



INSTITUTO FEDERAL
Sul-rio-grandense

Câmpus
Charqueadas

Desenvolvimento Back-end

Estruturas de Repetição e Arrays

Prof. Sérgio Yoshimitsu Fujii
sergiofujii@ifsul.edu.br

A estrutura WHILE

Para que serve?

Esse operador é utilizado para executar um bloco de código várias vezes, enquanto uma determinada condição for atendida. Traduzindo, esta estrutura funciona da seguinte forma:

ENQUANTO (condição for atendida) **FAÇA ALGO.**

Onde FAÇA ALGO pode ser um ou vários comandos PHP.

A estrutura WHILE

Sintaxe básica

ENQUANTO (condição for atendida) **FAÇA ALGO.**

```
while (condição) {  
  
    // comandos  
  
}
```

```
<?php  
  
    $contador = 0;  
    while($contador < 10) {  
        echo $contador;  
        $contador++;  
    }  
  
?>
```

A estrutura DO ... WHILE

Para que serve?

A estrutura **DO...WHILE** é quase idêntica que a **WHILE** com a diferença que o teste da variável é feita no final da estrutura , ou seja, o código é executado pelo menos uma vez:

FAÇA ALGO ... ENQUANTO (condição for atendida).

Onde FAÇA ALGO pode ser um ou vários comandos PHP.

A estrutura DO ... WHILE

Sintaxe básica

FAÇA ALGO ... ENQUANTO (condição for atendida).

```
do {  
  
    // comandos  
  
} while (condição);
```

```
<?php  
  
$contador = 0;  
do {  
    echo $contador;  
    $contador++;  
} while($contador < 10);  
  
?>
```

A estrutura FOR

Para que serve?

A estrutura de repetição **FOR** é utilizada para se executar um conjunto de comandos por um número definido de vezes. Para esse operador, são passados uma situação inicial, uma condição e uma ação a ser executada a cada repetição.

Em geral informamos uma variável que serve como contador de repetições, com seu valor inicial, uma condição a ser atendida para que cada repetição seja executada e um incremento ao contador.

A estrutura FOR

Sintaxe básica

PARA (valor inicial; teste; incremento) { comandos }.

```
for (valor; teste; incremento) {  
    // comandos  
}
```

```
<?php  
  
$contador = 0;  
for ($i=0; $i<10; $i++) {  
    echo $contador;  
    $contador++;  
}  
  
?>
```

A estrutura FOREACH

Para que serve?

O **FOREACH** é uma simplificação do operador **FOR** para se trabalhar em coleções de dados, ou seja, vetores e matrizes. Ele permite acessar cada elemento individualmente iterando sobre toda a coleção e sem a necessidade de informação de índices.

A sintaxe do FOREACH é:

```
foreach (vetor as variável) {  
    // variável representa um elemento do vetor a cada iteração  
}
```


A estrutura FOREACH

Exemplos

```
foreach (vetor as variável) {  
    // variável representa um elemento do vetor a cada iteração  
}
```

```
<?php  
  
$vetor = array(1, 2, 3, 4, 5);  
for($i = 0; $i < 5; $i++) {  
    echo = $vetor[$i];  
}  
  
?>
```

Utilizando **FOR**

```
<?php  
  
$vetor = array(1, 2, 3, 4, 5);  
foreach($vetor as $item) {  
    echo = $item;  
}  
  
?>
```

Utilizando **FOREACH**

A estrutura FOREACH

Variação

Utilizando **FOR**

Índice: **\$i**
Valor: **\$vetor[\$i]**

```
<?php
// NECESSÁRIO saber o tamanho do vetor
$vetor = array(1, 2, 3, 4, 5);
for($i = 0; $i < 5; $i++) {
    echo "Posição: ". $i. " Valor: ". $vetor[$i];
}

?>
```

Utilizando **FOREACH**

Índice: **\$índice**
Valor: **\$item**

```
<?php
// NÃO é necessário saber o tamanho do vetor
$vetor = array(1, 2, 3, 4, 5);
foreach($vetor as $índice => $item) {
    echo "Posição: ". $índice. " Valor: ". $item;
}

?>
```

Break e Continue

O comando break

O comando **break** é usado em laços de repetição **while**, **do...while**, **for** e com o comando **switch/case**. Quando usado em laço de repetição, causa uma interrupção imediata do mesmo, continuando a execução do programa na próxima linha após o laço.

Isso ocorre caso uma condição imposta seja atendida.

Break e Continue

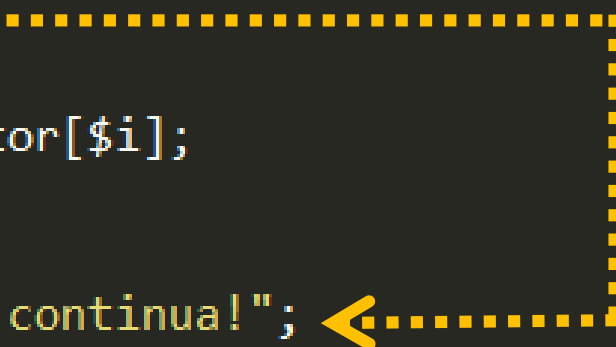
O comando break

```
<?php

    $vetor = array(1, 2, 3, 4, 5);
    for($i = 0; $i < 5; $i++) {
        if ($i == 3) {
            break;
        }
        echo $vetor[$i];
    }

    echo "O código continua!";

?>
```



Break e Continue

O comando continue

O comando **continue** é usado somente em laços de repetição. Quando ele é executado, o laço volta imediatamente para o teste de condição do laço de repetição. Normalmente, usamos o comando continue em um teste **if**.

Break e Continue


O comando continue

```
<?php

$vetor = array(1, 2, 3, 4, 5);
for($i = 0; $i < 5; $i++) {
    if ($i == 3) {
        continue;
    }
    echo $vetor[$i];
}

echo "O código continua!";

?>
```

A yellow dashed box with a yellow arrow pointing left from the 'continue;' statement to the closing brace of the 'for' loop, indicating that the rest of the loop body is skipped.

Arrays do PHP

Arrays

- Criado com a função [array](#)
- Armazenam diversos valores
- Acessamos posições com o `[]`

```
// declara array  
$nomes = array();
```

```
// atribui valores na declaração  
$nomes = array("João", "Carol", "Guilherme");
```

```
// acessa posições específicas do array  
echo $nomes[0];  
echo $nomes[1];  
echo $nomes[2];
```

```
// preenche mais informações no array, em  
posições específicas  
$nomes[3] = "Maria";  
$nomes[4] = "Bianca";  
$nomes[5] = "Pedro";
```

Arrays do PHP

Funções de arrays

[print_r](#) - imprime a estrutura do array.

```
print_r($nomes);
```

Array ([0] => João [1] => Carol [2] => Guilherme [3] => Maria [4] => Bianca [5] => Pedro)

[explode](#) - quebra uma string em um array.

```
$frase = "Esta é uma frase de  
teste";  
$palavras = explode(' ', $frase);  
print_r($palavras);
```

Array ([0] => Esta [1] => é [2] => uma [3] => frase [4] => de [5] => teste)

[count](#) - Retorna a quantidade de elementos de um array.

```
$palavras = array("lápis", "borracha",  
"caneta", "papel", "caderno");  
echo count($palavras); // 5
```

[implode](#) - Junta os elementos de um array.

```
$lista = array("morango", "abacaxi",  
"banana", "pêssego");  
$string = implode(',', $lista);  
echo $string;
```

morango,abacaxi,banana,pêssego

Mais funções de arrays, visite: https://www.w3schools.com/php/php_ref_array.asp

Arrays do PHP

Percorrendo arrays

Supondo o array:

```
$lista = array("lápiz", "borracha", "caneta", "papel", "caderno");
```

Podemos percorrer usando normalmente um for:

```
for ($i=0 ; $i < count($lista) ; $i++) {  
    echo "<div>$lista[$i]</div>";  
}
```

Ou podemos também usar o [foreach](#):

```
foreach ($lista as $item) {  
    echo "<div>$item</div>";  
}
```

Arrays do PHP

Arrays associativos

No PHP podemos nomear os índices dos arrays com strings

- Parecido com os objetos do JS

```
$pessoa = array(  
    "nome" => "João",  
    "idade" => 17  
);  
$pessoa["email"] = "joao.silva@gmail.com";
```

- A *flecha =>* é usada para separar o nome do campo do valor
- Após a declaração usamos o `[]` para acessar os campos do array associativo.

Arrays do PHP

Arrays associativos

Percorrendo arrays associativos usando **foreach**:

```
$pessoa = array();  
$pessoa["nome"] = "João";  
$pessoa["idade"] = 17;  
$pessoa["email"] = "joao.silva@gmail.com";  
foreach ($pessoa as $campo => $valor) {  
    echo "<div>$campo: $valor</div>";  
}
```

Variável que recebe
o **nome** do campo

Variável que recebe
o **valor** do array

Arrays do PHP

Arrays associativos

Acessando um cadastro com um array multidimensional:

```
$cadastro = array(  
    array(  
        "nome" => "João",  
        "idade" => 25,  
    ),  
    array(  
        "nome" => "Pedro",  
        "idade" => 29,  
    ),  
    ... // outros elementos  
);
```

```
foreach ($cadastro as $item) {  
    $row = "<div class='row'>";  
    foreach ($item as $campo => $valor) {  
        $row .= "<div class='col'>  
            <span>$campo</span>  
            <span class='value'>$valor</span>  
        </div>";  
    };  
    $row .= "</div>";  
    echo $row;  
}
```

nome	João	idade	25
nome	Pedro	idade	29
nome	Maria	idade	32
nome	Guilherme	idade	65
nome	Rafaela	idade	14

Arrays do PHP

Exercício:

- Use somente o PHP
- Crie um array com 5 elementos.
 - Cada elemento representará informações de uma pessoa.
- Cada elemento conterá um outro array.
 - Este array deverá conter os dados **nome**, **idade**, **estado civil**, **email** e **salário** da pessoa.
- Usando o php, percorra este array, e monte uma tabela HTML com as informações das pessoas.

Nome	Idade	E-mail	E.Civil	Salário
João Silva	25	joao.silva@gmail.com	Solteiro	1950
Rafael Cardoso	32	rafacardoso@gmail.com	Casado	5541
Gabriela Schidt	21	gabischidt@gmail.com	Solteira	3214
Roberta Oliveira	38	roberta.oliveira@gmail.com	Divorciada	4258
Pedro Santos	17	pebolado@gmail.com	Solteiro	2100