**ЛЕКСИЧЕСКИЙ АНАЛИЗ**

Первой фазой процесса компиляции является лексический анализ, то есть группирование строк литер, обозначающих идентификаторы, константы или слова языка и т.д., в единые символы (**лексемы**). Этот процесс может идти параллельно с другими фазами компиляции (например, в однопроходных компиляторах). Однако, в любом случае, при описании конструкции компилятора и его построения удобно представлять лексический анализ как самостоятельную фазу.

Блок сканирования (**сканер**) должен выдавать каждую лексему, устанавливая ее принадлежность тому или иному классу. Выбор классов зависит от особенностей транслируемого языка. Часто выделяют классы имен переменных, констант, ключевых слов, арифметических и логических операций ("+", "-", "\*", "/" и т.д.), специальных символов ("=", ";" и т. д.)

Характер распознаваемых строк может намного упростить процесс лексического анализа. Например:

идентификаторы:

a1 one

числа:

100 10.54

ключевые слова языка:

begin if end

Все эти строки можно генерировать с помощью регулярных выражений. Например, вещественные числа можно генерировать посредством регулярного выражения (+ | -) **d d**\*.**d**\* , где **d** обозначает любую цифру. Из выражения видно, что вещественное число состоит из следующих компонентов, расположенных именно в таком порядке:

1. возможного знака;

2. последовательности из одной или более цифр;

3. десятичной точки;

4. последовательности из нуля или более цифр.

Регулярные выражения эквивалентны грамматикам типа 3. Например, грамматика типа 3, соответствующая регулярному выражению для вещественного числа, имеет порождающие правила:

*R* → *+S | -S | dQ*

*S* → **d***Q*

*Q* → **d***Q | .F | .*

*F* → **d** *| dF*

где *R* - начальный символ, **d** – цифра.

Существует полное соответствие между регулярными выражениями (а потому и грамматиками типа 3) и конечными автоматами (учили).

Некоторые лексемы (например, \*) могут быть распознаны по одному считанному символу, другие (такие, как :=) – по двум символам, а для идентификации третьих необходимо u1087 прочитать неизвестное заранее число символов (например, код константы). В последнем случае, чтобы найти конец лексемы, приходится считывать один лишний символ, не входящий в состав данной лексемы. Этот символ необходимо запоминать, чтобы при разборе следующей лексемы он не был утерян.

Лексический анализатор, наряду с разбиением исходного потока символов на лексемы, может включать в исходный текст дополнительную информацию или исключать из него строки символов. Примером дополнительно включаемой информации являются номера строк. Если их не включить, то информация о том, в какой строке исходного текста находилась та или иная лексема, будет, в случае выполняющего сканирование в отдельном проходе компилятора, утеряна после лексического анализа.

Однако для диагностики на фазе синтаксического анализа необходимо иметь возможность ссылаться на ошибки в программе с указанием номеров строк оригинального исходного текста. С другой стороны, комментарии включаются в текст программы или описания объекта проектирования только для предоставления человеку дополнительных пояснений. Они никак

не влияют на генерируемый в дальнейшем код, и лексический анализатор обычно их просто исключает.

**Пример структуры программы сканирования**

Пусть реализуемый язык состоит только из оператора присваивания.

БНФ языка:

<Присваивание> ::= <Идент> = <Выражение>

Правило показывает, что в левой части присваивания – идентификатор,

далее следует символ присваивания (=), справа – выражение;

<Выражение> ::= <Операнд> | <Операнд> <Бин.оп> <Выражение>

Выражение – это операнд, или операнд, за которым следует бинарная

операция и выражение;

<Бин.оп> ::= "-" | "+" | "\*" | "/"

Бинарная операция – символ арифметической операции "-", "+","\*" или

"/";

<Операнд> ::= <Идент> | <Cons>

Операнд – это идентификатор или константа;

<Идент> ::= <Буква>

Идентификатор состоит из одной буквы;

<Const> ::= <Цифра> <Const> | <Цифра>

Константа – последовательность цифр, состоящая хотя бы из одной

цифры.

Лексический анализатор преобразует исходную программу в последовательность символов. Для удобства дальнейшей обработки лексем их разбивают на классы. В данном случае можно выделить следующие классы лексем:

1 – идентификатор;

2 – константа;

3 – символ присваивания;

4 – символы операций умножения и деления;

5 – символы операций сложения и вычитания.

Заметим, что необходимость разделения бинарных операций на две

группы диктуется их различным приоритетом, играющим важную роль при генерации постфиксной записи.

