Conceptos y Paradigmas de Lenguajes de Programación - 2025 - Primer Parcial 1ra Fecha. T1
25/04/2025

HOJA 1|3

Realice el parcial con lapicera de otra forma se desaprobará el/los ejercicio/s.
Se considera presentismo cuando se realiza completamente un ejercicio.

Legajo: 2306110								Corrigió:
Apellido y Nombres: Guaymas Matias Julián								Viviana
1	a B-	b B.						
2	a B	b B	c B	d B.				
3	a B.	b B						
4	a R	b B	c R-					
Resultado Final								Aprobado

Ejercicio 1

a) (Pts.25) Defina, utilizando EBNF, la gramática para una función de C. Puede considerar (además de otros) un ejemplo como (**no implemente solo el ejemplo**):

```
tipo  nombre_función (lista parámetros)
{         instrucciones
          return xxxx (valor/variable del tipo de la función)
}
```

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

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

Realice este ejercicio sobre esta misma hoja.

Identif	*L-value*	*R-Value*	*Alcance*	*T.V.*
Main (2)	——	——	3-12	2-12
b(4)	automática	basura	5-12	2-12
m(4)	automática	basura	5-12	2-12
t(4)	automática	basura	5-7.3 8-12	2-12
vec(5)	automática	basura	6-12	2-12
x(6)	automática	basura	7-12	2-12
Proc(7)	——	——	7.1-12	7-7.5
vec2(7.2)	semidinámica	basura	7.3-7.5	7-7.5
t(7.3)	automática	nil	7.4-7.5	7-7.5
t^	dinámica	basura	7.4-7.5	7.43-7.44

B

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 estática. b) (Pts. 5) Especifique cuáles son los campos que componen un registro de activación y para qué sirve cada uno de ellos.

