

编译原理程序设计I

实验报告

——C语言词法分析程序的设计与实现



姓 名：崔 思 颖

学 号：2018211290

学 院：计算机学院

班 级：2018211305

# 实验题目

采用C/C++作为实现语言，设计、实现一个C语言词法分析程序。

# 实验要求

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。

# 程序设计说明

词法分析的任务是从字符串表示的源程序中识别出具有独立意义的单词符号，其基本思想是根据扫描到的单词符号的第一个字符，判断该单词符号的种类并识别出整个单词。其中，识别关键字的思想为检索关键字列表；识别运算符、标识符、常数的思想是向后扫描直至单词结束，若不符合相应的格式则报错；识别界符、注释、字符串的思想是利用有限状态自动机，自动机卡死则报错。

输入的格式为一段C语言程序源代码，输出的格式为”<单词，行号，单词类别>”。

## 单词识别

(1)关键字

void main short long int double float while if else for break return char auto signed unsigned struct union enum static switch case default continue register const volatile typedef extern do goto sizeof

(2)运算符

= ! > < + - \* % & | ^ . >> << ++ -- && || -> == != >= <= += -= \*= %= &= |= ^=

(3)界符

, ; ( ) [ ] { } \ " # :

(4)标识符

ID = \_ | letter ( letter | digit )\*

(5)常数

NUM = digit ( digit\* )|(( e | . | x ) digit+ )

