Operadores de String

São utilizados para concatenar argumentos, tendo 2 tipos:

- Ponto**(.)** - realiza a concatenação dos argumentos presentes nos lados direito e esquerdo.

Ponto e igual**(.=)** - realiza a concatenação utilizando uma variável declarada anteriormente com o argumento desejado, variável ou não.

- Veja exemplos de Operadores de Strings na **Listagem 1**.

```php
<?php

$variavel = "Dev";
$variavel = $variavel . "media! ";

$variavel2 = "Plataforma";
$variavel2 .= " para programadores!";

echo $variavel . $variavel2;
?>
```

Listagem 1. Exemplo de operadores de Strings

Nota: No PHP o `+=` não é um operador de concatenação de Strings como em outras linguagens de programação, sendo apenas um operador de atribuição. E se utilizado será emitido um alerta do interpretador.

Operadores Aritméticos

São aqueles que estudamos na escola, aquelas funções básicas de somar, subtrair, multiplicar, dividir, etc. Neste caso se os operandos são números(INTEGER e FLOAT) podem ser usados normalmente, se forem de outro tipo, seus valores serão convertidos antes da realização da operação. Com estes operadores podemos fazer qualquer operação matemática com tipo de dados numéricos, veja abaixo os operadores aritméticos no PHP:

- Adição: **(+)**
- Subtração: **(-)**
- Multiplicação: **(*)**
- Divisão: **(/)**

```php
<?php
$num1 = 2;
$num2 = 4;
$num3 = 6;

//Resultado igual a 2
$resposta1 = $num2 - $num1;

//Resultado igual a 8
$resposta2 = $num3 + $num1;

//Resultado igual a 12
$resposta3 = $num1 * $num3;

//Resultado igual a 2
$resposta4 = $num2 / $num1;

//Resultado igual a 2, já que o resto da divisão de 8 por 6 é igual a 2
$resposta5 = $resposta2 % $num3;
?>
```

Listagem 2. Exemplo de operadores aritméticos

O PHP também possui outros operadores aritméticos que trabalham apenas com um operando, são os chamados operadores unários e que servem para: trocar sinais, incrementar ou decrementar valor.

Nota: No PHP o `+=` não é um operador de concatenação de Strings como em outras linguagens de programação, sendo apenas um operador de atribuição. E se utilizado será emitido um alerta do interpretador.

Operadores de decremento e incremento

- -operando: troca o sinal do operando.
- ++operando: conhecido como pré-incremento, primeiro incrementa depois realiza a operação.
- --operando: conhecido como pré-decremento, primeiro decrementa depois realiza a operação.
- operando++: conhecido como pós-incremento, primeiro faz a operação depois incrementa.
- operando--: conhecido como pós-decremento, primeiro faz a operação depois decrementa.

Vamos mostrar um exemplo simples, como mostra a **Listagem 3**.

```php
<?php
$num1 = 2;
$num2 = 4;
$num3 = 6;
$num4 = 8;

//Resultado igual a 3
$resposta1 = ++$num2 - $num1;

//Resultado igual a 8
$resposta2 = $num3-- + $num1;

//Resultado igual a 9
$resposta3 = --$num1 + $num4;
?>
```

Listagem 3. Exemplo de operadores aritméticos e decremento/incremento

- Na **linha 8** antes de ser realizada a operação aritmética entre `$num2` e `$num1`, a variável `$num2` é incrementada em 1 para assim realizar a operação.
- Na **linha 11** primeiro é realizada a operação aritmética entre `$num3` e `$num1`, em seguida a variável `$num3` é decrementada em 1.
- Na **linha 14** antes de ser realizada a operação aritmética entre `$num1` e `$num4`, a variável `$num1` é decrementada em 1 para assim realizar a operação.

No caso acima apresentamos os operadores aritméticos assim como as operações que incrementam e decrementam de forma simples.

Operadores de atribuição

No caso de operadores de atribuição, o PHP fornece um operador básico e diversos derivados, estes sempre retornam um valor atribuído. Nos operadores

derivados de atribuição, a operação é feita entre os dois operandos, sendo atribuído o resultado ao primeiro. Veja a seguir os operadores de atribuição:

- Atribuição simples**(=)**
- Atribuição com adição **(+=)**
- Atribuição com subtração **(-=)**
- Atribuição com multiplicação **(*=)**
- Atribuição com divisão **(/=)**
- Atribuição com módulos **(%=)**
- Atribuição com concatenação **(.=)**

Vejamos um exemplo na **Listagem 4**.

```
<?php

$num = 10;
$num += 5;


?>
```

Listagem 4. Exemplo de operador de atribuição

- Na **linha 3** é atribuída a variável `$num` o valor 10.
- Na **linha 4** a variável `$num` soma 5 ao seu valor 10, definido anteriormente.

