

MÓDULO

PROGRAMAÇÃO PHP

UNIDADE

VARIÁVEIS E OPERADORES EM PHP

ÍNDICE

OBJETIVOS.....	3
INTRODUÇÃO.....	4
1. TIPOS DE DADOS	5
2. OPÇÕES DE CONVERSÃO	14
3. VARIÁVEIS	18
4. CONSTANTES.....	25
5. OPERADORES	27
5.1. OPERADORES ARITMÉTICOS.....	29
5.2. OPERADORES DE ATRIBUIÇÃO.....	29
5.3. OPERADORES DE BIT A BIT.....	30
5.4. OPERADORES DE COMPARAÇÃO	31
5.5. OPERADOR DE CONTROLO DE ERROS.....	32
5.6. AUMENTAR/DIMINUIR OPERADORES.....	33
5.7. OPERADORES LÓGICOS	34
5.8. OPERADORES DE STRINGS.....	34
5.9. OPERADORES DE ARRAY	35
5.10. OPERADORES DE CLASSE	35
CONCLUSÃO.....	37

AUTOAVALIAÇÃO	39
SOLUÇÕES	43
PROPOSTAS DE DESENVOLVIMENTO DO ESTUDO	44
BIBLIOGRAFIA	45

OBJETIVOS

Com esta unidade didática, pretende-se que desenvolva os seguintes objetivos de aprendizagem:

- Reconhecer os tipos de dados da linguagem.
- Criar e trabalhar com variáveis e constantes.
- Reconhecer os operadores disponíveis.

INTRODUÇÃO

As linguagens de programação baseiam-se no trabalho com informação e o PHP não é diferente. Por isso, deve ter em conta que tipos de dados pode utilizar. Esta unidade dedicar-se-á à criação de variáveis que ajudam a conter esses dados e ao método de trabalhar com os mesmos.

Nesta unidade aprenderá a usar variáveis, constantes e operadores. Para realizar os exemplos, será necessário ter instalado um editor de código para editar PHP e um servidor web para poder testá-los.

1. TIPOS DE DADOS

Os tipos de dados primitivos utilizados em PHP são os seguintes:

Booleans (bool)

Este tipo expressa um valor true ou false. A sua sintaxe é a seguinte:

```
<?php $variavel = True; ?>
```

Inteiros (int)

É um número negativo ou positivo sem decimais, por exemplo:

```
<?php $variavel = 234; ?> /*positivo*/  
<?php $variavel = -23; ?> /*negativo*/
```

Este tipo de dados também aceita valores em hexadecimal e octal:

```
<?php $variavel = 0123; ?> /*octal*/  
<?php $varivel = 0x1B; ?> /*hexadecimal*/
```

Qualquer valor manipulado que após uma operação devolva um número fora da categoria dos inteiros torna-se automaticamente um float.

Float (float)

São os números com casas decimais, por exemplo:

```
<?php $variavel = 3.168; ?>
```

String (string)

É um conjunto de caracteres sem um limite estabelecido. Podem ser criados com plicas ou aspas, por exemplo:

```
<?php $variavel = 'primeira variávelem string'; ?>
<?php $variavel = "primeira variávelem string"; ?>
```

Quando pretender colocar plicas ou aspas dentro de uma string como valor e usar esses mesmos caracteres como delimitadores, terá de indicar que quer colocar aspas e não fechar a string. Para isso, é utilizada a barra invertida (\). Caso queira colocar uma barra invertida deverá utilizar duas barras invertidas (\\), por exemplo:

```
<?php $variavel = 'Citando o D\'artação "todos por um..."'; ?>
```

Se colocar a string dentro de aspas, existem vários caracteres de escape para criar efeitos no texto:

\n	Nova linha
\t	Guia horizontal
\r	Carriage return (volta ao início da linha)
\f	Avanço de página
\v	Guia vertical
\\	Barra invertida

<code>\"</code>	Aspas
<code>\\$</code>	Cifrão

Existem dois métodos menos utilizados na criação de strings que evitam o caráter de escape para as aspas: Heredoc e Nowdoc; em ambos, os símbolos utilizados são: três vezes o símbolo de menor (<<<), seguido pelo identificador e pelo conteúdo da string (numa nova linha), e finalmente o identificador atribuído para fechar a definição, seguido por um ponto e vírgula (;) para fechar o comando. O último identificador e o ponto e vírgula devem estar sozinhos na última linha.

