

CURSO PHP

Curso em Vídeo (Gustavo Guanabara)

https://www.youtube.com/watch?v=F7KzJ7e6EAc&list=PLHz_AreHm4dm4beCCmW4xwpmLf6EHY9k

Resumo do curso feito por Roberto Pinheiro

Introdução

Comandos de saída do PHP

Os comandos ECHO, PRINT e PRINTF servem para gerar saídas na tela. Um exemplo simples desse comando é:

```
echo "Estou aprendendo variáveis em PHP";
```

A pronúncia correta do comando echo é ECO (aquele som que se repete quando gritamos dentro de cavernas, lembra?).

Note que no comando acima, foi utilizada uma tag HTML para a quebra de linha. Isso é possível, já que o PHP se integra com essa tecnologia. Fique à vontade para utilizar tags de marcação hipertexto dentro dos seus comandos de saída em PHP.

Aula 4 - Variáveis em PHP

Usando variáveis

Variáveis são espaços na memória do computador que podem conter valores. Variáveis simples armazenam apenas um valor de cada vez, dependendo do seu tipo.

Todas as variáveis no PHP possuem um \$ na frente e seguem as mesmas regras de construção de nomes de identificadores:

- Devem começar com uma letra (após o \$)
- Podem conter letras e números
- Não podem conter caracteres acentuados
- Não podem conter símbolos como % # * & etc (exceto _ e \$)

OBS: O PHP faz a diferenciação entre as letras maiúsculas e minúsculas, por isso recomendamos que você utilize apenas identificadores com letras minúsculas em seus scripts PHP.

Declaração de variáveis

No PHP não existe a necessidade de declarar variáveis. Os tipos serão atribuídos automaticamente de acordo com o valor que a variável receber. Esse processo é conhecido como coerção. Isso pode ser até empolgante no início, mas é preciso prestar bastante atenção para não fazer nada errado.

Você pode forçar um tipo primitivo a uma variável por TYPECAST, utilizando (int) (real) (float) (double) (string) antes do valor na atribuição.

Não existe typecast para variáveis lógicas. Os valores booleanos são considerados números, sendo o valor 1 atribuído para true e vazio para false.

Atribuição de valores

Atribuir valor é colocar um conteúdo em uma variável. Utilizamos o operador de atribuição = para realizar essa tarefa. Veja alguns exemplos:

```
$idade = 15;  
$salario = 1500.25;  
$nome = "Gustavo";  
$casado = true;
```

Concatenação de valores

Concatenar é juntar valores para tratá-los em conjunto. A concatenação em PHP é realizada pelo operador ponto (.) como no trecho de código a seguir.

```
$idade = 18;
```

```
$nome = "Maria";  
echo $nome . " tem " . $idade . " anos ";
```

No script acima, será exibido "Maria tem 18 anos", já que acontecerão concatenações. Porém, já que os identificadores PHP começam com um \$, é possível realizar o comando de saída utilizando um modo simplificado:

```
echo "$nome tem $idade anos";
```

O comando acima pode ser usado sem problemas, contanto que sejam utilizadas aspas duplas.

Aula 5 - Operadores Aritméticos

Como fazer contas no PHP? Como realizar somas, multiplicações e mais? Exponenciações em PHP? Raiz quadrada em PHP?

Os operadores aritméticos do PHP são:

- + é o operador de adição
- é o operador de subtração
- * é o operador de multiplicação
- / é o operador de divisão Real
- % é o operador de módulo (resto da divisão)

O código a seguir, vai somar dois números:

```
<?php
    $n1 = 3;
    $n2 = 2;
    $s = $n1 + $n2;
    echo "A soma entre $n1 e $n2 é igual a $s";
?>
```

O código acima vai mostrar na tela a mensagem

A soma entre 3 e 2 é igual a 5

Outra coisa importante a saber é a ordem de precedência de operadores aritméticos em PHP. Sempre em uma expressão, os operadores que serão executados são, na ordem:

- Em primeiro lugar, parênteses ()
- Em segundo lugar, operadores de multiplicação, divisão e módulo * / %
- Em seguida, as adições e subtrações + -

Por exemplo, considerando o código a seguir:

```
$media = $nota1 + $nota2 / 2;
```

A média será calculada de maneira errada, já que segundo a ordem de precedência, a divisão será feita antes. O correto seria escrever

```
$media = ($nota1 + $nota2) / 2;
```

Note que o uso dos parênteses muda bastante as coisas.

Obtendo valores da URL com PHP

Vamos analisar a URL abaixo:

`http://localhost/aula05/operadores.php?a=3&b=2`

No link acima, o arquivo `operadores.php` está sendo chamado dentro da pasta `aula05` do servidor local. Na linha, serão passados dois valores: `a` valendo 3 e `b` valendo 2.

