

## 16) DFA Construction and Checking the string acceptance

Code:

```
#include<stdio.h>

#include<string.h>

#define max 20

int main() {

    int t[4][2] = {{1, 3}, {1, 2}, {1, 2}, {3, 3}};

    int f = 2, p = 0;

    char s[max];

    printf("Enter the String 's' : ");

    scanf("%s", s);

    int str_len = strlen(s);

    for(int i = 0; i < str_len; i++) {

        int i_s = (s[i] == 'a') ? 0 : ((s[i] == 'b') ? 1 : -1);

        if(i_s == -1){

            printf("Invalid Input \n");

            return 0;

        }

        p = t[p][i_s];

    }

    if(p == f){

        printf("The DFA Accepts the string \n");

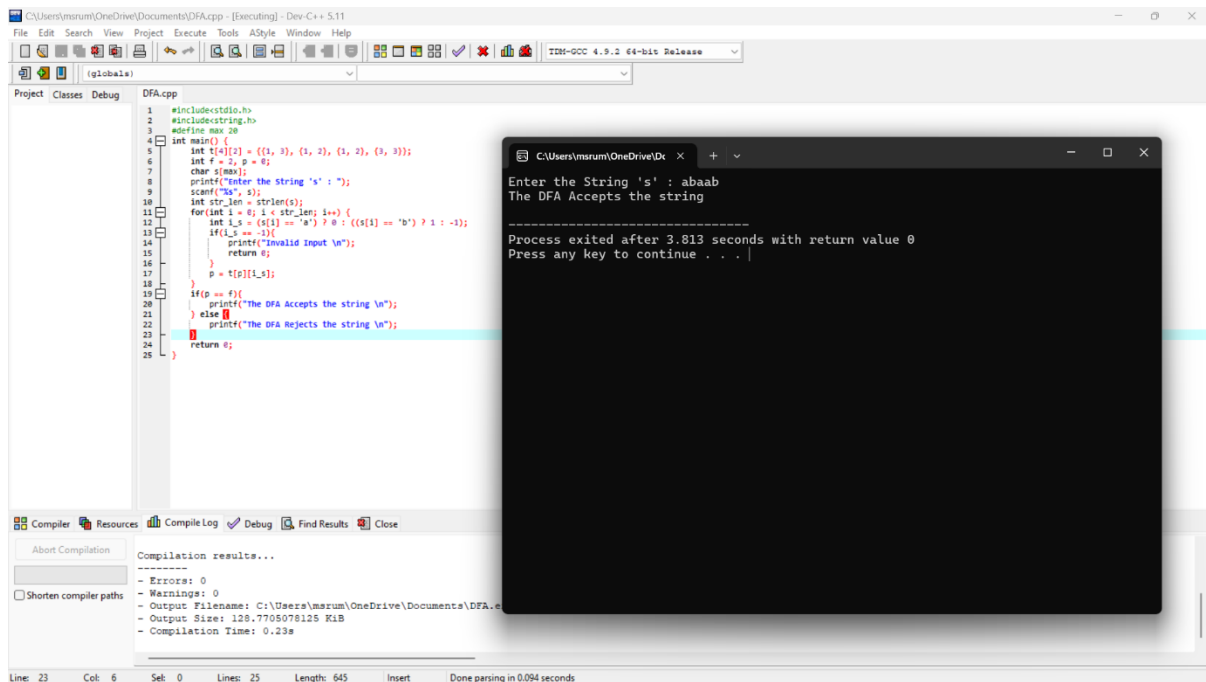
    } else {

        printf("The DFA Rejects the string \n");

    }

    return 0;

}
```



