PHP 8 - PARTE 1

Prof. Ruan Carlos Martins

19/08/2022

Introdução

O que é o PHP?

PHP é uma linguagem de programação voltada para o Desenvolvimento Web. Seu propósito é solucionar problemas Webs da maneira mais fácil possível.

Alô, Mundo!

Esse código, que está dentro do arquivo index.php, exibe o resultado da função `echo`:

```php
<?php
echo "Alô, mundo!";
```

Misturando PHP com HTML

Esse código, que está dentro do arquivo index.php, mistura PHP com HTML:

```
<?php
echo "Alô, mundo!";
?>

<!DOCTYPE html>
<html>
    <body>
        <h1>Alô, mundo HTML!</h1>
    </body>
</html>
```

PHP dentro do HTML

Esse código, que está dentro do arquivo index.php, coloca o PHP dentro do HTML:

```
<!DOCTYPE html>
<html>
    <body>
        <?php
            echo "Alô, mundo!";
        ?>
        <h1>Alô, mundo HTML!</h1>
    </body>
</html>
```

TEXTOS EM PHP

O exemplo a seguir mostra como produzir texto com o comando echo:

```php
<?php
echo"PHP";
echo'PHP';
echo ('PHP');
```

COMENTÁRIO PHP

Comentários em PHP

Um comentário no código PHP é uma linha que não é executada como parte do programa. Seu único propósito é ser lido por alguém que esteja olhando o código.

COMENTÁRIOS EM PHP

A seguir mostraremos comentário de linha única:

```php
<?php
echo"ALÔ MUNDO!"; // COMENTÁRIO DE UMA ÚNICA LINHA
```

A seguir mostraremos comentário de várias linhas:

```php
<?php
/*
echo "Alô mundo!"

*/
```

VARIÁVEIS EM PHP

Variáveis em PHP

Variáveis são containers temporários de dados, que ficam alocados na memória, no PHP uma variável pode conter diversos tipos de dados, strings, inteiros, números ponto flutuante, objetos, etc... Em php toda variável começa com ($).

Case-sensitive em PHP

As variáveis em php são Case-sensitive. Significa que caracteres em caixa alta e em caixa baixa são tratados de modo diferente.

```php
<?php
 $x = "Alô mundo1"; // Variável com letra minúscula
 $X = "Alô mundo2"; //Variável com letra maiúscula
```

Restrições de nomes de variáveis

Exemplos de como não podemos iniciar uma variável:

```php
<?php
$1var = 0; // Não podemos iniciar com um número
$!var = 0; // Não podemos iniciar com um símbolo
$variavel(1) = 0; // Não pode conter parênteses
$variavel um = 0; // Não pode conter espaços
$variavel-um = 0; // Não pode conter hífen
```

Variáveis

Exemplos de como podemos iniciar uma variável:

```php
<?php
$var1 = 0;    // Números após letras
$var_um = 0; // Palavras separadas com underscore (snake case)
$varUm = 0;  //Camel case
$_um = 0;
$_1 =0;
```

VARIÁVEIS EM PHP

A seguir mostraremos um exemplos de variáveis em php:

```php
<?php
$nome = "Ruan"; //String
$idade = 23; //Int

echo $nome;
echo $idade;
```

Tipos de dados

As variáveis em php podem ter vários tipos de dados:

```php
<?php
    $inteiro = 100;                 //int
    $float = 10.5;                  //float (com casas decimais)
    $bool = true;                   //verdadeiro e falso
    $string = "Olá mundo!";         //String (cadeia de caracteres)
    $array = [1,2,3];               //array - coleção de valores
    $pessoa = new Pessoa();         //Objeto com propiedades e métodos
    $nulo = null;                   //variável com valor nulo
```

Operadores aritméticos

A seguir mostraremos operadores aritméticos(para execução de cálculos):

```php
<?php
    $x = 5 + 2; //adição
    $x = 4 - 2; //subtração
    $x = 4 * 5; //multiplicação
    $x = 4 / 2; //divisão
    $x = 4 % 2; //modulus (resto da divisão)
    $x = 4**2; // expoente que calcula potência
```

Operadores de atribuição

Operadores de atribuição permitem alterar o valor de uma variável, a partir do valor original.
Mostraremos nos exemplos a seguir:

```php
<?php
$x =10;
$x = $x + 20;  // resultado = 30
$x = $x - 3;  // resultado = 7
$x = $x * 5;  // resultado = 50
$x = $x / 2; // resultado = 5;
$y = 10;
$y = $y % 2; // resultado = 0;

//Forma simplificada
$x += 20;
$x -= 3;
$x *= 5;
$x /= 2;
```