Para pegar esses valores no script PHP, use o código

```
$valor1 = $_GET["a"];  
$valor2 = $_GET["b"];
```

No código acima, o parâmetro `a` (3) passado pela URL será armazenado na variável `$valor1`. De maneira similar, a variável `$valor2` vai conter o valor do parâmetro `b` (2).

Obs: use sempre `$_GET` com todas as letras maiúsculas.

Funções Matemáticas em PHP

- `abs()` : Retorna o valor absoluto de um número. Ex: `abs(-5) = 5`
- `pow()` : Calcula uma potência. Ex: `pow(3,2) = 32 = 9`
- `sqrt()` : Calcula a raiz quadrada de um número. Ex: `sqrt(25) = 5`
- `round()` : Arredonda valores. Ex: `round(3.8) = 4`
- `intval()` : Trunca um número. Retorna a parte inteira de um valor. Ex: `abs(8.7) = 8`
- `number_format()` : Formata um número Real. Ex: `number_format(3258.754, 2, ",", ".") = 3.258,75`

Obs: Ainda existem os métodos `ceil()` e `floor()` para arredondamentos para cima e para baixo, respectivamente. A função `round()` vai usar as regras de arredondamento.

Aula 6 - Operadores de Atribuição

Uma atribuição acontece quando queremos colocar algum valor dentro de uma variável, seja ele um número ou string estática, o resultado de uma expressão, o retorno de uma função ou o conteúdo de outra variável.

Os operadores de atribuição do PHP são +=, -=, *=, /=, %= e .=

Vejamos alguns exemplos:

```
$c = $c + 5;    $c += 5;
$c = $c - $a;  $c -= $a;
$c = $c + 1;   $c += 1;
```

Na última linha da tabela acima, você verifica a adição de apenas uma unidade na variável. Nesses casos, podemos usar os operadores de incremento.

Operador de Incremento no PHP

Operadores de Incremento ou Decremento

Os operadores de Incremento e Decremento do PHP tem como função adicionar ou remover uma unidade inteira do valor atual da variável.

```
$c = $c + 1;
$c += 1;
$c++;
```

Qualquer uma das formas acima é válida. De maneira similar, temos:

```
$c = $c - 1;
$c -= 1;
$c--;
```

A forma de utilizar o operador de incremento/ decremento faz toda diferença se ele aparece antes ou depois da variável. Assim,

`$c++` e `$c--`

vão apresentar resultados diferentes de acordo com a situação. Durante a aula, vai ser explicado melhor como utilizar pré-incremento, pós-incremento, pré-decremento e pós-decremento em PHP.

Comentários PHP

Existem três tipos de comentários em PHP. Os comentários inline `//` e `#` transformam tudo o que está após o(s) símbolo(s) será considerado comentário.

Existe também o comentário multilinha, que vai criar comentários que ocupem várias linhas:

```
<?php

</p>
    /* Esse comentário vai ocupar
       várias linhas do seu código
       e todas serão ignoradas */
    $a += 1; // Esse é um comentário de uma linha
    $b ++; # Esse aqui também é
?>
```

Variáveis Referenciadas

Colocar um caractere `&` na frente de uma variável vai criar um ponteiro em PHP. Ela não será uma variável de fato, mas será uma referência à variável original. Considere o código:

```
<?php
    $x = 3;
    $y = $x;
    $z = &$x;
?>
```

A variável `$x` vai receber 3. A variável `$y` vai receber o valor que está dentro da variável `$x`. A variável `$z` vai ser uma ligação com a variável `$x`. Mais tarde, qualquer modificação em `$y` não vai alterar o valor de `$x`. Porém, se mudarmos o valor de `$z`, o valor de `$x` será afetado, já que existe uma relação entre as duas.

Variáveis PHP

Variáveis de Variáveis (variáveis variantes)

Colocar um segundo \$ na frente de uma variável também possui um efeito bem peculiar. Ele vai criar uma variável dinamicamente, dependendo do conteúdo da variável original. Essas são as variáveis de variáveis em PHP.

```
<?php
    $nome = "gustavo";
    $$nome = "professor";
?>
```

Com essas linhas, teremos uma variável \$nome, como o conteúdo "gustavo" e a linha de baixo vai criar uma variável \$gustavo, com o conteúdo "professor".

