# Léxico y gramática del lenguaje *Hivar*

Roberta González Garza - A01570010
Mariano García Alipi - A00822247

9 de abril de 2021

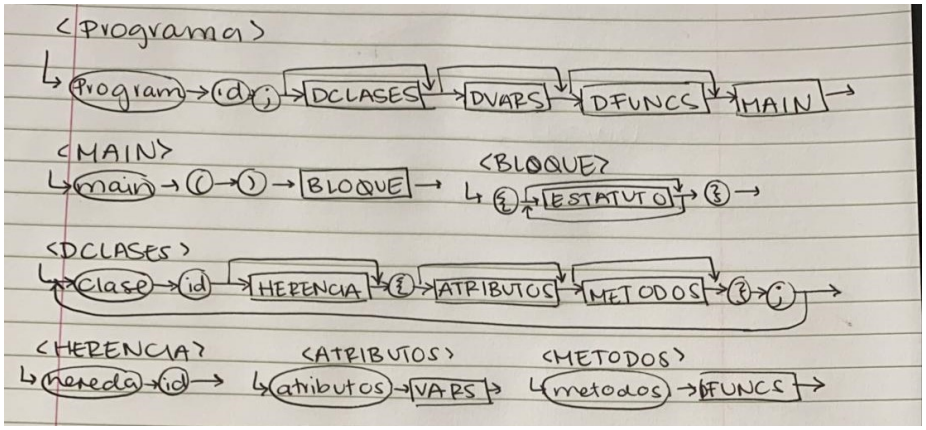
## Léxico

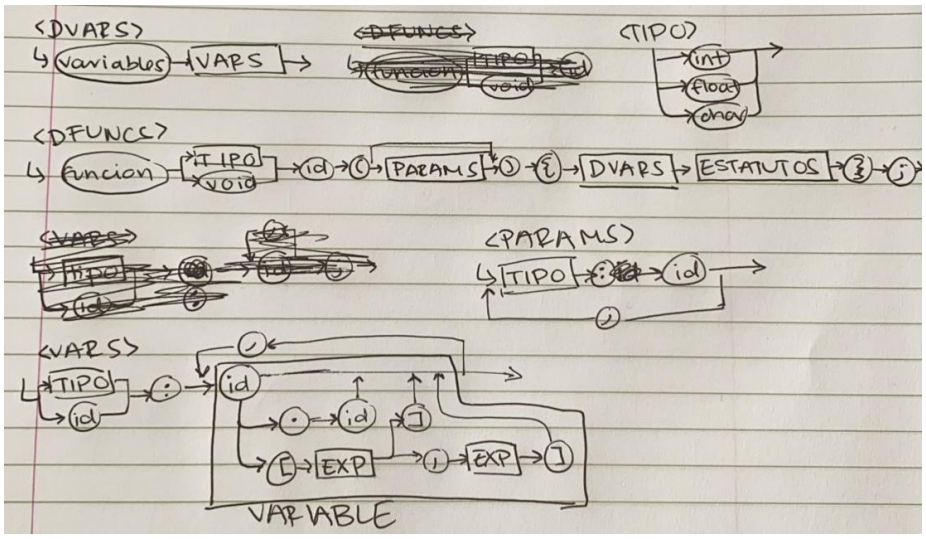
| Token | Expresión regular | Ejemplo |
|---|---|---|
| PROGRAM_KEYWORD | program | program |
| MAIN_KEYWORD | main | main |
| CLASS_KEYWORD | class | class |
| INHERITS | inherits | inherits |
| ATTRIBUTES_KEYWORD | attributes | attributes |
| VARS_KEYWORD | variables | variables |
| END_VARS | byevar | byevar |
| METHODS_KEYWORD | methods | methods |
| FUNCTION | function | function |
| RETURN | return | return |
| READ | read | read |
| WRITE | write | write |
| INT | int | int |
| FLOAT | float | float |
| CHAR | char | char |
| VOID | void | void |
| IF | if | if |
| ELSIF | elsif | elsif |
| ELSE | else | else |
| WHILE | while | while |
| DO | do | do |
| FROM | from | from |
| TO | to | to |
| COMMA | , | , |
| PERIOD | \. | . |
| COLON | : | : |
| SEMICOLON | ; | ; |
| LEFT_PARENTHESIS | \( | ( |
| RIGHT_PARENTHESIS | \) | ) |
| LEFT_CURLY | \{ | { |
| RIGHT_CURLY | \} | } |
| LEFT_BRACKET | \[ | [ |
| RIGHT_BRACKET | \] | ] |
| NOT_EQUALS | != | != |
| EQUALS_COMPARISON | == | == |
| LESS_EQUALS | <= | <= |
| GREATER_EQUALS | >= | >= |