Operadores de incremento e decremento

Estes operadores adicionam ou subtraem uma unidade à variável; Mostraremos nos exemplos a seguir:

```php
<?php
 $a= 10;
 $a++; // resultado = 11
 $a--; // resultado = 9

//Operadores podem ser pré-incrementados ou pós-incrementados

 ++$a;
 $a++;

 $a--;
 --$a;
```

Operadores de comparação

São operadores que permitem efetuar uma comparação entre valores, verificando se eles são valores iguais, diferentes, maiores ou menores, ou do mesmo tipo. Com resultado sempre sendo (true) ou (false).

```php
<?php

// Valores de igualdade
$a = (2 == 3); // falso
$a = (100 == 100); //true
$a = (50 == '50'); //true obs: o valor '50' é convertido de
//string para int
$a =  (50 === '50'); // false obs: A operação verifica o valor e tipo
```

Operadores de comparação

OBS: Operadores de comparação com 3 símbolos, comparam valor e tipo (===) (! ==)

```php
<?php
//Valores diferentes
$a = (2 != 3); // true (different)
$a = (2 <> 3); // true (alternative)
$a = (50 != 50); // true (different)
$a = (50 != '50'); // false
$a = (50 !== '50'); //true
```

Operadores de comparação

Comparação de valores maiores e menores

```php
<?php

$a = (2 > 3); //false
$a = (2 < 3); //true
$a = (50 >= 50);// true
$a = (50 <= 50); //true
$a = (50 < 50); //false
```

Operadores de comparação

No PHP 7 foi adicionado um novo operador de comparação, designado por spaceship operator. O operador spaceship é usado para comparar duas expressões. Ele retorna um inteiro menor do que, igual a, ou maior que zero. As comparações são realizadas de acordo com as regras de comparação de tipos habitual do PHP.

```php
<?php
 $x = 1 <=> 1; // resultado = 0 ( 1==1)
 $x = 3 <=> 2; // resultado = 1 (3 > 2)
 $x = 1 <=> 2; // resultado = -1 (1 < 2)
```

Operadores Lógicos

Operadores lógicos são frequentemente usados com os operadores de comparação. Eles permitem 'ligar' várias comparações avaliando se são verdadeiras ou falsas.

```php
<?php

$a = 100;
$b = 200;

$x = ($a < $b) && ($a < 1000); // AND -> true
$x = ($a > $b) || ($a > 1000); // OR -> false
$x = ($a < 150) && ($b > 300); // false
$x = ($a > 10) || ($b > 500); // true
$x = ($a == 100) && ($b < $a); // false
```

Operadores Lógicos

Exemplos:

```php
<?php
$x = (10 > 5) && (20 > 30);
$x = (false && true); //false (Na porta && se um for
//falso, o resultado será falso.)
$x = (2 < 1) && (3 > 2);
$x = (true && true); // true (Só é vedadeiro se todos forem verdadeiros)
$x = (2 < 4) || (4 > 10);
$x = (true || false); // true (Se um for verdadeiro, o
//resultado será verdadeiro)
$x = (2 == 3) || (3 > 1);
$x = (false || true); //true (Na porta || o resultado só é falso se todos forem
$a = false;
$x = !$a; // true (A porta not altera o valor do resultado.
```

STRINGS EM PHP

STRINGS

Uma string é uma série de caracteres que podem ser guardados numa variável. Podemos definir uma string como sendo um conjunto de letras, espaços, símbolos e algarismos que normalmente são designados por texto.

STRINGS EM PHP

Em php as strings podem ser definidas por:

```php
<?php
$a = 'String'; //Podemos utilizar aspas simples
$b = "string"; //Podemos utilizar aspas duplas

echo $a;
echo $b;
```

CONCATENAÇÃO DE STRINGS

Concatenação é um processo que, através de código, permite juntar várias strings.
Existem dois operadores de concatenação:

```php
<?php
$a = 'Ruan' . ' ' . 'Martins'; // o ponto liga as diferentes strings

$b ='Ruan';
$b .=' ' . 'Martins'; // .= Faz uma concatenação com o valor já existente na st

echo $a;

$nome = 'Ruan';
$sobrenome = 'Martins';

$nome_completo = $nome . ' ' . $sobrenome;

echo $nome_completo; // = Ruan Martins
```