## 17) NFA Construction and Checking the string acceptance

```
#include<stdio.h>
```

```
#include<string.h>
```

```
#define max 20
```

```
int main() {
```

```
    int t[4][2][2] = {{1, 2}, {1, 3}}, {{1, 2}, {1, 2}}, {{3, -1}, {3, -1}}, {{3, 3}, {3, 3}};
```

```
    int f = 3, p[max] = {0}, p_count = 1, next_states[max], i_s;
```

```
    char s[max];
```

```
    printf("Enter the String 's' : ");
```

```
    scanf("%s", s);
```

```
    for (int i = 0, str_len = strlen(s); i < str_len; i++) {
```

```
        i_s = (s[i] == 'a') ? 0 : ((s[i] == 'b') ? 1 : -1);
```

```
        if (i_s == -1) { printf("Invalid Input \n"); return 0; }
```

```
        int next_count = 0;
```

```
        for (int j = 0; j < p_count; j++) {
```

```
            for (int k = 0; k < 2; k++) {
```

```
                int new_state = t[p[j]][i_s][k];
```

```
                if (new_state != -1) {
```

```

    int found = 0;

    for (int l = 0; l < next_count; l++) {

        if (next_states[l] == new_state) { found = 1; break; }

    }

    if (!found) { next_states[next_count++] = new_state; }

}

}

}

p_count = next_count;

memcpy(p, next_states, p_count * sizeof(int));

}

int accepts = 0;

for (int i = 0; i < p_count; i++) { if (p[i] == f) { accepts = 1; break; } }

printf("The NFA %s the string \n", accepts ? "Accepts" : "Rejects");

return 0;

}

```

The screenshot displays a C++ IDE with the following components:

- Editor:** Shows the source code for `DFA.cpp`. The code defines a transition table for an NFA with 4 states and 2 characters ('a' and 'b'). It includes a `main` function that reads a string `s` and checks if it is accepted by the NFA.
- Compiler:** Shows the compilation results, indicating that the program compiled successfully with 0 errors and 0 warnings.
- Debug Console:** Shows the output of the program for two test cases:
  - Test Case 1: Input string is `aban`. The output is `Invalid Input`.
  - Test Case 2: Input string is `aaabaa`. The output is `The NFA Accepts the string`.

## 18) PDA $0^n 1^n$

```
#include <stdio.h>
#include <string.h>

char stack[20];
int top = 0;

void push() {
    stack[++top] = '0';
    stack[top + 1] = '\0';
}

int pop() {
    if (top < 1)
        return 0;
    else {
        stack[top--] = '\0';
        return 1;
    }
}

int main() {
    char input[20];
    int i, len;

    printf("Simulation of Pushdown Automaton for  $0^n 1^n$ \n");
    printf("Enter a string : ");
    scanf("%s", input);
    stack[top] = 'Z';
    printf("Stack\tInput\n");
    printf("%s\t%s\n", stack, input);
    len = strlen(input);
    for (i = 0; i < len; i++) {
        if (input[i] != '0' && input[i] != '1') {
            printf("Invalid input\n");
        }
    }
}
```

```

        return 0;
    }
}

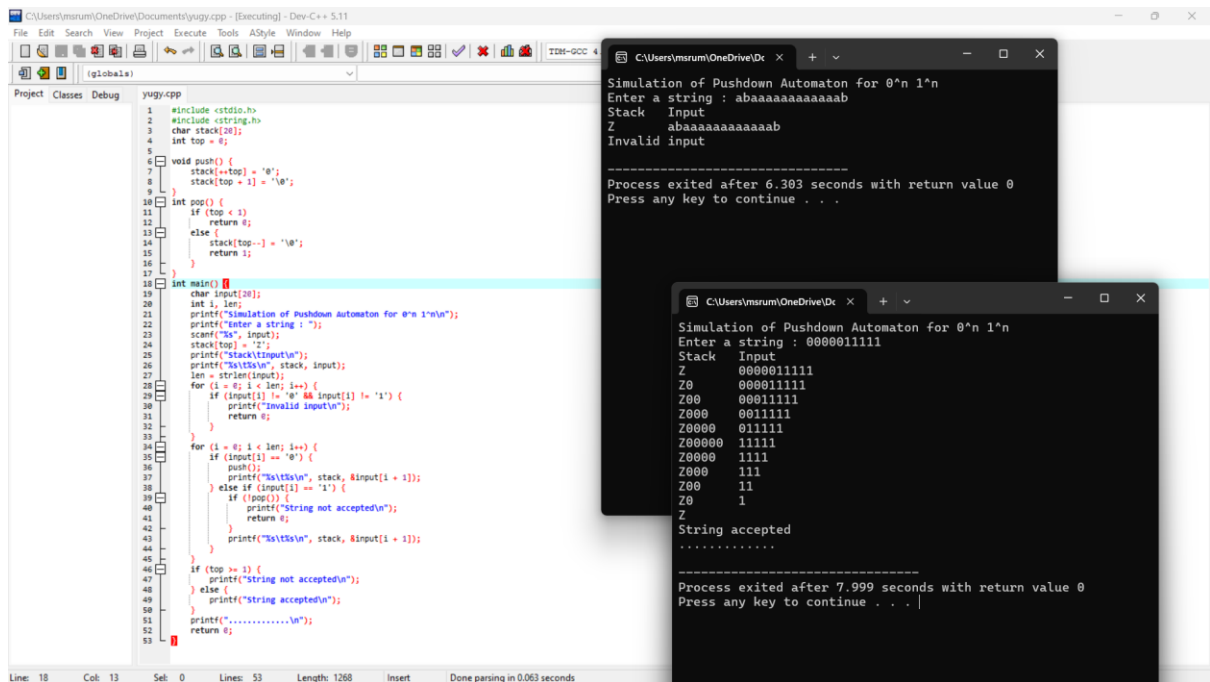
for (i = 0; i < len; i++) {
    if (input[i] == '0') {
        push();
        printf("%s\t%s\n", stack, &input[i + 1]);
    } else if (input[i] == '1') {
        if (!pop()) {
            printf("String not accepted\n");
            return 0;
        }
        printf("%s\t%s\n", stack, &input[i + 1]);
    }
}

if (top >= 1) {
    printf("String not accepted\n");
} else {
    printf("String accepted\n");
}

printf(".....\n");

return 0;
}

