PHP - INTRODUÇÃO

PROFESSOR EDUARDO

INTRODUÇÃO

Qualquer script PHP é construído por uma série de instruções. Uma instrução pode ser uma atribuição, uma chamada de função, um laço de repetição, uma instrução condicional, ou mesmo uma instrução que não faz nada (um comando vazio). Instruções geralmente terminam com um ponto e vírgula. Além disso, as instruções podem ser agrupados em um grupo de comandos através do encapsulamento de um grupo de comandos com chaves.

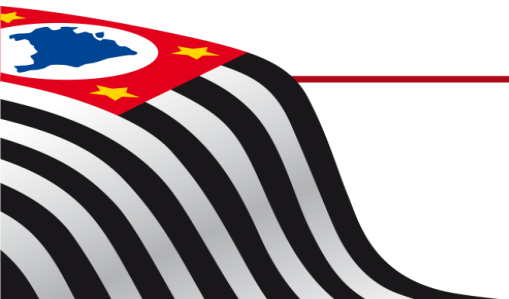
```
<!DOCTYPE html>
<html lang="pt-br">
<head>
<meta charset="utf-8"/>
<title>HTML5 – Estrutura básica</title>
</head>
<body>



</body>
</html>
```

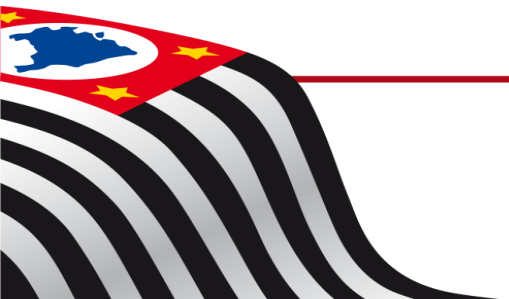
<?php

?>



VARIÁVEIS

As variáveis em PHP (ou em qualquer outra linguagem de programação), são objetos capazes de reter e representar valores, expressões e funções no tempo de execução do script. Cada variável está associada a uma posição de memória do seu computador ou servidor.



VARIÁVEIS

No PHP, utilizamos o sinal de **$** (***Cifrão***), para representar a criação ou utilização de uma variável. Ex:

$variavel = “Minha primeira página em PHP”;

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
<meta charset="utf-8"/>
<title>HTML5 – Estrutura básica</title>
</head>
<body>

<?php
$a="Meu primeiro arquivo com php";

?>

</body>
</html>
```

MOSTRAR A INFORMAÇÃO DA VARIÁVEL

```
$a="Meu primeiro arquivo com php";
echo $var1;
?>
```

CONCATENAR

É um termo usado em computação para designar a operação de unir duas ou mais conteúdo de variáveis.

1° opção

```
$fichacompleta  = $nome." - ".$idade." - ".$salario;
echo $fichacompleta;
```

2° opção:

```
$fichacompleta="$nome - ";
$fichacompleta .="$idade - ";
$fichacompleta .="$salario"
echo $fichacompleta;
```

CONCATENAÇÃO

3° opção

```php
echo "meu nome: {$nome} - minha idade: {$idade} - meu salario: {$salario}";
```

OBSERVAÇÃO

Em PHP é utilizado o sinal de cifrão ($) mais o nome que vai identificar a variável. Porém, o primeiro caractere após o cifrão deve ser uma letra ou um underline (_), **nunca um número ou um traço** (-).

Correto

```php
<?php
$_variavel                = 'Correto';
$_variavel_               = 'Correto';
$_variável_               = 'Correto';
$Variável                 = 'Correto';
$variavel_qualquer        = 'Correto';
$VARIÁVEL_QUALQUER        = 'Correto';
$variavelQualquer         = 'Correto';
$variavel_22_Qualquer_11 = 'Correto';
?>
```

Incorreto

```php
<?php
$1variavel         = 'Incorreto';
$ variavel         = 'Incorreto';
$variavel qualquer = 'Incorreto';
$-variavel         = 'Incorreto';
$variavel-qualquer = 'Incorreto';
?>
```

TIPO DE VARIÁVEIS

No PHP não existe declaração de variáveis, isso porque a linguagem PHP é de **tipagem dinâmica** ou **fracamente tipada**, ele faz a atribuição de variáveis por coerção.

COERÇÃO

Coerção se explica da seguinte forma, se é criada uma variável do tipo **"$nome"** é definido o tipo dela a partir do momento que a variável $nome recebe **"João da Silva"**, por exemplo, **$nome = "João da Silva"** automaticamente será reconhecido como uma variável String, mas nada impede de no meio do programa essa variável nome receber um valor **inteiro**,**booleano** ou de **ponto flutuante**, isso automaticamente ele irá converter a variável para o tipo de dado que ela está recebendo.

TIPOS