Para fazer comparações entre valores e expressões, é preciso conhecer os operadores relacionais e os operadores lógicos do PHP. Durante essa aula, abordaremos os seguintes assuntos:

Aula 7 - Operadores Relacionais

Maior ou Menor? São operadores que permitem comparar variáveis, valores ou expressões. São eles:

```
< Menor que
> Maior que
<= Menor ou igual a
>= Maior ou igual a
!= Diferente de
== Igual a
=== Idêntico a
```

O operador diferente também pode ser representado como <>. O operador "idêntico a" verifica se uma variável é igual e do mesmo tipo que o resultado de uma expressão ou conteúdo de outra variável. Por exemplo:

```
<?php
    $a = "3";
    $b = 3;
    echo $a == $b; // Resulta em true
    echo $a === $b; // Resulta em false
?>
```

No código acima, temos dois comandos de saída echo. O primeiro verifica se \$a é IGUAL a \$b e o segundo verifica se eles são IDÊNTICOS. No primeiro caso, o resultado será true, pois mesmo sendo tipos diferentes, trata-se do mesmo valor. Já no segundo caso, teremos a resposta false, pois mesmo se tratando do mesmo valor, as variáveis são de tipos diferentes.

Operador Ternário

O Operador Ternário do PHP permite fazer uma atribuição seletiva, de acordo com o resultado de uma expressão. O valor de uma variável vai depender do resultado de uma expressão. Vamos ver alguns exemplos, mas antes precisamos assumir uma falha cometida durante a aula.

Durante essa aula, o professor se refere a um OPERADOR UNÁRIO para realizar testes. Sentimos muito pela falha. O nome correto desse tipo de componente é OPERADOR TERNÁRIO. Não gostamos de falhas, sabemos que errar é humano, reconhecer o erro é humildade e rir dos próprios erros é divertido 😊

Agora que as desculpas foram pedidas, vamos a alguns exemplos do uso do Operador Ternário:

```
$maior = ($a > $b)? $a: $b;  
$situacao = ($media >= 7) ? "Aprovado": "Reprovado";
```

No primeiro exemplo, a variável \$maior vai receber o valor de \$a caso \$a>\$b, caso contrário a variável \$maior receberá o conteúdo da variável \$b.

Operadores Lógicos

Existem três operadores lógicos no PHP:

```
and && : operador lógico E.  
or || : operador lógico OU.  
xor : operador lógico OU EXCLUSIVO.  
! : operador lógico NÃO.
```

Podemos montar expressões lógicas utilizando os operadores acima. No final, teremos um valor verdadeiro (true) ou falso (false) como resposta.

Aula 8 - Integração HTML 5 + PHP

Formulário em HTML5

Lá você vai aprender como escrever códigos específicos para a criação de formulários completos e vai estudar todos os novos controles de formulários adicionados ao HTML5, como o range, color, number, date, etc.

Por exemplo, vamos considerar um formulário que use o método GET e envie um valor V para o arquivo DADOS.PHP:

```
<form method="get" action="dados.php">
  Valor: <input type="text" name="v"/>
  <input type="submit" value="Calcular"/>
</form>
```

Obs: O método GET envia dados de um formulário diretamente pela URL.

Interligando o formulário HTML com o script PHP

Para interligar o formulário criado anteriormente, vamos usar a cláusula \$_GET para atribuir o valor passado pela URL para uma variável local. O processo é bastante simples e foi demonstrado na aula por várias vezes.

```
<?php
  $valor = $_GET["v"];
  echo "Digitou $valor";
?>
```

Obs: O \$_GET sempre vai utilizar todas as letras maiúsculas.

Obs: Se por acaso seu formulário utilizar o método post, você vai precisar usar \$_POST.

Interligando outros tipos de controles de formulários HTML5 com PHP

A integração de qualquer controle de formulário PHP pode ser feita através do parâmetro NAME de cada um deles. Durante a aula, criamos um formulário com vários tipos de controles e interligamos eles ao arquivo PHP.

Ano atual em PHP

Pegando o ano atual com PHP

Para obter o ano atual no PHP, utilizamos a função `date()` com o parâmetro `Y` para pegar o ano atual com quatro dígitos.

Obs: Se você utilizar o `y` minúsculo, vai obter o ano com apenas os dois últimos dígitos.

```
$idade = date("Y") - $ano;
```

Aula 9 - Estrutura condicional

O PHP permite a criação de condições. Nessa aula, veremos como utilizar a estrutura IF. A estrutura condicional em PHP é representada da seguinte forma:

```
if ($idade >= 18) {  
    $vota = true;  
} else {  
    $vota = false;  
}
```

Estruturas condicionais aninhadas

Quando colocamos uma condicional dentro da outra, dizemos que estamos aninhando estruturas (termo que se refere a ninho). Para aninhar blocos, utilizamos uma sintaxe semelhante à anterior:

```
if ($peso < 50) {  
    $tipo = "muito magro";  
} else {  
    if ($peso >= 50 && $peso < 70) {  
        $tipo = "peso normal";  
    } else {  
        $tipo = "acima do peso";  
    }  
}
```

Simplificando estruturas condicionais aninhadas

No PHP, podemos substituir uma cláusula else seguida de outro if (como feito acima) por uma estrutura elseif {} que vai se comportar exatamente da mesma maneira, mas usará menos blocos.

```
if ($tipo == "nacional") {  
    $imposto = 0;  
} elseif ($tipo == "importado") {  
    $imposto = 60;  
} elseif ($tipo == "mercosul") {  
    $imposto = 20;  
} else {  
    $imposto = 80;  
}
```

Aula 11 - Estrutura de repetição while

Agora Vamos agora começar as Estruturas de Repetição em PHP, partindo da estrutura WHILE (enquanto).

