

Introdução à Programação e Configuração do Ambiente

O que é Programação?

Imagine que você quer dar instruções para um computador realizar uma tarefa. A programação é a forma de escrever essas instruções de maneira que o computador entenda e execute. É como se você estivesse ensinando o computador a fazer algo, passo a passo.

Para que serve a Programação?

A programação está presente em tudo o que nos rodeia: desde os aplicativos que usamos no celular, até os sistemas complexos que controlam o tráfego aéreo.

Com a programação, podemos criar soluções para os mais diversos problemas, automatizar tarefas, criar jogos, websites e muito mais.

Por que PHP?

PHP é uma linguagem de programação muito popular e utilizada para desenvolvimento web. Ela é relativamente fácil de aprender e possui uma grande comunidade de desenvolvedores dispostos a ajudar.

Além disso, o PHP é uma linguagem versátil que permite criar desde páginas web simples até sistemas complexos.

Nosso Ambiente de Desenvolvimento

Para começarmos a programar em PHP, precisaremos configurar nosso ambiente de desenvolvimento. Utilizaremos as seguintes ferramentas:

- **VSCode:** Um editor de código poderoso e personalizável, onde escreveremos nossos programas.
- **XAMPP:** Um pacote de software que inclui o servidor web Apache, o banco de dados MySQL e o PHP, tudo o que precisamos para rodar nossos programas PHP.
- **Workbench:** Uma ferramenta para gerenciar o banco de dados MySQL, que utilizaremos mais adiante em nossos projetos.

Instalando e Configurando as Ferramentas

VSCode:

- Acesse o site oficial do VSCode (<https://code.visualstudio.com/>) e baixe a versão para o seu sistema operacional.
- Instale o VSCode seguindo as instruções do instalador.
- Após a instalação, abra o VSCode e instale a extensão "PHP Intelephense" para ter suporte completo à linguagem PHP.

XAMPP:

- Acesse o site oficial do XAMPP (<https://www.apachefriends.org/index.html>) e baixe a versão para o seu sistema operacional.
- Instale o XAMPP seguindo as instruções do instalador.

- Após a instalação, abra o XAMPP Control Panel e inicie os serviços "Apache" e "MySQL".

Workbench:

- Acesse o site (<https://dev.mysql.com/downloads/workbench/>) e baixe a versão para o seu sistema operacional.
- Instale o Workbench seguindo as instruções do instalador.

Nosso Primeiro Programa

Com o ambiente configurado, podemos escrever nosso primeiro programa em PHP. Abra o VSCode e crie um novo arquivo com o nome "hello.php". Dentro desse arquivo, escreva o seguinte código:

```
<?php  
echo "Olá, mundo!";?>
```

Salve o arquivo e abra o seu navegador. Na barra de endereço, digite "localhost/hello.php" e pressione Enter.

Se tudo estiver configurado corretamente, você verá a mensagem "Olá, mundo!" na tela.

Curiosidade

Você sabia que o PHP foi criado em 1994 por Rasmus Lerdorf?

Inicialmente, era apenas um conjunto de scripts para monitorar o tráfego do seu site pessoal, mas com o tempo, evoluiu e se tornou uma das linguagens de programação mais utilizadas no mundo.

O PHP ainda é uma das linguagens de programação mais utilizadas na web.

Porcentagem de sites e sistemas que utilizam PHP:

- W3Techs: Estima que o PHP é usado por 78,9% de todos os sites que usam uma linguagem de programação do lado do servidor conhecida.
- Kinsta: Relata que o PHP está presente em mais de 70% dos sites na web atualmente.
- Hostinger: Afirma que o PHP é utilizado por 78,1% de todos os sites na internet, já que é a linguagem primária do WordPress.

Observações:

- A porcentagem exata de sites que utilizam PHP varia de acordo com a fonte e a metodologia de pesquisa.

Fatores que influenciam a popularidade do PHP:

- Facilidade de aprendizado: O PHP é considerado relativamente fácil de aprender, o que contribui para sua popularidade entre iniciantes.

- Grande comunidade: O PHP possui uma grande comunidade de desenvolvedores, o que facilita a busca por ajuda e recursos.
- WordPress: O WordPress, um dos CMSs mais populares do mundo, é escrito em PHP, o que garante a relevância da linguagem.
- Sistemas legados: Muitas empresas ainda utilizam sistemas legados escritos em PHP, o que mantém a demanda por desenvolvedores PHP.

