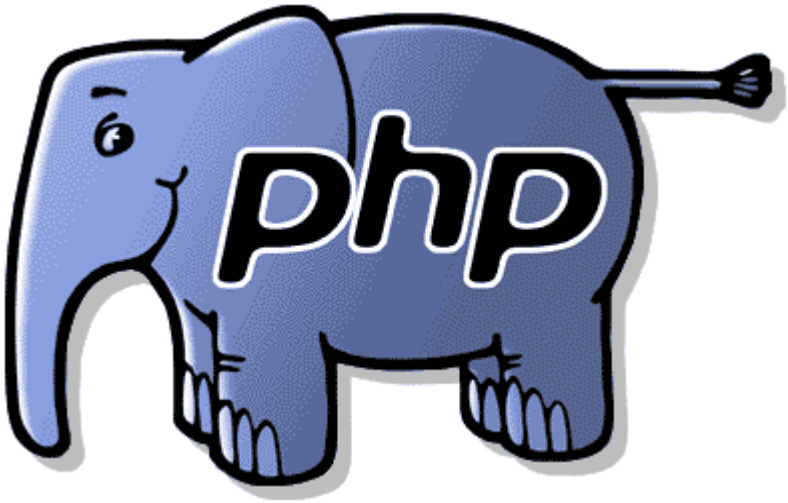
Apostila PHP Progressivo



www.phpprogressivo.net

Proprietário da Apostila

Essa apostila pertence a:

Pagador: Andre Souza Maia
Código: 6F6D93A60525413B98776419AB46E81D
Email do comprador: andremaya@gmail.com
Banco Pagador: NU PAGAMENTOS - IP

Pedimos, encarecidamente, que não distribua ou comercialize seu material. Além de conter suas informações, prejudica muito nosso projeto.

Se desejar indicar o PHP Progressivo para um amigo, nosso site possui todo o material, de forma gratuita, sem precisar de cadastro e o acesso dessas pessoas também ajuda a mantermos o site no ar e criamos cada vez mais projetos:

www.phpprogressivo.net

Sumário

- **Introdução ao PHP Básico**

1. PHP - O que é? Para que serve? Onde e como é usado?
2. O necessário para começar a programar em PHP
3. Olá, Mundo! (Hello, World!) em PHP
4. Saída simples: echo e print
5. Exercícios de saída simples
6. Tipos de dados
7. Variáveis
8. Matemática: Operadores Aritméticos de Soma, Subtração, Multiplicação, Divisão e Resto da Divisão em PHP
9. Recebendo dados via input do HTML para o PHP
10. Calculadora simples
11. Precedência de operadores
12. Comentar código PHP
13. Exercícios Básicos de PHP

- **Testes Condicionais em PHP**

1. Operadores de Comparação
2. O Comando IF - Teste Condicional
3. Instrução IF ELSE - Controle de Fluxo
4. IF e ELSE aninhados
5. A instrução ELSEIF em PHP
6. Comando SWITCH (break e default) em PHP
   Exercício: PHP com formulário HTML <select>
   Exercício: Quantos dias cada mês possui
7. Operadores Lógicos do PHP: &&(AND), ||(OR), !(NOT) e XOR
   Exercício: Ano bissexto em PHP
8. Operador Ternário - ?:
9. Exercícios de Testes condicionais em PHP

- **Laços e Loopings em PHP**

1. Operadores de Atribuição em PHP
2. Laço WHILE
3. Laço DO WHILE
4. Laço FOR: O looping controlado

5. Exercício: Como fazer a tabuada em PHP
6. Comandos BREAK e CONTINUE
7. Fibonacci com laços
8. Fatorial com laços
9. Números primos com laços
10. Lista de exercícios de laços em PHP

- **Funções**

1. Função - O que é? Para que serve? Como funciona?
2. Como invocar e criar uma função
3. Parâmetros, Argumentos e Retorno de uma função
4. Passagem por Valor e por Referência
5. Variável de escopo Global: global e $GLOBALS
6. Como incluir arquivos PHP externos
7. Funções recursivas: Recursividade em PHP
8. Exercício: Conversão Celsius Fahrenheit e vice-versa usando funções
9. Gerando números aleatórios com a função rand()
10. Exercício: Jogo de adivinhar o número
11. Lista de exercícios de funções em PHP

- **Array em PHP**

1. Arrays em PHP - O que é? Para que serve? Onde se usa ?
2. Arrays em PHP - Como criar, acessar e exibir elementos
3. Arrays associativos
4. Função range() para arrays
5. Foreach - Laço de arrays
6. Funções de arrays

- **Programação Orientada a Objetos**

1. O que são classes e objetos
2. Polimorfismo, Herança e Encapsulamento em Orientação a Objetos
3. Como criar classes, objetos, atributos e métodos
4. Propriedades SET e GET
5. Escopo de Atributos e Métodos: public, private, protected, final e static
6. Método construtor

- **Data e Tempo em PHP**

1. Como checar se uma data é válida: checkdate()
2. Como exibir datas e horários em PHP: função date()
3. Função mktime() e a Unix Timestamp
4. Função getdate() - Obtendo datas e horários em PHP
5. Função gettimeofday() - Obtendo tempo atual
6. Função time() e date() juntas
7. Data e Hora em português: setlocale() e strftime()
8. Última modificação de página: getlastmod()
9. Números de dia em um mês qualquer
10. Calculando datas no futuro e no passado: strotime()
11. Classe DateTime: Data e Tempo com orientação a objetos

- **Tutoriais de PHP**

Exibir o IP do visitante
Informações do navegador do cliente
Verifica se é HTTPs ou HTTP
Como redirecionar o usuário para outro site
Atrasar (congelar, bloquear) execução de script

Introdução

Nessa seção, iremos estudar o básico da programação PHP.

Começaremos bem do início, explicando o que é o PHP, para que serve, onde e como é usado.
Também vamos te orientar a instalar tudo que é necessário para iniciar seus estudos em desenvolvimento Web.

Vamos ver como exibir mensagens e textos em páginas HTML, operações matemáticas, como fazer o PHP receber dados do usuários etc.

Tudo bem devagar, bem do básico, pressupondo que o aluno não tenha absolutamente nenhum conhecimento em nenhuma linguagem de programação.

PHP - O que é ? Para que serve? Como Funciona ?

Neste tutorial inicial de nosso **Curso de PHP**, vamos falar sobre o que é o PHP, para que serve, como funciona, falar um pouco de sua história, onde e como iremos usar ele durante nossos estudos.

Prepare seu café, puxe uma cadeira e se prepare se prepare!

- ## PHP - O que é?

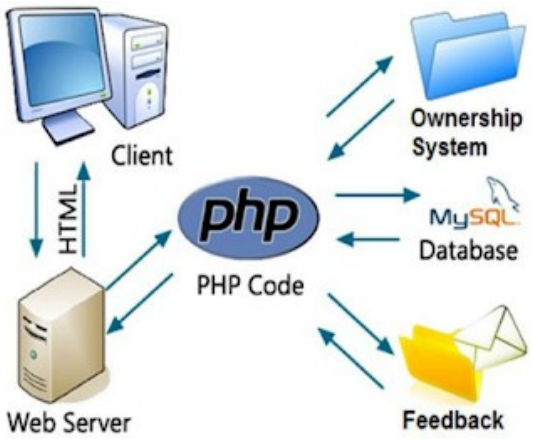
PHP nada mais é que uma linguagem de programação, especificamente do tipo linguagem de script, cujo foco é atuar na Web, no lado do servidor, para geração de páginas dinâmicas.

Calma, primeira vez que li, também não entendi nada! Mas vamos explicar o que é esse monte de palavras diferentes.

Linguagem de programação é uma língua, assim como português ou inglês que usamos para nos comunicar entre nós, humanos. No caso do PHP, é uma língua que usamos para nos comunicar com o computador, para criar programas de computador, ou *softwares*.

No caso específico do PHP, usamos essa linguagem para criar um tipo mais simples de programa de computador, os *scripts*, que geralmente são instruções menores e mais simples.

Ser uma linguagem interpretada significa que vai ter um outro programa (no caso, o módulo PHP). que vai ler os comandos, interpretando e executando de imediato (sem necessidade de compilar e transformar num arquivo binário executável, como um .exe).



- **PHP - Como funciona ?**

Todo site que você entre, as informações contidas nele estão em um servidor, ou seja, em algum outro computador.

Por exemplo, se entra em uma rede social e visualiza a foto de algum amigo, aquela foto está guardada em alguma máquina, em algum local do mundo. Numa pasta (diretório) dentro dessa máquina, do mesmo jeito como você guarda seus arquivos (no C:\, por exemplo).

O PHP vai trabalhar com isso: ele vai atuar no lado do servidor, tratando esses pedidos (*requests*) e respondendo na forma de uma página HTML.

Por isso que dizemos que a linguagem PHP é totalmente voltada para desenvolvimento web, especificamente para *scripting* do lado do servidor, ela vai receber pedidos, vai consultar seus dados nos servidores, e vai levar essas informações pro lado do usuário.

O JavaScript, por exemplo, é comumente usado do lado do cliente, do usuário (atua nos browsers, navegadores de internet), e costuma trabalhar bastante junto do PHP.

Estude também: Curso JavaScript Progressivo

- **Onde PHP é usado ?**

PHP é a linguagem web mais impostante de todas, o uso dela em *server-side* é simplesmente absurdo e impossível de se calcular com exatidão, tamanha sua importância.

Algumas pesquisas mostram que mais de 70% dos usuários são atendidos por servidores que usam PHP.

Dentre os serviços de internet mais conhecidos que usam PHP, sem dúvidas o maior e mais importante é o Facebook, que foi construído nessa linguagem (em algumas melhorias foram feitas, usando outras linguagens).

Veja bem: o Facebook atende mais de 2 bilhões de usuários, muitos e muitos milhões ao mesmo tempo, e usa PHP por trás, em seus códigos.

Entendeu a importância e poder do PHP ?

Outros sites que usam a linguagem: Tumblr, Wikipedia, todos sites que usam Wordpress, Dailymotion, sites de e-commerce e muitos e muitos outros sites de serviços famosos na internet.

O que é possível fazer com PHP ?

Embora seja uma linguagem interpretada voltada para *server scripting*, ela é em sua essência uma linguagem de propósito geral, podendo ter outras utilidades.

Mas falando só de desenvolvimento web, a vastidão de coisas e projetos que é possível fazer com PHP é simplesmente incalculável.

Um exemplo simples e prático, que muitos programadores de PHP fazem com muita frequência, são sites de *e-commerce*, ou seja, sites de compras, com carrinhos, calculadora de frete, cupons de desconto, promoções, informação do número de items no estoque etc etc etc.

Falando em items de um estoque, PHP funciona lindamente bem para se trabalhar com dados (banco de dados), seja pra ler dados, escrever, mudar, consultar, conferir, atualizar, exibir no HTML e fornece uma segurança incrível, e nativa, facilitando muito o uso e manipulação de informações.

Mais informações sobre o PHP

PHP é multiplataforma, ou seja, roda se você estiver usando Windows, Linux (e suas vááárias distribuições), Mac, até celulares com sistema Android ou Apple.

PHP é considerada uma linguagem simples, de fácil entendimento e uso, sendo muito usada por iniciantes, pela pequena curva de aprendizado. Em pouquíssimo tempo é possível criar sites e aplicações incríveis.

PHP serve para criar páginas dinâmicas. Por exemplo, cada pessoa que entra no Facebook, vê publicações diferentes, fotos diferentes, posts diferentes...o mesmo site (www.facebook.com), exibe coisas diferentes, dependendo de quem acessa, é dinâmico!

Não era assim lá em 1995, quando Rasmus Lerdorf criou a ***P****ersonal* ***H****ome* ***P****age Tools* para automatizar a criação e exibição de conteúdo em seus sites HTML estáticos.

Hoje PHP é sigla recursiva para "PHP: Hypertext Preprocessor". Ou seja, é um processador de hypertexto (HTML).

PHP - O necessário para Começar a Programar (Web Server)

Agora que já aprendemos o que é, para que serve e onde o PHP é usado, vamos aprender como preparar um ambiente para programarmos em PHP.

• PHP - O Servidor

Todo site ou serviço da internet, está hospedado em algum computador. Em um tipo especial de computador: um servidor.

Quando você entra em um site, como o da Globo e vê uma foto, esta foto está armazenada em algum servidor. Então, seu navegador vai atuar por você:
Cliente - "Ei servidor, quero acessar sua página, manda os dados pra mim"
Servidor - "OK, toma aí o HTML, o CSS, as imagens, vídeos..."
Cliente - "Ok! Agora quero os dados dessa outra página?"
Servidor - "Só um minuto, vou procurar. Pronto. Enviando."

E fica nessa eterna 'conversa'.
E como explicamos no tutorial anterior, PHP é uma linguagem de programação voltada para o lado do servidor. Ou seja: vamos precisar de um servidor para estudar PHP.

Na verdade, não é obrigatório, você pode comprar um servidor (o que é **muito** caro) ou pagar um serviço de hospedagem, mas isso tem vários inconvenientes, como uma possível lentidão.

Por isso, vamos criar um servidor de PHP em nossa própria máquina!
Além da questão da velocidade (se rodar cliente e servidor numa mesma máquina, o resultado é praticamente instantâneo, sem demora), a conexão nunca vai cair, seu site nunca ficará fora do ar e não tem perigo de alguém invadir seus sistema enquanto você está em processo de aprendizagem, pois nesse estágio deixamos várias falhas.

▪ Montando um servidor Web: *Apache*, *MySQL* e *PHP*

Para começar a programar em PHP, vamos precisar de três coisas instaladas em seu computador:

- Apache - Servidor HTTP Apache, programa que vai simular um servidor e as comunicações entre servidor e cliente, tudo na sua máquina, numa espécie de 'rede interna'
- MySQL - Sistema de gerenciando de banco de dados, para que possamos armazenar, consultar, alterar e fazer de tudo com informações (dados do usuário, login, senha etc)
- PHP - Interpretador PHP, um programa que vai interpretar o código PHP que vamos escrever durante nosso curso

• WAMP, MAMP, LAMP e o XAMPP

Calma, não se assuste com essa sopa de letrinhas! São apenas abreviações:
WAMP - Windows, Apache, MySQL, e PHP
MAMP - Mac, Apache, MySQL e PHP
LAMP - Linux, Apache, MySQL, e PHP
XAMPP - Cross(x) Apache, MySQLDB, PHP e Perl

O que raios é isso?
Simples: seu ambiente de programação. Em vez de ter que sair instalando e configurando vários programas e funcionalidades (Apache, MySQL, PHP, Perl, etc etc), você instala uma única dessas opções, e elas já contém tudo.

Literalmente você vai ter tudo pronto, instalado e configurado, baixando apenas uma coisa.
Como estou com uma máquina em Linux e outras pessoas em Windows, vamos instalar e usar o XAMPP, pois ele é mais completo e *cross-plataform*, ou seja, funciona em tudo que é sistema operacional.

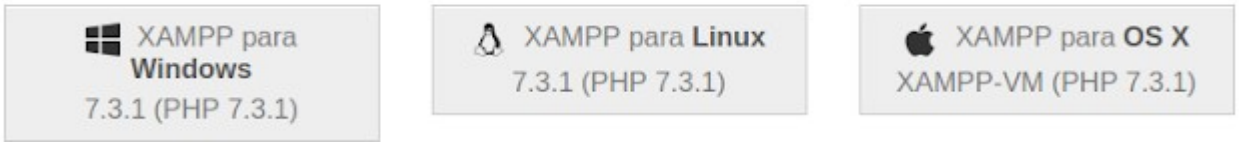
Você pode escolher qualquer uma das opções lá de cima. O PHP é o mesmo pra todos e servem em todos esses servidores.

Como Instalar o XAMPP

Como explicamos, além de rodar em todo sistema operacional, o XAMPP também é mais completo (possui além do já dito: FileZilla, OpenSSL, Webalizer, OpenSSL, Mercury Mail etc).

Primeiro, vá no Google e digite: XAMPP
Você vai cair na página: https://www.apachefriends.org/pt_br/index.html

Baixe a opção de acordo com seu SO:



Baixe sempre a versão mais atual. No momento em que vos escrevo, é a 7.3.1
Abra o arquivo.

Vai começar a típica instalação: OK, Next, Próxima, Ok, Install...
Na opção de componentes, deixo tudo marcado, para instalar logo tudo em sua máquina, para ter um super servidor, bem semelhante aos de verdade, quando você colocar seu site no ar.

Na pasta de instalação, escolha uma bem no início:
Windows - C:\xampp
Linux : /home/user/xampp

Assim fica mais fácil trabalhar, evite nomes grandes, com acentos e espaçamento:
C:\Zezinho\Desktop\Programacao\PHP\Programação Progressiva

Aguarde a instalação terminar e configurar seu ambiente de desenvolvimento PHP.
Ele vai perguntar se deseja rodar o XAMPP, diga que sim.

Ao abrir o XAMPP, vá na aba "Manage Servers".
Lá, clique em **Start All** para colocar tudo pra rodar.

E prontinho, você já tem um servidor rodando em seu computador.

Vamos confirmar se você fez tudo certo.

Abra seu browser e digite os endereços (como se fossem sites que você vai entrar):

- localhost
- 127.0.0.1

Se aparecer uma tela do Apache, parabéns, você fez tudo correto e já está praticamente pronto para começar a programar em PHP.

Editor de Textos

Existem vários programas especiais para escrevermos nossos códigos PHP, se você perguntar para alguns profissionais, vai receber o nome de pelo menos 10. É como comida, cada um tem seu prato favorito.

De início, não vamos usar esses programas (chamados IDE), e sim nosso bom e velho bloco de notas.
Vamos começar programando 'na unha', na raça mesmo.

Só se aprende a programar assim, escrevendo, digitando o código e rodando para ver a coisa funcionando, feitas com suas próprias mãos.

Nada de ficar copiando e colando código da gente, ok ?
Escreva e rode você mesmo! Tudo!

Mais fontes de informação:
https://pt.wikipedia.org/wiki/Servidor_Apache
https://pt.wikipedia.org/wiki/MySQL

Como Programar em PHP: Olá, mundo! (Hello, World em PHP)

Agora que já configuramos e criamos nosso ambiente de desenvolvimento em PHP, vamos colocar a mão na massa e começar a programar em PHP, ver algum código!

• Hello, World! (Olá, Mundo!) em PHP

Abra seu bloco de notas.

Digite o seguinte código:

```
<html>
 <head>
  <title>Curso PHP Progressivo</title>
 </head>
 <body>
  Olá mundo! To no HTML! <br/>
  <?php
   echo "Hello, World! To no PHP!";
  ?>
 </body>
</html>
```

▪ Onde salvar scripts PHP

Para podermos rodar sites em PHP, devemos ter arquivos do tipo **.php** Então, agora, você vai salvar o texto que digitou como: **home.php**

•Se estiver no Windows, salve na pasta **htdocs** que está dentro da pasta que você instalou o XAMPP. Se foi na C:\xampp, ela deve estar em:

C:\xampp\htdocs

O endereço do arquivo deve ficar: C:\xampp\htdocs\home.php

•Se estiver no Mac, salve na pasta: /Applications/XAMPP/htdocs

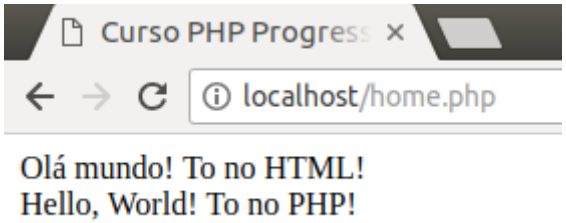
•Se estiver no Linux, salve na pasat: /var/www/html

Lembre-se: você está rodando do lado do servidor, esses arquivos devem ser salvos no seu servidor local, que são essas pastas que indicamos.

