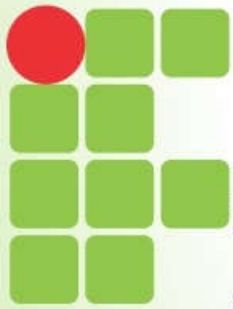


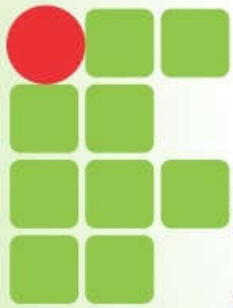
INSTITUTO FEDERAL
DO NORTE DE MINAS GERAIS

Campus Arinos

Linguagens e técnicas de programação II

Prof.: Wilson Prates de Oliveira

INSTITUTO FEDERAL INSTITUTO FEDERAL INSTITUTO FEDERAL INSTITUTO FEDERAL INSTITUTO FEDERAL INSTITUTO FEDERAL INSTITUTO FEDERAL



INSTITUTO FEDERAL
DO NORTE DE MINAS GERAIS

Campus Arinos

Aula 6: funções

Código da turma: 7ccvn2j

Prof.: Wilson Prates de Oliveira

INSTITUTO FEDERAL INSTITUTO FEDERAL INSTITUTO FEDERAL INSTITUTO FEDERAL INSTITUTO FEDERAL INSTITUTO FEDERAL INSTITUTO FEDERAL

Aula 6

- Tema: funções, subprogramas ou rotinas
- Objetivos:
  - Apresentar algumas das funções internas fornecidas pelo PHP;
  - Explicar a motivação para usar o recurso da criação de funções;

Aula 6

- Objetivos:
  - Ensinar a sintaxe da declaração de funções de usuário para que o usuário/desenvolvedor possa criar suas próprias funções;

Conteúdo da aula

- Funções
  - Internas do PHP
  - Definidas pelo usuário
    - ✓ Parâmetros da Função
      - ✓ Passagem de parâmetro
    - ✓ Retornando valor
    - ✓ Chamada de funções dinâmica
    - ✓ Especificação de tipo

Conteúdo da aula

- Inclusão de arquivos externos
- Exercícios
- Referências bibliográficas

Instruções para a aula

I. Descompacte o arquivo **aula 6.rar** no diretório aulas criado na aula 1;

II. Seguindo a padrão das aulas anteriores, crie uma entrada, link, com o texto - Aula 6: funções, subprogramas ou rotinas - no arquivo index.html da pasta aulas, apontando para a pasta **aula 6**;

III. Baixe o pdf da aula no local indicado no link da página index.html desta aula;

Instruções para a aula

IV. Para cada trecho de código **exemplo** **identificado** por **número**, crie um script PHP para testá-lo;

V. Para cada exercício de fixação identificado por número, escreva um script PHP com a solução sempre que for necessário; caso não seja necessário o script PHP, escreva a solução em outro formato de arquivo: html, txt, docx, pdf, etc. e salve na pasta **files**;

Instruções para a aula

VI. Teste todos scripts criados acessando links na página index.html da aula. **Siga o padrão** utilizado nas aulas anteriores: exemplo 1 (exemplo1.php), exercício de fixação 1 (exe-fix1.php) e exercício 1 (exe1.php).

VII. A localização dos arquivos criados deve estar de acordo com a amostra da página **index.html** da aula, exceto se estiver usando uma estrutura de arquivos diferente.

Funções

- Definição:

  uma função é um pedaço ou **trecho de código** que pode **receber uma ou mais entradas** na forma de **parâmetros**, **realizar** algum **processamento** e **retornar** um **valor**.

  - Em PHP, as funções são similar a outras linguagens de programação.

Funções internas do PHP

- Variedade de funções
  - A linguagem PHP possui mais de **1000 funções internas** divididas em várias **categorias**.
  - Estas funções podem ser chamadas diretamente, de dentro de um script, para realizar diferentes tarefas.

Funções internas do PHP

- Algumas categorias de funções internas:
  - **String** - manipular de strings;
  - **Array** – acessar e manipular de arrays;
  - **FileSystem** – acessar e manipular o sistema de arquivos;
  - **Mail** – enviar e-mail a partir de script;
  - **MySQLi** – acessar base de dados MYQL;

Funções internas do PHP

- Para ver a lista completa das funções e suas categorias visite:
  - ✓ https://www.w3schools.com/php/php_ref_overview.asp , ou
  - ✓ consulte o manual da linguagem PHP.

