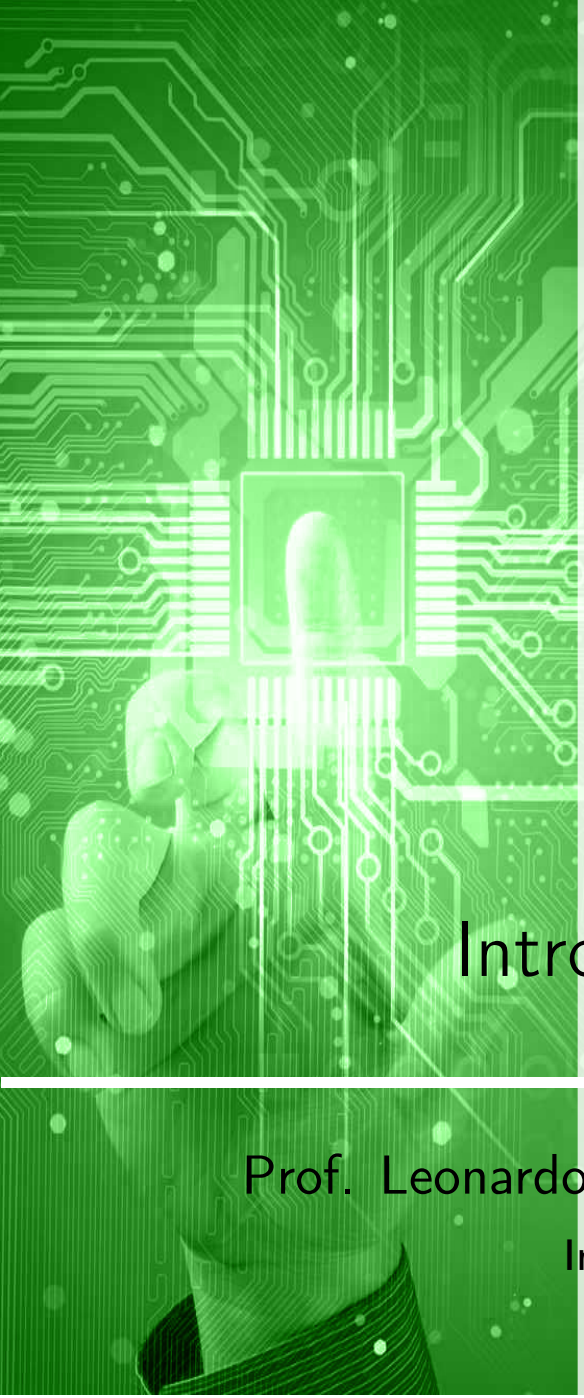

INSTITUTO FEDERAL
Sudeste de Minas Gerais
Campus Manhuaçu

Desenvolvimento Web III
Introdução geral à linguagem PHP

Prof. Leonardo C. R. Soares - *leonardo.soares@ifsudestemg.edu.br*
Instituto Federal do Sudeste de Minas Gerais

15 de outubro de 2024

Desenvolvimento Web III

Carga horária

- 75 horas;
- 5 aulas por semana;
- Quarta (08:00 - 10:15) e Sexta (08:45 - 10:15).

Desenvolvimento Web III

Carga horária

- 75 horas;
- 5 aulas por semana;
- Quarta (08:00 - 10:15) e Sexta (08:45 - 10:15).

Ementa

Introdução ao desenvolvimento Web com PHP; Fundamentos da linguagem; Programação Orientada a Objetos (POO) em PHP; Tratamento de exceções; Manipulação de bases de dados utilizando o PHP Data Objects (PDO); Transações em bases de dados; Manipulação de dados utilizando o JavaScript Object Notation (JSON); Padrão Model-View-Controller (MVC) e frameworks PHP.

Apresentação da disciplina

Bibliografia básica

- MILANI, A. Construindo Aplicações Web com PHP e MySQL. São Paulo, Novatec, 2010. 336p. ISBN: 9788575222195.
- DALL'OGLIO, P. PHP: Programando com Orientação a Objetos. 3ed. São Paulo, Novatec, 2015. 552p. ISBN: 9788575224656.
- SANDERS, W. Aprendendo Padrões de Projeto em PHP. Editora Novatec. 1ed. 2013. 368p. ISBN-13 9788575223437.

