编号：

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 实习 | 一 | 二 | 三 | 四 | 五 | 六 | 七 | 八 | 九 | 十 | 总评 | 教师签名 |
| 成绩 |  |  |  |  |  |  |  |  |  |  |  |  |

武汉大学计算机学院

**《编译原理》课程**

词法分析

实习报告

编 号：

实习题目： 小语言的词法分析程序

专业（班）：

学生学号：

学生姓名：

任课教师：

２０19 年 4 月 26 日

**目 录**

[1 语言形式化描述 3](#_Toc7269457)

[2 单词编码表 3](#_Toc7269458)

[3 状态转换图 5](#_Toc7269459)

[4 词法分析算法 8](#_Toc7269460)

[5 测试计划 14](#_Toc7269461)

# 1 语言形式化描述

编译程序使得高级语言源程序所描述的功能得以在计算机上实现。编译程序的设计这就是高级语言的实现者，源程序的编写者就是高级语言的使用者，他们必须遵循同样的准则——高级语言程序的构成规则，才能使写出的源程序能够被成功地翻译，文法描述的就是高级语言程序的构成规则。

人们把用一组数学符号和规则来描述语言的方式称为形式描述，把所用的数学符号和规则称为形式语言。语言的成分有字母表、文法和语义。字母表是符号的有穷集，符号构成了语言的句子；文法是结构规则的有穷集；语义是操作规则的有穷集。

# 2 单词编码表

关键字的编码表如表2.1所示。

表**2.1** 关键字编码表

|  |  |  |
| --- | --- | --- |
| CLASS | NAME | VALUE |
| 1 | int | $1 |
| 1 | void | $2 |
| 1 | main | $3 |
| 1 | if | $4 |
| 1 | else | $5 |
| 1 | while | $6 |
| 1 | for | $7 |
| 1 | bool | $8 |
| 1 | true | $9 |
| 1 | false | $10 |
| 1 | return | $11 |

运算符的编码表如表2.2所示。

表**2.2** 运算符编码表

|  |  |  |
| --- | --- | --- |
| CLASS | NAME | VALUE |
| 2 | + | $1 |
| 2 | - | $2 |
| 2 | \* | $3 |
| 2 | / | $4 |
| 2 | % | $5 |
| 2 | ! | $6 |
| 2 | = | $7 |
| 2 | < | $8 |
| 2 | > | $9 |
| 2 | & | $10 |
| 2 | | | $11 |
| 2 | += | $1$7 |
| 2 | -= | $2$7 |
| 2 | \*= | $3$7 |
| 2 | /= | $4$7 |
| 2 | %= | $5$7 |
| 2 | != | $6$7 |
| 2 | == | $7$7 |
| 2 | <= | $8$7 |
| 2 | >= | $9$7 |
| 2 | && | $10$10 |
| 2 | || | $11$11 |
| 2 | ++ | $1$1 |
| 2 | -- | $2$2 |

界限符的编码表如表2.3所示。

表**2.3** 界限符编码表

|  |  |  |
| --- | --- | --- |
| CLASS | NAME | VALUE |
| 3 | , | $1 |
| 3 | ; | $2 |
| 3 | ( | $3 |
| 3 | ) | $4 |
| 3 | { | $5 |
| 3 | } | $6 |

标识符、整数及出错情况的编码表如表2.4所示。

表**2.3** 标识符、整数编码表

|  |  |  |
| --- | --- | --- |
| CLASS | NAME | VALUE |
| 4 | 标识符 | 字符串值 |
| 5 | 整数 | 数值 |
| 6 | 出错 | 出错字符串值 |

# 3 状态转换图

整数的识别如图3.1所示。

数字

数字

非数字

空白

\*

**图3.1 整数的识别**

字母的识别如图3.2所示。

\*

非字母、数字或下划线

字母或下划线

空白

**图3.2 标志符和关键字的识别**

运算符的识别如图3.3所示。

非＝

空白

\*

非＝

+

＝

\*

-

\*

＝

\*

非＝

\*

\*

\*

＝

非＝

/

\*

＝

\*

\*

非＝

=

=

\*

非=

<

\*

\*

=

=

\*

\*

非=

%

（接上）

\*

\*

=

非=

!