El lenguaje evalua expresiones de izq. a der.

```
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) Indique dónde se almacenan las variables semi-dinámicas y cómo es el mecanismo que se utiliza para almacenar los valores de retorno en el registro de activación.
b) (Pts. 10) Indique cómo es la inicialización por defecto de las variables en lenguaje C al momento de su alocación en memoria.
c) (Pts. 5) ¿Qué diferencias hay entre ligadura estática y dinámica? Defina el concepto de ligadura (para los distintos atributos de las variables tomando como ejemplo al lenguaje C).

PARCIAL 1-1RA FECHA MATIAS GUAYMAS - LEGAJO: 23061/0 HOJA 2/3 25 04 25

1) a) G = {N, T, S, P}

N = {<funcionC>, <tipo>, <identificador>, <sentencia>, <retorno>, <letra>, <caracter>, <digito>, <sentencia_if>, <sentencia_for>, <sentencia_while>, <llamadoFuncion>}

T = {"(", ",", ")", "{", "}", "int", "void", "double", "String", "static", ..., "_", "a", "b", ..., "z", "A", "B", ..., "Z", 0, 1, ..., 9, "return"}

S = <funcionC>

P = {<funcionC> ::= <tipo> <identificador> "(" {<tipo> <identificador> ","}* ")" "{" {<sentencia>}* [<retorno>] "}"

<tipo> ::= ("int" | "void" | "double" | "String" | <identificador> | ...)

* ["static"]

<identificador> ::= (<letra> | "_") {<caracter>}*

<caracter> ::= (<letra> | "_" | <digito>)

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

<digito> ::= (0 | 1 | .. | 9)

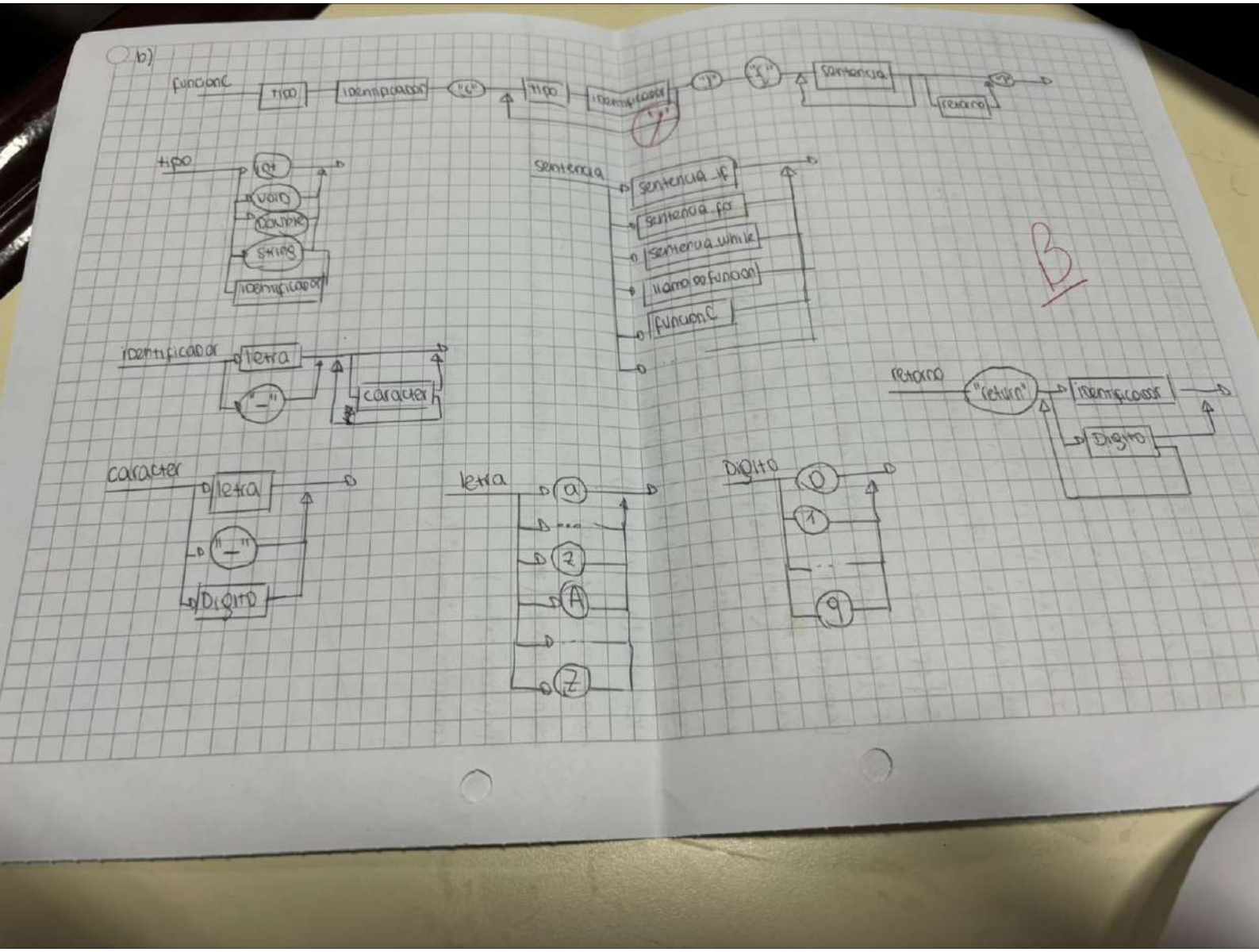
<sentencia> ::= (<sentencia_if> | <sentencia_for> | <sentencia_while> | <llamadoFuncion> | <funcionC> | ...)

<retorno> ::= "return" (<identificador> | {digito}+)

}

Le hace falta una parentesis que no cierra

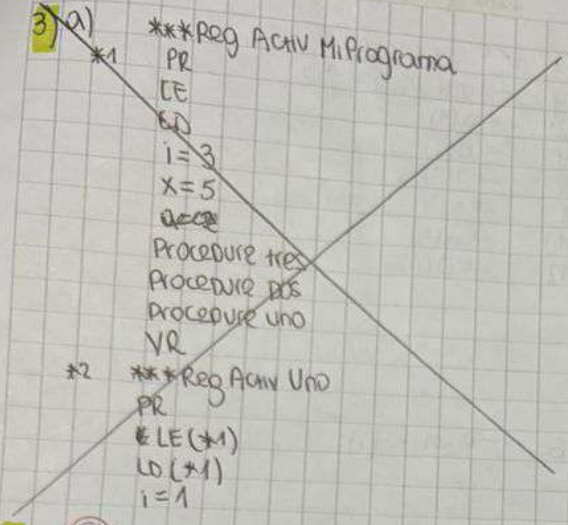
B-


b)
funcionC
tipo
identificador
"("
tipo
identificador
","
")"
"{"
sentencia
retorno
";"
tipo
int
void
DOUBLE
String
identificador
sentencia
sentencia_if
sentencia_for
sentencia_while
llamado_funcion
funcionC
B
identificador
letra
"_"
caracter
retorno
"return"
identificador
Digito
caracter
letra
"_"
Digito
letra
a
z
A
Z
Digito
0
1
9

MATÍAS GUAYMAS - LEGAJO: 23061/0

HOJA 3/3 (25) (04) (25)

3) a)



→ Está atras el punto 3

4) a) b) En C las variables globales y variables estáticas se inicializan por default. La diferencia clave independientemente de su l-valor.

entre estas es su alocación en memoria. Las variables estáticas se alocan en memoria de código mientras que las globales en la stack. Cabe mencionar que las variables locales no inicializan por default.

~~c) La ligadura estática~~

a) Las variables semidinámicas en memoria de datos. Un ejemplo podrían ser las variables de tipo vector en Ada con rango de 0 a N, N siendo una constante numérica tipada. → También soporte dinámico? No necesariamente