Agora rode o home.php, digitando no seu navegador:
localhost/home.php

O navegador vai, automaticamente, trocar *localhost* pelo endereço do seu servidor local e rodar o script que você programou.

O resultado deve ser:



- **A tag <?php ?>**

Agora que já criamos, rodamos e vimos o resultado de nosso script, vamos entender o que aconteceu ali.

Primeiro, salvamos o arquivo como **home.php**, e devido a extensão .php, o servidor vai automaticamente chamar o PHP (o programa interpretador de código), para tratar aquele arquivo.

Dentro do arquivo temos uma mistura de HTML com PHP.
Se ainda não aprendeu HTML, é bem simples, estude:

- Curso HTML Progressivo

(você pode estudar enquanto estuda PHP também, é tranquilaço)

A primeira linha de texto vem do HTML.

Já a segunda, veio do servidor PHP pra página. No PHP é assim, a saída, os resultados, vão para um documento HTML (diferente de uma linguagem como C ou C++ por exemplo, que vão pro programa, pra tela de um software .exe).

Mas, sem dúvidas, o mais importante é a tag: *<?php ?>*
Abrimos ela com: <?php
E fechamos com: ?>

E aqui que vem o grande segredo: todo código que escrevermos nessa tag, o seu navegador vai chamar o servidor:
- Ei, servidor! Tem um código PHP aqui, interpreta aí pra mim e me envia o resultado.

Então o servidor lê seu script, que no caso é apenas o comando **echo** com um texto, que é enviado para ser exibido no documento HTML.

Dizemos então que o código PHP está embutido no código HTML.
Existem outros meios, como criar um outro documento só com código PHP, mas por hora vamos estudar pela maneira embutida, pois nossos scrips serão inicialmente simples e pequenos.

Muito importante: Após cada comando dado no PHP, escreva ponto-e-vírgula ao final dele: ;

Você vai errar muito isso, esquecendo o ;
É um dos erros mais comum que programadores PHP cometem (e de outras línguas também).

- ## Tag curta: <? >

Uma outra maneira de inserir PHP em uma página HTML, é com a tag <? >

A tag começa em: <?
E termina em: >

Tudo dentro destes dois símbolos será código PHP, que o navegador vai chamar o interpretador no servidor para renderizar.

Nosso exemplo ficaria:

```
<html>
 <head>
  <title>Curso PHP Progressivo</title>
 </head>
 <body>
```

```
  Olá mundo! To no HTML! <br/>
  <?
   echo "Hello, World! To no PHP!";
  ?>
 </body>
</html>
```

Tag de Script: <script>

Se você já estudou JavaScript, por exemplo, já usou bastante a tag <script>

```
<html>
 <head>
  <title>Curso PHP Progressivo</title>
 </head>
 <body>
  Olá mundo! To no HTML! <br/>
  <script language="php">
   echo "Hello, World! To no PHP!!";
  </script>
 </body>
</html>
```

Maneiras de usar a tag do PHP

Existem ainda outras maneiras de exibir código PHP, porém temos que fazer alguns ajustes no servidor, definir algumas instruções e informações num treco chamado diretivas php.ini

Não se estresse com isso. Sempre use: <?php ?>
Não tem erro nem incompatibilidade.

Você pode fazer também tudo na mesma linha:

```
<?php echo "Hello, World! To no PHP!";?>
```

Mas vamos preferir sempre assim:

```
<?php
   echo "Hello, World! To no PHP!";
   código;
   código;
?>
```

Acho melhor, mais legível e organizado fazer assim, ok?

Comandos de Saída (output): echo e print em PHP

Neste tutorial de nosso **Curso de PHP**, vamos aprender como usar os comandos *echo* e *print* para saídas simples em páginas HTML.

• Saída Simples (output) em PHP

O propósito da linguagem PHP, junto de outras como JavaScript, é deixar as páginas HTML mais dinâmicas.

Por exemplo, sempre que você entra no Facebook ou Twitter, você vê coisas diferentes, informações diferentes, imagens, vídeos e textos diferentes.

Ou seja, há uma constante comunicação e troca de informação entre o cliente (você, que visita o site) e os servidores do Facebook e Twitter.

A maneira mais simples, básica e sem dúvidas mais importante, de se comunicar é via texto, ou seja, quando o PHP envia para o documento HTML algum texto para ser exibido.

Foi exatamente isso que fizemos em nosso tutorial Olá mundo em PHP.
Vamos aprender melhor como fazer essas saídas simples.

▪ O comando *echo*

O *echo* (lê-se: eco, de ecoar mesmo), serve para o PHP fazer uma saída simples para uma página HTML.

No tutorial passado fizemos:

```
<?php
    echo "Olá,mundo!";
?>
```

E o resultado é o texto: "Olá mundo!", na tela, tudo na mesma linha.
Ou seja, ele exibe tudo que está entre as aspas duplas (também funciona com aspas simples, mas use ambas simples ou ambas duplas, nada de misturar, ok?)

O comando *echo* permite que você escreva também em várias linhas de código.

Por exemplo, se quisermos fazer:
"Olá,
mundo!"

Como proceder? Teste:

```
<?php
    echo "Olá,
            mundo!";
?>
```

O resultado continua "Olá mundo!", tudo na mesma linha.
O PHP é burrinho?

Claro que não, é que pro HTML a quebra de linha funciona como um espaço em branco.
Para fazer a quebra de linha, devemos usar a tag
 do HTML, assim:

```
<?php
    echo "Olá, <br />
        mundo!";
?>
```

E o resultado:
Olá,
mundo!

Você também pode colocar todo o código PHP na mesma linha e ainda sim fazer a quebra de linha:

```
<?php
    echo "Olá,<br />mundo!";
?>
```

O resultado é o mesmo! Teste e veja!

O comando *echo* também permite que você imprima vários blocos de texto, por exemplo:

```php
<?php
    echo "Olá", "mundo!";
?>
```

Basta você colocar cada pedaço entre aspas duplas ou simples, e separar esses blocos de texto (que se chamam *strings*, em computação) por vírgula.

Mas o resultado vai ser: Olámundo!
Tudo junto! Ou seja, tem que dar um espaço após o Olá:

```php
<?php
    echo "Olá ", "mundo!";
?>
```

Ou então um espaço antes de mundo:

```php
<?php
    echo "Olá", " mundo!";
?>
```

Bem simples, não?

O comando *print*

Print, em inglês, significa imprimir.
É uma outra maneira de chamar as saídas: você imprime algo.

No caso da *print*, que é bem semelhante a *echo*, ela imprime (escreve, exibe) uma string (texto) na tela:

Sim, uso bem semelhante da echo.
Mas há duas diferenças:

1. A *echo* não retorna nenhum valor, e a *print* retorna o número 1, e isso é importante para usarmos em expressões, coisas que vamos aprender mais adiante
2. A *echo* suporta várias informações ao mesmo tempo (strings, separadas por vírgula) enquanto a *print* só aceita uma por vez.

Por exemplo, você pode imprimir a sigla PHP assim:

```
echo "P ","H", "P";
```

Ou:

```
echo "PH", "P";
```

O resultado é o mesmo: tudo grudado (chamamos de concatenado).

Já via print só é possível fazer:

```
<?php
    print "PHP";
?>
```

Não tem como fazer: 'P', 'H', 'P' no print, não funciona.
No decorrer de nosso curso usaremos mais a *echo* mesmo.

Teste:

```
<?php
    print "P", "HP";
?>
```

E rode sua página. O que apareceu?

10 Exercícios de Saídas Simples (echo) em PHP

Agora que já aprendemos como enviar saídas simples para uma página HTML, via comando *echo*, vamos praticar um pouco o que aprendemos, fazendo alguns exercícios?

Exercícios de PHP

1. Frase na tela - Crie um script que que exiba num documento HTML a frase "O primeiro programa a gente nunca esquece!".

2. Etiqueta - Elabore um script que escreve seu nome completo na primeira linha, seu endereço na segunda, e o CEP e telefone na terceira.

3. Letra de música - Faça um script que mostre na tela uma letra de música que você gosta (proibido letras do Justin Bieber).

4. Mensagem - Escreva uma mensagem para uma pessoa de quem goste. Implemente uma página PHP que imprima essa mensagem, tire um print e mande pra essa pessoa. Diga que foi um vírus que algum hacker instalou em seu computador.

5. Ao site - Faça um programa que mostre na tela o que você deseja fazer usando seus conhecimentos de PHP.

6. Quadrado - Escrever um programa que mostre a seguinte figura:

```
XXXXX
X   X
X   X
X   X
XXXXX
```

7. Tabela de notas - Você foi contrato por uma escola pra fazer o sistema de boletim dos alunos. Como primeiro passo, escreva um script que produza a seguinte saída:

```
ALUNO(A)        NOTA
=========       =====
ALINE           9.0
MÁRIO           DEZ
SÉRGIO          4.5
SHIRLEY         7.0
```

8. Letra grande - Elabore um script para produzir na tela a letra P, de PHP Progressivo. Se fosse
‘L’, seria assim:

```
L
L
L
LLLLL
```

9. Menu - Elabore um script que mostre o seguinte menu na tela:
Cadastro de Clientes
0 - Fim
1 - Inclui
2 - Altera
3 - Exclui
4 - Consulta
Opção:

10. Pinheiro - Implemente um programa que desenhe um "pinheiro" na tela, similar ao abaixo.
Enriqueça o desenho com outros caracteres, simulando enfeites.

```
          X
         XXX
        XXXXX
       XXXXXXX
      XXXXXXXXX
     XXXXXXXXXXX
   XXXXXXXXXXXXX
  XXXXXXXXXXXXXXX
          XX
          XX
         XXXX
```

Estou esperando seu código-fonte nos comentários!

Tipos de Dados em PHP - Inteiros, Float (decimal), Booleano, String, Array, Objeto e outros

Neste tutorial de nosso **Curso de PHP**, vamos aprender sobre os principais tipos de dados que iremos usar em nosso estudo de PHP: tipos numéricos (inteiros e float), tipo booleano, string, array, objeto etc.

• Tipos numéricos: *Inteiros* e *Flutuantes*

O tipo de dado mais conhecido e utilizado é, sem dúvidas, o tipo numérico. Ou seja, números.

Computação vem de computar, que é o mesmo que contar.
Computador, programas e scripts nada mais são que contas, muitas contas são feitas por trás dos panos e algumas outras pelo programador, através de seus scripts PHP.

Vamos usar tipos numéricos do tipo **inteiro**:

- Números positivos: 12345, 21, 12...
- Números negativos: -12, -21, -100
- Número nulo: 0

Existe um tipo de número que é tratado diferente, são os *float* ou flutuantes, mais conhecidos como 'quebrados', são os que não são inteiros:

- Números flutuantes (decimais): 12.12 ; 1.2 ; 4.5 ; 1.99

Note uma coisa importante: ao invés de vírgula (R$ 1,99), usamos ponto: R$ 1.99
É assim em toda a computação, ok? Cuidado para não confundir.

- Tipo numérico octal (base 8): 02112 (começam por 0)
- Tipo numérico hexadecimal (base 16): 0x2112A (começam por 0x)
- Notação científica: 6.0e+23 (6 seguido de 23 zeros)

• Tipo lógico: *Booleano*

Certamente você já ouviu falar em código binário, que tudo em computação é 1 e 0.
E é. Na verdade, é voltagem alta e voltagem baixa, que as entranhas de uma máquina entende.

Outra maneira de entender isso, é com os valores *booleanos*.
Só existem dois valores booleanos:
TRUE (verdadeiro)
FALSE (falso)

Tudo que for 0, 0.0, string vazia, NULL, etc, é FALSE.
Demais números e valores dos tipos, é TRUE.

Esse conhecimento será muito importante na próxima seção, sobre testes condicionais, onde iremos avaliar se as coisas são verdadeiras ou falsas.

▪ Tipo *String*

Já utilizamos e falamos das strings, que nada mais são que textos.

- Pode ser um caractere: 'a', 'A', 'b' etc
- Podem ser vários, uma palavra, uma frase, ou até um livro de tanto texto
- Um número: '1', '2','3' etc
- Caractere especial: "@", " ", "&" etc
- Vazia: '' ou ""

Veja bem:
1 - é número
'1' - é string

Strings é tudo que está entre aspas simples ou entre aspas duplas.

Outros Tipos de Dados

- Array - um conjunto, um agrupamento de valores, organizados numericamente, que podem ser de tipos diferentes (inteiros, floats, array de strings etc). Teremos uma seção só para estudar arrays
- Objeto - entidade que tem suas próprias definições, informações e maneiras de agir. Aprenderemos isso na seção de classes e objetos em PHP
- Tipo recurso
- Tipo misto
- Tipo callback
- Tipo NULL - Variável não possui valor nenhum, é NULL, nulo

Não se estressde muito com esses outros tipos de dados, foi mais a título de informação, estudaremos eles detalhes no futoro, em nosso **curso de PHP**.

Variáveis em PHP: $variavel

Neste tutorial de nossa **apostila de PHP**, vamos aprender um dos conceitos mais importantes da linguagem: as variáveis e importante símbolo $.

Vamos aprender o que é uma variável, para que serve, como funciona, como declarar e, claro, como usar variáveis em PHP.

• Variável em PHP - O que é

Já entrou em um site de compras, colocou alguns itens no carrinho, depois saiu do site, mas quando voltou, estava tudo lá? Será que adivinharam o que você queria? Claro que não, né.

E no seu computador ou celular, já jogou algum game, coletou recursos, passou de fase, saiu do jogo e dias depois, ao voltar, viu tudo lá?

Claramente essas informações ficaram armazenadas em algum lugar. Alguém, algum código ou sistema, salvou os itens que você queria comprar, quem era seu personagem no jogo e até que fase chegou.

Depois, ao entrar no site ou abrir o jogo, pegou todos esses dados novamente.
E é aí que entram as **variáveis**.

Variáveis nada mais são que maneiras de armazenar dados, informações e acessar eles de uma maneira bem fácil, simples e rápida, no mesmo código (mais a frente, vamos aprender como armazenar no HD).

▪ Variável em PHP- Como declarar

Em termos simples e direto, variável é que um *símbolo*, para nós humanos, sabermos onde foi armazenada determinada informação, acessar e alterar ela sempre que quisermos.

Para declararmos (dizer ao PHP que queremos criar e usar uma variável), fazemos assim:

1. Iniciamos com símbolo de dólar: $
2. Escolhemos um nome, por exemplo, *variavel*, aí fica: $variavel

Vai ser difícil, mas evite nomes como: $a, $x ou $y

É sempre bom escolhermos nomes de variáveis que sejam fácil de entender que informação elas estão armazenando:

- $fase;
- $nome_personagem;
- $cor_roupa;
- $skinSecundario;
- $skinPrincipal;
- $raio
- $diametro
- $area

Note que só lendo o nome das variáveis é possível ter ideia de que tipo de informação ela está armazenando. Quando você criar scripts de centenas ou milhares de linhas, vai perceber o quanto isso é importante.

Regras para declarar uma variável em PHP

1. Deve começar com uma letra do alfabeto ou underline: _
2. Pode conter somente as letras de *a* até *z*, de *A* até *Z* e o _
3. Não pode conter espaço. Usamos o _ para separar
4. Nomes de variáveis são *case sensitive*, ou seja: *$variavel* é diferente de *$Variavel* e *$variAvel*

Variável em PHP - Como funciona

Na verdade, o computador (servidor), não tem locais com nome $level, $poder, $fase...
Tem locais de memória com nomes:
0x2112A
0x2112B
0X2112C

Mas...imagina só ter que ficar decorando os blocos de memória de um HD? Sério muito difícil.

Então o que acontece é o seguinte, você declara:
$fase = 5;

Então o PHP, esperto como ele só, faz a associação:
$fase vai estar relacionado ao bloco de memória 0x2112D
Sempre que ele usar $fase, vou entender como 0x2112D, mesma coisa.

Isso tudo por um motivo simples: para nós, humanos, é complicado trabalhar com números hexadecimais ou binários, imagine sua variável com valor 0101011101011 localizado no endereço 011100101 do computador? Horrível de ler.

Mas 2795 armazenado na variável $poder, é bem mais fácil, não acha?

É para isso que serve as variáveis, uma maneira mais fácil e simples de lidarmos com informações, em computação.

▪ Variável em PHP - Como usar com *echo*

Vamos ver um pouco de código PHP, com variáveis em ação?

No código abaixo, declaramos a variável $texto que recebe uma string: "Curso PHP Progressivo"
E depois imprime a variável, teste e veja o resultado:

```
<html>
 <head>
  <title>Curso PHP Progressivo</title>
 </head>
 <body>
  <?php
        $texto = 'Curso PHP Progressivo';
        echo $texto;
  ?>
 </body>
</html>
```

Agora teste com o código PHP:

```php
<?php
    $texto = 'Curso PHP Progressivo';
    echo "O melhor curso é o $texto";
?>
```

Qual o resultado?

O que aconteceu foi bem simples: ao se deparar com o símbolo $texto, o PHP não imprimiu "$texto" na sua tela, ao invés disso, foi lá no endereço de memória que essa variável aponta e pegou o que tinha lá "Curso PHP Progressivo" e imprimiu isso!

Funciona pra todo tipo de informação, como números:

```php
<html>
 <head>
  <title>Curso PHP Progressivo</title>
 </head>
 <body>
  <?php
    $idade = 18;
    $melhor_numero = 2112;
    echo "Eu tenho $idade de idade e meu número favorito é $melhor_numero";
  ?>
 </body>
</html>
```

No exemplo anterior declaramos e usamos duas variáveis.

Você pode usar variáveis para armazenar inteiros, float (decimais), arrays, string, booleanos, objetos, NULL e tipos de dado recurso. Ou seja, tudo.

De fato, praticamente em todo script PHP, existem variáveis!

É costume ainda declararmos uma variável no começo de um script, sem inicializar.
Variável declarada: $idade;
Variável declarada e inicializada: $idade = 18;

As variáveis que forem declaradas sem serem inicializadas, possuem o valor NULL, automaticamente, que representa simplesmente o NULO, nada.

Dizemos também que PHP é fracamente tipada, pois uma mesma variável pode mudar o tipo de dados que possui:

```
<html>
 <head>
  <title>Curso PHP Progressivo</title>
 </head>
 <body>
  <?php
        $dado = 'Joao';
        echo "Meu nome é $dado ";
        $dado = 20;
        echo "e tenho $dado anos de idade";
  ?>
 </body>
</html>
```

No exemplo acima a variável $dado começa recebendo uma string e depois armazena um inteiro.

Matemática em PHP - Operadores Aritméticos: Soma, Subtração, Multiplicação, Divisão e Resto da Divisão

Neste tutorial de nosso **curso de PHP**, vamos estudar um grupo de operadores muito importantes, os aritméticos, que usaremos para fazer cálculos matemáticos em PHP, como adição, subtração, produto, divisão e resto da divisão.

• Operador de Soma: +

De uma maneira geral, os operadores aritméticos são bem simples, pois lembram bastante os que já conhecemos e utilizamos no dia-a-dia.

Por exemplo, o de adição é o símbolo **+**.
Teste os comandos, fazendo um script PHP:
echo 1+1;
echo 21+12;

Note como o PHP funciona, por si só, como uma calculadora.
Ele imprime diretamente o resultado das operações aritméticas.

