**BabyDuck – Entrega #0**

1.- Diseñar las Expresiones Regulares que representan a los diferentes elementos de léxico que ahí aparecen.

|  |  |
| --- | --- |
| **Token** | **Expresión Regular** |
| id | [a-z\_][a-zA-Z0-9\_]\* |
| cte\_int | [0-9]+ |
| cte\_float | [0-9]+\.[0-9]+ |
| cte\_string | “(.\*)?” |
| add | \+ |
| sub | - |
| mul | \\* |
| div | / |
| assign | = |
| equal | == |
| notequal | != |
| less | < |
| greater | > |
| semicolon | ; |
| colon | : |
| comma | , |
| lparen | ( |
| rparen | ) |
| lbrace | { |
| rbrace | } |
| lbrack | [ |
| rbrack | ] |
| program | program |
| main | main |
| end | end |
| var | var |
| print | print |
| if | if |
| else | else |
| while | while |
| do | do |

2.- Listar todos los Tokens que serán reconocidos por el lenguaje

**Palabras Clave**

* program
* main
* end
* if
* else
* while
* do
* print
* void
* type (int o float)

**Variables**

* id
* cte\_int
* cte\_float
* cte\_string

**Operadores aritméticos**

* +
* -
* \*
* /

**Operadores de comparación**

* =
* ==
* !=
* <
* >

**Delimitadores**

* (
* )
* {
* }
* [
* ]
* ;
* ,
* :

3.- Diseñar las reglas gramaticales (Context Free Grammar) equivalentes a los diagramas.

|  |  |
| --- | --- |
| **Regla** | **Definición** |
| Programa | program ID ; VARS FUNCS main Body end |
| VARS | var : Tipo ID ( , ID )\* ; VARS | ε |
| Tipo | int | float |
| FUNCS | void ID ( [ ] ) ( Tipo ) ID : VARS Body FUNCS | ε |
| Body | { STATEMENT } |
| STATEMENT | ASSIGN | CONDITION | CYCLE | Print |
| ASSIGN | ID = EXPRESIÓN ; |
| Print | print ( EXPRESIÓN ) ; |
| CONDITION | if ( EXPRESIÓN ) Body else Body |
| CYCLE | while ( EXPRESIÓN ) do Body |
| EXPRESIÓN | EXP ( < | > | == | != EXP )? |
| EXP | TÉRMINO ( (+ | -) TÉRMINO )\* |
| TÉRMINO | FACTOR ( (\* | /) FACTOR )\* |
| FACTOR | (+ | -)? ( ID | CTE | ( EXPRESIÓN ) ) |
| CTE | CTE\_INT | CTE\_FLOAT | CTE\_STRING |