STRINGS EM PHP

A string de aspas duplas interpretam os valores com efeito de Parse. Já as com aspas simples não são interpretadas.

```php
<?php
$nome = 'Ruan';
$sobrenome = 'Martins';
$nome_completo = "$nome $sobrenome";

echo $nome_completo;
```

HEREDOC E NOWDOC

(HEREDOC E NOWDOC) Apesar de não serem muito comuns, podem ser usadas para tratamento de maiores blocos de texto.

```
$cliente = 'Ruan Martins';
$email = 'rcam8029@gmail.com';
<?php
//HEREDOC
$str = <<<EOD
Exemplo de uma string
distribuída em várias linhas
utilizando a sintaxe heredoc.
EOD;

//NOWDOC
<?php
echo <<<'EOD'
Exemplo de string abrangendo várias linhas
usando a sintaxe nowdoc. As barras invertidas são sempre tratadas literalmente,
por exemplo: e.g. \\ and \'.

EOD;
```

CARACTERES DE ESCAPE

Os caracteres de escape são usados para escrever caracteres especiais, como é o caso de alguns símbolos que podem entrar em conflito com o código.

```php
<?PHP
$frase1 = 'I\'m going to Ruan\'s party!';
$frase2 = "I'm going to Ruan's party!";
$frase3 = "I'm not sure Ruan \"the great\"Will win this fight";
// "\" é o símbolo para definir um caractere de escape
```

Quebra de linha no terminal

Quebra de linha no terminal

```php
<?PHP
echo "Ruan \n\rTeste";  //\n\r qubra de linha aprensentada somente no terminal
```

Caractere de escape unicode

No php 7 foi introduzido o caracter de escape unicode, que permite apresentar caracteres a partir do seu código hexadecimal.

```php
<?PHP
echo "\u{A9}"; // copyright
echo "\u{BC}"; // 1/4
echo "\u{AE}"; //registered trademark

//Com HTML entities

echo '<br>';
echo '&copy;';
echo '&frac14';
echo '&reg;';
```

FUNÇÕES PARA OPERAR COM STRINGS

A seguir mostraremos exemplos de funções para tratamento de string:

```php
<?php
$frase = 'Esta frase tem 29 caracteres.';
$n = '<br>';

// aprensenta o primeiro caracter da frase
echo $frase[0];
echo $n;

// apresenta o sexto caracter da frase
echo $frase[5];

// retorna o número de caracteres da string
$numero_caracteres_total = strlen($frase);

//retorna as letras a partir da posição 0 e durante 4 caracteres
$primeira_palavra = strlen($frase);
```

FUNÇÕES PARA OPERAR COM STRINGS

A seguir mostraremos exemplos de funções para tratamento de string:

```php
<?php

// converte toda string em letras minúsculas
$todas_minusculas = strtolower($frase);

//converte em letras maiúsculas
$todas_maiusculas = strtoupper($frase);

//substitui uma letra pela outra
$nova_frase = str_replace('a','x', $frase); //substitui o 'a' pelo 'x'

//verifica qual a posição de um caracter dentro da string
$posicao = strpos($frase,'a');
```

FUNÇÕES PARA OPERAR COM STRINGS

Podemos comparar strings da seguinte forma:

```php
<?php
$a = 'Ruan';
$b = 'Carlos';

$c = ($a == $b); //false
$d = ($a != $b); //true

// compara strings
$nome = "Ruan Martins";
$x = str_contains($nome, 'Ruan'); // true

$x = str_starts_with($nome, 'ruan'); //false
$x = str_starts_with($nome, 'Ruan'); //true

$x = str_ends_with($nome, ' ins'); // false
$x = str_ends_with($nome, 'ins'); // true
```

ARRAYS

ARRAYS

Um ARRAY é uma coleção constituída por chaves/índices e valores. As chaves/índices podem ser númericos ou strings, e podem até coexistir os dois tipos de índices no array. os valores podem ser de qualquer tipo, inclusive serem outros arrays

ARRAYS

ARRAYS NUMÉRICOS E ARRAYS ASSOCIATIVOS

```php
<?php

$valores = array(1,2,3,20,40,50);
$nomes = array('Ruan', 'Raissa', 'Carlos');

// A partir do PHP 5,4, passou a ser possível escrever arrays de
//forma mais simplificada

$valores = [1,2,3,20,40,50];
$nomes = ['Ruan', 'Raissa', 'Carlos'];

//Para apresentar seus valores, procedemos da seguinte forma:
  echo $valores[0]; // 1
  echo $nomes[1]; // Raissa
//Os arrays tem índice de base 0
```

ARRAYS NUMÉRICOS E ARRAYS ASSOCIATIVOS

```php
<?php
//Não obrigatoriamente tem chave sequencial

$dados =[
    10 => 1000,
    20 => 2000,
    30 => 3000,
];

//Se adicionarmos um novo elemento no final da coleção, ele assumirá o último í
$dados[] = 4000; //$dados[31];

//podemos alterar valores do array usando o índice
 $valores = [10,20,30];
 $valores[1] = 2000;  //  $valores = [10,2000,30];

 // podemos também ignorar o índice para adicionar um novo elemento ao array
 $valores[] = 3000; //  $valores = [10,20,30,3000];
```

ARRAYS NUMÉRICOS

```php
<?php
 // ou de outra forma
 array_push($valores, 5000); //  $valores = [10,20,30,5000];

 //para apresentarmos valores de array em string
 echo 'os valores são:' . $valores[0] . ' e '. $valores[1];

 //ou
 echo "os valores são $valores[0] e $valores[1]";
```

ARRAYS NUMÉRICOS

```php
<?php
 // ou de outra forma
 array_push($valores, 5000); //  $valores = [10,20,30,5000];

 //para apresentarmos valores de array em string
 echo 'os valores são:' . $valores[0] . ' e '. $valores[1];

 //ou
 echo "os valores são $valores[0] e $valores[1]";
```

ARRAYS ASSOCIATIVOS

```php
<?php
//ARRAYS ASSOCIATIVOS
// Ao contrário dos arrays númericos, as chaves são do tipo string
$dados = [
    'A' => 20,
    'B' => 30,
    'C' => 40,
    'D' => 50
];

//ou
$dados = [
    'nome' => 'Ruan',
    'email' => 'rcam8029@gmail.com',
    'nacionalidade' => 'brasileiro'
];
```

ARRAYS ASSOCIATIVOS

```php
<?php
//Da mesma forma que os arrays númericos não não devem
//existir duas chaves iguais, nos associativos também não.
//Na aprensentação não é gerado nenhum erro, o valor a ser definido é sempre
//o último.

$dados = [
    'nome' => 'Ruan',
    'nome' => 'Carlos'
];
```

ARRAYS MISTOS

```php
<?php
/ É possível criar um array que combina índices números e strings.

$dados = [
    0 => 10,
    'nome' => 'Ruan',
    10 => 10000,
    11 => 'Belém',
    12 => 'Brasil'
];

//Uma estrutura deste tipo é mais complexa de gerir devido a mistura de índices
//e dados.
//Para apresentar dados do array, procedemos sempre da mesma forma

echo $dados[0]; //10
echo $dados['nome']; //Ruan
```

ARRAYS MULTIDIMENSIONAIS

```php
<?php
//são arrays númericos ou associativos, cujos valores são outros arrays.
$dados = [
    [10,20,30,40],
    [100, 200, 300, 400],
    [1000,2000,3000,4000]
];

// Neste caso, para apresentar-mos valores deste array:
echo $dados [1][2]; //300
echo $dados [0][3]; //40

//Podemos ter arrays multidimensionais associativos:
$cidades = [
    'Portugal' => ['Lisboa', 'Porto', 'Coimbra'],
    'Brasil' => ['Brasília', 'Pará', 'São paulo'],
    'Espanha' => ['Madrid', 'Barcelona', 'Sevilha']
];
echo $cidades = ['Espanha'][0]; //Madrid
echo $cidades = ['Brasil'][1]; //Pará
```

Declarações e instruções condicionais

Declarações e instruções condicionais

São instruções utilizadas para controlar o fluxo da aplicação de acordo com determinadas condições. Para a verificação das condições destas declarações, recorremos ao uso de operadores de comparação combinados com os operadores lógicos.

Declarações e instruções condicionais

Declaração IF - define blocos de código que só são executados se a condição for verdadeira (true)

```php
<?php

$nome = 'Ruan';
if($nome == 'Ruan'){
    echo 'Foi identificado o nome "Ruan"';
};

//IF ... ELSE
// Se nenhuma das opções anteriores não forem verdadeiras, o else é executado.

$nome = 'Ruan';
$idade = 23;

if($nome == 'Ruan' && $idade < 20 ){
    echo 'idade menor do que 20';
} else{
    echo 'idade maior do que 20 anos';
};
```

Declarações e instruções condicionais

IF... ELSEIF ... ELSE

```php
<?php
// IF ... ELSEIF ... ELSE

$nota = 8; // Notas de 0 a 10
if($nota == 1){
    echo 'Nota muito abaixo da média';
} elseif ($nota == 4) {
    echo 'Nota abaixo da média';
} elseif ($nota == 6) {
    echo 'Nota na média';
} elseif($nota == 8){
    echo 'Nota acima da média';
} else {
    echo 'Nota exelente! Parabéns!! :)';
};

//NOTA ACIMA DA MÉDIA
```

Declarações e instruções condicionais

Uso de operadores lógicos

```php
<?php
$numero = 10;
if($numero < 10 && $numero == 10){
    echo 'Executar código';
} else {
    echo 'Não executar código';
};

//Condições dentro de condições
$numero = 100;
if($numero > 0) {
    if($numero >= 100){
        echo 'Número é maior do que, ou igual a 100';
    } else {
        echo 'O número é positivo mas é menor que 100';
    };
} else {
    echo 'O número é negativo';
};
```

Declarações e instruções condicionais

SWITCH -Uma alternativa ao IF

```php
<?php
$nome = 'Ruan';
switch($nome) {
    case 'joão':
        //código 1
        break;
    case 'Ruan':
         //código 2
        break;
    case 'Raissa':
        //código 3
        break;

}; //código 1
```

Declarações e instruções condicionais

INSTRUÇÃO CONDICIONAL SWITCH E SINTAXE ALTERNATIVA

```php
<?php
//SINTAXE ALTERNATIVA
$varlor = 15;

if($valor == 10):
    //código 1
else:
    //código 2
endif;

//SWITCH
switch($variable):
  case 'value':
    //code
  break;
  default:
    //code
  break;
endswitch;
```

Declarações e instruções condicionais

INSTRUÇÃO UTILIZANDO HTML

```php
<?php

// INSTRUÇÃO UTILIZANDO HTML
<?php if(true): ?>
    <div>Executar código</div>
<?php else: ?>
    <h3>Executar aqui</h3>
<?php endif; ?>
```

OPERADOR TERNÁRIO

É um operador condicional que pode substituir uma estrutura simples IF ELSE

```php
<?php
// O operador necessita de três expressões ou áreas:
// 1. A expressão que será avaliada como verdadeira
// 2. A expressão que será executada com true
// 3. A expressão que será executada com false

$opcao = 0;
$nome = $opcao == 1 ? 'Joao' : 'Ruan';

//Podemos também usar da seguinte forma:

$opcao == 1 ? $nome = 'Joao' : $nome = 'Antonio';

//Podemos usar o operador ternário em operações distintas
//Por exemplo, para apresentar um texto

echo $opcao == 1 ? 'SIM' : 'NÃO';
```

OPERADOR TERNÁRIO

OPERADOR TERNÁRIO

```php
<?php

// Com instrução condicional IF
<?php if($opcao == 1): ?>
    <h2>Sim</h2>
<?php else: ?>
    <h2>Não</h2>
<?php endif; ?>

//Com operador ternário
<h2><?= $opcao == 1 ? 'SIM': 'NAO'?></h2>

//Controlar CSS
<h2 style="color: <?$opcao == 1? 'red' : 'blue'?>">Texto css</h2>
```

INSTRUÇÃO CONDICIONAL MATCH

No PHP 8 foi introduzida uma nova expressão condicional designada por March. É muito semelhante ao Switch, mas com uma sintaxe mais concisa.

```php
<?php
$x = 10;
echo match($x) {
    5 => 'Parou no 5',
    10 => 'Parou no 10',
    15 => 'Parou no 15',
    default => 'É um número diferente de 5,10 e 15'
};

//No match apenas uma expressão é valida por cada condição.
// E é possível executar a mesma operação para várias condições.

echo match($x) {
    5 => 'é 5',
    4, 5, 6 => 'é 4, 5 ou 6',
    default => 'É um número diferente'
};
```

CICLO WHILE

CICLOS/LOOPS

Existem 4 estruturas de ciclos PHP. Elas existem para permitir executar um determinado bloco de código várias vezes.

```php
<?php
//WHILE
//Definimos a condição durante a qual o ciclo deve continuar a ser executada.
// Quando essa condição for falsa, o ciclo termina e o código avança.

$x = 1;

while ($x < 10){
    echo 'ciclo em execução<br>';
    $x++;
}
echo '<hr>';
//outro exemplo
$i = 0;
while ($i < 10) {
    echo $i++ . '<br>';
}
```

CICLO WHILE

CICLOS/LOOPS

```php
<?php
//Neste caso podemos remover as chaves

$i = 0;
while ($i < 10) echo $i++ . '<br>';

//IMPORTANTE: Nunca esquecer a alteração da condição, caso contrário
//teremos um ciclo infinito.
```

CICLO DO WHILE

DO WHILE é muito semelhante ao ciclo WHILE, contudo, neste caso a condição é avaliada após a execução do ciclo.

```php
<?php
$x = 0;
do {
    echo $x++ . '<br>';
} while ($x < 10);

echo '<hr>';

//De igual modo, quando só temos uma expressão a executar, podemos ignorar
//as chaves

$x = 0;
do
  echo $x++. '<br>';
while ($x < 10);

echo '<hr>';
```

CICLO FOR

Permite executar um bloco de código determinado número de vezes, e é constituído por três parâmetros.

```php
<?php
//1. O iniciador - indica o valor inicial do contador
//2. A condição - indica qual a condição para que o ciclo termine
//3. O incremento - altera o valor do contador em cada ciclo

for ($x = 1; $x < 10; $x++) {
    echo $x . '<br>';
}
echo '<hr>';
//mais uma vez, se for utilizada apenas uma expressão, não é necessário usar
//chaves.

for ($x = 1; $x < 10; $x++)
    echo $x . '<br>';
```

CICLO FOR

```php
<?php
//Pode conter uma assinatura diferente.
$i = 1;
for (; $i < 10;){
    echo $i++ . '<br>';
}
echo '<hr>';

//O primeiro e último parâmetro também podem ser divididos em vários
//parâmetros usando a vírgula como separador
for ($contador = 1, $x = 10, $contador < 10; $contador++; $x--){
    echo $x . '<br>';
}
echo '<hr>';

//por exemplo, podemos apresentar todos os dados de um array
$nomes = ['joao', 'ana', 'carlos'];
for ($i = 0; $i < sizeof($nomes); $i++){
    echo $nomes[$i] . '<br>';
}
```

CICLO FOREACH

É um ciclo especialmente concebido para fazer uma iteração pelos valores de um array

```php
<?php
$nomes = ['Ruan','Lucas','Carlos'];
foreach($nomes as $nome){
    echo $nome . '<br>';
}
echo '<hr>';

//existem ainda uma outra assinatura que permite ir buscar a chave do valor
//do array
$capitais = [
    'Brasil' => 'Brasília',
    'Portugal' => 'Lisboa',
    'Espanha' => 'Madrid'
];
foreach($capitais as $key => $value){
    echo "Para o país $key a capital é $value<br>";
}
```

BREAK, CONTINUE E GOTO

Existem duas funções especiais dentro dos loops: break e continue

```php
<?php
//BREAK: Serve para interromper a execução de um loop

for($i = 0; $i < 20; $i++){
    echo $i.'<br>';
    if($i==10){
        break; //interrompe o loop
    }
}

echo '<hr>';

$nomes = ['Ruan', 'Raissa', 'Carlos'];
foreach($nomes as $nome){
    echo $nome . '<br>';
    if($nome =='Raissa'){
        break
```

CONTINUE

Permite avançar uma volta do ciclo ignorando o código que deveria ser executado

```php
<?php

for($i = 0; $i < 20; $i++){
    if($i==10){
        continue;//volta à linha inicial do loop
    }
    echo $i.'<br>';
}

echo '<hr>';

$nomes = ['Ruan', 'Ana', 'Raissa'];
foreach($nomes as $nome){
    if($nomes == 'Ana'){
        continue;
    }
    echo $nome . '<br>';
}
echo '<hr>';
```

GOTO

É pouco utilizado porque torna a leitura do código mais complexa. Permite sair do loop e ir para uma linha do código definida por uma label (nome seguido de:)

```php
<?php

for($i = 0; $i < 20; $i++){
    if($i==10){
        goto teste; // vai saltar para o label
    }
    echo $i. '<br>';
}

echo 'Fim do loop';

teste:
echo 'AQUI';
```