Harshit Aggarwal (RA1911003010782)

Experiment 8: Computation of Leading and Trailing

17/3/22

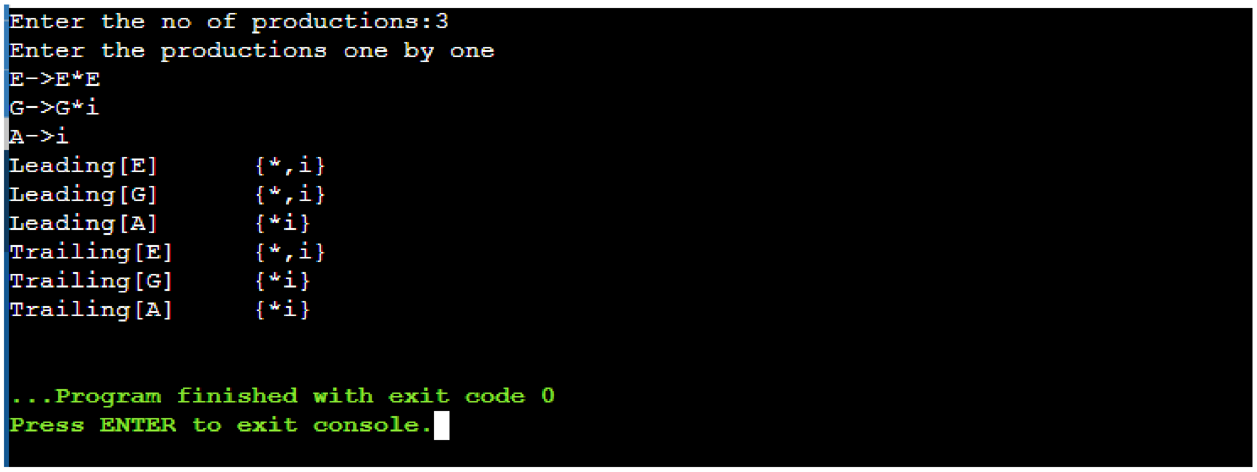
Aim:
To implement leading trailing using C++ program.

Algorithm:
1. Start
2. Get the input expression and store it in the input buffer.
3. Read the data from the input buffer one at the time
4. Using stack and push & pop operation with respect to production rules available.
5. Continue the process till Lead and trail is obtained.
6. Display the Leads and Trails of all non terminals.
7. Stop

Code:
```
#include<iostream>
#include<string.h>
// #include<conio.h>
using namespace std;

int nt,t,top=0;
char s[50],NT[10],T[10],st[50],l[10][10],tr[50][50];

int searchnt(char a)
{
        int count=-1,i;
        for(i=0;i<nt;i++)
        {
                if(NT[i]==a)
                        return i;
        }
        return count;
}

int searchter(char a)
{
        int count=-1,i;
        for(i=0;i<t;i++)
        {
                if(T[i]==a)
                        return i;
        }
        return count;
}

void push(char a)
{
        s[top]=a;
        top++;
}

char pop()
{
        top--;
        return s[top];
}

void installl(int a,int b)
{
        if(l[a][b]=='f')
        {
                l[a][b]='t';
                push(T[b]);
                push(NT[a]);
        }
}

void installt(int a,int b)
{
        if(tr[a][b]=='f')
        {
                tr[a][b]='t';
                push(T[b]);
                push(NT[a]);
        }
}

int main()
{
        int i,s,k,j,n;
        char pr[30][30],b,c;
        cout<<"Enter the no of productions:";
        cin>>n;
```

```
cout<<"Enter the productions one by one\n";
for(i=0;i<n;i++)
        cin>>pr[i];
nt=0;
t=0;
for(i=0;i<n;i++)
{
        if((searchnt(pr[i][0]))==-1)
                NT[nt++]=pr[i][0];
}
for(i=0;i<n;i++)
{
        for(j=3;j<strlen(pr[i]);j++)
        {
                if(searchnt(pr[i][j])==-1)
                {
                        if(searchter(pr[i][j])==-1)
                        {
                                T[t++]=pr[i][j];
                        }
                }
        }
}
for(i=0;i<nt;i++)
{
        for(j=0;j<t;j++)
                l[i][j]='f';
}
for(i=0;i<nt;i++)
{
        for(j=0;j<t;j++)
                tr[i][j]='f';
}
for(i=0;i<nt;i++)
{
        for(j=0;j<n;j++)
        {
                if(NT[(searchnt(pr[j][0]))]==NT[i])
                {
                        if(searchter(pr[j][3])!=-1)
                                installl(searchnt(pr[j][0]),searchter(pr[j][3]));
                        else
                        {
                                for(k=3;k<strlen(pr[j]);k++)
                {
                    if(searchnt(pr[j][k])==-1)
                    {
                        installl(searchnt(pr[j][0]),searchter(pr[j][k]));
                        break;
                    }
                }
            }
        }
    }
}
while(top!=0)
{
    b=pop();
    c=pop();
    for(s=0;s<n;s++)
    {
        if(pr[s][3]==b)
            installl(searchnt(pr[s][0]),searchter(c));
    }
}
for(i=0;i<nt;i++)
{
    cout<<"Leading["<<NT[i]<<"]"<<"\t{";
    for(j=0;j<t;j++)
    {
        if(l[i][j]=='t'&&j+1!=t)
            cout<<T[j]<<",";
        else
            cout<<T[j];
    }
    cout<<"}\n";
}
top=0;
for(i=0;i<nt;i++)
{
    for(j=0;j<n;j++)
    {
        if(NT[searchnt(pr[j][0])]==NT[i])
        {
            if(searchter(pr[j][strlen(pr[j])-1])!=-1)
                installt(searchnt(pr[j][0]),searchter(pr[j][strlen(pr[j])-1]));
            else
            {
                    for(k=(strlen(pr[j])-1);k>=3;k--)
                    {
                        if(searchnt(pr[j][k])==-1)
                        {
                            installt(searchnt(pr[j][0]),searchter(pr[j][k]));
                            break;
                        }
                    }
                }
            }
        }
    }
    while(top!=0)
    {
        b=pop();
        c=pop();
        for(s=0;s<n;s++)
        {
            if(pr[s][3]==b)
                installt(searchnt(pr[s][0]),searchter(c));
        }
    }
    for(i=0;i<nt;i++)
    {
        cout<<"Trailing["<<NT[i]<<"]"<<"\t{";
        for(j=0;j<t;j++)
        {
            if(tr[i][j]=='t'&&j+1!=t)
                cout<<T[j]<<",";
            else
                cout<<T[j];
        }
        cout<<"}\n";
    }
    return 0;
}
```

Output:



```
Enter the no of productions:3
Enter the productions one by one
E->E*E
G->G*i
A->i
Leading[E]      {*,i}
Leading[G]      {*,i}
Leading[A]      {*i}
Trailing[E]     {*,i}
Trailing[G]     {*i}
Trailing[A]     {*i}


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

Result:

Hence Leads and Trails is obtained for Given Grammar.