Por exemplo:

```
<?php
    $variavel = <<<TESTE
Este é um teste de Heredoc no
Qual não é necessário escapar da ' e escrever
O texto como pretender.
TESTE;
?>
```

Nowdoc é igual ao anterior, mas o identificador é definido entre plicas. A diferença é que neste nenhuma verificação é feita, ou seja, no anterior, se incluir uma variável, mostrará o valor da variável; no caso do Nowdoc, mostrará o cifrão e o nome da variável (é ideal para mostrar excertos de código). A sua sintaxe é a seguinte:

```
<?php
    $varivel = <<<'TESTE'
Este é um teste de Nowdoc no
Qual não é necessário escapar da ' e escrever
O texto como pretender.
    TESTE;
?>
```

No PHP, uma string é um array de caracteres, ou seja, pode tratá-la como se fosse um array e, por exemplo, aceder a alguns dos caracteres pelo índice de posição. Por exemplo:

```
<?php
    $variavel = "isto é um teste";
    echo $variavel[2]; /* mostrará 't'*/
?>
```

Arrays (ou vetores)

Um array é um mapa de computador, ou seja, um tipo de dado que associa valores a chaves, e que pode ser usado para matrizes, tabelas, filas de informações, entre outros. Também pode criar arrays que tenham outros arrays (arrays multidimensionais). Para criar um array utiliza-se a função de construção "array" do PHP, declarando os conjuntos (chave e valor) separados por vírgulas. As chaves podem ser números inteiros, floats ou strings, enquanto os valores podem ser de qualquer tipo. Segue-se um exemplo:

```
<?php
    $vector = array( "Idade" => 12, 20 => true);
?>
```

Também pode criar arrays sem indicar nenhuma chave. Nestes casos, criará automaticamente uma chave numérica consecutiva começando no 0.

O conteúdo de qualquer chave pode ser acedido a qualquer momento usando o identificador:

```
<?php
    echo $vector["Idade"]; //mostra 12
    echo $vector[20]; //mostra 1
?>
```

Para criar um array multidimensional, basta criar um array que contenha outro como valor-chave:

```
<?php
$vector = array(
    1=> array("nome"=>"Pedro","idade"=>12),
    2 => array("nome"=>"Ana","idade"=>22),
    3 => array("nome"=>"Luis","idade"=>41)
);

echo $vector[2]["nome"]; //mostra Ana
?>
```

Um array pode ser modificado ou eliminado, por isso, seguem-se os possíveis casos que podem ocorrer ao trabalhar com um array.

O array para o exemplo:

```
$vector = array("nome"=>"Maria","idade"=>23):
```

- Para modificar um valor, é feita a referência à chave do array com parênteses e é colocado o novo valor. Por exemplo:

```
$vector["nome"] = "Ana";
// altera o valor do nome para Ana
```

- Para adicionar um valor, é atribuído o valor desejado a uma nova chave. Por exemplo:

```
$vector["casada"] = true;
//nova chave com um valor boolean
```

- Para remover um elemento de um array, é utilizado o `unset` e a chave a eliminar. Por exemplo:

```
unset($vector["idade"]);
```

- Para remover um array inteiro, é utilizado o `unset` mas com apenas o nome do array. Por exemplo:

```
unset($vector);
```

Para trabalhar com arrays existem várias funções que facilitam o processo.

- **print_r**, função que permite exibir todos os conjuntos de valores-chave de um formulário no ecrã.

Neste exemplo, será utilizada a função `print_r` para exibir todos os dados de um array.

```
<html>
<body>
<p>
<pre>
<?php
    $a = array ('nome' => 'Pedro', 'Idade' => '34',
        'carros' => array ('renault', 'seat', 'ford'));
    print_r ($a);
?>
</pre>
</p>
</body>
</html>
```

- **array_values**, função que é utilizada para reindexar arrays que foram modificados. Por exemplo:

```
<?php
$vec = array('seat', 'renault', 'fiat');
unset($vec[1]);
/* deixa o array como (0=>'seat',2=>'fiat')*/

$vec_r = array_values($vec);
/* o array $vec_r será (0=>'seat',1=>'fiat')*/
?>
```

- **foreach**, permite percorrer chave a chave todos os elementos de um array, criando um item com o qual trabalhar.