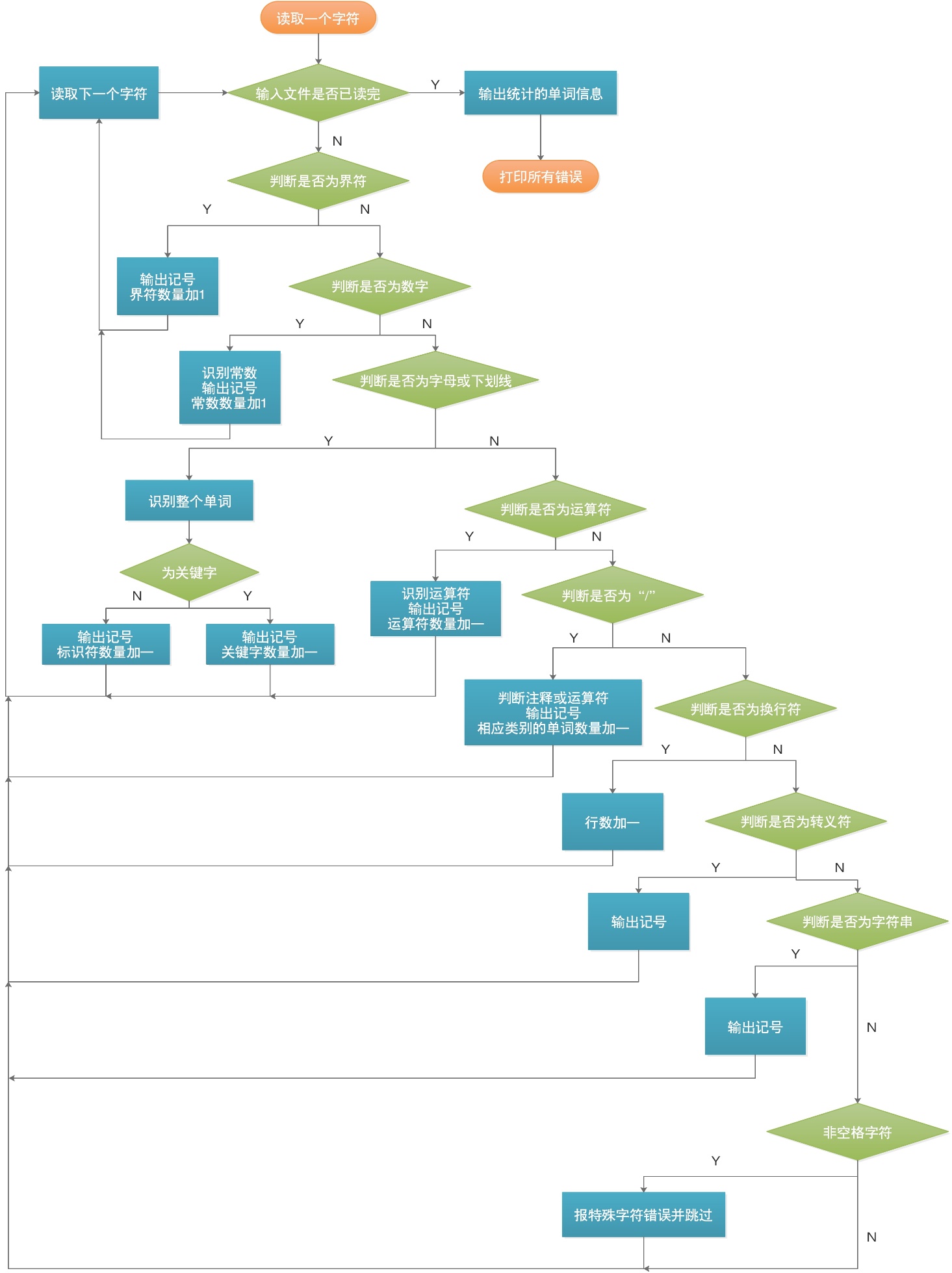
(6)注释

//… /\*…\*/

(7)字符串

“…”

## 算法流程图



## 源程序

* **数据结构**

enum errorType{DIGIT,COMMENT,CHAR};//拼写错误类型：数字、注释、特殊字符

typedef struct error

{

errorType type;//错误类型

int line;//行号

string s;//错误单词

}Error;

ifstream in("input.txt");//输入文件

ofstream out("output.txt");//输出文件

int Line=1;//语句行数

int Sum;//字符总数

int Opt;//运算符总数

int Id;//标识符总数

int Con;//常数总数

int Split;//分界符总数

int Key;//关键字总数

vector<Error> Errors;//存储错误位置

* **函数**

/\*查找s是否在保留字表keyword中\*/

bool lookupKeyword(string s)

/\*查找c是否在运算符表optword中\*/

bool lookupOpt(char c)

/\*查找c是否在分界符表splitword中\*/

bool lookupSplit(char c)

/\*读取一个字符，字符总数加一\*/

char getNextChar()

/\*拼写错误处理\*/

void spellError(char ch,string errorWord,errorType type)

/\*识别数字\*/

char recDigit(char ch)

/\*识别变量名或保留字\*/

char recAlpha(char ch)

/\*识别运算符\*/

char recOpt(char ch)

/\*识别注释并跳过\*/

char recCom(char ch)

/\*打印错误\*/

void printError()

**源代码及可执行程序见附件**

# 测试报告

## 输入（input.txt）

int main(int argc, char \*argv[])

{

int n,i,j,line,count;

scanf("%d",&n);

for(i=1;i<=2\*n-1;i++)

{

if(i>n)

line=2\*n-i;

else line=i;

for(j=1;j<=n-line;j++)

{printf(" ");}

for(count=1;count<=2\*line-1;count++)

{

if(count==1||count==2\*line-1)

printf("\*");

else printf(" ");

}

printf("\n");

}

return 0;