```



## 19) Turing Machine for $0^n 1^n$

```

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define TAPE_SIZE 1000

#define STATE_START 0

#define STATE_READ_0 1

#define STATE_READ_1 2

#define STATE_ACCEPT 3

#define STATE_REJECT 4

#define SYMBOL_BLANK '_'

#define SYMBOL_TAPE '*'

#define SYMBOL_0 '0'

#define SYMBOL_1 '1'

struct TuringMachine {
    char tape[TAPE_SIZE];

    int head;

    int state;

```

```

};

void initialize(struct TuringMachine *machine, const char *input) {
    memset(machine->tape, SYMBOL_BLANK, sizeof(machine->tape));
    strncpy(machine->tape, input, strlen(input));
    machine->head = 0;
    machine->state = STATE_START;
}

void moveLeft(struct TuringMachine *machine) {
    machine->head--;
}

void moveRight(struct TuringMachine *machine) {
    machine->head++;
}

void step(struct TuringMachine *machine) {
    switch (machine->state) {
        case STATE_START:
            if (machine->tape[machine->head] == SYMBOL_0) {
                machine->tape[machine->head] = SYMBOL_TAPE;
                moveRight(machine);
                machine->state = STATE_READ_0;
            } else if (machine->tape[machine->head] == SYMBOL_BLANK) {
                machine->state = STATE_ACCEPT;
            } else {
                machine->state = STATE_REJECT;
            }
            break;
        case STATE_READ_0:
            if (machine->tape[machine->head] == SYMBOL_0) {
                moveRight(machine);
            } else if (machine->tape[machine->head] == SYMBOL_1) {
                machine->state = STATE_READ_1;
                moveRight(machine);
            }
    }
}

```

```

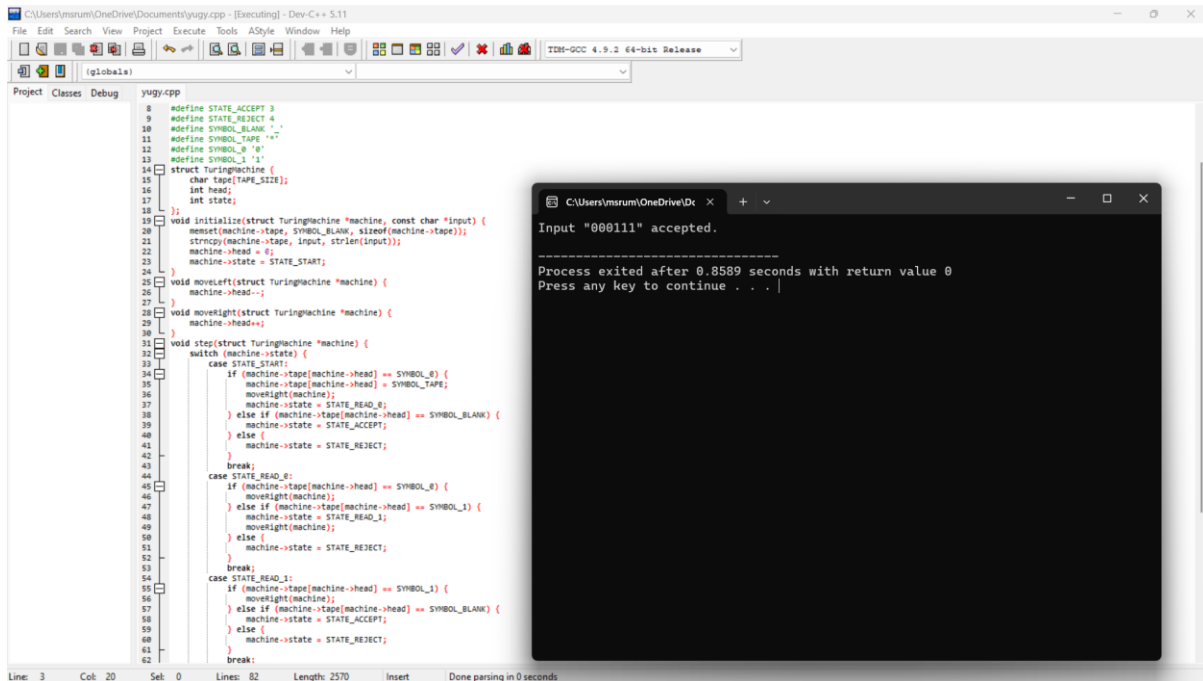
    } else {
        machine->state = STATE_REJECT;
    }
    break;
case STATE_READ_1:
    if (machine->tape[machine->head] == SYMBOL_1) {
        moveRight(machine);
    } else if (machine->tape[machine->head] == SYMBOL_BLANK) {
        machine->state = STATE_ACCEPT;
    } else {
        machine->state = STATE_REJECT;
    }
    break;
case STATE_ACCEPT:
    break;
case STATE_REJECT:
    break;
}
}

int main() {
    struct TuringMachine machine;
    const char *input = "000111";
    initialize(&machine, input);
    while (machine.state != STATE_ACCEPT && machine.state != STATE_REJECT) {
        step(&machine);
    }
    if (machine.state == STATE_ACCEPT) {
        printf("Input \"%s\" accepted.\n", input);
    } else {
        printf("Input \"%s\" rejected.\n", input);
    }
    return 0;
}

```



}



The screenshot shows a C++ IDE with a project named 'yugi.cpp'. The code defines a Turing Machine with states: STATE\_ACCEPT, STATE\_REJECT, STATE\_READ\_0, STATE\_READ\_1, and STATE\_READ\_1\_SECOND. It also defines symbols: SYMBOL\_BLANK, SYMBOL\_TAPE, SYMBOL\_0, and SYMBOL\_1. The machine's logic is implemented in the 'step' function, which uses a switch statement to handle different states and tape symbols. The execution output shows that the input '000111' was accepted, and the process exited after 0.8589 seconds with a return value of 0.

```
15 #define STATE_ACCEPT 2
16 #define STATE_REJECT 4
17 #define SYMBOL_BLANK ' '
18 #define SYMBOL_TAPE '*'
19 #define SYMBOL_0 '0'
20 #define SYMBOL_1 '1'
21 struct TuringMachine {
22     char tape[TAPE_SIZE];
23     int head;
24     int state;
25 };
26 void initialize(struct TuringMachine *machine, const char *input) {
27     memset(machine->tape, SYMBOL_BLANK, sizeof(machine->tape));
28     strcpy(machine->tape, input, strlen(input));
29     machine->head = 0;
30     machine->state = STATE_START;
31 }
32 void moveleft(struct TuringMachine *machine) {
33     machine->head--;
34 }
35 void moveright(struct TuringMachine *machine) {
36     machine->head++;
37 }
38 void step(struct TuringMachine *machine) {
39     switch (machine->state) {
40     case STATE_START:
41         if (machine->tape[machine->head] == SYMBOL_0) {
42             machine->tape[machine->head] = SYMBOL_TAPE;
43             moveright(machine);
44             machine->state = STATE_READ_0;
45         } else if (machine->tape[machine->head] == SYMBOL_BLANK) {
46             machine->state = STATE_ACCEPT;
47         } else {
48             machine->state = STATE_REJECT;
49         }
50         break;
51     case STATE_READ_0:
52         if (machine->tape[machine->head] == SYMBOL_0) {
53             moveright(machine);
54         } else if (machine->tape[machine->head] == SYMBOL_1) {
55             machine->state = STATE_READ_1;
56             moveright(machine);
57         } else {
58             machine->state = STATE_REJECT;
59         }
60         break;
61     case STATE_READ_1:
62         if (machine->tape[machine->head] == SYMBOL_1) {
63             moveright(machine);
64         } else if (machine->tape[machine->head] == SYMBOL_BLANK) {
65             machine->state = STATE_ACCEPT;
66         } else {
67             machine->state = STATE_REJECT;
68         }
69         break;
70     case STATE_READ_1_SECOND:
71         if (machine->tape[machine->head] == SYMBOL_1) {
72             moveright(machine);
73         } else if (machine->tape[machine->head] == SYMBOL_BLANK) {
74             machine->state = STATE_ACCEPT;
75         } else {
76             machine->state = STATE_REJECT;
77         }
78         break;
79     }
80 }
```

Input "000111" accepted.

Process exited after 0.8589 seconds with return value 0  
Press any key to continue . . .

## 20) Turing Machine for $0^n 1^{2n}$

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <string.h>
```

```
#define TAPE_SIZE 1000
```

```
#define STATE_START 0
```

```
#define STATE_READ_0 1
```

```
#define STATE_READ_1 2
```

```
#define STATE_READ_1_SECOND 3
```

```
#define STATE_ACCEPT 4
```

```
#define STATE_REJECT 5
```

```
#define SYMBOL_BLANK ' '
```

```
#define SYMBOL_TAPE '*'
```

```
#define SYMBOL_0 '0'
```

```
#define SYMBOL_1 '1'
```

```
struct TuringMachine {
```

```
    char tape[TAPE_SIZE];
```

```

    int head;

    int state;
};

void initialize(struct TuringMachine *machine, const char *input) {
    memset(machine->tape, SYMBOL_BLANK, sizeof(machine->tape));
    strncpy(machine->tape, input, strlen(input));
    machine->head = 0;
    machine->state = STATE_START;
}

void moveLeft(struct TuringMachine *machine) {
    machine->head--;
}

void moveRight(struct TuringMachine *machine) {
    machine->head++;
}

void step(struct TuringMachine *machine) {
    switch (machine->state) {
        case STATE_START:
            if (machine->tape[machine->head] == SYMBOL_0) {
                machine->tape[machine->head] = SYMBOL_TAPE;
                moveRight(machine);
                machine->state = STATE_READ_0;
            } else if (machine->tape[machine->head] == SYMBOL_BLANK) {
                machine->state = STATE_ACCEPT;
            } else {
                machine->state = STATE_REJECT;
            }
            break;
        case STATE_READ_0:
            if (machine->tape[machine->head] == SYMBOL_0) {

```