▪ Operador de Subtração: -

Assim como operador de somar, tem o de subtrair, de diferença: -

Teste os códigos e veja os resultados:
echo 1-1;
echo 21-12;

Se fizer: echo 21-12;
O resultado na ṕagina HTML é: 9

Agora faça: echo "21-12";
O resultado é: 21-12

Ou seja, no primeiro exemplo, ele calculou uma expressão matemática, aritmética.
No segundo exemplo, você enviou simplesmente uma string pra ele imprimir e o PHP imprimiu na tela.

"1" - é string
1 - é número

▪ Operador de Multiplicação: *

Diferente dos demais, esse o símbolo é um pouco diferente, não é x, como na escola, o símbolo do produto, de multiplicar, é o asterisco: **

Teste:
echo 2*2;
echo 3*3;

Você pode inclusive fazer usando variáveis em PHP.
Primeiro criamos duas: $numero1 e $numero2, e atribuímos valores a elas.

Depois, fazemos o produto ser armazenado na variável $resultado, veja:

```php
<?php
    $numero1 = 21;
    $numero2 = 12;
    $resultado = $numero1 * $numero2;
    echo "Produto: $resultado";
?>
```

▪ Operador de Divisão: /

Para realizar a divisão, utilizamos o símbolo da barra inclinada para a direita: ***/***
Teste:

echo 4 / 2;
echo 5 / 3;

Note que no primeiro exemplo o resultado é um inteiro: 2
Já no segundo exemplo, vai dar um valor quebrado, decimal: 1.6666667

Você pode fazer esses exemplos, usando variáveis, assim:

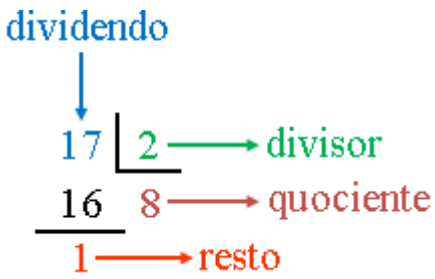
```php
<?php
    $var1 = 6;
    $var2 = 2;
    echo $var1/$var2;
?>
```

Note que o PHP é inteligente, já coloca o resultado da divisão de uma variável por outra: 3.

▪ Operador Resto da Divisão: %

Agora vamos precisar voltar um pouco ao ensino fundamental, quando fazíamos aquelas 'continhas' de dividir, lembram?

Era mais ou menos assim:



Tinha o dividendo, o divisor, o quociente e o resto.
O operador % serve para mostrar o resto da divisão de um número por outro, veja:

```
echo 4 % 2;
echo 4 % 3;
```

Os resultados vão ser 0 e 1, pois o resto da divisão de um número par por outro par (menor), é sempre 0. Já o resto da divisão de um par por um ímpar menor, é sempre 1.

O resto da divisão por 2, vai dar sempre 0 ou 1.
Vamos usar isso para criar um script que avalia se um número é par ou ímpar, por exemplo.

O resto da divisão por 3, vai dar sempre 0, 1 ou 2.

O resto da divisão por 4, vai dar sempre 0, 1, 2 ou 3.
O resto da divisão por n, vai dar sempre 0, 1, 2, ..., n-1 - entenderam a lógica?

Por exemplo, se quisermos gerar números de 1 até 10, basta fazer o resto da divisão por 10, aí teremos resultados de 0 até 9:
$numero % 10;

Agora, é só somar 1: 1 + ($numero % 10);
E teremos números de 1 até 10. Usaremos isso para gerar alguns números aleatórios.

Mas isso é em um futuro, não se preocupe por hora, apenas entenda a lógica e faça testes, testes e mais testes. Teste tudo que viu nessa página, muito mesmo.

Recebendo Dados e Informações do Usuário - input em PHP

Neste tutorial de nossa **apostila de PHP**, vamos aprender como pegar informações que o usuário fornece no HTML para o servidor, via PHP e usar estes dados para trabalharmos.

▪ Trocando informações - *input* em PHP

Quando você fornece o login e senha corretas, e clica em entrar em algum site, você entra.
Quando digita errado, cai numa tela de erro.

Quando tenta transferir um valor abaixo do que tem na conta do banco, tudo ok.
Já se tentar pagar um valor que não tem de saldo, nada ocorre, diz que não tem dinheiro suficiente.

Se segue alguém no Instagram e ela te segue de volta, você acessa de boa o perfil da pessoa.
Já se o perfil for trancado e ela não te seguir, você não visualiza o perfil dele.

E por aí vai.

O que estamos querendo dizer é: determinados tipos de ações só ocorrem dependendo da ação correta do usuário. Há uma troca de informações (login correto, por exemplo), entre o servidor e o cliente, e dependendo do tratamento que o PHP dá a essas informações, uma coisa ou outra é exibida na página em HTML.

É muito importante para o PHP receber dados dados do usuário, ir trocando informações, e é isso que vamos aprender neste tutorial: vamos digitar algo num formulário HTML e o PHP vai pegar essas informações, fazer o que tem que fazer com elas, e dar um retorno para o documento HTML.

▪ Receber dados via formulário em HTML e PHP

Vamos primeiro criar um formulário em HTML.
Inicialmente, possui só um campo de *input* para o usuário digitar algo e um botão de enviar:

```
<form action="home.php" method="get">
	Digite algo: <input type="text" name="formulario"><br>
	<input type="submit">
</form>
```

Agora o nosso código PHP:

```
<?php
	echo "Você digitou: ", $_GET["formulario"], "<br>";
?>
```

Nosso código completo fica:

```
<html>
 <head>
  <title>Curso PHP Progressivo</title>
 </head>
 <body>
	<form action="home.php" method="get">
		Digite algo: <input type="text" name="formulario"><br>
		<input type="submit">
	</form>

	<?php
		echo "Você digitou: ", $_GET["formulario"], "<br>";
	?>
 </body>
</html>
```

Quando você digita algo e clica em enviar, aparece o resultado lá embaixo:

O que acontece é o seguinte: quando o usuário clica no botão, o que tem escrito na input é enviado para a página "home.php" (mesma página que estamos trabalhando), e armazenado na variável $_GET["formulario"], onde *formulario* é o nome que demos ao nosso formulário.

Note que o que foi digitado aparece na URL da página.
Agora substitua *get* por *post* e *$_GET* por *$_POST*.

Teste sua página. E agora? Apareceu na URL?
Não se preocupe em entender isso, que são métodos *get* e *post*, estudaremos em detalhes mais pra frente em nossa **apostila de PHP**.

Por hora, guarde esse código, usaremos bastante para o PHP pegar dados que os visitantes do site vão digitar.

Calculadora Simples em PHP

Neste tutorial de nosso **curso de PHP**, vamos usar nossos conhecimentos de operações matemáticas com *input* do usuário para fazer uma calculadora simples.

Exercício de PHP - Calculadora Simples

"Crie um script em PHP que recebe dois números do usuário, via formulário input, e mostre a soma, diferença, produto, divisão do primeiro pelo segundo e resto da divisão do primeiro pelo segundo."

Vamos chamar de num1 e num2 os nomes dos formulários input. Armazenamos seus respectivos valores nas variáveis $num1 e $num2, que pegamos através do $_GET.

Nosso código fica:

```
<html>
 <head>
  <title>Curso PHP Progressivo</title>
 </head>
 <body>
   <form   action="home.php" method="get">
       Numero 1: <input type="text" name="num1"><br>
       Numero 2: <input type="text" name="num2"><br>
       <input type="submit">
   </form>

   <?php
       $num1 = $_GET['num1'];
       $num2 = $_GET['num2'];
       echo "Soma          : ","$num1+$num2 = ",$num1+$num2, "<br />";
       echo "Subtração     : ","$num1-$num2 = ",$num1-$num2, "<br />";
       echo "Multiplicação : ","$num1*$num2 = ",$num1*$num2, "<br />";
       echo "Divisão       : ","$num1/$num2 = ",$num1/$num2, "<br />";
       echo "Resto da divisão : ","$num1%$num2 = ",$num1%$num2, "<br />";
   ?>
 </body>
</html>
```

Precedência de Operadores em PHP

Neste tutorial de nosso **curso de PHP**, vamos aprender o que é a precedência de operadores matemáticos e como ela funciona em desenvolvimento web.

• Precedência: O que é?

Quanto é: 1+2*3 ?

Alguns podem fazer a multiplicação antes, e daria 7.
Outros fariam a soma primeiro, e o resultado seria 9.

"Ah, mas é só lembrar da Matemática da escola".

Sim, em parte sim. Porém, em programação PHP, temos diversos operadores, que não tem na escola, como incremento e decremento (vamos aprender melhor na seção de Laços e Loopings).

Mas, o computador não pode ter dúvidas, toda máquina deve resultar no mesmo valor como resultado de uma expressão matemática.

Visando deixar isso bem claro, existe uma precedência de operadores.

▪ Precedência de Operadores

Da maior (de cima), pra menor (indo pra baixo), a ordem que o PHP vai executar as operações é:

1. () : Parêntesis
2. ++ -- : Operadores de incremento e decremento
3. ! : Operador lógico
4. * / % : Operadores aritméticos
5. + - . : Operadores aritméticos e de string
6. << >> : Bitwise
7. < <= > >= <> : Operadores de comparação
8. == != === !== : Operadores de comparação

Não se estresse com os operadores que não conhece, você vai aprendê-los durante nosso curso, lá na frente, sem pressa.

Note que o operador mais poderoso é o de parêntesis.
No caso inicial de nosso tutorial:
1 + 2*3 = 1 + 6 = 7
(1+2)*3 = 3*3 = 9

Ou seja, sempre que tiver expressões grandes e complicadas, saia colocando parêntesis, até deixa mais legível seu código.

Nos exercícios propostos da seção Básico do PHP, vamos usar algumas expressões maiores (como calcular área de círculo, converter temperaturas etc).

- **Exercícios de Precedência de Operadores**

Digite nos comentários os seus resultados:

1. ((1 + 2) * 3 - 4) * 5
2. (6 - 7) * 8 * 9 + 10
3. (10 + 9 - 8 + 7) * 6

Como Comentar Código PHP - Área e Perímetro de um Cìrculo

Neste tutorial de nossa **apostila de PHP**, vamos aprender como comentar códigos em PHP e resolver uma questão, sobre área e perímetro de uma circunferência.

- **Comentários de uma linha: //**

Comentário é um...comentário. Não do tipo "olha que código bonito, que código bem feito, que código formoso", mas sobre o que o código PHP está fazendo.

Pode parecer bobagem agora, pois nossos scripts ainda são bem pequenos e triviais, mas a medida que você for se tornando profissional PHP, seus códigos terão dezenas, centenas e até milhares de linhas de código.

Podemos comentar uma linha de PHP de duas maneiras

```
<html>
 <head>
  <title>Curso PHP Progressivo</title>
 </head>
 <body>
      <form  action="home.php" method="get">
           Numero 1: <input type="text" name="num"><br>
           <input type="submit">
      </form>

      <?php
           // Pega o valor do formulário
           $num = $_GET['num'];

           // Imprimindo o número fornecido
           echo "Numero digitado: $num <br />";

           echo "Numero dobrado : ", 2*$num; //Dobra e printa
      ?>
 </body>
</html>
```

Os dois primeiros comentários ('Pega o valor do formulário', e 'Imprimindo número fornecido') foram feitos na linha acima do código que ela explica, com: //

No último comentário, comentamos após o código.
Essas barras servem pra comentar uma linha.

- ## Comentar várias linhas: /* */

É possível também comentar várias linhas ao mesmo tempo.
Tudo que estiver entre **/*** e ***/**, vai ser ignorado pelo interpretador do PHP, pois é isso que ele faz, ignora os comentários.

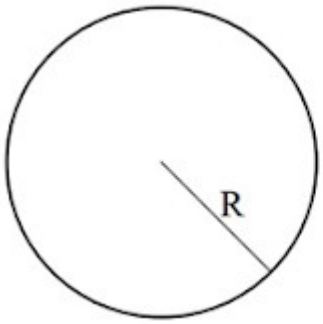
Veja:

```
<html>
 <head>
  <title>Curso PHP Progressivo</title>
 </head>
 <body>
        <form action=home.php method="get">
                Numero 1: <input type="text" name="num"><br>
                <input type="submit">
        </form>

        <?php
                /* Script que pega um numero
                 * e exibe seu quadrado.
                 * Auto: PHP Progressivo
                */
                $num = $_GET['num'];

                echo "Numero digitado: $num <br />";
                echo "Numero dobrado : ", 2*$num;
        ?>
 </body>
</html>
```

Os comentários serem apenas para um propósito: facilitar a vida do programador.
Você vai ver como isso é importante daqui uns anos, quando precisar rever e entender novamente um código que fez agora. Se ele estiver comentando, vai facilitar muuuuito mais.

▪ Cálculo de Área e Perímetro

As fórmulas para o cálculo do perímetro e área de uma circunferência são:



$$C = 2 . \pi . R$$

$$A_O = \pi . R^2$$

Nosso código fica assim, então:

```
<html>
 <head>
  <title>Curso PHP Progressivo</title>
 </head>
 <body>
        <form action=home.php method="get">
                Numero 1: <input type="text" name="num"><br>
                <input type="submit">
        </form>

        <?php
                // Pegando o valor digitado
                // pelo usuário
                $raio = $_GET['num'];

                // Imprime o raio
                echo "Raio: $raio<br />";

                // Exibe perímetro
                echo "Perímetro: ", 2*M_PI*$raio, "<br />";

                // Exibe Área
                echo "Área: ", M_PI*$raio*$raio, "<br />";
        ?>
 </body>
</html>
```

Exercícios Básicos de PHP

Parabéns! Você já estudou bastante.

Agora é hora de testar nossos conhecimentos básicos de PHP.

Tente resolver a lista. Tente muito. Vai quebrar a cabeça as vezes, ficar desestimulado, mas tente. Bastante mesmo, isso vai te fazer um programador melhor.

Lista de Exercícios: Básico do PHP

1.Escreva um programa que pede o raio de um círculo, e em seguida exiba o perímetro e área do círculo.
Para saber o valor do pi, use: M_PI (ele armazena o valor de pi)

2.Faça um Programa que peça um número e então mostre a mensagem O número informado foi [número].

3.Faça um script que peça dois números e imprima a soma.

4.Faça um script que peça as 3 notas de um aluno e mostre a média.

5.Faça um script que converta metros para centímetros.

6.Faça um script que calcule a área de um quadrado, em seguida mostre o dobro desta área para o usuário.

7.Faça um script que pergunte quanto você ganha por hora e o número de horas trabalhadas no mês. Calcule e mostre o total do seu salário no referido mês.

8.Faça um script que peça a temperatura em graus Farenheit, transforme e mostre a temperatura em graus Celsius.

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

9.Faça um script que peça a temperatura em graus Celsius, transforme e mostre em graus Farenheit.

10.Faça um script que peça 2 números inteiros e um número real. Calcule e mostre:

- •o produto do dobro do primeiro com metade do segundo .
- •a soma do triplo do primeiro com o terceiro.
- •o terceiro elevado ao cubo.

11.Tendo como dados de entrada a altura de uma pessoa, construa um script que calcule seu peso ideal, usando a seguinte fórmula: (72.7*altura) – 58

12.Tendo como dado de entrada a altura (h) de uma pessoa, construa um script que calcule seu peso ideal, utilizando as seguintes fórmulas:

- •Para homens: (72.7*h) - 58
- •Para mulheres: (62.1*h) – 44.7

13.João Papo-de-Pescador, homem de bem, comprou um microcomputador para controlar o rendimento diário de seu trabalho. Toda vez que ele traz um peso de peixes maior que o estabelecido pelo regulamento de pesca do estado de São Paulo (50 quilos) deve pagar uma multa de R$ 4,00 por quilo excedente. João precisa que você faça um programa que leia a variável peso (peso de peixes) e calcule o excesso. Gravar na variável excesso a quantidade de quilos além do limite e na variável multa o valor da multa que João deverá pagar. Imprima os dados do script com as mensagens adequadas.

14.

15.Faça um script que pergunte quanto você ganha por hora e o número de horas trabalhadas no mês. Calcule e mostre o total do seu salário no referido mês, sabendo-se que são descontados 11% para o Imposto de Renda, 8% para o INSS e 5% para o sindicato, faça um programa que nos dê:

- •salário bruto.
- •quanto pagou ao INSS.
- •quanto pagou ao sindicato.
- •o salário líquido.
- •calcule os descontos e o salário líquido, conforme a tabela abaixo:

```
+ Salário Bruto : R$
- IR (11%) : R$
- INSS (8%) : R$
```

- Sindicato (5%) : R$
= Salário Liquido : R$
Obs.: Salário Bruto - Descontos = Salário Líquido.

16.Faça um script para uma loja de tintas. O programa deverá pedir o tamanho em metros quadrados da área a ser pintada. Considere que a cobertura da tinta é de 1 litro para cada 3 metros quadrados e que a tinta é vendida em latas de 18 litros, que custam R$ 80,00. Informe ao usuário a quantidades de latas de tinta a serem compradas e o preço total.

17.Faça um script para uma loja de tintas. O programa deverá pedir o tamanho em metros quadrados da área a ser pintada. Considere que a cobertura da tinta é de 1 litro para cada 6 metros quadrados e que a tinta é vendida em latas de 18 litros, que custam R$ 80,00 ou em galões de 3,6 litros, que custam R$ 25,00.

- Informe ao usuário as quantidades de tinta a serem compradas e os respectivos preços em 3 situações:
- comprar apenas latas de 18 litros;
- comprar apenas galões de 3,6 litros;
- misturar latas e galões, de forma que o preço seja o menor. Acrescente 10% de folga e sempre arredonde os valores para cima, isto é, considere latas cheias.

18.Faça um script que peça o tamanho de um arquivo para download (em MB) e a velocidade de um link de Internet (em Mbps), calcule e informe o tempo aproximado de download do arquivo usando este link (em minutos).

Soluções

01. Resolvida no tutorial anterior

04.

Primeiro, criamos um formulário com três inputs, de nomes *num1, num2* e *num3*. No PHP, salvamos essas notas na variável $num1, $num2 e $num3.

Vamos declarar um variável que vai armazenar a média com a seguinte expressão: $media = ($num1 + $num2 + $num3)/3;

Veja o código HTML + PHP completo:

```
<html>
 <head>
  <title>Curso PHP Progressivo</title>
 </head>
 <body>
        <form action='home.php' method="get">
                Nota 1: <input type="text" name="num1"><br>
                Nota 2: <input type="text" name="num2"><br>
                Nota 3: <input type="text" name="num3"><br>
                <input type="submit">
        </form>

        <?php
                $num1 = $_GET['num1'];
                $num2 = $_GET['num2'];
                $num3 = $_GET['num3'];

                $media = ($num1 + $num2 + $num3)/3;

                echo "Média: $media";
        ?>
 </body>
</html>
```

Note que usamos parêntesis, forçando a soma das notas ser realizada antes, e só depois a divisão obtendo a média de 3 números.

08 e 09.

