

TC3048 Diseño de Compiladores

**Omedetou**

30 de marzo del 2022

|  |  |
| --- | --- |
| Sergio Guasso Idalias | Alejandro Cedillo Gamez |

# Visión

# Con nuestro nuevo lenguaje queremos poner a prueba los conocimientos adquiridos durante nuestra formación académica universitaria, implementando los estándares definidos para el desarrollo de compiladores, y añadiendo funcionalidades que creamos necesarias para complementar las librerías preestablecidas y aumentar la gama de problemas que es capaz de resolver.

# Objetivo principal

# Crear un lenguaje que cumpla con todas las reglas gramaticales necesarias para su funcionamiento básico, el cuál será orientado a objetos. Se tendrá que documentar toda la gramática, las estructuras utilizadas y los diagramas que describen la lógica del programa.

# Requerimientos de lenguaje

# Que soporte la creación de objetos, así como acceso a sus funciones y atributos.

## Elementos básicos (Tokens)

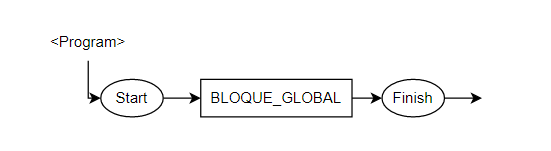
| Nombre | Signo |
| --- | --- |
| START | start |
| FINISH | finish |
| VAR | var |
| WHILE | while |
| IF | if |
| ELSE | else |
| DEF | def |
| RETURN | return |
| CLASS | class |
| EXTENDS | extends |
| PRINT | print |
| PRIVATE | Private |
| PUBLIC | Public |
| PROTECTED | Protected |
| LEFT\_CURRLY\_BRACES | { |
| RIGHT\_CURRLY\_BRACES | } |
| LEFT\_PARENTHESIS | ( |
| RIGHT\_PARENTHESIS | ) |
| COMMA | , |
| COLON | : |
| NEW\_THAN | \n |
| GREATER\_THAN | > |
| LESS\_THAN | < |
| LESS\_EQUAL | <= |
| GREATER\_EQUAL | >= |
| NOT\_EQUAL | != |
| EQUAL\_COMPARATION | == |
| EQUAL | = |
| PLUS | + |
| MINUS | - |
| ASTERISK | \* |
| SLASH | / |

## Diagramas de sintaxis

Ver siguiente link:

