**Лабораторная работа № 3  
Проектирование лексического анализатора с использованием регулярных выражений**

**Цель работы:** изучение основных понятий теории регулярных грамматик, ознакомление с назначением и принципами работы лексических анализаторов (сканеров), получение практических навыков построения сканера на примере заданного простейшего входного языка.

Для выполнения лабораторной работы требуется написать программу, которая выполняет лексический анализ входного текста в соответствии с заданием и порождает таблицу лексем с указанием их типов и значений. Текст на входном языке задается в виде символьного (текстового) файла. Программа должна выдавать сообщения о наличие во входном тексте ошибок, которые могут быть обнаружены на этапе лексического анализа.

Длину идентификаторов и строковых констант считать ограниченной 32 символами. Программа должна допускать наличие комментариев неограниченной длины во входном файле. Форму организации комментариев выбрать самостоятельно.

Для организации лексического разбора следует использовать регулярные выражения.

Регулярные выражения (регулярки) — это мощный инструмент для поиска, замены и обработки текстовых данных. Они представляют собой шаблоны, используемые для описания множества строк, которые удовлетворяют определенным критериям. Регулярные выражения применяются в различных областях, таких как текстовый поиск и замена, валидация данных, анализ текста и программирование.

Основные компоненты регулярных выражений

1. **Литералы**:
   * Это обычные символы, которые ищутся как есть. Например, cat в регулярном выражении будет искать текст "cat".
2. **Специальные символы**:
   * .: Соответствует любому одиночному символу, кроме символа новой строки.
   * \d: Соответствует любой цифре (эквивалентно [0-9]).
   * \D: Соответствует любому нецифровому символу (эквивалентно [^0-9]).
   * \w: Соответствует любому буквенному символу или цифре (эквивалентно [a-zA-Z0-9\_]).
   * \W: Соответствует любому не буквенно-цифровому символу (эквивалентно [^a-zA-Z0-9\_]).
   * \s: Соответствует любому пробельному символу (пробел, табуляция, перевод строки).
   * \S: Соответствует любому непробельному символу.
3. **Классы символов**:
   * [abc]: Соответствует любому из символов a, b или c.
   * [a-z]: Соответствует любому символу в диапазоне от a до z.
   * [^abc]: Соответствует любому символу, который не является a, b или c.
4. **Квантификаторы**:
   * \*: Соответствует нулю или более повторениям предыдущего символа или группы.
   * +: Соответствует одному или более повторениям предыдущего символа или группы.
   * ?: Соответствует нулю или одному повторению предыдущего символа или группы.
   * {n}: Соответствует ровно n повторениям предыдущего символа или группы.
   * {n,}: Соответствует n или более повторениям.
   * {n,m}: Соответствует от n до m повторениям.
5. **Группы и альтернация**:
   * () : Используется для группировки символов или выражений. Например, (abc)+ соответствует одной или более последовательностям "abc".
   * | : Альтернация (или). Например, cat|dog соответствует либо "cat", либо "dog".
6. **Якоря**:
   * ^: Соответствует началу строки.
   * $: Соответствует концу строки.
7. **Экранирование**:
   * \: Используется для экранирования специальных символов, чтобы они воспринимались как литералы. Например, \. соответствует точке, а не любому символу.

**Порядок выполнения работы**

Написать программу для решения задачи согласно варианту. Данная лабораторная работа аналогична лабораторной работе №2, но при разработке должны использоваться регулярные выражения.

Вариант лабораторной работы соответствует варианту лабораторной работы №2.

