

Домашнее задание №1.

Проектирование и реализация конечного распознавателя.

Вариант 1.22.

Выполнил: Кравченко Александр Андреевич
КТб01-6

1. Постановка задачи:

1. Построить детерминированный конечный автомат с входным алфавитом $\{a, b, c\}$, допускающий множество цепочек, начинающихся и заканчивающихся симметричной парой символов.
2. Разработать программу, реализующую разработанный конечный распознаватель

2. Словесное описание автомата:

а. Описание состояний автомата:

q_0 – начальное состояние

q_1 – если первый символ пары – $a(N)$;

q_2 – если первый символ пары – $b(N)$;

q_3 – если первый символ пары – $c(N)$;

q_4 – если второй символ пары, начинающейся с 'a' – 'a'(Y). Если последующий символ также будет 'a', автомат останется в этом состоянии;

q_5 – если следующий символ в последовательности по ветке пары 'aa' – 'b'(N);

q_6 – если следующий символ в последовательности по ветке пары 'aa' – 'c'(N);

q_7 – если следующий символ в последовательности по ветке пары 'aa' – 'a'(N);

q_8 – если второй или следующий символ пары, начинающейся с 'a' – 'b'(N);

- q9 – если следующий символ в последовательности по ветке пары ‘ab’, после символа ‘b’ – ‘a’(Y);
- q10 – если следующий символ в последовательности по ветке пары ‘ab’ – ‘a’(N);
- q11 - если следующий символ в последовательности по ветке пары ‘ab’ – ‘c’(N);
- q12 – если второй или следующий символ пары, начинающейся с ‘a’ – ‘c’(N);
- q13 – если следующий символ в последовательности по ветке пары ‘ac’, после символа ‘c’ – ‘a’(Y);
- q14 – если следующий символ в последовательности по ветке пары ‘ac’ – ‘a’(N);
- q15 - если следующий символ в последовательности по ветке пары ‘ac’ – ‘b’(N);
- q16 – если второй символ пары, начинающейся с ‘b’ – ‘b’(Y). Если последующий символ также будет ‘b’, автомат останется в этом состоянии;
- q17 – если следующий символ в последовательности по ветке пары ‘bb’ – ‘a’(N);
- q18 – если следующий символ в последовательности по ветке пары ‘bb’ – ‘c’(N);
- q19 – если следующий символ в последовательности по ветке пары ‘bb’ – ‘b’(N);
- q20 – если второй или следующий символ пары, начинающейся с ‘b’ – ‘a’(N);
- q21 – если следующий символ в последовательности по ветке пары ‘ba’, после символа ‘a’ – ‘b’(Y);

- q22 – если следующий символ в последовательности по ветке пары ‘ba’ – ‘b’(N);
- q23 - если следующий символ в последовательности по ветке пары ‘ba’ – ‘c’(N);
- q24 – если второй или следующий символ пары, начинающейся с ‘b’ – ‘c’(N);
- q25 – если следующий символ в последовательности по ветке пары ‘bc’, после символа ‘c’ – ‘b’(Y);
- q26 – если следующий символ в последовательности по ветке пары ‘bc’ – ‘b’(N);
- q27 - если следующий символ в последовательности по ветке пары ‘bc’ – ‘a’(N);
- q28 – если второй символ пары, начинающейся с ‘c’ – ‘c’(Y).
Если последующий символ также будет ‘c’, автомат останется в этом состоянии;
- q29 – если следующий символ в последовательности по ветке пары ‘cc’ – ‘b’(N);
- q30 – если следующий символ в последовательности по ветке пары ‘cc’ – ‘a’(N);
- q31 – если следующий символ в последовательности по ветке пары ‘cc’ – ‘c’(N);
- q32– если второй или следующий символ пары, начинающейся с ‘c’ – ‘b’(N);
- q33 – если следующий символ в последовательности по ветке пары ‘cb’, после символа ‘b’ – ‘c’(Y);
- q34 – если следующий символ в последовательности по ветке пары ‘cb’ – ‘c’(N);
- q35 - если следующий символ в последовательности по ветке пары ‘cb’ – ‘a’(N);

q36 – если второй или следующий символ пары, начинающейся с ‘с’ – ‘а’(N);

q37 – если следующий символ в последовательности по ветке пары ‘са’, после символа ‘а’ – с’(Y);

q38 – если следующий символ в последовательности по ветке пары ‘са’ – ‘а’(N);

q39 - если следующий символ в последовательности по ветке пары ‘са’ – ‘b’(N);

в. Общая логика работы автомата:

Общая логика работы автомата состоит в том, для каждой возможной пары элементов есть своя ветка(свой блок). Всего 9 веток. В каждом блоке 4 состояния – 3 состояния для каждой из букв алфавита, а также еще одно состояние, которое работает только тогда, когда текущие крайние 2 символа последовательности образуют симметричную пару входной. Это состояние и является допускающим, то есть “покинуть” блок не корректно(когда последнее состояние не является допускающим) невозможно. Таким образом автомат способен правильно обработать входную последовательность.

3. Формальное описание распознавателя.

а. Описание в виде пятёрки множеств:

$$V = \{a, b, c\}$$

$$K = \{q_0, q_1, q_2, \dots, q_{38}, q_{39}\}$$

M:

$$M[q_0, a] = q_1;$$

$M[q_1, a] = q_4;$	$M[q_7, a] = q_4;$	$M[q_{10}, a] = q_{10};$
$M[q_4, a] = q_4;$	$M[q_7, b] = q_5;$	$M[q_{10}, b] = q_8;$
$M[q_4, b] = q_5;$	$M[q_7, c] = q_6;$	$M[q_{10}, c] = q_{11};$
$M[q_4, c] = q_6;$	$M[q_1, b] = q_8;$	$M[q_{11}, a] = q_{10};$
$M[q_5, a] = q_7;$	$M[q_8, b] = q_8;$	$M[q_{11}, b] = q_8;$
$M[q_5, b] = q_5;$	$M[q_8, a] = q_9;$	$M[q_{11}, c] = q_{11};$
$M[q_5, c] = q_6;$	$M[q_8, c] = q_{11};$	$M[q_1, c] = q_{12};$
$M[q_6, a] = q_7;$	$M[q_9, a] = q_{10};$	$M[q_{12}, c] = q_{12};$
$M[q_6, b] = q_5;$	$M[q_9, b] = q_8;$	$M[q_{12}, b] = q_{15};$
$M[q_6, c] = q_6;$	$M[q_9, c] = q_{11};$	$M[q_{12}, a] = q_{13};$