<https://drive.google.com/file/d/1mg31Oh5NAC9qVDS2xzRFxlYAqREs92jk/view>

### <Program>



program -> Start BLOQUE\_GLOBAL Finish

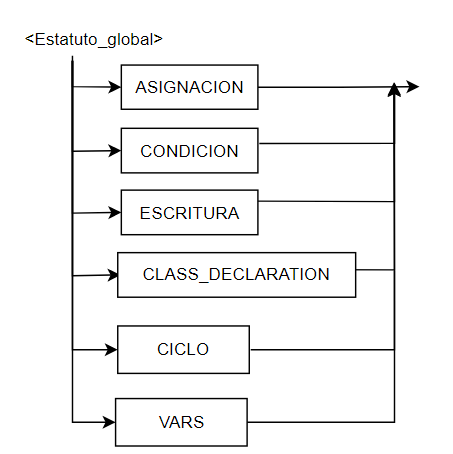
### <Bloque global>

bloque global -> { BG1 }

BG1-> ε | ESTATUTO\_GLOBAL BG1

### 

### <Estatuto global>

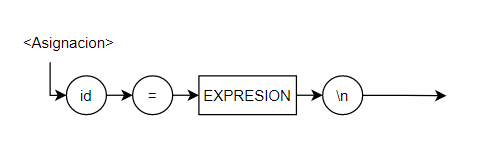


### 

Estatuto Global-> ASIGNACION | CONDICION | ESCRITURA | CLASS\_DECLARATION | CICLO | VARS

### 

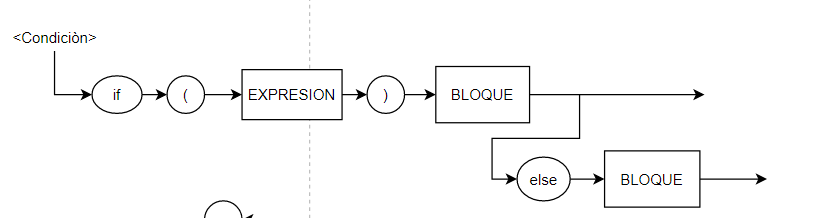
### <Asignacion>



### 

Asignacion -> id = EXPRESION \n

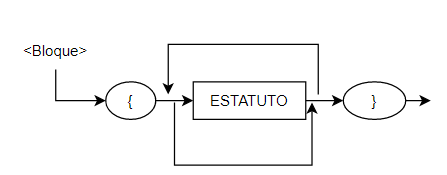
### <Condicion>



Condicion -> if ( EXPRESION ) BLOQUE C1

C1 -> else BLOQUE |

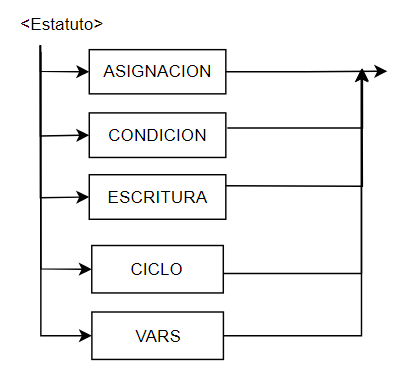
### <Bloque>



bloque global -> { B1 }

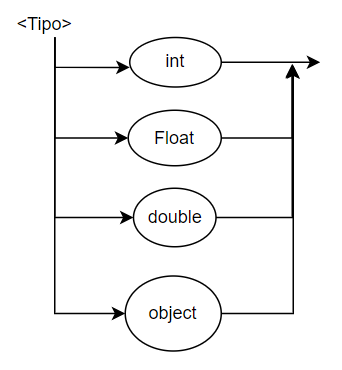
B1-> ε | ESTATUTO B1

### <Estatuto>



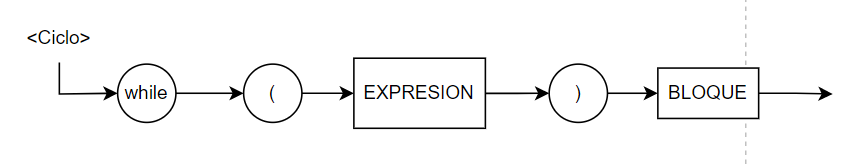
Estatuto-> ASIGNACION | CONDICION | ESCRITURA | CICLO | VARS

