

## 1.0 PHP

### Objetivos

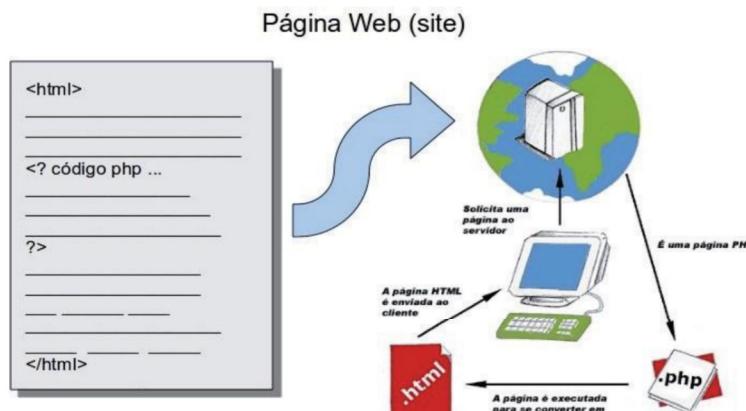
*Apresentar basicamente alguns tipos de dados a ser implementados e adotados; Mostrar sua importância; Diferenciar variável e constante; Apresentar comandos e processo de entrada.*

### 1.1 O que é PHP

PHP significa: Hypertext Preprocessor. O produto foi originalmente chamado de “Personal Home Page Tools”, mas como se expandiu em escopo um nome novo e mais apropriado foi escolhido por votação da comunidade. Você pode utilizar qualquer extensão que desejar para designar um arquivo PHP, mas os recomendados foram .php. O PHP está atualmente na versão 5.4.0, chamado de PHP5 ou, simplesmente de PHP, mas os seus desenvolvedores estão trabalhando para lançamento da versão 6, que causa uma preocupação para os programadores do mundo todo, uma vez que, algumas funcionalidades antigas deixam de funcionar quando passaram da versão 4 para a 5.



PHP é uma linguagem de criação de scripts embutida em HTML no servidor. Os produtos patenteados nesse nicho do mercado são as Active Server Pages(ASP) da Microsoft, o Coldfusion da Allaire e as Java Server Pages da antiga Sun que foi comprada pela Oracle. PHP é, às vezes, chamado de “o ASP de código-fonte aberto” porque sua funcionalidade é tão semelhante ao produto/conceito, ou o que quer que seja, da Microsoft.



Exploraremos a criação de script no servidor, mais profundamente, nos próximos capítulos, mas, no momento, você pode pensar no PHP como uma coleção de “supertags” de HTML que permitem adicionar funções do servidor às suas páginas da Web. Por exemplo, você pode

utilizar PHP para montar instantaneamente uma complexa página da Web ou desencadear um programa que automaticamente execute o débito no cartão de crédito quando um cliente realizar uma compra. Observe uma representação de como PHP e HTML se comportam:

O PHP tem pouca relação com layout, eventos ou qualquer coisa relacionada à aparência de uma página da Web. De fato, a maior parte do que o PHP realiza é invisível para o usuário final. Alguém visualizando uma página de PHP não será capaz de dizer que não foi escrita em HTML, porque o resultado final do PHP é HTML.

O PHP é um módulo oficial do servidor http Apache, o líder do mercado de servidores Web livres que constitui aproximadamente 55 por cento da World Wide Web. Isso significa que o mecanismo de script do PHP pode ser construído no próprio servidor Web, tornando a manipulação de dados mais rápida. Assim como o servidor Apache, o PHP é compatível com várias plataformas, o que significa que ele executa em seu formato original em várias versões do UNIX e do Windows. Todos os projetos da Apache Software Foundation – incluindo o PHP – são software de código-fonte aberto.

#### 1.1.1 Um pouco da História do PHP.

Rasmus Lerdorf, engenheiro de software, membro da equipe Apache é o criador e a força motriz original por trás do PHP. A primeira parte do PHP foi desenvolvida para utilização pessoal no final de 1994. Tratava-se de um wrapper de PerlCGI que o auxiliava a monitorar as pessoas que acessavam o seu site pessoal. No ano seguinte, ele montou um pacote chamado de Personal Home Page Tools (também conhecido como PHP Construction Kit) em resposta à demanda de usuários que por acaso ou por relatos falados depararam-se com o seu trabalho. A versão 2 foi logo lançada sob o título de PHP/FI e incluía o Form Interpreter, uma ferramenta para analisar sintaticamente consultas de SQL.



Em meados de 1997, o PHP estava sendo utilizado mundialmente em aproximadamente 50.000 sites. Obviamente estava se tornando muito grande para uma única pessoa administrar, mesmo para alguém concentrado e cheio de energia como Rasmus. Agora uma pequena equipe central de desenvolvimento mantinha o projeto sobre o modelo de “junta benevolente” do código-fonte aberto, com contribuições de desenvolvedores e usuários em todo o mundo. Zeev Suraski e Andi Gutmans, dois programadores israelenses que desenvolveram os analisadores de sintaxe PHP3 e PHP4, também generalizaram e estenderam seus

trabalhos sob a rubrica de Zend.com (Zeev, Andi, Zend).

O quarto trimestre de 1998 iniciou um período de crescimento explosivo para o PHP, quando todas as tecnologias de código-fonte aberto ganharam uma publicidade intensa. Em outubro de 1998, de acordo com a melhor suposição, mais de 100.000 domínios únicos utilizavam PHP de alguma maneira. Um ano depois, o PHP quebrou a marca de um milhão de domínios.

## 1.2 Instalação do Servidor PHP

As requisições são feita de forma cliente servidor, onde um determinado cliente faz uma requisição a um servidor, ele por sua vez recebe a requisição e faz todo o processo em diferentes camadas, retornando somente o que o cliente (browser) solicitou.

Existe uma aplicação chamada de LAMP (Linux, Apache, MYSQL, PHP) voltada para sistemas operacionais Linux, como também WAMP (Windows, Apache, MYSQL, PHP) voltada para sistemas operacionais Windows. Eles proporcionam uma simples configuração desses recursos e também muito indicado para ambiente de estudo PHP. Porém não iremos trabalhar com a instalação do LAMP. Pretendemos futuramente trabalhar com **frameWorks**, então o mais indicado será realizar a instalação e configuração manualmente. Iremos

ff

explicar um passo a passo de todo o processo de instalação e configuração desses recursos no tópico seguinte.

### 1.2.1 Instalação Apache.

O apache é um dos principais aplicativos para o funcionamento de programas web feito em PHP, ele é um dos responsáveis por compartilhar o nosso site dinâmico para outras outras máquinas, existe duas grandes versões, o apache 2.x e o Apache 1.3, que apesar de antigo ainda é muito utilizado em servidores. O Apache 2 trouxe muitas vantagens, sobretudo do ponto de vista do desempenho além de oferecer novos módulos e mais opções de segurança. Mas sua adoção foi retardada nos primeiros anos por um detalhe muito simples: o fato de ele ser incompatível com os módulos compilados para o Apache 1.3. Como os módulos são a alma do servidor web, muitos administradores ficavam amarrados ao Apache 1.3 devido à falta de disponibilidade de alguns módulos específicos para o Apache 2.

Iremos trabalhar com o Apache2 em sua versão para Linux, o procedimento de instalação é simples pois precisamos apenas de uma conexão com a internet e alguns comando, outra forma de instalação e baixando o mesmo no site: <http://httpd.apache.org/> e instalando de forma manual.

Antes de começar qualquer instalação, certifique-se que não existe nenhum pacote corrompido no sistema ou pacotes como LAMPP, XAMPP Instalados..

