词法分析程序设计

班级: 2013211308

姓名: 陈鹏辉

学号: 2013211391

**一．问题描述**

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.可以并识别读取源程序中的注释。

3.可以统计源程序中的语句行数、单词个数和字符数，其中标点和空格不计为单词，并输出统计结果。

4.检察源程序中存在的错误，并可以报告错误所在行列的位置。

5.发现原程序中存在的错误，进行适当修复，使词法分析可以继续进行，通过一次词法分析处理，可以检查并报告源程序中存在的所有错误。

**二、实验程序设计说明**

**1.数据结构和全局变量**

//关键字

string keywords[32]={ "auto","break","case","char","const","continue",

"default","do","double","else","enum","extern",float","for","goto","if","int","long","register","return","short","signed","sizeof","static","struct","switch","typedef","union","unsigned","void","volatile","while" };

//运算符

string operators[40] = { "+","-","\*","/","%","++","--",//算术运算符(0-6)

">","<","==",">=","<=","!=",//关系运算符(7-12)

"!","&&","||",//逻辑运算符(13-15)

"<<",">>","~","|","^","&",//位运算符(16-21)

"=","+=","-=","\*=","/=","%=",">>=","<<=","&=","^=","|=",//赋值运算符(22-32)

"?",":",//条件运算符(33-34)

".","->",//分量运算符(35-36)

"[","]",//下标运算符(37-38)

"\\" };//转义运算符(39)

//分隔符

string boundary[9] = { ",",";","'","\"","(",")","{","}","#" };

const int keywordNum = 32;//关键字个数

const int boundaryNum = 9;//分隔符个数

const int operatorNum = 40;//操作符个数

ifstream infile;//源程序文件指针

int lines = 0;//行数记录

int errors = 0;//错误记录

int tokenNum = 0;//单词数记录

int chNum = 0;//字符数记录

**2.主要函数说明**

void initTab();//初始化各表

void alpha(char ch);//字符，下划线开头的字符串处理（标识符或关键字）

void digit(char ch);//数字串处理

void strCon(char ch);//字符串字面量处理

void otherChar(char ch);//注释，运算符等处理

void output(string lexeme, string token, int location);//数据结果表

int findID(string id);//遍历标识符表返回标识符所在位置，若没有在表尾插入

int findNum(string digit);//遍历数字表返回数字常量所在位置，若没有在表尾插入

int findStrCon(string str);//遍历字符串常量表返回字符串所在位置，若没有在表尾插入

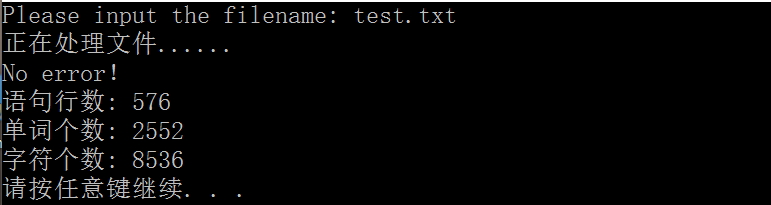
int findOperator(string s);//遍历运算符表返回运算符在表中位置，若不在返回-1

