

编译原理

算符优先语法分析设计原理与实现

学 院：

专 业：

学生姓名：

学 号：

北京交通大学

2023年10月

**实验要求：**

[**实验项目**]

实现算符优先分析算法，完成以下描述算术表达式的算符优先文法的算符优先分析过程。

G[E]:E→E+T∣E-T∣T

T→T\*F∣T/F∣F

F→(E)∣i

**[设计说明]**

终结符号 i 为用户定义的简单变量,即标识符的定义。

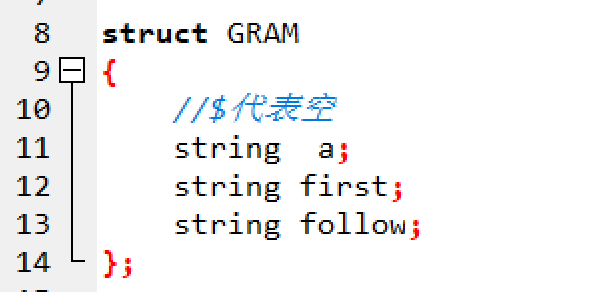
[**设计要求**]

1. 构造该算符优先文法的优先关系矩阵或优先函数；
2. 输入串应是词法分析的输出二元式序列，即某算术表达式“专题 1”的输出结果。输出为输入串是否为该文

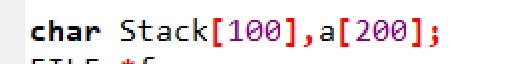
法定义的算术表达式的判断结果。

1. 算符优先分析过程应能发现输入串出错。
2. 设计两个测试用例（尽可能完备，正确和出错），并给出测试结果；
3. 考虑编写程序根据算符优先文法构造算符优先关系矩阵，并添加到你的算符优先分析程序中。
4. **程序功能描述**
5. 手动输入文法语句，将二元式序列文件读入程序
6. 依据文法求出的相应FirstVT和 LastVT 集，并生成算符优先分析表
7. 进行算符优先分析，判断读入字符串是否是文法的句子
8. **主要数据结构**

GRAM表示非终结符号的结构体，其中包含其first和follow集

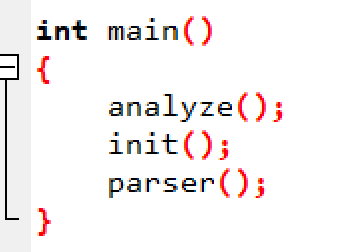


进行移进归约分析时，使用Stack和a数组。



1. **程序结构描述**

主函数包含三个子函数，analyze()作用是分析输入文法，依据文法求出的相应FirstVT和 LastVT 集，并生成算符优先分析表；init()的作用是读入字符串；parser的作用是依据生成的算符优先分析表进行算符优先分析，判断输入串是否为文法的合法语句，如果出错则进行相应提示。



