

17-Feb-22

Experiment 4 - Left Recursion and Left Factoring

Dhawal Patil

RA1911003010575

CSE A2

Experiment 4a

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.

Code:

```
#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;
}

```

```

1  #include <iostream>
2  #include <vector>
3  #include <string>
4  using namespace std;
5  int main()
6  {
7      int n;
8      cout<<"\nEnter number of non terminals: ";
9      cin>>n;
10     cout<<"\nEnter non terminals one by one: ";
11     int i;
12     vector<string> nonter(n);
13     vector<int> leftrecr(n,0);
14     for(i=0;i<n;++i) {
15         cout<<"\nNon terminal "<<i+1<<" : ";
16         cin>>nonter[i];
17     }
18     vector<vector<string> > prod;
19     cout<<"\nEnter '^' for null";
20     for(i=0;i<n;++i) {
21         cout<<"\nNumber of "<<nonter[i]<<" productions: ";
22         int k;
23         cin>>k;
24         int j;
25         cout<<"\nOne by one enter all "<<nonter[i]<<" productions";
26         vector<string> temp(k);
27         for(j=0;j<k;++j) {
28             cout<<"\nRHS of production "<<j+1<<": ";
29             string abc;
30             cin>>abc;
31             temp[j]=abc;
32             if(nonter[i].length()<=abc.length()&&nonter[i].compare(abc.substr(0,nonter[i].length()))==0)
33                 leftrecr[i]=1;
34         }
35         prod.push_back(temp);
36     }

```

```

37-   for(i=0;i<n;++i) {
38-       cout<<leftrecr[i];
39-   }
40-   for(i=0;i<n;++i) {
41-       if(leftrecr[i]==0)
42-           continue;
43-       int j;
44-       nonter.push_back(nonter[i]+"");
45-       vector<string> temp;
46-       for(j=0;j<prod[i].size();++j) {
47-           if(nonter[i].length()<=prod[i][j].length()&&nonter[i].compare(prod[i][j].substr(0,nonter[i].length()))==0) {
48-               string abc=prod[i][j].substr(nonter[i].length(),prod[i][j].length()-nonter[i].length()+nonter[i]+"");
49-               temp.push_back(abc);
50-               prod[i].erase(prod[i].begin()+j);
51-               --j;
52-           }
53-           else {
54-               prod[i][j]+=nonter[i]+"";
55-           }
56-       }
57-       temp.push_back("^");
58-       prod.push_back(temp);
59-   }
60-   cout<<"\n\n";
61-   cout<<"\nNew set of non-terminals: ";
62-   for(i=0;i<nonter.size();++i)
63-       cout<<nonter[i]<<" ";
64-   cout<<"\n\nNew set of productions: ";
65-   for(i=0;i<prod.size();++i) {
66-       int j;
67-       for(j=0;j<prod[i].size();++j) {
68-           cout<<"\n"<<nonter[i]<<" -> "<<prod[i][j];
69-       }
70-   }
71-   return 0;
72- }

```

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' -> ^

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

```

Result:

A program for Elimination of Left Recursion was run successfully.

Experiment 4b

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:
 $A \rightarrow aB1 \mid aB2$
4. If found, replace the particular productions with:
 $A \rightarrow aA'$
 $A' \rightarrow B1 \mid B2 \mid \epsilon$
5. Display the output
6. Exit

Code:

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    int n,j,l,i,m;
    int len[10] = {};
    string a, b1, b2, flag;
    char c;
    cout << "Enter the Parent Non-Terminal : ";
    cin >> c;
    a.push_back(c);
    b1 += a + "'->";
    b2 += a + "'\''->";
    a += "->";
    cout << "Enter total number of productions : ";
    cin >> n;
    for (i = 0; i < n; i++)
    {
        cout << "Enter the Production " << i + 1 << " : ";
        cin >> flag;
        len[i] = flag.size();
        a += flag;
        if (i != n - 1)
        {
```

```

        a += "|";
    }
}
cout << "The Production Rule is : " << a << endl;
char x = a[3];
for (i = 0, m = 3; i < n; i++)
{
    if (x != a[m])
    {
        while (a[m++] != '|');
    }
    else
    {
        if (a[m + 1] != '|')
        {
            b1 += "|" + a.substr(m + 1, len[i] - 1);
            a.erase(m - 1, len[i] + 1);
        }
        else
        {
            b1 += "#";
            a.insert(m + 1, 1, a[0]);
            a.insert(m + 2, 1, "\\");
            m += 4;
        }
    }
}
char y = b1[6];
for (i = 0, m = 6; i < n - 1; i++)
{
    if (y == b1[m])
    {
        if (b1[m + 1] != '|')
        {
            flag.clear();
            for (int s = m + 1; s < b1.length(); s++)
            {
                flag.push_back(b1[s]);
            }
            b2 += "|" + flag;
            b1.erase(m - 1, flag.length() + 2);
        }
        else
    }
}

