**编译原理与技术程序设计一**

**--词法分析程序的设计与实现**

**实**

**验**

**报**

**告**

**班 级 2016211302**

**姓 名 毛维嘉**

**学 号 2016211067**

1. 实验目的

通过本实验的编程实践，学生可以充分了解词法分析的任务和方法，并且掌握词法分析程序设计的设计方法，对编译、原理和方法有完整的和清楚的解释，能够正确运用词法分析程序分析源程序。

1. 实验内容

设计并设计C语言的词法词法分析程序，要求实现以下功能：

1. 可以识别出用C语言编写的源程序中每一个单词符号，并以记号的形式输出每个单词符号。
2. 可以识别并跳出源程序中的注释。
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语法错误，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回复，使得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语法问题。
6. 程序编写思路

程序总体的编写思路是首先按照顺序读取源程序的每一个字符，将读到的每一个字符存入缓冲区，首先存入缓冲区的一半，之后进行分析，如果已经将当前缓冲区内容分析结束，则将源程序继续读入另一半缓冲区，循环下去直到程序结束。

从缓冲区读取每个字符进行分析，识别当前字符及其前后字符是否可以组成一个新的标识符、常数、浮点数、关键字等，将组成的内容以记号形式存入一个新的文件。此时若发现错误，则就直接报错并且跳过当前字符继续分析。最终分析结束之后将所有识别到的标识符、关键字、浮点数或者常数放在相应的数组中进行保存并且用记号的方式进行输出到文件中，并将相应的错误输出到错误分析文件中。

1. 具体实现

首先我先设置总体框架，在主函数中我设置了一个while循环来判断是否读入缓冲区，然后用switch语句来分别判断提取出的当前字符到底属于哪一种情况，最终将结果打印到文件中，文件全部分析之后程序结束。下面我将具体说明每个模块功能的实现过程。

下面首先列出全局变量的设置

ifstream file; //定义输入文件

ofstream outfile; //定义输出文件(也包括错误判断)

ofstream errorfile; //错误判断程序

char buffer[buffersize]; //定义一个完整缓冲区，但是可以从中间分开为前半部分和后半部分来用

int charnum=0; //字符数量

int row=1;

int rownum=1; //行数

int columnnum=0; //列数

int position=0;

char tokenstring[100]; //标识符数组

int tokennum=0;

int notenum;//注释数目

char notestring[2000]; //注释数组

char quotationstr[2000]; //引号内容

int quotation=0;

char keyword[32][20] =

{ "include","main","int","float","double","char","long","bool","short",

"if","else","for","while","do","struct","typedef","const","default",

"return","case","switch","break","continue","enum","goto","sizeof",

"static","void","union","unsigned","signed","extern" }; //定义32个关键字

char numstring[1000][32];

char idstring[1000][32];

int numlength=0; //相应numstring 数组长度

int idlength=0; //相应 idstring 数组长度

int searchnum=0; //找到一致元素在数组中的位置

int searchid=0; // 找到一致元素在数组中的位置

1. 读取文件存入缓冲区、

函数名称是void cinbuffer(int start)，这个start是指当前是从缓冲区什么位置开始提取文件中的字符，通过file.read()函数进行文件字符提取，然后放在缓冲区中，每次放缓冲区一半的空间。

1. 从缓冲区提取字符并且更新缓冲区

void getchar(char&ch)此函数的目的是从缓冲区提取字符，并且判断当前缓冲区的指针位置，如果此时缓冲区已经读完，则就调用读取文件存入缓冲区的函数填充缓冲区的另一半。

