编译原理与技术实验报告

设计词法分析程序

一、实验环境

1. 实验平台：Windows10
2. 编译环境：Visual Studio Code，Dev C++
3. 编程语言：C++
4. 版本号：1.0.0

二、实验内容及要求

设计并分析C语言的词法分析程序，以记号的形式输出每个单词符号，可以识别并跳过注释，统计源程序的中的语句行数，各类单词的个数以及字符总数并输出统计结果。检查源程序中的词法错误，并报告错误的位置。

三、程序设计

主要数据结构

词法分析类

```
class analysis                          //词法分析类
{
    public:
    int forward;                        //字符指针，向前指针
    int Judge_Keyword(string &a);       //判断字符串是否是关键字
    char get_char(string& ab,char c); //根据字符指针读入下一个字符
    char get_nbc(string& ab,char c);  //跳过空格
    void ana(string& ab);               //构建自动机词法分析
    {
        char c;                         //当前读入的字符
        string token;                   //字符数组存放当前正在识别的字符串
        state=0;                        //置自动机状态为初始态
        do
        {
            switch(state)
            {
                case 0:                 //初始状态
                case 1:                 //标识符状态
                case 2:                 //常数状态
                case 3:                 //小数点状态
                case 4:                 //小数状态
                case 5:                 //指数状态
                case 6:
                case 7:
                case 8:                 //<状态
                case 9:                 //>状态
                case 10:                //:状态
                case 11:                //+状态
                case 12:                //-状态
                case 13:                //*状态
                case 14:                ///状态判断是否是除号或注释
                case 15:                //&状态
                case 16:                //|状态
                case 17:                //!状态
                case 18:                ///*注释状态
                case 19:                //=状态
                case 20:                //错误状态
                case 21:                //转义字符状态
            }
        }
        while(forward<ab.size());
    }
    void output();                      //输出函数
}
```

主函数

```
ifstream fin(filename,ios::in);
while(std::getline(fin,temp))
{
    string buffer;
    buffer=temp;                        //读入缓冲区
    if(buffer=="")//空行
    {
        line_count++;
        char_count+=buffer.length()+1;
        temp="";
        continue;
    }
    char a=buffer.at(0);
    if(a=='#')                          //删去预编译代码，不交给词法分析程序处理
    {
        line_count++;
        char_count+=buffer.length()+1;
        temp="";
        continue;
    }
    ANALYSIS.forward=0;
    ANALYSIS.ana(buffer);
    line_count++;
    char_count+=buffer.length()+1;    //添加'\n'符号
    temp="";
}
```

主函数将按行读取数据,将每行读到的字符读入缓冲区`buffer`中，再将`buffer`作为参数传入一个用来词法分析的类，其中包含利用自动机进行分析所用的函数。但是主函数若检测到`#include`等预编译内容，将不交给词法分析程序处理。在`ana()`函数中,使用循环结构逐个读取缓冲区中每个字符，读一个字符进行一次循环如果读到空字符就跳过不分析，直到前进字符指针指向缓冲区末尾。每次分析字符时通过`switch-case`结构进行选择，根据字符种类使自动机从初始状态进入不同状态，根据状态进行相应处理。

错误处理

由于C语言中不允许存在@、$符号，因此本程序在读到这两种字符时进入`state=20`即错误处理状态，继续读到该字符串末尾，然后字符串和错误发生的行数存入全局的`vector string`数组中。在数字的判断中，若程序读到E或e字符，则转入指数判断状态，即`state=5`。但是若在E或e后或者E或e后的+-号后不是数字，则判断为指数形式错误，将指数和行数存入`vector`中。另外，程序也会识别用户自定义标识符中出现数字开头的不合法情况。最后在输出模块`output()`将所有词法错误的行数和内容输出。

符号表

程序将以二元组形式输出符号表。即`<种类,内容>`形式。如`<NUM,1E+6>`、`<KEYWORD,int>`等。

四、程序输入输出及执行结果

输入形式

程序运行时将询问要读取的文本文件名，输入文件名后，程序将自行读取文件内容。

输出形式

程序首先逐行输出符号表，然后输出统计的行数、关键字、分隔符、运算符、标识符、数字和字符总数。

执行结果

test1.txt

```
#include<stdio.h>
char\o <c=string+=h 89 |double 7E+x 23
       ||int+forward=//123
1+2=3.78*5-753E-97
+-*/  67  10E+9 string token;10E+9
&      1&&state>=ab 2ab
       *=do=int /*xy@z*/
{@$td return
   int (state)31E+64
a \n
1E+6
1E6
```