$M[q13,a] = q14;$
 $M[q13,b] = q15;$
 $M[q13,c] = q12;$
 $M[q14,a] = q14;$
 $M[q14,b] = q15;$
 $M[q14,c] = q12;$
 $M[q15,a] = q14;$
 $M[q15,b] = q15;$
 $M[q15,c] = q12;$
 $M[q0,b] = q2;$
 $M[q2,b] = q16;$
 $M[q16,b] = q16;$
 $M[q16,a] = q17;$
 $M[q16,c] = q18;$
 $M[q17,a] = q17;$
 $M[q17,b] = q19;$
 $M[q17,c] = q18;$
 $M[q18,a] = q17;$
 $M[q29,a] = q30;$
 $M[q29,b] = q29;$
 $M[q29,c] = q31;$
 $M[q30,a] = q30;$
 $M[q30,b] = q29;$
 $M[q30,c] = q31;$
 $M[q31,a] = q30;$
 $M[q31,b] = q29;$
 $M[q31,c] = q28;$
 $M[q3,b] = q32;$
 $M[q32,a] = q35;$
 $M[q32,b] = q32;$
 $S = \{q0\}$

$M[q18,b] = q19;$
 $M[q18,c] = q18;$
 $M[q19,a] = q17;$
 $M[q19,b] = q16;$
 $M[q19,c] = q18;$
 $M[q2,a] = q20;$
 $M[q20,a] = q20;$
 $M[q20,b] = q21;$
 $M[q20,c] = q23;$
 $M[q21,a] = q20;$
 $M[q21,b] = q22;$
 $M[q21,c] = q23;$
 $M[q22,a] = q20;$
 $M[q22,b] = q22;$
 $M[q22,c] = q23;$
 $M[q23,a] = q20;$
 $M[q23,b] = q22;$
 $M[q23,c] = q23;$
 $M[q32,c] = q33;$
 $M[q33,a] = q35;$
 $M[q33,b] = q32;$
 $M[q33,c] = q34;$
 $M[q34,a] = q35;$
 $M[q34,b] = q32;$
 $M[q34,c] = q34;$
 $M[q35,a] = q35;$
 $M[q35,b] = q32;$
 $M[q35,c] = q34;$
 $M[q3,a] = q36;$
 $M[q36,a] = q36;$

$M[q2,c] = q24;$
 $M[q24,a] = q27;$
 $M[q24,b] = q25;$
 $M[q24,c] = q24;$
 $M[q25,a] = q27;$
 $M[q25,b] = q26;$
 $M[q25,c] = q24;$
 $M[q26,a] = q27;$
 $M[q26,b] = q26;$
 $M[q26,c] = q24;$
 $M[q27,a] = q27;$
 $M[q27,b] = q26;$
 $M[q27,c] = q24;$
 $M[q0,c] = q3;$
 $M[q3,c] = q28;$
 $M[q28,c] = q28;$
 $M[q28,a] = q30;$
 $M[q28,b] = q29;$
 $M[q36,b] = q39;$
 $M[q36,c] = q37;$
 $M[q37,a] = q36;$
 $M[q37,b] = q39;$
 $M[q37,c] = q38;$
 $M[q38,a] = q36;$
 $M[q38,b] = q39;$
 $M[q38,c] = q38;$
 $M[q39,a] = q36;$
 $M[q39,b] = q39;$
 $M[q39,c] = q38;$

$Z = \{q4, q9, q13, q16, q21, q25, q28, q33, q37\}$

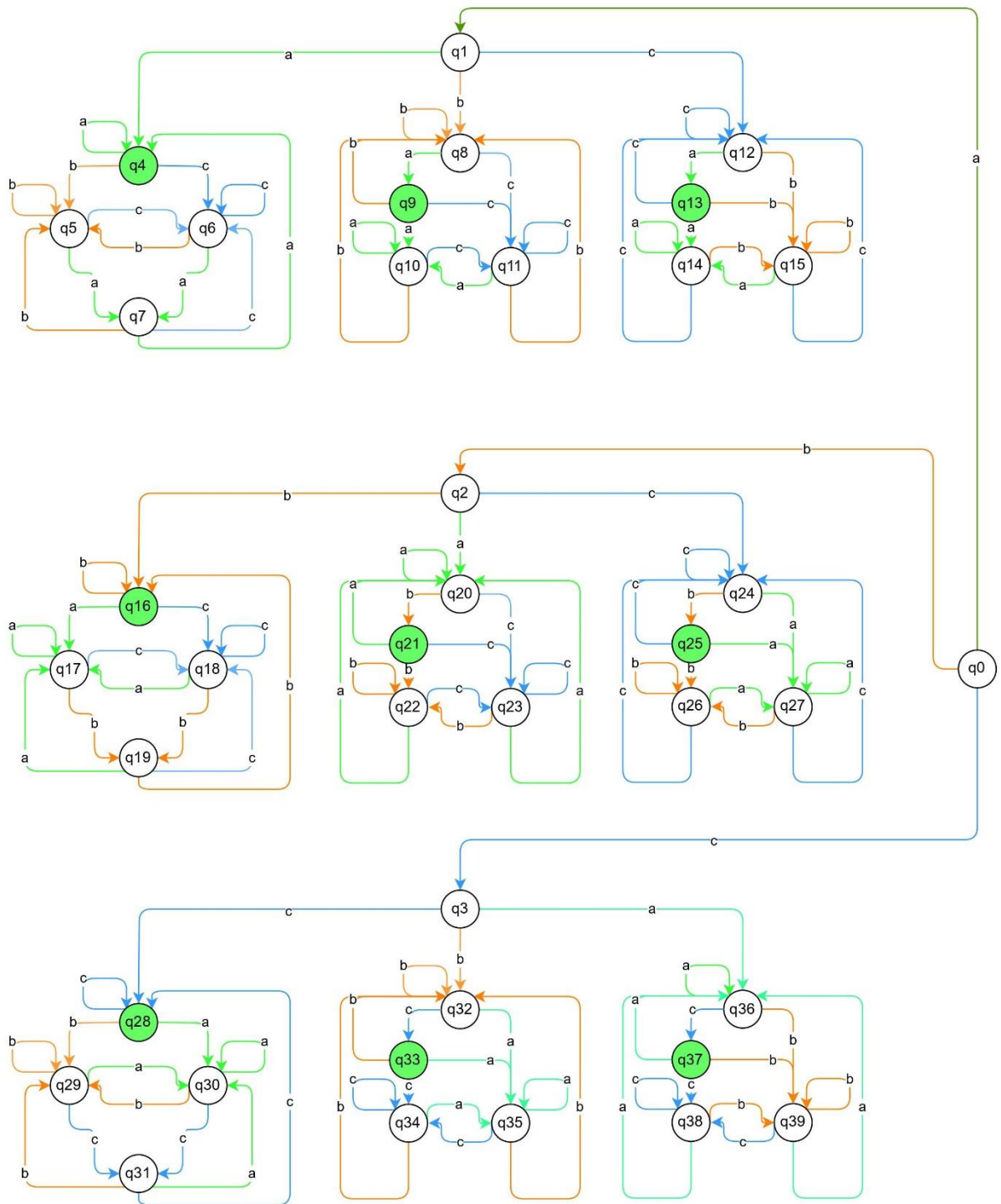
б. Описание в виде таблицы переходов:

Таблица 1

№	a	b	c	Y/N
q0	q1	q2	q3	N
q1	q4	q8	q12	N
q2	q20	q16	q24	N
q3	q36	q32	q28	N
q4	q4	q5	q6	Y
q5	q7	q5	q6	N
q6	q7	q5	q6	N
q7	q4	q5	q6	N
q8	q9	q8	q11	N
q9	q10	q8	q11	Y
q10	q10	q8	q11	N
q11	q10	q8	q11	N
q12	q13	q15	q12	N
q13	q14	q15	q12	Y
q14	q14	q15	q12	N
q15	q14	q15	q12	N
q16	q17	q16	q18	Y
q17	q17	q19	q18	N
q18	q17	q19	q18	N
q19	q17	q16	q18	N
q20	q20	q21	q23	N
q21	q20	q22	q23	Y
q22	q20	q22	q23	N
q23	q20	q22	q23	N
q24	q27	q25	q24	N
q25	q27	q26	q24	Y
q26	q27	q26	q24	N
q27	q27	q26	q24	N
q28	q30	q29	q28	Y
q29	q30	q29	q31	N
q30	q30	q29	q31	N
q31	q30	q29	q28	N
q32	q35	q32	q33	N
q33	q35	q32	q34	Y
q34	q35	q32	q34	N
q35	q35	q32	q34	N
q36	q36	q39	q37	N
q37	q36	q39	q38	Y
q38	q36	q39	q38	N
q39	q36	q39	q38	N

с. Описание в виде диаграммы переходов автомата:

Рисунок 1



Легенда:

Для удобства восприятия были использованы следующие условные обозначения:

● — линии зелёного цвета - буква 'a'

● — линии оранжевого цвета - буква 'b'

● — линии синего цвета - буква 'c'

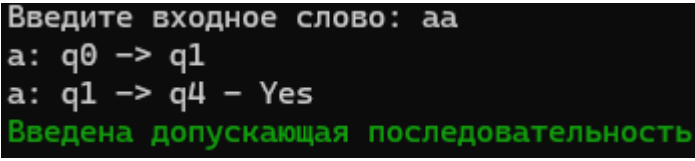
● — закрашенное зелёным состояние - допускающее(Y)

4. Набор тестов с пошаговой ручной проверкой

1. 'aa' – $q_0(a) \rightarrow q_1(a) \rightarrow q_4$ – Yes;
2. 'aba' – $q_0(a) \rightarrow q_1(b) \rightarrow q_8(a) \rightarrow q_9$ – Yes;
3. 'aca' – $q_0(a) \rightarrow q_1(c) \rightarrow q_{12}(a) \rightarrow q_{13}$ – Yes;
4. 'abab' – $q_0(a) \rightarrow q_1(b) \rightarrow q_8(a) \rightarrow q_9(b) \rightarrow q_8$ – No;
5. 'abba' – $q_0(a) \rightarrow q_1(b) \rightarrow q_8(b) \rightarrow q_8(a) \rightarrow q_9$ – Yes;
6. 'aaaa' – $q_0(a) \rightarrow q_1(a) \rightarrow q_4(a) \rightarrow q_4(a) \rightarrow q_4$ – Yes;
7. 'abcba' – $q_0(a) \rightarrow q_1(b) \rightarrow q_8(c) \rightarrow q_{11}(b) \rightarrow q_8(a) \rightarrow q_9$ – Yes;
8. 'acabca' – $q_0(a) \rightarrow q_1(c) \rightarrow q_{12}(a) \rightarrow q_{13}(b) \rightarrow q_{15}(c) \rightarrow q_{12}(a) \rightarrow q_{13}$ – Yes;
9. 'abacbbabba' – $q_0(a) \rightarrow q_1(b) \rightarrow q_8(a) \rightarrow q_9(c) \rightarrow q_{11}(b) \rightarrow q_8(b) \rightarrow q_8(a) \rightarrow q_9(b) \rightarrow q_8(a) \rightarrow q_9$ – Yes;
10. 'aabcaa' – $q_0(a) \rightarrow q_1(a) \rightarrow q_4(b) \rightarrow q_5(c) \rightarrow q_6(a) \rightarrow q_7(a) \rightarrow q_4$ – Yes;
11. 'acabcabca' – $q_0(a) \rightarrow q_1(c) \rightarrow q_{12}(a) \rightarrow q_{13}(b) \rightarrow q_{15}(c) \rightarrow q_{12}(a) \rightarrow q_{13}(b) \rightarrow q_{15}(c) \rightarrow q_{12}(a) \rightarrow q_{13}$ – Yes;
12. 'abcbcaa' – $q_0(a) \rightarrow q_1(b) \rightarrow q_{12}(c) \rightarrow q_{12}(b) \rightarrow q_{15}(c) \rightarrow q_{12}(a) \rightarrow q_{13}(a) \rightarrow q_{14}$ – No;
13. 'bb' – $q_0(b) \rightarrow q_2(b) \rightarrow q_{16}$ – Yes;
14. 'bbb' – $q_0(b) \rightarrow q_2(b) \rightarrow q_{16}(b) \rightarrow q_{16}$ – Yes;
15. 'bab' – $q_0(b) \rightarrow q_2(a) \rightarrow q_{20}(b) \rightarrow q_{21}$ – Yes;
16. 'bcb' – $q_0(b) \rightarrow q_2(c) \rightarrow q_{24}(b) \rightarrow q_{25}$ – Yes;
17. 'bbacbb' – $q_0(b) \rightarrow q_2(b) \rightarrow q_{16}(a) \rightarrow q_{17}(c) \rightarrow q_{18}(b) \rightarrow q_{19}(b) \rightarrow q_{16}$ – Yes;
18. 'bacaab' – $q_0(b) \rightarrow q_2(a) \rightarrow q_{20}(c) \rightarrow q_{23}(a) \rightarrow q_{20}(a) \rightarrow q_{20}(b) \rightarrow q_{21}$ – Yes;
19. 'bccabcb' – $q_0(b) \rightarrow q_2(c) \rightarrow q_{24}(c) \rightarrow q_{24}(a) \rightarrow q_{27}(b) \rightarrow q_{26}(c) \rightarrow q_{24}(b) \rightarrow q_{25}$ – Yes;
20. 'cc' – $q_0(c) \rightarrow q_3(c) \rightarrow q_{28}$ – Yes;
21. 'ccc' – $q_0(c) \rightarrow q_3(c) \rightarrow q_{28}(c) \rightarrow q_{28}$ – Yes;
22. 'cac' – $q_0(c) \rightarrow q_3(a) \rightarrow q_{36}(c) \rightarrow q_{37}$ – Yes;
23. 'cbc' – $q_0(c) \rightarrow q_3(b) \rightarrow q_{32}(c) \rightarrow q_{33}$ – Yes;
24. 'ccabcc' – $q_0(c) \rightarrow q_3(c) \rightarrow q_{28}(a) \rightarrow q_{30}(b) \rightarrow q_{29}(c) \rightarrow q_{31}(c) \rightarrow q_{28}$ – Yes;
25. 'caabac' – $q_0(c) \rightarrow q_3(a) \rightarrow q_{36}(a) \rightarrow q_{36}(b) \rightarrow q_{38}(a) \rightarrow q_{36}(c) \rightarrow q_{37}$ – Yes;
26. 'cbabbc' – $q_0(c) \rightarrow q_3(b) \rightarrow q_{32}(a) \rightarrow q_{35}(b) \rightarrow q_{32}(b) \rightarrow q_{32}(c) \rightarrow q_{33}$ – Yes;
27. 'bbabc' – $q_0(b) \rightarrow q_2(b) \rightarrow q_{16}(a) \rightarrow q_{17}(b) \rightarrow q_{19}(c) \rightarrow q_{18}$ – No;
28. 'aaab' – $q_0(a) \rightarrow q_1(a) \rightarrow q_4(a) \rightarrow q_4(b) \rightarrow q_5$ – No;
29. 'caabca' – $q_0(c) \rightarrow q_3(a) \rightarrow q_{36}(a) \rightarrow q_{36}(b) \rightarrow q_{39}(c) \rightarrow q_{38}(a) \rightarrow q_{36}$ – No;
30. 'cb' – $q_0(c) \rightarrow q_3(b) \rightarrow q_{32}$ – No;
31. 'ba' – $q_0(b) \rightarrow q_2(a) \rightarrow q_{20}$ – No;
32. 'ccbc' – $q_0(c) \rightarrow q_3(c) \rightarrow q_{28}(b) \rightarrow q_{29}(c) \rightarrow q_{31}$ – No;
33. 'acccba' – $q_0(a) \rightarrow q_1(c) \rightarrow q_{12}(c) \rightarrow q_{12}(c) \rightarrow q_{12}(b) \rightarrow q_{15}(a) \rightarrow q_{14}$ – No;
34. 'baaacb' – $q_0(b) \rightarrow q_2(a) \rightarrow q_{20}(a) \rightarrow q_{20}(a) \rightarrow q_{20}(c) \rightarrow q_{23}(b) \rightarrow q_{22}$ – No;
35. 'baaaab' – $q_0(b) \rightarrow q_2(a) \rightarrow q_{20}(a) \rightarrow q_{20}(a) \rightarrow q_{20}(a) \rightarrow q_{20}(b) \rightarrow q_{21}$ – Yes;