Empresas que utilizam PHP

- **Facebook:** A maior rede social do mundo, o Facebook, utiliza PHP em grande parte de sua estrutura. A linguagem é utilizada para gerar páginas dinâmicas, processar dados de usuários e interagir com o banco de dados.
- **Yahoo:** Um dos pioneiros da internet, o Yahoo utiliza PHP em diversos de seus serviços, como o portal de notícias, o serviço de e-mail e a plataforma de buscas.
- **Wikipedia:** A enciclopédia online mais famosa do mundo, a Wikipedia, utiliza PHP para gerar suas páginas, exibir conteúdo dinâmico e interagir com o banco de dados.
- **WordPress:** A plataforma de blogs e sites mais utilizada do mundo, o WordPress, é totalmente escrita em PHP. Isso garante a flexibilidade e a facilidade de uso da plataforma.

Empresas brasileiras

- **Locaweb:** Uma das maiores empresas de hospedagem de sites e serviços de internet do Brasil, a Locaweb utiliza PHP em seus sistemas para gerenciar contas de clientes, processar pagamentos e oferecer suporte técnico.
- **TOTVS:** Uma das maiores empresas de software da América Latina, a TOTVS utiliza PHP em alguns de seus sistemas, como o sistema de gestão empresarial RM.

Observação

É importante lembrar que as empresas podem utilizar diversas linguagens de programação em seus sistemas. PHP é apenas uma delas.

Espero que esta informação tenha sido útil. Se tiver mais alguma dúvida, pode perguntar!

Variáveis e Tipos de Dados em PHP

Variáveis

O que são?

Variáveis são como caixas onde podemos armazenar informações. Cada caixa tem um nome único que a identifica, e podemos colocar diferentes tipos de informações dentro dela.

Para que servem?

As variáveis nos permitem guardar dados temporariamente durante a execução do programa. Esses dados podem ser números, textos, resultados de cálculos, etc.

Como criar?

Em PHP, as variáveis são representadas por um símbolo de dólar (\$) seguido do nome da variável. O nome da variável deve começar com uma letra ou sublinhado, e pode conter letras, números e sublinhados.

Exemplos

```
$nome = "João"; // Variável que armazena um texto
$idade = 30; // Variável que armazena um número inteiro
$altura = 1.80; // Variável que armazena um número decimal
$casado = true; // Variável que armazena um valor booleano (verdadeiro ou falso)
```

Regras importantes

- **Nomes:** Devem começar com \$ seguido de uma letra ou sublinhado.
- **Case-sensitive:** PHP diferencia letras maiúsculas e minúsculas (\$nome é diferente de \$Nome).
- **Atribuição:** O sinal de igual (=) é usado para atribuir um valor a uma variável.

Tipos de Dados

O que são?

Os tipos de dados definem o tipo de informação que uma variável pode armazenar. PHP possui diversos tipos de dados, como:

- **Inteiros (int):** Números inteiros (ex: 10, -5, 0).
- **Decimais (float):** Números com casas decimais (ex: 3.14, -2.5).
- **Textos (string):** Sequências de caracteres (ex: "Olá", "PHP").
- **Booleanos (bool):** Valores verdadeiro (true) ou falso (false).
- **Arrays:** Coleções de dados.
- **Objetos:** Instâncias de classes.
- **Nulo (null):** Valor que representa a ausência de um valor.

Para que servem?

Os tipos de dados nos ajudam a organizar e manipular as informações de forma correta. Cada tipo de dado possui características e operações específicas.

Gettype() - Como descobrir o tipo de uma variável?

Podemos usar a função `gettype()` para descobrir o tipo de uma variável.

```
$numero = 10;echo gettype($numero); // Saída: integer
$texto = "Olá";echo gettype($texto); // Saída: string
$booleano = true;echo gettype($booleano); // Saída: boolean
```

Conversão de Tipos (Type Casting)

O que é?

A conversão de tipos é a forma de transformar um dado de um tipo para outro.

Para que serve?

A conversão de tipos é útil quando precisamos realizar operações entre dados de tipos diferentes.

Como fazer?

Podemos usar o operador de cast (tipo) para converter um dado para um tipo específico.

Exemplos

```
$numero = 10;
$texto = (string) $numero; // Converte o número para texto
echo gettype($texto); // Saída: string

$texto = "3.14";
$decimal = (float) $texto; // Converte o texto para decimal
echo gettype($decimal); // Saída: double
```

Dicas

- Escolha nomes de variáveis que sejam claros e descritivos.
- Utilize os tipos de dados corretos para cada situação.
- Faça a conversão de tipos quando necessário.

Curiosidade

Você sabia que PHP é uma linguagem de tipagem dinâmica? Isso significa que o tipo de uma variável é definido pelo valor que ela armazena, e pode mudar durante a execução do programa.