**ЛАБОРАТОРНАЯ РАБОТА №1.**

**РАЗРАБОТКА ЛЕКСИЧЕСКОГО АНАЛИЗАТОРА**

**1. Порядок выполнения работы.**

1.1. По варианту задания определить, какие классы лексем будут в вашем языке.

1.2. Составить контрольные примеры на реализуемом языке. Хотя бы один пример должен проверять поведение вашей программы при наличии недопустимых символов в транслируемом файле.

1.3. Запрограммировать и отладить модуль сканирования. Выполнить тестирование на контрольных примерах. Результатом работы должна быть таблица, содержащая лексемы и признаки их классов, для числовых констант их внутреннее представление (шестнадцатеричное). Необходимо включить

в результирующий файл информацию о номерах строк исходного текста

транслируемой программы.  
Одинаковые идентификаторы и константы в таблицу повторно не записываются.  
Необходимо предусмотреть восстановление после ошибок.

1.4. Оформить отчет.

**2. Содержание отчета.**

2.1. Название работы и ее исполнители.

2.2. Цель работы.

2.3. БНФ реализуемого языка.

2.4. Список классов лексем реализуемого языка.

2.5. Краткое (по 2-3 предложения) описание процедур (функций), из

которых состоит программа лексического анализа. Наилучший вариант –

включение описаний в текст программы в виде комментариев.

2.6. Листинг программы (не обязательно).

2.7. Распечатки контрольных примеров и результатов их выполнения.

2.8. Выводы по проделанной работе.

**ВАРИАНТЫ ЗАДАНИЙ К ЛАБОРАТОРНЫМ РАБОТАМ**

Во всех вариантах все переменные должны быть объявлены до начала

вычислений.

<Буква> – буква латинского алфавита (a...z).

<Цифра> – цифра от 0 до 9.  
**Комментарии.**

Для вариантов **1, 4, 7, 12** комментарий в стиле С++ однострочный  
// ----- Комментарий ------  
Для вариантов **2, 5, 8, 11** комментарий в стиле С++ многострочный  
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\* строки комментариев\*\*\*\*\*\*\*\*\*\*\*/  
Для вариантов **3, 6, 9, 10** – комментарий в стиле Паскаля   
{ Первая строка комментария.  
 Вторая строка комментария.

Последняя строка комментария. }

**Вариант 1**

<Программа> ::= <Объявление переменных> <Описание вычислений> .

<Описание вычислений> ::= <Список операторов>

<Объявление переменных> ::= Var <Список переменных>

<Список переменных> ::= <Идент> | <Идент> , <Список переменных>

<Список операторов> ::= <Оператор> | <Оператор> <Список операторов>  
<Оператор>::=<Присваивание> |<Сложный оператор>

<Присваивание> ::= <Идент> **=** <Выражение>

<Выражение> ::= <Ун.оп.> <Подвыражение> | <Подвыражение>

<Подвыражение> :: = ( <Выражение> ) | <Операнд> |

< Подвыражение > <Бин.оп.> <Подвыражение>

<Ун.оп.> ::= "-"|"not"

<Бин.оп.> ::= "-" | "+" | "\*" | "/" |"<"|">"|"=="

<Операнд> ::= <Идент> | <Const>

|<Сложный оператор>:: =<Оператор цикла>  
<Оператор цикла>:: =WHILE <Выражение> DO <Оператор>

<Идент> ::= <Буква> <Идент> | <Буква>

<Const> ::= <Цифра> <Const> | <Цифра>

**Вариант 2**

<Программа> ::= <Объявление переменных> <Описание вычислений>

<Описание вычислений> ::= Begin <Список операторов> End

<Объявление переменных> ::= Var <Список переменных> ;

<Список переменных> ::= <Идент> | <Идент> , <Список переменных>

<Список операторов> ::= <Оператор> | <Оператор> <Список операторов>  
<Оператор>::=<Присваивание> |<Сложный оператор>

<Присваивание> ::= <Идент> **:=** <Выражение> ;

<Выражение> ::= <Ун.оп.> <Подвыражение> | <Подвыражение>

<Подвыражение> :: = ( <Выражение> ) | <Операнд> |

< Подвыражение > <Бин.оп.> <Подвыражение>

<Ун.оп.> ::= "-"|"not"

<Бин.оп.> ::= "-" | "+" | "\*" | "/"|"=<"|">="|"="

<Операнд> ::= <Идент> | <Const>  
|<Сложный оператор>:: =<Оператор цикла>  
<Оператор цикла>:: =REPEAT <Список операторов> UNTIL <Выражение>