Norterminal()的作用是判断终结符和非终结符 返回1是非终结符 返回0是终结符。

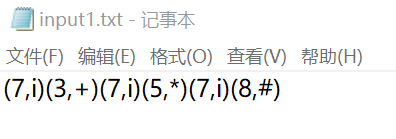
IsReduce()的作用是寻找与最左素短语匹配的产生式

getVtNumber()的作用是获取终结符号ch的序号

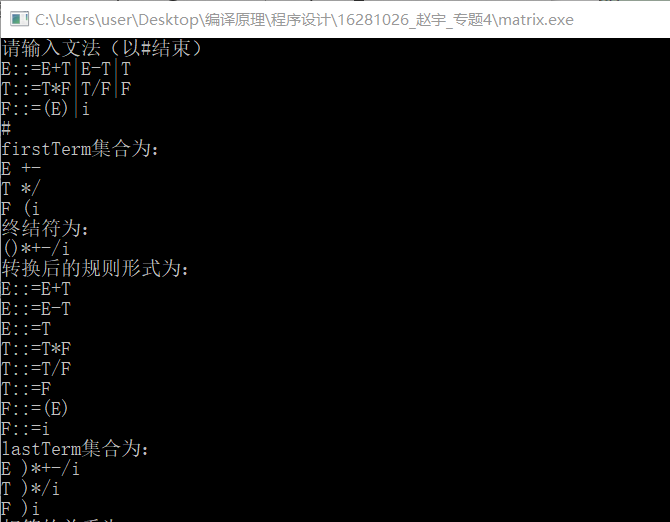
Higher、Lower、Equal均是对算符优先关系判断的函数

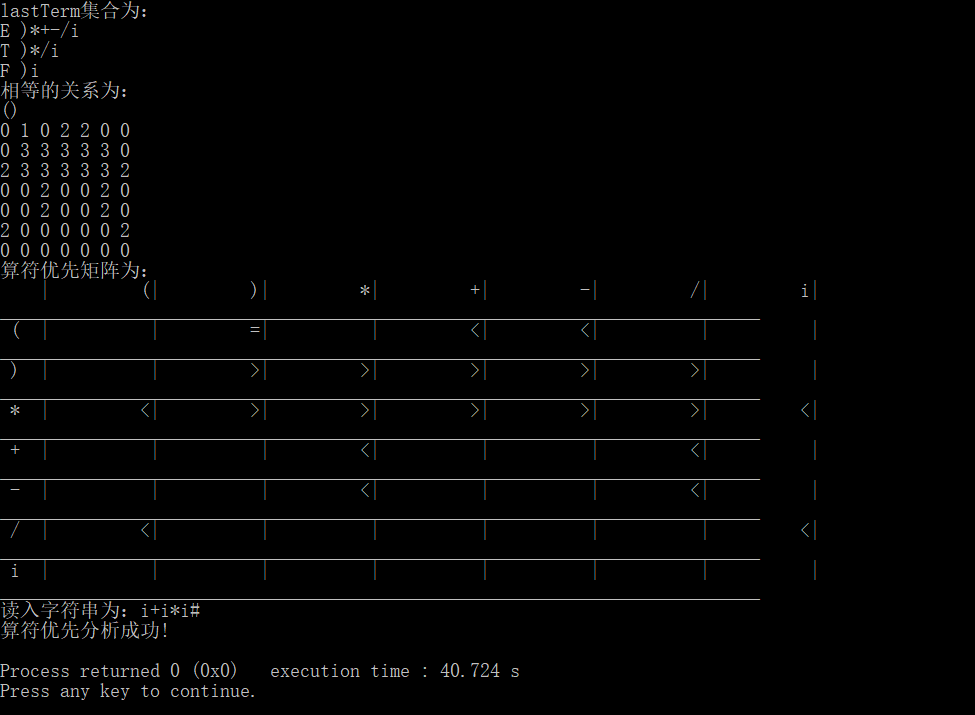
1. **实验结果及测试**

测试样例i+i\*i

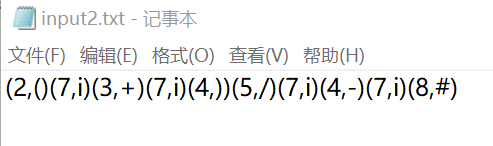


输入文法后，则打印优先关系矩阵，然后对读入串进行分析，得到结果是该输入串是算符优先分析的文法，见下图：

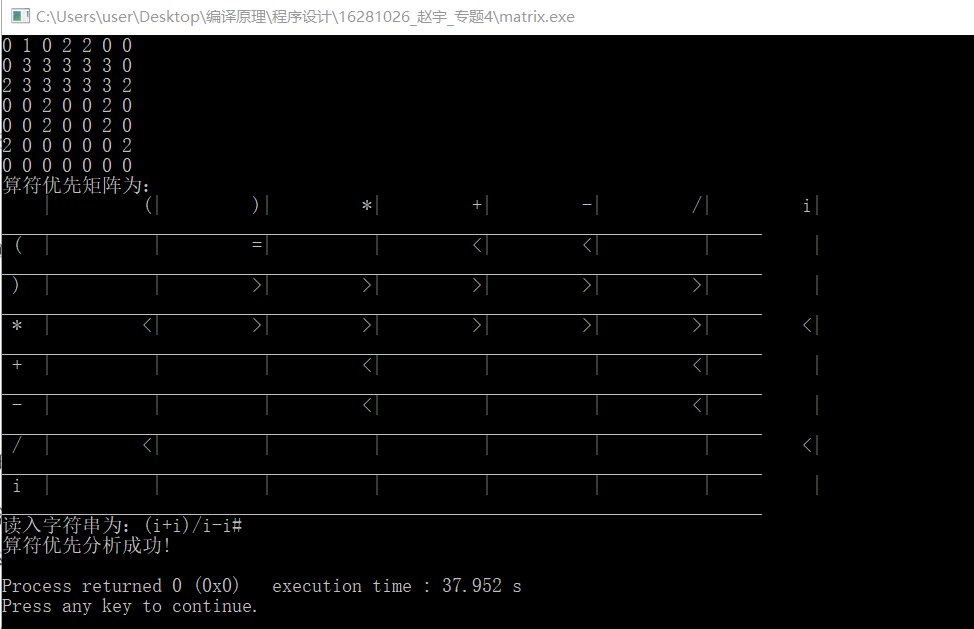




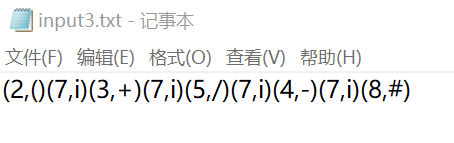
2、测试样例(i+i)/i-i



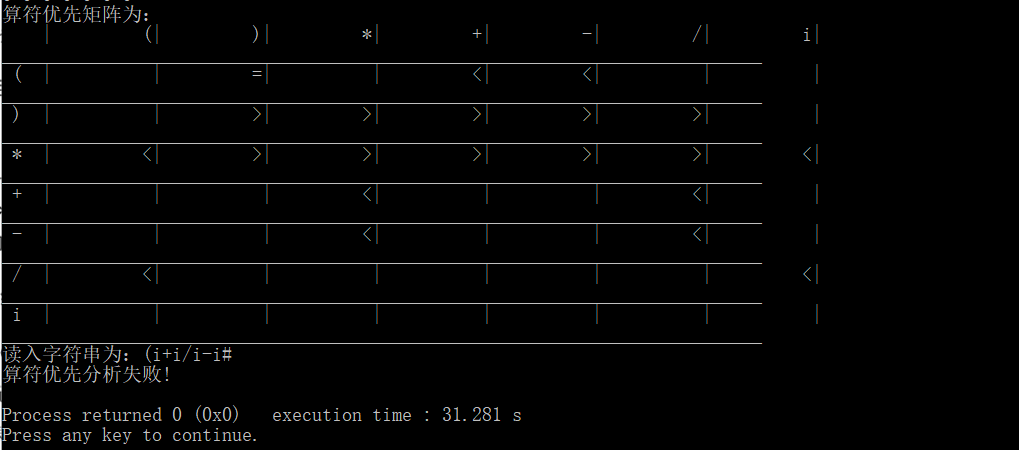
算符优先分析成功：



3、测试样例(i+i/i-i



该输入串不是文法的句子，所以分析失败



1. **实验结果分析**

由测试结果可见，我编写的算符优先语法分析程序可以正确的输出相应FirstVT和 LastVT 集，并且可以正确判断读入字符串是否是文法的句子。而且，我加入了错误判断，并且函数实现是模块化的，便于之后的扩展。

1. **研究性教学实验感受**

刚开始写实验的时候有些无从下手，后面通过看书和查阅资料逐步了解需要完成任务和实现的方法，然后开始逐步实现应有的功能函数。通过这次实验让我对算符优先文法的理解更深了一步，这也是做实验和学习这门课程的目的所在。一步步完善实验的过程也是在增进自己的知识。

1. **附件（源代码列表）**

#include<bits/stdc++.h>

# pragma warning (disable:4819)

using namespace std;

char Stack[100],a[200];

FILE \*fp;

struct GRAM

{

//$代表空

string a;

string first;

string follow;

};

//判断终结符和非终结符 返回1是非终结符 返回0是终结符

int Norterminal(char c)

{

if(c>='A'&&c<='Z')

return 1;

else if(c!=':'&&c!='='&&c!='<'&&c!='>'&&c!=' '&&c!='|')

return 0;

}

void analyze()

{

GRAM gram[50];

string grammer;

cout<<"请输入文法（以#结束）"<<endl;

std::vector<string> strN;

cin>>grammer;//输入规则

strN.push\_back(grammer);