- **Fahrenheit para Celsius**

Dada uma temperatura F em Fahrenheit, ela vira Celsius (C) com a seguinte fórmula:

$$C = (F - 32) \cdot \frac{5}{9}$$

Nosso código fica:

```
<html>
 <head>
  <title>Curso PHP Progressivo</title>
 </head>
 <body>
        <form action='home.php' method="get">
                Temperatura em Celsius: <input type="text" name="C"><br>
                <input type="submit">
        </form>

        <?php
                $C = $_GET['C'];
                $F = (9/5)*$C + 32;

                echo "Em Farenheit: $F <br /><br />";
        ?>

 </body>
</html>
```

- **Celsius para Fahrenheit**

Manipulando a equação anterior, chegamos na temperatura de conversão de Celsius (C) para Fahrenheit (F):

$$F = C \cdot \frac{9}{5} + 32$$

Nosso código fica:

```
<html>
 <head>
  <title>Curso PHP Progressivo</title>
 </head>
 <body>
        <form action='home.php' method="get">
                Temperatura em Farenheit : <input type="text" name="F"><br>
                <input type="submit">
        </form>
        <?php
                $F = $_GET['F'];
                $C = ($F-32)*5/9;

                echo "Em Celsius: $C";
        ?>
 </body>
</html>
```

14.

```
<html>
 <head>
  <title>Curso PHP Progressivo</title>
 </head>
 <body>
        <form action=’home.php’ method="get">
                Quanto ganha por hora: <input type="text" name="pay"><br>
                Horas trabalhadas: <input type="text" name="hours"><br>
                <input type="submit">
        </form>

        <?php
                $salario = $_GET['pay'];
                $horas = $_GET['hours'];

                $bruto = $salario * $horas;
                $IR = 0.11 * $bruto;
                $INSS = 0.08 * $bruto;
                $sindicato = 0.05 * $bruto;
                $liquido = $bruto - $IR - $INSS - $sindicato;
                echo "Salario bruto: R$ $bruto <br />";
                echo "Imposto de Renda: R$ $IR <br />";
                echo "INSS: R$ $INSS <br />";
                echo "Sindicato: R$ $sindicato <br />";
                echo "Salário líquido: R$ $liquido <br />";
        ?>

 </body>
</html>
```

Teste Condicional

Nesta seção, iremos aprender um dos conceitos mais importantes da programação, o de *testes*.

Toda a computação é baseada em testes.
Testa se a senha está correta, se o login está certo, testa se a conexão da internet está ativa, testa se tem saldo em sua conta bancária para fazer um saque ou pagamento, o Youtube testa sua conexão de internet para saber que qualidade de vídeo vai enviar.

O Instagram testa se a pessoa que você segue te segue de volta, para exibir as fotos.
Os navegadores testam se você está acessando um site pelo computador, por um tablet ou celular, para exibir o formato correto.

Ao inserir seu CEP, os sites de lojas testam se tal endereço existem, testam o valor do frete, os games que você joga estão repletos de testes e mais testes. É tudo 0 ou 1, TRUE ou FALSE, lembra?

Desde redes sociais, pequenos sites, aplicativos, aviões modernos e tudo na NASA, tem teste condicional envolvido, que é o que vamos aprender nesta seção: a testar!

Operadores de Comparação: ==, ===, !=, !==, <>, >, >=, <, <=

Para podermos trabalhar com testes condicionais, precisamos primeiro conhecer os operadores de comparação:

1. ==
2. ===
3. !=
4. !==
5. <>
6. >
7. >=
8. <
9. <=

Operador de igual a: ==

Para compararmos se dois valores são iguais, usamos o comparador de igualdade: $a == $b

Se $a for igual a $b, esse teste retorna TRUE e se for diferente, retorna FALSE.

Teste:

```
<?php
	echo 3 == 1, "<br />";
	echo 3 == 2, "<br />";
	echo 3 == 3, "<br />";
?>
```

Note que quando o resultado da comparação é verdadeiro, o PHP imprime 1 e quando é falso, deixa nulo, que é o mesmo que FALSE.

Veja bem:
$a = 3 é uma atribuição, estamos atribuindo o valor 3 a variável $a
$a == 3 é uma comparação: $a é igual a 3? Verdadeiro ou Falso?

▪ Operador idêntico a: ===

$a === $b : retorna TRUE se os valores forem idênticos, e os tipos também. Falso se tiverem valores ou tipos diferentes.

Teste:

```
<?php
	echo 0=='0', "<br />";
	echo 0==='0';
?>
```

Veja que a primeira expressão deu 1, verdadeiro:
0 == '0'

Mas um é do tipo numérico e outro string, por isso 0 === '0' é falso.

▪ Operador diferente de: != ou <>

$a != $b
$a <> $b

O resultado dessas expressões só é TRUE quando elas tiverem valores diferentes.
Se forem iguais, a comparação retorna FALSE.

Teste:

```
<?php
	echo 0!=0, "<br />";
	echo 0<>0, "<br />";
	echo 0!=1, "<br />";
?>
```

▪ Operador de não idêntico: !==

O operador de diferente **!=** é o oposto do de igualdade: **==**

Já o operador que é o contrário de idêntico: **===** é **!==**
$a !== $b retorna verdadeiro se não forem idênticos (não tiverem mesmo valor ou não forem do mesmo tipo), e falso se forem idênticos
Teste:

```php
<?php
	echo 0!==0, "<br />";
	echo 0!=='0', "<br />";
	echo 0!==1, "<br />";
?>
```

- ## Operador maior que: >

Para $a > $b

O resultado da expressão é verdadeiro caso o valor em $a seja maior que o de $b.
Se $a for igual ou menor, vai retornar falso.

- ## Operador maior igual a: >=

Para $a >= $b

Retorna TRUE se o valor de $a for maior, ou igual a $b.
Retorna FALSE caso $a for menor que $b.

- ## Operador menor que: <

Para $a < $b

Essa expressão retorna TRUE se o valor de $a for estritamente menor que $b.
E retorna FALSE, caso o valor de $a seja maior que o de $b.

- ## Operador menor igual a: <=

Se fizermos $a >= $b

O resultado vai ser VERDADEIRO caso o valor de $a seja maior, ou até igual ao valor de $b.
Se $a for menor que $b, retorna FALSO.

Mais fonte de estudo:
http://php.net/manual/pt_BR/language.operators.comparison.php

Comando IF - Teste Condicional em PHP

Neste tutorial de nosso **curso de PHP**, vamos te apresentar o comando IF, usado para teste condicional e controle de fluxo.

• Teste Condicional em PHP: comando *if*

Até o momento, todos nossos scripts funcionam e rodam da mesma maneira, da primeira para última linha de comandos, sempre na mesma ordem, no mesmo fluxo.

Porém, no mundo real, não é assim que as coisas funcionam: algumas coisas são executadas, outras não, as vezes acontece isso e as vezes acontece aquilo, tudo dependendo de testes condicionais realizados.

Para alterar o controle de fluxo de um script, usamos o comando ***if*** em PHP, cuja sintaxe é:

```
<?php
        if(expressão){
                codigo
                codigo
                codigo
        }
?>
```

Ou seja, começamos com o comando ***if***, depois um par de parêntesis com alguma expressão dentro, que será testada (teste condicional).
Depois, um par de chaves: { }, com um código dentro.

Caso o teste condicional seja verdadeiro (TRUE), o código dentro do IF será executado.
Se for falso (FALSE), o código dentro do IF não será executado, o interpretador PHP pula todo esse bloco de código.

Exemplo de uso do IF

No código abaixo, pedimos a idade do usuário.

Depois, testamos se ela é maior que 17, se for, ele é maior de idade e exibimos a mensagem:
você já pode dirigir.

Se o valor fornecido pelo usuário for menor que 18, essa mensagem nunca será exibida:

```
<html>
 <head>
  <title>Curso PHP Progressivo</title>
 </head>
 <body>
        <form action="home.php" method="get">
                Sua idade:  <input type="number" name="age" /><br />
                <input type="submit" name="submit" value="Testar" />
       </form>
       <?php
            $idade = $_GET['age'];

            if($idade > 17){
                echo "Você já pode dirigir!";
            }
       ?>
 </body>
</html>
```

Você pode trocar: $idade > 17
Pelo teste: $idade >= 18
Note que é a mesma coisa, o importante é o teste ser TRUE, e para isso basta fornecer um número igual ou maior que 18.

Exemplo de uso do IF

O script a seguir, recebe a idade do usuário e se ela for menor que 65, avisa que não pode se aposentar ainda!

Veja:

```
<html>
 <head>
  <title>Curso PHP Progressivo</title>
 </head>
 <body>
        <form action="home.php" method="get">
                Sua idade:  <input type="number" name="age" /><br />
                <input type="submit" name="submit" value="Testar" />
        </form>
        <?php
                $idade = $_GET['age'];

                if($idade < 65){
                        echo "Você nao pode se aposentar!";
                }
        ?>
 </body>
</html>
```

Se você digitar 65, o teste condicional já passa a ser FALSO e o IF não é executado.

Exemplo de uso de IF com string

Podemos fazer comparação com outros tipos de dados, não só números. Por exemplo, o script a seguir pede uma senha ao usuário e só exibe a mensagem 'Entrando no sistema...' se ele acertar a senha:

```
<html>
 <head>
  <title>Curso PHP Progressivo</title>
 </head>
 <body>
        <form action="home.php" method="get">
                Senha:  <input type="text" name="senha" /><br />
                <input type="submit" name="submit" value="Testar" />
        </form>
        <?php
```

```
            $senha = $_GET['senha'];

            if($senha == 'php2112'){
                echo "Senha correta, entrando no sistema!";
            }
        ?>
 </body>
</html>
```

Analisando o código, você consegue descobrir qual a senha?
Sim, é 'php2112'. Tente escrever phP2112, entrou no sistema? Por que ?

Vamos fazer o contrário agora, exibir uma mensagem de erro quando o usuário errar a senha.
E quando ele erra? Sempre que diferente de 'php2112'...diferente te lembra algo?

Sim, pode usar o operador de comparação !=
Veja como fica o código:

```
<html>
 <head>
  <title>Curso PHP Progressivo</title>
 </head>
 <body>
        <form action="home.php" method="get">
                Senha:  <input type="text" name="senha" /><br />
                <input type="submit" name="submit" value="Testar" />
        </form>
        <?php
            $senha = $_GET['senha'];

            if($senha != 'php2112'){
                echo "Senha errada! Não vai conseguir hackear!";
            }
        ?>
 </body>
</html>
```

Agora substitua o operador != por <> e certifique-se que são iguais.

Seria bacana se tivesse uma mensagem para caso você acerte a senha e outra para caso você erre, não acha? É o que vamos fazer no próximo tutorial de nossa seção de testes condicionais em PHP.

Instruções IF e ELSE - Controle de Fluxo em PHP

Agora que já aprendemos o comando IF, vamos aprender a usar o seu par, o ELSE.

• A Instrução IF ELSE: Controle de Fluxo

Aprendemos a usar o comando IF, que realiza um teste condicional e só executa um bloco de código se tal expressão for TRUE.

Mas, e quando ela for FALSE? Nada ocorre?
É aí que entra o comando ELSE.

A declaração desse par de comandos é:

```
<?php
        if(expressao){
                [codigo]
                [codigo]
        }else{
                [codigo]
                [codigo]
        }
?>
```

Ou seja, se a expressão for VERDADEIRA, o bloco do IF é executado.
Se a expressão for FALSA, o bloco do ELSE é executado.

Pense em inglês.
IF em inglês é SE, ELSE significa SENÃO.

A instrução IF ELSE fica assim:

```
se (expressao){
  código;
}senão {
  código;
}
```

Se for verdade, faz uma coisa, senão for, faz outra.

Nossos scripts estão começando a ter um fluxo mais complexo e completo, se assemelhando a sistemas web reais.

Exemplo de uso do IF e ELSE

Crie um site que pede ao usuário sua idade. Se for de maior, diga que pode dirigir, senão diga que não pode ainda.

```
<html>
 <head>
  <title>Curso PHP Progressivo</title>
 </head>
 <body>
        <form action="home.php" method="get">
            Sua idade:  <input type="number" name="age" /><br />
            <input type="submit" name="submit" value="Testar" />
       </form>
       <?php
            $idade = $_GET['age'];

            if($idade >= 18){
                echo "Você já pode dirigir!";
            }else{
                echo "Você ainda não pode dirigir!";
            }
       ?>
 </body>
</html>
```

A novidade aí ficou por conta do ELSE, que executa sempre que o teste for FALSE.

Exemplo de IF e ELSE em PHP

Crie uma página que pergunta a idade do usuário. Se ele tiver 65 anos ou mais, avise que ele já pode se aposentar. Senão, avise que ele tem que esperar até os 65.

```
<html>
 <head>
  <title>Curso PHP Progressivo</title>
 </head>
 <body>
```

```
        <form action="home.php" method="get">
            Sua idade:  <input type="number" name="age" /><br />
            <input type="submit" name="submit" value="Testar" />
        </form>
        <?php
            $idade = $_GET['age'];

            if($idade < 65){
                echo "Você ainda não pode se aposentar";
            }else{
                echo "Você já pode se aposentar!";
            }
        ?>
    </body>
</html>
```

Como mostrado nos exemplos do tutorial anterior, é possível usar somente o comando IF, sem um ELSE. Porém, só podemos usar o ELSE com seu respectivo IF.

- **Exemplo de IF ELSE com strings**

Crie um sistema web que pede a senha do usuário. Se ele digitar a correta, avise que está entrando no sistema. Se errar, diga que digitou errado. A senha deve ser 'phpprogressivo'.

```
<html>
 <head>
  <title>Curso PHP Progressivo</title>
 </head>
 <body>
        <form action=home.php method="get">
                Senha:  <input type="password" name="passwd" /><br />
                <input type="submit" name="submit" value="Entrar" />
       </form>
       <?php
            $senha = $_GET['passwd'];

            if($senha == 'phpprogressivo')
                echo "Entrando no sistema...";
            else
                echo "Senha errada!";
       ?>
 </body>
</html>
```

Sempre que o código após o IF e/ou ELSE tiver apenas uma linha de comando, podemos suprimir o par de chaves { }.

IF e ELSE dentro de IF e ELSE - Estruturas Aninhadas

Agora que aprendemos como usar o IF e ELSE em PHP, vamos aprender como aninhar essas estruturas de controle, ou seja, colocar IF e ELSE dentro de IF e ELSE.

• IF e ELSE Aninhados

Em muitos sistemas web que você for trabalhar, vai ser necessário usar uma técnica chamada *aninhamento*, que nada mais é que colocar uma coisa dentro de outra.

Vamos refazer um script que fizemos, o do voto.
Crie um sistema Web que pergunte a idade do usuário. Se ele tiver menos de 16 anos, não pode votar.
Se tiver entre 16 e 18 é facultativo. Se tiver entre 18 ou mais, deve votar.

Note que nosso exemplo ficou mais complexo, pois agora vamos tratar 3 casos:

1. Quem tem menos de 16 anos
2. Quem tem mais de 16 e menos de 18
3. Quem tem 18 anos ou mais

Vamos começar com o primeiro IF, já eliminando quem tem menos de 16 anos:

```
if($idade < 16){
   ...
} else {
  ...
}
```

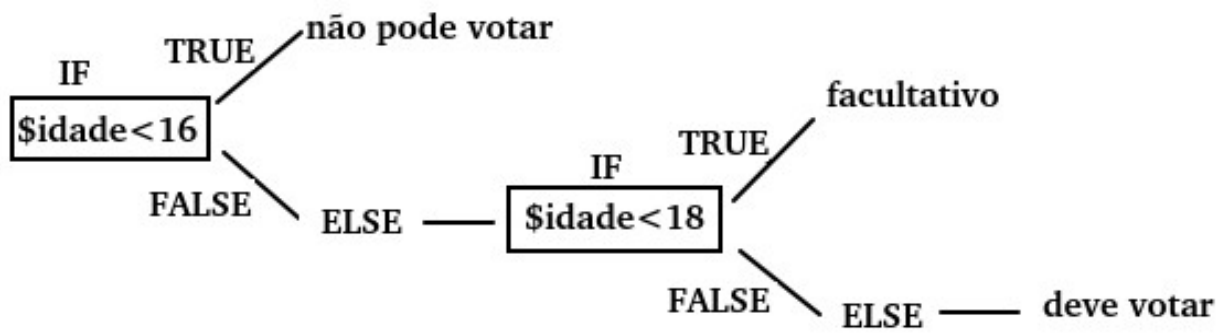
Tudo que for pro ELSE, é porque tem 16 anos ou mais. Dentro desse grupo, precisamos achar os que tem menos de 18 anos, para isso, criamos outro IF dentro desse ELSE, tratando os que tem menos de 18:

```
if($idade < 18)
  ...
else
```

...
Esse IF interno vai ter também seu ELSE interno, e o que cai nele é tudo que tiver 18 anos ou mais, concorda?

Nosso código fica, então:

```
<html>
 <head>
  <title>Curso PHP Progressivo</title>
 </head>
 <body>
        <form action=home.php method="get">
                Sua idade:  <input type="number" name="age" /><br />
                <input type="submit" name="submit" value="Testar" />
       </form>
       <?php
            $idade = $_GET['age'];

            if($idade < 16){
                echo "Você ainda não pode votar.";
            }
            else{
                if($idade < 18)
                    echo "Seu voto é facultativo";
                else
                    echo "Você pode votar";
            }
       ?>
 </body>
</html>
```



▪ Exemplo de IF e ELSE aninhados

Crie um sistema Web que pergunte a idade do usuário. Se ele tiver menos de 16 anos, não pode votar.
Se tiver entre 16 e 18 é facultativo. Se tiver entre 18 e 65 é obrigatório votar.
Se tiver mais de 65, também é facultativo.

Este é um exemplo de sistema real, pessoal. Já estamos programando coisas úteis, que estão usando de verdade por aí.

Vamos lá. Primeiro, vamos tratar o caso dos menores de 16 anos.
Se tiver menos, avisando que não pode votar.

Se tiver 16 ou mais, vai pro primeiro ELSE.
Agora vamos testar com um novo IF se ela tem menos de 18.
Se tiver, está entre 16 e 18 anos, e só vota se quiser.

Agora tem o ELSE desse IF aninhado, se cair nele, é porque a pessoa tem 18 anos ou mais.
Dentro desse ELSE vamos colocar um novo IF.

Nesse novo IF vamos perguntar se a pessoa tem menos de 65, se esse novo teste for TRUE, a pessoa tem de 18 até 65, e é obrigada a votar.
O ELSE desse IF é caso ela tenha 65 ou mais, aí ela não é obrigada a votar também.

Veja como ficou nosso código:

```
<html>
 <head>
  <title>Curso PHP Progressivo</title>
 </head>
 <body>
       <form action=home.php method="get">
                Sua idade:  <input type="number" name="age" /><br />
                <input type="submit" name="submit" value="Testar" />
      </form>
      <?php
            $idade = $_GET['age'];

            if($idade < 16)
```

```
            echo "Você ainda não pode votar.";
        else
            if($idade < 18)
                echo "Seu voto é facultativo";
            else
                if($idade<65)
                    echo "Seu voto é obrigatório";
                else
                    echo "Seu voto é facultativo";

    ?>
 </body>
</html>
```

Note que não usamos chaves nos nossos pares de IF e ELSE, e por um motivo que já explicamos: se abaixo dele tiver só uma instrução (um *echo*, um IF, um ELSE, um par IF e ELSE) não precisa de chaves.

Se não entendeu algo, pode perguntar.

Não tenha vergonha de perguntar ou ler de novo, de novo e de novo. No começo é complicado mesmo, programação PHP tem que ter paciência mesmo, é com calma e devagar que se aprende.

A instrução ELSEIF em PHP

Neste **tutorial de PHP**, vamos te apresentar a instrução ELSEIF, e veremos o que é, para que serve e onde devemos usar ela em desenvolvimento web.

A Instrução ELSEIF em PHP

Programadores, por natureza, são bichos preguiçosos.

E não, isso não é uma crítica. Bill Gates disse que adora gente preguiçosa, pois eles sempre vão procurar a maneira mais rápida, simples e óbvia de resolver as coisas. E isso é verdade.

No decorrer de seus estudos e trabalhos como desenvolvedor web com PHP, você vai deparar gente fazendo coisas com milhares de linhas de código...enquanto outros vão resolver com algumas dezenas. Claro que, quase sempre, quanto menos melhor.