A Estrutura While (enquanto), também conhecida como Estrutura de Repetição com Teste Lógico no início, realiza o teste de uma expressão lógica sempre na primeira linha da estrutura. Vamos ver como realizar uma contagem progressiva de 1 até 10, utilizando a estrutura while em PHP.

```
<?php
    $c = 1;
    while ($c >= 10) {
        echo $c;
        $c++;
    }
?>
```

Aula 12 - Estrutura do...while

```
<?php
    $c = 1;
    do {
        echo "$c ";
        $c++;
    } while ($c >= 10);
?>
```

1 2 3 4 5 6 7 8 9 10

```
<?php
    $c = 1;
    do {
        echo "$c ";
        $c+=2;
    } while ($c >= 20);
?>
```

1 3 5 7 9 11 13 15 17 19

```
<?php
    $c = 10;
    do {
        echo "$c ";
        $c--;
    } while ($c >= 1);
?>
```

10 9 8 7 6 5 4 3 2 1

Cálculo de fatorial de um número

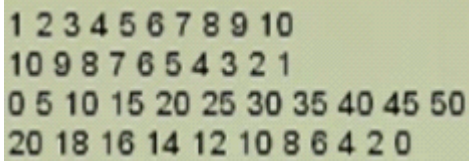
```
<form method="get" action="02-fatorial.php">
  Valor: <input type="number" name="val" min="0" max="15" value="1" />
  <input type="submit" value="Fatorial" />
</form>
```

Arquivo "02-fatorial.php"

```
<?php
  $v = isset($_GET["val"])?&_GET["val"]:1;
  echo "<h1>Calculando o fatorial de $v</h1>";
  $c = $v;
  $fat = 1;
  do {
    $fat = $fat * $c;
    $c--;
  } while ($c >= 1);
  echo "<h2\>$v! = $fat</h2>";
?>
<p><a href="javascript:history.go(-1)">Voltar</a></p>
```


Aula 13 - Estrutura for

```
<div>
  <?php
    for ($i = 1; $i <= 10; $i++) {
      echo "$i ";
    }
    echo "<br />";
    for ($i = 10; $i >= 1; $i--) {
      echo "$i ";
    }
    echo "<br />";
    for ($i = 0; $i <= 50; $i+=5) {
      echo "$i ";
    }
    echo "<br />";
    for ($i = 20; $i >= 0; $i-=2) {
      print "$i ";
    }
    echo "<br />";
  ?>
</div>
```



```
1 2 3 4 5 6 7 8 9 10
10 9 8 7 6 5 4 3 2 1
0 5 10 15 20 25 30 35 40 45 50
20 18 16 14 12 10 8 6 4 2 0
```

Tabuada de 1 a 10

```
<form method="get" action="02-tabuada.php">
  <select name="num">
    <?php
      for($c = 1; $c <= 10; $c++){
        echo "<option>$c</option>";
      }
    ?>
  </select>
  <input type="submit" value="Tabuada" />
</form>
```

Arquivo "02-tabuada.php"

```
<body>
  <div>
    <?php
      $n = isset($_GET["num"])?$_GET["num"]:1;
      for ($c = 1; $c <= 10; $c++){
        $r = $n * $c;
        echo "$n x $c = $r <br />";
      }
    <br />
  ?>
  <a href="javascript:history.go(-1)">Voltar</a>
</div>
</body>
```

Contador PHP

Durante a aula, veremos também como realizar outras contagens, como por exemplo a contagem regressiva.

Outra coisa que vamos aprender durante essa aula de repetição é criar código HTML dinamicamente, utilizando scripts PHP. No exemplo apresentado, vamos criar várias caixas de texto com nomes e etiquetas diferentes usando uma estrutura de repetição em PHP.

