Práctica 4 CPLP

Ejercicio 1:

a) Tome una de las variables de la línea 3 del siguiente código e indique y defina cuales son sus atributos:

```
1. Procedure Practica4();
2. var
3. a,i:integer
4. p:puntero
5. Begin
6. a:=0;
7. new(p);
8. p:= ^i
9. for i:=1 to 9 do
10.a:=a+i;
11.end;
12....
13.p:= ^a;
14....
15.dispose(p);
16.end;
```

a:

- L-Value automática
- Tiempo de vida 1-16
- Alcance de la 3-16
- Creada en la entrada del bloque.
- R-Value Dinámico, inicio indefinido.
- Tipo: Integer ligado estaticamente.

b) Compare los atributos de la variable del punto a) con los atributos de la variable de la línea 4. Que dato contiene esta variable?,

Diferencia	p	^p	i
L-value	automático	dinámico	automática
T.vida	1-16	7-15	1-16
Alcance	5-16	5-16	4-16
R-value	indefinido	nil	indefinido
Tipo	Puntero	Integer	Integer

Ejercicio 2:

a. Indique cuales son las diferentes formas de inicializar una variable en el momento de la declaración de la misma.

1. No inicializamos.
2. Inicializado por defecto, por ejemplo en java.
3. Inicializamos explícitamente, ejemplo int h = 0;

b. Analice en los lenguajes: Java, C, Python y Ruby las diferentes formas de inicialización de variables que poseen. Realice un cuadro comparativo de esta característica.

Modo de inicialización.	Python	Java	C	Ruby
Por defecto	-	int x; //Queda 0	int glob; // Solo funca con globales, queda 0	-
En la misma línea	x = 0;	int x = 0;	int x = 0;	x = 0;
Declaracion e inicializacion separada	-	int x; x = 0;	int x; x = 0;	-

Ejercicio 3:

Explique los siguientes conceptos asociados al atributo l-valor de una:

a. Variable estática.

Son alocadas en tiempo de compilación en un región de memoria fija, durante toda la vida útil del programa.

Ejemplo java: (Dentro de una clase)

- static String name = “Goku”; Desde ahora en adelante se puede referenciar y siempre, todos los objetos de la clase acceden a la misma variable. Puede cambiar de valor pero no de ubicación de memoria.

b. Variable automática o semi estática.

Son alocadas cuando se aloca el bloque de código donde estaba definido. Se crean en la pila de memoria y luego se eliminan cuando se va del alcance.

Ejemplo java: (Dentro de un método)

- int x = 0;

c. Variable dinámica.

Se asignan en tiempo de ejecución en la heap, su L-Valor se determina dinámicamente durante la ejecución y puede cambiar a medida que se asigna.
Ejemplo c:

- int *p = &x;

d. Variable semidinámica.

Se asigna dinámicamente durante la ejecución del programa pero su dirección de memoria se mantiene constante.

```
1  with Ada.Text_IO; use Ada.Text_IO;
2
3  procedure Ejemplo_Variable_Semidinamica is
4     type Matriz is array (Positive range <>, Positive range <>) of Integer;
5     -- Definición de una matriz de tipo semidinámico
```

De al menos un ejemplo de cada uno.
Investigue sobre qué **tipos de variables respecto de su l-valor hay en los lenguajes C y Ada**.

Tipo	C	Ada
Estática	Se declaran usando la keyword "static" y se guardan en memoria estática, se inicializan en 0, y mantienen sus valor entre llamada de funciones. Si es creada en una función su valor perdura, se le declara un valor inicial la primera vez y luego es ignorado. Ejemplo:	No existe una palabra clave para estática, sino que efectuar un efecto similar tenemos que declararlo a nivel de paquete: package My_Package is -- Declare a static variable Static_Var : Integer := 0; Y su valor perdurara entre ejecuciones de funciones.