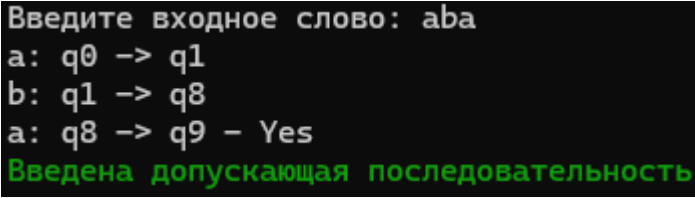
5. Скриншоты выполнения программы на тестовых примерах

Рисунок 2

1. 

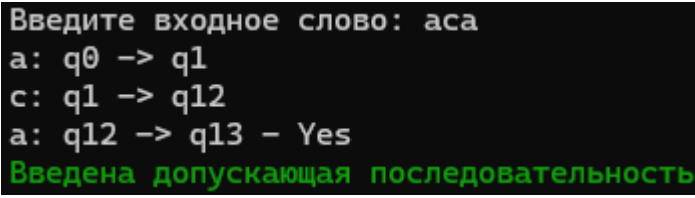
```
Введите входное слово: aa
a: q0 -> q1
a: q1 -> q4 - Yes
Введена допускающая последовательность
```

Рисунок 3

2. 

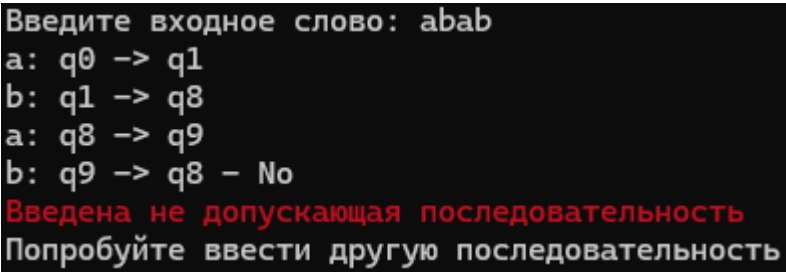
```
Введите входное слово: aba
a: q0 -> q1
b: q1 -> q8
a: q8 -> q9 - Yes
Введена допускающая последовательность
```

Рисунок 4

3. 

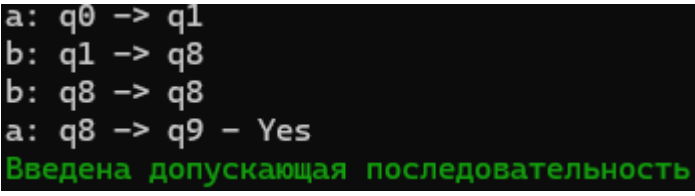
```
Введите входное слово: аса
a: q0 -> q1
c: q1 -> q12
a: q12 -> q13 - Yes
Введена допускающая последовательность
```

Рисунок 5

4. 

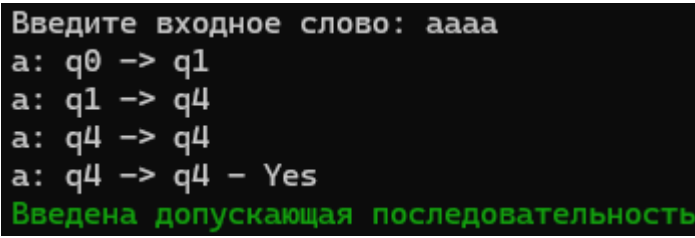
```
Введите входное слово: abab
a: q0 -> q1
b: q1 -> q8
a: q8 -> q9
b: q9 -> q8 - No
Введена не допускающая последовательность
Попробуйте ввести другую последовательность
```

Рисунок 6

5. 

```
a: q0 -> q1
b: q1 -> q8
b: q8 -> q8
a: q8 -> q9 - Yes
Введена допускающая последовательность
```

Рисунок 7

6. 

```
Введите входное слово: аaaa
a: q0 -> q1
a: q1 -> q4
a: q4 -> q4
a: q4 -> q4 - Yes
Введена допускающая последовательность
```

Рисунок 8

```
Введите входное слово: абсба
a: q0 -> q1
b: q1 -> q8
c: q8 -> q11
b: q11 -> q8
a: q8 -> q9 - Yes
Введена допускающая последовательность
```

Рисунок 9

```
Введите входное слово: асабса
a: q0 -> q1
c: q1 -> q12
a: q12 -> q13
b: q13 -> q15
c: q15 -> q12
a: q12 -> q13 - Yes
Введена допускающая последовательность
```