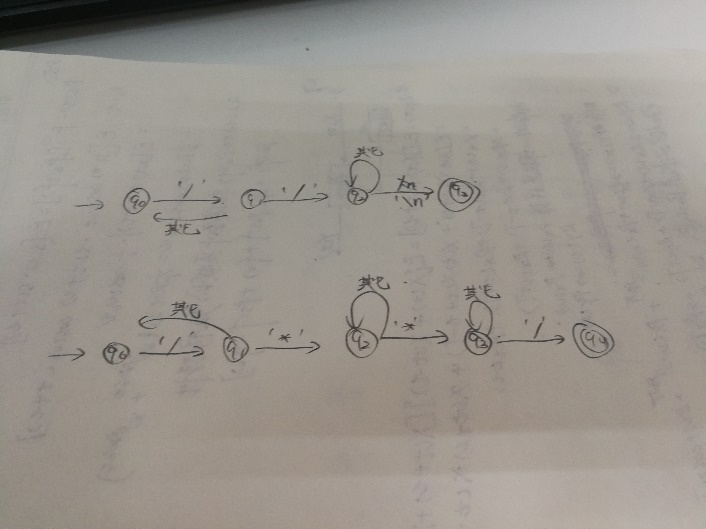
1. 去除空格

void ifspace(char&c)这个函数就是判断当前取出的字符是否是空格，如果是则再从缓冲区取出下一个字符。

1. 识别并且跳过注释

识别注释我们分为了两种情况，第一种情况是//双斜杠代表之后当前行的内容是注释，/\*则代表到\*/之前的内容都是注释。

此时我画出状态转移图

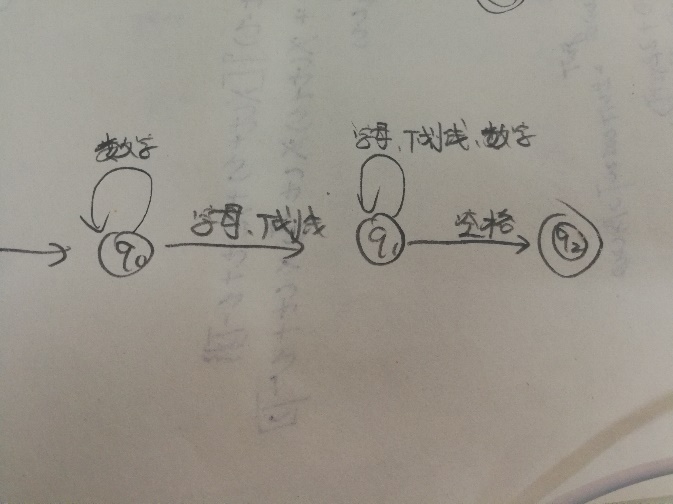


根据此状态转移图可以看出识别跳过注释函数实现的逻辑，并且我们将注释内容单独存放在一个位置，当做记号打印到输出文件中。

1. 识别标识符或者关键字

标识符或者关键字都是下划线或者字母开头，然后后面可以跟上数字字符下划线，直到遇到空格结束，识别完毕之后判断是否是关键字，如果不是关键词就当做标识符处理。

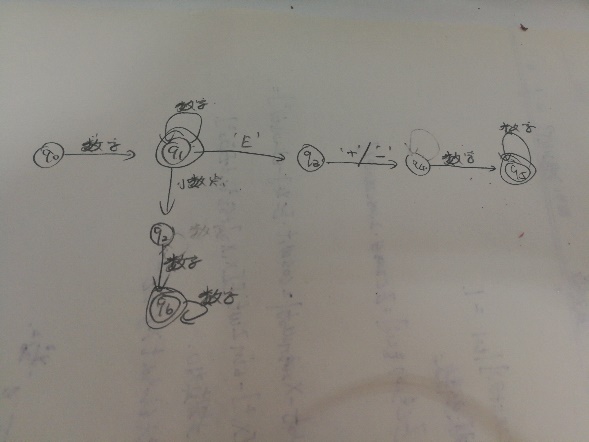
下面是识别标识符的DFA



1. 识别常数、浮点数

识别常数和浮点数首先是以数字开头，后面可以跟上若干数字或者一个小数点，然后可以跟上一个大写字符E表示浮点数，E后面跟上+或者-，之后再是若干数字。

下面是识别常数或者浮点数的DFA



1. 识别运算符

运算符有很多种，首先我们判断运算符是否要向前读取然后才能判断出来是哪一种运算符。这里面的运算符有=，==，+，-，||,&&,\*等等。

下面我们用+举例说明  
case'+':

getchar(ch);

if(ch=='=')

outfile<<"<assign\_op, += >"<< endl;

else if(ch=='+')

outfile<<"<++, - >"<< endl;

else

{

outfile<<"<+, - >"<< endl;

position--;

columnnum--;

}

break;

1. 识别转义字符：  
   下面我放一些部分识别转义字符的代码

case'\n':

outfile<<"<esc,换行>"<<endl;

row++;

columnnum=0;

break;

case'\t':

outfile<<"<esc,制表>"<<endl;

break;

