

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**  
**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**  
**ВЫСШЕГО ОБРАЗОВАНИЯ**  
**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ**  
**УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**  
**(БГТУ им. В.Г. Шухова)**



**ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЯЮЩИХ СИСТЕМ**

**Лабораторная работа №3**

по дисциплине: Теория автоматов и формальных языков  
тема: «Регулярные языки и конечные распознаватели»

Выполнил: ст. группы ПВ-223  
Пахомов Владислав Андреевич

Проверили:  
ст. пр. Рязанов Юрий Дмитриевич

Белгород 2024 г.

**Лабораторная работа №3**  
Регулярные языки и конечные распознаватели  
Вариант 8

**Цель работы:** изучить основные способы задания регулярных языков, способы построения, алгоритмы преобразования, анализа и реализации конечных распознавателей.

1. Язык  $L_1$  в алфавите  $\{0, 1\}$ , представляющий собой множество цепочек, в которых на предпоследнем месте стоит единица, задан грамматикой:

$S \rightarrow A10$

$A \rightarrow A011$

$A \rightarrow 0A$

$A \rightarrow 1A$

$A \rightarrow \varepsilon$

Построить детерминированный конечный распознаватель языка  $L_1$ .

Преобразуем заданную грамматику к автоматной правосторонней. Сейчас она является КС-грамматикой.

Приведём грамматику и устраним левую рекурсию.

Лишних символов в грамматике нет.

В грамматике есть  $\varepsilon$ -правило. Исключим его.

$S \rightarrow A10$

$S \rightarrow 10$

$A \rightarrow A011$

$A \rightarrow 011$

$A \rightarrow 0A$

$A \rightarrow 0$

$A \rightarrow 1A$

$A \rightarrow 1$

Цепных правил в грамматике нет.

В грамматике есть левая рекурсия. Исключим её.

$S \rightarrow A10$

$S \rightarrow 10$

$A \rightarrow 011B$

$A \rightarrow 0AB$

$A \rightarrow 0B$

$A \rightarrow 1AB$

$A \rightarrow 1B$

$A \rightarrow 011$

$A \rightarrow 0A$

$A \rightarrow 0$

$A \rightarrow 1A$

$A \rightarrow 1$

$B \rightarrow 011B$

$B \rightarrow \varepsilon$

В грамматике есть  $\varepsilon$ -правило. Исключим его.

$S \rightarrow A10$   
 $S \rightarrow 10$   
 $A \rightarrow 011$   
 $A \rightarrow 011B$   
 $A \rightarrow 0A$   
 $A \rightarrow 0AB$   
 $A \rightarrow 0$   
 $A \rightarrow 0B$   
 $A \rightarrow 1A$   
 $A \rightarrow 1AB$   
 $A \rightarrow 1$   
 $A \rightarrow 1B$   
 $A \rightarrow 011$   
 $A \rightarrow 0A$   
 $A \rightarrow 0$   
 $A \rightarrow 1A$   
 $A \rightarrow 1$   
 $B \rightarrow 011B$   
 $B \rightarrow 011$

Исключим правила-дубликаты:

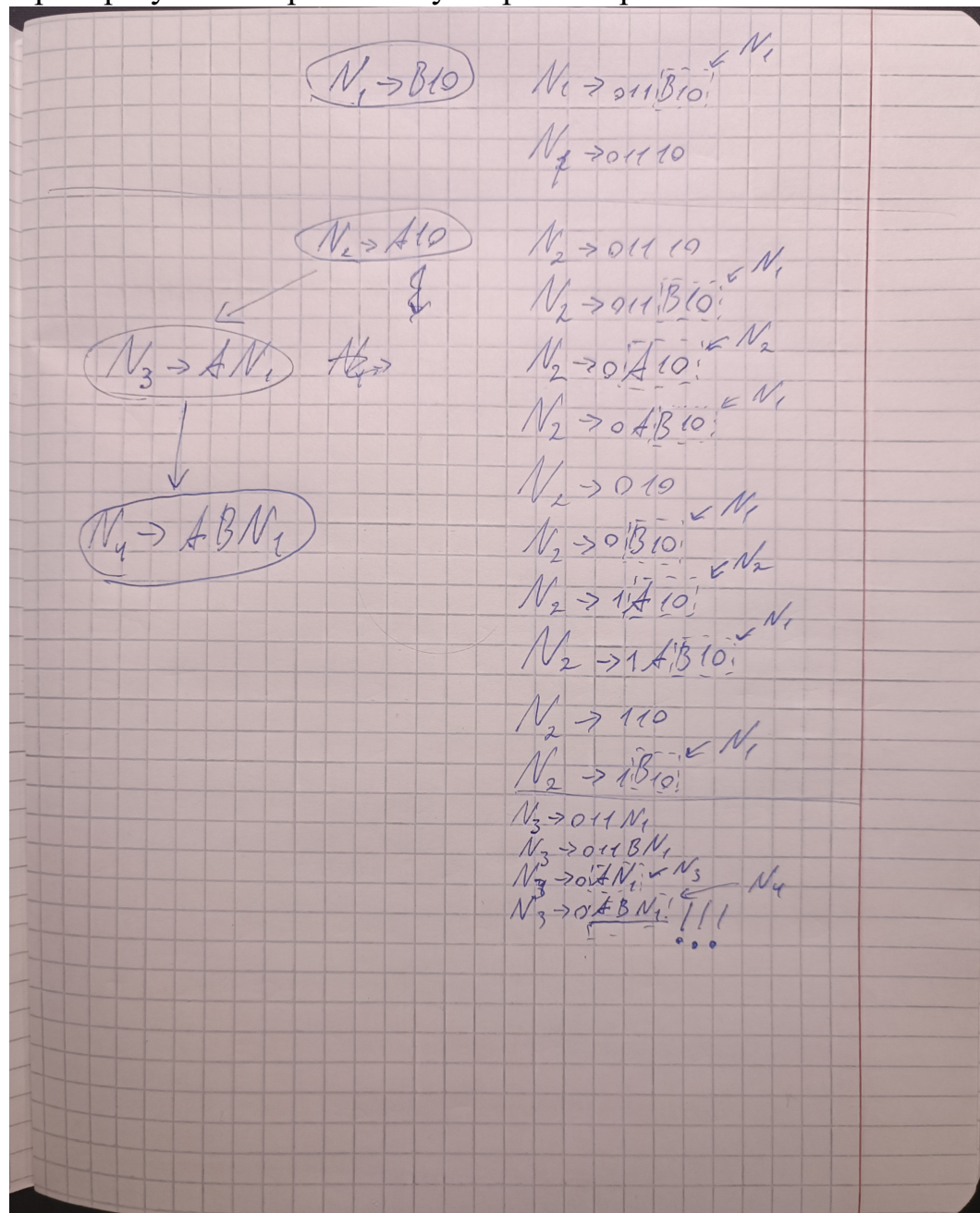
$S \rightarrow A10$   
 $S \rightarrow 10$   
 $A \rightarrow 011$   
 $A \rightarrow 011B$   
 $A \rightarrow 0A$   
 $A \rightarrow 0AB$   
 $A \rightarrow 0$   
 $A \rightarrow 0B$   
 $A \rightarrow 1A$   
 $A \rightarrow 1AB$   
 $A \rightarrow 1$   
 $A \rightarrow 1B$   
 $B \rightarrow 011B$   
 $B \rightarrow 011$

Грамматика приведена, а также в ней нет левой рекурсии. Преобразуем грамматику к такому виду, что каждое правило будет начинаться с терминала:

$S \rightarrow 01110$   
 $S \rightarrow 011B10$   
 $S \rightarrow 0A10$   
 $S \rightarrow 0AB10$   
 $S \rightarrow 010$   
 $S \rightarrow 0B10$   
 $S \rightarrow 1A10$   
 $S \rightarrow 1AB10$   
 $S \rightarrow 110$

$S \rightarrow 1B10$   
 $S \rightarrow 10$   
 $A \rightarrow 011$   
 $A \rightarrow 011B$   
 $A \rightarrow 0A$   
 $A \rightarrow 0AB$   
 $A \rightarrow 0$   
 $A \rightarrow 0B$   
 $A \rightarrow 1A$   
 $A \rightarrow 1AB$   
 $A \rightarrow 1$   
 $A \rightarrow 1B$   
 $B \rightarrow 011B$   
 $B \rightarrow 011$