Formulário: Fazer exemplo com get e post

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <!-- criar um formulario com os campos nome, email e senha -->
  <form action="b.php" method="post">
    <input type="text" name="nome" placeholder="Nome">
    <input type="email" name="email" placeholder="Email">
    <input type="password" name="senha" placeholder="Senha">
    <button type="submit">Enviar</button>
  </form>
</body>
</html>
```

=====

Sem validar

```
<?php
    $nome = $_POST["nome"];
    $email = $_POST["email"];
    $senha = $_POST["senha"];

    echo "Nome: {$nome}<br>";
    echo "Email: {$email}<br>";
    echo "Senha: {$senha}<br>";
?>
```

Validação básica

```
<?php
    // valide se o formulario foi enviado
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        $nome = $_POST["nome"];
        $email = $_POST["email"];
        $senha = $_POST["senha"];

        echo "Nome: {$nome}<br>";
        echo "Email: {$email}<br>";
        echo "Senha: {$senha}<br>";
    }else{
        echo "Formulario não enviado";
    }
?>
```

Saída de dados, concatenação e interpolação

Em programação, **saída de dados** refere-se ao processo de exibir ou retornar informações para o usuário ou para outro sistema. Em PHP, a saída de dados é comumente realizada utilizando a função `echo`, que imprime uma ou mais strings na tela.

Exemplo básico de saída de dados:

```
<?php
echo "Olá, mundo!";
?>
```

Neste exemplo, a função `echo` exibe a mensagem "Olá, mundo!" na tela.

Concatenação de Strings:

A **concatenação** é o processo de unir duas ou mais strings em uma única. Em PHP, o operador de concatenação é o ponto (`.`).

Exemplo de concatenação:

```
<?php
$nome = "João";
$idade = 25;
echo "Meu nome é " . $nome . " e tenho " . $idade . " anos.";
?>
```

Saída:

Meu nome é João e tenho 25 anos.

Neste exemplo, as variáveis `$nome` e `$idade` são concatenadas com strings literais para formar uma frase completa.

Interpolação de Strings:

A **interpolação** permite inserir o valor de variáveis diretamente dentro de uma string. Em PHP, isso é possível quando se utiliza aspas duplas (`"`) para definir a string.

Exemplo de interpolação:

```
<?php
$nome = "João";
$idade = 25;
echo "Meu nome é $nome e tenho $idade anos.";
?>
```

Saída:

Meu nome é João e tenho 25 anos.

Neste exemplo, as variáveis \$nome e \$idade são inseridas diretamente na string, sem a necessidade de concatenação explícita.

Observações Importantes:

Aspas Simples vs. Aspas Duplas: Quando se utiliza aspas simples ('), o PHP não realiza a interpolação de variáveis. Já com aspas duplas ("), a interpolação é realizada.

Exemplo com aspas simples:

```
<?php
$nome = "João";
echo 'Meu nome é $nome.'; // Saída: Meu nome é $nome.
?>
```

Uso de chaves para variáveis dentro de strings: Para evitar ambiguidades, especialmente quando a variável é seguida por caracteres alfanuméricos, é recomendável utilizar chaves {} ao redor da variável.

Exemplo com chaves:

```
<?php
$nome = "João";
echo "Meu nome é {$nome}123."; // Saída: Meu nome é João123.
?>
```

Desempenho: Em termos de desempenho, a concatenação utilizando o operador . pode ser mais eficiente em alguns casos, especialmente quando se trabalha com muitas variáveis.

Operadores Aritméticos em PHP

Operadores Aritméticos

O que são?

Os operadores aritméticos são símbolos que nos permitem realizar operações matemáticas básicas, como adição, subtração, multiplicação, divisão e resto da divisão.

Para que servem?

Os operadores aritméticos são essenciais para realizar cálculos em nossos programas.

Exemplos

Operador	Nome	Exemplo	Resultado
+	Adição	$\$a + \b	Soma de $\$a$ e $\$b$
-	Subtração	$\$a - \b	Diferença entre $\$a$ e $\$b$
*	Multiplicação	$\$a * \b	Produto de $\$a$ e $\$b$
/	Divisão	$\$a / \b	Quociente de $\$a$ e $\$b$
%	Módulo (resto)	$\$a \% \b	Resto da divisão de $\$a$ por $\$b$
**	Exponenciação	$\$a ** \b	a elevado à potência de b

```
$a = 10;
$b = 5;
echo $a + $b;      // Saída: 15
echo $a - $b;      // Saída: 5
echo $a * $b;      // Saída: 50
echo $a / $b;      // Saída: 2
echo $a % $b;      // Saída: 0
echo $a ** $b;     // Saída: 100000
```

Operadores de Atribuição Combinados

O que são?

Os operadores de atribuição combinados são formas abreviadas de realizar uma operação aritmética e atribuir o resultado à mesma variável.

Para que servem?

Os operadores de atribuição combinados tornam o código mais conciso e legível.

Exemplos

Operador	Equivalente a	Exemplo
+=	$\$a = \$a + \$b$	$\$a += \b
-=	$\$a = \$a - \$b$	$\$a -= \b
*=	$\$a = \$a * \$b$	$\$a *= \b
/=	$\$a = \$a / \$b$	$\$a /= \b
%=	$\$a = \$a \% \$b$	$\$a \% = \b

