Conceptos y Paradigmas de Lenguajes de Programación - Práctica 6

EJERCICIO 1

- **Parámetro:** es una forma de compartir datos entre diferentes unidades. Es la más flexible y permite la transferencia de diferentes datos en cada llamada. Proporciona ventajas en legibilidad y modificalidad. Nos permite compartir datos en forma abstracta ya que indican con precisión qué es exactamente lo que se comparte.
- **Parámetro real:** es un valor u otra entidad utilizada para pasar a un procedimiento o función. Se encuentra en la parte de la invocación.
- **Parámetro formal:** es una variable utilizada para recibir valores de entrada en una rutina, subrutina, etc. Se encuentra en la parte de la declaración. Es una variable local a su entorno.
- **Ligadura posicional:** los parámetros formales y reales se ligan según la posición en la llamada y en la declaración.
- **Ligadura por palabra clave o nombre:** los parámetros formales y reales se ligan por el nombre. Se deben conocer los nombres de los parámetros formales.

EJERCICIO 2

MODO IN	Valor Valor constante
MODO OUT	Por resultado Por resultado de funciones
MODO IN / OUT	Valor - resultado Referencia Nombre

EJERCICIO 3A

TIPO DE PASAJE DE PARÁMETROS	LENGUAJE
MODO IN: Pasaje por valor o por valor constante **MODO OUT:** Pasta por resultado **MODO IN/OUT:** Pasaje por referencia	ADA
MODO IN: Pasaje por valor **MODO OUT:** No tiene un modo OUT explicativo, pero se simula usando punteros (por referencia) **MODO IN/OUT:** Se usa el pasaje por referencia mediante punteros	C
MODO IN: Pasaje por valor (aunque técnicamente es pasaje por referencia de objetos, pero para tipos primitivos se comporta como pasaje por valor) **MODO OUT:** No tiene un modo OUT explícito **MODO IN/OUT:** Pasaje por referencia de objetos, pero los tipos primitivos se comportan como pasaje por valor	RUBY

MODO IN: Pasaje por valor (para tipos primitivos) y pasaje por referencia (para objetos, aunque técnicamente es una referencia a un objeto) **MODO OUT:** No tiene un modo OUT explícito **MODO IN/OUT:** NO se permite directamente. Se puede simular mediante el uso de objetos	JAVA
MODO IN: Pasaje por valor para tipos inmutables (int, float, str, etc.) y pasaje por referencia para tipos mutables (listas, diccionarios, etc) **MODO OUT:** No tiene un modo OUT explícito **MODO IN/OUT:** Pasaje por referencia para objetos mutables	PYTHON

EJERCICIO 3B

ADA es más seguro que Pascal en el pasaje de parámetros debido a que:
- **Requiere especificar el modo del parámetro** (in, out, in out), lo que evita errores de lectura o escritura indebida.
- **Hace verificaciones estrictas en compilación**, como el uso correcto de tipos y parámetros.
- **Controla mejor el aliasing**, reduciendo errores por referencias múltiples a la misma variable.

En cambio, Pascal no distingue claramente entre estos modos, lo que puede llevar a errores más difíciles de detectar.

EJERCICIO 3C

En ADA, el manejo de parámetros **in out** depende del **tipo de dato** y se hace de forma **segura y controlada**. Aquí tienes una explicación resumida:

- **Parámetros in out en ADA según el tipo de dato:**

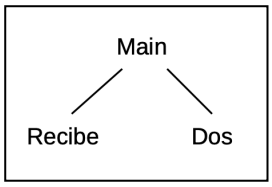
1. **Tipos escalares** (números, booleanos, caracteres): El compilador generalmente pasa por copia: crea una copia local del valor, y al final de la subrutina copia el resultado de vuelta. Esto evita efectos colaterales no deseados, como interferencias con otras variables.

