Vetores em PHP

- Vetores (ou arrays) são estruturas de dados que permitem armazenar mais de um valor na memória RAM, simultaneamente, no mesmo endereço. Uma mesma variável armazena mais de um valor ao mesmo tempo;
- Em PHP, não é necessário **declarar** o vetor antes de utilizá-lo, nem mesmo **inicializar** este vetor;
- Em PHP, não é necessário definir o **tamanho** de um vetor antes de usá-lo;
- Em PHP, após ter sido criado, o tamanho de um vetor pode mudar dinamicamente, aumentando ou diminuindo conforme necessário;
- Podemos endereçar o conteúdo de um vetor de duas formas: **com índice numérico e com índice associativo;**
- Vetores com chave numérica iniciam no índice zero;
- Qualquer tipo de dado pode ser armazenado em um vetor;
- Podemos misturar, em um mesmo vetor, índices numéricos e associativos;
- Um índice pode ser qualquer valor **inteiro (positivo ou negativo)** ou **string**.

Formas de criação de um vetor

- Em PHP, podemos usar colchetes ou o construtor array() para a criação de vetores:

```
$meuVetor = []; //cria um vetor vazio, sem nenhum elemento. Seu valor agora é null
$meuVetor = array(); //cria um vetor vazio, sem nenhum elemento. Seu valor é null
$diasDaSemana = ['domingo', 'segunda', 'terça'];
$diasDaSemana = array('domingo', 'segunda', 'terça');
$diasDaSemana[0] = 'domingo';
$diasDaSemana[1] = 'segunda';
$diasDaSemana[2] = 'terça';
```

Indexação numérica

```
$estado[0] = "RS";
$estado[1] = "SC";
$estado[2] = "PR";
$estado[3] = "SP";
```

- Não há a necessidade de se seguir uma sequência numérica dos índices. Poderíamos ter:

```
$estado[0] = "RS";
$estado[1] = "SC";
$estado[3] = "PR";
$estado[7] = "SP";
```

- Vetores deste tipo são conhecidos como "**vetores esparsos**" e o PHP não alocará memória para as posições com índices 2, 4, 5 e 6, já que elas não existem.

Indexação associativa

```
$estado["rs"] = "Rio Grande do Sul";
$estado["sc"] = "Santa Catarina";
$estado["pr"] = "Paraná";
```

- Ou, de forma equivalente:

```
$estado = ["rs"   => "Rio Grande do Sul",
           "sc"   => "Santa Catarina",
           "pr"   => "Paraná"];
```

Indexação automática

- Podemos usar colchetes vazios para indicar ao PHP a continuidade do vetor. Neste caso, o PHP adiciona o conteúdo após a última posição ocupada, usando o próximo índice numérico disponível. Exemplo:

```
$estado[] = "Rio Grande do Sul";
$estado[] = "Santa Catarina";
$estado[] = "Paraná";
$estado[] = "São Paulo";
```

O construtor array()

- Vetores em PHP podem ser criados de outra forma: com o uso do construtor **array()**. Exemplo:

```
$estados = array("Rio Grande do Sul", "Santa Catarina", "Paraná", "São Paulo");
```

Aqui, o índice é numérico e começa em 0.

```
$estados = array("rs" => "Rio Grande do Sul",
                 "sc" => "Santa Catarina",
                 "pr" => "Paraná");
```

Aqui, temos um vetor com índice associativo

Percorrendo elementos de um vetor

De duas maneiras:

- ✓com os laços comuns de repetição - while, for, do..while, etc...
- ✓com o laço foreach.

Usando for

Suponha que conheçamos, antecipadamente, o tamanho do vetor que, neste caso, tem três posições de memória. Com o uso de for, ficaria assim:

```
for($i = 0; $i < 3; $i++)

    echo "$estado[$i] <br>";
```

Usando foreach

- Sintaxe geral do laço foreach:

```
foreach($vetor as $indice => $valor)

{

echo "Índice indicando a posição do elemento: $indice <br>";

echo "Valor do elemento: $valor";

}
```

