

CONVERSION FROM REGULAR EXPRESSION TO NFA

EX. NO. 2

Anas Ahmed Ather (RA2011031010006)

Date: 24/01/23

AIM: To write a program for converting Regular Expression to NFA.

ALGORITHM:

1. Start
2. Get the input from the user
3. Initialize separate variables and functions for Postfix , Display and NFA
4. Create separate methods for different operators like +, *, .
5. By using Switch case Initialize different cases for the input
6. For ' . ' operator Initialize a separate method by using various stack functions do the same for the other operators like ' * ' and ' + '.
7. Regular expression is in the form like a.b (or) a+b
8. Display the output
9. Stop

PROGRAM :

```
#include<stdio.h>
#include<string.h>
int main()
{
    char reg[20]; int q[20][3],i=0,j=1,len,a,b;
    for(a=0;a<20;a++) for(b=0;b<3;b++) q[a][b]=0;
    printf("Enter the RE:-");
    scanf("%s",reg);
    printf("Given regular expression: %s\n",reg);
    len=strlen(reg);
    while(i<len)
    {
        if(reg[i]=='a'&&reg[i+1]!='|'&&reg[i+1]!='*') { q[j][0]=j+1; j++; }
        if(reg[i]=='b'&&reg[i+1]!='|'&&reg[i+1]!='*') { q[j][1]=j+1; j++; }
        if(reg[i]=='e'&&reg[i+1]!='|'&&reg[i+1]!='*') { q[j][2]=j+1; j++; }
        if(reg[i]=='a'&&reg[i+1]=='|'&&reg[i+2]=='b')
        {
            q[j][2]=((j+1)*10)+(j+3); j++;
            q[j][0]=j+1; j++;
            q[j][2]=j+3; j++;
            q[j][1]=j+1; j++;
            q[j][2]=j+1; j++;
            i=i+2;
        }
    }
}
```

```

    }
    if(reg[i]=='b'&&reg[i+1]=='l'&&reg[i+2]=='a')
    {
        q[j][2]=(j+1)*10+(j+3); j++;
        q[j][1]=j+1; j++;
        q[j][2]=j+3; j++;
        q[j][0]=j+1; j++;
        q[j][2]=j+1; j++;
        i=i+2;
    }
    if(reg[i]=='a'&&reg[i+1]=='*')
    {
        q[j][2]=(j+1)*10+(j+3); j++;
        q[j][0]=j+1; j++;
        q[j][2]=(j+1)*10+(j-1); j++;
    }
    if(reg[i]=='b'&&reg[i+1]=='*')
    {
        q[j][2]=(j+1)*10+(j+3); j++;
        q[j][1]=j+1; j++;
        q[j][2]=(j+1)*10+(j-1); j++;
    }
    if(reg[i]=='')&&reg[i+1]=='*')
    {
        q[0][2]=(j+1)*10+1;
        q[j][2]=(j+1)*10+1;
        j++;
    }
    i++;
}
printf("\n\tTransition Table \n");
printf("_____ \n");
printf("Current State \tInput \tNext State");
printf("\n_____ \n");
for(i=0;i<=j;i++)
{
    if(q[i][0]!=0) printf("\n q[%d]\t | a | q[%d]",i,q[i][0]);
    if(q[i][1]!=0) printf("\n q[%d]\t | b | q[%d]",i,q[i][1]);
    if(q[i][2]!=0)
    {
        if(q[i][2]<10) printf("\n q[%d]\t | e | q[%d]",i,q[i][2]);
        else printf("\n q[%d]\t | e | q[%d] , q[%d]",i,q[i][2]/10,q[i][2]%10);
    }
}
printf("\n_____ \n");
return 0;
}

```

INPUT : (a|b)*abb

OUTPUT :

```
>_ Console x Shell x +
> make -s
> ./main
Enter the RE:-(a|b)*abb
Given regular expression: (a|b)*abb

    Transition Table
-----
Current State | Input | Next State
-----
q[0]          | e     | q[7] , q[1]
q[1]          | e     | q[2] , q[4]
q[2]          | a     | q[3]
q[3]          | e     | q[6]
q[4]          | b     | q[5]
q[5]          | e     | q[6]
q[6]          | e     | q[7] , q[1]
q[7]          | a     | q[8]
q[8]          | b     | q[9]
q[9]          | b     | q[10]
-----
> |
```

RESULT :

The program to convert regular expressions to NFA was implemented successfully.