Преобразуем КС-грамматику к правосторонней:



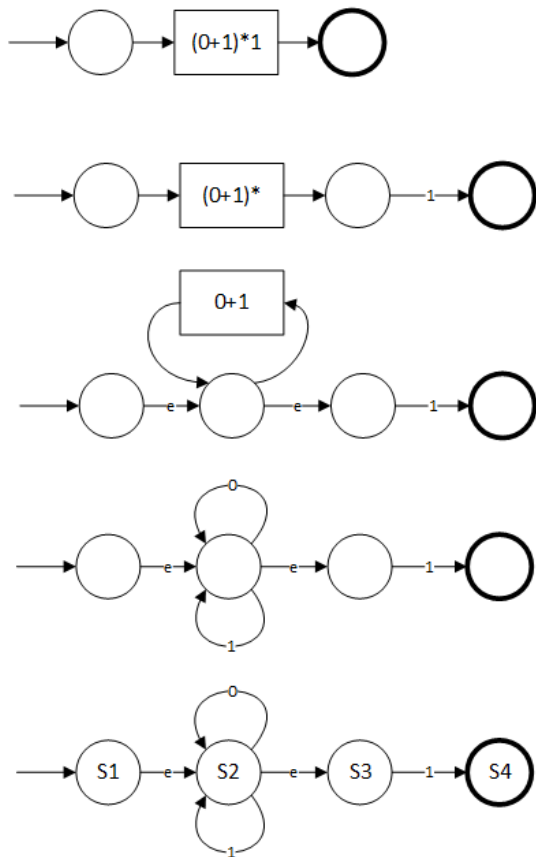
Преобразовать грамматику к правосторонней невозможно, так как в ходе преобразований получили правило (подчёркнутое с !!! в вычислениях)  $N_4 \rightarrow \underline{AB}N_1$ . С правилом  $N_3 \rightarrow AN_1$  они имеют общий префикс и постфикс, в дальнейшем мы будем получать правила вида  $AB^*N_1$ , получаем рекурсию, и следовательно правостороннюю грамматику с конечным числом правил получить нельзя. Задание невыполнимо.

2. Язык  $L_2$  в алфавите  $\{0, 1\}$ , представляющий собой множество цепочек, в которых на последнем месте стоит единица, задан регулярным выражением:  
 $(0+1)^*1$

Построить детерминированный конечный распознаватель языка  $L_2$ .

Для начала построим конечный недетерминированный распознаватель языка:

Получение  
 недетерминированного  
 конечного распознавателя:



Данный распознаватель языка не является детерминированным, так как он содержит  $\epsilon$ -переходы. Преобразуем данный конечный распознаватель языка в детерминированный:

	$\downarrow$			1
	S1	S2	S3	S4
1		S2	S4	
0		S2		
$\epsilon$	S2	S3		

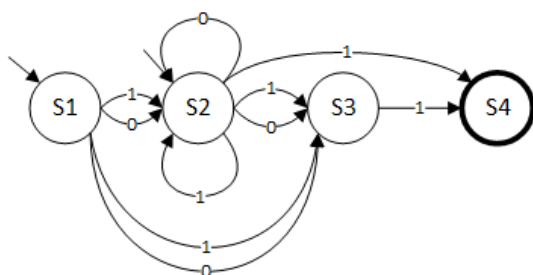
Удалим  $\epsilon$ -переходы:

$\epsilon$ -замыкания:  $\epsilon(S1) = \{S1, S2\}$ ,  $\epsilon(S2) = \{S2, S3\}$ ,  $\epsilon(S3) = \{S3\}$ ,  $\epsilon(S4) = \{S4\}$

	↓	↓		1
	$\epsilon(S1)$ {S1, S2}	$\epsilon(S2)$ {S2, S3}	$\epsilon(S3)$ {S3}	$\epsilon(S4)$ {S4}
1	$\epsilon(S2)$	$\epsilon(S2), \epsilon(S4)$	$\epsilon(S4)$	
0	$\epsilon(S2)$	$\epsilon(S2)$		