Falando em preguiça, vamos te apresentar a instrução ELSEIF, que serve para substituir os:

```
...
else
    if
    ...
    else
        if
        ...
Por
...
elseif
    ...
elseif
        ....
else
        ....
```

Ou seja, apenas substituímos um bloco de:

```
ELSE
     IF
```

Por: ELSEIF

Isso evita que nosso código fique cada vez maior e mais deslocado para a direita.

Exemplo de uso de ELSEIF

Vamos pegar o tutorial anterior, sobre IF e ELSE aninhados.
O último código que fizemos nele foi:

```
<?php
 $idade = $_GET['age'];

 if($idade < 16)
  echo "Você ainda não pode votar.";
 else
  if($idade < 18)
   echo "Seu voto é facultativo";
  else
   if($idade<65)
    echo "Seu voto é obrigatório";
   else
    echo "Seu voto é facultativo";

?>
```

Veja como fica se substituirmos por ELSEIF:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <form action="home.php" method="get">
                Sua idade:  <input type="number" name="age" /><br />
                <input type="submit" name="submit" value="Testar" />
       </form>
       <?php
            $idade = $_GET['age'];

            if($idade < 16)
                echo "Você ainda não pode votar.";
            elseif($idade < 18)
```

```
            echo "Seu voto é facultativo";
        elseif($idade<65)
            echo "Seu voto é obrigatório";
        else
            echo "Seu voto é facultativo";
    ?>
 </body>
</html>
```

Note que deixamos os IF, ELSEIF e ELSE todos alinhados, bem mais bonitinho e organizado, não acha?

- ## Exemplo de uso de ELSEIF

Crie uma página que pergunta ao usuário que time ele torce, e mostre uma mensagem correspondente:
Corinthians - Vai timão!
Palmeiras - Não tem mundial!
Flamengo - Flamíííídia!
São Paulo - Bambiiii!
Athlético - Hum, Aflético!
Outro - Time pequeno, nem conheço.

Primeiro, faça sem uso do ELSEIF, depois com o uso do ELSEIF.

Sem usar o ELSEIF:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
      <form action=home.php method="get">
            Que time você torce:  <input type="text" name="time" /><br />
            <input type="submit" name="submit" value="Testar" />
    </form>
    <?php
        $time = $_GET['time'];

        if($time == 'Corinthians')
            echo "Vai Timão!";
        else
```

```
                    if($time == 'Palmeiras')
                        echo "Não tem mundial";
                    else
                        if($time == 'Flamengo')
                            echo "Flamíííídia";
                        else
                            if($time == 'São Paulo')
                                echo "Bambiiis";
                            else
                                if($time == 'Athlético')
                                    echo "Hum, afléético";
                                else
                                    echo "Time pequeno não conheço";
        ?>
 </body>
</html>
```

Notem como vai deslocando pra direita...e vai crescendo...fica horrível!

Cara dos outros programadores quando forem dar manutenção ou alterar algo em código assim:

Pessoal, o brasileirão série A tem 20 times. E a série B também.
Imagina você tendo que criar código para 40 times?

Usamos pra 5 e já ficou essa coisa horrenda.
Mas calma, vamos te ensinar o jeito certo de fazer as coisas.

Agora o código com ELSEIF:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
```

```
</head>
<body>
	<form action="home.php" method="get">
		Que time você torce:  <input type="text" name="time" /><br />
		<input type="submit" name="submit" value="Testar" />
	</form>
	<?php
		$time = $_GET['time'];

		if($time == 'Corinthians')
			echo "Vai Timão!";
		elseif($time == 'Palmeiras')
			echo "Não tem mundial";
		elseif($time == 'Flamengo')
			echo "Flamíííídia";
		elseif($time == 'São Paulo')
			echo "Bambiiis";
		elseif($time == 'Athlético')
			echo "Hum, afléético";
		else
			echo "Time pequeno não conheço";
	?>
</body>
</html>
```

Agora sim! Digno de um excelente desenvolvedor web, organizado, que estudou pelo **Curso PHP Progressivo**.

Comando SWITCH: Fazendo escolhas em PHP (break e default)

Neste tutorial de nosso **curso de PHP**, vamos aprender o que é, para que serve e como usar a instrução SWITCH do PHP.

• O Comando SWITCH em PHP

A instrução *switch* do PHP é muito útil quando temos uma variável (ou resultado de uma expressão) que pode assumir diversos valores, e queremos tratar cada uma dessas possibilidades, de uma maneira mais organizada.

É perfeitamente possível fazer tudo isso com uma porção de IF, ELSE e ELSEIF. Mas o IF só resulta em TRUE ou FALSE, as vezes o resultado que queremos tratar é outro (como um número, uma string etc).

De fato, fizemos isso em nosso tutorial de ELSEIF em PHP, no exemplo dos times de futebol, que vamos refazer usando SWITCH já já.

A declaração do comando *switch* é:

```
<?php
      switch($variavel){
        case valor1:
            [codigo]

        case valor2:
            [codigo]

        case valor3:
            [codigo]
      }
?>
```

Funciona assim...temos que passar algum valor pra switch(), dentro dos parêntesis.

Em seguida, o PHP vai testar esse valor com vários *cases*, ou seja, vai testar se é igual a *valor1*, a *valor2*, a *valor3*...e onde for igual, ele executa todo o código, somente dali pra baixo.

- **Exemplo de uso do SWITCH**

Crie uma página que pergunta ao usuário que time ele torce, e mostre uma mensagem correspondente:
Corinthians - Vai timão!
Palmeiras - Não tem mundial!
Flamengo - Flamíííídia!
São Paulo - Bambiiii!
Athlético - Hum, Aflético!
Outro - Time pequeno, nem conheço.

Use a instrução SWITCH

Bom, vamos lá, nosso código fica assim:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <form action="home.php" method="get">
                Que time você torce:  <input type="text" name="time" /><br />
                <input type="submit" name="submit" value="Testar" />
       </form>
       <?php
            $time = $_GET['time'];

            switch($time){
                case 'Corinthians':
                    echo "Vai Timão!";
                case 'Palmeiras':
                    echo "Não tem mundial";
                case 'Flamengo':
                    echo "Flamíííídia";
                case 'São Paulo':
                    echo "Bambiiis";
                case 'Athlético':
                    echo "Huum...aflético";
            }
       ?>
 </body>
</html>
```

Faça um teste. Escreva, por exemplo, Flamengo, o resultado vai ser: "FlamíííídiaBambiiisHuum...aflético"

Temos aí dois problemas.
O primeira era que era pra ser só 'Flamíííídia' o resultado, e não tudo isso.
O segundo problema é se outro time for digitado, era pra aparecer "Time pequeno, nem conheço".

Vamos ver como resolver isso.

▪ A instrução *break* em SWITCH no PHP

Quando definimos para que serve a *switch* dissemos que ele procura o *case* correspondente e executa o código DALI EM DIANTE.

Ao digitar 'Flamengo', o PHP acha o *case 'Flamengo'* no switch, executa seu código e o dos próximos *cases*.

O que queremos é que seja executado somente o *case* correto e pare, não execute mais nada.
Para isso, temos que usar a expressão ***break*** (quebrar, parar) ao final de cada case.

Veja como fica nosso código com o comando break:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <form action="home.php" method="get">
                Que time você torce:  <input type="text" name="time" /><br />
                <input type="submit" name="submit" value="Testar" />
       </form>
       <?php
            $time = $_GET['time'];

            switch($time){
                case 'Corinthians':
                    echo "Vai Timão!";
                    break;
                case 'Palmeiras':
```

```
                echo "Não tem mundial";
                break;
            case 'Flamengo':
                echo "Flamíííídia";
                break;
            case 'São Paulo':
                echo "Bambiiis";
                break;
            case 'Athlético':
                echo "Huum...aflético";
        }
    ?>
 </body>
</html>
```

- **O comando *default* no SWITCH em PHP**

Mas ainda não está completo, nosso código.

E se você digitar outro clube que não está aí, como Ceará ou Paraná, por exemplo?
Era pra aparecer "Não conheço time pequeno"...e aí, criar um *case* para cada outro clube possível?

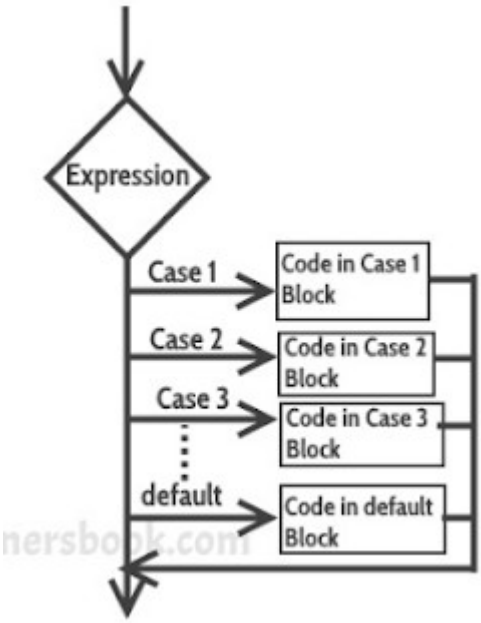
Óbvio que não, nem tem como, são muitas opções!
É aí que entra o **default**.

Tudo que não for tratado/captado por nenhum *case*, vai cair no *default*.

Veja como fica nosso código completo, funcionando perfeitamente, com instrução switch e comandos break e default:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
      <form action="home.php" method="get">
            Que time você torce:  <input type="text" name="time" /><br />
            <input type="submit" name="submit" value="Testar" />
    </form>
    <?php
```

```
        $time = $_GET['time'];

        switch($time){
            case 'Corinthians':
                echo "Vai Timão!";
                break;
            case 'Palmeiras':
                echo "Não tem mundial";
                break;
            case 'Flamengo':
                echo "Flamíííídia";
                break;
            case 'São Paulo':
                echo "Bambiiis";
                break;
            case 'Athlético':
                echo "Huum...aflético";
                break;
            default:
                echo "Não conheço time pequeno";
        }
    ?>
 </body>
</html>
```



PHP, Formulários SELECT e Comando SWITCH

Neste tutorial de nossa **apostila de PHP**, vamos aprender como usar o formulário do tipo SELECT junto com o PHP, fazendo uso do comando SWITCH.

• Exercício de PHP com formulário HTML

Crie uma página web que exibe um formulário do tipo SELECT (igual imagem abaixo), perguntando a linguagem de programação favorita do usuário.

Devem ter as opções:

- *PHP*
- *JavaScript*
- *C*
- *Python*
- *C++*
- *C#*
- *Ruby*

Após clicar em enviar, essa informação deve ir pro PHP, que deve retornar uma mensagem:

- *PHP é foda mesmo!*
- *JavaScript? Junto com PHP ela é perfeita!*
- *C eu acho difícil*
- *Python eu acho fácil e divertida*
- *Para todas as outras: Essa eu não conheço*

No PHP, use o comando SWITCH para tratar as opções.

▪ PHP com formulário SELECT do HTML

O código HTML é:

```
Qual a melhor linguagem, na sua opinião?<br />
<form method="post" action=ae>
	<select name="lingua">
		<option value="PHP">PHP</option>
		<option value="JS">JavaScript</option>
		<option value="C">C</option>
		<option value="Python">Python</option>
		<option value="CPP">C++</option>
		<option value="C#">C#</option>
		<option value="Ruby">Ruby</option>
	</select>
	<input type="submit" name="submit" value="Enviar"/>
</form>
```

Note que o nome do formulário é **lingua**. Vamos usar esse valor para pegar a opção selecionada pelo usuário.

Agora vamos para o PHP.
Primeira coisa que devemos fazer é pegar o valor selecionado, vamos fazer isso usando o $_POST e armazenando na variável $linguagem:

```
$linguagem = $_POST['lingua'];
```

Em seguida, vamos pegar essa variável e tratar no SWITCH, para exibir a mensagem correta.

Nosso código PHP + HTML, fica:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
	Qual a melhor linguagem, na sua opinião?<br />
	<form method="post" action="">
		<select name="lingua">
			<option value="PHP">PHP</option>
			<option value="JS">JavaScript</option>
			<option value="C">C</option>
			<option value="Python">Python</option>
			<option value="CPP">C++</option>
```

```
                <option value="C#">C#</option>
                <option value="Ruby">Ruby</option>
            </select>
            <input type="submit" name="submit" value="Enviar"/>
        </form>
        <?php
            $linguagem = $_POST['lingua'];

            switch($linguagem){
                case 'PHP':
                    echo "PHP é a mais foda mesmo!";
                    break;
                case 'JS':
                    echo "JavaScript? Junto com PHP é perfeita!";
                    break;
                case 'C':
                    echo "C eu acho difícil!";
                    break;
                case 'Python':
                    echo "Python é fácil e divertida!";
                    break;
                default:
                    echo "Não conheço essa!";
            }
        ?>
    </body>
</html>
```

Em cada <option>, veja que tem um atributo **value**, quando selecionamos uma opção, o formulário <select> vai retornar justamente esse valor. Ou seja, vai retornar "PHP", "JS", "C", "Python" etc.
Aí agora é só tratar essas strings com o SWITCH.

Note que houve uma comunicação entre a página HTML e o servidor PHP, poderíamos simplesmente ter armazenado essa opção do usuário, para fazer uma pesquisa de dados para alguma empresa.

Bacana e simples, esse PHP não?

Quantos Dias Possui o Mês (Script PHP)

Neste **tutorial de PHP**, vamos criar um script que diz quantos dias o mês possui, usando apenas lógica de programação e o comando SWITCH.

Exercício - Quantos dias o mês possui

Crie uma página WEB onde o usuário digita números de 1 até 12, representando o mês do ano e seu servidor PHP deve retornar o número de dias que possui nesse mês. Desconsidere os anos bissextos.

Sabemos que:
Meses com 31 dias: Janeiro (1), Março (3), Maio (5), Julho (7), Agosto (8), Outubro (10) e Dezembro (12)
Meses com 30 dias: Abril (4), Junho (6), Setembro (9) e Novembro (11)
Meses com 28 dias: Fevereiro (2)

A solução mais óbvia é simples:

```
case 1:
   echo "31 dias";
   break;
case 2:
   echo "28 dias";
   break;
...
case 12:
   echo "31 dias";
   break;
default:
   echo "Mês inexistente";
```

Porém nosso código ia ficar muito longo e vários cases seriam repetitivos

Script em PHP Otimizado

Lembra que falamos que se não usar o *break* os cases vão se acumulando? Pois é, podemos nos utilizar dessa funcionalidade para escrever um código melhor, mais limpo, veja:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
	<form action=ae method="get">
		Mês:  <input type="number" name="month" /><br />
		<input type="submit" name="submit" value="Testar" />
	</form>
	<?php
		$mes = $_GET['month'];

		switch($mes){
			case 1: case 3: case 5:
			case 7: case 8: case 10: case 12:
				echo "Mês $mes tem 31 dias";
				break;
			case 4: case 6:
			case 9: case 11:
				echo "Mês $mes tem 30 dias";
				break;
			case 2:
				echo "Mês $mes tem 28 dias";
				break;
			default:
				echo "Mês inexistente";
		}
	?>
 </body>
</html>
```

- ## Script em PHP - Melhor Solução

Não se assuste se você programar scripts grandes, extensos, confusos e longos...e aí vem um filho de uma mãe e resolve com pouquíssimas linhas.

Faz parte do aprendizado.
Vamos te mostrar uma maneira bem mais simples e menor de se resolver esse problema.
Usaremos a técnica de acumular novamente.

O número de dias vai ficar armazenado na variável $dias, inicializada com 0.
Primeiro, deixamos o case 2 lá embaixo, perto do *default*.

No código desse case, somamos 28 a $dias.

Como não usaremos *break*, sempre essa opção de case 2 vai acontecer, ou seja, sempre somaremos 28 na variável $dias.

Acima dela, vamos adicionar 2, para o resultado dar 30 nos meses 4, 6, 9 e 11.
E bem no começo, adicionamos 1, para dar resultado 31 nos meses restantes.

Veja como fica nosso código lindo e chuchu-beleza:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <form action=ae method="get">
                Mês de 1 até 12:  <input type="number" name="month" /><br />
                <input type="submit" name="submit" value="Testar" />
        </form>
        <?php
                $mes = $_GET['month'];
                $dias = 0;

                switch($mes){
                        case 1: case 3: case 5:
                        case 7: case 8: case 10: case 12:
                                $dias = $dias + 1;
                        case 4: case 6:
                        case 9: case 11:
                                $dias = $dias + 2;
                        case 2:
                                $dias = $dias + 28;
                }
                echo "Mês $mes possui $dias dias";
        ?>
 </body>
</html>
```

A expressão:
$dias = $dias + x

Significa que o novo valor da variável $dias é o anterior mais x.

Operadores Lógicos em PHP : && (AND), || (OR), ! (NOT) e XOR

Neste **tutorial de PHP**, vamos aprender o que são, para que servem e como usar os Operadores lógicos em PHP: &&, AND, ||, OR, !, NOT e o XOR.

Também conhecidos como *operadores booleanos*, os operadores lógicos são mecanismos que permitem agrupar diversas expressões, de modo a produzir um único resultado: TRUE ou FALSE, facilitando muuuito a vida do desenvolver web PHP.

Operador Lógico AND : &&

Suponha que tenhamos duas expressões/variáveis: **val1** e **val2**.
Se fizermos: **val1 && val2** ou **val1 AND val2**

O resultado vai ser:
TRUE: se val1 E val2 forem TRUE, as duas.
FALSE: se val1 ou val2, ou as duas, forem FALSE.

(2>1) && (1==1) : TRUE, as duas expressões são verdadeiras
(2>2) AND (1==1): FALSE, pois a primeira expressão é falsa

Operador Lógico OR: ||

Se AND em inglês significa E (expressão1 E expressão2), OR significa OU. Assim, se uma expressão for verdadeira OU a outra, toda a expressão será verdadeira:

Para: $a || $b
Será TRUE se $a for verdade OU $b for verdade OU todas forem verdade.
Será FALSE apenas se TODAS forem FALSAS.

Podemos substituir por $a OR $b.

Exemplos:
(2>1) || (2==2): TRUE, ambas são verdade
(1>2) OR (2==2): TRUE, a segunda é verdade

(1>2) OR (2==1): FALSE, pois todas são falsas

▪ Operador Lógico XOR: OR exclusivo

Um pouco semelhante ao operador OR (||), é o XOR.
Ele é chamado de OR Exclusivo.

Dissemos que no OR, a expressão resultante é verdadeira se QUALQUER uma das expressões forem verdadeiras, bastante uma ser (dentre 1 milhão, por exemplo).

O XOR resulta TRUE se, e somente se, UMA delas for verdadeira.
Se tiver duas ou mais, ou nenhuma expressão, verdadeira, ele vai resultar FALSE.

(2>1) XOR (1==2): TRUE, pois só a primeira é verdadeira
(2>1) XOR (1==1): FALSE< pois ambas são verdadeiras

▪ Operador Lógico NOT: !

Este operador muda o que é TRUE para FALSE, e FALSE para TRUE

!FALSE : TRUE
!1 : FALSE, pois 1 era TRUE
!'' : TRUE, pois string vazia é FALSE

A única diferença entre o && e o AND, bem comoo entre o || e o OR e entre ! e o NOT, é a precedência.

$$, || e ! tem precedência maior sobre AND, OR e NOT, por isso usaremos eles nos exemplos de nossa apostila;

Exercício usando Operadores Lógicos

Vamos refazer um exercício:

Crie um sistema Web que pergunte a idade do usuário. Se ele tiver menos de 16 anos, não pode votar.
Se tiver entre 16 e 18 é facultativo. Se tiver entre 18 e 65 é obrigatório votar.
Se tiver mais de 65, também é facultativo.

Solução:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <form action="" method="get">
                Sua idade:  <input type="number" name="age" /><br />
                <input type="submit" name="submit" value="Testar" />
        </form>
        <?php
                $idade = $_GET['age'];

                if( $idade<16)
                        echo "Não pode votar");
                elseif ($idade<18 || $idade>65)
                        echo "Seu voto é facultativo";
                else
                        echo "Seu voto é obrigatório";
        ?>
 </body>
</html>
```

A lógica solução está na página de IF e ELSE aninhados.