Neste exemplo será criado um array com números e será mostrado o resultado da multiplicação de cada número por 11, utilizando o foreach:

```
<body>
<p>
<?php
$vector = array (34,2,12,14);
foreach ($vector as $item => $value)
{
echo "Valor {$item} : " . ($value * 11) . "<br/>";
}
?>
</p>
</body>
```

Mostrando como resultado:

```
Valor 0: 374
Valor 1: 22
Valor 2: 132
```

Valor 3: 154

Objetos (class)

Os objetos são, como o nome sugere, elementos de programação orientada a objetos; conjuntos complexos de propriedades e funções. Segue-se um exemplo de um objeto:

```
class aluno{  
    $nome = "Pedro";  
    function darNome(){  
        echo $this->nome;  
    }  
}
```

Iteráveis

É um pseudotipo de dados introduzido no PHP 7.1 que aceita arrays ou objetos que implementam a interface Traversable.

Basicamente, é um tipo de dados que é percorrido com foreach.

Recursos (resource)

São tipos especiais de dados que contêm referência a recursos externos como a conexão ao MySQL, por exemplo. Na bibliografia desta unidade consta uma referência para ver todos os tipos de recursos existentes.

Nulo (null)

É um tipo que indica uma variável sem valor. Uma variável é considerada do tipo nulo se a constante NULL lhe for atribuída, se nenhum valor foi atribuído à variável ou se foi utilizado unset(). Por exemplo:

```
$variavel = NULL;
```

2. OPÇÕES DE CONVERSÃO

Para os tipos de dados vistos anteriormente, existem várias opções de conversão. Segue-se uma lista das conversões.

Booleans

Para converter um valor em boolean, é utilizado o `bool`, embora neste caso geralmente não seja necessário, uma vez que um valor é automaticamente convertido em `true` ou `false` quando uma relação deste tipo é estabelecida (por exemplo, numa condição `if`). Os seguintes valores são considerados falsos ao converter um valor num boolean:

- Inteiro com valor 0.
- Float com valor 0,0.
- String vazia.
- Array com 0 elementos.
- Objeto sem variável.
- O tipo `NULL`.
- Objetos 'SimpleXML' criados a partir de tags vazias.

O resto irá devolver true. Ainda assim, é possível utilizar expressão bool para forçar a conversão, por exemplo:

```
var_dump((bool) "");           // bool(false)
var_dump((bool) 1);            // bool(true)
var_dump((bool) array(12));    // bool(true)
var_dump((bool) array());      // bool(false)
```

A função `var_dump` dá informações sobre as variáveis, mostrando o que está indicado nos comentários depois de cada expressão.

Inteiros

Para converter um valor num inteiro, é utilizado o `int` ou `integer`; também é possível usar a função `intval()`. Seguem-se alguns exemplos:

```
$var = ((int) "12" ); // converte o texto "12" em número
$var = ((integer) "12" ); // converte o texto "12" em número
$var = intval('42'); // converte o texto "42" em número
```

Floats

Para todos os tipos de valores (exceto strings), a conversão é feita da mesma maneira como se o valor tivesse sido convertido para um inteiro.

No caso de strings, se a string não tiver os caracteres `.`, `e` ou `E` e o valor cair dentro dos limites dos inteiros, a string será avaliada como um inteiro; em todos os outros casos, será considerada um float. Seguem-se alguns exemplos para esclarecer:

```
$var = 1 + "10.5";           // $var é um float (11.5)
$var = 1 + "-1.3e3";          // $var é um float (-1299)
$var = 1 + "cas-1.3e3";       // $var é um inteiro (1)
$var = 1 + "cas3";            // $var é um inteiro (1)
```

```
$var = 1 + "10 pássaros";    // $var é um inteiro (11)
$var = 4 + "10.2 polvos";    // $var é um float (14.2)
$var = "10.0 polvos" + 1;    // $var é um float (11)
$var = "10.0 polvos" + 1.0;  // $var é um float (11)
```

Strings

Qualquer valor pode ser convertido numa string utilizando a função `String` ou `strval()`, tendo em consideração que o valor `true` dos booleans é convertido em `"1"` e o valor `false` em `"0"`.

```
$var = ((String) 23);
$var = strval(23);
```

Arrays

Para os inteiros, floats, strings, booleans e resources, converter um valor num array devolve um array de um único elemento com índice de chave 0 e o valor passado; se for utilizado um tipo de objeto, o resultado é um array cujos elementos são as propriedades do objeto, utilizando o `(array)`.

```
$var = ((array) 23);
```

Iteráveis

Não há nenhuma conversão para este tipo de dados.

Objetos

Se um valor de qualquer tipo for convertido num objeto utilizando o (object), uma instância genérica de classe stdClass é criada:

```
$var = (object) 'olá';  
echo $var->scalar; //mostra olá
```

Resources

Por serem tipos especiais de dados, não pode haver nenhuma conversão entre outros tipos de dados.