Funções definidas pelo usuário

- Além de fornecer funções internas, PHP permite que o usuário defina ou crie funções de acordo com suas necessidades.
  - As principais motivações para criar sua próprias funções são:
    - ✓ evitar a repetição de código; e
    - ✓ reaproveitar o código.

Funções definidas pelo usuário

- A declaração de uma função
  - começa com a palavra chave ***function***;
  - seguida do **nome** da função - que deve inicializar com **letra** ou **sublinhado** -;
  - seguido do **corpo** da função que deve ser delimitado por **chaves** (**{}**).

Funções definidas pelo usuário

- Sintaxe da declaração de uma função:

```
function nomeDaFuncao(< lista de parâmetros >) {
    < Código da função >
}
```

Funções definidas pelo usuário

- Exemplo 1:

```
<?php
   function escrevaMensagem() {
        echo "Alô mundo!";
   }
   escrevaMensagem();
?>
```

Funções definidas pelo usuário

- Parâmetros da Função
  - é a forma de passar informações para a função.
  - é como uma variável.
  - são especificados após o nome da função dentro de parêntesis **( )**.
  - devem ser passados quantos forem necessários.
  - devem ser separados por vírgula (,), caso existam mais de um.

Parâmetros da Função

- Exemplo 2 - função com dois parâmetros:

```
<?php
   function sobrenome($nome, $ano) {
        echo "$nome Silva. nasceu em $ano <br>";
   }
   sobrenome("José","1975");
   sobrenome("Joaquim","1978");
   sobrenome("Maria","1983");
?>
```

Parâmetros da Função

- Passagem de parâmetros
  - A linguagem PHP permite especificar como os parâmetros serão passados para as funções:
    - ✓ **passagem por referência -** a própria variável é passada para a função;
    - ✓ **passagem por valor -** uma **cópia** da variável é passada para a função;

Passagem de parâmetros

- Exemplo 3 – passagem de parâmetros por valor e por referência:

```
<?php
   function passagemPorValor($num) {
     $num += 5;
   }
   function passagemPorReferencia(&$num) {
     $num += 6;
   }
```

Passagem de parâmetros

```
    $numOriginal = 10;
    passagemPorValor( $numOriginal );
    echo "Valor original é $numOriginal<br />";
    passagemPorReferencia( $numOriginal );
    echo "Valor original é  $numOriginal<br />";
?>
```

Passagem de parâmetros

- Parâmetros com valor ***default***
  - É possível definir um **valor padrão** ou ***default*** para os parâmetros da função.
  - O valor ***default*** será **usado** caso o chamador da função **não passe** um **valor como argumento** durante a chamada,.

Passagem de parâmetros

- Exemplo 4 **-** valor ***default***:

```
<?php
   function meuPrint($param = "default"){
           print $param;
   }
   meuPrint("Este é um teste.<br>");
   meuPrint();
?>
```

Funções definidas pelo usuário

- Retornando valor

  Uma função pode retornar um valor, um objeto ou um conjunto de valores usando o comando ***return*** seguido do valor, objeto ou array de valores.

  - A chamada de ***return*** interrompe a execução da função e retorna o valor e o controle para o código que chamou a função.

Retornando valor

- Exemplo 5:

```
<?php
   function funcaoSoma($num1, $num2) {
            $sum = $num1 + $num2;
            return $sum;
   }
   $valorRetornado = funcaoSoma(10, 20);
   echo "O valor da soma é: $valorRetornado";
?>
```

Funções definidas pelo usuário

- Chamada de função dinâmica

  Em PHP, é possível atribuir nomes de funções como **strings** à variáveis e, em seguida, tratar essas variáveis como se fossem as próprias funções.

Chamada de função dinâmica

- Exemplo 6:

```
<?php
   function funcaoSoma($num1, $num2) {
      return ($num1 + $num2);
   }
   $soma = "funcaoSoma";
   $valorRetornado = $soma(10, 20);
   echo "o valor da soma é: $valorRetornado";
?>
```

Funções definidas pelo usuário

- Especificação de tipo
  - Tipo do parâmetro
  - Tipo do valor retornado

Especificação de tipo

- Tipo do parâmetro
  - A partir do **PHP 7** é possível especificar o **tipo** de dado que será passado como **argumento** ao declarar uma função.
  - A **passagem** do tipo esperado é forçada inserindo a declaração "**declare(strict_types=1);**" na primeira linha do script.