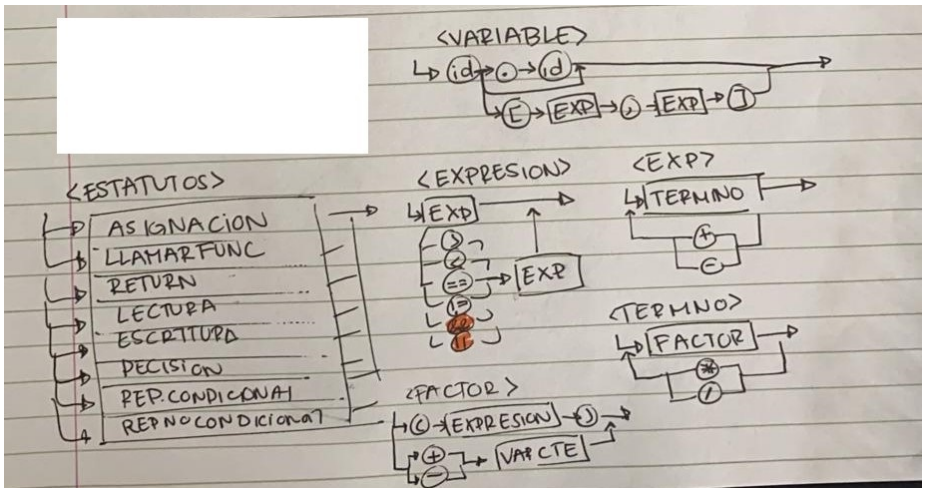
| Token | Expresión regular | Ejemplo |
|---|---|---|
| LESS_THAN | < | < |
| GREATER_THAN | > | > |
| EQUALS_ASSIGNMENT | = | = |
| PLUS | \+ | + |
| MINUS | \- | - |
| MULTIPLY | * | * |
| DIVIDE | \/ | / |
| AND | && | && |
| OR | \|\| | \|\| |
| ID | [a-zA-Z][a-zA-Z0-9_]* | foo |
| CONST_FLOAT | \d+\.\d+ | 3.14 |
| CONST_INT | \d+ | 123 |
| CONST_STRING | \"([^\\\n]\|(\\.))+\" | "Hello, world!" |

