实验一

一、实验名称： 词法分析 器的设计

二、实验目的： 1 ，词法分析器能够识别简单语言的单词符号

2 ，识别出并输出简单语言的基本字 . 标示符 . 无符号整数 . 运算符 . 和界符。

三 、实验要求： 给出一个简单语言单词符号的种别编码词法分析器

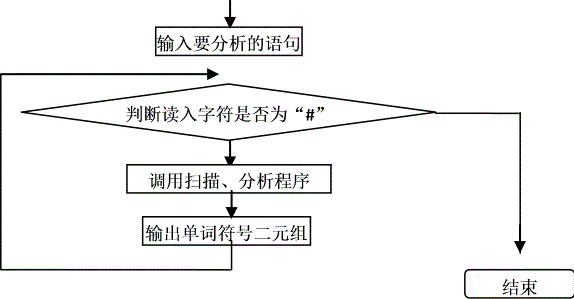
四、实验原理：

1 、 词法分析程序的算法思想

算法的基本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单词符号的第一个字符的种类，拼出相应的单词符号。

2 、程序流程图

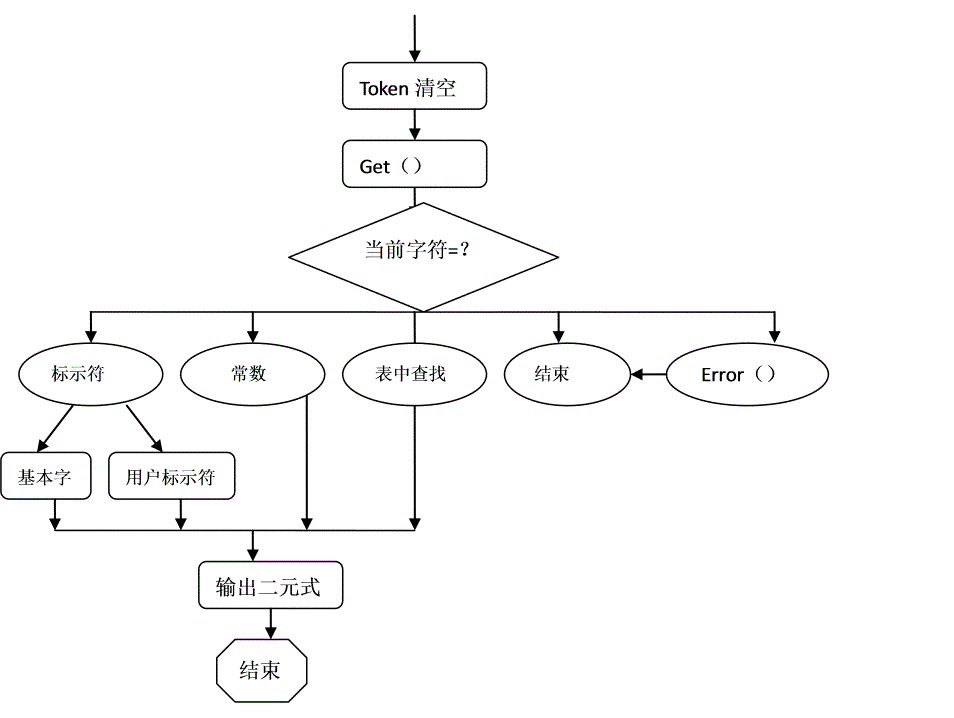
（ 1 ）主程序







（ 2 ）扫描子程序



3 、 各种单词符号对应的种别码

|  |  |  |  |
| --- | --- | --- | --- |
| 单词符号 | 种别码 | 助记符 | 内码值 |
| while | 1 | while | - |
| if | 2 | if | - |
| else | 3 | else | - |
| switch | 4 | switch | - |
| case | 5 | case | - |
| 标识符 | 6 | id | id 在符号表中的位置 |
| 常数 | 7 | num | num 在常数表中的位置 |
| + | 8 | + | - |
| - | 9 | - | - |
| \* | 1 0 | \* | - |
| <= | 11 | relop | LE |
| < | 11 | relop | LT |
| == | 11 | relop | EQ |
| = | 12 | = | - |
| ; | 13 | ; | - |
|  |  |  |  |

五 、 实验内容：

1 、 实验分析

编写程序 时， 先定义 几 个全局变量 a[] 、 token[] （ 均为字符串数组 ),c,s( char 型 ) ， i,j,k （ int 型） ， a[] 用来存放 输入的字符串， token[] 另一个则用来 帮助 识别单词符号 ， s 用来表示 正在分析的字符。字符串输入之后，逐个分析输入字符，判断其是否 ‘ # ’，若是 表 示 字符串输入 分析完毕，结束分析程序 ， 若否 则 通过 int digit(char c) 、 int letter(char c) 判断其是数字，字符还是算术符 , 分别为用以判断数字或字符的情况，算术符的判断可以在 switch 语句中进行，还要通过 函数 int lookup(char token[]) 来判断标识符和保留字。