Para instalar o Apache precisamos abrir o terminal e atualizar o nosso repositório do Linux:

```
#apt-get update
```

Com os pacotes básicos atualizados podemos instalar o apache:

```
#apt-get install apache2
```

Espere até a conclusão de toda instalação. Por padrão o apache automaticamente inicializa, podendo ser utilizado após isso. Digite no <http://127.0.0.1/> ou <http://localhost/> no browser,

deverá aparecer a seguinte tela:



Isto mostra que o apache está funcionando corretamente. Trata-se de um texto dentro de um arquivo HTML que vem como padrão dentro da pasta "/var/www/" , essa é a pasta principal do apache, onde trabalharemos com os arquivos PHP.

filf

to

O apache pode ser configurado de forma que podemos acessar mais de uma página com ip's ou portas diferentes, ou seja podemos ter www1,www2 ... em uma mesma máquina servidora fazendo acesso por endereços como <http://localhost:8060> ou <http://localhost:82>, entre outros. Mas trabalharemos apenas com a configuração padrão do apache, uma vez que existe muito conteúdo a respeito desse assunto na internet. Para obter mais informações consulte a documentação do site oficial do apache: <http://httpd.apache.org/docs/2.2/>

### 1.2.2 Instalação Php5.

Para o funcionamento dos programas em PHP precisamos de algumas dependências, onde funcionalidades, funções, classes e suporte a XML então embutidas. Por exemplo, se você quiser utilizar orientação a objeto ou simplesmente fazer uma conexão com o banco de dados, é necessário também a instalação do PHP5, onde algumas bibliotecas serão instaladas para dar suporte aos programas em PHP.

Após instalar o apache, faremos a instalação do PHP5 e algumas bibliotecas básicas com o seguinte comando:

```
#apt-get install php5 libapache2-mod-php5 php5-gd curl php5-curl php5-xmlrpc  
php5-ldap php5-odbc
```

Atualizaremos o Ubuntu com alguns pacotes de serviço para servidor com o seguinte comando :

```
#apt-get install openssh-server unattended-upgrades
```

E por fim iremos reiniciar o serviço Apache:

```
#/etc/init.d/apache2 restart
```

Após a conclusão da instalação, podemos testar criando um arquivo dentro da pasta “var/www”.

Entre na pasta principal:

```
$ cd /var/www
```

Renomeie o arquivo “index.html” para outro nome:

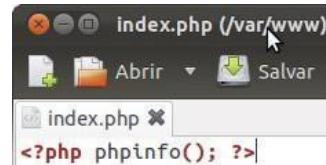
```
#mv  
index.html nome.html #mv
```

Qualquer arquivo com o nome “index.html” dentro da pasta **www** será o primeiro arquivo a ser executado. Lembrando que pode existir somente um arquivo “index.html” na pasta. Após renomear

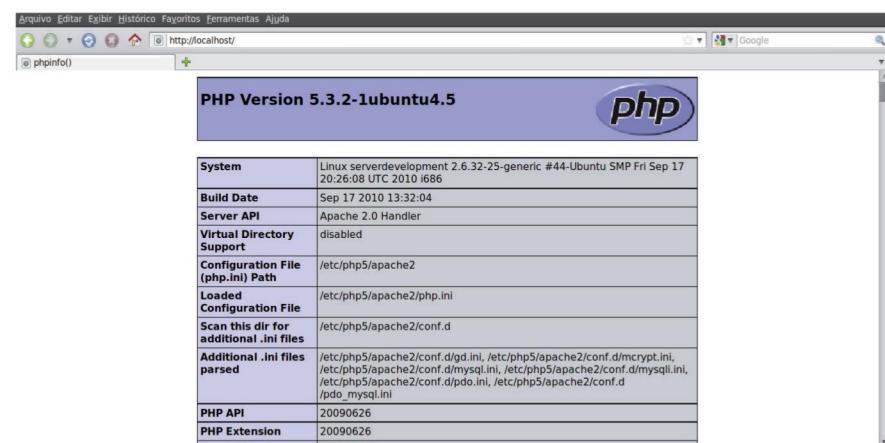
```
#gedit index.php
```

criaremos um novo arquivo.

Após executar esse comando, digite o seguinte código no editor de texto gedit:



Nesse momento estamos criando uma aplicação PHP que chama a função `phpinfo()` que tem como finalidade mostra informações sobre configuração do PHP. Lembre-se que toda operação em PHP estará dentro da Tag especial chamada: “<?php ?>”



Abra o browser e digite <http://localhost/>, se tudo estiver correto, observaremos seguinte tela:

### 1.3 Características de um programa PHP.

Assim como qualquer linguagem de programação, o PHP tem algumas características importantes, ao criamos um arquivo PHP podemos usar a seguinte extensão:

**.php \***

< Arquivo PHP contendo um programa.

**.class.php <**

adiante).

Arquivo PHP contendo uma classe(veremos o conceito de classe mais

**.ini.php**

< Arquivo PHP a ser incluído, pode incluir constantes ou configurações.

Outras extensões podem ser encontradas principalmente em programas antigos:

**\*.php3 <** Arquivo PHP contendo um programa PHP versão 3.

**\*.php4 <** Arquivo PHP contendo um programa PHP versão 4.

**\*.phtml <** Arquivo PHP contendo um programa PHP e HTML na mesma página.

### 1.3.1 Imprimindo algo no navegador WEB via PHP.

Usaremos comandos utilizados para gerar uma saída em tela (output). Se o programa PHP for executado via servidor de páginas web (Apache ou IIS), a saída será exibida na própria página HTML gerada no Browser (navegador), assim será mostrado de acordo com o conteúdo existente na saída, por exemplo, se tivermos o seguinte: "<h2> Hello Word!"

<h2>", será mostrado no navegador apenas a mensagem Hello Word! em um tamanho maior, pois trata-se de um código HTML dentro de comandos PHP.

Podemos então usar os seguintes comandos para gerar comandos de saída: echo e print.

#### echo

É um comando que imprime uma ou mais variáveis ou textos, onde os mesmos são

colocados em aspas simples '' ou duplas “ ”. Sintaxe: echo 'a','b','c';

resultado:

abc

#### print

É uma função que imprime um *texto* na tela. Exemplo:

resultado:

apostila de PHP

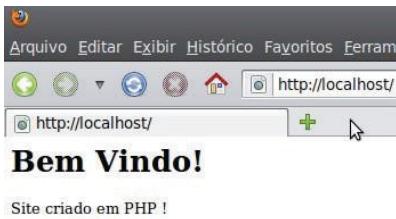
```
print( 'apostila de PHP' );
```

Observe um exemplo com o uso de echo e print:

#### Código:

```
1 □ <?php
2
3 echo '<h1>Bem Vindo!</h1>';
4 print('Site criado em PHP !');
5
6 ?>
```

#### Saída no Browser:



### 1.3.2 Comentários PHP

Usamos os comentários para nos ajudar a lembrar de partes do código ou para orientar outros programadores a algoritmo que está sendo escrito.

