专题 1\_ 词法分析程序构造原理与实现

李若森 13281132 计科1301

# 程序功能描述

**[功能]：**

完成下述正则文法所描述的C语言子集单词符号的词法分析程序。

**[要求]：**

1. 给出各单词符号的类别编码。
2. 能发现输入串的错误。
3. 将分析所得二元序列输出到中间文件中。

**[文法]：**

<标识符>→c|c<余留标识符>

<余留标识符>→d|c

<实数>→\((+|-)<无符号数>|d<余留无符号数>|.<小数部分>|d\)

<无符号数>→d<余留无符号数>|.<小数部分>|d

<余留无符号数>→d<余留无符号数>|.<十进小数>|(E|e)<指数部分>|.|d

<十进小数>→(E|e)<指数部分>|d<十进小数>|d

<小数部分>→d<十进小数>|d

<指数部分>→d<余留指数>|(+|-)<整指数>|d

<整指数>→d<余留整指数>|d

<余留整指数>→d<余留整指数>|d

<算数运算符>→+|-|\*|/|++|--

<关系运算符>→>|<|==|>=|<=|!=

<逻辑运算符>→!|&&|\|\|

<位操作运算符>→>>|<<

<赋值运算符>→=|+=|-=|\*=|/=|%=

<特殊运算符>→,|\(|\)|{|}

<分隔符>→;

保留字: void int float double if else for do while

**[说明]：**

1. 该语言对大小写不敏感
2. c代表字母a-z&&A-Z，d代表数字0-9。
3. “/\*..\*/”以及“//”为程序注释部分。
4. 文法中‘\’为转义字符
5. 所有数字都需要用括号表示

# 主要数据结构描述

**pair<int, string>:**

用pair<int, string>来存储单个二元组。其中第一个元素为类型号，第二个为元素的值。当类型号小于40时代表程序分界符，第二个元素不存储有效信息，用“-”代替；类型号为40时是标识符，第二个元素存储标识符字符串；类型号为41时代表实数，第二个元素存储的是该实数的二进制值。

**vector<>:**

vector是C++中的动态数组，用来存储每一行的二元组。

# 程序结构描述

## 设计方法：

### 状态转换图：（DFA M）

|  |
| --- |
|  |

### 本实验将标识符分为以下几类

1. 算术运算符

+、-、\*、/、%、++、--

1. 关系运算符

>、<、==、>=、<=、!=

1. 逻辑运算符

!、&&、||

1. 位操作运算符

<<、>>

1. 赋值运算符

=、+=、-=、\*=、/=、%=

1. 特殊运算符

,、(、)、{、}

1. 分隔符

;、/\*、\*/、//

1. 实数
2. 标识符
3. 保留字

void、int、float、double、if、else、for、do、while

### 需要识别的关键字及识别码

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 关键字 | 识别码 | 关键字 | 识别码 | 关键字 | 识别码 |
| + | 1 | && | 15 | } | 29 |
| - | 2 | || | 16 | ; | 30 |
| \* | 3 | << | 17 | void | 31 |
| / | 4 | >> | 18 | int | 32 |
| % | 5 | = | 19 | float | 33 |
| ++ | 6 | += | 20 | double | 34 |
| -- | 7 | -= | 21 | if | 35 |
| > | 8 | \*= | 22 | else | 36 |
| < | 9 | /= | 23 | for | 37 |
| == | 10 | %= | 24 | do | 38 |
| >= | 11 | , | 25 | while | 39 |
| <= | 12 | ( | 26 | 标识符 | 40 |
| != | 13 | ) | 27 | 实数 | 41 |
| ! | 14 | { | 28 |  |  |