	#include <stdio.h>
	
	void foo()
	{
	int a = 10;
	static int sa = 10;
	
	a += 5;
	sa += 5;
	
	printf("a = %d, sa = %d\n", a, sa);
	}
	
	int main()
	{
	int i;	

	for (i = 0; i < 10; ++i)
	foo();
	}
	
	Imprime:
	a = 15, sa = 15
	a = 15, sa = 20
	a = 15, sa = 25
	a = 15, sa = 30
	a = 15, sa = 35
	a = 15, sa = 40
	a = 15, sa = 45
	a = 15, sa = 50
	a = 15, sa = 55
	a = 15, sa = 60	

Automática	Declarado dentro de un bloque cualquiera:
	
	int x = 0;
	
	Su memoria es alocada cuando entra al bloque y desalojada cuando sale.
	Tiene valor inicial indefinido
		Lo mismo que C, pero se declara asi:
		
		procedure Ejemplo is
		begin
		declare
		Variable_Automatica : Integer := 10;
		begin
		-- Código dentro del bloque declarativo
		null;
		end;
		-- La variable automáticamente se elimina al salir del bloque declarativo
		end Ejemplo;

Dinámica	int *puntero;
	
	puntero = (int*)malloc(sizeof(int));
	
	free(puntero);
		Punteros pa'
		
		with Ada.Text_IO; use Ada.Text_IO;
		
		procedure Ejemplo_Dinamico is
		type Puntero_Int is access Integer; -- Tipo de puntero a entero
		
		Puntero : Puntero_Int; -- Declaración de un puntero a entero
		
		begin
		-- Asignación de memoria dinámica

		New (Puntero); *Puntero := 42; -- Asigna el valor 42 a la dirección de memoria apuntada por Puntero -- Imprime el valor almacenado en la dirección de memoria apuntada por Puntero Put_Line("El valor almacenado en la memoria dinámica es: " & Integer'Image(*Puntero)); -- Liberación de memoria dinámica Free (Puntero); end Ejemplo_Dinamico;
Semi Dinámica	-	Los arreglos de Ada inician con una posición inicial de memoria y tamaño predefinido en compilación, sin embargo pueden cambiar de tamaño sin cambiar de posición inicial.

Ejercicio 4:

a. ¿A qué se denomina variable local y a qué se denomina variable global?
Las variables locales son declaradas dentro de un bloque de código específico y solo tiene alcance para ese bloque.
Las globales se declaran fuera de todo bloque de código, en el bloque del programa en sí, y tiene alcance de todo el programa.
b. ¿Una variable local puede ser estática respecto de su l-valor? En caso afirmativo dé un ejemplo
Si, en C por ejemplo podemos definir una variable static para que mantenga su valor entre ejecuciones de la función, ejecutándose la línea donde se define sólo una vez efectivamente.

```
1.    void foo()
2.    {
3.        int a = 10;
4.        static int sa = 10;
5.
6.        a += 5;
7.        sa += 5;
8.
9.        printf("a = %d, sa = %d\n", a, sa);
10.   }
```

En este caso, la línea static int sa = 10; solo será ejecutada la primera vez, el resto de la veces no porque sino resetear su valor entre llamadas de funciones.

c. Una variable global ¿siempre es estática? Justifique la respuesta.

No, una variable global puede ser automática, estática o no eso es independiente de su alcance.

Las estáticas hasta pueden ser locales a una función, como vimos en el ejemplo anterior, y solo retener su valor ahí.

Si declaramos una variable static en bloque de programa, entonces sera global. En C sería solo una diferencia cuando hablamos de enlazado de programa.

En C el L-valor de las globales son estáticas, pero en Java no.

Ejemplo:

"**#include <stdio.h>**
#include <stdlib.h>

static int i;
int j;

int main ()
{
//Some implementation
}

i has internal linkage so you can't use the name i in other source files (strictly translation units) to refer to the same object.

j has external linkage so you can use j to refer to this object if you declare it extern in another translation unit."

d. Indique qué diferencia hay entre una variable estática respecto de su l-valor y una constante

Simplemente: Constante no es modificable, una estática si. El L-valor queda igual para ambos.