1. 连接函数

Void cat()这个函数的目的就是将读出的字符与原来读出的字符相连，这样最后可以当做一个整体识别是否是一个关键字还是浮点数等等。

void cat(char ch) //连接函数

{

tokenstring[tokennum]=ch;

tokennum++;

}

1. 错误识别程序

这里面识别的错误包括不认识的字符、汉字，以及在识别浮点数中出现的错误，比如浮点数输入不完全等等。然后将错误信息打印到相应文件中，说明在第几行第几列出现错误。

void error()

{

outfile << "词法分析程序发现错误:位于第" << row << "行，第" << columnnum << "列" << endl;

errorfile << "词法分析程序发现错误:位于第" << row << "行，第" << columnnum << "列" << endl;

}

1. 统计字符和总行数

这些总体信息的统计工作均在从文件中读入缓冲区中完成，当文件读入缓冲区的时候，我们设置识别当前字符，如果识别到\n，则判断行数增加。当前行数和列数的判断则在程序执行过程中，每读取一个字符列数加一，遇到换行符的时候当前行数加一，列数清零。

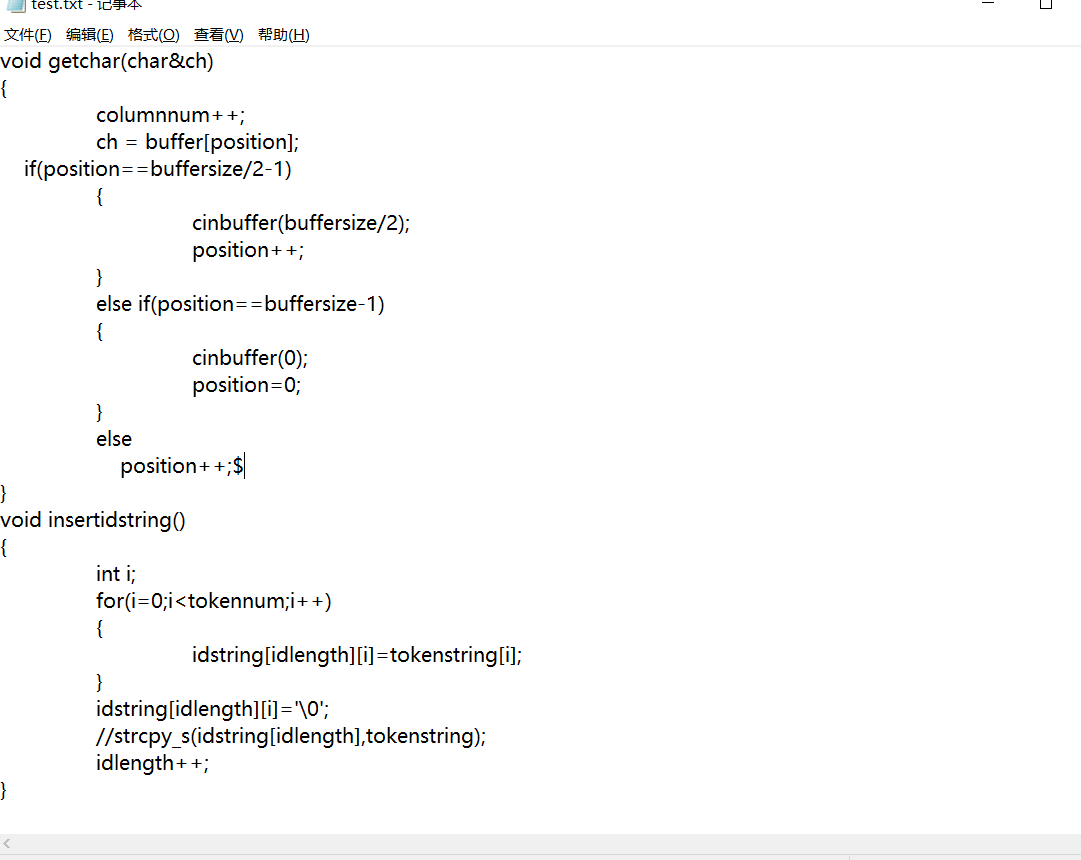
1. 函数说明

|  |  |
| --- | --- |
| 函数名称 | 函数功能 |
| void cinbuffer(int start) | 此时将分析文件内容部分放入缓冲区 |
| void getchar(char&ch) | 缓冲区中读取字符 |
| void ifspace(char&c) | 判断是否是空格 |
| int ifletter(char ch) | 判断是否是字母 |
| int ifnum() | 判断连接而成的数字是否可以组成常数或者浮点数 |
| void error() | 报错函数 |
| int ifid() | 判断是否是标识符 |
