7.0 Manipulação de arrays

Objetivos

Abordar de forma clara as principais estruturas de um array; Mostrar a sua criação e manipulações possíveis; Definir arrays multidimensionais ou matrizes; Determinar formas de interações e acessos.

Um array no PHP é atualmente um conjunto de valores ordenado. Podemos relacionar cada valor com uma chave, para indicar qual posição o valor está armazenado dentro do *array*. Ele é otimizado de várias maneiras, então podemos usá-lo como um array real, lista (vetor), hashtable (que é uma implementação de mapa), dicionário, coleção, pilha, fila e provavelmente muito mais. Além disso, o php nos oferece uma gama enorme de funções para manipulá-los.

A explicação dessas estruturas estão além do escopo dessa apostila, mas todo conteúdo aqui abordado trás uma boa base para quem estar iniciando o conteúdo de array.

7.1 Criando um Array

Arrays são acessados mediante uma posição, como um índice numérico. Para criar um array pode-se utilizar a função array([chave =>] valor, …). Exemplo:

Sintaxe:

```
$nomes = array('Maria','João','Alice','Alex');
```

```
$nomes = array(0=>'Maria', 1=>'João', 2=>'Alice', 3=>'Alex');
```

ou

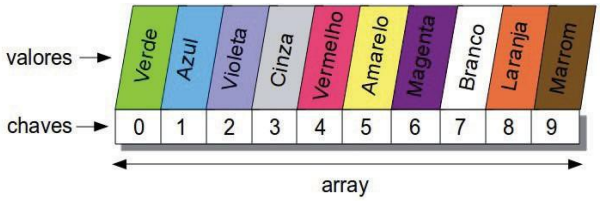
Nessa sintaxe temos duas formas de declarar uma variável do tipo array. Onde a chave o índice podem ser de forma automática como no primeiro exemplo, ou manual como no segundo. Outro detalhe importante é que: todo array começa pela chave ou índice de numero 0, quando o mesmo não é declarado.

Também temos outras formas de criar um array, onde simplesmente podemos adicionar valores conforma a sintaxe abaixo:

```
$nome[] = 'Maria';
$nome[] = 'João';
$nome[] = 'Carlos';
$nome[] = 'José';
```

A figura abaixo representa um array que tem como valor representação de cores, e possui dez posições,

Cada posição representa uma cor, seu índice (chave) vai de 0 até 9. Veja:



Em código temos:

```
$cores = array('Verde','Azul','Violeta','Cinza','Vermelho','Amarelo',
               'Margenta','Branco','Laranja','Marrom');
```

7.2 Arrays Associativos.

Aos arrays associativos associa-se um determinado valor ou nome a um dos valores do array.

O array associativo usa strings como índice, onde cada string pode representas uma chave.

Observe a sintaxe:

```
$var = array('texto1'=>'valor1','texto2'=>'valor2',...,'textoN'=>'valorN');
```

Observe que quando usamos arrays associativos, a compreensão é mais fácil, dando mais legibilidade ao código. Porém não é utilizado quando usamos um array dentro de um laço

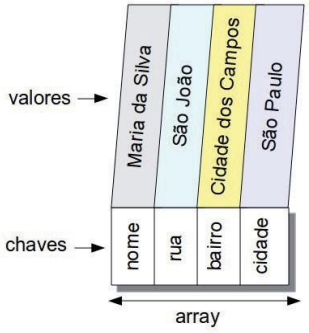
```
1  <?php
2  $dados = array('nome'=>'Maria Cristina','rua'=>'São João',
3                 'bairro'=>'Cidade dos Campos',
4                 'cidade'=>'São Paulo');
5  ?>
```

(loop), mas em outros casos sempre é bom utilizar arrays associativos. Veja um exemplo:

Outra forma de iniciarmos o mesmo array é adicionar valores conforme abaixo:

```
1  <?php
2  $dados['nome']='Maria Cristina';
3  $dados['rua']='São João';
4  $dados['bairro']='Cidade dos Campos';
5  $dados['cidade']='São Paulo';
6  ?>
```

A imagem abaixo representa os exemplo anteriores:



Umas das vantagem do array associativo é quando fazemos o acesso ao array, onde temos de forma clara e compreensível o valor que aquela chave pode conter. Como por exemplo nome, onde só vai existir o nome de pessoas. Veja abaixo um exemplo de acesso ao valores armazenados em um array dessa natureza.