非=

>

=

\*

\*

\*

\*

&

&

非&

|

非|

|

\*

\*

**图3.3 运算符的识别**

界限符的识别如图3.4所示。

空白

;

\*

，

\*

(

\*

{

)

\*

\*

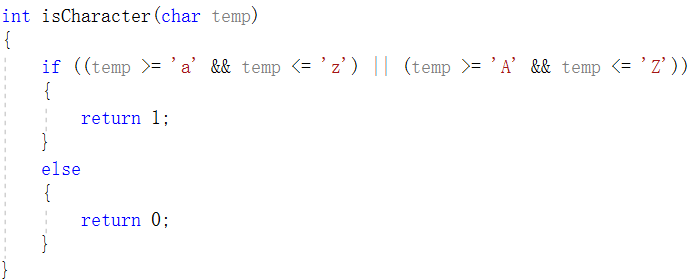
\*

}

**图3.4 界限符的识别**

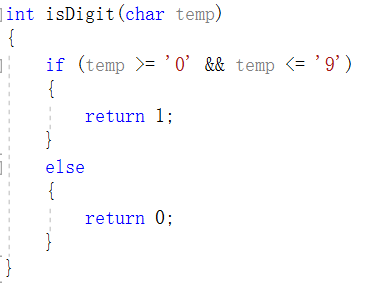
# 4 词法分析算法

判断是否为字母的算法如图4.1所示。



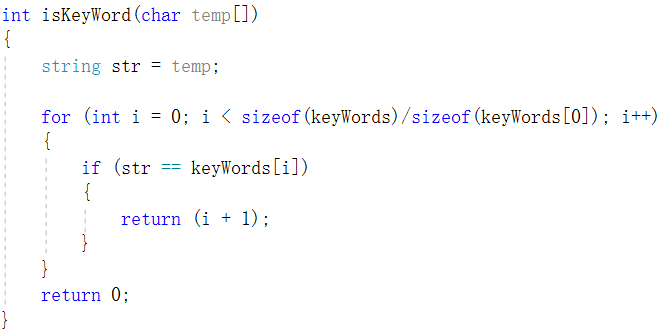
**图4.1 判断是否为字母**

判断是否为数字的算法如图4.2所示。



**图4.2 判断是否为数字**

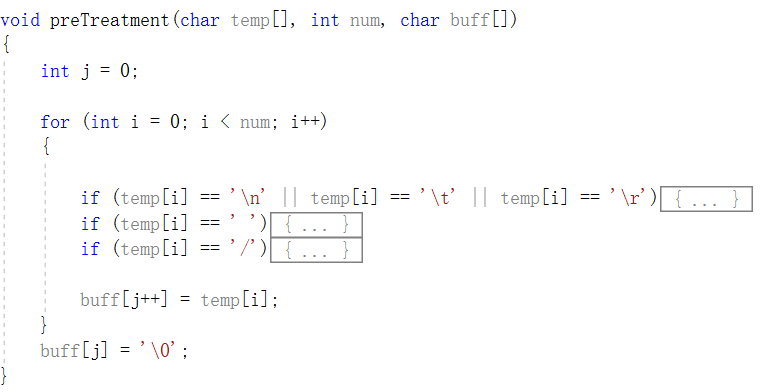
判断是否为关键字并返回关键字序列的算法如图4.3所示。



**图4.3 判断是否为关键字并返回其序列**

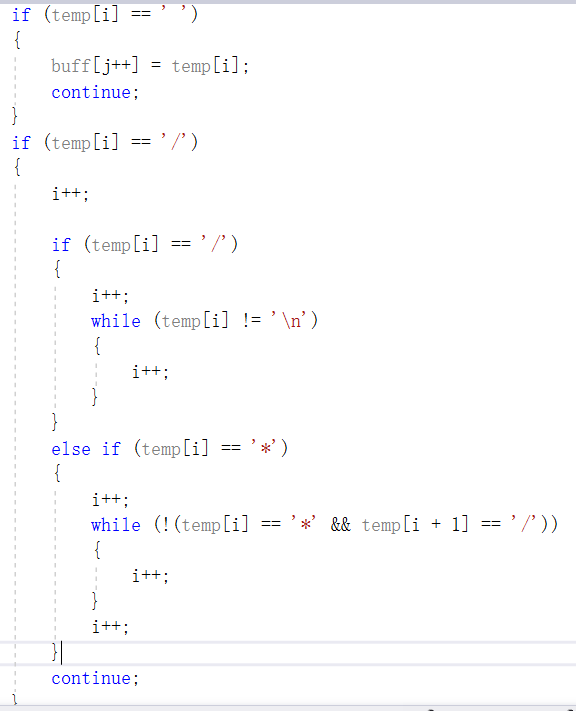
判断是否为界限符、运算符并返回其序列的算法与上述判断关键字的算法类似。在此不予赘述。