```php
<?php

$var1='Meu primeiro arquivo com php';
$nome ='seu nome';
$idade=35;
$salario=1850.30;
$casado=false;

echo $var1;
echo "<br/>";
echo $nome;
echo "<br/>";
echo $idade;
echo "<br/>";
echo $salario;
echo "<br/>";
echo $casado;?>
```

FORÇAR O TIPO DA VARIÁVEL

Para forçar o tipo da variável basta inserir depois do símbolo de igual o tipo da variável. Exemplo:

```
$var = (String) "João da Silva";
```

TIPOS

Inteiro = (Int), (Integer);

Real = (real),(float), (double);

Caractere =(String);

Lógico (?????) -> true=1, false= 0;

OPERAÇÕES ARITMÉTICAS

São funções básicas de somar, subtrair, multiplicar, dividir e etc.

- adição (+)
- subtração (-)
- multiplicação (*)
- divisão (/)
- módulo - resto da divisão - (%)

OPERADORES LÓGICOS

Operador	Nome	Exemplo	Resultado
==	Igual	$a == $b	Verdadeiro se $a for igual a $b
!=	Diferente	$a != $b	Verdadeiro se $a **não** for igual a $b
<>	Diferente	$a <> $b	Verdadeiro se $a **não** for igual a $b
===	Idêntico	$a === $b	Verdadeiro se $a for igual a $b e for do mesmo **tipo**
!==	Não idêntico	$a !== $b	Verdadeiro se $a **não** for igual a $b, ou eles não são do mesmo **tipo**
<	Menor que	$a < $b	Verdadeiro se $a for menor que $b
>	Maior que	$a > $b	Verdadeiro se $a for maior que $b
<=	Menor ou igual	$a <= $b	Verdadeiro se $a for menor ou igual a $b.
>=	Maior ou igual	$a >= $b	Verdadeiro se $a for maior ou igual a $b.

ORDEM DE PRECEDÊNCIA

1. Parênteses ();
2. Multiplicação *;
3. Divisão /;
4. Modulo %;
5. Adição +;
6. Subtração -

FUNÇÕES ARITMÉTICAS

OBS: documentação - https://www.php.net/manual/pt_BR/ref.math.php

abs — Valor absoluto;
acos — Cosseno Inverso (arco cosseno);
acosh — Cosseno Hiperbólico Inverso;
asin — Seno Inverso (arco seno);
asinh — Seno Hiperbólico Inverso;
atan2 — Tangente inversa de duas variáveis;
atan — Tangente Inversa (arco tangente);
atanh — Tangente hiperbólica inversa;
base_convert — Converte um número entre bases arbitrárias;

FUNÇÕES ARITMÉTICAS

bindec — Binário para decimal;
ceil — Arredonda frações para cima;
cos — Coseno;
cosh — Cosseno hiperbólico;
decbin — Decimal para binário;
dechex — Decimal para hexadecimal;
decoct — Decimal para octal;
deg2rad — Converte o número em graus ao equivalente em radianos;

FUNÇÕES ARITMÉTICAS

exp — Calcula o expoente de e;
expm1 — Retorna exp(numero) - 1, computado de forma que é preciso mesmo quando o valor do número é perto de zero;
floor — Arredonda frações para baixo;
fmod — Retorna o resto em ponto flutuante (módulo) da divisão dos argumentos;
getrandmax — Retorna o maior valor aleatório possível;

FUNÇÕES ARITMÉTICAS

hexdec — Hexadecimal para decimal;
hypot — Calcula o tamanho da hipotenusa de um ângulo reto do triângulo Retorna a raiz quadrada de (num1*num1 + num2*num2);
intdiv — Dividir números inteiros;
is_finite — Verifica se um valor é um número finito;
is_infinite — Descrição;
is_nan — Verifica se um valor não é um número;
lcg_value — Gerador congruente linear combinado;

FUNÇÕES ARITMÉTICAS

log10 — Logaritmo Base-10;
log1p — Retorna o log(1 + numero), calculado de forma que o valor do número seja próximo de zero;
log — Logaritmo natural;
max — Localiza o maior valor;
min — Encontra o menor valor;
mt_getrandmax — Retorna o maior valor aletório possível;
mt_rand — Gerador melhorado de números aleatórios;

FUNÇÕES ARITMÉTICAS

mt_srand — Semeia o gerador melhorado de números aleatórios;
octdec — Octal para decimal;
pi — Obtém o valor de pi;
pow — Potência;
rad2deg — Converte o número em radianos para o equivalente em graus;
rand — Gera um inteiro aleatório;

FUNÇÕES ARITMÉTICAS

round — Arredonda um número;
sin — Seno;
sinh — Seno hiperbólico;