	↓	↓		1
	S1	S2	S3	S4
1	S2, S3	S2, S3, S4	S4	
0	S2, S3	S2, S3		

Устранение  $\epsilon$ -переходов



Преобразуем недетерминированный конечный распознаватель в детерминированный:

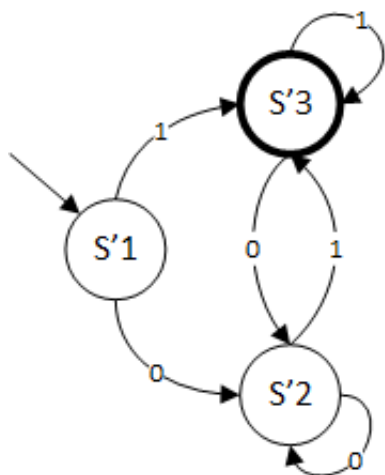
	{S1, S2}	{S2, S3}	{S2, S3, S4}
1	{S2, S3, S4}	{S2, S3, S4}	{S2, S3, S4}
0	{S2, S3}	{S2, S3}	{S2, S3}

Обозначим множества состояний как S'1, S'2, S'3...

S'1 обозначим как начальное состояние, согласно алгоритму, а S'3 обозначим как допускающее состояние, так как множество {S2, S3, S4} включает в себя допускающее состояние S4.

	↓		1
	S'1	S'2	S'3
1	S'3	S'3	S'3
0	S'2	S'2	S'2

Переход к  
детерминированному  
распознавателю

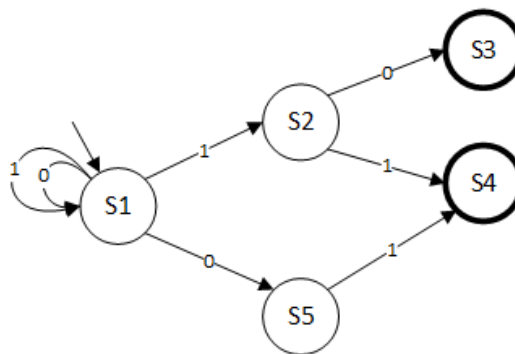


Построили детерминированный конечный распознаватель языка  $L_2$ .

3. Построить минимальный детерминированный конечный распознаватель языка  $L_3$  в алфавите  $\{0,1\}$ , представляющий собой множество цепочек, в которых хотя бы на одной из последних двух позиций стоит единица.

Пусть у нас будет исходный распознаватель языка  $L_3$ . В начальном состоянии S1 мы итеративно получаем 0 и 1, для окончания работы переходим в состояние S2 под действием символа 1, из него можем попасть в допускающие состояния S3 или S4 под действием символов 0 и 1 соответственно, так как если 1 - предпоследний символ, то строку можем закончить либо 1 либо 0. Если же предпоследний символ - 0, то из состояния S1 можно перейти в состояние S5 под действием символа 0. Однако из S5 мы теперь можем попасть только в S3, так как если предпоследний символ - 0, то последним обязательно должен быть 1. Получили недетерминированный конечный алгоритм без  $\epsilon$ -переходов:

Исходный  
недетерминированный  
распознаватель

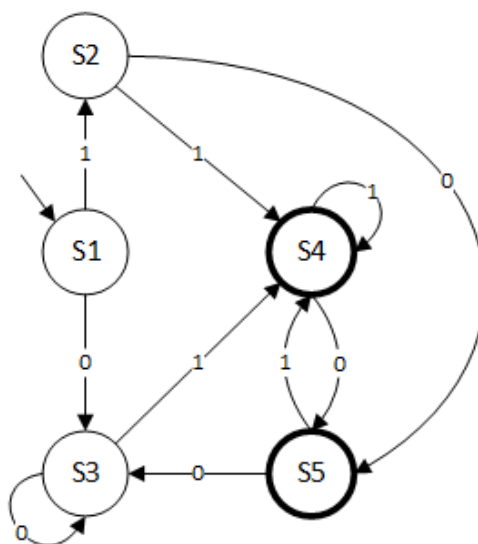


	↓		1	1	
	S1	S2	S3	S4	S5
1	S1, S2	S4			S4
0	S1, S5	S3			

Преобразуем распознаватель в детерминированный:

	{S1}	{S1, S2}	{S1, S5}	{S1, S2, S4}	{S1, S5, S3}
1	{S1, S2}	{S1, S2, S4}	{S1, S2, S4}	{S1, S2, S4}	{S1, S2, S4}
0	{S1, S5}	{S1, S5, S3}	{S1, S5}	{S1, S5, S3}	{S1, S5}

Детерминированный  
распознаватель



	↓			1	1
	S1	S2	S3	S4	S5
1	S2	S4	S4	S4	S4
0	S3	S5	S3	S5	S3

Полученный распознаватель является детерминированным, однако является ли он минимальным?

В распознавателе нет состояний, недостижимых из начального.

Перейдём к поиску и исключению эквивалентных состояний:

	↓			1	1
	S1	S2	S3	S4	S5
1	S2	S4	S4	S4	S4
0	S3	S5	S3	S5	S3

Отвергающие состояния {S1, S2, S3} объединим в класс K1 0-эквивалентных состояний, а допускающие состояния {S4, S5} - в класс K2.

	K1			K2	
	S1	S2	S3	S4	S5
1	K1	K2	K2	K1	K2
0	K1	K2	K1	K1	K2

Получили таблицу переходов в классы 0-эквивалентных состояний. На основе этой таблицы можем построить таблицу переходов в классы 1-эквивалентных состояний.

	K1	K2	K3	K4	K5
	S1	S3	S2	S4	S5
1	K3	K1	K4	K4	K4
0	K2	K1	K2	K5	K5

Получили таблицу переходов в классы 1-эквивалентных состояний. На основе этой



таблицы можем построить таблицу переходов в классы 2-эквивалентных состояний.

	K1	K2	K3	K4	K5	K6
	S1		S3	S2	S4	S5
1	K4	K2	K5	K5	K5	K5
0	K3	K2	K3	K6	K6	K3

Ни в одном из получившихся классов эквивалентных состояний не получилось более одного столбца, значит распознаватель является минимальным.

Построили минимальный детерминированный конечный распознаватель языка  $L_3$ .

4. Написать программу компиляционного типа для реализации минимального детерминированного конечного распознавателя языка  $L_3$ .

```
MESSAGES = {
    -1: "Отвергнуть, последовательность пуста",
    -2: "Отвергнуть, невалидный входной символ",
    -3: "Отвергнуть, слишком короткая цепочка",
    -4: "Отвергнуть, последние два символа не содержат 1",
    0: "Допустить"
}

def S1(input):
    if len(input) == 0:
        return -1

    if input[0] == '1':
        return S2(input[1:])
    elif input[0] == '0':
        return S3(input[1:])
    else:
        return -2

def S2(input):
    if len(input) == 0:
        return -3

    if input[0] == '1':
        return S4(input[1:])
    elif input[0] == '0':
        return S5(input[1:])
    else:
        return -2

def S3(input):
    if len(input) == 0:
        return -4

    if input[0] == '1':
        return S4(input[1:])
    elif input[0] == '0':
        return S3(input[1:])
    else:
```

```

        return -2

def S4(input):
    if len(input) == 0:
        return 0

    if input[0] == '1':
        return S4(input[1:])
    elif input[0] == '0':
        return S5(input[1:])
    else:
        return -2

def S5(input):
    if len(input) == 0:
        return 0

    if input[0] == '1':
        return S4(input[1:])
    elif input[0] == '0':
        return S3(input[1:])
    else:
        return -2

def L3validator(input):
    result = S1(input)
    print(input, MESSAGES[result])
    return result

