

LR(0)

NAME : ATHRESH KUMAR LABDE
RA1911033010146
M1

AIM : To verify leading and trailing in this experiment by implementing and running the code.

ALGORITHM :

Input – Context Free Grammar G

Output – LEADING (A) = {a} iff Boolean Array L [A, a] = true

Method – Procedure Install (A, a) will make L (A, a) to true if it was not true earlier.

begin

For each non-terminal A and terminal a

L [A, a] = false ;

For each production of form $A \rightarrow a\alpha$ or $A \rightarrow B a \alpha$

Install (A, a) ;

While the stack not empty

Pop top pair (B, a) form Stack ;

For each production of form $A \rightarrow B \alpha$

Install (A, a);

end

Procedure Install (A, a)

begin

If not L [A, a] then

L [A, a] = true

push (A, a) onto stack.

end

Algorithm to compute TRAILING

Input – Context Free Grammar G

Output – TRAILING (A) = {a} iff Boolean Array T [A, a] = true

Method

begin

For each non-terminal A and terminal a

$T[A, a] = \text{false}$;

For each production of form $A \rightarrow \alpha a$ or $A \rightarrow \alpha a B$

 Install (A, a) ;

While the stack not empty

 Pop top pair (B, a) form Stack ;

 For each production of form $A \rightarrow \alpha B$

 Install (A, a);

end

Procedure Install (A, a)

begin

If not $T[A, a]$ then

$T[A, a] = \text{true}$

 push (A, a) onto stack.

end

Algorithm for Computing Operator Precedence Relations

Input – An Operator Grammar

Output – A Precedence Relations between terminals and symbols.

Method

begin

For each production $A \rightarrow B_1, B_2, \dots \dots \dots B_n$

 for $i = 1$ to $n - 1$

 If B_i and B_{i+1} are both terminals then

 set $B_i = B_{i+1}$

 If $i \leq n - 2$ and B_i and B_{i+2} are both terminals and B_{i+1} is non-terminal then

 set $B_i = B_{i+1}$

 If B_i is terminal & B_{i+1} is non-terminal then for all a in LEADING (B_{i+1})

set $B_i < .a$

If B_i is non-terminal & B_{i+1} is terminal then for all a in TRAILING (B_i)

set $a . > B_{i+1}$

End

CODE :

```
#include<iostream>
```

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

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

```
using namespace std;
```

```
char prod[20][20],listofvar[26]="ABCDEFGHIJKLMNOPQRSTUVWXYZ";
```

```
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 augmented 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--)

```

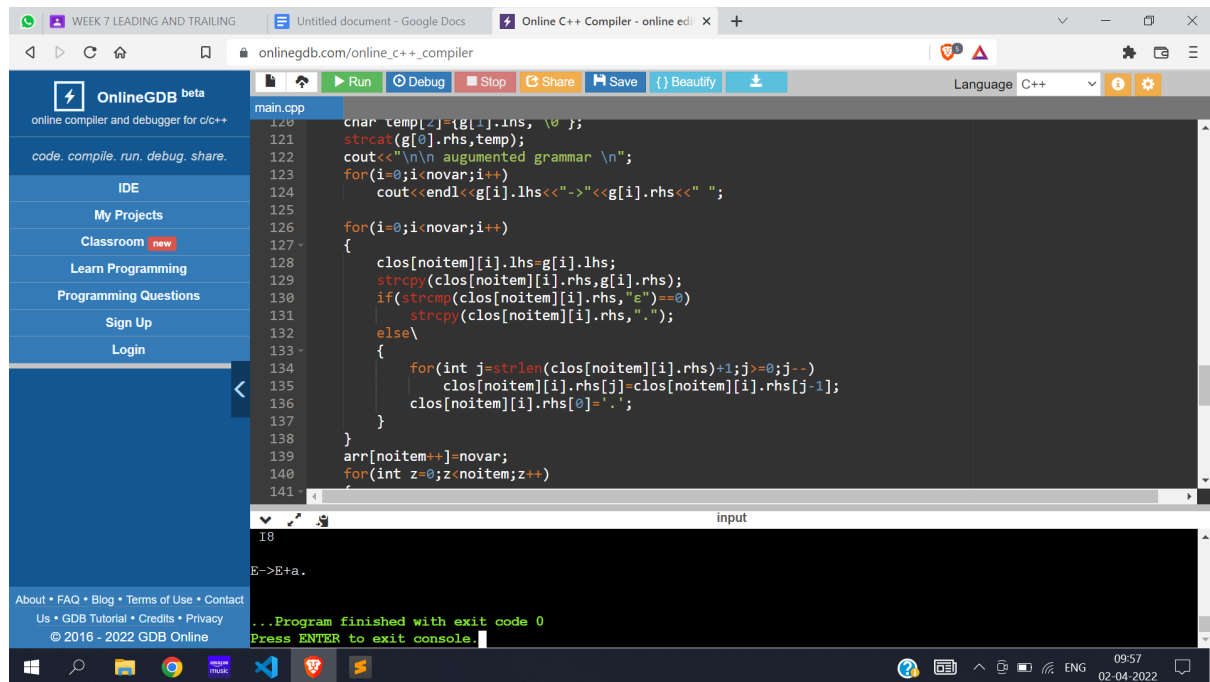
```

        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 l"<<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";
    }
}
}

