**北京邮电大学软件学院**

**2016－2017学年第一学期实验报告**

**课程名称： 编译原理**

**项目名称： 词法分析程序的设计和实现**

**项目完成人：**

**姓名：\_肖逸敏\_\_\_\_\_\_\_学号：\_\_2014211990\_\_\_\_\_\_**

**姓名：\_\_\_\_\_\_\_\_学号：\_\_\_\_\_\_\_\_**

**姓名：\_\_\_\_\_\_\_\_学号：\_\_\_\_\_\_\_\_**

**姓名：\_\_\_\_\_\_\_\_学号：\_\_\_\_\_\_\_\_**

**姓名：\_\_\_\_\_\_\_\_学号：\_\_\_\_\_\_\_\_**

**指导教师：\_\_\_\_\_\_杨正球\_\_\_\_\_\_\_\_\_\_\_\_**

**日 期： 2016 年 11 月 16 日**

1. **实验目的**

学习词法分析，掌握词法分析的内容。

1. **实验内容**

设计并实现C语言的词法分析程序，要求如下：

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，单词个数和字符个数，其中标点和空格不计算为单词，并输出统计结果。
4. 检查源程序中存在的错误，并可以报告错误所在的行列位置。
5. 发现源程序中存在错误后，进行适当的恢复，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。

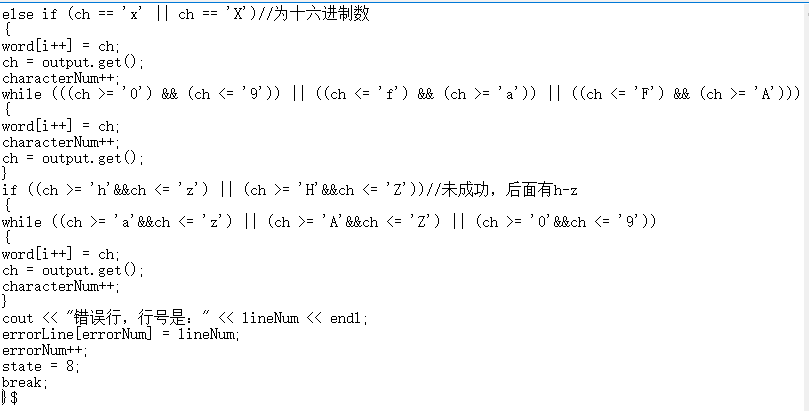
要求：采用C/C++作为实现语言，手工编写词法分析程序。

1. **实验环境**

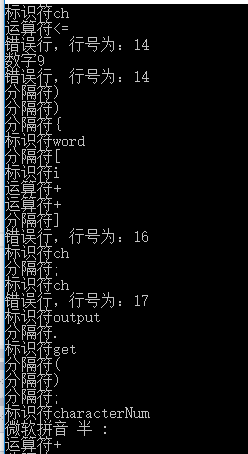
VS2013

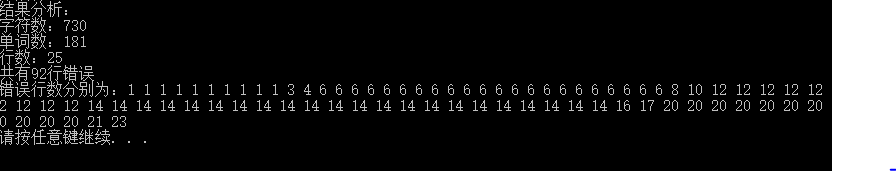
1. **实验结果**

输入文件：



输出结果：





1. **附录**

程序：

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<iostream>

#include<fstream>

#include<string.h>

using namespace std;

int wordNum=0;//单词数

int characterNum=0;//字符数

int lineNum=1;//行数

int errorNum = 0;//错误数

int errorLine[100];//错误行信息

void scanner()

{

ifstream output;

output.open("input.txt");//读取文件

char structer[11][10] = { "if", "else", "for", "return", "switch", "case", "break", "continue", "while", "do" ,"goto"};//基本结构

char dataType[14][10] = { "int", "char", "float", "double", "void","signed","unsigned","short","long","static","struct","union","typedef","volatile" };//数据类型

char preDeal[2][10] = { "include", "define" };//预处理符号

char word[30] = { '\0' };

int i = 0;//扫描结果数组下标

bool k = false;//记录单词状态

int a = 0;//扫描小数时循环的控制变量

int b = 0;//扫描注释时循环的控制变量

int state = 1;//循环中case的默认取值

char ch;

ch = output.get();

characterNum++;

while (ch != '$')//未结束

{

//输入字符是分隔符

if (ch == '\n' || ch == ' ' || ch == '\0' || ch == '\t')