Exemplo:

```php
<?php
$dados['nome']='Maria Cristina';
$dados['rua']='São João';
$dados['bairro']='Cidade dos Campos';
$dados['cidade']='São Paulo';
echo $dados['nome'];   # Resultado = Maria Cristina
echo $dados['rua'];    # Resultado = São João
echo $dados['bairro']; # Resultado = Cidade dos Campos
echo $dados['cidade']; # Resultado = São Paulo
?>
```

Dessa forma podemos acessar o array. Basta determinar o nome do array e qual a chave, onde cada chave tem um valor já determinado. Resultará em um erro o uso de uma chave errada.

7.3 Interações

Quando falamos de interações em um array estamos dizendo o mesmo que percorrer esse array usando mecanismos da própria linguagem. Como isso as interações podem ser feitas de várias formas, mas no PHP podem ser iterados pelo operador FOREACH que já vimos anteriormente.

Exemplo:

```php
<?php
$dados['nome']='Maria Cristina';
$dados['rua']='São João';
$dados['bairro']='Cidade dos Campos';
$dados['cidade']='São Paulo';
//loop que percorre todo array.
foreach($dados as $chave => $dados){
    echo $chave." = ".$dados."<br>";
}
?>
```

resultado:

```
nome = Maria Cristina
rua = São João
bairro = Cidade dos Campos
cidade = São Paulo
```

Esse tipo de interação é muito utilizado, principalmente quando temos arrays associativos.

Dicas: Sempre que se depararem com arrays, onde haja a necessidade de percorrer suas informações independentemente da chave, procure sempre utilizar mecanismos de programação mais simplificados como FOREACH.

7.4 Acessando um Array

Quando criamos um array temos que ter em mente que estamos criando uma variável que possui vários valores e que os mesmo podem ser acessados a qualquer momento. Cada valor está guardado em uma posição que pode ser acessada através de uma chave.

```
nome_do_array[ chave_de_acesso ];
```

A sintaxe para acesso simplificado de um array é a seguinte:

Temos que ter cuidado ao passar uma chave para o array, pois ela deve conter o mesmo nome de qualquer umas das chaves existentes no array. Caso a chave não exista, o valor não poderá ser resgatado. A sintaxe acima retorna um valor contido no array, por esse motivo temos que

```php
<?php
//criamos um array com valores.
$meu_array = array('nome','telefone','rua','cidade');

//vamos acessar uma das posições.
$r = $meu_array[1]; //atribuindo valor resgatado a $r.
//imprimi na tela.
echo $r;

?>
```

atribuir esse valor como mostra o exemplo abaixo:

Resultado: telefone.

7.5 Alterando um Array

Podemos alterar qualquer valor de um array. É muito semelhante ao acesso, onde, a diferença está na chamada do array. É nesse momento que atribuímos um novo valor.

Veja a sintaxe:

```
nome_do_array[ chave_de_acesso ] = <novo_valor>;
```

Observe o exemplo abaixo:

```php
<?php
//criamos um array com valores.
$meu_array = array('nome','telefone','rua','cidade');
//alterando o valor de uma das posições.
$meu_array[1] = 'sobrenome';
//imprimi na tela.
foreach($meu_array as $valores)
echo $valores."<br>";

?>
```

Resultados:

nome
sobrenome
rua
cidade

```php
<?php
//criamos um array com valores.
$produto['produto'] = 'Arroz';
$produto['valor'] = 1.35;
// alterando o valor do produto.
$produto['valor'] += .63;
// alterando o nome do produto.
$produto['produto'] .= ' Tio João ';
//imprimi valores na tela.
foreach($produto as $valores)
echo $valores."<br>";

?>
```

