实验编号：3**四川师大 编译原理 实验报告 2019**年**4**月**12**日

**计算机科学学院** 2017级 4 班 实验名称： LR1分析法

姓名：\_\_韩勇\_\_ 学号：\_\_2017110409\_ 指导老师：\_\_吴贞东\_\_ 实验成绩:\_\_\_\_\_

**实验 三 LR1分析法\_**

1. 实验目的及要求

构造LR(1)分析程序，利用它进行语法分析，判断给出的符号串是否为该文 法识别的句子，了解LR（K）分析方法是严格的从左向右扫描，和自底向上的语 法分析方法。

1、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。

2、如果遇到错误的表达式，应输出错误提示信息。

3、程序输入/输出实例： 输入一以#结束的符号串(包括+\*（）i#)：在此位置输入符号串 输出过程如下： 步骤 状态栈 符号栈 剩余输入串 动作 1 0 # i+i\*i# 移进 i+i\*i的LR 分析过程 步骤 状态栈 符号栈 输入串 动作说明 1 0 # i+i\*i# ACTION[0,i]=S5,状态5 入栈 2 05 #i +i\*i# r6: F→i归约,GOTO(0,F)=3 入栈 3 03 #F +i\*i# r4: T→F 归约,GOTO(0,T)=3 入栈 4 02 #T +i\*i# r2: E→T 归约,GOTO(0,E)=1 入栈 19 5 01 #E +i\*i# ACTION[1,+]=S6,状态6 入栈 6 016 #E+ i\*i# ACTION[6,i]=S5,状态5 入栈 7 0165 #E+i \*i# r6: F→i归约,GOTO(6,F)=3 入栈 8 0163 #E+F \*i# r4: T→F 归约,GOTO(6,T)=9 入栈 9 0169 #E+T \*i# ACTION[9,\*]=S7,状态7 入栈 10 01697 #E+T\* i# ACTION[7,i]=S5,状态5 入栈 11 016975 #E+T\*i # r6:F→i归约,GOTO(7,F)=10 入栈 12 0169710 #E+T\*F # r3: T→T\*F 归约,GOTO(6,T)=9 入栈 13 0169 #E+T # r1:E→E+T,GOTO(0,E)=1 入栈 14 01 #E # Acc：分析成功 4、输入符号串为非法符号串(或者为合法符号串) 算术表达式文法的LR 分析表 状 态 ACTION GOTO i + \* ( ) # E T F 0 S5 S4 1 2 3 1 S6 acc 2 r2 S7 r2 r2 3 r4 r4 r4 r4 4 S5 S4 8 2 3 5 r6 r6 r6 r6 6 S5 S4 9 3 7 S5 S4 10 8 S6 S11 9 r1 S7 r1 r1 10 r3 r3 r3 r3 11 r5 r5 r5 r5

1. 实验内容

对下列文法，用LR（1）分析法对任意输入的符号串进行分析：

（1）E-> E+T

（2）E->T

（3）T-> T\*F

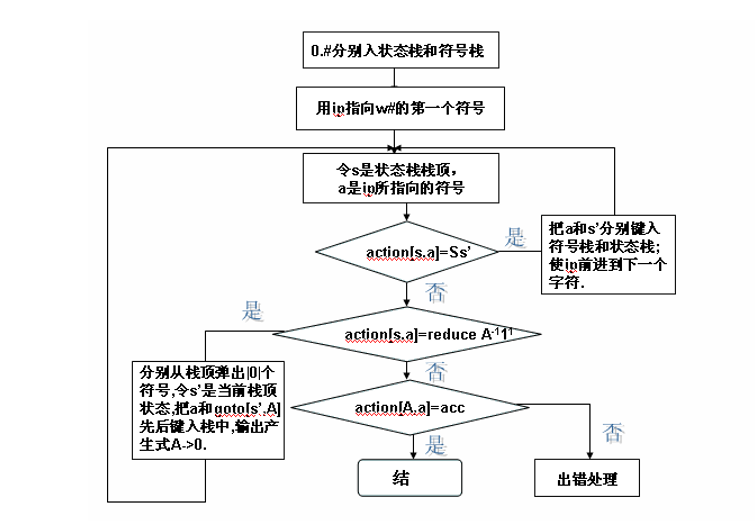
（4）T->F

（5）F-> (E)

（6）F->i

1. 实验主要流程、基本操作或核心代码、算法片段（该部分如不够填写，请另加附页）

流程图：



源程序：

#include<stdio.h>

#include<string.h>

char\*action[10][3]= {"S3#","S4#",NULL, /\*ACTION\*/

NULL,NULL,"acc",

"S6#","S7#",NULL,

"S3#","S4#",NULL,

"r3#","r3#",NULL,

NULL,NULL,"r1#",

"S6#","S7#",NULL,

NULL,NULL,"r3#",

"r2#","r2#",NULL,

NULL,NULL,"r2#"

