CONVERSION FROM NFA TO DFA

NAME: **RISHAL RAMESH** ***EXP NO:*** **3**

REG NO: **RA1911030010084**

DATE: **03/02/2022**

AIM:

To study and perform NFA (non deterministic automata to DFA (deterministic automata) conversion in any of the programming languages.

LANGUAGE USED:

C++

ALGORITHM:

- Start
- Get the input from the user.
- Set the only state in SDFA to “unmarked”.
- While SDFA contains an unmarked state do:
  a) Let T be that unmarked state
  b) For each a in % do S=e-Closure(MoveNFA(T,a))
  c) If S is not in the SDFA already then, add S to SDFA(as an “unmarked state”)
  d) Set MoveDFA(T,a) to S
- For each S in SDFA if any s &S us a final state in NFA then, mark S as a final state in the DFA.
- Print the result
- Stop the program

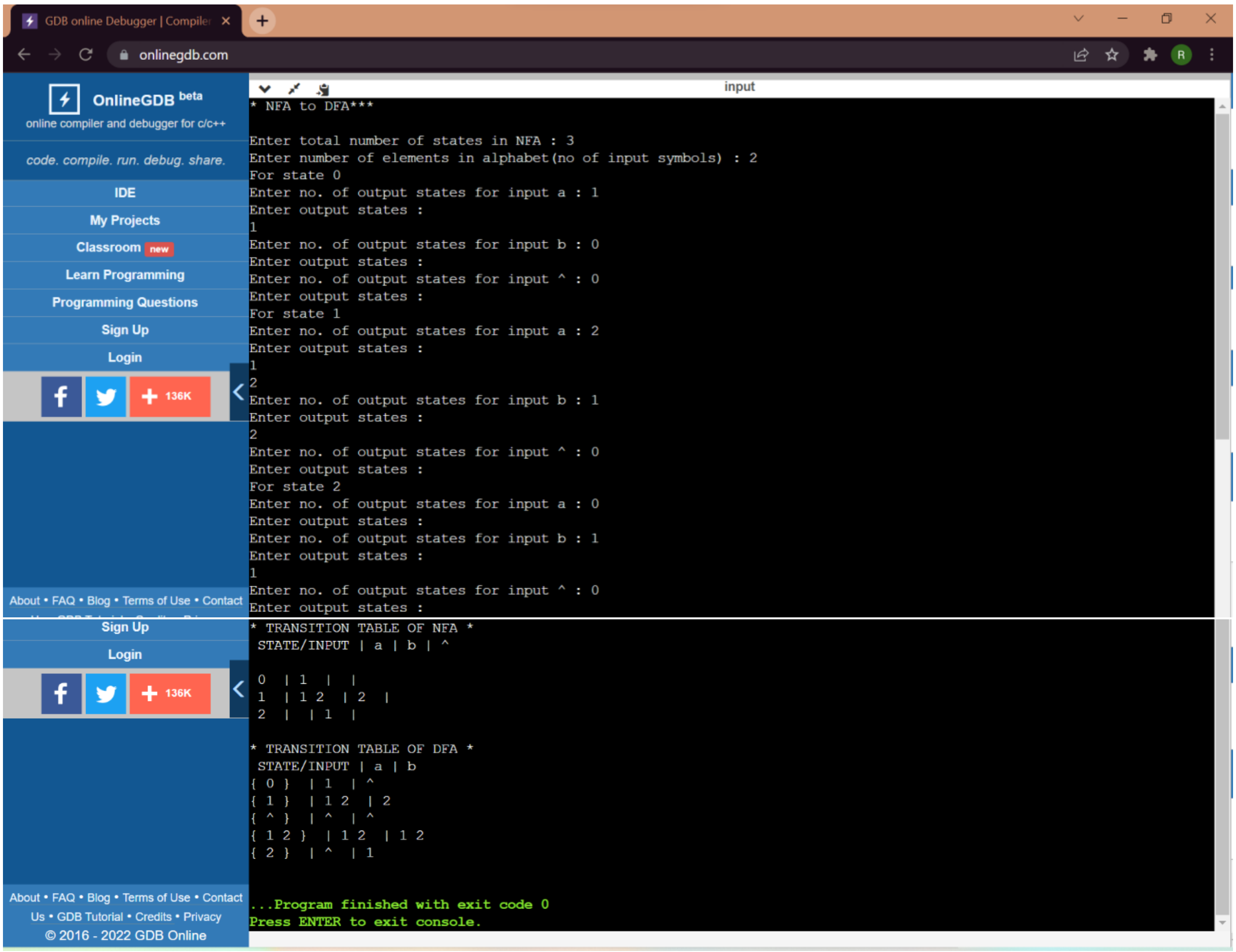
CODE:

```
#include <iostream>
#include <bits/stdc++.h>
using namespace std;
void print(vector<vector<vector<int>>> table)
{
    cout << " STATE/INPUT |";
    char a = 'a';
    for (int i = 0; i < table[0].size() - 1; i++)
    {
        cout << " " << a++ << " |";
    }
    cout << " ^ " << endl
         << endl;
    for (int i = 0; i < table.size(); i++)
    {
        cout << " " << i << " ";
        for (int j = 0; j < table[i].size(); j++)
        {
            cout << " | ";
            for (int k = 0; k < table[i][j].size(); k++)
            {
                cout << table[i][j][k] << " ";
            }
        }
        cout << endl;
    }
}
void printdfa(vector<vector<int>> states, vector<vector<vector<int>>> dfa)
{
```

```
cout << " STATE/INPUT ";
char a = 'a';
for (int i = 0; i < dfa[0].size(); i++)
{
    cout << "| " << a++ << " ";
}
cout << endl;
for (int i = 0; i < states.size(); i++)
{
    cout << "{ ";
    for (int h = 0; h < states[i].size(); h++)
        cout << states[i][h] << " ";
    if (states[i].empty())
    {
        cout << "^ ";
    }
    cout << "} ";
    for (int j = 0; j < dfa[i].size(); j++)
    {
        cout << " | ";
        for (int k = 0; k < dfa[i][j].size(); k++)
        {
            cout << dfa[i][j][k] << " ";
        }
        if (dfa[i][j].empty())
        {
            cout << "^ ";
        }
    }
    cout << endl;
    }
}
vector<int> closure(int s, vector<vector<vector<int>>> v)
{
    vector<int> t;
    queue<int> q;
    t.push_back(s);
    int a = v[s][v[s].size() - 1].size();
    for (int i = 0; i < a; i++)
    {
        t.push_back(v[s][v[s].size() - 1][i]);
        // cout<<"t[i]"<<t[i]<<endl;
        q.push(t[i]);
    }
    while (!q.empty())
    {
        int f = q.front();
        q.pop();
        if (!v[f][v[f].size() - 1].empty())
        {
            int u = v[f][v[f].size() - 1].size();
            for (int i = 0; i < u; i++)
            {
                int y = v[f][v[f].size() - 1][i];
                if (find(t.begin(), t.end(), y) == t.end())
                {
                    // cout<<"y"<<y<<endl;
                    t.push_back(y);
                    q.push(y);
                }
            }
        }
    }
    return t;
}
int main()
{
    int n, alpha;
    cout << "* NFA to DFA***" << endl
         << endl;
    cout << "Enter total number of states in NFA : ";
    cin >> n;
    cout << "Enter number of elements in alphabet(no of input symbols) : ";
    cin >> alpha;
    vector<vector<vector<int>>> table;
    for (int i = 0; i < n; i++)
    {
        cout << "For state " << i << endl;
        vector<vector<int>> v;
        char a = 'a';
        int y, yn;
        for (int j = 0; j < alpha; j++)
        {
            vector<int> t;
            cout << "Enter no. of output states for input " << a++ << " : ";
            cin >> yn;
            cout << "Enter output states :" << endl;
            for (int k = 0; k < yn; k++)
            {
                cin >> y;
                t.push_back(y);
            }
            v.push_back(t);
        }
        vector<int> t;
        cout << "Enter no. of output states for input ^ : ";
        cin >> yn;
        cout << "Enter output states :" << endl;
        for (int k = 0; k < yn; k++)
        {
            cin >> y;
            t.push_back(y);
        }
        v.push_back(t);
        table.push_back(v);
    }
    cout << "* TRANSITION TABLE OF NFA *" << endl;
    print(table);
    cout << endl
         << "* TRANSITION TABLE OF DFA *" << endl;
    vector<vector<vector<int>>> dfa;
    vector<vector<int>> states;
    states.push_back(closure(0, table));
    queue<vector<int>> q;
    q.push(states[0]);
    while (!q.empty())
    {
        vector<int> f = q.front();
        q.pop();
        vector<vector<int>> v;
```

```
        for (int i = 0; i < alpha; i++)
        {
            vector<int> t;
            set<int> s;
            for (int j = 0; j < f.size(); j++)
            {
                for (int k = 0; k < table[f[j]][i].size(); k++)
                {
                    vector<int> cl = closure(table[f[j]][i][k], table);
                    for (int h = 0; h < cl.size(); h++)
                    {
                        if (s.find(cl[h]) == s.end())
                            s.insert(cl[h]);
                    }
                }
            }
            for (set<int>::iterator u = s.begin(); u != s.end(); u++)
                t.push_back(*u);
            v.push_back(t);
            if (find(states.begin(), states.end(), t) == states.end())
            {
                states.push_back(t);
                q.push(t);
            }
        }
        dfa.push_back(v);
    }
    printdfa(states, dfa);
}
```