{

if (ch == '\n')

{

lineNum++;

}

ch = output.get();

characterNum++;

}

else if (((ch >= 'a') && (ch <= 'z')) || ((ch >= 'A') && (ch <= 'Z')))//输入为字母

{

word[i++] = ch;

characterNum++;

ch = output.get();

while (((ch >= 'a') && (ch <= 'z')) || ((ch >= 'A') && (ch <= 'Z')) || ((ch >= '0') && (ch <= '9')||(ch=='\_')))

{

word[i++] = ch;

ch = output.get();

characterNum++;

}

//查询是不是基本结构

for (int j = 0; j < 11; j++)

{

if (strcmp(word, structer[j]) == 0)

{

k = true;//单词被找到

cout << "基本结构" << word << endl;

break;

}

}

//查询是不是数据类型

for (int j = 0; j < 11; j++)

{

if (strcmp(word, dataType[j]) == 0)

{

k = true;//单词被找到

cout << "数据类型" << word << endl;

break;

}

}

if (k == false)//为标识符

{

cout << "标识符" << word << endl;

}

wordNum++;

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

{

word[j] = '\0';

}

i = 0;

k = false;

}

else if ((ch >= '0') && (ch <= '9'))//输入为数字

{

word[i++] = ch;

ch = output.get();

characterNum++;

while (a != 1)

{

switch (state){

case 1://判断八进制或十六进制

char chpro;

chpro = word[i-1];

if (chpro == '0')//第一个数字为0

{

if ((ch >= '0') && (ch <= '7'))//为八进制

{

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

{

word[i++] = ch;

ch = output.get();

characterNum++;

}

if ((ch >= 'a'&&ch <= 'z') || (ch >= 'A'&&ch <= 'Z') || ch == '8' || ch == '9')//不成功，后面有字母或8，9

{

while ((ch >= 'a'&&ch <= 'z') || (ch >= 'A'&&ch <= 'Z') || (ch >= '0'&&ch <= '9'))

{

word[i++] = ch;

ch = output.get();

characterNum++;

}

cout << "错误行，行号是：" << lineNum << endl;

errorLine[errorNum] = lineNum;

errorNum++;

state = 8;

break;

}

else //成功为八进制数

{

cout << "八进制数" << word << endl;

wordNum++;

state = 8;

break;

}

}

else if (ch == 'x' || ch == 'X')//为十六进制数

{

word[i++] = ch;

ch = output.get();

characterNum++;

while (((ch >= '0') && (ch <= '9')) || ((ch <= 'f') && (ch >= 'a')) || ((ch <= 'F') && (ch >= 'A')))

{

word[i++] = ch;

characterNum++;

ch = output.get();

}

if ((ch >= 'h'&&ch <= 'z') || (ch >= 'H'&&ch <= 'Z'))//未成功，后面有h-z

{

while ((ch >= 'a'&&ch <= 'z') || (ch >= 'A'&&ch <= 'Z') || (ch >= '0'&&ch <= '9'))

{

word[i++] = ch;

ch = output.get();

characterNum++;

}

cout << "错误行，行号是：" << lineNum << endl;

errorLine[errorNum] = lineNum;

errorNum++;

state = 8;

break;

}

else

{

cout << "十六进制数" << word << endl;

wordNum++;

state = 8;

break;

}

}

else//既没可能是八进制也没可能是十六进制，跳转到状态2

{

state = 2;

break;

}

}

else//输入的第一个数字不是0

{

state = 2;

break;

}

case 2:

if ((ch >= '0') && (ch <= '9'))//输入数字

{

state = 2;//仍保持在状态2

word[i++] = ch;

ch = output.get();

characterNum++;

}

else if (ch == '.')//输入小数点

{

state = 3;//跳转到状态3

word[i++] = ch;

ch = output.get();

characterNum++;

}

else if (ch == 'e' || ch == 'E')//输入指数符号

{

state = 5;//转到状态5

word[i++] = ch;

ch = output.get();

characterNum++;

