



Introdução

Prof. Anderson Henrique

# Capítulo 2

# Manipuladores

## Funções

Uma função é um pedaço de código com um objetivo específico, encapsulado sob uma estrutura única que recebe um conjunto de parâmetros e retorna um dado. Uma função é declarada uma única vez, mas pode ser utilizada diversas vezes. É uma das estruturas mais básicas para prover reusabilidade.

## Sintaxe de declaração de uma função:

```
13  function nome_funcao($arg1, $arg2, $argN) {  
14      $valor = $arg1 + $arg2 + $argN;  
15      return $valor;  
16  }
```

## Variáveis globais

Todas as variáveis declaradas dentro do escopo de uma função são locais. Para acessar uma variável externa ao contexto de uma função sem passá-la como parâmetro, é necessário declará-la como **global**. Uma variável global é acessada a partir de qualquer ponto da aplicação.

```
25     $total = 0;
26     function km2mi($quilometros){
27         global $total;
28         $total += $quilometros;
29         return $quilometros * 0.6;
30     }
31     echo 'percorreu ' . km2mi(100) . " milhas <br>";
32     echo 'percorreu ' . km2mi(200) . " milhas <br>";
33     echo 'percorreu no total ' . $total . " quilometros <hr>";
```

## Variáveis estáticas

Dentro do escopo de uma função podemos armazenar variáveis de forma estática. Assim, elas mantêm o valor que lhes foi atribuído na última execução.

```
41 function percorre($quilometros){  
    static $total;  
43     $total += $quilometros;  
44     echo "percorreu mais $quilometros do total de $total <br>";  
45 }  
46 percorre(100);  
47 percorre(200);  
48 percorre(50);
```

## Passagem de parâmetros

Existem dois tipos de passagem de parâmetros: por valor (**by value**) e por referência (**by reference**). Por padrão, os valores são passados **by value** para as funções. Assim, o parâmetro que a função recebe é tratado como variável local dentro do contexto da função, não alterando o seu valor externo.

```
59 function incrementaByValue($variavel, $valor){
60     $variavel += $valor;
61 }
62 $a = 10;
63 incrementaByValue($a, 20);
64 echo $a;
65
66 function incrementaByReference(&$variavel, $valor){
67     $variavel += $valor;
68 }
69 $b = 10;
70 incrementaByReference($b, 20);
71 echo $b;
```

## Parâmetros com valor padrão

O PHP permite definir valores default para parâmetros. Assim, se o programador executar a função sem especificar o parâmetro, será assumido o valor predefinido.

```
79 function incrementaParamDefault(&$variavel, $valor = 40) {  
80     $variavel += $valor;  
81 }  
82 $c = 10;  
83 incrementaParamDefault($c);  
84 echo $c;
```



## Argumentos variáveis

O PHP permite definir uma função com o número de argumentos variáveis, ou seja, permite obtê-los de forma dinâmica, mesmo sem saber quais são ou quantos são. Para obter quais são, utilizamos a função **func\_get\_args( )**; para obter a quantidade de argumentos, utilizamos a função **func\_num\_args( )**

```
94  function Ola(){
95      $argumentos = func_get_args();
96      $quantidade = func_num_args();
97
98      for($n=0;$n < $quantidade; $n++){
99          echo '<br> Olá ' . $argumentos[$n];
100      }
101  }
102  Ola('João', 'Maria', 'José', 'Pedro');
```



## Recursão

O PHP permite chamada de funções recursivamente. No caso a seguir criaremos uma função para calcular um fatorial de um número.


```
110 function Fatorial($numero){  
111     if($numero == 0 || $numero == 1){  
112         return 1;  
113     }else{  
114         return $numero * Fatorial($numero - 1);  
115     }  
116 }  
117 echo Fatorial(5);  
118 echo Fatorial(7);
```

# Strings

Uma string é uma cadeia de caracteres alfanuméricos. Para declarar uma string podemos utilizar aspas simples " ou aspas duplas "".

```
 $variavel = 'Isto é um teste';  
 $variavel = "Isto é um teste";
```

A diferença é que todo conteúdo contido dentro de aspas duplas é avaliado pelo PHP. Assim, se a string contém uma variável, esta variável será traduzida pelo seu valor.

```
 $fruta = 'maçã';  
16 print "como $fruta"; //resultado 'como maçã'  
17 print 'como $fruta'; //resultado 'como $fruta'
```

## Concatenação

Para concatenar strings, pode-se utilizar o operador "." ou colocar múltiplas variáveis dentro de strings com aspas duplas "", uma vez que seu conteúdo é interpretado.

```
37  $fruta = 'maçã';  
38  echo $fruta.' é a fruta de adão';  
39  echo "{$fruta} é a fruta de adão";
```

O PHP realiza automaticamente a conversão entre tipos. Como neste exemplo de concatenação entre uma string e um número:

```
46  $a = 1234;  
47  
48  echo 'O salário é '. $a . '<br>';  
49  echo "O salário é $a <br>";
```

## Caracteres de escape

Dentro de aspas duplas podemos utilizar controles especiais interpretados diferentemente pelo PHP, que são os caracteres de escape (\). Veja a seguir os mais utilizados:

<code>\n</code>	Nova linha, proporciona quebra de linha.
<code>\r</code>	Retorno de carro.
<code>\t</code>	Tabulação.
<code>\\</code>	Barra invertida é o mesmo que “\”.
<code>\"</code>	Aspas duplas.
<code>\\$</code>	Símbolo de \$.

```
58 echo "seu nome é \"Paulo \".\";
59 echo 'seu nome é "Paulo".';
60 echo 'seu salário é $650,00';
61 echo "seu salário é \$650,00";
```

As funções a seguir formam um grupo cuja característica comum é a manipulação de cadeias de caracteres (strings), como conversões, transformações, entre outras funcionalidades.

