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\_AUX** |
| **PROGRAM\_AUX** | **BLOCK**  | **PROGRAM\_AUX BLOCK** |
| **BLOCK** | **BLOCK\_AUX** block id **RECEIVES\_AUX RETURNS\_AUX BLOCK\_BODY** |
| **BLOCK\_AUX** | starting  | ɛ |
| **RECEIVES\_AUX** | receives colon id of type **TYPE** **RECEIVES\_AUX1**  | ɛ |
| **RECEIVES\_AUX1** | comma id of type **TYPE** **RECEIVES\_AUX1**  | ɛ |
| **RETURNS\_AUX** | block\_returns **TYPE**  | ɛ |
| **BLOCK\_BODY** | curlybraces\_open **BLOCK\_BODY\_AUX** curlybraces\_close |
| **BLOCK\_BODY\_AUX** | **DECLARATIONS** **BLOCK\_BODY\_AUX**  | **BLOCK\_BODY\_AUX1** |
| **BLOCK\_BODY\_AUX1** | **STATEMENT BLOCK\_BODY\_AUX1**  | ɛ |
| **TYPE** | whole  | decimal  | words  | boolean |
| **BODY** | curlybraces\_open **BODY\_AUX** curlybraces\_close |
| **BODY\_AUX** | **STATEMENT BODY\_AUX**  | ɛ |
| **CONDITION** | if parenthesis\_open **EXPRESSION** parenthesis\_close **BODY CONDITION\_AUX** |
| **CONDITION\_AUX** | else **BODY**  | ɛ |
| **CONSTANT** | id **CONSTANT\_AUX**  | cst\_whole  | cst\_decimal  | cst\_words  | cst\_boolean |
| **CONSTANT\_AUX** | squarebracket\_open **ITEM** squarebracket\_close  | parenthesis\_open **CONSTANT\_AUX1** parenthesis\_close  | ɛ |
| **CONSTANT\_AUX1** | **EXPRESSION CONSTANT\_AUX2**  | ɛ |
| **CONSTANT\_AUX2** | comma **EXPRESSION CONSTANT\_AUX2**  | ɛ |
| **DECLARATIONS** | **VAR\_DECLARATION**  **| LIST\_DECLARATION** |
| **VAR\_DECLARATION** | variable id **VAR\_DECLARATION\_AUX** of type **TYPE** semicolon |
| **VAR\_DECLARATION\_AUX** | comma id **VAR\_DECLARATION\_AUX**  | ɛ |
| **LIST\_DECLARATION** | list id squarebracket\_open **cst\_whole** squarebracket\_close of type **TYPE** semicolon |
| **EXPRESSION** | **EXPRESSION\_AUX EXP EXPRESSION\_AUX1** |
| **EXPRESSION\_AUX** | op\_negation  | ɛ |
| **EXPRESSION\_AUX1** | **op\_and EXPRESSION**  **| op\_or EXPRESSION**  **| ɛ** |
| **EXP** | **ITEM EXP\_AUX** |
| **EXP\_AUX** | 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\_AUX** |
| **FACTOR\_AUX** | **op\_addition CONSTANT**  **| op\_subtraction CONSTANT**  **| CONSTANT** |
| **ITEM** | **TERM ITEM\_AUX** |
| **ITEM\_AUX** | op\_addition **ITEM**  | op\_subtraction **ITEM**  | ɛ |
| **TERM** | **FACTOR TERM\_AUX** |
| **TERM\_AUX** | op\_multiplication **TERM**  | op\_division **TERM**  | ɛ |
| **STATEMENT** | **ASSIGN**  **| CONDITION**  **| READ**  **| WRITE**  **| LOOP**  **| RETURN**  **| CALL** |
| **CALL** | call id parenthesis\_open **CALL\_AUX** parenthesis\_close semicolon |
| **CALL\_AUX** | **EXPRESSION CALL\_AUX2**  | ɛ |
| **CALL\_AUX2** | comma **EXPRESSION CALL\_AUX2**  | ɛ |
| **LOOP** | do **BODY** until parenthesis\_open **EXPRESSION** parenthesis\_close |
| **ASSIGN** | id **ASSIGN\_AUX** op\_assign **EXPRESSION** semicolon |
| **ASSIGN\_AUX** | squarebracket\_open **ITEM** squarebracket\_close  | ɛ |
| **RETURN** | return\_statement **EXPRESSION** semicolon |
| **READ** | input parenthesis\_open id **READ\_AUX** parenthesis\_close semicolon |
| **READ\_AUX** | squarebracket\_open **ITEM** squarebracket\_close  | ɛ |
| **WRITE** | print parenthesis\_open **EXPRESSION WRITE\_AUX** parenthesis\_close semicolon |
| **WRITE\_AUX** | comma **EXPRESSION WRITE\_AUX**  | ɛ |