暨南大学本科实验报告专用纸

课程名称 编译原理 成绩评定

实验项目名称 词法分析 指导教师 王娜

实验项目编号 实验项目类型 综合 实验地点

学生姓名 陈宇 学号 2020101642

学院 信息科学技术学院 系 计算机系 专业 软件工程

实验时间 2023 年 3 月 20 日下午～3 月 20 日 下 午 温度 ℃湿度

实验目的

1. 理解词法分析在编译程序中的作用；
2. 加深对有穷自动机模型的理解；
3. 掌握词法分析程序的实现方法和技术。

实验内容

1. 选择部分 C 语言的语法成分，设计其词法分析程序。
2. 要求能够识别关键字、运算符、分界符、标识符、常量（至少是整型常量，可以自己扩充识别其他常量）等。
3. 并能处理注释、部分复合运算符（如>=等）。

实验要求

（1）待分析的简单的语法

关键字：begin if then while do end

运算符和界符：

:= + - * / < <= > >= <> = ; ( ) #

其他单词是标识符 id 和整型常数 num，通过以下正规式定义：

id=l(l|d)*

num=dd*

空格、注释：在词法分析中要去掉。

（2）各种单词符号对应的种别编码

单词符号	种别码	单词符号	种别码
begin	1	\0	1000
if	2	(	26
then	3	)	27
while	4	[	28
do	5	]	29
end	6	{	30
int	7	}	31
main	8	,	32
return	12	:	33
cout	13	;	34
l (l|d) *	10	>	35
:=	18	<	36
dd*	20	>=	37
==	21	<=	38
+	22	!=	40
-	23	“	41
*	24	#	0
/	25	!	-1

程序实现词法分析，从文件 data.txt 中读取一段小程序，分解出一个个的单词，其中有关键词，有界符、运算符等等，代码还需实现去掉空格、回车、注释等等情况，最后的输出结果是以单词二元组（单词种别码，单词自身的值）的形式输出。

实验环境

编程环境：

操作系统：Windows11，64 位

处理器：AMD Ryzen 7 5800HS

编程语言：java 11

编程思想：

面向对象

部分 c 语言文本：

```
int main()begin

    int value=10;  //定义一个变量

    int a = value;

    /*

        对 a 赋值

    */

    a:=a;

    return 0;

end
```

其中各种文件名和功能如下表所示：

文件名	类型	功能
FilReader.java	java 文件	用于实现读取文本文件，并且去掉注释功能
Analyzer.java	java 文件	用于实现词法分析器功能：拆分单词，识别关键字等功能