对于一个源文件，首先对其作预处理。去除其中的注释、换行符、回车符、多个空格等非必要信息，并将剩余必要信息其存入一个字符数组中。此过程框架如图4.4所示。其中三个参数分别代表未经处理的源文件字符数组、未经处理的源文件数组长度、预处理后形成的缓冲字符数组。



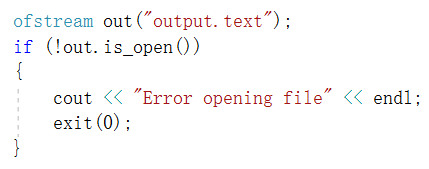
**图4.4 预处理代码框架**

由图4.4可知，首先去除其中的换行符、回车符、制表符，再保留一个空格，以便保留其中的必要空格。最后去除其中的注释信息。具体重要代码如图4.5所示。



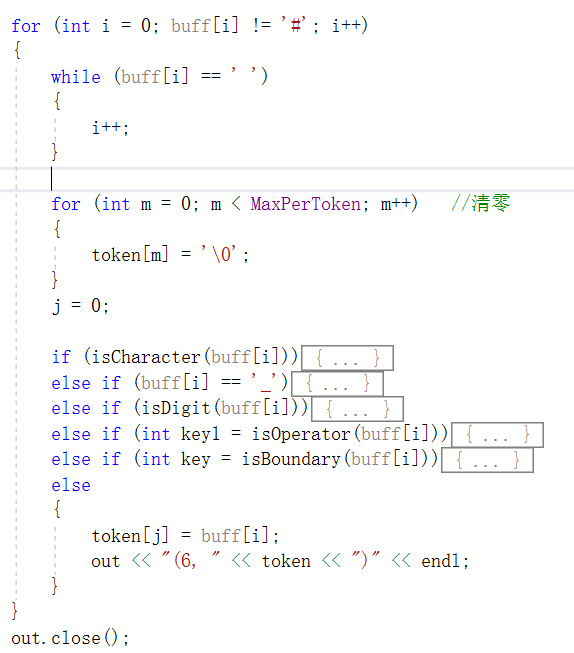
**图4.5 预处理保留必要空格、删除注释部分代码**

接着，对预处理后的字符数组进行词法分析。首先需要新建文件。此部分代码如图4.6所示。



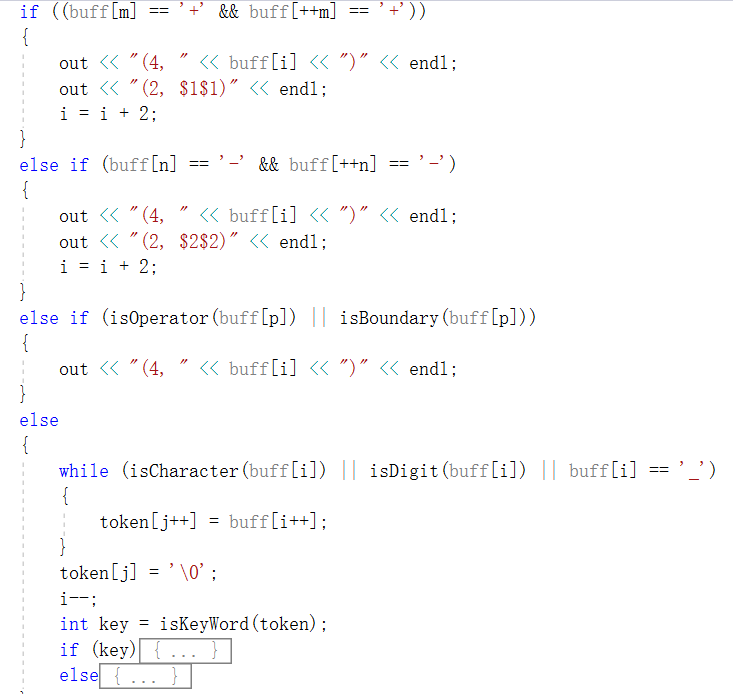