b) El mecanismo que se utiliza para almacenar los valores de retorno en un registro de activación es así: sólo si es se invoca una función el valor de retorno se anota en el registro de activación de quien llamó a la función y no en el registro de activación de la función llamada, de no ser así los valores se perderían porque una vez finalizada la subrutina, el registro se desaloca de memoria.

c) ligadura: momento en el que se asocia el r-valor con el l-valor.

La ligadura estática es realizada antes de la ejecución mientras que la ligadura dinámica es en ejecución.

Ejemplos en C:

R= int a → Se liga el tipo int con la variable a en compilación

static int a = 0 → Se liga el valor por default (r-valor) en compilación.

int a = 1
int a = 1+2 } El r-valor cambia en ejecución → ligadura dinámica.

Falto explicar el concepto de ligadura para cada atributo de las variables

3) a)

*1 ***Reg Activ Mi Programa
PR
LE
LD
i = 8 - 2 - 4 - 1 - 6
x = 5 - 4
a(1) = 2
a(2) = 4
a(3) = 6 - 8
a(4) = 8
a(5) = 10
a(6) = 12
Procedure Tres
Procedure DOS
Procedure Uno
VR

*2 ***Reg Activ Uno
PR
LE(*1)
LD(*1)
i = 1..6 - 6
VR

*3 ***Reg Activ DOS
PR
LE(*1)
LD(*2)
x = 7
Function Fun
VR 8

*4 ***Reg Activ Fun
PR
LE(*3)
LD(*3)
PR

*5 ***Reg Activ Tres
PR
LE(*1)
LD(*2)

Cuando hago 1..6 me reparo a que esta toma valor el valor 1, luego el 2, luego 3, ..., hasta el 6 inclusive

Imprime: 4 → var global "i"
4 → var global "x"
2 → a(1)
4 → a(2)
8 → a(3)
8 → a(4)
10 → a(5)
12 → a(6)

Imprime: 6 → write(i)

B

b)

Registro de Activación	→ Representa los valores y atributos de un bloque (procedimiento, main, función) en forma de pila.
Punto de Retorno (PR)	→ Dirección de memoria que permite continuar la ejecución normal del programa (se guarda en el punto de parada cuando el rep. se desplaza en memoria)
Link Estático (LE)	→ Tiene que ver con donde está contenido el registro de activación en base a la estructura del programa.
Link Dinámico (LD)	→ Hace referencia al rep. de activación que lo invocó. Ligado a la ejecución del programa.
Variables / Parámetros	→ Son las variables locales al registro de activación. Toman cualquier tipo y se representa la modificación de su valor en la pila.
Procedures	→ Procedimientos declarados dentro del bloque del Registro de Activación
Funciones	→ Funciones declaradas dentro del bloque del Registro de Activación
Valor de Retorno (VR)	→ Si se llamase a una función dentro de la lógica del bloque, el valor retornado por dicha función es almacenado aquí, en el que la invocó.

B