| void insertnumstring() | 判断出是一个数字之后存入到相应二维数组进行保存 |
| void insertidstring() | 判断出是标识符之后，存入二维数组中 |
| void cat(char ch) | 连接函数 |
| int ifkey() | 判断是否是关键字 |
| void cin1(char&ch) | 读取下一个字符并连接 |
| void copynote1(char&ch) | 识别第一种注释函数 |
| void copynote2(char&ch) | 识别第二种注释函数 |
| void copyquotation(char ch) | 识别出注释之后将注释存入相应数组 |

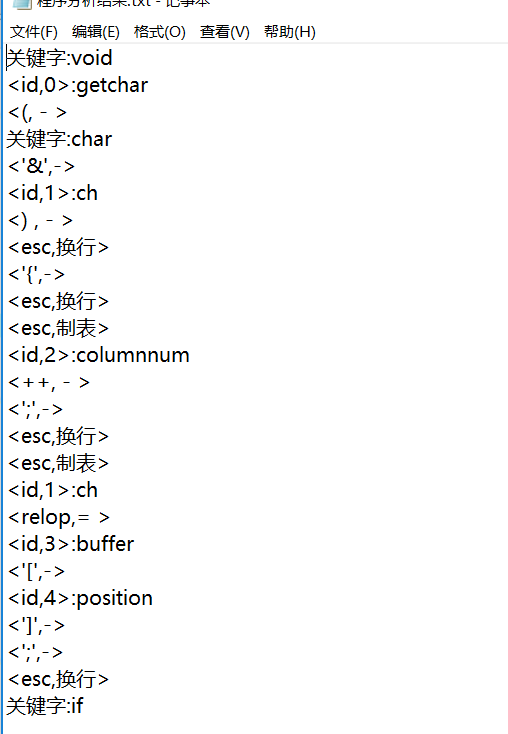
1. 编程环境

Windows 10

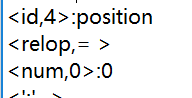
1. 程序测试分析
2. 首先我们自己写了一个测试程序进行分析，此分析程序中选取了改词法分析程序中的两个函数吧，并且人为加上了几个错误判断是否可以识别出来，下面我放上检测程序的截图。在第一个函数结束增加了一个$符号，判断词法分析程序是否可以识别出错误。



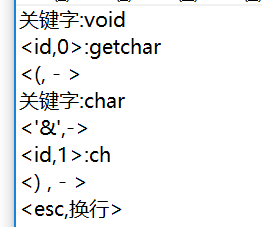
分析此程序之后，结果将放在两个文件中，一个是程序分析结果，另外一个是错误分析结果。



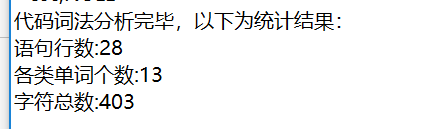
1. 本程序还将识别出来的标识符和所有数字存在相应的数组中并且按照记号的格式输出到分析文件中。



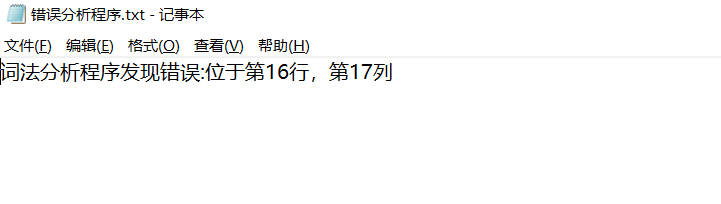
1. 关键字和换行的识别



1. 最后的统计结果



5、错误处理部分



1. 实验总结

进行完本次实验后，我对词法分析有了充分并且深刻的了解，知道了词法分析的整个过程并且也知道了我们最终要得到的结果。在实验中遇到了一些问题，并且通过解决这些问题进一步巩固了概念并且提升了编程的能力。

