Diferencias parámetros de valor y referencia

En primer lugar, **la distinción "pasar por valor frente a pasar por referencia" tal como se define en la teoría CS ahora es obsoleta** porque **la técnica originalmente definida como "pasar por referencia" ha caído en desgracia desde entonces** y rara vez se usa ahora. [1]

Los lenguajes más nuevos [2] tienden a usar un par de técnicas diferentes (pero similares) para lograr los mismos efectos (ver más abajo), que es la principal fuente de confusión.

Una fuente secundaria de confusión es el hecho de que **en "pasar por referencia", "referencia" tiene un significado más limitado que el término general "referencia"** (porque la frase lo precede

Ahora, la definición auténtica es:

- Cuando se **pasa** un parámetro **por referencia** , la persona que llama y la persona que llama **utilizan la misma variable** para el parámetro. Si la persona que llama modifica la variable del parámetro, el efecto es visible para la variable de la persona que llama.
- Cuando se **pasa** un parámetro **por valor** , la persona que llama y la persona que llama tienen **dos variables independientes** con el mismo valor. Si la persona que llama modifica la variable del parámetro, el efecto no es visible para la persona que llama.

Las cosas a tener en cuenta en esta definición son:

- **"Variable" aquí significa la variable del llamante (local o global)** , es decir, si paso una variable local por referencia y la asigno, cambiaré la variable de la persona que llama, no, por ejemplo, a lo que esté apuntando si es un puntero .
    - Esto ahora se considera una mala práctica (como una dependencia implícita). Como tal, **prácticamente todos los idiomas más nuevos son exclusivos, o casi exclusivamente de paso por valor.** Pass-by-reference ahora se usa principalmente en forma de "argumentos de salida / entrada" en lenguajes donde una función no puede devolver más de un valor.
- **El significado de "referencia" en "pasar por referencia"** . La diferencia con el término general de "referencia" es que **esta "referencia" es temporal e implícita.** Lo que el destinatario básicamente obtiene es **una "variable" que de alguna manera es "la misma" que la original.** La forma específica en que se logra este efecto es irrelevante (por ejemplo, el lenguaje también puede exponer algunos detalles de implementación - direcciones, punteros, desreferenciación - todo esto es irrelevante; si el efecto neto es este, es paso por referencia

Ahora, en los lenguajes modernos, las variables tienden a ser de "tipos de referencia" (otro concepto inventado más tarde que "pasar por referencia" e inspirado en él), es decir, los datos del objeto real se almacenan por separado en algún lugar (generalmente, en el montón), y solo las "referencias" a él se mantienen en variables y se pasan como parámetros. [3]

Pasar dicha referencia cae bajo el valor de paso por valor porque el **valor de** una variable es técnicamente la referencia en sí misma, no el objeto referido. Sin embargo, **el efecto neto en el programa puede ser el mismo que el de pasar por valor o pasar por referencia:**

- Si se toma una referencia de la variable de un llamante y se pasa como argumento, esto tiene el mismo efecto que pasar por referencia: si el objeto referido está *mutado* en la persona que llama, la persona que llama verá el cambio.
    - Sin embargo, si se *reasigna* una variable que contiene esta referencia *,* dejará de apuntar a ese objeto, por lo que cualquier otra operación en esta variable afectará a lo que esté apuntando ahora.
- Para tener el mismo efecto que el paso por valor, se realiza una copia del objeto en algún momento. Las opciones incluyen:
    - La persona que llama solo puede hacer una copia privada antes de la llamada y, en su lugar, puede proporcionarle una referencia.
    - En algunos idiomas, algunos tipos de objetos son "inmutables": cualquier operación en ellos que parezca alterar el valor en realidad crea un objeto completamente nuevo sin afectar el original. Por lo tanto, pasar un objeto de ese tipo como argumento siempre tiene el efecto de pasar por valor: se realizará automáticamente una copia para la persona que llama si necesita un cambio, y el objeto de la persona que llama nunca se verá afectado.
        - En lenguajes funcionales, *todos los* objetos son inmutables.