/////////////////////////////////////////Syntax////////////////////////////////////////////////////////

double media\_aritmetica ( int|double num1 , int|double num2, #arg)

{

double result=num1+num2;

for item in #arg

{

result+=item;

}

result/=sup.len+2;

return result;

}

main(){

print(media\_aritmetica(1 , 2 , 3 , 4 , 5 , 6.5 , 8.3));

}

///////////////////////////////gramatica///////////

<basic symbol>::= <letter> | <digit> | <logical value> | <special symbol> | <delimiter>

<letter> ::= A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | Y | W | X | Y | Z

<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<logical value> ::= TRUE | FALSE

<special symbol> ::= <any symbol in CDC 64-charaeter set>

<delimiter> ::= <operator> | <separator> | <bracket> | <declarator> | <specificator >

<operator> ::= <arithmetic operator> | <relational operator> | <logical operator > | <sequential operator>

<arithmetic operator> ::= + | - | \* | /

<relational operator> ::= < | <= | = | >= | > | ~=

<bracket> ::= ) | ( | ] | [ | } | {

<declarator> ::= OWN | BOOLEAN | INTEGER | REAL | ARRAY | STRING | LIST

<identifier> ::= <letter> | <identifier><letter> | < identifier><digit>

<unsigned integer> ::= <digit> | <unsigned integer><digit>

<integer> ::= <unsigned integer> | + <unsigned integer> | - <unsigned integer>

<decimal fraction> ::= .<unsigned integer>

<decimal number> ::= <unsigned integer> | <decimal fraction> | <unsigned integer><decimal fraction>

<number> ::= <unsigned number> | + <unsigned number > | - <unsigned number >

<expression> ::= <arithmetic expression> | <Boolean expression> | <designational expression>

<adding operator> ::= + | -

< multiplying operator> ::= \* | /

<primary> ::= <unsigned number> | <variable> | <function designator> | (<arithmetic expression>)

<factor> ::= <primary> | <factor> \*\* <primary> | <factor> ^ <primary>

<term> ::= <factor> | <term> <multiplying operator> <factor>

<simple arithmetic> ::= <term> | <adding operator > <term> | <simple arithmetic> <adding operator> <term>

<if clause> ::= IF <Boolean expression> THEN

<arithmetic expression> ::= <simple arithmetic> | <if clause> <simple arithmetic> #ELSE# <arithmetic expression>

<relational operator > ::= < | <= | = | >= | > | ~=

<function statement>::=<data type> <identifier><Left\_paranthesis> <list of parameters> <Right\_paranthesis> <body>

Gramatica V2

primaryExpression

: Identifier

| Constant

| StringLiteral+

| '(' expression ')'

;

multiplicativeExpression

: castExpression

| multiplicativeExpression '\*' castExpression

| multiplicativeExpression '/' castExpression

| multiplicativeExpression '%' castExpression

;

relationalExpression

: relationalExpression '<' shiftExpression

| relationalExpression '>' shiftExpression

| relationalExpression '<=' shiftExpression

| relationalExpression '>=' shiftExpression

;

equalityExpression

: relationalExpression

| equalityExpression '==' relationalExpression

| equalityExpression '!=' relationalExpression

;

andExpression

: equalityExpression

| andExpression '&' equalityExpression

;

exclusiveOrExpression

: andExpression

| exclusiveOrExpression '^' andExpression

;

inclusiveOrExpression

: exclusiveOrExpression

| inclusiveOrExpression '|' exclusiveOrExpression

;

logicalAndExpression

: inclusiveOrExpression

| logicalAndExpression '&&' inclusiveOrExpression

;

logicalOrExpression

: logicalAndExpression

| logicalOrExpression '||' logicalAndExpression

;

conditionalExpression

: logicalOrExpression ('?' expression ':' conditionalExpression)?

;

assignmentOperator

: '=' | '\*=' | '/=' | '%=' | '+=' | '-='

;

expression

: assignmentExpression

| expression ',' assignmentExpression

;

constantExpression

: conditionalExpression

;

declaration

: declarationSpecifiers initDeclaratorList ';'

| declarationSpecifiers ';'

| staticAssertDeclaration

;