```
options {
  IGNORE_CASE = true;
}

PARSER_BEGIN(AnalisadorLexico)

public class AnalisadorLexico {
   public static void main(String[] args) {}
}

PARSER_END(AnalisadorLexico)

TOKEN :
{
  // Palavras Reservadas
< BEGIN: "begin" >
| < DEFINE: "define" >
| < START: "start" >
| < END: "end" >
| < NUM: "num" >
| < REAL: "real" >
| < TEXT: "text" >
| < FLAG: "flag" >
| < SET: "set" >
| < READ: "read" >
| < SHOW: "show" >
| < IF: "if" >
| < THEN: "then" >
| < ELSE: "else" >
| < TRUE: "true" >
| < FALSE: "false" >
| < LOOP: "loop" >
| < WHILE: "while" >

  // Símbolos Especiais
| < ATRIBUICAO: "=" >
| < PONTO_E_VIRGULA: ";" >
| < VIRGULA: "," >
| < ABRE_PARENTESES: "(" >
| < FECHA_PARENTESES: ")" >
| < ABRE_COLCHETES: "[" >
| < FECHA_COLCHETES: "]" >
| < ABRE_CHAVES: "{" >
| < FECHA_CHAVES: "}" >
| < MAIS: "+" >
| < MENOS: "-" >
| < VEZES: "*" >
| < DIVIDIR: "/" >
```

```
| < POTENCIA: "**" >
| < DIVISAO_INTEIRA: "%" >
| < RESTO_DIVISAO: "%%" >
| < IGUAL: "==" >
| < DIFERENTE: "!=" >
| < MENOR: "<<" >
| < MAIOR: ">>" >
| < MENOR_OU_IGUAL: "<<=" >
| < MAIOR_OU_IGUAL: ">>=" >
| < E: "&" >
| < OU: "|" >
| < NAO: "!" >

  //Identificador
| < IDENTIFICADOR: ( <LETRA> | "_" ) ( <LETRA> | <DIGITO> <LETRA>| <DIGITO> "_" | "_" )* >

  // Constantes
| < CONSTANTE_INTEIRA: (<DIGITO> (<DIGITO>)? (<DIGITO>)?) >
| < CONSTANTE_REAL: (<DIGITO> (<DIGITO>)? (<DIGITO>)? (<DIGITO>)?) "." (<DIGITO>
(<DIGITO>)?)>
| < CONSTANTE_LITERAL: <CL_ASPAS_DUPLAS> | <CL_ASPAS_SIMPLES> >
| < CONSTANTE_LOGICA: ( "true" | "false" ) >
| < LITERAL_NAO_FECHADO: "\"" (<ASCII>)* ( "\n" | "\r" | "\r\n" | "\u0000" )
                | "'"  (<ASCII>)* ( "\n" | "\r" | "\r\n" | "\u0000" ) >


  // Tokens de apoio das definições regulares
| < #LETRA: [ "A"-"Z", "a"-"z" ] >
| < #DIGITO: [ "0"-"9" ] >
| < #CL_ASPAS_DUPLAS: "\"" (<ASCII>)* "\"" >
| < #CL_ASPAS_SIMPLES: "'" (<ASCII>)* "'" >
| < #ASCII: ~["\u0000" - "\u001F", "\u007F"] >

}

SKIP :
{
  < ESPACO_BRANCO: ( " " | "\t" | "\n" | "\r" )+ >
  | < COMENTARIO_LINHA: ("//"(<ASCII>)*) >
  | < COMENTARIO_BLOCO: ("/*"(<ASCII>)* "*/") >
}

TOKEN :{
  //tratamento de erros
< CONST_INT_INVALIDA: (<DIGITO><DIGITO><DIGITO>(<DIGITO>)+)>

|<CONST_REAL_INVALIDA: (<DIGITO>)+ "." (<DIGITO>)* ("." (<DIGITO>)+)+
   |(<DIGITO><DIGITO><DIGITO><DIGITO>(<DIGITO>)+"."<DIGITO><DIGITO>(<DIGITO>)+)
```

```
   |(<DIGITO> (<DIGITO>)? (<DIGITO>)? (<DIGITO>)?"."<DIGITO><DIGITO>(<DIGITO>)+)
   |(<DIGITO><DIGITO><DIGITO><DIGITO>(<DIGITO>)+"."<DIGITO> (<DIGITO>)?)>

|< IDENTIFICADOR_INVALIDO:
     ( <DIGITO> ( <LETRA> | <DIGITO> | "_" )* )
   | ( ( <LETRA> | "_" ) ( <LETRA> | <DIGITO> | "_" )* <DIGITO> )
   | ( ( <LETRA> | "_" ) ( <LETRA> | "_" )* <DIGITO>(<DIGITO>)+ ( <LETRA> | <DIGITO> | "_" )* )>

| < COMENTARIO_NAO_FECHADO: "/*" (~["*"])* >

|< SIMBOLO_INVALIDO: ~[] >
}
```