Рисунок 10

```
Введите входное слово: абасббабба
a: q0 -> q1
b: q1 -> q8
a: q8 -> q9
c: q9 -> q11
b: q11 -> q8
b: q8 -> q8
a: q8 -> q9
b: q9 -> q8
b: q8 -> q8
a: q8 -> q9 - Yes
Введена допускающая последовательность
```

Рисунок 11

```
Введите входное слово: аабсаа
a: q0 -> q1
a: q1 -> q4
b: q4 -> q5
c: q5 -> q6
a: q6 -> q7
a: q7 -> q4 - Yes
Введена допускающая последовательность
```

Рисунок 12

```
Введите входное слово: асавсабса
a: q0 -> q1
c: q1 -> q12
a: q12 -> q13
b: q13 -> q15
c: q15 -> q12
a: q12 -> q13
b: q13 -> q15
c: q15 -> q12
a: q12 -> q13 - Yes
11. Введена допускающая последовательность
```

Рисунок 13

```
Введите входное слово: абсбсаа
a: q0 -> q1
b: q1 -> q8
c: q8 -> q11
b: q11 -> q8
c: q8 -> q11
a: q11 -> q10
a: q10 -> q10 - No
Введена не допускающая последовательность
12. Попробуйте ввести другую последовательность
```

Рисунок 14

```
Введите входное слово: bb
b: q0 -> q2
b: q2 -> q16 - Yes
13. Введена допускающая последовательность
```

Рисунок 15

```
Введите входное слово: bbb
b: q0 -> q2
b: q2 -> q16
b: q16 -> q16 - Yes
14. Введена допускающая последовательность
```

Рисунок 16

```
Введите входное слово: bab
b: q0 -> q2
a: q2 -> q20
b: q20 -> q21 - Yes
15. Введена допускающая последовательность
```

Рисунок 17

16. Введите входное слово: bcb
b: q0 -> q2
c: q2 -> q24
b: q24 -> q25 - Yes
Введена допускающая последовательность

Рисунок 18

17. Введите входное слово: bbacbb
b: q0 -> q2
b: q2 -> q16
a: q16 -> q17
c: q17 -> q18
b: q18 -> q19
b: q19 -> q16 - Yes
Введена допускающая последовательность

Рисунок 19

18. Введите входное слово: басааб
b: q0 -> q2
a: q2 -> q20
c: q20 -> q23
a: q23 -> q20
a: q20 -> q20
b: q20 -> q21 - Yes
Введена допускающая последовательность

Рисунок 20

19. Введите входное слово: бссавсб
b: q0 -> q2
c: q2 -> q24
c: q24 -> q24
a: q24 -> q27
b: q27 -> q26
c: q26 -> q24
b: q24 -> q25 - Yes
Введена допускающая последовательность

Рисунок 21

20. Введите входное слово: сс
c: q0 -> q3
c: q3 -> q28 - Yes
Введена допускающая последовательность

Рисунок 22

Введите входное слово: ссс
с: q0 -> q3
с: q3 -> q28
с: q28 -> q28 - Yes
Введена допускающая последовательность

21.

Рисунок 23

Введите входное слово: сас
с: q0 -> q3
а: q3 -> q36
с: q36 -> q37 - Yes
Введена допускающая последовательность

22.

Рисунок 24

Введите входное слово: сbs
с: q0 -> q3
b: q3 -> q32
с: q32 -> q33 - Yes
Введена допускающая последовательность

23.

Рисунок 25

Введите входное слово: сsabсс
с: q0 -> q3
с: q3 -> q28
а: q28 -> q30
b: q30 -> q29
с: q29 -> q31
с: q31 -> q28 - Yes
Введена допускающая последовательность

24.

Рисунок 26

Введите входное слово: саабас
с: q0 -> q3
а: q3 -> q36
а: q36 -> q36
b: q36 -> q39
а: q39 -> q36
с: q36 -> q37 - Yes
Введена допускающая последовательность

25.

Рисунок 27

```
Введите входное слово: cbabbc
с: q0 -> q3
b: q3 -> q32
a: q32 -> q35
b: q35 -> q32
b: q32 -> q32
с: q32 -> q33 - Yes
26. Введена допускающая последовательность
```

Рисунок 28

```
Введите входное слово: bbabbc
b: q0 -> q2
b: q2 -> q16
a: q16 -> q17
b: q17 -> q19
с: q19 -> q18 - No
Введена не допускающая последовательность
27. Попробуйте ввести другую последовательность
```

Рисунок 29

```
Введите входное слово: aaab
a: q0 -> q1
a: q1 -> q4
a: q4 -> q4
b: q4 -> q5 - No
Введена не допускающая последовательность
28. Попробуйте ввести другую последовательность
```

Рисунок 30

```
Введите входное слово: саабса
с: q0 -> q3
a: q3 -> q36
a: q36 -> q36
b: q36 -> q39
с: q39 -> q38
a: q38 -> q36 - No
Введена не допускающая последовательность
29. Попробуйте ввести другую последовательность
```

Рисунок 31

```
Введите входное слово: cb
с: q0 -> q3
b: q3 -> q32 - No
Введена не допускающая последовательность
30. Попробуйте ввести другую последовательность
```

Рисунок 32

Введите входное слово: ba
b: q0 -> q2
a: q2 -> q20 - No
Введена не допускающая последовательность
31. Попробуйте ввести другую последовательность

Рисунок 33

Введите входное слово: ccbc
c: q0 -> q3
c: q3 -> q28
b: q28 -> q29
c: q29 -> q31 - No
Введена не допускающая последовательность
32. Попробуйте ввести другую последовательность

Рисунок 34

Введите входное слово: accsba
a: q0 -> q1
c: q1 -> q12
c: q12 -> q12
c: q12 -> q12
b: q12 -> q15
a: q15 -> q14 - No
Введена не допускающая последовательность
33. Попробуйте ввести другую последовательность

Рисунок 35

Введите входное слово: baaacb
b: q0 -> q2
a: q2 -> q20
a: q20 -> q20
a: q20 -> q20
c: q20 -> q23
b: q23 -> q22 - No
Введена не допускающая последовательность
34. Попробуйте ввести другую последовательность

6. Описание структуры данных, используемой в программе для хранения системы переходов автомата

В качестве структуры данных, используемой в программе для хранения системы переходов автомата был использован контейнер “map”, первым элементом которого является “string”, являющийся состоянием автомата. Второй же элемент “map”а – также “map”. Он же в свою очередь состоит из “char” и “string”, где “string” – состояние в которое ведёт состояние, отраженное в первом элементе “map”, а “char” – символ, который относится к этому состоянию.

