



Министерство образования и науки Российской Федерации  
Федеральное государственное образовательное учреждение  
высшего образования  
«Национальный исследовательский университет «МЭИ»  
Институт ИВТ Кафедра ПМИИ

## Отчет по лабораторной работе номер 3 Проведение лексического анализа восходящим методом

**Подготовил:** Желтиков Александр Алексеевич  
**Дата:** 2 сентября 2022 г.

# 1. Постановка задачи и ее условие.

**Задание.** С использованием лексического анализатора lex разработать и реализовать программу восходящего лексического анализа.

Номер варианта по журналу → 10

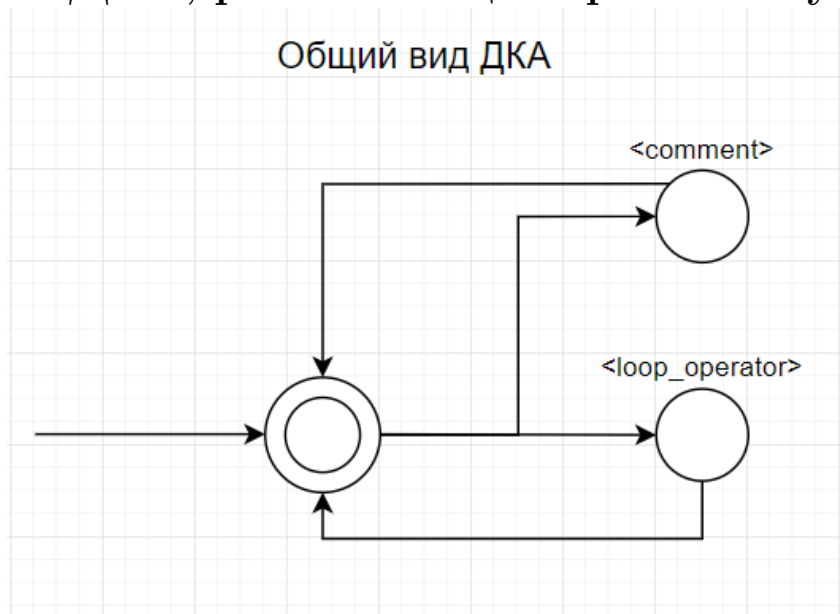
Задания предусматривают **возможность наличия комментариев неограниченной длины** во входном языке. Форма организации комментариев соответствует языку Паскаль.

Входной язык содержит операторы цикла типа **for (...; ...; ...) do**, разделенные символом ; (точка с запятой). Операторы цикла содержат идентификаторы, знаки сравнения <, >, =, римские числа, знак присваивания (:=).

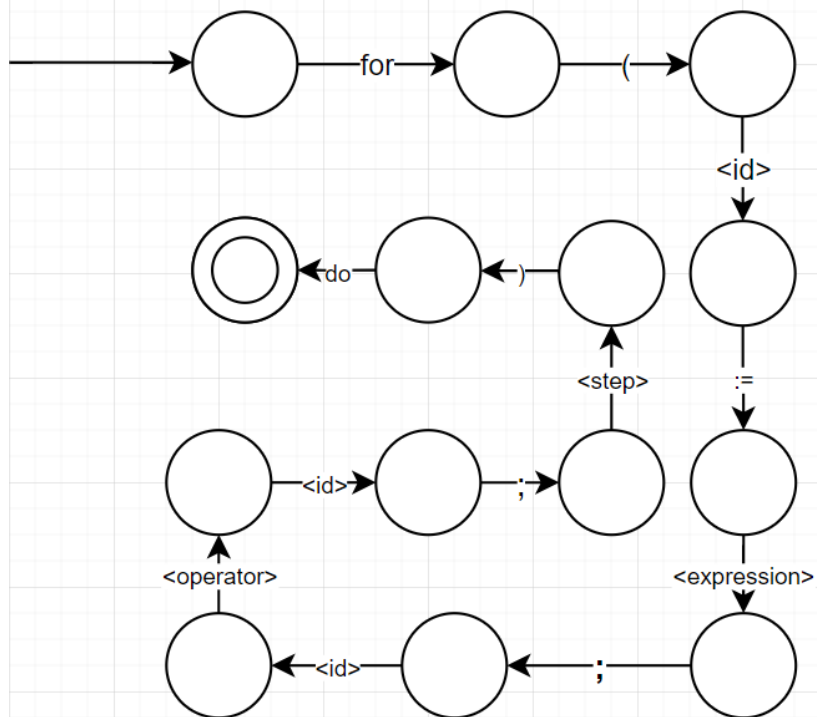
## 2. Грамматика лексем входного языка в форме БНФ.

```
<lang> ::= <loop_operator> ; <lang> | <comment> <lang> | λ  
<comment> ::= //[^\\n ] | "{" [^} ] "  
<loop_operator> ::= for "(" <assign> ";" <compare> ";" <step> ")" do  
<id> ::= <letter> | <letter><number><id> | <letter><id>  
<assign> ::= <id> := <expression>  
<expression> ::= <value> | <id>  
<value> ::= <number>  
<compare> ::= <id> <operator> | <id>  
<operator> ::= < | > | = | <= | >=  
<step> ::= <roman> | <number>  
<letter> ::= a | b | c | ... | z | A | B | C | ... | Z  
<roman> ::= I | X | V  
<number> ::= <roman> | <roman><number>
```

