

# LÓGICA DE PROGRAMAÇÃO

Professor Jefferson Chaves  
[jefferson.chaves@ifc-araquari.edu.br](mailto:jefferson.chaves@ifc-araquari.edu.br)



Técnico Integrado em  
**INFORMÁTICA**

- Definição de funções;
  - Quais problemas as funções resolvem;
- Anatomia de uma função;
- Passagens de parâmetros;
- Retorno de funções;



# LÓGICA DE PROGRAMAÇÃO | FUNÇÕES OU PROCEDIMENTOS

Livro desenvolvendo websites com PHP  
Páginas 75 a 87

- **Funções** são blocos de código que podem realizar **qualquer tipo** de tarefa;
- Podemos criar **funções** para somar dois números, ou mesmo verificar se a data de aniversário foi digitada corretamente.

# LÓGICA DE PROGRAMAÇÃO | FUNÇÕES

- Mesmo sem saber, já usamos algumas **funções**. Perceba a semelhança entre as anatomias das funções abaixo:
  - **print\_r( );**
  - **var\_dump( );**
  - **array( );**
  - **sort( );**

- O **modelo** para a criação de uma função é a seguinte:

```
function nomeDaFuncao( ) {  
    //instruções  
}
```

# LÓGICA DE PROGRAMAÇÃO | FUNÇÕES

- Uma função deve ter a palavra chave **function** seguida do nome da função;
- As instruções que uma função executam devem estar delimitadas em um bloco por meio do uso das chaves **{ };**
- O nome de uma função deve seguir as mesmas regras dos nomes de variáveis;
- É necessário que o nome de uma função deve refletir o que a função faz;
- Use os verbos **set, get, is, [...]**;

# LÓGICA DE PROGRAMAÇÃO | FUNÇÕES

- Funções são educadas: elas só são executadas quando forem **chamadas**.
- A função será executada por uma chamada para a função;



- A função será executada por uma chamada para a função;

```
//definição da função
function digaOla( ) {
    echo "Olá";
}
```

**digaOla( );** //chamada para a função

- Funções são educadas: elas só são executadas quando forem **chamadas**.
- A chamada para função deve ser feita por meio de seu nome seguido dos parenteses vazios ou com parâmetros quando houver.





**LÓGICA DE  
PROGRAMAÇÃO**

**PARÂMETROS  
DE UMA FUNÇÃO**

- Uma função pode receber alguns valores que serão usados por instruções da função;
- Eses valores recebem o nome de **parâmetros** ou **argumentos** da função;
- Eses valores devem estar definidos na estrutura da função e devem ser informados na chamada da função;

- Passagem de parâmetros por valor:
  - Os parâmetros que a função precisa deve estar contido entre os parenteses em frente ao nome da função;

```
function validarCPF( $cpf ) {  
    //código com a lógica para validar um CPF
```

```
}
```

```
validarCPF( '099 988 355 - 77' );
```

- Quando uma função possui um parâmetro em sua definição, ao chamá-la, é **obrigatório** informar o valor para esse parâmetro.
- Contudo é possível informar um valor padrão: neste caso o parâmetro torna-se opcional.

- Passagem de parâmetros por valor:
  - Os parâmetros que a função precisa deve estar contido entre os parenteses em frente ao nome da função;

```
function saudacaoUsuario( $nome = “anônimo” ) {
```

```
    echo $nome;
```

```
}
```

```
saudacaoUsuario();
```

# LÓGICA DE PROGRAMAÇÃO

# PARÂMETROS POR REFERÊNCIA

Jefferson de Oliveira Chaves

- Passagem de parâmetros por referência:
  - Os parâmetros que a função precisa deve estar contido entre os parenteses em frente ao nome da função;

```
function adicionarValor($numeros){  
    array_push($numeros, 10);  
}  
  
$numeros = [ 1, 2, 3, 4 ];  
adicionarValor($numeros);  
print_r($numeros);
```

- Ao passarmos um parâmetro a uma função, uma cópia(valor) é passada para a função;
- Isso significa que qualquer alteração que seja efetuada nesse parâmetro dentro da função(escopo local), não afetará seu ambiente externo(escopo global).

- Porém, existem casos em que precisamos que as alterações sejam mantidas no ambiente externo(escopo global),
- Para esses casos, devemos passar o parâmetro como referência, utilizando **&** na frente do parâmetro no momento em que definimos a função.

```
<?php  
function adicionarValor(&$numeros){  
    array_push($numeros, 10);  
}  
  
$numeros = [ 1, 2, 3, 4 ];  
adicionarValor($numeros);  
print_r($numeros);
```

# LÓGICA DE PROGRAMAÇÃO

**QUAIS PROBLEMAS  
UMA FUNÇÃO  
RESOLVE?**

# LÓGICA DE PROGRAMAÇÃO | FUNÇÕES

```
1 <?php
2
3     $nome = "Pluto";
4     $peso = 23;
5
6     if ($peso > 20 ) {
7         echo "$nome late: WOOF WOOF";
8     } else {
9         echo "$nome late: au au au";
10    }
11
12
13    $nome = "Bobby";
14    $peso = 13;
15
16    if ($peso > 20 ) {
17        echo "$nome late: WOOF WOOF";
18    } else {
19        echo "$nome late: au au au";
20    }
21
22
23    $nome = "Spike";
24    $peso = 20;
25
26    if ($peso > 20 ) {
27        echo "$nome late: WOOF WOOF";
28    } else {
29        echo "$nome late: au au au";
30    }
```



Código está  
errado?