sqrt — Raiz quadrada;
srand — Semeia o gerador de números aleatórios;
tan — Tangente;
tanh — Tangente hiperbólica;

TRABALHANDO COM FUNÇÕES DATA E HORA

DATA

Caractere de `format`	Descrição	Exemplo de valores retornados
Dia	---	---
d	Dia do mês, 2 dígitos com zero à esquerda	*01* até *31*
D	Uma representação textual de um dia, três letras	*Mon* até *Sun*
j	Dia do mês sem zero à esquerda	*1* até *31*
l ('L' minúsculo)	A representação textual completa do dia da semana	*Sunday* até *Saturday*
N	Representação numérica ISO-8601 do dia da semana (adicionado no PHP 5.1.0)	*1* (para Segunda) até *7* (para Domingo)
S	Sufixo ordinal inglês para o dia do mês, 2 caracteres	*st*, *nd*, *rd* ou *th*. Funciona bem com *j*
w	Representação numérica do dia da semana	*0* (para domingo) até *6* (para sábado)

DATA

z	O dia do ano (iniciando em 0)	*0* até *365*
Semana	---	---
W	Número do ano da semana ISO-8601, começa na Segunda (adicionado no PHP 4.1.0)	Exemplo: *42* (a 42ª semana do ano)
Mês	---	---
F	Um representação completa de um mês, como January ou March	*January* até *December*
m	Representação numérica de um mês, com zero à esquerda	*01* a *12*
M	Uma representação textual curta de um mês, três letras	*Jan* a *Dec*
n	Representação numérica de um mês, sem zero à esquerda	*1* a *12*
t	Número de dias de um dado mês	*28* até *31*
Ano	---	---
L	Se está em um ano bissexto	*1* se está em ano bissexto, *0*, caso contrário.

DATA

o	Número do ano ISO-8601. Este tem o mesmo valor como *Y*, exceto que se o número da semana ISO (*W*) pertence ao anterior ou próximo ano, o ano é usado ao invés. (adicionado no PHP 5.1.0)	Exemplos: *1999* ou *2003*
Y	Uma representação de ano completa, 4 dígitos	Exemplos: *1999* ou *2003*
y	Uma representação do ano com dois dígitos	Exemplos: *99* ou *03*

HORA

a	Antes/Depois de meio-dia em minúsculo	*am* or *pm*
A	Antes/Depois de meio-dia em maiúsculo	*AM* or *PM*
B	Swatch Internet time	*000* até *999*
g	Formato 12-horas de uma hora sem zero à esquerda	*1* até *12*
G	Formato 24-horas de uma hora sem zero à esquerda	*0* até *23*
h	Formato 12-horas de uma hora com zero à esquerda	*01* até *12*
H	Formato 24-horas de uma hora com zero à esquerda	*00* até *23*
i	Minutos com zero à esquerda	*00* até *59*
s	Segundos, com zero à esquerda	*00* até *59*
u	Microssegundos (adicionado no PHP 5.2.2). Note que a função **date()** sempre gerará *000000*, já que aceita um parâmetro integer, enquanto que DateTime::format() possui suporte a microssegundos se DateTime foi criado com microssegundos.	Example: *654321*

FUSO HORÁRIO

e	Identificador do fuso horário (adicionado no PHP 5.1.0)	Exemplos: *UTC*, *GMT*, *Atlantic/Azores*
I (i maiúsculo)	Se a data está ou não no horário de verão	*1* se horário de verão, *0*, caso contrário.
O	Deslocamento ao Horário de Greenwish (GMT) em horas	Exemplo: *+0200*
P	Deslocamento ao Horário de Greenwish (GMT) com dois pontos entre horas e minutos (adicionado no PHP 5.1.3)	Exemplo: *+02:00*
T	Abreviação do fuso horário	Exemplos: *EST*, *MDT* ...
Z	Deslocamento, em segundos, do fuso horário. O deslocamento para fusos horários a oeste de UTC sempre será negativa, e para aqueles à leste de UTC sempre será positiva.	*-43200* até *50400*

DATA E HORA COMPLETA

c	Data ISO 8601 (adicionado no PHP 5)	2004-02-12T15:19:21+00:00
r	» RFC 2822 formatted date	Exemplo: *Thu, 21 Dec 2000 16:01:07 +0200*
U	Segundos desde Unix Epoch (January 1 1970 00:00:00 GMT)	Veja também time()

DATA DE HOJE

```php
<?php
echo "Data atual: " . date("Y-m-d");
?>
```

OBTER A HORA ATUAL:

```php
<?php
echo "Hora atual: " . date("H:i:s");
?>
```

OBTER A DATA E HORA ATUAL COMPLETA:

```php
<?php
echo "Data e hora atuais: " . date("Y-m-d H:i:s");
?>
```

FORMATAR UMA DATA ESPECÍFICA:

```php
<?php
$data = "2024-06-02";
$data_formatada = date("d/m/Y", strtotime($data));
echo "Data formatada: " . $data_formatada;
?>
```

ADICIONAR/SUBTRAIR DIAS A PARTIR DE UMA DATA:

```php
<?php
$data = "2024-06-02";
$nova_data = date("Y-m-d", strtotime($data . " +1 day"));
echo "Nova data: " . $nova_data;
?>
```

ADICIONAR/SUBTRAIR DIAS A PARTIR DE UMA DATA:

```php
<?php
$data1 = "2024-06-02";
$data2 = "2024-06-10";
$diferenca = strtotime($data2) - strtotime($data1);
$dias = floor($diferenca / (60 * 60 * 24));
echo "Diferença em dias: " . $dias;
?>
```

FUNÇÃO DATE()

date(): Esta função é usada para formatar uma data e/ou hora de acordo com um formato especificado.

```php
<?php
echo date("Y-m-d H:i:s"); // Saída: 2024-06-02 14:30:45
?>
```

FUNÇÃO TIME()

time(): Retorna a data atual em segundos

```php
<?php
echo time(); // Saída: 1622624473 (um número que representa os segundos desde a Época
?>
```

FUNÇÃO GETDATE():

getdate(): Retorna um array associativo contendo informações sobre a data/hora fornecida ou a data/hora atual, se não for fornecida.

```php
<?php
$data = getdate();
print_r($data);
/*
Saída (exemplo):
Array
(
    [seconds] => 45
    [minutes] => 30
```

```
    [hours]   => 14
    [mday]    => 2
    [wday]    => 5
    [mon]     => 6
    [year]    => 2024
    [yday]    => 153
    [weekday] => Friday
    [month]   => June
    [0]       => 1622625045
)
*/
?>
```

FUNÇÃO MKTIME()

A função mktime() retorna o timestamp Unix para uma data/hora específica.
Um **timestamp** (ou marca temporal) é um número inteiro que representa o número de segundos desde o início da **Era Unix** (também conhecido como o Epoch Unix), que começou à meia-noite de 1º de janeiro de 1970, no horário UTC (Tempo Universal Coordenado). Esse sistema de contagem é amplamente utilizado em sistemas de computadores e linguagens de programação para rastrear e armazenar informações de data e hora de maneira uniforme.

FUNÇÃO MKTIME()

```php
<?php
// mktime(hour, minute, second, month, day, year)
$timestamp = mktime(14, 30, 0, 6, 14, 2024);
echo "Timestamp específico: " . $timestamp . "\n";
echo "Data formatada: " . date("Y-m-d H:i:s", $timestamp) . "\n";
?>
```

FUSO HORÁRIO

Para trabalhar com fusos horários, você pode usar a classe **DateTimeZone.**

ÁfricaAfrica/Cairo
Africa/Johannesburg
Africa/Lagos
Africa/Nairobi

FUSO HORÁRIO

América

America/Argentina/Buenos_Aires
America/Chicago
America/Los_Angeles
America/Mexico_City
America/New_York
America/Sao_Paulo

FUSO HORÁRIO

Ásia

Asia/Bangkok
Asia/Dubai
Asia/Hong_Kong
Asia/Kolkata
Asia/Shanghai
Asia/Tokyo

FUSO HORÁRIO

Europa

Europe/Berlin
Europe/Lisbon
Europe/London
Europe/Madrid
Europe/Paris
Europe/Rome

FUSO HORÁRIO

Oceania

Australia/Sydney

Pacific/Auckland

Pacific/Fiji

FUSO HORÁRIO

```php
<?php
$date = new DateTime("now", new DateTimeZone("America/Sao_Paulo"));
echo "Data e hora em São Paulo: " . $date->format("Y-m-d H:i:s") . "\n";

$date->setTimezone(new DateTimeZone("Europe/London"));
echo "Data e hora em Londres: " . $date->format("Y-m-d H:i:s") . "\n";
?>
```

FUNÇÃO STRTOTIME

Serve para fazer cálculos com data.

Ex:

```
date_default_timezone_set('America/Sao_Paulo');
echo date("d/m/Y", strtotime("now")); //data atual
echo "<br />";
echo date("d/m/Y",strtotime("+1 day"));//data atual mais 1 dia
echo "<br />";
```

FUNÇÃO STRTOTIME

```
echo date("d/m/Y",strtotime("+1 week"));//data atual mais uma semana
echo "<br />";
echo date("d/m/Y",strtotime("next Tuesday"));// proxima terça feira
```

FUNÇÃO STRTOTIME

```
echo "<br />";
echo date("d/m/Y",strtotime("+1 month"));// acrescenta um mes
echo "<br />";
echo date("d/m/Y",strtotime("last tuesday")); //última terça feira
echo "<br />";
```

ATIVIDADE

Crie uma página de internet utilizando PHP onde o usuário digita a data e a hora e converta para um pais que o usuário também posso escolher.

Em PHP, você pode criar strings usando aspas simples (') ou aspas duplas (").