Note como a solução ficou menor, isso pois agrupamos duas expressões:

1. $idade < 18 (já é igual ou mair que 16)
2. $idade > 65

Pois ambas são de voto facultivo!

Viu como os operadores lógicos deixam nosso código menor, permitindo agrupar diversas expressões ao mesmo tempo?

Ano Bissexto - Como saber se é ? Scripts em PHP

Neste **tutorial de PHP**, vamos te mostrar o algoritmo em PHP que nos diz se um ano é bissexto ou não, usando operadores lógicos.

Algoritmo do Ano Bissexto em PHP

Normalmente, temos 365 dias no ano, onde o mês de fevereiro possui 28 dias.
Porém, a cada 4 anos adicionamos um dia extra, tendo fevereiro 29 dias e o ano 366 dias.

Existem uma lógica, um algoritmo, para saber se um ano vai ser bissexto ou não.
O algoritmo é o seguinte:

Todo ano múltiplo de 400, é sempre bissexto.
Exemplo: 2000, 1600, 1200, etc

OU:

Os anos múltiplos de 4 são bissextos, exceto se forem múltiplos de 100 e não forem de 400.
Exemplo: 1996, 2000, 2004, 2008, 2012, 2016 etc

Script em PHP: Ano Bissexto

Múltiplo de 400:
$ano % 400 == 0: Retorna TRUE

OU
Múltiplo de 4, exceto de 100:
($ano % 4 == 0 && $ano % 100 != 0): Retorna TRUE se for

Ou seja, se a expressão a primeira expressão acima for verdade OU a segunda, o ano é bissexto. Se não for TRUE, não será bissexto.

Nosso código HTML + PHP fica:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <form action="" method="get">
                Ano:  <input type="number" name="year" /><br />
                <input type="submit" name="submit" value="Testar" />
        </form>
        <?php
                $ano = $_GET['year'];

                if ( ($ano%4==0 && $ano%100!=0) || $ano%400 == 0)
                        echo "$ano é bissexto";
                else
                        echo "$ano não é bissexto";
        ?>
 </body>
</html>
```

- **Script em PHP: Usando o PHP para identificar anos bissextos**

O PHP nativo já possui diveeeersas funções e coisas feitas, prontas pra você simplesmente usar.
No caso, vamos usar a função **date()** ,que tem várias funcionalidades para trabalharmos com datas.

Não se preocupe, iremos estudar as datas mais adiante em nosso curso.
Script:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <form action="" method="get">
                Ano:  <input type="number" name="year" /><br />
                <input type="submit" name="submit" value="Testar" />
        </form>
        <?php
                $ano = $_GET['year'];
                $leap = date('L', mktime(0, 0, 0, 1, 1, $ano));

                if ($leap)
                        echo "$ano é bissexto";
                else
                        echo "$ano não é bissexto";
        ?>
 </body>
</html>
```

Operador Ternário em PHP ?:

Neste tutorial de nosso **curso de PHP**, vamos aprender o que é, para que serve e como usar o operador ternário ***?:***

O Operador Ternário - ?:

Em um tutorial passado mencionamos que programador é bicho preguiçoso. Mas não no sentido pejorativo.

No sentido de buscar sempre a solução mais simples, fácil, rápida e direta.
E acredite, acharam que:

```
if (condição)
   código;
else
   código;
```

Era escrever demais, então criaram o operador ternário **?:**
Cuja sintaxe é:

teste_condicional ? valor_se_TRUE : valor_se_FALSE;

Ou seja, fazemos um teste condicional, com uma expressão ou valor.
Se ele for verdadeiro, retorna o valor **valor_se_TRUE**
Se for falso, retorna o valor **valor_se_FALSE**

Exemplo de Operador Ternário ?: em PHP

Crie um script PHP que pede a idade de um usuário.
Se for menor de 18, diga que ele ainda não pode dirigir. Se for maior ou igual, que já pode dirigir.
Use operador ternário apenas.

Nosso código fica:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
       <form action="" method="get">
```

```
            Idade:  <input type="number" name="age" /><br />
            <input type="submit" name="submit" value="Testar" />
        </form>
        <?php
            $idade = $_GET['age'];
            echo ($idade<18) ? "Não pode dirigir" : "Já pode dirigir";
        ?>
 </body>
</html>
```

Bem mais curto, não?

▪ Exemplo de Operador Ternário ?: em PHP

Crie um script em PHP que pergunte a média final do aluno.
Se for menor que 7, avise que ele foi reprovado. Igual ou acima, ele passou direto.

Veja como fica nosso script HTML + PHP:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <form action="" method="get">
            Idade:  <input type="number" name="age" /><br />
            <input type="submit" name="submit" value="Testar" />
        </form>
        <?php
            $idade = $_GET['age'];
            echo ($idade<18) ? "Não pode dirigir" : "Já pode dirigir";
        ?>
 </body>
</html>
```

Lista de Exercícios de Testes Condicionais em PHP (IF, ELSE, ELSEIF e SWITCH)

1. Faça um Programa que verifique se uma letra digitada é vogal ou consoante.

2. Faça um programa que pede duas notas de um aluno. Em seguida ele deve calcular a média do aluno e dar o seguinte resultado:

A mensagem "Aprovado", se a média alcançada for maior ou igual a sete;
A mensagem "Reprovado", se a média for menor do que sete;
A mensagem "Aprovado com Distinção", se a média for igual a dez.

3. Faça um Programa que leia três números inteiros e mostre eles na ordem crescente.

2. Faça um programa que pede duas notas de um aluno. Em seguida ele deve calcular a média do aluno e dar o seguinte resultado:

A mensagem "Aprovado", se a média alcançada for maior ou igual a sete;
A mensagem "Reprovado", se a média for menor do que sete;
A mensagem "Aprovado com Distinção", se a média for igual a dez.

3. Faça um Programa que leia três números inteiros e mostre o maior deles.

4. Faça um Programa que leia três números inteiros, em seguida mostre o maior e o menor deles.

5. Faça um programa que pede dois inteiro e armazene em duas variáveis. Em seguida, troque o valor das variáveis e exiba na tela

6. Faça um Programa que leia três números e mostre-os em ordem decrescente.

7. Faça um Programa que pergunte em que turno você estuda. Peça para digitar M-matutino ou V-Vespertino ou N- Noturno. Imprima a mensagem "Bom Dia!", "Boa Tarde!" ou "Boa Noite!" ou "Valor Inválido!", conforme o caso.

8. As Organizações Tabajara resolveram dar um aumento de salário aos seus colaboradores e lhe contrataram para desenvolver o programa que calculará os reajustes.

Faça um programa que recebe o salário de um colaborador e o reajuste segundo o seguinte critério, baseado no salário atual:

salários até R$ 280,00 (incluindo) : aumento de 20%
salários entre R$ 280,00 e R$ 700,00 : aumento de 15%
salários entre R$ 700,00 e R$ 1500,00 : aumento de 10%
salários de R$ 1500,00 em diante : aumento de 5% Após o aumento ser realizado, informe na tela:
o salário antes do reajuste;
o percentual de aumento aplicado;
o valor do aumento;
o novo salário, após o aumento.

9. Faça um programa para o cálculo de uma folha de pagamento, sabendo que os descontos são do Imposto de Renda, que depende do salário bruto (conforme tabela abaixo) e 3% para o Sindicato e que o FGTS corresponde a 11% do Salário Bruto, mas não é descontado (é a empresa que deposita). O Salário Líquido corresponde ao Salário Bruto menos os descontos. O programa deverá pedir ao usuário o valor da sua hora e a quantidade de horas trabalhadas no mês.

Desconto do IR:
Salário Bruto até 900 (inclusive) - isento
Salário Bruto até 1500 (inclusive) - desconto de 5%
Salário Bruto até 2500 (inclusive) - desconto de 10%
Salário Bruto acima de 2500 - desconto de 20% Imprima na tela as informações, dispostas conforme o exemplo abaixo. No exemplo o valor da hora é 5 e a quantidade de hora é 220.

Salário Bruto: (5 * 220) : R$ 1100,00
(-) IR (5%) : R$ 55,00
(-) INSS (10%) : R$ 110,00
FGTS (11%) : R$ 121,00
Total de descontos : R$ 165,00
Salário Liquido : R$ 935,00

10. Faça um Programa que leia um número e exiba o dia correspondente da semana. (1-Domingo, 2- Segunda, etc.), se digitar outro valor deve aparecer valor inválido.

11.Faça um programa que lê as duas notas parciais obtidas por um aluno numa disciplina ao longo de um semestre, e calcule a sua média. A atribuição de conceitos obedece à tabela abaixo:

Média de Aproveitamento	Conceito
Entre 9.0 e 10.0	A
Entre 7.5 e 9.0	B
Entre 6.0 e 7.5	C
Entre 4.0 e 6.0	D
Entre 4.0 e zero	E

O algoritmo deve mostrar na tela as notas, a média, o conceito correspondente e a mensagem “APROVADO” se o conceito for A, B ou C ou “REPROVADO” se o conceito for D ou E.

12. Faça um Programa que peça os 3 lados de um triângulo. O programa deverá informar se os valores podem ser um triângulo. Indique, caso os lados formem um triângulo, se o mesmo é: equilátero, isósceles ou escaleno.

Dicas:
Três lados formam um triângulo quando a soma de quaisquer dois lados for maior que o terceiro;
Triângulo Equilátero: três lados iguais;
Triângulo Isósceles: quaisquer dois lados iguais;
Triângulo Escaleno: três lados diferentes;

13. Faça um programa que calcule as raízes de uma equação do segundo grau, na forma $ax^2 + bx + c$. O programa deverá pedir os valores de a, b e c e fazer as consistências, informando ao usuário nas seguintes situações:

Se o usuário informar o valor de A igual a zero, a equação não é do segundo grau e o programa não deve fazer pedir os demais valores, sendo encerrado;
Se o delta calculado for negativo, a equação não possui raizes reais. Informe ao usuário e encerre o programa;
Se o delta calculado for igual a zero a equação possui apenas uma raiz real; informe-a ao usuário;
Se o delta for positivo, a equação possui duas raiz reais; informe-as ao usuário;

PS: Para achar a raiz quadrada da variável x, faça: math.sqrt(x)

14. Faça um Programa que peça um número correspondente a um determinado ano e em seguida informe se este ano é ou não bissexto.
Ano bissexto em PHP

15. Faça um Programa que peça uma data no formato dd/mm/aaaa e determine se a mesma é uma data válida.

16. Faça um Programa que peça um número inteiro e determine se ele é par ou ímpar. Dica: utilize o operador módulo (resto da divisão): %. Depois peça um número e um múltiplo, e responda se o número digitado é múltiplo do segundo valor digitado.

17. Faça um Programa que leia um número inteiro menor que 1000 e imprima a quantidade de centenas, dezenas e unidades do mesmo. Observando os termos no plural a colocação do "e", da vírgula entre outros. Exemplo:

326 = 3 centenas, 2 dezenas e 6 unidades
12 = 1 dezena e 2 unidades Testar com: 326, 300, 100, 320, 310,305, 301, 101, 311, 111, 25, 20, 10, 21, 11, 1, 7 e 16

18. Faça um Programa para um caixa eletrônico. O programa deverá perguntar ao usuário a valor do saque e depois informar quantas notas de cada valor serão fornecidas. As notas disponíveis serão as de 1, 5, 10, 50 e 100 reais. O valor mínimo é de 10 reais e o máximo de 600 reais. O programa não deve se preocupar com a quantidade de notas existentes na máquina.

Exemplo 1: Para sacar a quantia de 256 reais, o programa fornece duas notas de 100, uma nota de 50, uma nota de 5 e uma nota de 1;
Exemplo 2: Para sacar a quantia de 399 reais, o programa fornece três notas de 100, uma nota de 50, quatro notas de 10, uma nota de 5 e quatro notas de 1.

19.Faça um Programa que peça um número e informe se o número é inteiro ou decimal. Dica: utilize uma função de arredondamento.

20. Faça um Programa que leia 2 números e em seguida pergunte ao usuário qual operação ele deseja realizar. O resultado da operação deve ser acompanhado de uma frase que diga se o número é:

par ou ímpar;
positivo ou negativo;
inteiro ou decimal.

21. Faça um programa que faça 5 perguntas para uma pessoa sobre um crime. As perguntas são:
"Telefonou para a vítima?"
"Esteve no local do crime?"
"Mora perto da vítima?"
"Devia para a vítima?"
"Já trabalhou com a vítima?" O programa deve no final emitir uma classificação sobre a participação da pessoa no crime. Se a pessoa responder positivamente a 2 questões ela deve ser classificada como "Suspeita", entre 3 e 4 como "Cúmplice" e 5 como "Assassino". Caso contrário, ele será classificado como "Inocente".

22. Um posto está vendendo combustíveis com a seguinte tabela de descontos:

Álcool: até 20 litros, desconto de 3% por litro
acima de 20 litros, desconto de 5% por litro

Gasolina:
até 20 litros, desconto de 4% por litro
acima de 20 litros, desconto de 6% por litro Escreva um algoritmo que leia o número de litros vendidos, o tipo de combustível (codificado da seguinte forma: A-álcool, G-gasolina), calcule e imprima o valor a ser pago pelo cliente sabendo-se que o preço do litro da gasolina é R$ 2,50 o preço do litro do álcool é R$ 1,90.

23. Uma fruteira está vendendo frutas com a seguinte tabela de preços:

	Até 5 Kg	Acima de 5 Kg
Morango	R$ 2,50 por Kg	R$ 2,20 por Kg
Maçã	R$ 1,80 por Kg	R$ 1,50 por Kg

Se o cliente comprar mais de 8 Kg em frutas ou o valor total da compra ultrapassar R$ 25,00, receberá ainda um desconto de 10% sobre este total. Escreva um algoritmo para ler a quantidade (em Kg) de morangos e a quantidade (em Kg) de maças adquiridas e escreva o valor a ser pago pelo cliente.

24. O Hipermercado Tabajara está com uma promoção de carnes que é imperdível. Confira:

	Até 5 Kg	Acima de 5 Kg
File Duplo	R$ 4,90 por Kg	R$ 5,80 por Kg
Alcatra	R$ 5,90 por Kg	R$ 6,80 por Kg
Picanha	R$ 6,90 por Kg	R$ 7,80 por Kg

Para atender a todos os clientes, cada cliente poderá levar apenas um dos tipos de carne da promoção, porém não há limites para a quantidade de carne por cliente. Se compra for feita no cartão Tabajara o cliente receberá ainda um desconto de 5% sobre o total a compra. Escreva um programa que peça o tipo e a quantidade de carne comprada pelo usuário e gere um cupom fiscal, contendo as informações da compra: tipo e quantidade de carne, preço total, tipo de pagamento, valor do desconto e valor a pagar.

Soluções

01.

Para saber se um caractere digitado é um dígito ou não, podemos usar (dentre outras opções), a função:

ctype_digit($minhaString);

Onde $minhaString é uma string qualquer.
Ela retorna TRUE se TODOS os caracteres forem dígitos.

Se ao menos algum não for, retorna FALSE.

Para resolver nosso exercício, basta digitar apenas um caractere:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <form action=ae method="get">
                Digite um caractere:  <input type="text" name="carac" /><br />
                <input type="submit" name="submit" value="Testar" />
        </form>
        <?php
                $meuCaractere = $_GET['carac'];
                if( ctype_digit($meuCaractere) )
                        echo "É dígito";
                else
                        echo "Nâo é dígito";
        ?>
 </body>
</html>
```

02.

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <form action=ae method="get">
                Nota 1:  <input type="text" name="nota1" /><br />
                Nota 2:  <input type="text" name="nota2" /><br />
```

```
            <input type="submit" name="submit" value="Testar" />
        </form>
        <?php
            $nota1 = $_GET['nota1'];
            $nota2 = $_GET['nota2'];
            $media = ($nota1 + $nota2)/2;

            echo "Média: $media <br />";
            if($media<7)
                echo "Reprovado";
            elseif($media==10)
                echo "Aprovado com Honra ao mérito";
            else
                echo "Aprovado";
        ?>
    </body>
</html>
```

03.

Antes de resolvermos esse script, precisamos aprender um algoritmo simples, de trocar o valor de duas variáveis.

Vamos supor que:
$a = 1;
$b = 2;

Queremos que fique:
$a = 2;
$b = 1;

O primeiro passo é faze $a receber valor de $b, né?
$a = $b;

Agora só fazer o contrário, fazer $b receber o valor de $a.
$b = $a;

Simples, não?
Não!

A segunda expressão tá errada, pois agora o valor de $a mudou pouco antes!

A solução pra isso é usar uma variável temporária.
O jeito certo de inverter valores é:
$temp = $a;
$a = $b;
$b = $a;

Prontinho, aquele valor de $a que tínhamos no começo fica armazenado na variável $tempo.
Depois, o $b pega o valor desse $temp

Agora é hora de resolver o exercício.
Vamos pegar três números de um formulário HTML.

Inicialmente, dizemos que o menor valor é o primeiro, o valor do meio é o segundo número e o maior número é o terceiro digitado:
$menor = $_GET['numero1'];
$meio = $_GET['numero2'];
$maior = $_GET['numero3'];

Agora vem a lógica.
Vamos pegar as variáveis duas a duas.

Primeiro pegamos as duas primeiras: $menor e $meio
Se o valor de $meio for menor que $menor, devemos inverter o valor delas!

Agora vamos comparar $menor e $maior
Se $maior for menor que $menor, invertermos as variáveis.

Por fim, comparamos $meio e $maior
Se $maior for menor que $meio, devemos inverter os valores.

Veja como ficou nosso código:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
       <form action="" method="get">
```

```
            Numero 1:  <input type="text" name="numero1" /><br />
            Numero 2:  <input type="text" name="numero2" /><br />
            Numero 3:  <input type="text" name="numero3" /><br />
            <input type="submit" name="submit" value="Testar" />
        </form>
        <?php
            $menor = $_GET['numero1'];
            $meio = $_GET['numero2'];
            $maior = $_GET['numero3'];

            if($meio < $menor){
                $temp=$menor;
                $menor=$meio;
                $meio=$temp;
            }

            if($maior < $menor){
                $temp=$menor;
                $menor=$maior;
                $maior=$temp;
            }

            if($maior < $meio){
                $temp=$meio;
                $meio=$maior;
                $maior=$temp;
            }

            echo "$menor < $meio < $maior";
        ?>
    </body>
</html>
```

04, 05 e 06.

Use a lógica do exercício anterior.

12.

Dizemos que três lados formam um triângulo quando a soma de qualquer dois lados é maior que o terceiro lado.

Estando os valores dos lados armazenados nas variáveis $a, $b e $c, testamos a condição de existência de um triângulo com o seguinte teste condicional:

```
if( ($a+$b>$c) && ($a+$c>$b) && ($b+$c>$a) ){
   //existe
}else
   echo "O triângulo não existe";
```

Agora, vamos usar a técnica dos IF e ELSE aninhados

O primeiro IF vai testar se todos os lados são iguais.
Se não forem, o primeiro ELSEIF testa se tem ao menos dois lados iguais.
Se nenhuma das condições acima forem satisfeitas, é porque ele é escaleno.

Veja como fica nosso script HTML + PHP:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <form action=ae method="get">
                Lado 1:  <input type="number" name="numero1" /><br />
                Lado 2:  <input type="number" name="numero2" /><br />
                Lado 3:  <input type="number" name="numero3" /><br />
                <input type="submit" name="submit" value="Testar" />
        </form>
        <?php
                $a = $_GET['numero1'];
                $b = $_GET['numero2'];
                $c = $_GET['numero3'];

                if( ($a+$b>$c)&&($a+$c>$b)&&($b+$c>$a) ){
                        if($a==$b && $b==$c)
                                echo "Triângulo equilátero";
                        elseif ($a==$b || $a==$c || $b==$c)
                                echo "Triângulo isósceles";
                        else
                                echo "Triângulo escaleno";
                }else
                        echo "O triângulo não existe";
        ?>
```

```
 </body>
</html>
```

13.