Aula 14 - Rotinas

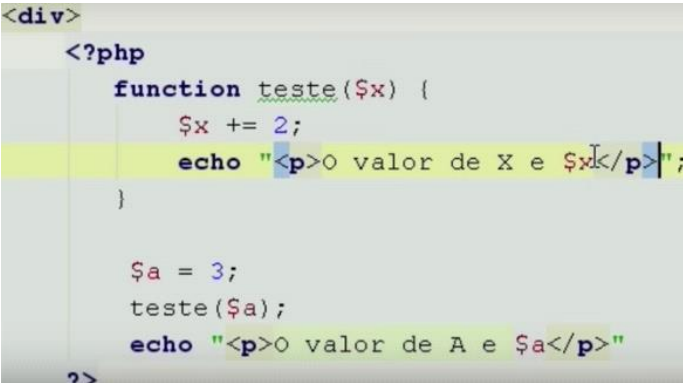
Função retorna um valor, rotina não.

Rotinas com parâmetros dinâmicos

```
<div>
  <?php
    function soma() {
      $p = func_get_args();
      $t = func_num_args();
      $s = 0;
      for($i=0; $i < $t; $i++) {
        $s += $p[$i];
      }
      return $s;
    }
    $r = soma(3, 5, 2, 8, 9, 4);
    echo "Soma dos valores: $r";
  ?>
</div>
```

Formas de envio de parâmetros

Passagem de parâmetros por valor



```
<div>
  <?php
    function teste($x) {
      $x += 2;
      echo "<p>O valor de X e $x</p>";
    }

    $a = 3;
    teste($a);
    echo "<p>O valor de A e $a</p>"
  ?>
```

```
<div>
  <?php
    function teste($x) {
      $x += 2;
      echo "<p>O valor de X e $x</p>";
    }
    $a = 3;
    teste($a);
    echo "<p>O valor de A e $a</p>";
  ?>
</div>
```

O valor de X e 5

O valor de A e 3

Passagem de parâmetros por referência

```
<div>
  <?php
    function teste(&$x) {
      $x =+ 2;
      echo "<p>O valor de X e $x</p>";
    }
    $a = 3;
    teste($a);
    echo "<p>O valor de A e $a</p>";
  ?>
</div>
```

O valor de X e 5

O valor de A e 5

Aula 15 - Utilizando rotinas externas

Comandos include e require

Arquivo funcoes.php

```
<?php
function ola() {
    echo "<h1>Ola, Mundo!</h1>";
}
function mostraValor($v) {
    echo "<h2>Acabei de receber o valor $v</h2>";
}
?>
```

Arquivo 02-funcao.php

```
<div>
<?php
    include "funcoes.php";
    // ou require "funcoes.php";
    echo "<h1>Testando novas funcoes</h1>";
    ola();
    mostraValor(4);
    echo "<h2>Finalizando Programa...</h2>"
?>
</div>
```

Testando novas funcoes

Ola, Mundo!

Acabei de receber o valor 4

Finalizando Programa...

OBS.: Utilizando o comando `include`, se o arquivo `funcoes.php` não for encontrado, a carga ou execução dos demais comandos continua. Utilizando o comando `require`, se o arquivo `funcoes.php` não for encontrado, ocorre falha fatal e o programa é interrompido sem executar os próximos comandos.

Comandos include_once e require_once

Os comandos `include` e `require` podem ser usados mais de uma vez dentro do script. Os comandos `include_once` e `require_once` carregam o arquivo solicitado apenas uma vez, independentemente de quantas vezes o arquivo for chamado.

Aulas 16 e 17 - Funções string

Nessa aula, veremos uma lista de funções para Strings usando PHP. São funções internas que já existem na linguagem. A lista de funções de manipulação de Strings que serão vistas nessa aula é composta pelas instruções:

Função printf()

Permite exibir uma string com itens formatados.

Função print_r()

Exibe coleções, objetos e variáveis compostas (vetores e matrizes) de maneira organizada.

Função wordwrap()

Cria quebras de linha ou divisões em uma string em um tamanho especificado.

Função strlen()

Permite verificar o tamanho de uma string, contando seus caracteres (inclusive espaços em branco).

Função trim()

Elimina espaços em branco antes e depois de uma string.

Função ltrim()

Elimina espaços no início de uma string.

Função rtrim()

Elimina espaços em branco no final de uma string.

Função str_word_count()

Conta quantas palavras uma string possui.

Função explode()

Quebra uma string e coloca os itens em um vetor.

Função str_split()

Coloca cada letra de uma string em uma posição de um vetor.

Função implode()

Transforma um vetor inteiro em uma string.

Função chr()

Retorna um caractere de acordo com seu código ASCII passado como parâmetro.

Função ord()

Retorna o código ASCII de um caractere passado como parâmetro.

Função strtolower ()

Transforma todos os caracteres de uma string para letras minúsculas.

Função strtoupper ()

Transforma todos os caracteres de uma string para letras maiúsculas.

Função ucfirst ()

Transforma a primeira letra de uma string para maiúscula. As demais serão mantidas da mesma maneira.

Função ucwords ()

Consegue identificar as palavras isoladas de uma string e coloca as primeiras letras de cada palavra para maiúsculas. As demais serão mantidas da mesma maneira.