2 实验词法分析器源程序：

#include <stdio.h>

#include <math.h>

#include <string.h>

int i,j,k;

char c,s,a[20],token[20]={'0'};

int letter(char s){

if((s>=97)&&(s<=122)) return(1);

else return(0);

}

int digit(char s){

if((s>=48)&&(s<=57)) return(1);

else return(0);

}

void get(){

s=a[i];

i=i+1;

}

void retract(){

i=i-1;

}

int lookup(char token[20]){

if(strcmp(token,"while")==0) return(1);

else if(strcmp(token,"if")==0) return(2);

else if(strcmp(token,"else")==0) return(3);

else if(strcmp(token,"switch")==0) return(4);

else if(strcmp(token,"case")==0) return(5);

else return(0);

}

void main()

{

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

i=0;

do{i=i+1;

scanf("%c",&a[i]);

}while(a[i]!='#');

i=1;

j=0;

get();

while(s!='#'){ memset(token,0,20);

switch(s)

{

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':

while(letter(s)||digit(s))

{token[j]=s;

j=j+1;

get();

}

retract();k=lookup(token);

if(k==0)

printf("(%d,%s)",6,token);

else printf("(%d,-)",k);

break;

case '0':

case '1':

case '2':

case '3':

case '4':

case '5':

case '6':

case '7':

case '8':

case '9':

while(digit(s)){

token[j]=s;

j=j+1;

get();

}

retract();

printf("%d,%s",7,token);

break;

case '+':printf("('+',NULL)");break;

case '-':printf("('-',null)");break;

case '\*':printf("('\*',null)");break;

case '<':get();

if(s=='=') printf("(relop,LE)");

else{retract();

printf("(relop,LT)");

}

break;

case '=':

get();

if(s=='=')

printf("(relop,EQ)");

else{

retract();

printf("('=',null)");

}

break;

case ';':

printf("(;,null)");

break;

case ' ':break;

default:printf("!\n");

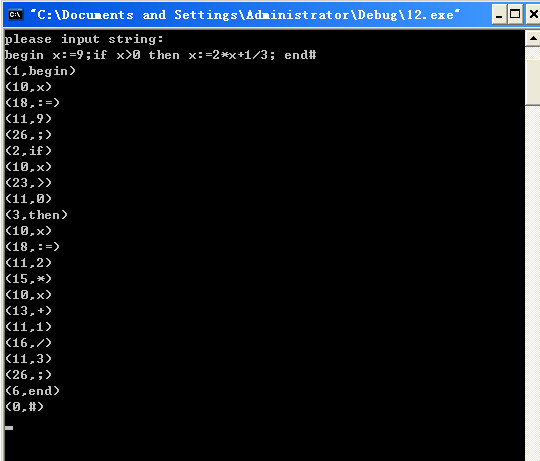
}

j=0;

get();

} }

六： 实验结果：



实验二

一、 实验名称：语法分析 器的设计

二、 实验目的：

用 C 语言编写对一个 算术 表达式实现语法分析的语法分析程序，并以四元式的形式输出，以 加深对语法语义分析原理的理解， 掌握语法分析程序的实现方法和技术。

三、 实验原理：

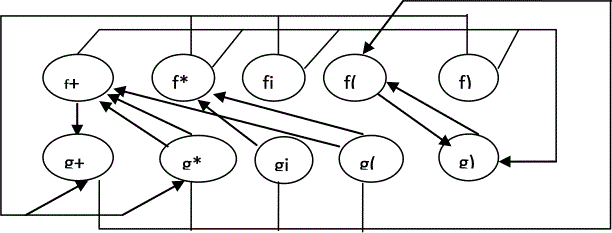
1 、算术 表达式 语法分析程序的算法思想

首先通过关系图法构造出终结符间的左右优先函数 f(a),g(a) 。在分析的过程中，通过左右优先函数比较当前读入终结符与前一个读入终结符间的优先关系，分析后适时的以四元式形式输出相关的符号。

2 、通过优先函数关系图构造优先函数

优先函数表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | + | \* | i | ( | ) |
| f | 4 | 6 | 6 | 2 | 6 |
| g | 3 | 5 | 7 | 7 | 2 |
|  |  |  |  |  |  |