```
$a = 10;
$b = 5;
$a += $b;  // $a = $a + $b;
echo $a;   // Saída: 15
```

```
$a -= $b;    // $a = $a - $b;
echo $a;    // Saída: 10

$a *= $b;    // $a = $a * $b;
echo $a;    // Saída: 50

$a /= $b;    // $a = $a / $b;
echo $a;    // Saída: 10

$a %= $b;    // $a = $a % $b;
echo $a;    // Saída: 0
```

Operadores de Incremento e Decremento

O que são?

Os operadores de incremento e decremento são utilizados para aumentar ou diminuir o valor de uma variável em 1.

Para que servem?

Os operadores de incremento e decremento são úteis em loops e outras situações onde precisamos alterar o valor de uma variável de forma rápida.

Exemplos

Operador	Nome	Exemplo
++	Pré-incremento	++\$a
--	Pré-decremento	--\$a
++	Pós-incremento	\$a++
--	Pós-decremento	\$a--

```
$a = 10;

echo ++$a; // Saída: 11 (pré-incremento)
echo $a;   // Saída: 11

echo $a--; // Saída: 11 (pós-decremento)
echo $a;   // Saída: 10
```

Lista de exercícios

01 - Cálculo de Distância Percorrida

Crie um programa que leia o **tempo** (em horas) e a **velocidade média** (em km/h) e calcule a **distância percorrida** utilizando a fórmula:

$$\text{distância} = \text{Velocidade média} / \text{tempo}$$

Exemplo:

Digite o tempo (em horas): 3

Digite a velocidade média (km/h): 60

Distância percorrida: 180.00 km

02 - Cálculo da Área de uma Esfera

Crie um programa que leia o **raio** (r) de uma esfera e calcule a **área superficial** da esfera usando a fórmula:

$$A = 4 * \pi * r^2$$

Onde π é aproximadamente 3,14159.

Exemplo:

Digite o raio da esfera: 5

Área superficial: 314.16

03 - Cálculo do Volume de um Cone

Crie um programa que leia o **raio** (r) e a **altura** (h) de um cone e calcule o **volume** utilizando a fórmula:

$$V = 1/3 * \pi * r^2 * h$$

Exemplo:

Digite o raio do cone: 3

Digite a altura do cone: 5

Volume: 141.37

04 - Conversão de Temperatura (Escala Fahrenheit para Celsius)

Crie um programa que leia uma **temperatura** em **Fahrenheit** e converta para **Celsius** usando a fórmula:

$$C = 5 / 9 * (F - 32)$$

Exemplo:

Digite a temperatura em Fahrenheit: 100

Temperatura em Celsius: 37.8

05 - Cálculo do Salário com Desconto de Impostos

Crie um programa que leia o **salário bruto** de um trabalhador e aplique os seguintes descontos:

- Desconto de **INSS** de 8%
- Desconto de **Imposto de Renda** de 12%

Exiba o **salário líquido** após os descontos.

Exemplo:

Digite o salário bruto: 3000

Desconto de INSS: 240

Desconto de Imposto de Renda: 360

Salário líquido: 2400

06 - Cálculo da Média Ponderada

Crie um programa que leia **três notas** e seus respectivos **pesos**, e calcule a **média ponderada** utilizando a fórmula:

$$Mp = [(N1 * P1) + (N2 * P2) + (N3 * P3)] / (P1 + P2 + P3)$$

Exemplo:

Digite a primeira nota: 7

Digite o peso da primeira nota: 3

Digite a segunda nota: 8

Digite o peso da segunda nota: 2

Digite a terceira nota: 9

Digite o peso da terceira nota: 1

Média ponderada: 7.67

07 - Cálculo da Perda de Massa de um Corpo

Crie um programa que leia a **massa inicial** de um corpo (em kg) e calcule a **massa final** após perder 10% de sua massa inicial, 20% e 30%.

Exemplo:

Digite a massa inicial: 80

Massa final após perder 10%: 72.00

Massa final após perder 20%: 64.00

Massa final após perder 30%: 56.00

Observação:

Trabalhe cada exercício de forma independente, realizando as **operações matemáticas** corretas e utilizando os operadores de forma precisa.

Ao final, **teste os cálculos** com diferentes valores para garantir que o programa está funcionando corretamente.

Faça **reflexões** sobre como os operadores podem ser usados de forma eficiente para resolver problemas matemáticos e lógicos.

Operadores Relacionais

O que são?

Os operadores relacionais são utilizados para comparar dois valores e retornar um valor booleano (verdadeiro ou falso).

Para que servem?

Os operadores relacionais são essenciais para criar estruturas de controle de fluxo, como condicionais e loops.

Exemplos

Operador	Nome	Exemplo	Resultado
==	Igual	<code>\$a == \$b</code>	Verdadeiro se a for igual a b
!=	Diferente	<code>\$a != \$b</code>	Verdadeiro se a for diferente de b
>	Maior que	<code>\$a > \$b</code>	Verdadeiro se a for maior que b
<	Menor que	<code>\$a < \$b</code>	Verdadeiro se a for menor que b
>=	Maior ou igual a	<code>\$a >= \$b</code>	Verdadeiro se a for maior ou igual a b
<=	Menor ou igual a	<code>\$a <= \$b</code>	Verdadeiro se a for menor ou igual a b
===	Idêntico	<code>\$a === \$b</code>	Verdadeiro se a for igual a b e do mesmo tipo
!==	Não idêntico	<code>\$a !== \$b</code>	Verdadeiro se a for diferente de b ou de tipo diferente