}

else if (ch == 'l')//判断是不是long常数

{

ch = output.get();

characterNum++;

word[i++] = ch;

cout << "long常数" << word << endl;//输出long常数

wordNum++;

state = 8;//跳转到状态8，扫描完毕

}

else if (ch == 'u')//判断是不是unsigned常数

{

ch = output.get();

characterNum++;

word[i++] = ch;

cout << "unsigned常数" << word << endl;//输出unsigned常数

wordNum++;

state = 8;;//跳转到状态8，扫描完毕

}

else

state= 7;

break;

case 3://识别小数点以后的操作

if (ch >= '0' && ch <= '9')//小数点后是数字

{

state = 4;//跳转到状态4

word[i++] = ch;

ch = output.get();

characterNum++;

}

else//小数点后不是数字，报错

{

cout << "错误行，行号是：" << lineNum << endl;

errorLine[errorNum] = lineNum;

errorNum++;

state = 8;

}

break;

case 4://小数点后是常数后

if (ch >= '0' && ch <= '9')//小数点后的常数后仍是常数

{

state = 4;

word[i++] = ch;

ch = output.get();

characterNum++;

}

else if (ch == 'e' || ch == 'E')//小数点后的常数后是指数符号

{

state = 5;//跳转到状态5

word[i++] = ch;

ch = output.get();

characterNum++;

}

else

state= 7;

break;

case 5://识别到是指数之后的操作

if (ch == '+' || ch == '-')//指数符号后是正负号

{

state = 6;//跳转到状态6

word[i++] = ch;

ch = output.get();

characterNum++;

}

else if (ch >= '0'&&ch <= '9')//指数符号后是数字

{

state = 6;//跳转到状态6

word[i++] = ch;

ch = output.get();

characterNum++;

}

else//指数符号后既不是数字也不是正负号，报错

{

cout << "错误行，行号是：" << lineNum << endl;

errorLine[errorNum] = lineNum;

errorNum++;

state = 8;

}

break;

case 6://指数之后的之后

if (ch >= '0'&&ch <= '9')

{

state = 6;

word[i++] = ch;

ch = output.get();

characterNum++;

}

else

state= 7;

break;

case 7:

if ((ch<'a'||ch>'z') && (ch<'A'||ch>'Z'))//输入符号，输出数字

{

cout << "数字" << word << endl;

wordNum++;

}

else if ((ch >= 'a'&&ch <= 'z') || (ch >= 'A'&&ch <= 'Z'))//输入字母，报错

{

while ((ch >= 'a'&&ch <= 'z') || (ch >= 'A'&&ch <= 'Z'))

{

word[i++] = ch;

ch = output.get();

characterNum++;

}

cout << "错误行，行号是：" << lineNum << endl;

errorLine[errorNum] = lineNum;

errorNum++;

}

else//输入其他，报错

{

cout << "错误行，行号是：" << lineNum << endl;

errorLine[errorNum] = lineNum;

errorNum++;

}

a = 1;

break;

case 8://扫描完成，跳出循环

a = 1;

break;

}

}

//扫描完毕，将参数初始化

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

{

word[j] = '\0';

}

i = 0;

a = 0;

state = 1;

}

else if (ch == '#')//识别是否为预处理符

{

ch = output.get();

characterNum++;

if (ch >= 'a'&&ch <= 'z')//#后输入的是小写字母

{

word[i++] = ch;

ch = output.get();

characterNum++;

while (ch >= 'a'&&ch <= 'z')

{

word[i++] = ch;

ch = output.get();

characterNum++;

}

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

{

if (strcmp(word, preDeal[j]) == 0)//判断是不是预处理符

{

k = true;//单词被找到，是预处理符

cout << "预处理符#" << word << endl;

wordNum++;

break;

}

}

for (int j = 0; j <= i; j++)//初始化

{

word[j] = '\0';

}

i = 0;

k = false;

}

else//不是预处理符，则#作为单独的运算符

{

cout << "运算符#" << endl;

wordNum++;

}

}

else if (ch == '<')//识别以<开头的符号

{

word[i++] = ch;

ch = output.get();

characterNum++;

if (ch == '=' || ch == '>')//识别<=,<>

{

word[i++] = ch;

ch = output.get();

characterNum++;

wordNum++;

cout << "运算符" << word << endl;

}

else if(ch=='<')//识别<<,<<=

{

word[i++] = ch;

ch = output.get();

characterNum++;

if (ch == '=')//识别<<=

{

word[i++] = ch;

wordNum++;

ch = output.get();

characterNum++;

cout << "运算符" << word << endl;

