实验四 递归下降分析分析法

【实验目的】

（1）掌握自上而下语法分析的要求与特点。

（2）掌握递归下降语法分析的基本原理和方法。

（3）掌握相应数据结构的设计方法。

【实验内容】

用递归下降法编写一个语法分析程序，使之与词法分析器结合，能够根据语言的上下文无关文法，识别输入的单词序列是否文法的句子。

【实验要求】

对下列文法，用递归下降分析法对任意输入的符号串进行分析：

E->TG G->+TG|-TG G->ε

T->FS S->\*FS|/FS S->ε

F->(E) F->i

【实验结果】

**一：源程序**

#include <stdio.h>

#include<dos.h>

#include<stdlib.h>

#include<string.h>

char a[50], b[50], d[200], e[10];

char ch;

int n1, i1 = 0, flag = 1, n = 5;

/\*规定的文法

E→TG

G→＋TG|-TG

G→ε

T→FS

S→\*FS|/FS

S→ε

F→(E)

F→i

\*/

int E();

int E1();

int T();

int G();

int S();

int F();

void input();

void input1();

void output();

void main()

{

int f, p, j = 0;

char x;

d[0] = 'E';

d[1] = '=';

d[2] = '>';

d[3] = 'T';

d[4] = 'G';

d[5] = '#';

printf("请输入字符串(长度<50,以#号结束）\n");

do{

scanf("%c", &ch);

a[j] = ch;

j++;

} while (ch != '#');

n1 = j;

ch = b[0] = a[0];

printf("文法\t分析串\t\t分析字符\t剩余串\n");

f = E1();

if (f == 0) return;

if (ch == '#')

{

printf("accept\n");

p = 0;

x = d[p];

while (x != '#')

{

printf("%c", x); p = p + 1; x = d[p]; /\*输出推导式\*/

}

}

else

{

printf("error\n");

printf("回车返回\n");

getchar(); getchar();

return;

}

printf("\n");

printf("回车返回\n");

getchar();

getchar();

}

//================================================

int E1()

{

int f, t;

printf("E->TG\t");

flag = 1;

input();

input1();

f = T();

if (f == 0) return(0);

t = G();

if (t == 0) return(0);

else return(1);

}

//================================================

int E()

{

int f, t;

printf("E->TG\t");

e[0] = 'E'; e[1] = '='; e[2] = '>'; e[3] = 'T'; e[4] = 'G'; e[5] = '#';

output();

flag = 1;

input();

input1();

f = T();

if (f == 0) return(0);

t = G();

if (t == 0) return(0);

else return(1);

}

//================================================

int T()

{

int f, t;

printf("T->FS\t");

e[0] = 'T'; e[1] = '='; e[2] = '>'; e[3] = 'F'; e[4] = 'S'; e[5] = '#';

output();

flag = 1;

input();

input1();

f = F();

if (f == 0) return(0);

t = S();

if (t == 0) return(0);

else return(1);

}

//================================================

int G()

{

int f;

if (ch == '+')

{

b[i1] = ch;

printf("G->+TG\t");

e[0] = 'G'; e[1] = '='; e[2] = '>'; e[3] = '+'; e[4] = 'T'; e[5] = 'G'; e[6] = '#';

output();

flag = 0;

input(); input1();

ch = a[++i1];

f = T();

if (f == 0) return(0);

G();

return(1);

}

if (ch == '-')

{

b[i1] = ch;

printf("G->-TG\t");

e[0] = 'G';

e[1] = '=';

e[2] = ' ';

e[3] = '-';

e[4] = 'T';

e[5] = 'G';

e[6] = '#';

output();

flag = 0;

input(); input1();

ch = a[++i1];

f = T();

if (f == 0) return(0);

G();

return(1);

}

printf("G--^\t");

e[0] = 'G'; e[1] = '='; e[2] = ' '; e[3] = '^'; e[4] = '#';

output();

flag = 1;

input(); input1();

return(1);

}

//================================================

int S()

{

int f, t;

if (ch == '\*') {

b[i1] = ch; printf("S->\*FS\t");

e[0] = 'S'; e[1] = '='; e[2] = ' '; e[3] = '\*'; e[4] = 'F'; e[5] = 'S'; e[6] = '#';

output();

flag = 0;

input();

input1();

ch = a[++i1];

f = F();

if (f == 0) return(0);

t = S();

if (t == 0) return(0);

else return(1);

}

if (ch == '/') {

b[i1] = ch; printf("S->/FS\t");

e[0] = 'S';

e[1] = '=';

e[2] = ' ';

e[3] = '/';

e[4] = 'F';

e[5] = 'S';

e[6] = '#';

output();

flag = 0;

input(); input1();

ch = a[++i1];

f = F();

if (f == 0) return(0);

t = S();

if (t == 0) return(0);

else return(1);

}

printf("S->^\t");

e[0] = 'S'; e[1] = '='; e[2] = ' '; e[3] = '^'; e[4] = '#';

output();

flag = 1;

a[i1] = ch;

input(); input1();

return(1);

}

//================================================

int F()

{

int f;

if (ch == '(') {

b[i1] = ch; printf("F->(E)\t");

e[0] = 'F'; e[1] = '='; e[2] = ' '; e[3] = '('; e[4] = 'E'; e[5] = ')'; e[6] = '#';

output();

flag = 0;

input(); input1();

ch = a[++i1];

f = E();

if (f == 0) return(0);

if (ch == ')') {

b[i1] = ch; printf("F--(E)\t");

flag = 0; input(); input1();

ch = a[++i1];

}

else {

printf("error\n");

return(0);

}

}

else if (ch == 'i') {

b[i1] = ch; printf("F->i\t");

e[0] = 'F'; e[1] = '='; e[2] = ' '; e[3] = 'i'; e[4] = '#';

output();

flag = 0; input(); input1();

ch = a[++i1];

}

else { printf("error\n"); return(0); }

return(1);

}

//================================================

void input()

{

int j = 0;

for (; j <= i1 - flag; j++)

printf("%c", b[j]); /\*输出分析串\*/

printf("\t\t");

printf("%c\t\t", ch); /\*输出分析字符\*/

}

//================================================

void input1()

{

int j;

for (j = i1 + 1 - flag; j<n1; j++)

printf("%c", a[j]); /\*输出剩余字符\*/

printf("\n");

}

//================================================

void output(){ /\*推导式计算\*/

int m, k, j, q;

int i = 0;

m = 0; k = 0; q = 0;

i = n;

d[n] = '='; d[n + 1] = '>'; d[n + 2] = '#'; n = n + 2; i = n;

i = i - 2;

while (d[i] != '>'&&i != 0)

i = i - 1;

i = i + 1;

while (d[i] != e[0])

i = i + 1;

q = i;

m = q; k = q;

while (d[m] != '>')

m = m - 1;

m = m + 1;

while (m != q)

{

d[n] = d[m]; m = m + 1; n = n + 1;

}

d[n] = '#';

for (j = 3; e[j] != '#'; j++)

{

d[n] = e[j];

n = n + 1;

}

k = k + 1;

while (d[k] != '=')

{

d[n] = d[k]; n = n + 1; k = k + 1;

}

d[n] = '#';

}

**二：程序结果截图**