int findBoundary(string s);//遍历分隔符表返回分隔符在表中位置，若不在返回-1

void error(string tip, int line); //错误处理输出

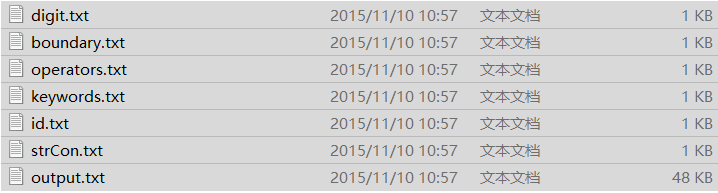
**3.输入示例**

输入文件test.txt为本次实验的词法分析器源码

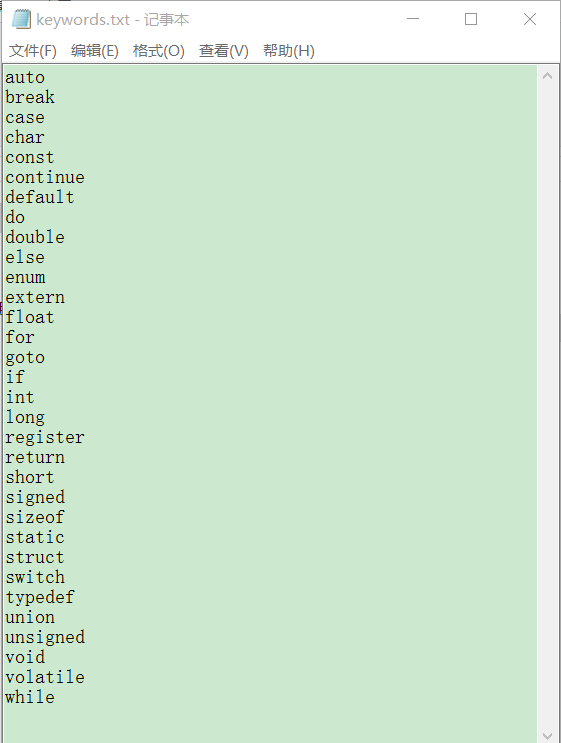


相关的统计结果和错误信息输出到屏幕

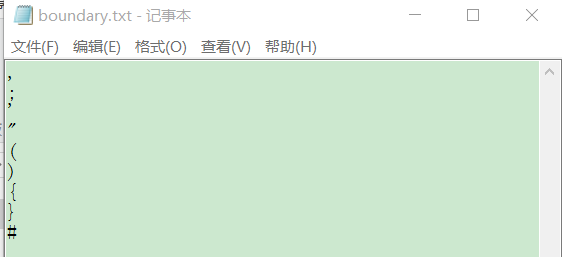
另得到7个输出结果文件



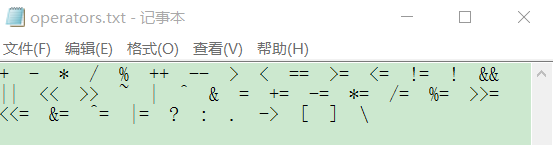
**1.keywords.txt** 记录C语言所有的关键字



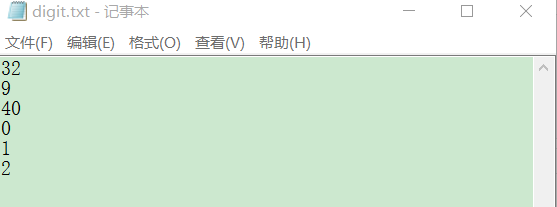
**2.boundary.txt** 记录C语言所有的分隔符



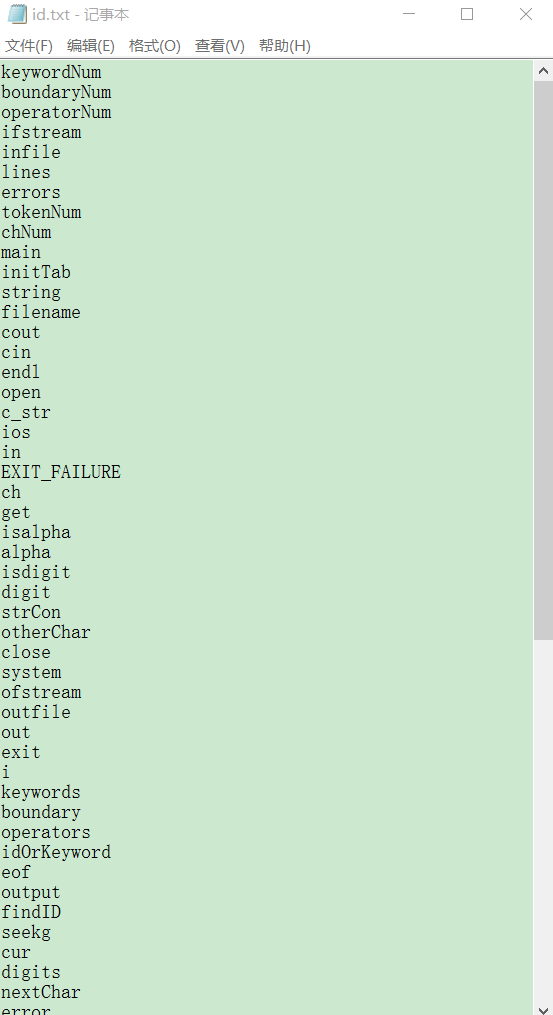
**3.operators.txt**  记录C语言所有的运算符



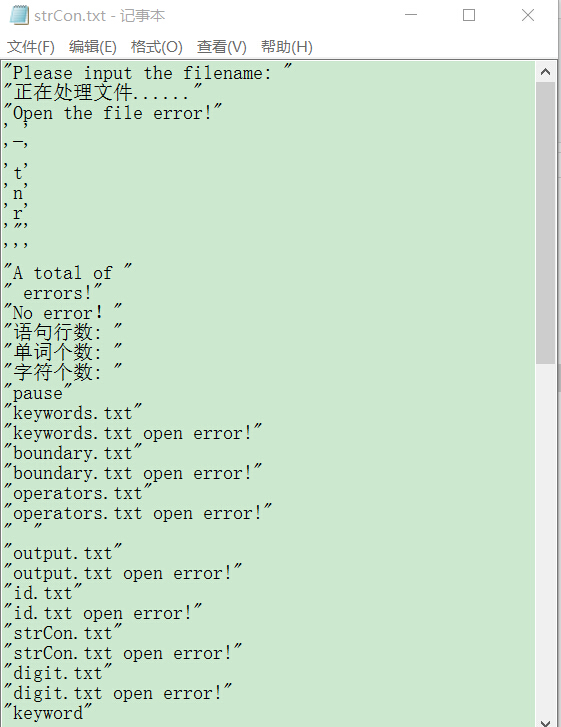
**4.digit.txt**  记录输入文件中所有的数字常量



**5.id.txt**  记录输入文件中所有的标识符



**6.strCon.txt** 记录输入文件中所有的字符串常量

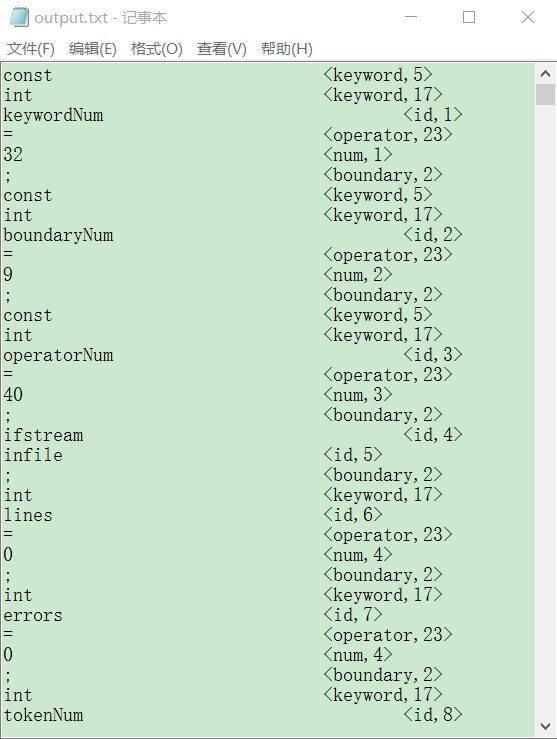


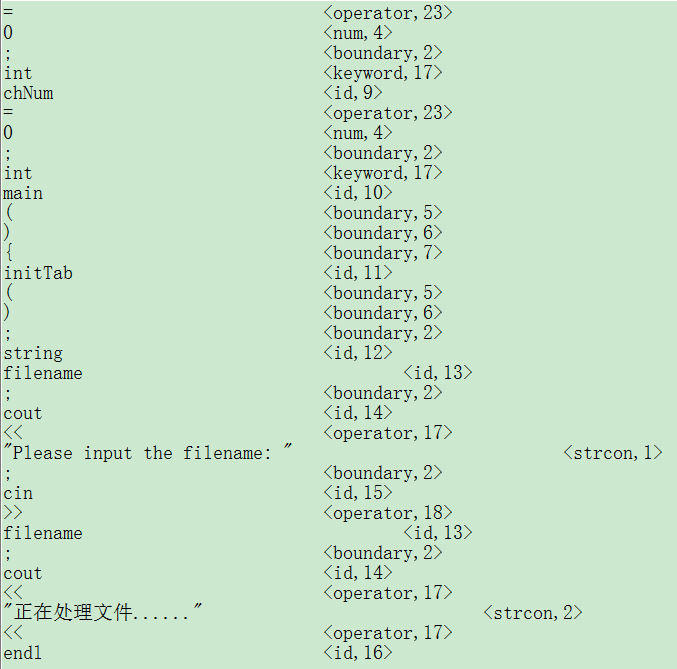
**7.output.txt**  记录词法分析的输出结果

output.txt 格式如下:

单词 <单词属性，单词在对应表中的第几个位置>

例如:





**PS:限于篇幅有限，只截取部分输出结果**

**三、程序源代码**

C:\Users\CPH\AppData\Roaming\Tencent\Users\867118422\QQ\WinTemp\RichOle\R0BWZ]]%IEQG6EDSU}]3M)2.png

1. lexAnalysis.h

#ifndef LEXANALYSIS\_H

#define LEXANALYSIS\_H

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