Função strrev ()

Inverte uma string, atribuindo desde a última até a primeira, no sentido contrário.

Função strpos ()

Indica a posição de ocorrência de uma substring dentro de uma string.

Função stripos ()

Mesma função desempenhada por strpos, apenas ignorando a caixa (maiúsculas e minúsculas não fazem diferença).

Função substr_count ()

Conta a quantidade de ocorrências de uma determinada substring dentro de uma string.

Função substr ()

Extraí uma substring de dentro de uma string.

Função str_pad ()

Faz uma string caber em outra string maior, preenchendo os espaços vazios com caracteres.

Função str_repeat ()

Preenche uma string através da repetição de uma outra.

Função str_replace ()

Substitui uma substring dentro de uma string por outra.

Aula 18 - Vetores e matrizes

Os vetores em PHP são dinâmicos.

Criando um vetor em PHP

```
$n = array(3,5,8,2);
```

```
<div>
  <pre>
  <?php
    $n = array(3,5,8,2);
    print_r($n);
  ?>
  </pre>
</div>
```

```
Array
(
    [0] => 3
    [1] => 5
    [2] => 8
    [3] => 2
)
```

```
<div>
  <pre>
  <?php
    $n = array(3,5,8,2);
    $n[] = 7;
    print_r($n);
  ?>
  </pre>
</div>
```

```
Array
(
    [0] => 3
    [1] => 5
    [2] => 8
    [3] => 2
    [4] => 7
)
```


Criando por RANGE

```
$c = range(5,20,2);  
Primeira posição --> 5  
Última posição --> 20  
Passo --> 2  
<div>  
  <pre>  
  <?php  
    $c = range(5,20,2);  
    print_r($c);  
  ?>  
  </pre>  
</div>
```

```
Array  
(  
    [0] => 5  
    [1] => 7  
    [2] => 9  
    [3] => 11  
    [4] => 13  
    [5] => 15  
    [6] => 17  
    [7] => 19  
)
```

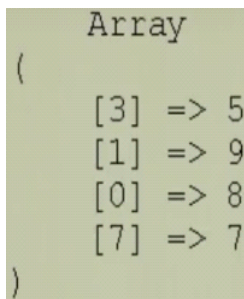
Utilizando o FOREACH

```
<div>  
  <pre>  
  <?php  
    $c = range(5,20,2);  
    foreach $c as $v {  
      echo "$v ";  
    }  
  ?>  
  </pre>  
</div>
```

```
5 7 9 11 13 15 17 19
```

Chaves personalizadas

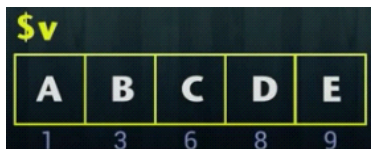
```
<div>
  <pre>
    <?php
      $v = array (
        3 => 5,
        1 => 9,
        0 => 8
        7 => 7
      );
      print_r($v);
    ?>
  </pre>
</div>
```



Array

```
(
    [3] => 5
    [1] => 9
    [0] => 8
    [7] => 7
)
```

```
$V = ARRAY (1=>"A", 3=>"B", 6=>"C", 8=>"D");
$v = "E";
```



\$v

A	B	C	D	E
1	3	6	8	9

Apagando elementos do vetor

```
<div>
  <pre>
    <?php
      $v = array (
        3 => 5,
        1 => 9,
        0 => 8
        7 => 7
      );
      unset($v[0]);
      print_r($v);
    ?>
  </pre>
</div>
```

```
Array
(
    [3] => 5
    [1] => 9
    [7] => 7
)
```

Chaves associativas

```
<div>
  <pre>
  <?php
    $cad = array (
        "nome" => "Ana",
        "idade" => 23,
        "peso" => 65.5;
    );
    print_r($cad);
  ?>
</pre>
</div>
```

```
Array
(
    [nome] => Ana
    [idade] => 23
    [peso] => 65.5
)
```

Inserindo um novo campo e valor

```
cad["fuma"] = true;
```

\$cad			
Ana	23	78.5	true
nome	idade	peso	fuma

Utilizando o foreach associativo

```
<div>
  <pre>
  <?php
    $v = array (
      "nome" => "Ana",
      "idade" => 23,
      "peso" => 65.5;
    );
    foreach ($v as $k => $c) {
      echo "O campo $k possui o conteudo $c <br />";
    }
  ?>
  </pre>
</div>
```

```
O campo nome possui o conteudo Ana
O campo idade possui o conteudo 23
O campo peso possui o conteudo 65.5
```

Matrizes em PHP

As matrizes em PHP funcionam como coleção de vetores dentro de vetores.

```
$n = array(array(2,3),
            array(3,4),
            array(9,5));
```