Apresentação da disciplina

Bibliografia complementar

- LOCKHART, J. PHP Moderno. São Paulo, Novatec, 2015. 296p. ISBN: 9788575224281.
- SICA, C. PHP Orientado a Objetos. Editora Ciência Moderna. 1ed. 2006. 216p. ISBN: 9788573935530.
- MEDINA, M.; FERTIG, C. Algoritmos e Programação - Teoria e Prática. São Paulo, Novatec, 2005. 384p. ISBN: 857522073X.
- SILVA, M. S. JavaScript - Guia do Programador. São Paulo, Novatec, 2010. 608p. ISBN: 9788575222485.
- SAMPAIO, C. Javascript de Cabo a Rabo. Rio de Janeiro, Ciência Moderna, 2015. 352p. ISBN: 9788539906581.

Apresentação da disciplina

Avaliações

Nossa disciplina possuirá três atividades avaliativas, todas, individuais e sem consulta. As duas primeiras serão teóricas e a última prática.

- 04/12/2024 - T - 3pts;
- 07/02/2025 - T - 3pts;
- 12/03/2025 - P - 4pts.

Apresentação da disciplina

Avaliações

Nossa disciplina possuirá três atividades avaliativas, todas, individuais e sem consulta. As duas primeiras serão teóricas e a última prática.

- 04/12/2024 - T - 3pts;
- 07/02/2025 - T - 3pts;
- 12/03/2025 - P - 4pts.

O plano de ensino está disponível no SIGAA. Separe um tempinho e leia.

Apresentação da disciplina

Avaliações

Nossa disciplina possuirá três atividades avaliativas, todas, individuais e sem consulta. As duas primeiras serão teóricas e a última prática.

- 04/12/2024 - T - 3pts;
- 07/02/2025 - T - 3pts;
- 12/03/2025 - P - 4pts.

O plano de ensino está disponível no SIGAA. Separe um tempinho e leia.
Todo professor possui 4 horas dedicada ao atendimento individualizado aos alunos. Use-as.

Apresentação da disciplina

Avaliações

Nossa disciplina possuirá três atividades avaliativas, todas, individuais e sem consulta. As duas primeiras serão teóricas e a última prática.

- 04/12/2024 - T - 3pts;
- 07/02/2025 - T - 3pts;
- 12/03/2025 - P - 4pts.

O plano de ensino está disponível no SIGAA. Separe um tempinho e leia.
Todo professor possui 4 horas dedicada ao atendimento individualizado aos alunos. Use-as.

Alguma dúvida?

Introdução

Definição

De acordo com o manual do PHP (disponível em `http://php.net`) **PHP (acrônimo recursivo para PHP: Hypertext Preprocessor)** é uma linguagem de script de código aberto amplamente utilizada que é especialmente adequada para desenvolvimento web e pode ser incorporada em páginas HTML.

Breve histórico

- Criado em 1994 por **Rasmys Lerdorf** utilizando a linguagem de programação C;
- Projeto seminal foi nomeado como *Pesonal Home Page Tools (ou PHP Tools)*;
- Código fonte disponibilizado ao público em junho de 1995;
- Em setembro de 1995 o nome PHP foi temporariamente abondonado e adotado o nome *tools as FI (Forms Interpreter)*;
- Em abril de 1996 a versão 2.0 foi lançado sob o nome PHP/FI;
- Em junho de 1998, o PHP 3.0 foi lançado. Esta versão é a primeira com similariedades a versão atual. O nome **PHP** foi adotado novamente agora com o significado *Hypertext Preprocessor*;
- Atualmente, o PHP encontra-se na versão 8.3.

O que o PHP pode fazer:

- **Server-side scripting**: Desenvolvimento de sistemas web. Para utilizarmos o PHP desta forma, precisaremos do PHP parser (CGI ou módulo servidor), um servidor web e um browser web;

O que o PHP pode fazer:

- **Server-side scripting**: Desenvolvimento de sistemas web. Para utilizarmos o PHP desta forma, precisaremos do PHP parser (CGI ou módulo servidor), um servidor web e um browser web;
- **Command line scripting**: Scripts desenvolvidos para execução sem a utilização de servidores *web* ou *web browsers*. Utilizado na automatização de tarefas utilizando *cron* ou *gerenciador de tarefas*;

O que o PHP pode fazer:

- **Server-side scripting**: Desenvolvimento de sistemas web. Para utilizarmos o PHP desta forma, precisaremos do PHP parser (CGI ou módulo servidor), um servidor web e um browser web;
- **Command line scripting**: Scripts desenvolvidos para execução sem a utilização de servidores *web* ou *web browsers*. Utilizado na automatização de tarefas utilizando *cron* ou *gerenciador de tarefas*;
- **Aplicações desktop**: Embora, provavelmente não seja a melhor linguagem para isso, podemos utilizar PHP para escrever aplicações desktop com interface gráfica em PHP (PHP-GTK).