}

## 运行结果（output.txt）

< int , 1 , 保留字 >

< main , 1 , 保留字 >

< ( , 1 , 界符 >

< int , 1 , 保留字 >

< argc , 1 , 标识符 >

< , , 1 , 界符 >

< char , 1 , 保留字 >

< \* , 1 , 运算符 >

< argv , 1 , 标识符 >

< [ , 1 , 界符 >

< ] , 1 , 界符 >

< ) , 1 , 界符 >

< { , 2 , 界符 >

< int , 3 , 保留字 >

< n , 3 , 标识符 >

< , , 3 , 界符 >

< i , 3 , 标识符 >

< , , 3 , 界符 >

< j , 3 , 标识符 >

< , , 3 , 界符 >

< line , 3 , 标识符 >

< , , 3 , 界符 >

< count , 3 , 标识符 >

< ; , 3 , 界符 >

< scanf , 4 , 标识符 >

< ( , 4 , 界符 >

< "%d" , 4 , 字符串 >

< , , 4 , 界符 >

< & , 4 , 运算符 >

< n , 4 , 标识符 >

< ) , 4 , 界符 >

< ; , 4 , 界符 >

< for , 5 , 保留字 >

< ( , 5 , 界符 >

< i , 5 , 标识符 >

< = , 5 , 运算符 >

< 1 , 5 , 常数 >

< ; , 5 , 界符 >

< i , 5 , 标识符 >

< <= , 5 , 运算符 >

< 2 , 5 , 常数 >

< \* , 5 , 运算符 >

< n , 5 , 标识符 >

< - , 5 , 运算符 >

< 1 , 5 , 常数 >

< ; , 5 , 界符 >

< i , 5 , 标识符 >

< ++ , 5 , 运算符 >

< ) , 5 , 界符 >

< { , 6 , 界符 >

< if , 7 , 保留字 >

< ( , 7 , 界符 >

< i , 7 , 标识符 >

< > , 7 , 运算符 >

< n , 7 , 标识符 >

< ) , 7 , 界符 >

< line , 8 , 标识符 >

< = , 8 , 运算符 >

< 2 , 8 , 常数 >

< \* , 8 , 运算符 >

< n , 8 , 标识符 >

< - , 8 , 运算符 >

< i , 8 , 标识符 >

< ; , 8 , 界符 >

< else , 9 , 保留字 >

< line , 9 , 标识符 >

< = , 9 , 运算符 >

< i , 9 , 标识符 >

< ; , 9 , 界符 >

< for , 10 , 保留字 >

< ( , 10 , 界符 >

< j , 10 , 标识符 >

< = , 10 , 运算符 >

< 1 , 10 , 常数 >

< ; , 10 , 界符 >

< j , 10 , 标识符 >

< <= , 10 , 运算符 >

< n , 10 , 标识符 >

< - , 10 , 运算符 >

< line , 10 , 标识符 >

< ; , 10 , 界符 >

< j , 10 , 标识符 >

< ++ , 10 , 运算符 >

< ) , 10 , 界符 >

< { , 11 , 界符 >

< printf , 11 , 标识符 >

< ( , 11 , 界符 >

< " " , 11 , 字符串 >

< ) , 11 , 界符 >

< ; , 11 , 界符 >

< } , 11 , 界符 >

< for , 12 , 保留字 >

< ( , 12 , 界符 >

< count , 12 , 标识符 >

< = , 12 , 运算符 >

< 1 , 12 , 常数 >

< ; , 12 , 界符 >

< count , 12 , 标识符 >

< <= , 12 , 运算符 >

< 2 , 12 , 常数 >

< \* , 12 , 运算符 >

< line , 12 , 标识符 >

< - , 12 , 运算符 >

< 1 , 12 , 常数 >

< ; , 12 , 界符 >

< count , 12 , 标识符 >

< ++ , 12 , 运算符 >

< ) , 12 , 界符 >

< { , 13 , 界符 >

< if , 14 , 保留字 >

< ( , 14 , 界符 >

< count , 14 , 标识符 >

< == , 14 , 运算符 >

< 1 , 14 , 常数 >

< || , 14 , 运算符 >

< count , 14 , 标识符 >

< == , 14 , 运算符 >

< 2 , 14 , 常数 >

< \* , 14 , 运算符 >

< line , 14 , 标识符 >

< - , 14 , 运算符 >

< 1 , 14 , 常数 >

< ) , 14 , 界符 >

< printf , 15 , 标识符 >

< ( , 15 , 界符 >

< "\*" , 15 , 字符串 >

< ) , 15 , 界符 >

< ; , 15 , 界符 >

< else , 16 , 保留字 >

< printf , 16 , 标识符 >

< ( , 16 , 界符 >

< " " , 16 , 字符串 >

< ) , 16 , 界符 >

< ; , 16 , 界符 >

< } , 17 , 界符 >

< printf , 18 , 标识符 >

< ( , 18 , 界符 >

< "\n" , 18 , 字符串 >

< ) , 18 , 界符 >

< ; , 18 , 界符 >

< } , 19 , 界符 >

< return , 20 , 保留字 >

< 0 , 20 , 常数 >

< ; , 20 , 界符 >

< } , 21 , 界符 >

总计21行语句，334个字符。

其中包含26个运算符，36个标识符，12个常数，53个分界符，13个关键字。

## 输入（input.txt）

int main(int argc, char \*argv[])

{

int score;

scanf("%d",&score);

switch(score/10){

case 10:@

case 9:printf("A\n");break;

case 8:printf("B\n");break;

case 7:printf("C\n");break;

case 6:printf("D\n");break;

case 5:case 4:case 3:case 2:case 1:

case 0:printf("E\n");break;

default:printf("The score is out of range!\n");break;

}

return 0;