## 函数定义：

|  |
| --- |
| **getOutputName:** |
| void getOutputName( char \*inputName, char \*outputName ); |
| **功能：** |
| 得到输出文件名 |
| **传入参数：** |
| inputname:需处理的文件的文件名 |
| **传出参数：** |
| Outputname:输出文件的文件名 |
| **返回值：** |
| （无） |
|  |
| **errMsg:** |
| void errMsg( string filename, int rowNo, int colNo, string errmsg ); |
| **功能：** |
| 向屏幕输出错误信息 |
| **传入参数：** |
| filename:正在处理的文件的文件名称 |
| rowNo:出错行 |
| colNo:出错列 |
| errmsg:错误信息 |
| **传出参数：** |
| （无） |
| **返回值：** |
| （无） |
|  |
| **spaceProcess:** |
| void spaceProcess( char \*line, string &errmsg ); |
| **功能：** |
| 预处理，删除字符串内空格 |
| **传入参数：** |
| line:需处理的字符串 |
| **传出参数：** |
| line:处理过的字符串  errmsg:错误信息 |
| **返回值：** |
| （无） |
|  |
| **noteProcess:** |
| bool noteProcess( string &str, bool &noteflag ); |
| **功能：** |
| 预处理，删除字符串内注释 |
| **传入参数：** |
| str:需处理的字符串 |
| Noteflag:该字符串是否处于多行注释中 |
| **传出参数：** |
| str:处理过的字符串 |
| noteflag:当前行是否处于多行注释中 |
| **返回值：** |
| 是否成功处理。如果注释匹配则返回true，否则返回false。 |
|  |
| **print:** |
| void print( FILE \*fp ); |
| **功能：** |
| 将结果输出到文件中 |
| **传入参数：** |
| str:需处理的字符串 |
| Noteflag:该字符串是否处于多行注释中 |
| **传出参数：** |
| str:处理过的字符串 |
| **返回值：** |
| 是否成功处理。如果注释匹配则返回true，否则返回false。 |
|  |
| **lindAnalyse:** |
| bool lineAnalyse( string &line, int &colNo ); |
| **功能：** |
| 行词法分析 |
| **传入参数：** |
| line:包含该行的字符串 |
| colNo:起始处理位置 |
| **传出参数：** |
| colNo:处理完成位置 |
| **返回值：** |
| 该行是否成功解析。成功返回true，否则返回false。 |
|  |
| **isTerminalStage:** |
| bool isTerminalStage( int stage ); |
| **功能：** |
| 判断当前阶段是否为终结阶段 |
| **传入参数：** |
| line:包含该行的字符串 |
| **传出参数：** |
| （无） |
| **返回值：** |
| 该阶段是否成功解析。是则返回true，否则返回false。 |
|  |
| **characterType:** |
| int characterType( char ch ); |
| **功能：** |
| 判断当前字符的类型 |
| **传入参数：** |
| ch:待判断字符 |
| **传出参数：** |
| （无） |
| **返回值：** |
| 该字符的类型对应的值。 |
|  |
| **dtob:** |
| void dtob( double d, string &str ); |
| **功能：** |
| 将double类型值转换为二进制表示。 |
| **传入参数：** |
| d:待转换的double值 |
| **传出参数：** |
| str:该double类型值得二进制表示字符串 |
| **返回值：** |
| （无） |
|  |
| **addPair:** |
| void addPair( string str, vector<PIS> &vTable ); |
| **功能：** |
| 将得到的合法字符串加入到二元式序列中 |
| **传入参数：** |
| str:待加入二元式的合法字符串 |
| **传出参数：** |
| vTable:更新后的二元式序列 |
| **返回值：** |
| （无） |
|  |
| **stage\_\*:** |
| bool stage\_\*( int &stage, int chtype );//(\*从0-17共18个状态) |
| **功能：** |
| 阶段转换 |
| **传入参数：** |
| stage:当前阶段 |
| chtype:下一个字符的类别 |
| **传出参数：** |
| stage:下一个阶段 |
| **返回值：** |
| 是否进入下一个合法的阶段。进入返回true，否则返回false。 |
|  |
| **excute:** |
| bool excute( int &stage, char ch ); |
| **功能：** |
| 将得到的合法字符串加入到二元式序列中 |
| **传入参数：** |
| Stage:当前阶段 |
| ch:下一个字符 |
| **传出参数：** |
| stage:下一个阶段 |
| **返回值：** |
| 是否进入下一个合法的阶段。进入返回true，否则返回false。 |
|  |
| **lineAnalyse:** |
| bool lineAnalyse( string &line, int &colNo,  string &errmsg, vector<PIS> &vTable ); |
| **功能：** |
| 行词法分析 |
| **传入参数：** |
| line:包含该行的字符串 |
| colNo:起始处理位置 |
| **传出参数：** |
| colNo:处理完成位置 |
| errmsg:错误信息 |
| vTable:该行的二元式序列 |
| **返回值：** |
| 该行是否成功解析。成功返回true，否则返回false。 |

# 程序测试