**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение высшего образования  
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ  
УНИВЕРСИТЕТ имени Н.Э.БАУМАНА  
(национальный исследовательский университет)»**

Факультет: Информатика и системы управления

Кафедра: Теоретическая информатика и компьютерные технологии

**Лабораторная работа № 4**

“Объектно-ориентированный лексический анализатор”

по дисциплине «Конструирование компиляторов»

Вариант 34

Работу выполнил

студент группы ИУ9-62Б

Сербин Денис

Москва, 2022

# **1. Цель работы**

# Целью данной работы является приобретение навыка реализации лексического анализа тора на объектно-ориентированном языке без применения каких-либо средств автоматизации решения задачи лексического анализа.

# **2. Задание**

# В лабораторной работе предлагается реализовать на языке Java две первые фазы стадии анализа: чтение входного потока и лексический анализ. При этом следует придерживаться схемы реализации объектно-ориентированного лексического анализатора, рассмотренной на лекции.Входной поток должен загружаться из файла (в UTF-8). В результате работы программы в стандартный поток вывода должны выдаваться описания распознанных лексем в формате:

# Тег (координаты\_фрагмента): атрибут лексемы

# При этом для лексем, не имеющих атрибутов, нужно выводить только тег и координаты. Например.

# IDENT (1, 2)-(1, 4): str

# ASSIGN (1, 8)-(1, 9);

# STRING (1, 11)-(1, 16); qwerty

# Лексемы во входном файле могут разделяться пробельными символами (пробел, горизонтальная табуляция, маркеры окончания строки), а могут быть записаны слитно (если это не приводит к противоречиям).

# Идентификаторы и числовые литералы не могут содержать внутри себя пробельных сим волов, если в задании явно не указано иного (варианты 3, 5 и 34). Комментарии, строковые и символьные литералы могут содержать внутри себя пробельные символы.

# Лексический анализатор должен иметь программный интерфейс для взаимодействия с пар сером. Рекомендуется реализовывать его как метод nextToken() для императивных языков или функцию, возвращающую список лексем, для функциональных языков.

# Входной файл может содержать ошибки, при обнаружении которых лексический анализа тор должен выдавать сообщение с указанием координаты, восстанавливаться и продолжать работу.

# Для лексических доменов должны вычисляться их атрибуты:

# • для целых чисел атрибут должен быть целым числом наибольшей разрядности (напри мер, в Java long).

# • для вещественных чисел атрибут должен быть вещественным числом (например, double в Java или С++),

# • для идентификаторов номер в таблице идентификаторов (см. слайды лекции).

# • для строковых констант значение, изображаемое самой строковой константой (т.е. без окружающих кавычек и с интерпретацией escape-последовательностей).

# • для комментариев токен не порождается, вместо этого координаты комментария помещается в список комментариев (см. слайды лекции).

**4. Индивидуальный вариант**

# Комментарии: начинаются с <//> и продолжаются до окончания строки текста. Идентификаторы: любой текст, не содержащий </> и ограниченный символами </>. Ключевые слова: </while/», «/do/>, <</end/».

# **5. Результат выполнения**

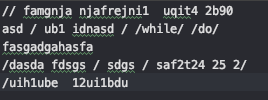


Рисунок 1. Текст входного потока

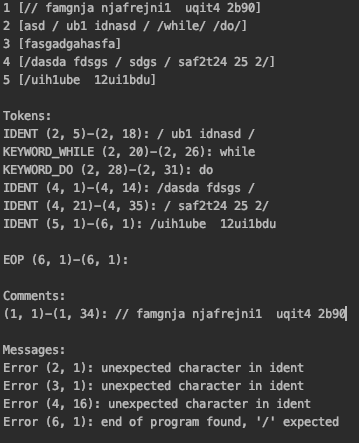


Рисунок 2. Лексический анализ

# **6. Вывод**

# В рамках данной лабораторной работы были приобретены навыки реализации лексического анализа тора на объектно-ориентированном языке без применения каких-либо средств автоматизации решения задачи лексического анализа.