Vimos no exemplo anterior o valor da posição 1 do array ('telefone') foi alterada para sobrenome. Vale ressaltar que esse array tem suas chaves definidas de forma automática. A primeira posição é 0, a segunda é 1, e assim sucessivamente. Veja mais um exemplo onde

alteramos o valor, mas usando o operador de atribuição "+=" e concatenação ".=":

Resultados: Arroz Tio João 1.98

Podemos observar que assim como as variáveis "comuns", a forma de alterar o valor de um array é igual. A diferença está na chamada do array, pois temos que passar a chave além do valor que queremos atribuir. Outro detalhe importante é o tipo de valor, onde supostamente devemos atribuir os tipos compatíveis. Ou seja, se o valor atribuído a chave produto for do tipo string, não podemos usar os operadores de atribuição para atribuir um outro tipo, porém podemos mudar o tipo do valor pelo operador de atribuição simples (=).

Exemplo:

```
$var[2] += 1.90;            ff o tipo é um ponto flutuante antes e depois.
$var[2] = 'bom  dia' ;           ff agora temos a mudança de tipo ponto
flutuante para string.
```

7.6 Arrays multidimensionais

Os arrays multidimensionais são estruturas de dados que armazenam os valores em mais de uma dimensão. Os arrays que vimos até agora armazenam valores em uma dimensão, por isso para acessar às posições utilizamos somente um índice ou chave. Os arrays de 2 dimensões salvam seus valores de alguma forma como em filas e colunas e por isso, necessitaremos de dois índices para acessar a cada uma de suas posições.

Em outras palavras, um array multidimensional é como um "contêiner" que guardará mais valores para cada posição, ou seja, como se os elementos do array fossem por sua vez outros arrays.

Outra ideia que temos é que matrizes são arrays nos quais algumas de suas posições podem conter outros arrays de forma recursiva. Um array multidimensionais pode ser criado pela função array():

Na figura abaixo temos a representação de um array com duas dimensões.

Linhas \ Colunas	0	1	2	3	4
0	0.0	0.1	0.2	0.3	0.4
1	1.0	1.1	1.2	1.3	1.4
2	2.0	2.1	2.2	2.3	2.4
3	3.0	3.1	3.2	3.3	3.4
4	4.0	4.1	4.2	4.3	4.4

Uma diferença importante de um array comum para um multidimensional é a quantidades de chaves (índices), onde cada um dos índices representa uma dimensão. Observe o código da representação ao lado.

Código:

```php
<?php

$num = array(array(0.0 , 0.1 , 0.2 , 0.3),
             array(1.0 , 1.1 , 1.2 , 1.3),
             array(2.0 , 2.1 , 2.2 , 2.3),
             array(3.0 , 3.1 , 3.2 , 3.3));

?>
```

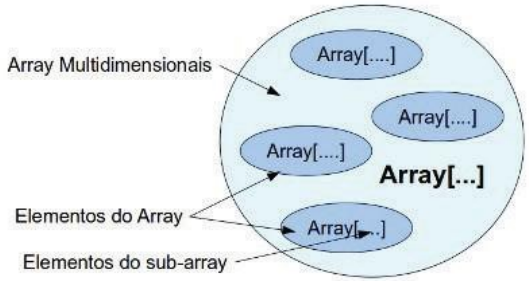
Outra forma de iniciar o array:

```php
<?php

$num[0][0] = 0.0; $num[0][1] = 0.1; $num[0][2] = 0.2; $num[0][3] = 0.3;
$num[1][0] = 1.0; $num[1][1] = 1.1; $num[1][2] = 1.2; $num[1][3] = 1.3;
$num[2][0] = 2.0; $num[2][1] = 2.1; $num[2][2] = 2.2; $num[2][3] = 2.3;
$num[3][0] = 3.0; $num[3][1] = 3.1; $num[3][2] = 3.2; $num[3][3] = 3.3;

?>
```

Observe que temos uma chave para representar a linha e outra para representar a coluna, assim, determinando uma matriz 4x4. Podemos ver também que inicializamos um array dentro do outro. Cada sub-array é uma linha, e cada elemento do array maior representa as colunas.