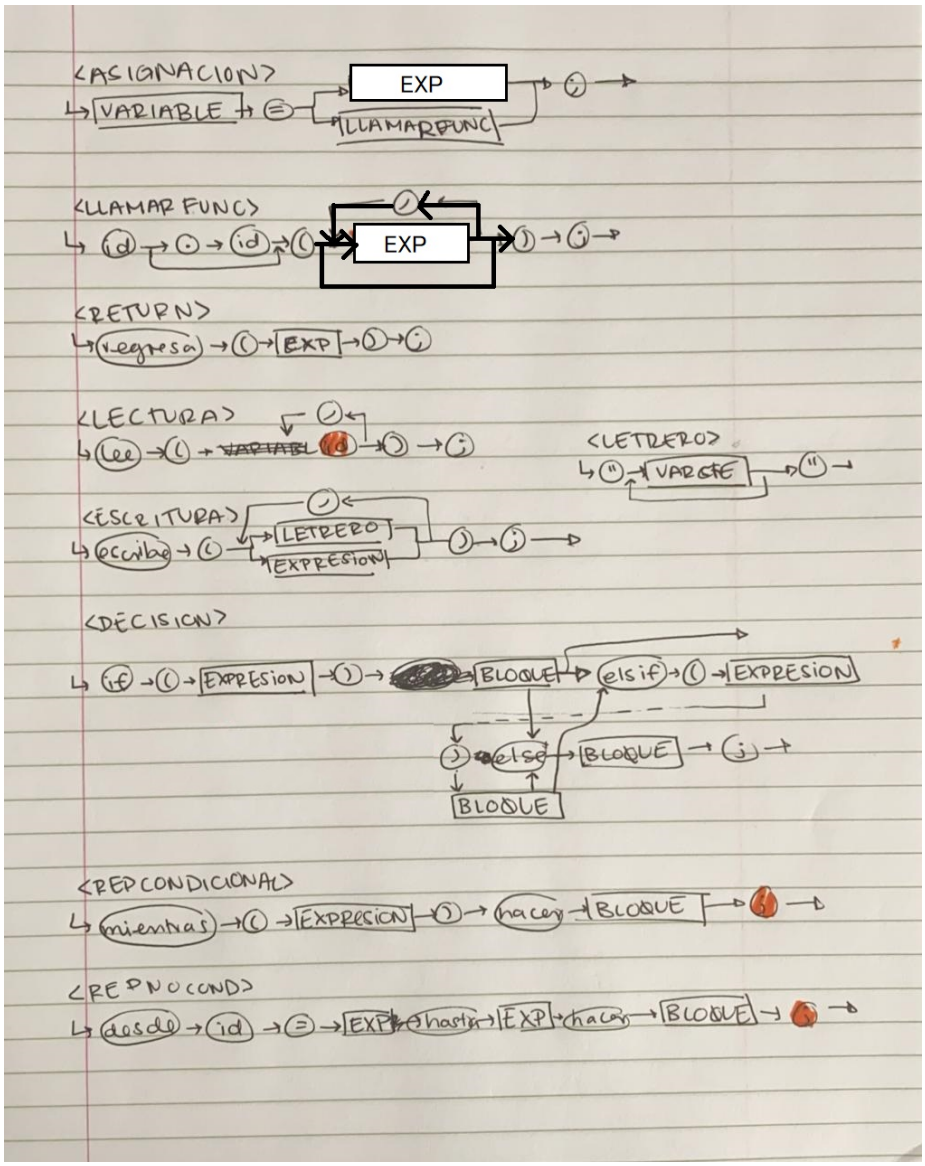
## Gramática

### Diagramas de sintaxis




<DVARS>
variables
VARS
<DFUNCS>
<TIPO>
int
float
char
<DFUNCS>
funcion
TIPO
void
id
PARAMS
DVARS
ESTATUTOS
<PARAMS>
TIPO
id
<VARS>
TIPO
id
EXP
EXP
VARIABLE


<VARIABLE>
id
id
EXP
EXP
<ESTATUTOS>
ASIGNACION
LLAMARFUNC
RETURN
LECTURA
ESCRITURA
DECISION
REP.CONDICIONAL
REP.NOCONDICIONAL
<EXPRESION>
EXP
EXP
<EXP>
TERMINO
<TERMINO>
FACTOR
<FACTOR>
EXPRESION
VAR_CTE




