

ELIMINATION OF LEFT RECURSION

EX. NO. 4(a)

Anas Ahmed Ather (RA2011031010006)

Date: 14/2/23

AIM: A program for Elimination of Left Recursion.

ALGORITHM:

1. Start the program.
2. Initialize the arrays for taking input from the user.
3. Prompt the user to input the no. of non-terminals having left recursion and no. of productions for these non-terminals.
4. Prompt the user to input the production for non-terminals.
5. Eliminate left recursion using the following rules:-

$$A \rightarrow A\alpha_1 \mid A\alpha_2 \mid \dots \mid A\alpha_m$$
$$A \rightarrow \beta_1 \mid \beta_2 \mid \dots \mid \beta_n$$

Then replace it by

$$A \rightarrow \beta_i A' \quad i=1,2,3,\dots,m$$
$$A' \rightarrow \alpha_j \quad j=1,2,3,\dots,n$$
$$A' \rightarrow \epsilon$$

6. After eliminating the left recursion by applying these rules, display the productions without left recursion.
7. Stop.

PROGRAM:

```
#include <iostream>
#include <vector>
#include <string>
using namespace std;
int main()
{
    int n;
    cout<<"\nEnter number of non terminals: ";
    cin>>n;
    cout<<"\nEnter non terminals one by one: ";
    int i;
    vector<string> nonter(n);
    vector<int> leftrecr(n,0);
    for(i=0;i<n;++i) {
        cout<<"\nNon terminal "<<i+1<<" : ";
        cin>>nonter[i];
```

```

}
vector<vector<string> > prod;
cout<<"\nEnter '^' for null";
for(i=0;i<n;++i) {
cout<<"\nNumber of "<<nonter[i]<<" productions: ";
int k;
cin>>k;
int j;
cout<<"\nOne by one enter all "<<nonter[i]<<" productions";
vector<string> temp(k);
for(j=0;j<k;++j) {
cout<<"\nRHS of production "<<j+1<<": ";
string abc;
cin>>abc;
temp[j]=abc;
if(nonter[i].length()<=abc.length()&&nonter[i].compare(abc.substr(0,nonter[i].length()))==0)
leftrecr[i]=1;
}
prod.push_back(temp);
}
for(i=0;i<n;++i) {
cout<<leftrecr[i];
}
for(i=0;i<n;++i) {
if(leftrecr[i]==0)
continue;
int j;
nonter.push_back(nonter[i]+"");
vector<string> temp;
for(j=0;j<prod[i].size();++j) {
if(nonter[i].length()<=prod[i][j].length()&&nonter[i].compare(prod[i][j].substr(0,nonter[i].length
()))==0) {
string
abc=prod[i][j].substr(nonter[i].length(),prod[i][j].length()-nonter[i].length()+nonter[i]+"");
temp.push_back(abc);
prod[i].erase(prod[i].begin()+j);
--j;
}
else {
prod[i][j]+=nonter[i]+"";
}
}
temp.push_back("^");
prod.push_back(temp);
}
cout<<"\n\n";
cout<<"\nNew set of non-terminals: ";
for(i=0;i<nonter.size();++i)
cout<<nonter[i]<<" ";
cout<<"\n\nNew set of productions: ";

```

```
for(i=0;i<nonter.size();++i) {  
    int j;  
    for(j=0;j<prod[i].size();++j) {  
        cout<<"\n"<<nonter[i]<<" -> "<<prod[i][j];  
    }  
}  
return 0;  
}
```

OUTPUT :

```
Enter number of non terminals: 3

Enter non terminals one by one:
Non terminal 1 : E

Non terminal 2 : T

Non terminal 3 : F

Enter '^' for null
Number of E productions: 2

One by one enter all E productions
RHS of production 1: E+T

RHS of production 2: T

Number of T productions: 2

One by one enter all T productions
RHS of production 1: T*F

RHS of production 2: F

Number of F productions: 2

One by one enter all F productions
RHS of production 1: (E)

RHS of production 2: i
110

New set of non-terminals: E T F E' T'

New set of productions:
E -> TE'
T -> FT'
F -> (E)
F -> i
E' -> +TE'
E' -> ^
T' -> *FT'
T' -> ^
```

RESULT :

A program for Elimination of Left Recursion was run successfully.

LEFT FACTORING

EX. NO. 4(b)

Anas Ahmed Ather (RA2011031010006)

AIM : A program for implementation Of Left Factoring

ALGORITHM :

1. Start
2. Ask the user to enter the set of productions
3. Check for common symbols in the given set of productions by comparing with:

$A \rightarrow aB1 | aB2$

4. If found, replace the particular productions with:

$A \rightarrow aA'$

$A' \rightarrow B1 | B2 | \epsilon$

5. Display the output
6. Exit

CODE :

```
#include <iostream>
#include <cstring>
#include <cstdlib>

using namespace std;

int main()
{
    char ch, lhs[20][20], rhs[20][20][20], temp[20], temp1[20];
    int n, n1, count[20], x, y, i, j, k, c[20];
    cout << "\nEnter the no. of nonterminals : ";
    cin >> n;
    n1 = n;
    for (i = 0; i < n; i++)
    {
        cout << "\nNonterminal " << i + 1 << " \nEnter the no. of productions : ";
        cin >> c[i];
        cout << "\nEnter LHS : ";
        cin >> lhs[i];
        for (j = 0; j < c[i]; j++)
        {
            cout << lhs[i] << "->";
            cin >> rhs[i][j];
        }
    }
    for (i = 0; i < n; i++)
    {
```

```

    count[i] = 1;
    while (memcmp(rhs[i][0], rhs[i][1], count[i]) == 0)
        count[i]++;
}
for (i = 0; i < n; i++)
{
    count[i]--;
    if (count[i] > 0)
    {
        strcpy(lhs[n1], lhs[i]);
        strcat(lhs[i], "");
        for (k = 0; k < count[i]; k++)
            temp1[k] = rhs[i][0][k];
        temp1[k++] = '\0';
        for (j = 0; j < c[i]; j++)
        {
            for (k = count[i], x = 0; k < strlen(rhs[i][j]); x++, k++)
                temp[x] = rhs[i][j][k];
            temp[x++] = '\0';
            if (strlen(rhs[i][j]) == 1)
                strcpy(rhs[n1][1], rhs[i][j]);
            strcpy(rhs[i][j], temp);
        }
        c[n1] = 2;
        strcpy(rhs[n1][0], temp1);
        strcat(rhs[n1][0], lhs[n1]);
        strcat(rhs[n1][0], "");
        n1++;
    }
}
cout << "\n\nThe resulting productions are : \n";
for (i = 0; i < n1; i++)
{
    if (i == 0)
        cout << "\n " << lhs[i] << " -> " << (char)238 << "|";
    else
        cout << "\n " << lhs[i] << " -> ";
    for (j = 0; j < c[i]; j++)
    {
        cout << " " << rhs[i][j] << " ";
        if ((j + 1) != c[i])
            cout << "|";
    }
    cout << "\b\b\b\n";
}
return 0;
}

```

OUTPUT :

```
Enter the no. of nonterminals : 2

Nonterminal 1
Enter the no. of productions : 3

Enter LHS : S
S->iCtSeS
S->iCtS
S->a

Nonterminal 2
Enter the no. of productions : 1

Enter LHS : C
C->b

The resulting productions are :

S' ->  $\epsilon$  | eS | |

C -> b

S -> iCtSS' | a
```

RESULT : A program for implementation Of Left Factoring was compiled and run successfully