//关键字

string keywords[32] = { "auto","break","case","char","const","continue","default","do","double","else","enum","extern",

"float","for","goto","if","int","long","register","return","short","signed","sizeof","static",

"struct","switch","typedef","union","unsigned","void","volatile","while" };

//运算符

string operators[40] = { "+","-","\*","/","%","++","--",//算术运算符(0-6)

">","<","==",">=","<=","!=",//关系运算符(7-12)

"!","&&","||",//逻辑运算符(13-15)

"<<",">>","~","|","^","&",//位运算符(16-21)

"=","+=","-=","\*=","/=","%=",">>=","<<=","&=","^=","|=",//赋值运算符(22-32)

"?",":",//条件运算符(33-34)

".","->",//分量运算符(35-36)

"[","]",//下标运算符(37-38)

"\\" };//转义运算符(39)

//分隔符

string boundary[9] = { ",",";","'","\"","(",")","{","}","#" };

void initTab();//初始化各表

void alpha(char ch);//字符，下划线开头的字符串处理（标识符或关键字）

void digit(char ch);//数字串处理

void strCon(char ch);//字符串字面量处理

void otherChar(char ch);//注释，运算符等处理

void output(string lexeme, string token, int location);//数据结果表

int findID(string id);//遍历标识符表返回标识符所在位置，若没有在表尾插入

int findNum(string digit);//遍历数字表返回数字常量所在位置，若没有在表尾插入