```
// Aspas simples
$single_quote_string = 'Hello, World!';

// Aspas duplas
$double_quote_string = "Hello, World!";
```

FUNÇÕES COMUNS PARA MANIPULAÇÃO DE STRINGS

strlen() - Retorna o tamanho da string.

```
<?php
$str = "Senac";
$tamanho = strlen($str); // 5
echo $tamanho;
?>
```

FUNÇÕES COMUNS PARA MANIPULAÇÃO DE STRINGS

strpos() – Mostra quantos caracteres existem antes da palavra.

```
<?php
$texto = "Seja bem vindo ao Senac";
$resultado = strpos($texto, "Senac"); // 18
echo $resultado."<br>";
?>
```

FUNÇÕES COMUNS PARA MANIPULAÇÃO DE STRINGS

str_replace – Substituir uma palavra.

```
<?php
$frase = "Seja bem vindo ao Google";
$substituir = str_replace("Google", "Senac", $frase);
echo $substituir."<br>";
?>
```

FUNÇÕES COMUNS PARA MANIPULAÇÃO DE STRINGS

substr – Retorna uma parte do texto.

```
<?php
$frase = "Curso técnico no Senac";
$sub = substr($frase, 17, 6);
echo $sub."<br>";
?>
```

FUNÇÕES COMUNS PARA MANIPULAÇÃO DE STRINGS

substr – Retorna uma parte do texto.

```
<?php
$frase = "Curso técnico no Senac";
$sub = substr($frase, 17, 6);
echo $sub."<br>";
?>
```

FUNÇÕES COMUNS PARA MANIPULAÇÃO DE STRINGS

strtolower – transforma em minúscula.

```
<?php
$texto="AULA DE PHP";
$string = strtolower($texto);
echo $string."<br>";
?>
```

FUNÇÕES COMUNS PARA MANIPULAÇÃO DE STRINGS

strtolower – transforma em minúscula.

```
<?php
$texto="AULA DE PHP";
$string = strtolower($texto);
echo $string."<br>";
?>
```

FUNÇÕES COMUNS PARA MANIPULAÇÃO DE STRINGS

strtoupper() – Converte uma string para maiúsculas.

```
<?php
$texto="aula de php";
$string = strtoupper($texto);
echo $string."<br>";
?>
```

FUNÇÕES COMUNS PARA MANIPULAÇÃO DE STRINGS

ucfirst() – Converte o primeiro caractere de uma string para maiúscula.

```
<?php
$texto="seja bem vindo ao Senac";
$string = ucfirst($texto);
echo $string."<br>";
?>
```

FUNÇÕES COMUNS PARA MANIPULAÇÃO DE STRINGS

str_split() – Repete uma string um número especificado de vezes.

```
<?php
$texto = "Curso de informática <br>";
$fraseRepetcao = str_repeat($texto, 3);
echo $fraseRepetcao;
?>
```

FUNÇÕES COMUNS PARA MANIPULAÇÃO DE STRINGS

strrev() – Inverte uma string

```
<?php
$frase = "Senac";
$inverso = strrev($frase);
echo $inverso."<br>";
?>
```

FUNÇÕES COMUNS PARA MANIPULAÇÃO DE STRINGS

strcmp() – Compara duas strings (sensível a maiúsculas e minúsculas).

```
<?php
$frase1 = "Senac";
$frase2 = "senac";
$comparar = strcmp($frase1, $frase2);
echo
```

FUNÇÕES COMUNS PARA MANIPULAÇÃO DE STRINGS

strcmp() – Compara duas strings (sensível a maiúsculas e minúsculas).

```
<?php
$frase1 = "Senac";
$frase2 = "senac";
$comparar = strcmp($frase1, $frase2);
echo $comparar."<br>";
?>
```

FUNÇÕES COMUNS PARA MANIPULAÇÃO DE STRINGS

substr_replace() – Compara duas strings (não sensível a maiúsculas e minúsculas).

