Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| Identificadores y variables | | | |
| Declaracion | Declaración y asignacion | | Uso |
| Num identifier; | Num identifier=23;  Num identifier=20.5; | | identifier++;  identifier--;  identifier\*\*;  identifier+=2;  identifier\*=2;  identifier/=2;  identifier-=2;  identifier=23;  identifier=identifier+2; |
| Str identifier; | Str identifier=”any string”; | | Identifier+=”other”;  Identifier=identifier+”other”;  Identifier=”other string”; |
| Default identifier; | Default identifier = anything; | | Todas las sobrecargas y asignaciones (infere el tipo, tomándolo como un string) |
| Any dentifier; | Any identifier=[“element1”, “element2”,…, “elementN”];  Any identifier=[1,2,3,…,N] | | Identifier[1]=”replace element”;  Identfier[1]=20;  identifier.IndexFirst(element); <-obtener el índice de la primer incidencia |
| Long identifier; | Long identifier = 23456789876543456; | | Todas las sobrecargas y asignaciones |
|  | | | |
|  | |  | |
| [Loop #name]  ---  Identifier[name.Index];  ---  [Loop :Index=N,Iter=M #name]  ---  Identifier[name.Index];  --- | | Contendrá una palabra reservada Index (representara el indice de iteración) e Iter (el valor con el que operara decremento o decremento), por defecto tendrán Index=0, Iter=1  Indicar el valor del Index y del Iter cambiara su función por defecto.  --- marcan el inicio y fin del contexto del ciclo | |
| [If:(condicion) #name]  ---  identifier = 20;  ---  [If:(condicion) #name]  ---  identifier = 20;  ---[:]  ---  identifier = 12;  --- | |  | |
| FN name(type paremeter1,…, type paremeterN)  ---  //code  ---  FN name(type paremeter1,…, type paremeterN, Out outIdentifier)  ---  //code    ---  FN name(type paremeter1,…, type paremeterN)  ---  //code  Go;  ---  //  FN name(type paremeter1,…, type paremeterN, Out outIdentifier)  ---  //code  [If:(condicion) #name]  ---  Go;  ---[:]  //more code  --- | | el tipo Out define el nombre de la variable que se usara para retornar un valor, de este modo si en la función se indica este parámetro el ultimo valor que tenga es el que tomara y retornara, tendra Default por defecto, a menos que se redefina dentro de la función con otro tipo (para interrumpir el flujo de la función inmediato se puede usar Go) | |
| [Write(“message”)];  [Write : Url=docUrl]; | | Salida de datos  salida de datos de un archivo | |
| Identifier=[Read];  [Read(“entrada”):Url=docUrl];  [Read(“entrada”)]; | | Entrada de datos  Entrada de datos a un archivo | |
| Go;  Go #name; | | Interrumpe el flujo de una estructura de control  Interrumpe el flujo de una estructura de control y/o desde cierto punto de código redirige a la estructura que referencie el dentificador | |
| [Entity #name]  ---  Str identifier;  --- | | Permite definir variables y funciones propias de su contexto | |

Palabras reservadas={Any, Default, FN, Go, If, Index, IndexFirst, Iter, Long, Loop, Num, Read, Str, Url, Write, Entity}

Simbolos= {'b', 'B', 'c', 'C', 'g', 'h', 'H', 'j', 'J', 'k', 'K', 'M', 'O', 'q', 'Q', 'T', 'v', 'V', 'w', 'X', 'Y', 'z', 'Z',’\_’}

Digitos={0,1,2,3,4,5,6,7,8,9}

Delimitadores={---, ; , “ “, “,”, [, ],(,),#,”}

Operadores aritmeticos = {‘+’,’-‘,’\*’,’/’,’%’,’=’}

Operadores relacionales = {‘==’,’!=’, ‘<’,’<=’,’>’,’>=’}

Operadores lógicos = {&&,||,!}

Expresiones regulares:

Identificador -> símbolo (símbolos+Digitos)

Estructura del AST

Tipos de Datos: Num, Str, Any, Default, Loop

Operaciones Binarias: +, -, \*, /, etc.

Variables: Declaración y uso de variables.

Literales: Números, cadenas, caracteres, booleanos.

Declaraciones: Asignaciones, declaraciones de variables, declaraciones de funciones.

Estructuras de Control: if, loop, etc.

Bloques de Código: Secuencias de declaraciones y expresiones.