2. **Tipos compuestos** (arrays, registros, objetos): Puede usarse pasaje por referencia (o una optimización equivalente) para evitar copias costosas.Esto permite modificar directamente el objeto original, pero ADA asegura que el aliasing esté controlado.

3. **Tipos con restricciones** (subtipos, tipos derivados): ADA verifica en tiempo de ejecución que los valores modificados aún cumplan las restricciones del tipo.
Esto ayuda a evitar errores como desbordamientos o violaciones de rangos.

- **Ventajas del enfoque de ADA**
- Seguridad: evita errores por aliasing y mantiene la integridad de los datos.
- Claridad: el modo `in out` deja claro que la variable será leída y modificada.
- Eficiencia: permite optimizaciones dependiendo del tipo de dato.

EJERCICIO 4-A



*1	Reg Act Main PR LE LD j: m: i: Procedure Recibe Procedure Dos VR	*2	Reg Act Recibe PR LE : *1 LD VR	*3	Reg Act Dos PR LE: *1 LD m: VR

EJERCICIO 4-B **(consultar: cuando es por referencia tengo q poner el parámetro en el reg de activación del procedimiento q lo recibe? con la flecha que apunta al procedimiento q contiene la var?)**

i- MODO IN/OUT: Referencia.
Cadena estática

*1	Reg Act Main PR LE LD j: ~~3~~ 5 m: ~~2~~ 6 i: ~~1~~ 5 Procedure Recibe Procedure Dos VR	*write (i, j, m);* Imprime 5, 5, 6	*2	Reg Act Dos PR LE: *1 LD: *1 m: 5 VR	*write (i, j, m);* Imprime 5, 5, 5	*3	Reg Act Recibe PR LE : *1 LD: *2 x: ———> i (*1) y: ———> j (*1) VR	*write (x, y, i, j, m);* Imprime 5, 5, 5, 5, 6

i- MODO IN/OUT: Referencia.
Cadena dinámica

*1	Reg Act Main PR LE LD j: ~~3~~ 8 m: 2 i: ~~1~~ 5 Procedure Recibe Procedure Dos VR	*write (i, j, m);* Imprime 5, 8, 2	*2	Reg Act Dos PR LE: *1 LD: *1 m: ~~5~~ 9 VR	*write (i, j, m);* Imprime 5, 8, 9	*3	Reg Act Recibe PR LE : *1 LD: *2 x: ———> i (*1) y: ———> j (*1) VR	*write (x, y, i, j, m);* Imprime 5, 8, 5, 8, 9

ii- MODO IN: Valor.
Cadena estática

*1	Reg Act Main PR LE LD j: 3 m: ~~2~~ 6 i: 1 Procedure Recibe Procedure Dos VR	*write (i, j, m);* Imprime 1, 3, 6	*2	Reg Act Dos PR LE: *1 LD: *1 m: 5 VR	*write (i, j, m);* Imprime 1, 3, 5	*3	Reg Act Recibe PR LE : *1 LD: *2 x: ~~1~~ 5 y: ~~3~~ 5 VR	*write (x, y, i, j, m);* Imprime 5, 5, 1, 3, 6

ii- MODO IN: Valor.
Cadena dinámica

*1	Reg Act Main PR LE LD j: 3 m: 2 i: 1 Procedure Recibe Procedure Dos VR	*write (i, j, m);* Imprime 1, 3, 2	*2	Reg Act Dos PR LE: *1 LD: *1 m: ~~5~~ 9 VR	*write (i, j, m);* Imprime 1, 3, 9	*3	Reg Act Recibe PR LE : *1 LD: *2 x: ~~1~~ 5 y: ~~3~~ 8 VR	*write (x, y, i, j, m);* Imprime 5, 8, 1, 3, 9

