**北京邮电大学《编译原理与技术》课程实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **实验**  **名称** | C语言词法分析程序的设计与实现 | | **学 院** | 计算机学院 | **指导教师** |  |
| **班 级** | **班内序号** | **学 号** | | **学生姓名** | **成绩** | |
| 2018211305 | 27 | 2018211279 | | 索政铎 |  | |
|  |  |  | |  |  | |
|  |  |  | |  |  | |
| **实**  **验**  **内**  **容** | 利用所学词法分析原理，自行设计实现一个简易的词法分析程序，用于C语言的词法分析。 | | | | | |
| **学生实验报告** | （详见“实验报告和源程序”册） | | | | | |
| **课**  **程**  **设**  **计**  **成**  **绩**  **评**  **定** | **评语**:  **成绩**:  指导教师签名：  年 月 日 | | | | | |

注：评语要体现每个学生的工作情况，可以加页。

1. **实验内容与实验环境描述**

**实验内容与要求：**

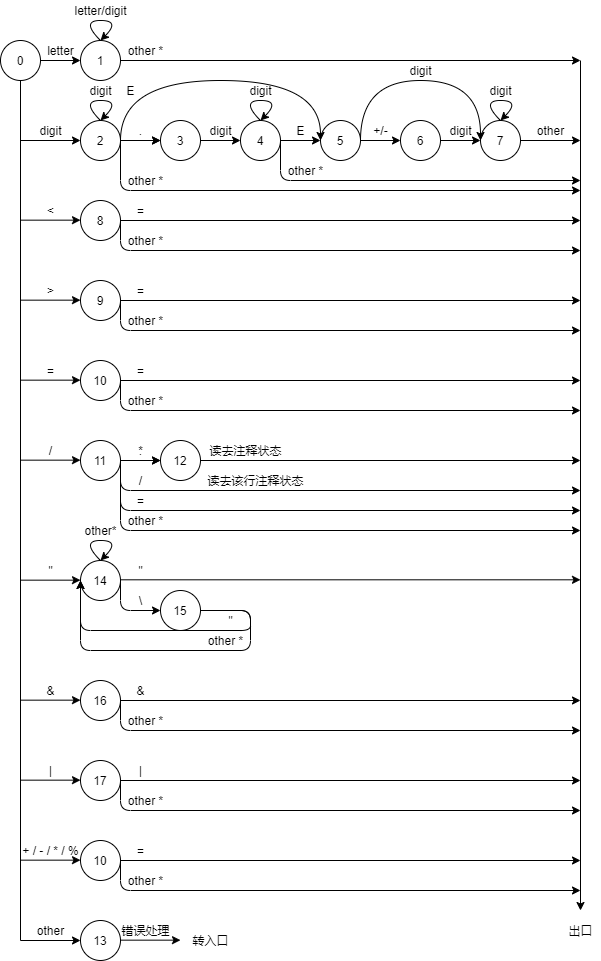
1. 可以识别出用C语言编写的源程序中的每个单词符号，并以记号的形式输出每个单词符号。
2. 可以识别并跳过源程序中的注释。
3. 可以统计源程序中的语句行数、各类单词的个数、以及字符总数，并输出统计结果。
4. 检查源程序中存在的词法错误，并报告错误所在的位置。
5. 对源程序中出现的错误进行适当的恢复，使词法分析可以继续进行，对源程序进行一次扫描，即可检查并报告源程序中存在的所有词法错误。

**实验环境：**

Microsoft Visual Studio Community 2019。

1. **程序设计说明**
2. **程序流程**

针对C语言构造的状态转换图如下



1. **使用方法**

在与LexicalAnalysor.exe放置待分析程序(program.c)与保留字信息(key.txt)后执行即可。

1. **符号说明**

|  |  |
| --- | --- |
| 名称 | 含义 |
| analyse() | 词法分析函数 |
| state | 当前状态指示 |
| C | 存放当前读入的字符 |
| token | 存放当前正在识别的字符串 |
| buffer | 输入程序缓冲区 |
| forward | 指向后一个待读入字符的指针 |
| get\_char | 函数，根据forward指示从buffer中读入一个字符并存入C中，然后移动forward指向下一个字符 |
| get\_nbc | 函数，检查C中的字符是否为空格、换行符、制表符、空字符等，若是则调用get\_char |
| cat | 函数，将C中的字符连接在token字符串后 |
| iskey | 表示识别出的标识符记号 |
| letter | 函数，判断字符C是否为字母 |
| digit | 函数，判断字符C是否为数字 |
| retract | 函数，将forward后退一个字符 |
| reserve | 函数，根据token中的字符串查找符号表，并返回标识符记号 |
| SToI | 将token中字符串转换为整数 |
| SToF | 将token中字符串转换为浮点数 |
| table\_insert | 函数，将识别出的标识符插入符号表，犯规标识符记号 |
| error | 函数，错误处理 |
| ret | 函数，将识别出的记号按格式返回 |
| buffer\_size | 表示输入程序缓冲区的大小 |
| table\_keyword | 记录保留字及识别出的标识符 |
| line | 当前正在分析的行号 |
| col | 当前正在分析的列号 |
| statistic | 函数，输出统计信息 |
| cnttoken | 记录识别出的符号数 |