};

int goto1[10][2]= {1,2, /\*QOTO\*/

0,0,0,5,0,8,0,0,0,0,0,9,0,0,0,0,

0,0

};

char vt[3]= {'a','b','#'}; /\*存放非终结符\*/

char vn[2]= {'S','B'}; /\*存放终结符\*/

char \*LR[4]= {"E->S#","S->BB#","B->aB#","B->b#"}; /\*存放产生式\*/

int a[10];

char b[10],c[10],c1;

int top1,top2,top3,top,m,n;

int main() {

int g,h,i,j,k,l,p,y,z,count;

char x,copy[10],copy1[10];

top1=0;

top2=0;

top3=0;

top=0;

a[0]=0;

y=a[0];

b[0]='#';

count=0;

z=0;

printf("----------------请输入表达式(以#结尾)--------------\n");

do {

scanf("%c",&c1);

c[top3]=c1;

top3=top3+1;

} while(c1!='#');

printf("步骤\t状态栈\t\t符号栈\t\t输入串\t\tACTION\tGOTO\n");

do {

y=z;

m=0;

n=0; /\*y,z指向状态栈栈顶\*/

g=top;

j=0;

k=0;

x=c[top];

count++;

printf("%d\t",count);

while(m<=top1) { /\*输出状态栈\*/

printf("%d",a[m]);

m=m+1;

}

printf("\t\t");

while(n<=top2) { /\*输出符号栈\*/

printf("%c",b[n]);

n=n+1;

}

printf("\t\t");

while(g<=top3) { /\*输出输入串\*/

printf("%c",c[g]);

g=g+1;

}

printf("\t\t");

while(x!=vt[j]&&j<=2)j++;

if(j==2&&x!=vt[j]) {

printf("error\n");

return 0;

}

if(action[y][j]==NULL) {

printf("error\n");

return 0;

} else

strcpy(copy,action[y][j]);

if(copy[0]=='S') {

/\*处理移进\*/

z=copy[1]-'0';

top1=top1+1;

top2=top2+1;

a[top1]=z;

b[top2]=x;

top=top+1;

i=0;

while(copy[i]!='#') {

printf("%c",copy[i]);

i++;

}

printf("\n");

}

if(copy[0]=='r') {

/\*处理归约\*/

i=0;

while(copy[i]!='#') {

printf("%c",copy[i]);

i++;

}

h=copy[1]-'0';

strcpy(copy1,LR[h]);

while(copy1[0]!=vn[k])k++;

l=strlen(LR[h])-4;

top1=top1-l+1;

top2=top2-l+1;

y=a[top1-1];

p=goto1[y][k];

a[top1]=p;

b[top2]=copy1[0];

z=p;

printf("\t");

printf("%d\n",p);

}

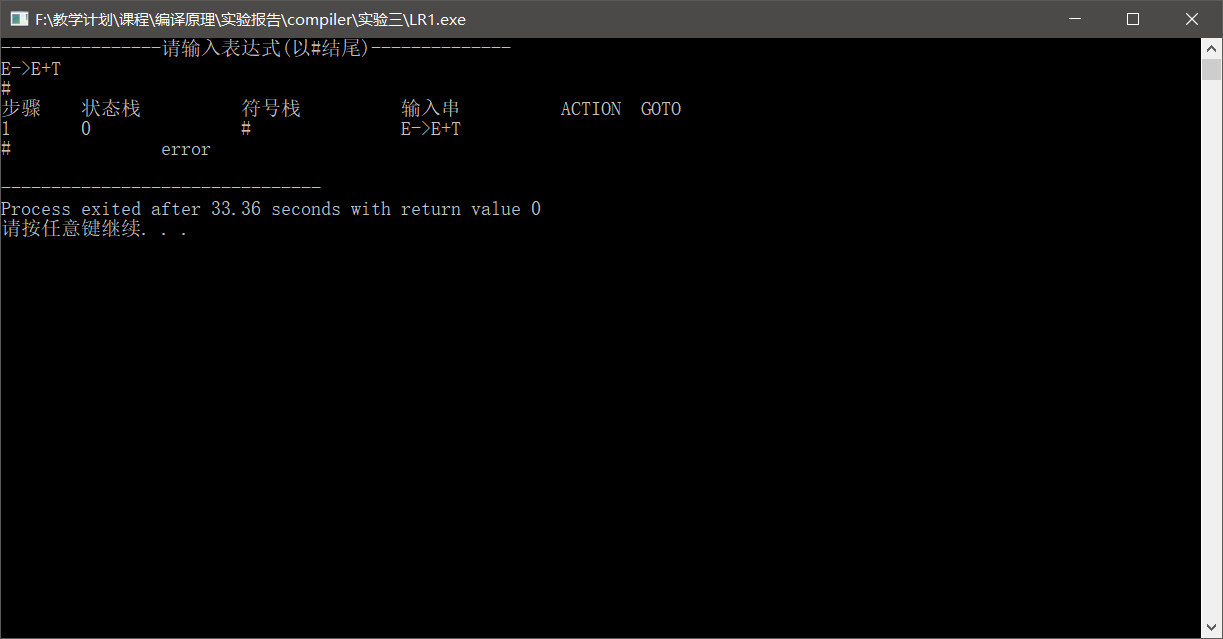
} while(action[y][j]!="acc");

printf("acc\n");

getchar();

}

运行结果截图：



1. 实验结果的分析与评价（该部分如不够填写，请另加附页）

本实验的难度相比于上一个的难度要小一些，关键在于弄清楚识别活前缀的DFA的构造过程，以及存储各类结果的数据结构即可。

注：实验成绩等级分为（90－100分）优，（80－89分）良，(70-79分)中，（60－69分）及格，（59分）不及格。