<Идент> ::= <Буква> <Идент> | <Буква>

<Const> ::= <Цифра> <Const> | <Цифра>

**Вариант 3**

<Программа> ::= <Объявление переменных> <Описание вычислений>

<Описание вычислений> ::= [ <Список операторов> ]

<Объявление переменных> ::= Var <Список переменных> ;

<Список переменных> ::= <Идент> | <Идент> , <Список переменных>

<Список операторов> ::= <Оператор> | <Оператор> <Список операторов>  
<Оператор>::=<Присваивание> |<Сложный оператор>

<Присваивание> ::= <Идент> **=** <Выражение> ;

<Выражение> ::= <Ун.оп.> <Подвыражение> | <Подвыражение>

<Подвыражение> :: = ( <Выражение> ) | <Операнд> |

< Подвыражение > <Бин.оп.> <Подвыражение>

<Ун.оп.> ::= "-"

<Бин.оп.> ::= "-" | "+" | "\*" | "/"|"<"|">"|"=="

<Операнд> ::= <Идент> | <Const>  
|<Сложный оператор>:: =<Оператор цикла>  
<Оператор цикла>:: =FOR <Идент> ":=" <СписокЦикла> DO <Оператор>  
<СписокЦикла> :: = <НачЗнач> TO <КонЗнач>  
<НачЗнач> :: = <Выражение>

<КонЗнач>:: = <Выражение>  
<Идент> ::= <Буква> <Идент> | <Буква>

<Const> ::= <Цифра> <Const> | <Цифра>

**Вариант 4**

<Программа> ::= <Объявление переменных> <Описание вычислений>

<Описание вычислений> ::= <Список операторов>

<Объявление переменных> ::= Var <Список переменных> ;

<Список переменных> ::= <Идент> | <Идент> , <Список переменных>

<Список операторов> ::= <Оператор> | <Оператор> <Список операторов>  
<Оператор>::=<Присваивание> |<Сложный оператор>

<Присваивание> ::= <Идент> **:=** <Выражение> ;

<Выражение> ::= <Ун.оп.><Подвыражение> | <Подвыражение>

<Подвыражение> :: = ( <Выражение> ) | <Операнд> |

< Подвыражение > <Бин.оп.> <Подвыражение>

<Ун.оп.> ::= "-"

<Бин.оп.> ::= "-" | "+" | "\*" | "/"|"<"|">"|"=="

<Операнд> ::= <Идент> | <Const>  
|<Сложный оператор>:: = IF <Выражение> THEN <Оператор> |

IF <Выражение> THEN <Оператор> ELSE <Оператор>

<Идент> ::= <Буква> <Идент> | <Буква>

<Const> ::= <Цифра> <Const> | <Цифра>

**Вариант 5**

<Программа> ::= <Объявление переменных> <Описание вычислений> .

<Описание вычислений> ::= Begin < Список операторов > End

<Объявление переменных> ::= Var <Список переменных>

<Список переменных> ::= <Идент> | <Идент> , <Список переменных>

<Список операторов> ::= <Оператор> | <Оператор> <Список операторов>  
<Оператор>::=<Присваивание> |<Сложный оператор>

<Присваивание> ::= <Идент> **:=** <Выражение>

<Выражение> ::= <Ун.оп.> <Подвыражение> | <Подвыражение>

<Подвыражение> :: = ( <Выражение> ) | <Операнд> |

< Подвыражение > <Бин.оп.> <Подвыражение>

<Ун.оп.> ::= "-"

<Бин.оп.> ::= "-" | "+" | "\*" | "/" | ">>" | "<<" | ">" | "<" | "="

<Операнд> ::= <Идент> | <Const>

<Сложный оператор> ::= IF "("< Выражение> ")" Оператор|

IF "(" <Выражение> ")" <Оператор> ELSE <Оператор>|<Составной оператор>  
<Составной оператор>::= Begin < Список операторов > End

<Идент> ::= <Буква> <Идент> | <Буква>

<Const> ::= <Цифра> <Const> | <Цифра>

**Вариант 6**

<Программа> ::= <Объявление переменных> <Описание вычислений>

<Описание вычислений> ::= Begin < Список операторов > End**.**

<Объявление переменных> ::= Var <Список переменных>

<Список переменных> ::= <Идент>; | <Идент> , <Список переменных> |

<Идент> ; <Список переменных>

<Список операторов> ::= <Оператор> | <Оператор> <Список операторов>  
<Оператор>::=<Присваивание> |<Сложный оператор>|<Составной оператор>   
<Составной оператор>::= Begin < Список операторов > End