Ambiente de desenvolvimento

Em nossa disciplina, focaremos apenas em Server-side scripting. Para que você possa fazer os exemplos (e exercícios) em seu computador pessoal, será necessário instalar o Servidor Web e o PHP (sem mencionar o MySQL). Existem algumas aplicações que facilitam este processo de instalação, entre elas (se você for usar uma), recomendamos que você instale o XAMPP.

Embora o processo de instalação seja completamente intuitivo, caso tenha dificuldades, assista a este vídeo.

Como funciona uma aplicação web em PHP

- O código PHP pode ser escrito utilizando-se qualquer editor de texto não formatado (ou uma IDE como VS Code ou NetBeans). O arquivo **deve** ser salvo com a extensão .php dentro da pasta virtual.

Como funciona uma aplicação web em PHP

- O código PHP pode ser escrito utilizando-se qualquer editor de texto não formatado (ou uma IDE como VS Code ou NetBeans). O arquivo **deve** ser salvo com a extensão .php dentro da pasta virtual.
- Para acessar a aplicação web, deveremos navegar até o servidor e solicitar o arquivo utilizando um *web browser*. Para aplicações locais, pode-se substituir o endereço do servidor pelo endereço de loopback.

Como funciona uma aplicação web em PHP

- O código PHP pode ser escrito utilizando-se qualquer editor de texto não formatado (ou uma IDE como VS Code ou NetBeans). O arquivo **deve** ser salvo com a extensão .php dentro da pasta virtual.
- Para acessar a aplicação web, deveremos navegar até o servidor e solicitar o arquivo utilizando um *web browser*. Para aplicações locais, pode-se substituir o endereço do servidor pelo endereço de loopback.
- O arquivo PHP será interpretado pelo servidor web e o resultado será enviado ao *browser* cliente.

Como funciona uma aplicação web em PHP

- O código PHP pode ser escrito utilizando-se qualquer editor de texto não formatado (ou uma IDE como VS Code ou NetBeans). O arquivo **deve** ser salvo com a extensão .php dentro da pasta virtual.
- Para acessar a aplicação web, deveremos navegar até o servidor e solicitar o arquivo utilizando um *web browser*. Para aplicações locais, pode-se substituir o endereço do servidor pelo endereço de loopback.
- O arquivo PHP será interpretado pelo servidor web e o resultado será enviado ao *browser* cliente.

Vejamos na prática.

Delimitadores de código

O código PHP é embutido em documentos HTML. Para identificar quais trechos de texto são códigos em PHP, devemos demarcá-los utilizando delimitadores de códigos. Os seguintes delimitadores são válidos em PHP:

<?php ... ?>
<? ... ?>
<script language='php'> ... </script>
<% ... %>

Case sensitivity

As palavras reservadas da linguagem (*keywords*) como echo, while, class etc, bem como os nomes de funções e classes definidas pelo usuário são **case-insensitive**, ou seja, não diferenciam maiúsculas de minúsculas. As linhas abaixo são equivalentes:

```
echo("hello, world");
ECHO("hello, world");
EcHo("hello, world");
```

Case sensitivity

As palavras reservadas da linguagem (*keywords*) como echo, while, class etc, bem como os nomes de funções e classes definidas pelo usuário são **case-insensitive**, ou seja, não diferenciam maiúsculas de minúsculas. As linhas abaixo são equivalentes:

echo("hello, world");
ECHO("hello, world");
EcHo("hello, world");

Variáveis, por outro lado, são **case-sensitive**, ou seja, $name, $NAME, and $NaME são três variáveis diferentes.