```
$a = 10;
$b = 5;

var_dump($a == $b); // Saída: boolean false
var_dump($a != $b); // Saída: boolean true
var_dump($a > $b); // Saída: boolean true
var_dump($a < $b); // Saída: boolean false
var_dump($a >= $b); // Saída: boolean true
var_dump($a <= $b); // Saída: boolean false

$c = "10";

var_dump($a == $c); // Saída: boolean true
var_dump($a === $c); // Saída: boolean false
```

Operadores Lógicos

O que são?

Os operadores lógicos são utilizados para combinar expressões booleanas e retornar um valor booleano.

Para que servem?

Os operadores lógicos são essenciais para criar condições mais complexas em nossos programas.

Exemplos

&& (E):	Retorna verdadeiro se ambas as expressões forem verdadeiras.
 (OU):	Retorna verdadeiro se pelo menos uma das expressões for verdadeira.
! (NÃO):	Inverte o valor da expressão (verdadeiro se torna falso e vice-versa).

```
$a = true;
$b = false;

var_dump($a && $b);           // Saída: boolean false
var_dump($a || $b);           // Saída: boolean true
var_dump(!$a);                 // Saída: boolean false
```

Exemplos Práticos

Operador &&

```
$a = true;
$b = false;

if ($a && $b) {
    echo "Ambas as condições são verdadeiras.";
} else {
    echo "Pelo menos uma das condições é falsa.";
}
// Saída: Pelo menos uma das condições é falsa.
```

Operador ||

```
$a = true;
$b = false;

if ($a || $b) {
    echo "Pelo menos uma das condições é verdadeira.";
} else {
    echo "Ambas as condições são falsas.";
}
// Saída: Pelo menos uma das condições é verdadeira.
```

Aqui, a condição if verifica se pelo menos uma das variáveis \$a ou \$b é verdadeira. Como \$a é verdadeiro, a mensagem "Pelo menos uma das condições é verdadeira." é exibida.

Operador NÃO (NOT) !

```
$a = false;

if (!$a) {
    echo "A condição é falsa.";
} else {
    echo "A condição é verdadeira.";
}
// Saída: A condição é falsa.
```

Neste caso, o operador ! inverte o valor de \$a. Como \$a é falso, !\$a é verdadeiro, e a mensagem "A condição é falsa." é exibida.

Estrutura de controle condicional

Usando o if

O if é usado para verificar se uma condição é verdadeira. Se for, o código dentro do bloco será executado.

```
<?php
$idade = 20;

if ($idade >= 18) {
    echo "Você tem 18 anos ou mais.";
}
?>
```

Usando if else

O if else permite que você forneça um bloco de código alternativo caso a condição do if seja falsa.

```
<?php
$cpf = '123.456.789-00';

if ($cpf == '123.456.789-00') {
    echo "CPF válido.";
} else {
    echo "CPF inválido.";
}
?>
```

Usando ifs encadeados

Os ifs encadeados ocorrem quando há uma estrutura de if dentro de outra. Cada if adicional depende da verificação anterior.

