Programa de desenvolvedor Full-Stack

Conceitos Básicos

Case sensitivity

- Significa sensibilidade a casas maiúsculas e minúsculas;
- Para **instruções PHP não temos essa diferença**, ou seja: echo = ECHO;
- Porém para **variáveis são case sensitive**;
- Ou seja, $nome != $NOME;
- Obs: veremos variáveis em detalhes mais adiante;

Instruções de código

- As instruções simples de PHP são **separadas por ponto e vírgula**;
- Instruções simples são instruções de uma linha;
- Quando há uma **instrução maior**, como de condição ou repetição, a definição da mesma é dada por **abertura e fechamento de chaves**;
- Nestes casos **não precisaremos** de ponto e vírgula;

Espaços em branco

- Para interpretação do código em PHP **o espaço em branco é ignorado**;
- Isso acontece pois o mesmo **é removido antes da execução**;
- A **quebra de linha também também é ignorada**;
- Porém se utilizada de má forma pode gerar erros inesperados no código;

Comentários

- Os comentários servem para dar **informações e direções importantes** de como o código funciona;
- Iniciamos um comentário com **//**
- Todo conteúdo que está em um **comentário é ignorado** na execução;
- **Não insira informações sensíveis** nos comentários;
- Outra forma de inserir comentários é com **#**
- Comentários multi linhas são feitos com: **/* comentário */**

Exercício 2

- Crie um arquivo PHP;
- Neste arquivo descreva características básicas de uma pessoa ou objeto utilizando comentários;
- Pelo menos três, em linhas separadas;

Palavras reservadas

- Algumas palavras são **reservadas da linguagem** e já tem suas funcionalidades definidas, então não podemos utilizar em nossos programas;
- Pois caso fosse possível **poderíamos substituir** a sua função original;
- **Alguns exemplos são:** echo, insteadof, else, interface, namespace, pow,_____DIR___, _____FILE___, endif, print, private, protected, and, require, public, as, break, case, for, finally, switch, throw e etc;

Tipos de dados

Inteiros (integers)

- Os inteiros são os **números inteiros** da matemática, como: 1, 2, 15;
- Incluindo os **números negativos**;
- Os números positivos **não precisam** de um sinal de + na frente;
- Já os números negativos devem ser descritos assim, ex: -12;

Exercício 3

- Crie um arquivo PHP;
- Imprima três números inteiros;

Exercício 3

2_exercicio_3 > index.php

```php
<?php

    echo 5;
    echo "<br>";
    echo 50;
    echo "<br>";
    echo 5000;

?>
```

Números decimais (floats)

- Os floats são todos os números com **casas decimais**;
- Como o padrão universal é da língua inglesa, temos a separação de casas **com . e não ,**
- Exemplos de floats: 2.123, 0.04, -12.8

Exercício 4

- Crie um arquivo PHP;
- Imprima três floats;
- Utilize a função is_float em um deles;

Exercício 4

```php
<?php

echo 4.15;
echo "<br>";
echo 12.12;
echo "<br>";

$c = -78.1;

echo $c;
echo "<br>";
```

Textos (strings)

- Os textos são conhecidos como **strings**;
- Em PHP podemos escrever textos em **aspas simples ou duplas**, não há diferença para texto puro;
- As aspas duplas **interpretam variáveis**;

Exercício 5

- Crie um arquivo PHP;
- Imprima textos com aspas duplas e também com aspas simples;

Exercício 5

```php
<?php

echo 'Testando aspas simples <br>';
echo "Teste de aspas duplas <br>";

$nome = "Matheus";
$idade = 29;

echo "Olá, eu sou o $nome e tenho $idade anos";
```

Booleanos

- O boolean é um tipo de dado que só possui **dois valores**:
- **True** - verdadeiro;
- **False** - falso;
- Alguns valores são considerados como falsos: 0, 0.0, “0”, [ ], NULL;

Arrays (conjunto, lista)

- O array é um tipo de dado que serve para **agrupar um conjunto de valores**;
- Podemos inserir **qualquer tipo de dado** na lista;
- A sintaxe é: [1, 2, 3, 4, 5];
- Sempre entre **[ ]**, dados separados por **,**
- Veremos arrays em mais detalhes futuramente, é uma estrutura de dados muito importante e muito utilizada;

Array Associativo

- O **array associativo** é basicamente um array, porém com **chave e valor**;
- A **estrutura base é a mesma**, mas vamos construir dessa maneira:
- $arr = [‘nome’ => ‘Matheus’, idade => 29]
- Chave entre aspas, seta para apontar o valor e valor;

Exercício 6

- Crie um arquivo PHP;
- Crie um array com características de um carro;
- Imprima duas características;

Exercício 6

```php
<?php

$carro = [
    'marca' => 'BMW',
    'rodas' => 4,
    'teto_solar'=> true,
    'velocidade_max' => 300,
    'blindado' => false
];

print_r($carro);

$marca = $carro['marca'];
$velocidade_maxima = $carro['velocidade_max'];

echo "<br>";

echo "O carro é da marca $marca e atinge no maximo $velocidade_maxima km/h";
```

Variáveis

Introdução

Sobre as variáveis

- É a forma que temos para **declarar um valor e salvá-lo na memória**;
- Uma variável em PHP tem o **$** na frente do seu nome;
- Ex: $nome = “Rafael”;
- Podemos salvar **qualquer tipo de dado**;
- Podemos **alterar o valor de uma variável** no decorrer do programa;
- Podemos imprimir o valor de uma variável com **echo**;

Exercício 8

- Crie um arquivo PHP;
- Crie três variáveis com tipos de dados diferentes;
- Imprima estas variáveis;

Exercício 8

```php
<?php

$velocidade = 100; // int
$marca = "Ferrari"; // string
$itens = ["Teto solar", "Motor 2.0", "Porta malas grande", "Piloto automático"]; // array

echo $velocidade;
echo "<br>";
echo $marca;
echo "<br>";
print_r($itens);
```

Exercício 9

- Crie um arquivo PHP;
- Crie duas variáveis com números;
- Cria uma terceira com a soma destes dois números;
- Lembrando: a soma pode ser feita com o símbolo +;
- Ex: 2 + 4

Exercício 9

```php
<?php

$a = 11; // int
$b = 99.324; // float

$c = $a + $b;

echo $c;
```

Escopo de uma Variável

- Como em outras linguagens em PHP também temos escopo de variáveis;
- **Local:** variável declarada em uma função;
- **Global:** variáveis declaradas fora de funções;
- **Static:** variável declarada dentro da função, porém o seu valor permanece salvo entre chamadas da função;
- **Parâmetros de função:** variáveis passadas para uma função, podendo ser utilizadas ao longo da mesma;

Variável Local

- A variável local tem seu **escopo definido dentro de uma função**;
- Ela **não é acessível** fora da mesma;
- O **seu valor sempre é resetado** quando a função é finalizada;
- Obs: veremos funções em detalhes futuramente;

Variável Global

- A principal característica da variável global **é ser declarada fora de funções**;
- Por comportamento padrão **não são acessíveis dentro de funções**;
- Precisamos utilizar a palavra **global** para isso;
- Essa função da variável global não ser acessível dentro de funções, previne muitos problemas no software;

Variável Estática

- A variável estática é declarada com a instrução **static**;
- O valor da mesma **é mantido e alterado a cada execução de uma função**;
- É interessante este comportamento pois as variáveis de **escopo local sempre são resetadas**;

Parâmetros de função

- Os parâmetros de função **também são considerados tipos de variáveis**;
- Este recurso nos ajuda a **criar funções com valores dinâmicos**;
- Podendo **alterá-los a cada invocação** da mesma;
- Podemos passar mais de um parâmetro para uma função;

Expressões e Operadores

Introdução

O que é uma expressão?

- Uma **instrução de código** que será avaliada **e resultará em um valor**;
- Uma **simples impressão de um texto** é uma expressão;
- **Uma soma ou operação matemática mais complexa** também;
- Na programação realizaremos **diversas expressões** durante nosso código, para formar nosso software;

O que é um operador?

- Operadores são **recursos que utilizamos para compor expressões mais complexas**;
- Alguns deles: +, -, **, /, ++, >, <, >=, <= e etc…
- Estas operações podem ser matemáticas ou até mesmo comparações;
- A ideia principal é que um **novo valor é gerado** ou também um **boolean pode ser retornado**;

Ordem dos operadores

- O PHP e as linguagens de programação **executam os operadores na mesma ordem que na matemática**;
- Ou seja em: 2 + 2 * 4, teremos o resultado de **10**;
- Pois **a multiplicação é avaliada antes da soma**;
- Mesmo que a primeira operação seja soma;
- Podemos utilizar **( )** para separar operações;

Exercício 10

- Crie um arquivo PHP;
- Crie uma operação que utiliza subtração (-), divisão (/) e multiplicação
- Armazene todos os valores em variáveis;
- Imprima o resultado final na tela;

Exercício 10

```php
<?php

// -, /, *
$a = 10;
$x = 5;
$z = 9;

$operacao = ($x - $z) / $a * $z;

echo $operacao;
```

Mudança de tipo implícito

- O PHP em certas operações **muda o tipo de dado** de forma implícita;
- Por exemplo 5 / 2 = 2.5 (gera um **float**)
- E 5 . 5 resulta em 55 (gera uma **string**, o **.** é o operador de concatenação)
- Por isso, temos que **tomar cuidado** com algumas expressões que podem gerar resultados indesejados;
- Este recurso é chamado de **auto cast**;

Exercício 11

- Crie um arquivo PHP;
- Teste a expressão “5” * 12;
- Utilize a função **gettype()** com o resultado como parâmetro para checar o tipo resultante da operação;

Exercício

```php
<?php

$operacao = "5" * 12;

echo $operacao . "<br>";

echo gettype($operacao);
echo "<br>";
echo gettype([]);
echo "<br>";
echo gettype(12.2);
echo "<br>";
echo gettype("teste");
```

Operadores aritméticos

- Temos os **operadores básicos** da matemática em PHP;
- Soma: +
- Subtração: -
- Divisão: /
- Multiplicação: *

Exercício 12

- Crie um arquivo PHP;
- Crie uma operação com cada um dos operadores básicos;
- Cada operação deve estar em uma variável diferente;
- Imprima cada uma das etapas;
- Ex: soma -> multiplicação -> divisão -> subtração;

Exercício 12

```php
// +, -, /, *
$a = 3;
$b = 12;

$op1 = $a - $b;
echo $op1;
echo "<br>";

$c = 12.4;

$op2 = $op1 * $c;
echo $op2;
echo "<br>";

$d = 4.8;

$op3 = $op2 + $d;
echo $op3;
echo "<br>";

$e = 9.2;
```

Operador de módulo

- O operador de módulo é inserido no código pelo símbolo de **%**
- Sua função é realizar **uma divisão**;
- Mas como resultado ele **apresenta apenas o resto** da mesma;

Exercício 13

- Crie um arquivo PHP;
- Teste o operador de resto em duas divisões;
- Uma não exata e outra exata;

Exercício 13

```php
<?php

$a = 14;
$b = 2;
$c = 3;

echo $a % $b;
echo "<br>";
echo $a % $c;
echo "<br>";
```

Operador de concatenação

- Em PHP podemos concatenar valores com **.** (ponto)
- Concatenar é o ato de **juntar vários textos e/ou números** em apenas uma string;
- **Não há limites** de quantas expressões podem ser concatenadas;

Exercício 14

- Crie um arquivo PHP;
- Crie uma variável saudação, nome e outra de sobrenome;
- Imprima com echo a concatenação de saudação, nome e sobrenome;

Exercício 14

```php
<?php

$saudacao = "Sr.";
$nome = "Pedro";
$sobrenome = "Amaral";

$frase = "O " . $saudacao . " " . $nome . " " . $sobrenome . ", vem hoje para a reunião as 14h";

echo $frase;
```

Auto incremento e auto decremento

- Podemos incrementar um valor ou decrementar com os operadores: **++** e

  --;
- Exemplo: $n++ ou $x--
- Onde n e x são variáveis, e **terão seus valores alterados com +1 e -1**;
- Estes operadores são muito utilizados em **estruturas de repetição**;

Operadores de comparação

- As operações com operadores de comparação resultarão em true or false;
- Igualdade: **==**
- Idêntico a: **===**
- Diferença: **!=**
- Não idêntico a: **!==**
- Maior e maior ou igual a: **> e >=**
- Menor e menor ou igual a: **< e <=**

Operador de igualdade

- Com o **operador de igualdade** verificamos se um valor é igual ao outro;
- O símbolo é: **==**
- Exemplo: 5 == 4 # false
- Exemplo: 3 == 3 # true

Operador idêntico a

- Com o **operador idêntico a** verificamos se um valor é igual ao outro, avaliando o seu tipo também;
- O símbolo é: **===**
- Exemplo: 5 === 5 # true
- Exemplo: 3 === “3” # false

Operador de diferença

- Com o **operador de diferença** verificamos se um valor é diferente de outro;
- O símbolo é: **!=**
- Exemplo: 5 != 5 # false
- Exemplo: 10 != 5 # true

Operador maior e maior ou igual

- Com o **operador maior que** verificamos se um valor é maior que outro;
- O símbolo é: **>**
- Exemplo: 5 > 4 # true
- Com o **operador maior ou igual a** verificamos se um valor é maior ou igual a outro;
- O símbolo é: **>=**
- Exemplo: 5 >= 5 # true

Operador menor e menor ou igual

- Com o **operador menor que** verificamos se um valor é menor que outro;
- O símbolo é: **<**
- Exemplo: 5 < 4 # false
- Com o **operador menor ou igual a** verificamos se um valor é menor ou igual a outro;
- O símbolo é: **<=**
- Exemplo: 11 <= 12 # true

Operadores lógicos

- Com os operadores lógicos podemos **encadear várias comparações**;
- Operador AND: **&&**
- Operador OR: **||**
- Operador NOT: **!**

Tabela verdade

- Com a tabela verdade, temos um resumo dos operadores lógicos:

NOT

x	x'
0	1
1	0

AND

x	y	xy
0	0	0
0	1	0
1	0	0
1	1	1

OR

x	y	x+y
0	0	0
0	1	1
1	0	1
1	1	1

fonte: https://introcs.cs.princeton.edu/java/home/

Operador lógico AND

- Os operadores lógicos em conjunto dos de comparação **também retornam uma booleano** (true ou false);
- No caso de **AND** temos **true** apenas quando **as duas comparações são verdadeiras**;
- Símbolo: &&
- Ex: 5 > 2 && 10 < 100 # true

Exercício 17

- Verifique as seguintes operações com AND;
- 15 > 5 AND “João” === “João”
- 10 > 5 AND 1
- 2 == 3 AND 5 >= 3

Exercício 17

```php
<?php

// comparacao 1
if(15 > 5 && "João" === "João") {
    echo "A comparação 1 é verdadeira <br>";
}

// comparacao teste
if(10 > 5 && 1) {
    echo "A comparação teste é verdadeira <br>";
}
```

Operador lógico OR

- O operador lógico OR resulta em **verdadeiro** caso **qualquer um dos lados da operação seja verdadeiro**;
- E só resulta em **falso** caso os **dois lados sejam falsos**;
- Símbolo: ||
- Exemplo: 5 > 15 || “teste” == “teste” # true

Exercício 18

- Verifique as seguintes operações com OR;
- 12 < 5 OR “João” === “João”
- 1 > 5 OR 1
- 20 === “20” AND 51 >= 31

Exercício 18

```php
<?php

// comparação 1
if(12 < 5 || "João" === "João") {
  echo "A operação 1 é verdadeira <br>";
}

// comparação 2
if(1 > 5 || 1) {
  echo "A operação 2 é verdadeira <br>";
}

// comparação 3
if(20 === "20" && 51 >= 31) {
  echo "A operação 3 é verdadeira <br>";
}
```

Operador lógico NOT

- O operador lógico **NOT** apenas **inverte o resultado booleano** de uma operação, se é true vira false e se é false vira true;
- Símbolo: **!**
- Exemplo: !true # false
- Exemplo: !(5 > 2) # false

Operador lógico NOT

```php
<?php

if(!(5 > 2)) { // true > false
  echo "A operação 1 é verdadeira <br>";
}

if(!(5 > 20)) { // false > true
  echo "A operação 2 é verdadeira <br>";
}

$a = 10;
$b = 20;

if(!($a >= $b)) {
  echo "A operação 3 é verdadeira <br>";
}
```

Operadores de conversão (cast)

- Com os **operadores de conversão** podemos **forçar uma variável ser de um determinado tipo**;
- **Nem todos são úteis**, os mais utilizados são para converter uma string em número;
- Operadores: int, bool, float, string, array, object e unset;
- Exemplo: $a = (float) “5.34243” # string é convertida para float

Operadores de atribuição

- Com estes operadores podemos **atribuir valor a uma variável**;
- O mais conhecido é o **=**, porém temos algumas variações do mesmo;
- Operadores: **+=, -=, /=, *= e %=**;
- Cada um destes fará uma **operação antes da atribuição**;

Operador ternário

- Este operador constitui uma **estrutura de condição resumida**;
- **Na maioria dos casos** vamos optar por if/else;
- Porém em situações simples podemos utilizar o ternário;
- Exemplo: 5 > 2 **?** echo “5 é maior que dois” **:** echo “5 é menor que 2”
- A primeira interrogação vem **depois da comparação**;
- E o **:** é utilizado para uma segunda situação, caso a primeira seja falsa;

Exercício 20

- Atribua dois números a variáveis distintas;
- Faça uma comparação de menor ou igual com o operador ternário;
- Imprima resultados para ambas as possibilidades;

Exercício 20

```php
<?php

$a = 10;
$b = 5;

echo $a <= $b ? "A é menor ou igual a B <br>" : "B é menor que A <br>";
```

Referências

Fernandes, Matheus Campos Linguagens de servidor: uma abordagem prática com PHPE-book. Disponível em: https://www.bibliotecadigitalsenac.com.br/?from=busca%3FcontentInfo%3D3269%26term%3Dphp#/legacy/epub/3269Acesso em 11/11/2023

Use A Cabeça! PHP e MySQLUse A Cabeça! PHP e MySQL

Desenvolvendo Websites com PHP