Неполный вид таблицы в исходном коде программы:

Рисунок 36

```
map<string, map<char, string>> transitionTable =
{
    {"q0" , { {'a', "q1" }, {'b', "q2"}, {'c', "q3"} } },
    {"q1" , { {'a', "q4" }, {'b', "q8"}, {'c', "q12"} } },
    {"q2" , { {'a', "q20"}, {'b', "q16"}, {'c', "q24"} } },
    {"q3" , { {'a', "q28"}, {'b', "q24"}, {'c', "q32"} } },
    {"q4" , { {'a', "q36"}, {'b', "q32"}, {'c', "q40"} } },
    {"q8" , { {'a', "q44"}, {'b', "q40"}, {'c', "q48"} } },
    {"q12" , { {'a', "q52"}, {'b', "q48"}, {'c', "q56"} } },
    {"q16" , { {'a', "q60"}, {'b', "q56"}, {'c', "q64"} } },
    {"q20" , { {'a', "q68"}, {'b', "q64"}, {'c', "q72"} } },
    {"q24" , { {'a', "q76"}, {'b', "q72"}, {'c', "q80"} } },
    {"q28" , { {'a', "q84"}, {'b', "q80"}, {'c', "q88"} } },
    {"q32" , { {'a', "q92"}, {'b', "q88"}, {'c', "q96"} } },
    {"q36" , { {'a', "q100"}, {'b', "q96"}, {'c', "q104"} } },
    {"q40" , { {'a', "q108"}, {'b', "q104"}, {'c', "q112"} } },
    {"q44" , { {'a', "q116"}, {'b', "q112"}, {'c', "q120"} } },
    {"q48" , { {'a', "q124"}, {'b', "q120"}, {'c', "q128"} } },
    {"q52" , { {'a', "q132"}, {'b', "q128"}, {'c', "q136"} } },
    {"q56" , { {'a', "q140"}, {'b', "q136"}, {'c', "q144"} } },
    {"q60" , { {'a', "q148"}, {'b', "q144"}, {'c', "q152"} } },
    {"q64" , { {'a', "q156"}, {'b', "q152"}, {'c', "q160"} } },
    {"q68" , { {'a', "q164"}, {'b', "q160"}, {'c', "q168"} } },
    {"q72" , { {'a', "q172"}, {'b', "q168"}, {'c', "q176"} } },
    {"q76" , { {'a', "q180"}, {'b', "q176"}, {'c', "q184"} } },
    {"q80" , { {'a', "q188"}, {'b', "q184"}, {'c', "q192"} } },
    {"q84" , { {'a', "q196"}, {'b', "q192"}, {'c', "q200"} } },
    {"q88" , { {'a', "q204"}, {'b', "q200"}, {'c', "q208"} } },
    {"q92" , { {'a', "q212"}, {'b', "q208"}, {'c', "q216"} } },
    {"q96" , { {'a', "q220"}, {'b', "q216"}, {'c', "q224"} } },
    {"q100" , { {'a', "q228"}, {'b', "q224"}, {'c', "q232"} } },
    {"q104" , { {'a', "q236"}, {'b', "q232"}, {'c', "q240"} } },
    {"q108" , { {'a', "q244"}, {'b', "q240"}, {'c', "q248"} } },
    {"q112" , { {'a', "q252"}, {'b', "q248"}, {'c', "q256"} } },
    {"q116" , { {'a', "q260"}, {'b', "q256"}, {'c', "q264"} } },
    {"q120" , { {'a', "q268"}, {'b', "q264"}, {'c', "q272"} } },
    {"q124" , { {'a', "q276"}, {'b', "q272"}, {'c', "q280"} } },
    {"q128" , { {'a', "q284"}, {'b', "q280"}, {'c', "q288"} } },
    {"q132" , { {'a', "q292"}, {'b', "q288"}, {'c', "q296"} } },
    {"q136" , { {'a', "q300"}, {'b', "q296"}, {'c', "q304"} } },
    {"q140" , { {'a', "q308"}, {'b', "q304"}, {'c', "q312"} } },
    {"q144" , { {'a', "q316"}, {'b', "q312"}, {'c', "q320"} } },
    {"q148" , { {'a', "q324"}, {'b', "q320"}, {'c', "q328"} } },
    {"q152" , { {'a', "q332"}, {'b', "q328"}, {'c', "q336"} } },
    {"q156" , { {'a', "q340"}, {'b', "q336"}, {'c', "q344"} } },
    {"q160" , { {'a', "q348"}, {'b', "q344"}, {'c', "q352"} } },
    {"q164" , { {'a', "q356"}, {'b', "q352"}, {'c', "q360"} } },
    {"q168" , { {'a', "q364"}, {'b', "q360"}, {'c', "q368"} } },
    {"q172" , { {'a', "q372"}, {'b', "q368"}, {'c', "q376"} } },
    {"q176" , { {'a', "q380"}, {'b', "q376"}, {'c', "q384"} } },
    {"q180" , { {'a', "q388"}, {'b', "q384"}, {'c', "q392"} } },
    {"q184" , { {'a', "q396"}, {'b', "q392"}, {'c', "q400"} } },
    {"q188" , { {'a', "q404"}, {'b', "q400"}, {'c', "q408"} } },
    {"q192" , { {'a', "q412"}, {'b', "q408"}, {'c', "q416"} } },
    {"q196" , { {'a', "q420"}, {'b', "q416"}, {'c', "q424"} } },
    {"q200" , { {'a', "q428"}, {'b', "q424"}, {'c', "q432"} } },
    {"q204" , { {'a', "q436"}, {'b', "q432"}, {'c', "q440"} } },
    {"q208" , { {'a', "q444"}, {'b', "q440"}, {'c', "q448"} } },
    {"q212" , { {'a', "q452"}, {'b', "q448"}, {'c', "q456"} } },
    {"q216" , { {'a', "q460"}, {'b', "q456"}, {'c', "q464"} } },
    {"q220" , { {'a', "q468"}, {'b', "q464"}, {'c', "q472"} } },
    {"q224" , { {'a', "q476"}, {'b', "q472"}, {'c', "q480"} } },
    {"q228" , { {'a', "q484"}, {'b', "q480"}, {'c', "q488"} } },
    {"q232" , { {'a', "q492"}, {'b', "q488"}, {'c', "q496"} } },
    {"q236" , { {'a', "q500"}, {'b', "q496"}, {'c', "q504"} } },
    {"q240" , { {'a', "q508"}, {'b', "q504"}, {'c', "q512"} } },
    {"q244" , { {'a', "q516"}, {'b', "q512"}, {'c', "q520"} } },
    {"q248" , { {'a', "q524"}, {'b', "q520"}, {'c', "q528"} } },
    {"q252" , { {'a', "q532"}, {'b', "q528"}, {'c', "q536"} } },
    {"q256" , { {'a', "q540"}, {'b', "q536"}, {'c', "q544"} } },
    {"q260" , { {'a', "q548"}, {'b', "q544"}, {'c', "q552"} } },
    {"q264" , { {'a', "q556"}, {'b', "q552"}, {'c', "q560"} } },
    {"q268" , { {'a', "q564"}, {'b', "q560"}, {'c', "q568"} } },
    {"q272" , { {'a', "q572"}, {'b', "q568"}, {'c', "q576"} } },
    {"q276" , { {'a', "q580"}, {'b', "q576"}, {'c', "q584"} } },
    {"q280" , { {'a', "q588"}, {'b', "q584"}, {'c', "q592"} } },
    {"q284" , { {'a', "q596"}, {'b', "q592"}, {'c', "q600"} } },
    {"q288" , { {'a', "q604"}, {'b', "q600"}, {'c', "q608"} } },
    {"q292" , { {'a', "q612"}, {'b', "q608"}, {'c', "q616"} } },
    {"q296" , { {'a', "q620"}, {'b', "q616"}, {'c', "q624"} } },
    {"q300" , { {'a', "q628"}, {'b', "q624"}, {'c', "q632"} } },
    {"q304" , { {'a', "q636"}, {'b', "q632"}, {'c', "q640"} } },
    {"q308" , { {'a', "q644"}, {'b', "q640"}, {'c', "q648"} } },
    {"q312" , { {'a', "q652"}, {'b', "q648"}, {'c', "q656"} } },
    {"q316" , { {'a', "q660"}, {'b', "q656"}, {'c', "q664"} } },
    {"q320" , { {'a', "q668"}, {'b', "q664"}, {'c', "q672"} } },
    {"q324" , { {'a', "q676"}, {'b', "q672"}, {'c', "q680"} } },
    {"q328" , { {'a', "q684"}, {'b', "q680"}, {'c', "q688"} } },
    {"q332" , { {'a', "q692"}, {'b', "q688"}, {'c', "q696"} } },
    {"q336" , { {'a', "q700"}, {'b', "q696"}, {'c', "q704"} } },
    {"q340" , { {'a', "q708"}, {'b', "q704"}, {'c', "q712"} } },
    {"q344" , { {'a', "q716"}, {'b', "q712"}, {'c', "q720"} } },
    {"q348" , { {'a', "q724"}, {'b', "q720"}, {'c', "q728"} } },
    {"q352" , { {'a', "q732"}, {'b', "q728"}, {'c', "q736"} } },
    {"q356" , { {'a', "q740"}, {'b', "q736"}, {'c', "q744"} } },
    {"q360" , { {'a', "q748"}, {'b', "q744"}, {'c', "q752"} } },
    {"q364" , { {'a', "q756"}, {'b', "q752"}, {'c', "q760"} } },
    {"q368" , { {'a', "q764"}, {'b', "q760"}, {'c', "q768"} } },
    {"q372" , { {'a', "q772"}, {'b', "q768"}, {'c', "q776"} } },
    {"q376" , { {'a', "q780"}, {'b', "q776"}, {'c', "q784"} } },
    {"q380" , { {'a', "q788"}, {'b', "q784"}, {'c', "q792"} } },
    {"q384" , { {'a', "q796"}, {'b', "q792"}, {'c', "q800"} } },
    {"q388" , { {'a', "q804"}, {'b', "q800"}, {'c', "q808"} } },
    {"q392" , { {'a', "q812"}, {'b', "q808"}, {'c', "q816"} } },
    {"q396" , { {'a', "q820"}, {'b', "q816"}, {'c', "q824"} } },
    {"q400" , { {'a', "q828"}, {'b', "q824"}, {'c', "q832"} } },
    {"q404" , { {'a', "q836"}, {'b', "q832"}, {'c', "q840"} } },
    {"q408" , { {'a', "q844"}, {'b', "q840"}, {'c', "q848"} } },
    {"q412" , { {'a', "
```