iii- MODO IN/OUT: Valor-Resultado
Cadena estática

*1	Reg Act Main PR LE LD j: ~~3~~ 5 m: ~~2~~ 6 i: ~~1~~ 5 Procedure Recibe Procedure Dos VR	*write (i, j, m);* Imprime 5, 5, 6	*2	Reg Act Dos PR LE: *1 LD: *1 m: 5 VR	*write (i, j, m);* Imprime 5, 5, 5	*3	Reg Act Recibe PR LE : *1 LD: *2 x: ~~1~~ 5 y: ~~3~~ 5 VR	*write (x, y, i, j, m);* *5 → Este valor se alocara en i cuando termine el procedure* *5 → Este valor se alocara en y cuando termine el procedure* Imprime 5, 5, 1, 3, 6

iii- MODO IN/OUT: Valor-Resultado
Cadena dinámica

*1	Reg Act Main PR LE LD j: ~~3~~ 8 m: 2 i: ~~1~~ 5 Procedure Recibe Procedure Dos VR	*write (i, j, m);* Imprime 5, 8, 2	*2	Reg Act Dos PR LE: *1 LD: *1 m: ~~5~~ 9 VR	*write (i, j, m);* Imprime 5, 8, 9	*3	Reg Act Recibe PR LE : *1 LD: *2 x: ~~1~~ 5 y: ~~3~~ 8 VR	*write (x, y, i, j, m);* *5 → Este valor se alocara en i cuando termine el procedure* *8 → Este valor se alocara en y cuando termine el procedure* Imprime 5, 8, 1, 3, 9

iii- MODO IN/OUT: Nombre
Cadena estática

*1	Reg Act Main PR LE LD j: ~~3~~ 5 m: ~~2~~ 6 i: ~~1~~ 5 Procedure Recibe Procedure Dos VR	*write (i, j, m);* Imprime 5, 5, 6	*2	Reg Act Dos PR LE: *1 LD: *1 m: 5 VR	*write (i, j, m);* Imprime 5, 5, 5	*3	Reg Act Recibe PR LE : *1 LD: *2 x: ↑i —> *1 y: ↑j —> *1 VR	*write (x, y, i, j, m);* Imprime 5, 5, 5, 5, 6

iii- MODO IN/OUT: Nombre
Cadena dinámica

*1	Reg Act Main PR LE LD j: ~~3~~ 8 m: 2 i: ~~1~~ 5 Procedure Recibe Procedure Dos VR	*write (i, j, m);* Imprime 5, 8, 2	*2	Reg Act Dos PR LE: *1 LD: *1 m: ~~5~~ 9 VR	*write (i, j, m);* Imprime 5, 8, 9	*3	Reg Act Recibe PR LE : *1 LD: *2 x: ↑i —> *1 y: ↑j —> *1 VR	*write (x, y, i, j, m);* Imprime 5, 8, 5, 8, 9

iii- MODO OUT: Resultado
Cadena estática y dinámica
No es posible, se produce un error al ejecutar m:= m+1+y, ya que y no fue inicializada, lo mismo con x:= i+x+j.

EJERCICIO 4-C

Si existió, en Resultado. Explicado en el inciso anterior.

EJERCICIO 4-D

El error explicado anteriormente se da tanto en la cadena estática como en la dinámica. Para el resto de tipos de pasaje de parámetros se realizaron las cadenas estáticas y dinámicas.

EJERCICIO 5

a) Para conseguir esos resultados se debe usar MODO IN/OUT por Referencia en los 3 parámetros.
b) Para conseguir esos resultados se debe usar MODO IN/OUT por Referencia para los parámetros **iteración** y **vit** y MODO IN por Valor para el parámetro **vector**.
c) Para conseguir esos resultados se puede usar MODO IN por Valor o MODO IN / OUT por Valor/Resultado en cada uno de los parámetros cuales fuera.

EJERCICIO 6

- **Un valor entero:** un único valor se comporta exactamente igual que el pasaje por referencia. (Se lee y modifica correctamente)

- **Una constante:** es equivalente a por valor. (No se puede modificar)
- **Un elemento de un arreglo:** puede cambiar el suscripto entre las diferentes referencias. (Se accede y modifica correctamente)
- **Una expresión:** se evalúa cada vez.