实验流程图

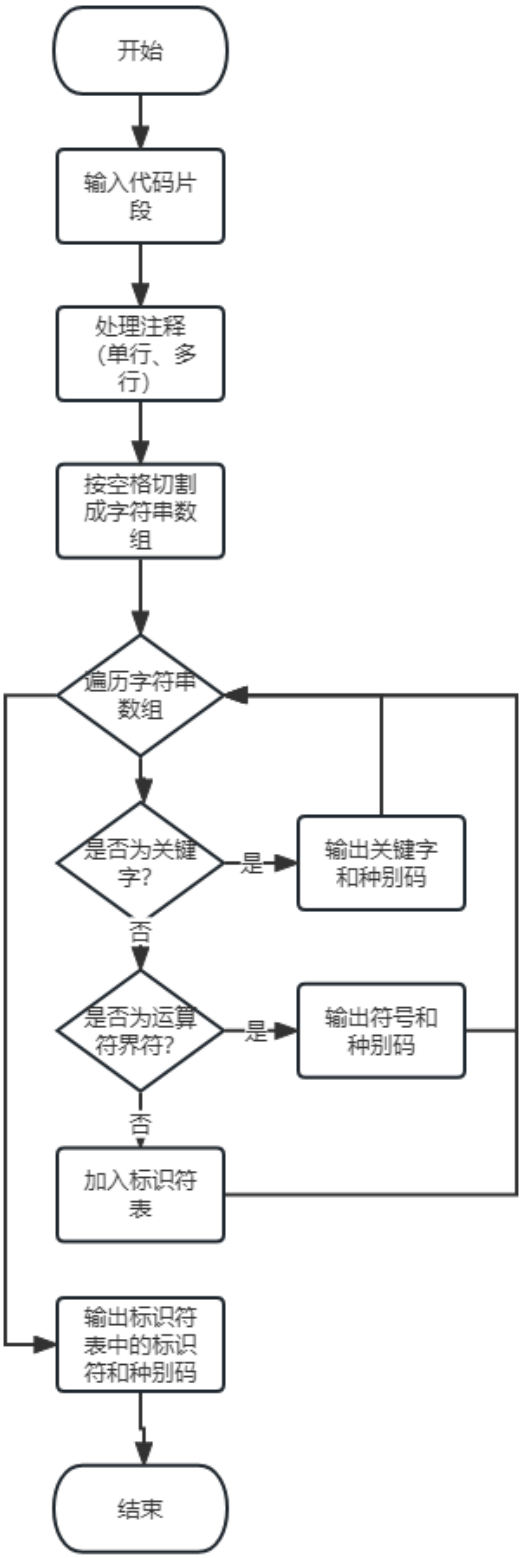

图 1 实验流程图

实验代码及结果

FileReader.java　　用于实现读取文本文件，并且去掉注释功能

```
import java.io.*;
import java.util.regex.Matcher;
import java.util.regex.Pattern;


/**
 * @author 小宇
 * @date {2023}-{04}-{20}:{22:20}
 * @preference: 类：大驼峰 方法：蛇形 变量：全小写
 * @description: 读取文件类
 */
public class FileReader {
    static private String text;    // 代码文件
    static public boolean readFile (String filePath)
{       // 读取文件
        // 返回的单词组
        try (FileInputStream fp = new
FileInputStream(filePath)) {
            // 如果找到文件则读取全部字符
            text = new String(fp.readAllBytes());
```

```
        } catch (IOException e) {
            // 否则返回空
            return false;
        }
        return true;
    }
    static public String outputText(){        // 输出文件
        FileReader.filter();
        return text;
    }

    static private void filter() {     //  正则表达式处理 单行注释 & 多行注释
        String pattern1 =
"/\\*([^\\*^\\/]*|[\\*^\\/*]*|[^\\**\\/]*)*\\*\\/";
        // 匹配 /**/ 注释
        String pattern2 = "\\/\\/[^\\r\n]*";
        // 匹配 // 注释

        Pattern p = Pattern.compile(pattern1);
        Matcher m = p.matcher(text);
        text = m.replaceAll(" ");
```

```
        // 去除 /**/ 注释
        p = Pattern.compile(pattern2);
        m = p.matcher(text);
        text = m.replaceAll(" ");
        // 去除 // 注释
    }



    public static void main(String[] args) {
        FileReader.readFile("code.text");
        FileReader.filter();
        System.out.println(FileReader.outputText());
    }
}
```

