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:**

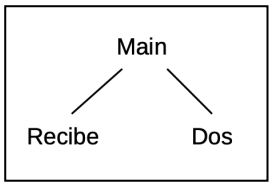
1. **Tipos escalares** (números, booleanos, caracteres): El compilador generalmente pasa por copia: crea una copia local del valor, y al final de la subrutina copia el resultado de vuelta. Esto evita efectos colaterales no deseados, como interferencias con otras variables.

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

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	*3	Reg Act Dos PR LE: *1 LD: *1 m: 5 VR	*write (i, j, m);* Imprime 5, 5, 5	*2	Reg Act Recibe PR LE : *1 LD: *2 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	*3	Reg Act Dos PR LE: *1 LD: *1 m: ~~5~~ 9 VR	*write (i, j, m);* Imprime 5, 8, 9	*2	Reg Act Recibe PR LE : *1 LD: *2 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	*3	Reg Act Dos PR LE: *1 LD: *1 m: 5 VR	*write (i, j, m);* Imprime 1, 3, 5	*2	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	*3	Reg Act Dos PR LE: *1 LD: *1 m: ~~5~~ 9 VR	*write (i, j, m);* Imprime 1, 3, 9	*2	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	*3	Reg Act Dos PR LE: *1 LD: *1 m: 5 VR	*write (i, j, m);* Imprime 5, 5, 5	*2	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	*3	Reg Act Dos PR LE: *1 LD: *1 m: ~~5~~ 9 VR	*write (i, j, m);* Imprime 5, 8, 9	*2	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	*3	Reg Act Dos PR LE: *1 LD: *1 m: 5 VR	*write (i, j, m);* Imprime 5, 5, 5	*2	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	*3	Reg Act Dos PR LE: *1 LD: *1 m: ~~5~~ 9 VR	*write (i, j, m);* Imprime 5, 8, 9	*2	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
EJERCICIO 7-B
EJERCICIO 8-A
EJERCICIO 8-B
EJERCICIO 9-A
EJERCICIO 9-B
EJERCICIO 10-A
EJERCICIO 10-B