我通过这次实验首先明白了在编程之前提前设计和准备框架的重要性，首先我确定了整个词法分析程序的流程和步骤，第一步是设计全局变量，比如缓冲区的设置、读入字符提取符号，当前行列以及总体行列数目统计等等全局变量。其次我又设计了整体的大框架是通过while 循环并且通过switch语句进行判断来进行识别。然后我又将整个词法分析程序分了很多不同的模块，每个模块设计了相应的自动机进行识别，然后编写各个函数，不断填充在已经设计好的框架中，每添加一个模块的功能首先进行编辑调试及时发现错误，这样比较好检查出来。

我遇到的最大的问题其实是缓冲区的设置问题，最初将缓冲区设置的很大，导致我刚开始调试的时候测试程序比较下,无法发现缓冲区无法自动读入，结果先将缓冲区容量设置小了以后再去调试最终发现问题所在。

总体实验结束之后，对整个编译原理的词法分析这一部分有了非常深刻的理解和体会，但其实这也只是编译过程的第一步，这也为我以后语法分析和其他分析过程打下了坚实的基础。

1. 词法分析源代码

#include "string.h"

#include<iostream>

#include<string>

#include <fstream>

using namespace std;

#define buffersize 2000

ifstream file; //定义输入文件

ofstream outfile; //定义输出文件(也包括错误判断)

ofstream errorfile; //错误判断程序

char buffer[buffersize]; //定义一个完整缓冲区，但是可以从中间分开为前半部分和后半部分来用

int charnum=0; //字符数量

int row=1;

int rownum=1; //行数

int columnnum=0; //列数

int position=0;

char tokenstring[100]; //标识符数组

int tokennum=0;

int notenum;//注释数目

char notestring[2000]; //注释数组

char quotationstr[2000]; //引号内容

int quotation=0;

char keyword[32][20] =

{ "include","main","int","float","double","char","long","bool","short",

"if","else","for","while","do","struct","typedef","const","default",

"return","case","switch","break","continue","enum","goto","sizeof",

"static","void","union","unsigned","signed","extern" }; //定义32个关键字

char numstring[1000][32];

char idstring[1000][32];

int numlength=0; //相应numstring 数组长度

int idlength=0; //相应 idstring 数组长度

int searchnum=0; //找到一致元素在数组中的位置

int searchid=0; // 找到一致元素在数组中的位置

void print() //调试函数

{

int i;

for(i=0;i<20;i++)

cout<<buffer[i];

}

void cinbuffer(int start) //此时将分析文件内容部分放入缓冲区

{

int i=0;

char c;

while(i<buffersize/2&&!file.eof())

{

file.read(&c, 1);

buffer[i+start]=c;

if(c=='\n')

{

rownum++;

}

else charnum++;

i++;

}

if (file.eof())

{

buffer[start+i-1] = '\0';

}

}

void getchar(char&ch) //从缓冲区中读取字符

{

columnnum++;

ch = buffer[position];

if(position==buffersize/2-1)

{

cinbuffer(buffersize/2);

position++;

}

else if(position==buffersize-1)

{

cinbuffer(0);

position=0;

}

else

position++;

}

void ifspace(char&c) //判断是否是空格

{

if(c==' ')

{

getchar(c);

ifspace(c);

}

}

int ifletter(char ch) //判断是否是字母

{

if ((ch >= 97 && ch <= 122) || (ch >= 65 && ch <= 90))

return 1;

else

return 0;

}

int ifnum() //判断连接而成的数字是否可以组成常数或者浮点数

{

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

{

if (strcmp(tokenstring, numstring[i]) == 0)

{

searchnum = i;

return 1;

}

}

return 0;

}

void error() //报错函数

{

outfile << "词法分析程序发现错误:位于第" << row << "行，第" << columnnum << "列" << endl;

errorfile << "词法分析程序发现错误:位于第" << row << "行，第" << columnnum << "列" << endl;

}

int ifid() //判断是否是标识符

{

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

{

if(strcmp(tokenstring,idstring[i])==0)

{

searchid=i;

return 1;

}

}

return 0;

}

void insertnumstring() //判断出是一个数字之后存入到相应二维数组进行保存

{

int i;

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

{

numstring[numlength][i]=tokenstring[i];

}

numstring[numlength][i]='\0';

//strcpy\_s(numstring[numlength], tokenstring);

numlength++;

