

Conceptos y Paradigmas de Lenguajes de Programación - 2025 - Primer Parcial 1ra Fecha. T2

25/04/2025

Realice el parcial con lapicera de otra forma se desaprobará el/los ejercicio/s.

Se considera presentismo cuando se realiza completamente un ejercicio.

Legajo: 14350/7					Corrigió:				
Apellido y Nombres: Carballo Pedro					SEBASTIAN				
1	a	B	b	B					
2	a	B	b	B	c	B	d	B	
3	a	B	b	B					
4	a	B	b	B	c	B			
Resultado Final									A

Ejercicio 1

a) (Pts.25) Defina, utilizando EBNF, la gramática para la sentencia FOR de lenguaje C. Puede ser un ejemplo como:

```
for (expresionInicio;; condicionFin; ExpresionLoop)
{
    instrucciones
}
```

b) (Pts. 5) Realice el diagrama sintáctico de la gramática correspondiente.

Ejercicio 2

Sea el siguiente código en Ada, indique para todos los identificadores

a) (Pts. 5) Su tipo de ligadura con l-valor. b) (Pts. 5) Su r-valor al momento de declaración (inicialización).

c) (Pts. 5) Tiempo de vida . d) (Pts. 5) Alcance

→ Se toma desde la línea siguiente a la declaración (como en la práctica)

1. with text_io; use text_io;
2. Procedure Main is;
3. type arreglo_int is array(integer range <>);
4. c, o, q: integer;
5. vec: arreglo(1..100);
6. t: constant integer:=5;
7. Procedure Proc is;
 - 7.1. type pointer is access integer;
 - 7.2. vec2: arreglo(0..t);
 - 7.3. q: pointer;
 - 7.4. begin
 - 7.4.1. o:=4;
 - 7.4.2. vec2(o):= vec2(1) + vec(4);
 - 7.4.3. q:= new pointer;
 - 7.4.4. free q;
 - 7.5. end;
8. begin
9. o:=5;
10. Proc;
11. c:= o + 2;
12. end.

Realice este ejercicio sobre esta misma hoja.

Identif	L-value	R-Value	Alcance	T.V.
main	-	-	3-12	2-12
c	autom.	basura	5-12	2-12
o	autom.	basura	5-12	2-12
q (línea 4)	autom.	basura	5-7.3 8-12	2-12
vec	autom.	basura	6-12	2-12
t	autom.	basura	7-12	2-12
Proc	-	-	7.1-7.5	7-7.5
vec2	semidin.	basura	7.3-7.5	7-7.5
q	autom.	null	7.4-7.5	7-7.5
q^	dinámica	basura	7.4-7.5	7.4.3-7.4.4

(línea 7.3)

25/04/2025

Realice el parcial con lapicera de otra forma se desaprobará el/los ejercicio/s.

Se considera presentismo cuando se realiza completamente un ejercicio.

Ejercicio 3

Sea el siguiente programa escrito en Pascal like, realice la ejecución del programa presentado siguiendo:

a) (Pts. 20) Cadena dinámica. b) (Pts. 5) Indique cuáles son los campos que componen un registro de activación y para qué sirve cada uno.

El lenguaje evalúa expresiones de izquierda a derecha.

Program MiPrograma

var i: integer;

var x: integer;

var a: array[1..6] of integer;

Procedure Tres

begin

i := i + 2;

x := x - 1;

end

Procedure Dos

var x: integer;

Function Fun: integer;

begin

i := i - 1;

return x + 1;

end

begin

x := 7;

a[i] := Fun();

end

Procedure Uno

var i: integer;

begin

for i:=1 to 6 begin

a[i] := i * 2;

end

i := 6

Dos();

Tres();

write(i);

end

begin

i := 3;

x := 5;

Uno();

write(i);

write(x);

for i := 1 to 6 begin

write(a[i]);

end

end

Ejercicio 4

a) (Pts. 10) Describa a qué se denomina variable local y a qué se denomina variable global

b) (Pts. 10) Indique la diferencia entre un error sintáctico y uno semántico. Puede usar ejemplos.

c) (Pts. 5) Alcanza con definir una gramática EBNF, para la construcción de números romanos. Justifique su respuesta