## **strtoupper**

Transforma uma string para maiúsculo. Retorna a string com todos os caracteres alfabéticos convertidos para maiúsculo.

```
75 echo strtoupper("Convertendo para maiusculo");
```

## **strtolower**

Transforma uma string para minúsculo. Retorna a string com todos os caracteres alfabéticos convertidos para minúsculo.

```
83 echo strtolower("CONVERTENDO PARA MINUSCULO");
```

## substr

Retorna parte de uma string. Retorna uma porção de conteúdo, começando em início, contendo comprimento em caracteres. Se comprimento for negativo, conta n caracteres antes do final.

```
93  $rest = substr("América", 1);  
    echo "$rest <br>";  
95  $rest = substr("América", 1, 3);  
    echo "$rest <br>";  
97  $rest = substr("América", 0, -1);  
    echo "$rest <br>";  
99  $rest = substr("América", -2);  
    echo "$rest <br>";
```



## str\_pad

Preenche uma string com uma outra string, dentro de um tamanho específico.

```
106 $texto = "The Beatles";  
107 print str_pad($texto, 20) . "<br>";  
108 print str_pad($texto, 20, "*", STR_PAD_LEFT) . "<br>";  
109 print str_pad($texto, 20, "*", STR_PAD_BOTH) . "<br>";  
109 print str_pad($texto, 20, "*", STR_PAD_RIGHT) . "<br>";
```

## str\_repeat

Repete uma string uma certa quantidade de vezes.

```
116 $txt = ".oOOOo.";  
117 print str_repeat($txt, 5) . "<br>";
```



## strlen

Retorna o comprimento de uma string.

```
123     $com = "O Rato roeu a roupa do rei de roma";  
124     print 'O comprimento é: '.strlen($com). "<br>";
```

## str\_replace

Substitui uma string por outra em um dado contexto.

```
130     $rep = "O Rato roeu a roupa do rei de roma";  
131     print str_replace("Rato", "Leão", $rep) . "<br>";
```

## strpos

Encontra a primeira ocorrência de uma string dentro de outra.

```
137 $minha_string = "O Rato roeu a roupa do rei de roma";
138 $encontrar = 'roupa';
139 $posicao = strpos($minha_string, $encontrar);
140 if($posicao) {
141     echo "String encontrada na posição $posicao";
142 }else{
143     echo "String não encontrada";
144 }
```

## sha1

Função utilizada para criptografar senhas, pode ser utilizada de forma encadeada, retorna 40 caracteres hexadecimais.

```
151 $senha = '123abc';  
152 echo 'Criptografada (40 caracteres): ' . sha1($senha) . "<br>";
```

## md5

Função utilizada para criptografar senhas, pode ser utilizada de forma encadeada, retorna 32 caracteres hexadecimais.

```
159 echo 'Criptografada (32 caracteres): ' . md5($senha) . "<hr>";
```

# Números

O PHP nos permite facilmente formatar números, definindo quantidade de casas decimais, o separador de decimais, separador de milhares utilizando a função **number\_format( )**.

Define quantidade de casas decimais para 02

```
6 $dec = 25.87499;  
7 echo number_format($dec, 2) . "<br>;
```

Define o caractere utilizado para separar decimais (vírgula)

```
13 $n = 8.89;  
14 echo number_format($n, 2, ",", NULL) . "<br>;
```

Define o caractere utilizado para separar milhares (ponto)

```
19 $s = 32578.97;  
20 echo number_format($s, 2, ",", ".") . "<br>;
```

Retorna o valor da constante matemática PI

```
25 echo pi() . "<br>;
```

Retorna a raiz quadrada de um número

```
30 echo sqrt(81) . "<br>;
```

Retorna a potência de um número pow(base, potencia)

```
35 echo pow(2, 3) . "<br>;
```

Retorna um número randômico dentro de um intervalo definido

```
40 echo rand(1, 10) . "<br>;
```

Arredonda número decimal para o primeiro número inteiro acima

```
45 echo ceil(1.01) . "<br>;
```

Converte dado binário em hexadecimal

```
50 echo bin2hex(001) . "<br>;
```

Converte dado binário em decimal

```
55 echo bindec(1001) . "<br>;
```

Converte hexadecimal em dado binário

```
60 echo hex2bin("1C") . "<br>;
```

Converte hexadecimal em decimal

```
65 echo hexdec("B45") . "<br>;
```

Converte decimal em dado binário

```
70 echo decbin(9) . "<br>;
```

Converte decimal em octo

```
75 echo decoct(9) . "<br>;
```

Converte decimal em hexadecimal

```
80 echo dechex(1010) . "<br>;
```