Para acessarmos o valor de um array multidimensional, basta colocar as duas ou mais chaves da posição que queremos acessar. É muito semelhante ao array de uma única dimensão.

Observe o acesso aos exemplos anteriores: Sintaxe:

```
nome_do_array[ chave_1 ][ chave_2 ]...[chave_n] ;
```


Exemplo:

```php
<?php

$num = array(array(0.0 , 0.1 , 0.2 , 0.3),
             array(1.0 , 1.1 , 1.2 , 1.3),
             array(2.0 , 2.1 , 2.2 , 2.3),
             array(3.0 , 3.1 , 3.2 , 3.3));
// acessando o array linha 1 coluna 2.
echo $num[1][2];
//resultado 1.2;

?>
```

Dessa forma podemos acessar o elemento numérico 1.2 que está guardado na posição linha 1 coluna 2, lembrando que o primeiro elemento de um array é 0. Abaixo, um exemplo que acessa todos os valores do array e imprime com quebra de linha:

```php
<?php
$num = array(array(0.0 , 0.1 , 0.2 , 0.3),
             array(1.0 , 1.1 , 1.2 , 1.3),
             array(2.0 , 2.1 , 2.2 , 2.3),
             array(3.0 , 3.1 , 3.2 , 3.3));

foreach($num as $linha){
    foreach($linha as $valores){
        $i++;
        echo $valores." , ";
        if($i == 4){
           echo "<br>";
           $i = 0;
        }
    }
}
?>
```

Resultado:

0 , 0.1 , 0.2 , 0.3 ,
1 , 1.1 , 1.2 , 1.3 ,
2 , 2.1 , 2.2 , 2.3 ,
3 , 3.1 , 3.2 , 3.3 ,

0.0 é igual a 0 , 1.0 é igual a 1 e assim sucessivamente.

Explicando o código:

Linha 2 – criamos um array de duas dimensões.

Linha 7 – temos um foreach para percorrer o primeiro array, ele retorna na variável $linha os sub-arrys contido no array maior.

Linha 8 – agora temos outro foreach que vai percorrer os valores dos arrays passando pra

variável $linha.

Linha 9 – criamos uma variável $i para contar os elementos. Linha 10 – imprime os valores.

Linha 11 – temos um IF, quando $i for igual a 4, significa que podemos executar o código pertencente ao If, determinando que chegou ao quarto elemento do array.

Linha 12 – quebra de linha com o
.

Linha 13 – zera a variável $i para começar a contagem de novo.

7.7 Funções com Arrays

Em PHP temos um conjuntos de funcionalidades e que já vem prontas para serem utilizadas. Trata-se de funções que já estão pré-definidas, você pode encontrá-las facilmente no site php.net.

Abordaremos agora funções utilizadas exclusivamente para manipulação de array, funções de acesso, ordenação, dentre outras. Obviamente que não falaremos de todas, pois existem muitas funções, mas mostraremos as mais utilizadas, e outras que são definidas como principais.

var_dump

Essa função é muito usada por programadores que pretendem realizar debug (análise mais detalhado para encontrar supostos erros). Observe um exemplo prático:

```php
<?php

// declaração de um array
$vetor = array('Palio','Gol','Fiesta','Corsa');
//chamada da função var_dump passando $vetor
var_dump($vetor);

?>
```

Saída no Browser:

print_r

Imprime o conteúdo de uma variável assim como var_***dump***(), porém com um formato menos detalhado e mais legível ao programador. Exemplo:

código:

```php
<?php

// declaração de um array
$vetor = array('Palio','Gol','Fiesta','Corsa');
//chamada da função print_r passando $vetor
print_r($vetor);

?>
```

Saída no Browser:

Dicas: Ao olhar o código fonte da página aberta, nota-se o código bem organizado, porém os comentários não podem ser vistos. Procure olhar o código-fonte sempre que tiver dúvida de que código HTML o PHP está gerando. Clique com botão direito do mouse, procure código fonte, e observe o que é gerado!