}

else//识别<<

{

cout << "运算符" << word << endl;

wordNum++;

}

}

else//<作为单独的运算符

{

cout << "运算符" << word << endl;

wordNum++;

}

for (int j = 0; j <= i; j++)//初始化

{

word[j] = '\0';

}

i = 0;

}

else if (ch == '>')//识别以>开头的符号

{

word[i++] = ch;

ch = output.get();

characterNum++;

//识别>=

if (ch == '=')

{

wordNum++;

word[i++] = ch;

ch = output.get();

characterNum++;

cout << "运算符" << word << endl;

}

else if (ch == '>')//识别>>=,>>

{

word[i++] = ch;

ch = output.get();

characterNum++;

if (ch == '=')

{

word[i++] = ch;

wordNum++;

ch = output.get();

characterNum++;

cout << "运算符" << word << endl;

}

}

else//>为单独运算符

{

cout << "运算符" << word << endl;

wordNum++;

}

for (int j = 0; j <= i; j++)//word初始化

{

word[j] = '\0';

}

i = 0;

}

else if (ch == ':')//识别以:开头的符号

{

word[i++] = ch;

ch = output.get();

characterNum++;

//识别:=

if (ch == '=')

{

word[i++] = ch;

ch = output.get();

characterNum++;

wordNum++;

cout << "运算符" << word << endl;

}

else//报错

{

cout << "错误行，行数为：" << lineNum << endl;

errorLine[errorNum] = lineNum;

errorNum++;

}

for (int j = 0; j <= i; j++)//word初始化

{

word[j] = '\0';

}

i = 0;

}

else if (ch == '\*' || ch == '+' || ch == '-')//识别运算符

{

word[i++] = ch;

ch = output.get();

characterNum++;

//识别\*=，/=，+=，-=

if (ch == '=')

{

word[i++] = ch;

ch = output.get();

characterNum++;

wordNum++;

cout << "运算符" << word << endl;

}

else//\*，+，-作为单独运算符

{

wordNum++;

cout << "运算符" << word << endl;

}

for (int j = 0; j <= i; j++)//word初始化

{

word[j] = '\0';

}

i = 0;

}

else if (ch == '/')//识别注释

{

ch = output.get();

if (ch == '/')//注释是//式

{

characterNum--;

ch = output.get();

while (ch != '\n')//注释//式读完

{

word[i++] = ch;

ch = output.get();

}

cout << "注释" << word << endl;

for (int j = 1; j <= i; j++)

{

word[j] = '\0';

}

i = 0;

characterNum++;

}

else if (ch == '\*')//注释是/\*\*/式

{

characterNum--;

ch = output.get();

while (ch != '\*')

{

word[i++] = ch;

ch = output.get();

}

ch = output.get();

if (ch == '/')

cout << "注释" << word << endl;

for (int j = 1; j <= i; j++)

{

word[j] = '\0';

}

i = 0;

ch = output.get();

characterNum++;

}

else

{

word[i++] = '/';

characterNum++;

if (ch == '=')

{

wordNum++;

word[i++] = ch;

ch = output.get();

characterNum++;

cout << "运算符" << word << endl;

}

else

{

wordNum++;

cout << "运算符" << word << endl;

}

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

{

word[j] = '\0';

}

i = 0;

}

}

else if (ch == '(' || ch == ')' || ch == '{' || ch == '}' || ch == ',' ||

ch == ';' || ch == '.' || ch == '[' || ch == ']' || ch == '\_' || ch == '"')

{

cout << "分隔符" << ch << endl;

wordNum++;

ch = output.get();

characterNum++;

}

else

{

cout << "错误行，行号为：" << lineNum << endl;

errorLine[errorNum] = lineNum;

errorNum++;

ch = output.get();

characterNum++;

}

}

output.close();

}

void main()

{

cout << "~~~~~~~~~~~~~~~~~语法分析器~~~~~~~~~~~~~~~~~~" << endl;

cout << "请把你的内容放在程序所在目录下的input.txt文档，以$结尾" << endl;

scanner();

cout << "结果分析：" << endl;

cout << "字符数：" <<characterNum<< endl;

cout << "单词数：" <<wordNum<< endl;

cout << "行数：" <<lineNum<< endl;

cout << "共有" <<errorNum<<"行错误"<< endl;

cout << "错误行数分别为：";

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

{

cout << errorLine[i] << " ";

}

cout << endl;

system("pause");

}