1. **测试结果与分析**

**输入样例程序：**

void func()

{

return -1 - 180.3e-2 \* 1e-1 / 2.0E-2 + 200%2;

}

int main()

{

int a = 4.5, b = 4.4.6; /\*error\*/

if(a==b && 2|3 || 1&2) a = a+b; //test

else a = a > b + a < b;

a += b;

b /= a;

if(a>=2 && a<=3 || b>=3 && b<=4) return sizeof(double);

printf("%d\"123\3%d", a.b);

return 0;

}

**样例分析输出：**

<ID=28, 1, void>

<ID=31(new), 1, func>

<symbol, 1, (>

<symbol, 1, )>

<symbol, 2, {>

<ID=19, 3, return>

<symbol, 3, ->

<num, 3, 1>

<symbol, 3, ->

<num, 3, 1.803000>

<symbol, 3, \*>

<num, 3, 0.100000>

<symbol, 3, />

<num, 3, 0.020000>

<symbol, 3, +>

<num, 3, 200>

<symbol, 3, %>

<num, 3, 2>

<symbol, 3, ;>

<symbol, 4, }>

<ID=16, 6, int>

<ID=32(new), 6, main>

<symbol, 6, (>

<symbol, 6, )>

<symbol, 7, {>

<ID=16, 9, int>

<ID=33(new), 9, a>

<symbol, 9, ASSIGN>

<num, 9, 4.500000>

<symbol, 9, ,>

<ID=34(new), 9, b>

<symbol, 9, ASSIGN>

<num, 9, 4.400000>

error<line:9, column:23>

<num, 9, 6>

<symbol, 9, ;>

<ID=15, 10, if>

<symbol, 10, (>

<ID=33, 10, a>

<symbol, 10, EQ>

<ID=34, 10, b>

<symbol, 10, &&>

<num, 10, 2>

<symbol, 10, |>

<num, 10, 3>

<symbol, 10, ||>

<num, 10, 1>

<symbol, 10, &>

<num, 10, 2>

<symbol, 10, )>

<ID=33, 10, a>

<symbol, 10, ASSIGN>

<ID=33, 10, a>

<symbol, 10, +>

<ID=34, 10, b>

<symbol, 10, ;>

<ID=9, 11, else>

<ID=33, 11, a>

<symbol, 11, ASSIGN>

<ID=33, 11, a>

<symbol, 11, GT>

<ID=34, 11, b>

<symbol, 11, +>

<ID=33, 11, a>

<symbol, 11, LS>

<ID=34, 11, b>

<symbol, 11, ;>

<ID=33, 12, a>

<symbol, 12, +=>

<ID=34, 12, b>

<symbol, 12, ;>

<ID=34, 13, b>

<symbol, 13, /=>

<ID=33, 13, a>

<symbol, 13, ;>

<ID=15, 14, if>

<symbol, 14, (>

<ID=33, 14, a>

<symbol, 14, GE>

<num, 14, 2>

<symbol, 14, &&>

<ID=33, 14, a>

<symbol, 14, LE>

<num, 14, 3>

<symbol, 14, ||>

<ID=34, 14, b>

<symbol, 14, GE>

<num, 14, 3>

<symbol, 14, &&>

<ID=34, 14, b>

<symbol, 14, LE>

<num, 14, 4>

<symbol, 14, )>

<ID=19, 14, return>

<ID=22, 14, sizeof>

<symbol, 14, (>

<ID=8, 14, double>

<symbol, 14, )>

<symbol, 14, ;>

<ID=35(new), 15, printf>

<symbol, 15, (>

<array, 15, "%d"123\3%d">

<symbol, 15, ,>

<ID=33, 15, a>

<symbol, 15, )>

<symbol, 15, ;>

<ID=19, 16, return>

<num, 16, 0>

<symbol, 16, ;>

<symbol, 17, }>

**结果分析：**

样例输入基本完全包含所有需要分析的情形、涵盖了程序基本所有分支，且输出结果均符合预期。

1. **总结与心得体会**

通过词法分析程序的实现实验，我经历了了从最初对词法分析程序感到复杂且难以设计的状态，经过老师的教导与学习，借助上学期形式语言与自动机中关于状态转换图的设计，让我对词法分析程序的实现感到豁然开朗，并且还在原本的设计基础上添加了C语言中单行注释、二目运算符、常量字符串等等的处理。这让我深深地感觉到了编译原理的奥妙与设计的精巧，让我对程序编译有了更为深入的认识和理解。

非常感谢老师的实验设计！

1. **源程序**

另附