array_push

Adiciona elementos ao final de um array. Tem o mesmo efeito de utilizar a sintaxe:

```
var_dump( nome_do_array , novo_valor)
```

Exemplo:

```php
<?php
$var = array("valor-1","valor-2","valor-3");
array_push($var,"novo_valor");
var_dump($var);
?>
```

Resultado:

```
array(4) {
  [0]=>
  string(7) "valor-1"
  [1]=>
  string(7) "valor-2"
  [2]=>
  string(7) "valor-3"
  [3]=>
  string(10) "novo_valor"
}
```

array_pop

Remove um valor no final de um array. Sintaxe:

```
array_pop(nome_do_array);
```

Exemplo:

```php
<?php
$var = array("valor-1","valor-2","valor-3");
array_pop($var);
var_dump($var);
?>
```

Resultado:

```
array(2) {
  [0]=>
  string(7) "valor-1"
  [1]=>
  string(7) "valor-2"
}
```

array_shift

Remove um elemento do início de um array, sintaxe:

```
array_shift( nome_do_array )
```

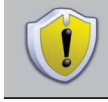
Exemplo:

```php
<?php
$var = array("valor-1","valor-2","valor-3");
array_shift($var);
var_dump($var);
?>
```

Resultado:

```
array(2) {
  [0]=>
  string(7) "valor-2"
  [1]=>
  string(7) "valor-3"
}
```

Como já sabemos como lidar com essas funções, observemos que basta conhecer a sintaxe para montarmos um exemplo. Apresentaremos agora de forma resumida as demais funções.



Atenção!

Todas as funções aqui apresentadas são para mostrar ao aluno as formas de trabalharmos com determinadas funções. Fica a critério do aluno se aprofundar ou não nesse conhecimento, uma vez que exista inúmeras funções.

Observe a tabela abaixo com outras funções:

Funções	Definição	Sintaxe:
array_unshift	*Adicionar um elemento no inicio de um array.*	array_unshift(nome_array , novo_valor)
array_pad	*Preenche um array com valores, determina a quantidade de posições.*	array_pad(nome_array, tamanho ,valor)
array_reverse	*Recebe um array e retorna-o na ordem inversa.*	array_reverse(nome_array, valor_booleano)
array_merge	*Uni dois arrays criando um novo array.*	$novo_array = array_merge(array_1,array_2)
array_keys	*Retorna somente as chaves do array.*	array_keys(nome_array)
array_values	*Cria um novo array com os valores de outro.*	$novo_array = array_values(outro_array)
array_slice	*Extrai posições de um array.*	$var = array_slice(nome_array, inicio, tamanho)
count	*Conta quantos elementos tem um array*	$var = count(nome_array)

Funções	Definição	Sintaxe:
in_array	*Verifica se um array possui determinado valor.*	in_array(valor_buscado, nome_array)
sort	*Ordena um array pelos valores.*	sort(nome_array)
rsort	*Ordena um array pelos valores de ondem reversa.*	rsort(nome_array)
explode	*Converte uma string em um array.*	explode(separador, nomer_string)
implode	*Converte um array em uma string*	implode(separador, nome_array)

Essa tabela mostra as funções mais comuns. De repente você pode se deparar com algumas delas em códigos já feito, ou baixados da internet. Exemplo disso são as ferramentas CMS, como joomla ou wordpress, que são feitas em PH. Existem muitas combinações dessas funções e outras a mais em seu código-fonte. Veja açguns exemplos utilizando algumas das funções anteriores.

```php
<?php
//Criamos dois arrays.
$array_1 = array("i","r","c","o","u","g","t","k","y","q","b","n");
$array_2 = array("f","m","l","v","j","x","z","a","p","h","d","s","e");
//Adicionar o elemento w no início do $array_1.
array_unshift($array_1,"w");
//Unimos dois arrays criando um novo array $alfabeto.
$alfabeto = array_merge($array_1,$array_2 );
sort($alfabeto);  // ordena em ordem alfabética de a-z.
foreach($alfabeto as $letra){echo "-".$letra;} //imprime na tela
rsort($alfabeto); // ordena em ordem alfabética de z-a.
echo "<br>"; // quebra de linha.
foreach($alfabeto as $letra){echo "-".$letra;} //imprime na tela
?>
```