int findStrCon(string str);//遍历字符串常量表返回字符串所在位置，若没有在表尾插入

int findOperator(string s);//遍历运算符表返回运算符在表中位置，若不在返回-1

int findBoundary(string s);//遍历分隔符表返回分隔符在表中位置，若不在返回-1

void error(string tip, int line); //错误处理输出

#endif

2.lexAnalysis.cpp

#include "lexAnalysis.h"

const int keywordNum = 32;

const int boundaryNum = 9;

const int operatorNum = 40;

ifstream infile;//源程序文件指针

int lines = 0;//行数记录

int errors = 0;//错误记录

int tokenNum = 0;//单词数记录

int chNum = 0;//字符数记录

int main()

{

initTab();//初始化各输出表

string filename; //打开输入文件

cout << "Please input the filename: ";

cin >> filename;

infile.open(filename.c\_str(), ios::in);

if (!infile)

{

cout << "Open the file error!" << endl;

system("pause");

return EXIT\_FAILURE;

}

cout << "正在处理文件......" << endl;

char ch;

while (infile.get(ch)) //词法分析主体

{

++chNum;

if (isalpha(ch) || ch == '\_') //字符，下划线开头的字符串处理（标识符或关键字）

alpha(ch);

else if (isdigit(ch)) //数字串处理

digit(ch);

else if (ch == ' ' || ch == '\t') //空白符，制表符处理，直接跳过

--chNum;

else if (ch == '\n' || ch == '\r') //换行符处理，记录行数

{

--chNum;