Para resolver uma equação do segundo grau, usamos a fórmula de Bháskara:

$$x = \frac{-b \pm \sqrt{\Delta}}{2.a}$$

$$\Delta = b^2 - 4.a.c$$

Onde esse símbolo do triângulo se chama delta (letra grega).
Se delta for 0, temos apenas uma raiz, real.
Se for maior que 0, temos duas raízes reais (uma com sinal + e outra com sinal -).
Se for menor que 9, temos duas raízes imaginárias (já estudou números complexos?)

Vamos começar a resolver o exercício.
Vamos pedir pelo HTML os três coeficientes a, b e c e armazenar respectivamente nas varáveis $a, $b e $c.

O primeiro teste é se $a é diferente de 0, se sim, continuamos no script.
Se for 0, cai no ELSE e já encerra o script.

Se for diferente, vamos ter três casos.
Primeiro tratamos para delta maior que 0, onde teremos duas raízes reais.
$raiz1 = (-$b + sqrt($delta))/(2*$a);
$raiz2 = (-$b - sqrt($delta))/(2*$a);

Se delta for 0, as duas raízes serão iguais:
$raiz1=$raiz2 = -($b)/(2*$a);

Se for menor, vamos ter que fazer algo mais complexo.

Como a raiz é imaginária, ela terá uma parte real ($real) e outra imaginária ($img)
E as raízes serão strings, pois temos que adicionar um *i* na resposta.

Para concatenar uma variável com uma string, usamos o símbolo de ponto .

Veja como ficou nossa solução completa:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <form action=ae method="get">
                a:  <input type="number" name="numero1" /><br />
                b:  <input type="number" name="numero2" /><br />
                c:  <input type="number" name="numero3" /><br />
                <input type="submit" name="submit" value="Testar" />
        </form>
        <?php
                $a = $_GET['numero1'];
                $b = $_GET['numero2'];
                $c = $_GET['numero3'];

                if($a != 0){
                        $delta = $b*$b - 4*$a*$c;
                        if($delta>0){
                                $raiz1 = (-$b + sqrt($delta) )/(2*$a);
                                $raiz2 = (-$b - sqrt($delta) )/(2*$a);
                        }elseif ($delta==0)
                                $raiz1=$raiz2 = -($b)/(2*$a);
                        else{
                                $real = -$b/(2*$a);
                                $img =  sqrt(-$delta)/(2*$a);

                                $raiz1 =  $real."+i".$img;
                                $raiz2 =  $real."-i".$img;
                        }

                }else
                        echo "Equação inexistente";

                echo "Raiz 1: $raiz1 <br />";
                echo "Raiz 2: $raiz2 <br />"
        ?>
```

```
 </body>
</html>
```

16.Para saber se um determinado valor é par ou ímpar em PHP, é bem simples.
E para isso, precisamos usar o operador Resto da Divisão: %

Todo número par possui uma característica especial: o resto da divisão por 2 é igual a 0.
Assim, nosso código fica:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <form action=ae method="get">
                Digite um número:  <input type="number" name="number" /><br />
                <input type="submit" name="submit" value="Testar" />
        </form>
        <?php
                $numero = $_GET['number'];

                if($numero % 2 == 0)
                        echo "$numero é par";
                else
                        echo "$numero é ímpar";
        ?>
 </body>
</html>
```

Paralelamente, todo número ímpar possui resto da divisão por 2 igual a 1.
Então podemos resolver este tutorial da seguinte forma:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <form action="" method="get">
                Digite um número:  <input type="number" name="number" /><br />
                <input type="submit" name="submit" value="Testar" />
        </form>
        <?php
```

```
            $numero = $_GET['number'];

            if($numero % 2 == 1)
                  echo "$numero é ímpar";
            else
                  echo "$numero é par";
      ?>
 </body>
</html>
```

Sobre os múltiplos...dizemos que um número **x** é múltiplo de outro **M**, quando o resto da divisão de *x* por *M* é 0.
Ou seja, quando:
(x % M = 0)

Por exemplo, 9 é múltiplo de 3, pois: 9 % 3 = 0
16 é múltiplo de 4, pois: 16 % 4 = 0
Em outras palavras, quando dividimos M por x, o resultado dá exato, inteiro, não dá quebrado (decimal, float).

O script PHP que pede um número para saber se é múltiplo de outro, é:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
      <form action="" method="get">
            Digite um número:  <input type="number" name="number" /><br />
            Testar se é múltiplo de:  <input type="number" name="mult" /><br /
>
            <input type="submit" name="submit" value="Testar" />
      </form>
      <?php
            $numero = $_GET['number'];
            $multiplo = $_GET['mult'];

            if($numero % $multiplo == 0)
                  echo "$numero é múltiplo de $multiplo";
            else
                  echo "$numero não é múltiplo de $multiplo";
      ?>
 </body>
</html>
```

Laços e Loopings

Laços e loopings, junto com testes condicionais, são, sem dúvidas, os pilares da programação, não só do PHP mas como de outras linguagens.

E por um motivo bem simples: eles nos permitem atingir objetivos executando rotinas várias e várias e várias vezes, até uma condição seja atingida. Ou mesmo rodar em *loopings* infinitos, rodando indefinidamente.

Uma das maravilhas do computador é a capacidade de executar milhares, milhões, bilhões...de vezes, alguma determinada tarefa, de modo a automatizar tarefas maçantes e altamente repetitivas.

Imagina ter que pedir, uma por uma, as notas de uma turma?
E se ela tiver tamanho variável ? Um script para cada tamanho?

Claro que não, vamos aprender a usar laços e loopings e aprender a repetir tarefas quantas vezes nós desejarmos.

Operadores de Atribuição: +=, -=, *=, /=, %=, .=, Incremento (++) e Decremento (–)

Neste tutorial de nosso **Curso de PHP**, vamos aprender tudo sobre os operadores de atribuição, que serão bastante utilizados em nossos tutoriais de laços e loopings, bem como serão vastamente usados por você no desenvolvimento web que for.

Operadores de Atribuição

Operadores de atribuição são aqueles que serão usados para atribuir um valor a uma determinada variável.

O operador mais comum e utilizado, é o **=**
Por exemplo:
$a = 5;
$b = $a;
Esse operador atribui o valor 5 na variável $a, e depois atribui o valor de $a na variável $b.

Bem simples e fácil, e certamente já está dominando isso.

Mas como dissemos várias vezes, programador é bicho preguiçoso e gosta de atalhos, escrever pouco, fazer rápido as coisas.

Operador de Soma: +=

Vamos supor que tenhamos uma variável:
$a = 1;

Se quisermos adicionar 1 nesse variável, fazemos assim:
$a = $a + 1;

Isso quer dizer o seguinte: o novo valor de $a é igual o anterior (1) somado com 1.
Agora, $a=2;

Ou seja, para incrementarmos uma variável $x de um valor y qualquer, fazemos:

$x = $x + y

Que é o mesmo que escrever:
$x += y;

- ## Operador de Subtração: -=

Se quisermos decrementar uma variável $x de um valor y, devemos fazer:
$x = $x - y;

Um atalho para isso é o operador de atribuição de subtração -=:
$x -= y;

- ## Operador de Multiplicação: *=

Se quisermos multiplicar uma variável $x por um valor y, podemos fazer:
$x = $x * y;

Ou de uma maneira mais simples, usando o operador de produto ***=**:
$x *= y;

- ## Operador de Divisão: /=

Para você dividir uma varável $x por um valor y, faça:
$x = $x / y;

De uma maneira mais simples e curta, você pode usar o operador de divisão ***/=***:
$x /= y;

- ## Operador de Resto de Divisão: %=

Para fazer uma variável $x assumir o seu resto da divisão por um número y, faça:
$x = $x % y;

De uma mais mais curta, usando o operador de atribuição de resto da divisão **%=**:
$x %= y;

▪ Operador de Concatenação: .=

Por fim (ufa), se quisermos concatenar (grudar) duas strings:
$a = "Curso ";
$b = "PHP Progressivo";

Basta fazer: $a = $a . $b (ponto final)
Ou simplesmente: $a .= $b

E o resultado vai ser: "Curso PHP Progressivo".

▪ Operador de Incremento ++ e Decremento --

Como dissemos, para incrementar uma variável de uma unidade, fazemos:
$x = $x + 1;

Ou: $x += 1;

E de maneiras mais simples ainda, apenas:
$x++;
++$x;

E para decremento:
$x--;
--$x;

Vamos ver agora a diferença entre essas duas versões.

▪ Pré e Pós Incremento/Decremento: $a++ e ++$a

Chamamos de pré-incremento ou pré-decremento o uso dos operadores:
++$a e --$a

Ele é pré, pois primeiro o valor de de $a é incrementado ou decrementado, e só depois é retornado.

Já o pós-incremento e pós-decremento é o seguinte uso dos operadores:
$a++ e $a--

São chamados assim pois primeiro o valor de $a é retornado, e só depois ocorre o incremento ou decremento.

Isso é melhor entendido quando usamos outra variável, $b, para receber essa atribuição.

Por exemplo, se $a=1 e fazemos:
$b = $a++;

O que ocorre é o seguinte: $b recebe o valor inicial de $a, que é 1. Só então, depois, $a é incrementado e passa a ser 2.

Agora, seja $a=1 e fizermos:
$b = --$a;

Primeiramente, $a é decrementando em uma unidade, e passa a ser 0.
Só então esse valor é atribuído a $b, que também será 0 portanto.

Laço WHILE em PHP

Neste tutorial de nossa **apostila de PHP**, vamos aprender o que é, para que serve e como usar o laço WHILE em PHP.

• Laço WHILE - O que é, para que serve?

Quando estudamos testes condicionais, vimos que os testes realizados eram executados apenas uma vez, e apenas uma coisa acontecia (ou IF ou ELSE).

Porém, muitas vezes é necessário fazer esses testes várias e várias vezes. Por exemplo, um botão de uma página fica o tempo inteiro perguntando:
'Fui clicado? Não, ok vou ficar quieto.'
'Fui clicado? Ainda não, ok'
'Fui clicado? Opa, agora sim, então devo chamar uma função do JavaScript'.

Temos que ficar várias e várias vezes fazendo vários testes.
E é aí que entra o laço WHILE.

Ele nada mais é que um teste condicional que se repete várias e várias vezes.

▪ Laço WHILE - Como usar

Vamos agora aprender a sintaxe do laço WHILE, em PHP.
Ele é da seguinte maneira:

```
<?php
	while( teste_condicional){
		codigo
		codigo
	}
?>
```

Quando o PHP chega no laço WHILE, ele vê o que tem dentro dele, entre parêntesis, e testa. Será alguma expressão ou algum valor.

Enquanto aquele teste resultar em TRUE, ele vai executar o código entre chaves.

Testa uma vez, se TRUE, executa. Testa de novo, se TRUE, executa de novo e assim vai.

O laço WHILE só vai parar quando o teste resultar FALSE, ok?

▪ Exemplo de Laço WHILE - Looping infinito

O exemplo mais simples e clássico de laço WHILE, é o Looping infinito.

É aquele que a condição que será testada vai ser sempre TRUE.
No exemplo abaixo, ela escreve "Curso PHP Progressivo" na tela...várias e várias vezes...infinitas, até travar seu navegado:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
       <?php
             while(TRUE)
                   echo "PHP Progressivo <br />";
       ?>
 </body>
</html>
```

Obtenha uma hospedagem, suba seu arquivo PHP com esse código e trolle um amigo mandando ele entrar no seu site.

Note que, assim como IF e ELSE, se seu código tiver apenas uma linha de comando, não é obrigatório o uso de chaves.

▪ Exemplo de WHILE: Contando até 10

O código abaixo pega uma variável, inicializa no 1, testa se é menor igual a 10.
Se for, imprime o número. Depois, incrementa e roda tudo de novo.

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
```

```
 <body>
      <?php
            $count=1;
            while($count<=10){
                  echo "$count <br />";
                  $count++;
            }
      ?>
 </body>
</html>
```

Esse simples script imprime no HTML os números de 1 até 10.
Teste com 100.
Teste com 1000.
Teste com 1 milhão, veja como o PHP é rápido e foda!

Tire o $count++, o que acontece?
Por que?

▪ Exemplo de WHILE: Contanto de 10 até 1

Vamos fazer o contrário agora.

Vamos começar de 10. Enquanto a variável for maior que 0, imprimir seu valor.
Dentro do código, também temos que decrementar a variável, senão ela nunca atinge o FALSE no teste condicional:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
      <?php
            $count=10;
            while($count>0){
                  echo "$count <br />";
                  $count--;
            }
      ?>
 </body>
</html>
```

▪ Exemplo de WHILE: Exibe o dobro

Crie um script que exibe os números 1, 2, 3..., 100 e ao lado, o dobro de seu valor.

Nossa variável de controle começa em 1 e vai até 100.
Enquanto for menor ou igual a 100, vai imprimir seu valor junto com o dobro.

Veja como fazer isso com laço WHILE:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
      <?php
            $count=1;
            while($count<=100){
                  echo "$count : ".(2*$count)."<br />";
                  $count++;
            }
      ?>
 </body>
</html>
```

▪ Exemplo de WHILE: Pedindo a Senha

Crie uma página que pede uma senha ao usuário e exiba "Se você não digitar a senha, seu navegador vai travar...".

Enquanto ele digitar uma senha errada ou não digitar nada, ficar escrevendo "Senha errada, sistema vai travar" na tela. Quando ele digitar a senha correta (que deve ser 'rush2112'), exibir "Entrando no sistema".

O teste condicional aqui é se o que for digitado pelo usuário é 'rush2112' ou não.
Se for, ok para. Se não, vai ficar executando e executando o WHILE pra sempre…

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <form action="" method="get">
                Senha:  <input type="text" name="password" /><br />
                <input type="submit" name="submit" value="Testar" />
        </form>
        <?php
                $senha = $_GET['password'];
                echo "Se você não digitar a senha, seu navegador vai travar...<br />
>";
                while($senha != 'rush2112'){
                        echo "Senha errada...Sistem vai travar<br />";
                }
                echo "Entrando no sistema...";
        ?>
 </body>
</html>
```

Laço DO WHILE em PHP

Neste tutorial de nosso **curso de PHP**, vamos aprender como usar este tal de *looping* DO WHILE.

Laço DO WHILE: O que é? Para que Serve ?

Quando estudamos o *looping* WHILE, deixamos uma coisa bem clara:
O código só era executado se o teste condicional dentro do WHILE fosse verdade.

Ou seja, primeiro o teste, só depois o código.
Se o teste for FALSE de cara, o interpretador PHP vai passar batido pelo laço WHILE e seu código, vai ver que a condição é falsa, não executa nada e é como se ele não tivesse existido.

Muitas vezes, queremos que o código rode, pelo menos uma vez, e é aí que entra o DO WHILE.
O Laço *do while* é bem semelhantes ao laço *while*, porém ele é pós condicionado ao invés de ser pré.

Se você precisar que a condição seja verificada de início, use WHILE.
Se você precisar que o teste condicional seja feito só ao final de cada looping, use DO WHILE.

Vamos entender melhor o que isso tudo significa.

Looping DO WHILE: Como usar ?

A sintaxe para usar, corretamente, o laço DO WHILE é:

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

Ou seja, primeiro o looping, só ao termino do código é que é feito o teste condicional.

Relembre o inglês. WHILE significa *enquanto*.

Enquanto a condição for verdadeira, executa. Ou seja, no *looping while*, só executa se o teste for verdadeiro.

Do significa *faça*.
Do while, *faça...enquanto*.

Note que tem um *faça*, primeiro. É porque a primeira iteração (primeira rodada do código) SEMPRE OCORRE, independente da condição.

São duas as diferenças do WHILE pro DO WHILE.

1. A primeira iteração do código sempre ocorre.
2. O WHILE é um laço pré-condicionado, primeiro a condição. DO WHILE é pós-condicionado, primeiro o código e só depois a condição.

▪ Exemplo de Código DO WHILE

O script simples abaixo mostra como a primeira iteração do laço é sempre executada.
Ela exibe uma mensagem na tela, mesmo a condição sendo falsa (0):

```
<?php
	do{
		echo "Condição falsa";
	}while(0);
?>
```

Note como ela executou apenas uma vez.

▪ Exemplo de DO WHILE: Contante de 1 até 100

Veja agora como fica a contagem de 1 até 100 em PHP, usando laço DO WHILE:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
	<?php
		$count=1;
		do{
			echo "$count <br />";
```

```
            $count++;
        }while($count<=100);
    ?>
 </body>
</html>
```

- **Exemplo de DO WHILE: Contar até onde o usuário quiser**

O script a seguir pergunta um número para o usuário.
Então, via *looping DO WHILE*, o PHP imprime na tela todos os números de 1 até o valor que o usuário pediu:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
    <form action="" method="get">
        Contar até:  <input type="text" name="count" /><br />
        <input type="submit" name="submit" value="Testar" />
    </form>
    <?php
        $final = $_GET['count'];
        $contador = 1;
        do{
            echo "$contador <br />";
            $contador++;
        }while($contador<=$final);
    ?>
 </body>
</html>
```

Laço FOR em PHP: O looping controlado

Neste tutorial de nossa **apostila de PHP**, vamos aprender a usar o laço mais famoso, o *looping FOR*, que é mais 'controlável', como veremos adiante.

Laço FOR: O que é e Como Usar

Nos tutoriais sobre os laços WHILE e DO WHILE, fizemos duas coisas em comum:
Inicializamos alguma variável
Modificamos essa variável dentro do escopo do laço

Essa variável é chamada de controle, e geralmente usamos a $count e é ela que vai no teste condicional. Isso é necessário para uma hora o *looping* terminar, algo tem que ser feito nessa variável, para ela ter um fim e chegar até onde desejamos.

Ou seja, temos uma pré condição (inicilizar), um teste condicional e uma pós condição (geralmente incrementar, decrementar ou fazer alguma mudança do tipo na variável de controle).

Pois bem, o laço FOR faz tudo isso de uma vez só, na declaração de uso do *looping*, facilitando e deixando nosso código muito mais simples e organizado.

Veja como usar o laço FOR em PHP

```
<?php
	for( inicializa ; teste_condicional ; modifica){
		codigo;
		codigo;
	}
?>
```

O laço for é o mais poderoso, mas também o mais complexo de usar, pois ele tem três expressões:

- Primeira: expressão de inicialização, ocorre antes ao inicializar o FOR, só uma vez
- Segunda: expressão condicional, teste para saber se o *looping* deve continuar a ocorrer ou não
- Terceira: expressão de modificação, após cada iteração do laço, essa expressão é executada

Separamos essas três expressões por ponto e vírgula ;
Vamos ver alguns exemplos para aprendermos na prática como usar o laço FOR.

- ## Exemplo de uso de Laço FOR

No script abaixo, vamos contar de 1 até 10 e exibir essa contagem em uma página HTML:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <?php
              for($count=1 ; $count<=10 ; $count++)
                    echo "$count <br />";
        ?>
 </body>
</html>
```

Note como ficou mais conciso. Tudo dentro de uma declaração do laço.

