МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

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

**Факультет программной инженерии и компьютерной техники**

**Домашнее задание №1**

по дисциплине

“Разработка компиляторов”

**Вариант № 9**

**Студент:**

Румский А.

Группа P3307

**Преподаватель:**

Лаздин Артур Вячеславович

Санкт-Петербург, 2025 год

## Пункт 1.

**Задача:** для каждой грамматики из списка [9, 18, 38] определить тип по классификации Хомского. Для грамматик типа 2|3 привести вывод не менее двух предложений языка, принадлежащий порождаемому языку.

### Грамматика 1.

**Анализ:**

Грамматика не соответствует ограничениям типа 3, так как, как минимум, присутствует рекурсия () В данной грамматике слева присутствуют только одиночные не-терминалы, справа же набор из терминалов и не-терминалов. Это соответствует типу 2.

**Вывод: Грамматика второго типа (контекстно-свободная)**

**Предложения языка:**

1.   
Возьмем:

Применим правило:

Получим:

2.

Возьмем:

Применим правило, выбрав:

Сохраняя результат, применим правило:

Подставим в , получим:

### Грамматика 2.

**Анализ:** Грамматика соответствует всем параметрам для грамматики 3 типа: один нетерминальный символ слева; не более одного нетерминального символа, с одной и той же стороны, от цепочки терминалов во всех правилах одной грамматики справа.

**Вывод: Грамматика третьего типа (регулярная)**

**Предложения языка:**

1.

Возьмём

Получим

Применим

Получим

Применим

Получим

Применим

Получим

2.

Возьмём

Получим

Применим

Получим

Применим

Получим

Применим

Получим

### Грамматика 3.

Множество строк четной длины, которые содержат ровно один a.

**Анализ:** второй тип(контекстно-свободная).

Код для проверки (Python)

def print\_error(inpt, symbol, index):

print(f"Обнаружен недопустимый сценарий:\n\tСтрока: {inpt}\n\tЛитерал: {symbol}\n\tИндекс: {index}")

class DFA:

def \_\_init\_\_(self):

# Определение состояний

# S→aX|bX|cX

# X→bF|cF|

# F→bX|cX|aG

# G→bD|cD|E

# D→bG|cG

# E→ε

self.states = {"q0", "q1", "q2", "q3", "q4","q5"} # Узлы

self.alphabet = {"a", "b", "c"} # Допустимые символы

self.start\_state = "q0" # Начальное состояние

self.accept\_states = {"q5"} # Конечное состояние

# Таблица переходов (состояние -> {входной символ -> следующее состояние})

self.transitions = {

"q0": {"a": "q1", "b": "q3", "c": "q3"},

"q1": {"b": "q2", "c": "q2"},

"q2": {"b": "q1", "c": "q1", "#": "q5"},

"q3": {"a": "q2", "b": "q4", "c": "q4"},

"q4": {"a": "q1", "b": "q3", "c": "q3"},

}

def process\_input(self, input\_string, debug=False):

if debug:

print(f"Получена строка: {input\_string}")

state = self.start\_state

for i, symbol in enumerate(input\_string):

if symbol in self.transitions[state]:

if debug:

print(f"\tТекущий символ: {symbol}, позиция {i}, состояние: {state}")

state = self.transitions[state][symbol]

if debug:

print(f"\t\tСимвол найден в словаре, новое состояние: {state}")

if state == "q5":

return True

else:

print\_error(input\_string, symbol, i)

return False # Недопустимый символ или отсутствие перехода

return False

dfa = DFA()

TEST\_CASES = ["cbcbc#", "cbcbbca#", "ccccca#", "abacb#", "abc#", "abcb#", "bcba#", "bcac#", "cbac#", "ccac#", "bbbab#", "cbcacb#", "caccbb#"]

for string in TEST\_CASES:

result = dfa.process\_input(string, debug=False)

print(f"Строка '{string}' -> {'Допустима' if result else 'Недопустима'}")