Agora vamos ver um exemplo mais completo sobre operadores de atribuição, como mostra a **Listagem 5**.

```php
<?php

//Declaração das variáveis e atribuições simples
$SOMAR = 0;
$numero1 = 2;
$numero2 = 3;
$numero3 = 4;

//Atribuição com adição
$SOMAR += $numero1;
$SOMAR += $numero2;

//Atribuição com multiplicação
$SOMAR *= $numero3;

//Atribuição com módulos
$SOMAR %= 3;

echo $SOMAR;
?>
```

Listagem 5. Exemplo de operador de atribuição

- Nas **linhas 4 a 7** são realizadas as declarações das variáveis, `$SOMAR`, `$numero1`, `$numero2`, `$numero3`.
- Na **linha 10** é realizada a atribuição com soma a variável `$SOMAR`, somando o valor de `$numero1` a ela ($SOMAR = 0 + 2).
- Na **linha 11** também é realizada a atribuição com soma a variável `$SOMAR`, somando o valor de `$numero2` ao valor obtido na atribuição anterior ($SOMAR = 2 + 3).
- Na **linha 14** é realizada a atribuição com multiplicação a variável `$SOMAR`, multiplicando o valor de `$numero3` ao valor obtido na atribuição anterior ($SOMAR = 5 * 4).
- Na **linha 17** também é realizada a atribuição com modulo a variável `$SOMAR`, calculando o resto da divisão do valor obtido na atribuição anterior por 3 ($SOMAR = 20 / 3).
- Na **linha 19** é impresso o valor 2 após realizar todas as atribuições durante a execução do script.

Veja o resultado na **Figura 2**.

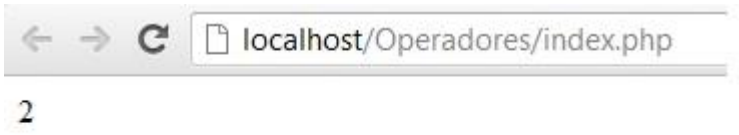

Figura 2. Resultado do código de operadores de atribuição

O resultado mostrado na figura é o resto da divisão que foi feita no final do código.

Operadores lógicos

Os operadores lógicos trabalham com valores booleanos, com o objetivo de avaliar expressões cujo valor pode ser verdadeiro ou falso, ou seja, implementando a lógica booleana.

- **and(e)** – O resultado verdadeiro só será obtido quando ambos dos dados comparados forem verdadeiros.
- **or(ou)** – O resultado verdadeiro só será obtido quando pelo menos um dos dados comparados for verdadeiro.
- **xor** – O resultado verdadeiro só será obtido quando pelo menos um dos dados comparados for verdadeiro, mas não ambos.
- **!(não)** – O resultado verdadeiro só será obtido se o valor dado não for verdadeiro.
- **&&(e)** – Igual ao operador AND.
- **||(ou)** – Igual ao operador OR.

Muito cuidado com a ordem de precedência (ordem de execução), ou você pode ter problemas com resultados, uma informação importante:

- **&& e ||** = precedência alta;
- **and e or** = precedência baixa.

Veja um exemplo com o uso dos operadores lógicos na **Listagem 6**.

```php
<?php
$booleano1 = true;
$booleano2 = false;
$booleano3 = true;

if ($booleano1 and $booleano2) {

  echo "Verdadeiro";

} else {

  //Resultado é falso pois os valores das variáveis são diferentes
  echo "Falso";

}

if ($booleano1 or $booleano2) {

  //Resultado é verdadeiro pois uma das variáveis é verdadeira
  echo "Verdadeiro";

} else {

  echo "Falso";

}

if ($booleano1 xor $booleano3) {

  echo "Verdadeiro";

} else {

  //Resultado é falso pois os valores de ambas as variáveis são verdadeiras
  echo "Falso";

}

if (!$booleano2) {

  //Resultado é verdadeiro pois o valor da variável é falso
  echo "Verdadeiro";

} else {

  echo "Falso";

}
```

```php
if ($booleano1 && $booleano3) {

  //Resultado é verdadeiro pois ambos os valores das
variáveis são verdadeiros
  echo "Verdadeiro";

} else {

  echo "Falso";

}

if (!$booleano1 || !$booleano3) {

  echo "Verdadeiro";

} else {

  //Resultado é falso, pois o valor de ambas as variáveis
foram invertidas para falso
  echo "Falso";

}




?>
```

Listagem 6. Exemplo de operador de lógicos

No exemplo acima é realizado o teste de cada tipo de operador.

