# SI401

## Programação para a Web

2º Semestre - 2024

# Programação *Back-End*: PHP
# Unidade 14

Prof. Guilherme Palermo Coelho
guilherme@ft.unicamp.br

# Roteiro

- Operadores;
- Estruturas de Controle;
- Funções;
- Referências.

# OPERADORES

# Operadores

- **Operadores Aritméticos:**

| Operator | Name | Description | Example | Result |
|---|---|---|---|---|
| x + y | Addition | Sum of x and y | 2 + 2 | 4 |
| x - y | Subtraction | Difference of x and y | 5 - 2 | 3 |
| x * y | Multiplication | Product of x and y | 5 * 2 | 10 |
| x / y | Division | Quotient of x and y | 15 / 5 | 3 |
| x % y | Modulus | Remainder of x divided by y | 5 % 2<br>10 % 8<br>10 % 2 | 1<br>2<br>0 |
| - x | Negation | Opposite of x | - 2 | |
| a . b | Concatenation | Concatenate two strings | "Hi" . "Ha" | HiHa |

- Exponencial (introduzido em PHP 5.6):
  - $x ** $y  // equivale a $x^y$

# Operadores

- **Operadores de Atribuição:**

| Assignment | Same as... | Description |
|---|---|---|
| x = y | x = y | The left operand gets set to the value of the expression on the right |
| x += y | x = x + y | Addition |
| x -= y | x = x - y | Subtraction |
| x *= y | x = x * y | Multiplication |
| x /= y | x = x / y | Division |
| x %= y | x = x % y | Modulus |
| a .= b | a = a . b | Concatenate two strings |

- **Operadores de Incremento e Decremento:**

| Operator | Name | Description |
|---|---|---|
| ++ x | Pre-increment | Increments x by one, then returns x |
| x ++ | Post-increment | Returns x, then increments x by one |
| -- x | Pre-decrement | Decrements x by one, then returns x |
| x -- | Post-decrement | Returns x, then decrements x by one |

# Operadores

- **Operadores de Comparação:**

| Operator | Name | Description | Example |
|---|---|---|---|
| x == y | Equal | True if x is equal to y | 5==8 returns false |
| x === y | Identical | True if x is equal to y, and they are of same type | 5==="5" returns false |
| x != y | Not equal | True if x is not equal to y | 5!=8 returns true |
| x <> y | Not equal | True if x is not equal to y | 5<>8 returns true |
| x !== y | Not identical | True if x is not equal to y, or they are not of same type | 5!=="5" returns true |
| x > y | Greater than | True if x is greater than y | 5>8 returns false |
| x < y | Less than | True if x is less than y | 5<8 returns true |
| x >= y | Greater than or equal to | True if x is greater than or equal to y | 5>=8 returns false |
| x <= y | Less than or equal to | True if x is less than or equal to y | 5<=8 returns true |

- Há também um ***operador ternário*** semelhante ao de outras linguagens

# Operadores

- **Operadores de Comparação (PHP 7.x):**
  - Operador ***spaceship*** (<=>):
    - **$a <=> $b**:
      - Retorna 0: **$a** e **$b** são iguais;
      - Retorna inteiro positivo: se **$a** > **$b**
      - Retorna um inteiro negativo: se **$a** < **$b**
  - Operador ***null coalescing*** (??):
    - Deve ser usado quando busca-se verificar se algo existe e, caso não exista, retornar um valor padrão:
    - **$username** = **$_GET[**‘user’**]** ?? ‘ninguem’;
      - Se não houver a entrada ‘user’ no vetor **$_GET**, é atribuída a *string* ‘ninguem’ a **$username**.