**图4.6 C++文件输入和输出**

字符处理过程如图4.7所示。将所得的词存入 token字符数组中，分析其类别并输出到文件中。需要注意的是，每分析完一个词之后需要对 token字符数组进行清除。



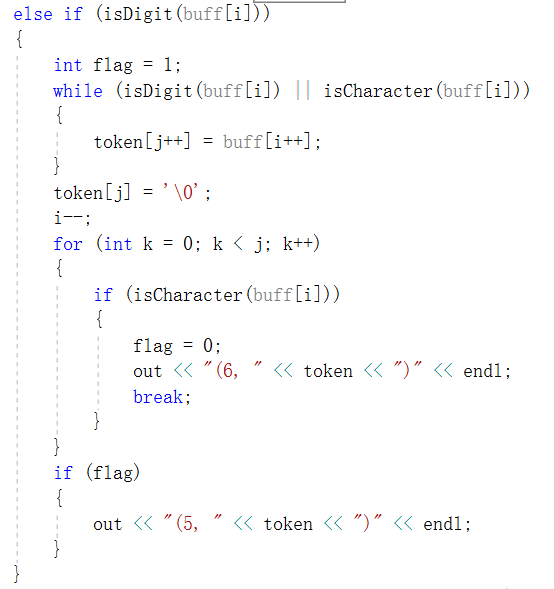
**图4.7 词法分析处理过程概要**

分析以字母开头的词的重要代码如图4.8所示。



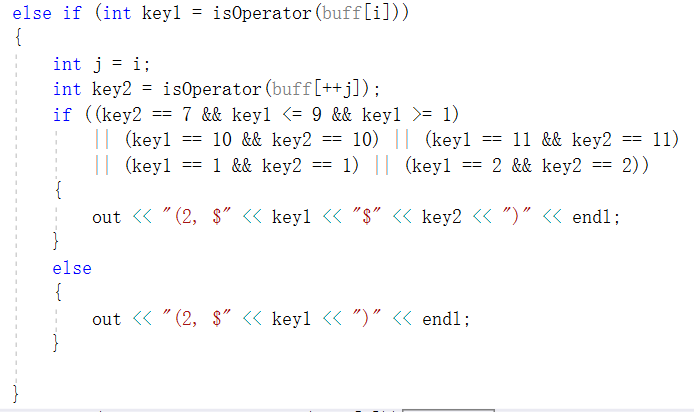
**图4.8 分析以字母开头的词的部分代码**

分析以数字开头的数字时，不得为标识符。只能为数字。此部分代码如图4.9所示。



**图4.9 分析以数字开头的词的代码**

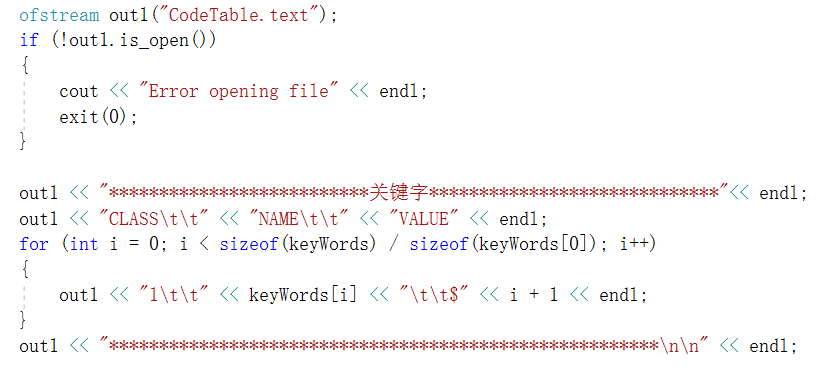
分析以运算符开头的词时，需注意两个单个运算符合并为一个新运算符的情况。此部分代码如图4.10所示。



