电 子 科 技 大 学

实 验 报 告

学生姓名：董文龙　　学 号：2018081309003　　指导教师：陈昆

实验地点：科 A-504　　实验时间：2021 年 5 月 15 日

一、实验室名称：科 A-504

二、实验项目名称：词法分析器的设计与实现

三、实验学时：4 学时

四、实验原理

1.编译程序要求对高级语言编写的源程序进行分析和合成，生成目标程序。词法分析是对源程序进行的首次分析，实现词法分析的程序为词法分析程序。

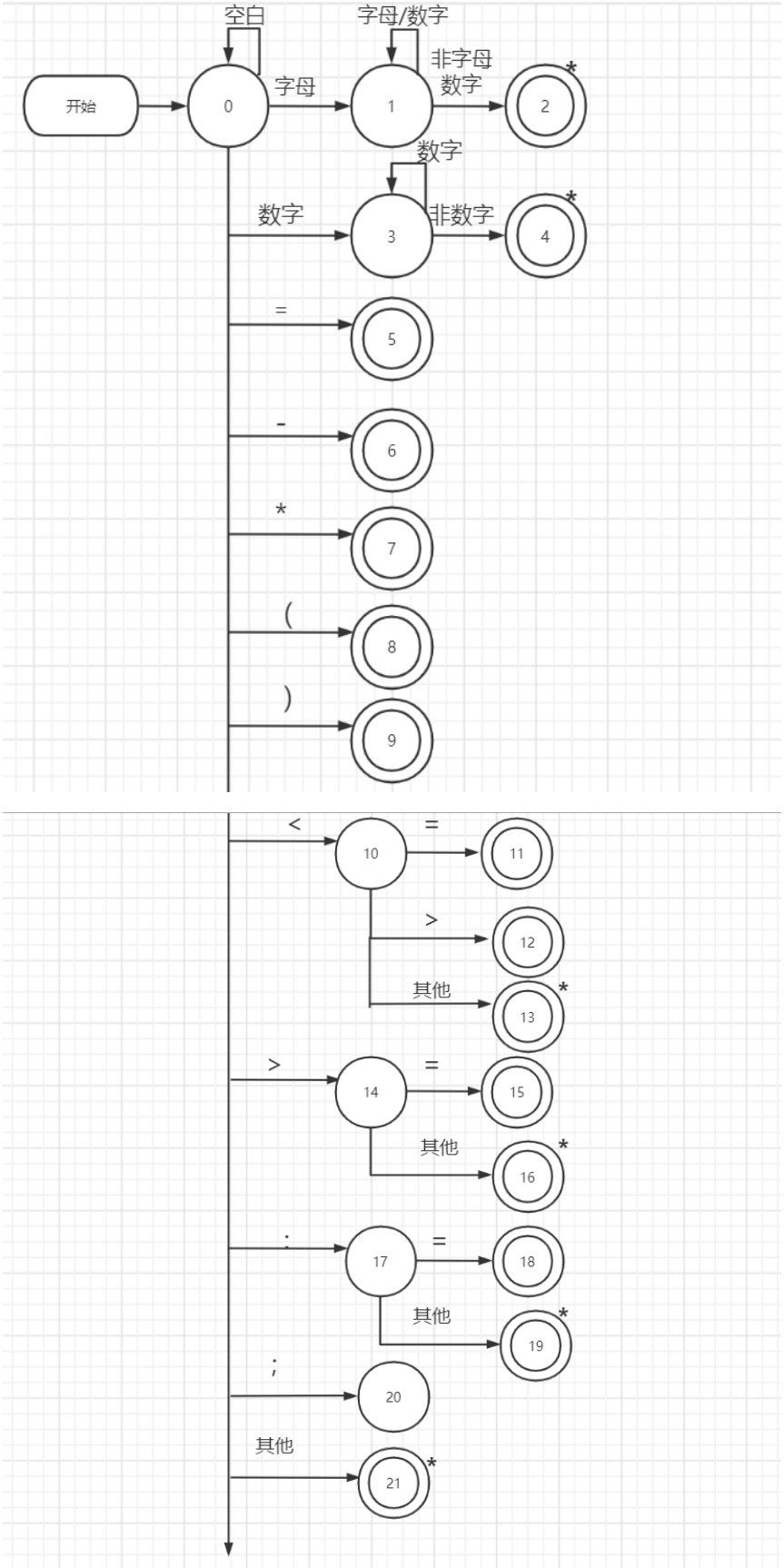
2.词法分析的功能是从左到右逐个地扫描源程序字符串，按照词法规则识别出单词符号作为输出，对识别过程中发现的词法错误，输出相关信息。

3.本实验的种别表如下图所示：

单词符号	种别	单词符号	种别	单词符号	种别
begin	1	end	2	integer	3
if	4	then	5	else	6
function	7	read	8	write	9
标识符	10	常数	11	=	12
<>	13	<=	14	<	15
>=	16	>	17	-	18
*	19	:=	20	(	21
)	22	;	23		

4.状态转换图是有限有向图，是设计词法分析器的有效工具。本实验的

状态转换图如下图所示：



5.二元式格式文件：

① 二元式文件*.dyd

单词符号(长度 16) ∪种别(长度 2)

② 每行结尾对应为 ∪∪∪...∪EOLN∪24

③ 文件结尾对应为 ∪∪∪...∪EOF∪25

五、实验目的

通过设计词法分析器的实验，了解和掌握词法分析程序设计的原理及相应的程序设计方法，同时提高编程能力。

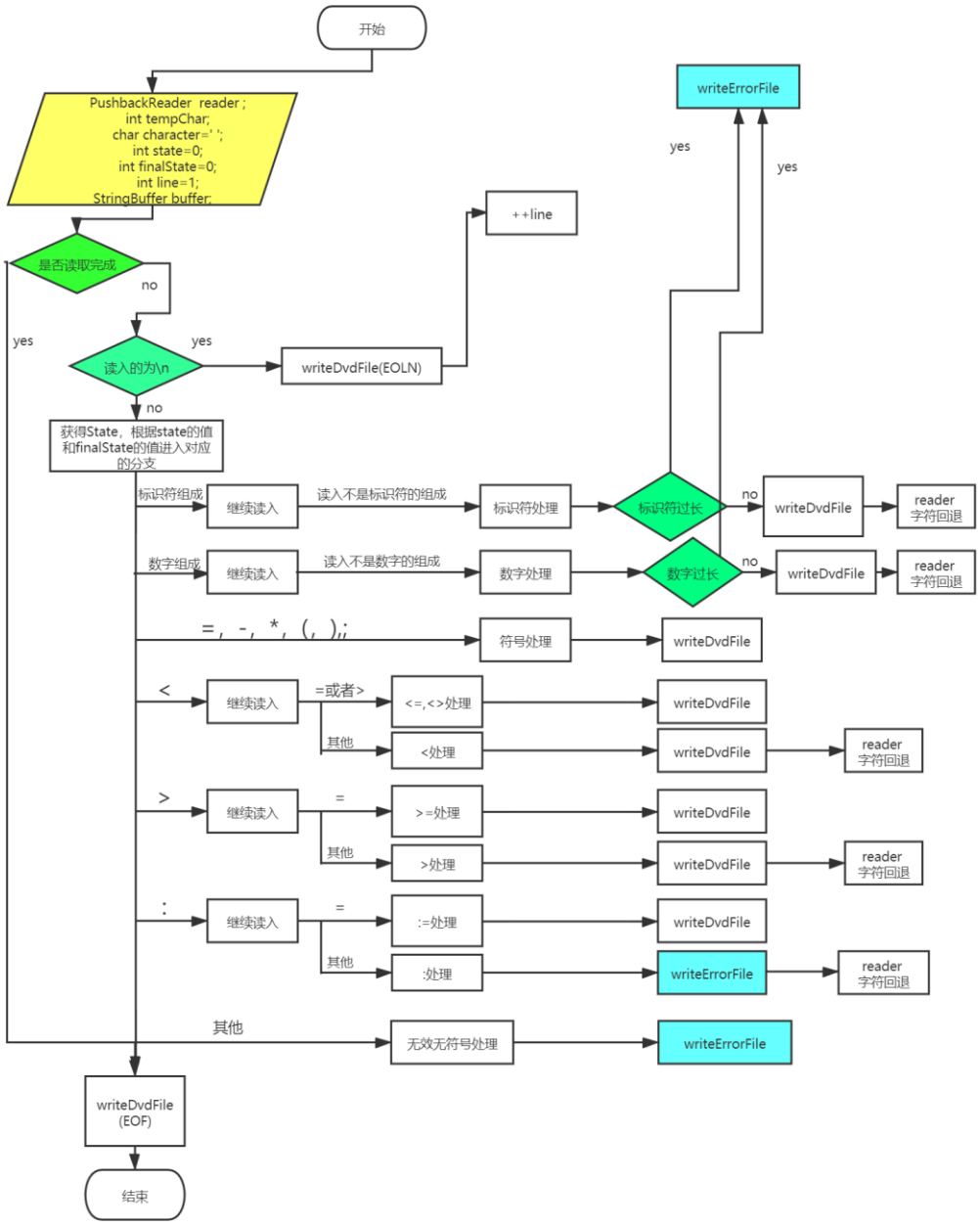
六、实验内容

实现求 n！的极小语言的词法分析程序，返回二元式作为输出。

七、实验器材（设备、元器件）

1. 操作系统：Windows 10
2. 开发工具：IntelliJ IDEA，JDK1.8
3. 电脑配置：Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz　2.00 GHz，内存 20GB