EJERCICIO 7-A

Cadena estática

*1	Reg Act Uno PR LE LD y: ~~1..6 1..5~~ ~~1~~ ~~2 1..6~~ ~~1...~~5 z: ~~2 3~~ 3 r1[1]: 2 r1[2]: ~~2~~ 3 r1[3]: 2 r1[4]: ~~2~~ 4 r1[5]: 2 r1[6]: 2 r2[1]: 1 r2[2]: ~~1 2 3~~ 4 r2[3]: ~~1 3~~ 5 r2[4]: 1 r2[5]: 1 Procedure Dos VR	*for y:= 1 to 6 do write (r1(y));* *for y:= 1 to 5 do write (r2(y));* Imprime 2 3 2 4 2 2 1 4 5 1 1	*2	Reg Act Dos PR LE: *1 LD: *1 x: ↑r1[y + r2(y)] —> *1 t: ↑r2[z] —> *1 io: ———> y (*1) y: ~~2~~ 3 Procedure Dos VR	*3 → Este valor se alocara en z cuando termine el procedure*	*3	Reg Act Dos PR LE : *2 LD: *2 t1:↑t —> *2 Procedure Tres VR	*4	Reg Act Tres PR LE : *3 LD: *3 VR

EJERCICIO 7-B

Cadena dinámica

*1	Reg Act Uno PR LE LD y: ~~1..6 1..5~~ ~~1~~ ~~2 1..6~~ ~~1...~~5 z: ~~2 3~~ 3 r1[1]: 2 r1[2]: ~~2~~ 3 r1[3]: 2 r1[4]: ~~2~~ 4 r1[5]: 2 r1[6]: 2 r2[1]: 1 r2[2]: ~~1 2 3~~ 4 r2[3]: ~~1 3~~ 5 r2[4]: 1 r2[5]: 1 Procedure Dos VR	*for y:= 1 to 6 do write (r1(y));* *for y:= 1 to 5 do write (r2(y));* Imprime 2 3 2 4 2 2 1 4 5 1 1	*2	Reg Act Dos PR LE: *1 LD: *1 x: ↑r1[y + r2(y)] —> *1 t: ↑r2[z] —> *1 io: ———> y (*1) y: ~~2~~ 3 Procedure Dos VR	*3 → Este valor se alocara en z cuando termine el procedure*	*3	Reg Act Dos PR LE : *2 LD: *2 t1:↑t —> *2 Procedure Tres VR	*4	Reg Act Tres PR LE : *3 LD: *3 VR

EJERCICIO 8-A

Shallow/Superficial: busca donde se ejecutará la función pasada como parámetro / “busco los parámetros donde fue llamada” (parecido a la cadena dinámica)
Deep/Profunda: se busca en el entorno donde está definida/declarada la función pasada como parámetro “busco los parámetros donde fue declarada” (parecido a la cadena estática)

EJERCICIO 8-B

SHALLOW: **Cadena estática y dinámica**

*1	Reg Act A PR LE LD x: 0 y: a Procedure B Procedure C Function D VR	*Write (x,y);* Imprime 0 a	*2	Reg Act B PR LE: *1 LD: *1 h: 1 VR	*Write (y);* Imprime a *Write (y);* Imprime a	*3	Reg Act C PR LE : *1 LD: *2 x: ~~3~~ 6 y: ~~b~~ j VR: 6	*Write (x,y);* Imprime 6 j	*4	Reg Act D PR LE : *1 LD: *3 j: ~~3~~ 6 k: ~~b~~ b VR	*Write (k);* Imprime b