Para comentar uma única linha nos código PHP podemos utilizar tanto o separador “//” como também “#”, observe o exemplo abaixo:

```
1 □ <?php
2
3 // echo "a";
4 // imprime a;
5 # echo b;
6 # comentários!!
7
8 ?>
```

Lembrando que os comentários são trechos de código que não são executados, onde eles servem somente para quem tem acesso aos códigos-fonte ou está ligado diretamente a programação desse código. Podemos também comentar muitas linhas, isso serve quando queremos que boa parte do código não execute ou simplesmente colocar um comentário mais extenso. Tudo que ficar entre: “//” e “#” será um comentário, independente do número de linhas, observe o exemplo abaixo:

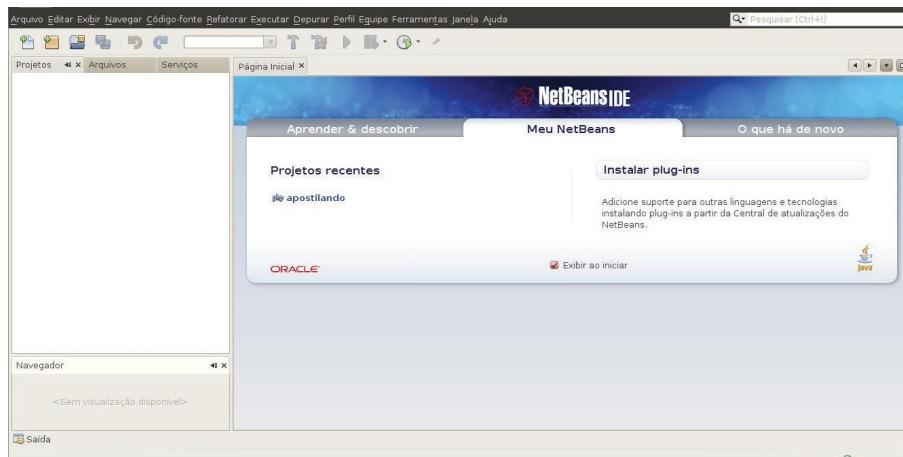
\* \* \*

```
1 □ <?php
2
3 /* parte menos importante do código!
4 *
5 * echo '<h1>Bem vindo so Site</h1>'; ou
6 * if($a==2){
7 * echo "logout";
8 * }
9 */
10
11 ?>
```

```
1 □ <?php
2
3 /* código principal!
4 * não inicializa as
5 * variáveis
6 */
7
8 ?>
```

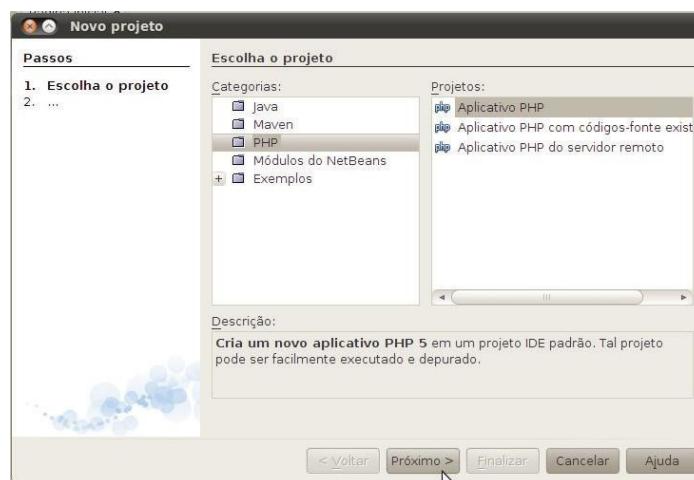
## 2.0 Iniciando um projeto com netbeans.

Para facilitar nosso trabalho, escolhemos usar a IDE Netbeans. Procure no site <http://netbeans.org/>. Uma versão com o Netbeans instalada ou baixe plugin PHP em : Ferramentas > Plugins.



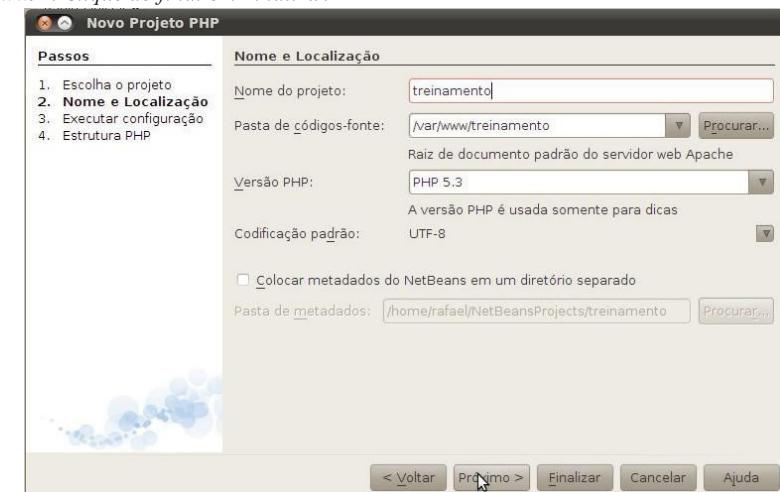
Com a IDE instalada devidamente, encontraremos o seguinte layout:

Para criar um novo projeto, vamos em: Arquivo > Novo Projeto .

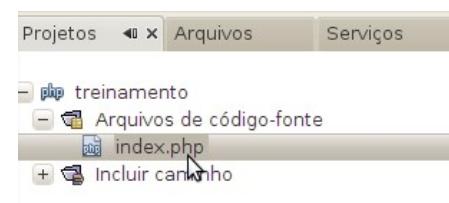


Escolhendo um projeto em PHP. Depois clicamos em Próximo:

Alteraremos o nome do Projeto para treinamento e certificar-se que ele estará dentro da pasta "/var/www/". Clique ao final em Finalizar.



Obs: Caso não tenha a pasta "/var/www/", veja como está a instalação do servidor APACHE na sua máquina.



Na área de Projetos a esquerda, irá aparecer um novo projeto em PHP, através dele, estudaremos todo o conteúdo criando o novo projeto do site.

No palco Principal, o Netbeans já faz por nós toda a criação básica das estrutura HTML incluindo a TAG PHP na página. Para testar que a nossa aplicação, vamos dar um velho e conhecido : "Olá Mundo" em PHP. Para nos certificarmos que está tudo funcionando.

```

1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
5      <title>Página Inicial</title>
6    </head>
7    <body>
8      <?php
9        echo "Olá Mundo";
10       ?>
11     </body>
12   </html>
13

```

Já o resultado no seu navegador: Acessando: "<http://localhost/treinamento/>".

Caso algo deu errado, veja novamente as configurações do APACHE ou PHP.

## 2.1 Tipos Primitivos

Todo o trabalho realizado por um computador é baseado na manipulação das informações contidas em sua memória. A grosso modo de expressar-se, estas informações podem ser classificadas em dois tipos:

As instruções, que comandam o funcionamento da máquina e determinam a maneira como devem ser tratados os dados. As instruções são específicas para cada modelo de computador, pois são funções do tipo particular de processador utilizado em sua implementação. Os dados propriamente ditos, que correspondem a porção das informações a serem processadas pelo computador.

### 2.1.1 Tipos de Primitivos de Dados

Quaisquer dados a ser tratado na construção de um algoritmo deve pertencer a algum tipo, que irá determinar o domínio de seu conteúdo. Os tipos mais comuns de dados são conhecidos como tipos primitivos de dados, são eles: inteiro, real, numérico, caractere e lógico. A classificação que será apresentada não se aplica a nenhuma linguagem de programação específica, pelo contrário, ela sintetiza os padrões utilizados na maioria das linguagens.

- **Inteiro:** Todo e qualquer dado numérico que pertença ao conjunto de números inteiros relativos (negativo, nulo ou positivo). Exemplos: {...-4,-3,-2,-1,0,1,2,3,4,...}.
- **Real:** Todo e qualquer dado numérico que pertença ao conjunto de números reais (negativo, nulo ou positivo). Exemplos: {15.34; 123.08 ; 0.005 -12.0 ; 510.20}.
- **Numérico:** Trata-se de todo e qualquer número que pertença ao conjunto dos inteiros ou reais, também abrange números binários, octal e hexadecimal.
- **Caracteres:** são letras isoladas. Exemplo : {'a', 'd', 'A', 'h'}.
- **Dados literais:** Conhecido também como **Conjunto de caracteres** ou String, é todo e qualquer dado composto por um conjunto de caracteres alfanuméricos (números, letras e caracteres especiais). Exemplos: {"Aluno Aprovado", "10% de multa", "Confirma a exclusão ??", "S", "99-3000-2", "email", "123nm", "fd54fd"}.
- **Lógico:** A existência deste tipo de dado é, de certo modo, um reflexo da maneira como

os computadores funcionam. Muitas vezes, estes tipos de dados são chamados de **booleanos**, devido à significativa contribuição de **Boole** a área da lógica matemática. A todo e qualquer dado que só pode assumir duas situações dados biesáveis, algo como por exemplo {0/ 1, verdadeiro/falso, sim/não, true/false }.

