Producciones de la Gramática en formato BNF

Identificadores para variables y números

integer ::= [0 - 9]+
number ::= -?[0-9]+(\.[0-9]+)?
string ::= \"[^\"]*\"
varName ::= [a-zA-Z][a-zA-Z0-9]*

Producciones

El símbolo distinguido de la gramática es **entrypoint**

```
entrypoint::=  START hyperstatements END | START END

hyperstatements::= hyperstatement hyperstatements | hyperstatement

hyperstatement::= statement SEMICOLON | block | ifsentence  | while

inblockstatements::= inblockstatement inblockstatements | inblockstatement

inblockstatement::=  statement SEMICOLON | ifsentence | while

ifsentence::=           IF OPEN_P generalexpression CLOSE_P block
                        | IF OPEN_P generalexpression CLOSE_P block elsetrain

elsetrain::=     ELSE block
                | ELSE_IF OPEN_P generalexpression CLOSE_P block
                | ELSE_IF OPEN_P generalexpression CLOSE_P block elsetrain

while::=         WHILE OPEN_P generalexpression CLOSE_P block

statement::=            generalexpression
                        | assignment
                        | vardeclaration
                        | vardeclassignment
                        | foreach | print | getfunctions | exit

vardeclaration::=       type VAR
                        | type  VAR OPEN_BRACK NUMBER_LITERAL CLOSE_BRACK


vardeclassignment::=          type VAR ASSIGN_EQ generaloperation
```

| type VAR ASSIGN_EQ literal
| type VAR ASSIGN_EQ generalexpression
| type VAR ASSIGN_EQ arrayliteral
| type VAR ASSIGN_EQ getfunctions

print::= PRINT unity | PRINT literal

getfunctions: GET_INT OPEN_P CLOSE_P
| GET_DOUBLE OPEN_P CLOSE_P
| GET_STRING OPEN_P CLOSE_P

foreach::= VAR DOT FOREACH OPEN_P VAR RIGHT_ARROW foreachbody CLOSE_P

foreachbody::= statement | block

block::= OPEN_B inblockstatements CLOSE_B

assignment::= VAR ASSIGN_EQ literal
| VAR ASSIGN_EQ generaloperation
| VAR ASSIGN_EQ generalexpression
| arraccess ASSIGN_EQ generalexpression
| arraccess ASSIGN_EQ generaloperation
| VAR ASSIGN_EQ getfunctions
| arraccess ASSIGN_EQ getfunctions

arraccess::= VAR OPEN_BRACK expunity CLOSE_BRACK

literal::= STRING_LITERAL

arrayliteral::= OPEN_BRACK numlist CLOSE_BRACK

numlist::=   numlist COMMA NUMBER_LITERAL | NUMBER_LITERAL

type::=INT | STR | DOUBLE | INT_ARR | DOUBLE_ARR

generalexpression::= generalexpression AND expression
| generalexpression OR expression
| expression

expression::= expunity EQ expunity
| expunity GT expunity
| expunity GE expunity
| expunity LT expunity
| expunity LE expunity
| expunity NE expunity
| NOT expunity

expunity::= VAR
| NUMBER_LITERAL
| arraccess

generaloperation::= operation
| operation ADD generaloperation
| operation SUBS generaloperation

operation ::= unity
| unity PROD operation
| unity DIV operation
| unity CROSS operation
| unity SPROD operation
| SUBS operation

unity ::= VAR
| NUMBER_LITERAL
| arraccess

exit::= EXIT

IF::= “if”
ELSE_IF ::= “else if”
ELSE ::= “else”
WHILE ::= “while”
COMMA ::= “,”
SEMICOLON ::= “;”
DOT ::= “.”
INT ::= “int”
INT_ARR ::= “int[]”
STR ::= “str”
DOUBLE ::= “double”
DOUBLE_ARR ::= “double[]”
FOREACH ::= “forEach”
RIGHT_ARROW ::= “->”
ASSIGN_EQ ::= “=”
EQ ::= “==”
GT ::= “>”
GE ::= “>=”
LT ::= “<”
LE ::= “<=”
NE ::= “!=”
ADD ::= “+”
SUBS ::= “-”
SPROD ::= “*.”

PROD ::= “*”
DIV ::= “/”
AND ::= “&&”
OR ::= “||”
NOT ::= “!”
OPEN_B ::= “{“
CLOSE_B ::= “}”
OPEN_P ::= “(“
CLOSE_P ::= “)”
OPEN_BRACK ::= “[“
CLOSE_BRACK ::= “]”
PRINT ::= “print”
START ::= “start”
END ::= “end”
CROSS ::= “*x”
GET_INT ::= “getInt”
GET_DOUBLE ::= “getDouble”
GET_STRING ::= “getString”
EXIT ::= “exit”