# LÓGICA DE PROGRAMAÇÃO | FUNÇÕES

```
1 <?php
2
3     $nome = "Pluto";
4     $peso = 23;
5
6     if ($peso > 20 ) {
7         echo "$nome late: WOOF WOOF";
8     } else {
9         echo "$nome late: au au au";
10    }
11
12
13    $nome = "Bobby";
14    $peso = 13;
15
16    if ($peso > 20 ) {
17        echo "$nome late: WOOF WOOF";
18    } else {
19        echo "$nome late: au au au";
20    }
21
22
23    $nome = "Spike";
24    $peso = 20;
25
26    if ($peso > 20 ) {
27        echo "$nome late: WOOF WOOF";
28    } else {
29        echo "$nome late: au au au";
30    }
```



**Se eu  
pudesse  
reusar o  
código**

# LÓGICA DE PROGRAMAÇÃO

## FUNÇÕES

```
function latir ($nome, $peso)
```

```
{
```

```
// instruções
```

```
}
```

corpo da função

define-se os parâmetros  
da função



# LÓGICA DE PROGRAMAÇÃO

# FUNÇÕES

```
function latir ($nome, $peso) {  
    if ($peso > 20) {  
        echo "$nome late: WOOF WOOF";  
    } else {  
        echo "$nome late: au au au";  
    }  
}
```



# LÓGICA DE PROGRAMAÇÃO

# FUNÇÕES

```
latir ("rover", 23);
```

```
function latir ($nome, $peso) {  
    if ($peso > 20) {  
        echo ("$nome late WOOF WOOF");  
    } else {  
        echo ("$nome late woof woof");  
    }  
}
```

Estamos passando 2 parâmetros: nome e peso.  
Quando chamamos a função latir, os argumentos  
são atribuídos aos nomes dos parâmetros

# LÓGICA DE PROGRAMAÇÃO | FUNÇÕES

```
function latir ($nome, $peso) {  
    if ($peso > 20) {  
        echo (" $nome late WOOF WOOF");  
    } else {  
        echo (" $nome late au au au");  
    }  
}
```

```
latir ("rover", 23);  
latir ("spot", 13);  
latir ("spike", 53);  
latir ("lady", 17);
```





**LÓGICA DE  
PROGRAMAÇÃO**

**FUNÇÕES COM  
RETORNO**

Jefferson de Oliveira Chaves

# LÓGICA DE PROGRAMAÇÃO | FUNÇÕES

- Uma função pode além de apenas exibir o resultado, retornar um valor para quem a chamou;
- O retorno deve ser indicado pelo uso do comando **return**.

## LÓGICA DE PROGRAMAÇÃO

## FUNÇÕES COM RETORNO

```
function calculaAumento ( $salario ) {  
    $novoSalario = $salario + ($salario * 0.10);  
    return $novoSalario;  
}  
  
$novoSalario = calculaAumento(1500);  
echo $novoSalario;
```

# LÓGICA DE PROGRAMAÇÃO

# ESCOPO DE VARIÁVEIS

Jefferson de Oliveira Chaves

- Você deve lembrar o que é escopo, e o que é o **escopo de variáveis**.
- O **escopo** é o conjunto de regras que determinam o uso e a validade de variáveis nas diversas partes da aplicação.

## LÓGICA DE PROGAMAÇÃO

## ESCOPO DE VARIÁVEIS

```
aumento = 8%;
```

```
funcao calcularAumento ( salario ){
```

```
    salario = salario + aumento;
```

```
    exibe salario;
```

```
}
```

```
calcularAumento ( 1000.00 );
```

Vai gerar um erro: a variável **aumento** não existe;

## LÓGICA DE PROGAMAÇÃO

## ESCOPO DE VARIÁVEIS

```
aumento = 8%;
```

```
funcao calcularAumento( salario ){
```

```
    global aumento;
```

```
    salario = salario + aumento;
```

```
    exibe salario;
```

```
}
```

```
calcularAumento( 1000.00 );
```

Solução elegante:  
**Usar o atributo global**  
para a variável.

- O verdadeiro **PODER**, de uma linguagem de programação começa com o uso funções;
- Podemos programar nossas próprias funções ou usar funções internas (built-in) no **PHP**;
- No PHP são mais de **1000** funções!
- Podemos consultar essas funções no **php.net**

# **EXERCÍCIOS | FUNÇÕES**

Jefferson de Oliveira Chaves

1. Faça um uma função chamada `converteMoeda( $valor, $moeda )`, que converta um valor em reais para dólar e euro;

**Retorne um vetor com os valores convertidos;**

2. Implemente um algoritmo que modularizado que a partir de um vetor de **N** inteiros, possibilite:
- a) A digitação de valores no vetor;
  - b) Imprimir o somatório de seus elementos;
  - c) Imprimir a média de seus elementos;
  - d) Imprimir a média e o somatório
  - e) Substituir por zero todos os valores negativos e imprimir a média;
  - f) Substituir por zero todos os valores repetidos e imprimir a média e o somatório;