Veremos a seguir como esses valores serão armazenados e como o PHP irá interpretá-los.

### EXERCÍCIOS PROPOSTOS

Como podemos definir PHP e qual a sua relação com HTML?

Descreva um exemplo estrutural relacionando tag's HTML e delimitações PHP.

Quais aplicativos básicos podemos instalar para criarmos um servidor de páginas em PHP?

Em relação ao apache, qual a sua principal finalidade?

Quais as principais extensões de arquivos PHP e diga a diferença entre elas?

Crie dois exemplos de código PHP usando seus delimitadores. EP02.7: Qual as finalidades de usar comentários dentro de um código-fonte? EP02.8: Cite os principais comandos de saída usados em PHP.

Qual a diferença entre echo e print?

Observe o seguinte código e diga qual o item correto:

```
<?php
//página web.
```

```
echo '<h1>Olá, essa é sua primeira página!<h1>';
```

```
echo '<h2>Responda!</h2>' print 'O que é PHP?';
```

```
print 'Qual sua finalidade?'; echo '<h4>Resposta: item a!</h4>';
```

```
?>
```

I - Esse código pode ser interpretado pelo apache normalmente. II - As tag's HTML são interpretadas pelo servidor.

III - Esse código possui erros.

IV - As tag's são interpretadas pelo navegador do cliente.

a) I, II, IV estão corretas.

b) somente a I está correta.

c) III, IV estão corretas.

d) somente IV está correta.

e) I e IV estão corretas.

**Prático:**

Crie um arquivo PHP dentro da pasta www com o nome index.php, após isso pegue o código anterior e adicione a esse arquivo, defina quais textos são visualizadas em seu navegador. Caso exista erros, faça uma correção.

Crie dois arquivos diferentes, um com nome index.php, outro com o nome teste.php, após isso inclua o arquivo teste.php dentro do arquivo index.php, ambos arquivos deverá ter no seu código impresso mensagens diferentes utilize o comando print.

## 3.0 Atribuições em PHP

### Objetivos

*Mostrar a definição de variáveis e os seus tipos; Mostrar como pode-se atribuir um valor a ela, seu uso e aplicação em PHP; Mostrar a definição de constantes e sua importância.*

### 3.1 Variáveis

Variáveis são identificadores criados para guardar valores por determinado tempo. Em PHP elas são declaradas e inicializadas, porém são armazenadas na memória RAM do servidor web. Esse é um dos motivos pelo qual os servidores precisam de grande quantidades de memória.

Imagine um servidor com mais de 20 mil acessos simultâneos ao mesmo tempo, onde cada usuário está acessando a mesma página feita em PHP. São criadas neste processo variáveis diferentes para cada usuário, logo, isso faz com que muitos processos sejam gerados e processados pelo servidor.

A tipagem em PHP é dinâmica, ou seja, as variáveis não precisam ser obrigatoriamente inicializadas após a declaração. Uma variável é inicializada no momento em que é feita a primeira atribuição. O tipo da variável será definido de acordo com o valor atribuído. Esse é um fator importante em PHP, pois uma mesma variável pode ser de um mesmo tipo ou não, e pode assumir no decorrer do código um ou mais valores de tipos diferentes.

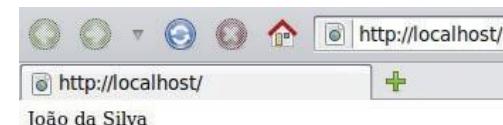
Para criar uma variável em PHP, precisamos atribuir-lhe um nome de identificação, sempre precedido pelo caractere cifrão (\$). Observe um exemplo:

```

1 <?php
2
3 $nome = "João";
4 $sobrenome = "da Silva";
5 echo "$nome $sobrenome";
6
7 ?>

```

Para imprimirmos as duas variáveis usamos aspas duplas no comando “echo”, no exemplo anterior temos a seguinte saída:



**Atenção!**

Obs.: Podem acontecer erros na exibição das mensagens por conta das codificações de acentuação. Caso isso aconteça, mude a codificação do seu navegador ou utilize as metas de codificação.

**Algumas dicas Importantes:**

- Nomes de variáveis devem ser significativa e transmitir a ideia de seu conteúdo dentro do contexto no qual está inserido.
- Utilize preferencialmente palavras em minúsculo (separadas pelo caracter “\_”) ou somente as primeiras letras em maiúsculo quando da ocorrência de mais palavras.

Exemplo:

```
<?PHP
$codigo_cliente; //exemplo de variável
$CodigoCliente; //exemplo de variável
?>
```

- Nunca inicie a nomenclatura de variáveis com números. Ex: \$1nota;
- Nunca utilize espaço em branco no meio do identificados da variável. Ex: \$nome um;
- Nunca utilize caracteres especiais( ! @ # ^ & \* / | [ ] { } ) na nomenclatura das variáveis.
- Evite criar variáveis com mais de 15 caracteres em virtude da clareza do código-fonte.

Com exceção de nomes de classes e funções, o PHP é case sensitive, ou seja, é sensível a letras maiúsculas e minúsculas. Tome cuidado ao declarar variáveis. Por exemplo a variável

\$codigo é tratada de forma totalmente diferente da variável \$Codigo.

Em alguns casos, precisamos ter em nosso código-fonte nomes de variáveis que podem mudar de acordo com determinada situação. Neste caso, não só o conteúdo da variável é mutável, mas também variante (variable variables). Sempre que utilizarmos dois sinais de cifrão (\$) precedendo o nome de uma variável, o PHP irá referenciar uma variável representada pelo conteúdo da primeira. Nesse exemplo, utilizamos esse recurso quando declaramos a variável \$nome (conteúdo de \$variável) contendo 'maria'.

```
1 | <?php
2 | //define o nome da variável
3 | $variavel = 'nome';
4 | //cria variável identificada pelo conteúdo de $variavel
5 | $variavel = 'maria';
6 | // exibe variável $nome na tela
7 | echo "$nome";
8 | ?>
9 |
```

*Resultado = maria.*

Quando uma variável é atribuída a outra, sempre é criada uma nova área de armazenamento na memória. Veja neste exemplo que, apesar de \$b receber o mesmo conteúdo de \$a, após qualquer modificação em \$b, \$a continua com o mesmo valor,veja:

```
1 | <?php
2 | $a = 5;
3 | $b = $a;
4 | $b = 10;
5 | echo $a; // resultado = 5
6 | echo $b; // resultado = 10
7 | ?>
8 |
```

Para criar referência entre variáveis, ou seja, duas variáveis apontando para a mesma região da memória, a atribuição deve ser precedida pelo operador &. Assim, qualquer alteração em qualquer uma das variáveis reflete na outra,veja:

```
1 | <?php
2 | $a = 5;
3 | $b = &$a;
4 | $b = 10;
5 | echo $a; // resultado = 10
6 | echo $b; // resultado = 10
7 | ?>
```

No exemplo anterior percebemos que tanto \$a como \$b apontam para a mesma referência na memória, dessa forma se atribuirmos um novo valor em \$a ou em \$b, esse valor será gravado no mesmo endereço, fazendo com que, ambas variáveis possam resgatar o mesmo valor.

