## 实验二 LL(1)分析法

**一、实验目的**

根据某一文法编制调试LL（1）分析程序，以便对任意输入的符号串进行分析。本次实验的目的主要是加深对预测分析LL（1）分析法的理解。

**二、实验预习提示**

1、LL（1）分析法的功能

LL（1）分析法的功能是利用LL（1）控制程序根据显示栈栈顶内容、向前看符号以及LL（1）分析表，对输入符号串自上而下的分析过程。

2、LL（1）分析法的前提

改造文法：消除二义性、消除左递归、提取左因子，判断是否为LL（1）文法，

3、LL（1）分析法实验设计思想及算法

X∈VN

‘#’‘S’进栈，当前输入符送a

上托栈顶符号放入X

若产生式为

X→ X1X2…Xn

按逆序即Xn…X2X1入栈

出错

X=’#’

X∈VT

X=a

M[X,a]是产生式吗

出错

X=a

读入下一个符号

结束

是

是

是

是

否

否

否

否

否

是

**三、实验过程和指导：**

（一）准备：

1.阅读课本有关章节，

2.考虑好设计方案；

3.设计出模块结构、测试数据，初步编制好程序。

（二）上课上机：

将源代码拷贝到机上调试，发现错误，再修改完善。第二次上机调试通过。

（三）程序要求：

程序输入/输出示例：

对下列文法，用LL（1）分析法对任意输入的符号串进行分析：

（1）E->TG

（2）G->+TG|—TG

（3）G->ε

（4）T->FS

（5）S->\*FS|/FS

（6）S->ε

（7）F->(E)

（8）F->i

输出的格式如下：

(1)LL（1）分析程序，编制人：姓名，学号，班级

(2)输入一以#结束的符号串(包括+—\*/（）i#)：在此位置输入符号串

(3)输出过程如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **步骤** | **分析栈** | **剩余输入串** | **所用产生式** |
| 1 | E | i+i\*i# | E->TG |

(4)输入符号串为非法符号串(或者为合法符号串)

备注：(1)在“**所用产生式**”一列中如果对应有推导则写出所用产生式；如果为匹配终结符则写明匹配的终结符；如分析异常出错则写为“分析出错”；若成功结束则写为“分析成功”。

(2) 在此位置输入符号串为用户自行输入的符号串。

(3)上述描述的输出过程只是其中一部分的。

注意：1.表达式中允许使用运算符（+-\*/）、分割符（括号）、字符i，结束符#；

2.如果遇到错误的表达式，应输出错误提示信息（该信息越详细越好）；

3.对学有余力的同学，测试用的表达式事先放在文本文件中，一行存放一个表达式，同时以分号分割。同时将预期的输出结果写在另一个文本文件中，以便和输出进行对照；

（四）程序思路（仅供参考）：

模块结构：

（1）定义部分：定义常量、变量、数据结构。

（2）初始化：设立LL(1)分析表、初始化变量空间（包括堆栈、结构体、数组、临时变量等）；

（3）控制部分：从键盘输入一个表达式符号串；

（4）利用LL(1)分析算法进行表达式处理：根据LL(1)分析表对表达式符号串进行堆栈（或其他）操作，输出分析结果，如果遇到错误则显示错误信息。

（五）练习该实验的目的和思路：

程序相当复杂，需要利用到大量的编译原理，也用到了大量编程技巧和数据结构，通过这个练习可大大提高软件开发能力。

（六）为了能设计好程序，注意以下事情：

1.模块设计：将程序分成合理的多个模块（函数），每个模块做具体的同一事情。

2.写出（画出）设计方案：模块关系简图、流程图、全局变量、函数接口等。

3.编程时注意编程风格：空行的使用、注释的使用、缩进的使用等。

**四、上交：**

1.程序源代码（源程序文件c/c++）；

2.已经测试通过的测试数据3组；

3.实验报告：

（一）实验内容

（1）功能描述：该程序具有什么功能？

（2）程序结构描述：函数调用格式、参数含义、返回值描述、函数功能；函数之间的调用关系图。

（3）程序总体执行流程图

（二）实验过程记录：出错次数、出错严重程度、解决办法摘要。

（三）实验总结：你在编程过程中花时多少？多少时间在纸上设计？多少时间上机输入和调试？多少时间在思考问题？遇到了哪些难题？你是怎么克服的？你对你的程序的评价？你的收获有哪些？

