Computation of Leading and Trailing
Compiler Design – Lab 8

Name: Ojas Patil
Reg No: 21BAI1106

AIM

Write a C / C++ / Java program to Compute the Leading and Trailing of a given Grammar.

Algorithm

1. Input and Grammar Initialization: Accept user input for variables, terminals, and production rules.
2. Leading Set Computation: Calculate leading sets for each variable based on production rules.
3. Propagate Leading Sets: Update leading sets by propagating them through productions with variable symbols.
4. Trailing Set Computation: Compute trailing sets for each variable using production rules.
5. Propagate Trailing Sets: Update trailing sets by propagating them through productions with variable symbols.
6. Display Leading and Trailing Sets: Output computed leading and trailing sets for each variable.

Explanation

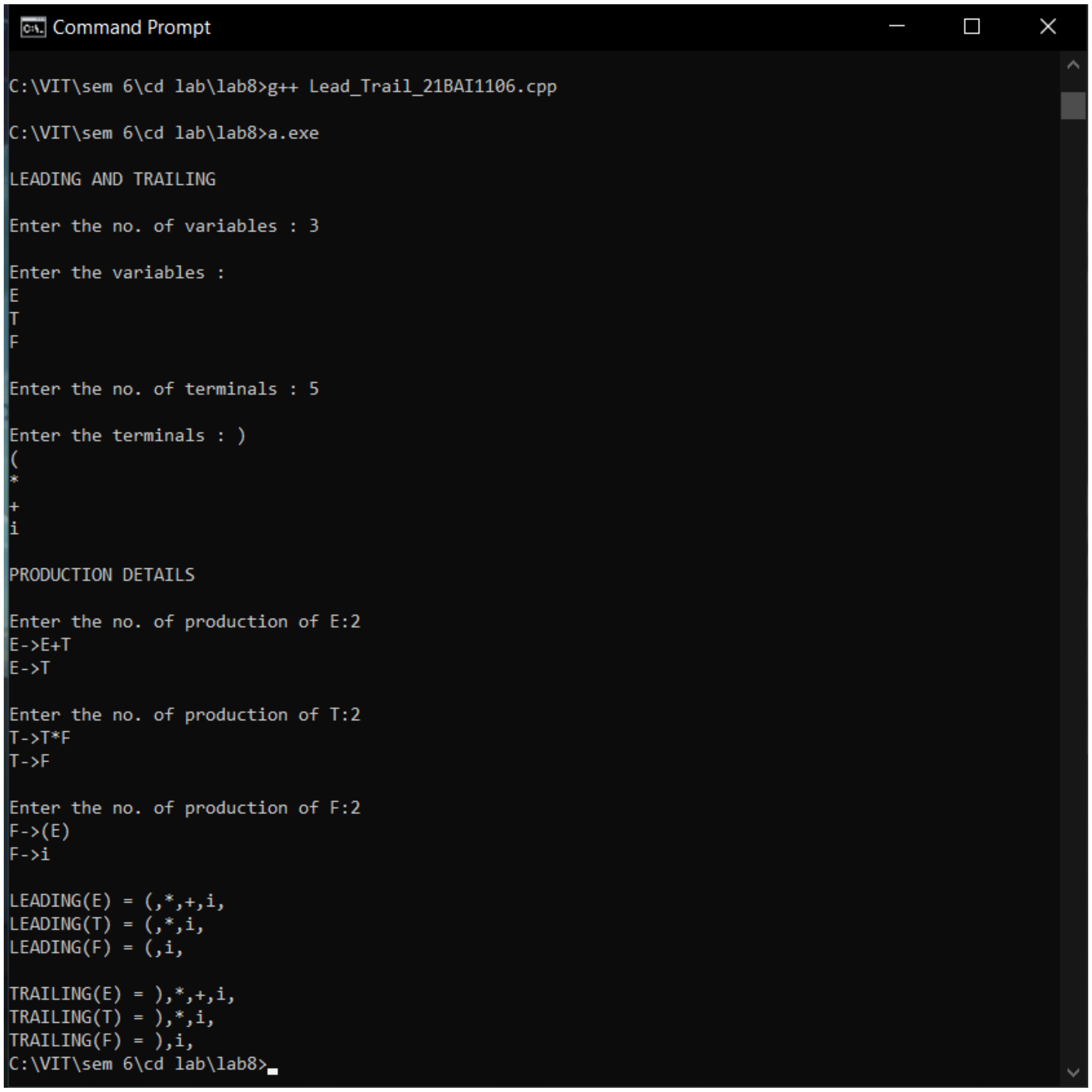
This code computes the leading and trailing sets for each variable in a context-free grammar. It first accepts user-defined grammar rules, initializes variables, and terminals. Then, it calculates the leading sets by analyzing the first symbols of productions and propagating leading sets of variables. Similarly, trailing sets are computed by examining the last symbols of productions. Finally, the leading and trailing sets for each variable are displayed.