/\*std::vector<string> strN;

int i=0,j=0;

if((fp=fopen("wenfa.txt","r"))==NULL)

{

cout<<"文件打开失败！"<<endl;

exit(0);

}

grammer[i++]=fgetc(fp);

/\*while(!feof(fp))

{

a1[i++]=fgetc(fp);

if((i%5)==0)

a[j++]=a1[i-2];

}

fclose(fp);\*/

char str[10][80];

const char \*ch = "|";

char \*result;

vector<string> strN1; //存处理过“|”规则的

for(int i=0;i<grammer.length();i++)

{

str[0][i]=grammer[i];

}

for(int h=grammer.length();h<80;h++)

{

str[0][h]=NULL;

}

result = strtok(str[0],ch);

while(result!=NULL)

{

strN1.push\_back(result);

result = strtok(NULL,ch);

}

for(int i=1;grammer!="#";i++)

{

cin>>grammer;

strN.push\_back(grammer);

/\*处理转换的规则形式输入部分\*/

for(int h=0;h<grammer.length();h++)

{

str[i][h]=grammer[h];

}

for(int h=grammer.length();h<80;h++)

{

str[i][h]=NULL;

}

result = strtok(str[i],ch);

while(result!=NULL)

{

strN1.push\_back(result);

result = strtok(NULL,ch);

}

}

/\*

获取firstTerm集合

\*/

for(unsigned i=0;i<strN.size();i++)

{

gram[i].a=strN[i][0];

gram[i].first="";

if(!Norterminal(strN[i][4]))

{

gram[i].first=strN[i][4];

}

else

{

if(!Norterminal(strN[i][5]))

{

gram[i].first=strN[i][5];

}

}

for(unsigned k=5;k<strN[i].length();k++)

{

if(strN[i][k]=='|')

{

if(!Norterminal(strN[i][k+1]))

{

gram[i].first+=strN[i][k+1];

}

else

{

if(!Norterminal(strN[i][k+2]))

{

gram[i].first+=strN[i][k+2];

}

}

}

}

}

int num=strN1.size();

//当第一个字符是非终结符时

for(int i=num-1;i>=0;i--)

{

for(unsigned l=0;l<strN.size()-1;l++)

{

if(gram[l].a[0]==strN1[i][0])

{

for(unsigned h=0;h<strN.size()-1;h++)

{

if(strN1[i][4]==gram[h].a[0]&&strN1[i][4]!=strN1[i][0])

{

gram[l].first+=gram[h].first;

string ustr(gram[l].first);

sort(ustr.begin(), ustr.end());

ustr.erase(unique(ustr.begin(), ustr.end()), ustr.end() );

gram[l].first=ustr;

break;

}

}

}

}

}

cout<<"firstTerm集合为："<<endl;

for(unsigned i=0;i<strN.size()-1;i++)

{

for(int k=0;k<gram[i].first.length();k++)

{

if(gram[i].first[k]==NULL)

{

gram[i].first.erase(gram[i].first.begin()+k);

}

}

cout<<gram[i].a<<" "<<gram[i].first<<endl;

}

string term="";

cout<<"终结符为："<<endl;

for(unsigned i=0;i<strN.size()-1;i++)

{

for(int j=0;j<strN[i].length();j++)

{

if(!Norterminal(strN[i][j])&&strN[i][j]!='$')

term+=strN[i][j];

}

string ustr(term);

sort(ustr.begin(), ustr.end());

ustr.erase(unique(ustr.begin(), ustr.end()), ustr.end() );

term=ustr;

}

cout<<term<<endl;

/\*转换规则形式\*/

cout<<"转换后的规则形式为："<<endl;

int listnumber =0;

for(unsigned i=0;i<strN1.size()-1&&i-1>=0;i++)

{

if(strN1[i].find("::")==-1)

{

string add="";

add=add+strN1[i-1][0]+"::="+strN1[i];

strN1[i]=add;

}

cout<<strN1[i]<<endl;

}

/\*获取lastTerm\*/

//直接获取lastTerm

for(int i=0;i<num-1;i++)

{

for(unsigned l=0;l<strN.size()-1;l++)

{

if(gram[l].a[0]==strN1[i][0])

{

for(unsigned h=0;h<strN.size()-1;h++)

{

int k=strN1[i].length();

if(!Norterminal(strN1[i][k-1]))

