CONCEPTOS Y PARADIGMAS DE LENGUAJA DE PROGRAMACION

TRABAJO INTEGRADOR 2018

INTEGRANTES:

- **SALGADO IVAN, Legajo 11823/6**
- **ALONSO DIAZ MARIANO HUGO IGNACIO, Legajo 11046/7**
- **VARELA JUAN MANUEL, Legajo 11997/9**

GRUPO: 43

LENGUAJES: Processing – Python

AYUDANTE: Viviana

TURNO: Viernes (M)

BIBLIOGRAFIA:

- **https://www.python.org/**
- **https://processing.org/**
- **Teorías de la catedra**

A. Desarrolle en EBNF el bloque for (o su correspondiente en el lenguaje principal) donde se puedan utilizar expresiones numéricas en los delimitadores. Estas expresiones pueden tener paréntesis.

G = (N, T, S, P)

N = { <sentencia_for>, <variable>, <inicializada>, <var>, <letra>, <alfanumérica>, <digito>, <real>, <declarada_inicializada>, <tipo_dato>, <comparación>, <signo_comparacion>, <calculo>, <incremento>, <decremento>, <expresión>, <signo> }

T = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, .. , z, A, .. , Z, for, ;, (,), =, <, >, >=, <=, ==, !=, +, -, *, /, ++, - -}

S = { <sentencia_for> }

P = { <sentencia_for> ::= 'for' '(' <variable> ';' <comparación> ';' <calculo> ')' '{' <bloque> '}'

<variable> ::= (<declarada_inicializada> | <inicializada>)

<inicializada> ::= <var> '=' <real>

<var> ::= <letra> {<alfanumérico>}*

<letra> ::= ('a' | .. | 'z' | 'A' | .. | 'Z' |)

<alfanumérica> ::= (<letra> | <digito>)

<digito> ::= ('0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9')

<real> ::= [(+ | -)] {<digito>} + ['.'{<digito>}+]

<declarada_inicializada> ::= <tipo_dato> <inicializada>

<tipo_dato> ::= ('int' | 'float' | 'byte')

<comparación> ::= (<var> | <real>) <signo_comparacion> (<var> | <real>)

<signo_comparación> ::= ('<' | '>' | '>=' | '<=' | '==' | '!=')

<calculo> ::= (<incremento> | <decremento> | <expresión>)

<incremento> ::= <var> '++'

<decremento> ::= <var> '- -'

<expresión> ::= (<var> '=' (<var> | <real>) <signo> (<var> | <real>) | <var> <signo> '=' <real>)

<signo> ::= ('+' | '-' | '*' | '/') }

B. Enuncie y compare distintos aspectos semánticos (tanto de la semántica estática y dinámica) de la sentencia de asignación entre los lenguajes asignados.

Nuestros lenguajes asignados (Processing y Python) cuentan con diferentes formas de asignar los datos a las variables ya que una de las diferencias principales es que Processing es fuertemente tipado y Python dinámicamente tipado.

Asignación en Processing:

Int x = 10;

String p = “Hola”;

Persona persona = new Persona();

Al ser fuertemente tipado hay que declarar el tipo de dato a la variable antes de asignarle un valor.

Asignación en Python:

x = 10;

p = “hola”;

x = “un string”;

Python es un lenguaje dinámicamente tipado, esto quiere decir que no necesita indicar el tipo de variable en el momento de declararlo ya que lo identifica automáticamente.

En el ejemplo de arriba x = 10 transforma automáticamente a x en un entero y eso implica que se le asigne el rango y operaciones válidas para el tipo de dato. Luego la variable x cambia su valor a un cadena de texto, entonces de forma automática cambia el tipo de dato a string ajustando sus nuevo rango y operaciones posibles.

Python no posee semántica estática porque al ser un lenguaje interpretado ejecuta y evalúa línea a línea las instrucciones de un programa dado. Si en alguna línea estoy usando un identificador que no fue declarado el programa lanzara un error cuando llegue a ejecutar esa instrucción, por lo tanto, seria un error de semántica dinámica.

En cambio en Processing que es un lenguaje compilado y verifica todos los identificadores y tipos en tiempo de compilación asegura que no ocurran errores en ejecución como identificadores no declarados, asignaciones inválidas (por el tipo de dato)

C. Defina una porción de código donde pueda apreciarse las características más relevantes de las variables en cuanto a sus atributos. Elija alguno de los lenguajes asignados que presente mayores posibilidades para mostrar estas características y desarrolle el ejercicio de la misma forma que se realiza en la práctica. Luego, si es necesario realice las explicaciones que permitan una mayor comprensión del ejercicio.

Lenguaje elegido Processing.

Código de ejemplo:

```
1.  int x = 10;
2.  static float y = 5.5;
3.
4.  public void funcionA () {
5.      String a = "Mundo:";
6.      println("Hola " + a);
7.  }
8.
9.  public void funcionB () {
10.     int x = 50;
11.     println ("Variable local x: " + x);
12. }
13.
14. void setup () {
15.   int j;
16.   println("Variable global x: " + x);
17.   funcionA();
18.   funcionB();
19. }
20.
```

Identificador	Tipo	r-valor	Alcance	T.V.
x	Automática	10	(1 - 9) y (13 - 20)	(1 - 20)
y	Estática	5.5	(2 - 20)	(1 - 20)
a	Automática	Mundo	(5 - 7)	(4 - 7)
x	Automática	50	(10 - 12)	(9 - 12)
j	Automática	indefinido	(15 - 19)	(14 - 19)
funcionA()	-	-	(5 - 7)	(4 - 7)
funcionB()	-	-	(10 - 12)	(9 - 12)

Nombre de las variables:

- Es un identificador (una secuencia de letras y/o dígitos).
- Representa a la variable en el programa fuente.
- Permite al programador aportar información sobre el significado del valor almacenado.

Tipo de las variables:

- Determina el conjunto de valores que puede tomar la variable, y las operaciones que pueden hacerse sobre ella.

Tiempo de vida:

- La ejecución de un programa ocupa un intervalo de tiempo:
- El tiempo de vida de una variable es el intervalo o intervalos de tiempo durante los cuales la variable existe.
- Los intervalos están normalmente incluidos en el intervalo de tiempo de ejecución del programa.
- Una variable existe durante la ejecución cuando se puede acceder a su valor, en el sentido de que se puede garantizar que dicho valor está almacenado en su localización usando la representación asociada a su tipo.

Alcance:

- Es la porción o porciones del programa en los cuales la aparición del nombre de la variable es correcta y hace referencia a dicha variable, a sus atributos o a su valor.
- Define en qué secciones de código una variable estará disponible. Fuera de este ámbito, una variable no podrá ser accedida (no existe).

R-valor:

- Valor almacenado en el l-valor de la variable.
- Se interpreta de acuerdo al tipo de variable.