# **《编译技术》实验报告**

实验名称：词法分析程序

姓名：牟洪锋

学号：U201417220

班级：软件工程1404班

指导教师：胡福林

2016年12月17日

1. 实验题目

对下述单词表定义的语言设计编制一个词法分析器。单词符号及种别表和词法分析器功能及基本要求如下：

（1）单词符号及种别表

|  |  |  |
| --- | --- | --- |
| 单词符号 | 种别编码 | 单词值 |
| main | 1 |  |
| int | 2 |  |
| float | 3 |  |
| double | 4 |  |
| char | 5 |  |
| if | 6 |  |
| else | 7 |  |
| do | 8 |  |
| while | 9 |  |
| l(l|d)\* | 10 | 内部字符串 |
| ( +|-|ε ) dd\*(.dd\* | ε)( e ( +|-|ε ) dd\*|ε) | 20 | 二进制数值表示 |
| = | 21 |  |
| + | 22 |  |
| - | 23 |  |
| \* | 24 |  |
| / | 25 |  |
| ( | 26 |  |
| ) | 27 |  |
| { | 28 |  |
| } | 29 |  |
| , | 30 |  |
| ; | 31 |  |
| > | 32 |  |
| >= | 33 |  |
| < | 34 |  |
| <= | 35 |  |
| == | 36 |  |
| != | 37 |  |
| # | 0 |  |

1. 实验目的

通过该实验,熟练应用编译原理关于词法分析的基本理论和方法；学会用C/C++高级程序设计语言设计一个词法分析器；加深对编译原理理论的分析理解，提高实际操作和解决具体问题的能力。

1. 实验要求

处理用户提交的符合上述词法的源代码序列，进行词法分析，并输出单词二元组。

例如输入：a+b-c#

输出:(10,a)

(22,+)

(10,b)

(23,-)

(10,c)

(0,#)

1. 实验步骤

基本设计思路

1. 首先将基本字作为特殊符号处理，识别基本字，查表，给出其种别码。基本字的初始值定为：

**char\*rwtab[9]={"main","int","float","double","char","if","else","do","while"}; （字符指针数组）**

1. 识别无符号整数是将数字串转换为无符号整数。我们在getchar()的时候是把数字当做字符从外部输出读取的。将数字串789#转换为整数：(7\*10+8)\*10+9=789送到sum中
2. 对于科学计数法的数采取分步处理：首先将e前面的数字用double型的变量存储起来，

再对e后面的数字判断正负，若为正，则用10的后面数字次方处理。若为负责用0.1的数字次方处理。

1. 对两个正负号重叠进行处理，判断其为加减号还是正负号。
2. 对于有词法错误的部分采取屏蔽的方法，不显示错误的部分。

正规式

状态转换图

流程框图

开始

接收字符串

调用scanner函数

出错处理

种别码=0?

否

Warn=0？

是

否

是

结束

算法设计

词法分析程序所用的较为重要的全局变量和需调用的函数如下：

char prog[80]= {'\0'},

token[8]; /\*存放构成单词符号的字符串\*/

char ch; /\*存放需要判断的字符\*/

char ch2; /\*存放需要判断的字符\*/

char ch3; /\*存放需要判断的字符\*/

char ch4; /\*存放需要判断的字符\*/

int syn, /\*存放单词字符的种别码\*/

n,

sum, sum2, /\*存放整数型单词\*/

m,p; /\*p是缓冲区prog的指针，m是token的指针\*/

double xiaoshu; /\*存放double型单词\*/

float k; /\*存放0.1进行科学计数法负次幂运算\*/

int flag=1; /\*正负判断符\*/

int flag2=1; /\*正负判断符\*/

int j;

int t=10; /\*存放10进行科学计数法负次幂运算\*/

int warn=0; /\*错误判断符\*/

double g; /\*存放科学计数法变量\*/

char \*gjz[9]= {"main","int","float","double","char","if","else","do","while"};