### <Tipo>



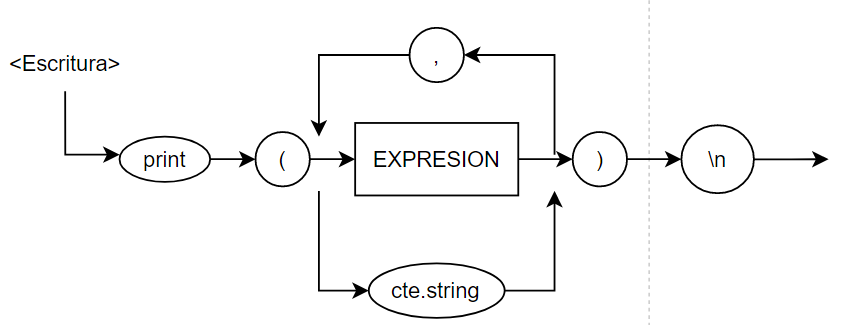
Tipo-> int | Float | double | object

### <Ciclo>



Ciclo-> while ( EXPRESION ) BLOQUE

### <Escritura>



Escritura -> print ( ESC1 ESC2 ) \n

ESC1 -> EXPRESION | cte.string

ESC2 -> , ESC1 | ε

### <Vars>

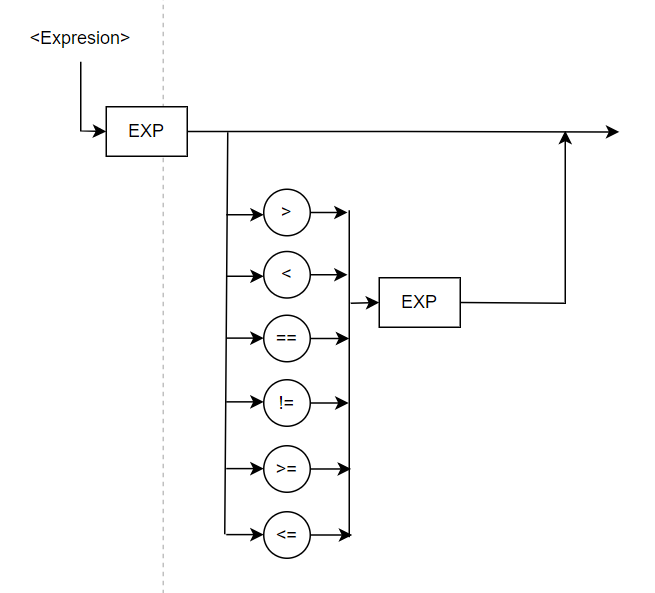
## 

VAR-> var VAR2

VAR2 -> id VAR3

VAR3-> , VAR2 | : TIPO \n

### <Expresion>

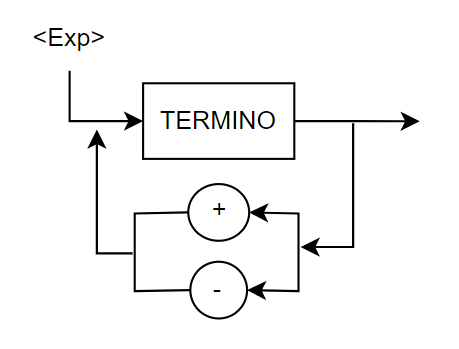


Expresion-> EXP E1

E1 -> ε | E2 EXP

E2 -> > | < | == | != | >= | <=

### <Exp>

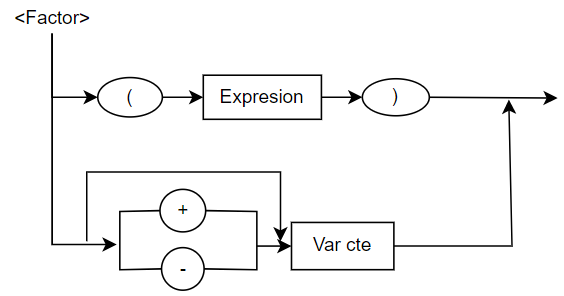


Exp-> Termino EX1

EX1-> ε | EX2 Exp

EX2-> + | -

### <Factor>



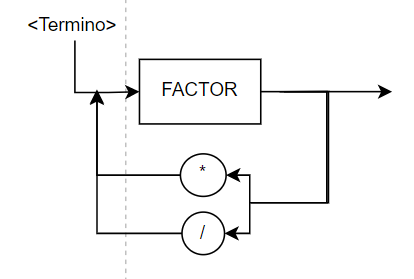
Factor-> F1 | F2

F1-> ( EXPRESION )

F2-> F3 Varcte

F3-> ε | + | -

### <Termino>

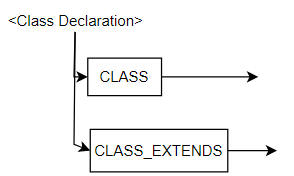


Termino-> FACTOR T1

T1 -> ε | T2 Termino

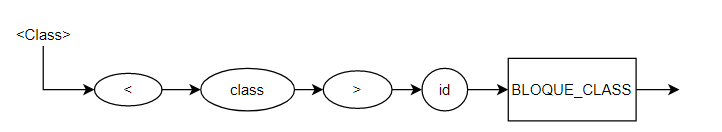
T2-> \* | /

### <Class Declaration>



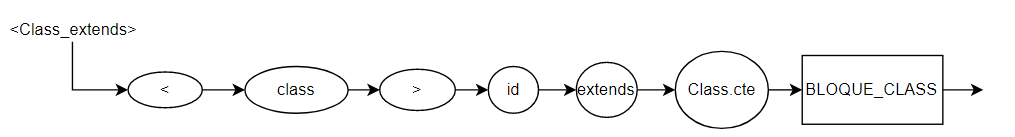
Class Declaration -> CLASS | CLASS\_EXTENDS