Sample Input

```
Enter the no. of variables : 3
Enter the variables : E, T, F
Enter the no. of terminals : 5
Enter the terminals : ), (, *, +, i

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
```

Output Screenshot



```
C:\VIT\sem 6\cd lab\lab8>g++ Lead_Trail_21BAI1106.cpp

C:\VIT\sem 6\cd lab\lab8>a.exe

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,
C:\VIT\sem 6\cd lab\lab8>
```

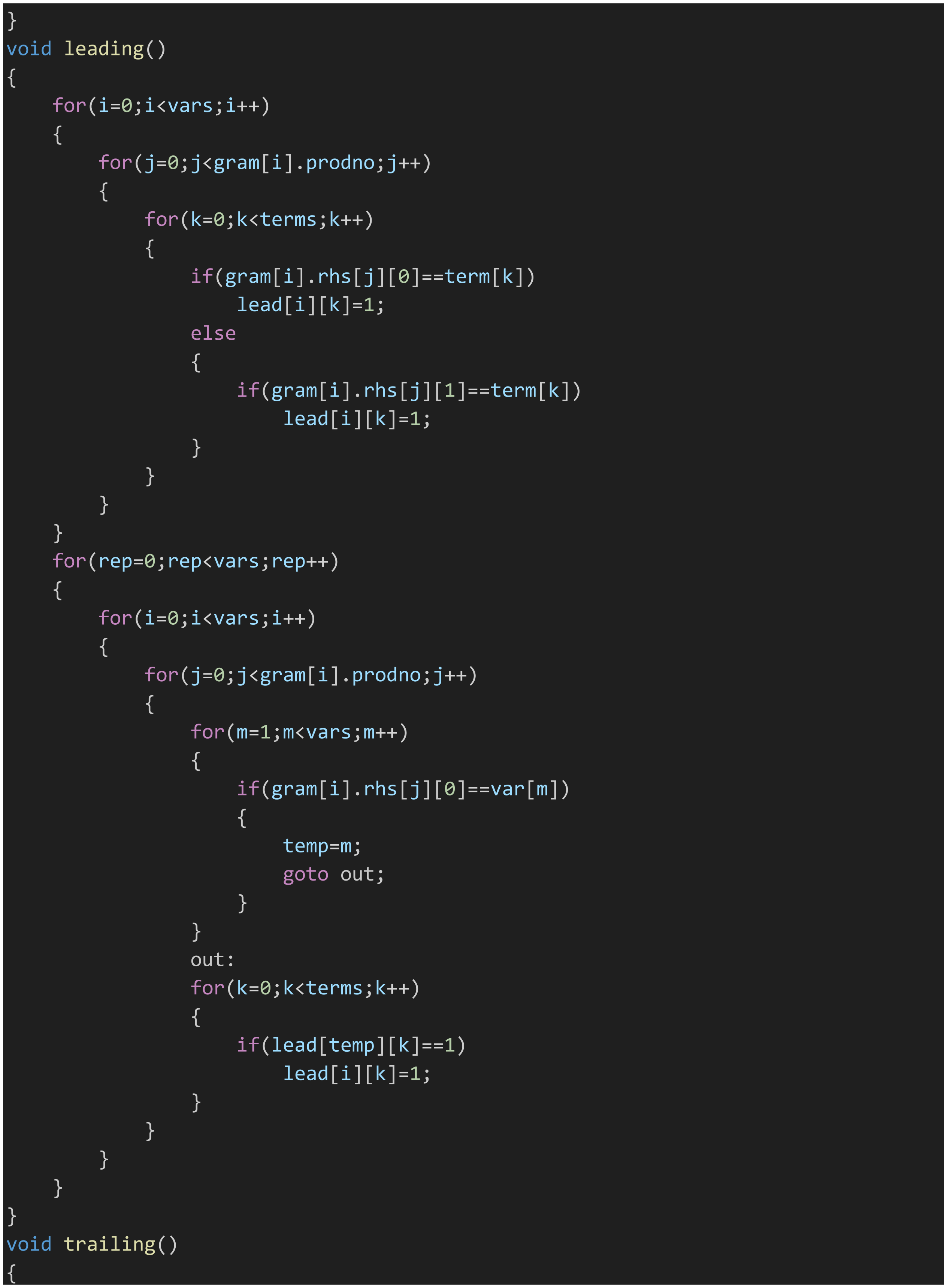
Source Code

```
#include<iostream>
#include<conio.h>
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
using namespace std;

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<<"\nLEADING 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<<"\nPRODUCTION 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()
{
```

```c
    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]<<",";
        }
    }
}
int main()
{
    get();
    leading();
    trailing();
    display();

    return 0;
}
```

Result

Thus, we have written an CPP Code to take the given grammar as input and derives the leading and trailing sets.