7. Словесное описание идеи программной реализации одного шага работы автомата

Идея программы очень проста, имеется “map”, являющийся таблицей переходов автомата, а также введенная последовательность. Есть цикл, который перебирает все символы в последовательности. Также, имеется изначальное состояние автомата – q_0 . На каждом шаге работы автомата, в зависимости от символа, на котором сейчас находится итератор, используя таблицу переходов, выбирается следующее состояние. Цикл заканчивается тогда, когда все символы последовательности были обработаны, следовательно была построена полная цепочка состояний, у которой есть конечное состояние, которое затем ищется в коллекции допускающих состояний. Если оно было найдено – цепочка допускающая. Иначе – не допускающая.

8. Листинг программы

```
//ЮФУ, ИКТИБ, МОП ЭВМ
//Программирование и основы теории алгоритмов
//ДЗ1 - Проектирование и реализация конечного распознавателя.
//КТб01-6, Кравченко Александр Андреевич
//22.03.2024
```

```
#include <iostream>
#include <map>
#include <set>
#include <string>
#include <regex>
#include <windows.h>
#include <lmcons.h>
```



```

using namespace std;

// Функция инициализации, обработки и хранения таблицы переходов, принимает
введённую строку
// Хранит в себе таблицу переходов, а также принимает введенную пользователем
последовательность
// Перед отправкой введенной последовательности на проверку допускающего
состояния, использует функцию sequenceValid,
// и в зависимости от возвращенного этой функцией значения либо продолжает работу
над строкой, либо передает в main код ошибки,
// возвращенный функцией sequenceValid
// Если sequenceValid прошла без ошибок, проверяет строку на допускающее
состояние функцией sequenceChecker
// Если вернулось значение 1, то значит последовательность допускающая, иначе
возвращает 0
// Структура данных:
// Контейнер "map", первым элементом которого является "string",
// являющийся состоянием автомата. Второй же элемент "map"а – также "map".
// Он же, в свою очередь, состоит из "char" и "string", где "string" – состояние
в которое ведёт состояние,
// отраженное в первом элементе "map", а "char" – символ, который относится к
этому состоянию.
int DFA(string);

// Функция, проверяющая последовательность на корректность ввода, а также
проверяющая, не был ли введен символ выхода
// Возвращает значения:
// 5 - если введен символ выхода
// 2 - если введенная строка пустая
// 3 - если в последовательности используются символы не из алфавита(a,b,c)
// 4 - если длина строки меньше 2(отсутствие пары)
int sequenceValid(string);

// Главная функция проверки последовательности, принимает введённую строку, а
также таблицу переходов
// Последовательно идёт по каждому символу последовательности, начиная с
состояния q0
// Для каждого символа получает его следующее состояние из таблицы переходов
transitionTable и сохраняет его в переменной nextState
// Пройдя по всем символам последовательности проверяет, является ли последнее
состояние допускающим(ищет его в сете допускающих состояний valids)
// Если состояние допускающее - возвращает 1
// Иначе - 0
int sequenceChecker(string, const map<string, map<char, string>>& );

int main()
{
    HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    setlocale(LC_ALL, "Russian");