Ejercicio 5:

a. En Ada hay dos tipos de constantes, las numéricas y las comunes. Indique a qué se debe dicha clasificación.

Las numéricas se usan para números, las comunes para caracteres o cadenas.

b. En base a lo respondido en el punto a), determine el momento de ligadura de las constantes del siguiente código:

H: constant Float:= 3,5;
I: constant:= 2;
K: constant float:= H*I;

Las constantes están ligadas en compilación.

H: Se liga cuando se le asigna el valor 3,5.

I: Se liga cuando se le asigna el valor 2.
K: Se liga luego de ligar I y H.

Ejercicio 6:

Sea el siguiente archivo con funciones de C:
Archivo.c

```
{ int x=1; (1)
        int func1();{
                int i;
                for (i:=0; i < 4; i++) x=x+1;
        }
        int func2();{
                int i, j;
                /*sentencias que contienen declaraciones y
                sentencias que no contienen declaraciones*/
                ......
                for (i:=0; i < 3; i++) j=func1 + 1;
        }
}
```

Analice si llegaría a tener el mismo comportamiento en cuanto a alocación de memoria, sacar la declaración (1) y colocar dentro de func1() la declaración static int x =1;
Es igual ya que x es global y su L-valor sería estático por defecto en C. (En ambos casos la memoria se asigna una sola vez).

Ejercicio 7:

Sea el siguiente segmento de código escrito en Java, indique para los identificadores si son globales o locales.

Clase Persona {
public long id
public string nombreApellido
public Domicilio domicilio
private string dni;
public string fechaNac;
public static int cantTotalPersonas;
//Se tienen los getter y setter de cada una de las variables
//Este método calcula la edad de la persona a partir de la fecha de
	public int getEdad(){
	public int edad=0;
	public string fN =
	this.getFechaNac();
	...
	...
	return edad;
	}
	}
	Clase Domicilio {
	public long id;

nacimiento
}
	public static int nro
	public string calle
	public Localidad loc;
	//Se tienen los getter y setter de cada una de las variables
	}

id: Local
nombreApellido: Local
domicilio: Local
dni: Local
fechaNac: Local
cantTotalPersonas: Global
--
edad: Local
fN: Local
id: Local
nro: Global
calle: Local
loc: Local

Ejercicio 8:

Sea el siguiente ejercicio escrito en Pascal
1- Program Uno;
2- type tpuntero= ^integer;
3- var mipuntero: tpuntero;
4- var i:integer;
5- var h:integer;
6- Begin
7- i:=3;
8- mipuntero:=nil;
9- new(mipuntero);
10- mipunterno^:=i;
11- h:= mipuntero^+i;
12- dispose(mipuntero);
13- write(h);
14- i:= h- mipuntero;
15- End.
a) Indique el rango de instrucciones que representa el tiempo de vida de las variables i, h y mipuntero.
Tiempo de vida de i, h y mipuntero: de 1-15
^mipuntero: 9-12 //O 15?

b) Indique el rango de instrucciones que representa el alcance de las variables i, h y mipuntero.
i: 5-15
h: 6-15
mipuntero: 4-15
^mipuntero: 4-15

c) Indique si el programa anterior presenta un error al intentar escribir el valor de h. Justifique
No, h consigue valor despues de hacerle new al puntero y asignarle valor a lo apuntado, que termina siendo i+i el valor i.

d) Indique si el programa anterior presenta un error al intentar asignar a i la resta de h con mipuntero. Justifique.
Si, mipuntero está en nil después del dispose.

e) Determine si existe otra entidad que necesite ligar los atributos de alcance y tiempo de vida para justificar las respuestas anteriores. En ese caso indique cuál es la entidad y especifique su tiempo de vida y alcance.
Para poder ligar las variables locales del programa principal, primero hay que ligar el programa principal.

f) Especifique el tipo de variable de acuerdo a la ligadura con el l-valor de las variables que encontró en el ejercicio.
i: entero
h: entero
mipuntero: puntero
^mipuntero: entero

