Propuesta de Proyecto

Lenguaje ALOOP

Diseño de compiladores (Gpo. 2)

Adriana Fernández López A01197148
Jeasika Alejandra López López A00821873
4 de abril de 2022

1. Propósito del proyecto

En este documento se describen las características generales de un lenguaje de programación orientado a objetos.

2. Objetivo del lenguaje

3. Requerimientos el lenguaje

3.1. Tokens

TOKEN	DESCRIPCIÓN INFORMAL	LEXEMAS DE EJEMPLO
program	caracteres **p, r, o, g, r, a, m**	program
main	caracteres **m, a, i, n**	main()
end	caracteres **e, n, d**	end;
type	caracteres **t, y, p, e.** Declaración de clase.	type
def	caracteres **d, e, f.** Declaración de variable.	def
func	caracteres **f, u, n, c.** Declaración de función.	func
ret	caracteres **r, e, t**	ret();
call	caracteres **c, a, l, l.** Llamar a una función.	call
if	caracteres **i, f**	if
then	caracteres **t, h, e, n**	then
else	caracteres **e, l, s, e**	else
while	caracteres **w, h, i, l, e**	while
do	caracteres **d, o**	do
for	caracteres **f, o, r**	for
to	caracteres **t, o**	to
id	letra seguida por letras y dígitos	pi, M2
number	caracteres **n, u, m, b, e, r**	number
num	un dígito, opcionalmente seguido por más dígitos, un punto y más dígitos	12, 15.26
string	caracteres **s, t, r, i, n, g**	string
str	Comillas dobles seguidas por cualquier	“hola”

	número y tipo de caracteres y terminando con comillas dobles	
to_number	caracteres **t, o, _, n, u, m, b, e, r**	to_number(“1”)
to_string	caracteres **t, o, _, s, t, r, i, n, g**	to_string(123)
input	caracteres **i, n, p, u, t**	input()
print	caracteres **p, r, i, n, t**	print()

Expresiones regulares

`id → [a-zA-Z][a-zA-Z_0-9]*`

`num → [0-9]+(\.[0-9]+)?`

`str → ".*"`

3.2. Diagramas de sintaxis

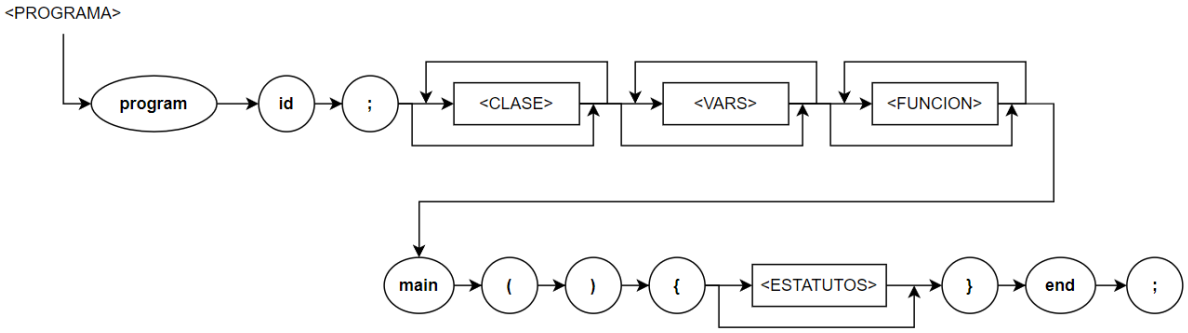
La estructura general del lenguaje es:

```
program <id>;
<Declaración de clases>
<Declaración de variables globales>
<Declaración de funciones>

# comentario
main() {
      <estatutos>
}
end;
```

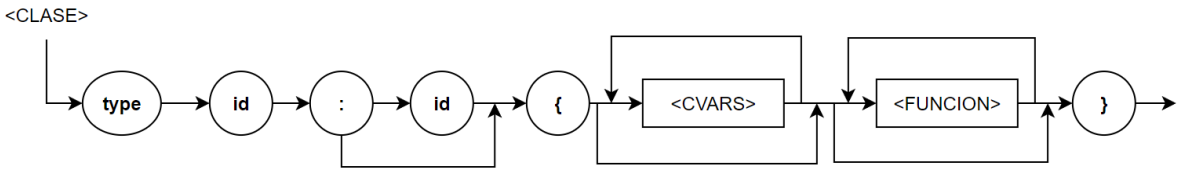
* Las secciones en itálicas son opcionales.