**Варианты заданий**

1. Входной язык содержит арифметические выражения, разделенные символом ;(точка с запятой). Арифметические выражения состоят из идентификаторов, десятичных чисел с плавающей точкой (в обычной и логарифмической форме), знаков операций и скобок.
2. Входной язык содержит арифметические выражения, разделенные символом ;(точка с запятой). Арифметические выражения состоят из идентификаторов, римских чисел, знаков операций и скобок. (Римскими считать числа, записанные большими буквами X, V и I).
3. Входной язык содержит упрощенные операторы цикла типа  
   while <логическое выражение> do <оператор присваивания>; Логическое выражение может содержать идентификаторы, знаки операций сравнения, целые десятичные числа без знака, скобки и логические операции and и or. Оператор присваивания должен состоять из идентификатора, знака присваивания и целой десятичной константы без знака.
4. Входной язык содержит упрощенные условные операторы типа  
   if <логическое выражение> then <оператор присваивания> else <оператор присваивания>; (часть else в операторе может отсутствовать). Логическое выражение может содержать идентификаторы, знаки операций сравнения, целые десятичные числа без знака, скобки и логические операции and и not. Оператор присваивания должен состоять из двух идентификаторов, разделенных знаком присваивания.
5. Входной язык содержит выражения над строковыми константами, разделенные символом ;(точка с запятой). Выражения состоят из идентификаторов, строковых констант, заключенных в двойные кавычки, одиночных символов, заключенных в одинарные кавычки и знаков операции конкатенации +.
6. Входной язык содержит последовательность вызовов процедур, разделенных символом ;(точка с запятой). Вызов процедуры должен состоять из имени процедуры и списка параметров. В качестве параметров могут выступать идентификаторы, целые десятичные числа без знака, шестнадцатеричные числа, десятичные числа с плавающей точкой.
7. Входной язык содержит последовательность вызовов процедур, разделенных символом ;(точка с запятой). Вызов процедуры должен состоять из имени процедуры и списка параметров. В качестве параметров могут выступать идентификаторы, строковые константы, заключенные в двойные кавычки и одиночные символы, заключенные в одинарные кавычки.
8. Входной язык содержит последовательность вызовов процедур, разделенных символом ;(точка с запятой). Вызов процедуры должен состоять из имени процедуры и списка параметров. В качестве параметров могут выступать идентификаторы и римские цифры со знаком. (Римскими считать числа, записанные большими буквами X, V и I)
9. Входной язык содержит последовательность описаний массивов в соответствии со спецификацией языка Паскаль, разделенных символом ;(точка с запятой). Считать, что массивы могут содержать только элементы скалярных типов integer, real, byte, word и char.
10. См. вариант №9, но размер массива должен указываться с помощью римских чисел.
11. Входной язык содержит последовательность описаний записей (record) в соответствии со спецификацией языка Паскаль, разделенных символом ;(точка с запятой). Считать, что записи могут содержать только поля скалярных типов integer, real, byte, word, char и строки string с возможным указанием длины строки в квадратных скобках.
12. Входной язык содержит последовательность описаний констант в соответствии со спецификацией языка Паскаль. Константы могут быть целыми десятичными числами со знаком, целыми шестнадцатеричными числами, целыми десятичными числами с плавающей точкой, строками или символами.
13. Входной язык содержит последовательность команд ассемблера в форме:  
    <метка>: <команда> <операнд1>,<операнд2> (метка, а также один или оба операнда могут отсутствовать). В качестве операндов могут выступать регистры процессора 80х86, идентификаторы, целые десятичные числа или целые шестнадцатеричные числа. (Предусмотреть наличие не менее 6 допустимых команд).
14. Входной язык содержит выражения, разделенные символом; (точка с запятой). Выражения состоят из идентификаторов, бинарных операций (сложение, вычитание, умножение, деление) и функций (например, sqrt, log, exp). Функции могут принимать идентификаторы или числовые константы в качестве аргументов. Также поддерживаются скобки для задания приоритетов операций.
15. Входной язык содержит последовательность логических выражений, разделенных символом; (точка с запятой). Логические выражения могут включать идентификаторы, булевы значения (true, false), знаки операций сравнения (равно, больше, меньше) и логические операции (and, or, not). Кроме того, присутствуют упрощенные условные операторы if <логическое выражение> then <оператор>, где оператор может быть присваиванием или вызовом процедуры.
16. Входной язык содержит описание массивов и записей в виде текстовых строк, разделенных символом; (точка с запятой). Массивы и записи могут включать поля различных скалярных типов: целых чисел, чисел с плавающей точкой, символов и строк. В описании массивов используются только целые числа для указания размеров, в описании записей — различные скалярные типы и строки.
17. Входной язык содержит многоуровневые арифметические выражения, разделенные символом; (точка с запятой). Выражения могут включать идентификаторы, числовые константы, скобки для задания приоритетов, а также функции (например, sin, cos). Операции могут включать сложение, вычитание, умножение и деление.
18. Входной язык содержит цикл for и условные операторы if. Цикл имеет вид for <идентификатор> := <начало> to <конец> do <оператор>, а условный оператор — if <логическое выражение> then <оператор> else <оператор>. В качестве операторов могут быть использованы присваивания или вызовы процедур.
19. Входной язык содержит описание типов данных и констант, разделенных символом; (точка с запятой). Типы данных могут включать простые (целые, вещественные, строковые) и составные (массивы, записи). Константы могут быть целыми числами, строками и символами. Описание типов должно включать указание имен типов и их соответствующих значений.
20. Входной язык содержит описание и вызовы процедур и функций, разделенные символом; (точка с запятой). Описание процедуры или функции включает имя, список параметров и тело. Параметры могут быть идентификаторами, строковыми или числовыми константами. Также предусмотрены вызовы процедур и функций с аргументами, которые могут быть идентификаторами или константами.