```

```

    {
        b1.insert(m + 1, 1, b1[0]);
        b1.insert(m + 2, 2, '\\');
        b2 += "#";
        m += 5;
    }
}
}
b2.erase(b2.size() - 1);
cout << "After Left Factoring : " << endl;
cout << a << endl;
cout << b1 << endl;
cout << b2 << endl;
return 0;
}

```

```

1  #include <iostream>
2  #include <string>
3  using namespace std;
4  int main()
5  {
6      int n,j,l,i,m;
7      int len[10] = {};
8      string a, b1, b2, flag;
9      char c;
10     cout << "Enter the Parent Non-Terminal : ";
11     cin >> c;
12     a.push_back(c);
13     b1 += a + "\\'->";
14     b2 += a + "\\'''->";
15     a += "->";
16     cout << "Enter total number of productions : ";
17     cin >> n;
18     for (i = 0; i < n; i++)
19     {
20         cout << "Enter the Production " << i + 1 << " : ";
21         cin >> flag;
22         len[i] = flag.size();
23         a += flag;
24         if (i != n - 1)
25         {
26             a += "|";
27         }
28     }
29     cout << "The Production Rule is : " << a << endl;
30     char x = a[3];
31     for (i = 0, m = 3; i < n; i++)
32     {
33         if (x != a[m])
34         {
35             while (a[m++] != '|');
36         }
37     }
38 }

```

```

37     else
38     {
39         if (a[m + 1] != '|')
40         {
41             b1 += "|" + a.substr(m + 1, len[i] - 1);
42             a.erase(m - 1, len[i] + 1);
43         }
44         else
45         {
46             b1 += "#";
47             a.insert(m + 1, 1, a[0]);
48             a.insert(m + 2, 1, '\\');
49             m += 4;
50         }
51     }
52 }
53 char y = b1[c];
54 for (i = 0, m = 6; i < n - 1; i++)
55 {
56     if (y == b1[m])
57     {
58         if (b1[m + 1] != '|')
59         {
60             flag.clear();
61             for (int s = m + 1; s < b1.length(); s++)
62             {
63                 flag.push_back(b1[s]);
64             }
65             b2 += "|" + flag;
66             b1.erase(m - 1, flag.length() + 2);
67         }
68         else
69         {
70             b1.insert(m + 1, 1, b1[0]);
71             b1.insert(m + 2, 2, '\\');
72             b2 += "#";
73             m += 5;
74         }
75     }
76 }
77 b2.erase(b2.size() - 1);
78 cout << "After Left Factoring : " << endl;
79 cout << a << endl;
80 cout << b1 << endl;
81 cout << b2 << endl;
82 return 0;
83 }

```

Output:

```

Enter the Parent Non-Terminal : M
Enter total number of productions : 4
Enter the Production 1 : i
Enter the Production 2 : iM
Enter the Production 3 : (M)
Enter the Production 4 : iM+M
The Production Rule is : M->i|iM|(M)|iM+M
After Left Factoring :
M->iM'| (M)
M' ->#|MM''
M' ->#|+M

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

```

Result:

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