{

gram[l].follow+=strN1[i][k-1];

string ustr(gram[l].follow);

sort(ustr.begin(), ustr.end());

ustr.erase(unique(ustr.begin(), ustr.end()), ustr.end() );

gram[l].follow=ustr;

break;

}

else

{

if(!Norterminal(strN1[i][k-2]))

{

gram[l].follow+=strN1[i][k-2];

string ustr(gram[l].follow);

sort(ustr.begin(), ustr.end());

ustr.erase(unique(ustr.begin(), ustr.end()), ustr.end() );

gram[l].follow=ustr;

break;

}

}

}

}

}

}

for(int i=num-1;i>=0;i--)

{

for(unsigned l=0;l<strN.size()-1;l++)

{

if(gram[l].a[0]==strN1[i][0])

{

for(unsigned h=0;h<strN.size()-1;h++)

{

int k=strN1[i].length();

if(Norterminal(strN1[i][k-1]))

{

for(unsigned h1=0;h1<strN.size()-1;h1++)

if(gram[h1].a[0]==strN1[i][k-1])

{

gram[l].follow+=gram[h1].follow;

string ustr(gram[l].follow);

sort(ustr.begin(), ustr.end());

ustr.erase(unique(ustr.begin(), ustr.end()), ustr.end() );

gram[l].follow=ustr;

break;

}

}

}

}

}

}

cout<<"lastTerm集合为："<<endl;

for(unsigned i=0;i<strN.size()-1;i++)

{

cout<<gram[i].a<<" "<<gram[i].follow<<endl;

}

int list[term.length()][term.length()];

for(int i=0;i<term.length();i++)

{

for(int j=0;j<term.length();j++)

{

list[i][j]=0;

}

}

cout<<"相等的关系为："<<endl;

for(int i=num-1;i>=0;i--)

{

for(int k=0;k<strN1[i].length()&&k+1<strN1[i].length();k++)

{

if(Norterminal(strN1[i][k]))

{

if(!Norterminal(strN1[i][k-1])&&!Norterminal(strN1[i][k+1]))

{

cout<<strN1[i][k-1]<<strN1[i][k+1]<<endl;;

for(int h=0;h<term.length();h++)

{

if(strN1[i][k-1]==term[h])

for(int g=0;g<term.length();g++)

{

if(strN1[i][k+1]==term[g])

{

list[h][g]=1;

}

}

}

}

}

else

{

if(!Norterminal(strN1[i][k+1]))

{

cout<<strN1[i][k]<<strN1[i][k+1]<<endl;

for(int h=0;h<term.length();h++)

{

if(strN1[i][k]==term[h])

for(int g=0;g<term.length();g++)

{

if(strN1[i][k+1]==term[g])

{

list[h][g]=1;

}

}

}

}

}

}

}

//cout<<"小于关系为："<<endl;

for(int i=0;i<num-1;i++)

{

for(int k=0;k<strN1[i].length();k++)

{

if(!Norterminal(strN1[i][k]))

{

if(Norterminal(strN1[i][k+1]))

{

for(int h=0;h<term.length();h++)

{

if(strN1[i][k]==term[h])

{

for(int j=0;j<num-1;j++)

{

if(strN1[i][k+1]==gram[j].a[0])

{

for(int g=0;g<term.length();g++)

{

for(int f=0;f<gram[j].first.length();f++)

if(gram[j].first[f]==term[g])

{

list[h][g]=2;

}

}

}

}

}

}

}

}

}

}

//大于关系

for(int i=0;i<num-1;i++)

{

for(int k=0;k<strN1[i].length();k++)

{

if(Norterminal(strN1[i][k]))

{

if(!Norterminal(strN1[i][k+1]))

{

for(int h=0;h<term.length();h++)

{

if(strN1[i][k+1]==term[h])

for(int j=0;j<num-1;j++)

{

if(strN1[i][k]==gram[j].a[0])

{

for(int g=0;g<term.length();g++)

{

for(int f=0;f<gram[j].first.length();f++)

if(gram[j].follow[f]==term[g])

{

list[g][h]=3;

}

}

}

}

}

}

}

}

}

for(int i=0;i<term.length();i++)

{

for(int j=0;j<term.length();j++)