### Gramática formal

```
program      →  PROGRAM_KEYWORD ID SEMICOLON classes vars funcs main

classes      →  CLASS_KEYWORD ID inheritance LEFT_CURLY attributes methods
                RIGHT_CURLY SEMICOLON classes
             |  empty
```

```
inheritance  →  INHERITS ID
             |  empty

attributes   →  ATTRIBUTES_KEYWORD vars_1
             |  empty

methods      →  METHODS_KEYWORD funcs
             |  empty

vars         →  VARS_KEYWORD vars_1 END_VARS
             |  empty

vars_1       →  var_type COLON vars_2 vars_arr SEMICOLON vars_1
             |  var_type COLON vars_2 vars_arr SEMICOLON

vars_arr     →  LEFT_BRACKET vars_arr_1 RIGHT_BRACKET
             |  empty

vars_arr_1   →  vars_arr_2 COMMA vars_arr_2
             |  vars_arr_2

vars_arr_2   →  CONST_INT
             |  exp

type         →  INT
             |  FLOAT
             |  CHAR

var_type     →  type
             |  ID

vars_2       →  ID COMMA vars_2
             |  ID

funcs        →  FUNCTION func_type ID
                LEFT_PARENTHESIS parameters RIGHT_PARENTHESIS
                LEFT_CURLY vars block_1 RIGHT_CURLY SEMICOLON funcs_1

funcs_1      →  funcs
             |  empty

func_type    →  type
             |  VOID

parameters   →  parameters_1
             |  empty

parameters_1 →  var_type COLON ID parameters_2

parameters_2 →  COMMA parameters_1
             |  empty

main         →  MAIN_KEYWORD LEFT_PARENTHESIS RIGHT_PARENTHESIS block
                SEMICOLON
```

```
block        →  LEFT_CURLY block_1 RIGHT_CURLY

block_1      →  statement block_1
             |  empty

statement    →  statement_1 SEMICOLON

statement_1  →  assignment
             |  func_call
             |  return
             |  read
             |  write
             |  decision
             |  cond_loop
             |  non_cond_loop
             |  empty

assignment   →  variable EQUALS_ASSIGNMENT exp
             |  variable EQUALS_ASSIGNMENT func_call

variable     →  ID LEFT_BRACKET exp COMMA exp RIGHT_BRACKET
             |  ID PERIOD ID
             |  ID

expression   →  exp relational_op exp
             |  exp

relational_op → NOT_EQUALS
              | EQUALS_COMPARISON
              | LESS_THAN
              | GREATER_THAN
              | AND
              | OR

exp          →  term PLUS exp
             |  term MINUS exp
             |  term

term         →  factor MULTIPLY factor
             |  factor DIVIDE factor
             |  factor

factor       →  LEFT_PARENTHESIS expression RIGHT_PARENTHESIS
             | constant
             | variable
             | func_call
             | PLUS constant
             | MINUS constant

constant    →   CONST_INT
            |   CONST_FLOAT
```

```
func_call   →   ID PERIOD ID LEFT_PARENTHESIS func_call_1
                RIGHT_PARENTHESIS
            |   ID LEFT_PARENTHESIS func_call_1 RIGHT_PARENTHESIS

func_call_1  →  func_call_2
             |  empty

func_call_2  →  exp func_call_3

func_call_3  →  COMMA func_call_2
             |  empty

return       →  RETURN LEFT_PARENTHESIS exp RIGHT_PARENTHESIS

read         →  READ LEFT_PARENTHESIS read_1 RIGHT_PARENTHESIS

read_1       →  variable read_2

read_2       →  COMMA variable read_2
             |  empty

write        →  WRITE LEFT_PARENTHESIS write_1 RIGHT_PARENTHESIS

write_1      →  expression write_2
             |  CONST_STRING write_2

write_2      →  COMMA write_1
             |  empty

decision     →  IF LEFT_PARENTHESIS expression RIGHT_PARENTHESIS
                block elsif else

elsif        →  ELSIF LEFT_PARENTHESIS expression RIGHT_PARENTHESIS
                block elsif
             |  empty

else         →  ELSE block
             |  empty

cond_loop    →  WHILE LEFT_PARENTHESIS expression RIGHT_PARENTHESIS DO block

non_cond_loop → FROM ID EQUALS_ASSIGNMENT exp TO exp DO block
```