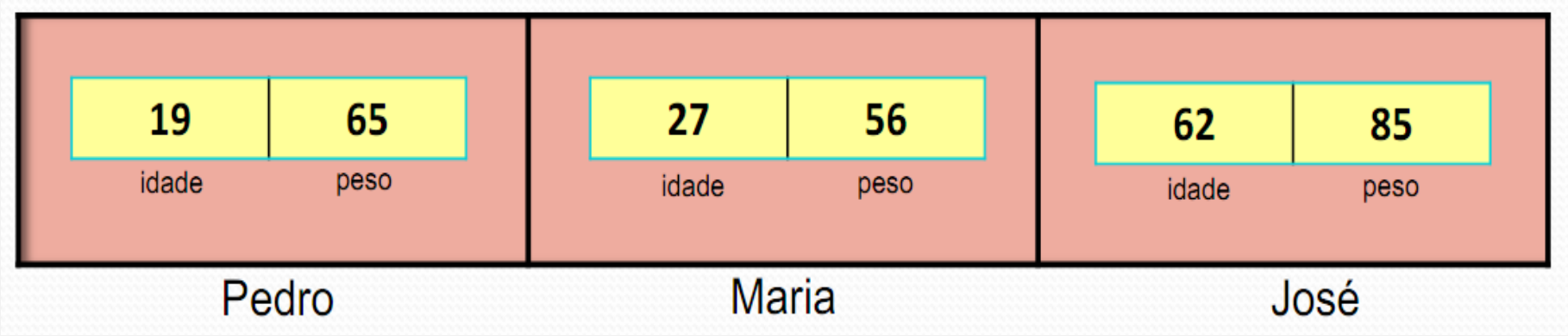
- O laço percorre cada elemento do vetor, até encontrar o seu final. A cada repetição, o índice é colocado na variável **$indice** e o conteúdo daquela posição do vetor fica armazenado na variável **$valor**. Notar que **$indice** e **$valor** podem ser quaisquer nomes válidos de variáveis para o PHP.

Matrizes em PHP

- De forma bastante simples, PHP implementa matrizes (também chamadas de arrays multidimensionais) armazenando um vetor dentro de cada posição de memória de outro vetor. Em outras palavras, uma matriz é um vetor dentro de outro vetor. Considere a tabela abaixo, representando uma matriz bidimensional (dois índices):

Nome	Idade	Peso
Pedro	19	65
Maria	27	56
José	62	85

- Uma representação gráfica para a matriz criada pelo PHP para armazenar os dados acima poderia ser como o que se segue:



Criação de matrizes

- A matriz do exemplo anterior poderia ser criada em PHP de acordo com os comando abaixo, usando colchetes:

```
$biometria["Pedro"]["idade"] = 19;
$biometria["Pedro"]["peso"]  = 65;
$biometria["Maria"]["idade"] = 27;
$biometria["Maria"]["peso"]  = 56;
$biometria["José"]["idade"]  = 62;
$biometria["José"]["peso"]   = 85;
```

- Ou, com colchetes em uma única linha:

```
$biometria = ["Pedro" => ["idade" => 19, "peso"  => 65],
              "Maria" => ["idade" => 27, "peso" => 56],
              "José"  => ["idade" => 62, "peso" => 85]];
```

- Ou, com o construtor array():

```
$biometria = array("Pedro" => array("idade" => 19, "peso"  => 65),
                   "Maria" => array("idade" => 27, "peso"  => 56),
                   "José"  => array("idade" => 62, "peso"  => 85));
```

Mostrar valores do vetor com echo;

- **CUIDADO**: usar um vetor com índice associativo dentro de echo para mostrar o conteúdo daquela posição não produzirá o efeito desejado. O navegador mostrará a palavra Array(). Devemos deixar o vetor fora das aspas. Este problema não ocorre para vetores com índice numérico.

- echo *"$vetor["nome"]";* ou echo *"$vetor['nome']";* não funcionará
- echo "O nome da pessoa é", $vetor["nome"]; funcionará
- echo "O nome da pessoa é " . $vetor["nome"]; funcionará
- echo "o nome da pessoa é {$vetor["nome"]}"; funcionará

Conferindo elementos do vetor com print_r()

- Quando queremos apenas visualizar os elementos de um vetor para, por exemplo, conferir se estão armazenados corretamente, podemos escrevê-los de forma sucinta na página web, por meio da função

***print_r*($vetor);**

- Os elementos do vetor são mostrados na página web sem formatação, em uma única linha, dificultando a visualização;

- Para aplicar formatação, faça:

```
echo "<pre>";
print_r($vetor);
echo "</pre>";
```

Vetores em PHP – funções especiais

- Determinar o tamanho de um vetor:

```php
$tamanho = count($vetor);
```

Localizar determinado elemento em um vetor

```
$existe = in_array($valor, $vetor);
```

- A função acima retorna true se **$valor** foi encontrado dentro do vetor. False, caso contrário.