**图4.10 分析以运算符开头的词的代码**

其他的分析过程类似。在此不予赘述。

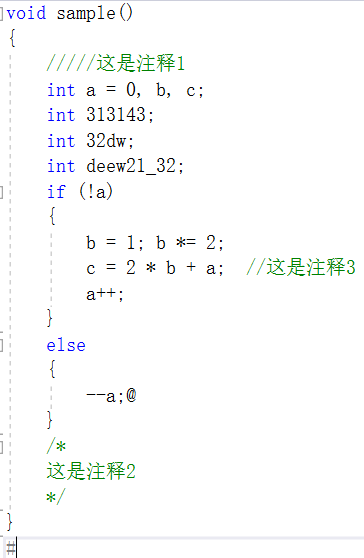
输出编码表的部分代码如图4.11所示。



**图4.11 输出编码表的部分代码**

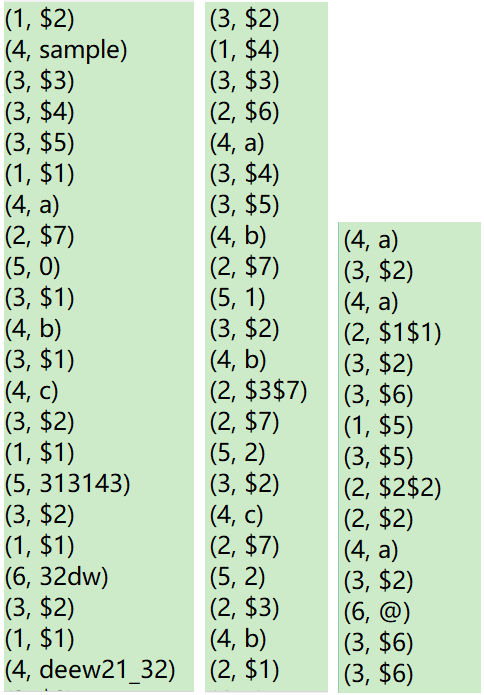
# 5 测试计划

测试代码如图5.1所示。其中涵盖了注释、空格等非必要信息，涵盖了正确关键字、数字、标识符等，也有错误标识符。

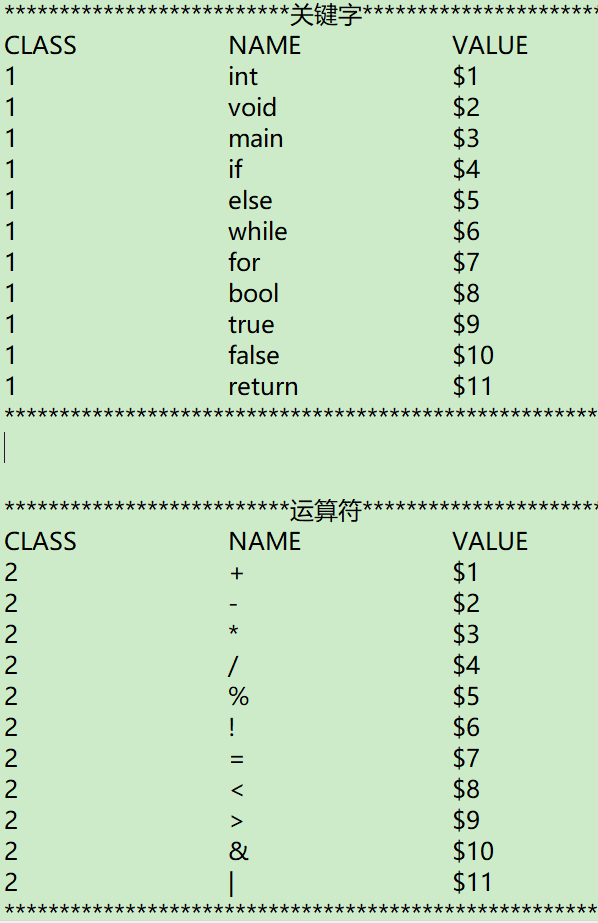
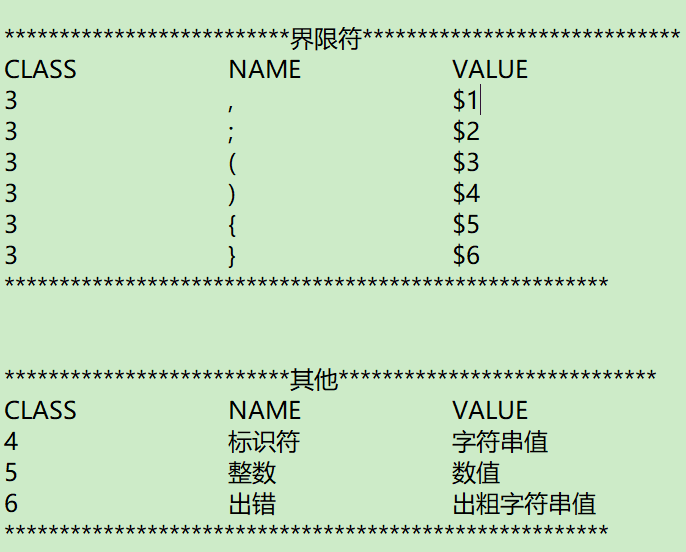


**图5.1 测试代码**

输出如图5.2所示，图片经过调整。输出顺序为从左到右，从上到下。字符编码表文件如图5.3所示。



**图5.2 输出单词符号文件**

**图5.3 输出字符编码表文件**

由图5.2可知，关键字、运算符、界限符、标识符、整数、出错类型分别用数字1、2、3、4、5、6表示。其中，错误单词已显示出来。