}

## 运行结果（output.txt）

< int , 1 , 保留字 >

< main , 1 , 保留字 >

< ( , 1 , 界符 >

< int , 1 , 保留字 >

< argc , 1 , 标识符 >

< , , 1 , 界符 >

< char , 1 , 保留字 >

< \* , 1 , 运算符 >

< argv , 1 , 标识符 >

< [ , 1 , 界符 >

< ] , 1 , 界符 >

< ) , 1 , 界符 >

< { , 2 , 界符 >

< int , 3 , 保留字 >

< score , 3 , 标识符 >

< ; , 3 , 界符 >

< scanf , 4 , 标识符 >

< ( , 4 , 界符 >

< "%d" , 4 , 字符串 >

< , , 4 , 界符 >

< & , 4 , 运算符 >

< score , 4 , 标识符 >

< ) , 4 , 界符 >

< ; , 4 , 界符 >

< switch , 5 , 保留字 >

< ( , 5 , 界符 >

< score , 5 , 标识符 >

< /,运算符 >

< 10 , 5 , 常数 >

< ) , 5 , 界符 >

< { , 5 , 界符 >

< case , 6 , 保留字 >

< 10 , 6 , 常数 >

< : , 6 , 界符 >

Illegal Character!

< case , 6 , 保留字 >

< 9 , 6 , 常数 >

< : , 6 , 界符 >

< printf , 6 , 标识符 >

< ( , 6 , 界符 >

< "A\n" , 6 , 字符串 >

< ) , 6 , 界符 >

< ; , 6 , 界符 >

< break , 6 , 保留字 >

< ; , 6 , 界符 >

< case , 7 , 保留字 >

< 8 , 7 , 常数 >

< : , 7 , 界符 >

< printf , 7 , 标识符 >

< ( , 7 , 界符 >

< "B\n" , 7 , 字符串 >

< ) , 7 , 界符 >

< ; , 7 , 界符 >

< break , 7 , 保留字 >

< ; , 7 , 界符 >

< case , 8 , 保留字 >

< 7 , 8 , 常数 >

< : , 8 , 界符 >

< printf , 8 , 标识符 >

< ( , 8 , 界符 >

< "C\n" , 8 , 字符串 >

< ) , 8 , 界符 >

< ; , 8 , 界符 >

< break , 8 , 保留字 >

< ; , 8 , 界符 >

< case , 9 , 保留字 >

< 6 , 9 , 常数 >

< : , 9 , 界符 >

< printf , 9 , 标识符 >

< ( , 9 , 界符 >

< "D\n" , 9 , 字符串 >

< ) , 9 , 界符 >

< ; , 9 , 界符 >

< break , 9 , 保留字 >

< ; , 9 , 界符 >

< case , 10 , 保留字 >

< 5 , 10 , 常数 >

< : , 10 , 界符 >

< case , 10 , 保留字 >

< 4 , 10 , 常数 >

< : , 10 , 界符 >

< case , 10 , 保留字 >

< 3 , 10 , 常数 >

< : , 10 , 界符 >

< case , 10 , 保留字 >

< 2 , 10 , 常数 >

< : , 10 , 界符 >

< case , 10 , 保留字 >

< 1 , 10 , 常数 >

< : , 10 , 界符 >

< case , 11 , 保留字 >

< 0 , 11 , 常数 >

< : , 11 , 界符 >

< printf , 11 , 标识符 >

< ( , 11 , 界符 >

< "E\n" , 11 , 字符串 >

< ) , 11 , 界符 >

< ; , 11 , 界符 >

< break , 11 , 保留字 >

< ; , 11 , 界符 >

< default , 12 , 保留字 >

< : , 12 , 界符 >

< printf , 12 , 标识符 >

< ( , 12 , 界符 >

< "The score is out of range!\n" , 12 , 字符串 >

< ) , 12 , 界符 >

< ; , 12 , 界符 >

< break , 12 , 保留字 >

< ; , 12 , 界符 >

< } , 13 , 界符 >

< return , 14 , 保留字 >

< 0 , 14 , 常数 >

< ; , 14 , 界符 >

< } , 15 , 界符 >

总计15行语句，359个字符。

其中包含3个运算符，12个标识符，13个常数，53个分界符，25个关键字。

-------------------------------Error-------------------------------

Line 6：存在非法字符

## 输入（input.txt）

int main(int argc, char \*argv[])

{

int a,b,c ,t;

scanf("%d%d%d",&a,&b,&c);//输入

/\*排序\*/

if(a>b){t=a;a=b;b=t;}

if(a>c){t=a;a=c;c=t;}

if(b>c){

t=b;b=c;c=t;

}

/\*输出\*

printf("%d%d%d\n,a,b,c);

return 0;