八、实验步骤（此步骤给出算法流程图即可）



九、实验数据及结果分析

1.无错误 source.pas 文件

```
1 begin
2   integer k;
3   integer function F(n);
4     begin
5       integer n;
6       if n<=0 then F:=1
7       else F:=n*F(n-1)
8     end;
9   read(m);
10  k:=F(m);
11  write(k)
12 end
```

对应的 dyd 文件：

```
1           begin 01
2             EOLN 24
3         integer 03
4               k 10
5               ; 23
6            EOLN 24
7         integer 03
8        function 07
9               F 10
10              ( 21
11              n 10
12              ) 22
13              ; 23
14           EOLN 24
15          begin 01
16           EOLN 24
17        integer 03
18              n 10
19              ; 23
20           EOLN 24
21             if 04
22              n 10
23             <= 14
24              0 11
25           then 05
26              F 10
27             := 20
28              1 11
29           EOLN 24
30           else 06
31              F 10
32             := 20
33              n 10
34              * 19
35              F 10
36              ( 21
37              n 10
38              - 18
39              1 11
40              ) 22
41           EOLN 24
42            end 02
43              ; 23
44           EOLN 24
45           read 08
46              ( 21
47              m 10
48              ) 22
49              ; 23
50           EOLN 24
51              k 10
52             := 20
53              F 10
54              ( 21
55              m 10
56              ) 22
57              ; 23
58           EOLN 24
59          write 09
60              ( 21
61              k 10
62              ) 22
63           EOLN 24
64            end 02
65            EOF 25
```

此时 err 文件没有报错。

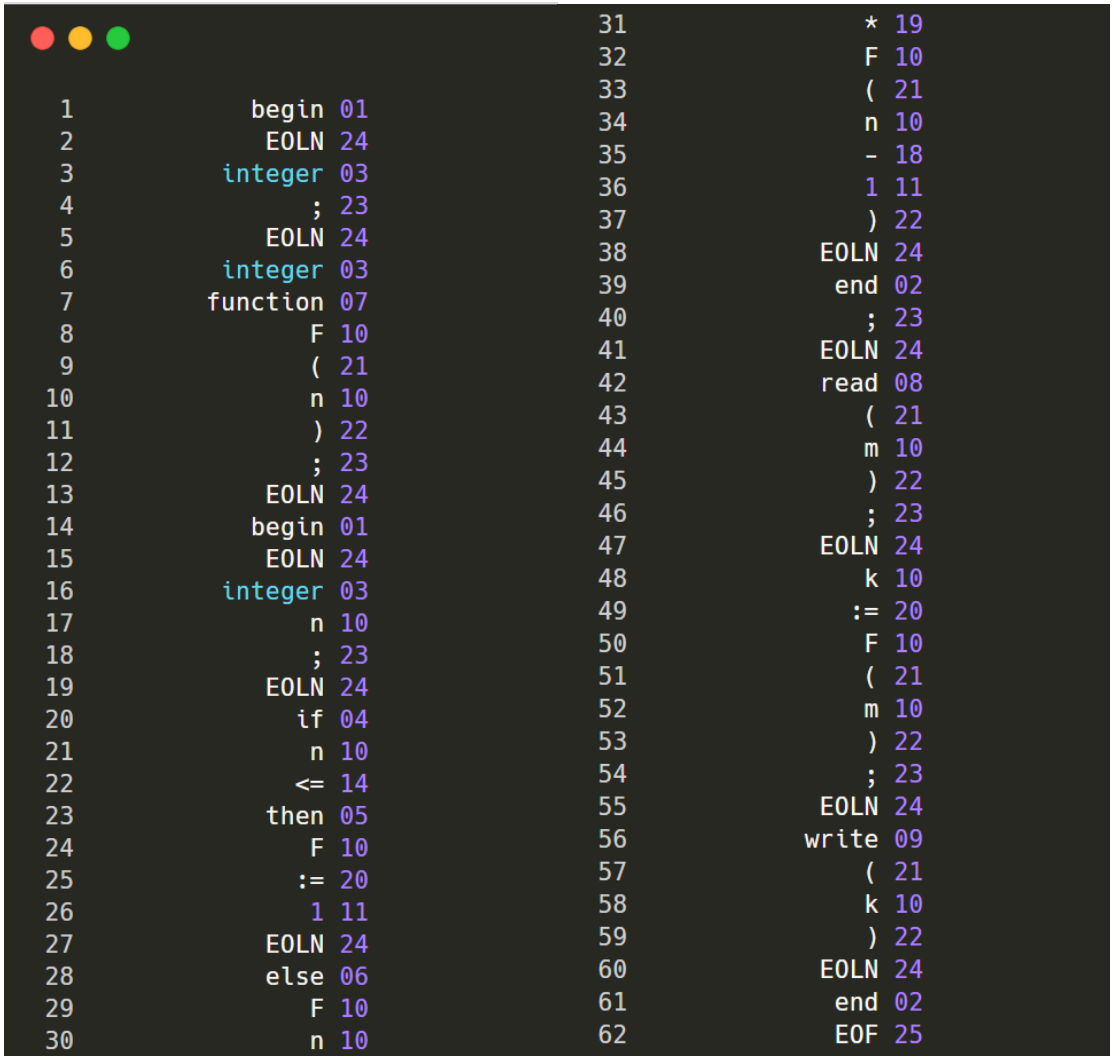
2.有错误 source.pas 文件

```
begin
  integer #;
  integer function @F(n);
    begin
      integer n;
      if n<=201808130900320180813090003 then F:=1
      else F:n*F(n-1)
    end;
  read(m);
  k:=F(m);
  write(k)
end
```

err 文件报错信息：

```
***LINE2: '#'is invalid symbol.
***LINE3: '@'is invalid symbol.
***LINE6:  Number '201808130900320180813090003'is too long.
***LINE7:  Missing "=" after ":".
```

dyd 文件输出：



十、实验结论

本实验程序较好地完成了词法分析程序的设计与实现，能够对所给文法的程序进行词法分析，在没有词法错误的时候生成相应的二元式文件。如果有错给出出错信息和所在行数，该实验程序可一次性给出源程序中的词法错误。