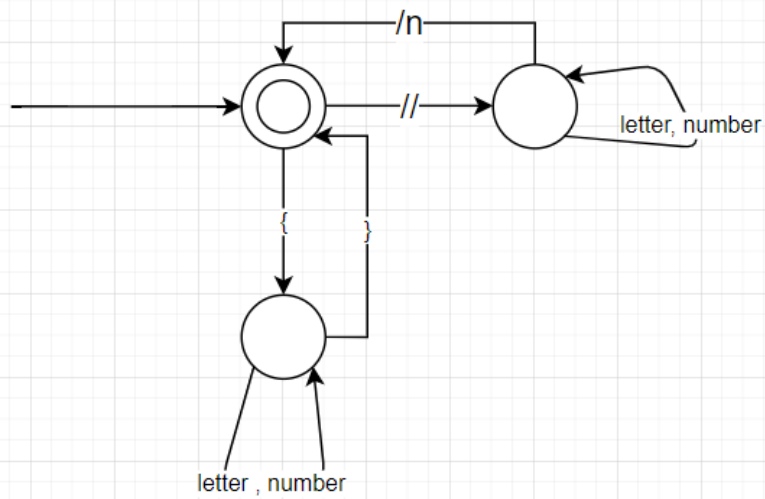
## 3. ДКА, распознающий грамматику.

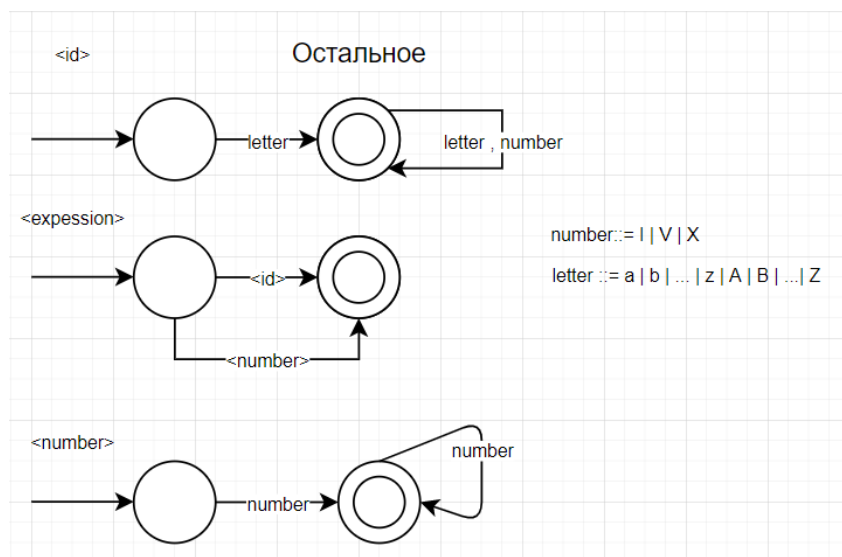


### Блок цикла for



### Блок Комментариев





## 4. Разработка алгоритма синтаксического анализа.

Flex (Fast Lexical Analyzer) — генератор лексических анализаторов. Заменяет Lex в системах на базе пакетов GNU и имеет аналогичную функциональность. При этом Flex не является частью проекта GNU

Lex — это инструмент для лексического анализа, который может использоваться для выделения из исходного текста определенных строк заранее заданным способом. Yacc — это инструмент для грамматического разбора; он читает текст и может использоваться для конвертирования последовательности слов в структурированный формат для дальнейшей обработки.

На входе программа получает текст в свободном формате и правила выделения лексем, а на выходе даёт код анализатора, в виде функции на языке Си.[9]

Правила задаются в виде регулярных выражений слева и, обычно, кода на языке Си справа. Они содержат три секции, отделяющиеся строкой «%%»:

Блок определений

%%

Блок правил

%%

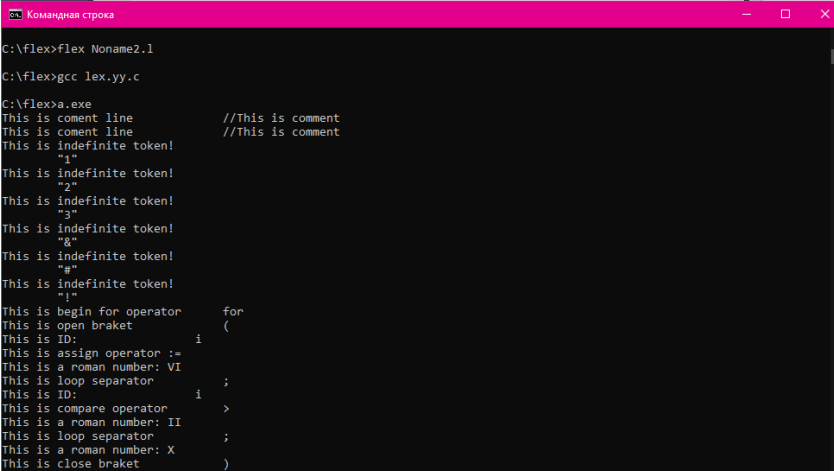
Блок кода на Си

Определения содержат стартовые значения и определения, правила, непосредственно сами выражения и соответствующие им действия; пользовательский код просто включается в вывод flex. Некоторые секции могут отсутствовать.

Функция анализатора получает текст на входе и выполняет заданный код для каждой найденной лексемы.

Что бы воспользоваться данным средством нужно отдельно установить его для Windows или через терминал на Linux с помощью команды. Далее (Будет использоваться система windows 10.0) после написания кода в файле с расширением .l нужно с помощью терминала собрать полученную программу, для этого используем команду flex \*.l, после компилируем полученный C-файл командой gcc lex.yy.c \*.exe (где \* любое имя .exe файла). В итоге, мы получаем .exe файл, который мы можем

вызвать через консоль.



```
C:\flex>flex Noname2.1
C:\flex>gcc lex.yy.c
C:\flex>a.exe
This is coment line      //This is comment
This is coment line      //This is comment
This is indefinite token!
"1"
This is indefinite token!
"2"
This is indefinite token!
"3"
This is indefinite token!
"8"
This is indefinite token!
"#"
This is indefinite token!
"1"
This is begin for operator for
This is open braket (
This is ID: i
This is assign operator :=
This is a roman number: VI
This is loop separator ;
This is ID: i
This is compare operator <
This is a roman number: II
This is loop separator ;
This is a roman number: X
This is close braket )
```

## 5. Тестирование работы алгоритма.

Протестируем работу лексического анализатора

---

Тест номер 1:

```
//This is comment
{This is
    Multi line
    comment}
for(i := I ; i < V ; XV ) do
123 //indefinite token!
```

Результат:

```
This is coment line //This is comment
This is coment line {This is
Multi line
comment}
This is begin for operator for
This is open braket (
This is ID: i
This is assign operator :=
This is a roman number: I
This is loop separator ;
This is ID: i
This is compare operator <
This is a roman number: V
This is loop separator ;
This is a roman number: XV
This is close braket )
This is end for operator do
This is indefinite token!
"1"
This is indefinite token!
"2"
This is indefinite token!
```

"3"

This is coment line //indefinite token!

---

Тест номер 2:

*//This is comment*

*//This is comment*

123 &#!

for(i := VI ; i > II ; X) do

Результат:

This is coment line //This is comment

This is coment line //This is comment

This is indefinite token!

"1"

This is indefinite token!

"2"

This is indefinite token!

"3"

This is indefinite token!

""

This is indefinite token!

""

This is indefinite token!

"!"

This is begin for operator for

This is open braket (

This is ID: i

This is assign operator :=

This is a roman number: VI

This is loop separator ;

This is ID: i

This is compare operator >

This is a roman number: II

This is loop separator ;

This is a roman number: X

This is close braket )

This is end for operator do

---

## 6. Разработанный код

```
%option noyywrap
```

```
%{
```

```
    #include <stdio.h>
```

```
%}
```

```
FORBRA "for"
```

```
FORKET "do"
```

```
LETTER [_a-zA-Z]
```

```
NUMBER [IVX]+
```

```
MOREID {LETTER}|{NUMBER}
```

```
ID {LETTER}{MOREID}*
```

```
OPERATOR "<" | ">" | "="
```

```

ASSIGN ":@"
BRA "("
KET ")"
COMMENT "//" [^\n]* | "{" [^}]* }"
SEPARATOR ";"
%%

[ \t\n]
{FORBRA}      {printf("This is begin for operator      %s\n", yytext ); }
{FORKET}      {printf("This is end for operator      %s\n", yytext ); }
{NUMBER}      {printf("This is a roman number:      %s\n", yytext ); }
{ID}          {printf("This is ID:      %s\n", yytext ); }
{OPERATOR}    {printf("This is compare operator      %s\n", yytext ); }
{ASSIGN}      {printf("This is assign operator      %s\n", yytext ); }
{BRA}         {printf("This is open braket      %s\n", yytext ); }
{KET}         {printf("This is close braket      %s\n", yytext ); }
{COMMENT}     {printf("This is coment line      %s\n", yytext ); }
{SEPARATOR}   {printf("This is loop separator      %s\n", yytext ); }
.             {printf("This is indefinite token! \n\t\"%s\n", yytext );}
%%

int main()
{
    yyin = fopen( "test.txt", "r" );
    yylex();

    return 0;
}

```