МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

**Домашняя работа №1**

Вариант 10

Выполнил:

Круглов Егор Ильич, P3324

Преподаватель:

Лаздин Артур Вячеславович

Санкт-Петербург

2025

**Задача №10:**

*S → aASB | ε*

*A → ad | d*

*B → bb*

Тип грамматики: 2

Примеры предложений:

1. ε
2. aadbb
3. adbb

Множественно-теоретический вид:

L(G) =

**Задача №19:**

*S → A | SA | SB*

*A → a*

*B → b*

Тип грамматики: 2

Примеры предложений:

1. a
2. aa
3. ab

Множественно-теоретический вид:

**Задача №39:**

*Множество строк с нечетным числом вхождений подстроки ab.*

*А = {a, b, c}*

Грамматика:

S₀ → aS₁ | bS₀ | cS₀

S₁ → aS₁ | bS₂ | cS₀

S₂ → aS₃ | bS₂ | cS₂ | ε

S₃ → aS₃ | bS₀ | cS₂ | ε

Опишем 4 состояния :

четное вхождение подстрок , нет «подвешенной» (крайний правый символ) - .

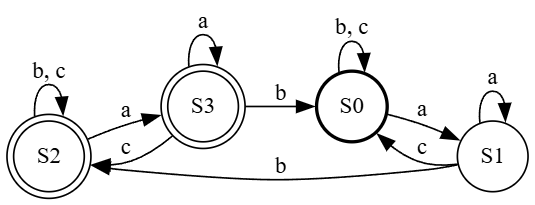
четное вхождение подстрок , есть «подвешенная» (крайний правый символ) - .

нечетное вхождение подстрок , нет «подвешенной» (крайний правый символ) - .

нечетное вхождение подстрок , есть «подвешенная» (крайний правый символ) - .

Переходы получаются простой логикой. Для добавили переход для выхода, так как нас удоволетворяют данные состояния (нечетное число вхождений подстроки .

ДКА:



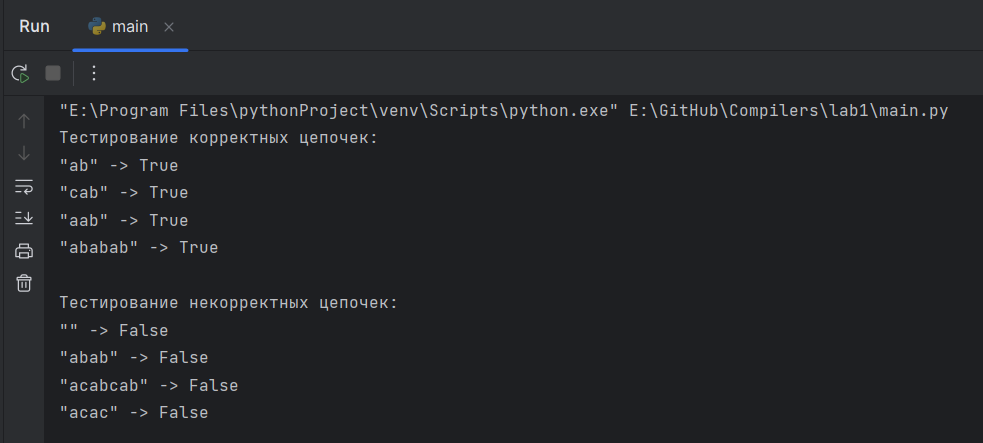
Код:

Ссылка на github: <https://github.com/KruglovEgor/Compilers/blob/main/lab1/main.py>

1. **def** simulate\_dfa(input\_str):
2. """
3. Моделирует работу ДКА для языка с нечетным числом вхождений подстроки 'ab'
4. Алфавит: {a, b, c}
6. Состояния:
7. S0: чётное число 'ab', нет подвешенного 'a'
8. S1: чётное число 'ab', есть подвешенное 'a'
9. S2: нечетное число 'ab', нет подвешенного 'a'
10. S3: нечетное число 'ab', есть подвешенное 'a'
12. Допускающие состояния: S2 и S3
13. """
14. # Начальное состояние: S0
15. state **=** 'S0'
17. # Переходы
18. transitions\_dic **=** {
19. 'S0': {
20. 'a': 'S1',
21. 'b': 'S0',
22. 'c': 'S0'
23. },
24. 'S1': {
25. 'a': 'S1',
26. 'b': 'S2',
27. 'c': 'S0'
28. },
29. 'S2': {
30. 'a': 'S3',
31. 'b': 'S2',
32. 'c': 'S2'
33. },
34. 'S3': {
35. 'a': 'S3',
36. 'b': 'S0',
37. 'c': 'S2'
38. }
39. }
41. # Допускающие состояния
42. exit\_states **=** ('S2', 'S3')
44. **for** ch **in** input\_str:
45. possible\_transitions **=** transitions\_dic[state]
46. **if** ch **in** possible\_transitions:
47. state **=** possible\_transitions[ch]
48. **else**:
49. # Неизвестный символ (не из нашего алфавита)
50. **return** False
52. **return** state **in** exit\_states

55. # Тестирование автомата
56. test\_strings **=** {
57. "valid": [
58. "ab",
59. "cab",
60. "aab",
61. "ababab"
62. ],
63. "invalid": [
64. "",
65. "abab",
66. "acabcab",
67. "acac"
68. ]
69. }
71. print("Тестирование корректных цепочек:")
72. **for** s **in** test\_strings["valid"]:
73. result **=** simulate\_dfa(s)
74. print(f'"{s}" -> {result}')
76. print("\nТестирование некорректных цепочек:")
77. **for** s **in** test\_strings["invalid"]:
78. result **=** simulate\_dfa(s)
79. print(f'"{s}" -> {result}')

Вывод в консоль:

**