

RA1811028010049

SHUSHRUT KUMAR

CSE - CC J2

COMPILER DESIGN

## **LAB EXP 8 : LEADING AND TRAILING**

**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.h>
#include<conio.h>
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
int vars,terms,i,j,k,m,rep,count,temp=-1;
char var[10],term[10],lead[10][10],trail[10][10];
struct grammar
{
    int prodno;
```

```

        char lhs,rhs[20][20];
    }gram[50];
    void get()
    {
        cout<<"\n----- LEADING AND TRAILING ----- \n";
        cout<<"\nEnter the no. of variables : ";
        cin>>vars;
        cout<<"\nEnter the variables : \n";
        for(i=0;i<vars;i++)
        {
            cin>>gram[i].lhs;
            var[i]=gram[i].lhs;
        }
        cout<<"\nEnter the no. of terminals : ";
        cin>>terms;
        cout<<"\nEnter the terminals : ";
        for(j=0;j<terms;j++)
            cin>>term[j];
        cout<<"\n----- PRODUCTION DETAILS ----- \n";
        for(i=0;i<vars;i++)
        {
            cout<<"\nEnter the no. of production of "<<gram[i].lhs<<": ";
            cin>>gram[i].prodno;
            for(j=0;j<gram[i].prodno;j++)
            {
                cout<<gram[i].lhs<<"->";
                cin>>gram[i].rhs[j];
            }
        }
    }
}

void leading()

```

```

{
    for(i=0;i<vars;i++)
    {
        for(j=0;j<gram[i].prodno;j++)
        {
            for(k=0;k<terms;k++)
            {
                if(gram[i].rhs[j][0]==term[k])
                    lead[i][k]=1;
                else
                {
                    if(gram[i].rhs[j][1]==term[k])
                        lead[i][k]=1;
                }
            }
        }
    }
}

for(rep=0;rep<vars;rep++)
{
    for(i=0;i<vars;i++)
    {
        for(j=0;j<gram[i].prodno;j++)
        {
            for(m=1;m<vars;m++)
            {
                if(gram[i].rhs[j][0]==var[m])
                {
                    temp=m;
                    goto out;
                }
            }
        }
    }
}

```

```

        out:
        for(k=0;k<terms;k++)
        {
            if(lead[temp][k]==1)
                lead[i][k]=1;
        }
    }
}

void trailing()
{
    for(i=0;i<vars;i++)
    {
        for(j=0;j<gram[i].prodno;j++)
        {
            count=0;
            while(gram[i].rhs[j][count]!='\x0')
                count++;
            for(k=0;k<terms;k++)
            {
                if(gram[i].rhs[j][count-1]==term[k])
                    trail[i][k]=1;
                else
                {
                    if(gram[i].rhs[j][count-2]==term[k])
                        trail[i][k]=1;
                }
            }
        }
    }
}

```

```

for(rep=0;rep<vars;rep++)
{
    for(i=0;i<vars;i++)
    {
        for(j=0;j<gram[i].prodno;j++)
        {
            count=0;
            while(gram[i].rhs[j][count]!='\x0')
                count++;
            for(m=1;m<vars;m++)
            {
                if(gram[i].rhs[j][count-1]==var[m])
                    temp=m;
            }
            for(k=0;k<terms;k++)
            {
                if(trail[temp][k]==1)
                    trail[i][k]=1;
            }
        }
    }
}

void display()
{
    for(i=0;i<vars;i++)
    {
        cout<<"\nLEADING("<<gram[i].lhs<<") = ";
        for(j=0;j<terms;j++)
        {
            if(lead[i][j]==1)

```

```

        cout<<term[j]<<",";

    }

}

cout<<endl;
for(i=0;i<vars;i++)
{
    cout<<"\nTRAILING("<<gram[i].lhs<<") = ";
    for(j=0;j<terms;j++)
    {
        if(trail[i][j]==1)
            cout<<term[j]<<",";

    }

}

}

void main()
{
    clrscr();
    get();
    leading();
    trailing();
    display();
    getch();
}

```

## OUTPUT :

```
----- LEADING AND TRAILING -----  
  
Enter the no. of variables : 3  
  
Enter the variables :  
E  
T  
F  
  
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,
```

## RESULT :

The program was successfully compiled and run.