Statements

- Uma *statement* é uma coleção de código PHP que executa alguma coisa. Pode ser uma simples atribuição de valor a variável ou uma complexa estrutura de repetição;
- PHP utiliza ponto-e-vígurla para separar statements simples;
- PHP utiliza { ... } para delimitar blocos de código;

```
$a = 1;
$name = "Elphaba";
$b = $a / 25.0;
if ($a == $b) { echo "Rhyme And Reason"; }
```

Statements

- Uma *statement* é uma coleção de código PHP que executa alguma coisa. Pode ser uma simples atribuição de valor a variável ou uma complexa estrutura de repetição;
- PHP utiliza ponto-e-vígurla para separar statements simples;
- PHP utiliza { ... } para delimitar blocos de código;

```
$a = 1;
$name = "Elphaba";
$b = $a / 25.0;
if ($a == $b) { echo "Rhyme And Reason"; }
```

- O ponto-e-vírgula antes do final de bloco (}) é obrigatório. Antes do delimitador de fim de código (?>), é opcional.

Comentários

Comentários são utilizados para fornecer informações úteis a quem estiver lendo o código.
Em PHP podemos utilizar comentários do tipo *shell-style* (#) e *C++ Comments* (//). Após estes caracteres, tudo o que for digitado será considerado comentário até o final da linha.

```
$a = 1; // A variável a recebeu o valor 1
$b = 2; # A variável b recebeu o valor 2
```

Variáveis

Identificadores de variáveis em PHP são *case-sensitive*, iniciados pelo caracter dollar ($). O primeiro caracter após o $ não pode ser um número. Além disso, identificadores de variáveis não podem possuir espaços ou caracteres especiais (exceto o underline).

Variáveis

Identificadores de variáveis em PHP são *case-sensitive*, iniciados pelo caracter dollar ($). O primeiro caracter após o $ não pode ser um número. Além disso, identificadores de variáveis não podem possuir espaços ou caracteres especiais (exceto o underline).

Exemplos de variáveis válidas:

$bill
$head_count
$I_HEART_PHP
$_underscore

Variáveis

Identificadores de variáveis em PHP são *case-sensitive*, iniciados pelo caracter dollar ($). O primeiro caracter após o $ não pode ser um número. Além disso, identificadores de variáveis não podem possuir espaços ou caracteres especiais (exceto o underline).

Exemplos de variáveis válidas:

$bill
$head_count
$I_HEART_PHP
$_underscore

Exemplos de variáveis inválidas:

$not valid
$#
$3wa

Variáveis

PHP é uma linguagem fracamente tipada. Não há um comando específico para se declarar variáveis e o tipo da variável é associado automaticamente dependendo de seu valor. Como os dados não são definidos de uma maneira estrita, podemos fazer coisas como somar *strings* em inteiros sem erros.

Variáveis

PHP é uma linguagem fracamente tipada. Não há um comando específico para se declarar variáveis e o tipo da variável é associado automaticamente dependendo de seu valor. Como os dados não são definidos de uma maneira estrita, podemos fazer coisas como somar *strings* em inteiros sem erros.

A partir do PHP 7, declarações de tipo foram adicionadas, fornecendo uma opção para definirmos o tipo esperado quando declaramos uma função. Ao habilitarmos este requisito estrito (veremos mais sobre *strict* and *non-strict* no futuro próximo), caso o tipo seja incompatível será lançada uma exceção do tipo *Fatal Error*.

Variáveis

O PHP suporta os seguintes tipos:

- String
- Integer
- Float
- Boolean
- Array
- Object
- Null
- Resource

```
1 <?php
2 $x = 5; // $x e integer
3 $x = "John"; // $x agora e string (e ta tudo bem).
4 ?>
```

Constantes

Constante é um identificador para um valor simples e imutável durante toda a execução do programa. Para definirmos o identificador da constante, utilizamos o comando **define** ou **const**. Após sua criação, o valor poderá ser acessado através de seu identificador.

```
1 <?php
2 define('PROFESSOR', "Leonardo C. R. Soares");
3 const MATERIA = "Desenvolvimento web III";
4
5 echo PROFESSOR;
6 echo MATERIA;
7 ?>
```