OUTPUT:


GDB online Debugger | Compiler
onlinegdb.com
OnlineGDB beta
online compiler and debugger for c/c++
code. compile. run. debug. share.
IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Sign Up
Login
136K
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
© 2016 - 2022 GDB Online
input

```
* NFA to DFA***

Enter total number of states in NFA : 3
Enter number of elements in alphabet(no of input symbols) : 2
For state 0
Enter no. of output states for input a : 1
Enter output states :
1
Enter no. of output states for input b : 0
Enter output states :
Enter no. of output states for input ^ : 0
Enter output states :
For state 1
Enter no. of output states for input a : 2
Enter output states :
1
2
Enter no. of output states for input b : 1
Enter output states :
2
Enter no. of output states for input ^ : 0
Enter output states :
For state 2
Enter no. of output states for input a : 0
Enter output states :
Enter no. of output states for input b : 1
Enter output states :
1
Enter no. of output states for input ^ : 0
Enter output states :
```

```
* TRANSITION TABLE OF NFA *
 STATE/INPUT | a | b | ^

 0  | 1  |  |
 1  | 1 2  | 2  |
 2  |  | 1  |

* TRANSITION TABLE OF DFA *
 STATE/INPUT | a | b
{ 0 }  | 1  | ^
{ 1 }  | 1 2  | 2
{ ^ }  | ^  | ^
{ 1 2 }  | 1 2  | 1 2
{ 2 }  | ^  | 1


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

RESULT:

NFA to DFA conversion was successfully executed in C++.