Operadores de comparação

Operadores de comparação ou condicionais, são aqueles capazes de fazer comparações entre variáveis, com eles podemos saber se uma variável é maior que a outra, diferente, etc. Veja abaixo os operadores de comparação:

- **(==)** Igual a
- **(===)** Idêntico, e do mesmo tipo de dados
- **(!=),(<>)** Diferente de
- **(!==)** Não idêntico
- **(<)** Menor que
- **(>)** Maior que
- **(<=)** Menor ou igual a
- **(>=)** Maior ou igual a
- **(<=>)** Define um inteiro(-1,0,1) caso o resultado da comparação for menor que, igual ou maior que, respectivamente

Veja um exemplo com o uso dos operadores de comparação na **Listagem 7**.

```php
<?php
$numero1 = 15;
$numero2 = 5;
$numero3 = 5;

//Verifica se $numero1 é igual a $numero2
if ($numero1 == $numero2) {
  echo "Sim";
} else {
  echo "Não";
}
```

```php
//Verifica se $numero1 é diferente de $numero2
if ($numero1 != $numero2) {
  echo "Sim";
} else {
  echo "Não";
}

//Verifica se $numero1 é menor que $numero2
if ($numero1 < $numero2) {
  echo "Sim";
} else {
  echo "Não";
}

//Verifica se $numero3 é maior que $numero1
if ($numero3 > $numero1) {
  echo "Sim";
} else {
  echo "Não";
}

//Verifica se $numero1 é menor ou igual a $numero2
if ($numero1 <= $numero2) {
  echo "Sim";
} else {
  echo "Não";
}

//Verifica se $numero1 é maior ou igual a $numero2
if ($numero1 >= $numero2) {
  echo "Sim";
} else {
  echo "Não";
}

//Verifica se $numero1 é idêntico a $numero2
if ($numero3 === $numero2) {
  echo "Sim";
} else {
  echo "Não";
}

//Verifica se $numero1 não é idêntico a $numero2
if ($numero1 !== $numero2) {
  echo "Sim";
} else {
  echo "Não";
}

/*Define o inteiro 1 como resultado da comparação,
```

```
pois $numero1 é maior que $numero2*/
if ($numero1 <=> $numero2) {
  echo "Sim";
} else {
  echo "Não";
}
```

```
?>
```

Listagem 7. Exemplo de operador de comparação

Precedência de operadores

Numa expressão onde é utilizado mais um operador o PHP verifica a precedência de cada um deles, onde determinados operadores possuirão menor, igual ou maior prioridade do que os demais. Por exemplo, os 20 + 4 * 2, a resposta é 28 e não 48, porque o operador de multiplicação "*" tem prioridade maior do que o operador de adição "+".

Estruturas condicionais

If/else

Os operadores condicionais são um dos recursos mais básicos da programação, e são essenciais no desenvolvimento da lógica de qualquer sistema ou aplicação. Eles são utilizados quando é necessário que determinado código seja executado apenas se atender a uma condição.

Neste documento apresentaremos as estruturas de condição `if/else`, `else if`

O `if/else` é um operador condicional utilizado quando desejamos executar um bloco de código apenas se determinada condição for atendida, por exemplo, exibir um conteúdo apenas se o usuário for maior de idade.

Sintaxe do `if/else`:

```php
<?php
if ($condicao)
{
// código se a condição for atendida
}
else
{
// código se a condição não for atendida
}
?>
```

Assim, caso a condição seja atendida, o primeiro bloco de código será executado, senão, é executado o segundo bloco.

Nota:O uso do `else` não é obrigatório. Em muitos casos apenas o `if` será suficiente. Além disso, o uso das chaves para delimitar o bloco de código é opcional caso nele seja declarada apenas uma linha. Ainda assim, recomenda-se o uso das chaves visando melhor legibilidade do código.

Exemplo de uso:

```php
<?php
$a = 4;
$b = 2;

if ($a/$b == 2)
{
echo "O resultado da divisão é 2";
}
else
{
echo "O resultado da divisão não é 2";
}
?>
```

Neste código verificamos se a divisão de um número é igual a 2. Caso verdadeiro, executamos o primeiro bloco de código, caso contrário, o segundo.

Else if

Além do `if/else`, existe também a condicional `elseif` ou `else if`. Essa opção é utilizada caso seja necessário verificar duas ou mais condições.

Sintaxe do `if/else` com `elseif`:

Observe que agora temos duas condições. Caso nenhuma delas seja atendida, o código declarado dentro do `else` é executado.

Nota: Podemos declarar quantos `elseif` forem necessários para atender a lógica a ser implementada.

Exemplo de uso:

```php
<?php
if ($condicaoUm)
{
// código se a condiçãoUm for atendida
}
elseif ($condicaoDois)
{
// código se a condiçãoDois for atendida
}
else
{
// código se nenhum das condições forem atendidas
}
?>
```

Este código é semelhante ao anterior, mas como declaramos mais uma condição foi necessário utilizar o `elseif`.