* Las palabras y símbolos en bold son reservadas y el # indica comentario.



Declaración de clases

```
type <id> : <id> {
        <declaración de variables>
        <declaración de funciones>
}
```



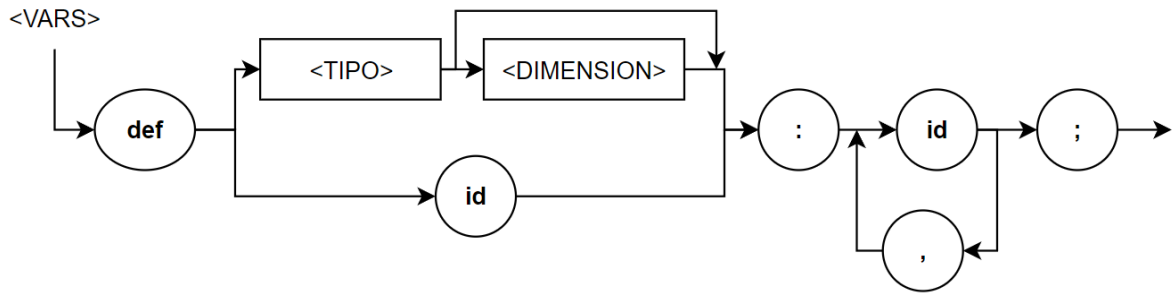
Para acceder o llamar un método de un objeto, se utilizará la siguiente sintaxis:

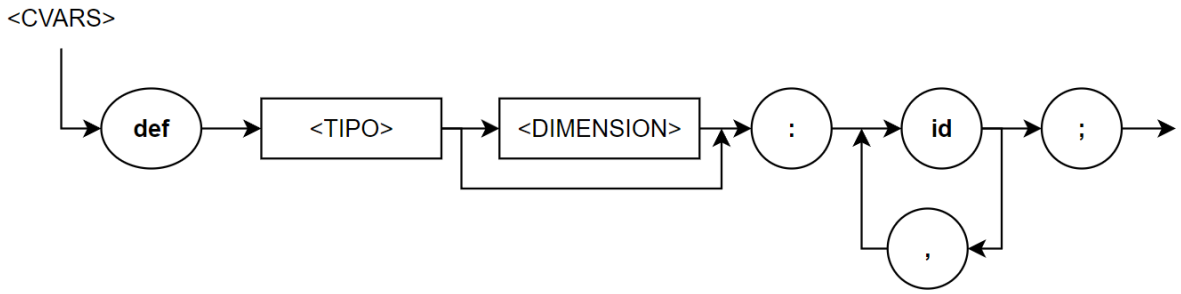
`objeto:attributo` o bien `objeto:metodo(<parametros>)`

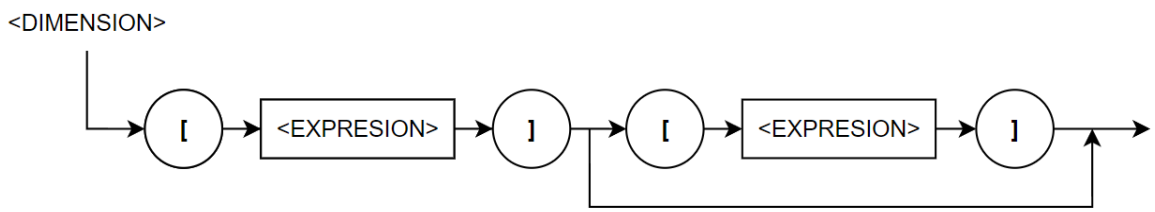
Declaración de variables

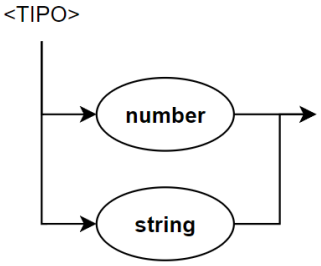
```
def <tipo><dimension> : <lista_ident>;
```

* Dentro de una clase solo se podrán definir variables de tipos simples (number o string), mientras que para las variables globales también se podrán definir objetos