Analyzer.java 用于实现词法分析器功能：拆分单词，识别关键字等功能

```
import java.lang.reflect.Array;
import java.util.*;
import java.io.*;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
```

```
/**
 * @author 小宇
 * @date 2023-04-19:18:40
 * @preference: 类：大驼峰 方法：蛇形 变量：全小写
 * @description: 词法分析器
 */
public class Analyzer {
//    成员----------------------------
//   关键字表
    final public static Map<String, Integer> keyWord = new
HashMap<>();
    static {
        keyWord.put("begin",1);
        keyWord.put("if",2);
        keyWord.put("then",3);
        keyWord.put("while",4);
        keyWord.put("do", 5);
        keyWord.put("end", 6);
        keyWord.put("int", 7);
        keyWord.put("main", 8);
        keyWord.put("return", 9);
```

```
    }
//    界符&运算符
    final public static Map<String, Integer>
operatorDelimiter = new HashMap<>();
    static {
        operatorDelimiter.put(":=", 18);
        operatorDelimiter.put("+", 22);
        operatorDelimiter.put("-", 23);
        operatorDelimiter.put("*", 24);
        operatorDelimiter.put("/", 25);
        operatorDelimiter.put("<", 36);
        operatorDelimiter.put("<=",38);
        operatorDelimiter.put(">", 35);
        operatorDelimiter.put(">=", 37);
        operatorDelimiter.put("<>", 41);
        operatorDelimiter.put("=", 42);
        operatorDelimiter.put(";", 34);
        operatorDelimiter.put("(", 26);
        operatorDelimiter.put(")", 27);
        operatorDelimiter.put("#", 0);
    }
//    标识符表
```

```
    private List<String> identifierList = new
ArrayList<String>();



//   函数--------------------------------
//    判断是否为数字
    private boolean is_Digit(char ch){
        if(ch >= '0' && ch <='9'){
            return true;
        }
        return false;
    }
    private boolean is_Digit(String ch){
        if(ch.charAt(0) >= '0' && ch.charAt(0) <='9'){
            return true;
        }
        return false;
    }
//    判断是否为字母
    private  boolean is_Letter(char ch){
        if(ch >='a' && ch <= 'z' || ch >= 'A' && ch <= 'Z'){
            return true;
        }
```

```
        return false;
    }
//    判断是否为关键字
    private int is_Keyword(String str){
        if(keyWord.containsKey(str)){
            return keyWord.get(str);
        }
        return -1;
    }

//  判断是否为符号
    private int is_op_deli(String str){
        if(operatorDelimiter.containsKey(str)){
            return operatorDelimiter.get(str);
        }
        return -1;
    }

//      将符号和变量数字用空格分开
    private String regular(String text){
        char[] textlist = text.toCharArray();
        String record = "";
```

```
        int index = 0;
        for(int i = 0; i < textlist.length-1; ){
            String temp = textlist[i]
+Character.toString(textlist[i+1]);
//            :=   +   -   *   /   <   <=   >   >=   <>
=   ;   (   )   #
            if(temp.equals(":=") || temp.equals("<=") ||
temp.equals(">=") ||  temp.equals("<>")){
                record = record + " " + temp + " ";
                i = i+2;
            } else if (!is_Digit(textlist[i])
&& !is_Letter(textlist[i])) {
                record = record + " " + textlist[i] + " ";
                i++;
            }else {
                record = record + textlist[i];
                i++;
            }
            index = i;
        }
        record = record + textlist[index];
        String pattern = "\\s+";
```

```
        Pattern p = Pattern.compile(pattern);
        Matcher m = p.matcher(record);
        record = m.replaceAll(" ");
        // 去除 多个空格
        return record;
    }
//   遍历，处理元素前后空格 删除空格元素
    private List filter(String resource){
        List<String> wordlist = new ArrayList<>();
        String[] temp =resource.trim().split(" ");
        for (int i = 0; i < temp.length; i++ ){
           temp[i] = temp[i].trim();
           if(temp[i] != null && temp[i] != ""){
               wordlist.add(temp[i]);
           }
        }
        return wordlist;
    }

//    扫描列表
    private void scanner(List<String> wordlist){
        System.out.println("类别\t 值\t 种别码");
```

```
        for (String element:wordlist) {
            if(is_Keyword(element)>=0){
                System.out.println("关键字
"+'\t'+element+'\t'+is_Keyword(element));
            }
            else if(is_op_deli(element)>=0){
                System.out.println("运算符&界符
"+'\t'+element+'\t'+is_op_deli(element));
            }
            else if(is_Digit(element)){
                System.out.println("整型常数
"+'\t'+element+'\t'+"20");
            }
            else {
                identifierList.add(element);
            }
        }
        System.out.println("-----------标识
符:10------------");
        for (String element:identifierList) {
            System.out.println(element);
        }
```

```
    }


    public static void main(String[] args) {
        Analyzer myanalyzer = new Analyzer();
        FileReader.readFile("code.text");
        List<String> display =
myanalyzer.filter(myanalyzer.regular(FileReader.outputT
ext()));
        System.out.println(display);
        myanalyzer.scanner(display);
    }
}
```

实验结果：

```
D:\jdk-11\jdk-11\bin\java.exe "-javaagent:D:
[int, main, (, ), begin, int, value, =, 10, 
类别 值   种别码
关键字   int 7
关键字   main   8
运算符&界符   (   26
运算符&界符   )   27
关键字   begin   1
关键字   int 7
运算符&界符   =   42
整型常数 10  20
运算符&界符   ;   34
关键字   int 7
运算符&界符   =   42
运算符&界符   ;   34
运算符&界符   :=  18
运算符&界符   ;   34
关键字   return  9
整型常数 0   20
运算符&界符   ;   34
关键字   end 6
-----------标识符:10------------
value
a
value
a
a

Process finished with exit code 0
```

图 2 实验结果图

实验总结

通过此次实验，我理解了词法分析在编译程序中的作用，加深对有穷自动机模型的理解，掌握了词法分析程序的实现方法和技术。最后，并成功通过 java 实现了对一小段代码文本进行词法分析。