Grammar

**Token list:**

* block
* starting
* receives
* block\_returns
* call
* variable
* list
* of
* type
* return\_statement
* do
* until
* if
* else
* colon
* semicolon
* comma
* input
* print
* curlybraces\_open
* curlybraces\_close
* parenthesis\_open
* parenthesis\_close
* squarebracket\_open
* squarebracket\_close
* op\_assign
* op\_less
* op\_less\_equal
* op\_greater
* op\_greater\_equal
* op\_equal
* op\_not\_equal
* op\_and
* op\_or
* op\_negation
* op\_addition
* op\_subtraction
* op\_multiplication
* op\_division
* whole
* decimal
* words
* boolean
* cst\_whole
* cst\_decimal
* cst\_words
* cst\_boolean
* id

|  |  |
| --- | --- |
| **Syntactic Rule** | **Context Free Grammar (LR Parsing)** |
| **PROGRAM** | **PROGRAM’** |
| **PROGRAM’** | **BLOCK**  | **PROGRAM’ BLOCK** |
| **BLOCK** | **BLOCK’** block id **RECEIVES\_AUX RETURNS\_AUX BLOCK\_BODY** |
| **BLOCK’** | starting  | ɛ |
| **RECEIVES\_AUX** | receives colon id of type **TYPE** **RECEIVES\_AUX’**  | ɛ |
| **RECEIVES\_AUX’** | comma id of type **TYPE** **RECEIVES\_AUX’**  | ɛ |
| **RETURNS\_AUX** | block\_returns **TYPE**  | ɛ |
| **BLOCK\_BODY** | curlybraces\_open **BLOCK\_BODY’** curlybraces\_close |
| **BLOCK\_BODY’** | **DECLARATIONS** **BLOCK\_BODY’**  | **BLOCK\_BODY’’** |
| **BLOCK\_BODY’’** | **STATEMENT BLOCK\_BODY’’**  | ɛ |
| **TYPE** | whole  | decimal  | words  | boolean |
| **BODY** | curlybraces\_open **BODY’** curlybraces\_close |
| **BODY’** | **STATEMENT BODY’**  | ɛ |
| **CONDITION** | if parenthesis\_open **EXPRESSION** parenthesis\_close **BODY CONDITION’** |
| **CONDITION’** | else **BODY**  | ɛ |
| **CONSTANT** | id **CONSTANT’**  | cst\_whole  | cst\_decimal  | cst\_words  | cst\_boolean |
| **CONSTANT’** | squarebracket\_open **ITEM** squarebracket\_close  | parenthesis\_open **EXPRESSION CONSTANT’’** parenthesis\_close  | ɛ |
| **CONSTANT’’** | comma **EXPRESSION CONSTANT’’**  | ɛ |
| **DECLARATIONS** | **VAR\_DECLARATION DECLARATION**  | **LIST\_DECLARATION DECLARATION**  |ɛ |
| **VAR\_DECLARATION** | variable id VAR\_DECLARATION’ of type **TYPE** semicolon |
| **VAR\_DECLARATION’** | comma id **VAR\_DECLARATION’**  | ɛ |
| **LIST\_DECLARATION** | list id squarebracket\_open **ITEM** squarebracket\_close of type **TYPE** semicolon |
| **EXPRESSION** | **EXPRESSION’ EXP EXPRESSION’’** |
| **EXPRESSION’** | op\_negation  | ɛ |
| **EXPRESSION’’** | op\_and **EXPRESSION’ EXP**  | op\_or **EXPRESSION’ EXP**  **|** ɛ |
| **EXP** | **ITEM EXP’** |
| **EXP’** | op\_less **ITEM**  | op\_less\_equal **ITEM**  | op\_greater **ITEM**  | op\_greater\_equal **ITEM**  | op\_equal **ITEM**  | op\_not\_equal **ITEM**  | ɛ |
| **FACTOR** | parenthesis\_open **EXPRESSION** parenthesis\_close  | **FACTOR’** |
| **FACTOR’** | op\_addition **CONSTANT**  | op\_subtraction **CONSTANT**  | **CONSTANT** |
| **ITEM** | **TERM** **ITEM’** |
| **ITEM’** | op\_addition **ITEM**  | op\_subtraction **ITEM**  | ɛ |
| **TERM** | **FACTOR TERM’** |
| **TERM’** | op\_multiplication **TERM**  | op\_division **TERM**  | ɛ |
| **STATEMENT** | **ASSIGN**  | **CONDITION**  | **READ**  | **WRITE**  | **LOOP**  | **RETURN**  | **CALL** |
| **CALL** | call id parenthesis\_open **EXPRESSION CALL’** parenthesis\_close semicolon |
| **CALL’** | comma **EXPRESSION CALL’**  | ɛ |
| **LOOP** | do **BODY** until parenthesis\_open **EXPRESSION** parenthesis\_close |
| **ASSIGN** | id **ASSIGN’** op\_assign **EXPRESSION** semicolon |
| **ASSIGN’** | squarebracket\_open **ITEM** squarebracket\_close |
| **RETURN** | return\_statement **EXPRESSION** semicolon |
| **READ** | input parenthesis\_open id **READ’** parenthesis\_close semicolon |
| **READ’** | squarebracket\_open **ITEM** squarebracket\_close  | ɛ |
| **WRITE** | print parenthesis\_open **EXPRESSION WRITE’** parenthesis\_close semicolon |
| **WRITE’** | comma **EXPRESSION WRITE’**  | ɛ |