++lines;

}

else if (ch == '"' || ch == '\'') //字符串字面量

strCon(ch);

else //需要超前搜索才能识别的内容

otherChar(ch);

}

if (errors) //错误统计

cout << "A total of " << errors << " errors!" << endl;

else

cout << "No error！" << endl;

cout << "语句行数: " << lines << endl; //结果统计

cout << "单词个数: " << tokenNum << endl;

cout << "字符个数: " << chNum << endl;

infile.close();

system("pause");

return 0;

}

void initTab()//初始化各表

{

ofstream outfile;

//初始化关键字表

outfile.open("keywords.txt", ios::out);

if (!outfile)

{

cout << "keywords.txt open error!" << endl;

system("pause");

exit(-1);

}

for (int i = 0; i < keywordNum; ++i)

outfile << keywords[i] << endl;

outfile.close();

//初始化分隔符表

outfile.open("boundary.txt", ios::out);

if (!outfile)

{

cout << "boundary.txt open error!" << endl;

system("pause");

exit(-1);

}

for (int i = 0; i < boundaryNum; ++i)

outfile << boundary[i] << endl;

outfile.close();

//初始化运算符表

outfile.open("operators.txt", ios::out);

if (!outfile)

{

cout << "operators.txt open error!" << endl;

system("pause");

exit(-1);

}

for (int i = 0; i < operatorNum; ++i)

outfile << operators[i] << " ";

outfile << endl;

outfile.close();

//创建并初始化输出文件

outfile.open("output.txt", ios::out);

if (!outfile)

{

cout << "output.txt open error!" << endl;

system("pause");

exit(-1);

}

outfile.close();

//创建并初始化标识符输出文件

outfile.open("id.txt", ios::out);

if (!outfile)

{

cout << "id.txt open error!" << endl;

system("pause");

exit(-1);

}

outfile.close();

//创建并初始化字符串常量输出文件

outfile.open("strCon.txt", ios::out);

if (!outfile)

{

cout << "strCon.txt open error!" << endl;

system("pause");

exit(-1);

}

outfile.close();

//创建并初始化数字常量输出文件

outfile.open("digit.txt", ios::out);

if (!outfile)

{

cout << "digit.txt open error!" << endl;

system("pause");

exit(-1);

}

outfile.close();

}

void alpha(char ch)//字符，下划线开头的字符串处理（标识符或关键字）

{

string idOrKeyword;

while (isalpha(ch) || isdigit(ch) || ch == '\_')

{

idOrKeyword += ch;

infile.get(ch);

++chNum;

if (infile.eof()) break;

}

int i;

for (i = 0; i < keywordNum; ++i)

if (idOrKeyword == keywords[i])//是关键字

{

output(keywords[i], "keyword", i + 1);

break;

}

if (i == keywordNum)//不是关键字

output(idOrKeyword, "id", findID(idOrKeyword));

infile.seekg(-1, ios::cur);

--chNum;

}

void digit(char ch)//数字串处理

{

string digits;

char nextChar = infile.get();

infile.seekg(-1, ios::cur);

if (ch == '0' && nextChar != '.'

&& nextChar != 'e' && nextChar != 'E'

&& nextChar != 'x' && !isdigit(nextChar)) //整形0

digits += ch;

else if (ch == '0') //0开头的8进制或16进制数

{

digits += ch;

infile.get(ch);

++chNum;

if (ch == 'x' || ch == 'X') //16进制处理

{

digits += ch;

infile.get(ch);

++chNum;

if (!isdigit(ch) && !(ch >= 'a' && ch <= 'f') && !(ch >= 'A' && ch <= 'F')) //非法0x 或 0X 字符串

{

infile.seekg(-1, ios::cur);

--chNum;

error("Illegal hexadecimal!", lines + 1);

return;