}

## 运行结果（output.txt）

< int , 1 , 保留字 >

< main , 1 , 保留字 >

< ( , 1 , 界符 >

< int , 1 , 保留字 >

< argc , 1 , 标识符 >

< , , 1 , 界符 >

< char , 1 , 保留字 >

< \* , 1 , 运算符 >

< argv , 1 , 标识符 >

< [ , 1 , 界符 >

< ] , 1 , 界符 >

< ) , 1 , 界符 >

< { , 2 , 界符 >

< int , 3 , 保留字 >

< a , 3 , 标识符 >

< , , 3 , 界符 >

< b , 3 , 标识符 >

< , , 3 , 界符 >

< c , 3 , 标识符 >

< , , 3 , 界符 >

< t , 3 , 标识符 >

< ; , 3 , 界符 >

< scanf , 4 , 标识符 >

< ( , 4 , 界符 >

< "%d%d%d" , 4 , 字符串 >

< , , 4 , 界符 >

< & , 4 , 运算符 >

< a , 4 , 标识符 >

< , , 4 , 界符 >

< & , 4 , 运算符 >

< b , 4 , 标识符 >

< , , 4 , 界符 >

< & , 4 , 运算符 >

< c , 4 , 标识符 >

< ) , 4 , 界符 >

< ; , 4 , 界符 >

< //输入 , 5 , 注释 >

< /\*排序\*/ , 5 , 注释 >

< if , 6 , 保留字 >

< ( , 6 , 界符 >

< a , 6 , 标识符 >

< > , 6 , 运算符 >

< b , 6 , 标识符 >

< ) , 6 , 界符 >

< { , 6 , 界符 >

< t , 6 , 标识符 >

< = , 6 , 运算符 >

< a , 6 , 标识符 >

< ; , 6 , 界符 >

< a , 6 , 标识符 >

< = , 6 , 运算符 >

< b , 6 , 标识符 >

< ; , 6 , 界符 >

< b , 6 , 标识符 >

< = , 6 , 运算符 >

< t , 6 , 标识符 >

< ; , 6 , 界符 >

< } , 6 , 界符 >

< if , 7 , 保留字 >

< ( , 7 , 界符 >

< a , 7 , 标识符 >

< > , 7 , 运算符 >

< c , 7 , 标识符 >

< ) , 7 , 界符 >

< { , 7 , 界符 >

< t , 7 , 标识符 >

< = , 7 , 运算符 >

< a , 7 , 标识符 >

< ; , 7 , 界符 >

< a , 7 , 标识符 >

< = , 7 , 运算符 >

< c , 7 , 标识符 >

< ; , 7 , 界符 >

< c , 7 , 标识符 >

< = , 7 , 运算符 >

< t , 7 , 标识符 >

< ; , 7 , 界符 >

< } , 7 , 界符 >

< if , 8 , 保留字 >

< ( , 8 , 界符 >

< b , 8 , 标识符 >

< > , 8 , 运算符 >

< c , 8 , 标识符 >

< ) , 8 , 界符 >

< { , 8 , 界符 >

< t , 9 , 标识符 >

< = , 9 , 运算符 >

< b , 9 , 标识符 >

< ; , 9 , 界符 >

< b , 9 , 标识符 >

< = , 9 , 运算符 >

< c , 9 , 标识符 >

< ; , 9 , 界符 >

< c , 9 , 标识符 >

< = , 9 , 运算符 >

< t , 9 , 标识符 >

< ; , 9 , 界符 >

< } , 10 , 界符 >

/\*输出\* is error!

< printf , 12 , 标识符 >

< ( , 12 , 界符 >

"%d%d%d\n,a,b,c); is error!

< return , 13 , 保留字 >

< 0 , 13 , 常数 >

< ; , 13 , 界符 >

< } , 14 , 界符 >

总计14行语句，215个字符。

其中包含16个运算符，35个标识符，1个常数，40个分界符，9个关键字。

-------------------------------Error-------------------------------

Line 11：未找到注释“/\*输出\*”的结束符

Line 12：字符串"%d%d%d\n,a,b,c);缺少右引号

## 分析说明

三个样例中，程序均能准确将input.txt中的C语言源代码的单词符号识别出来输出到output.txt中。在输出结尾可以把统计出的信息，包括语句行数、各类单词的个数以及字符总数，清晰地展示出来。若出现错误，程序能够将错误信息输出并跳过。如样例2中第六行的“@”符号无法识别，将其归为特殊符号错误；样例3中注释没有结束符、字符串缺少右引号程序都能检查出来，并将词法错误的位置和所在字段输出。

通过本次实验，我加深了对词法分析原理的理解，也进一步了解了平时使用的编译器其背后的原理，接下来的实验我也要再接再厉、认真完成。