3. VARIÁVEIS

Como foi visto nos exemplos em PHP, as variáveis são representadas por um cifrão (\$) antes do nome da variável; os nomes das variáveis são case sensitive, ou seja, são sensíveis a maiúsculas e minúsculas.

As variáveis devem começar com uma letra ou underscore (_) seguido de quantas letras e números pretender.

```
$_variavel = 23;  
$variavel = "23";  
$_3 = 3;
```

A variável \$this é especial para o navegador e não pode ser atribuída.

Como todas as linguagens de programação, o PHP possui várias variáveis predefinidas, que fornecem informações e valores para qualquer script executado. As variáveis predefinidas mais comuns são:

- **\$GLOBALS**, referência a todas as variáveis globais.
- **\$_GET**, variáveis do HTTP GET.
- **\$_POST**, variáveis do HTTP POST.
- **\$_SERVER**, informações do ambiente.
- **\$_FILES**, carrega ficheiros do HTTP.

- **\$_REQUEST**, variáveis de solicitação HTTP.
- **\$_SESSION**, variáveis de sessão.
- **\$_ENV**, variáveis de ambiente.
- **\$_COOKIE**, cookies HTTP.
- **\$http_response_header**, cabeçalhos HTTP.
- **\$argc**, número de argumentos passados para um script.
- **\$argv**, array de argumentos passado para um script.

Um dos aspetos mais importantes a ter em consideração ao utilizar variáveis é onde é que estas são definidas; as variáveis padrão criadas da forma normal são variáveis locais, ou seja, podem ser utilizadas no contexto em que foram criadas, mas não dentro de funções, pois definir uma função cria um novo local.

Segue-se um exemplo para ver como as variáveis e os seus diferentes locais funcionam:

```
<html>
<body>

<p>

<?php

    $var = 2;
    $var_2 = 5;

    echo "valor somado: " . ($var + $var_2). "<br/>";

    function teste() {
        echo "valor somado com a função: " . ($var + $var_2);
    }

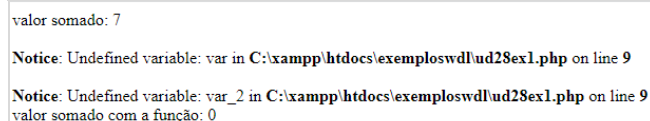
    teste();
```

```
?>

</p>

</body>
</html>
```

Este exemplo devolverá 7 no primeiro echo fora da função e 0 no segundo, dentro da função, já que neste segundo caso as variáveis que tenta utilizar são locais à função e não estão definidas. Pode-se também observar, na imagem abaixo, que devolve dois erros, indicando que a `var` e a `var_2` não estão definidas.



```
valor somado: 7
Notice: Undefined variable: var in C:\xampp\htdocs\exemplos\ud28ex1.php on line 9
Notice: Undefined variable: var_2 in C:\xampp\htdocs\exemplos\ud28ex1.php on line 9
valor somado com a função: 0
```

Para evitar o caso anterior, é possível utilizar a palavra reservada “global”, para a qual são passadas as variáveis que se pretende considerar enquanto globais como parâmetro. O exemplo anterior ficaria desta forma:

```
<?php
$var = 2;
$var_2 = 5;
echo "valor somado: " . ($var + $var_2). "<br/>";
function teste()
{
    global $var, $var_2; // utiliza as locais em relação ao código
    echo "valor somado com a função: " . ($var + $var_2);
}
teste();
?>
```

A visualização da página será:

```
valor somado: 7  
valor somado com a função: 7
```

Neste caso, 7 será exibido nos dois echos que aparecem no código. Outra alternativa seria utilizar o array `$GLOBALS` reservado que contém o nome e o valor das variáveis globais. Por exemplo:

```
<?php  
    $var = 2;  
    $var_2 = 5;  
    echo "valor somado: " . ($var + $var_2) . "<br/>";  
    function teste(){  
        echo "somado com a função: " . ($GLOBALS[ 'var' ] +  
        $GLOBALS[ 'var_2' ] );  
    }  
    teste();  
?>
```

A visualização da página será:

```
valor somado: 7  
somado com a função: 7
```

Outra característica do local das variáveis é o uso da variável estática: esta existe no local da função e não perde o valor ao sair do mesmo, por exemplo, para criar um contador.