Ejercicio 9:

Elija un lenguaje y escriba un ejemplo:
a. En el cual el tiempo de vida de un identificador sea mayor que su alcance

```
#include <stdio.h>
int example(){
	static int x = 0;
	x++;
	return x;
}
int main(){
	for (int i =0; i<10; i++)
		printf(example());
}
```

b. En el cual el tiempo de vida de un identificador sea menor que su alcance

```
program example;
type intpointer = ^integer;
var x: intpointer;
begin
        new(x);
        dispose(x);
        writeln(^x);
end.
```

c. En el cual el tiempo de vida de un identificador sea igual que su alcance

```
int x=0;
int main(){
        x++;
}
```

Ejercicio 10:

Si tengo la siguiente declaración al comienzo de un procedimiento:

- **int c; en C**
- **var c:integer; en Pascal**
- **c: integer; en ADA**

Y ese procedimiento NO contiene definiciones de procedimientos internos.
¿Puedo asegurar que el alcance y el tiempo de vida de la variable "c" es siempre todo el procedimiento en donde se encuentra definida?.
Analícelo y justifique la respuesta, para todos los casos.

No exactamente, el tiempo de vida sí será el de todo el procedimiento, pero su alcance recién sería en la próxima línea de su declaración.
Sin embargo, en términos prácticos, si, su tiempo de vida y su alcance es el mismo que el de el procedimiento si no hay cosas que lo modifiquen.

Ejercicio 11:

a) Responda Verdadero o Falso para cada opción.
El tipo de dato de una variable es?
I) Un string de caracteres que se usa para referenciar a la variable y operaciones que se pueden realizar sobre ella.
F
II) Conjunto de valores que puede tomar y un rango de instrucciones en el que se conoce el nombre.
F
III) Conjunto de valores que puede tomar y lugar de memoria asociado con la variable.
F

IV) Conjunto de valores que puede tomar y conjunto de operaciones que se pueden realizar sobre esos valores.

V

b) Escriba la definición correcta de tipo de dato de una variable.

El tipo de dato de una variable representa qué valores puede tomar, y qué operaciones puede realizar, para poder usar una variable primero se debe ligar el tipo, además con chequeo de tipos se verifica el uso correcto de la variable.

Ejercicio 12:

Sea el siguiente programa en ADA, completar el cuadro siguiente indicando para cada variable de que tipo es en cuanto al momento de ligadura de su l-valor, su r-valor al momento de alocación en memoria y para todos los identificadores cuál es su alcance y cual es su el tiempo de vida. Indicar para cada variable su r-valor al momento de alocación en memoria.

```
1. with text_io; use text_io;
2. Procedure Main is;
3. type vector is array(integer range
<>);
4. a, n, p:integer;
5. v1:vector(1..100);
6. c1: constant integer:=10;
7. Procedure Uno is;
8. type puntero is access integer;
9. v2:vector(0..n);
10. c1, c2: character;
11. p,q: puntero;
12. begin
13. n:=4;
14. v2(n):= v2(1) + v1(5);
15. p:= new puntero;
16. q:= p;
17. .......
18. free p;
19. ......
20. free q;
21. ......
22. end;
23.begin
24. n:=5;
25. .....
26. Uno;
27. a:= n + 2;
28. .....
29. end
```

Identificador	Tipo	R-valor	Alcance	T.V.
Main (linea 2)	-	-	3-29	2-29
a (línea 4)	automática	basura	5-29	2-29
n (línea 4)	automática	basura	5-29	2-29
p (línea 4)	automática	basura	5-11 23-29	2-29
v1 (linea 5)	automática	basura	6-29	2-29
c1 (linea 6)	automática	10	7-10 23-29	2-29
uno (linea 7)	-	-	8-29	7-22
v2 (línea 9)	semidinamica	basura	10-22	7-22
c1 (linea 10)	automática	basura	11-22	7-22
c2 (linea 10)	automática	basura	11-22	7-22
p	automática	nil	12-22	7-22
p^	dinámica	basura	12-22	15-18
q	automática	nil	12-22	7-29
q^	dinámica	basura	12-22	16-20

Ejercicio 13:

El nombre de una variable puede condicionar:
a) Su tiempo de vida.
b) Su alcance.
c) Su r-valor.
d) Su tipo.
Justifique la respuesta

Esta pregunta es una chota, pero suponiendo que esta bien redactada, la unica correcta en mi opinión es "su alcance", ya que el alcance es donde se conoce el nombre de una variable.