```
<?php
$frase1 = "Senac";
$frase2 = "senac";
$comparar = strcasecmp($frase1, $frase2);
echo $comparar."<br>";
?>
```

CONCLUINDO

PHP oferece uma rica coleção de funções para manipulação de strings, cada uma atendendo a diferentes necessidades de processamento e formatação de texto. Com essas funções, é possível realizar desde tarefas simples, como modificar o caso de caracteres, até operações mais complexas, como substituir substrings e formatar. A prática com essas funções fortalecerá sua habilidade de manipular strings de forma eficaz e eficiente em PHP, tornando o desenvolvimento mais fluido e robusto.

EXERCÍCIO

Em php, crie um formulário onde o usuário possa digitar uma frase e repita essa frase a quantidade de vezes dos caracteres dessa frase.

IF

O construtor if é um dos recursos mais importantes em muitas linguagens, inclusive no PHP. Permite a execução condicional de fragmentos de código. O PHP apresenta uma estrutura if semelhante a do C:

ESTRUTURAS DE CONTROLE DE FLUXO CONDICIONAIS

INTRODUÇÃO

Estruturas de controle de fluxo são responsáveis pela forma como o fluxo do programa será executado, ou seja, são nelas que colocamos condições para uma ou outra rotina ser executada.

INTRODUÇÃO

Permite dinamismo ao programa, sem estruturas de controle não teríamos como controlar a execução de nosso programa de modo a determinar quais ações devem ser executadas em determinados casos, sendo assim, uma linguagem depende muito das estruturas de controle.

INTRODUÇÃO

A estrutura de controle no PHP é divida em duas partes: **comandos de seleção** e comand**os de repetição.**

COMANDOS DE SELEÇÃO.

são também conhecidos como comandos condicionais, neles é possível executar comandos ou bloco de comandos com base em testes feitos durante a execução, o comandos são: IF e SWITCH.

COMANDOS DE REPETIÇÃO.

São comandos utilizados para que um conjunto de instruções seja executado repetidamente por um determinado número de vezes ou até que uma condição seja atingida, os comandos são: WHILE, DO..WHILE, FOR e FOREACH.

IF

O operador IF é utilizado para avaliar o valor de uma condição booleana, ou seja, que pode assumir apenas dois valores distintos: VERDADEIRO (true) ou FALSO (false). IF significa **SE** em português e a estrutura funciona da seguinte forma: SE (condição) ENTÃO (faça algo).

ESTRUTURA

```
If(condição){

Código

}
```

EXEMPLO

Verificar se a Variável I é maior que a Variável 2.

EXEMPLO

```php
<?php
$valor1=$_GET["a"];
$valor2=$_GET["b"];
if($valor1>$valor2){
    echo "O valor da varivel 1 é maior que o valor da variavel 2";

?>
```

IF E ELSE

O operador IF pode ser complementado com o operador **ELSE**, que significa **SENÃO**. Ou seja, SE uma condição for atendida, fazer uma coisa. SENÃO, fazer outra. O ELSE só precisa ser utilizado quando houver algo específico a ser feito se a condição não for atendida.

ESTRUTURA

```
If(condição){
Código quando condição for verdadeira
}
Else{
Código quando condição for falsa
}
```

ESTRUTURA

Verificar qual variável é maior

EXEMPLO

```php
<?php
$valor1=$_GET["a"];
$valor2=$_GET["b"];
if($valor1>$valor2){
    echo "O valor da varivel 1 é maior que o valor da variavel 2";
    } else{
        echo "O valor da varivel 2 é maior que o valor da variavel 1";
    }

?>
```

EXEMPLO 2

Inserir 4 notas, mostrar a média do aluno e se o mesmo obter a média até 5 está reprovado, senão estará aprovado.

EXEMPLO 2

```php
<?php
$valor1=$_GET["a"];
$valor2=$_GET["b"];
$valor3=$_GET["c"];
$valor4=$_GET["d"];
$media=($valor1+$valor2+$valor3+$valor4)/4;

if($media<=5){
    echo "A média do aluno é: $media e está Reprovado";
    } else{
        echo "A média do aluno é: $media e está Aprovado";
    }

?>
```

ELSEIF

Essa opção é utilizada caso seja necessário verificar duas ou mais condições.

ESTRUTURA

```
If(condição){
Código quando condição for verdadeira
}
Elseif (condição){
Código quando condição for falsa
}
Else{
Condição
}
```

EXEMPLO 3

Inserir 4 notas, mostrar a média do aluno e se o mesmo obter a média de 0 até 5 reprovado, se a média estiver entre 5.1 e 6.9 recuperação, se a média for maior que 7 aprovado.

EXEMPLO 3

```php
<?php
$valor1=$_GET["a"];
$valor2=$_GET["b"];
$valor3=$_GET["c"];
$valor4=$_GET["d"];
$media=($valor1+$valor2+$valor3+$valor4)/4;

if($media<=5){
    echo "A média do aluno é: $media e está Reprovado";
    }
    elseif($media>5&&$media<7){
        echo "A média do aluno é: $media e está Recuperação";

