词法分析程序说明文档

141250149 吴秦月

目录

1. 实验目的

编写、调试一个词法分析程序，并对语句进行词法分析，掌握记号、模式与单词，掌握正规式与正规集，掌握有限自动机，掌握如何从正规式到词法分析器的各种算法，从而更好理解词法分析原理。

2. 内容描述

此程序用 Java 编写。

1） input：程序读取一个文本文件，并对其中内容进行词法分析，此程序实现了对 java 程序的词法识别，可识别关键字、标识符（关键字优先于标识符）、操作符、分隔符 、整数、注释符、浮点数

2） output：输出格式为：种类，识别的单词符号的 TOKEN 序列

3） error 报告：对未识别字符、整型过大（超过 java 最大的 4 字节）、文件过大进行异常报错。

3. 思路方法

1) 对要识别的单词符号写出正则表达式 RE

2) 根据正则表达式 RE 写出对应的 NFA

3) 合并所有 NFA 并化简为 DFA

4) 最小化 DFA

5) 基于最小化 DFA 编程。

4. 假设

假设输入的文件内容是正常的 java 程序，即包含合法的关键字和操作符。

Token 分为标识符、关键字、操作符、分隔符、整数和浮点数。

标识符：identifier = letter (letter | digit)*

关键字：

public|class|static|void|main|while|if|else|for|switch|case|package|String|int|float

|double|char|boolean|true|false；

操作符：[,],~,(,),*,/,.,’ ,” ,+,-,<,>,=,!,&,|,+=,-=,<=,>=,!=,&&,|| ,++,--；

分隔符：;|{|} | ，;

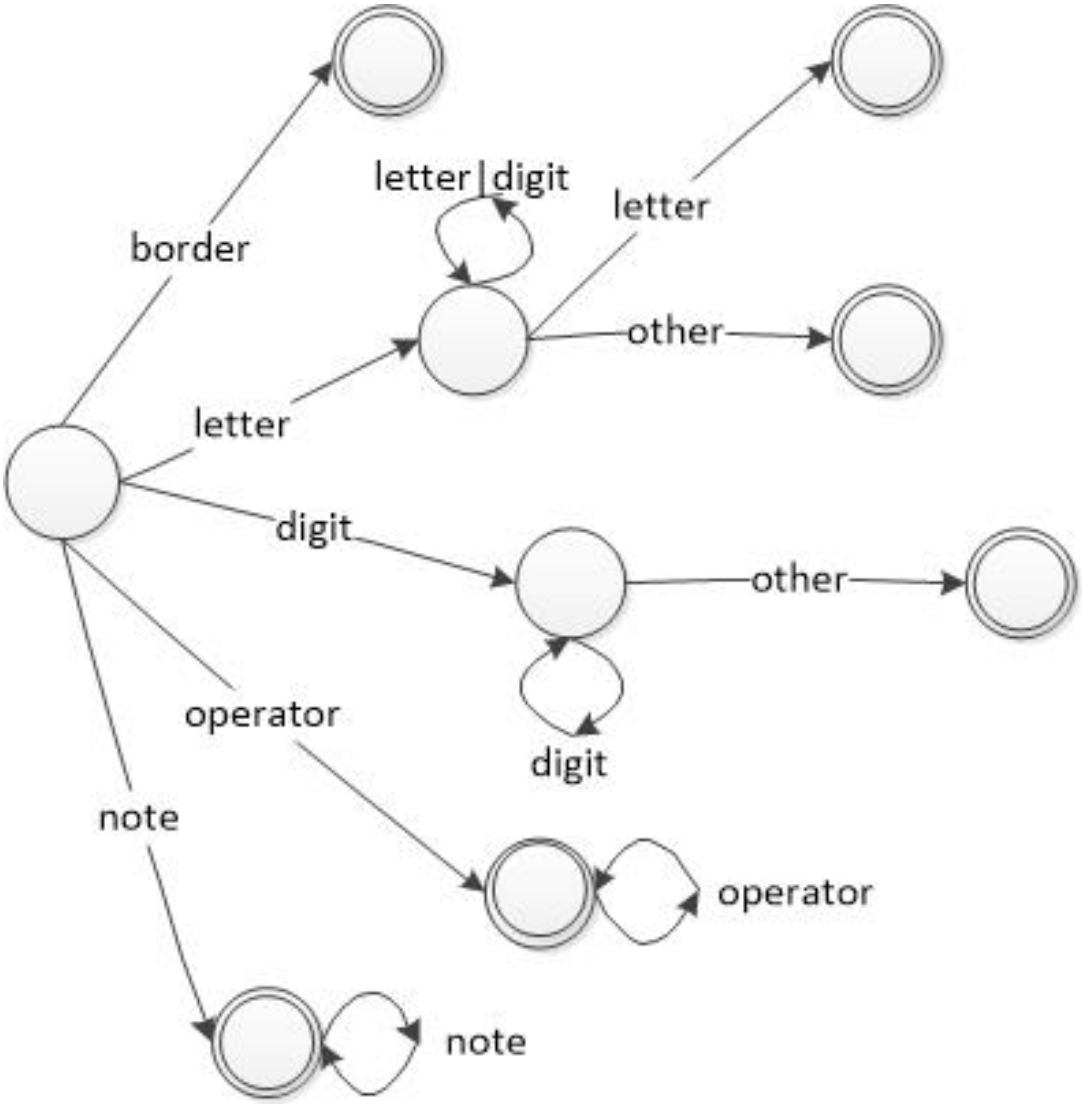
注释符：//|/*|*/；

整数：digit (digit)*

浮点数：digit (digit)*. digit (digit)*

另外，空格、\n、\t、\r 在词法分析阶段忽略。

5. 相关 FA 描述



ID->letter (letter | digit)*

Int->digit digit*

Double->digit digit* . digit digit*

digit->0|1|2|3|4|5|6|7|8|9

letter->a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x

|y|z

6. 重要数据结构

```
    //存储输入的字符数组
    private static char input[] = new char[600];
    private static int p = 0;//input下标
    //存储输出的token 序列
    private static ArrayList<Token> output = new ArrayList<>();
    //单词符号
    private static char word[];
    private static char ch;// 当前读的字符
    private static String code;// 单词种别码
```

```
  private static String[] keyWords = {"class", "public", "protected",
          "private", "void", "static", "int", "char", "float", "double",
          "if", "else", "do", "while", "try", "catch", "switch",
          "case", "for", "main", "String","boolean","true","false"};
```

//对于每个输出都实例化一个 token 对象，并加入输出链表

```
public class Token {
    //标识码
    private String code;
    //单词符号
    private String str;

    private String error;

    public Token(String code,String str){
       this.code=code;
        this.str=str;
        this.error=null;
    }

    public Token(String error) { this.error=error; }

    public String toString(){
        if(this.error!=null){
            return "Error:"+this.error;
        }
        return this.code+" : "+this.str;
    }
}
```

7. 核心算法

程序主要的方法有：

readFile()——读取文件得到输入

scanner()——扫描输入进行分析，核心算法

output()——将结果输出到控制台和输出文件中

下面介绍 scanner()方法：

Scanner()是该程序分析词法的核心方法，此方法每次只能识别一个单词符号，所以程序通过反复调用 scanner()方法分析整个输入。

通过读取的第一个字符的类型，预测接下来的单词符号可能的类型。读到英文字符，可能为保留字或变量名（类型一）；读到数字，就是常数（正数）（类型二）；读到其他字符（类型三），则可能是操作符或边界符或注释符，如果是‘-’符，后面是数字就组成了负数，当然也可能是换行符或是未定义的字符。

类型一，则继续读取，每读一位，都判断是否属于保留字，若是就直接输出（因为保留字优先于变量名），否则一直读到不是英文字符为止，此时需要退回一位，并输出变量名。

类型二，一直读到不是数字为止，同样需要退回一位，并输出正常数。

类型三相对复杂一点，如果单字符就可以确定种别，则直接输出；否则继续读取下一位，直到可以确定种别为止，若负号（减号）后面是数字，按类型二读取，最后需要输出负数。同样多读的需要退回。

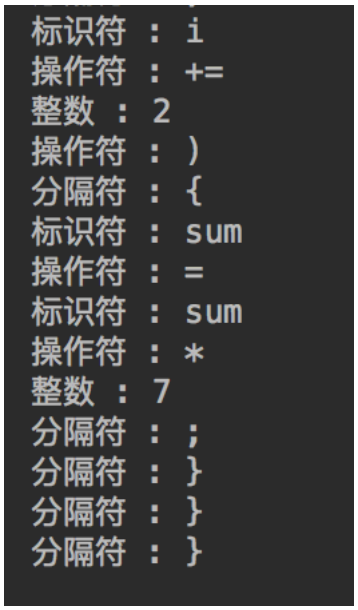
8. 运行截图

Input.txt：

```
public static void main(String[] args){
    int theHeightOfTA = 99999999999999999999999999999999999999999999999999999999999999999999999999999;
    boolean isTANice = true;
    int sum = 0;
    for(int i = 0;i<10;i+=2){
    sum = sum * 7;
    }
    }
}
```

Output：

```
关键字 : public
关键字 : static
关键字 : void
关键字 : main
操作符 : (
关键字 : String
操作符 : [
操作符 : ]
标识符 : args
操作符 : )
分隔符 : {
关键字 : int
标识符 : theHeightOfTA
操作符 : =
Error:integer overflow
分隔符 : ;
关键字 : boolean
标识符 : isTANice
操作符 : =
关键字 : true
分隔符 : ;
关键字 : int
标识符 : sum
操作符 : =
整数 : 0
分隔符 : ;
关键字 : for
操作符 : (
关键字 : int
标识符 : i
操作符 : =
整数 : 0
分隔符 : ;
标识符 : i
操作符 : <
整数 : 10
分隔符 : ;
```



并输出了 output .txt 文件，保存结果

9. 问题与解决

1) 控制台输入较麻烦，且不易有多行输入。所以进行读取文本来读取输入，这样对于读取源码文件也较为方便。

2) 对于 int 类型常数，如果超过了 max_value 则会变为负数，所以根据其结果是否为负，判断是否溢出。

10. 感受与总结

经过自己动手查找资料、编写简单的词法分析程序，有助于对词法分析过程和方法有更深入的理解，受益匪浅！