<Присваивание> ::= <Идент> := <Выражение> ;

<Выражение> ::= <Ун.оп.><Подвыражение> | <Подвыражение>

<Подвыражение> :: = ( <Выражение> ) | <Операнд> |

< Подвыражение > <Бин.оп.> <Подвыражение>

<Ун.оп.> ::= "-"

<Бин.оп.> ::= "-" | "+" | "\*" | "/" | ">" | "<" | "="

<Операнд> ::= <Идент> | <Const>  
|<Сложный оператор>:: =<Оператор цикла>|<Составной оператор>

<Оператор цикла>:: =WHILE <Выражение> DO <Оператор>

<Составной оператор>::= Begin < Список операторов > End

<Идент> ::= <Буква> <Идент> | <Буква>

<Const> ::= <Цифра> <Const> | <Цифра>

**Вариант 7**

<Программа> ::= <Объявление переменных> <Описание вычислений>

<Описание вычислений> ::= Begin < Список операторов > End**.**

<Объявление переменных> ::= Var <Список переменных>

<Список переменных> ::= <Идент>; | <Идент> , <Список переменных> |

<Идент> ; <Список переменных>

<Список операторов> ::= <Оператор> | <Оператор> <Список операторов>  
<Оператор>::=<Присваивание> |<Сложный оператор>|<Составной оператор>   
<Составной оператор>::= Begin < Список операторов > End;

<Присваивание> ::= <Идент> = <Выражение> ;

<Выражение> ::= <Ун.оп.><Подвыражение> | <Подвыражение>

<Подвыражение> :: = ( <Выражение> ) | <Операнд> |

< Подвыражение > <Бин.оп.> <Подвыражение>

<Ун.оп.> ::= "-" | not

<Бин.оп.> ::= "-" | "+" | "\*" | "/" | "\*\*" | ">" | "<" | "=="

<Операнд> ::= <Идент> | <Const>  
<Сложный оператор>::= IF "("< Выражение> ")" <Оператор>;

<Идент> ::= <Буква> <Идент> | <Буква>

<Const> ::= <Цифра> <Const> | <Цифра>

**Вариант 8**

<Программа> ::= <Объявление переменных> <Описание вычислений>.

<Описание вычислений> ::= Begin <Список присваиваний> End

<Объявление переменных> ::= Var <Список переменных> :Boolean;

Var <Список переменных> :Boolean;| <Объявление переменных>  
<Объявление переменных> ::= Var <Список переменных> :Decimal;|  
Var <Список переменных> :Decimal; <Объявление переменных>

<Список операторов> ::= <Оператор> | <Оператор> <Список операторов>  
<Оператор>::=<Присваивание> |<Сложный оператор>

<Присваивание> ::= <Идент> **:=** <Выражение>

<Выражение> ::= <Ун.оп.> <Подвыражение> | <Подвыражение>

<Подвыражение> :: = ( <Выражение> ) | <Операнд> |

< Подвыражение > <Бин.оп.> <Подвыражение>

<Ун.оп.> ::= "!"

<Бин.оп.> ::= "&" | "|" | "^" | "-" | "+" | "\*" | "/" | ">" | "<" | "=="

<Операнд> ::= <Идент> | <Const>

<Сложный оператор> ::= IF "("< Выражение> ")" Оператор|

IF "(" <Выражение> ")" <Оператор> ELSE <Оператор>

<Операнд> ::= <Идент> | <Const>

<Идент> ::= <Буква> <Идент> | <Буква>

<Const> ::= < LConst >|< DConst >   
< DConst ><Цифра> |<Цифра><DConst>

< LConst > ::= 0|1

**Вариант 9**

<Программа> ::= <Объявление переменных> <Описание вычислений>.

<Описание вычислений> ::= Begin <Список присваиваний> End

<Объявление переменных> ::= Var <Список переменных> :Boolean;

Var <Список переменных> :Boolean;| <Объявление переменных>  
<Объявление переменных> ::= Var <Список переменных> :Decimal;|  
Var <Список переменных> :Decimal; <Объявление переменных>

<Список операторов> ::= <Оператор> | <Оператор> <Список операторов>  
<Оператор>::=<Присваивание> |<Сложный оператор>

<Присваивание> ::= <Идент> **:=** <Выражение>

<Выражение> ::= <Ун.оп.> <Подвыражение> | <Подвыражение>

<Подвыражение> :: = ( <Выражение> ) | <Операнд> |

< Подвыражение > <Бин.оп.> <Подвыражение>

|<Сложный оператор>:: =<Оператор цикла>  
<Оператор цикла>:: =FOR <Идент> ":=" <СписокЦикла> DO <Оператор>  
<СписокЦикла> :: = <НачЗнач> TO <КонЗнач>  
<НачЗнач> :: = <Выражение>

<КонЗнач>:: = <Выражение>  
<Ун.оп.> ::= ".NOT."

