

Introdução ao Desenvolvimento Web com PHP



Autores:

Cristiane Akemi Yaguinuma

Fernando Vieira Duarte

Francisco Rocha Pirolla

José Arnaldo Mascagni de Holanda

Prefácio

O objetivo desta apostila é fornecer uma introdução ao desenvolvimento web com a linguagem PHP, apresentando desde elementos de sintaxe de programação básica até exemplos de websites com tratamento de sessão, cookies e envio de e-mails. A organização da apostila utilizou como base o livro de Niederauer (2017), do qual foram estendidos diversos exemplos de programas PHP e exercícios de fixação.

O desenvolvimento deste material foi resultado do trabalho de um grupo de estudo formado por docentes do Instituto Federal de São Paulo, Câmpus Araraquara, com experiência na área de programação e banco de dados. O conteúdo foi especialmente elaborado tanto para estudantes quanto para docentes que desejam iniciar seus estudos no desenvolvimento de sistemas web utilizando linguagem PHP. Bons estudos!

SUMÁRIO

Capítulo 1 - Introdução à arquitetura web	6
1.1. Programação desktop vs. programação web	6
1.2. Sites de conteúdo estático	6
1.3. Sites de conteúdo dinâmico e aplicações web	7
1.4. Introdução ao PHP	8
1.5. Histórico	8
1.6. Arquitetura Cliente/Servidor com PHP	9
1.7. Instalação e configuração do PHP	10
Capítulo 2 - Introdução à Linguagem PHP	11
2.1. Estrutura de um Programa PHP	11
2.1.1. Exibição de páginas no browser	14
2.1.2. Exercícios de fixação	16
2.3. Comandos de Saída de Dados	17
2.3.1. Comando echo	17
2.3.2. Comando print	17
2.3.3. Comando printf	18
2.4. Manipulação de dados em PHP	18
2.4.1. Dados Numéricos	18
2.4.2. Dados Alfanuméricos (Textos)	19
Aspas simples ('')	19
Aspas duplas (")	20
2.4.3. Variáveis	21
Maiúsculas e minúsculas	22
Variáveis numéricas	23
Variáveis alfanuméricas (strings)	23
Conversão de variáveis	24
2.4.4. Exercícios de fixação	25
2.4.5. Interpolação de variáveis	26
2.4.6. Operadores	28
Operadores aritméticos	28
Operadores de comparação	30
Operadores de atribuição	31
Operadores lógicos	32
Operador ternário	32

Precedência de operadores	33
2.2.7. Constantes	34
2.2.8. Exercícios de Fixação	35
Capítulo 3 - Entrada de dados	36
3.1 Formulários em HTML	36
3.1.1. Tag input	37
3.1.2. Tag select	40
3.1.3. Tag textarea	40
3.2 Envio de informações para um programa PHP	40
3.2.1. Método GET	41
3.2.2. Método POST	42
3.3. Recebimento dos dados	42
3.4 Exercícios de Fixação	43
Capítulo 4 - Estruturas de Controle	44
4.1. Estruturas de Controle Condicional	44
4.1.1. Comando if	45
4.1.2. Comando switch	49
4.1.1 Exercícios de Fixação	51
4.2 Estruturas de Controle de Repetição	54
4.2.1 Comando FOR	54
4.2.2 Comando WHILE	55
4.2.3 Comando DO..WHILE	56
4.2.4 Comando FOREACH	57
4.3. Juntando tudo!	57
4.4. Exercícios de Fixação	61
Capítulo 5 - Variáveis compostas	65
5.1. Vetores	65
5.1.1. Leitura e escrita	66
5.2. Matrizes	67
5.3. Exercícios de Fixação	69
Capítulo 6 - Sessões e Cookies	72
6.1. Sessões	72
6.1.1. Juntando tudo	73
6.1.2. Exercícios de fixação	76
6.2. Cookies	78
6.2.1 Juntando tudo	80
6.3. Exercícios de fixação	84

Capítulo 7 - Funções	85
7.1. Definição	85
7.2. Escopo de variáveis	87
7.2.1. Passagem de parâmetros: valor e referência	87
7.3. Funções recursivas	88
7.4. Exercícios de fixação	89
Capítulo 8 - Includes em PHP	90
8.1. Definição	90
8.2. Includes e funções	91
8.3. Juntando tudo	91
8.4. Exercícios de Fixação	95
Capítulo 9 - Manipulação de arquivos	97
9.1. Funções de manipulação de arquivos	97
9.1.1. Abertura – fopen()	97
9.1.2. Fechamento – fclose()	99
9.1.3. Leitura – fread()	99
9.1.4. Leitura linha a linha – fgets()	100
9.1.5. Escrita – fwrite()	100
9.2. Exemplo completo	101
9.3. Exercícios de fixação	102
Capítulo 10 - Envio de e-mails com PHP	104
10.1. Função mail	104
10.2. Configurando o sendmail no WAMP Server	105
10.3. Juntando tudo	106
10.4. Exercícios de fixação	108
Referências Bibliográficas	110

Capítulo 1 - Introdução à arquitetura *web*

1.1. Programação desktop vs. programação *web*

As aplicações desktop são escritas usando uma linguagem de programação (C, Java, C#, etc.) e geram um programa (software). Este software é instalado e executado em um computador realizando tarefas específicas. Também podem ser usados por vários usuários em rede.

As linguagens de programação *web*, por sua vez, permitem desenvolver *websites* ou *sites* acessados via internet por meio de programas chamados navegadores ou *browsers*. Os sites podem ser classificados quanto à natureza de seu conteúdo como: sites de conteúdo estático, que não sofrem grandes alterações ao longo do tempo, e sites de conteúdo dinâmico, que utilizam aplicações *web* para responder às requisições dos usuários.

1.2. Sites de conteúdo estático

Os sites estáticos não sofrem grandes alterações em seu conteúdo ao longo do tempo, pois contêm páginas escritas diretamente usando a linguagem de marcação HTML (*HyperText Markup Language*), sem o apoio de aplicações que automatizam a geração do seu conteúdo. Os sites estáticos são similares às páginas de uma revista ou de um jornal impresso, cujo conteúdo não se modifica com a interação do leitor.

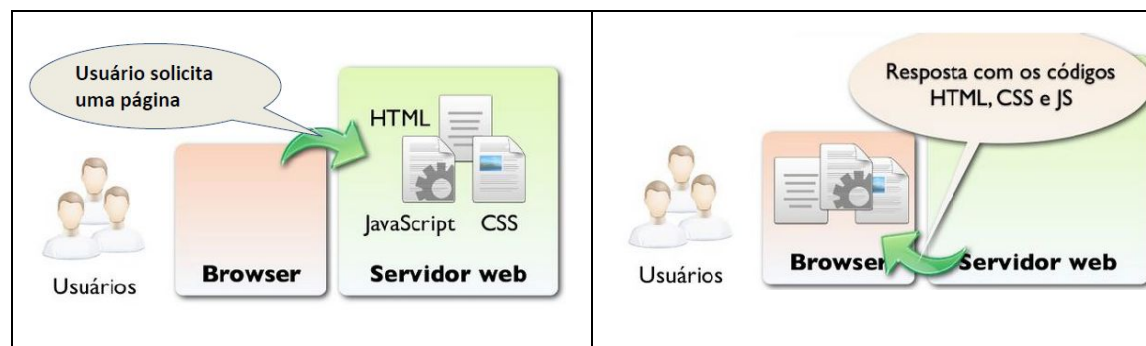


Figura 1: Funcionamento de um site de conteúdo estático.

Fonte: (WATANABE, 2018).

A Figura 1 ilustra o funcionamento geral de um site de conteúdo estático. Primeiramente, os desenvolvedores do site escrevem o conteúdo como páginas HTML, podendo utilizar também folhas de estilo *Cascading Style Sheets* (CSS) e scripts na linguagem JavaScript (JS). Este conteúdo é então disponibilizado por meio de um servidor web. Para

acessá-lo, os usuários devem utilizar um navegador ou *browser* para solicitar uma página web ao servidor digitando um endereço URL (por ex.: <http://sitedeinteresse.com>). Por fim, o servidor web responde à solicitação enviando os códigos HTML, CSS e JS originalmente escritos pelos desenvolvedores, que são processados e exibidos pelo navegador aos usuários.

1.3. Sites de conteúdo dinâmico e aplicações web

Os sites dinâmicos, por sua vez, permitem que o conteúdo seja gerado por *aplicações web* que realizam o processamento de informações enviadas pelo usuário. Assim, os sites tornam-se interativos e seu conteúdo pode ser modificado com facilidade. Exemplos de sites de conteúdo dinâmico são portais de notícias, sites de comércio eletrônico, redes sociais, entre outros.

As aplicações web, que geram o conteúdo dos sites dinâmicos, são escritas usando uma combinação de linguagem de marcação (por ex.: HTML) com linguagem de programação web (por ex.: PHP). As aplicações web são executadas em um servidor web, cuja função é receber uma solicitação (requisição), geralmente via formulário web, e devolver uma resposta para o cliente, como uma página HTML, por exemplo. A Figura 2 ilustra esse processo.

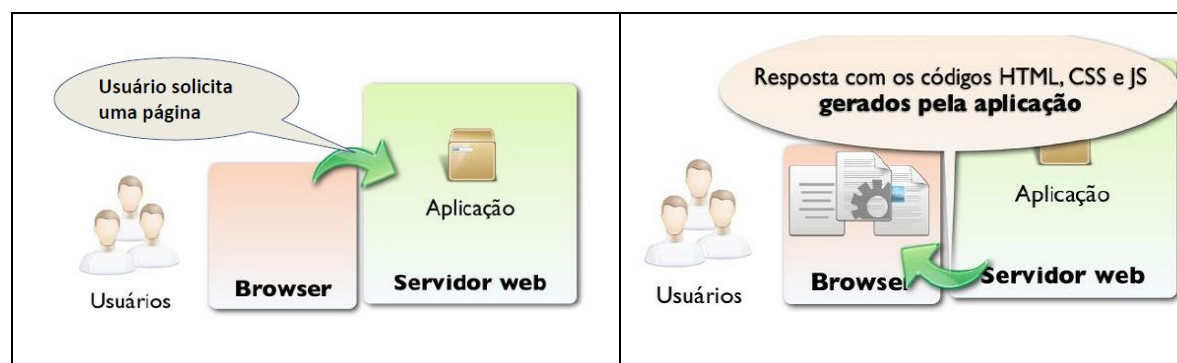


Figura 2: Funcionamento de um site de conteúdo dinâmico com aplicação web.
Fonte: (WATANABE, 2018).

A Figura 2 ilustra o funcionamento geral de um site de conteúdo dinâmico. Primeiramente, os desenvolvedores do site programam uma aplicação web combinando linguagem de marcação (por ex.: HTML) com linguagem de programação web (por ex.: PHP). Esta aplicação é então carregada em um servidor web. Os usuários devem utilizar um navegador ou *browser* para solicitar ao servidor o processamento de uma página web, por exemplo um formulário preenchido e submetido via clique de botão. O servidor recebe essa solicitação e a direciona para a aplicação web realizar o processamento dos valores preenchidos e gerar uma resposta com os códigos HTML, CSS e JS. Por fim, o servidor web envia a resposta gerada pela aplicação web para o navegador, que a exibe para os usuários.

Existem várias linguagens de programação para desenvolver aplicações *web*, sendo que a linguagem PHP é uma das mais populares, com a vantagem de ser gratuita e de código aberto.

1.4. Introdução ao PHP

O PHP (um acrônimo recursivo para *PHP: Hypertext PreProcessor*) é uma linguagem de *script open source* de uso geral, muito utilizada, e especialmente adequada para o desenvolvimento *web* e que é embutida dentro de uma página HTML (*HyperText Markup Language*).

É uma linguagem interpretada e executada em um servidor *web*, ou seja, os scripts escritos em PHP são interpretados e executados pelo próprio servidor da *web*. O resultado do processamento é inserido na página e enviado ao navegador do usuário em formato HTML. Desta forma, usando PHP é possível transformar sites estáticos, com HTML puro, em sites interativos ou dinâmicos.

Trata-se de uma linguagem gratuita e open-source (software livre de código-fonte aberto), ou seja, seu código-fonte pode ser adaptado para diferentes finalidades, independentemente de plataforma, sendo possível executar o PHP no Windows, Unix ou Linux.

Outra característica é a tipagem dinâmica ou fracamente tipada, onde uma variável pode armazenar tipos de dados diferentes durante a execução do código.

O código PHP é delimitado pelas instruções de processamento (*tags*) de início e fim `<?php` e `?>`, que permite a escrita de códigos PHP dentro da página HTML. A instalação pode ser realizada a partir do site <http://www.php.net>. Neste site também são disponibilizadas documentação e correções para os defeitos (*bugs*) encontrados em versões anteriores.

1.5. Histórico

O PHP foi criado por Rasmus Lerdorf em 1994. Essa linguagem era formada por um conjunto de scripts em linguagem C, voltados à criação de páginas dinâmicas, que Rasmus utilizava para monitorar o acesso ao seu currículo na Internet. Embora isso possa parecer extremamente rudimentar para os nossos dias, vale lembrar que tais ferramentas não estavam disponíveis naquela época. Esta inovação rendeu a Lerdorf uma avalanche de e-mails de pessoas curiosas em saber como isso era possível. Devido ao grande sucesso que sua página obteve, Lerdorf resolveu distribuir seu conjunto de ferramentas, que permitia a interação com banco de dados, ao qual deu o nome de *Personal Home Page* (PHP).

A segunda versão foi lançada em 1997 e denominada PHP/FI 2.0. Nesta época, aproximadamente 60 mil domínios (1% da Internet) já utilizavam PHP, que era mantido principalmente por Rasmus.

O crescente sucesso do PHP levou Andi Gutmans e Zeev Suraski, estudantes que utilizavam a linguagem para um projeto acadêmico de comércio eletrônico, resolvessem

cooperar com Rasmus para o aprimorar o PHP. Este trabalho em conjunto deu início ao PHP 3, disponibilizado oficialmente em 1998. Entre as principais características do PHP 3 estavam a extensibilidade, a possibilidade de conexão com vários bancos de dados, novos protocolos, uma sintaxe mais consistente, suporte à orientação a objetos e uma nova API, que possibilitava a criação de novos módulos e que atraiu vários desenvolvedores para o PHP.

No final de 1998 o PHP já estava presente em cerca de 10% dos domínios da Internet. Nessa época, o significado da sigla PHP mudou para PHP: Hypertext Preprocessor, retratando a nova realidade de uma linguagem de propósitos mais amplos.

Mais tarde, Zeev e Andi modificaram por completo o PHP 3, dando origem ao PHP 4 em 2000. Embora a versão 4 do PHP tenha sido utilizada por muitos anos, ela possuía alguns problemas sérios em relação à orientação a objetos, que somente foram definitivamente corrigidos em sua versão 5, disponibilizada oficialmente em julho de 2004.

1.6. Arquitetura Cliente/Servidor com PHP

Na arquitetura cliente/servidor, o computador cliente envia uma solicitação para o servidor através da conexão de rede. A solicitação é processada pelo servidor que retorna a resposta para o cliente. A internet é baseada na arquitetura cliente/servidor, onde o servidor *web* processa solicitações de clientes simultaneamente.

A arquitetura cliente/servidor em PHP funciona conforme a Figura 3.

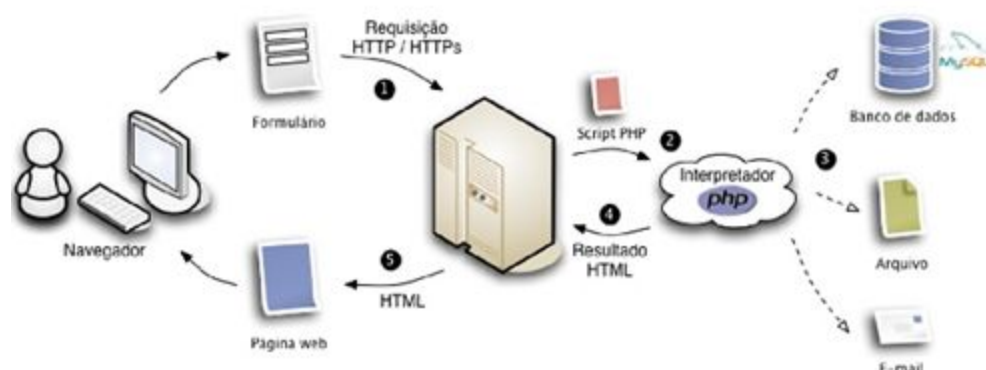


Figura 3: Processamento de um script PHP na arquitetura cliente/servidor.

Fonte: (LIMA JR, 2018)

Primeiramente, o usuário cliente preenche e envia o formulário ao servidor web por meio de uma requisição HTTP ❶. No servidor, uma aplicação PHP deve responder à sua requisição. Quando recebe seus dados, o interpretador PHP processa o código (script PHP) ❷. Dependendo das instruções, o um banco de dados pode ser acessado (consultado, atualizado, etc.), um e-mail é enviado, etc. ❸. Uma página HTML é montada pelo interpretador

PHP contendo os resultados desta requisição ④. Finalmente, a página web gerada no formato html é enviada e carregada pelo navegador do cliente ⑤.

1.7. Instalação e configuração do PHP

Para desenvolver aplicações utilizando PHP geralmente se utiliza duas ferramentas básicas: um servidor web com suporte a PHP e um servidor de banco de dados (por ex.: MySQL). Para se preparar o ambiente de trabalho são necessários downloads e instalações das versões atualizadas destes servidores.

Para facilitar essa preparação inicial, existem pacotes que incluem essas e outras ferramentas em um único instalador, entre eles estão:

- XAMPP: X (para qualquer sistemas operacionais), Apache, MySQL, PHP, Perl. Pode ser obtido em: https://www.apachefriends.org/pt_br/index.html
- WAMP: Windows, Apache, MySQL, PHP - Perl – Python. Pode ser obtido em: <http://www.wampserver.com/en/>
- VertrigoServ: PHP, Apache, MySQL. Pode ser obtido em: <https://www.vswamp.com/>.

Estas ferramentas de desenvolvimento permitem aos programadores testarem seus códigos em seus próprios computadores, sem necessitar acesso algum à Internet. É importante ressaltar que é possível utilizar um servidor de hospedagem com suporte a PHP com banco de dados MySQL para o funcionamento de scripts PHP sem a necessidade da instalação de softwares.

Capítulo 2 - Introdução à Linguagem PHP

Depois de estudar os conceitos básicos sobre arquitetura *web*, instalar e configurar corretamente o PHP, é hora de começar a escrever os primeiros programas em PHP. Neste capítulo, serão explicados códigos em PHP para exibir informações na tela do *browser*, manipular diferentes tipos de dados, além de aplicar operadores para realizar comparações entre os dados.

2.1. Estrutura de um Programa PHP

Primeiramente, abra um editor de texto (Bloco de Notas, Notepad++) e escreva o seguinte código:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8" />
    <title>PHP</title>
  </head>
  <body>
    <?php
      echo "<h2>Olá mundo!</h2>";
    ?>
  </body>
</html>
```

Salve esse programa como *programa1.php*, dentro da pasta *www* do *WampServer* (*C:\wamp64\www* ou *C:\wamp\www*) ou na pasta *C:\xampp\htdocs* do *Xampp*. Esse primeiro programa vai gerar como resultado a frase “*Olá mundo!*” na página.

Para ver o resultado, acesse pelo *browser* (navegador) o endereço *http://localhost/programa1.php*, como mostra a Figura 4. Vamos ver com detalhes cada uma das linhas do código do programa.

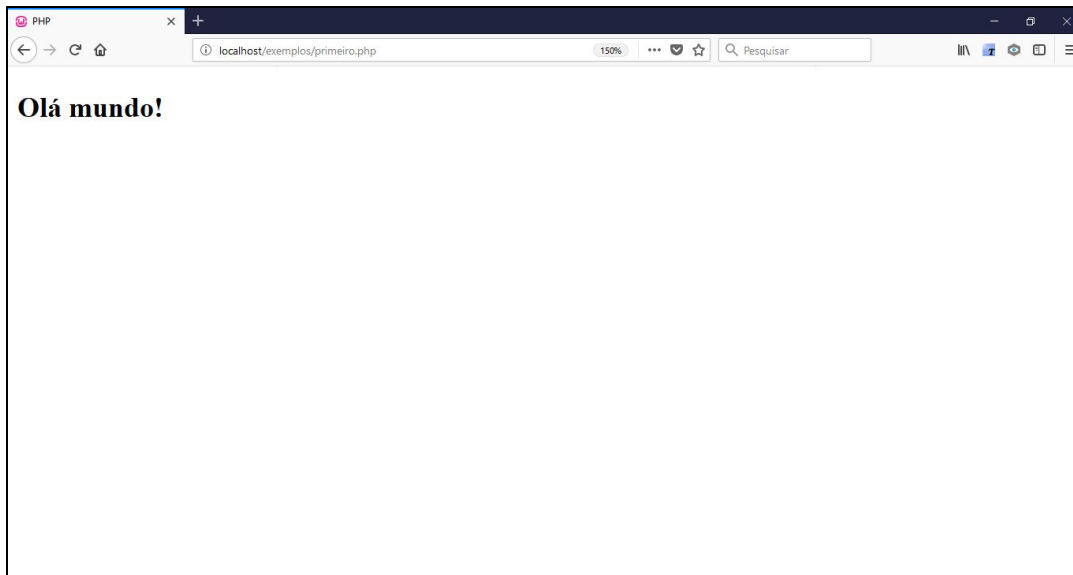


Figura 4: Processamento de um script PHP na arquitetura cliente/servidor.
Fonte: próprios autores.

O trecho de programação PHP deve estar entre as tags `<?php` e `?>`, para que o servidor web saiba que esse trecho deve ser processado.

Elemento	Descrição
<code><?php</code>	Informa que aqui começa um programa PHP.
<code>//</code>	Representam uma linha de comentário. Tudo que vem após estas barras na mesma linha é ignorado pelo PHP. Os comentários são úteis para uma boa documentação do seu programa. As duas barras servem para transformar uma única linha em comentário, mas pode-se usar o <code>/*</code> para iniciar uma sequência de comentários e depois finalizar os comentários com o <code>*/</code> .
<code>echo</code>	É um dos comandos mais utilizados em PHP. Serve para escrever alguma coisa na tela.
<code>?></code>	Informa que aqui termina o programa PHP.

Por meio da opção **Exibir** → **Código-fonte** do seu browser, pode-se ver o código recebido pelo browser, que foi o seguinte:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <title>PHP</title>
  </head>
  <body>
    <h2>Olá mundo!</h2>
  </body>
</html>
```

Note que o browser não recebe nenhuma linha em PHP, somente recebe código HTML puro, pois, como já foi comentado, o PHP executa no servidor. Toda a programação em PHP é processada no servidor, que retorna somente o resultado final para o browser.

Dica: quando as páginas possuírem extensão *.html*, o servidor web as tratará como HTML puro, e não reconhecerá códigos PHP. Se a página possuir extensão *.php*, o servidor web ativará o processador de hipertexto do PHP para verificar linha a linha em busca de códigos de programação, por isso o processo fica um pouco mais lento. A dica é a seguinte: só coloque extensão *.php* nas páginas que realmente possuem códigos PHP, senão será gasto um tempo desnecessário, procurando, em cada linha, códigos que não existem na página.

Como pode-se notar, um programa escrito em PHP pode possuir comandos em HTML e códigos em PHP.

Os comandos HTML devem aparecer fora das tags **<?php** e **?>**, pois essas tags delimitam um trecho de programa PHP. Dentro dessas tags até podem aparecer comandos HTML, mas somente se utilizarmos o comando **echo** para escrevê-los.

Pode-se concatenar scripts PHP com comandos HTML e, assim, escrever vários scripts PHP em uma única página. Cada script PHP existente na página deve começar com a tag **<?php** e terminar com **?>**. As linhas de programação que serão escritas entre as tags devem sempre terminar com ; (ponto e vírgula), senão ocorrerão erros no momento de execução da página. Entre essas tags, pode-se escrever programas utilizando todos os recursos que o PHP oferece, tais como definição e chamada de funções, acesso a banco de dados, atribuição de valores a variáveis, fórmulas matemáticas, entre outros.

A mistura entre o HTML e o PHP é muito útil, pois pode-se utilizar o PHP para gerar os dados dinamicamente, enquanto o HTML é usado para formatar e exibir esses dados nas páginas apresentadas no browser.

Por exemplo, vamos misturar HTML e PHP para mostrar a data atual. Digite o seguinte programa em seu editor de textos e salve como *programa2.php*:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <title>Data atual</title>
  </head>
  <body>
    <?php
      $data_de_hoje = date("d/m/Y", time());
    ?>
    <p align="center">
      Hoje é dia <?php echo $data_de_hoje; ?>
    </p>
  </body>
</html>
```

Note a combinação existente entre os comandos HTML e código PHP. No início do programa, a data atual foi atribuída à variável `$data_de_hoje`, utilizando o comando `date`. Em PHP as variáveis começam sempre pelo símbolo de cifrão (\$). Depois, utilizamos comandos HTML para escrever “Hoje é dia”, e completamos abrindo um novo trecho PHP, escrevendo a data atual armazenada na variável `$data_de_hoje`, por meio do comando `echo`.

2.1.1. Exibição de páginas no browser

Para o browser mostrar alguma coisa na tela é necessário que a página tenha pelo menos um comando `echo` para escrever algo, ou então comandos HTML que escrevam o conteúdo da página. Porém, somente se for usado o comando `echo` ou algum outro comando PHP que produza uma saída na tela, a página terá informações dinâmicas, pois o HTML é estático, imprime sempre as mesmas informações na tela.

Veja o seguinte exemplo:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <title>Exibição no Browser</title>
  </head>
  <body>
    <?php
      $dia = date("d/m/Y", time());
      $base = 5.5;
      $altura = 10;
      $area = $base * $altura;
    ?>
  </body>
</html>
```

Salve esse programa como *programa3.php* e veja o resultado em seu *browser*. Note que não há nenhum comando `echo` no programa, por isso seu *browser* mostrará uma tela em branco. Os valores atribuídos às variáveis ficaram armazenados apenas na memória, mas não foram mostrados na tela. Ao visualizar o código-fonte recebido pelo *browser*, vemos apenas as tags do HTML:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <title>Exibição no Browser</title>
  </head>
  <body>
  </body>
</html>
```

Vamos utilizar agora o comando `echo` para mostrar as informações na tela. Digite o seguinte código em seu editor de texto e salve-o como *programa4.php*:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <title>Exibição no Browser</title>
  </head>
  <body>
    <?php
      $curso = "Técnico em Informática Integrado ao Ensino Médio";
      $ano = "2o";
      $disciplina = "IEB";
      $professor1 = "Fernando";
      $professor2 = "José Arnaldo";
      $frase1 = "Curso: $curso";
      $frase2 = "$ano ano - Disciplina: $disciplina";
      $frase3 = "Professores: $professor1 e $professor2";
      echo "<h1>$frase1</h1>";
      echo "<h2>$frase2</h2>";
      echo "<h3>$frase3</h3>";
    ?>
  </body>
</html>
```

A Figura 5 mostra o resultado gerado pelo *programa4.php* quando abrimos a página no *browser*.

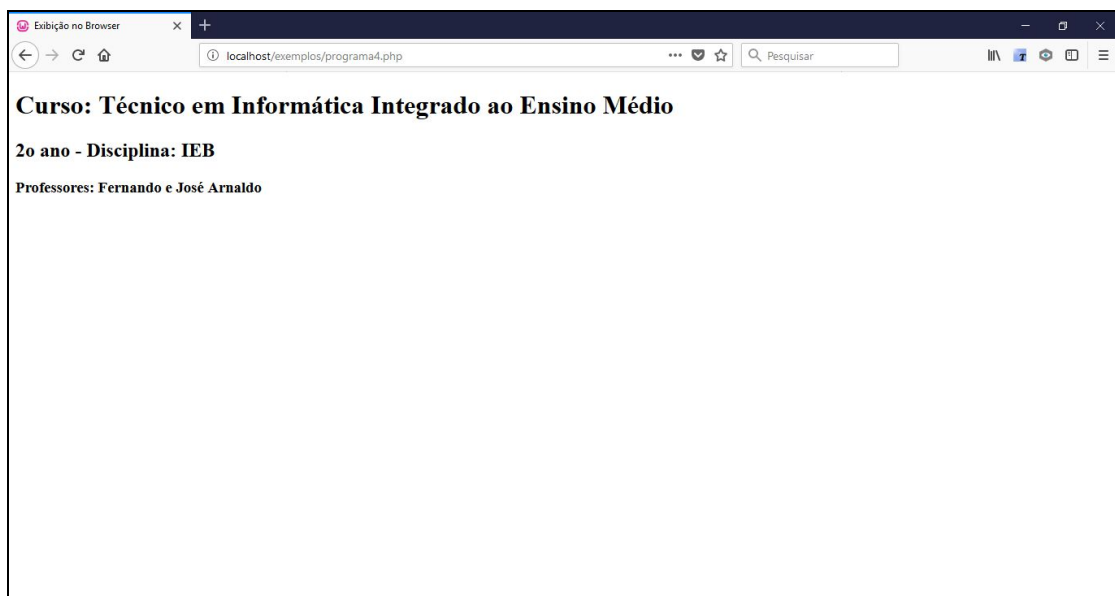


Figura 5: Processamento de um script PHP na arquitetura cliente/servidor.
Fonte: próprios autores.

Lembre-se que em PHP as variáveis começam sempre pelo símbolo de cifrão (\$). Quando houver um cifrão diante de um nome qualquer, o PHP interpretará isso como uma variável.

2.1.2. Exercícios de fixação

Exercício 1. O PHP é:

- a) Uma linguagem de programação (scripts) que é executada no navegador do usuário.
- b) Uma linguagem de programação (scripts) usada na criação de páginas web dinâmicas.
- c) Uma linguagem de programação (scripts) usada pelos provedores da Internet.
- d) As alternativas A e B estão corretas.

Exercício 2. As páginas construídas com PHP:

- a) São processadas no navegador do usuário.
- b) São processadas no navegador do usuário, desde que esse navegador possua um suporte run-time (em tempo de execução) para a linguagem PHP.
- c) São processadas no servidor web.
- d) Nenhuma das alternativas anteriores.

Exercício 3. Os scripts processados em PHP são delimitados por:

- a) `</php e />`
- b) `<&php e &>`
- c) `<* e *>`
- d) `<?php e ?>`

Exercício 4. Com base no *programa4.php*, faça um programa em PHP que armazena os seguintes dados em variáveis e utiliza comandos echo para apresentar os dados armazenados nessas variáveis:

- Nome (atribua à variável o seu nome);
- Idade (atribua à variável sua idade);
- Sexo (atribua à variável o seu sexo);
- Endereço (atribua à variável o seu endereço);
- Cidade (atribua à variável a sua cidade);
- Telefone (atribua à variável o seu telefone).

2.3. Comandos de Saída de Dados

A exibição de dados em uma linguagem de programação é uma das operações mais fundamentais. Em PHP, é possível imprimir na tela do navegador por intermédio de três

funções principais diferentes: **echo**, **print** e **printf**. Cada uma delas, possui suas próprias características, que veremos adiante.

2.3.1. Comando echo

O comando **echo** é o comando mais comum da linguagem PHP para a saída de qualquer tipo de informação, seja ela texto, números ou variáveis. Abaixo, é apresentado um exemplo de uso desse comando.

```
<?php
    $nome = "Ana Maria";
    echo "<h2>$nome, seja bem-vinda ao comando echo.</h2>";
?>
```

2.3.2. Comando print

O uso do comando **print** é similar ao comando **echo**. A diferença entre os dois comandos está no fato que o comando **print** retorna um valor, que pode ser usado para verificar se a mensagem foi realmente exibida. Por esta razão, o comando **echo** é um pouco mais rápido que o comando **print**.

```
<?php
    $nome = "Ana Maria";
    print "<h2>$nome, seja bem-vinda ao comando print.</h2>";
?>
```

2.3.3. Comando printf

A função **printf** é utilizada para exibir no navegador uma *string* (cadeia de caracteres) formatada. Seu uso é bastante similar ao do comando **printf** da linguagem C. Um exemplo pode ser visto a seguir.

```
<?php
    $nome = "Ana Maria";
    printf("<h2>%s, seja bem-vinda ao comando printf.</h2>", $nome);
?>
```

Os valores das variáveis passadas como argumentos à função são inseridos na *string* por meio dos especificadores com sinais de porcentagem (%). Assim, no exemplo, o conteúdo da variável `$nome` é inserido no mesmo local da string onde está o especificador `%s`.

Os especificadores mais comuns para a formatação de strings são:

- `%c` para caracteres
- `%d` para números decimais
- `%f` para números em ponto flutuante
- `%s` para *strings*

2.4. Manipulação de dados em PHP

Uma das etapas fundamentais para dominar uma linguagem de programação é aprender a manipular apropriadamente os diversos tipos de dados. A seguir, veremos como tratar dados numéricos e literais, assim como criar e utilizar variáveis e constantes. Também serão vistos vários operadores fornecidos pela linguagem PHP para produzir expressões lógicas, aritméticas e relacionais.

2.4.1. Dados Numéricos

Os dados numéricos podem ser números inteiros, reais, positivos, decimais, octais ou hexadecimais. Esses dados normalmente são utilizados para efetuar cálculos. Para representar números muito pequenos ou muito grandes, pode-se utilizar a notação científica, por exemplo, 1.500.000.000 pode ser expresso como 1.5E+9. Seguem alguns exemplos de dados numéricos:

Dados	Descrição
5	Valor inteiro na base decimal.
4.012	Valor real (ou ponto flutuante) com três casas decimais.
.14	Valor real com duas casas decimais. É o mesmo que 0.14. Quando o número começa com um ponto, o 0 (zero) é considerado o algarismo inicial.
033	Valor inteiro na base octal. Todo valor iniciado com 0 (zero) é tratado como base 8 (na base 8 são utilizados apenas os algarismos 0, 1, 2, 3, 4, 5, 6, 7).
0xBC	Valor inteiro na base hexadecimal. Todo valor iniciado com 0x é tratado como

	base 16 (na base 16 são utilizados os 10 algarismos do sistema decimal e mais as letras A, B, C, D, E e F, que representam os valores de 10 a 15).
43000000	É um número real grande e que pode ser expresso em notação científica: 4.3E+7.

2.4.2. Dados Alfanuméricos (Textos)

Os dados alfanuméricos também são conhecidos como strings. São sequências de caracteres, que podem ser delimitadas por aspas simples (') e aspas duplas ("), dependendo da utilização desejada. O PHP trata os dados alfanuméricos de forma diferente, de acordo com o delimitador utilizado. Vejamos alguns exemplos de cada um dos delimitadores.

Aspas simples (')

Esse tipo de aspas pode ser usado para delimitar qualquer dado alfanumérico (sequência de caracteres), por exemplo:

```
<?php
    echo '<p>Texto utilizando aspas simples</p>';
?>
```

Deve-se ter cuidado com textos que possuem o caractere ' em uma string, por exemplo:

```
<?php
    echo '<h2>Welcome to the John's Page</h2>';
?>
```

Como existe uma aspa simples no meio da string, o PHP vai interpretá-la como delimitador de finalização do texto, e isso causará um erro de execução. Esse problema pode ser facilmente resolvido inserindo antes da aspa o caractere de controle \ (barra invertida), que indica ao PHP que aquela aspa deve ser tratada como um texto comum, e não como um delimitador. Portanto, a forma correta da string seria:

```
<?php
    echo '<h2>Welcome to the John\'s Page</h2>';
?>
```

Para apresentar quebras de linha no formato HTML, para que sejam exibidas na tela, deve-usar a tag **
** do HTML. Por exemplo:

```
<?php
    echo 'Isto é um teste.<br/>Coloquei uma quebra de linha';
?>
```

Aspas duplas (")

As aspas duplas são muito parecidas com as aspas simples. Uma diferença importante entre esses dois tipos de delimitadores é que, com as aspas duplas, pode-se fazer uma interpolação de variáveis. Uma interpolação de variáveis consiste em incluir o valor de uma variável dentro de outra, por exemplo:

```
<?php
    $palavra = "teste";
    $frase = "Isto é um $palavra.";
    echo "$frase";
?>
```

Outra diferença entre as aspas simples e as aspas duplas é que, utilizando as aspas duplas como delimitadores, pode-se incluir sequências de caracteres de controle nos dados alfanuméricos. Algumas sequências de controle que podem ser inseridas são:

Controle	Descrição
\"	Insere no texto o caractere \".
\'	Insere no texto o caractere \'.
\\$	Insere no texto o caractere \$.
\\	Insere no texto o caractere \.

Além disso, um dado alfanumérico delimitado por aspas duplas pode conter normalmente um caractere ' (aspas simples), por exemplo:

```
<?php
    echo "<h2>Welcome to the John's Page</h2>";
?>
```

Entretanto, para indicar que as aspas fazem parte do texto é necessário incluir o caractere de controle (ou de escape) \, tal como:

```
<?php
    echo "Estou colocando \"aspas duplas\" dentro de um texto";
?>
```

2.4.3. Variáveis

As variáveis servem para armazenar dados que podem ser usados em qualquer ponto do programa. Cada variável está associada a uma posição de memória do computador. Diferentemente de outras linguagens de programação, tais como C, Pascal e Java, no PHP não é necessário fazer declaração de variáveis. Basta atribuir diretamente um valor para uma variável, e a partir desse momento já está criada e associada a um tipo (numérico, alfanumérico, entre outros), dependendo do valor que lhe foi atribuído.

Como já foi visto, em PHP as variáveis devem iniciar com o símbolo \$. Após esse símbolo deve vir um identificador (nome) da variável, que será utilizado para referenciá-la durante a execução do programa. Esse identificador não pode iniciar com um número. Os números podem aparecer em qualquer posição do identificador, menos na primeira. Alguns exemplos de variáveis válidas e inválidas:

Válidas
\$nota1
\$casa120
\$bisc8

Inválidas
\$100vergonha
\$5
\$20assustar
\$60nacadeira

Maiúsculas e minúsculas

É recomendável que se utilize sempre identificadores com letras minúsculas, pois o PHP faz a distinção entre maiúsculas e minúsculas. Se houver uma mistura dos dois tipos de letras, pode ocorrer uma confusão na utilização da variável, já que `$nota_aluno` não é a mesma coisa que `$Nota_aluno`. Imagine que um aluno tirou nota 10 e que esse valor foi armazenado na variável `$nota_aluno`. Na hora de imprimir a nota do aluno, foi utilizada a seguinte linha:

```
<?php
    echo $Nota_aluno;
?>
```

Pobre coitado! Está reprovado porque você imprimiu a variável errada! Por isso, tome cuidado: o PHP distingue letras maiúsculas e minúsculas.

No PHP, existem variáveis dos tipos numéricos, alfanuméricas (strings), arrays (vetores) e objetos.

Variáveis numéricas

As variáveis numéricas podem possuir valores inteiros ou reais (ponto flutuante). Uma variável é definida como numérica no momento em que atribuímos um valor numérico a ela. Exemplos:

```
<?php
    $numero = 15;
    $x = 7;
    $numero_hexa = 0x0b; // valor em hexadecimal
    $y = 18.005;
    $a = 400.7;
?>
```

Variáveis alfanuméricas (strings)

São cadeias de caracteres que podem ser delimitadas por aspas simples (') ou aspas duplas ("). Quando utilizamos aspas duplas, pode-se incluir caracteres de controle no meio da string. Exemplos:

```
<?php
    $nome = 'Fernando';
    $profissao = "Professor";
    $texto = "Bom dia! <br/>Seja bem-vindo(a) ao meu site!";
?>
```

OBS: os arrays (vetores) e os objetos serão apresentados posteriormente.

Vale lembrar que a tipagem de dados no PHP é fraca. Desta forma, uma mesma variável pode receber um valor de qualquer tipo de dado, por exemplo:

```
<?php
    $minhaVariavel = 10;
```



```
echo "Conteúdo da minha variável: $minhaVariavel <br />";  
$minhaVariavel = 10.65;  
echo "Conteúdo da minha variável: $minhaVariavel <br />";  
$minhaVariavel = "Ana Maria";  
echo "Conteúdo da minha variável: $minhaVariavel";  
?>
```

Conversão de variáveis

Se tivermos uma string contendo somente números, o PHP somará normalmente esse valor com outra variável do tipo numérico. Se houver textos e números em uma string, o PHP utilizará somente a parte em que estão os números para efetuar operações aritméticas. Por exemplo:

```
$string = "5";  
$numero = 3;  
$texto = "3 vezes sem juros";
```

- Se somarmos `$numero + $string`, o resultado será 8.
- Se somarmos, por exemplo, `3.45 + $string`, o resultado será 8.45.
- Se somarmos `$numero + $texto`, o resultado será 6, pois somente a parte numérica da string será considerada.

Algumas vezes, precisamos fazer a conversão manualmente para realizar certos tipos de cálculos. Veja os conversores de tipos existente no PHP:

Conversor	Descrição
(int), (integer)	Converte para inteiro.
(real), (float), (double)	Converte para ponto flutuante.
(string)	Converte em string.
(array)	Converte em array (vetor).

O conversor converterá o tipo daquela variável que aparece imediatamente após o conversor, por exemplo:

```
<?php
    $x = 50;
    $y = 2.35;
    $soma = (int) $y + $x;
    echo $soma;
?>
```

No exemplo, a variável `$y`, que é do tipo float, é convertida para inteiro por meio do conversor `int`. Se for necessário converter o resultado de toda a expressão para inteiro, basta colocar os parênteses para delimitar a expressão cujo resultado deve ser convertido. Por exemplo:

```
$soma = (int)($x + $y);
```

2.4.4. Exercícios de fixação

Exercício 1. Um funcionário recebe um salário fixo de R\$ 1500,00 mais 4% de comissão sobre as suas vendas. Faça um programa em PHP para calcular e mostrar a comissão e o salário final do funcionário, sabendo-se que ele vendeu R\$10.000,00.

Exercício 2. Sabe-se que um(a) aluno(a) da disciplina IEB do Curso Técnico em Informática obteve nota 9.0 na primeira prova, 7.5 na segunda prova e 6.0 no projeto prático. Sabe-se ainda que o peso da primeira prova é 3, da segunda prova 4 e do projeto prático 3. Faça um programa em PHP para calcular e apresentar a nota final do aluno no bimestre.

Exercício 3. Dado que uma pessoa nasceu no ano de 1999 e o ano atual é 2017, faça um programa em PHP que calcule e mostre:

- a idade da pessoa em anos;
- a idade da pessoa em meses;
- a idade da pessoa em dias;
- a idade da pessoa em semanas.

Exercício 4. Sabe-se que uma pessoa fez um depósito de R\$ 4000,00 em sua conta poupança. Sabe-se também que a taxa de juros foi de 0.67% ao mês. Faça um programa em PHP para calcular e mostrar o valor do rendimento após um mês e o valor total depois do rendimento.

Exercício 5. Sabe-se que um garçom vai trabalhar 12 horas em uma festa de Carnaval e que o valor do salário mínimo é de R\$ 937,00 em 2017. Faça um programa em PHP para calcular o salário a receber do empregado, seguindo as seguintes regras:

- a hora trabalhada vale 5% do salário mínimo;
- o salário bruto equivale ao número de horas trabalhadas multiplicado pelo valor da hora trabalhada;
- o imposto equivale a 3% do salário bruto;
- o salário a receber equivale ao salário bruto menos o imposto.

2.4.5. Interpolação de variáveis

A interpolação de variáveis consiste em escrever o valor de uma ou mais variáveis dentro da string que será mostrada na tela ou atribuída a outra variável. Por exemplo:

```
<?php
    $curso = "Técnico em Informática Integrado ao Ensino Médio";
    $ano = "20";
    $disciplina = "IEB";
    echo "Estou estudando $disciplina no $ano do Curso $curso.";
?>
```

Após a execução deste programa, o resultado apresentado será:

```
Estou estudando IEB no 2o do Curso Técnico em Informática Integrado ao Ensino
Médio.
```

Com certeza, vamos utilizar bastante a interpolação de variáveis. Deve-se tomar cuidado, porém, quando se escrever uma variável dentro de uma string e houver um ou mais caracteres logo ao lado da variável, por exemplo:

```
<?php
    $curso = "Técnico em Informática Integrado ao Ensino Médio";
    $ano = 2;
    echo "Estou no $ano do Curso $curso.";
?>
```

O objetivo do programa seria imprimir na tela a frase “Estou no 2o do Curso Técnico em Informática Integrado ao Ensino Médio.”. Mas ao executá-lo, a será mostrada na tela o seguinte resultado:

Estou no do Curso Técnico em Informática Integrado ao Ensino Médio.

Isso ocorre porque o PHP procurará pelo valor de uma variável chamada \$ano, que na verdade não existe, e por isso nada será mostrado nesse local. Para resolver esse problema, existem duas maneiras. A primeira delas é:

```
<?php
    $curso = "Técnico em Informática Integrado ao Ensino Médio";
    $ano = 2;
    echo "Estou no ${ano}o do Curso $curso.";
?>
```

Nessa maneira, a variável é colocada entre chaves para que o PHP saiba onde termina o identificador da variável. Outra maneira é escrever o comando de forma diferente, sem usar a interpolação de variáveis e usar a concatenação:

```
<?php
    $curso = "Técnico em Informática Integrado ao Ensino Médio";
    $ano = 2;
    echo "Estou no " . $ano . "o do Curso " . $curso . ".";
?>
```

2.4.6. Operadores

Pode-se pensar em um operador como um transformador, o qual, fornecidos um ou mais valores ou expressões, transforma-os em outro valor.

No PHP existem três grupos de operadores. Os operadores unários, os quais, como o nome sugere, manipulam um único valor, por exemplo os operadores ! (negação) e ++ (incremento). Existem os operadores binários, que manipulam dois valores ou expressões e retornam um valor, por exemplo OR (ou) e >= (maior ou igual), sendo a maioria dos operadores no PHP desta categoria. Finalmente, existe o operador ternário, o qual retorna um valor entre dois possíveis, conforme o resultado de um terceiro valor ou expressão. Só existe um operador ternário no PHP.

Vamos ver os seguintes tipos de operadores:

- Operadores aritméticos;
- Operadores de comparação;
- Operadores de atribuição;
- Operadores lógicos;
- Operador ternário.

Operadores aritméticos

Nesse grupo de operadores estão as operações matemáticas básicas, tais como somar, subtrair, multiplicar e dividir. São eles:

Operador	Operação	Exemplo
+	Adição	\$valor + \$acrescimo
-	Subtração	\$valor - \$desconto
*	Multiplicação	\$valor * 10
/	Divisão	\$valor / 5
%	Resto da divisão	\$valor % 2

Nota: O operador de divisão “/” retorna sempre um ponto flutuante, mesmo que ambos os valores sejam inteiros.

O PHP possui também outros operadores aritméticos, que atuam em apenas um operador. Esses operadores são bastante úteis, pois permitem realizar de forma simples algumas operações, tais como troca de sinal, incremento ou decremento de valores.

Operador	Operação	Exemplo
-oper	Troca o sinal do operando	-\$valor
++oper	Pré-incremento. Primeiro incrementa o valor do operando e depois realiza a operação.	++\$contador
--oper	Pré-decremento. Primeiro decrementa o valor do operando e depois realiza a operação.	--\$contador
oper++	Pós-incremento. Primeiro realiza a operação e depois incrementa o valor do operando.	\$contador++
oper--	Pós-decremento. Primeiro realiza a operação e depois decrementa o valor do operando.	\$contador--

O pré-incremento (++oper) e o pós-incremento (oper++) diferem um do outro se tivermos uma atribuição de valor a uma variável ou a avaliação de uma expressão. Vamos ver um exemplo dos dois tipos:

```
<?php
```

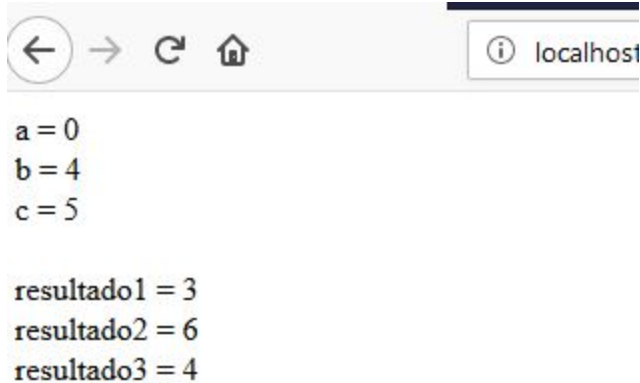
```

$a = 1;
$b = 3;
$c = 5;
$resultado1 = ++$b - $a;
$resultado2 = $c-- + $a;
$resultado3 = --$a + $c++;
echo "a = $a <br/>b = $b <br/>c = $c <br/><br/>";
echo "resultado1 = $resultado1<br/>resultado2 = $resultado2 <br/>
      resultado3 = $resultado3";

```

```
?>
```

Salve o programa e execute em seu navegador:



```
a = 0
b = 4
c = 5

resultado1 = 3
resultado2 = 6
resultado3 = 4
```

Vamos analisar os resultados. A variável `$resultado1` recebeu o valor gerado pela expressão `++$b - $a`, ou seja, a variável `$b` foi incrementada (passou de 3 para 4) e desse valor de `$b` foi subtraído do valor da variável `$a`, que vale 1. O resultado foi $4 - 1 = 3$, como mostra o resultado da figura acima.

A variável `$resultado2` recebeu o valor gerado pela expressão `$c-- + $a`, ou seja, primeiro foi feita a soma da variável `$c` com valor da variável `$a`, o resultado foi atribuído à variável `$resultado2` e depois dessa atribuição o valor da variável `$c` foi decrementado. Ao terminar a execução dessa linha, a variável `$resultado2` ficou como o valor da soma `$c + $a`, ou seja, $5 + 1 = 6$, e a variável `$c`, que possuía o valor 5, passou para o valor 4 após o decremento.

A variável `$resultado3` recebeu o valor gerado pela expressão `--$a + $c++`, ou seja, a variável `$a` foi decrementada (passou de 1 para 0) e esse valor foi somado ao valor da variável `$c`, resultando em: $0 + 4 = 4$. Esse resultado foi atribuído à variável `$resultado3` e depois dessa atribuição o valor da variável `$c` foi incrementado. Ao terminar a execução dessa linha, a variável `$resultado3` ficou como o valor 4, e a variável `$c`, que possuía o valor 4, passou para o valor 5 após o incremento.

Operadores de comparação

Os operadores de comparação, como o próprio nome diz, comparam dois valores, retornando verdadeiro ou falso.

Operador	Operação	Exemplo
==	Igualdade	\$v == \$a (retorna verdadeiro se \$v for igual a \$a)
===	Idênticos	\$v === \$a (retorna verdadeiro se \$v for igual a \$a e ambos forem do mesmo tipo)
!=	Diferente	\$v != \$a (retorna verdadeiro se \$v for diferente de \$a)
<>	Diferente	\$v <> \$a (retorna verdadeiro se \$v for diferente de \$a)
!==	Não idênticos	\$v !== \$a (retorna verdadeiro se \$v for diferente a \$a e se não forem do mesmo tipo)
<	Menor	\$v < \$a (retorna verdadeiro se \$v for menor que \$a)
<=	Menor ou igual	\$v <= \$a (retorna verdadeiro se \$v for menor ou igual a \$a)
>	Maior	\$v > \$a (retorna verdadeiro se \$v for maior que \$a)
>=	Maior ou igual	\$v >= \$a (retorna verdadeiro se \$v for maior ou igual a \$a)

Operadores de atribuição

O operador de atribuição deve ser utilizado quando deseja-se atribuir o valor da expressão que está à sua direita ao operando (geralmente uma variável) que está à sua esquerda.

O operador básico de atribuição é o sinal de igual (=). Pode-se também realizar operações combinadas, tais como `$valor = ($total = 12) - 7`. Desta forma, tem-se `$total = 12` e `$valor = 5`.

Além do formato básico, pode-se combinar o operador de atribuição com todos os operadores aritméticos e de string.

Operador	Operação	Exemplo
=	Atribuição	\$valor = \$total + 5
+=	Atribuição e adição	\$valor += 5 (\$valor = \$valor + 5)
-=	Atribuição e subtração	\$valor -= 5 (\$valor = \$valor - 5)
*=	Atribuição e multiplicação	\$valor *= 5 (\$valor = \$valor * 5)
/=	Atribuição e divisão	\$valor /= 5 (\$valor = \$valor / 5)
%=	Atribuição e módulo	\$valor %= 2 (\$valor = \$valor % 2)
.=	Atribuição e concatenação	\$texto .= \$s (\$texto = \$texto . \$s)

Operadores lógicos

Os operadores lógicos realizam comparações entre expressões (com exceção do operador “!” que é unário, todos os outros operadores são binários), retornando verdadeiro ou falso como resultado.

Operador	Operação	Exemplo
AND	E	op1 AND op2 (verdadeiro se op1 E op2 forem verdadeiros)
OR	Ou	op1 OR op2 (verdadeiro se op1 OU op2 forem verdadeiros)
XOR	Ou exclusivo	op1 XOR op2 (verdadeiro só se op1 ou só op2 for verdadeiro)
!	Negação	!op1 (verdadeiro se op1 for falso)
&&	E	op1 AND op2 (verdadeiro se op1 E op2 forem verdadeiros)
 	Ou	op1 OR op2 (verdadeiro se op1 OU op2 forem verdadeiros)

A diferença entre os operadores AND e &&, e também entre OR e || é a precedência dos operadores na avaliação de expressões. Os operadores && e || têm precedência mais alta.

Operador ternário

É uma forma abreviada de usar a estrutura condicional if. Uma condição é avaliada, se for verdadeira atribui-se um valor à variável, e se for falsa atribui-se outro valor. A sintaxe é:

```
condicao ? expressao1 : expressao2;
```

Por exemplo:

```
$salarioNovo = ($salario <= 1500) ? ($salario + 500) : ($salario+200);
```

Precedência de operadores

Para evitar erros de lógica em seus programas é fundamental conhecer a ordem de precedência de operadores utilizada pelo PHP. A tabela abaixo apresenta a ordem decrescente (do mais alta para a mais baixa) de precedência de operadores do PHP:

Operador	Descrição
- ! ++ --	Negativo, negação, incremento e decremento.
* / %	Multiplicação, divisão e resto da divisão.
+ - .	Adição, subtração e concatenação.
> < >= <=	Maior, menor, maior ou igual, menor ou igual.
== != <>	Igualdade, desigualdade.
&&	AND (E lógico).
	OR (Ou lógico).
?:	Operador ternário.
= += -= *= /= % =	Operadores de atribuição.
AND	E lógico de menor prioridade.
XOR	Ou exclusivo.
OR	Ou lógico de menor prioridade.

É importante lembrar que primeiro o PHP executará todas as operações que estiverem entre parênteses. Se dentro dos parênteses houver diversas operações, a precedência de operadores será utilizada para definir a ordem. Depois de resolver todas as operações que aparecem entre parênteses, o PHP resolverá o resto da expressão com base na tabela de precedência de operadores para determinar a ordem de avaliação dos operadores.

Quando houver operadores de mesma prioridade em uma expressão e não existirem parênteses, o PHP resolverá a expressão da esquerda para a direita.

```
<?php
    $numero = 5;
    $resultado = 8 + 3 * 2 + ++$numero;
    echo "Número = $numero <br/>Resultado = $resultado";
?>
```

O resultado mostrado na página será 6 e 20.

2.2.7. Constantes

Constantes são valores predefinidos no início do programa PHP e que não mudam ao longo de sua execução. Pode-se definir suas próprias constantes utilizando o comando `define`. Veja um exemplo de como deve-se usar as constantes:

```
<?php
    define ("nome", "Fernando");
    define ("peso", 64);
    echo "Meu nome é " . nome . "<br/>";
    echo "Meu peso é " . peso . " quilos.";
?>
```

É importante notar que as constantes são referidas diretamente pelo nome escolhido para elas e não utiliza-se, na frente delas, o símbolo `$`, pois esse símbolo é utilizado apenas para representar variáveis.

Outro recurso utilizado, no exemplo, foi a concatenação por meio do ponto (.). pode-se concatenar quantos dados quisermos e todos serão exibidos como apenas uma string.

O PHP possui também diversas constantes próprias predefinidas. Algumas delas são:

Constante	Descrição
TRUE	Valor verdadeiro (utilizado para comparação).
FALSE	Valor falso.
__FILE__	Contém o nome do script que está sendo executado.
__LINE__	Contém o número da linha do script que está sendo executado.
PHP_VERSION	Contém a versão corrente do PHP.
PHP_OS	Contém o nome do sistema operacional no qual o PHP está executando.

2.2.8. Exercícios de Fixação

Exercício 1. Dado que um espetáculo teatral tem o custo de R\$ 2665,00 e o preço do convite esse espetáculo é R\$ 65,00. Faça um programa em PHP para calcular e mostrar:

- A quantidade de convites que devem ser vendidos para que pelo menos o custo do espetáculo seja alcançado.
- A quantidade de convites que devem ser vendidos para que se tenha um lucro de 23%.

Exercício 2. Faça um programa em PHP para efetuar o cálculo da quantidade de combustível gasto em uma viagem, sabendo-se que o automóvel que faz 12Km por litro, que o tempo gasto foi de 2 horas e meia e a velocidade média durante a viagem foi 96 Km/hora.

Exercício 3. Sabe-se que o quilowatt de energia custa R\$ 0,48 e que uma residência consumiu 137 quilowatts no mês de fevereiro de 2017. Faça um programa em PHP que calcule e mostre:

- O valor, em reais, a ser pago por essa residência.
- O valor, em reais, a ser pago com desconto de 15%.

Exercício 4. Um hotel deseja fazer uma promoção especial de final de semana, concedendo um desconto de 25% na diária. O hotel possui 42 apartamentos e o valor normal da diária por apartamento é R\$ 167,00. Implemente um programa em PHP para calcular:

- Valor promocional da diária;
- Valor total a ser arrecadado caso a ocupação neste final de semana (2 diárias por apartamento) atinja 100%;
- Valor total a ser arrecadado caso a ocupação neste final de semana atinja 70%;
- Valor que o hotel deixará de arrecadar em virtude da promoção, caso a ocupação atinja 100%.

Exercício 5. Faça um programa em PHP que calcule o índice de massa corpórea de uma pessoa com 67 kg de peso e a 172 cm de altura. O índice de massa corpórea mede a relação entre peso e altura (peso em Kg, dividido pelo quadrado da altura em metros).

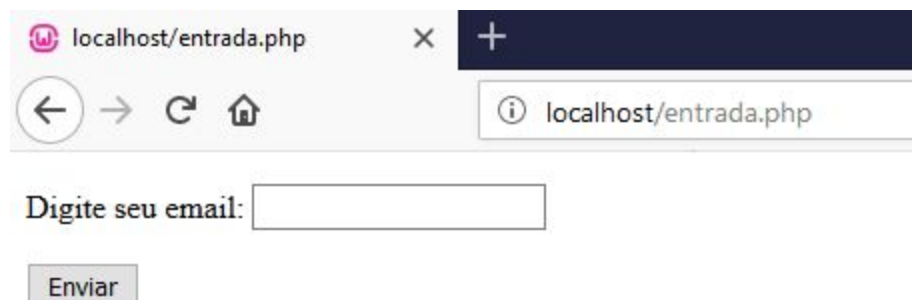
Capítulo 3 - Entrada de dados

Neste capítulo, vamos revisar a criação de formulários HTML, para que os usuários possam preencher com seus dados. E também aprender como enviar esses dados ao programa PHP, para que sejam tratados.

3.1 Formulários em HTML

Em HTML, um formulário possui, no mínimo, um campo para entrada de dados e um botão para o envio das informações desse campo.

```
<form>
  <p>Digite seu email: <input type="text" name="email" size="20" /></p>
  <p><input type="submit" value="Enviar" /></p>
</form>
```



No exemplo mostrado acima, ao clicar no botão Enviar, o valor preenchido no campo email não será enviado a lugar nenhum. Para tornar o formulário útil, devemos informar ao navegador para onde devem ser mandadas as informações. Isso é feito por meio da opção action da tag form.

```
<form action="processa_dados.php">
  <p>Digite seu email: <input type="text" name="email" size="20" /></p>
  <p><input type="submit" value="Enviar" /></p>
</form>
```

Agora , ao clicar no botão Enviar, o conteúdo do campo email será enviado ao programa `processa_dados.php` e esse programa irá tratar a informação recebida.

Há diferentes tags presentes na linguagem HTML para a entrada de dados, dependendo do que se deseja informar.

3.1.1. Tag input

Uma das tags que é utilizada para criar campos de entrada de dados em HTML é a `input`. Esta tag pode conter várias opções, das quais destacam-se:

- **name:** informa o nome do campo;
- **value:** informa um valor padrão para o campo;
- **size:** informa qual o tamanho do campo que será exibido na tela;
- **maxlength:** informa o número máximo de caracteres que pode ser digitado no campo; e
- **type:** informa qual é o tipo do campo de entrada de dados.

Os valores que podem ser utilizados na opção **type** permitem definir diferentes formas de entrada de dados. Os mais frequentes são:

- **radio:** permite criar botões de seleção para escolha de uma entre várias opções disponíveis.
- **text:** exibe uma caixa de texto de uma linha, a qual pode receber tanto valores numéricos quanto alfanuméricos.
- **checkbox:** mostra uma caixa de seleção que pode ou não ser marcada.
- **email:** mostra um campo de texto de uma linha para a digitação de endereço de e-mail. Alguns navegadores processam o texto do campo, verificando se este contém um e-mail em um formato válido.
- **tel:** mostra um campo de uma linha para inserção de número de telefone. Por meio da opção `pattern` é possível definir uma expressão regular para verificar o formato do número telefônico.
- **submit:** é utilizado para criar um botão que aciona o envio dos dados dos formulários.
- **reset:** cria um botão que, quando acionado, limpa todos os campos de um formulário, retornando-os ao seus valores padrão.

```
<form action="processa_dados.php">
  <p><b>O que você achou do site?</b></p>
  <p>
    <input type="radio" name="avaliacao" value="muitobom" checked="checked" />
    Muito bom<br />
    <input type="radio" name="avaliacao" value="bom" />Bom<br />
    <input type="radio" name="avaliacao" value="regular" />Regular<br />
    <input type="radio" name="avaliacao" value="umlixo" />Um lixo<br />
  </p>
</form>
```

```

</p>
<p><b>Diga-nos como entrar em contato com você:</b></p>
<p>Nome: <input type="text" name="nome" size="35" maxlength="256" /></p>
<p>E-mail: <input type="email" name="email" size="35" maxlength="256" /></p>
<p>Fone: <input type="tel" name="fone" size="35" maxlength="256" /></p>
<p><input type="checkbox" name="novidades" value="novidades" />
    Quero receber as novidades do site por email.
</p>
<p>
    <input type="submit" value="Enviar" />
    <input type="reset" value="Limpar" />
</p>
</form>

```

- **password:** é utilizado especificamente para a digitação de senhas. No lugar dos caracteres digitados são exibidos asteriscos, contudo a informação é enviada normalmente.
-

```

<form action="login.php">
  <fieldset>
    <legend>Digite seus dados:</legend>
    <p>
      <label>Usuário:</label>
      <input type="text" name="usuario" size="35" maxlength="256" />
    </p>
    <p>
      <label>Senha:</label>
      <input type="password" name="senha" size="35" maxlength="256" />
    </p>
    <p>
      <input type="submit" value="Entrar" />
    </p>
  </fieldset>
</form>

```


-
- **hidden:** este campo permanece escondido, não aparecendo na tela. Pode ser utilizado para passar algumas informações aos programas que recebem os dados.
-

```
<form action="processa_dados.php">
  <input type="hidden" name="operacao" value="inclusao">
  <p>
    Código do produto: <input type="number" name="codigo_produto" size="5" />
    <br />
    Nome do produto: <input type="text" name="nome_produto" size="20" /><br />
    Descrição do produto: <input type="text" name="descricao_produto"
      size="20" />
  </p>
  <p><input type="submit" value="Incluir" name="incluir" /></p>
</form>
```

- **file:** permite o envio de arquivos.
-

```
<form action="processa_dados.php">
  <p>
    Envie seu documento:
    <input type="file" name="arquivo" accept=".doc,.docx" />
  </p>
  <p>
    <input type="submit" value="Enviar" />
  </p>
</form>
```

-
- **image**: funciona de forma similar ao submit, mas utiliza uma imagem ao invés do botão tradicional do formulário.
-

```
<form action="login.php">
  <fieldset>
    <legend>Digite seus dados:</legend>
    <p>
      <label>Usuário:</label>
      <input type="text" name="usuario" size="35" maxlength="256" />
    </p>
    <p>
      <label>Senha:</label>
      <input type="password" name="senha" size="35" maxlength="256" />
    </p>
    <p>
      <input type="image" src="enter.png" alt="Entrar" width="48"
        height="48" />
    </p>
  </fieldset>
</form>
```

3.1.2. Tag select

Esta tag do HTML mostra uma lista de seleção no estilo *drop-down*.

```
<form action="processa_dados.php">
  <p><b>Qual a seção que você mais gostou?</b></p>
  <p>
    <select name="secao" size="1">
      <option value="emcartaz">Em cartaz</option>
      <option value="trilhasonora">Trilha sonora</option>
      <option value="fotos">Fotos</option>
      <option value="bilheteria">Bilheteria</option>
      <option value="outra">Outra</option>
    </select>
  <p>
    <input type="submit" value="Enviar" />
  </p>
</form>
```

3.1.3. Tag textarea

A tag textarea exibe uma caixa de texto contendo várias linhas e é geralmente utilizada para a entrada de textos maiores do que aqueles da opção text da tag input.

```
<form action="processa_dados.php">
  <p><b>Digite seus comentários no espaço abaixo:</b></p>
  <p>
    <textarea name="comentarios" rows="5" cols="42"></textarea>
  </p>
  <p>
    <input type="submit" value="Enviar" />
  </p>
</form>
```

3.2 Envio de informações para um programa PHP

Para que os dados fornecidos nos campos dos formulário possam ser devidamente processados, é necessário especificar qual programa PHP irá recebê-los. Isto é feito por meio da opção action da tag form. À opção action, atribui-se o nome do programa PHP que receberá os dados, incluindo a extensão do arquivo (.php).

Além da opção action, é preciso também especificar o método que será utilizado para o envio dos dados. Isto é feito por meio da opção method da tag form. Os possíveis valores para essa opção são GET ou POST, os quais indicam ao navegador formas diferentes de transmissão dos dados para o servidor que executará o programa em PHP.

3.2.1. Método GET

O método GET é o padrão para envio de dados. Isto significa que, se nada for especificado na opção method da tag form, o método GET será escolhido automaticamente pelo navegador. Neste método, os dados dos campos do formulário são enviados juntamente com o nome da página (URL), fazendo com que o conteúdo dos campos fique visível na URL presente na barra de endereços do navegador.

Por exemplo, suponha que, no formulário abaixo, preenchamos o campo nome com o valor Joaquim e o campo idade com o valor 20.

```
<form action="processa_dados.php">
  <p>
    <label>Digite seu nome:</label>
    <input type="text" name="nome" size="30" />
  </p>
  <p>
    <label>Digite sua idade:</label>
    <input type="number" name="idade" size="3" />
  </p>
  <p>
    <input type="submit" value="Enviar" />
  </p>
</form>
```

O endereço ativado será:

http://www.seusite.com.br/processa_dados.php?nome=Joaquim&idade=20

Neste exemplo, os campos do formulário são passados como parâmetros no endereço de destino. O caractere ? representa o início de uma cadeia de variáveis. O caractere & identifica o início de uma nova variável. As variáveis e seu valores são separados pelo caractere =.

A vantagem do uso deste método é a possibilidade de passar parâmetros por meio de links. Contudo, o número de caracteres em um endereço é limitada (em torno de 2000), o que limita a quantidade de informações que pode ser enviada. Além disso, todos os parâmetros ficam visíveis na barra de endereço do navegador, o que nem sempre é desejável.

3.2.2. Método POST

O método POST envia os dados presentes no formulário por meio do corpo da mensagem encaminhada ao servidor. Desta forma, o usuário do navegador não consegue visualizar os dados transmitidos, pois o endereço de destino não irá conter a cadeia de variáveis.

```
<form action="processa_dados.php" method="post">
  <p>
    <label>Digite seu nome:</label>
    <input type="text" name="nome" size="30" />
  </p>
  <p>
    <label>Digite sua idade:</label>
    <input type="number" name="idade" size="3" />
  </p>
  <p>
    <input type="submit" value="Enviar" />
  </p>
</form>
```

No exemplo anterior, ao utilizar o método POST, o endereço de destino seria:

`http://www.seusite.com.br/recebe_dados.php`

O método POST tem a vantagem de não possuir uma limitação no tamanho dos dados que estão sendo enviados. Portanto, é recomendado utilizá-lo nos casos em que o formulário possui muitos dados, ou ainda, quando é necessário enviar dados que não podem ser enviados pelo método GET, como imagens e arquivos.

3.3. Recebimento dos dados

Os dados do formulário enviados pelo navegador ao servidor são disponibilizados ao programa PHP por meio de arrays superglobais. Dependendo do método escolhido para envio (GET ou POST) um array diferente deve ser acessado. Dados enviados pelo método GET são armazenados no array `$_GET` e dados enviados pelo método POST são armazenados no array `$_POST`.

Assim, o programa em PHP deve acessar o array apropriado, fornecendo o nome dos campos do formulário como chave associativa. Isto significa que, ao invés de acessar os elementos do array pela forma tradicional, utilizando índices numéricos, strings serão utilizadas como índices.

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

    echo "<h2>Nome recebido: $nome</h2>";
    echo "<h3>Idade recebida: $idade</h3>"
?>
```

3.4 Exercícios de Fixação

Exercício 1. Em épocas de pouco dinheiro, os comerciantes estão procurando aumentar suas vendas oferecendo desconto. Faça um programa em PHP que possa receber de uma página web (HTML) o valor de um produto e que retorne uma outra página web com o novo valor tendo em vista que o desconto foi de 9%.

Exercício 2. Faça o programa em PHP que calcule o valor em reais, correspondente aos dólares que um turista possui no cofre do hotel. O programa deve receber os seguintes dados de uma página web:

- Quantidade de dólares guardados no cofre e;
- Cotação do dólar naquele dia.

Exercício 3. Uma loja de informática está vendendo seus produtos em 5 (cinco) prestações sem juros. Faça um programa em PHP que receba, por meio de uma página web, o valor de uma compra e retorne uma outra página web com o valor das prestações.

Exercício 4. Faça um programa em PHP que receba de uma página web o preço de custo de um produto e a margem de lucro do produto (percentual informado pelo usuário). O programa deve retornar uma página web com o valor de venda do produto.

Exercício 5. Cada degrau de uma escada tem uma altura X. Faça um programa em PHP que receba essa altura e a altura que o usuário deseja alcançar subindo a escada. Calcule e mostre quantos degraus o usuário deverá subir para atingir seu objetivo. Todas as medidas fornecidas devem estar em metros.

Exercício 6. Sabe-se que, para iluminar de maneira correta os cômodos de uma casa, para cada m², deve-se usar 18 W de potência. Faça um programa em PHP que receba as duas dimensões do cômodo (em metros), calcule e retorne a área do cômodo em m² e a potência de iluminação que deverá ser utilizada.

Capítulo 4 - Estruturas de Controle

Neste capítulo, são apresentadas as estruturas de controle que são utilizadas para realizar decisões lógicas, testar se determinada expressão é verdadeira, repetir um bloco de comandos por determinado número de vezes ou até para que uma condição seja atingida.

4.1. Estruturas de Controle Condicional

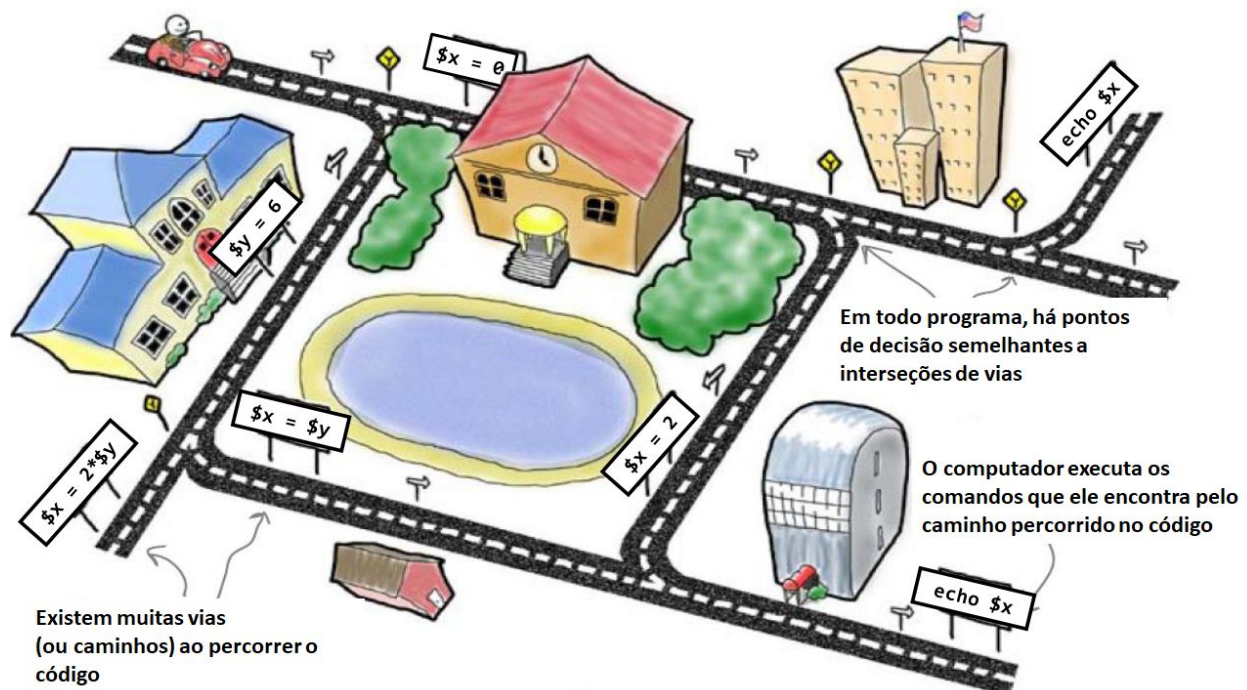
Até este ponto, vimos programas em PHP que são apenas sequências simples de comandos. Tais programas possuem um único fluxo de execução, assim como um automóvel que trafega em uma via sem desvios ou retornos.

```
<?php
echo "Bem vindo ao meu programa!";
echo "Abastecendo...";
echo "Volte sempre!";
?>
```



Em geral, programas computacionais envolvem diversas estruturas que permitem acionar fluxos de execução diferentes, caso certas condições sejam ou não atendidas. De forma análoga, é como se um automóvel estivesse agora trafegando pelas ruas de uma cidade e, dependendo das condições e necessidades, diferentes caminhos pudessem ser tomados.

Os pontos de decisão de um programa são semelhantes a interseções de vias, que podem levar a caminhos diferentes. Assim, um mesmo programa pode ter diversos caminhos possíveis, mas somente são percorridos aqueles que atendem às condições verificadas nos pontos de decisão.



Para permitir o acionamento de diferentes caminhos ou fluxos de execução nos programas em PHP, serão vistas, inicialmente, as estruturas de controle condicional, representadas pelos comandos `if` e `switch`.

4.1.1. Comando `if`

Este comando é formado pela palavra reservada `if`, que em português significa “se”, seguida de uma expressão entre parênteses. Quando o comando é executado, a expressão é avaliada e, caso ela seja verdadeira (*true*), o bloco de comandos que estiver entre chaves será executado. Se a expressão entre parênteses for falsa (*false*), o bloco de comandos dentro das chaves não será executado.

```
if (expressão) {
    //instruções...
}
```

O exemplo abaixo mostra um programa em PHP que verifica se um determinado número é par. Nele é utilizado um comando `if` simples.

```
<?php
    $num = 4;
    if($num % 2 == 0){
        echo "Número é par!";
    }
?>
```

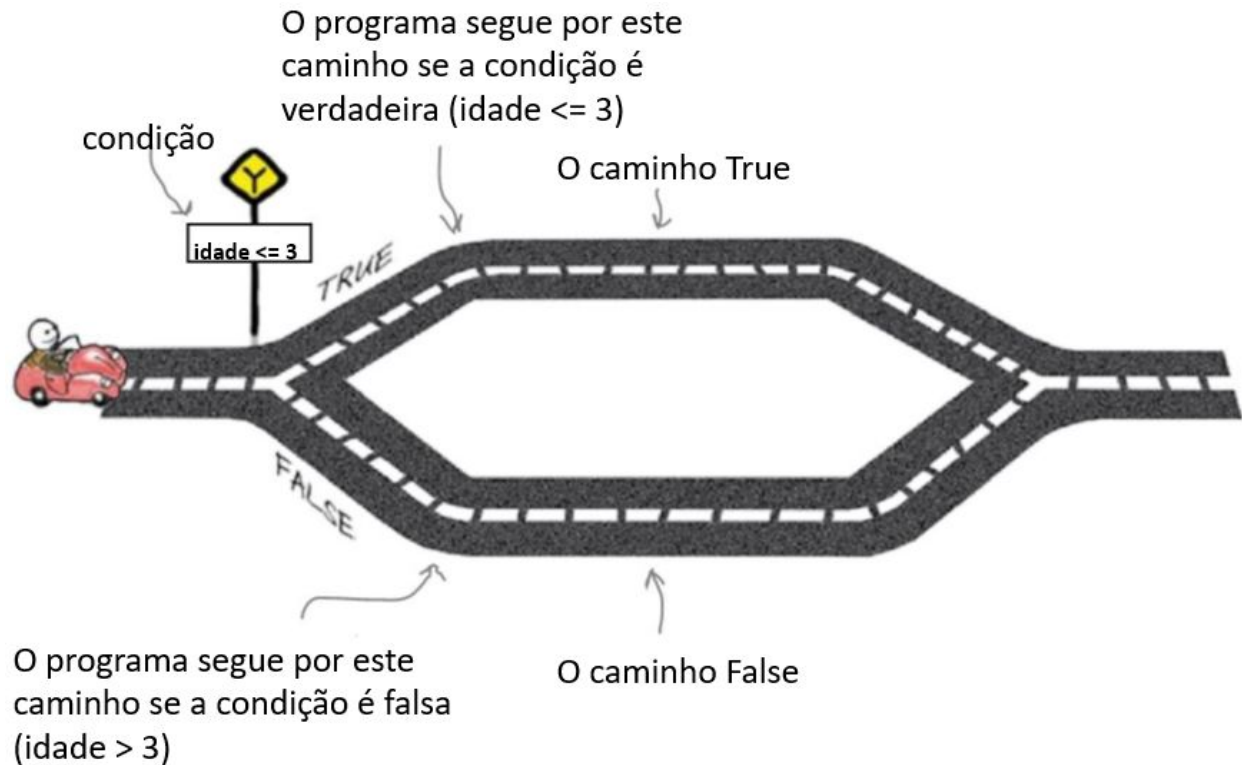
O comando if pode ser utilizado também de forma composta, com o comando else (em português, “senão”) na sequência para especificar o bloco de comandos que será executado caso a expressão seja avaliada como falsa.

```
if (expressão) {
    //instruções...
} else {
    //instruções...
}
```

Se o bloco de comandos possuir somente um comando, o uso das chaves torna-se opcional. Considera-se, contudo, uma boa prática de programação o uso das chaves mesmo nesta situação. No exemplo abaixo, o comando if testa se um determinado número é par. Caso o teste seja avaliado como negativo, o bloco de comandos dentro do else é executado.

```
<?php
    $num = 7;
    if($num % 2 == 0){
        echo "Número é par!";
    }else{
        echo "Número é ímpar!";
    }
?>
```

Analogamente, um programa executar um comando condicional é como um automóvel que trafega por uma via e se depara com uma bifurcação. Neste momento, baseado na avaliação feita pelo condutor (no caso do programa, uma condição), toma-se uma das direções, conforme a ilustração abaixo.



Continuando a analogia, é possível ainda que, tomada umas das direções, mais a frente se encontre outra bifurcação. Neste caso, deve-se fazer uma nova avaliação para determinar o caminho a seguir.

Considere o código abaixo. Caso o valor da variável `$salario` seja maior que 1200, a execução do programa segue para o bloco de comandos dentro do `else`. Ali há outro comando condicional que avalia novamente a variável `$salario`. Em situações como esta, onde há estruturas condicionais dentro de estruturas condicionais, dizemos que há estruturas condicionais aninhadas.

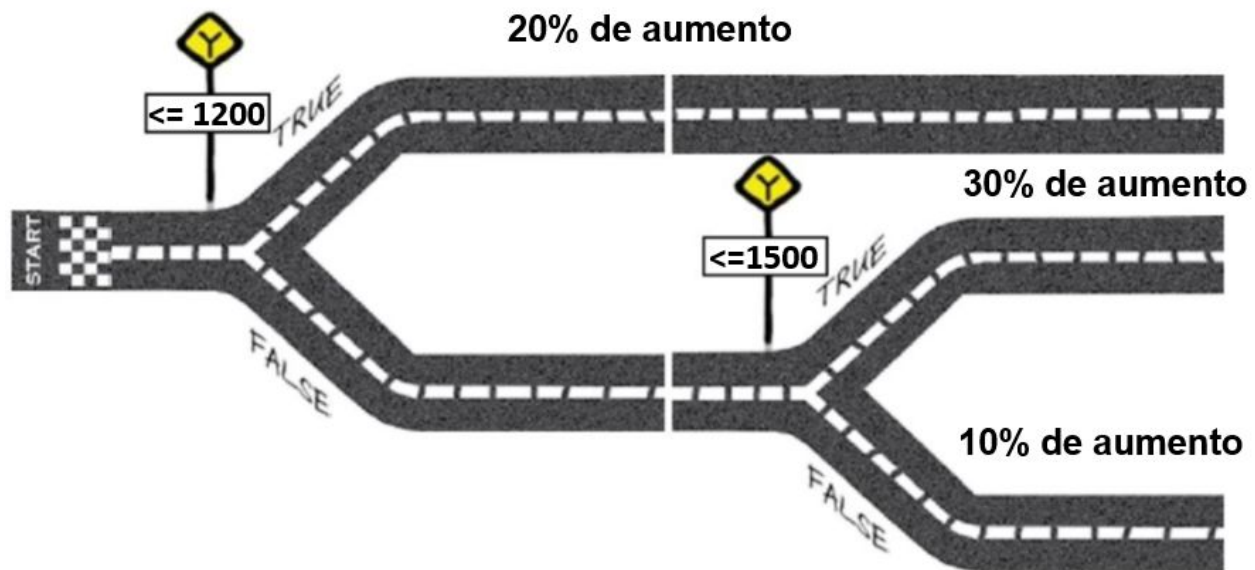
```
<?php
    $salario = $_POST["salario"];

    if ($salario <= 1200){
        $aumento = $salario * 0.2;
    } else {
        if ($salario <= 1500) {
            $aumento = $salario * 0.3;
        } else {
            $aumento = $salario * 0.1;
        }
    }

    $novoSalario = $salario + $aumento;
```

```
echo "Valor do Aumento: R$ " . $aumento;  
echo "<br/><br/>Novo Salario: R$" . $novoSalario;  
?>
```

A figura abaixo ilustra esta situação, utilizando a analogia com as vias de trânsito. Caso a primeira condição avaliada seja falsa, ainda deve-se avaliar outras condições.



Para resolver exemplos como esse, pode-se utilizar o comando `elseif`. Este comando é apropriado para os casos em que há múltiplas avaliações de condições e onde cada condição leva a um fluxo de execução diferente. Reescrevendo o exemplo, tem-se o seguinte código:

```
<?php  
$salario = $_POST["salario"];  
  
if ($salario <= 1200){  
    $aumento = $salario * 0.2;  
} elseif ($salario <= 1500) {  
    $aumento = $salario * 0.3;  
} else {  
    $aumento = $salario * 0.1;  
}  
  
$novoSalario = $salario + $aumento;  
echo "Valor do Aumento: R$ " . $aumento;  
echo "<br/><br/>Novo Salario: R$" . $novoSalario;  
?>
```

O comando `else` ao final permite contemplar todas as demais possibilidades que as condições anteriores não contemplaram.

4.1.2. Comando `switch`

O comando `switch` é parecido com o comando `if`, uma vez que ambos avaliam o valor de um argumento teste para escolher qual bloco de instruções deve ser executado. O argumento pode ser numérico, um caractere ou uma `String`.

O valor do argumento teste é avaliado e se for igual ao valor de uma das constantes, a execução do código é desviada para aquele ponto.

```
switch(expressão) {  
    case constante1:  
        //instruções...  
        break;  
    case constante2:  
        //instruções...  
        break;  
    case constante3:  
        //instruções...  
        break;  
    default:  
        //instruções...  
}
```

Enquanto o `if` utiliza várias cláusulas (`if`, `else`, `elseif`), a estrutura `switch..case` utiliza somente uma cláusula (`case`), tornando o código um pouco mais organizado. O bloco `default` é opcional e executado se nenhuma combinação for encontrada.

O exemplo abaixo mostra um programa em PHP que verifica se um determinado número é igual a 0, 1, 2, 3 ou 4. Se o número fornecido for diferente destes, o código exibe a mensagem “é cinco ou mais”.

```

<?php
    $i = $_POST["numero"];
    switch ($i){
        case 0:
            echo "$i é zero";
            break;
        case 1:
            echo "$i é um";
            break;
        case 2:
            echo "$i é dois";
            break;
        case 3:
            echo "$i é três";
            break;
        case 4:
            echo "$i é quatro";
            break;
        default:
            echo "$i é cinco ou mais";
    }
?>

```

Note que após cada bloco de instruções deve ser utilizado o comando break, para que o comando switch seja encerrado e a execução do restante do código continue após ele.

```

<?php
    $mes = date("F", time());

    switch ($mes){
        case "January":
            echo "Janeiro";
            break;
        case "February":
            echo "Fevereiro";
            break;
        case "March":
            echo "Março";
            break;
        case "April":
            echo "Abril";
            break;
        default:
            echo "Outro mês";
    }
?>

```

No exemplo acima, o valor do argumento teste é o conteúdo da variável `$mes` (meses do ano em inglês). As saídas possíveis são “janeiro”, “fevereiro” ou “março”, ou ainda “outro mês”, se o valor fornecido for diferente destes.

4.1.1 Exercícios de Fixação

Exercício 1. Faça uma página HTML para enviar três notas de um aluno para um programa PHP, que recupera as notas, calcula e retorna uma nova página HTML com a média aritmética das notas do aluno e a mensagem constante na tabela a seguir:

Média aritmética	Mensagem
Média < 4,0	Reprovado
4,0 >= Média < 6,0	Reavaliação
Média >= 6,0	Aprovado

Exercício 2. Faça uma página HTML para enviar três números inteiros para um programa PHP, que recupera os números e retorna-os uma nova página HTML em ordem crescente. Suponha que o usuário sempre digitará três números diferentes.

Exercício 3. Elabore um programa PHP que calcule o valor a ser pago por uma compra, considerando o preço normal de etiqueta e a escolha da condição de pagamento (enviados por meio de um formulário HTML), e de acordo com os seguintes critérios:

Código	Condição de pagamento
1	À vista em dinheiro ou cheque, recebe 10% de desconto.
2	À vista no cartão de crédito, recebe 5% de desconto.
3	Em 2 vezes, preço normal da etiqueta sem juros.
4	Em 3 vezes, preço normal da etiqueta mais juros de 10%.

Exercício 4. Elaborar um programa em PHP que recebe de uma página web 3 valores inteiros e positivos A, B, C e verifica se eles formam ou não um triângulo. Caso os valores formem um triângulo, deve-se calcular e imprimir o valor do perímetro do triângulo, indicando com uma mensagem se este é equilátero (três lados iguais), isósceles (apenas 2 lados iguais) ou

escaleno (todos os lados são diferentes). Se os valores não formam um triângulo, escrever uma mensagem informando o fato. Lembre-se que em um triângulo, o comprimento de cada lado deve ser menor que a soma dos outros dois lados.

Exercício 5. Escreva um programa em PHP que recebe de uma página HTML as coordenadas (X,Y) de um ponto no sistema cartesiano e escrever o quadrante ao qual o ponto pertence. Caso o ponto não pertença a nenhum quadrante, escrever se ele está sobre o eixo X, eixo Y ou na origem.

Exercício 6. Uma empresa concederá um aumento de salário aos seus funcionários, variável de acordo com o cargo, conforme a tabela abaixo. Faça um programa em PHP que recebe de uma página web o salário e o cargo de um funcionário, e calcule o novo salário. Se o cargo do funcionário não estiver na tabela, ele deverá, então, receber 40% de aumento. Mostre o salário antigo, o novo salário e a diferença entre o salário antigo e o novo salário.

Cargo	Percentual
Gerente	10%
Engenheiro	20%
Técnico	30%

Exercício 7. O IMC (Índice de Massa Corporal) é um critério da Organização Mundial de Saúde para dar uma indicação sobre a condição de peso de uma pessoa adulta. A fórmula é $IMC = \text{peso} / (\text{altura})^2$. Elabore um programa em PHP que receba o peso e a altura de um adulto e mostre sua condição de acordo com a tabela abaixo.

IMC em adultos	Condição
Abaixo de 18,5	Abaixo do peso
Entre 18,5 e 25	Peso normal
Entre 25 e 30	Acima do peso
Acima de 30	Obeso

Exercício 8. Fazer um programa na Linguagem PHP para receber a idade de uma pessoa e informar sua classe eleitoral, que pode ser:

- Não eleitor (abaixo de 16 anos);
- Eleitor obrigatório (entre 18 e 65 anos);
- Eleitor facultativo (entre 16 e 18 anos e maior de 65).

Exercício 9. Faça uma página HTML para enviar número inteiro entre 1 e 7 para um programa PHP, que deverá escrever o dia da semana correspondente. Caso o usuário digite um número fora desse intervalo, deverá aparecer uma mensagem informando que não existe dia da semana com esse número.

Exercício 10. Criar um programa em PHP que informe a quantidade total de calorias de uma refeição a partir do usuário, que deverá informar o prato, a sobremesa e a bebida por meio de uma página HTML. Veja a tabela de calorias a seguir:

Prato	Calorias	Sobremesa	Calorias	Bebida	Calorias
Vegetariano	180 cal	Abacaxi	75 cal	Chá	20 cal
Peixe	230 cal	Sorvete <i>diet</i>	110 cal	Suco de laranja	70 cal
Frango	250 cal	Mouse <i>diet</i>	170 cal	Suco de melão	100 cal
Carne	350 cal	Mouse chocolate	200 cal	Refrigerante <i>diet</i>	65 cal

Exercício 11. O governo federal abriu uma linha de crédito para os servidores federais. O valor máximo da prestação não poderá ultrapassar 30% do salário bruto. Fazer um programa em PHP que receba, por meio de uma página HTML, o salário bruto e o valor da prestação, e informar se o empréstimo pode ou não ser concedido.

Exercício 12 (Desafio). Criar uma página HTML que leia o número correspondente ao mês atual e os dígitos (somente os quatro números) de uma placa de veículo, e através do número finalizador da placa (algarismo da casa das unidades) determine se o IPVA do veículo vence no mês corrente, de acordo com a tabela abaixo.

Final 1 – mês (1) – Janeiro	Final 6 – mês (6) – Junho
Final 2 – mês (2) – Fevereiro	Final 7 – mês (7) – Julho

Final 3 – mês (3) – Março	Final 8 – mês (8) – Agosto
Final 4 – mês (4) – Abril	Final 9 – mês (9) – Setembro
Final 5 – mês (5) – Maio	Final 0 – mês (10) – Outubro

Exercício 13 (Desafio). Criar um programa em PHP que, a partir da idade e peso do paciente recebido por meio de uma página HTML, calcule a dosagem de determinado medicamento e imprima a receita informando quantas gotas do medicamento o paciente deve tomar por dose. Considere que o medicamento em questão possui 500 mg por ml, e que cada ml corresponde a 20 gotas.

- Adultos ou adolescentes desde 12 anos, inclusive, se tiverem peso igual ou acima de 60 quilos devem tomar 1000 mg; com peso abaixo de 60 quilos devem tomar 875 mg.
- Para crianças e adolescentes abaixo de 12 anos a dosagem é calculada pelo peso corpóreo conforme a tabela a seguir:

Peso	Dosagem
5 kg a 9 kg	125 mg
9.1 kg a 16 kg	250 mg
16.1 kg a 24 kg	375 mg
24.1 kg a 30 kg	500 mg
Acima de 30 kg	750 mg

4.2 Estruturas de Controle de Repetição

Os comandos de repetição são utilizados para repetir a execução um conjunto de instruções por um número determinado de vezes ou até que uma condição seja atingida. Em termos práticos, isto significa repetir comandos até que uma variável atinja determinado valor ou que seja diferente de um valor.

A linguagem PHP define vários comandos de repetição. Veremos os comandos for, while e do...while. O comando foreach é utilizado para percorrer arrays e será estudado mais adiante.

4.2.1 Comando FOR

O comando for é utilizado para repetir a execução de um conjunto de instruções por um número determinado de vezes. Existem três parâmetros para o comando for: inicialização, condição de parada e operação de incremento ou decremento.

```
for (inicialização; condição_de_parada; in(de)cremento) {  
    // instruções  
}
```

O parâmetro de inicialização define o valor inicial da variável que controlará a repetição das instruções. Por exemplo, o parâmetro de inicialização `$contador = 0` define o valor inicial da variável `$contador` que será utilizado para controlar a repetição. O comando da inicialização é executado uma única vez, ao iniciar o laço for. Em seguida, a condição de parada é avaliada. O parâmetro `condição_de_parada` define a condição para que a repetição continue executando.

Por exemplo, o parâmetro de condição de parada `$contador < 10` define que o conjunto de instruções deverá ser repetido enquanto esta condição seja verdadeira. Quando esta condição for falsa, a repetição será encerrada. Após a execução das instruções dentro do laço for, o comando de `in(de)cremento` é executado. No exemplo abaixo esse comando é o `$contador++`. Após o incremento, a condição de parada é avaliada novamente. Caso a condição seja verdadeira, as instruções são executadas novamente e, caso a condição seja falsa, a repetição é encerrada.

```
<?php  
    for($contador = 0; $contador < 10; $contador++){  
        echo "$contador ";  
    }  
?>
```

O exemplo abaixo mostra um programa que efetua a soma dos números inteiros do intervalo de 1 a 10. Nele, a variável `$soma` é responsável por acumular as somas dos números a cada repetição. Para isso, deve-se inicializar a variável `$soma` antes do laço, de forma que ela tenha um valor válido quando for utilizada pela primeira vez.

```
<?php
    $soma = 0;
    for($i = 1; $i <= 10; $i++){
        $soma += $i; // $soma = $soma + $i;
    }
    echo "A soma dos números inteiros do intervalo de 1 a 10 é
        <b>$soma</b>.";
?>
```

4.2.2 Comando WHILE

Neste tipo de estrutura de repetição, não se conhece previamente o número de repetições que serão executadas. Também são chamadas de estruturas de repetição condicionais pelo fato de encerrarem sua execução mediante uma determinada condição.

O número de repetições está ligado a uma condição sujeita à modificação pelas instruções do interior do laço. A sintaxe do comando é descrita abaixo.

```
// inicialização
while (condição_de_parada){
    // instruções
    // in(de)cremento
}
```

Vale ressaltar que, ao contrário do for, no while deve-se fazer o gerenciamento do incremento. No exercício resolvido abaixo, é descrito um programa que exibe a soma dos números inteiros do intervalo de 1 a 10.

```
<?php
    $soma = 0;
    $i = 1;
    while($i <= 10){
        $soma += $i; // $soma = $soma + $i;
        $i++;
    }
    echo "A soma dos números inteiros do intervalo de 1 a 10 é
        <b>$soma</b>.";
?>
```

A sequência de comandos será repetida enquanto a condição for verdadeira (`$i <= 10`). Se a condição não for mais verdadeira, a repetição é interrompida e a sequência de comandos, que estiver logo após o `}` da estrutura, passa a ser executada.

4.2.3 Comando DO..WHILE

A estrutura `do..while` é muito parecido com a estrutura `while`. A diferença está no fato de que o bloco de instruções é executado no mínimo uma vez. A sintaxe do comando é descrita abaixo.

```
// inicialização
do {
    // instruções
    // in(de)cremento
} while (condição_de_parada);
```

No exercício resolvido abaixo, é descrito um programa que exibe a soma dos números inteiros do intervalo de 1 a 10.

```
<?php
    $soma = 0;
    $i = 1;
    do{
        $soma += $i; // $soma = $soma + $i;
        $i++;
    }while($i <= 10);
    echo "A soma dos números inteiros do intervalo de 1 a 10 é
<b>$soma</b>.";
?>
```

Note que o `do..while` executa o bloco de comandos e depois testa a condição (`$i <= 10`) no final da estrutura. Assim como na estrutura `while`, deve-se fazer o gerenciamento do incremento (`$i++`).

4.2.4 Comando FOREACH

A estrutura `foreach` oferece uma maneira mais conveniente de percorrer os elementos de um array. A sintaxe do comando é descrita abaixo.

```
foreach ($nome_do_vetor as $elemento) {
    // instruções
}
```

O vetor `$nome_do_vetor` é percorrido do primeiro ao último índice e a cada iteração o valor do elemento corrente do vetor é atribuído à variável `$elemento` e o ponteiro interno do array é avançado.

4.3. Juntando tudo!

Até aqui, foram apresentados diversos conceitos básicos para a construção de uma página web dinâmica utilizando a linguagem PHP. O exemplo descrito nessa seção visa juntar os conteúdos já vistos e aplicá-los em um caso prático.

Considere um sistema que permita o cálculo da média final dos alunos de uma turma e, automaticamente, informe se os alunos estão aprovados, em reavaliação ou reprovados. O número de alunos da turma deve ser informado ao sistema e este deve fornecer um formulário que permita inserir as informações de cada aluno. As informações são o nome do aluno e suas notas na prova, no trabalho e no exercício prático. O sistema deve calcular a média final de cada aluno, que consiste na média aritmética das três notas.

Caso um aluno obtenha média final inferior a 4.0, ele está reprovado. Se sua média final estiver entre 4.0 (inclusive) e 6.0, o aluno está em reavaliação. Se a média for maior ou igual a 6.0, o aluno está aprovado. Ao final da execução, o sistema deve apresentar uma página web com uma tabela, cujas linhas mostram o nome do aluno, suas notas da prova, do trabalho e do exercício prático, sua média final e sua situação na disciplina.

A página `index.html`, abaixo, pede ao usuário do sistema a quantidade de alunos, cujas notas serão inseridas.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Fechamento de Notas - Página Inicial</title>
  </head>
  <body>
    <form action="form_notas.php" method="post">
      <fieldset>
        <legend>Digite a quantidade de alunos:</legend>
        <input type="number" name="quantidadeAlunos" min="1"
step="1" />
        <input type="submit" value="Enviar" />
      </fieldset>
    </form>
  </body>
</html>
```

A página form_notas.php recebe a quantidade de alunos e gera o formulário com os campos apropriados para inserir as informações de nome e notas de cada aluno. É importante ressaltar aqui, a estratégia para gerar os valores dos nomes das tags input. Para que haja campos com nomes únicos para cada aluno, utiliza-se a variável \$i concatenada ao valor da opção name das tags.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Fechamento de Notas - Formulário de Notas</title>
  </head>
  <body>
    <form action="processa_notas.php" method="post">
      <?php
        $quantidadeAlunos = $_POST["quantidadeAlunos"];
        for($i = 1; $i <= $quantidadeAlunos; $i++){
          echo '<fieldset>
            <legend>Dados do aluno ' . $i . ' :</legend>
            <p>
              <label>Nome:</label>
              <input type="text" name="nome' . $i . '" />
            </p>
            <p>
              <label>Prova:</label>
              <input type="number" name="prova' . $i . '"
                min="0" max="10" step="0.1" />
              <label>Trabalho:</label>
              <input type="number" name="trabalho' . $i . '"
                min="0" max="10" step="0.1" />
              <label>Exercícios de Fixação:</label>
              <input type="number" name="exercicio' . $i . '"
                min="0" max="10" step="0.1" />
            </p>
          </fieldset>';
        }
        echo '<input type="hidden" name="quantidadeAlunos"
          value="' . $quantidadeAlunos . '" />';
      ?>
      <input type="submit" value="Enviar" />
      <input type="reset" value="Limpar" />
    </form>
  </body>
</html>
```

A página `processa_notas.php` é responsável por receber as notas, calcular as médias e exibir a situação final de cada aluno. As informações de cada aluno são recebidas, utilizando um artifício semelhante ao da página anterior. Aqui, contudo, a string usada como índice associativo do supervetor (`$_GET` ou `$_POST`) é interpolada com a variável `$i`, obtendo-se assim, os nomes dos campos referentes aos alunos enviados por meio do formulário. Após o cálculo das médias e a definição da situação do aluno, uma linha da tabela é gerada para cada aluno com suas informações.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Fechamento de Notas - Resultado</title>
  </head>
  <body>
    <table border="1">
      <thead>
        <tr>
          <th>Nome</th>
          <th>Prova</th>
          <th>Trabalho</th>
          <th>Exercícios</th>
          <th>Média</th>
          <th>Situação</th>
        </tr>
      </thead>
      <tbody>
        <?php
          $quantidadeAlunos = $_POST["quantidadeAlunos"];
          $mediaTurma = 0;
          for($i = 1; $i <= $quantidadeAlunos; $i++){
            $nome = $_POST["nome$i"];
            $prova = $_POST["prova$i"];
            $trabalho = $_POST["trabalho$i"];
            $exercicio = $_POST["exercicio$i"];
            $media = ($prova + $trabalho + $exercicio) / 3;
            if($media < 4){
              $situacao = "Reprovado";
            }elseif($media < 6){
              $situacao = "Reavaliação";
            }else{
              $situacao = "Aprovado";
            }
            $mediaTurma += $media;
            echo '<tr>
              <td>' . $nome . '</td>
              <td>' . number_format($prova, 1) . '</td>
              <td>' . number_format($trabalho, 1) . '</td>
              <td>' . number_format($exercicio, 1) . '</td>
              <td>' . $media . '</td>
              <td>' . $situacao . '</td>
            </tr>';
          }
        </tbody>
    </table>
  </body>
</html>
```

```

        <td>' . number_format($exercicio, 1) . '</td>
        <td>' . number_format($media, 1) . '</td>
        <td>' . $situacao . '</td>
    </tr>';
    }
    ?>
</tbody>
<tfoot>
<tr>
    <td colspan="5">Média da turma</td>
    <td>
        <?php
            $mediaTurma /= $quantidadeAlunos;
            echo number_format($mediaTurma, 1);
        ?>
    </td>
</tr>
</tfoot>
</table>
</body>
</html>

```

4.4. Exercícios de Fixação

Exercício 1. Faça uma página HTML para enviar dois números inteiros quaisquer para um programa PHP, que deverá calcular e escrever a multiplicação entre os números recebidos utilizando para isso apenas o operador “+”, por exemplo:

$$(3 * 5) = 5 + 5 + 5$$

$$(4 * 12) = 12 + 12 + 12 + 12$$

Exercício 2. Dado que, um número é primo quando é divisível apenas por 1 e por ele mesmo, faça um programa em PHP que receba, por meio de uma página HTML, um número inteiro maior que 1, verifique se o número fornecido é primo ou não e mostre uma mensagem de “número primo” ou de “número não primo”.

Exercício 3. Faça um programa em PHP que receba, por meio de uma página HTML, um número, calcule e mostre a tabela (gerar uma tabela do HTML) deste número, conforme figura abaixo:



← ⓘ localhost/testes/tabuada			
Tabuada do 9			
9	x	1	= 9
9	x	2	= 18
9	x	3	= 27
9	x	4	= 36
9	x	5	= 45
9	x	6	= 54
9	x	7	= 63
9	x	8	= 72
9	x	9	= 81
9	x	10	= 90

Exercício 4. Um funcionário de uma empresa recebe, anualmente, aumento salarial. Sabe-se que:

- Esse funcionário foi contratado em 2005, com salário inicial de R\$ 1000,00;
- Em 2006, ele recebeu um aumento de 1,5% sobre seu salário inicial;
- A partir de 2007 (inclusive), os aumentos salariais sempre corresponderam ao percentual do ano anterior mais 0,1%.

Desta forma, criar um programa em PHP que obtém o ano atual (do sistema), que determina e apresenta o salário atual deste funcionário.

Exercício 5. Criar um programa em PHP que receba de uma página HTML o valor de um carro novo, calcule e mostre uma tabela (tabela HTML) com os seguintes dados: preço final, quantidade de parcelas e valor das parcelas. Considere que:

- O preço final para compra à vista tem desconto de 20%;
- A quantidade de parcelas pode ser: 6, 12, 18, 24, 30, 36, 42, 48, 54 e 60; e
- Os percentuais de acréscimo, dependendo da quantidade de parcelas, encontra-se na tabela a seguir:

Quantidade de parcelas	Percentual de acréscimo sobre o preço final
6	3%
12	6%
18	9%
24	12%

30	15%
36	18%
42	21%
48	24%
54	27%
60	30%

Exercício 6. Faça uma página HTML para enviar um número inteiro positivo para um programa PHP, que deverá calcular e escrever o fatorial do número recebido. Sabe-se que:

n	n!
0	1
1	1
2	2
3	6
4	24
5	120
6	720
7	5.040
...	...

Exercício 7. Fazer um programa em PHP para gerar e escrever os 4 (quatro) primeiros números perfeitos. Um número perfeito é aquele que é igual a soma dos seus divisores. Por exemplo:

$$6 = 1 + 2 + 3$$

$$28 = 1 + 2 + 4 + 7 + 14$$

Exercício 8. Fazer um programa em PHP para escrever os X primeiros termos da Sequência de Fibonacci. O valor de X é lido por meio de um formulário HTML. Por exemplo, os oito primeiros termos são:

$$0 - 1 - 1 - 2 - 3 - 5 - 8 - 13$$

Exercício 9. Um cinema de Araraquara fez uma pesquisa entre os seus espectadores. Cada espectador respondeu a um questionário, no qual constava sua idade e sua opinião em relação ao filme assistido:

Opinião	Código
Ótimo	3
Bom	2
Regular	1

Fazer um programa em PHP para receber os dados da pesquisa (idades e opiniões dos espectadores, por meio de páginas HTML – uma para saber o número de espectadores e outra com o formulário de coleta de dados), calcule e apresente:

- A média das idades das pessoas que responderam ótimo;
- A quantidade de pessoas que responderam regular;
- A porcentagem de pessoas que responderam bom, entre todos os espectadores analisados.

Exercício 10. Foi feita uma pesquisa sobre a audiência de canal de TV em várias casas de Araraquara, em determinado dia. Para cada casa foi fornecido o número do canal (4, 5, 9, 11) e o número de pessoas que estavam assistindo aquele canal. Somente as casas com TV ligada entraram na pesquisa. Faça um programa em PHP que:

- Receba um número de casas pesquisadas;
- Gere um formulário para coletar os dados das casas pesquisadas (canal e números de espectadores);
- Calcule e apresente a porcentagem de audiência de cada canal.

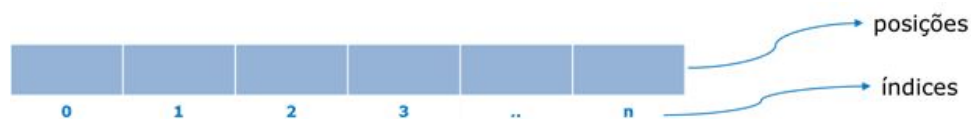
Exercício 11. Foi feita uma pesquisa entre os habitantes de uma região da cidade de Araraquara (coletar o número de habitantes que participaram da pesquisa). Foram coletados os dados de idade, sexo (M/F) e salário. Faça um programa em PHP que, após receber os dados, calcule e mostre:

- A média dos salários do grupo pesquisado;
- A maior e menor idade do grupo;
- A quantidade de mulheres com salário até R\$ 1000,00;
- A idade e o sexo da pessoa que possui o menor salário.

Capítulo 5 - Variáveis compostas

5.1. Vetores

Um vetor é uma variável composta (array) que pode armazenar vários valores ao mesmo tempo, agrupados sob o mesmo nome (identificador) e diferenciados entre si através de índices. Trata-se de uma variável unidimensional, em que cada índice indica uma posição de memória em que fica armazenado um elemento do vetor. Os índices podem ser um número ou um texto e devem aparecer entre colchetes “[]” logo após o identificador do vetor. Abaixo, temos uma representação gráfica.



Em PHP não é necessário que um array seja declarado antes de seu uso, nem mesmo indicar o número máximo de elementos que eles devem conter. Além disso, os índices de um array podem ser numéricos ou caracteres.

No exemplo abaixo, é descrito um programa que declara quatro vetores de maneiras diferentes.

```
<?php
    $vetor1 = array();
    $vetor1[0] = 1;
    $vetor1[1] = 2;
    $vetor1[2] = 3;

    $vetor2 = array(4, 5, 6);

    $vetor3 = array();
    $vetor3["codigo"] = 1234;
    $vetor3["nome"] = "Ana Maria";
    $vetor3["endereco"] = "Rua Nova, 999";

    $vetor4 = array("codigo" => 1234, "nome" => "Ana Maria",
                    "endereco" => "Rua Nova, 999");
?>
```

O `$vetor1` possui índices de 0 a 2 e armazena respectivamente números inteiros de 1 a 3 em cada posição. O `$vetor2` também armazena números inteiros de 4 a 6, sendo 0, 1 e 2 os índices correspondentes. O `$vetor3` armazena no índice “código” o valor “1234”, no índice “nome” o valor “Ana Maria” e no índice “endereço” o valor “Rua Sem Nome, sem número”. Note que o `$vetor4` armazena as mesmas informações, com os mesmos índices do `$vetor3`, mas é declarado de maneira diferente.

5.1.1. Leitura e escrita

Em diversas situações, necessitamos ler e/ou escrever em vetores. Para tanto, é comum a utilização da estrutura de repetição `for` ou `for..each` seguindo os exemplos abaixo.

```
<?php
    $vetor = array();

    for($i = 0; $i < 5; $i++){
        $vetor[$i] = $i + 1;
    }

    echo "[ ";
    for($i = 0; $i < sizeof($vetor); $i++){
        echo "$vetor[$i] ";
    }
    echo " ] <br /><br />";

    $soma = 0;
    foreach($vetor as $item){
        $soma = $soma + $item;
    }

    echo "Soma = " . $soma;
?>
```

No primeiro laço, cinco posições do `$vetor` são percorridas e são atribuídos valores de acordo com o valor de cada índice. Já no segundo laço o `$vetor` é exibido. Note que a função `sizeof()` determina o tamanho do vetor. No terceiro laço, é utilizada a instrução `for..each`, onde o vetor é percorrido do primeiro ao último índice e a cada iteração o valor do elemento corrente do vetor é atribuído à variável `$item` e o ponteiro interno do array é avançado. Cada item do vetor é somado e armazenado na variável acumuladora `$soma`, que é exibida no final do código.

Os dados do formulário enviados pelo navegador ao servidor são disponibilizados ao programa PHP por meio de arrays superglobais.

No exemplo abaixo, é descrito um programa que armazena nas variáveis `$nome` e `$idade` as informações disponibilizadas pela variável superglobal `$_POST`.

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

    echo "<h2>Nome recebido: $nome</h2>";
    echo "<h3>Idade recebida: $idade</h3>"
?>
```

O programa em PHP deve acessar o array apropriado, fornecendo o nome dos campos do formulário como chave associativa. Isto significa que, ao invés de acessar os elementos do array pela forma tradicional, utilizando índices numéricos, strings serão utilizadas como índices. No exemplo acima, as chaves associativas são `"nome"` e `"idade"`, respectivamente.

5.2. Matrizes

Os arrays são estruturas lineares, ou seja, cada elemento de um array armazena um valor por vez, por exemplo um número ou uma string. Em algumas situações, pode ser necessário que cada elemento de um array armazene vários valores por vez, ou seja, armazene um outro array. Por exemplo, um array pode armazenar uma agenda telefônica, sendo que cada elemento corresponde a um contato e pode conter um outro array armazenando uma lista de números de telefone daquele contato. Para tanto, devem ser utilizadas estruturas multidimensionais chamadas *matrizes*.

As matrizes possuem um único identificador, mas possuem dois ou mais índices para referenciar uma posição de memória. O acesso para cada dimensão de uma matriz é realizado por meio dos colchetes []. No caso do exemplo da agenda telefônica, `$matrizAgendaTel[0]` permite acessar o array que contém a lista de números de telefone do primeiro contato. Para acessar o segundo número de telefone do primeiro contato, deve-se utilizar `$matrizAgendaTel[0][1]`. No trecho de código php abaixo, são definidos três vetores com números de telefone e depois criada uma matriz para armazenar esses vetores. Depois, para cada índice da primeira dimensão da matriz, são exibidos na tela os telefones armazenados em cada vetor, utilizando o comando `foreach`.

```

<?php
    $telefones0 = array("1234-5678", "98765-4321");
    $telefones1 = array("1234-0011", "98765-9999", "91234-5555");
    $telefones2 = array("3322-8765");
    $matrizAgendaTel = array($telefones0, $telefones1, $telefones2);

    for($i = 0; $i < 3; $i++){
        echo "Telefones do contato " . $i . ": ";
        foreach($matrizAgendaTel[$i] as $telefone){
            echo "$telefone ";
        }
        echo "<br />";
    }
?>

```

No exemplo abaixo, a variável `matriz` é um array com nove elementos (números inteiros). Note que inicialmente, a variável pode ser um vetor (unidimensional) ou uma matriz (bidimensional). Dentro do primeiro laço `for`, cada posição (índice) da variável `$matriz` é inicializado como um novo array, ou seja, é criado um array com duas dimensões (matriz). No segundo laço `for`, cada posição da matriz armazena o valor da variável `$cont`.

```

<?php
    $matriz = array();

    $cont = 1;
    for($i = 0; $i < 3; $i++){
        $matriz[$i] = array();
        for($j = 0; $j < 3; $j++){
            $matriz[$i][$j] = $cont++;
        }
    }

    foreach($matriz as $vetor){
        echo "[ ";
        foreach($vetor as $item){
            echo "$item ";
        }
        echo " ]<br />";
    }

?>

```

Na sequência do exemplo, temos duas estruturas **foreach**: a primeira percorre a variável **\$matriz** dividindo-a num vetor e a segunda dividindo este vetor em itens, exibindo-os na tela.

O exemplo seguinte cria duas variáveis do tipo array (vetores) compostos por letras (a, b, c, d, e, f) e a matriz é criada a partir deste dois vetores.

```
<?php
    $vetor1 = array("a", "b", "c");
    $vetor2 = array("d", "e", "f");
    $matriz = array($vetor1, $vetor2);

    echo "[ ";
    foreach($matriz as $vetor){
        echo "[ ";
        foreach($vetor as $item){
            echo "$item ";
        }
        echo " ]";
    }
    echo " ]";
?>
```

Novamente, na sequência do exemplo, temos duas estruturas **foreach**: a primeira percorre a variável **\$matriz** dividindo-a num vetor e a segunda dividindo este vetor em itens, exibindo-os na tela.

5.3. Exercícios de Fixação

Exercício 1. Faça um formulário HTML para ler os elementos de um array (vetor) com 10 valores reais, enviar os elementos lidos para um programa em PHP que encontra e mostra o maior e o menor valor armazenado no array.

Exercício 2. Faça um formulário HTML para ler os elementos de dois vetores: R de 3 elementos inteiros e S de 7 elementos inteiros. Os números lidos devem ser enviados para um programa em PHP que gera um vetor X de 10 elementos cujas 3 primeiras posições contenham os elementos de R e as 7 últimas posições, os elementos de S. Mostrar o vetor X.

Exercício 3. Faça um formulário HTML para ler um vetor U de 10 elementos reais. A seguir, os dados devem ser enviados para um programa em PHP, que troca o primeiro elemento com o último, o segundo com penúltimo etc. até o quinto com o sexto e escreve o vetor U assim modificado.

Exercício 4. Faça um formulário HTML para ler dois vetores, X e Y de 10 elementos inteiros cada um. Os dados lidos são enviados a um programa em PHP, que intercala os elementos desses dois vetores, formando assim um novo vetor R de 20 elementos, onde nas posições

ímpares de R estejam os elementos de X e nas posições pares os elementos de Y (Considere o zero como PAR). O programa deve escrever o vetor R, após sua completa geração.

Exercício 5. Faça um programa em PHP que receba o total das vendas de cada vendedor de uma loja e armazene-os em um vetor. Receba também o percentual de comissão de cada vendedor e armazene-os em outro vetor. Receba os nomes desses vendedores e armazene-os em um terceiro vetor. Existem apenas 5 vendedores na loja. O programa deve calcular e mostrar:

- Um relatório com os nomes dos vendedores e os valores a receber referentes à comissão por suas vendas;
- O total das vendas de todos os vendedores;
- O maior valor a receber e o nome de quem o receberá;
- O menor valor a receber e o nome de quem o receberá.

Exercício 6. Faça um programa em PHP que armazena os nomes de sete alunos em um vetor e que armazena também a média final desses alunos. O programa deve calcular e mostrar:

- O nome do aluno com maior média (desconsiderar empates);
- A média da turma;
- A diferença entre a maior e a menor médias.

Exercício 7. Faça um programa em PHP que armazena os nomes de cinco produtos em um vetor e os seus respectivos preços em outro vetor. O programa deve calcular e mostrar:

- A quantidade de produtos com preço inferior a R\$ 50,00;
- O nome dos produtos com preço entre R\$ 50,00 e R\$ 100,00;
- A média dos preços dos produtos com preço superior a R\$ 100,00.

Exercício 8. Faça uma página Web com um formulário HTML para preencher uma matriz 5x5 com números inteiros. Um programa em PHP deve calcular e mostrar a soma:

- dos elementos da linha 4;
- dos elementos da coluna 2;
- dos elementos da diagonal principal;
- dos elementos da diagonal secundária;
- de todos os elementos da matriz.

Exercício 9. Faça uma página Web com um formulário HTML para receber os valores das vendas de uma loja. Os dados devem ser armazenados em uma matriz 12x4, em que cada linha representa um mês do ano e cada coluna representa uma semana do mês. Um programa deverá calcular e mostrar:

- O total vendido em cada mês do ano, mostrando o nome do mês por extenso;
- O total vendido em cada semana durante todo o ano, que os proprietários da loja possam identificar a semana que mais teve vendas;

- O total vendido pela loja no ano.

Exercício 10. Faça uma página Web com um formulário HTML para receber o estoque atual e o custo de três produtos de uma distribuidora, que são armazenados em quatro armazéns. Os dados devem ser armazenados em uma matriz 5x3, sendo que as 4 primeiras linhas contêm os estoques dos produtos nos armazéns e a última linha contém o custo de cada produto. Um programa deverá calcular e mostrar (OBS: devem ser desconsiderados empates):

- A quantidade de itens de produto armazenados em cada armazém;
- Qual armazém possui maior estoque do produto 2;
- Qual armazém possui menor estoque;
- Qual o custo total de cada produto;
- Qual o custo total de cada armazém.

Capítulo 6 - Sessões e Cookies

Ao desenvolver um programa Web, é comum a necessidade de manter informações sobre os usuários enquanto eles navegam pelas páginas ou até que eles saiam do site. Para isto, é importante saber utilizar sessões e cookies em linguagem PHP.

Quando diferentes usuários acessam e navegam em páginas em HTML de um site, o servidor web trata esses acessos como requisições independentes. Ele não sabe que pessoas diferentes estão acessando e requisitando suas páginas. Os mecanismos fornecidos pelas sessões e pelos cookies permitem armazenar informações de cada usuário enquanto eles estiverem navegando entre as páginas de um site, possibilitando a utilização dessas informações para vários fins, tais como autenticação de usuário, carrinho de compras, exibição de anúncios e personalização de páginas, entre outros.

6.1. Sessões

Uma sessão pode ser definida como o período de tempo no qual uma pessoa navega pelas páginas de um site. Assim, quando uma pessoa acessa o site, uma nova sessão pode ser iniciada. Nela, são registradas diversas variáveis que ficam armazenadas em arquivos no servidor e que podem ser acessadas em qualquer página do site, enquanto a sessão estiver aberta. Cada sessão possui um identificador único, chamado de session ID (SID).

Em PHP, uma sessão pode ser criada utilizando-se a função `session_start`, que serve também para restaurar os dados de uma sessão, com base no session ID corrente. Essa função deve ser chamada antes de qualquer saída produzida pelo navegador (no início do arquivo). Uma outra forma de criar uma sessão é habilitando a diretiva `session.auto_start` do arquivo `php.ini`, presente no servidor web. Com essa diretiva, sempre que um usuário entrar no site, será automaticamente criada uma sessão. A primeira forma será adotada nos exemplos apresentados.

Ao registrarmos uma variável em uma sessão, ela se torna disponível para todas as páginas que serão acessadas até o encerramento da sessão. Para registrar uma variável, deve-se adicionar diretamente entradas ao array superglobal `$_SESSION`, conforme o exemplo:

Caso uma variável de sessão tenha sido registrada e não for mais utilizada nos próximos acessos, ela pode ser eliminada. Isto é feito por meio do comando `unset` do PHP, como mostra o exemplo:

Para demonstrar o funcionamento do registro de variáveis em uma sessão, considere o exemplo abaixo, que utiliza duas páginas chamadas de `pagina1.php` e `pagina2.php`. Na primeira página, são definidas e registradas duas variáveis de sessão e é exibido um link para a segunda página. A segunda página recupera os dados da sessão e exibe os valores atribuídos ainda na primeira página. O código da `pagina1.php` é o seguinte:

```
<?php
    session_start();
    echo "Página 1";
    $_SESSION["nome"] = "Joao";
    $_SESSION["sobrenome"] = "da Silva";
    echo "<br><a href=\"pagina2.php\">Página 2</a>";
?>
```

Na segunda página, é preciso, primeiramente, restaurar os dados da sessão por meio do comando `session_start()`. Em seguida, os valores das variáveis `nome` e `sobrenome` são recuperados pela acesso ao vetor `$_SESSION`. Veja o código:

```
<?php
    session_start();
    echo "Página 2";
    echo $_SESSION["nome"] . "<br>";
    echo $_SESSION["sobrenome"] . "<br>";
    echo "<br><a href=\"pagina1.php\">Página 1</a>";
?>
```

6.1.1. Juntando tudo

Neste exemplo, veremos como utilizar formulários, envio de informações e sessões para realizar o cadastro de diversos produtos em uma sessão utilizando uma única página HTML.

Antes de entender o exemplo, é preciso conhecer a função `empty()`. Esta função testa se uma determinada variável, informada como parâmetro, está vazia. Uma variável é considerada vazia se ela não existir ou se o seu valor for igual a `FALSE` (falso). Para um array, isto significa não possuir elemento algum.

A função `empty()` será utilizada para determinar o contexto do carregamento da página. Ao aplicar, por exemplo, a função `empty()` no vetor superglobal `$_POST`, é possível saber se o carregamento da página atual foi feito por meio da submissão de um formulário (`$_POST` contendo dados de formulário), ou se a página está sendo aberta sem receber informações da página anterior (`$_POST` vazio). Utilizaremos esse status do vetor superglobal para saber o momento de exibir o formulário de cadastro do produto e o momento de registrar as informações de um produto na sessão.

```
<?php
    session_start();
?>
```

```

<html lang="pt-BR">
  <head>
    <meta charset="utf-8"/>
    <title>Formulário de Cadastro de Produtos</title>
  </head>
  <body>
    <?php

        if(empty($_POST)){
            $_SESSION["cadastrou"] = 0;
            echo ' <form action="form_cadastro_produto.php" method="post">
                <fieldset>
                    <legend>Cadastro de produto</legend>
                    <p>
                        <label>Nome:</label>
                        <input type="text" name="nome" size="30"/>
                    </p>
                    <p>
                        <label>Preço</label>
                        <input type="number" step="0.01"
                            name="preco"/>
                    </p>
                    <input type="submit" value="Enviar"/>

                </fieldset>
            </form>
            <a href="lista_produtos.php">
                Listar produtos cadastrados</a>';
        }else{
            if($_SESSION["cadastrou"] == 0){
                $_SESSION["nomes"][] = $_POST["nome"];
                $_SESSION["precos"][] = $_POST["preco"];
                $_SESSION["cadastrou"] = 1;
                echo ' <br/><br/>
                    <div class="center">
                        <h2>Produto cadastrado com sucesso.</h2>
                        <p><a href="form_cadastro_produto.php">Voltar</a></p>
                    </div>';
            }else{
                echo ' <br/><br/>
                    <div class="center">
                        <h2>Produto já cadastrado.</h2>
                        <p><a href="form_cadastro_produto.php">Voltar</a></p>
                    </div>';
            }
        }
    <?>
  </body>
</html>

```

No exemplo, o vetor superglobal `$_SESSION` também armazena uma *flag* com o nome “cadastrou” para verificar se os dados de um formulário já foram devidamente submetidos e registrados na sessão. Isto previne que, ao atualizar a página, o usuário registre novamente os dados submetidos pelo formulário. Este artifício não previne, contudo, que usuários cadastrem produtos com o mesmo nome.

O código a seguir pertence à página `lista_produtos.php`. Sua função é exibir todos os produtos cadastrados na sessão e seus respectivos preços em uma tabela. O link para essa página encontra-se abaixo do formulário exibido na página `form_cadastro_produto.php`. Caso a sessão não possua produtos cadastrados (vetor `$_SESSION` vazio), a página emite uma mensagem de alerta.

```
<?php
    session_start();
?>
<html lang="pt-BR">
    <head>
        <meta charset="utf-8"/>
        <title>Lista de Produtos</title>
        <style>
            .center{
                margin: auto;
            }
            .tabela1 tbody tr:nth-child(odd){
                background-color: #E9E9E9;
            }
        </style>
    </head>
    <body>
        <?php
            if(!empty($_SESSION["nomes"])){
                echo ' <div class="center">
                    <table class="tabela1" width="30%" border="1">
                        <thead>
                            <tr>
                                <th>Nome</th>
                                <th>Preço</th>
                            </tr>
                        </thead>
                        <tbody>';
                for($i = 0; $i < sizeof($_SESSION["nomes"]); $i++){
                    $nome = $_SESSION["nomes"][$i];
                    $preco = $_SESSION["precos"][$i];
                    echo ' <tr>
                        <td>' . $nome . '</td>
                        <td>' . $preco . '</td>
                    </tr>';
                }
            }
        </?php>
    </body>
</html>
```

```

    }
    echo          '</tbody>'
                </table>
            </div>
            <a href="form_cadastro_produto.php">
                Cadastrar produto</a>';
    }else{
        echo      '<br/><br/>'
                <div class="center">
                    <h2>Nenhum produto cadastrado.</h2>
                </div>
                <a href="form_cadastro_produto.php">
                    Cadastrar produto</a>';
    }
?>
</body>
</html>

```

6.1.2. Exercícios de fixação

Exercício 1. Faça um programa em PHP que gerencia os dados de pessoas, armazenando-os em uma agenda de contatos. Assim, crie:

- Um programa `form_cadastro_pessoa.php`, como mostra a Figura 1, para receber os dados de uma pessoa (nome, telefone e endereço) e armazene-os na sessão. Após armazenar os dados, o programa deve apresentar a mensagem de sucesso.
- Um programa `lista_pessoas.php` para gerar uma tabela com os dados de todas as pessoas, ou seja, a agenda de contatos do usuário, Figura 2.
- Um programa `form_consulta_pessoa.php` para que o usuário informe o nome da pessoa e retorne os dados dessa pessoa ou uma mensagem que a pessoa não foi encontrada, Figura 3.

form_cadastro_pessoa.php (sem post)

[Cadastrar Pessoa](#) | [Listar pessoas](#) | [Consultar Pessoa](#)

Nome: Telefone:

Endereço:

form_cadastro_pessoa.php (com post)

[Cadastrar Pessoa](#) | [Listar pessoas](#) | [Consultar Pessoa](#)

Pessoa cadastrada com sucesso.

Figura 1. Formulário de cadastro de pessoas.

lista_pessoas.php (com pessoas cadastradas)

[Cadastrar Pessoa](#) | [Listar pessoas](#) | [Consultar Pessoa](#)

Nome	Telefone	Endereço
Ana Maria	3333-1111	Rua 0, 100. Centro, Araraquara, SP
Juliana	3322-2222	Av. 36, 1000. Centro, Araraquara, SP
Mariana	2211-3333	Alameda Paulista, 30, Vila Xavier, Araraquara, SP

lista_pessoas.php (sem pessoas cadastradas)

[Cadastrar Pessoa](#) | [Listar pessoas](#) | [Consultar Pessoa](#)

Ainda não há pessoas cadastradas.

Figura 2. Lista de pessoas cadastradas.

form_consulta_pessoa.php (sem post)

[Cadastrar Pessoa](#) | [Listar pessoas](#) | [Consultar Pessoa](#)

Nome:

form_consulta_pessoa.php (com post)

[Cadastrar Pessoa](#) | [Listar pessoas](#) | [Consultar Pessoa](#)

Ana Maria encontrada:
Nome: Ana Maria
Telefone: 3333-1111
Endereço: Rua 0, 100, Centro, Araraquara, SP

Figura 3. Formulário de consulta de pessoas.

6.2. Cookies

Uma outra forma de compartilhar dados entre diferentes páginas, ou até mesmo em diferentes acessos, é por meio de cookies. Um cookie é um arquivo texto que fica armazenado no computador do usuário e contém informações registradas durante o acesso do usuário a uma página. Esse arquivo pode ser recuperado posteriormente pelo servidor para utilização dessas informações.

Cookies podem permanecer armazenados no computador dos usuários por vários dias, ao contrário do que acontece com as sessões, que mantêm os dados somente durante o tempo em que usuário permanece no site.

Um cookie é formado por um par nome/valor, ou seja, ele possui um nome pelo o qual ele é referenciado e um valor associado a esse nome. Assim como ocorre com as sessões, a linguagem PHP disponibiliza funções para a manipulação dos cookies.

A principal função é chamada de `setcookie()` e é utilizada para enviar cookies ao computador do usuário. A execução dessa função serve tanto para a definição de um cookie, como para a sua exclusão. Por exemplo, para definir um cookie que armazena o nome de uma pessoa, pode-se usar o comando

```
setcookie("username", "xpto");
```

Neste caso, o valor "xpto" estará associado ao nome "username". Para excluir esse cookie, basta utilizar a mesma função da seguinte forma:

```
setcookie("username");
```

A função `setcookie` possui uma série de parâmetros, que permitem determinar diversas configurações de um cookie. Por exemplo, o seguinte comando define um cookie válido por dois dias (48 horas):

```
setcookie("username", "xpto", time()+172800);
```

Outros parâmetros possíveis para a função `setcookie` estão descritos na tabela abaixo.

Parâmetro	Descrição
Nome	Indica o nome do cookie que está sendo enviado e é o único parâmetro obrigatório para a função
Valor	É o valor do cookie. Se não for fornecido, o servidor tentará excluir o cookie com o nome especificado
Validade	Define o tempo de validade do cookie. Deve ser expresso no formato-padrão de tempo do UNIX (número de segundos após 1º de janeiro de 1970, às 0h).
Caminho	Caminho no servidor para o qual o cookie estará disponível. Se for definido o valor "/", ele estará disponível para todo o domínio especificado no parâmetro domínio. O valor padrão é o diretório corrente a partir do qual o cookie foi definido
Domínio	Domínio para o qual o cookie estará disponível.
Seguro	É um valor inteiro (0 ou 1) que indica se o cookie é seguro. Se for utilizado o valor 1, o cookie só será transmitido se a conexão for segura (HTTPS).
Somente HTTP	Se for igual a TRUE, o cookie estará acessível apenas sobre o protocolo HTTP e não acessível para linguagens de script, tal como JavaScript.

Para que o código PHP de uma página consiga acessar os dados de um cookie no computador do usuário, utiliza-se o array superglobal `$_COOKIE`. Se o cookie foi definido como

```
setcookie("username", "xpto");
```

na próxima página acessada pelo usuário, esse valor poderia ser acessado da seguinte maneira:

```
$_COOKIE["username"]
```

6.2.1 Juntando tudo

O exemplo a seguir mostra como fazer um sistema básico de autenticação. Ao acessar a página login.html, o usuário deverá fornecer o nome de usuário e sua senha. Estas informações, após o acionamento do botão “Enviar”, serão enviadas ao programa login.php, que redirecionará o usuário para uma página apropriada, de acordo com a validação de seu login.

login.html

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8"/>
    <title>Página de Login</title>
    <link rel="stylesheet" href="estilos.css"/>
  </head>
  <body>
    <form action="login.php" method="post">
      <fieldset>
        <p>
          <label>Nome de usuário:</label>
          <input type="text" name="nome_usuario" size="10"/>
        </p>
        <p>
          <label>Senha:</label>
          <input type="password" name="senha" size="10"/>
        </p>
        <p>
          <input type="submit" value="Enviar"/>
        </p>
      </fieldset>
    </form>
  </body>
</html>
```

Neste exemplo, utiliza-se uma folha de estilo simples, descrita a seguir, para formatação da página.

estilos.css

```
fieldset{
  margin: auto;
  width: 30%;
```

```

}

p{
    text-align: center;
}

label{
    display: inline-block;
    width: 120px;
    text-align: right;
}

input{
    width: 100px;
}

```

Ao receber as informações de login, a página login.php verifica se o nome de usuário e a senha estão corretos. Neste caso, ambos são fixos e possuem os valores “admin” e “123”. Em uma aplicação real, seria necessário acessar um banco de dados para recuperar essas informações. Se os dados estiverem corretos, são criados cookies para o nome de usuário e senha, com seus respectivos valores e o usuário é redirecionado automaticamente para a página index.php, por meio da função header(). Por outro lado, se as informações de login estiverem incorretas, o usuário é redirecionado para páginas com mensagens de erro específicas (erronomeusuario.html e errosenha.html).

login.php

```

<?php
    $nome_usuario = $_POST["nome_usuario"];
    $senha = $_POST["senha"];

    if($nome_usuario != "admin"){
        header("Location: erro_nome_usuario.html");
    }else{
        if($senha != "123"){
            header("Location: erro_senha.html");
        }else{ // usuário e senha corretos
            setcookie("nome_usuario", $nome_usuario);
            setcookie("senha", $senha);
            header("Location: index.php");
        }
    }
?>

```

erro_nome_usuario.html

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8"/>
    <title>Página de Erro</title>
    <link rel="stylesheet" href="estilos.css"/>
  </head>
  <body>
    <p>
      Usuário não encontrado!
    </p>
    <p>
      <a href="login.html">Voltar</a>
    </p>
  </body>
</html>
```

erro_senha.html

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8"/>
    <title>Página de Erro</title>
    <link rel="stylesheet" href="estilos.css"/>
  </head>
  <body>
    <p>
      Senha incorreta!
    </p>
    <p>
      <a href="login.html">Voltar</a>
    </p>
  </body>
</html>
```

index.php

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8"/>
    <title>Página de Boas-vindas</title>
    <link rel="stylesheet" href="estilos.css"/>
  </head>
  <body>
    <?php
      if(isset($_COOKIE["nome_usuario"])){
        $nome_usuario = $_COOKIE["nome_usuario"];
      }
      if(isset($_COOKIE["senha"])){
        $senha = $_COOKIE["senha"];
      }

      if(!(empty($nome_usuario) || empty($senha))){
        if($nome_usuario != "admin" || $senha != "123"){
          setcookie("nome_usuario");
          setcookie("senha");
          echo "Você não realizou o login!";
          exit;
        }
      }else{
        echo "Você não realizou o login";
        exit;
      }
      echo '
        <p>Seja bem-vindo(a)!</p>
        <p>
          <a href="logout.php">Logout</a>
        </p>';
    <?>
  </body>
</html>
```

Quando o usuário clique no link “Logout”, a página logout.php será acessada. Nessa página, a função `setcookie()` é chamada para cada nome de cookie, fazendo com que estes sejam excluídos.

logout.php

```
<?php
    setcookie("nome_usuario");
    setcookie("senha");
    header("Location: login.html");
?>
```

6.3. Exercícios de fixação

Exercício 1. Faça uma página HTML com um formulário para o usuário informar as preferências: cor de fundo (background-color) e tamanho do texto (text-size). Os dados informados devem ser enviados para um programa PHP, que recebe os dados e os armazena em cookies. Crie também um arquivo PHP que recupera os dados dos cookies e gera uma página com as preferências do usuário.

Capítulo 7 - Funções

Ao resolver um problema complexo, é comum dividi-lo em partes menores, ou seja, resolver o problema em etapas. A cada uma das partes menores bem definidas chamamos de módulo.

As funções (functions) são muito úteis para deixar o código dos programas mais organizado e mais modular. Além disso, as funções nos poupam da tarefa de ter de repetir determinado código toda vez que se precisa realizar uma tarefa.

7.1. Definição

As funções são programas menores inseridos em um programa principal ou em outro arquivo, que pode ser chamado a qualquer instante para executar uma tarefa específica. As funções podem realizar qualquer tipo de tarefa, por exemplo: somar dois números, testar se o valor de uma variável é válido, verificar se um número de CPF é válido, transformar uma string para letras maiúsculas, entre outras.

A sintaxe para a construção de uma função é a seguinte:

```
function nome_funcao(arg1, arg2, arg3, ... , argn){  
    comandos  
    [return <expressão>]  
}
```

Onde `function nome_funcao` é um identificador único, seguindo as mesmas regras de declaração de variáveis: não pode iniciar com número, não pode conter espaços, vírgulas, ponto, etc

Quando uma função é chamada, ela pode receber diversos valores, denominados argumentos (arg1, arg2,..., argn). Estes argumentos ou parâmetros são valores recebidos pela função no momento em que a mesma é chamada. Após a função ser chamada, estes valores são processados pela mesma. Cabe observar que é opcional a utilização de parâmetros. Neste caso, a sintaxe é escrita da seguinte maneira:

```
function nome_funcao()
```

Neste caso, utilizamos o comando `return` quando queremos atribuir o valor retornado a uma variável ou quando precisamos testar o valor de retorno de uma função. No exemplo seguinte, a função `somar` é criada e uma chamada para ela é incluída no programa principal:

```
<?php
function somar($operando1, $operando2){
    $resultado = $operando1 + $operando2;
    echo "$operando1 + $operando2 = $resultado";
}
$numero1 = 5;
$numero2 = 10;
somar($numero1, $numero2);
?>
```

No momento em que a função é chamada, a variável `$operando1` receberá o valor da variável `$numero1` (primeiro argumento), e a variável `$operando2` receberá o valor da variável `$numero2` (segundo argumento). Observe que as variáveis passadas como parâmetro não precisam ter o mesmo nome dos argumentos definidos na função. O objetivo da função é somar dois números e mostrar o resultado na tela e por isso o comando `return` não é usado.

No exemplo seguinte, a mesma função `somar` é criada, porém, o valor final (resultado da soma) é atribuído a variável `$resultado` (no programa principal).

```
<?php
function somar($operando1, $operando2){
    return $operando1 + $operando2;
}
$numero1 = 5;
$numero2 = 10;
somar($numero1, $numero2);
echo "$numero1 + $numero2 = " . somar($numero1, $numero2);
?>
```

Uma função também pode retornar um array contendo vários elementos ao invés de somente um, conforme o próximo exemplo:

```
<?php
function ordenar($vetor){
    for($i = 0; $i < sizeof($vetor); $i++){
        for($j = $i + 1; $j < sizeof($vetor); $j++){
            if($vetor[$i] > $vetor[$j]){
                $aux = $vetor[$i];
                $vetor[$i] = $vetor[$j];
                $vetor[$j] = $aux;
            }
        }
    }
}
```

```

        return $vetor;
    }
    $nomes = array("Juliana","Mariana", "Eliana", "Ana", "Adriana");
    $nomes = ordenar($nomes);
    foreach($nomes as $nome){
        echo "$nome ";
    }
?>

```

No exemplo acima, a função ordenar classifica os nomes contidos na variável \$vetor de forma a ordená-lo alfabeticamente.

7.2. Escopo de variáveis

O escopo de uma variável se refere ao contexto onde ela foi definida. As variáveis que são declaradas dentro de uma função são chamadas variáveis locais e são conhecidas somente dentro de seu próprio bloco, entre os símbolos abre e fecha chaves({ }). Variáveis locais existem apenas durante a execução do bloco de instruções onde estão declaradas Ou seja, são criadas quando se entra no bloco e destruída na saída.

No exemplo anterior, a variável \$vetor é local, ou seja, ela só existe e pode ser processada dentro da função ordenar. Já a variável \$vet é global, ou seja, ela pode ser acessada de qualquer parte do código (escopo global da aplicação).

As variáveis super globais são variáveis nativas do PHP e recebem esse nome pois estarão presentes em qualquer escopo do programa. As variáveis \$_GET, \$_POST (variáveis de requisição) e \$_SESSION são consideradas superglobais pois são arrays predefinidos contendo as variáveis do servidor web.

7.2.1. Passagem de parâmetros: valor e referência

Passagem de parâmetros por valor significa que a função receberá cópias dos valores passados no momento em que for chamada. Passagem de parâmetros por referência significa que a função receberá endereços de memória com o conteúdo de variáveis. O exemplo abaixo ilustra as duas situações.

```

<?php
function dobro1($a){
    $a = 2 * $a;
}

```

```
function dobro2(&$b){
    $b = 2 * $b;
}

$valor = 5;
dobro1($valor);
echo "Valor = $valor <br/>";
dobro2($valor);
echo "Valor = $valor";
?>
```

No exemplo acima, a função **dobro1** não altera o valor da variável após sua chamada, pois seu valor é alterado somente dentro da função e não gera retorno. Já a função **dobro2** altera o valor da variável após sua chamada, pois o endereço de memória da variável é acessado e seu valor pode ser acessado por outras funções ou pelo programa principal.

7.3. Funções recursivas

Uma função recursiva é definida em termos de si mesma. Ou seja, é recursiva quando dentro dela está chamada a ela própria. Como exemplo, segue o código que calcula o fatorial de um número utilizando uma função recursiva.

```
<?php
function fatorial($numero){
    if ($numero < 0){
        return -1;
    }
    if ($numero <=1){
        return 1;
    }
    return $numero * fatorial($numero-1);
}

echo "O fatorial de 3 é " . fatorial(3) . " <br/>";
echo "O fatorial de 4 é " . fatorial(4) . " <br/>";
echo "O fatorial de 5 é " . fatorial(5) . " <br/>";
?>
```

No momento em que a função é chamada, um número inteiro `$numero` é passado como parâmetro e dentro da função o valor `$numero-1` é passado como parâmetro novamente. A mesma variável `$numero` acumula os valores das multiplicações sucessivas e em seguida é retornada.

7.4. Exercícios de fixação

Exercício 1. Faça um programa em PHP que receba três valores (obrigatoriamente maiores que zero), representando as medidas dos três lados de um triângulo. Elabore funções (sub-rotinas) para:

- Validar se os valores informados são maiores que zero;
- Determinar se esses lados informados forma um triângulo (sabe-se que, para ser triângulo, a medida de um lado qualquer deve ser inferior à soma das medidas dos outros dois lados);
- Determinar e retornar o tipo de triângulo (equilátero, isósceles ou escaleno), caso as medidas formem um triângulo.

* Todas as mensagens devem ser mostradas no programa principal (fora que qualquer função).

Exercício 2. Faça um programa em PHP para analisar as temperaturas médias de cada mês do ano. Assim o programa deve conter as seguintes funções para:

- Receber a temperatura média de cada mês do ano e armazene-as em um vetor (array);
- Receber um mês em número e retornar o mês por extenso: 0 – janeiro; 1 – fevereiro; 2 – março; ...);
- Calcular e retornar a maior temperatura do ano e em qual mês ocorreu;
- Calcular e retornar a menor temperatura do ano e em qual mês ocorreu;
- Calcular e retornar a média anual de temperaturas.

* Todas as mensagens devem ser mostradas no programa principal (fora que qualquer função).

Capítulo 8 - Includes em PHP

8.1. Definição

As includes permitem reutilizar uma ou mais funções ou arquivos em diversas páginas de um site. Em geral, as includes são usadas em situações em que uma alteração deve ser realizada em todas as páginas do site, sem ter que alterar cada página individualmente. Ao utilizar uma include, as alterações são feitas em apenas um arquivo, que então será acessado por todas as páginas do site.

Um exemplo de uso de include é a criação de menu contendo links para as seções de um site. Suponha que o site tenha vinte páginas e todas elas devem exibir o menu ao lado esquerdo. Caso seja necessário adicionar ou remover um link do menu, seria muito trabalhoso alterar manualmente cada uma das vinte páginas. Para resolver esse problema usando include, primeiramente, deve-se criar um arquivo contendo o código html que define o menu:

```
<p> <strong> Cursos </strong><br/> </p>
<p> <a href="ads.php">Análise e Desenvolvimento de Sistemas</a> </p>
<p> <a href="si.php">Sistemas de Informação</a> </p>
<p> <a href="cc.php">Ciência da Computação</a> </p>
```

Ao salvar esse arquivo com o nome "menu.inc", pode-se utilizar a função `include` para incluir um arquivo dentro de outro, utilizando a sintaxe:

```
<?php include "menu.inc"; ?>
```

Assim, em todas as vinte páginas do site deve haver a chamada da função `include` no local onde deve aparecer o menu. Caso seja necessário alterar o menu, basta modificar o conteúdo html do arquivo "menu.inc", que as alterações serão acessadas por todas as páginas do site. Para adicionar o curso de Engenharia da Computação no menu de cursos, basta acrescentar a seguinte linha no arquivo "menu.inc":

```
<p> <a href="enc.php">Engenharia de Computação</a> </p>
```

Da mesma forma, outros includes podem ser usados para criar cabeçalhos e rodapés de um site. Para isso, devem ser criados arquivos "cabecalho.inc" e "rodape.inc" e depois se deve colocar em cada página do site, no topo e no rodapé, as chamadas include para os arquivos criados:

```
<?php include "cabecalho.inc"; ?>
. . .
<?php include "rodape.inc"; ?>
```

Todas as páginas que utilizam includes devem possuir extensão *.php*, pois a função `include` é um comando da linguagem php.

8.2. Includes e funções

Dentro das includes também é possível definir funções, que estarão disponíveis para todas as páginas php que chamarem a include. Com isso, as funções podem ser reutilizadas em várias páginas, evitando retrabalho e facilitando a manutenção do código. Por exemplo, suponha que seja necessário exibir a data atual em várias páginas de um site. Para reutilizar o código, é possível definir uma função chamada `exibirDataAtual()` dentro do arquivo `"dataAtual.inc"`:

```
<?php
function exibirDataAtual() {
    $meses = array("Janeiro", "Fevereiro", "Março", "Abril", "Maio", "Junho",
        "Julho", "Agosto", "Setembro", "Outubro", "Novembro", "Dezembro");
    $dia = date("d", time());
    $mes = date("m", time());
    $ano = date("Y", time());
    echo $dia . " de " . $meses[$mes-1] . " de " . $ano;
}
?>
```

Nas páginas php em que seja necessário exibir a data atual, basta colocar a chamada `include` para o arquivo `"dataAtual.inc"` e utilizar a função `exibirDataAtual()`:

```
<?php
include "dataAtual.inc";
exibirDataAtual();
?>
```

8.3. Juntando tudo

Considere um site para locação de carros, contendo páginas para cadastro de clientes, cadastro de carros e cadastro de locações realizadas. Todas as páginas devem conter o mesmo cabeçalho e um menu com as funcionalidades do site. Assim, esses elementos podem

ser definidos em arquivos separados e depois incluídos em todas as páginas com o uso de includes. O código html do arquivo "cabecalho.inc" é o seguinte:

```
<header>
  <center>
    
    <h2>IFSP-Car - Sua Locadora de Carros</h2>
  </center>
  <hr>
</header>
```

O código html do arquivo "menu.inc" é o seguinte:

```
<nav style="float: left; max-width: 160px; padding: 1em; background-color:
LightGray;">
  <ul>
    <li><a href="index.php">Home</a></li>
    <li><a href="form_cadastro_cliente.php">Cadastro de Clientes</a></li>
    <li><a href="form_cadastro_carro.php">Cadastro de Carros</a></li>
    <li><a href="form_locacao_carros.php">Cadastro de Locações</a></li>
  </ul>
</nav>
```

Desta forma, todas as páginas do site deverão conter includes para os arquivos de cabeçalho e menu. O código da página inicial do site ("index.php") é o seguinte:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>Locadora de Carros - Página Inicial</title>
  </head>
  <body>
    <div class="container">
      <?php
        include "cabecalho.inc";
        include "menu.inc";
```

```
        ?>
        <center><h2>Seja bem-vindo(a)!</h2></center>
        <br/>
        <br/>
        <br/>
    </div>
</body>
</html>
```

Note que todas as páginas que utilizam includes devem ter extensão .php.

O código da página de cadastro de clientes (arquivo "form_cadastro_cliente.php") é o seguinte:

```
<?php
    session_start();
?>
<!DOCTYPE html>
<html lang="pt-BR">
    <head>
        <meta charset="utf-8">
        <title>IFSP-Car - Cadastro de Clientes</title>
    </head>
    <body>
        <div class="container">
            <?php
                include "funcoes.inc";
                include "cabecalho.inc";
                include "menu.inc";
                if(empty($_POST)){
                    include "form_cliente.inc";
                }else{
                    ler_dados_cliente();
                }
            ?>
        </div>
    </body>
</html>
```

No arquivo `"form_cadastro_cliente.php"`, também se deve incluir os arquivos para cabeçalho e menu. Se o array `$_POST` for vazio, ou seja, o conteúdo do formulário ainda não foi submetido via método POST, é realizado um include para o arquivo `"form_cliente.inc"`, cujo código html é o seguinte:

```
<article>
  <form action="form_cadastro_cliente.php" method="post">
    <fieldset>
      <legend>Cadastro de cliente</legend>
      <p>
        <label>Nome:</label>
        <input type="text" name="nome" size="30"/>
        <label>E-mail</label>
        <input type="text" name="email" size="30"/>
      </p>
      <p>
        <label>Endereço:</label>
        <input type="text" name="endereco" size="30"/>
        <label>Telefone:</label>
        <input type="text" name="telefone" />
      </p>
      <input type="submit" value="Enviar"/>
    </fieldset>
  </form>
  <br/>
  <br/>
  <br/>
</article>
```

O arquivo `"form_cliente.inc"` contém o formulário html de cadastro de cliente, com os campos nome, email, endereço e telefone. Quando o usuário insere os dados e clica no botão `"Enviar"`, o arquivo `"form_cadastro_cliente.php"` fará o tratamento da requisição enviada via método POST. Desta forma, o array `$_POST` não será mais vazio, pois estará preenchido com os dados dos campos do formulário. Na sequência, será executada a função `ler_dados_cliente()`, definida no arquivo `"funcoes.inc"`:

```

<?php
function ler_dados_cliente(){
    if(!isset($_SESSION["contadorCliente"])){
        // registra as variáveis na sessão
        $_SESSION["nomesCliente"][0] = $_POST["nome"];
        $_SESSION["emailCliente"][0] = $_POST["email"];
        $_SESSION["enderecoCliente"][0] = $_POST["endereco"];
        $_SESSION["telefoneCliente"][0] = $_POST["telefone"];
        $_SESSION["contadorCliente"] = 0;
    }else{
        $_SESSION["contadorCliente"]++;
        $contador = $_SESSION["contadorCliente"];
        $_SESSION["nomesCliente"][$contador] = $_POST["nome"];
        $_SESSION["emailCliente"][$contador] = $_POST["email"];
        $_SESSION["enderecoCliente"][$contador] = $_POST["endereco"];
        $_SESSION["telefoneCliente"][$contador] = $_POST["telefone"];
    }
    echo "<article><br/><br/><div class=\"center\">
        <h2>Cliente cadastrado com sucesso.</h2></div>
        <br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/>
        <br/><br/><br/><br/></article>";
}
?>

```

A função `ler_dados_cliente()` adiciona os dados recebidos do array `$_POST` no array superglobal `$_SESSION`, para armazenar os clientes cadastrados durante uma sessão. O vetor superglobal `$_SESSION` também armazena um contador ("`contadorCliente`") que guarda o índice do último cliente cadastrado na sessão. Se esse contador ainda não foi definido, é feito o primeiro cadastro dos dados de cliente na sessão e o contador é inicializado com o valor 0.

Assim, nas próximas execuções dessa função, o valor do contador já estará definido e então será incrementado e utilizado para adicionar os próximos dados no vetor superglobal `$_SESSION`. Por fim, a função exibe uma mensagem dizendo que o cliente foi cadastrado com sucesso.

8.4. Exercícios de Fixação

Exercício 1. Com base no site de locadora de carros definido na Seção 8.3., faça o programa php para o cadastro de carros ("`form_cadastro_carro.php`" presente no menu), de forma similar ao cadastro de clientes. Devem ser utilizados includes para o código html do cabeçalho, do menu e para o arquivo "`funcoes.inc`", que deve conter uma função `ler_dados_carro()` para cadastrar os dados dos carros na sessão. Os dados de cadastro dos carros são: marca, modelo, ano, placa, valor da diária em reais e estado (valor 0 para disponível e valor 1 para locado).

Exercício 2. A partir do exercício anterior, faça um programa em PHP para o cadastro de locações ("`form_locacao_carros.php`" presente no menu), de forma similar ao cadastro de clientes e de carros. Devem ser utilizados includes para o código html do cabeçalho, do menu e para o arquivo "`funcoes.inc`", que deve conter uma função `ler_dados_locacao()` para cadastrar os dados das locações na sessão. Os dados de cadastro das locações são: data da locação, um combobox para selecionar um cliente dentre os clientes já cadastrados, um combobox para selecionar um carro dentre os carros já cadastrados e disponíveis, e um campo para inserir a quantidade de diárias.

Capítulo 9 - Manipulação de arquivos

Uma das maneiras de armazenar dados com o objetivo de recuperá-los posteriormente é trabalhando com o sistema de arquivos do sistema operacional. O php permite abrir, fechar, ler, escrever e realizar outras funções sobre um arquivo de formato conhecido. Para tanto, não é necessária a instalação de um SGBD (Sistema Gerenciador de Banco de Dados) no servidor.

Nestes exemplos, os dados são gravados em arquivos no formato texto. Vale lembrar que a manipulação desses arquivos é recomendada somente quando o volume de dados é pequeno, evitando a utilização de um SGBD.

9.1. Funções de manipulação de arquivos

As principais funções de manipulação de arquivos são: abertura, fechamento, leitura, gravação e escrita.

9.1.1. Abertura – fopen()

O primeiro passo para se ler ou gravar um arquivo é abri-lo, usando a função `fopen()`, que possui a seguinte sintaxe:

```
fopen(string nome_arquivo, string modo_acesso)
```

O parâmetro `nome_arquivo` pode referenciar um arquivo local ou um arquivo em um computador remoto. Se iniciar com `http://`, será aberta uma conexão HTTP. Senão, o arquivo será aberto na própria máquina. Se a abertura do arquivo falhar, a função retorna `FALSE`. O segundo parâmetro se refere ao modo de acesso ao arquivo referenciado, a saber:

Modo	Descrição
'r'	Abre somente para leitura, colocando o ponteiro no início do arquivo.
'r+'	Abre para leitura e escrita, colocando o ponteiro no início do arquivo.
'w'	Abre somente para escrita, colocando o ponteiro no início do arquivo, deixando-o com tamanho zero. Se o arquivo não existir, tentará criá-lo.

'w+'	Abre para leitura e escrita, colocando o ponteiro no início do arquivo, deixando-o com tamanho zero. Se o arquivo não existir, tentará criá-lo.
'a'	Abre somente para escrita, colocando o ponteiro no final do arquivo. Se o arquivo não existir, tentará criá-lo.
'a+'	Abre para leitura e escrita, colocando o ponteiro no final do arquivo. Se o arquivo não existir, tentará criá-lo.
'x'	Cria e abre o arquivo somente para escrita, colocando o ponteiro no início do arquivo. Se o arquivo já existir, retorna FALSE e gera um erro do tipo E_WARNING. Se o arquivo não existir, tenta criá-lo.
'x+'	Cria e abre o arquivo para leitura e escrita, colocando o ponteiro no início do arquivo. Se o arquivo já existir, retorna FALSE e gera um erro de nível E_WARNING. Se o arquivo não existir, tenta criá-lo.

É possível acrescentar a letra 'b' como último caractere do parâmetro. Isto é útil somente em sistemas que diferenciam entre arquivos binários e texto (por exemplo Windows). Se não necessário, será ignorado. Boas práticas de programação levam a incluir o modo 'b' de forma a tornar seus scripts mais portáteis. Seguem alguns exemplos:

```
<?php
$arquivo1 = fopen ("/home/usuario/comunicado.txt", "r");
$arquivo2 = fopen ("/home/usuario/aviso.txt", "wb");
$arquivo3 = fopen ("/home/usuario/aviso.txt", "a+");
$arquivo4 = fopen ("http://www.dominio.com.br", "r");
?>
```

Na plataforma Windows, utilize uma segunda barra invertida (escape) nos caminhos de arquivos, ou a barra normal:

```
<?php
$novoarq = fopen ("c:\\dados\\infome.txt", "r");
?>
```

9.1.2. Fechamento – fclose()

Para fechar um arquivo, utiliza-se a função `fclose`, que possui a seguinte sintaxe:

```
bool fclose($arquivo)
```

A função retorna `true` se o arquivo foi fechado com sucesso e retorna `false` se houver alguma falha. O parâmetro é o nome da variável para qual foi atribuído a referência do arquivo aberto. Exemplo:

```
<?php
    $arquivo1 = fopen ("teste.txt", "r");
    ...
    fclose($arquivo1);
?>
```

9.1.3. Leitura – fread()

Com o arquivo aberto, utiliza-se o comando `fread` para obter seus dados, conforme a seguinte sintaxe:

```
string fread(arquivo, int tamanho)
```

Essa função percorre o arquivo e lê o número de bytes especificado no parâmetro `tamanho`. A leitura termina quando o número de bytes especificado é lido ou o fim de arquivo é alcançado. Exemplo:

```
<?php
    $arquivo = fopen("teste.txt", "r");
    $conteudo = fread($arquivo, 40);
    echo $conteudo;
    fclose($arquivo);
?>
```

O exemplo acima lê os primeiros 40 bytes do arquivo “teste.txt” armazenados na variável \$conteúdo. Em seguida o valor obtido é exibido na tela e o arquivo é fechado.

9.1.4. Leitura linha a linha – fgets()

É possível realizar a leitura de um arquivo obtendo uma linha de cada vez. Uma das maneiras de se obter uma linha de um arquivo é utilizando a função fgets:

```
string fgets(nome_arquivo, int tamanho_da_linha)
```

A leitura é feita seguindo a quantidade de bytes especificado em tamanho_da_linha ou quando a linha terminar(\n). O valor padrão do tamanho da linha é 1024 bytes. O exemplo abaixo ilustra a função:

```
<?php
$arquivo = fopen("teste.txt", "r");
$linha1 = fgets($arquivo, 200);
echo "<br />".$linha1;
$linha2 = fgets($arquivo, 200);
echo "<br />".$linha2;
fclose($arquivo);
?>
```

9.1.5. Escrita – fwrite()

Para gravar dados em um arquivo, utiliza-se o comando fwrite, depois de ter aberto o arquivo. A sintaxe é apresentada a seguir:

```
int fwrite(nome_arquivo, string conteúdo)
```

Essa função escreve o conteúdo especificado na variável nome_arquivo. O exemplo abaixo ilustra a função:

```
<?php
$conteudo = "Texto a ser gravado no arquivo";
$arquivo = fopen("teste.txt", "w");
```

```
fwrite($arquivo,$conteudo);

echo $conteudo;

fclose($arquivo);
?>
```

Note que o código acima tenta abrir o arquivo "teste.txt" no modo de escrita, sobrescrevendo seu conteúdo e gravando o texto contido na variável \$conteudo. Se o arquivo não existir, tentará criá-lo. Em seguida o conteúdo do arquivo é exibido na tela e fechado.

9.2. Exemplo completo

Segue um exemplo contendo as funções vistas até agora.

```
<?php
//Abrindo um arquivo:
$arquivo = fopen("dados.txt", "w");

//Escrevendo no arquivo:
fwrite($arquivo,'Texto a se escrito ');
fwrite($arquivo,'no arquivo. ');

//Fechando arquivo:
fclose($arquivo);

//Lendo um arquivo:
$arquivo = fopen("dados.txt", "r");
$conteudo = fread($arquivo, 2000);

//Fechando arquivo:
fclose($arquivo);

//Exibindo o conteúdo do arquivo:
echo $conteudo;
?>
```


No exemplo acima, o arquivo "`dados.txt`" é aberto com atributo "w" (aberto para escrita) e em seguida são escritas duas linhas de texto. Em seguida o mesmo arquivo é aberto, com atributo "r" (aberto para leitura) e seu conteúdo é exibido na tela. É importante ressaltar que é preciso ter permissão de acesso aos arquivos, seja na pasta local ou no servidor web.

9.3. Exercícios de fixação

Exercício 1. Crie uma página web, com uso da linguagem PHP, que contém formulário (<form>) para que o usuário configure a apresentação visual da página de boas-vindas, gerando um arquivo CSS para:

- Aplicar uma cor de fundo ao corpo da página.
- Aplicar um tamanho de fonte, uma cor e um alinhamento ao texto (direita, esquerda, centralizado) dos parágrafos.
- Aplicar um alinhamento e uma cor de texto ao texto dos <h1> .

Exercício 2. Crie uma página web, com uso da linguagem PHP, que contém formulário para entrada de dados de livros (título, autor, ano de publicação, número de páginas e editora). Os dados inseridos no formulário deverão ser salvos em um arquivo texto. Crie também uma página para apresentar os dados dos livros armazenados no arquivo.

Exercício 3. Crie uma página web que contém formulário para entrada de dados de alunos (nome, nota da primeira prova, nota da segunda prova). Os dados inseridos no formulário deverão ser salvos em um arquivo texto. Crie também uma página para apresentar os dados dos alunos armazenados no arquivo, a média de notas de cada aluno e a média de notas da turma.

Exercício 4. Sistema de Vendas de Produtos

Faça um programa em PHP que atua como um sistema de vendas de produtos, que deve gerenciar basicamente as vendas de produtos de um supermercado. Assim, crie (utilize funções, arquivos ".inc" – include, e arquivos texto):

- Uma página inicial (index.php) com um cabeçalho (nome do supermercado), um menu de opções e um rodapé com o contato da supermercado (endereço, telefone, e-mail). Uma página de cadastro de clientes (form_cadastro_cliente.php) para receber os dados de um cliente (nome, endereço, telefone, e-mail e CPF) e armazene-os em **um arquivo texto ("clientes.txt")**. Após armazenar os dados, o programa deve apresentar uma mensagem de sucesso.

- Uma página de cadastro de produtos (form_cadastro_produto.php) para receber os dados de um produto (código, descrição e preço) e armazená-los em **um arquivo texto (“produtos.txt”)**. Após armazenar os dados, o programa deve apresentar uma mensagem de sucesso.
- Uma página de venda de produtos (form_venda_produtos.php) para receber os dados de uma venda (data da venda, cliente – previamente cadastrado, produto – previamente cadastrado, e quantidade vendida. Com base no preço do produto e quantidade vendida, o programa deve calcular o valor total da venda. Os dados da venda devem ser armazenados em **um arquivo texto (“vendas.txt”)**. Após armazenar os dados, o programa deve apresentar uma mensagem de sucesso.
- Uma página de listagem de clientes (lista_clientes.php) para gerar uma tabela com os dados de todos os clientes cadastrados.
Uma página de listagem de produtos (lista_produtos.php) para gerar uma tabela com os dados de todos os produtos cadastrados.
- Uma página de listagem de vendas (lista_vendas.php) para gerar uma tabela com os dados de todas as vendas realizadas.

Capítulo 10 - Envio de e-mails com PHP

O envio de e-mails com PHP é um recurso utilizado com frequência pelos desenvolvedores de páginas web. Existem diversas situações em que o envio de e-mails é de grande utilidade. Por exemplo, quando um usuário preenche um formulário de cadastramento de um site e escolhe um *username* (nome de usuário) e uma senha, é comum que o sistema lhe envie automaticamente um e-mail confirmando seu cadastro e também os dados que ele forneceu por meio do formulário. Assim, o usuário pode guardar essas informações e recuperá-las no caso de esquecer suas credenciais.

Outro exemplo comum é o de uma loja online, em que cada pedido de compra é armazenado no banco de dados. Para facilitar a identificação dos pedidos, uma alternativa é fazer com a loja receba um e-mail informando que houve uma compra. Esse e-mail poderia conter, por exemplo, nome e endereço do cliente e os produtos comprados. Além disso, seria interessante o cliente também receber um e-mail informando que seu pedido foi recebido, com outras instruções que ele deve seguir para receber os produtos comprados.

Para garantir que os usuários informem e-mails corretos e válidos no momento do cadastro, alguns desenvolvedores utilizam o PHP para enviar um e-mail para o endereço informado, enviando nesse e-mail uma senha especial. Com essa senha, o usuário deve finalizar o processo de cadastramento.

Outra situação, que será implementada como exemplo aqui, é o envio de sugestões ou comentários pelos usuários de um site. Muitos sites apresentam um formulário para o usuário emitir opiniões. Quando o usuário termina e submete os dados, o programa PHP os recebe e envia dois e-mails: um para o próprio site, contendo os dados do formulário, e outro para o usuário, agradecendo e confirmando o recebimento de sua opinião.

10.1. Função *mail*

A função `mail()` é responsável pelo envio de e-mails por meio de programas PHP. Veja um exemplo simples:

```
mail("nome@dominio.com.br", "Bem-vindo", "Olá. Seja bem-vindo!");
```

Essa função envia automaticamente um e-mail para o endereço especificado no primeiro parâmetro (destinatário), com o título do e-mail especificado no segundo parâmetro (assunto) e com o texto do terceiro parâmetro (mensagem).

Pode-se também enviar um e-mail para múltiplos destinatários. Para isso basta separar os endereços pelo caractere `;` (ponto e vírgula):

```
mail("nome@dominio.com.br; nome2@dominio.com.br", "Bem-vindo", "Olá. Seja bem-vindo!");
```

Entretanto, a função `mail` não funciona diretamente no servidor localhost. A função `mail()` carrega as configurações do servidor SMTP a partir do arquivo `php.ini`, onde não é suportado passar informações como usuário e senha, temos um problema porque a maioria dos servidores SMTP (Gmail, Hotmail, Yahoo e etc.) necessitam de autenticação.

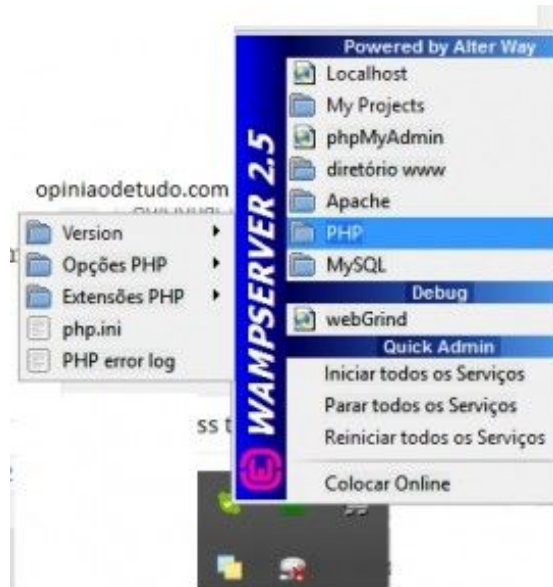
Quando estamos trabalhando em servidores de hospedagem, os mesmos já possuem um servidor SMTP instalado e com os dados configurados no `php.ini`, por isso na maioria das hospedagens a função `mail()` funciona tranquilamente.

Quando estamos trabalhando localmente não temos um servidor SMTP instalado em nossas máquinas. Pode-se tentar configurar o `php.ini` para usar um servidor externo como gmail, por exemplo. Porém, seria necessário passar o usuário e a senha, mas o `php.ini` não tem essa opção de configuração e nesse caso não iria funcionar.

10.2. Configurando o sendmail no WAMP Server

Uma alternativa ao uso do servidor externo, é configurar o sendmail no WAMP Server. Assim, siga os passos a seguir.

- 1) Faça o download: <http://www.glob.com.au/sendmail/sendmail.zip>
- 2) Depois que terminar o download, crie uma pasta chamada `sendmail` dentro da pasta do WAMP e descompacte o arquivo `sendmail.zip` na pasta criada no WAMP, por padrão fica em `"c:\wamp\sendmail"` ou `"c:\wamp64\sendmail"`.
- 3) Após finalizar o processo anterior, procure o arquivo `sendmail.ini` e abra-o para editá-lo utilizando o bloco de notas. Aqui vamos inserir os dados do servidor do Gmail.
- 4) Procure no arquivo pela linha **`smtp_server=mail.mydomain.com`** e substitua por **`smtp_server=smtp.gmail.com`**
- 5) Depois, procure pela linha **`smtp_port=25`** e substitua por **`smtp_port=587`**, que é a porta de saída de email do Gmail.
- 6) Onde estiver **`smtp_ssl=auto`**, altere para **`smtp_ssl=tls`**
- 7) Depois procure as informações de login e insira seus dados do gmail
 - `auth_username=usuario@gmail.com` (**inserir o seu usuário do Gmail**)
 - `auth_password=senha` (**inserir sua senha do Gmail**)
- 8) Para concluir a configuração, procure pelo ícone do WAMP na Barra de Tarefas e selecione o arquivo de configuração do php `"php.ini"`, como mostra a imagem abaixo:



9) No arquivo de configuração do PHP, procure pela seguinte informação:

- `;sendmail_path =`

10) Nessa linha, você deverá informar o caminho do sendmail que você descompactou, se estiver no diretório padrão ficará assim:

- **`sendmail_path="c:\wamp\SendEmail\sendmail.exe -t"`**

11) Não se esqueça de tirar o ponto e vírgula, que serve para deixar a linha comentada.

Após efetuar todos os procedimentos citados, não se esqueça de reiniciar os serviços do WAMP e, em seguida, seus scripts de email já devem funcionar.

10.3. Juntando tudo

Para testar o envio de e-mail, temos um programa que recebe sugestões enviadas por visitantes de um site e as envia por e-mail ao webmaster. Esse programa é ativado por um formulário HTML, no qual o usuário vai digitar seu nome, seu e-mail e sua sugestão.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Envio de e-mail com PHP</title>
  </head>
  <body>
    <form action="email.php" method="post">
      <p>
        <label>Nome:</label>
        <input type="text" size="30" name="nome" />
      </p>
      <p>
        <label>E-mail:</label>
        <input type="text" size="30" name="email" />
      </p>
      <p>
        <label>Sugestão:</label>
        <input type="text" size="100" name="sugestao" />
      </p>
      <input type="submit" value="Enviar" />
    </form>
  </body>
</html>
```

Quando o usuário clicar no botão Enviar, os dados são enviados ao programa PHP abaixo, que obtém os dados que o usuário digitou e os coloca em forma de mensagem, para enviá-los utilizando a função `mail`.

```

<?php

$nome = $_POST["nome"];
$email = $_POST["email"];
$sugestao = $_POST["sugestao"];

$mensagem = "Sugestão enviada por um visitante:<br /><br />";
$mensagem .= "Nome: $nome<br />";
$mensagem .= "E-mail: $email<br />";
$mensagem .= "Sugestão: $sugestao";

$de = "seusite@seusite.com.br";
$para = "webmaster@seusite.com.br";
$reply = "seusite@seusite.com.br";
$headers = "From: $de\r\n" .
           "Reply-To: $de\r\n" .
           "X-Mailer: PHP/" . phpversion() . "\r\n";
$headers .= "MIME-Version: 1.0\r\n";
$headers .= "Content-Type: text/html; charset=utf-8\r\n";

if(mail($para, "Sugestão", $mensagem, $headers)){
    echo "<h2>Obrigado por enviar sua sugestão.</h2>";
}else{
    echo "<h2>ERRO ao enviar os dados.</h2>";
}

?>

```

Observe que, neste exemplo, a função mail recebe quatro parâmetros. Os três primeiros parâmetros já foram vistos anteriormente. O quarto parâmetro contém os cabeçalhos (headers), exigidos pela maioria dos provedores.

10.4. Exercícios de fixação

Exercício 1. Altere a página de formulário de venda de produtos (`form_venda_produtos.php`), do Exercício 4 do Capítulo 9, para enviar também um e-mail para o gerente de vendas do supermercado com os dados da venda.

Exercício 2. Altere o programa PHP do Exercício 3 do Capítulo 9, para que este envie um e-mail ao professor da disciplina com os dados dos alunos, a média de notas de cada aluno e a média de notas da turma.

Referências Bibliográficas

LIMA JR, L. E. de **Um pouco da história da linguagem de programação PHP**. Disponível em: <<http://www.naninho.blog.br/web/php/um-pouco-da-historia-da-linguagem-de-programacao-php.html>>. Acesso em: 26 mar. 2018.

MILANI, A. **Construindo Aplicações Web com PHP e MySQL**. 2 ed. São Paulo: Novatec, 2010.

NIEDERAUER, J. **Desenvolvendo Websites com PHP**. 3. ed. São Paulo: Novatec, 2017.

WATANABE, W. M. **Comunicação cliente/servidor - HTTP**. Disponível em: <<https://pt.slideshare.net/watinha1/apresentacao-17018075>>. Acesso em: 20 mar. 2018.