Segue-se um exemplo de como as variáveis estáticas funcionam:

```
<html>
<body>
<p>
<?php
function contador()
{
    static $ct = 0;
    echo $ct;
    $ct++;
}
contador();
contador();
contador();
contador();
contador();
?>
</p>
</body>
</html>
```

A visualização da página será:

01234

Este exemplo devolve 01234, porque, mesmo ao sair da função e inserir as chamadas e definindo a variável `$ct` como `static`, são obtidas duas coisas: a primeira, que é inicializada com 0 apenas na primeira vez que se entra na função, e a segunda, o valor que não deve ser perdido ao sair da função.

Existe um conceito em PHP, chamado “variables variables”, que permite utilizar os valores das variáveis como nomes de outras variáveis. Segue-se um exemplo para entender melhor:

```
<?php
$var = 'olá'; //definir a variável $var que contem 'olá'

$$var = 'mundo'; //definir a variável $ola usando $var e dando a
esta o valor mundo

echo "$var ${$var}"; //mostra a mensagem, assim como coloca $var
$ola

?>
```

A visualização da página será:

olá mundo

Na programação web o mais comum é que os dados das variáveis sejam preenchidos pelo utilizador através de um formulário. Esses formulários HTML podem enviar os dados de duas maneiras diferentes: com o método POST ou com o método GET. Na página onde é recebido o envio do formulário, poderá aceder aos dados deste formulário utilizando as variáveis predefinidas \$_GET, \$_POST ou \$_REQUEST; as duas primeiras serão usadas em remessas GET e POST, respetivamente, enquanto a terceira contém ambas as remessas.

Neste exemplo, será criada uma página que contém um formulário com vários campos enviados com o método POST, que chama um ficheiro exemplo.php, e mostrará esses dados na página web. O ficheiro HTML será composto da seguinte forma:

```
<html>
<body>
<p>
    <form action="exemplo.php" method="post">
```

```
Nome: <input type="text" name="nome"/><br/>
Email: <input type="text" name="email"/><br/>
<input type="submit" name="submit" value="enviar"/>

</form>
</p>
</body>
</html>
```

O ficheiro exemplo.php será:

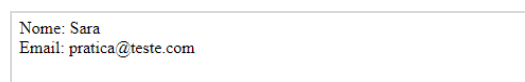
```
<?php
    echo "Nome: ".$_POST['nome']."<br/>";
    echo "Email: ".$_POST['email'];
?>
```

A visualização da página HTML será a mostrada na imagem abaixo:



A screenshot of a web form. It contains two text input fields, one for 'Nome' and one for 'Email', stacked vertically. Below the 'Email' field is a submit button labeled 'enviar'.

Após a inserção de dados, a visualização será:



A screenshot of the same web form, but now it displays the submitted data. The text 'Nome: Sara' and 'Email: pratica@teste.com' is shown in the form's container area.

4. CONSTANTES

Uma constante é um identificador ao qual está associado um valor que não pode variar durante a execução do script. Por convenção, são declaradas em maiúsculas, e deve-se ter cuidado, pois, tal como as variáveis, são case sensitive.

```
<?php
    define("CONSTANTE", "azul");
echo CONSTANTE;
?>
```

O exemplo acima define uma constante chamada "CONSTANTE" e atribui-lhe o valor "azul".

Apenas os valores dos tipos boolean, inteiros, float, string e array podem ser constantes (o último, a partir do PHP 7). E como pode observar no exemplo, para se referir ao valor, basta colocar o nome da constante, sem precisar do cifrão.

As diferenças entre variável e constante são:

- As constantes não têm um cifrão (\$).
- As constantes devem ser sempre definidas através do `define()`, não sendo suficiente atribuir-lhes um valor, como acontece com as variáveis.

- As constantes não seguem as regras dos locais das variáveis e podem ser definidas e acedidas a partir de qualquer lugar.
- As constantes não podem alterar o seu valor ou apagar o valor depois de definidas.
- As constantes contêm apenas valores booleanos, inteiros, floats, strings ou arrays.

Desde a versão 5.3 do PHP, as constantes podem ser criadas sem usar a função `define` e usando o `const`, resultando em:

```
<?php const CONSTANTE = "olá mundo"; ?>
```

No sistema de programação do PHP também existem várias constantes predefinidas. Muitas destas são criadas por extensões diferentes e só podem ser usadas no caso de terem essas extensões. Seguem-se seis constantes deste tipo que mudam dependendo de onde são utilizadas:

- `__LINE__`, linha atual do ficheiro.
- `__FILE__`, caminho absoluto do ficheiro.
- `__DIR__`, diretório do ficheiro.
- `__FUNCTION__`, nome da função.
- `__CLASS__`, nome da classe.
- `__METHOD__`, nome do método da classe.