结果:

```
请输入进行词法分析的文件名
test1.txt
<KEYWORD,char>
<SEPARATOR,\o>
<OPERATOR,<>
<IDENTIFIER,c>
<OPERATOR,=>
<IDENTIFIER,string>
<OPERATOR,+=>
<IDENTIFIER,h>
<NUM,89>
<OPERATOR,|>
<KEYWORD,double>
<IDENTIFIER,x>
<NUM,23>
<OPERATOR,||>
<KEYWORD,int>
```

```
<NUM,31E+64>
<IDENTIFIER,a>
<SEPARATOR,\n>
<NUM,1E+6>
<NUM,1E6>
行数为：12
关键字个数为：7
分隔符个数为：6
运算符个数为：20
标识符个数为：12
数字总数为：15
字符总数为：237
错误为:
第2行指数7E+非法
第8行@$td非法

--------------------------------
Process exited after 14.77 seconds with ret
请按任意键继续. . .
```

test2.txt

```
#include<stdio.h>
int main ()
{
int a,b;
scanf("%d  %d",&a,&b);
printf("%d",a+b);
return 0;
}
```

结果:

```
请输入进行词法分析的文件名
test2.txt
<KEYWORD,int>
<IDENTIFIER,main>
<SEPARATOR,(>
<SEPARATOR,)>
<SEPARATOR,{>
<KEYWORD,int>
<IDENTIFIER,a>
<SEPARATOR,,>
<IDENTIFIER,b>
<SEPARATOR,;>
<IDENTIFIER,scanf>
<SEPARATOR,(>
<SEPARATOR,">
<OPERATOR,%>
<IDENTIFIER,d>
<OPERATOR,%>
<IDENTIFIER,d>
<SEPARATOR,">
<SEPARATOR,,>
```

```
<OPERATOR,+>
<IDENTIFIER,b>
<SEPARATOR,)>
<SEPARATOR,;>
<KEYWORD,return>
<NUM,0>
<SEPARATOR,;>
<SEPARATOR,}>
行数为: 8
关键字个数为: 3
分隔符个数为: 20
运算符个数为: 6
标识符个数为: 12
数字总数为: 1
字符总数为: 94

--------------------------------
Process exited after 7.412 seconds with retu
请按任意键继续. . .
```

test3.txt

```
#include<stdio.h>
/*递推算法*/
int main()
{
	int n,a[501],k;
	scanf("%d",&n);
	k=n/2;
	a[0]=1;
	a[1]=2;
	for(int i=1;i<=k;i++)
	{
		a[i]=a[i-1]+a[i/2];
	}
	printf("%d",a[k]);
	return 0;
}
```

结果:

```
请输入进行词法分析的文件名
test3.txt
<KEYWORD,int>
<IDENTIFIER,main>
<SEPARATOR,(>
<SEPARATOR,)>
<SEPARATOR,{>
<KEYWORD,int>
<IDENTIFIER,n>
<SEPARATOR,,>
<IDENTIFIER,a>
<SEPARATOR,[>
<NUM,501>
<SEPARATOR,]>
<SEPARATOR,,>
<IDENTIFIER,k>
<SEPARATOR,;>
<IDENTIFIER,scanf>
<SEPARATOR,(>
<SEPARATOR,">
<OPERATOR,%>
```

```
<SEPARATOR,]>
<SEPARATOR,)>
<SEPARATOR,;>
<KEYWORD,return>
<NUM,0>
<SEPARATOR,;>
<SEPARATOR,}>
行数为：16
关键字个数为：5
分隔符个数为：44
运算符个数为：14
标识符个数为：25
数字总数为：10
字符总数为：193

--------------------------------
Process exited after 6.374 seconds with retu
请按任意键继续. . .
```

test4.txt

```
#include<stdio.h>
#include<string.h>
int main()
{
        char a[14];
        int n=0,k=0;
        scanf("%s",a);
        for(int i=0;i<12;i++)
        {
                if(a[i]>47&&a[i]<58)
                {
                        k++;
                        n=n+(a[i]-'0')*k;
                }
        }
        if(n%11==(a[12]-'0')||n%11==10&&a[1
```

结果：

```
请输入进行词法分析的文件名
test4.txt
<KEYWORD,int>
<IDENTIFIER,main>
<SEPARATOR,(>
<SEPARATOR,)>
<SEPARATOR,{>
<KEYWORD,char>
<IDENTIFIER,a>
<SEPARATOR,[>
<NUM,14>
<SEPARATOR,]>
<SEPARATOR,;>
<KEYWORD,int>
<IDENTIFIER,n>
<OPERATOR,=>
<NUM,0>
<SEPARATOR,,>
<IDENTIFIER,k>
<OPERATOR,=>
<NUM,0>
<SEPARATOR,;>
```

