LEADING AND TRAILING

NAME : ATHRESH KUMAR LABDE
RA1911033010146
M1

AIM : A program to implement Leading and Trailing

ALGORITHM :
1. For Leading, check for the first non-terminal.
2. If found, print it.
3. Look for next production for the same non-terminal.
4. If not found, recursively call the procedure for the single non-terminal present before the comma or End Of Production String.
5. Include it's results in the result of this non-terminal.
6. For trailing, we compute same as leading but we start from the end of the production to the beginning.
7. Stop.

CODE :

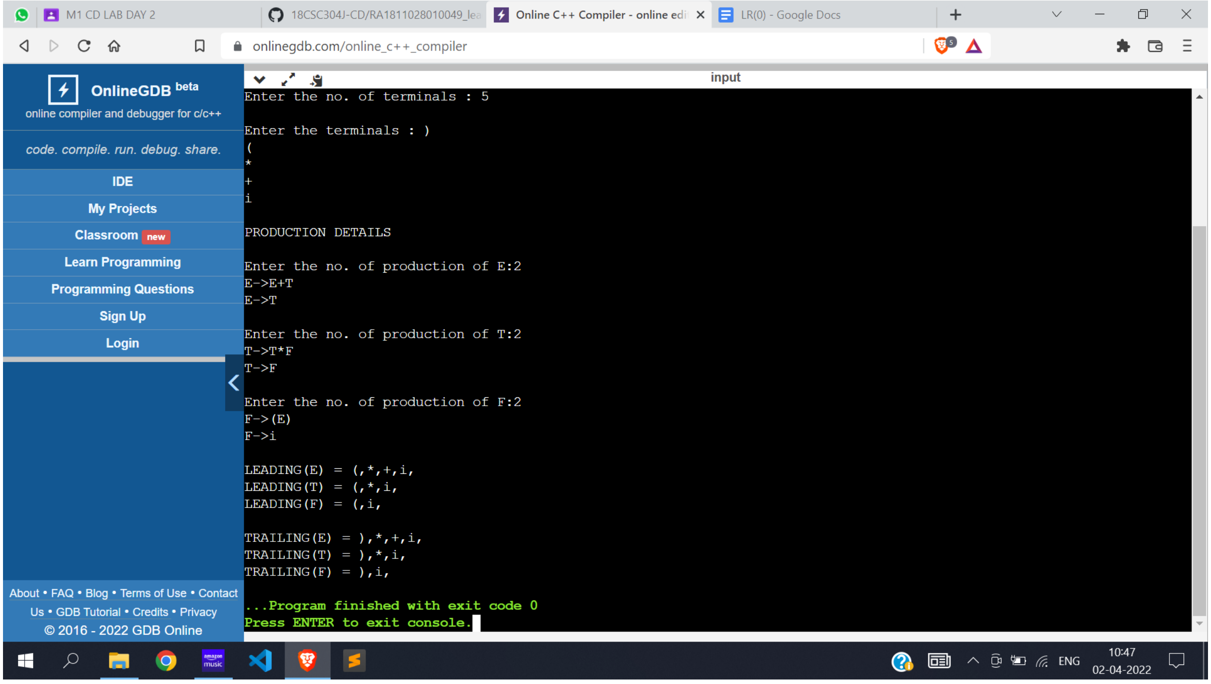
```
#include<iostream>
#include<conio.h>
#include<string.h>

using namespace std;

char prod[20][20],listofvar[26]="ABCDEFGHIJKLMNOPQR";
int novar=1,i=0,j=0,k=0,n=0,m=0,arr[30];
int noitem=0;

struct Grammar
{
        char lhs;
        char rhs[8];
}g[20],item[20],clos[20][10];

int isvariable(char variable)
{
        for(int i=0;i<novar;i++)
                if(g[i].lhs==variable)
                        return i+1;
        return 0;
}
void findclosure(int z, char a)
{
        int n=0,i=0,j=0,k=0,l=0;
        for(i=0;i<arr[z];i++)
```

```
    {
        for(j=0;j<strlen(clos[z][i].rhs);j++)
        {
            if(clos[z][i].rhs[j]=='.' && clos[z][i].rhs[j+1]==a)
            {
                clos[noitem][n].lhs=clos[z][i].lhs;
                strcpy(clos[noitem][n].rhs,clos[z][i].rhs);
                char temp=clos[noitem][n].rhs[j];
                clos[noitem][n].rhs[j]=clos[noitem][n].rhs[j+1];
                clos[noitem][n].rhs[j+1]=temp;
                n=n+1;
            }
        }
    }
    for(i=0;i<n;i++)
    {
        for(j=0;j<strlen(clos[noitem][i].rhs);j++)
        {
            if(clos[noitem][i].rhs[j]=='.' && isvariable(clos[noitem][i].rhs[j+1])>0)
            {
                for(k=0;k<novar;k++)
                {
                    if(clos[noitem][i].rhs[j+1]==clos[0][k].lhs)
                    {
                        for(l=0;l<n;l++)
                            if(clos[noitem][l].lhs==clos[0][k].lhs &&
strcmp(clos[noitem][l].rhs,clos[0][k].rhs)==0)
                                break;
                        if(l==n)
                        {
                            clos[noitem][n].lhs=clos[0][k].lhs;
                        strcpy(clos[noitem][n].rhs,clos[0][k].rhs);
                            n=n+1;
                        }
                    }
                }
            }
        }
    }
    arr[noitem]=n;
    int flag=0;
    for(i=0;i<noitem;i++)
    {
        if(arr[i]==n)
        {
            for(j=0;j<arr[i];j++)
            {
                int c=0;
                for(k=0;k<arr[i];k++)
                    if(clos[noitem][k].lhs==clos[i][k].lhs &&
strcmp(clos[noitem][k].rhs,clos[i][k].rhs)==0)
                        c=c+1;
                if(c==arr[i])
                {
                    flag=1;
                    goto exit;
                }
            }
        }
    }
    exit:;
    if(flag==0)
        arr[noitem++]=n;
}

int main()
{