Ejercicio 14:

Sean los siguientes archivos en C, los cuales se compilan juntos
Indicar para cada variable de qué tipo es en cuanto al momento de ligadura de su l-valor.
Indicar para cada identificador cuál es su alcance y cual es su el tiempo de vida.
Indicar para cada variable su r-valor al momento de alocación en memoria

ARCHIVO1.C
```
1. int v1;
2. int *a;
3. Int fun2 ()
4. { int v1, y;
5. for(y=0; y<8; y++)
6. { extern int v2;
7. ...}
8. }
9. main()
10. {static int var3;
11. extern int v2;
12. int v1, y;
13. for(y=0; y<10; y++)
14. { char var1='C';
15. a=&v1;}
16. }
```
ARCHIVO2.C
```
17. static int aux;
18. int v2;
19. static int fun2( )
20. { extern int v1;
21. aux=aux+1;
22. ...
23. }
```

Identificador	Tipo	R-valor	alcance	T.V
v1 (linea 1)	automático	0	2-4 9-12 16-> 21-23	1-28
a (línea 2)	automática	null	3-16	1-28
*a	dinámica	basura	3-16	15-16
fun2	-	-	4-16	3-8
v1	automática	basura	5-8	3-8
y	automática	basura	5-8	3-8
main	-	-	10-16	9-16
var3	estática	0	11-16	<1-28>
v1	automática	basura	13-16	9-16
y	automática	basura	13-16	9-16
var1	automática	'C'	15	13-15

```
24. int fun3( )
25. { int aux;
26. aux=aux+1;
27. ...
28. }
```

aux	estática	0	18-25	<1-28>
v2	automática	0	7 12-16 19-28	1-28
fun2	-	-	20-28	19-23
fun3	-	-	25-28	24-28
aux	automática	basura	26-28	24-28

Ejercicio 15:

Para javascript investigue la diferencia semántica para declarar una variable utilizando los modificadores const, var, let y la ausencia de cualquiera de estos. Compárelo con un lenguaje de su preferencia.

Claro, puedo explicarte la diferencia semántica entre const, var, let y la ausencia de cualquiera de estos en JavaScript, y compararlo con otro lenguaje. Comencemos con JavaScript:

const: Se utiliza para declarar una variable cuyo valor no cambiará a lo largo del tiempo. Una vez que se asigna un valor a una variable const, no se puede reasignar. Además, la declaración const requiere que se inicialice la variable al momento de la declaración.

```
const pi = 3.1416;
```

var: Antes de la llegada de let y const, var era la forma principal de declarar variables en JavaScript. Sin embargo, tiene alcance de función en lugar de alcance de bloque, lo que significa que una variable var declarada dentro de un bloque (como un bucle for o una declaración if) estará disponible fuera de ese bloque.

```
var x = 10;
```

let: Introducido en ECMAScript 6 (ES6), let tiene un alcance de bloque, lo que significa que la variable solo es válida dentro del bloque en el que se declara.

```
let y = 20;
```

Ausencia de modificadores: Si declaras una variable sin utilizar const, var o let, se considera global (si se declara fuera de cualquier función) o local al ámbito de la función (si se declara dentro de una función). Esta forma de declarar variables puede tener consecuencias inesperadas, especialmente si se realiza en un entorno con muchas variables globales.

```
z = 30; // Global
```

Comparado con python, no hay const, las variables tiene alcance dentro del bloque declarado y todos los bloques que estén declarados dentro de su bloque después de la variable.