    }
    else{
        echo "A média do aluno é: $media e está Aprovado";
    }

?>
```

SWITCH CASE

SWITCH CASE

No switch/case a variável recebe um valor como parâmetro e verifica se ele atende alguma das condições especificadas. Em caso positivo o trecho de código relacionado a essa condição é executado.

SWITCH CASE

Diferente da estrutura if/else, o switch/case avalia apenas condições de igualdade. Ou seja, ela verifica se o valor recebido como parâmetro é igual a alguma das opções especificadas em seu corpo.

ESTRUTURA

```
switch (expressão) {
      case valor1:
        //código a ser executado se a expressão for igual ao valor1
        break;
    case valor2:
        //código a ser executado se a expressão for igual ao valor2
        break;
  case valorN:
        //código a ser executado se a expressão for igual ao valorN
        break;
}
```

EXEMPLO

O usuário deverá inserir 2 valores e escolher se vai querer somar, subtrair, multiplicar ou dividir.

```
echo "Digite 1 para somar <br/>";
echo "Digite 2 para subtrair <br/>";
echo "Digite 3 para multiplicar <br/>";
echo "Digite 1 para dividir <br/>";
$valor1=$_GET["a"];
$valor2=$_GET["b"];
$valor3=$_GET["c"];
$resultado;

switch($valor3){
        case 1;
        $resultado=$valor1+$valor2;
        echo "Escolheu a opção 1 - a soma dos resultados foi: $resultado";
        break;
        case 2;
        $resultado=$valor1-$valor2;
        echo "Escolheu a opção 2 - a subtração dos resultados foi: $resultado";
        break;
        case 3;
        $resultado=$valor1*$valor2;
        echo "Escolheu a opção 3 - a multiplicação dos resultados foi: $resultado";
        break;
        case 4;
        $resultado=$valor1/$valor2;
        echo "Escolheu a opção 4 - a soma dos resultados foi: $resultado";
        break;
}
?>
```

ESTRUTURAS DE REPETIÇÃO

ESTRUTURAS DE REPETIÇÃO

É uma estrutura que permite executar mais de uma vez o mesmo comando ou conjunto de comandos, de acordo com uma condição ou com um contador.

ESTRUTURAS DE REPETIÇÃO

São utilizadas, por exemplo, para repetir ações semelhantes que são executadas para todos os elementos de uma lista de dados, ou simplesmente para repetir um mesmo processamento até que a condição seja satisfeita.

FOR

O **for** é a estrutura de repetição do PHP que utilizamos quando sabemos a quantidade de repetições que devem ser executadas.

SINTAXE

```
for (expressão 1; expressão 2; expressão 3) {
    // bloco de código
}
```

EXPRESSÃO 1

Executada somente uma vez, ao iniciar o loop. Normalmente a utilizamos para declarar e inicializar as variáveis que faremos uso para controlar o número de iterações do loop;

EXPRESSÃO 2

Expressão booleana, validada antes de cada iteração do loop. Se a expressão contiver múltiplas expressões, todas serão executadas, mas somente o resultado da última será considerado. Se a expressão for vazia, ela será interpretada como verdadeira. O loop somente será executado enquanto essa expressão retornar **true**;

EXPRESSÃO 3

Executada ao final de cada iteração, normalmente a utilizamos para declarar a forma de atualização do valor da variável avaliada na expressão 2.

EXEMPLO

Crie uma página da web que mostre os numerais de 0 até 9.

RESPOSTA

```
for ($i=0; $i < 10; $i++) {
    echo $i."<br>";
}
```

EXEMPLO 2

Crie uma página da web que mostre a tabuada do 9.

RESPOSTA

```
for ($i=0; $i <= 10; $i++) {
    echo "<br />9 X $i =".'9'*$i;
}
```

WHILE

O while é a estrutura de repetição mais simples do PHP. Com ele informamos que um bloco de código deve ser repetido enquanto a condição declarada for verdadeira.

ESTRUTURA

```
while (condição) {
    // bloco de código
}
```

EXEMPLO

Crie uma página da web que mostre os numerais de 1 até 10.

RESPOSTA

```
while ($i <= 10) {
    echo "$i<br>";
    $i++;
```

EXEMPLO

Crie uma página da web que mostre a tabuada do 8.

RESPOSTA

```
while ($i <= 10) {
    echo "<br />8 x $i = ".'8'*$i;
    $i++;

}
```

DO WHILE

O loop do-while tem comportamento parecido com o while, diferenciando-se somente na validação do loop, que é feita no final de cada iteração.

DO WHILE

Devido a essa característica, normalmente utilizamos essa estrutura de repetição quando desejamos que o bloco de código declarado no loop seja executado pelo menos uma vez.

ESTRUTURA