    cout<<"ENTER THE PRODUCTIONS OF THE GRAMMAR(0 TO END) :\n";
    do
    {
        cin>>prod[i++];
    }while(strcmp(prod[i-1],"0")!=0);
    for(n=0;n<i-1;n++)
    {
        m=0;
        j=novar;
        g[novar++].lhs=prod[n][0];
        for(k=3;k<strlen(prod[n]);k++)
        {
            if(prod[n][k] != '|')
            g[j].rhs[m++]=prod[n][k];
            if(prod[n][k]=='|')
            {
                g[j].rhs[m]='\0';
                m=0;
                j=novar;
                g[novar++].lhs=prod[n][0];
            }
        }
    }
    for(i=0;i<26;i++)
        if(!isvariable(listofvar[i]))
            break;
    g[0].lhs=listofvar[i];
    char temp[2]={g[1].lhs,'\0'};
    strcat(g[0].rhs,temp);
```

```
cout<<"\n\n augumented grammar \n";
for(i=0;i<novar;i++)
    cout<<endl<<g[i].lhs<<"->"<<g[i].rhs<<" ";

for(i=0;i<novar;i++)
{
    clos[noitem][i].lhs=g[i].lhs;
    strcpy(clos[noitem][i].rhs,g[i].rhs);
    if(strcmp(clos[noitem][i].rhs,"ε")==0)
        strcpy(clos[noitem][i].rhs,".");
    else\
    {
        for(int j=strlen(clos[noitem][i].rhs)+1;j>=0;j--)
            clos[noitem][i].rhs[j]=clos[noitem][i].rhs[j-1];
        clos[noitem][i].rhs[0]='.';
    }
}
arr[noitem++]=novar;
for(int z=0;z<noitem;z++)
{
    char list[10];
    int l=0;
    for(j=0;j<arr[z];j++)
    {
        for(k=0;k<strlen(clos[z][j].rhs)-1;k++)
        {
            if(clos[z][j].rhs[k]=='.')
            {
                for(m=0;m<l;m++)
                    if(list[m]==clos[z][j].rhs[k+1])
                        break;
                if(m==l)
                    list[l++]=clos[z][j].rhs[k+1];
            }
        }
    }
    for(int x=0;x<l;x++)
        findclosure(z,list[x]);
}
cout<<"\n THE SET OF ITEMS ARE \n\n";
for(int z=0; z<noitem; :z++)
{
    cout<<"\n I"<<z<<"\n\n";

    for(j=0;j<arr[z];j++)
        cout<<clos[z][j].lhs<<"->"<<clos[z][j].rhs<<"\n";
    if(z==1){
       cout<<"Special output\n";
```

```
                cout<<clos[1][0].lhs<<"->"<<clos[1][0].rhs<<"\n";
                cout<<clos[5][0].lhs<<"->"<<clos[5][0].rhs<<"\n";
            }

        }
}
```

OUTPUT :


M1 CD LAB DAY 2
18CSC304J-CD/RA1811028010049_lea
Online C++ Compiler - online edi
LR(0) - Google Docs
onlinegdb.com/online_c++_compiler
OnlineGDB beta
online compiler and debugger for c/c++
code. compile. run. debug. share.
IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Sign Up
Login
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
© 2016 - 2022 GDB Online
input
ENG
10:47
02-04-2022

```
Enter the no. of terminals : 5

Enter the terminals : )
(
*
+
i

PRODUCTION DETAILS

Enter the no. of production of E:2
E->E+T
E->T

Enter the no. of production of T:2
T->T*F
T->F

Enter the no. of production of F:2
F->(E)
F->i

LEADING(E)  =  (,*,+,i,
LEADING(T)  =  (,*,i,
LEADING(F)  =  (,i,

TRAILING(E)  = ),*,+,i,
TRAILING(T)  = ),*,i,
TRAILING(F)  = ),i,

...Program finished with exit code 0
Press ENTER to exit console.
```

RESULT : hence we have successfully verified the Leading and trailing experiment by implementing and running the code.