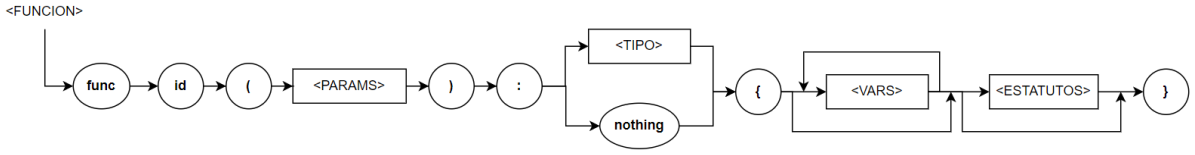


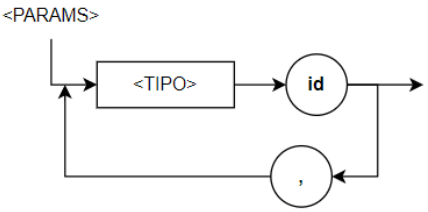




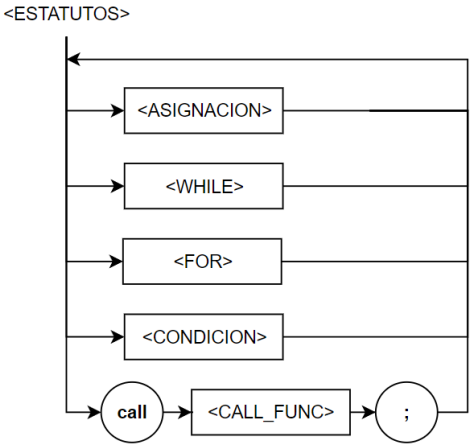
Declaración de funciones

```
func <id> ( <params> ) : <tipo_func> {
        <Declaración de variables>
        <estatutos>
}
```