优先函数关系图

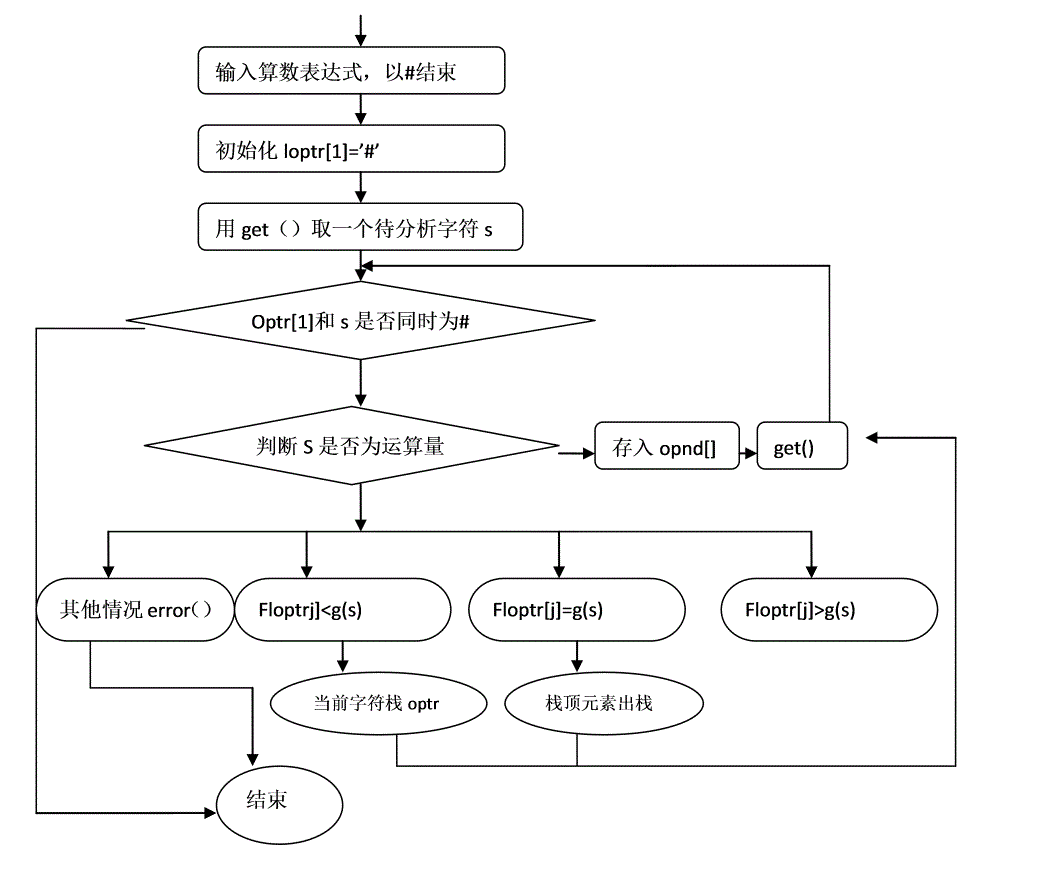
四 、 实验内容：

1 、 实验分析

本实验所用文法是： G[E]:E->E+E|E-E|E\*E|E/E|(E)

E->0|1|2|3| …… |9

根据此文法编写程序时，算术表达式的求解过程需要用到算术符号的优先判断，定义两个字符串数组 optr[] 、 opnd[] 用来模拟栈，存放算术符和操作数 , 用 a[] 来存放表达式字符串，在分析时还要通过函数 int f(char c) 和 int g(char c) ，判断运算符之间的优先关系，根据不同情况作各种不同操作。流程图如下：



2 ，实验程序源代码：

#include<stdio.h>

char a[21],optr[10],op,s;

int opnd[10],i,j,k,x1,x2,x3;

int operand(char s)

{if((s>=48)&&(s<=57)) return 1;

else return 0;

}

int f(char s)

{switch(s){

case '+':return 4;break;

case '\*':return 6;break;

case 'i':return 6;break;

case '(':return 2;break;

case ')':return 6;break;

default:return 0;}

}

int g(char s)

{switch(s){

case '+':return 3;break;

case '\*':return 5;break;

case 'i':return 7;break;

case '(':return 7;break;

case ')':return 2;break;

default:return 0;}

}

void get()

{s=a[i];

i++;}

void main()

{printf(" 请输入表达式（以‘ # ’结束）： \n");

i=0;

do{i++;

scanf("%c",&a[i]);

}while(a[i]!='#');

i=j=k=1;

optr[j]='#';

get();

while(!((optr[j]=='#')&&(s=='#')))

{if(operand(s)) {opnd[k]=s-48;k++;get();}

else if (f(optr[j])<g(s)) {j++;optr[j]=s;get();}

else if (f(optr[j])>g(s))

{op=optr[j];

j--;

x1=opnd[k-1];

x2=opnd[k-2];

k=k-2;

switch(op)

{case '+':x3=x1+x2;break;

case '\*':x3=x1\*x2;break;

default:break;

}

opnd[k]=x3;

k++;

printf("(%c,%d,%d,%d)\n",op,x2,x1,x3);

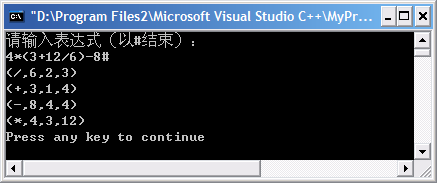
}

else {j--;get();}

}

}

3 实验结果：



心得体会：

通过 本次 实验， 我更加深刻的学习到了很多 ：

1 、通过实验我对词法分析和语法分析 原理有 了更深刻 的理解。

2 、而且 对词法分析 和语法分析 在实践中的应用有 了 深 入的掌握 。

3 、 更加熟悉了构造 词法分析程序和 语法分析程序的手工方式的相关原理，能够实现对词法分析程序所提供的单词符号序列进行相应的语法检查和结构分析，达到了学以致用的目的 。