```
do {
    //bloco de código
} while (condição);
```

EXEMPLO

Crie uma página da web que mostre os numerais de 0 até 10.

RESPOSTA

```
$i = 0;
do {
    echo"<br> $i";
    $i++;
} while ($i < 11);
```

EXEMPLO

Crie uma página da web que mostre a tabuada do 7.

RESPOSTA

```
$i = 0;
do {
    echo"<br> 7 X $i = ".'7'*$i;
    $i++;
}
while ($i < 11);
```

FOREACH

O foreach é uma estrutura de repetição da linguagem de programação PHP. Ele é usado para facilitar a iteração de estruturas como arrays, objetos e outros tipos que são iteráveis.

FOREACH

Existem vários modos para percorrer arrays, no entanto, o mais simples deles é utilizando o laço de repetição foreach em PHP. Este comando funciona só com arrays e objetos, e retorna um erro quando utilizado com outros tipos de expressões.

FOREACH

O **PHP foreach** é um laço de repetição especializada na iteração de Arrays. Ou seja, ele funciona como uma estrutura que está projetada para percorrer todos os elementos de uma Array. Dessa forma, além de melhorar a legibilidade do código, também evitamos alguns problemas, como o acesso a elementos não existentes. Este é um problema que pode ocorrer quando trabalhamos com uma estrutura do laço de repetição **for** em sua definição básica.

FOREACH

Com o **PHP Foreach** temos acesso a todos os elementos, da mesma forma que teríamos se trabalhássemos com o **for** normal. Dessa forma, temos um laço de repetição que percorrerá todos os elementos e a cada ciclo será definido o próximo elemento contido na estrutura que está sendo iterada. Dessa forma, podemos entender o **foreach** como uma função que, a cada elemento de uma array, executa um bloco de ações definidas.

SINTAXE BÁSICA DO PHP FOREACH

A sintaxe básica do PHP Foreach é conforme o código abaixo:

```
foreach ($array as $value) {
    //código a ser executado;
}
```

Dessa forma, para cada iteração do laço de repetição, o valor do elemento atual da Array é atribuído ao valor $value. Consequentemente o ponteiro da array é movido um a um, até atingir seu último elemento.

SINTAXE BÁSICA DO PHP FOREACH

Vejamos então o exemplo abaixo:

```
<!DOCTYPE html>
<html>
<body>
<?php
    $cores = array("azul", "vermelho", "amarelo", "verde");
    foreach ($cores as $value) {
        echo "$value <br>";
    }
?>
</body>
</html>
```

SINTAXE BÁSICA DO PHP FOREACH

Caso deseje modificar diretamente elementos de um array dentro de um laço, preceda $value com &, conforme o exemplo abaixo:

```
<!DOCTYPE html>
<html>
<body>
 <?php
   $arr = array(1, 2, 3, 4);
   foreach ($arr as &$value) {
       $value = $value * 3;
   }
   print_r($arr);
 // agora a nosa $arr possui os valores (3, 6, 9, 12)
 ?>
</body>
</html>
```

PHP FOREACH: UTILIZANDO O VALOR DA CHAVE

Além da sintaxe básica, também possuímos outras formas de utilizar o PHP **Foreach**. Dentre elas, existe uma forma onde para cada valor do **array**, também podemos utilizar a **chave** dela para trabalhar o nosso código. Para isso basta usar a seguinte sintaxe:

PHP FOREACH: UTILIZANDO O VALOR DA CHAVE

Além da sintaxe básica, também possuímos outras formas de utilizar o PHP **Foreach**. Dentre elas, existe uma forma onde para cada valor do **array**, também podemos utilizar a **chave** dela para trabalhar o nosso código. Para isso basta usar a seguinte sintaxe:

PHP FOREACH: UTILIZANDO O VALOR DA CHAVE

```
<?php
    foreach ($array as $key => $value) {
        #codigo
    }
?>
```

PHP FOREACH: UTILIZANDO O VALOR DA CHAVE

Dessa forma, vejamos o seguinte exemplo:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>PHP Foreach utilizando $key</title>
</head>
<body>
<?php
    $cores = array("Azul","Amarelo","Vermelho","Rosa","Verde");
    foreach ($cores as $key => $value) {
        echo "Minha cor no indice ".$key." é ".$value."<br/>";
    }
?>
</body>
</html>
```

PHP FOREACH: UTILIZANDO O VALOR DA CHAVE

Dessa forma, enquanto percorremos a array $cores, vamos também estar pegando o valor da sua chave. Consequentemente, vamos imprimir a mensagem utilizando a chave e também o valor de cada campo do array. Vejamos então o resultado conforme a imagem abaixo: