

<b>SPRAWOZDANIE</b>		<b>Data wykonania:</b> 28/11/2018
<b>Tytuł zadania:</b>	<b>Wykonał:</b>	<b>Sprawdził:</b>
<i>Symulacja automatu skończonego</i>	<i>Krzysztof Maciejewski</i>  299262	<i>dr inż. Konrad Markowski</i>

## Spis treści

1. Cel projektu .....	1
2. Teoria.....	2
3. Szczegóły implementacyjne .....	2
4. Sposób wywołania programu.....	4
4.1. Brak podania nazwy pliku do odczytu .....	4
4.2. Nie można otworzyć podanego pliku .....	5
4.3. Odczytano zły typ danej wejściowej.....	5
4.4. Prawidłowe dane wejściowe, zły stan końcowy.....	5
4.5. Prawidłowe dane wejściowe, prawidłowy stan końcowy .....	6
4.6. Ilość danych wejściowych przekracza określone maksimum .....	6
5. Wnioski i spostrzeżenia .....	7

## 1. Cel projektu

Celem naszego projektu było napisanie w języku C programu symulującego działanie automatu skończonego, zgodnego z założeniami otrzymanego zadania. Ja otrzymałem zadanie nr. 19.

### Zadanie 19

Napisać program symulujący działanie automatu skończonego  $M=(Q, \Sigma, \delta, q_0, F)$ , gdzie

$$Q=\{q_0, q_1, q_2, q_3\}, \Sigma=\{0, 1\}, F=\{q_3\}$$

$\delta$	Wejścia	
Stany	0	1
$q_0$	$q_1$	$q_2$
$q_1$	$q_3$	$q_0$
$q_2$	$q_0$	$q_3$
$q_3$	$q_1$	$q_2$

Symulator powinien dla ciągów składających się z symboli wejściowych rysować diagram przejść i zaznaczać aktualny stan przy wczytywanych kolejnych symbolach z ciągu. Po wczytaniu całego ciągu program powinien wyświetlić komunikat czy ciąg został zaakceptowany?

## 2. Teoria

W zadaniu mamy do czynienia z deterministycznym automatem skończonym.

**Deterministyczny automat skończony** (skrót: **DAS**) to abstrakcyjna maszyna o skończonej liczbie stanów. W danej chwili może znajdować się w jednym ze skończonej liczby stanów. Na początku znajduje się w stanie określanym jako początkowy, od tego stanu rozpoczyna działanie. Praca automatu opiera się na odczytywaniu kolejnych symboli wejściowych ze skończonego alfabetu. Każdy odczytany symbol wymusza przejście do określonego stanu. Jeżeli po przeczytaniu wszystkich symboli, stan automatu będzie należał do zbioru stanów końcowych, ciąg symboli zostaje zaakceptowany, w innym wypadku nie zostaje zaakceptowany.

DAS przedstawia się za pomocą **grafów skierowanych**. Przejścia do kolejnych stanów są zaznaczane za pomocą **łuków**.

DAS przedstawiamy jako **formalnie uporządkowaną piątkę**:

$\langle Q, \Sigma, \delta, q_0, F \rangle$

$Q$  – skończony zbiór stanów,

$\Sigma$  – skończony zbiór symboli wejściowych,

$\delta$  – funkcja przejścia, która przyjmuje jako argument stan i symbol wejściowy i zwraca stan

$q_0$  – zbiór stanów początkowych,

$F$  – zbiór stanów końcowych

Dla funkcji przejścia  $\delta$  możemy stworzyć **tablicę przejść**, np.:

$\delta$	Wejścia	
	0	1
Stany		
$q_0$	$q_1$	$q_2$
$q_1$	$q_3$	$q_0$
$q_2$	$q_0$	$q_3$
$q_3$	$q_1$	$q_2$

## 3. Szczegóły implementacyjne

Na początku dodajemy niezbędne biblioteki oraz definiujemy limit danych wejściowych, które odczyta nasz program.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX_DANYCH_WEJ 20
```

Program składa się z dwóch funkcji: *main* oraz *diagram*.

## Funkcja *diagram*

Do funkcji tej przekazujemy stan początkowy oraz plik, z którego czytamy dane.

```
void diagram(FILE *in, char *stan){
```

W funkcji inicjujemy pętlę while, która czyta znaki z pliku dopóki nie napotka końca tego pliku.

```
while ((c=getc(in))!=EOF){
```

Jeżeli przeczytany znak jest równy 0, inicjujemy instrukcję sterującą switch. Zależnie od aktualnego stanu, program zmieni stan oraz doda diagram tego przejścia (strzałka + nowy stan) do tablicy znaków *wiersz*, która przechowuje nasz dotychczasowy diagram przejść. Program wyświetla wczytany znak oraz łańcuch *wiersz*.

```
if (c=='0'){
    switch (stan[1]) {
        case '0': stan[1]='1';
                    strcat(wiersz, q1);
                    printf("Wczytano %c: %s\n", c, wiersz);
                    break;
        case '1': stan[1]='3';
                    strcat(wiersz, q3);
                    printf("Wczytano %c: %s\n", c, wiersz);
                    break;
        case '2': stan[1]='0';
                    strcat(wiersz, q0);
                    printf("Wczytano %c: %s\n", c, wiersz);
                    break;
        case '3': stan[1]='1';
                    strcat(wiersz, q1);
                    printf("Wczytano %c: %s\n", c, wiersz);
                    break;}
    cnt++;
    if (cnt==MAX_DANYCH_WEJ) break;}
}
```

Dodatkowo zwiększamy zmienną *cnt* odpowiedzialną za zliczanie wczytanych znaków i sprawdzamy czy nie osiągnęła określonego limitu (jeżeli tak przerywamy pętlę).

Analogicznie robimy, gdy wczytany znak jest równy 1.

Jeżeli wczytany znak jest znakiem białym, ignorujemy go, zaś gdy znak jest niezgodny z określonym w zadaniu typem danych wejściowych (inny niż 0 i 1), to kończymy działanie programu.

```
else if (c=='\n' || c=='\t' || c==' ');

else {
    printf("Wczytano %c: ERROR\n\nZle dane wejsciowe\n", c);
    exit(0);
}
```

## Funkcja *main*

Nasze dane będziemy odczytywać z pliku podanego jako argument przy wywołaniu. Jeżeli argument nie zostanie podany, bądź pliku nie można otworzyć, program zwróci błąd.

```
int main(int argc, char **argv){  
    if (argc<2){  
        fprintf(stderr, "Nie podano nazwy pliku z danymi\n");  
        return 1;}  
  
    FILE *in = fopen (argv[1], "r");  
    if (in == NULL){  
        fprintf(stderr, "Nie mozna otworzyc pliku\n");  
        return 2;}  
}
```

Tworzymy zmienną stan przechowującą aktualny stan (na początku będzie to stan q0).

Inicjujemy funkcję diagram.

Sprawdzamy wartość zmiennej stan po wypisaniu diagramu, czyli nasz stan końcowy. Jeżeli jest zgodny z założeniami automatu akceptujemy ciąg, w innym wypadku nie akceptujemy. W tym miejscu kończymy program.

```
if (stan[1]!='3'){  
    printf("\nCiąg nie został zaakceptowany - zla wartosc koncowa\n");  
    return 3;}  
  
else printf("\nCiąg został zaakceptowany\n");  
  
return 0;
```

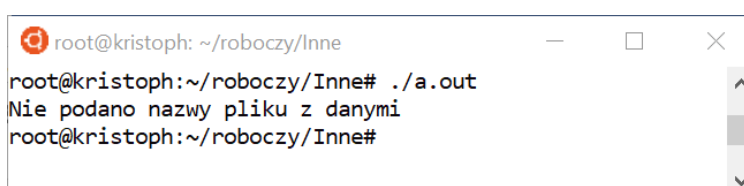
## 4. Sposób wywołania programu

Przy wywoływaniu programu należy podać nazwę pliku, z którego chcemy odczytywać dane wejściowe, jako pierwszy argument.

Znaki w pliku, z którego odczytujemy, mogą być oddzielone spacjami, tabulacjami, znakami nowej linii, lub być wypisane jeden po drugim. Program odczytuje kolejno znaki i przerwie działanie, gdy napotka znak inny niż „0”, „1” lub znak biały.

Przykłady wywołania programu:

### 4.1. Brak podania nazwy pliku do odczytu



```
root@kristoph: ~/roboczy/Inne  
root@kristoph:~/roboczy/Inne# ./a.out  
Nie podano nazwy pliku z danymi  
root@kristoph:~/roboczy/Inne#
```

Jeżeli nie podamy nazwy pliku, program zwróci nam błąd i wyświetli komunikat.

#### 4.2. Nie można otworzyć podanego pliku

```
root@kristoph: ~/roboczy/Inne
root@kristoph:~/roboczy/Inne# ./a.out plik.txt
Nie mozna otworzyc pliku
root@kristoph:~/roboczy/Inne#
```

Jeżeli nie można otworzyć podanego pliku, program zwróci nam błąd i wyświetli komunikat.

#### 4.3. Odczytano zły typ danej wejściowej

```
root@kristoph: ~/roboczy/Inne
root@kristoph:~/roboczy/Inne# cat plik.txt
011010
102010
root@kristoph:~/roboczy/Inne# ./a.out plik.txt

Stan początkowy: q0
Wczytano 0: q0 -> q1
Wczytano 1: q0 -> q1 -> q0
Wczytano 1: q0 -> q1 -> q0 -> q2
Wczytano 0: q0 -> q1 -> q0 -> q2 -> q0
Wczytano 1: q0 -> q1 -> q0 -> q2 -> q0 -> q2
Wczytano 0: q0 -> q1 -> q0 -> q2 -> q0 -> q2 -> q0
Wczytano 1: q0 -> q1 -> q0 -> q2 -> q0 -> q2 -> q0 -> q2
Wczytano 0: q0 -> q1 -> q0 -> q2 -> q0 -> q2 -> q0 -> q2 -> q0
Wczytano 2: ERROR

Złe dane wejściowe
root@kristoph:~/roboczy/Inne#
```

Kiedy program napotka zły typ danej wejściowej, kończy działanie i wyświetla komunikat.

#### 4.4. Prawidłowe dane wejściowe, zły stan końcowy

```
root@kristoph: ~/roboczy/Inne
root@kristoph:~/roboczy/Inne# cat plik.txt
010101011
001110010
root@kristoph:~/roboczy/Inne# ./a.out plik.txt

Stan początkowy: q0
Wczytano 0: q0 -> q1
Wczytano 1: q0 -> q1 -> q0
Wczytano 0: q0 -> q1 -> q0 -> q1
Wczytano 1: q0 -> q1 -> q0 -> q1 -> q0
Wczytano 0: q0 -> q1 -> q0 -> q1 -> q0 -> q1
Wczytano 1: q0 -> q1 -> q0 -> q1 -> q0 -> q1 -> q0
Wczytano 0: q0 -> q1 -> q0 -> q1 -> q0 -> q1 -> q0 -> q1
Wczytano 1: q0 -> q1 -> q0 -> q1 -> q0 -> q1 -> q0 -> q1 -> q0
Wczytano 0: q0 -> q1 -> q0 -> q1 -> q0 -> q1 -> q0 -> q1 -> q0 -> q2
Wczytano 1: q0 -> q1 -> q0 -> q1 -> q0 -> q1 -> q0 -> q1 -> q0 -> q2 -> q0
Wczytano 0: q0 -> q1 -> q0 -> q1 -> q0 -> q1 -> q0 -> q1 -> q0 -> q2 -> q0 -> q1
Wczytano 1: q0 -> q1 -> q0 -> q1 -> q0 -> q1 -> q0 -> q1 -> q0 -> q2 -> q0 -> q1 -> q0
Wczytano 0: q0 -> q1 -> q0 -> q1 -> q0 -> q1 -> q0 -> q1 -> q0 -> q2 -> q0 -> q1 -> q0 -> q2
Wczytano 1: q0 -> q1 -> q0 -> q1 -> q0 -> q1 -> q0 -> q1 -> q0 -> q2 -> q0 -> q1 -> q0 -> q2 -> q3
Wczytano 0: q0 -> q1 -> q0 -> q1 -> q0 -> q1 -> q0 -> q1 -> q0 -> q2 -> q0 -> q1 -> q0 -> q2 -> q3 -> q1
Wczytano 1: q0 -> q1 -> q0 -> q1 -> q0 -> q1 -> q0 -> q1 -> q0 -> q2 -> q0 -> q1 -> q0 -> q2 -> q3 -> q1 -> q3
Wczytano 0: q0 -> q1 -> q0 -> q1 -> q0 -> q1 -> q0 -> q1 -> q0 -> q2 -> q0 -> q1 -> q0 -> q2 -> q3 -> q1 -> q3 -> q2
Wczytano 1: q0 -> q1 -> q0 -> q1 -> q0 -> q1 -> q0 -> q1 -> q0 -> q2 -> q0 -> q1 -> q0 -> q2 -> q3 -> q1 -> q3 -> q2 -> q0
Ciąg nie został zaakceptowany - zła wartość końcowa
root@kristoph:~/roboczy/Inne#
```

Jeżeli stan końcowy będzie niezgodny z założeniami automatu, program wyświetli komunikat o niezaakceptowaniu ciągu.

#### 4.5. Prawidłowe dane wejściowe, prawidłowy stan końcowy

```
root@kristoph: ~/roboczy/Inne
root@kristoph:~/roboczy/Inne# cat plik.txt
0101011
00111001011
root@kristoph:~/roboczy/Inne# ./a.out plik.txt

Stan początkowy: q0
Wczytano 0: q0 -> q1
Wczytano 1: q0 -> q1 -> q0
Wczytano 0: q0 -> q1 -> q0 -> q1
Wczytano 1: q0 -> q1 -> q0 -> q1 -> q0
Wczytano 0: q0 -> q1 -> q0 -> q1 -> q0 -> q1
Wczytano 1: q0 -> q1 -> q0 -> q1 -> q0 -> q1 -> q0
Wczytano 0: q0 -> q1 -> q0 -> q1 -> q0 -> q1 -> q0 -> q2
Wczytano 1: q0 -> q1 -> q0 -> q1 -> q0 -> q1 -> q0 -> q2 -> q0
Wczytano 0: q0 -> q1 -> q0 -> q1 -> q0 -> q1 -> q0 -> q2 -> q0 -> q1
Wczytano 1: q0 -> q1 -> q0 -> q1 -> q0 -> q1 -> q0 -> q2 -> q0 -> q1 -> q0
Wczytano 0: q0 -> q1 -> q0 -> q1 -> q0 -> q1 -> q0 -> q2 -> q0 -> q1 -> q0 -> q2
Wczytano 1: q0 -> q1 -> q0 -> q1 -> q0 -> q1 -> q0 -> q2 -> q0 -> q1 -> q0 -> q2 -> q3
Wczytano 0: q0 -> q1 -> q0 -> q1 -> q0 -> q1 -> q0 -> q2 -> q0 -> q1 -> q0 -> q2 -> q3 -> q1
Wczytano 1: q0 -> q1 -> q0 -> q1 -> q0 -> q1 -> q0 -> q2 -> q0 -> q1 -> q0 -> q2 -> q3 -> q1 -> q3
Wczytano 0: q0 -> q1 -> q0 -> q1 -> q0 -> q1 -> q0 -> q2 -> q0 -> q1 -> q0 -> q2 -> q3 -> q1 -> q3 -> q2
Wczytano 1: q0 -> q1 -> q0 -> q1 -> q0 -> q1 -> q0 -> q2 -> q0 -> q1 -> q0 -> q2 -> q3 -> q1 -> q3 -> q2 -> q0
Wczytano 0: q0 -> q1 -> q0 -> q1 -> q0 -> q1 -> q0 -> q2 -> q0 -> q1 -> q0 -> q2 -> q3 -> q1 -> q3 -> q2 -> q0 -> q2
Wczytano 1: q0 -> q1 -> q0 -> q1 -> q0 -> q1 -> q0 -> q2 -> q0 -> q1 -> q0 -> q2 -> q3 -> q1 -> q3 -> q2 -> q0 -> q2 -> q3

Ciąg został zaakceptowany
root@kristoph:~/roboczy/Inne#
```

Jeżeli stan końcowy będzie zgodny z założeniami automatu, program wyświetli komunikat o zaakceptowaniu ciągu.

#### 4.6. Ilość danych wejściowych przekracza określone maksimum (maksimum określone jako 10)

```
root@kristoph: ~/roboczy/Inne
root@kristoph:~/roboczy/Inne# cat plik.txt
01101101010011000101
10101100011000110100
010001001011101010010
root@kristoph:~/roboczy/Inne# ./a.out plik.txt

Stan początkowy: q0
Wczytano 0: q0 -> q1
Wczytano 1: q0 -> q1 -> q0
Wczytano 1: q0 -> q1 -> q0 -> q2
Wczytano 0: q0 -> q1 -> q0 -> q2 -> q0
Wczytano 1: q0 -> q1 -> q0 -> q2 -> q0 -> q2
Wczytano 0: q0 -> q1 -> q0 -> q2 -> q0 -> q2 -> q3
Wczytano 1: q0 -> q1 -> q0 -> q2 -> q0 -> q2 -> q3 -> q1
Wczytano 0: q0 -> q1 -> q0 -> q2 -> q0 -> q2 -> q3 -> q1 -> q0
Wczytano 1: q0 -> q1 -> q0 -> q2 -> q0 -> q2 -> q3 -> q1 -> q0 -> q1
Wczytano 0: q0 -> q1 -> q0 -> q2 -> q0 -> q2 -> q3 -> q1 -> q0 -> q1
Wczytano 1: q0 -> q1 -> q0 -> q2 -> q0 -> q2 -> q3 -> q1 -> q0 -> q1 -> q0

Ciąg nie został zaakceptowany - zła wartość końcowa
root@kristoph:~/roboczy/Inne#
```

Jeżeli plik, z którego odczytujemy dane wejściowe, posiada ich więcej niż wynosi określony przez nas limit, program wczyta tyle danych ile wynosi limit.

## 5. Wnioski i spostrzeżenia

Wykonanie tego programu było ciekawym wyzwaniem. Zadanie zmusiło mnie do zagłębienia się w temat automatów skończonych, ale też pomogło lepiej zrozumieć ich działanie.

Uważam, że program był stosunkowo łatwy do napisania. Największą napotkaną trudnością było dla mnie znalezienie prostego sposobu przechowywania dotychczasowego diagramu przejść i dodawania nowych. Szukając rozwiązania poznałem funkcję *strcat*, która umożliwiła bardzo łatwe dodawanie kolejnych przejść do dotychczasowego diagramu i szybko rozwiązała ten problem.

Jestem bardzo zadowolony z całości mojego programu. Uważam, że jest napisany w sposób możliwie przejrzysty i działa poprawnie pod każdym względem. Dodatkowo, jest zabezpieczony na wypadek każdej opcji. Najbardziej zadowolony jestem ze sprytnego, moim zdaniem, sposobu, w jaki funkcja odczytuje i zmienia stan.