### <Class>



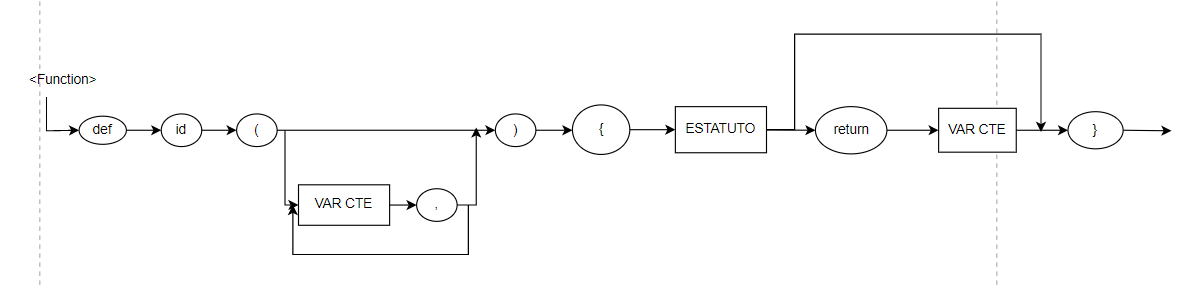
Class-> < class > id BLOQUE\_CLASS

### <Class Extends>



class extends -> < class > id extends

### <Function>

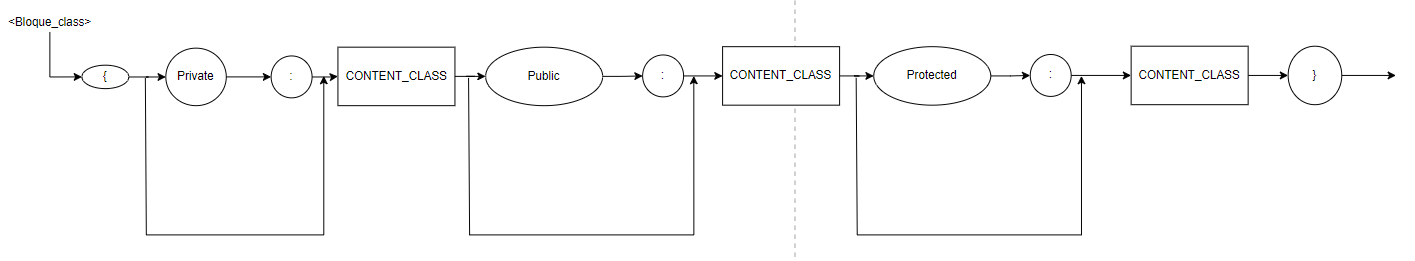


Function-> def id ( F1 ) { ESTATUTO F2 }

F1-> ε | VARCTE , F1

F2-> ε | return VARCTE

### <Bloque Class>



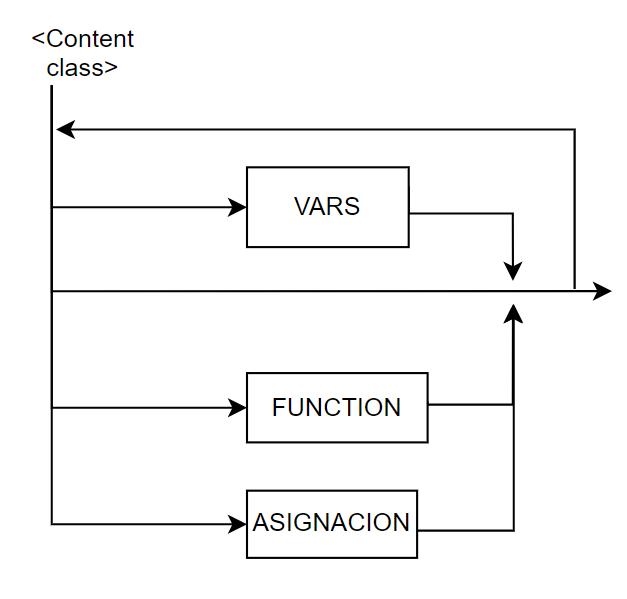
Bloque Class -> { BC1 CONTENT\_CLASS BC2 CONTENT\_CLASS BC3 CONTENT\_CLASS }

BC1 -> Private : | ε

BC2 -> Public : | ε

BC3 -> Protected : | ε

### <Content Class>



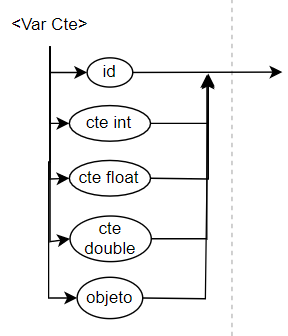
Content Class-> CC1 | ε

CC1 -> CC2 CONTENT\_CLASS

CC2 -> VARS | FUNCTION | ASIGNACION

### 

### <Var Cte>



VarCte -> id | cte int | cte float | cte double | objeto

## Características principales de semántica

* Soporte de programación orientada a objetos
* Seguir PEMDAS

## Descripciones específicas

Ejemplo de un código en lenguaje Omedetou:

Start

{

<class> Person

{

Public:

var num : int

var spreak : String

num = 4

speak = "hola mundo"

}

var aux : int

aux = 4

while (aux > 0) {

print("hola mundo" )

print("\n")

aux = aux - 1

}

}

Finish

## Tipos de datos

* Int
* Float
* String
* Object

# Language and OS

Language : Python

OS : Windows

# Bibliografía