Model language

```
<global_declaration>> → {<global_declaration>}
<global declaration> → func <function> | array of <array> | <variable>
\langle array \rangle \rightarrow \langle type \rangle \langle array size \rangle \langle id \rangle \{, \langle id \rangle \}
\langle \text{variable} \rangle \rightarrow \langle \text{type} \rangle \langle \text{id} \rangle [\langle \text{init} \rangle] 
<declaration> → <u>array of</u> <array> | <variable>
\langle array \ size \rangle \rightarrow [ \langle num \rangle_{\lambda} \rangle \langle num \rangle_{\lambda}]
\langle \text{init} \rangle \rightarrow = \langle \text{constant} \rangle
\langle type \rangle \rightarrow \underline{int} \mid \underline{real} \mid \underline{bool} \mid \underline{str} \mid \underline{char}
<function> → <type> <id> <arguments> <body>
<main_arguments> \rightarrow ( [< type > < id > ] {, < type > < id >} )
\langle arguments \rangle \rightarrow ([\langle type \rangle \langle id \rangle] \{, \langle type \rangle \langle id \rangle\})
\langle body \rangle \rightarrow \{ \{body operator\} \}
<block> \rightarrow : | {<}operator > } | <operator >
<br/>
<br/>
declaration> | < operator > <br/>
<br/>
<br/>
<br/>
<br/>
declaration> | < operator > <br/>
<br
                                                 <u>;</u> | <u>if</u> <operator_if> | <u>while</u> <operator_while> |
<operator > →
                                                  <u>do</u> <operator do> | <u>for</u> <operator for> | <u>switch</u> <operator switch> |
                                                  <u>read</u> <operator read> | <u>write</u> <operator write> |
                                                  break<operator break> | continue < operator continue> |
                                                  return < operator return> | < operator simple>
\langle operator simple \rangle \rightarrow \langle mono1 \rangle : | = \langle expression \rangle
\langle operator if \rangle \rightarrow (\langle expression \rangle) \langle block \rangle [\underline{else} \langle block \rangle] |
<operator while> → (<expression>) <block> |
<operator do> \rightarrow <block> <u>while</u> (<expression>)
\langle operator for \rangle \rightarrow ([\langle expression \rangle]; [\langle expression \rangle]; [\langle expression \rangle]) \langle block \rangle
\langle \text{operator read} \rangle \rightarrow (\langle \text{identificator} \rangle )
\langle operator write \rangle \rightarrow (\langle expression \rangle \{ , \langle expression \rangle \} )
<operator return> → <expression>
<operator break> →
<operator continue> →
\langle index \rangle \rightarrow [\langle expression \rangle \{ \langle expression \rangle \}]
<expression> \rightarrow <value> \{ \equiv <value> \}
\langle value \rangle \rightarrow \langle or value \rangle \{ or value \rangle \}
\langle \text{or value} \rangle \rightarrow \langle \text{and value} \rangle  and value\langle \text{and value} \rangle 
<and value> \rightarrow [ not ] <bool value> \{<bool operation> <bool value> \}
<bool value> \rightarrow <add value> \{ + | - <add value> \}
< add value > \rightarrow < mul value > {* | / < mul value > }
<mul value> \rightarrow [+ | -] (<expression>) | <sign>
< mono > \rightarrow < mono 1 > | < constant >
<mono1> \rightarrow <id> <call func> | <var ident> |
\langle \text{var ident} \rangle \rightarrow [\langle \text{index} \rangle]
<call func> \rightarrow (<expression> \{, <expression> \})
```