Operadores

Aritméticos

- Adição: +
- Subtração: -
- Multiplicação: *
- Divisão: /
- Módulo: %

Operadores

Aritméticos

- Adição: +
- Subtração: -
- Multiplicação: *
- Divisão: /
- Módulo: %

Atribuição

- Recebe: =
- Incremento: ++
- Decremento: - -

Operadores

Operadores de atribuição podem ser utilizados em conjunto com operadores aritméticos:

```
1  <?php
2    $a = 1; // A variavel $a recebe 1;
3    $a += 2; // $a recebe o proprio valor somado a
         2;
4    echo $a; // imprime 3;
5    $a -= 2; // $a recebe o proprio valor menos 2;
6    $a *= 2; // $a recebe o proprio valor
       multiplicado por 2;
7    $a /= 2; // $a recebe o proprio valor dividido
         por 2.
8    echo $a;
9  ?>
```

Operadores

Relacionais

- Igual: ==
- Idêntico: ===
- Diferente: != ou <>
- Menor que: <
- Maior que: >
- Menor ou igual: <=
- Maior ou igual: >=

Operadores

Lógicos Como a maioria das linguagens de programação, PHP possui os operadores lógicos E, OU, OU Exclusivo e Não. Os operadores E e OU possuem duas variações de escrita, a diferença entre elas é a precedência concedida a cada um.

Example	Name	Result
$a and $b	And	**true** if both *$a* and *$b* are **true**.
$a or $b	Or	**true** if either *$a* or *$b* is **true**.
$a xor $b	Xor	**true** if either *$a* or *$b* is **true**, but not both.
! $a	Not	**true** if *$a* is not **true**.
$a && $b	And	**true** if both *$a* and *$b* are **true**.
$a || $b	Or	**true** if either *$a* or *$b* is **true**.

Operador

Ternário

O operador ternário é um operador condicional.

Sua sintaxe básica é: (expr1) ? (expr2) : (expr3). A expr2 será considerada se expr1 for **verdadeira**. Se expr1 for **falsa**, será considerada a expr3.

Operador

Ternário

O operador ternário é um operador condicional.
Sua sintaxe básica é: (expr1) ? (expr2) : (expr3). A expr2 será considerada se expr1 for **verdadeira**. Se expr1 for **falsa**, será considerada a expr3.

```
1 <?php
2 $salario = 2100;
3 echo $salario > 2000 ? "Voce ganha bem." : "
      Precisamos melhorar o salario.";
4 // Sera impresso voce ganha bem.
5 ?>
```

Estruturas condicionais

Frequentemente precisamos atrelar a execução de um trecho de código a uma determinada condição. Para isso utilizamos as estruturas condicionais.
Em PHP temos as seguintes estruturas condicionais:

- **if**: Executa algum código se uma condição for verdadeira;

Estruturas condicionais

Frequentemente precisamos atrelar a execução de um trecho de código a uma determinada condição. Para isso utilizamos as estruturas condicionais.
Em PHP temos as seguintes estruturas condicionais:

- **if**: Executa algum código se uma condição for verdadeira;
- **if...else**: Executa algum código se uma condição for verdadeira e outro caso a condição seja falsa;

Estruturas condicionais

Frequentemente precisamos atrelar a execução de um trecho de código a uma determinada condição. Para isso utilizamos as estruturas condicionais.
Em PHP temos as seguintes estruturas condicionais:

- **if**: Executa algum código se uma condição for verdadeira;
- **if...else**: Executa algum código se uma condição for verdadeira e outro caso a condição seja falsa;
- **if...elseif...else**: Executa diferentes códigos atrelados a mais de duas condições;

Estruturas condicionais

Frequentemente precisamos atrelar a execução de um trecho de código a uma determinada condição. Para isso utilizamos as estruturas condicionais.
Em PHP temos as seguintes estruturas condicionais:

- **if**: Executa algum código se uma condição for verdadeira;
- **if...else**: Executa algum código se uma condição for verdadeira e outro caso a condição seja falsa;
- **if...elseif...else**: Executa diferentes códigos atrelados a mais de duas condições;
- **switch**: Seleciona um entre um bloco de códigos para ser executado.

Estruturas condicionais

Estrutura If - sintaxe

```
1 if (condition) {
2   // code
3 }
```


Estruturas condicionais

Estrutura If - sintaxe

```
1 if (condition) {
2   // code
3 }
```

Estrutura If - Exemplo

```
1 <?php
2 $t = date("H");
3 if ($t < 18) {
4   echo "Tenha um bom dia!";
5 }
6 ?>
```

Estruturas condicionais

Estrutura If ... else - sintaxe

```
1 if (condition) {
2   // condicao verdadeira
3 } else {
4  // condicao falsa
5 }
```

Estruturas condicionais

Estrutura If ... else - Exemplo

```
1  <?php
2  $t = date("H");
3  if ($t < 18) {
4    echo "Tenha um bom dia!";
5  } else {
6    echo "Tenha uma boa noite!";
7  }
8  ?>
```

Estruturas condicionais

Estrutura If ... elseif ... else - sintaxe

```
1 if (condition) {
2   // codigo para esta condicao verdadeira
3 } elseif (condition) {
4   // codigo para condicao anterior falsa e esta
      verdadeira
5 } else {
6   // codigo para o caso de todas as condicoes
      serem falsas
7 }
```

Estruturas condicionais

Estrutura If ... elseif... else - Exemplo

```
1  <?php
2  $t = date("H");
3  if ($t < 12) {
4     echo "Bom dia!";
5  } elseif ($t < 18) {
6     echo "Boa tarde!";
7  } else {
8     echo "Boa noite!";
9  }
10 ?>
```

Estruturas condicionais

switch - sintaxe

```
1  switch (n) {
2    case label1:
3      codigo a ser executado se n=label1;
4      break;
5    case label2:
6      codigo a ser executado se n=label2;
7      break;
8    case label3:
9      codigo a ser executado se n=label3;
10     break;
11     ...
12   default:
13     codigo a ser executado  se n for diferente
         de todos os labels;
14 }
```

Estruturas condicionais

switch - exemplo

```
1  $favcolor = "red";
2  switch ($favcolor) {
3    case "red":
4      echo "Sua cor favorita e vermelho!";
5      break;
6    case "blue":
7      echo "Sua cor favorita e azul!";
8      break;
9    case "green":
10     echo "Sua cor favorita e verde!";
11     break;
12   default:
13     echo "Voce nao e uma pessoa RGB!";
14 }
```

Exercício rápido

Faça um programa em PHP que sorteie um número aleatório entre 1 e 100 ($x=rand(1,100);) e diga se o mesmo é par ou ímpar.

Exercício rápido

Faça um programa em PHP que sorteie um número aleatório entre 1 e 100 ($x=rand(1,100);) e diga se o mesmo é par ou ímpar.

```
1  <?php
2  $x = rand(1,100);
3  echo "O numero sorteado foi $x e ele e ";
4  if ($x%2==0)
5          echo "par.";
6  else
7          echo "impar.";
8  ?>
```

Exercício rápido

Faça um programa em PHP que sorteie dois números aleatórios entre 1 e 100. Imprima o maior número sorteado ou a mensagem 'Numeros iguais' caso eles sejam iguais.

Exercício rápido

Faça um programa em PHP que sorteie dois números aleatórios entre 1 e 100. Imprima o maior número sorteado ou a mensagem 'Numeros iguais' caso eles sejam iguais.

```
1  <?php
2  $x = rand(1,100);
3  $y = rand(1,100);
4  echo "Os numeros sorteado foram $x e $y <br>";
5  if ($x>$y)
6          echo "$x e o maior";
7  elseif($y>$x)
8          echo "$y e o maior";
9  else
10     echo "Numeros iguais";
11 ?>
```

Estruturas de repetição

Loops

Estruturas de repetições ou *loops* são utilizadas quando necessitamos que um bloco de código seja executado mais de uma vez.

Estruturas de repetição

Loops

Estruturas de repetições ou *loops* são utilizadas quando necessitamos que um bloco de código seja executado mais de uma vez.
Em PHP tempos os seguintes tipos de *loops*:

- **while**: O bloco de código será repetido enquanto a condição for verdadeira;

Estruturas de repetição

Loops

Estruturas de repetições ou *loops* são utilizadas quando necessitamos que um bloco de código seja executado mais de uma vez.
Em PHP tempos os seguintes tipos de *loops*:

- **while**: O bloco de código será repetido enquanto a condição for verdadeira;
- **do .. while**: O bloco de código é executado pelo menos uma vez. Depois disso, bloco de código repete enquanto a condição for verdadeira;