/\*关键字\*/

bool isalpha() /\*判断是否为字母\*/

bool isdigit() /\*判断是否为数字\*/

函数相关说明

isalpha()和 isdigit()函数均为ctype.h头文件内的分类函数，对于字母和数字进行判断，返回值为布尔值。

输入与输出

词法分析程序的输入输出情况

输入：判断文法的一段字符串（以#为结束符）

输出：单词二元组

例如：a+b\*c+12.345

输出:

(10,a)

(22,+)

(10,b)

(24,\*)

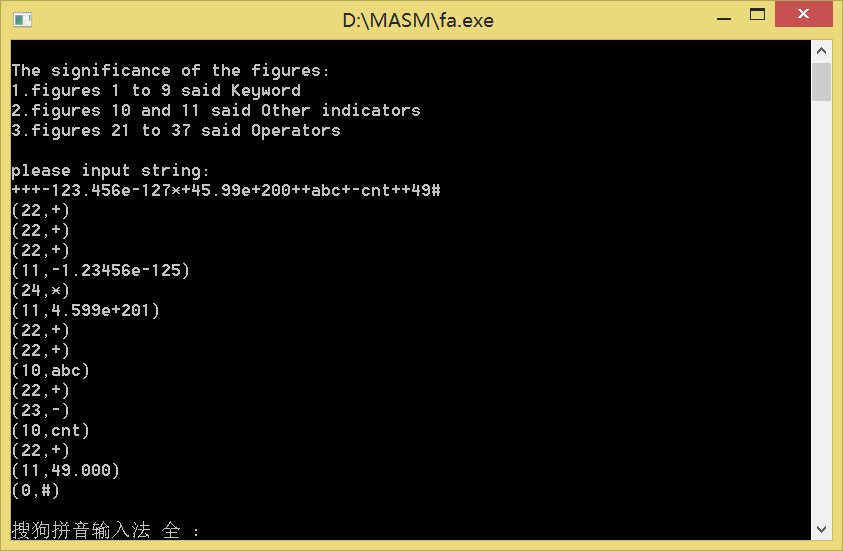
(10,c)

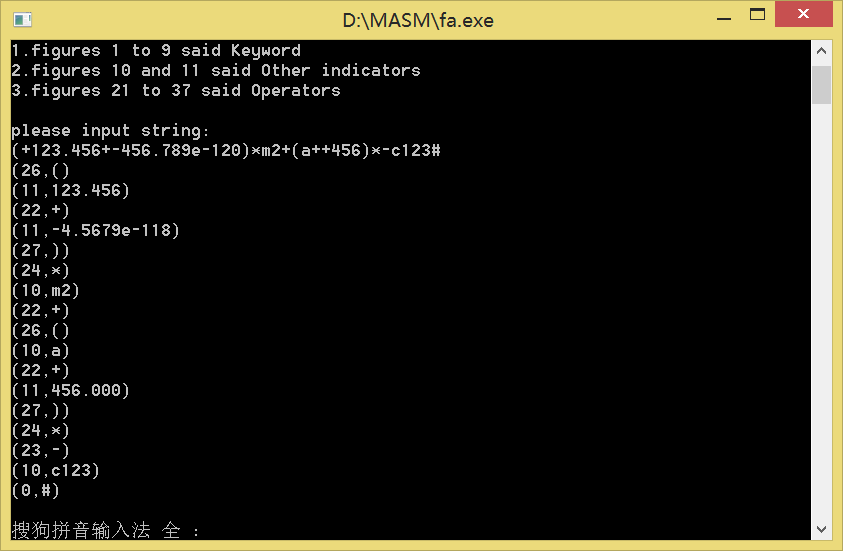
(22,+)

(11,12.345)