}

void insertidstring() //判断出是标识符之后，存入二维数组中

{

int i;

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

{

idstring[idlength][i]=tokenstring[i];

}

idstring[idlength][i]='\0';

//strcpy\_s(idstring[idlength],tokenstring);

idlength++;

}

void cat(char ch) //连接函数

{

tokenstring[tokennum]=ch;

tokennum++;

}

void print1() //调试函数

{

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

cout<<tokenstring[i];

cout<<endl;

}

int ifkey() //判断是否是关键字

{

for (int i = 0; i<32; i++)

{

if (strcmp(tokenstring, keyword[i]) == 0)

return 1;

}

return 0;

}

void cin1(char&ch) //读取下一个字符并连接

{

while(isdigit(ch))

{

cat(ch);

getchar(ch);

}

}

void copynote1(char&ch) //识别第一种注释函数

{

while(ch!='\n')

{

notestring[notenum]=ch;

getchar(ch);

notenum++;

}

}

void copynote2(char&ch) //识别第二种注释函数

{

char c;

while(1)

{

if(ch=='\*')

{

getchar(c);

if(c=='/')

break;

else position--;

}

notestring[notenum]=ch;

getchar(ch);

notenum++;

}

}

void copyquotation(char ch) //识别出注释之后将注释存入相应数组

{

quotationstr[quotation]=ch;

quotation++;

}

int main()

{

string str;

char ch=' ';

cout<<"您好，欢迎大家使用词法分析程序。"<<endl;

cout<<"首先请大家输入想要分析的文件名称:"<<endl;

cin>>str;

file.open(str, ios::in);

while(!file)

{

cout<<"不好意思,无法找到您想要分析的文件，请您重新输入。"<<endl;

cin>>str;

file.open(str, ios::in);

}

outfile.open("程序分析结果.txt");

errorfile.open("错误分析程序.txt");

cinbuffer(0); //首先将文件内容放入缓冲区的一半

while(ch!='\0')

{

getchar(ch);

ifspace(ch);

switch(ch)

{

case'/': //首先判断注释

getchar(ch);

notenum=0;

if(ch=='/')

{

getchar(ch);

copynote1(ch);

notestring[notenum]='\0';

outfile << "<注释," <<notestring<<">" << endl;

}

else if(ch=='\*')

{

getchar(ch);

copynote2(ch);

notestring[notenum]='\0';

outfile << "<注释," <<notestring<<">" << endl;

}

break;

case'a':case'b':case'c':case'd':case'e':case'f':case'g':case'h':case'i':case'j':case'k':case'l':case'm':case'n':

case'o':case'p':case'q':case'r':case's':case't':case'u':case'v':case'w':case'x':case'y':case'z':case'A':case'B':

case'C':case'D':case'E':case'F':case'G':case'H':case'I':case'J':case'K':case'L':case'M':case'N':case'O':case'P':

case'Q':case'R':case'S':case'T':case'U':case'V':case'W':case'X':case'Y':case'Z':case'\_': //判断关键字标识符

tokennum=0;

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

{

cat(ch);

getchar(ch);

//if(ch=='\n') row--;

}

position--;

columnnum--;

tokenstring[tokennum]='\0';

if(ifkey()==0)

{

if(ifid())

{

outfile << "<id," <<searchid<<">:"<<tokenstring << endl;

}

else

{

insertidstring();

outfile << "<id," <<idlength-1<<">:"<<tokenstring << endl;

}

}

else

{

outfile << "关键字:" << tokenstring << endl;

}

break;

case '0':case'1':case'2':case'3':case'4':case'5':case '6':case'7':case'8':case'9':

//判断常数及数字

tokennum = 0;

cin1(ch);

if(ch=='.')