Exercising 1

PEDRO
CARBALLO
1435017

Hoja 1 de 4

a)

$$G = (N, T, S, P)$$

$N = \{ \langle \text{for} \rangle, \langle \text{expressionInicio} \rangle, \langle \text{condicionFin} \rangle, \langle \text{expressionLoop} \rangle, \langle \text{instruccion} \rangle, \langle \text{palabras} \rangle, \langle \text{letras} \rangle, \langle \text{digitos} \rangle, \langle \text{numeros} \rangle, \langle \text{variables} \rangle, \langle \text{whites} \rangle, \langle \text{if} \rangle, \dots \langle \text{struct} \rangle, \langle \text{tips} \rangle, \langle \text{variables} \rangle, \langle \text{signes} \rangle \}$

$$T = \{ '}', '{', ':', ';', a..z, A..Z, 0..9, 'for', '(', ')', '=', '>', '<', '+', '-' \}$$

$$S = \{ \langle \text{for} \rangle \}$$

$$P = \{$$

$$\langle \text{for} \rangle ::= \text{'for' ' (' } \langle \text{expressionInicio} \rangle \text{' : ' ') ' } \langle \text{condicionFin} \rangle \text{' ; ' } \langle \text{expressionLoop} \rangle \text{') '}$$

$$\text{' { ' } \{ \text{instruccion} \}^* \text{' } \}$$

$\langle \text{expression / value} \rangle ::= \langle \text{HPO} \rangle \langle \text{variable} \rangle \text{ ':' '}' \langle \text{numeral} \rangle$

$\langle \text{HPD} \rangle ::= (\text{'int' } | \text{'float' } | \dots)$ // se aşman tipe ya deyinir.

$\langle \text{variables} \rangle ::= \langle \text{letra} \rangle \{ (\langle \text{letra} \rangle \mid \langle \text{dígito} \rangle)^* \}$

$$\langle \text{number} \rangle ::= \{ \langle \text{digit} \rangle \}^+$$

$$\langle \text{letra} \rangle := (a | \dots | z | A | \dots | Z)$$

$$\langle \text{digit} \rangle ::= (0 \mid \dots \mid 9)$$

$$\langle \text{condition} \rangle ::= \langle \text{variable} \rangle \langle \text{signo} \rangle \langle \text{numero} \rangle$$

$$\langle \text{signo} \rangle ::= (' > ' \mid ' < ' \mid ' = ')$$

$\langle \text{expresión binaria} \rangle ::= \langle \text{variables} \rangle ('++' | '--')$

$\langle \text{instrucción} \rangle ::= \{ \langle \text{varias} \rangle \}^+$ NO HACE FALTA PONERLE LA OBLICUADA

$\langle \text{varias} \rangle ::= \{ (\langle \text{letras} \rangle | \langle \text{palabras} \rangle | \langle \text{estruct} \rangle | \langle \text{churneo} \rangle | \dots) \}^+$

$\langle \text{estruct} \rangle ::= (\langle \text{for} \rangle | \langle \text{while} \rangle | \langle \text{if} \rangle | \dots)$ // se asume que el resto de las estruct. de control están definidas.

$\langle \text{palabras} \rangle ::= \{ \langle \text{letras} \rangle \}^+$

Exercising 1b

PEDRO
CARBALLO
1435017

1b) Dig 10 \rightarrow 0 \rightarrow 1

Diagram illustrating the recursive process of finding the last digit of a number:

Input: 12345

Step 1: Extract the last digit (5) and call the function on the remaining number (1234).

Step 2: Extract the last digit (4) and call the function on the remaining number (123).

Step 3: Extract the last digit (3) and call the function on the remaining number (12).

Step 4: Extract the last digit (2) and call the function on the remaining number (1).

Step 5: Extract the last digit (1) and call the function on the remaining number (0).

Step 6: Base case reached (0), return 0.

Step 7: Return the last digit (1) to the previous call.

Step 8: Return the last digit (2) to the previous call.

Step 9: Return the last digit (3) to the previous call.

Step 10: Return the last digit (4) to the previous call.

Step 11: Return the last digit (5) to the initial call.

Final Output: 5

Palavra → LETRA → LETRA

Signo \rightarrow \rightarrow
 \downarrow
 \downarrow
 \downarrow

variable → LETRA

```
graph LR; var[variable] --> LETRA[LETRA]; LETRA --> stack[LETRA / DIGITO]; stack --> LETRA;
```

```

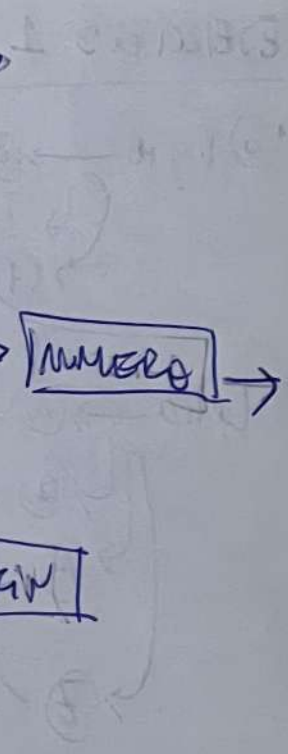
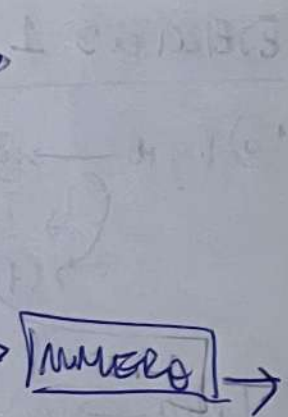
graph LR
    Tip0[Tip0] --> int((int))
    int --> Tip1[Tip1]
    Tip0 --> float(float)
    float --> Tip1