### 3.2 Tipos de Variáveis

Algumas linguagens de programação tem suas variáveis fortemente “tipadas”, diferentemente disso o PHP tem uma grande flexibilidade na hora de operar com variáveis. De fato, quando definimos uma variável dando-lhe um valor, o computador atribui-lhe um tipo. Isso permite que o programador não se preocupe muito na definição de tipos de variáveis, uma vez que isso é feita de forma automática. Porém deve ter cuidado com as atribuições de valores, evitando erros na hora de iniciar uma variável em PHP.

## TIPO BOOLEANO.

Um booleano expressa um valor lógico que pode ser verdadeiro ou falso. Para especificar um valor booleano, utilize a palavra-chave TRUE para verdadeiro ou FALSE para falso. No exemplo a seguir, declaramos uma variável booleana \$exibir\_nome, cujo conteúdo é TRUE para verdadeiro. Em seguida, testamos o conteúdo dessa variável para verificar se ela é realmente verdadeira imprimindo na tela caso seja. Usaremos a estrutura IF, uma estrutura de controle que veremos com mais detalhes no capítulo 4, para testar a variável. Observe:

```

1 | <?php
2 | //Atribuição de valor.
3 | $exibir_nome = true;
4 | //Imprime na 1 na tela.
5 | print($exibir_nome);
6 | ?>
```

*Resultado = 1 (esse valor representa verdadeiro ou true).*

Também podemos atribuir outros valores booleanos para representação de valores falso em operação booleanas.

- Inteiro 0 ;
- Ponto flutuante 0.0 ;
- Uma String vazia “ ” ou “0” ;
- Um array vazio ;
- Um objeto sem elementos ;
- Tipo NULL .

## TIPO INTEIRO

São os números que pertencem ao conjunto dos números inteiros, abrangendo valores negativos e positivos, trata-se de valores decimais.

```

1 | <?php
2 | $a = 125;
3 | $b = -1456;
4 | $c = 7;
5 | ?>
```

## TIPO PONTO FLUTUANTE:

Os números de ponto flutuante (floats e doubles) são números com casas decimais, onde a vírgula é substituída por um ponto. Exemplo:

```

1 | <?php
2 | $a = 10.0;
3 | $a = 10.3e4;
4 | $a = 1.234;
5 | ?>
```

## TIPO NUMÉRICO

Números podem ser especificados em notação decimal (base 10), hexadecimal (base 16) ou octal (base 8), opcionalmente precedido de sinal (- ou +), esse tipo abrange todos os valores abaixo:

```

1 | <?php
2 |
3 | $a = 1234; // número decimal
4 | $b = -123; // um número negativo
5 | $c = 0123; // numero octal (equivalente a 83 em decimal)
6 | $d = 0x1a; // número hexadecimal (equivalente a 26 em decimal)
7 | $e = 1.234; // ponto flutuante
8 | $f = 4e23; // notação científica
9 |
10| ?>
```

Não entraremos em detalhes em relação a conversão desses valores, porém que fique claro a ideia de que uma variável numérica pode assumir diferentes tipos de valores.

## TIPO STRING

Uma string é uma cadeia de caracteres alfanuméricos. Para declará-las podemos utilizar aspas

```

1 | <?php
2 | $a = 'Atribuição do tipo string'; //aspas simples
3 | $a = "Atribuição do tipo string"; //aspas dupla
4 | ?>
```

simples ( ' ) ou aspas duplas ( " " ). Exemplo:

Observe na tabela abaixo o que podemos também inserir em uma String:

Sintaxe	Significado
\n	Nova linha
\r	Retorno de carro (semelhante a \n)
\t	Tabulação horizontal
\\\	A própria barra (\ )
\\$	O símbolo \$
'	Aspa simples
"	Aspa dupla

Observe o exemplo:

```

1 | <?php
2 | $a = '\\tipo de dado: \'inteiro\'';
3 | echo $a;
4 | ?>
```

*Resultado: \tipo de dado: 'inteiro'*

## TIPO ARRAY

Array é uma lista de valores armazenados na memória, os quais podem ser de tipos diferentes (números, strings, objetos) e podem ser acessados a qualquer momento, pois cada valor é relacionado a uma chave. Um array também pode crescer dinamicamente com a adição de novos itens. Veja no capítulo 6 como manipular esse tipo de estrutura.

## TIPO OBJETO

Um objeto é uma entidade com um determinado comportamento definido por seus métodos (ações) e propriedade (dados). Para criar um objeto deve-se utilizar o operador new. Para mais informações sobre orientação a objeto, consulte o site <http://php.net> e pesquise sobre object.

## TIPO RECURSO

Recurso (resource) é uma variável especial que mantém uma referência de recursos externos. Recursos são criados e utilizado por funções especiais, como uma conexão ao banco de dados. Um exemplo é a função mysql\_connect(), que ao conectar-se ao banco de dados, retorna um variável de referência do tipo recurso. Exemplo:

*Resource mysql\_connect(...) Outro exemplo: mysql\_fetch\_row(...)*

## TIPO NULL

Quando atribuímos um valor do tipo null (nulo) a uma variável estamos determinando que a mesma não possui valor, e que seu único valor é nulo. Exemplo:

\$abc = null;

### 3.3 Constantes

#### 3.3.1 Constantes pré-definidas

O PHP possui algumas constantes pré-definidas, indicando a versão do PHP, o Sistema Operacional do servidor, o arquivo em execução e diversas outras informações. Para ter acesso a todas as constantes pré-definidas, pode-se utilizar a função phpinfo(), que exibe uma tabela contendo todas as constantes pré-definidas, assim como configurações da máquina, sistema operacional, servidor HTTP e versão do PHP instalada, como foi feito em exemplos anteriores.

#### 3.3.2 Definindo constantes

Para definir constantes utiliza-se a função define. Uma vez definido, o valor de uma constante não poderá mais ser alterado. Uma constante só pode conter valores escalares, ou seja, não pode conter nem um array nem um objeto. A assinatura da função define é a seguinte:

```

1 | <?php
2 | define("NOME_DA_CONSTANTE","valor inalterável");
3 | echo NOME_DA_CONSTANTE;
4 | ?>

```

define("NOME\_DA\_CONSTANTE","valor inalterável"); Exemplo:

*Resultado: valor inalterável*

O nome de uma constante tem as mesmas regras de qualquer identificador no PHP. Um nome de constante válida começa com uma letra ou sublinhado, seguido por qualquer número de letras, números ou sublinhados. Você pode definir uma constante utilizando-se da função define(). Quando uma constante é definida, ela não pode ser mais modificada ou anulada.

Estas são as diferenças entre constantes e variáveis:

- Constantes podem ser definidas e acessadas de qualquer lugar sem que as regras de escopo de variáveis sejam aplicadas;
- Constantes só podem conter valores escalares;
- Constantes não podem ter um sinal de cifrão (\$) antes delas;
- Constantes só podem ser definidas utilizando a função define( ), e não por simples assimilação;
- Constantes não podem ser redefinidas ou eliminadas depois que elas são criadas.

### 3.4 Conversão de variável

PHP utiliza checagem de tipos dinâmica, ou seja, uma variável pode conter valores de diferentes tipos em diferentes momentos da execução do script. Por este motivo não é necessário declarar o tipo de uma variável para usá-la. O interpretador PHP decidirá qual o tipo daquela variável, verificando o conteúdo em tempo de execução.

Ainda assim, é permitido converter os valores de um tipo para outro desejado, utilizando o typecasting ou a função settype (ver adiante).