{

cat(ch);

getchar(ch);

if(!isdigit(ch))

{

error();

break;

}

else cat(ch);

}

cin1(ch);

if(ch=='E')

{

cat(ch);

getchar(ch);

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

{

cat(ch);

getchar(ch);

if(!isdigit(ch))

{

error();

break;

}

else

cin1(ch);

}

}

else if (isdigit(ch))

cin1(ch);

tokenstring[tokennum]='\0';

position--;

columnnum--;

if(ifnum()==1)

{

outfile << "<num," << searchnum << ">:" << tokenstring << endl;

}

else

{

insertnumstring();

outfile << "<num," << numlength - 1 << ">:" << tokenstring << endl;

}

break;

case'<': //下面就是各种各样的符号

getchar(ch);

if(ch=='=')

outfile << "<relop,<= >"<< endl;

else if(ch=='>')

outfile << "<relop,<> >"<<endl;

else

{

outfile << "<relop,< >"<<endl;

position--;

columnnum--;

}

break;

case'>':

getchar(ch);

if(ch=='=')

outfile << "<relop,>= >"<< endl;

else

{

outfile << "<relop,> >"<<endl;

position--;

columnnum--;

}

break;

case'=':

getchar(ch);

if(ch=='=')

outfile << "<relop,== >"<< endl;

else

{

outfile << "<relop,= >"<< endl;

position--;

columnnum--;

}

break;

case':':

getchar(ch);

if(ch=='=')

outfile << "<assign\_op, := >"<< endl;

else

{

outfile << "<:,- >"<< endl;

position--;

columnnum--;

}

break;

case'+':

getchar(ch);

if(ch=='=')

outfile<<"<assign\_op, += >"<< endl;

else if(ch=='+')

outfile<<"<++, - >"<< endl;

else

{

outfile<<"<+, - >"<< endl;

position--;

columnnum--;

}

break;

case'-':

getchar(ch);

if(ch=='=')

outfile<<"<assign\_op, -= >"<< endl;

else if(ch=='-')

outfile<<"<--, - >"<< endl;

else

outfile<<"<-, - >"<< endl;

break;

case'\*':

getchar(ch);

if(ch=='=')

outfile<<"<assign\_op, \*= >"<< endl;

else

{

outfile<<"<\*, - >"<< endl;

position--;

columnnum--;

}

break;

case'\\':

getchar(ch);

if(ch=='=')

outfile<<"<assign\_op, \\= >"<< endl;

else

{

outfile<<"<\\, - >"<< endl;

position--;

columnnum--;

}

break;

case'(':

outfile<<"<(, - >"<< endl;

break;

case')':

outfile<<"<) , - >"<< endl;

break;

case'\n':

outfile<<"<esc,换行>"<<endl;

row++;

columnnum=0;

break;

case'\t':

outfile<<"<esc,制表>"<<endl;

break;

case ';':

outfile << "<';',->" << endl;

break;

case '{':

outfile << "<'{',->" << endl;

break;

case '}':

outfile << "<'}',->" << endl;

break;

case '[':

outfile << "<'[',->" << endl;

break;

case ']':

outfile << "<']',->" << endl;

break;

case ',':

outfile << "<',',->" << endl;

break;

case '#':

outfile << "<'#',->" << endl;

break;

case '.':

outfile << "<'.',->" << endl;

break;

case '&':

getchar(ch);

if(ch=='&')

outfile << "<logicop,and>" << endl;

else

{

position--;

columnnum--;

outfile << "<'&',->" << endl;

}

break;

case '|':

getchar(ch);

if(ch=='|')

outfile << "<logicop,or>" << endl;

else

{

position--;

columnnum--;

outfile << "<'|',->" << endl;

}

break;

case '!':

getchar(ch);

if(ch=='=')

outfile << "<relop,!=>" << endl;

else

{

position--;

columnnum--;

outfile<<"<'!',->" << endl;

}

break;

case '?':

outfile << "<?,->" << endl;

break;

case '\0':

outfile << "<esc,NULL>" << endl;

break;

case'"':

outfile <<"<\",->"<<endl;

quotation=0;

getchar(ch);

while(ch!='"')

{

copyquotation(ch);

getchar(ch);

}

quotationstr[quotation]='\0';

outfile<<"<quotation，"<<quotationstr<<">"<<endl;

outfile <<"<\",->"<<endl;

break;

case'\'':

outfile<<"<',->"<<endl;

break;

default:

error();

break;

}

}

outfile<<"代码词法分析完毕，以下为统计结果："<<endl;

outfile<<"语句行数:"<<rownum<<endl;

outfile<<"各类单词个数:"<<idlength<<endl;

outfile<<"字符总数:"<<charnum<<endl;

cout<<"感谢您的使用，代码全部的词法分析已结束，相应分析结果以及错误已在文件中显示。"<<endl;

file.close();

outfile.close();

errorfile.close();

system("pause");

return 0;

}