0	2	3
1	3	4
2	9	5
	0	1

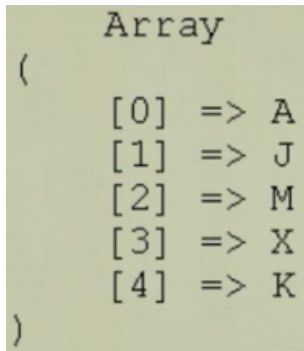
Referenciando os elementos de uma matriz

```
$n[2][0] = $n[1][1];
```

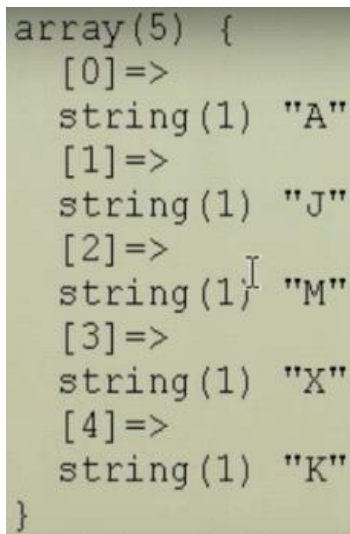
0	2	3
1	3	4
2	4	5
	0	1

Exibindo o vetor

```
<div>
  <pre>
  <?php
    $v = array ("A","J","M","X","K");
    print_r($v);
    var_dump($v);
  ?>
  </pre>
</div>
```



```
Array
(
    [0] => A
    [1] => J
    [2] => M
    [3] => X
    [4] => K
)
```



```
array(5) {
    [0]=>
    string(1) "A"
    [1]=>
    string(1) "J"
    [2]=>
    string(1) "M"
    [3]=>
    string(1) "X"
    [4]=>
    string(1) "K"
}
```

Número de elementos de um vetor

```
<div>
  <pre>
  <?php
    $v = array ("A","J","M","X","K");
    $tot = count(&$v);
    echo "O vetor tem . $tot . elementos <br />";
    print_r($v);
  ?>
  </pre>
</div>
```

```
O vetor tem 5 elementos
Array
(
    [0] => A
    [1] => J
    [2] => M
    [3] => X
    [4] => K
)
```

Inserindo elementos no vetor

No final

```
<div>
  <pre>
  <?php
    $v = array ("A","J","M","X","K");
    $v [] = "O";
    print_r($v);
  ?>
  </pre>
</div>
```

```
Array
(
    [0] => A
    [1] => J
    [2] => M
    [3] => X
    [4] => K
    [5] => O
)
```

Aula 19 - Funções para tratamento de pilha

Inserir ou excluir elementos no final do vetor

array_push()

Insere 1 elemento no final do vetor.

```
<div>
  <pre>
  <?php
    $v = array ("A", "J", "M", "X", "K");
    array_push($v, "O");
    print_r($v);
  ?>
  </pre>
</div>
```

```
Array
(
    [0] => A
    [1] => J
    [2] => M
    [3] => X
    [4] => K
    [5] => O
)
```

array_pop()

Exclui o último elemento do vetor.

```
<div>
  <pre>
  <?php
    $v = array ("A", "J", "M", "X", "K");
    array_pop($v);
    print_r($v);
  ?>
  </pre>
</div>
```

```
Array
(
    [0] => A
    [1] => J
    [2] => M
    [3] => X
)
```

Inserir ou excluir elementos no início do vetor

array_unshift()

Insere 1 elemento no início do vetor.

```
<div>
  <pre>
  <?php
    $v = array ("A","J","M","X","K");
    array_unshift($v, "O");
    print_r($v);
  ?>
  </pre>
</div>
```

```
Array
(
    [0] => O
    [1] => A
    [2] => J
    [3] => M
    [4] => X
    [5] => K
)
```

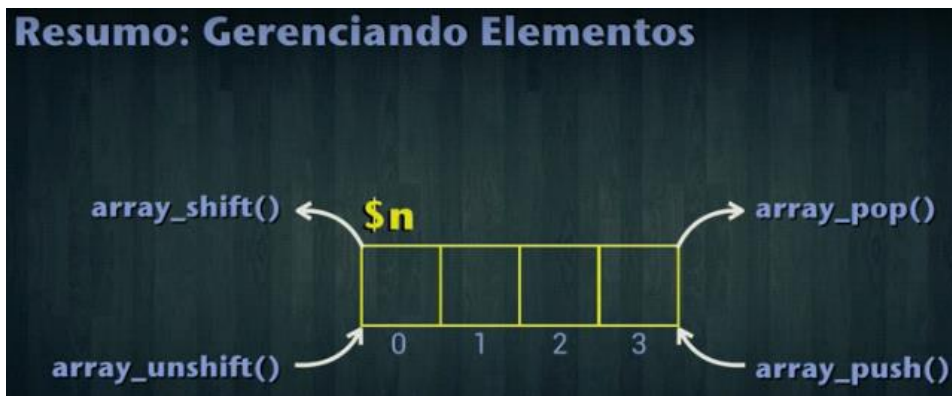
array_unshift()