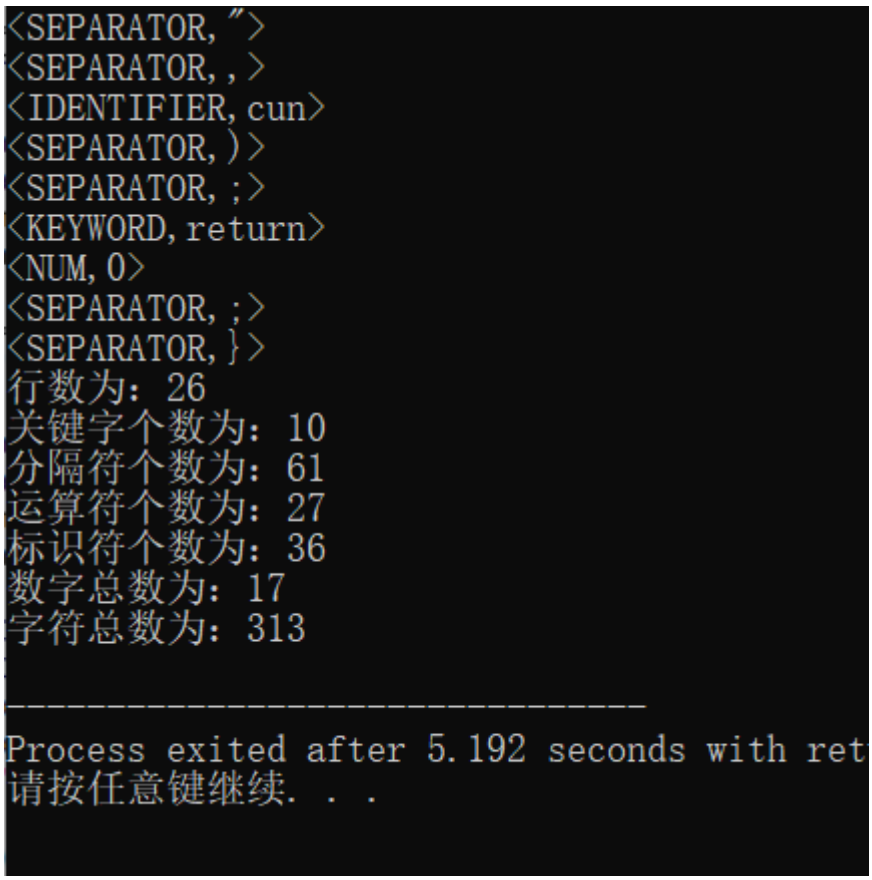
```
<NUM,11>
<SEPARATOR,)>
<SEPARATOR,;>
<SEPARATOR,}>
<SEPARATOR,}>
<KEYWORD,return>
<NUM,0>
<SEPARATOR,;>
<SEPARATOR,}>
行数为: 35
关键字个数为: 12
分隔符个数为: 86
运算符个数为: 28
标识符个数为: 37
数字总数为: 21
字符总数为: 427

--------------------------------
Process exited after 6.869 seconds with ret
请按任意键继续. . .
```

test5.txt

```
#include<stdio.h>
int main()
{
	int a[12],b=0,c[12],cun=0,month=0;
	for(int i=0;i<12;i++)
	{
		scanf("%d",&a[i]);
	}
	for(int i=0;i<12;i++)
	{
		if(b+300<a[i])
		{
			printf("-%d",i+1);
			return 0;
		}
		else
		{
			b=b+300-a[i];
```

结果:

```
请输入进行词法分析的文件名
test5.txt
<KEYWORD,int>
<IDENTIFIER,main>
<SEPARATOR,(>
<SEPARATOR,)>
<SEPARATOR,{>
<KEYWORD,int>
<IDENTIFIER,a>
<SEPARATOR,[>
<NUM,12>
<SEPARATOR,]>
<SEPARATOR,,>
<IDENTIFIER,b>
<OPERATOR,=>
<NUM,0>
<SEPARATOR,,>
<IDENTIFIER,c>
<SEPARATOR,[>
<NUM,12>
<SEPARATOR,]>
```



五、程序改进思路

程序读取每行末尾时，若有空格或`Tab`则会因为字符前进指针自加导致超出缓冲区范围而程序中止。后来优化了`forward`指针自加的条件，使程序正常运行。