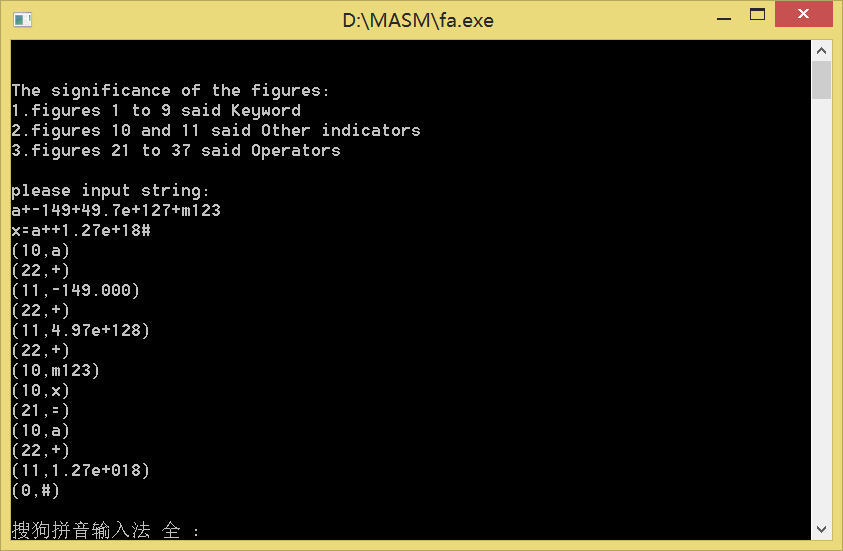
(0,#)