# Operadores

- **Operadores de *Arrays*:**

| Operator | Name | Description |
|---|---|---|
| x + y | Union | Union of x and y |
| x == y | Equality | True if x and y have the same key/value pairs |
| x === y | Identity | True if x and y have the same key/value pairs in the same order and of the same types |
| x != y | Inequality | True if x is not equal to y |
| x <> y | Inequality | True if x is not equal to y |
| x !== y | Non-identity | True if x is not identical to y |

- O operador de **união** (+) adiciona os elementos do *array* da direita no *array* da esquerda, **sem sobrescrever os elementos de mesmo índice** (NÃO é append);

```
$a = array("a" => "maçã", "b" => "banana");
$b = array("a" =>"pêra", "b" => "framboesa", "c" => "morango");
$c = $a + $b; // Uniao de $a e $b

//c["a"] == "maçã"; c["b"] == "banana"; c["c"] == "morango"
```

# Estruturas de Controle

# *If* e *switch*

```
// Forma 1:
if (expr 1)
{
    bloco 1;
}
elseif (expr 2)
{
    bloco 2;
}
else
{
    bloco 3;
}
```

```
switch (operador)
{
    case valor1:
        <comandos>
        break;
    case valor2:
        <comandos>
        break;
        ....
    case valorN:
        <comandos>
        break;
    default:
        <comandos>
        break;
}
```

Não é obrigatório

# For

```
// Forma 1:
for (inicialização; condição; operador)
{
    <comandos>
}
```

# *Foreach*

- É usado para percorrer todos os elementos de um *array*;
- Foi implementado a partir da versão 4 de PHP;
- Existem duas formas de utilização do comando ***foreach***:

```
// Forma 1:
foreach ($nome_array as $valor)
{
    <comandos>
}
```

  - Todos os elementos do *array* são percorridos (do primeiro ao último);
  - A cada iteração, o valor do ***elemento corrente*** do *array* é atribuído à variável **$valor**;

# *Foreach*

- É usado para percorrer todos os elementos de um *array*;
- Foi implementado a partir da versão 4 de PHP;
- Existem duas formas de utilização do comando ***foreach***:

```
// Forma 2:
foreach ($nome_array as $chave => $valor)
{
    <comandos>
}
```

  - Faz a mesma coisa da **Forma 1**, com a diferença que a ***chave*** (ou *índice*) do elemento atual também é atribuída à variável **$chave**;
    - Útil principalmente quando se tem *arrays associativos*.

# Foreach

```
// Exemplo:
$vetor = array(5, 6, 7, 8);
foreach ($vetor as $v)
{
        echo "O valor atual é $v <br />”;
}

foreach($vetor as $chave => $v)
{
        echo "\$vetor[$chave] => $v <br />";
}
```

```
O valor atual é 5 <br />
O valor atual é 6 <br />
O valor atual é 7 <br />
O valor atual é 8 <br />
$vetor[0] => 5 <br />
$vetor[1] => 6 <br />
$vetor[2] => 7 <br />
$vetor[3] => 8 <br />
```

# While

```
// Forma 1:
while (expr)
{
    <comandos>
}
```

# Do ... While

```
do
{
    <comandos>
} while (expr)
```

# *Break* e *Continue*

- O comando ***break*** encerra a execução do comando atual (***if***, ***for***, ***while*** ou ***switch***);
  - A execução passa para o primeiro comando após o término da estrutura que estava em execução;
  - ***break*** pode receber um **parâmetro numérico** (opcional) que indica o número de estruturas (aninhadas) em execução que devem ser interrompidas:
    - **Ex.**: "***break*** 2;" pode interromper dois loops aninhados.
- O comando ***continue*** permite ***ignorar*** as instruções restantes no laço corrente, e ir para o início da próxima iteração ;
  - Também aceita um parâmetro numérico opcional, que indica o número de níveis que devem ser reiniciados.

# FUNÇÕES

# Funções

- Funções em PHP permitem:
  - O agrupamento de trechos de códigos que realizam funções específicas (como em outras linguagens de programação);
  - O encapsulamento de trechos de códigos que ***não devem*** ser executados quando a página é carregada;
- PHP disponibiliza uma grande quantidade de funções pré-definidas, que permitem realizar as mais diferentes tarefas:

http://www.php.net/manual/pt_BR/funcref.php

# Funções

- Para criar suas próprias funções:

```
function nome_funcao(arg1, arg2, arg3, ..., argN)
{
    <comandos>
    [return <expressao>]
}
```

  - O ***retorno*** e os ***parâmetros*** são opcionais;
  - É possível retornar um *array*.

# Funções: Passagem de Parâmetros

- Quando se passa uma determinada variável para uma função, por padrão esta passagem se dá ***por valor***;
  - Alterações internas na variável não se refletem externamente:

```
function incrementa($x)
{
     $x = $x + 1;
}

$x = 1;
incrementa($x);
echo $x; // Será impresso o valor 1
```

# Funções: Passagem de Parâmetros

- PHP também permite a passagem de variáveis por ***referência***;
  - Alterações nestas variáveis ***dentro*** de uma função se refletem fora dela;
- Para isso, deve-se colocar o símbolo "**&**" ***antes*** do nome da variável no cabeçalho da função.

```
function incrementa(&$x)
{
     $x = $x + 1;
}

$x = 1;
incrementa($x);
echo $x; // Será impresso o valor 2
```

# Funções: valor padrão para parâmetros

- PHP também permite que sejam definidos ***valores-padrão*** que os parâmetros de uma função devem ter:
  - Estes valores padrão são utilizados quando o parâmetro ***não é passado*** para a função no instante de sua chamada.
  - Para definir estes valores padrão, basta colocar um ***operador de atribuição*** após o parâmetro no cabeçalho da função, seguido pelo ***valor que deve ser considerado padrão***.

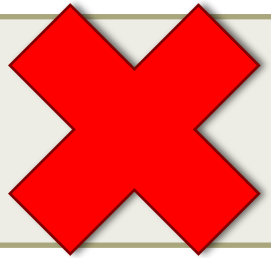
```
function teste($time, $titulo = "Campeão Mundial")
{
     echo "O $time é $titulo <br />";
}

teste("Internacional", "Campeão Gaúcho");
teste("Santos", "Campeão Paulista");
teste("Barcelona");
```

```
O Internacional é Campeão Gaúcho <br />
O Santos é Campeão Paulista <br />
O Barcelona é Campeão Mundial <br />
```

# Funções: valor padrão para parâmetros

- **Atenção**: os parâmetros que terão valores-padrão definidos ***devem ser os últimos*** parâmetros no cabeçalho da função.

```
function errada($a = 10, $b, $c)
{
     ...
}
```



# Exercício Sugerido 1

# Exercício Sugerido 1

- A instrução abaixo cria uma vetor, em PHP, com as capitais de alguns países europeus. Pesquise sobre ordenação de *arrays* em PHP e crie um script que exiba os nomes da capital e do país. Esta exibição deve ser feita em ordem alfabética por país.
  - Ex. de saída: “A capital da Alemanha é Berlim”.

```
$ceu = array("Italia"=>"Roma", "Luxemburgo"=>"Luxemburgo", "Belgica"=> "Bruxelas",
"Dinamarca"=>"Copenhagen", "Finlandia"=>"Helsinki", "França" => "Paris",
"Eslovaquia"=>"Bratislava", "Eslovenia"=>"Ljubljana", "Alemanha" => "Berlim", "Grecia" =>
"Atenas", "Irlanda"=>"Dublin", "Holanda"=>"Amsterdam", "Portugal"=>"Lisboa",
"Espanha"=>"Madrid", "Suecia"=>"Stockholm", "Reino Unido"=>"Londres", "Chipre"=>"Nicosia",
"Lituania"=>"Vilnius", "Republica Tcheca"=>"Praga", "Estonia"=>"Tallin", "Hungria"=>"Budapest",
"Latvia"=>"Riga", "Malta"=>"Valetta", "Austria" => "Vienna", "Polônia"=>"Varsovia");
```

# Referências

# Referências

[1] Niederauer, J. “Desenvolvendo Websites com PHP”, 2ª ed. Novatec, 2011.

[2] W3Schools PHP Tutorial:

http://w3schools.com/php/default.asp

[3] Manual do PHP:

http://php.net/manual/pt_BR/index.php

[4] Hudson, P. “Hacking with PHP”:

http://www.hackingwithphp.com/

[5] Prettyman, S. "Learn PHP 7“, Apress, 2016:

https://link.springer.com/book/10.1007/978-1-4842-1730-6