# 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<br>Valor constante |
| MODO OUT | Por resultado<br>Por resultado de funciones |
| MODO IN / OUT | Valor - resultado<br>Referencia<br>Nombre |

EJERCICIO 3A

| TIPO DE PASAJE DE PARÁMETROS | LENGUAJE |
|---|---|
| **MODO IN:** Pasaje por valor o por valor constante<br>**MODO OUT:** Pasta por resultado<br>**MODO IN/OUT:** Pasaje por referencia | ADA |
| **MODO IN:** Pasaje por valor<br>**MODO OUT:** No tiene un modo OUT explicativo, pero se simula usando punteros (por referencia)<br>**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)<br>**MODO OUT:** No tiene un modo OUT explícito<br>**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)<br>**MODO OUT:** No tiene un modo OUT explícito<br>**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)<br>**MODO OUT:** No tiene un modo OUT explícito<br>**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 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.