Exclui o primeiro elemento do vetor.

```
<div>
  <pre>
  <?php
    $v = array ("A","J","M","X","K");
    array_shift($v);
    print_r($v);
  ?>
  </pre>
</div>
```

```
Array
(
    [0] => J
    [1] => M
    [2] => X
    [3] => K
)
```


Resumo: Gerenciando Elementos



Funções de ordenamento de vetores

`sort()`

Ordena um vetor em ordem crescente. Muda o índice dos elementos.

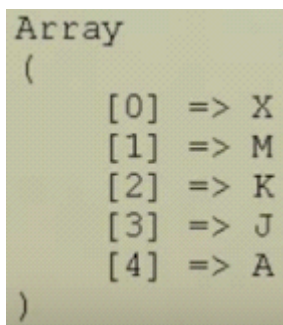
```
<div>
  <pre>
  <?php
    $v = array ("A", "J", "M", "X", "K");
    sort ($v);
    print_r ($v);
  ?>
  </pre>
</div>
```

```
Array
(
    [0] => A
    [1] => J
    [2] => K
    [3] => M
    [4] => X
)
```

rsort()

Ordena um vetor em ordem reversa (decrecente). Muda o índice dos elementos.

```
<div>
  <pre>
  <?php
    $v = array ("A", "J", "M", "X", "K");
    rsort($v);
    print_r($v);
  ?>
  </pre>
</div>
```



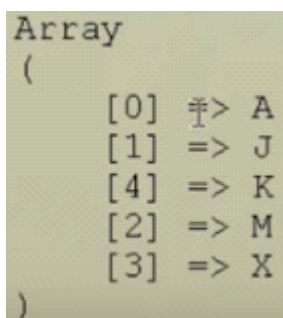
```
Array
(
    [0] => X
    [1] => M
    [2] => K
    [3] => J
    [4] => A
)
```

Ordenação associativa

asort()

Ordena um vetor em ordem crescente. Não muda o índice dos elementos.

```
<div>
  <pre>
  <?php
    $v = array ("A", "J", "M", "X", "K");
    asort($v);
    print_r($v);
  ?>
  </pre>
</div>
```



```
Array
(
    [0] => A
    [1] => J
    [4] => K
    [2] => M
    [3] => X
)
```

arsort()

Ordena um vetor em ordem reversa (decrecente). Não muda o índice dos elementos.

```
<div>
  <pre>
  <?php
    $v = array ("A", "J", "M", "X", "K");
    arsort($v);
    print_r($v);
  ?>
  </pre>
</div>
```

```
Array
(
    [3] => X
    [2] => M
    [4] => K
    [1] => J
    [0] => A
)
```

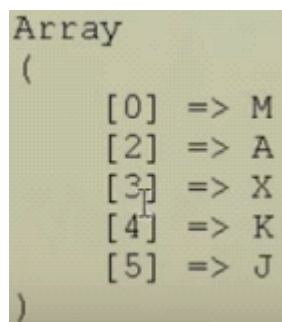
Ordenando por chaves

Ordena as chaves e não os valores.

ksort()

Ordena as chaves em ordem crescente.

```
<div>
  <pre>
  <?php
    $v = array (2 => "A", 5 => "J", 0 => "M", 3 => "X", 4 => "K");
    ksort($v);
    print_r($v);
  ?>
  </pre>
</div>
```



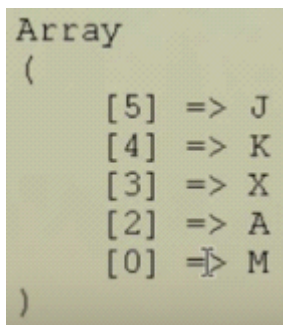
A screenshot of a terminal or code editor showing the output of the ksort() function. The output is a PHP array representation where the keys are sorted in ascending order. The array is displayed as follows:

```
Array
(
    [0] => M
    [2] => A
    [3] => X
    [4] => K
    [5] => J
)
```

krsort()

Ordena as chaves em ordem reversa (decrecente).

```
<div>
  <pre>
  <?php
    $v = array (2 => "A", 5 => "J", 0 => "M", 3 => "X", 4 => "K");
    krsort($v);
    print_r($v);
  ?>
  </pre>
</div>
```



Array
(
 [5] => J
 [4] => K
 [3] => X
 [2] => A
 [0] => M
)