5. OPERADORES

Antes de se passar para os operadores da linguagem PHP, serão analisadas várias expressões que ajudarão no desenvolvimento de programas. O primeiro aspeto que deverá entender é que “expressões” são todas as linhas de código que indicam que algo tem um valor, por exemplo, a expressão mais simples é igualdade:

```
$var = 5;
```

Neste exemplo foi atribuído a uma variável o valor 5. Se pretender atribuir o valor 5 a duas variáveis, não é necessário criar duas linhas de atribuição; sabendo que o código é sempre executado da esquerda para a direita, pode fazer uma dupla atribuição:

```
$var = $var2 = 5;
```

Outras expressões simples amplamente utilizadas em PHP são pré-incremento, pós-incremento, pré-decremento e pós-decremento. Seguem-se exemplos com o seu uso:

```
<?php
function doble($i)
{
    return $i*2;
}
```

```
}  
$b = $a = 5;  
/* atribuir o valor cinco às variáveis $a e $b */  
$c = $a++;  
/* pós-incremento, atribuir o valor original de $a (5) a $c */  
$e = $d = ++$b;  
/* pré-incremento, atribuir o valor incrementado de $b (6) a $d e  
$e, neste momento $d e $e são iguais a 6 */  
$f = dobro($d++);  
/* atribuir o dobro do valor de $d antes do incremento, 2 * 6 = 12,  
para o $f */  
$g = dobro(++$e)  
/* atribuir o dobro do valor de $e após o incremento, 2 * 7 = 14,  
para $g */  
$h = $g += 10;  
/* primeiro, $g é incrementado em 10 e termina com o valor 24. O  
valor do incremento (24) é então atribuído a $h, e $h também  
termina com o valor 24. */  
?>
```

Existe uma expressão utilizada em muitas linguagens de programação chamada condicional ternária. Baseia-se na avaliação da primeira expressão e, se esta for verdadeira, a segunda é avaliada; caso contrário, a terceira é avaliada. Segue-se um exemplo ilustrativo:

```
<?php  
$primeiro = 1;  
$segundo = 20;  
$terceiro = 30;  
$var = $primeiro ? $segundo : $terceiro;  
echo $var; //devolve 20  
?>
```

5.1. OPERADORES ARITMÉTICOS

São os operadores que permitem fazer cálculos matemáticos básicos:

- **Negativo**, oposto a um determinado valor (-valor).
- **Soma**, soma de dois valores (valor + valor).
- **Subtração**, subtração de dois valores (valor - valor).
- **Multiplicação**, multiplicação de dois valores (valor * valor).
- **Divisão**, quociente de dois valores (valor / valor).
- **Módulo**, resto de uma divisão (valor % valor).

Por exemplo:

```
<?php
$a = 2;
$b = 4;
$c = -$a;          //-2
$c = $a + $b;      //6
$c = $a - $b;      //-2
$c = $a * $b;      //8
$c = $b / $a;      //2
$c = $b % $a;      //0
?>
```

5.2. OPERADORES DE ATRIBUIÇÃO

O operador de atribuição básico é o símbolo de igual (=) e indica que o que fica à esquerda do operador assume o valor do que está à direita, podendo fazer uma série de atribuições complexas. Por exemplo:

```
<?php $a = ($b =4) + 5; ?>
```

No exemplo acima, \$a será 9 e \$b será 4.

Além deste operador de atribuição básico, existem outros que são combinações deste e de outros operadores, como:

```
<?php
$a = 4;
$a += 4; // define $a como 8, como se $a = $a + 4;
$b = "olá ";
$b .= "mundo"; // define $b como "olá, mundo"
?>
```

Há também a atribuição por referência. Neste caso é utilizado o valor contido na variável para criar uma nova variável com esse valor e nome:

```
<?php
$a = 3;
$b = &$a; // $b é uma referência $a
?>
```

5.3. OPERADORES DE BIT A BIT

Estes operadores permitem avaliar bits específicos dentro de um valor inteiro. As operações possíveis são:

- valor **&** valor2, (e), devolve os bits ativos em ambos.
- valor **|** valor2, (ou), devolve os bits ativos em qualquer um dos valores.
- valor **^** valor2, (ou exclusivo), devolve ativos de um ou outro valor, não de ambos.
- **~** valor, (não), os bits do valor ativo são ativados e vice-versa.
- valor **<<** valor2, desloca os bits de valor tantos passos à esquerda quanto o valor2; cada passo é multiplicado por dois.