Gramáticas:

Statements -> Expression SEMICOLON Statements | Function Statements | If Statements| Declaration SEMICOLON Statements| Out\_Put Statements | In\_Put Statements | Entity | EPSILON

------------------------Declarations-----------------------------------

Entity -> OPEN\_CORCHETES ENTITY SHARP IDENTIFIER COLSE\_CORCHETES CONTEXT\_TOKEN Statements CONTEXT\_TOKEN

//

Function -> FN IDENTIFIER OPEN\_PARENTHESIS Parameter\_List Function\_Out CLOSE\_PARENTHESIS CONTEXT\_TOKEN Statements CONTEXT\_TOKEN

Function\_Out -> OUT IDENTIFIER | EPSILON

Parameter\_List -> Type IDENTIFIER Parameter\_List\_Prime | EPSILON

Parameter\_List\_Prime -> COMMA Type IDENTIFIER Parameter\_List\_Prime | EPSILON

Type -> NUM | STR | DEFAULT | LONG | ANY

Arithmetic\_Operator -> EQUAL | DOUBLE\_ASTERISK | DOUBLE\_MINUS | DOUBLE\_PLUS | EQUAL\_PLUS | EQUAL\_MINUS | EQUAL\_SLASH | EQUAL\_ASTERISK | ASTERISK | SLASH

//

Declaration -> Type IDENTIFIER Assign\_Value | IDENTIFIER Assign\_Value | IDENTIFIER Function\_Call | Atribute\_Access | Flow\_Control

Flow\_Control -> GO | GO SHARP IDENTIFIER

Atribute\_Access -> IDENTIFIER DOT Atribute\_Access\_Prime

Atribute\_Access\_Prime -> IDENTIFIER Atribute\_Acces\_Prime | IDENTIFIER DOT Atribute\_Acces\_Prime | ESPSILON

Function\_Call -> OPEN\_PARENTHESIS Parameter\_Group CLOSE\_PARENTHESIS

Parameter\_Group -> EPSILON | Expression Parameter\_Group | Expression COMMA Parameter\_Group

Assign\_Value -> EPSILON | Arithmetic\_Operator Expression Assign\_Value | IDENTIFIER Assign\_Value | COMMA IDENTIFIER Assign\_Value | OPEN\_CORCHETES Assign\_Value\_Prime CLOSE\_CORCHETES Assign\_Value | EQUAL OPEN\_CORCHETES READ CLOSE\_CORCHETES

Assign\_Value\_Prime -> EPSILON | Val Assign\_Value\_Prime |COMMA Val Assign\_Value\_Prime

Out\_Put -> OPEN\_CORCHETES WRITE OPEN\_PARENTHESIS Variable CLOSE\_PARENTHESIS CLOSE\_CORCHETES SEMICOLON | OPEN\_CORCHETES WRITE DOUBLE\_DOT URL EQUAL STRING CLOSE\_CORCHETES SEMICOLON | OPEN\_CORCHETES WRITE DOUBLE\_DOT URL EQUAL IDENTIFIER CLOSE\_CORCHETES SEMICOLON

In\_Put -> OPEN\_CORCHETES READ OPEN\_PARENTHESIS Variable CLOSE\_PARENTHESIS DOUBLE\_DOT URL EQUAL IDENTIFIER CLOSE\_CORCHETES SEMICOLON | OPEN\_CORCHETES READ OPEN\_PARENTHESIS Variable CLOSE\_PARENTHESIS DOUBLE\_DOT URL EQUAL STRING CLOSE\_CORCHETES SEMICOLON

-------------------------------------------------------------------

----------------expressions-------------------------

Expression -> Term Expression\_Prime

Term -> Factor Term\_Prime

Term\_Prime -> ASTERISC Factor Term\_Prime | SLASH Factor Term\_Prime | EPSILON

Factor -> Variable | OPEN\_PARENTHESIS Expression CLOSE\_PARENTHESIS

Expression\_Prime -> PLUS Term Expression\_Prime | MINUS Term Expression\_Prime | EPSILON

Variable -> IDENTIFIER | NUMBER | DECIMAL\_NUMBER | (Expression) | STRING | Atribute\_Access

--------------- Control Structures -----------------------

If -> OPEN\_CORCHETES IF DOUBLE\_DOT OPEN\_PARENTHESIS Condition CLOSE\_PARENTHESIS Name\_Optional CLOSE\_CORCHETES CONTEXT\_TOKEN Statements CONTEXT\_TOKEN Else

Else -> OPEN\_CORCHETES DOUBLE\_DOT CLOSE\_CORCHETES CONTEXT\_TOKEN Statements CONTEXT\_TOKEN | EPSILON

Condition -> Val Condition\_Prime

Condition\_Prime -> Logic\_Operator Val Condition\_Prime | EPSILON

Logic\_operatior = OR || AND || LESS\_THAN || MORE\_THAN || DOUBLE\_EQUAL

Val -> NUMBER | DECIMAL\_NUMBER | IDENTIFIER | STRING

Name\_Optional -> SHARP IDENTIFIER | ESPSILON

//Loop

Cycle -> OPEN\_CORCHETES LOOP Optional\_Properties\_Loop Name\_Optional CLOSE\_CORCHETES CONTEXT\_TOKEN Satetments CONTEXT\_TOKEN

Optional\_Properties\_Loop -> DOUBLE\_DOT INDEX Assign\_Property\_Value COMMA ITER Assign\_Property\_Value | EPSILON

Assign\_Property\_Value -> EQUAL Val