Testar se uma chave existe

```php
$existe = array_key_exists($indice, $vetor);
```

- A função acima retorna true se o índice associativo $indice estiver presente no vetor. False, caso contrário.

Pesquisar um valor, retornando seu índice

A função abaixo pesquisa um determinado valor no vetor e:

- ✓ retorna o índice associado àquele valor, se este valor for encontrado no vetor;
- ✓ retorna false, se o conteúdo pesquisado não está no vetor.

```
$chave = array_search($valor, $vetor);
```

Criar vetor a partir de uma string

```
explode(separador, $string);
```

- Exemplo: se tivermos uma variável string armazenando os valores, digamos, $nomes = "Pedro – Maria – José ";, após invocarmos a função

```
$vetorNomes = explode("-", $nomes);
```

- Teremos criado, automaticamente, um vetor com índice numérico chamado $vetorNomes, onde cada posição armazenará as strings Pedro, Maria e José. O caractere "-" foi o separador usado para o PHP conseguir formar cada item do vetor a partir da string original.

Criar uma string a partir de um vetor

```
implode(delimitador, $vetor);
```

- Esta função faz o inverso da função explode(). A partir de um vetor, e fornecendo-se um caractere separador, a função cria a variável string correspondente, separando cada elemento do vetor com o delimitador. Exemplo:

```
$vetorNomes = array("Pedro", "Maria", "José");

$nomes = implode(", ", $vetorNomes);
```

- Agora, dentro da string $nomes, teremos o conteúdo "Pedro, Maria, José".

Excluir um elemento do vetor ou excluir o vetor inteiro

```
unset($vetor[$indice]); //não altera os índices das demais posições do vetor – exclui um único elemento
```

```
unset($vetor); //exclui o vetor inteiro da memória
```

Somando os elementos de um vetor

```
array_sum($vetor);
```

- Se tivermos o vetor abaixo

```
$vetor = ["Pedro" => 50, "Maria" => 12.7];
```

- E aplicarmos a função

```
$soma = array_sum($vetor);
```

- Teremos, agora, na variável $soma, a soma de todos os elementos do vetor, isto é, $soma conterá 62,7.

Recuperando o valor numérico máximo e mínimo

- A função ***max()*** permite que o PHP devolva o valor numérico máximo de um conjunto de valores. Os parâmetros da função podem ser dois ou mais valores numéricos, ou então, um vetor. A função ***min()*** age da mesma forma, porém devolve o menor valor numérico do conjunto ou do vetor.

```
$maior = max($vetor);

$menor = min($vetor);

$maior = max($valor1, $valor2, $valor3, ...);

$menor = min($valor1, $valor2, $valor3, ...);
```

Invertendo os elementos de vetor

```
array_reverse($vetor);
```

- Se tivermos o vetor abaixo

```
$vetor = array("Pedro", "Maria", "José");
```

- E aplicarmos a função

```
$invertido = array_reverse($vetor);
```

- Teremos, agora, um novo vetor com os valores do original invertidos, isto é, o vetor **`$invertido`** agora contém, nesta ordem, os elementos *José, Maria e Pedro*.

Ordenando vetores

- **`sort($array)`** – ordena um vetor em ordem crescente de seus valores. Não leva os índices junto ao fazer as trocas – cria novos índices numéricos.

- **`rsort($array)`** – ordena um vetor em ordem decrescente de seus valores. Não mantém associação entre chave e valor – cria novos índices numéricos.

- **`asort($array)`** – ordena o conteúdo das células de um vetor mantendo a associação entre chave e valor.

- **`arsort($array)`** – ordena o conteúdo das células de um vetor em ordem decrescente, mantendo a associação entre chave e valor.

- **`ksort($array)`** – ordena os índices de um vetor em ordem crescente de suas chaves, mantendo a associação entre índices e valores.

- **`krsort($array)`** – ordena os índices de um vetor em ordem decrescente dos valores de suas chaves, preservando a associação entre valor e chave.

- Todas as funções ou usam ordenação numérica ou alfabética;

- Funções que têm a letra **k** atuam sobre o índice (key);

- Funções que têm a letra **r** fazem ordenação na ordem decrescente (reverse);

- Internamente, usam o algoritmo QuickSort.