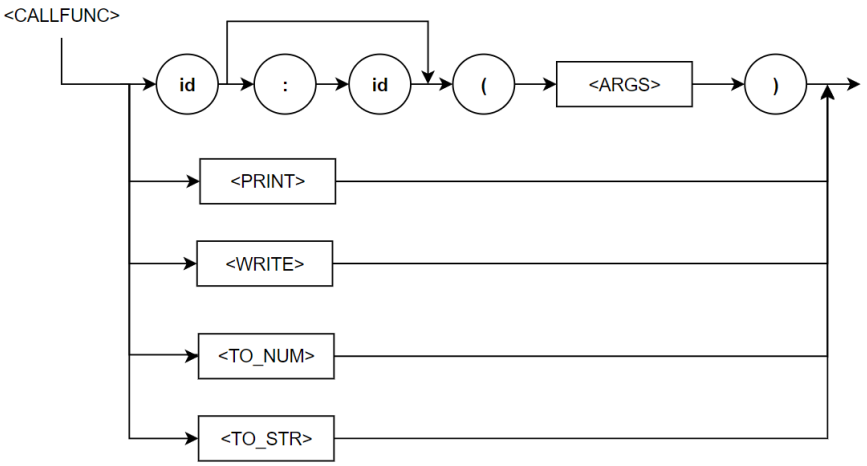


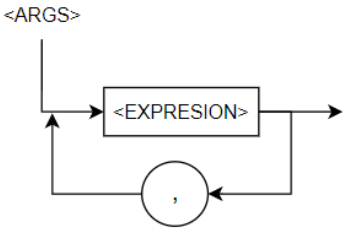
Estatutos



Llamar a una función

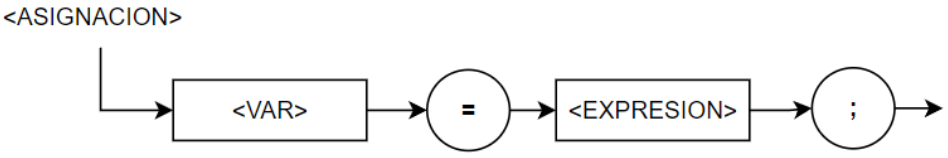
```
call <id>:<id>( <args> );
```

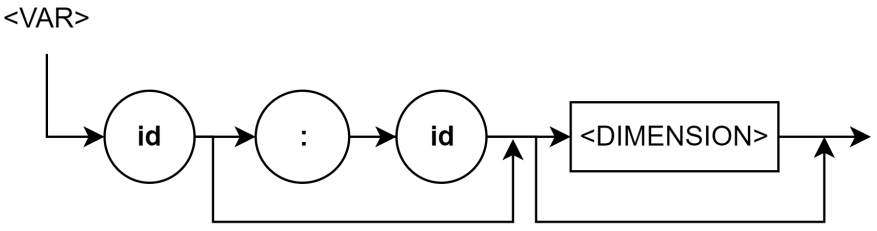


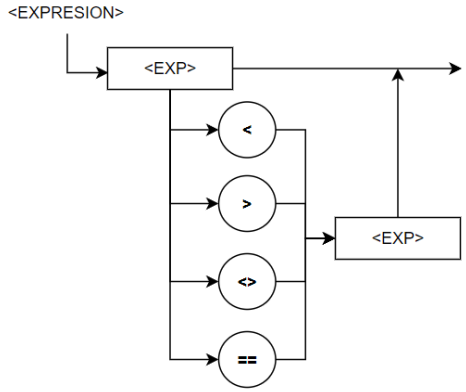


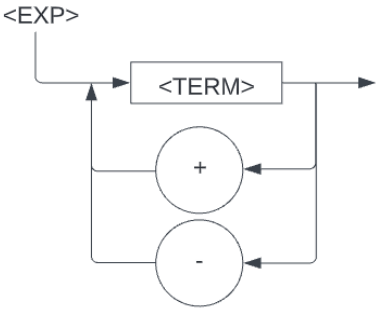
Asignación

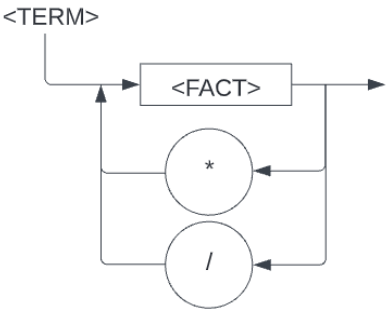
```
<id>:<id><dimension> = <expresion> ;
```

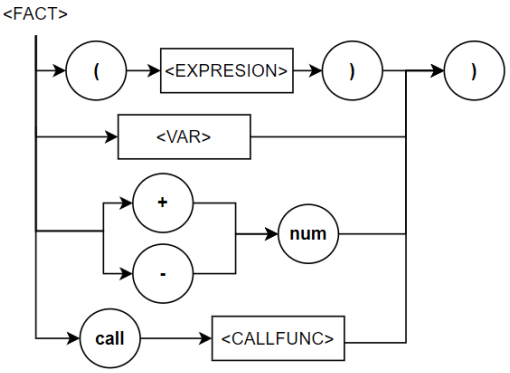






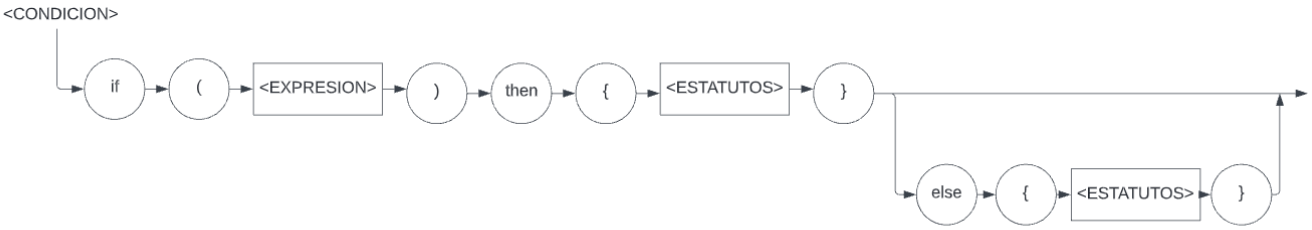






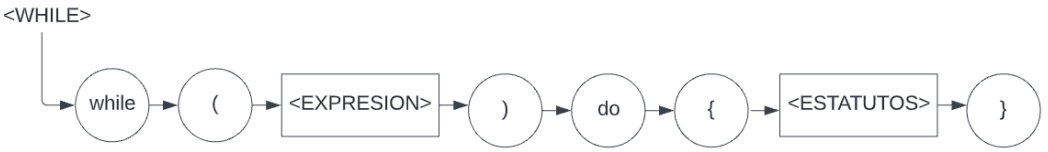
Condicional

```
if ( <expresión> ) then {
      <estatutos>
} else {
      <estatutos>
}
```



