```
%option c++

%{
//TODO: спросить про esc посл.
//TODO: сделать лог. операции + тесты к ним. Сделать тест для комментов.
#include <iostream>
%}

WS          [ \t\n\r\v]+
SCOLONE      ";"
COLON        ":"
EQ          "="
NEQ          "/="
ASG          ":="
LPAR         "("
RPAR         ")"
DOT          "."
AMPER        "&"
APOSTR       "'"
NAME         [a-zA-Z_][a-zA-Z0-9_]*
DIGIT        [0-9]
HEXDIGIT     [0-9A-Fa-f]

IF          if
THEN         then
ELSE        else
ELSIF       elsif

FOR          for
LOOP         loop
IN          in
EXIT        exit
```

WHEN when

WHILE while

WITH with

USE use

PROCEDURE procedure

FUNCTION function

BEGIN begin

IS is

END end

RETURN return

PACKAGE package

BODY body

TYPE type

TAGGED tagged

RECORD record

OVERRIDING overriding

NEW new

INTEGERTY Integer

STRINGTY String

CHARACTERTY Character

FLOATTY Float

BOOLTY Boolean

INTEGER {DIGIT}(_?{DIGIT})*

BINARY 2#[01](_?[01])*#

OCT 8#[0-7](_?[0-7])*#

HEX 16#{HEXDIGIT}(_?{HEXDIGIT})*#

FLOAT {INTEGER}\.{INTEGER}([Ee][+-]?{INTEGER})?

```
FALSE        False
TRUE         True
NULL         null

CHAR         \'([^\'\n]|\\.)\'
STRING       \"([^\"\n]|\"\")*\"

%x COMMENT

PLUS         "+"
MINUS        "-"
DIV          "/"
MUL          "*"
LESS         "<"
MORE         ">"

OR           or
NOT          not
AND          and

NEWLINE      New_Line
PUTLINE      Put_Line
IMAGE        Image

%%

"--"             { BEGIN(COMMENT); }   /* вошли в состояние COMMENT */

<COMMENT>.*        {
                    std::cout << "COMMENT " << yytext << std::endl;
                    BEGIN(INITIAL);
                   }
```

```
{WS}          /* игнорируем пробелы и переводы строк */

{SCOLONE}      { std::cout << "SCOLONE ;" << std::endl; }
{COLON}        { std::cout << "COLON :" << std::endl; }
{EQ}           { std::cout << "EQ =" << std::endl; }
{NEQ}          { std::cout << "NEQ /=" << std::endl; }
{ASG}          { std::cout << "ASG :=" << std::endl; }
{LPAR}         { std::cout << "LPAR (" << std::endl; }
{RPAR}         { std::cout << "RPAR )" << std::endl; }
{DOT}          { std::cout << "DOT ." << std::endl; }
{AMPER}        { std::cout << "AMPER &" << std::endl; }
{APOSTR}       { std::cout << "APOSTR '" << std::endl; }

{WHILE}        { std::cout << "WHILE " << yytext << std::endl; }
{ELSE}         { std::cout << "ELSE " << yytext << std::endl; }
{IF}           { std::cout << "IF " << yytext << std::endl; }
{THEN}         { std::cout << "THEN " << yytext << std::endl; }
{ELSIF}        { std::cout << "ELSIF " << yytext << std::endl; }
{FOR}          { std::cout << "FOR " << yytext << std::endl; }
{LOOP}         { std::cout << "LOOP " << yytext << std::endl; }
{IN}           { std::cout << "IN " << yytext << std::endl; }
{EXIT}         { std::cout << "EXIT " << yytext << std::endl; }
{WHEN}         { std::cout << "WHEN " << yytext << std::endl; }
{WITH}         { std::cout << "WITH " << yytext << std::endl; }
{USE}          { std::cout << "USE " << yytext << std::endl; }
{PROCEDURE}    { std::cout << "PROCEDURE " << yytext << std::endl; }
{FUNCTION}     { std::cout << "FUNCTION " << yytext << std::endl; }
{BEGIN}        { std::cout << "BEGIN " << yytext << std::endl; }
{IS}           { std::cout << "IS " << yytext << std::endl; }
{END}          { std::cout << "END " << yytext << std::endl; }
{RETURN}       { std::cout << "RETURN " << yytext << std::endl; }
```

```
{PACKAGE}      { std::cout << "PACKAGE " << yytext << std::endl; }
{BODY}         { std::cout << "BODY " << yytext << std::endl; }
{TYPE}         { std::cout << "TYPE " << yytext << std::endl; }
{TAGGED}       { std::cout << "TAGGED " << yytext << std::endl; }
{RECORD}       { std::cout << "RECORD " << yytext << std::endl; }
{OVERRIDING}   { std::cout << "OVERRIDING " << yytext << std::endl; }
{NEW}          { std::cout << "NEW " << yytext << std::endl; }

{INTEGERTY}    { std::cout << "INTEGERTY " << yytext << std::endl; }
{STRINGTY}     { std::cout << "STRINGTY " << yytext << std::endl; }
{CHARACTERTY}  { std::cout << "CHARACTERTY " << yytext << std::endl; }
{FLOATTY}      { std::cout << "FLOATTY " << yytext << std::endl; }
{BOOLTY}       { std::cout << "BOOLTY " << yytext << std::endl; }

{INTEGER}      { std::cout << "INTEGER " << yytext << std::endl; }
{BINARY}       { std::cout << "BINARY " << yytext << std::endl; }
{OCT}          { std::cout << "OCT " << yytext << std::endl; }
{HEX}          { std::cout << "HEX " << yytext << std::endl; }
{FLOAT}        { std::cout << "FLOAT " << yytext << std::endl; }
{CHAR}         { std::cout << "CHAR " << yytext << std::endl; }
{STRING}       { std::cout << "STRING " << yytext << std::endl; }
{FALSE}        { std::cout << "FALSE " << yytext << std::endl; }
{TRUE}         { std::cout << "TRUE " << yytext << std::endl; }
{NULL}         { std::cout << "NULL " << yytext << std::endl; }

{PLUS}         { std::cout << "PLUS +" << std::endl; }
{MINUS}        { std::cout << "MINUS -" << std::endl; }
{DIV}          { std::cout << "DIV /" << std::endl; }
{MUL}          { std::cout << "MUL *" << std::endl; }
{LESS}         { std::cout << "LESS <" << std::endl; }
{MORE}         { std::cout << "MORE >" << std::endl; }
```

```
{OR}          { std::cout << "OR " << yytext << std::endl; }
{NOT}         { std::cout << "NOT " << yytext << std::endl; }
{AND}         { std::cout << "AND " << yytext << std::endl; }

{NEWLINE}     { std::cout << "NEWLINE " << yytext << std::endl; }
{PUTLINE}     { std::cout << "PUTLINE " << yytext << std::endl; }
{IMAGE}       { std::cout << "IMAGE " << yytext << std::endl; }

{NAME}        { std::cout << "NAME " << yytext << std::endl; }

.             { std::cout << "ERROR " << yytext << std::endl; }


%%
```