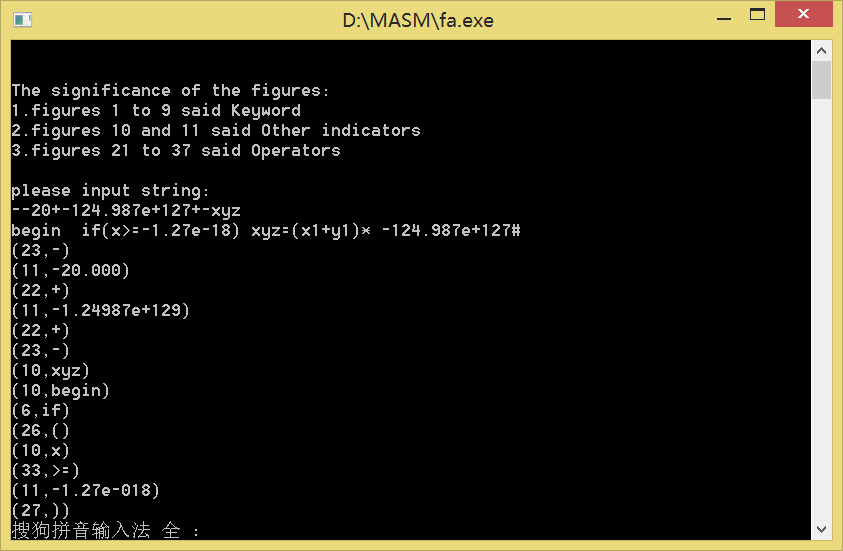
程序运行结果

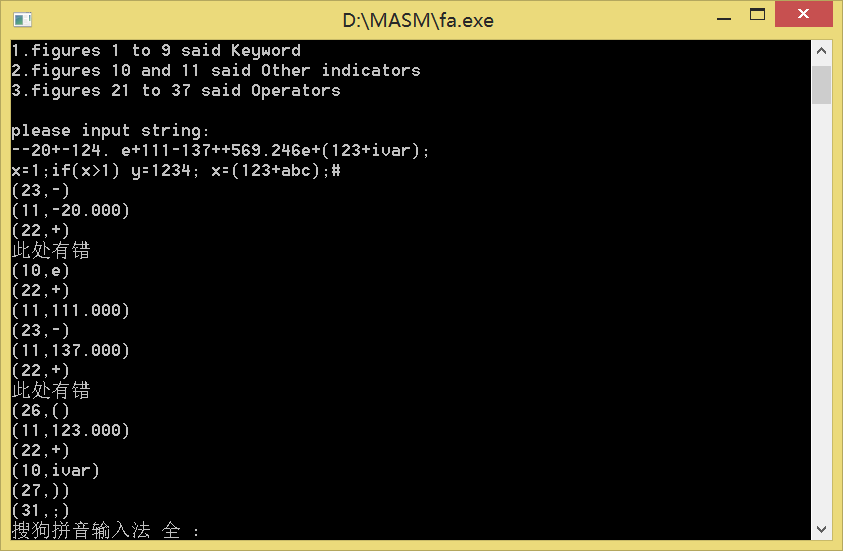












1. 实验方案设计实现

代码由c语言编写，通过设置判断符和数字处理完成词法分析。

1. 程序亮点描述

调用了c语言的库中的分类函数，能够对错误的地方指出并屏蔽掉有问题的部分。

1. 实验程序使用说明

用户输入待识别字符串（并以“#”结尾，表示字符串输入结束），回车后程序自动输出词法分析结果。

1. 实验心得体会

这次实验是这个学期的第一个c语言实验，很久没用过c语言了，对其中的一些函数用法已经开始陌生了。通过这次实验，我一遍遍理清了词法分析的流程和思路，用一天的时间完成了基本的词法判断。但是错误处理的问题苦恼了我很久，在老师检查的时候也没有解决这个问题，后来我又慢慢分析了错误发生的各种种类，现在已经能基本解决大部分的错误问题，对其进行屏蔽和修改。改程序是一个需要耐心和细心的过程，需要冷静的分析和考量，不能着急。总之，这个实验对我的编程能力又很大的帮助，让我受益匪浅。

1. 源程序清单

#include "stdio.h" /\*定义I/O库所用的某些宏和变量\*/

#include "string.h" /\*定义字符串库函数\*/

#include "conio.h" /\*提供有关屏幕窗口操作函数\*/

#include "ctype.h" /\*分类函数\*/

char prog[80]= {'\0'},

token[8]; /\*存放构成单词符号的字符串\*/

char ch;

char ch2;

char ch3;

char ch4;

int syn, /\*存放单词字符的种别码\*/

n,

sum, sum2, /\*存放整数型单词\*/

m,p; /\*p是缓冲区prog的指针，m是token的指针\*/

double xiaoshu;

float k;

int flag=1;

int flag2=1;

int j;

int t=10;

int warn=0;

double g;

//char \*rwtab[6]={"begin","if","then","while","do","end"};

char \*gjz[9]= {"main","int","float","double","char","if","else","do","while"};

void scaner()

{

m=0;

sum=0;

xiaoshu=0;

k=0.1;

for(n=0; n<8; n++)

token[n]='\0';

ch=prog[p++];

while(ch==' ')

ch=prog[p++];

while(ch=='\n')

ch=prog[p++];

if(isalpha(ch)) /\*ch为字母字符\*/

{

while(isalpha(ch)||isdigit(ch)) /\*ch 为字母字符或者数字字符\*/

{

token[m++]=ch;

ch=prog[p++];

}

token[m++]='\0';

ch=prog[p--];

syn=10;

for(n=0; n<9; n++)

if(strcmp(token,gjz[n])==0) /\*字符串的比较\*/

{

syn=n+1;

break;

}

}

else if(isdigit(ch)) /\*ch是数字字符\*/

{

while(isdigit(ch)) /\*ch是数字字符\*/

{

//sum=sum\*10+ch-'0';

xiaoshu=xiaoshu\*10+ch-'0';

ch=prog[p++];

}

if(ch=='.')

{

ch=prog[p++];

if(!isdigit(ch)){

warn=1;

if(ch=='e')

warn=2;

}

while(isdigit(ch)) /\*ch是数字字符\*/

{

xiaoshu=xiaoshu+k\*(ch-'0');

k=k\*0.1;

ch=prog[p++];

}

}

if(ch=='e'&&warn!=2)

{

g=xiaoshu;

ch=prog[p++];

ch4=ch;

if(ch4>23&&ch4<38){

warn=1;

g=0;

xiaoshu=0;

}

if(ch=='+')

{

ch=prog[p++];

if(!isdigit(ch))

{

warn=1;

}

while(isdigit(ch))

{

sum2=sum2\*10+ch-'0';

ch=prog[p++];

}

for(; sum2>0; sum2--)

{

g=g\*10;

}

}

if(ch=='-')

{

k=0.1;

ch=prog[p++];

if(!isdigit(ch))

warn=1;

flag2=-1;

while(isdigit(ch))

{

sum2=sum2\*10+ch-'0';

ch=prog[p++];