Estruturas de repetição

Loops

Estruturas de repetições ou *loops* são utilizadas quando necessitamos que um bloco de código seja executado mais de uma vez.
Em PHP tempos os seguintes tipos de *loops*:

- **while**: O bloco de código será repetido enquanto a condição for verdadeira;
- **do .. while**: O bloco de código é executado pelo menos uma vez. Depois disso, bloco de código repete enquanto a condição for verdadeira;
- **for**: O bloco de código é executa um número específico de vezes;

Estruturas de repetição

Loops

Estruturas de repetições ou *loops* são utilizadas quando necessitamos que um bloco de código seja executado mais de uma vez.
Em PHP tempos os seguintes tipos de *loops*:

- **while**: O bloco de código será repetido enquanto a condição for verdadeira;
- **do .. while**: O bloco de código é executado pelo menos uma vez. Depois disso, bloco de código repete enquanto a condição for verdadeira;
- **for**: O bloco de código é executa um número específico de vezes;
- **foreach**: O bloco de código será repetido para cada elemento em um vetor.

Estruturas de repetição

while

Executa um bloco de código enquanto a condição for verdadeira.
Sintaxe:

```
1 <?php
2 while (condicao) {
3   //codigo;
4 }
5 ?>
```

Estruturas de repetição

while - Exemplo

```
1  <?php
2  $x = 1;
3  while($x <= 5) {
4    echo "O numero esta em: $x <br>";
5    $x++;
6  }
7  ?>
```

Estruturas de repetição

do while

Executa um bloco de código uma vez. Após isso, repete este bloco enquanto a condição for verdadeira.

Sintaxe:

```php
<?php
do {
  // codigo
} while (condicao);
?>
```

Estruturas de repetição

do while - Exemplo

```
1  <?php
2  $x = 1;
3  do {
4     echo "O numero esta em: $x <br>";
5     $x++;
6  } while ($x <= 5);
7  ?>
```

Estruturas de repetição

for

A estrutura de repetição for é utilizada quando desejamos que o bloco de código seja repetido um número previamente conhecido de vezes.
Sintaxe:

```
1 <?php
2 for (inicio contador; teste contador; incremento
      contador) {
3   // codigo
4 }
5 ?>
```

Estruturas de repetição

for - Exemplo

```php
<?php
for ($x = 0; $x <= 10; $x++) {
  echo "O numero esta em: $x <br>";
}
?>
```

Estruturas de repetição

foreach

A estrutura de repetição foreach foi desenvolvida exclusivamente para iteração sobre vetores. O bloco de código é repetido para cada elemento do vetor.

Sintaxe:

```
1 <?php
2 foreach ($vetor as $valor) {
3   // codigo
4 }
5 ?>
```

Estruturas de repetição

foreach - Exemplo

```
1  <?php
2  $colors = array("red", "green", "blue", "lime");
3
4  foreach ($colors as $value) {
5    echo "$value <br>";
6  }
7  ?>
```

Estruturas de repetição

Caso queiramos, é possível acessar, além do valor de cada elemento, a chave do mesmo.

foreach - Exemplo

```
1  <?php
2  $age = array("Leia"=>35, "Ben"=>37, "Joe"=>43);
3
4  foreach($age as $chave => $valor) {
5    echo "$chave tem $valor anos.<br>";
6  }
7  ?>
```

Break e Continue

break

Caso seja necessário interromper a execução de um *loop* antes que a condição de parada seja atingida, podemos utilizar a palavra-chave **break**.

```
1  <?php
2  for ($x = 0; $x < 10; $x++) {
3    if ($x == 4) {
4      break;
5    }
6    echo "O numero esta em: $x <br>";
7  }
8  ?>
```

Break e Continue

continue

Caso seja necessário pular **uma** iteração da estrutura de repetição, podemos utilizar a palavra-chave **continue**.

```
1  <?php
2  for ($x = 0; $x < 10; $x++) {
3    if ($x == 4) {
4      continue;
5    }
6    echo "O numero esta em: $x <br>";
7  }
8  ?>
```

Exercício rápido

Faça um programa em PHP que imprima todos os números pares entre 1 e 100.