```

instruction → varies → varies

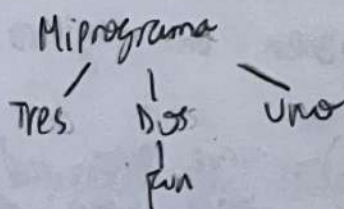
Various \rightarrow NETRO
 \rightarrow PARABRA
 \rightarrow ESTRUCT
 \rightarrow NUM

A diagram of a linked list structure. It consists of four rectangular boxes arranged vertically, each containing text: "LETRA", "PMABRA", "ESTUO", and "NUM". To the left of each box is a small arrow pointing to it. To the right of each box is a curved arrow pointing to the next box below it. A large curved arrow on the far left points from the bottom of the "NUM" box back up to the top of the "LETRA" box, indicating a circular or linked structure.



EJERCICIO 3

a) CADENA DINÁMICA :



PEDRO
CARBALLO
14350/7

Hoja 3
de 4

*1 Reg. Act. MiPrograma

PR

EE = -

ED = -

i = 1..6

x = 4

a[1] = 2

a[2] = 4

a[3] = 6

a[4] = 8

a[5] = 10

a[6] = 12

Procedure Tres

Procedure Dos

Procedure Uno

VR

i = 3
x = 5

imprime
3
4

i = 1..6

imprime
2
4
6
8
10
8

*5 Reg Act Tres

PR

EE = *1

ED = *2

VR

i = i + 2
i = 5 + 2 = 7
x = x - 1
x = 5 - 1 = 4

*2 Reg. Act. Uno

PR

EE = *1

ED = *1

i = 1..6

VR

imprime
7

*3 Reg Act Dos

PR

EE = *1

ED = *2

x = 7

Function Fun

VR = 8

x = 7
a[i] = Fun();
a[6] = Fun();
a[6] = 8

*4 Reg Act Fun

PR

EE = *3

ED = *3

VR

i = i - 1
return x + 1
return 8

b) Campos que componen un registro de activación:

Punto de Retorno (PR) = sirve para indicar el lugar en memoria donde sigue el programa luego de volver de otro procedimiento/función que fue llamado.

Enlace o Link estático (EE) = sirve para indicar la unidad que lo contiene

Enlace o Link dinámico (ED) = sirve para indicar la unidad que lo llamo

Variables = Son las variables de esa unidad

Procedimientos / funciones = son los procedimientos / funciones que contiene esa unidad.

Valor de retorno (VR) = sirve para indicar el valor que devuelve la función que llamo. En caso de devolver más de un valor, el VR se actualiza.

Ejercicio 4

PEDRO
CARBALLO
1435017

Hoja 4
de 4

- a) Variable local se denomina a aquellas variables que se encuentran declaradas dentro de una unidad local tal como un procedimiento o una función. Su alcance es (desde la línea siguiente a su declaración) dentro de esa unidad únicamente, y su tiempo de vida es desde la declaración de esa unidad hasta el final.

Variable global se denomina a aquellas variables que se encuentran declaradas fuera de todos los procedimientos, funciones locales, generalmente arriba de ellas.

Su alcance (desde la línea siguiente a su declaración) es todo el programa, y su tiempo de vida es todo el programa.

- b) Un error sintáctico es un error de cómo está escrito ese código, y no lo que significa. Es decir, es un error sintáctico cuando no se siguen las reglas léxicas del programa. Ej: (en ADA)

`C = 2;`

} error sintáctico ya que faltan los dos puntos en la ~~declaración~~ asignación:
debería ser `C := 2;`

Depende de cada lenguaje.

Estos errores se detectan en compilación.

Un error semántico es aquel error de código que está bien escrito sintácticamente pero carece de sentido para el lenguaje: ~~El error semántico~~ Depende de cada lenguaje.

Ejemplo en Pascal

```
Procedre uno  
var i: integer;  
var s: string;  
begin  
  i := 6;  
  s := "a";  
  i := i + s;  
end
```

sintácticamente
está bien
escrito pero no tiene
sentido ya que no se
puede sumar una variable
entera con un string.

Los errores semánticos se detectan en ejecución.
(Semántica dinámica)

c) No alcanza con definir una gramática EBNF para la construcción de números romanos, ya que, la gramática EBNF es una gramática libre de contexto.