}

for(; sum2>0; sum2--)

{

g=g\*k;

}

}

if(isdigit(ch))

{

while(isdigit(ch))

{

sum2=sum2\*10+ch-'0';

ch=prog[p++];

}

for(; sum2>0; sum2--)

{

g=g\*10;

}

}

}

ch=prog[p--];

syn=11;

}

else

switch(ch)

{

case'<':

m=0;

token[m++]=ch;

ch=prog[p++];

if(ch=='>')

{

syn=41;

token[m++]=ch;

}

else if(ch=='=')

{

syn=35;

token[m++]=ch;

}

else

{

syn=34;

ch=prog[p--];

}

break;

case'>':

m=0;

token[m++]=ch;

ch=prog[p++];

if(ch=='=')

{

syn=33;

token[m++]=ch;

}

else

{

syn=32;

ch=prog[p--];

}

break;

case':':

m=0;

token[m++]=ch;

ch=prog[p++];

if(ch=='=')

{

syn=18;

token[m++]=ch;

}

else

{

syn=17;

ch=prog[p--];

}

break;

case'!':

m=0;

token[m++]=ch;

ch=prog[p++];

if(ch=='=')

{

syn=37;

token[m++]=ch;

}

else

{

ch=prog[p--];

}

break;

case'=':

m=0;

token[m++]=ch;

ch=prog[p++];

if(ch=='=')

{

syn=36;

token[m++]=ch;

}

else

{

syn=21;

ch=prog[p--];

}

break;

case'+':

m=0;

ch2=ch;

j=p-2;

ch3=prog[j];

ch=prog[p++];

if((isdigit(ch))&&(!isdigit(ch3)))

{

if(isalpha(ch3)||ch3==')')

{

token[m++]=ch2;

syn=22;

ch=prog[p--];

}

else

{

syn=99;

ch=prog[p--];

}

}

else

{

token[m++]=ch2;

syn=22;

ch=prog[p--];

}

break;

case'-':

m=0;

ch2=ch;

j=p-2;

ch3=prog[j];

ch=prog[p++];

if((isdigit(ch))&&(!isdigit(ch3)))

{

if(isalpha(ch3)||ch3==')')

{

token[m++]=ch2;

syn=23;

ch=prog[p--];

}

else

{

syn=99;

ch=prog[p--];

flag=-1;

}

}

else

{

token[m++]=ch2;

syn=23;

ch=prog[p--];

}

break;

case'\*':

syn=24;

token[0]=ch;

break;

case'/':

syn=25;

token[0]=ch;

break;

case',':

syn=30;

token[0]=ch;

break;

case';':

syn=31;

token[0]=ch;

break;

case'(':

syn=26;

token[0]=ch;

break;

case')':

syn=27;

token[0]=ch;

break;

case'}':

syn=29;

token[0]=ch;

break;

case'{':

syn=28;

token[0]=ch;

break;

case'#':

syn=0;

token[0]=ch;

break;

default:

syn=-1;

}

}

main()

{

printf("\n\nThe significance of the figures:\n"

"1.figures 1 to 9 said Keyword\n"

"2.figures 10 and 11 said Other indicators\n"

"3.figures 21 to 37 said Operators\n");

p=0;

printf("\nplease input string:\n");

do

{

ch=getchar();

prog[p++]=ch;

}

while(ch!='#');

p=0;

do

{

scaner();

if(warn==0){

switch(syn)

{

case 11:

if(g==0)

{

xiaoshu=flag\*xiaoshu;

flag=1;

printf("(%d,%.3f)\n",syn,xiaoshu);

}

else

{

g=g\*flag;

flag=1;

printf("(%d,%g)\n",syn,g);

// printf("lalala\n");

g=0;

}

break;

case -1:

printf("\n ;\n");

break;

case 99:

break;

default:

printf("(%d,%s)\n",syn,token);

}

}

if(warn==1){

printf("error\n");

// printf("%.4f",xiaoshu);

warn=0;

xiaoshu=0;

g=0;

flag=1;

}

if(warn==2){

printf("error\n");

warn=0;

}

}

while(syn!=0);

getch();

}