Exercício rápido

Faça um programa em PHP que imprima todos os números pares entre 1 e 100.

```
1  <?php
2  for ($x = 1; $x <= 100; $x++)
3    if ($x%2==0)
4       echo "$x <br>";
5  ?>
```

Exercício rápido

Faça um programa em PHP que imprima todos os números pares entre 1 e 100.

```php
<?php
for ($x = 1; $x <= 100; $x++)
  if ($x%2==0)
     echo "$x <br>";
?>
```

```php
<?php
for ($x = 2; $x <= 100; $x+=2)
  echo "$x <br>";
?>
```

Exercício rápido

Faça um programa em PHP que calcule e imprima o valor da soma de todos os números inteiros entre 1 e 100.

Exercício rápido

Faça um programa em PHP que calcule e imprima o valor da soma de todos os números inteiros entre 1 e 100.

```php
<?php
$soma = 0;
for ($i = 1; $i <= 100; $i++) {
  $soma +=$i;
}
echo "A soma e $soma;";
?>
```

Exercícios

1. Faça um programa em PHP que sorteie um número aleatório entre 0 e 10 e o imprima por extenso;
2. Faça um programa em PHP que sorteie um número aleatório entre 1 e 10. Caso o número seja entre 1 e 7, imprima o dia da semana correspondente (sendo 1 domingo e 7 sábado). Caso o número sorteado não seja um dia da semana válido, imprima uma mensagem com esta informação.
3. Faça um programa em PHP que sorteie três números aleatórios entre 0 e 100 e imprima o maior e o menor entre eles. Utilize operadores lógicos em conjunto com as estruturas condicionais se for preciso;

Exercícios

4. Escreva um programa em PHP que sorteie um número referente a idade de uma pessoa (entre 0 e 90). Informe a situação eleitoral desta pessoa sabendo que:
   - Menor de 16 anos não vota;
   - Entre 18 e 64 anos o voto é obrigatório;
   - Entre 16 e 17 e após os 65 anos, inclusive, o voto é facultativo.
5. Escreva um programa em PHP que sorteie 4 números aleatórios, i entre 1 e 3; a, b e c entre 1 e 100. Em seguida:
   - Se i=1 escrever os três valores a, b, c em ordem crescente;
   - Se i=2 escrever os três valores a, b, c em ordem decrescente;
   - Se i=3 escrever os três valores a, b, c de forma que o maior entre a, b, c fique dentre os dois.

Exercícios

6. Faça um programa em PHP que sorteie dois números aleatórios entre 1 e 100. Imprima todos os números inteiros existentes entre eles, em ordem crescente;
7. Faça um programa em PHP que sorteie um número aleatório entre 1 e 100. Calcule e imprima todos os divisores deste número;
8. Faça um programa em PHP que sorteie um número aleatório entre 1 e 100. Imprima se o número sorteado é ou não um número primo;
9. Faça um programa em PHP que sorteie um número aleatório entre 1 e 100. Imprima se o número sorteado é ou não um número perfeito;

Exercícios

10. Faça um programa em PHP que sorteie um número aleatório $x entre 1 e 10. Depois, imprima a pirâmide de caracteres conforme exemplificado abaixo. A pirâmide deve possuir $x linhas e a cada linha, o número de linhas em asteriscos.
    Por exemplo, considere $x = 5. A saída deverá ser:
    *
    **
    ***
    ****
    *****


thank you
danke
謝謝
ngiyabonga
teşekkür ederim
gracias
merci
obrigado
dziękuję
bedankt
go raibh maith agat
sukriya
kop khun krap
terima kasih
grazie
arigatō
dakujem
mochchakkeram
tapadh leat
спасибо
рахмат
Баярлалаа
mersi
barka
wetalin
tack
dank je
misaotra
matondo
paldies
grazzi
mahalo
vinaka
blagodaram
спасибі
faafetai lava
kiitos
dankie
dhanyavad
nanni
nandri
mauruuru
hvala
gracie
bayarlalaa
enkosi
koszönöm
ačiū
dankon
akun
tau
djiere dieuf
хвала
asante
manana
obrigada
murakoze
tenki
chokrane
дякую
mamnun
chnorakaloutioun
gratias ago
gracies
sulpay
taiku
sobodi
dekuji
sagolun
mèsi
didi madloba
kam sah hamnida
najis tuke
ありがとう
tanemirt
rahmet
rahmat
তোমাকে ধন্যবাদ
감사합니다
xiexie
ευχαριστώ
diolch
dhanyavadagalu
shukriya
merce
takk
trugarez
мерси