

Construction of NFA

```
#include<stdio.h>
#include<conio.h>
int Fa[10][10][10],states[2][10],curr,row=0,col=0,sr=0,sc=0,th=0,in;
char *str;
int nfa(char *string,int state)
{
    int i,j;
    for(i=0;i<=row;i++)
    {
        if(*string)
        {
            curr=Fa[state][*string-97][i];

            if (curr==-1)
                break;
            if(nfa(string+1,curr))
                return 1;
        }
        else
        {
            if(states[1][i]==-1)
                break;
            if(state==states[1][i])
                return 1;
        }
    }
    return 0;
}
int main()
{
    FILE *fp;
    int i,j,k,flag=0;
    char c,ch;
    clrscr();
    fp=fopen("Nfa_ip.txt","r");

    for(i=0;i<2;i++)
    for(j=0;j<10;j++)
        states[i][j]=-1;
```

```

for(i=0;i<10;i++)
for(j=0;j<10;j++)
    for(k=0;k<10;k++)
        Fa[i][j][k]=-1;

while(fscanf(fp,"%d",&in)!=EOF)
{
    fscanf(fp,"%c",&c);

```

```

        if(flag)
        {
            states[sr][sc++]=in;
            if(c=='\n')
            {
                sr++;
                sc=0;
            }
        }
        else if(c=='#')
        {
            flag=1;
            Fa[row][col][th]=in;
            printf("\nFa[%d][%d][%d]=%d",row,col,th,Fa[row][col][th]);
        }
        else if(!flag)
        {
            Fa[row][col][th]=in;
            printf("\nFa[%d][%d][%d]=%d",row,col,th,Fa[row][col][th]);
            if(c==',')
            {
                th++;
            }
        }
        else if(c=='\n')
        {
            col=0;
            row++;
            th=0;
        }
        else if(c!='.')

```

```

        {
            col++;
            th=0;
        }
    }
}

printf("\n\nEnter the string : \n");
scanf("%s",str);
if(nfa(str,states[0][0]))
    printf("\nString Is Accepted");
else
    printf("\nString Not Accepted");
getch();
return 0;
}

```

SAMPLE OUTPUT:

Enter the string :

1,2 1

STRING ACCEPTED

Enter the string :

a 1 2 2

STRING NOT ACCEPTED

Ex.No:2 Construction of Minimized DFA

AIM : Write a program to Convert regular expression into DFA

ALGORITHM:

STEP1: Declare the necessary variables such as €,a, union, concatenation, kleen closure, parenthesis.

STEP2: Define the rule for €, and draw the transition table.

STEP3: Define the rule for a and draw the transition table.

STEP4: Define the rule for union and draw the transition table.

STEP5: Define the rule for concatenation and draw the transition table.

STEP6: Define the rule for kleen closure and draw the transition table.

STEP7: Define the rule for parenthesis(a) and draw the transition table.

STEP8: Combine the entire transition table.

STEP9: Display the result of DFA

PROGRAM

```
#include<stdio.h>
#include<conio.h>
int main()
{
    FILE * fp;
    int Fa[10][10],states[2][10],row=0,col=0,sr=0,sc=0,flag=0,i,j,in,curr;
    char k,*str;
    clrscr();
    fp = fopen("Dfa_ip.txt","r");
    if(fp==NULL)
        printf("file could not find\n");

    for(i=0;i<3;i++)
        for(j=0;j<10;j++)
            states[i][j]=-1;

    while(fscanf(fp,"%d",&in)!=EOF)
    {
        fscanf(fp,"%c",&k);

        if (flag)
        {
```

```

        states[sr][sc++]=in;
        if(k=='\n')
        {
            sr++;
            sc=0;
        }
    }
    else if(k=='#')
    {
        flag=1;
        Fa[row][col++]=in;
    }
    else if(!flag)
    {
        Fa[row][col++]=in;
        if(k=='\n')
        {
            row++;
            col=0;
        }
    }
}

```

```

printf("THE AUTOMATA IS : \n\n");
for (i=0;i<=row;i++)
{
    for (j=0;j<col;j++)
    {
        printf("%2d ",Fa[i][j]);
    }
    printf("\n");
}

```

```

printf("\n\nEnter the string : ");
gets(str);
curr=states[0][0];
i=0;
while(str[i]!='\0')
{
    curr=Fa[curr][str[i]-97];
}

```

```

        if(curr==-1)
            break;
        i++;
    }

    flag=0;
    if(curr!=-1)
    {
        for(i=0;i<=sc&&!flag;i++)
        {
            if(curr==states[1][i])
            {
                printf("\n\nSTRING ACCEPTED\n");
                flag=1;
                break;
            }
        }
    }
    if(flag==0)
        printf("\n\nSTRING NOT ACCEPTED ");
    getch();
    return 0;

}

```

SAMPLE OUTPUT:

Enter the string : 1 3

STRING ACCEPTED

Enter the string : 5 5#

STRING ACCEPTED

Enter the string : a 1 3

STRING NOT ACCEPTED

RESULT : Thus the program has been executed successfully and verified.

Ex.no.3 Implementation of Lexical Analyzer Using LexTool

AIM: To write a program for lexical analyzer to using lex tool

ALGORITHM:

STEP 1: Open gedit text editor from accessories in applications

STEP 2: Specify the header files to be included inside the declaration part (i.e. between %{ and %})

STEP 3: Define the digits i.e. 0-9 and identifiers a-z and a-z 0-9

STEP 4: Using translation rule, we defined the regular expression for digit, if it is matched with the given input then store and display it as digit in yytext.

STEP 5: Using translation rule, we defined the regular expression for keywords, if it is matched with the given input then store and display it as keyword in yytext.

STEP 6: Using translation rule, we defined the regular expression for identifiers, if it is matched with the given input then store and display it as identifier in yytext.

STEP 7: Using translation rule, we defined the regular expression for operators, if it is matched with the given input then store and display it as operator in yytext.

STEP 8: Inside procedure main(), use yyin() to point the current file being parsed by the lexer.

STEP 9: The call yylex(), which starts the analysis.

PROGRAM:

```
%{  
#include<math.h>  
#include<stdlib.h>  
%}  
  
DIGIT [0-9]  
ID    [a-z][a-z 0-9]*
```


%%

```
{DIGIT}+
{
printf("An integer:%s(%d)\n",yytext,atoi(yytext));
}
{DIGIT}+"."{DIGIT}*
{
printf("A float:%s(%g)\n",yytext,atof(yytext));
}
```

```
if|then|begin|end|procedure|function
{
printf("A keyboard:%s\n",yytext);
}
{ID}
printf("An identifier:%s\n",yytext);
```

```
"+"|"-"|"*"|"|"/"
printf("An operator:%s\n",yytext);
```

```
[\t\n]+
printf("Unrecognized chareacter:%s\n",yytext);
%%
```

```
main(argc,argv)
int argc;
char **argv;
{
++argv,--argc;
if(argc > 0)
yyin=fopen(argv[0],"r");
else
yyin=stdin;
yylex();
}
```

FILE PROGRAM:

ram is a good boy
sita is a good girl
gowtham is a clever boy
geetha is a smart girl
 $a+b*c/d=e$.

SAMPLE OUTPUT:

An identifier:ram is a good boy
An identifier:sita is a good girl
An identifier:gowtham is a clever boy
An identifier:geetha is a smart girl
An identifier:a
An operator:+
An identifier:b
An operator:*
An identifier:c
An operator:/
An identifier:d
Unrecognized chareacter:=

RESULT : Thus the program has been executed successfully and verified.