- `valor >> valor2`, desloca os bits de `valor` tantos passos para a direita quanto `valor2`; cada passo deve ser dividido por dois.

Por exemplo:

```
<?php
echo 12 ^ 9; // devolve 5
echo "12" ^ "9"; // devolve o carácter de retrocesso (ascii 8)
//('1' (ascii 49)) ^ ('9' (ascii 57)) = #8
echo "hallo" ^ "hello";
// devolve os valores ascii #0 #4 #0 #0 #0
// 'a' ^ 'e' = #4
echo 2 ^ "3"; // Sale 1
// 2 ^ ((int)"3") == 1
echo "2" ^ 3; // Sale 1
// ((int)"2") ^ 3 == 1
?>
```

5.4. OPERADORES DE COMPARAÇÃO

Estes operadores permitem comparar dois valores entre si, devolvendo valores `true` ou `false`. São os seguintes:

- `valor == valor2`, (igual), devolve `true`, se o valor for igual ao `valor2`.
- `valor === valor2`, (idêntico), devolve `true`, se o valor for igual ao `valor2` e eles forem do mesmo tipo.
- `valor != valor2`, (diferente), devolve `true`, se o valor não for igual ao `valor2`.
- `valor <> valor2`, (diferente), devolve `true`, se o valor não for igual ao `valor2`.
- `valor !== valor2`, (não idênticos), devolve `true`, se o valor não for igual ao `valor2` ou se não forem do mesmo tipo.

- `valor < valor2`, (menor que), devolve `true`, se o valor for menor que `valor2`.
- `valor > valor2`, (maior que), devolve `true`, se o valor for maior que `valor2`.
- `valor <= valor2`, (menor ou igual), devolve `true`, se o valor for menor ou igual a `valor2`.
- `valor >= valor2`, (maior ou igual), devolve `true`, se o valor for maior ou igual a `valor2`.

Se um número for comparado a uma string, a string será convertida num número antes da verificação. Por exemplo:

```
<?php
var_dump(0 == "a"); // 0 == 0 -> true
var_dump("1" == "01"); // 1 == 1 -> true
var_dump("10" == "1e1"); // 10 == 10 -> true
var_dump(100 == "1e2"); // 100 == 100 -> true
?>
```

5.5. OPERADOR DE CONTROLO DE ERROS

O PHP tem um símbolo como operador de erro, o arroba (@). Quando é prefixado antes de qualquer expressão em PHP, qualquer mensagem de erro que ocorrer será ignorada. Este operador pode ser anexado a funções, constantes, variáveis, etc.

O código seguinte, por exemplo, evita que um erro seja registado se o índice `$item` não existir:

```
$valor = @$vector[$item];
```

Desta forma, quando ocorrer um erro, este não será exibido no navegador, evitando que o utilizador veja acidentalmente informações confidenciais.

5.6. AUMENTAR/DIMINUIR OPERADORES

Estes operadores afetam apenas os tipos numéricos e strings, e são os seguintes:

- Pré-incremento, incrementa valor em um valor e devolve-o (++valor).
- Pós-incremento, devolve o valor e então incrementa-o em um valor (valor++).
- Pré-decremento, diminui o valor em um valor e devolve-o (--valor).
- Pós-decremento, devolve o valor e diminui-o em um valor (valor--).

Segue-se um exemplo de como pré-incrementar um número e um carácter.

```
<html>
<body>
<p>
<?
$var=34;
echo "pré-incremento de $var ";
echo "agora a variável vale ".$++$var."<br/><br/>";
$var2='F';
echo "pré-incremento de $var2 ";
echo "agora a variável vale ".$++$var2."<br/><br/>";
?>
</p>
</body>
</html>
```

5.7. OPERADORES LÓGICOS

Estes operadores ajudam a perguntar sobre o estado de várias afirmações ao mesmo tempo, por exemplo, para saber se duas expressões são verdadeiras ou se as duas comparações são verdadeiras ao mesmo tempo. São os seguintes:

- `$um and $dois`, (e), devolve true, se as duas comparações forem true.
- `$um && $dois`, (e), como o anterior, mudando apenas a expressão.
- `$um or $dois`, (ou), devolve true, se qualquer um dos dois for true.
- `$um || $dois`, (ou), como o anterior, mudando apenas a expressão.
- `$um xor $dois`, (exclusivo), devolve true, se algum dos elementos for true, mas não os dois ao mesmo tempo.
- `!$um`, (não), devolve true, se `$um` não for true.

