# Referência

No PHP, uma **referência** significa que **dois ou mais nomes de variáveis apontam para o mesmo conteúdo na memória**, funcionando como "apelidos" para o mesmo valor.

**O que é uma referência no PHP?**

* **Referência é uma associação entre nomes de variáveis e um conteúdo de variável.**
* Várias variáveis podem se referir ao mesmo conteúdo.
* Texto

  Descrição gerada automaticamenteA modificação feita por qualquer uma das variáveis **afeta todas as outras** que referenciam o mesmo conteúdo.

**O que Referências Não São?**

1. **Referências não são ponteiros**:
   * Não podemos realizar operações de aritmética de ponteiros, como mover endereços.
   * O PHP cuida dos endereços de memória para nós. Isso difere de ponteiros em linguagens como C, pois **não acessamos diretamente os endereços de memória**.
2. **Referências não criam cópias**:
   * Quando uma variável é atribuída por referência, ela não cria um novo espaço na memória.
3. **Referências não podem ser usadas para acessar diretamente a memória**:
   * Referências são sobre conteúdo, não sobre localização.

Para usarmos a referência a uma memóra com PHP, apenas precisamos colocar o um “&” antes do sifrão da declaração da variável.

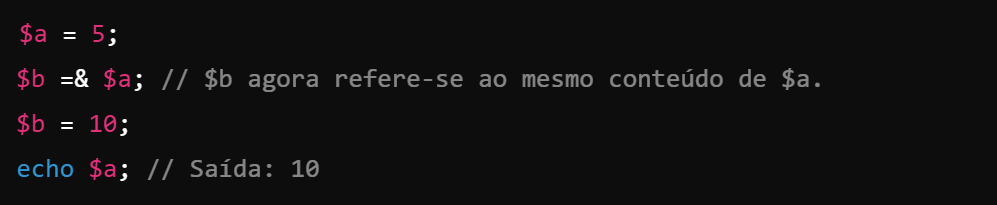


# Operações básicas

Há três operações básicas ao se utilizar referências: [atribuição por referência](https://www.php.net/manual/pt_BR/language.references.whatdo.php#language.references.whatdo.assign), [passagem por referência](https://www.php.net/manual/pt_BR/language.references.whatdo.php#language.references.whatdo.pass), e [retorno por referência](https://www.php.net/manual/pt_BR/language.references.whatdo.php#language.references.whatdo.return).

## Atribuição por referência

Quando duas variáveis apontam para o mesmo conteúdo. Qualquer modificação feita em uma variável afeta a outra.

**Variáveis indefinidas** atribuídas por referência são automaticamente criadas com o valor null.

## Passagem por referência

Quando passamos uma variável para uma função **por referência**, qualquer modificação feita na variável dentro da função **reflete diretamente na variável original**.

Texto

Descrição gerada automaticamente

## Retorno por referência

Permite que uma função retorne uma referência ao invés de um valor.

Texto

Descrição gerada automaticamenteIsso permite modificar diretamente o valor retornado.

## retornando referências

No PHP, **retornar referências** permite que você devolva um "apelido" para uma variável ou propriedade em vez de devolver uma cópia. Isso é útil quando você deseja que uma função **modifique diretamente o valor original** em vez de trabalhar com uma cópia.

**Conceito Principal**

* **Quando usamos retorno por referência (&)**, a função retorna o "endereço" de uma variável, permitindo que ela seja manipulada diretamente.
* **Cuidado**: Retornar por referência não é uma otimização de desempenho. Use somente quando houver uma **razão técnica** para modificar o valor original diretamente.

**Sintaxe Básica**

* Para retornar por referência, coloque o operador & na assinatura da função:

Texto

Descrição gerada automaticamente

* Use & ao receber o valor retornado

Texto

Descrição gerada automaticamente com confiança média

## Retorno por Referência como Propriedade de Classe

**Por que usar & no retorno?**

* Sem &, a função getValue() retornaria uma **cópia do valor** de $this->value, e mudanças na variável $myValue não afetariam $obj->value.

Texto

Descrição gerada automaticamente

Parece que não vemos muita utilidade em referências, pois elas são muito pouca usadas, usadas somente em cenários que realmente iremos sentir necessidade delas, que raramente acontece.

Alguns cenários comuns onde podem ser usadas são:

**Evitar Cópias Desnecessárias**

* Texto

  Descrição gerada automaticamenteQuando você está lidando com **estruturas de dados grandes** (como arrays), as referências podem evitar duplicação de memória.

**Passar Parâmetros por Referência**

* Texto

  Descrição gerada automaticamenteQuando você precisa que uma função **modifique diretamente os valores das variáveis passadas**.

Fora essas e outras formas que você achar que pode usar referência, evite utiliza-las, os bugs que elas causam são difícies de rastrear, a não ser que realmente precise.

# Unset ( ) & referências

Usar **unset()** em referências dentro de loops no PHP é importante para **remover a referência ativa** no final da iteração. Se você não fizer isso, a variável usada como referência no loop continuará vinculada ao **último elemento do array**, o que pode levar a comportamentos inesperados em usos futuros dessa variável.

**Por Que Isso Acontece?**

No PHP, ao usar **referências em loops** (como com foreach), a variável de referência continua existindo após o término do loop. Isso significa que, ao final da iteração:

* A variável ainda estará **vinculada ao último elemento do array**.
* Se você modificar a variável fora do loop, ela **altera diretamente o último elemento do array**.

### Exemplo Sem unset()

Texto

Descrição gerada automaticamente

**Por Que Isso Aconteceu?**

1. Dentro do loop, $valor é uma referência ao elemento atual do array.
2. Após o loop, a **referência persiste** no último elemento do array (3).
3. Quando $valor é modificado fora do loop, ele altera diretamente o último elemento.

### Solução: Usar unset()

Texto

Descrição gerada automaticamenteO comando unset($valor) quebra a referência ativa após o loop, impedindo que a variável continue vinculada ao último elemento.

**O Que Aconteceu Aqui?**

* O comando unset($valor) remove a referência, tornando $valor uma variável independente.
* Alterar $valor fora do loop não afeta mais o array