{

cout<<list[i][j]<<" ";

}

cout<<endl;

}

cout<<"算符优先矩阵为："<<endl;

cout<<" |";

for(int i=0;i<term.length();i++)

{

cout<<setw(10)<<term[i];

cout<<"|";

}

cout<<endl;

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl;

for(int i=0;i<term.length();i++)

{

printf("%2c",term[i]);

cout<<" "<<"|";

for(int j=0;j<term.length();j++)

{

if(list[i][j]==0)

cout<<setw(10)<<" "<<"|";

if(list[i][j]==1)

cout<<setw(10)<<"="<<"|";

if(list[i][j]==2)

cout<<setw(10)<<"<"<<"|";

if(list[i][j]==3)

cout<<setw(10)<<">"<<"|";

}

cout<<endl;

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl;

}

}

int array[8][8]={ //优先矩阵，1='>',-1='<',0='='

{ 1, 1,-1,-1,-1, 1,-1, 1},

{ 1, 1,-1,-1,-1, 1,-1, 1},

{ 1, 1, 1, 1,-1, 1,-1, 1},

{ 1, 1, 1, 1,-1, 1,-1, 1},

{-1,-1,-1,-1,-1, 0,-1, 1},

{ 1, 1, 1, 1,' ',1,' ',1},

{ 1, 1, 1, 1,' ',1,' ',1},

{-1,-1,-1,-1,-1,-1,-1, 0}

};

void init()

{

char a1[200];

int i=0,j=0;

if((fp=fopen("input1.txt","r"))==NULL)

{

cout<<"文件打开失败！"<<endl;

exit(0);

}

a1[i++]=fgetc(fp);

while(!feof(fp))

{

a1[i++]=fgetc(fp);

if((i%5)==0)

a[j++]=a1[i-2];

}

fclose(fp);

cout<<"读入字符串为：";

cout<<a<<endl;

}

void error()

{

cout<<"算符优先分析失败!"<<endl;

exit(0);

}

int getVtNumber(char ch)//获取终结符号ch的序号

{

char vt[8]={'+','-','\*','/','(',')','i','#'};

int i;

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

if(ch==vt[i])

break;

return i;

}

bool Higher(char ch1,char ch2)

{

int c1,c2;

c1=getVtNumber(ch1);

c2=getVtNumber(ch2);

if(array[c1][c2]==1)

return true;

return false;

}

bool Lower(char ch1,char ch2)

{

int c1,c2;

c1=getVtNumber(ch1);

c2=getVtNumber(ch2);

if(array[c1][c2]==-1)

return true;

return false;

}

bool Equal(char ch1,char ch2)

{

int c1,c2;

c1=getVtNumber(ch1);

c2=getVtNumber(ch2);

if(array[c1][c2]==0)

return true;

return false;

}

//寻找与最左素短语匹配的产生式

bool IsReduce(int begin,int end,int len)//begin,end,len分别表示最左素短语的开头和结束位置及最左素短语的长度

{

if(len==1)

if(Stack[begin]=='i')

return true;

if(len==3)

if(Stack[begin]=='(' && Stack[end]==')' && Norterminal(Stack[begin+1]))

return true;

else

if(Norterminal(Stack[begin]))

if(Norterminal(Stack[end]))

if(Stack[begin+1]=='+' || Stack[begin+1]=='-' || Stack[begin+1]=='\*' || Stack[begin+1]=='/')

return true;

return false;

}

//寻找最左素短语、控制规约与移进

void parser(void)

{

int i,k;

char r;

i=0;k=0;

Stack[0]='#';

do

{

int j;

r=a[i++];

if(!Norterminal(Stack[k]))

j=k;

else

j=k-1;

while(Higher(Stack[j],r))

{

char q;

do

{

q=Stack[j];

if(!Norterminal(Stack[j-1]))

j--;

else

j-=2;

}while(!Lower(Stack[j],q));

if(IsReduce(j+1,k,k-j))

{

k=j+1; //规约

Stack[k]='N';

}

else

error();

}

if(Lower(Stack[j],r) || Equal(Stack[j],r))

Stack[++k]=r; //移进

else

error();

}while(r != '#');

cout<<"算符优先分析成功!"<<endl;

}

int main()

{

analyze();

init();

parser();

}