<Бин.оп.> ::= ".AND." | ".OR." | ".XOR." | "-" | "+" | "\*" | "/" | ">" | "<" | "=="

<Операнд> ::= <Идент> | <Const>

<Идент> ::= <Буква> <Идент> | <Буква>

<Const> ::= < BConst >|< DConst >   
< DConst ><Цифра> |<Цифра><DConst>

< BConst > ::= 0|1

**Вариант 10**

<Программа> ::= <Объявление переменных> <Описание вычислений> .

<Описание вычислений> ::= <Список операторов>

<Объявление переменных> ::= Var <Список переменных>

<Список переменных> ::= <Идент> | <Идент> , <Список переменных>

<Список операторов> ::= <Оператор> | <Оператор> <Список операторов>  
<Оператор>::=<Присваивание> |<Сложный оператор>

<Присваивание> ::= <Идент> **=** <Выражение>

<Выражение> ::= <Ун.оп.> <Подвыражение> | <Подвыражение>

<Подвыражение> :: = ( <Выражение> ) | <Операнд> |

< Подвыражение > <Бин.оп.> <Подвыражение>

<Ун.оп.> ::= "-"|"not"

<Бин.оп.> ::= "-" | "+" | "\*" | "/" |"<"|">"|"=="

<Операнд> ::= <Идент> | <Const>

|<Сложный оператор>:: =<Оператор цикла>|<Составной оператор>  
<Оператор цикла>:: =WHILE <Выражение> DO <Оператор>  
<Составной оператор>::= Begin < Список операторов > End

<Идент> ::= <Буква> <Идент> | <Буква>

<Const> ::= <Цифра> <Const> | <Цифра>

**Вариант 11**

<Программа> ::= <Объявление переменных> <Описание вычислений>

<Описание вычислений> ::= Begin < Список операторов > End**.**

<Объявление переменных> ::= Var <Список переменных>

<Список переменных> ::= <Идент>; | <Идент> , <Список переменных> |

<Идент> ; <Список переменных>

<Список операторов> ::= <Оператор> | <Оператор> <Список операторов>  
<Оператор>::=<Присваивание> |<Сложный оператор>|<Составной оператор>   
<Составной оператор>::= Begin < Список операторов > End

<Присваивание> ::= <Идент> := <Выражение> ;

<Выражение> ::= <Ун.оп.><Подвыражение> | <Подвыражение>

<Подвыражение> :: = ( <Выражение> ) | <Операнд> |

< Подвыражение > <Бин.оп.> <Подвыражение>

<Ун.оп.> ::= "-"

<Бин.оп.> ::= "-" | "+" | "\*" | "/" | ">" | "<" | "=" | "AND" | "OR" | "XOR"

<Операнд> ::= <Идент> | <Const>  
|<Сложный оператор>:: =<Оператор цикла>

<Оператор цикла>:: =WHILE <Выражение> DO <Оператор>

<Идент> ::= <Буква> <Идент> | <Буква>

<Const> ::= <Цифра> <Const> | <Цифра>

**Вариант 12**

<Программа> ::= <Объявление переменных> <Описание вычислений>

<Описание вычислений> ::= Begin < Список операторов > End

<Объявление переменных> ::= Int <Список переменных> |

Int <Список переменных> <Объявление переменных> | Bin <Список переменных> | Bin <Список переменных> <Объявление переменных>

<Список переменных> ::= <Идент>; | <Идент> , <Список переменных>

<Список присваиваний>::= <Присваивание> |

<Присваивание> <Список присваиваний>

<Присваивание> ::= <Идент> := <Выражение> ;

<Выражение> ::= <Ун.оп.> <Подвыражение> | <Подвыражение>

<Подвыражение> :: = ( <Выражение> ) | <Операнд> |

< Подвыражение > <Бин.оп.> <Подвыражение>

<Ун.оп.> ::= "-"

<Бин.оп.> ::= "&" | "|" | "^" | "-" | "+" | "\*" | "/"

<Операнд> ::= <Идент> | <Const>

<Идент> ::= <Буква> <Идент> | <Буква>

<Const> ::= < BConst >|< DConst >   
< DConst ><Цифра> |<Цифра><DConst>

< BConst > ::= 0|1