```
<?php
$diaSemana = 3;           // 1 - Domingo, 2 - Segunda-feira, ... 7 - Sábado

if ($diaSemana == 1) {
    echo "Hoje é domingo.<br>";
} else {
    if ($diaSemana == 2) {
        echo "Hoje é segunda-feira.<br>";
    } else {
        if ($diaSemana == 3) {
            echo "Hoje é terça-feira.<br>";
        } else {
            if ($diaSemana == 4) {
                echo "Hoje é quarta-feira.<br>";
            } else {
                if ($diaSemana == 5) {
                    echo "Hoje é quinta-feira.<br>";
                } else {
                    if ($diaSemana == 6) {
                        echo "Hoje é sexta-feira.<br>";
                    } else {
                        if ($diaSemana == 7) {
                            echo "Hoje é sábado.<br>";
                        } else {
                            echo "Dia inválido.<br>";
                        }
                    }
                }
            }
        }
    }
}

?>
```

Usando ifs identados

O if identado é apenas uma forma de organizar melhor o código, especialmente quando se tem várias condições e verificações dentro de um bloco.

```
<?php
$diaSemana = 3;           // 1 - Domingo, 2 - Segunda-feira, ... 7 - Sábado

if ($diaSemana == 1) {
    echo "Hoje é domingo.<br>";
} elseif ($diaSemana == 2) {
    echo "Hoje é segunda-feira.<br>";
} elseif ($diaSemana == 3) {
    echo "Hoje é terça-feira.<br>";
} elseif ($diaSemana == 4) {
    echo "Hoje é quarta-feira.<br>";
} elseif ($diaSemana == 5) {
```

```

        echo "Hoje é quinta-feira.<br>";
    } elseif ($diaSemana == 6) {
        echo "Hoje é sexta-feira.<br>";
    } elseif ($diaSemana == 7) {
        echo "Hoje é sábado.<br>";
    } else {
        echo "Dia inválido.<br>";
    }
?>

```

Exemplo com formulário

```

<form method="post" action="***">
    <label for="nome">Nome:</label>
    <input type="text" id="nome" name="nome">
    <input type="submit" value="Enviar">
</form>

```

```

=====
<?php
    if ($_SERVER["REQUEST_METHOD"] == "POST") {

        $nome = $_POST['nome'];

        // Verifica se o campo "nome" foi preenchido
        if (!empty($nome)) {
            echo "Olá, $nome! Bem-vindo ao nosso site.<br>";
        } else {
            echo "Por favor, preencha o campo de nome.<br>";
        }
    }
?>

```

Resumo

- if: Executa um bloco de código se a condição for verdadeira.
- if else: Executa um bloco de código se a condição for verdadeira e outro se for falsa.
- ifs encadeados: São ifs dentro de outros, com cada um dependendo do anterior.
- ifs identados: A indentação melhora a legibilidade do código, facilitando a visualização da estrutura condicional.

Switch

A estrutura **switch** é utilizada quando temos várias condições para verificar com base no valor de uma única variável. Ela funciona como uma alternativa ao **if-elseif-else**, tornando o código mais organizado e legível quando lidamos com múltiplas comparações de igualdade.

Quando Usar switch em Vez de if-else?

Prefira **switch** quando:

- Você precisa comparar **um único valor** contra **múltiplas opções**.
- Cada caso representa um possível valor fixo, como números ou strings.
- Você deseja evitar múltiplos elseif, deixando o código mais limpo.

Prefira **if-else** quando:

- As condições envolvem comparações mais complexas (maior, menor, entre dois valores, etc.).
- Precisamos verificar múltiplas variáveis ao mesmo tempo.
- Existem expressões booleanas ou cálculos envolvidos na condição.

Exemplo Prático:

Agora, vamos criar um script PHP onde o usuário digita um número de **1 a 7**, e o programa exibe o dia correspondente.

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $numero = $_POST['numero'];

    echo "Você digitou: $numero <br>";

    switch ($numero) {
        case 1:
            echo "Hoje é domingo.<br>";
            break;
        case 2:
            echo "Hoje é segunda-feira.<br>";
            break;
        case 3:
            echo "Hoje é terça-feira.<br>";
            break;
        case 4:
            echo "Hoje é quarta-feira.<br>";
            break;
        case 5:
            echo "Hoje é quinta-feira.<br>";
            break;
        case 6:
            echo "Hoje é sexta-feira.<br>";
            break;
        case 7:
            echo "Hoje é sábado.<br>";
            break;
        default:
            echo "Número inválido! Digite um valor entre 1 e 7.<br>";
    }
}
```

```

?>

<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Switch - Dias da Semana</title>
</head>
<body>

  <h2>Digite um número de 1 a 7 para ver o dia da semana:</h2>
  <form method="post" action="">
    <input type="number" name="numero" min="1" max="7" required>
    <input type="submit" value="Verificar">
  </form>

</body>
</html>

```

Por que usamos switch aqui?

O valor digitado é **comparado diretamente** a números fixos (1 a 7).
 O código fica **mais organizado e fácil de entender** em comparação com múltiplos if-elseif.
 Usamos default para tratar **casos inválidos** automaticamente.

