed(ActionEvent e){
166.                 text = TA.getText().split("\n");//这样分割后的String没有
    \n
167.                 //for(String str:text) System.out.println(str);
168.                 vecRes.clear();
169.                 List<Result> resS = sl.Solve(text);
170.                 for(Result result:resS){
171.                     vecRes.add(result.toStringArrary());
172.                 }
173.                 //vecRes.forEach(Strings -> {for(String str:Strings) System
    .out.print(str+" ");System.out.println();});
174.                 table.validate();
175.                 table.updateUI();
176.                 restablescrollPane.updateUI();
177.             }
178.         }
179.         clear = new JButton("清空");
180.         clear.addActionListener(new clearListen());
181.
182.         analyse = new JButton("分析");
183.         analyse.addActionListener(new analyseListen());
184.
185.         clear.setBounds(400,320,70,35);
186.         analyse.setBounds(500,320,70,35);
187.         add(clear);
188.         add(analyse);
189.     }
190.     private void initResultTable(){
191.         tableDataModel = new TableDataModel(vecRes);
192.         table = new JTable(tableDataModel);
193.         table.setVisible(true);
194.         table.setPreferredScrollableViewportSize(new Dimension(550, 100));

195.         table.setRowHeight(24);
196.         restablescrollPane = new JScrollPane(table);
197.         restablescrollPane.setBounds(10, 367, 565, 363);
198.         add(restablescrollPane);
199.         pack();
200.     }
201. }
202.
203. class TableDataModel extends AbstractTableModel{
204.     private Vector<String[]> TableData;//用来存放表格数据的线性表
205.     private Vector<String> TableTitle;//表格的 列标题
```

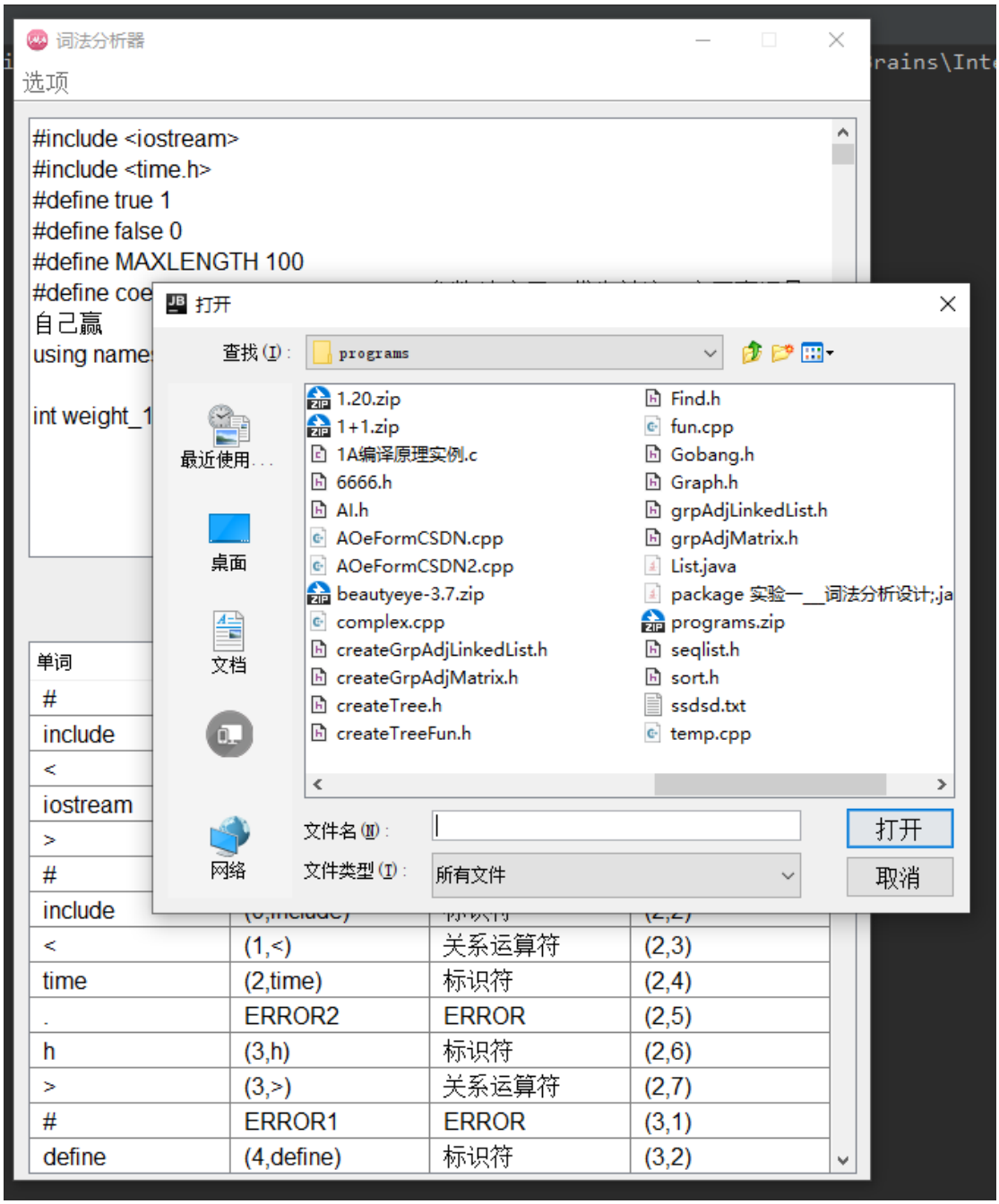
```
206.     public TableDataModel(Vector data){
207.         String Names[] = {"单词","二元序列","类 型","位置（行，列）"};
208.         Vector Namessss = new Vector();
209.         for(String str:Names){
210.             Namessss.add(str);
211.         }
212.         TableTitle = Namessss;
213.         TableData = data;
214.     }
215.
216.     @Override
217.     public int getRowCount(){
218.         return TableData.size();
219.     }
220.     public int getColumnCount(){
221.         return TableTitle.size();
222.     }
223.     @Override
224.     public String getColumnName(int colum){
225.         return TableTitle.get(colum);
226.     }
227.     public Object getValueAt(int rowIndex, int columnIndex){
228.         String LineTemp[] = this.TableData.get(rowIndex);
229.         return LineTemp[columnIndex];
230.     }
231.     @Override
232.     public boolean isCellEditable(int rowIndex, int columnIndex){//不允许编
    辑
233.         return false;
234.     }
235. }
```

运行结果：



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。