```

        moveRight(machine);
    } else if (machine->tape[machine->head] == SYMBOL_1) {
        machine->state = STATE_READ_1;
        moveRight(machine);
    } else {
        machine->state = STATE_REJECT;
    }
    break;
case STATE_READ_1:
    if (machine->tape[machine->head] == SYMBOL_1) {
        moveRight(machine);
    } else if (machine->tape[machine->head] == SYMBOL_BLANK) {
        machine->state = STATE_REJECT;
    } else {
        machine->state = STATE_READ_1_SECOND;
        moveRight(machine);
    }
    break;
case STATE_READ_1_SECOND:
    if (machine->tape[machine->head] == SYMBOL_1) {
        moveRight(machine);
    } else if (machine->tape[machine->head] == SYMBOL_BLANK) {
        machine->state = STATE_ACCEPT;
    } else {
        machine->state = STATE_REJECT;
    }
    break;
case STATE_ACCEPT:
    break;
case STATE_REJECT:

```

```

        break;
    }
}

int main() {
    struct TuringMachine machine;

    const char *input = "000111";

    initialize(&machine, input);

    while (machine.state != STATE_ACCEPT && machine.state != STATE_REJECT) {
        step(&machine);
    }

    if (machine.state == STATE_ACCEPT) {
        printf("Input \"%s\" accepted.\n", input);
    } else {
        printf("Input \"%s\" rejected.\n", input);
    }

    return 0;
}

```

The screenshot shows a C++ IDE with the source code of a Turing Machine implementation. The code defines a TuringMachine struct with tape, head, and state, and implements functions for initialization, movement, and stepping. The main function tests the machine with the input "000111".

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #define TAPE_SIZE 1000
5 #define STATE_START 0
6 #define STATE_READ_0 1
7 #define STATE_READ_1 2
8 #define STATE_READ_1_SECOND 3
9 #define STATE_ACCEPT 4
10 #define STATE_REJECT 5
11 #define SYMBOL_BLANK '-'
12 #define SYMBOL_TAPE '+'
13 #define SYMBOL_0 '0'
14 #define SYMBOL_1 '1'
15 struct TuringMachine {
16     char tape[TAPE_SIZE];
17     int head;
18     int state;
19 };
20 void initialize(struct TuringMachine *machine, const char *input) {
21     memset(machine->tape, SYMBOL_BLANK, sizeof(machine->tape));
22     strcpy(machine->tape, input, strlen(input));
23     machine->head = 0;
24     machine->state = STATE_START;
25 }
26 void moveLeft(struct TuringMachine *machine) {
27     machine->head--;
28 }
29 void moveRight(struct TuringMachine *machine) {
30     machine->head++;
31 }
32 void step(struct TuringMachine *machine) {
33     switch (machine->state) {
34     case STATE_START:
35         if (machine->tape[machine->head] == SYMBOL_0) {
36             machine->tape[machine->head] = SYMBOL_TAPE;
37             moveRight(machine);
38             machine->state = STATE_READ_0;
39         } else if (machine->tape[machine->head] == SYMBOL_BLANK) {
40             machine->state = STATE_ACCEPT;
41         } else {
42             machine->state = STATE_REJECT;
43             break;
44         }
45     case STATE_READ_0:
46         if (machine->tape[machine->head] == SYMBOL_0) {
47             moveRight(machine);
48         } else if (machine->tape[machine->head] == SYMBOL_1) {
49             machine->state = STATE_READ_1;
50             moveRight(machine);
51         } else {
52             machine->state = STATE_REJECT;
53             break;
54         }
55     case STATE_READ_1:

```

The execution output window shows the following text:

```

Input "000111" rejected.
-----
Process exited after 0.3512 seconds with return value 0
Press any key to continue . . .

```