Exemplo Prático:

Identificando Vogais e Consoantes

Agora, vamos criar um formulário onde o usuário digita uma letra, e o PHP verifica se é uma vogal (A, E, I, O, U). Caso contrário, informamos que a letra é uma consoante. Além disso, vamos garantir que a entrada seja convertida para **maiúscula** (uppercase) para evitar distinção entre 'a' e 'A', por exemplo.

Código Completo:

```

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $letra = strtoupper($_POST['letra']);      // Converte para maiúscula

  echo "Você digitou: $letra <br>";

  switch ($letra) {
    case 'A':
      echo "A letra digitada é a vogal A.<br>";
      break;
    case 'E':
      echo "A letra digitada é a vogal E.<br>";
      break;

```

```

        case 'I':
            echo "A letra digitada é a vogal I.<br>";
            break;
        case 'O':
            echo "A letra digitada é a vogal O.<br>";
            break;
        case 'U':
            echo "A letra digitada é a vogal U.<br>";
            break;
        default:
            echo "A letra digitada é uma consoante.<br>";
    }
}
?>

<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Switch - Vogais e Consoantes</title>
</head>
<body>

    <h2>Digite uma letra para verificar se é vogal ou consoante:</h2>
    <form method="post" action="">
        <input type="text" name="letra" maxlength="1" required>
        <input type="submit" value="Verificar">
    </form>

</body>
</html>

```

Operador Ternário

O operador ternário (?:) é uma forma simplificada de escrever uma estrutura if-else em uma única linha. Ele é útil para expressões curtas e diretas, tornando o código mais compacto e legível.

Sintaxe do Operador Ternário

condição ? valor_se_verdadeiro : valor_se_falso;

- Se a condição for verdadeira (true), retorna o valor após ?.
- Se a condição for falsa (false), retorna o valor após :.

Exemplo Simples: Verificar se um Número é Par ou Ímpar

```

<?php
$numero = 10;

```

```
$resultado = ($numero % 2 == 0) ? "O número $numero é Par." : "O número $numero é Ímpar.";
echo $resultado;
?>
```

Explicação:

- A condição ($\$numero \% 2 == 0$) verifica se o número é par.
- Se for **verdadeiro**, retorna "O número é Par."
- Se for **falso**, retorna "O número é Ímpar."
- O resultado é armazenado na variável `$resultado`, que é exibida com `echo`.

Exemplo com Formulário: Maioridade

Vamos criar um **formulário** onde o usuário digita a idade, e o PHP verifica se ele é maior ou menor de idade usando o operador ternário.

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $idade = $_POST['idade'];

    // Usando operador ternário para verificar a maioridade
    $mensagem = ($idade >= 18) ? "Você é maior de idade." : "Você é menor de idade.";

    echo $mensagem . "<br>";
}
?>

<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Verificação de Idade</title>
</head>
<body>

    <h2>Digite sua idade:</h2>
    <form method="post" action="">
        <input type="number" name="idade" required>
        <input type="submit" value="Verificar">
    </form>

</body>
</html>
```

Por que usar o Operador Ternário?

Código mais curto e legível do que um if-else.

Ideal para atribuir valores diretamente a variáveis.

Útil para expressões simples, sem necessidade de blocos grandes de código.

Quando NÃO Usar o Operador Ternário?

Se a lógica for **complexa** ou precisar de várias condições aninhadas.

Se a leitura do código ficar difícil, prefira if-else.

Exemplo **NÃO recomendado**:

```
$resultado = ($idade < 12) ? "Criança" : (($idade < 18) ? "Adolescente" : "Adulto");
```

Neste caso, um if-else seria mais claro!

Lista de exercícios

01 - Identificação do Mês

Crie um programa que leia um número de 1 a 12 e informe o nome do mês correspondente. Caso o número esteja fora dessa faixa, exiba uma mensagem de erro.

02 - Calculadora de IMC

Crie um programa que leia o peso (em kg) e a altura (em metros) de uma pessoa e calcule o **Índice de Massa Corporal (IMC)**. O IMC deve ser calculado utilizando a fórmula:

$$\text{IMC} = \text{peso} / (\text{altura} * \text{altura})$$

Em seguida, classifique a pessoa de acordo com o IMC calculado:

- Abaixo de 18.5: **Abaixo do peso**
- De 18.5 a 24.9: **Peso normal**
- De 25 a 29.9: **Sobrepeso**
- De 30 ou mais: **Obesidade**

03 - Verificação de Ano Bissexto

Crie um programa que leia um ano e informe se ele é **bissexto** ou não. Um ano é bissexto se:

- É divisível por 4, mas não por 100.
- Ou é divisível por 400.

04 - Calculadora de Desconto

Crie um programa que leia o valor de uma compra e informe o preço final após aplicar um **desconto**. Se o valor da compra for superior a **R\$ 100**, aplique um **desconto de 10%**. Caso contrário, aplique um desconto de **5%**.

05 - Verificação de Triângulo

Crie um programa que leia os **três lados** de um triângulo (números positivos) e informe se o triângulo é:

- **Equilátero**: todos os lados são iguais.
- **Isósceles**: dois lados são iguais.
- **Escaleno**: todos os lados são diferentes.

Caso os lados informados não formem um triângulo válido (por exemplo, a soma de dois lados deve ser maior que o terceiro), exiba uma mensagem de erro.

06 - Ordem Crescente de Três Números

Crie um programa que leia **três números** e exiba-os em ordem crescente. Se os números forem iguais, exiba uma mensagem dizendo que todos são iguais.

Estrutura de repetição

As **estruturas de repetição** (ou **loops**) são usadas em programação para **executar um bloco de código múltiplas vezes** enquanto uma condição for verdadeira.

Elas evitam a repetição manual de código, tornando o programa mais eficiente e organizado.

While (Enquanto)

Executa um bloco de código **enquanto** uma condição for verdadeira.

Sintaxe:

```
while (condição) {  
    // Código a ser repetido  
}
```

Exemplo - Contar de 1 a 5:

```
<?php  
$contador = 1;  
  
while ($contador <= 5) {  
    echo "Número: $contador <br>";  
    $contador++; // Incrementa o contador  
}  
?>
```

Explicação: Enquanto \$contador for menor ou igual a 5, o loop continuará rodando.

Do-While (Faça...Enquanto)

Semelhante ao while, mas **executa o código pelo menos uma vez** antes de verificar a condição.

Sintaxe:

```
do {  
    // Código a ser repetido  
} while (condição);
```

Exemplo - Contar de 1 a 5:

```
<?php  
$contador = 1;
```

```
do {  
    echo "Número: $contador <br>";  
    $contador++;  
} while ($contador <= 5);  
>
```

Diferença do while: Se \$contador começasse com 6, o while **nunca rodaria**, mas o do-while **executaria pelo menos uma vez** antes de verificar a condição.

For (Para)

Usado quando sabemos **quantas vezes** o loop deve ser executado.

Sintaxe:

```
for (inicialização; condição; incremento) {  
    // Código a ser repetido  
}
```

Exemplo - Contar de 1 a 5:

```
<?php  
for ($i = 1; $i <= 5; $i++) {  
    echo "Número: $i <br>";  
}  
?>
```

Explicação:

\$i = 1; → Inicializa a variável.

\$i <= 5; → Define a condição de execução.

\$i++ → Incrementa o contador a cada iteração.

foreach (Para cada)

Usado para percorrer **arrays** de forma simplificada.

Sintaxe:

```
foreach ($array as $valor) {  
    // Código a ser repetido para cada valor  
}
```

Exemplo - Exibir uma lista de frutas:

```
<?php
```

```
$frutas = ["Maçã", "Banana", "Laranja", "Manga"];

foreach ($frutas as $fruta) {
    echo "Fruta: $fruta <br>";
}
?>
```

Explicação: O loop percorre o array \$frutas, armazenando cada item na variável \$fruta.

Quando usar cada tipo de loop?

- **While**
 - ✓ Quando **não sabemos exatamente** quantas vezes o loop deve rodar.
- **Do-While**
 - ✓ Quando precisamos executar pelo menos uma vez antes de verificar a condição.
- **For**
 - ✓ Quando sabemos quantas vezes o loop deve ser repetido.
- **Foreach**
 - ✓ Quando queremos percorrer arrays de forma simples e eficiente.

As estruturas de repetição são essenciais para **automatizar tarefas repetitivas** e **tornar o código mais eficiente!**

Lista de exercícios

01 - Contagem de 1 a 100

Crie um programa para imprimir de **1 a 100**. Em seguida, faça o programa imprimir somente os números **ímpares** compreendidos entre 1 e 100 (inclusive).

02 - Tabuada de um Número

Crie um programa que leia um número e, usando o laço while, imprima a **tabuada** desse número de 1 a 10. Exemplo de saída para o número 5:

```
5 x 1 = 5
5 x 2 = 10
...
5 x 10 = 50
```

03. Soma de Números

Crie um programa que leia números positivos do usuário e calcule e imprima a soma desses números até que o usuário digite um número negativo (quando o programa deve parar de pedir entradas e mostrar a soma final).

04. Fatorial de um Número

Crie um programa que leia um número e calcule seu **fatorial**. O fatorial de um número N é dado pela multiplicação de todos os números inteiros de N até 1 (exemplo: $5! = 5 \times 4 \times 3 \times 2 \times 1$). O programa deve exibir o valor do fatorial.