DEEP: **Cadena estática y dinámica**

*1	Reg Act A PR LE LD x: 0 y: a Procedure B Procedure C Function D VR	*Write (x,y);* Imprime 0 a	*2	Reg Act B PR LE: *1 LD: *1 h: 1 VR	*Write (y);* Imprime a *Write (y);* Imprime a	*3	Reg Act C PR LE : *1 LD: *2 x: ~~3~~ 3 y: ~~b~~ j VR: 3	*Write (x,y);* Imprime 3 j	*4	Reg Act D PR LE : *1 LD: *3 j: ~~3~~ 3 k: ~~b~~ a VR	*Write (k);* Imprime a

EJERCICIO 9-A

- **Nombre**: el parámetro formal es sustituido textualmente por una expresión del parámetro real más un puntero al entorno del parámetro real. (se maneja una estructura aparte que resuelve esto). Se establece la ligadura entre parámetro formal y parámetro real en el momento de la invocación, pero la "ligadura de valor" se difiere hasta el momento en que se lo utiliza (la dirección se resuelve en ejecución). Distinto a por referencia. Es decir, no apunta a una dirección fija, puede ir cambiando (pero el nombre tiene que ser el mismo).

- **Referencia**: el parámetro formal será una variable local que contiene la dirección al parámetro real de la unidad llamadora que estará entonces en un ambiente no local. Cualquier cambio que se realice en el parámetro formal dentro del cuerpo del subprograma quedará registrado en el parámetro real.

- **Valor/Resultado**: el parámetro formal es una variable local que recibe una copia (a la entrada) del contenido del parámetro real y el parámetro real (a la salida) recibe una copia de lo que tiene el parámetro formal. Básicamente lo que hace la rutina lo copio en la variable. Cada referencia al parámetro formal es una referencia local.

- **Resultados de Impresión:**
  - **Nombre**: 3, 2, 0, 1, 1
  - **Referencia**: 1, 1, 3, 1, 1
  - **Valor/Resultado:** 1, 1, 4, 1, 1

EJERCICIO 9-B

El único que cambia en su impresión es el de por Nombre, termina imprimiendo 1,1,3,1,1.

EJERCICIO 10-A

```
program Uno;
var x: integer;

function Dos(x: integer): integer;
begin
    x := x + 1;
    Dos := x;
end;

procedure Tres(x: integer);  (* Cambiamos a pasaje por valor *)
begin
    x := x + 5;
    x := Dos(x) + 10;  (* Pasamos x a Dos *)
    write(x);  (* Para simular el efecto, escribimos aquí *)
end;

begin
    x := 8;
    Tres(x);
    (* No podemos asignar el valor de vuelta sin una variable temporal *)
end.
```

EJERCICIO 10B

Referencia	Valor/Resultado

`with Ada.Text_IO; use Ada.Text_IO;`
 	
`procedure Uno is`
`  X : Integer := 8;`
 	
`  function Dos return Integer is`
`  begin`
`  X := X + 1;  -- Modifica directamente la variable global X`
`  return X;`
`  end Dos;`
 	
`  procedure Tres (X : in out Integer) is`
`  begin`
`  X := X + 5;`
`  X := Dos + 10;`
`  end Tres;`
 	
`begin`
`  Tres(X);`
`  Put_Line(Integer'Image(X));`
`end Uno;`
	`with Ada.Text_IO; use Ada.Text_IO;`
	
	`procedure Uno is`
	`  X : Integer := 8;`
	
	`  function Dos (Val : Integer) return Integer is`
	`  Temp : Integer := Val;`
	`  begin`
	`  Temp := Temp + 1;  -- Trabaja con una copia local, no con la variable global`
	`  return Temp;`
	`  end Dos;`
	
	`  procedure Tres (X : in out Integer) is`
	`  Local_X : Integer := X;  -- Copia local para simular valor-resultado`
	`  begin`
	`  Local_X := Local_X + 5;`
	`  Local_X := Dos(Local_X) + 10;  -- Pasa Local_X a Dos`
	`  X := Local_X;  -- Asigna el valor final de vuelta a X`
	`  end Tres;`
	
	`begin`
	`  Tres(X);`
	`  Put_Line(Integer'Image(X));`
	`end Uno;`