Bucles

```
while ( <expresion> ) do {
      <estatutos>
}
```




```
for <expresion> to <expresion> {
      <estatutos>
}
```

3.3. Características principales de la semántica

Type matching

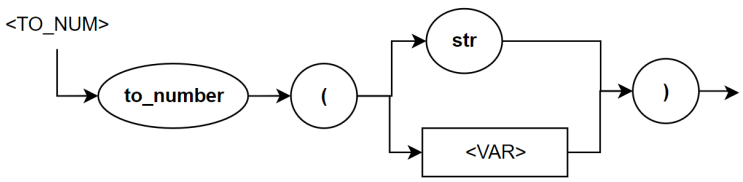
* 0 se manejará como falso y cualquier otro valor como verdadero

left	**right**	+	-	*	/	>	<	<>	==	&
num	num	num	num	num	num	num*	num*	num*	num*	error
num	str	error	error	error	error	error	error	error	error	error
str	str	error	error	error	error	error	error	num*	num*	str
str	num	error	error	error	error	error	error	error	error	error

3.4. Funciones especiales

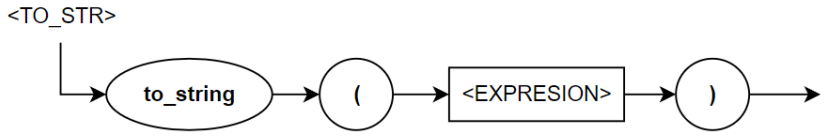
- `to_number(<constante_str>)`

  Transforma el tipo de dato de la constante que recibe a `number`.



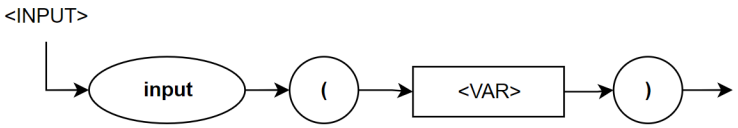
- `to_string(<constante>)`

  Transforma el tipo de dato de la constante que recibe a `string`.



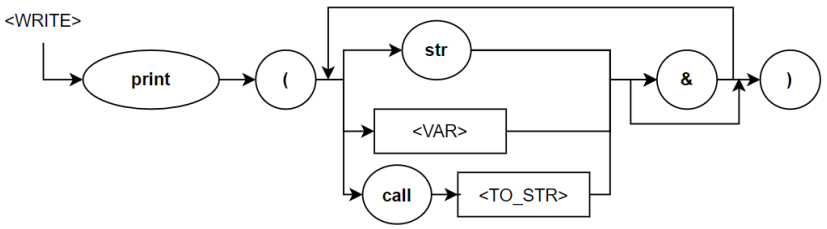
- `input()`

  Lee una cadena hasta que encuentra un espacio. Regresa el mismo valor que lee, así que se debe asignar a un id. Por ejemplo, `x = input();`



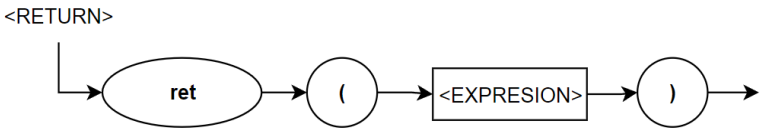
- `print(<string> & <string> & ... );`

  Imprime los string dentro de los paréntesis. Se utiliza el símbolo & para concatenar



- `ret(<expresion> );`

  Se utiliza dentro de una función para regresar un valor.



3.5. Tipos de datos

Para este lenguaje de programación se tienen los siguientes tipos:

- Number
- String
- Objetos

4. Lenguaje y sistema operativo

Lenguaje: Python Lex-Yacc (PLY)
Sistema operativo: Windows

5. Bibliografía

Compilers Principles, Techniques, and Tools 2ndEd. Alfred V. Aho, Monica Lam, Ravi Sethi, Jeffrey D. Ullman. Addison Wesley, 2007.