# 8 - Passaggio argomenti

Quando passiamo argomenti a una funzione, ci sono due modalità principali:

1. **passaggio per valore ( `T` )**
   - si fa una copia dell'argomento e questa copia viene utilizzata all'interno della funzione;
   - **impatti**:
     - **pro**: la funzione lavora su una copia, quindi non modifica l'argomento originale.
     - **contro**: se l'argomento è un oggetto "grande" (ad esempio, un contenitore come un `std::vector`), la copia può essere costosa in termini di prestazioni.
   - è opportuno utilizzarlo per tipi semplici e piccoli, come int, float, o puntatori (che sono solo indirizzi di memoria).
2. **passaggio per riferimento ( `const T&` o `T&` )**
   - la funzione lavora su un riferimento, cioè un alias dell'argomento originale. Non viene effettuata alcuna copia.
   - tipi di riferimento:
     - **riferimento a costante** ( `const T&` ): usato quando la funzione non deve modificare l'argomento originale;
     - **riferimento modificabile** ( `T&` ): usato quando la funzione deve modificare l'argomento originale.
   - **impatti**:
     - **pro**: evita la copia, quindi è più efficiente per oggetti grandi;
     - **contro**: con un riferimento modificabile, l'argomento originale può essere modificato, il che potrebbe essere indesiderato.
   - è opportuno utilizzare:
     - `const T&` per oggetti grandi che non devono essere modificati;
     - `T&` se la funzione deve modificare l'oggetto originale.

## Ritorno dei valori dalle funzioni

Due modalità principali:

1. **ritorno per valore**:
   - caso più comune: restituisce una copia del valore;
   - preferibile rispetto alla seconda modalità perché:
     - le variabili locali della funzione vengono distrutte quando la funzione termina;
     - non si possono restituire riferimenti a queste variabili locali, altrimenti si otterrebbero riferimenti dangling.
2. **ritorno per riferimento**:
   - la funzione restituisce un riferimento a un oggetto esistente (non una copia);
   - è sicuro solo se l'oggetto a cui si restituisce il riferimento ha un ciclo di vita che supera quello della funzione;
   - **pro**: evita copie costose;
   - **contro**: deve essere usato con attenzione per evitare riferimenti dangling.

***Riferimenti a rvalue***

A partire da C++11, sono stati introdotti i riferimenti a rvalue ( `T&&` ), che permettono di lavorare con oggetti temporanei in modo efficiente. Questo concetto è utile soprattutto nella gestione delle risorse e soprattutto per sostituire operazioni di copia (costose) con operazioni di spostamento (più efficienti). L'uso esplicito di riferimenti a rvalue è raro, perchè gran parte di queste ottimizzazioni avvengono automaticamente.

## Caso specifico del passaggio per valore: "passaggio per puntatore"

Quando passiamo un puntatore a una funzione, stiamo passando *per valore* l'indirizzo di memoria contenuto nel puntatore; tuttavia poiché un puntatore consente di accedere direttamente alla memoria, la funzione può modificare l'oggetto puntato. In ogni caso, è quasi sempre rimpiazzabile dal passaggio per riferimento. Un argomento di tipo puntatore ha senso quando l'argomento è *opzionale*: in questo caso, passando il puntatore nullo si segnala alla funzione che quell'argomento non è di interesse per una determinata chiamata.
9 - Array e puntatori