Resultado: -a-b-c-d-e-f-g-h-i-j-k-l-m-n-o-p-q-r-s-t-u-v-w-x-y-z
-z-y-x-w-v-u-t-s-r-q-p-o-n-m-l-k-j-i-h-g-f-e-d-c-b-a

Exercício resolvido com array.

Quais serão as letras que serão geradas a partir desse código?

```php
<?php

$valor = 2;
$indice = $valor;

$letras = array('d','b','e','a','i','o','r');

echo $letras[$indice++]." ";
echo $letras[$indice]." ";
echo $letras[$indice-2]." ";
echo $letras[0]." ";
echo $letras[$valor+1]." ";


?>
```

Entendendo o algorítmo:

- No escopo temos uma variável chamada "valor" = 2 e "indice" que recebe o conteúdo que está em valor que é 2 também.
- Temos também um array que irá receber um conjunto descrito, lembrando que o indice inicial de um vetor é zero até a sua dimensão menos um. Nesse caso temos 7 elementos, então os indices para o meu vetor são de zero até 6.
- Para cada impressão, temos uma letra seguido de uma concatenação com um espaço em branco, então podemos chegar a conclusão que irá ser impresso um conjunto de letras com e espaços.
- O primeiro "echo" irá imprimir o valor de "indice" = 2 que é representado pela letra **"e"**, em seguida indice será incrementado passando a ser igual a 3.
- O segundo "echo" irá imprimir o valor de indice = 3 que é representado pela letra **"a"**.
- O terceiro "echo" irá imprimir o valor de (indice - 2) = (3-2) = 1 que é representado pela letra **"b"**
- O quarto "echo" irá imprimir o indice zero do meu array representado pela letra **"d"**
- O quinto "echo" irá imprimr o indice (valor +1)= (2 + 1) = 3 representando pela letra **"a"**
- Por fim podemos identificar a mensagem no PHP : **e a b d a.**

EXERCÍCIOS PROPOSTOS

O que é um array, e qual a sua principal finalidade?

Declare um array chamado "nomes" com 8 posições, e grave nomes de pessoas que você conhece em cada uma delas. Após criar o array responsa as questões 2, 4, 5, 10, 11:

Utilizado o array responda.

a) Qual nome é impresso no navegador se colocarmos o código: echo nomes[3];

b) Quais nomes aparecerá se adicionamos os seguintes códigos: for($i= 6; $i>1 ; i--)
echo nomes[$i];

c) O que acontece se chamarmos uma posição que não existe no array, exemplo: nomes[15];

O que é um array associativo, de exemplos:

Usando o comando *foreach*, crie uma interação onde todos os nomes possa ser impresso na tela.

Utilizando o mesmo código, altere alguns nomes do array.

O que é um array multidimensional?

Crie um array "palavras" multidimensional 5x3 com os valores da tabela abaixo, e responda as questões 7,8,9:

"oi"	"tudo"	"estar"
"você"	"vai"	"?"
"com"	"dia"	"!"
","	"bem"	"sim"
"casa"	"hoje"	"em"

Crie um código PHP onde com os valores do array possa ser impresso na tela com a frase "oi, tudo bem com você?".

Utilizando as posições da sequencia [1][0],[1][1],[0][2],[4][2],[4][0],[4][1], [1][2] do array palavras, qual frase podemos formular?utilize a função print() para mostrar na tela do navegador.

Construa um código PHP para mostra uma resposta para a pergunta da questão 7. (use o comando echo para imprimir na tela.

Utilizando a função sort, imprima em ordem alfabética os nomes do array "nomes".

Use o comando array_unshift() para adicionar mais dois nomes no array.