```

5. Написать программу интерпретационного типа для реализации минимального детерминированного конечного распознавателя языка  $L_3$ .

```

MESSAGES = {
    0: "Отвергнуть, последовательность пуста",          # -1
    3: "Отвергнуть, невалидный входной символ",          # -2
    1: "Отвергнуть, слишком короткая цепочка",           # -3
    2: "Отвергнуть, последние два символа не содержат 1", # -4
    4: "Допустить",                                       # 0
}

PERMITTING = [3, 4]

MATRIX = {
    "1": [1, 3, 3, 3, 3],
    "0": [2, 4, 2, 4, 2]
}

def L3validator(input):
    input_origin = input
    S = 0

```

```

while len(input) > 0 and S >= 0:
    S = MATRIX[input[0]][S]
    input = input[1:]

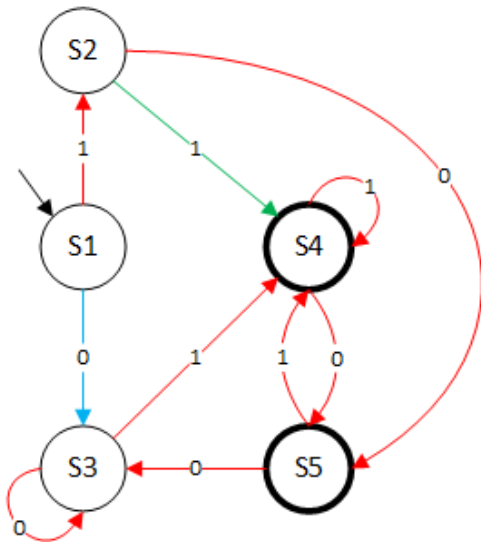
if S in PERMITTING:
    S = 4

print(input_origin, MESSAGES[S])
return S

```

6. Подобрать наборы тестовых данных так, чтобы в процессе тестирования сработал каждый переход конечного распознавателя.

- (a) 10001101 - уникальные тестируемые переходы отмечены красным
- (b) 11 - уникальные тестируемые переходы отмечены зелёным
- (c) 01 - уникальные тестируемые переходы отмечены голубым



Тесты для компиляционного варианта программы:

```

# Тестовые данные для всех переходов
assert L3validator("10001101") == 0
assert L3validator("11") == 0
assert L3validator("01") == 0

```

Тесты для интерпретационного варианта программы:

```

# Тестовые данные для всех переходов
assert L3validator("10001101") == 4
assert L3validator("11") == 4
assert L3validator("01") == 4

```

7. Подобрать наборы тестовых данных так, чтобы в процессе тестирования распознаватель закончил обработку цепочек в каждом состоянии конечного распознавателя.

- (a) <пустая строка> - состояние S1
- (b) 1 - состояние S2
- (c) 0 - состояние S3
- (d) 11 - состояние S4
- (e) 10 - состояние S5

Тесты для компиляционного варианта программы:

```
# Тестовые данные для всех состояний
assert L3validator("") == -1
assert L3validator("1") == -3
assert L3validator("0") == -4
assert L3validator("11") == 0
assert L3validator("10") == 0
```

Тесты для интерпретационного варианта программы:

```
# Тестовые данные для всех состояний
assert L3validator("") == 0
assert L3validator("1") == 1
assert L3validator("0") == 2
assert L3validator("11") == 4
assert L3validator("10") == 4
```

8. Выполнить тестирование программ для реализации минимального детерминированного конечного распознавателя языка  $L_3$ .

Результаты выполнения компиляционного варианта программы:

```
10001101 Допустить
11 Допустить
01 Допустить
Отвергнуть, последовательность пуста
1 Отвергнуть, слишком короткая цепочка
0 Отвергнуть, последние два символа не содержат 1
11 Допустить
10 Допустить

Process finished with exit code 0
```

## Результаты выполнения интерпретационного варианта программы:

```
10001101 Допустить
11 Допустить
01 Допустить
    Отвергнуть, последовательность пуста
1 Отвергнуть, слишком короткая цепочка
0 Отвергнуть, последние два символа не содержат 1
11 Допустить
10 Допустить

Process finished with exit code 0
```

Оба варианта программы завершились без ошибок, а значит проверки в проверках истинные, следовательно программа написана верно.

**Вывод:** в ходе лабораторной работы изучили основные способы задания регулярных языков, способы построения, алгоритмы преобразования, анализа и реализации конечных распознавателей.