}

else //合法的16进制数

{

while (isdigit(ch) || (ch >= 'a' && ch <= 'f') || (ch >= 'A' && ch <= 'F'))

{

digits += ch;

infile.get(ch);

++chNum;

if (infile.eof()) break;

}

}

}

else //8进制处理

{

digits += ch;

infile.get(ch);

++chNum;

if (ch <= '0' || ch >= '7') //非法8进制数

{

infile.seekg(-1, ios::cur);

--chNum;

error("Illegal octal!", lines + 1);

}

else //合法8进制数

{

while (ch >= '0' && ch <= '7')

{

digits += ch;

infile.get(ch);

++chNum;

if (infile.eof()) break;

}

}

}

}

else //处理非零十进制

{

while (isdigit(ch))

{

digits += ch;

if (infile.eof()) break;

infile.get(ch);

++chNum;

}

if (ch == '.') //处理小数部分

{

digits += ch;

infile.get(ch);

++chNum;

if (!isdigit(ch))

{

error("Illegal float!", lines + 1);

return;

}

else

{

while (isdigit(ch))

{

digits += ch;

if (infile.eof()) break;

infile.get(ch);

++chNum;

}

if (ch == 'e' || ch == 'E')

{

digits += ch;

infile.get(ch);

++chNum;

if (ch == '+' || ch == '-')

{

digits += ch;

infile.get(ch);

++chNum;

if (!isdigit(ch))

{

error("Illegal number!", lines + 1);

return;

}

else

{

while (isdigit(ch))

{

digits += ch;

if (infile.eof()) break;

infile.get(ch);

++chNum;

}

}

}

else if (isdigit(ch))

{

digits += ch;

infile.get(ch);

++chNum;

while (isdigit(ch))

{

digits += ch;

if (infile.eof()) break;

infile.get(ch);

++chNum;

}

}

else

{

infile.seekg(-1, ios::cur);

--chNum;

error("Illegal number", lines + 1);

return;

}

}

}

}

else

{

infile.seekg(-1, ios::cur);

--chNum;

}

}

output(digits, "num", findNum(digits));

}

void strCon(char ch)//字符串字面量处理

{

string strCon;

int curLine = lines;//记录字符串面值量所在行号，用于错误处理

char tmpc = ch; //记录第一次得到的是单引号 还是 双引号，用于匹配

while (1)

{

infile.get(ch);

++chNum;

if (ch == '\\')

{

infile.get(ch);

++chNum;

if (ch == '\n' || ch == '\r')

{

--chNum;

++lines;

}

else if (infile.eof())

{

error("Can not find matching string : \"\"\" or \"'\" !", curLine + 1);

break;

}

else

strCon += ch;

}

else if (ch == tmpc)

{

output(tmpc + strCon + tmpc, "strcon", findStrCon(tmpc + strCon + tmpc));

break;

}

else if (ch == '\n' || ch == '\r')

{

--chNum;

++lines;

}

else if (infile.eof())

{

error("Can not find matching string : \"\"\" or \"'\" !", curLine+1);

break;

}

else

strCon += ch;

}

}

void otherChar(char ch)//注释，运算符等处理

{

int cur = infile.tellg();//记录下文件指针当前位置

int curChNum = chNum;

if (ch == '/') //首字母为'/'的串

{

infile.get(ch);

++chNum;

if (ch == '/') //注释方法为 //

{

while (ch != '\n' && ch != '\r' && !infile.eof())

{

infile.get(ch);

++chNum;

}

++lines;

}

else if (ch == '\*')//注释方法为 /\*...\*/

{

int curLine = lines;

char charCur = infile.get();

++chNum;

char charNext = infile.get();

++chNum;

while (charCur != '\*' || charNext != '/')

{

if (charCur == '\n' || charCur == '\r')

{

lines++;

--chNum;

}

if (infile.eof())

{

error("Can not find matching string:\"\*/\" !", curLine+1);

break;

}

charCur = charNext;

charNext = infile.get();

++chNum;

}

}

else//单字运算符 /

{

infile.seekg(-1, ios::cur);

--chNum;

output("/", "operator", findOperator("/"));