Assim podemos definir novos valores para terminadas variáveis:

typecasting	Descrição
(int),(integer)	Converte em inteiro.
(real),(float),(double)	Converte em ponto flutuante.
(string)	Converte em string.
(object)	Converte em objeto.

```

1 | <?php
2 | $a =(int)(48.56 + 145 + 15 - 0.21);
3 | echo $a;
4 |
5 |

```

**Convertendo de ponto flutuante para inteiro.**

Resultado: 208

**Convertendo de String para Object.**

```

1 | <?php
2 | $a =(object) ("Bem vindo ao site!");
3 | print_r($a);
4 |
5 |

```

Resultado:

stdClass Object ( [scalar] =>Bem vindo ao site! )

**Convertendo de inteiro para ponto flutuante.**

```

1 | <?php
2 | $a = (double)(542);
3 | print($a);
4 |
5 |

```

Resultado: 542

O resultado poderia ser 542.0, mas lembrando que o interpretador do PHP faz outra conversão ao notar que o numero 542.0 tem a mesma atribuição de 542. O resultado seria o mesmo se tentarmos atribuir \$a = 542.0.

## **EXERCÍCIOS PROPOSTOS**

Qual a principal finalidade de uma variável?

O que significa tipagem automática.

Cite algumas dicas importantes na nomenclatura de variáveis:

Das variáveis abaixo, quais possuem nomenclaturas válidas.

\$a__b;	\$a_1_;	\$_início;
\$_@nome;	\$val_!;	\$_-nome;
\$a_ _;	\$_#valor;	\$palavra;
\$tele#;	\$123;	\$_=;
\$VALOR_MAIOR;	\$_____;	\$all;

Resposta:

---

Crie dez variáveis atribuindo valores diversos, logo após use o comando **echo** pra imprimir na tela do browser, exemplo:

```

<?php
$nome = "Maria Cavalcante"; echo $nome;
...
?>

```

Quais os tipos de variáveis que podemos citar em PHP.

Como podemos distinguir um tipo de variável de outro, uma vez que a tipagem é feita de forma automática em PHP.

Faça a ligação com os seguintes tipos:

- |                        |                      |
|------------------------|----------------------|
| 1 - \$var = -10;       | ( ) ponto flutuante. |
| 2 - \$var = "palavra"; | ( ) tipo null.       |
| 3 - \$var = 10.22;     | ( ) tipo objeto.     |
| 4 - \$var = true;      | ( ) String.          |
| 5 - \$var = null;      | ( ) numérico.        |
| 6 - \$var = new abc;   | ( ) booleano.        |

Qual a principal finalidade de um constante e como elas são definidas em PHP. **EP03.10:** Em que momentos precisamos converter uma variável de um tipo em outro. **EP03.11:** Quais os typecasting usados em PHP.

Crie uma constante com o comando define e imprima com o comando print();

Crie conversões e imprima na tela com o comando print() com as seguintes variável.

\$var1 = "paralelepípedo", \$var2 = 15.20, \$var3 = 10.

- Converte a variável \$var1 em objeto.
- Converte a variável \$var3 em ponto flutuante.
- Converte a variável \$var2 em inteiro.

## 4.0 Operadores em PHP

### Objetivos

Demostrar os tipos e quais os operadores; Falar do conceito de atribuição e concatenação de strings;  
Exemplificar os operadores, sua importância e funcionamento.

Os operadores tem seu papel importante dentro de qualquer linguagem de programação.

É através deles que podemos realizar diversos operações dentro de um programa, seja ela de atribuição, aritmética, relacional, lógico, dentre outros. Em PHP não é diferente, os operadores são utilizados constantemente, porém existem algumas regras que veremos mais adiante.

### 4.1 Operadores de strings

São operadores utilizados para unir o conteúdo de uma string a outra, com isso podemos dizer que há dois operadores de string. O primeiro é o operador de concatenação ('.') que já utilizamos em exemplos anteriores, ele retorna a concatenação dos seus argumentos direito e esquerdo. O segundo é o operador de atribuição de concatenação ('.='), que acrescenta o argumento do lado direito no argumento do lado esquerdo.

Observe o exemplo abaixo:

```

1 | <?php
2 |
3 | $a = " Bem vindo ";
4 | $b = " ao site ";
5 | $c = " de pesquisa ";
6 | echo $a.$b.$c; //imprime: Bem vindo ao site de pesquisa
7 | $d = $a.$b.$c.=" tecnológica";
8 | echo $d; //imprime: Bem vindo ao site de pesquisa tecnológica
9 | $a .= " Aluno ";
10 | echo $a; //imprime: Bem vindo Aluno
11 | ?>

```

Nesse exemplo pode-se observar a declaração da variável **\$d**, logo após temos uma inicialização e atribuição de concatenação em uma mesma linha, isso é possível em PHP, deixando o código mais otimizado porém menos legível.

## 4.2 Operadores Matemáticos

### 4.2.1 Aritméticos

Chamamos de **operadores aritméticos** o conjunto de símbolos que representa as operações básicas da matemática.

Operações	Operadores	Exemplo	Resposta
Adição	+	3 + 5	8
subtração	-	20 - 5	15
Multiplicação	*	7 * 8	56
Divisão	/	15 / 3	5
Módulo (Resto da divisão)	%	10 % 3	1
Negação (Número Posto)	- (valor)	Se for 15 a atribuição	-15

Veja:

```

1 | <?php
2 | $a = 3;
3 | $b = 4;
4 | echo $a*$b+5-2*3; // 12+5-6 resultado: 11
5 | echo $a*($b+5) -2*3; // 3*9-6 resultado: 21
6 |
7 | ?>

```

Nesse exemplo fizemos algumas operações, porém ao utilizar parênteses, estamos determinando quem executa primeiro, no caso a soma de  $b+5$ .

### 4.2.2 Atribuição

O operador básico de atribuição é "**=**" (igual). Com ele podemos atribuir valores as variáveis como foi visto em exemplos anteriores. Isto quer dizer que o operando da esquerda recebe o valor da expressão da direita (ou seja, "é configurado para"). Mas podemos usar algumas técnicas, observe o exemplo abaixo:

```

1 | <?php
2 |
3 | $a = ($b=4)+5;
4 | echo "a = $a,b = $b";
5 |
6 | ?>

```

Resultado:  $a = 9, b = 4$

Além do operador básico de atribuição, há "operadores combinados" usados para array e string, eles permitem pegar um valor de uma expressão e então usar seu próprio valor para o resultado daquela expressão.

Por exemplo:

```

1 <?php
2
3 $a = 3;
4 $a += 5; // $a recebe 8, então: $a = $a + 5;
5 $b = "Bom ";
6 $b .= "Dia!"; // $b recebe "Bom Dia!", então $b = $b . "Dia!";
7 echo "a = ".$a." , b = ".$b;
8
9 ?>
```

*Resultado: a = 8,b = Bom Dia!*

Observe a expressão: \$a = 3 e logo após \$a+=5. Isto significa a mesma coisa de \$a = \$a + 5, ou, \$a = 3 +5. A ideia pode ser usada para string, como foi feito com a variável \$b, onde \$b = “Bom”, logo após usamos ponto(.) e igual(=) para concatenar os valores, ficando assim: \$b.=“Dia!”. Lembrando que isso significa a mesma coisa que \$b = \$b.“Dia”. Observe mais um exemplo:

```

1 <?php
2
3 $a = "Dia "; #inicia $a com Dia
4 $b = "Bom "; #inicia $b com Bom
5 $b .= $a.= "turma"; #concatena primeiro $a,logo depois $b
6 echo $b; #imprime na tela.
7
8 ?>
```

*Resultado: Bom Dia turma*

Podemos definir uma sequência com duas concatenações, onde \$a = “Dia”.“turma” e logo após temos \$b = “Bom”.“Dia turma”.