EXERCICIOS

1. Calculadora Simples:
Crie uma calculadora simples em PHP que permita aos usuários realizar operações de adição, subtração, multiplicação e divisão em dois números fornecidos como entrada. Use os operadores aritméticos para realizar as operações.

2. Concatenação de Strings:
Crie um programa que concatene duas strings fornecidas pelos usuários usando diferentes operadores de concatenação.

3. Verificação de Igualdade e Maior/Menor:
Crie um programa que verifica se dois números fornecidos pelos usuários são iguais ou se um é maior ou menor que o outro usando operadores de comparação.

4. Operadores Lógicos:
Crie um programa que usa operadores lógicos para verificar se um número fornecido pelo usuário está dentro de um intervalo específico.

Estrutura de Repetição

For

O `for` é a estrutura de repetição do PHP que utilizamos quando sabemos a quantidade de repetições que devem ser executadas.

Sintaxe do `for`:

```
for (expressão 1; expressão 2; expressão 3) {
    // bloco de código
}
```

A sintaxe é composta por três expressões separadas por ponto e vírgula, cada uma delas podendo ter zero, uma ou mais declarações separadas por vírgula:

- **Expressão 1**: Executada somente uma vez, ao iniciar o loop. Normalmente a utilizamos para declarar e inicializar as variáveis que faremos uso para controlar o número de iterações do loop;
- **Expressão 2**: Expressão booleana, validada antes de cada iteração do loop. Se a expressão contiver múltiplas expressões, todas serão executadas, mas somente o resultado da última será considerado. Se a expressão for vazia, ela será interpretada como verdadeira. O loop somente será executado enquanto essa expressão retornar `true`;
- **Expressão 3**: Executada ao final de cada iteração, normalmente a utilizamos para declarar a forma de atualização do valor da variável avaliada na expressão 2.

Exemplo de uso:

```php
for ($i=0; $i < 10; $i++) {
    echo $i."<br>";
}
```

Ao executar este código serão impressos os números de 0 a 9.

Nota: O `for` pode ser declarado sem um bloco de código, colocando as ações dentro da terceira expressão. Ademais, caso haja apenas uma linha a ser executada, as chaves são opcionais. No entanto, recomenda-se o uso delas mesmo nestas ocasiões, visando manter uma boa legibilidade do código.

Em PHP, também podemos declarar o `for` utilizando a sintaxe alternativa, com "dois pontos" e `endfor`.

Exemplo de uso:

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

Foreach

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.

Por exemplo, supondo que fosse preciso percorrer um vetor com alguns elementos com o FOR. Seria necessário utilizar um contador para servir também de índice para acessar cada elemento, conforme a listagem a seguir.

Listagem 12: Iteração em vetor com FOR

Esse código poderia ser simplificado com a utilização do operador FOREACH, cuja sintaxe é mostrada abaixo.

Então, reescrevendo o código da Listagem 12, teríamos:

Listagem 14: Exemplo de uso do FOREACH

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

Assim, a cada iteração (repetição) a variável $item representa um elemento do vetor.

Break

As estruturas de repetição possuem um comando para finalizar um loop caso seja necessário, o comando `break`. Com ele podemos interromper a execução de um loop a qualquer momento.

Exemplo de uso:

```php
$pararLoop = 9;

for($numero = 0; ; $numero++){
    if($numero == $pararLoop){
        break;
    }
    echo $numero."<br>";
}
```

Ao executar este código o loop declarado dentro do `for` será processado até que a condição criada para chamar o `break` seja verdadeira; neste caso, `numero` ser igual a `pararLoop`. O resultado será a impressão dos valores de 0 a 8.

Continue

Além do `break`, podemos declarar a palavra reservada `continue` em uma estrutura de repetição. Quando executada, os comandos subsequentes do bloco de código serão desconsiderados e uma nova iteração iniciada.

Exemplo de uso:

```php
for($i = 0; $i < 10; $i++){
    if($i % 2 == 0){
        continue;
    }
    echo $i."<br>";
}
```

Ao executar este código serão impressos os números ímpares entre 0 e 10. Para não imprimir os números pares, utilizamos o `continue`, que pulará para a próxima iteração do loop toda vez que o resto da divisão do valor de `i` por 2 for igual a 0.

Exemplo prático

Suponha que você precisa apresentar ao usuário o total da soma dos valores de uma coleção de números. Para programar esse código, podemos utilizar a estrutura de repetição `for`.

Exemplo de uso:

```php
$colecao = [1, 3, 6, 9, 13];
$soma = 0;
for($i = 0, $j = count($colecao); $i < $j; $i++){
    $soma += $colecao[$i];
}
echo $soma;
```

Note que na expressão 1 do `for` declaramos e inicializamos duas variáveis (`i` e `j`) utilizando a vírgula para separar esses códigos. Ao executar esta implementação será impresso o resultado da soma dos números da coleção: 32.