**五、参考源代码**

/\*LL(1)分析法源程序，已在VC++中运行\*/

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<dos.h>

char A[20];/\*分析栈\*/

char B[20];/\*剩余串\*/

char v1[20]={'i','+','\*','(',')','#'};/\*终结符 \*/

char v2[20]={'E','G','T','S','F'};/\*非终结符 \*/

int j=0,b=0,top=0,l;/\*L为输入串长度 \*/

typedef struct type/\*产生式类型定义 \*/

{

char origin;/\*大写字符 \*/

char array[5];/\*产生式右边字符 \*/

int length;/\*字符个数 \*/

}type;

type e,t,g,g1,s,s1,f,f1;/\*结构体变量 \*/

type C[10][10];/\*预测分析表 \*/

void print()/\*输出分析栈 \*/

{

int a;/\*指针\*/

for(a=0;a<=top+1;a++)

printf("%c",A[a]);

printf("\t\t");

}/\*print\*/

void print1()/\*输出剩余串\*/

{

int j;

for(j=0;j<b;j++)/\*输出对齐符\*/

printf(" ");

for(j=b;j<=l;j++)

printf("%c",B[j]);

printf("\t\t\t");

}/\*print1\*/

void main()

{

int m,n,k=0,flag=0,finish=0;

char ch,x;

type cha;/\*用来接受C[m][n]\*/

/\*把文法产生式赋值结构体\*/

e.origin='E';

strcpy(e.array,"TG");

e.length=2;

t.origin='T';

strcpy(t.array,"FS");

t.length=2;

g.origin='G';

strcpy(g.array,"+TG");

g.length=3;

g1.origin='G';

g1.array[0]='^';

g1.length=1;

s.origin='S';

strcpy(s.array,"\*FS");

s.length=3;

s1.origin='S';

s1.array[0]='^';

s1.length=1;

f.origin='F';

strcpy(f.array,"(E)");

f.length=3;

f1.origin='F';

f1.array[0]='i';

f1.length=1;

for(m=0;m<=4;m++)/\*初始化分析表\*/

for(n=0;n<=5;n++)

C[m][n].origin='N';/\*全部赋为空\*/

/\*填充分析表\*/

C[0][0]=e;C[0][3]=e;

C[1][1]=g;C[1][4]=g1;C[1][5]=g1;

C[2][0]=t;C[2][3]=t;

C[3][1]=s1;C[3][2]=s;C[3][4]=C[3][5]=s1;

C[4][0]=f1;C[4][3]=f;

printf("提示:本程序只能对由'i','+','\*','(',')'构成的以'#'结束的字符串进行分析,\n");

printf("请输入要分析的字符串:");

do/\*读入分析串\*/

{

scanf("%c",&ch);

if ((ch!='i') &&(ch!='+') &&(ch!='\*')&&(ch!='(')&&(ch!=')')&&(ch!='#'))

{

printf("输入串中有非法字符\n");

exit(1);

}

B[j]=ch;

j++;

}while(ch!='#');

l=j;/\*分析串长度\*/

ch=B[0];/\*当前分析字符\*/

A[top]='#'; A[++top]='E';/\*'#','E'进栈\*/

printf("步骤\t\t分析栈 \t\t剩余字符 \t\t所用产生式 \n");

do

{

x=A[top--];/\*x为当前栈顶字符\*/

printf("%d",k++);

printf("\t\t");

for(j=0;j<=5;j++)/\*判断是否为终结符\*/

if(x==v1[j])

{

flag=1;

break;

}

if(flag==1)/\*如果是终结符\*/

{

if(x=='#')

{

finish=1;/\*结束标记\*/

printf("acc!\n");/\*接受 \*/

getchar();

getchar();

exit(1);

}/\*if\*/

if(x==ch)

{

print();

print1();

printf("%c匹配\n",ch);

ch=B[++b];/\*下一个输入字符\*/

flag=0;/\*恢复标记\*/

}/\*if\*/

else/\*出错处理\*/

{

print();

print1();

printf("%c出错\n",ch);/\*输出出错终结符\*/

exit(1);

}/\*else\*/

}/\*if\*/

else/\*非终结符处理\*/

{

for(j=0;j<=4;j++)

if(x==v2[j])

{

m=j;/\*行号\*/

break;

}

for(j=0;j<=5;j++)

if(ch==v1[j])

{

n=j;/\*列号\*/

break;

}

cha=C[m][n];

if(cha.origin!='N')/\*判断是否为空\*/

{

print();

print1();

printf("%c->",cha.origin);/\*输出产生式\*/

for(j=0;j<cha.length;j++)

printf("%c",cha.array[j]);

printf("\n");

for(j=(cha.length-1);j>=0;j--)/\*产生式逆序入栈\*/

A[++top]=cha.array[j];

if(A[top]=='^')/\*为空则不进栈\*/

top--;

}/\*if\*/

else/\*出错处理\*/

{

print();

print1();

printf("%c出错\n",x);/\*输出出错非终结符\*/

exit(1);

}/\*else\*/

}/\*else\*/

}while(finish==0);

}/\*main\*/