Os operadores de atribuição são usados para economizar linhas de código, deixando assim o código mais funcional e otimizado. A tabela abaixo mostra os principais operadores de atribuição:

Operadores	Descrição
=	Atribuição simples.
+=	Soma, depois atribui.
-=	Subtrai, depois atribui.
*=	Multiplica, depois atribui.
/=	Divide, depois atribui.
%=	Modulo(resto) da divisão, depois atribui.
.=	Concatena, depois atribui.

Exemplo:

```

1 <?php
2
3 $a = 43; # inicia $a com 43
4 $b = 62; # inicia $b com 62
5
6 $a += 17; # atribui o valor anterior(43) mais 17 em $a
7 echo $a; # imprime o valor(60) de $a na tela.
8
9 $b -= 12; # atribui o valor anterior(62) menos 12 em $b
10 echo $b; # imprime o valor(50) de $b na tela.
11
12 ?>
```

Observe mais um exemplo aplicando os demais operadores.

```

1 <?php
2 $a = 8; $b = 2; // inicializa duas variáveis
3 # multiplica o valor de $a(8) por 3 e atribui.
4 echo ( $a *= 3 )."<br>" ;
5 # divide o valor de $a(24) por 2 e atribui.
6 echo ( $a /= 2 )."<br>" ;
7 # resto da divisão de $a(12) por 5 e atribui.
8 echo ( $a %= 5 )."<br>" ;
9 // os pontos “.” concatena com "<br>" .
10 ?>
```

*Resultado: 24*

8  
2

Vale ressaltar que a cada echo, o valor de \$a sofre modificações. Isso devido a atribuição feita após a operação. Usamos o operador ponto(.) para concatenar os valores obtidos com <br> código usado em HTML para quebra de linha.

#### 4.2.3 Operadores de decremento e incremento

São operadores usados para atribuir em 1 ou -1 a variável, isso pode ser feito antes ou depois da execução de determinada variável. A tabela abaixo mostra tais operadores:

Operadores	Descrição
++\$a	Pré-incremento. Incrementa \$a em um e, então, retorna \$a.
\$a++	Pós-incremento. Retorna \$a, então, incrementa \$a em um.
-\$a	Pré-decremento. Decrementa \$a em um e, então, retorna \$a.
\$a-	Pós-decremento. Retorna \$a, então, decrementa \$a em um.

Exemplo:

```

1 <?php
2 $a = 1;
3 print(++$a); // incrementa 1 em $a(1), depois imprime(2).
4 print($a++); // imprime $a(2) depois incrementa 1.
5 print($a); // imprime $a(3)
6 print(--$a); // decrementa 1 em $a(3), depois imprime a$(2).
7 print($a--); // imprime $a(2), depois decrementa 1.
8 print($a); // imprime $a(1)
9 ?>
10

```

Nesse exemplo temos uma forma aplicada do uso de decremento e incremento, lembrando que a variável \$a pode ter qualquer nome. Também podemos fazer um comparativo com o Pré-incremento ou incremento-prefixado com operações que já conhecemos, observe:

Operador	Forma extensa.	Forma simplificada
<code>++\$a</code>	<code>\$a = \$a + 1</code>	<code>\$a+=1</code>
<code>--\$a</code>	<code>\$a = \$a - 1</code>	<code>\$a -=1</code>

### 4.3 Operadores relacionais

Os operadores relacionais ou conhecidos também como operadores de comparação, são utilizados para fazer determinadas comparações entre valores ou expressões, resultando sempre um valor booleano verdadeiro ou falso(TRUE ou FALSE). Para utilizarmos esses operadores usamos a seguinte sintaxe:

( valor ou expressão ) + ( comparador ) + ( segundo valor ou expressão )

Observe a tabela abaixo:

Comparadores	Descrição
<code>==</code>	Igual. Resulta em TRUE se as expressões forem iguais.
<code>==</code>	Idêntico. Resulta em TRUE se as iguais e do mesmo tipo de dados.
<code>!= ou &lt;&gt;</code>	Diferente. Resulta verdadeiro se as variáveis foram diferentes.
<code>&lt;</code>	Menor ou menor que. Resulta TRUE se a primeira expressão for menor.
<code>&gt;</code>	Maior ou maior que. Resulta TRUE se a primeira expressão for maior.
<code>&lt;=</code>	Menor ou igual. Resulta TRUE se a primeira expressão for menor ou igual.
<code>&gt;=</code>	Maior ou igual. Resulta TRUE se a primeira expressão for maior ou igual.

Veja um exemplo prático:

`$a <= $b`

Compara se \$a é menor ou igual a \$b, onde, retorna verdadeiro (TRUE), caso contrário retorna falso (FALSE).

Para testarmos essas comparações podemos utilizar o condicional “?:” (ou **ternário**), sua sintaxe é a seguinte:

**(expressão booleana) ? (executa caso verdadeiro) : (executa caso falso);**

Agora podemos ver um exemplo envolvendo as sintaxes e empregabilidade dos comparadores:

```

1 <?php
2 $a = 15;
3 $b = 42;
4 $c = 42.0; #resposta:
5 echo $b == $c ? "verdadeiro" : "falso"; // verdadeiro
6 echo $b === $c ? "verdadeiro" : "falso"; // falso
7 echo $b != $c ? "verdadeiro" : "falso"; // falso
8 echo $a <> $c ? "verdadeiro" : "falso"; // verdadeiro
9 echo $a < $c ? "verdadeiro" : "falso"; // verdadeiro
10 echo $a > $c ? "verdadeiro" : "falso"; // falso
11 echo $a <= $c ? "verdadeiro" : "falso"; // verdadeiro
12 echo $a >= $c ? "verdadeiro" : "falso"; // falso
13 ?>

```

Nesse exemplo declaramos e iniciamos três variáveis. Usamos então o comando **echo** para imprimir o resultado, onde o condicional “?:” foi utilizado. Iniciamos as comparações de \$a, \$b e \$c, caso a comparação individual retorne TRUE, imprime verdadeiro, caso retorne FALSE, imprime falso. Observe que o comparador “==” compara o valor e o tipo, retornando FALSE por \$b se tratar de um tipo inteiro, e \$c um tipo ponto flutuante, já o comparador “==” compara somente os valores onde 45 é igual a 45.0 retornando verdadeiro. Também podemos usar o operador “!==” onde tem a função semelhantemente ao operador “!=”, mas retorna TRUE se os tipos forem diferentes. Se a variável for do tipo booleano, podemos compará-los assim:

`$a == TRUE, $a == FALSE`

### 4.4 Operadores lógicos ou booleanos

São utilizados para avaliar expressões lógicas. Estes operadores servem para avaliar expressões que resultam em valores lógico sendo verdadeiro ou falso:

E	AND
OU	OR
Não	NOT

Esses operadores tem a finalidade de novas proposições lógicas composta a partir de outra proposições lógicas simples. Observe:

Operador	Função
<code>não</code>	negação
<code>e</code>	conjunção

<b>ou</b>	disjunção
-----------	-----------

Com isso podemos construir a Tabela verdade onde trabalhamos com todas as

possibilidades combinatoria entre os valores de diversas variáveis envolvidas, as quais se encontram em apenas duas situações (V e F), e um conjunto de operadores lógicos. Veja o comportamento dessas variáveis:

Operação de Negação:

A	(not) não A
F	V
V	F

Trazendo para o nosso cotidiano:

Considerando que A = “Está chovendo”, sua negação seria : “Não está chovendo”. Considerando que A = “Não está chovendo” sua negação seria : “Está chovendo”

Operação de conjunção:

A	B	A e B
F	F	F
F	V	F
V	F	F
V	V	V

Trazendo para o nosso cotidiano:

Imagine que acontecerá um casamento:

Considerando que A = “Noivo presente” e B = “Noiva presente”.

Sabemos que um casamento só pode se realizar, se os 2 estejam presente.