```

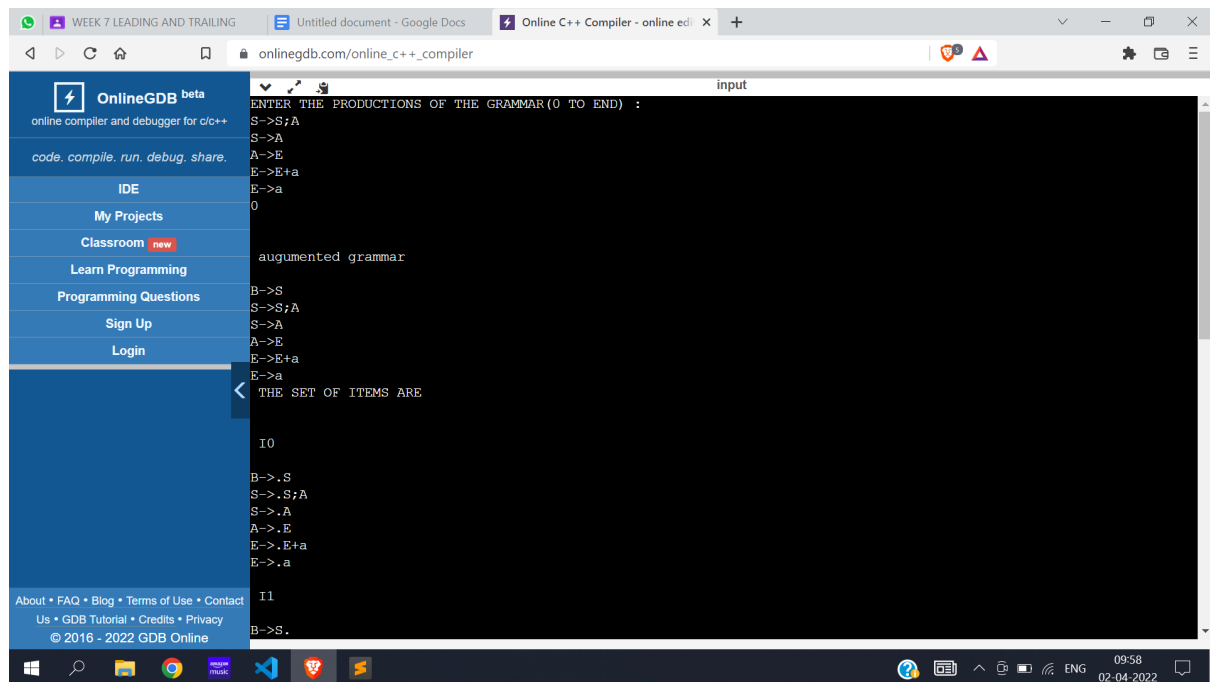
CODE SCREENSHOT :



The screenshot shows the OnlineGDB IDE interface. The left sidebar contains navigation links: IDE, My Projects, Classroom (new), Learn Programming, Programming Questions, Sign Up, and Login. The main editor displays a C++ program in `main.cpp` that calculates LR(0) item sets and transitions for a grammar. The grammar rules are: `S -> S;A`, `S -> A`, `A -> E`, and `E -> E+a`. The code uses arrays to store the LR(0) items and transitions. The output console at the bottom shows the program finished with exit code 0 and the prompt "Press ENTER to exit console."

```
120 char temp[z]={g[i].lhs, '\0'};
121 strcat(g[0].rhs,temp);
122 cout<<"\n\n augmented grammar \n";
123 for(i=0;i<novar;i++)
124     cout<<endl<<g[i].lhs<<"->"<<g[i].rhs<<" ";
125
126 for(i=0;i<novar;i++)
127 {
128     clos[noitem][i].lhs=g[i].lhs;
129     strcpy(clos[noitem][i].rhs,g[i].rhs);
130     if(strcmp(clos[noitem][i].rhs,"e")==0)
131         strcpy(clos[noitem][i].rhs,".");
132     else\
133     {
134         for(int j=strlen(clos[noitem][i].rhs)+1;j>=0;j--)
135             clos[noitem][i].rhs[j]=clos[noitem][i].rhs[j-1];
136         clos[noitem][i].rhs[0]='.';
137     }
138 }
139 arr[noitem++]=novar;
140 for(int z=0;z<noitem;z++)
```

OUTPUT :



The screenshot shows the output of the program in the OnlineGDB IDE. The output is displayed in the console area, showing the LR(0) item sets and transitions for the grammar. The output is as follows:

```
ENTER THE PRODUCTIONS OF THE GRAMMAR(0 TO END) :
S->S;A
S->A
A->E
E->E+a
E->a
0
augmented grammar
B->S
S->S;A
S->A
A->E
E->E+a
E->a
THE SET OF ITEMS ARE
I0
B->S
S->S;A
S->A
A->E
E->E+a
E->a
I1
B->S.
```

The screenshot shows the OnlineGDB website in a web browser. The browser's address bar displays 'onlinegdb.com/online_c++_compiler'. The website has a blue sidebar menu on the left with the following items: 'OnlineGDB beta', 'code. compile. run. debug. share.', 'IDE', 'My Projects', 'Classroom new', 'Learn Programming', 'Programming Questions', 'Sign Up', and 'Login'. The main area of the website is a dark-themed editor with the title 'input'. The console output on the right shows the following text: 'I3', 'A->E.', 'E->E+.a', 'I4', 'E->a.', 'I5', 'S->S;.A', 'A->.E', 'E->.E+a', 'E->.a', 'I6', 'E->E+.a', 'I7', 'S->S;.A', 'I8', 'E->E+.a', and '...Program finished with exit code 0 Press ENTER to exit console.' The browser's taskbar at the bottom shows the Windows logo, search icon, and several application icons. The system clock in the bottom right corner indicates the time is 09:58 on 02-04-2022.

RESULT : hence we have successfully verified the LR(0)experiment by implementing and running the code.