```

```

int returnCode;
string input;
SetConsoleTextAttribute(hConsole, 7);

TCHAR username[UNLEN + 1];
DWORD username_len = UNLEN + 1;

if (GetUserName(username, &username_len)) {
    wcout << L"Здравствуй!", " << username << endl;
}

while (true)
{
    cout << "Введите входное слово или введите '0' для выхода из программы:
";
    cin>>input;

    returnCode = DFA(input);

    switch (returnCode) {
        case 0:
            cout << " - No" << endl;
            SetConsoleTextAttribute(hConsole, 4);
            cout << "Введена не допускающая последовательность" << endl;
            SetConsoleTextAttribute(hConsole, 7);
            cout << "Попробуйте ввести другую последовательность" << endl;
            break;
        case 1:
            cout << " - Yes" << endl;
            SetConsoleTextAttribute(hConsole, 2);
            cout << "Введена допускающая последовательность" << endl;
            break;
        case 2:
            SetConsoleTextAttribute(hConsole, 4);
            cout << "Введена пустая последовательность" << endl;
            SetConsoleTextAttribute(hConsole, 7);
            cout << "Попробуйте ввести другую последовательность" << endl;
            break;
        case 3:
            SetConsoleTextAttribute(hConsole, 4);
            cout << "Последовательность содержит недопустимые символы" << endl;
            SetConsoleTextAttribute(hConsole, 7);
            cout << "Попробуйте ввести другую последовательность, используя
символы a,b,c" << endl;
            break;
        case 4:
            SetConsoleTextAttribute(hConsole, 4);
            cout << "Длина строки менее 2, невозможно найти пару" << endl;
            SetConsoleTextAttribute(hConsole, 7);
            cout << "Попробуйте ввести другую последовательность" << endl;

```

```

        break;
    case 5:
        cout << "Спасибо за использование!";
        exit(0);
        break;
    }
    SetConsoleTextAttribute(hConsole, 7);
    cout << endl;
}
}

int sequenceValid(string input) {
    const regex validSymbols("(a|b|c)+$");
    if (input == "0") return 5;
    if (input.empty()) return 2;
    if (!regex_match(input, validSymbols)) return 3;
    if (input.length() < 2) return 4;
    return 0;
}

int sequenceChecker(string input, const map<string, map<char, string>>&
transitionTable)
{
    string nextState, currentState = "q0";
    set<string> valids = { "q4", "q9", "q13", "q16", "q21", "q25", "q28", "q33", "q37" };

    for (int i = 0; i < input.size(); ++i) {
        nextState = transitionTable.at(currentState).at(input[i]);
        cout << input[i] << ": " << currentState << " -> " << nextState;
        if (i != input.size() - 1)
            cout << endl;
        currentState = nextState;
    }

    if (valids.contains(currentState)) return 1;
    else return 0;
}

int DFA(string input)
{
    map<string, map<char, string>> transitionTable =
    {
        {"q0" , { {'a', "q1" }, {'b', "q2"}, {'c', "q3"} } },
        {"q1" , { {'a', "q4" }, {'b', "q8"}, {'c', "q12"} } },
        {"q2" , { {'a', "q20" }, {'b', "q16"}, {'c', "q24"} } },
        {"q3" , { {'a', "q36" }, {'b', "q32"}, {'c', "q28"} } },
        {"q4" , { {'a', "q4" }, {'b', "q5"}, {'c', "q6"} } },
        {"q5" , { {'a', "q7" }, {'b', "q5"}, {'c', "q6"} } },
        {"q6" , { {'a', "q7" }, {'b', "q5"}, {'c', "q6"} } },
        {"q7" , { {'a', "q4" }, {'b', "q5"}, {'c', "q6"} } },
    }
}

```

```

{"q8" , { {'a', "q9" }, {'b', "q8"}, {'c', "q11"} } },
{"q9" , { {'a', "q10" }, {'b', "q8"}, {'c', "q11"} } },
{"q10" , { {'a', "q10" }, {'b', "q8"}, {'c', "q11"} } },
{"q11" , { {'a', "q10" }, {'b', "q8"}, {'c', "q11"} } },
{"q12" , { {'a', "q13" }, {'b', "q15"}, {'c', "q12"} } },
{"q13" , { {'a', "q14" }, {'b', "q15"}, {'c', "q12"} } },
{"q14" , { {'a', "q14" }, {'b', "q15"}, {'c', "q12"} } },
{"q15" , { {'a', "q14" }, {'b', "q15"}, {'c', "q12"} } },
{"q16" , { {'a', "q17" }, {'b', "q16"}, {'c', "q18"} } },
{"q17" , { {'a', "q17" }, {'b', "q19"}, {'c', "q18"} } },
{"q18" , { {'a', "q17" }, {'b', "q19"}, {'c', "q18"} } },
{"q19" , { {'a', "q17" }, {'b', "q16"}, {'c', "q18"} } },
{"q20" , { {'a', "q20" }, {'b', "q21"}, {'c', "q23"} } },
{"q21" , { {'a', "q20" }, {'b', "q22"}, {'c', "q23"} } },
{"q22" , { {'a', "q20" }, {'b', "q22"}, {'c', "q23"} } },
{"q23" , { {'a', "q20" }, {'b', "q22"}, {'c', "q23"} } },
{"q24" , { {'a', "q27" }, {'b', "q25"}, {'c', "q24"} } },
{"q25" , { {'a', "q27" }, {'b', "q26"}, {'c', "q24"} } },
{"q26" , { {'a', "q27" }, {'b', "q26"}, {'c', "q24"} } },
{"q27" , { {'a', "q27" }, {'b', "q26"}, {'c', "q24"} } },
{"q28" , { {'a', "q30" }, {'b', "q29"}, {'c', "q28"} } },
{"q29" , { {'a', "q30" }, {'b', "q29"}, {'c', "q31"} } },
{"q30" , { {'a', "q30" }, {'b', "q29"}, {'c', "q31"} } },
{"q31" , { {'a', "q30" }, {'b', "q29"}, {'c', "q28"} } },
{"q32" , { {'a', "q35" }, {'b', "q32"}, {'c', "q33"} } },
{"q33" , { {'a', "q35" }, {'b', "q32"}, {'c', "q34"} } },
{"q34" , { {'a', "q35" }, {'b', "q32"}, {'c', "q34"} } },
{"q35" , { {'a', "q35" }, {'b', "q32"}, {'c', "q34"} } },
{"q36" , { {'a', "q36" }, {'b', "q39"}, {'c', "q37"} } },
{"q37" , { {'a', "q36" }, {'b', "q39"}, {'c', "q38"} } },
{"q38" , { {'a', "q36" }, {'b', "q39"}, {'c', "q38"} } },
{"q39" , { {'a', "q36" }, {'b', "q39"}, {'c', "q38"} } },

```

```
};
```

```
int sqVaR = sequenceValid(input);
```

```
if (sqVaR != 0) return sqVaR;
```

```
if (sequenceChecker(input, transitionTable)) return 1;
```

```
return 0;
```

```
}
```