十一、总结及心得体会

通过该实验，对词法分析程序的设计，以及运用 JAVA 语言进行编程有了更深刻的理解，同时加深了自己对词法分析程序的原理的理解与掌握，提高了自己的动手能力。

十二、对本实验过程及方法、手段的改进建议

程序设计合理，代码可进一步优化，对类进行更好的封装。

报告评分：

指导教师签字：

词法分析代码：

Lexical_CharToState 文件：

```
/**
 * @author YiGeDian
 * @date 2021/5/21 15:49
 */
public class Lexical_CharToState {
    public static int CharToState(char s) {
        switch (s) {
            case ' ':
            case '\r':
            case '\n':
                return -1;
            case '\t':
                return 0;
            case 'a':
            case 'b':
            case 'c':
            case 'd':
            case 'e':
            case 'f':
```

```
                case 'g':
                case 'h':
                case 'i':
                case 'j':
                case 'k':
                case 'l':
                case 'm':
                case 'n':
                case 'o':
                case 'p':
                case 'q':
                case 'r':
                case 's':
                case 't':
                case 'u':
                case 'v':
                case 'w':
                case 'x':
                case 'y':
                case 'z':
                case 'A':
                case 'B':
                case 'C':
                case 'D':
                case 'E':
                case 'F':
                case 'G':
                case 'H':
                case 'I':
                case 'J':
                case 'K':
                case 'L':
                case 'M':
                case 'N':
                case 'O':
                case 'P':
                case 'Q':
                case 'R':
                case 'S':
                case 'T':
                case 'U':
                case 'V':
                case 'W':
                case 'X':
```

```
                case 'Y':
                case 'Z':
                    return 1; // 英文字母'a'~'z'和'A'~'Z'
                case '0':
                case '1':
                case '2':
                case '3':
                case '4':
                case '5':
                case '6':
                case '7':
                case '8':
                case '9':
                    return 2; // 数字'1'~'9'
                case '=':
                    return 3;
                case '-':
                case '*':
                case '(':
                case ')':
                case ';':
                    return 4;
                case '<':
                    return 5;
                case '>':
                    return 6;
                case ':':
                    return 7;
                default:
                    return 8;
            }
        }
}
```

Lexical_WordToIdType 文件：

```
/**
 * @author YiGeDian
 * @date 2021/5/21 11:11
 */
public class Lexical_WordToIdType {
    public static String wordToIdType (String string) {
        switch (string) {
            case "begin":
                return "01";
```

```
            case "end":
                return "02";
            case "integer":
                return "03";
            case "if":
                return "04";
            case "then":
                return "05";
            case "else":
                return "06";
            case "function":
                return "07";
            case "read":
                return "08";
            case "write":
                return "09";
            case "=":
                return "12";
            case "<>":
                return "13";
            case "<=":
                return "14";
            case "<":
                return "15";
            case ">=":
                return "16";
            case ">":
                return "17";
            case "-":
                return "18";
            case "*":
                return "19";
            case ":=":
                return "20";
            case "(":
                return "21";
            case ")":
                return "22";
            case ";":
                return "23";
            default:
                if (string.matches("[0-9]+")) {
                    return "11";//常量
                } else {
```

```
                        return "10";//标识符
                    }
                }
            }
        }
```

Lexical_Analyzer 文件：

语法分析：

```
import java.io.*;



/**
 * @author YiGeDian
 * @date 2021/5/21 11:26
 */
public class Lexical_Analyzer {
    public static void writeDydFile(String str) throws IOException {
        File file_dyd = new File("source.dyd");
        FileWriter fileWriter_dyd=new FileWriter(file_dyd,true);
        fileWriter_dyd.write(str);
        fileWriter_dyd.close();
    }
    public static void writeErrorFile(String str) throws IOException {
        File file_err=new File("source.err");
        FileWriter fileWriter_err=new FileWriter(file_err,true);
        fileWriter_err.write(str);
        fileWriter_err.close();
    }
    public static void main(String[] args) throws IOException {
        /*读取文件*/
        File file_pas = new File("source.pas");
        File file_err = new File("source.err");
        File file_dyd = new File("source.dyd");
        /*读取文件变量*/
        PushbackReader    reader = null;
        /*接受 reader 读取变量*/
        int tempChar;
        /*存储 reader 的字符*/
        char character=' ';
        /*根据读取的字符数得到返回值*/
        int state=0;
        int finalState=0;
        /*文件行号*/
```

```
        int line=1;
        /*存储一个串*/
        StringBuffer buffer=new StringBuffer();
        /*文件是否存在*/
        if ((!file_dyd.exists())||(!file_dyd.isFile())){
            file_dyd.createNewFile();
        }
        if ((!file_err.exists())||(!file_err.isFile())){
            file_err.createNewFile();
        }
        /*初始化读取文件变量*/
        try {
            reader = new PushbackReader(new FileReader(file_pas));
            /*System.out.println("Read successfully!");*/
        } catch (IOException e) {
            System.out.print(e);
        }
        /*写初始化，每次写会重写文件内容*/
        FileWriter fileWriter_dyd=new FileWriter(file_dyd);
        FileWriter fileWriter_err=new FileWriter(file_err);
        fileWriter_dyd.write("");
        fileWriter_err.write("");
        /*开始读取内容并进行判断*/
        while
((tempChar=reader.read())!=-1||((tempChar=reader.read())==-1)&&(finalState==1||finalState==
2||finalState==5||finalState==6||finalState==7)){
            if (((char)tempChar)=='\n'){
                line++;
                String str=String.format("%16s","EOLN")+" "+"24\n";
                writeDydFile(str);
                buffer.delete(0,buffer.length());
            }else{
                character=(char)tempChar;
                state= Lexical_CharToState.CharToState(character);
                if (finalState==1||state==1&&finalState==0){//标识符:字母或者数字
                    if (state==1||state==2){
                        buffer.append(character);
                        finalState=1;
                        continue;
                    }else{
                        String word=buffer.toString();
                        if (word.length()>16){
                            String str = "***LINE" + line + ":    Identifier
"+"'"+buffer.toString()+"'"+ "is too long.\n";
```

```
                writeErrorFile(str);
            }else {
                String str=String.format("%16s", word) + " " +
Lexical_WordToIdType.wordToIdType(word) + "\n";
                writeDydFile(str);
            }
            buffer.delete(0,buffer.length());
            reader.unread(tempChar);
        }
    }
    if (finalState == 2 || state == 2&&finalState==0) {// 接收到数字串
        if (state == 2) {
            buffer.append(character);
            finalState = 2;
            continue;
        } else {
            String word = buffer.toString();
            if (word.length()>16){
                String str = "***LINE" + line + ":    Number
"+"'"+buffer.toString()+"'"+ "is too long.\n";
                writeErrorFile(str);
            }else{
                String str=String.format("%16s", word) + " " +
Lexical_WordToIdType.wordToIdType(word) + "\n";
                writeDydFile(str);
            }
            buffer.delete(0, buffer.length());
            reader.unread(tempChar);
        }
    }
    if (finalState == 0 && state == 3) {// 接收到等号
        buffer.append(character);
        String word = buffer.toString();
        String str=String.format("%16s", word) + " " +
Lexical_WordToIdType.wordToIdType(word) + "\n";
        writeDydFile(str);
        buffer.delete(0, buffer.length());
    }
    if (finalState == 0 && state == 4) {// 接收到-，*，(，);
        buffer.append(character);
        String word = buffer.toString();
        String str=String.format("%16s", word) + " " +
Lexical_WordToIdType.wordToIdType(word) + "\n";
        writeDydFile(str);
```

```
                    buffer.delete(0, buffer.length());
                }
                if (finalState == 5 || state == 5&&finalState==0) {//接受到<,<=,<>
                    if (finalState == 0 && state == 5) {//接受到<
                        buffer.append(character);
                        finalState = 5;
                        continue;
                    } else if (finalState == 5 && (state == 3 || state == 6)) {//如果为<=,<>
                        buffer.append(character);
                        String word = buffer.toString();
                        String str=String.format("%16s", word) + " " +
Lexical_WordToIdType.wordToIdType(word) + "\n";
                        writeDydFile(str);
                        buffer.delete(0, buffer.length());
                        finalState=0;
                        continue;
                    } else {// 为小于号
                        String word = buffer.toString();
                        String str=String.format("%16s", word) + " " +
Lexical_WordToIdType.wordToIdType(word) + "\n";
                        writeDydFile(str);
                        buffer.delete(0, buffer.length());
                        reader.unread(tempChar);
                    }
                }
                if (finalState == 6 || state == 6&&finalState==0) {// 接受>=,>
                    if (finalState == 0 && state == 6) {
                        buffer.append(character);
                        finalState = 6;
                        continue;
                    } else if (finalState == 6 && state == 3) {
                        buffer.append(character);
                        String word = buffer.toString();
                        String str = String.format("%16s", word) + " " +
Lexical_WordToIdType.wordToIdType(word) + "\n";
                        writeDydFile(str);
                        buffer.delete(0, buffer.length());
                        finalState=0;
                        continue;
                    } else {
                        String word = buffer.toString();
                        String str = String.format("%16s", word) + " " +
Lexical_WordToIdType.wordToIdType(word) + "\n";
                        writeDydFile(str);
```

```
                        buffer.delete(0, buffer.length());
                        reader.unread(tempChar);
                    }
                }
                if (finalState==7||state==7&&finalState==0){//:=
                    if (finalState == 0 && state == 7) {//先接受到:
                        buffer.append(character);
                        finalState= 7;
                        continue;
                    }else if (finalState == 7 && state == 3) {
                        buffer.append(character);
                        String word = buffer.toString();
                        String str=String.format("%16s", word) + " "
+Lexical_WordToIdType.wordToIdType(word) + "\n";
                        writeDydFile(str);
                        buffer.delete(0, buffer.length());
                        finalState=0;
                        continue;
                    }else {
                        String str = "***LINE" + line + ":    Missing \"=\" after \":\".\n";
                        writeErrorFile(str);
                        buffer.delete(0, buffer.length());
                        reader.unread(tempChar);
                    }
                }
                if (tempChar==-1){//文件结束
                    break;//跳出循环
                }
                if (state == 8&&finalState==0) {
                    buffer.append(character);
                    String str = "***LINE" + line + ": "+"'"+buffer.toString()+"'"+"is invalid
symbol.\n";
                    writeErrorFile(str);
                    buffer.delete(0, buffer.length());
                }
                finalState=0;
            }
        }
        /*文件读取完毕*/
        reader.close();
        String str=String.format("%16s","EOF")+" "+"25";
        writeDydFile(str);
    }
}
```

电 子 科 技 大 学

实　验　报　告

学生姓名：董文龙　　学 号：2018081309003　　指导教师：陈昆

实验地点：科 A-504　　实验时间： 2021 年 5 月 30 日

一、实验室名称：科 A-504

二、实验项目名称：递归下降分析器的设计与实现

三、实验学时：4 学时

四、实验原理

1.语法分析是根据某种给定的形式文法对由单词序列构成的输入文本进行分析并确定其语法结构的一种过程。语法分析器使用一个独立的词法分析器从输入字符流中分离出一个个的“单词”，并将单词流作为其输入。

2.递归下降分析法是确定的自上而下分析法，首先要将语法消除左递归和提取公共左因子。为语法的每一个非终结符构造一个函数，而产生式右端即为该函数的内容。若遇到非终结符则调用相应函数，遇到终结符则尝试匹配及进行错误处理。

3.变量名表：

- 变量名 vname: char(16)
- 所属过程 vproc:char(16)
- 分类 vkind: 0..1(0—变量、1—形参)
- 变量类型 vtype: int
- 变量层次 vlev: int
- 变量在变量表中的位置 vadr: int(相对第一个变量而言)

4.过程名表：

- 过程名 pname: char(16)
- 过程类型 ptype: types
- 过程层次 plev: int(采用 main 函数为 0，以后每一层加一)
- 第一个变量在变量表中的位置 fadr: int
- 最后一个变量在变量表中的位置 ladr: int

5.本实验提取公共左因子和消除左递归后的文法如下：

```
/**
 * @author YiGeDian
 * @date 2021/6/8 20:40
 */
/**
 * <program>-><subProgram>
 * <subProgram>->begin<declareStatementList>;<executeStatementList>end
 * <declareStatementList>-><declareStatement><declareStatementList_L()>
 * <declareStatementList_L>->;<declareStatement><declareStatementList_L>|epsilon
 * <declareStatement>-><declareVariable>|<declareFunction>
 * <declareVariable>->integer<variable>
 * <variable>-><identifier>
 * <identifier>-><letter><identifier_I>
 * <identifier_I>-><letter><identifier_I> | <digit><identifier_I>|epsilon
 * <letter>->a~z|A~Z
 * <digit>->0~9
 * <declareFunction>->integer function <identifier>(<parameter>);<functionBody>
 * <parameter>-><variable>
 * <functionBody>->begin<declareStatementList>;<executeStatementList>end
 * <executeStatementList>-><executeStatement><executeStatementList_L>
 * <executeStatementList_L>->;<executeStatement><executeStatementList_L>|epsilon
 *
<executeStatement>-><readStatement>|<writeStatement>|<assignStatement>|<conditionalStatem
ent>
 * <readStatement>->read(<variable>)
 * <writeStatement>->write(<variable>)
 * <assignStatement>-><variable>:=<arithmeticalExpression>
 * <arithmeticalExpression>-><item><arithmeticalExpression_E>
 * <arithmeticalExpression_E>->-<item><arithmeticalExpression_E>|epsilon
 * <item>-><factor><item_I>
 * <item_I>->*<factor><item_I>|epsilon
 * <factor>-><variable>|<constant>|<functionCall>;
 * <constant>-><unsignedInteger>
 * <unsignedInteger>-><digit><unsignedInteger_I>
 * <unsignedInteger_I>-><digit><unsignedInteger_I>|epsilon
 * <functionCall>-><identifier>(<arithmeticalExpression>);
 *
<conditionalStatement>->if<conditionalExpression>then<executeStatement>else<executeStatemen
t>
 *
<conditionalExpression>-><arithmeticalExpression><relationalOperator><arithmeticalExpression>
 * <relationalOperator>->< | <= | > | >= | = | <>
 */
```

五、实验目的

通过设计递归下降分析器的设计与实现实验，使同学们掌握自上而下的递归分析法的语法分析原理和程序设计方法。

六、实验内容

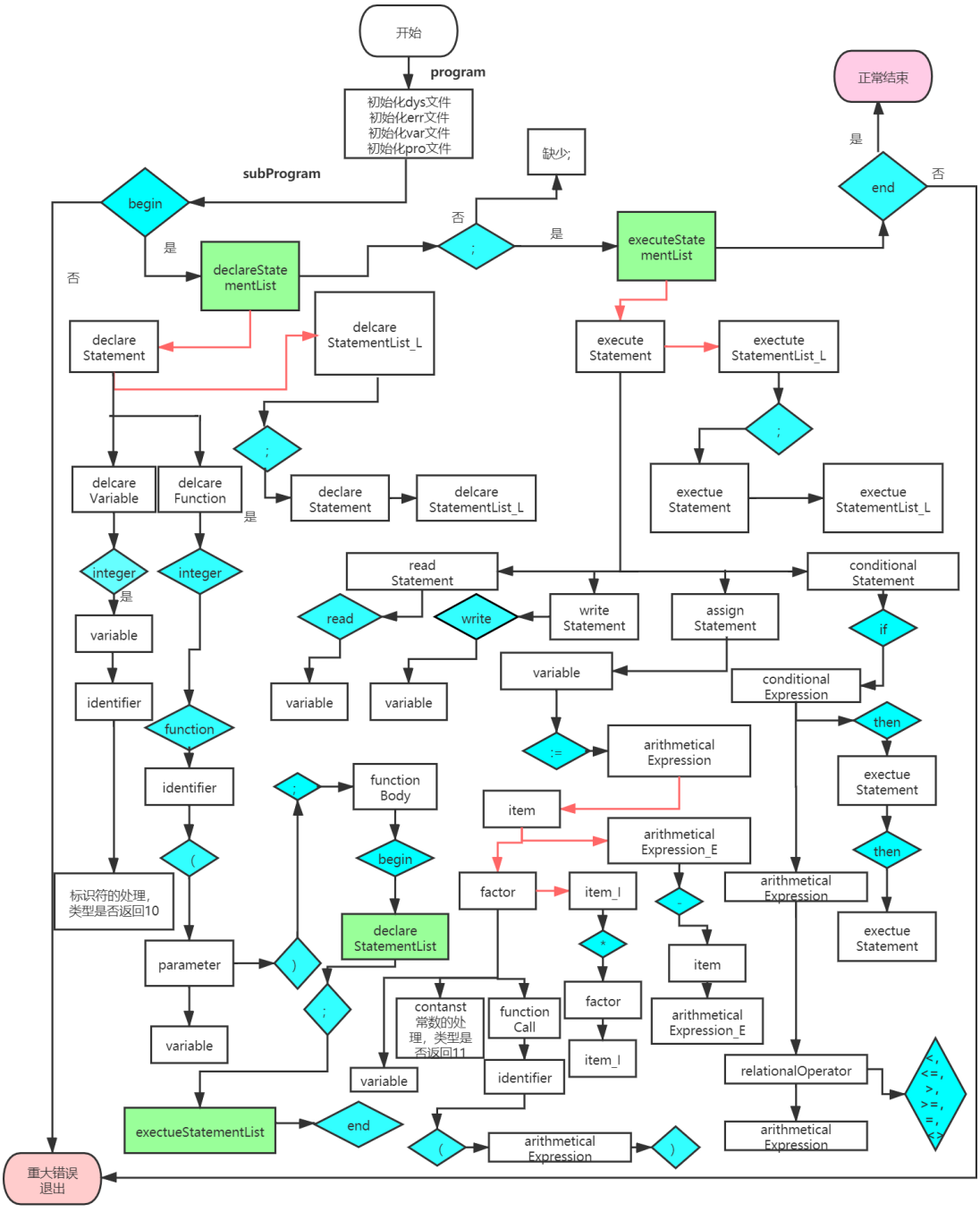
根据给定的方法，编写相应的递归下降的语法分析程序，实现对词法分析后的单词序列的语法检查和程序结构的分析，生成相应的变量名表和过程名表，并将编译中语法检查出来的错误写入相应的文件。
语法错分类:
(1)缺少符号错;
(2)符号匹配错
(3)符号无定义或重复定义。

七、实验器材（设备、元器件）

1. 操作系统：Windows 10
2. 开发工具：IntelliJ IDEA，JDK1.8
3. 电脑配置：Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz　2.00 GHz，内存 20GB

八、实验步骤

本实验的实验流程图如下，消除左递归的线已经用红色标出：



九、实验数据及结果分析

包含（初始文件（0），只修改了 intger m（1），只修改了 function（2），只修改了 readm）（3），第九行加上 integer m（4）,以及 1，3，4 的组合和其他重大错误）

1.初始文件：

Pas 文件：

```
begin
  integer k;
  integer function F(n);
    begin
      integer n;
      if n<=0 then F:=1
      else F:=n*F(n-1)
    end;
  read(m);
  k:=F(m);
  write(k)
end
```

Var 文件：

```
          vname            vproc  vkind           vtype  vlev  vadr
              k             main      0         INTEGER     0     0
              n                F      1         INTEGER     1     1
              n                F      0         INTEGER     1     2

```

Pro 文件：

```
           pname            ptype  plev  fadr  ladr
            main             void     0     0     0
               F          INTEGER     1     1     2

```

Err 文件

```
***LINE 9: m is not defined
***LINE 10: m is not defined

```

由于 m 未定义，所以第 9,10 行进行报错。

2.第一次修改（intger m;）：

Pas 文件：

```
begin
  intger m;
  integer function F(n);
    begin
      integer n;
      if n<=0 then F:=1
      else F:=n*F(n-1)
    end;
  read(m);
  k:=F(m);
  write(k)
end
```

Var 文件：

```
          vname            vproc  vkind           vtype  vlev  vadr
              n                F      1         INTEGER     1     0
              n                F      0         INTEGER     1     1
```

Pro 文件：

```
           pname            ptype  plev  fadr  ladr
            main             void     0     0     0
               F          INTEGER     1     0     1
```

Err 文件：

```
***LINE 2: missing "integer"
***LINE 9: m is not defined
***LINE 10: k is not defined
***LINE 10: m is not defined
***LINE 11: k is not defined
```

k,m 均未定义，begin 后面需要匹配 integer

3.第二次修改（integer funtion F(n);）：

Pas 文件：

```
begin
  integer k;
  integer funtion F(n);
    begin
      integer n;
      if n<=0 then F:=1
      else F:=n*F(n-1)
    end;
  read(m);
  k:=F(m);
  write(k)
end
```

此时产生重大语法错误，没有 Var 文件和 Pro 文件，除了这个还有一个重大语法错误就是文件的开始需要有 begin，并且文件的结尾需要有 end，缺少任何一个都会报重大错误（后面会有例子）。

Err 文件：

```
***LINE 3: Fatal Error: missing: "function"

```

4.第三次修改（readm）;）：

Pas 文件：

```
begin
  integer k;
  integer function F(n);
    begin
      integer n;
      if n<=0 then F:=1
      else F:=n*F(n-1)
    end;
  readm);
  k:=F(m);
  write(k)
end
```

Var 文件：

```
          vname            vproc  vkind            vtype  vlev  vadr
              k             main      0          INTEGER     0     0
              n                F      1          INTEGER     1     1
              n                F      0          INTEGER     1     2

```

Pro 文件：

```
          pname            ptype  plev  fadr  ladr
           main             void     0     0     0
              F          INTEGER     1     1     2
```

Err 文件：

```
***LINE 9: readm is not defined
***LINE 10: m is not defined

```

Readm 是一个未定义的变量，理论上接下来要匹配:=，但是实际上为）,接下来的处理就很复杂了，此处第 9 行报错为 readm 为定义。第 10 行的 m 也未定义。

5.第四次修改（第九行后加一行 integer m;）：

Pas 文件：

```
 1 begin
 2   integer k;
 3   integer function F(n);
 4     begin
 5       integer n;
 6       if n<=0 then F:=1
 7       else F:=n*F(n-1)
 8     end;
 9   read(m);
10   integer m;
11   k:=F(m);
12   write(k)
13 end
```

Var 文件：

```
 1           vname            vproc  vkind            vtype  vlev  vadr
 2               k             main      0          INTEGER     0     0
 3               n                F      1          INTEGER     1     1
 4               n                F      0          INTEGER     1     2
 5
```

Pro 文件：

```
 1           pname            ptype  plev  fadr  ladr
 2            main             void     0     0     0
 3               F          INTEGER     1     1     2
 4
```

Err 文件

```
1 ***LINE 9: m is not defined
2 ***LINE 10: Invalid execution statement
3 ***LINE 11: m is not defined
4
```

第 9，11 行的 m 都未定义，此处匹配的应该为执行语句，但未匹配<readStatement>|<writeStatement>|<assignStatement>|<conditionalStatement>的任意一个，为无效的赋值语句。

6.（修改 1，3，4 组合的文件）：

Pas 文件：

```
begin
  intger m;
  integer function F(n);
    begin
      integer n;
      if n<=0 then F:=1
      else F:=n*F(n-1)
    end;
  readm);
  integer m;
  k:=F(m);
  write(k)
end
```

Var 文件：

```
          vname            vproc  vkind            vtype  vlev  vadr
              n                F      1          INTEGER     1     0
              n                F      0          INTEGER     1     1

```

Pro 文件：

```
          pname            ptype  plev  fadr  ladr
           main             void     0     0     0
              F          INTEGER     1     0     1

```

Err 文件：

```
***LINE 2: missing "integer"
***LINE 9: readm is not defined
***LINE 10: Invalid execution statement
***LINE 11: k is not defined
***LINE 11: m is not defined
***LINE 12: k is not defined

```

组合情况已在上面分步阐述完毕。

7.（其他重大错误，开头缺少 begin）：

Pas 文件：

```
integer k;
integer function F(n);
  begin
    integer n;
    if n<=0 then F:=1
    else F:=n*F(n-1)
  end;
read(m);
k:=F(m);
write(k)
end
```

Err 文件：

```
***LINE 1: Fatal Error: missing "begin"

```

可以对源程序进行语法分析，图中给出了出错行数及出错类型。

十、 实验结论

本实验程序较好地完成了递归下降分析器的设计与实现，能够对所给文法的程序进行语法分析，生成变量名表和过程名表，如果源程序有语法错误则给出出错类型及所在行数。

十一、总结及心得体会

通过该实验，加深了自己对语法分析程序的原理的理解与掌握，提高了自己的动手能力。对递归下降分析程序的设计，以及运用 JAVA 语言进行编程有了更深刻的理解，同时提高了自己的写代码水平。

十二、对本实验过程及方法、手段的改进建议

程序设计合理，代码可进一步优化，可进一步提高程序的健壮性。

报告评分：

指导教师签字：

语法分析代码：

Syntactic_Analysis 文件：

```
import java.io.*;
import java.nio.charset.StandardCharsets;

/**
 * @author YiGeDian
 * @date 2021/6/9 10:08
 */
public class Syntactic_Analysis {
    public static Syntactic_Table.VarTable[] varTables=new Syntactic_Table.VarTable[100];
    public static Syntactic_Table.ProTable[] proTables=new Syntactic_Table.ProTable[100];
    String[] words=new String[150];      //用来存放二元式中的单词
    String currentWord;                  //当前单词
    String[] types=new String[150];      //用来存放二元式中的种类
    String currentType;                  //当前种类
    int line=0;                          //当前运行的行数
    int level=0;                         //程序的层次
    int proNumber=0;                     //程序的数目
    int address=0;                       //变量在变量表中的位置
    String lastWord;                     //上一个单词
    String lastType;                     //上一个单词类型
    int index=0;                         //当前单词的位置
    String err;                          //出错信息
    int errNumber=0;                     //出错数量
    int errFlag=0;                       //出错标志位
    int declareFlag=0;                   //说明语句标志位
    int proFlag=0;                       //程序标志位
    String form;                         //变量类型
    int isArgument;                      //是否为参数

    /**
     * 初始化变量名表
     * @param file
     * @throws IOException
     */
    public void initializeVarFile(File file) throws IOException {
        FileWriter fileWriter_var = new FileWriter(file,true);
        fileWriter_var.write(String.format("%16s", "vname") + "    ");
        fileWriter_var.write(String.format("%16s", "vproc") + "    ");
```

```
            fileWriter_var.write(String.format("%5s", "vkind") + "    ");
            fileWriter_var.write(String.format("%16s", "vtype") + "    ");
            fileWriter_var.write(String.format("%4s", "vlev") + "    ");
            fileWriter_var.write(String.format("%4s", "vadr") + "\n");
            fileWriter_var.close();
        }

        /**
         * 初始化过程名表
         * @param file
         * @throws IOException
         */
        public void initializeProFile(File file) throws IOException {
            FileWriter fileWriter_pro = new FileWriter(file, true);
            fileWriter_pro.write(String.format("%16s", "pname") + "    ");
            fileWriter_pro.write(String.format("%16s", "ptype") + "    ");
            fileWriter_pro.write(String.format("%4s", "plev") + "    ");
            fileWriter_pro.write(String.format("%4s", "fadr") + "    ");
            fileWriter_pro.write(String.format("%4s", "ladr") + "\n");
            fileWriter_pro.close();
        }

        /**
         * 将*.dyd 初始化
         * @param file
         * @throws IOException
         */
        public void initializeDydFile(File file) throws IOException {
            FileInputStream fileInputStream = new FileInputStream(file);
            InputStreamReader inputStreamReader = new InputStreamReader(fileInputStream,
StandardCharsets.UTF_8);
            BufferedReader bufferedReader = new BufferedReader(inputStreamReader);
            String readline;
            int i=0;
            while ((readline =bufferedReader.readLine()) != null) {
                words[i]=readline.substring(0,16).trim();
                types[i]=readline.substring(17,19);
                i++;
            }
            currentWord=words[0];
            currentType=types[0];
            varTables[0]=new Syntactic_Table.VarTable("","","",0,0,0);
            bufferedReader.close();
            inputStreamReader.close();
```

```
        fileInputStream.close();
    }

    /**
     * 当前变量写入写*.dys 文件
     * @param file
     * @param word
     * @param type
     * @throws IOException
     */
    public void writeDysFile(File file,String word,String type) throws IOException {
        FileWriter fileWriter_dys = new FileWriter(file,true);
        fileWriter_dys.write(String.format("%16s", word) + " ");
        if (word.equals("EOF")){
            fileWriter_dys.write(type);
        }else {
            fileWriter_dys.write(type+"\n");
        }
        fileWriter_dys.close();
    }

    /**
     * 错误信息写入*.err 文件
     * @param file
     * @param err
     * @throws IOException
     */
    public void writeErrFile(File file, String err) throws IOException {
        FileWriter fileWriter_err = new FileWriter(file,true);
        fileWriter_err.write("***LINE "+line+": "+err+"\n");
        fileWriter_err.close();
        errNumber++;
        if (err.indexOf("Fatal")!=-1) {//重大错误退出
            System.exit(0);
        }
    }

    /**
     * 变量表的内容写入*.var 文件
     * @param file
     * @throws IOException
     */
    public void writeVarFile(File file) throws IOException {
        FileWriter fileWriter_Var = new FileWriter(file,true);
```

```
            for (int n = 0; n < address; n++) {
                fileWriter_Var.write(String.format("%16s", varTables[n].vname) + "    ");
                fileWriter_Var.write(String.format("%16s", varTables[n].vproc) + "    ");
                fileWriter_Var.write(String.format("%5s", varTables[n].vkind) + "    ");
                fileWriter_Var.write(String.format("%16s", varTables[n].vtype) + "    ");
                fileWriter_Var.write(String.format("%4s", varTables[n].vlev) + "    ");
                fileWriter_Var.write(String.format("%4s", varTables[n].vadr) + "\n");
            }
            fileWriter_Var.close();
        }

        /**
         * 过程表的内容写入*.pro 文件
         * @param file
         * @throws IOException
         */
        public void writeProFile(File file) throws IOException {
            FileWriter fileWriter_Pro = new FileWriter(file,true);
            for (int n = 0; n < proNumber; n++) {
                fileWriter_Pro.write(String.format("%16s", proTables[n].pname) + "    ");
                fileWriter_Pro.write(String.format("%16s", proTables[n].ptype) + "    ");
                fileWriter_Pro.write(String.format("%4s", proTables[n].plev) + "    ");
                fileWriter_Pro.write(String.format("%4s", proTables[n].fadr) + "    ");
                fileWriter_Pro.write(String.format("%4s", proTables[n].ladr) + "\n");
            }
            fileWriter_Pro.close();
        }

        /**
         * 进行预测
         * @param fileDys
         * @param fileErr
         * @param word
         * @param type
         * @param err
         * @throws IOException
         */
        public void forecast(File fileDys,File fileErr,String word,String type,String err) throws
IOException {
            if (currentWord.equals(word)){
                errFlag=0;
            }else{
                writeErrFile(fileErr,err);
                writeDysFile(fileDys,currentWord,currentType);
```

```
            currentWord=word;
            currentType=type;
            errFlag=1;
        }
    }

    /**
     * 读入下一个单词
     * @throws IOException
     */
    public void readNextWord() throws IOException {
        File file_dys = new File("source.dys");
        lastWord=currentWord;
        lastType=currentType;
        index++;
        currentWord=words[index];
        currentType=types[index];
        if (currentWord.equals("EOLN")){
            writeDysFile(file_dys,currentWord,currentType);
            index++;
            currentWord=words[index];
            currentType=types[index];
            line++;
            if (currentWord.equals("EOF")){//解决空行问题
                writeDysFile(file_dys,currentWord,currentType);
                currentWord="";
                currentType="";
            }
        }else if(currentWord.equals("EOF")){
            writeDysFile(file_dys,currentWord,currentType);
            currentWord="";
            currentType="";
        }
    }

    /**
     * <program>-><subProgram>
     * @throws IOException
     */
    public void program() throws IOException {
        line++;
        proTables[0]= new Syntactic_Table.ProTable();
        proTables[0].pname="main";
        proTables[0].plev=level; //level=1
```

```
        proTables[0].ptype="void";
        proNumber++;
        subProgram();

        int fadr=100;
        int ladr=0;
        int tempadr=0;
        for (int n = 0; n < address; n++) {
            if (varTables[n].vlev==0) {//main 函数中的变量
                tempadr=varTables[n].vadr;
                if (tempadr<fadr) {
                    proTables[0].fadr=tempadr;
                    fadr=tempadr;
                }
            }
        }
        if (tempadr>ladr) {
            proTables[0].ladr=tempadr;
            ladr=tempadr;
        }
    }

    /**
     * <subProgram>->begin<declareStatementList>;<executeStatementList>end
     * @throws IOException
     */
    public void subProgram() throws IOException {
        File file_dys = new File("source.dys");
        File file_err = new File("source.err");
        if (currentWord.equals("begin")) {
            writeDysFile(file_dys,currentWord, currentType);
            readNextWord();
            declareStatementList();
            if(lastWord.equals(";")) {
                executeStatementList();
                if (currentWord.equals("end")) {
                    writeDysFile(file_dys,currentWord, currentType);
                    readNextWord();
                }else{
                    err="Fatal Error: missing: \"end\"";
                    writeErrFile(file_err,err);
                    readNextWord();
                }
            }else{
```

```
                    if (!lastWord.equals("function")){
                        err="Fatal Error: missing: \"function\"";
                        writeErrFile(file_err,err);
                    }
                    err="Error: missing: \";\"";
                    writeErrFile(file_err,err);
                    readNextWord();
                }
            }else{
                err="Fatal Error: missing \"begin\"";
                writeErrFile(file_err,err);
            }
        }

        /**
         * <declareStatementList>-><declareStatement><declareStatementList_L()>
         * @throws IOException
         */
        public void declareStatementList() throws IOException {
            File file_err = new File("source.err");
            File file_dys = new File("source.dys");
            forecast(file_dys,file_err,"integer", "03", "missing \"integer\"");
            declareStatement();
            declareStatementList_L();
        }

        /**
         * <declareStatementList_L>->;<declareStatement><declareStatementList_L>
         * @throws IOException
         */
        public void declareStatementList_L() throws IOException {
            if (currentWord.equals(";")) {
                File file_dys = new File("source.dys");
                writeDysFile(file_dys,currentWord, currentType);
                readNextWord();
                declareStatement();
                declareStatementList_L();
            }
        }

        /**
         * <declareStatement>->integer<variable>|integer function
<identifier>(<parameter>);<functionBody>
         * <declareStatement>-><declareVariable>|<declareFunction>
```

```
     * <declareVariable>->integer <variable>
     * <declareFunction>->integer function <identifier>(<parameter>);<functionBody>
     * @throws IOException
     */
    public void declareStatement() throws IOException {
        File file_dys = new File("source.dys");
        File file_err = new File("source.err");
        declareFlag=1;
        if (currentWord.equals("integer")) {
            if (errFlag==0) {
                writeDysFile(file_dys,currentWord,currentType);
            }
            form=currentWord.toUpperCase();
            readNextWord();
            if (currentWord.equals("function")) {
                level++;
                proFlag=1;
                writeDysFile(file_dys,currentWord,currentType);
                readNextWord();
                identifier();
                forecast(file_dys,file_err,"(","21", "missing: \"(\"");
                if (currentWord.equals("(")) {
                    if (errFlag==0) {
                        writeDysFile(file_dys,currentWord,currentType);
                    }else {

                    }
                    errFlag=0;
                    readNextWord();
                    isArgument=1;
                    parameter();
                    forecast(file_dys,file_err,")","22", "missing: \")\"");
                    if (currentWord.equals(")")) {
                        if (errFlag==0) {
                            writeDysFile(file_dys,currentWord,currentType);
                        }else {

                        }
                        errFlag=0;
                        readNextWord();
                        forecast(file_dys,file_err,";","23", "missing: \";\"");
                        if (currentWord.equals(";")) {
                            if (errFlag==0) {
                                writeDysFile(file_dys,currentWord,currentType);
```

```
                                }else {

                                }
                                errFlag=0;
                                readNextWord();
                                functionBody();
                            }
                        }
                    }
                }else {
                    variable();
                }
            }
            declareFlag=0;
        }

        /**
         * <variable>-><identifier>
         * @throws IOException
         */
        public void variable() throws IOException {
            identifier();
        }
        /**
         * <identifier>->currentType=10
         * <identifier>-><letter> | <identifier><letter> | <identifier><digit>
         * @throws IOException
         */
        public void identifier() throws IOException {
            File file_dys = new File("source.dys");
            File file_err = new File("source.err");
            if (currentType.equals("10")) {
                if (declareFlag!=1) {
                    int checkFlag=0;
                    for (int n = 0; n < proNumber; n++) {
                        if (proTables[n].pname.equals(currentWord)) {
                            writeDysFile(file_dys,currentWord,currentType);
                            readNextWord();
                            checkFlag=1;
                        }
                    }
                    for (int n = 0; n < address; n++) {
                        if (varTables[n].vname.equals(currentWord)) {
                            writeDysFile(file_dys,currentWord,currentType);
```

```
                        readNextWord();
                        checkFlag=1;
                    }
                }
                if (checkFlag==0) {
                    err=currentWord+ " is not defined";
                    writeErrFile(file_err,err);
                    writeDysFile(file_dys,currentWord,currentType);
                    readNextWord();
                }
            }
            if (declareFlag==1) {
                if (proFlag==1) {
                    int checkFlag=0;
                    for (int n = 0; n < proNumber; n++) {
                        if (proTables[n].pname.equals(currentWord)) {
                            checkFlag=1;
                        }
                    }
                    if (checkFlag==0) {
                        proTables[proNumber]=new
Syntactic_Table.ProTable(currentWord,form,level,0,0);
                        proNumber++;
                        writeDysFile(file_dys,currentWord,currentType);
                        readNextWord();
                    }else {
                        err="duplicate local variable "+currentWord;
                        writeErrFile(file_err,err);
                        readNextWord();
                    }
                    proFlag=0;
                }else {
                    if (errFlag==0){
                        int checkFlag=0;
                        for (int n = 0; n < proNumber; n++) {
                            if (proTables[n].pname.equals(currentWord)) {
                                checkFlag=1;
                            }
                        }
                        for (int n = 0; n < address; n++) {
                            if
(varTables[n].vname.equals(currentWord)&&varTables[n].vlev==level&&varTables[n].vkind=
=isArgument) {
                                checkFlag=1;
```

```
                                }
                            }
                            if (checkFlag==0) {
                                varTables[address]=new
Syntactic_Table.VarTable(currentWord,"",form,isArgument,level,address);
                                for (int n = 0; n < proNumber; n++) {
                                    if (proTables[n].plev==level) {
                                        varTables[address].vproc=proTables[n].pname;
                                    }
                                }
                                address++;
                                isArgument=0;
                                writeDysFile(file_dys,currentWord,currentType);
                                readNextWord();
                            }else {
                                err="duplicate local variable "+currentWord;
                                writeErrFile(file_err,err);
                                readNextWord();
                            }
                        }
                        else{
                            errFlag=0;
                            writeDysFile(file_dys,currentWord,currentType);
                            readNextWord();
                        }
                    }
                }
            }else{
                err="Invalid execution statement";
                writeErrFile(file_err,err);
                writeDysFile(file_dys,currentWord,currentType);
                readNextWord();
            }
        }

        /**
         * <parameter>-><variable>
         * @throws IOException
         */
        public void parameter() throws IOException {
            variable();
        }

        /**
```

```
     * <functionBody>->begin<declareStatementList>;<executeStatementList>end
     * @throws IOException
     */
    public void functionBody() throws IOException {
        File file_dys = new File("source.dys");
        File file_err = new File("source.err");
        if (currentWord.equals("begin")) {
            writeDysFile(file_dys,currentWord,currentType);
            readNextWord();
            declareStatementList();
            if (lastWord.equals(";")) {
                executeStatementList();
                if (currentWord.equals("end")) {
                    writeDysFile(file_dys,currentWord,currentType);
                    int fadr=100;
                    int ladr=0;
                    int tempadr=0;
                    for (int n = 0; n < address; n++) {
                        if (varTables[n].vlev==level) {
                            tempadr=varTables[n].vadr;
                            if (tempadr<fadr) {
                                proTables[level].fadr=tempadr;
                                fadr=tempadr;
                            }
                        }
                    }
                    if (tempadr>ladr) {
                        proTables[level].ladr=tempadr;
                        ladr=tempadr;
                    }
                    readNextWord();
                    level--;
                }else{
                    err="missing: \"end\"";
                    writeErrFile(file_err,err);
                }
            }
        }else{
            err="Fatal Error: missing \"begin\"";
            writeErrFile(file_err,err);
        }
    }

    /**
```

```
     * <executeStatementList>-><executeStatement><executeStatementList_L>
     * @throws IOException
     */
    public void executeStatementList() throws IOException {
        executeStatement();
        executeStatementList_L();
    }

    /**
     * <executeStatementList_L>->;<executeStatement><executeStatementList_L>
     * @throws IOException
     */
    public void executeStatementList_L() throws IOException {
        File file_dys = new File("source.dys");
        if (currentWord.equals(";")) {
            writeDysFile(file_dys,currentWord,currentType);
            readNextWord();
            executeStatement();
            executeStatementList_L();
        }
    }

    /**
     *
<executeStatement>->read(<variable>)|write(<variable>)|if<conditionalExpression>then<executeS
tatement>else<executeStatement>|<variable>:=<arithmeticalExpression>
     *
<executeStatement>-><readStatement>|<writeStatement>|<assignStatement>|<conditionalStatem
ent>
     * <readStatement>->read(<variable>)
     * <writeStatement>->write(<variable>)
     *
<conditionalStatement>->if<conditionalExpression>then<executeStatement>else<executeStatemen
t>
     * <assignStatement>-><variable>:=<arithmeticalExpression>
     * @throws IOException
     */
    public void executeStatement( ) throws IOException {
        File file_dys = new File("source.dys");
        File file_err = new File("source.err");
        if (currentWord.equals("read")) {
            writeDysFile(file_dys,currentWord,currentType);
            readNextWord();
            forecast(file_dys,file_err,"(","21", err="missing: \"(\"");
```

```
            if (currentWord.equals("(")) {
                if (errFlag==0) {
                    writeDysFile(file_dys,currentWord,currentType);
                }else {

                }
                errFlag=0;
                readNextWord();
                variable();
                forecast(file_dys,file_err,")","22", err="missing: \")\"");
                if (currentWord.equals(")")) {
                    if (errFlag==0) {
                        writeDysFile(file_dys,currentWord,currentType);
                    }else {

                    }
                    errFlag=0;
                    readNextWord();
                }
            }
        }else if (currentWord.equals("write")) {
            writeDysFile(file_dys,currentWord,currentType);
            readNextWord();
            forecast(file_dys,file_err,"(","21", err="missing: \"(\"");
            if (currentWord.equals("(")) {
                if (errFlag==0) {
                    writeDysFile(file_dys,currentWord,currentType);
                }else {

                }
                errFlag=0;
                readNextWord();
                isArgument=1;
                variable();
                forecast(file_dys,file_err,")","22", err="missing: \")\"");
                if (currentWord.equals(")")) {
                    if (errFlag==0) {
                        writeDysFile(file_dys,currentWord,currentType);
                    }else {

                    }
                    errFlag=0;
                    readNextWord();
                }
```

```
            }
        }else if (currentWord.equals("if")) {
            writeDysFile(file_dys,currentWord,currentType);
            readNextWord();
            conditionalExpression();
            forecast(file_dys,file_err,"then","05", err="missing: \"then\"");
            if (currentWord.equals("then")) {
                if (errFlag==0) {
                    writeDysFile(file_dys,currentWord,currentType);
                }else {

                }
                errFlag=0;
                readNextWord();
                executeStatement();
                forecast(file_dys,file_err,"else","06", err="missing: \"else\"");
                if (currentWord.equals("else")) {
                    if (errFlag==0) {
                        writeDysFile(file_dys,currentWord,currentType);
                    }else {

                    }
                    errFlag=0;
                    readNextWord();
                    executeStatement();
                }
            }
        }else {
            variable();
            if (currentWord.equals(":=")) {
                writeDysFile(file_dys,currentWord,currentType);
                readNextWord();
                arithmeticalExpression();
            }else {
                writeDysFile(file_dys,currentWord,currentType);
                readNextWord();
            }
        }
    }

    /**
     *
<conditionalExpression>-><arithmeticalExpression><relationalOperator><arithmeticalExpression>
     * @throws IOException
```

```
     */
    private void conditionalExpression() throws IOException {
        arithmeticalExpression();
        relationalOperator();
        arithmeticalExpression();
    }

    /**
     * <relationalOperator>->12<=type<=17
     * <relationalOperator>-><|<=|>|>=|=|<>
     * @throws IOException
     */
    private void relationalOperator() throws IOException {
        File file_dys = new File("source.dys");
        File file_err = new File("source.err");
        int num=Integer.parseInt(currentType);
        if (num>=12&&num<=17) {
            writeDysFile(file_dys,currentWord,currentType);
            readNextWord();
        }else{
            err="Invalid Symbol.";
            writeErrFile(file_err,err);
        }

    }

    /**
     * <arithmeticalExpression>-><item><arithmeticalExpression_E>
     * @throws IOException
     */
    private void arithmeticalExpression() throws IOException {
        item();
        arithmeticalExpression_E();
    }

    /**
     * <arithmeticalExpression_E>->-<item><arithmeticalExpression_E>
     * @throws IOException
     */
    private void arithmeticalExpression_E() throws IOException {
        File file_dys = new File("source.dys");
        if (currentWord.equals("-")) {
            writeDysFile(file_dys,currentWord,currentType);
            readNextWord();
```

```
            item();
            arithmeticalExpression_E();
        }
    }

    /**
     * <item>-><factor><item_I>
     * @throws IOException
     */
    private void item() throws IOException {
        factor();
        item_I();
    }

    /**
     * <item_I>->*<factor><item_I>
     * @throws IOException
     */
    private void item_I() throws IOException {
        File file_dys = new File("source.dys");
        if (currentWord.equals("*")) {
            writeDysFile(file_dys,currentWord,currentType);
            readNextWord();
            factor();
            item_I();
        }
    }

    /**
     *    <factor>->type=10<variable>(<arithmeticalExpression>)|type=10<variable>|type=11
     *    <factor>-><variable>|<constant>|<functionCall>;
     *    <functionCall>->identifier(<arithmeticalExpression>);
     *    <constant>-><unsignedInteger>
     *    <unsignedInteger>-><digit>|<unsignedInteger><digit>
     * @throws IOException
     */
    private void factor() throws IOException {
        File file_dys = new File("source.dys");
        if (currentType.equals("10")) {
            variable();
            if (currentWord.equals("(")) {
                writeDysFile(file_dys,currentWord,currentType);
                readNextWord();
                arithmeticalExpression();
```

```
                if (currentWord.equals(")")) {
                    writeDysFile(file_dys,currentWord,currentType);
                    readNextWord();
                }
            }
        }else if (currentType.equals("11")) {
            writeDysFile(file_dys,currentWord,currentType);
            readNextWord();
        }
    }

    public static void main(String[] args) throws IOException {
        File file_dyd = new File("source.dyd");
        File file_err = new File("source.err");
        File file_dys = new File("source.dys");
        File file_var = new File("source.var");
        File file_pro = new File("source.pro");
        //不存在,则创建文件
        if ((!file_dyd.exists())||(!file_dyd.isFile())){
            file_dyd.createNewFile();
        }
        if ((!file_err.exists())||(!file_err.isFile())){
            file_err.createNewFile();
        }
        if ((!file_dys.exists())||(!file_dys.isFile())){
            file_dys.createNewFile();
        }
        if ((!file_var.exists())||(!file_var.isFile())){
            file_var.createNewFile();
        }
        if ((!file_pro.exists())||(!file_pro.isFile())){
            file_pro.createNewFile();
        }
        /*写初始化，每次写会重写文件内容*/
        FileWriter fileWriter_var=new FileWriter(file_var);
        FileWriter fileWriter_pro=new FileWriter(file_pro);
        FileWriter fileWriter_dys=new FileWriter(file_dys);
        FileWriter fileWriter_err=new FileWriter(file_err);
        fileWriter_var.write("");
        fileWriter_pro.write("");
        fileWriter_dys.write("");
        fileWriter_err.write("");
        Syntactic_Analysis syntacticAnalysis = new Syntactic_Analysis();
        syntacticAnalysis.initializeVarFile(file_var);
```

```
            syntacticAnalysis.initializeProFile(file_pro);
            syntacticAnalysis.initializeDydFile(file_dyd);
            syntacticAnalysis.program();
            syntacticAnalysis.writeVarFile(file_var);
            syntacticAnalysis.writeProFile(file_pro);
        }
}
```

Syntactic_Table 文件：

```
/**
 * @author YiGeDian
 * @date 2021/6/9 16:10
 */
public class Syntactic_Table {



    static class VarTable{
        String vname;//变量名
        String vproc;//所说过程
        String vtype;//分类，0 变量，1 形参
        int vkind;//变量类型
        int vlev;//变量层次
        int vadr;//变量在变量表中的位置，相对第一个变量而言
        VarTable() {}
        VarTable(String vname, String vproc, String vtype, int vkind, int vlev, int vadr) {
            this.vname = vname;
            this.vproc = vproc;
            this.vtype = vtype;
            this.vkind = vkind;
            this.vlev = vlev;
            this.vadr = vadr;
        }
    }

    static class ProTable{
        String pname;
        String ptype;
        int plev;
        int fadr;
        int ladr;
        ProTable() {}
        ProTable(String pname, String ptype, int plev, int fadr, int ladr) {
            this.pname = pname;
            this.ptype = ptype;
```

```
                this.plev = plev;
                this.fadr = fadr;
                this.ladr = ladr;
            }
        }

}
```