Seguem-se alguns exemplos:

```
$var = (2=2 && 3=4)           //false
$var = ('a' = 'a' xor 'b' = 'b') //false
$var = (1 = 3 or 5 = 5)         //true
```

5.8. OPERADORES DE STRINGS

Existem dois operadores especiais de strings que podem ser utilizados para concatenar várias strings: o primeiro é um ponto (.) que devolve a união das strings à esquerda e à direita do ponto; e o segundo é um ponto e um igual (.=) que adiciona ao lado esquerdo o lado direito do símbolo.

Por exemplo:

```
<?php
$um = "olá ";
$dois = $um."mundo"; // $dois será "olá, mundo"
```

```
$tres = "olá ";  
$tres .= "mundo"; // $tres será "olá, mundo"  
?>
```

5.9. OPERADORES DE ARRAY

Estes operadores permitem interagir com vários arrays simultaneamente:

- `$vec + $vec3`, a união de dois arrays.
- `$vec == $vec2`, devolve true, se tiver os mesmos pares de chaves => valor.
- `$vec === $vec2`, devolve true, se tiver os mesmos pares chave => valor na mesma ordem e dos mesmos tipos.
- `$vec != $vec2` ou `$vec <> $vec2`, devolve true, se não forem iguais.
- `$vec !== $vec2`, devolve true, se não forem idênticos.

5.10. OPERADORES DE CLASSE

Existe apenas um, chamado `instanceof`, que devolve true, se uma variável for uma instância de uma classe criada. Por exemplo:

```
<?php  
class aluno{}  
class professor{}  
$var = new aluno;  
var_dump($var instanceof aluno);    //devolve true  
var_dump($var instanceof professor); //devolve false  
?>
```


CONCLUSÃO

Os operadores são ferramentas que permitirão criar programas mais complexos e úteis em PHP. A eles juntam-se os tipos, variáveis e constantes, para uma mais versátil capacidade de utilização das funcionalidades do PHP.

AUTOAVALIAÇÃO

1. Quantos tipos de dados primitivos existem?
 - a) Quatro.
 - b) Seis.
 - c) Oito.
 - d) Dez.

2. Qual dos seguintes não é um tipo de dados em PHP?
 - a) Boolean.
 - b) String.
 - c) Array.
 - d) Bit.

3. Qual é o carácter de escape das strings que permite saltar linhas num texto?
 - a) \f.
 - b) \v.
 - c) \n.
 - d) \t.

4. Qual é o método que permite escrever um texto sem ter de escapar a nenhum carácter, uma vez que é exibido sem verificar se contém código?
- a) Text.
 - b) Nowdoc.
 - c) NoCode.
 - d) Heredoc.
5. Que função de construção é utilizada para criar um array?
- a) array().
 - b) vector().
 - c) new array().
 - d) Não existe, basta declarar os valores pretendidos dentro de parênteses.
6. Que função é utilizada para eliminar uma parte de um array ou o array por inteiro?
- a) delKey.
 - b) remove.
 - c) unset.
 - d) delete.
7. Para que é utilizada a função foreach?
- a) Percorrer um array.
 - b) Encontrar uma variável.
 - c) Preencher um array com números aleatórios.
 - d) Separar todos os valores em variáveis.

8. Quais são as maneiras de converter um valor em string?
- a) toStr e String.
 - b) Apenas String.
 - c) Apenas toString.
 - d) String e strval.
9. Qual é o sinal usado para representar as variáveis?
- a) \$.
 - b) %.
 - c) &.
 - d) &.
10. O que é que é alcançado ao definir uma variável dentro de uma função com um valor static?
- a) Iniciar a variável apenas na primeira vez que se entra na função.
 - b) Não perder o valor ao sair da função.
 - c) Iniciar sempre a variável ao entrar na função e não a perder ao sair da função.
 - d) Iniciar a variável apenas na primeira vez que se entra na função e não a perder ao sair da função.

SOLUÇÕES

1.	c	2.	d	3.	c	4.	b	5.	a
6.	c	7.	a	8.	d	9.	a	10.	d

PROPOSTAS DE DESENVOLVIMENTO DO ESTUDO

Para descobrir quais são todos os tipos de recursos disponíveis em PHP, visite a seguinte página web:

- http://www.php.net/manual/pt_BR/resource.php

BIBLIOGRAFIA

- Cabezas, L. M. (2010), *PHP 5*. Madrid: Anaya Multimedia.
- The PHP Group (2001), "Instalação e Configuração". Disponível em:
https://www.php.net/manual/pt_BR/install.php. Consultado a 2 de março de 2021.