Tipo do parâmetro

- Exemplo 7:

```
<?php
   /* estritamente necessário */
   declare(strict_types=1);
   function funcaoSoma(int $num1, int $num2) {
          return ($num1 + $num2);
   }
   $soma = "funcaoSoma";
   $valorRetornado = $soma(10, 20);
   //$valorRetornado = $soma(10, "20");
   echo "o valor da soma é: $valorRetornado";
?>
```

Especificação de tipo

- Tipo do valor retornado
  - A partir do **PHP 7**, é possível declarar o tipo do valor retornado por uma função.
  - Como ocorre com a especificação do tipo de parâmetro da função, ao ativar o requisito **estrito**, será lançado um "***Erro Fatal***" caso ocorra uma incompatibilidade entre tipos.

Tipo do valor retornado

- Exemplo 8:

```
<?php
   /* estritamente necessário */
   declare(strict_types=1);
   function funcaoSomaInt(int $num1, int $num2):
   int {
           return ($num1 + $num2);
   }
   $soma = "funcaoSomaInt";
   $valorRetornado = $soma(10.0, 20);
   echo "o valor da soma é: $valorRetornado";
?>
```

Inclusão de arquivos externos

- Motivação

  A inclusão de arquivos externos é útil quando precisamos utilizar o mesmo código PHP, HTML ou texto em várias páginas de um site.

Inclusão de arquivos externos

- Instruções **include** e **require**
  - inserem o texto, código ou marcação existente no arquivo especificado no arquivo que contém a instrução.
  - são idênticas, exceto quando ocorre uma **falha**.

Instruções **include** e **require**

- Em caso de falha:
  - **require**
    - ✓ produz o **fatal erro E_COMPILE_ERROR** e **interrompe** o script;
  - **include**
    - ✓ produz o alerta ou ***warnning***, **E_WARNING**, e **continua** a **execução** do script.

Instruções **include** e **require**

- Exemplo 9:

```
<?php
   function funcaoSomaInt(int $num1,
   int $num2): int {
          return ($num1 + $num2);
     }
?>
```

Inclusão de arquivos externos

- Exemplo 10:

```
<?php
    declare(strict_types=1);
    include "exemplo9.php";
    $soma = "funcaoSomaInt";
    $valorRetornado = $soma(10, 20);
    echo "o valor da soma é: $valorRetornado";
?>
```

Inclusão de arquivos externos

- Exercício de fixação 1

I. crie , no diretório **src**, um diretório chamado **util**.

II. crie, no diretório **util,** o arquivo **funcoes.php**.

III. insira o código do exemplo 9 em **funcoes.php**.

IV. crie, no diretório src/exercícios/, o arquivo **exe-fix1.php**.

V. insira o código do exemplo 10 em **exe-fix1.php**.

VI. crie um link – Teste de inclusão de arquivo externo - na seção exercícios de fixação para testar os scripts criados.

Exercícios

1) Na matemática, o fatorial de um número natural **n**, representado por **n!**, é o produto de todos os inteiros positivos menores ou iguais a n. A notação **n!** foi introduzida por *Christian Kramp* em 1808.
   a) Escreva uma função fatorial($n) que calcule e retorne o valor do fatorial de n. A função deve tratar os casos: n negativo e n igual a 0. Teste a função com valores de n maiores que 10;
   b) Crie um formulário HTML, em um arquivo **exe1.html**, para que o usuário entre com os dados.
   c) Mostre a saída, dentro de um elemento HTML de sua preferência , no script **exe1.php**,

Exercícios

2) A sequência de ***Fibonacci*** é definida recursivamente pela fórmula: $f_n = f_{n-1} + f_{n-2}$, e valores iniciais: $f_1 = 1 \; e f_2 = 1$.
   a) Escreva uma função para calcular e mostrar os 10 primeiros termos da sequência.
   b) Crie um formulário HTML, em um arquivo **exe2.html**, para que o usuário entre com os dados.
   c) Mostra a saída, dentro de um elemento HTML de sua preferência ,no script **exe2.exe**;

Exercícios

3) Quando devemos usar os comandos include ou require e qual a diferença entre eles?

4) O que ocorre quando especificamos o tipo do parâmetro que deve ser passado para uma função e a chamamos usando um valor do tipo diferente do especificado.

5) Remova as funções criadas, nos exercícios 1 e 2, e as inclua no arquivo **funcoes.php**. Insira o script funcoes.php, usando require ou include, nos scripts exe1.php e exe2.php. **Observação**: ***as localizações não devem ser alteradas***.

Referências bibliográficas

1. https://www.w3schools.com/php/php_functions.asp
2. https://www.tutorialspoint.com/php/php_functions.htm
3. SEBESTA, Robert W., Conceitos de Linguagens de Programação, Editora: Bookman. 9ª Edição, 2010.

Fim da aula!