Operação de disjunção não exclusiva:

A	B	A ou B
F	F	F
F	V	V
V	F	V
V	V	V

Trazendo para o nosso cotidiano:

Imagine que acontecerá uma prova e para realizá-la você precisará da sua Identidade ou título de eleitor no dia da prova.

Considerando que A = “Identidade” e B = “Título de eleitor”.

Sabemos que o candidato precisa de pelo menos 1 dos documentos para realizar a prova.

São chamados de operadores lógicos ou booleanos por se tratar de comparadores de duas ou mais expressões lógicas entre si, fazendo agrupamento de testes condicionais e tem como retorno um resultado booleano.

Na tabela abaixo temos os operadores e suas descrições:

Operador	Descrição
<b>( \$a and \$b )</b>	E : Verdadeiro se tanto \$a quanto \$b forem verdadeiros.
<b>( \$a or \$b )</b>	OU : Verdadeiro se \$a ou \$b forem verdadeiros.

<b>( \$a xor \$b )</b>	XOR : Verdadeiro se \$a ou \$b forem verdadeiro, de forma exclusiva.
<b>( ! \$a )</b>	NOT : Verdadeiro se \$a for falso, usado para inverter o resultado da condição.
<b>( \$a &amp;&amp; \$b )</b>	E : Verdadeiro se tanto \$a quanto \$b forem verdadeiros.
<b>( \$a    \$b )</b>	OU : Verdadeiro se \$a ou \$b forem verdadeiros.



Dicas: or e and tem procedência maior que && ou ||, ou seja, em uma comparação extensa, onde ambos estão aplicados. Eles tem prioridade de executar sua comparação primeiro.

No próximo exemplo usamos os operadores lógicos que tem procedência maior:

```

1  <?php
2   $a = 10 > 2;      // verdadeiro, pois 10 é maior que 2.
3   $b = 12 >= 12;    // verdadeiro, pois 12 é igual a 12.
4   $c = FALSE;       // inicia com FALSE(tipo booleano).
5   echo ($b and $a) ? "sim" : "não" ; //sim, pois $a e $b são verdadeiros.
6   echo ($b or $c) ? "sim" : "não" ; //sim, pois $b é verdadeiro.
7   echo ($b xor $a) ? "sim" : "não" ; //não, pois $b e $a são verdadeiros.
8   ?>
9

```

Em outro exemplo temos os operadores lógicos mais comuns:

```

1 <?php
2 $a = 10 > 2;      // verdadeiro, pois 10 é maior que 2.
3 $b = 12 >= 12;    // verdadeiro, pois 12 é igual a 12.
4 $c = FALSE;        // inicia com FALSE(tipo booleano).
5 echo (!$c) ? "sim" : "não"; //sim, pois $c é falso.
6 echo ($a && $c) ? "sim" : "não"; //não, pois $c é falso.
7 echo ($b || $c) ? "sim" : "não"; //sim, pois $b é verdadeiros.
8 ?>
9

```

```

1 <?php
2 $a = 0;
3 $b = 0;
4 ($a = 2 or $b = 3);
5 echo $a."/".$b;
6 ($a = 5 and $b = 3);
7 echo $a."/".$b;
8 ?>
9

```

Também podemos atribuir valores as variáveis usando os operadores lógicos:

O primeiro **echo** mostra 2 e 0, pois não atribui valor a \$b uma vez que a primeira condição já é satisfatória.

O segundo **echo** mostra 5 e 3, pois tanto a primeira quanto a segunda precisam ser executadas.

#### 4.5 Precedência de Operadores

Agora já conhecemos uma boa quantidade de operadores no PHP, falta agora conhecer a precedência de cada um deles, ou seja, quem é mais importante, qual operador é avaliado primeiro e qual é avaliado em seguida. Observe o seguinte exemplo:

```

1 <?php
2
3 echo 5 + 2 * 6; // resultado: 17
4
5 ?>

```

O resultado será **17**, pois o operador **\*** tem maior precedência em relação ao operador **+**. Primeiro ocorre a multiplicação 2 6, resultando em 12, em seguida a soma de 5 + 12. Caso desejar realizar a operação com o operador **+** para só em seguida realizar a operação com o operador **\***, temos que fazer conforme o exemplo abaixo:

```

1 <?php
2
3 echo (5 + 2) * 6; // resultado: 42
4
5 ?>
6
7

```

Observe que utilizamos os parênteses para determinarmos quem deve ser executado primeiro, assim alterando o resultado para **42**. Os parênteses determina qual bloco de código executa

primeiro, e também serve para isolar determinadas operações. Veja mais um exemplo onde as operações são feitas separadamente. Primeiro executa a soma, em seguida a subtração e só então é executado a multiplicação, imprimindo um resultado final **21**:

Exemplo:

```

1 <?php
2
3 echo (5 + 2) * (6-3); // resultado: 21
4
5 ?>
6

```

A tabela seguinte mostra a precedência dos operadores, da maior precedência para os de menor precedência.

Operador	Descrição
- ! + +	Negativo, negação, incremento e decremento
/ *	Multiplicação, divisão e resto da divisão
+ -	Adição, subtração e concatenação
> < >= <=	Maior que, menor que, maior ou igual e menor ou igual
== != <>	Igual e diferente
&&	E
	OU
= += -=	Operadores de atribuição
= /= %=	
AND	E com menor prioridade
XOR	Ou exclusivo
OR	Ou com 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 dos operadores será utilizada para definir a ordem. Após resolver todas as operações dos parentes, o PHP volta a resolver o que está fora dos parênteses baseando-se na tabela de precedência de operadores. Havendo operadores de mesma prioridade o PHP resolverá a operação da esquerda para direita.

Também podemos trabalhar com precedência de parênteses, fazendo associações com um ou mais operadores, observe o seguinte exemplo:

```

1 <?php
2
3 echo (3 + 2) * (9-3)/(16-((5+2)*2));
4 // resultado: 15
5
6 ?>
7

```

Seguindo a ordem de procedência temos:

(5)  $*(6 / (16 - ((7 * 2))) >>> 5 * 6 / (16 - (14)) >>> 5 * 6 / 2 >>> 30 / 2$

Resultado : 15

Observe que primeiro executa todos os parênteses, e só então temos as procedências das demais operações.

### EXERCÍCIOS PROPOSTOS

Qual a finalidade dos operadores de strings?

Quais os operadores de decremento e incremento? Cite alguns exemplos:

Qual a finalidade do operador aritmético %(modulo)?

Cite os operadores relacionais, mostre alguns exemplos.

Quais operadores lógicos ou booleanos?

Quais os operadores de atribuição?

Qual a sintaxe do uso de ternário e cite um exemplo?

Quais os operadores utilizados e o resultado final do código abaixo:

```
<?php
$a = 10;
$b = 12.5;
$c = $a+$b;
print($a>$b? "verdadeiro" : "falso"); print($c>$b? "verdadeiro" : "falso");
?>
```

Observe o código abaixo e diga quais das operações são executadas primeiro, coloque a resposta em ordem decrescente.

$\$a = 8*5-3+4*f2+19%5f2+1;$

Faça testes com os operadores relacionais substituindo o operando > do código-fonte abaixo.

```
<?php
$var1 = 2.2564;
$var2 = 2.2635;
print($var1 > $var2 ? "sim" : "não");
?>
```

Usando o operador de String “.” para montar a seguinte frase abaixo:

```
<?php
$a = "de";
$b = "é um";
$c = "comunicação";
$c = "a";
$d = "internet";
$e = "meio"; print( .... );
?>
```

Observe o código-fonte abaixo e diga qual o resultado booleano final. Justifique sua resposta.

```
<?
$a = 12.0 < 11.2;
$b = 10*2-3 > 19%3+10;
$c = 10;
print( ($a || $c = 10 && $b) ? "true" : "false");
?>
```