}

}

else//各类操作符

{

string str1 = "0"; //长度为1的字符串，记录有一个字符的运算符

string str2 = "00"; //长度为2的字符串，记录有两个字符的运算符

string str3 = "000"; //长度为3的字符串，记录有三个字符的运算符

str3[0] = str2[0] = str1[0] = ch;

str3[1] = str2[1] = infile.get();

++chNum;

str3[2] = infile.get();

++chNum;

if (findOperator(str3) != -1) //优先考虑三字符运算符

output(str3, "operator", findOperator(str3));

else

{

infile.seekg(-1, ios::cur);

--chNum;

if (findOperator(str2) != -1) //考虑两字符运算符

output(str2, "operator", findOperator(str2));

else

{

infile.seekg(cur);//文件指针返回到判断前的位置

chNum = curChNum;

if (findOperator(str1) != -1) //考虑单字符运算符

output(str1, "operator", findOperator(str1));

else if (findBoundary(str1) != -1) //考虑分隔符

output(str1, "boundary", findBoundary(str1));

else //非法字符

error("Illegal character " + str1 + "!", lines+1);

}

}

}

}

void output(string lexeme, string token, int location)//数据结果表

{

ofstream outfile("output.txt", ios::app);

if (!outfile)

{

cout << "output.txt open error!" << endl;

system("pause");

exit(-1);

}

outfile << lexeme << "\t\t\t\t" << "<" << token << "," << location << ">" << endl;

++tokenNum;

outfile.close();

}

int findID(string id)//遍历标识符表返回标识符所在位置，若没有在表尾插入

{

string str;

ifstream idIn("id.txt", ios::in);

if (!idIn)

{

cout << "id.txt open error!" << endl;

system("pause");

exit(-1);

}

int location = 0;

while (idIn >> str)

{

++location;

if (str == id)

{

idIn.close();

return location;//标识符已存在表中

}

}

idIn.close();

//标识符不在表中

ofstream idOut("id.txt", ios::app);

if (!idOut)

{

cout << "id.txt open error!" << endl;

system("pause");

exit(-1);

}

idOut << id << endl;

idOut.close();

return location + 1;

}

int findNum(string digit)

{

string str;

ifstream numIn("digit.txt", ios::in);

if (!numIn)

{

cout << "digit.txt open error!" << endl;

system("pause");

exit(-1);

}

int location = 0;

while (numIn >> str)

{

++location;

if (str == digit)

{

numIn.close();

return location;//标识符已存在表中

}

}

numIn.close();

//标识符不在表中

ofstream numOut("digit.txt", ios::app);

if (!numOut)

{

cout << "digit.txt open error!" << endl;

system("pause");

exit(-1);

}

numOut << digit << endl;

numOut.close();

return location + 1;

}

int findStrCon(string strCon)//遍历字符串常量表返回字符串所在位置，若没有在表尾插入

{

string str;

ifstream strIn("strCon.txt", ios::in);

if (!strIn)

{

cout << "strCon.txt open error!" << endl;

system("pause");

exit(-1);

}

// cout << strCon << endl;

int location = 0;

while (getline(strIn, str))

{

++location;

if (str == strCon)

{

strIn.close();

return location;//标识符已存在表中

}

}

strIn.close();

//标识符不在表中

ofstream strOut("strCon.txt", ios::app);

if (!strOut)

{

cout << "strCon.txt open error!" << endl;

system("pause");

exit(-1);

}

strOut << strCon << endl;

strOut.close();

return location + 1;

}

int findOperator(string s)//遍历运算符表返回运算符在表中位置，若不在返回-1

{

for (int loc = 0; loc < operatorNum; ++loc)

if (operators[loc] == s)

return loc + 1;

return -1;

}

int findBoundary(string s)//遍历分隔符表返回分隔符在表中位置，若不在返回-1

{

for (int loc = 0; loc < boundaryNum; ++loc)

if (boundary[loc] == s)

return loc + 1;

return -1;

}

void error(string tip, int line) //错误处理输出

{

cout << "Error! No." << line << " line:" << tip << endl;

errors++;

}