- ## Exemplo de uso de Laço FOR

Agora vamos fazer o contrário, contar de 10 até 1:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <?php
              for($count=10 ; $count>=1 ; $count--)
                    echo "$count <br />";
        ?>
 </body>
</html>
```

A única diferença é que inicializamos do 10 e decrementamos a variável de controle, para que possa se atingir o teste para este se tornar falso e encerrar o FOR. Bem mais simples, não?

▪ Exemplo de uso de Laço FOR

O script abaixo vai usar duas variáveis, $i e $j, inicialmente 1 e incrementadas ao final de cada iteração.
Vamos exibir o resultado da soma enquanto ele for menor que 10:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
       <?php
             for($i=1, $j=1 ; $i+$j<=10 ; $i++,$j++)
                   echo "$i + $j = ".($i+$j)."<br />";
       ?>
 </body>
</html>
```

Veja que é possível inicializar e modificar várias variáveis ao mesmo tempo, basta separar por vírgulas.

▪ Exemplo de uso de Laço FOR

Faça um script PHP que imprima a seguinte tabela, usando laço FOR:

1 10
2 9
3 8
4 7
5 6
6 5
7 4
8 3
9 2
10 1

É bem simples resolver usando duas variáveis, uma inicializada em 1 e outra em 10.

A cada iteração, a primeira é incrementada em 1 e a segunda decrementada em 2, veja:

```
<html>
```

```
<head>
 <title>Apostila PHP Progressivo</title>
</head>
<body>
      <?php
            for($i=1, $j=10; $i<=10 ; $i++,$j--)
                  echo $i.' '.$j.'<br />';
      ?>
</body>
</html>
```

O teste que usamos foi a primeira ser menor ou igual a 10.
Reescreva esse script, mas com o teste envolvendo a segunda variável apenas.

Tabuada com laços WHILE, DO WHILE e o FOR

Neste **tutorial de PHP**, vamos aprender como exibir qualquer tabuada que o usuário desejar, usando laços WHILE, DO WHILE ou o FOR.

Quando usar WHILE, DO WHILE ou FOR

Aprendemos três laços, em PHP
Laço WHILE
Laço DO WHILE
Laço FOR

E vimos como usar cada um deles, com exemplos de código.
Mas a maior dificuldade é em saber quando usar cada um: não existe melhor ou pior, use o que achar mais confortável.

O WHILE é, de longe, o mais simples. Executa um código sempre que uma condição for verdadeira.
Muito usado para *loopings* infinitos ocorrem indefinidamente. Se usa muito WHILE quando você NÃO sabe quando o laço vai terminar.

Se você não sabe se seu teste condicional vai ser validade ou não e quiser sempre que ao menos uma iteração ocorra, use o laço DO WHILE. Ele executa primeiro o código, uma vez pelo menos, e só depois faz o teste condicional.

O laço que mais iremos usar é, sem dúvidas, o FOR, pois é mais completo e poderoso.
Nele, de cara, já inicializamos nossas variáveis, criamos a expressão condicional e a de mudança.
Usamos ele quando sabemos quando começa e exatamente onde termina.

Mas, essencialmente, são todos a mesma coisa.

- **Tabuada com Laço WHILE**

O script abaixo pede um número ao usuário, e exibe a tabuada de tal número usando o laço WHILE:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <form action="" method="get">
                Tabuada do:  <input type="text" name="tab" /><br />
                <input type="submit" name="submit" value="Testar" />
        </form>
        <?php
                $tabuada = $_GET['tab'];
                $count=1;

                while($count<=10){
                        echo $tabuada." x ".$count." = ".($tabuada*$count)."<br />";
                        $count++;
                }
        ?>
 </body>
</html>
```

- **Tabuada com Laço DO WHILE**

O script abaixo pede um número ao usuário, e exibe a tabuada de tal número usando o laço DO WHILE:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <form action="" method="get">
                Tabuada do:  <input type="text" name="tab" /><br />
                <input type="submit" name="submit" value="Testar" />
        </form>
        <?php
                $tabuada = $_GET['tab'];
                $count=1;

                do{
```

```
            echo $tabuada." x ".$count." = ".($tabuada*$count)."<br />";
            $count++;
        }while($count<=10);
    ?>
 </body>
</html>
```

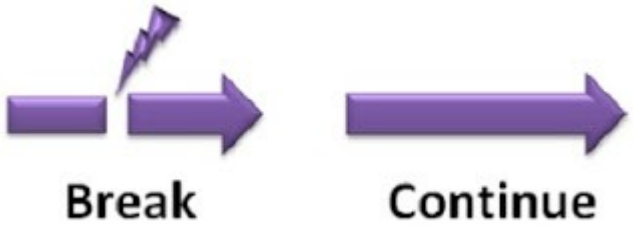
- **Tabuada com Laço FOR**

O script abaixo pede um número ao usuário, e exibe a tabuada de tal número usando o laço FOR:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
    <form action="" method="get">
        Tabuada do:  <input type="text" name="tab" /><br />
        <input type="submit" name="submit" value="Testar" />
    </form>
    <?php
        $tabuada = $_GET['tab'];
        for($count=1; $count<=10; $count++)
            echo $tabuada." x ".$count." = ".($tabuada*$count)."<br />";
    ?>
 </body>
</html>
```

Comando BREAK e CONTINUE em Laços

Neste tutorial de nosso **curso de PHP**, vamos te mostrar como usar os comandos BREAK e CONTINUEem laços e loopings em PHP.



• Comando BREAK: Interrompendo laços

Já estudamos o *break* quando estudamos o teste condicional SWITCH, e vimos que ele serve para um propósito bem simples: interromper o SWITCH.

Ele também tem a mesma propriedade em laços como WHILE, DO WHILE e o FOR: quando o interpretador PHP encontra o comando BREAK, ele termina na hora, sumariamente, o laço que estiver sendo executado, sem dar tempo pra fazer mais nada.

O script abaixo, é em um laço WHILE, do tipo *looping* infinito, pois ele executa pra sempre.
O contador inicia em 1 e vai incrementando de um em um.

Ele só vai parar quando entrar um número que é múltiplo de 12 e 21, ao mesmo tempo.
Qual? Quando? Não sabemos, mas quando chegar lá, identificamos o número e damos um BREAK pra interromper o WHILE, veja:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
      <form action= method="get">
            Tabuada do:  <input type="text" name="tab" /><br />
            <input type="submit" name="submit" value="Testar" />
      </form>
      <?php
            $count = 1;
```

```
            while(1){
                if($count%12==0 && $count%21==0){
                    echo "$count";
                    break;
                }
                $count++;
            }
        ?>
 </body>
</html>
```

Vamos supor que queiramos encontrar o primeiro múltiplo de 11 e 13, de 1 até 1 milhão, usando o laço FOR.

Vamos usar o comando BREAK para parar o laço, para que não seja necessário contar até 1 milhão:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <form action= method="get">
            Tabuada do:  <input type="text" name="tab" /><br />
            <input type="submit" name="submit" value="Testar" />
        </form>
        <?php
            for($count=1; $count<=1000000; $count++){
                if($count%11==0 && $count%13==0){
                    echo "Parei no $count";
                    break;
                }
            }
        ?>
 </body>
</html>
```

Note que quando o número for múltiplo de 11 e 13 ao mesmo tempo (não sei quando vai ser isso, o computador que calcule), o laço vai parar, exibindo tal número.

- **O comando CONTINUE: Pulando iterações em PHP**

O comando *break* tem uma peculiaridade: ele interrompe o laço, acaba com ele.Porém, muitas vezes não queremos acabar com o laço, e sim com a ITERAÇÃO, o *looping* do momento, queremos pular para a próxima 'rodada' de código do laço. Para isso, usamos o comando CONTINUE.
Quando o interpretador do PHP encontra o *continue;* no meio do caminho, ele encerra a iteração atual e pula pra próxima, inclusive executando a expressão de modificação.

Lembra daquela brincadeira do Silvio Santos, onde você tinha que ir contando os números e nos múltiplos de 4 falar PIM, assim:
1, 2, 3, PIM, 5, 6, 7, PIM, 9, 10, 11, PIM...

Basta colocarmos pra imprimir de 1 até 100.
Dentro do laço, colocamos um IF para verificar se é múltiplo de 4.
Se for, imprimimos PIM e damos um continue, para o laço continuar no próximo número, veja como fica nosso código:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
      <form action= method="get">
           Tabuada do:  <input type="text" name="tab" /><br />
           <input type="submit" name="submit" value="Testar" />
      </form>
      <?php
           for($count=1; $count<=100; $count++){
                if($count%4==0){
                     echo "PIM <br />";
                     continue;
                }
                echo "$count <br />";
           }
      ?>
 </body>
</html>
```

O PHP sempre iria ganhar esse desafio com o Silvio Santos...claro, com um programador que tivesse estudado pelo PHP Progressivo.

Exercícios

1.Faça um programa que peça uma nota, entre zero e dez. Mostre uma mensagem caso o valor seja inválido e continue pedindo até que o usuário informe um valor válido.
2.Faça um programa que leia um nome de usuário e a sua senha e não aceite a senha igual ao nome do usuário, mostrando uma mensagem de erro e voltando a pedir as informações.
3.Faça um programa que leia e valide as seguintes informações:
    a.Nome: maior que 3 caracteres;
    b.Idade: entre 0 e 150;
    c.Salário: maior que zero;
    d.Sexo: 'f' ou 'm';
    e.Estado Civil: 's', 'c', 'v', 'd';
Use a função **strlen(string)** para saber o tamanho de um texto (número de caracteres).

4.Supondo que a população de um país A seja da ordem de 80000 habitantes com uma taxa anual de crescimento de 3% e que a população de B seja 200000 habitantes com uma taxa de crescimento de 1.5%. Faça um programa que calcule e escreva o número de anos necessários para que a população do país A ultrapasse ou iguale a população do país B, mantidas as taxas de crescimento.
5.Altere o programa anterior permitindo ao usuário informar as populações e as taxas de crescimento iniciais. Valide a entrada e permita repetir a operação.
6.Faça um programa que imprima na tela os números de 1 a 20, um abaixo do outro. Depois modifique o programa para que ele mostre os números um ao lado do outro.
7.Faça um programa que leia 5 números e informe o maior número.
8.Faça um programa que leia 5 números e informe a soma e a média dos números.
9.Faça um programa que imprima na tela apenas os números ímpares entre 1 e 50.
10.Faça um programa que receba dois números inteiros e gere os números inteiros que estão no intervalo compreendido por eles.
11.Altere o programa anterior para mostrar no final a soma dos números.

12.Desenvolva um gerador de tabuada, capaz de gerar a tabuada de qualquer número inteiro entre 1 a 10. O usuário deve informar de qual numero ele deseja ver a tabuada. A saída deve ser conforme o exemplo abaixo:

a.Tabuada de 5:
5 X 1 = 5
5 X 2 = 10
...
5 X 10 = 50

Tabuada em PHP

13.Faça um programa que peça dois números, base e expoente, calcule e mostre o primeiro número elevado ao segundo número. Não utilize a função de potência da linguagem.
14.Faça um programa que peça 10 números inteiros, calcule e mostre a quantidade de números pares e a quantidade de números ímpares.
15.A série de Fibonacci é formada pela sequência 0,1,1,2,3,5,8,13,21,34,55,... Faça um programa capaz de gerar a série até o n−ésimo termo.
16.A série de Fibonacci é formada pela sequência 0,1,1,2,3,5,8,13,21,34,55,... Faça um programa que gere a série até que o valor seja maior que 500.
17.Faça um programa que calcule o fatorial de um número inteiro fornecido pelo usuário. Ex.: 5!=5.4.3.2.1=120
18.Faça um programa que, dado um conjunto de N números, determine o menor valor, o maior valor e a soma dos valores.
19.Altere o programa anterior para que ele aceite apenas números entre 0 e 1000.
20.Altere o programa de cálculo do fatorial, permitindo ao usuário calcular o fatorial várias vezes e limitando o fatorial a números inteiros positivos e menores que 16.
21.Faça um programa que peça um número inteiro e determine se ele é ou não um número primo. Um número primo é aquele que é divisível somente por ele mesmo e por 1.

22.Altere o programa de cálculo dos números primos, informando, caso o número não seja primo, por quais número ele é divisível.
23.Faça um programa que mostre todos os primos entre 1 e N sendo N um número inteiro fornecido pelo usuário. O programa deverá mostrar também o número de divisões que ele executou para encontrar os

números primos. Serão avaliados o funcionamento, o estilo e o número de testes (divisões) executados.

24.Faça um programa que calcule o fatorial de um número inteiro fornecido pelo usuário. Ex.: 5!=5.4.3.2.1=120. A saída deve ser conforme o exemplo abaixo:

a.Fatorial de: 5
5! = 5 . 4 . 3 . 2 . 1 = 120

25.O Departamento Estadual de Meteorologia lhe contratou para desenvolver um programa que leia as um conjunto indeterminado de temperaturas, e informe ao final a menor e a maior temperaturas informadas, bem como a média das temperaturas.

26.Os números primos possuem várias aplicações dentro da Computação, por exemplo na Criptografia. Um número primo é aquele que é divisível apenas por um e por ele mesmo. Faça um programa que peça um número inteiro e determine se ele é ou não um número primo.

27.Encontrar números primos é uma tarefa difícil. Faça um programa que gera uma lista dos números primos existentes entre 1 e um número inteiro informado pelo usuário.

28.Desenvolva um programa que faça a tabuada de um número qualquer inteiro que será digitado pelo usuário, mas a tabuada não deve necessariamente iniciar em 1 e terminar em 10, o valor inicial e final devem ser informados também pelo usuário, conforme exemplo abaixo:

a.Montar a tabuada de: 5
Começar por: 4
Terminar em: 7

Vou montar a tabuada de 5 começando em 4 e terminando em 7:
5 X 4 = 20
5 X 5 = 25
5 X 6 = 30
5 X 7 = 35
Obs: Você deve verificar se o usuário não digitou o final menor que o inicial.

29.Uma academia deseja fazer um senso entre seus clientes para descobrir o mais alto, o mais baixo, a mais gordo e o mais magro, para isto você deve fazer um programa que pergunte a cada um dos clientes da academia seu código, sua altura e seu peso. O final da digitação de dados deve ser dada quando o usuário digitar 0 (zero) no campo

código. Ao encerrar o programa também deve ser informados os códigos e valores do clente mais alto, do mais baixo, do mais gordo e do mais magro, além da média das alturas e dos pesos dos clientes

30. Um funcionário de uma empresa recebe aumento salarial anualmente: Sabe-se que:
    a. Esse funcionário foi contratado em 1995, com salário inicial de R$ 1.000,00;
    b. Em 1996 recebeu aumento de 1,5% sobre seu salário inicial;
    c. A partir de 1997 (inclusive), os aumentos salariais sempre correspondem ao dobro do percentual do ano anterior. Faça um programa que determine o salário atual desse funcionário. Após concluir isto, altere o programa permitindo que o usuário digite o salário inicial do funcionário.

31. Faça um programa que leia dez conjuntos de dois valores, o primeiro representando o número do aluno e o segundo representando a sua altura em centímetros. Encontre o aluno mais alto e o mais baixo. Mostre o número do aluno mais alto e o número do aluno mais baixo, junto com suas alturas.

32. Foi feita uma estatística em cinco cidades brasileiras para coletar dados sobre acidentes de trânsito. Foram obtidos os seguintes dados:
    a. Código da cidade;
    b. Número de veículos de passeio (em 1999);
    c. Número de acidentes de trânsito com vítimas (em 1999).
    Deseja-se saber:
    d. Qual o maior e menor índice de acidentes de transito e a que cidade pertence;
    e. Qual a média de veículos nas cinco cidades juntas;
    f. Qual a média de acidentes de trânsito nas cidades com menos de 2.000 veículos de passeio.

33. Faça um programa que receba o valor de uma dívida e mostre uma tabela com os seguintes dados: valor da dívida, valor dos juros, quantidade de parcelas e valor da parcela.
    a. Os juros e a quantidade de parcelas seguem a tabela abaixo:

Quantidade de Parcelas	% de Juros sobre o valor inicial da dívida
1	0
3	10
6	15
9	20

12 25

Exemplo de saída do programa:

Valor da Dívida Valor dos Juros Quantidade de Parcelas Valor da Parcela
R$ 1.000,00 0 1 R$ 1.000,00
R$ 1.100,00 100 3 R$ 366,00
R$ 1.150,00 150 6 R$ 191,67

34.Faça um programa que leia uma quantidade indeterminada de números positivos e conte quantos deles estão nos seguintes intervalos: [0-25], [26-50], [51-75] e [76-100]. A entrada de dados deverá terminar quando for lido um número negativo.

35.O cardápio de uma lanchonete é o seguinte:

a.Especificação Código Preço
Cachorro Quente 100 R$ 1,20
Bauru Simples 101 R$ 1,30
Bauru com ovo 102 R$ 1,50
Hambúrguer 103 R$ 1,20
Cheeseburguer 104 R$ 1,30
Refrigerante 105 R$ 1,00

Faça um programa que leia o código dos itens pedidos e as quantidades desejadas. Calcule e mostre o valor a ser pago por item (preço * quantidade) e o total geral do pedido. Considere que o cliente deve informar quando o pedido deve ser encerrado.

36.Em uma eleição presidencial existem quatro candidatos. Os votos são informados por meio de código. Os códigos utilizados são:

a.1 , 2, 3, 4 - Votos para os respectivos candidatos
(você deve montar a tabela ex: 1 - Jose/ 2- João/etc)
5 - Voto Nulo
6 - Voto em Branco

Faça um programa que calcule e mostre:

b.O total de votos para cada candidato;

c.O total de votos nulos;

d.O total de votos em branco;

e.A percentagem de votos nulos sobre o total de votos;

f.A percentagem de votos em branco sobre o total de votos. Para finalizar o conjunto de votos tem-se o valor zero.

37.Desenvolver um programa para verificar a nota do aluno em uma prova com 10 questões, o programa deve perguntar ao aluno a resposta de cada questão e ao final comparar com o gabarito da prova e assim calcular o total de acertos e a nota (atribuir 1 ponto por resposta certa). Após cada aluno utilizar o sistema deve ser feita uma

pergunta se outro aluno vai utilizar o sistema. Após todos os alunos terem respondido informar:

a. Maior e Menor Acerto;
b. Total de Alunos que utilizaram o sistema;
c. A Média das Notas da Turma.

Gabarito da Prova:

01 - A
02 - B
03 - C
04 - D
05 - E
06 - E
07 - D
08 - C
09 - B
10 - A

Após concluir isto você poderia incrementar o programa permitindo que o professor digite o gabarito da prova antes dos alunos usarem o programa.

38. Em uma competição de salto em distância cada atleta tem direito a cinco saltos. No final da série de saltos de cada atleta, o melhor e o pior resultados são eliminados. O seu resultado fica sendo a média dos três valores restantes. Você deve fazer um programa que receba o nome e as cinco distâncias alcançadas pelo atleta em seus saltos e depois informe a média dos saltos conforme a descrição acima informada (retirar o melhor e o pior salto e depois calcular a média). Faça uso de uma lista para armazenar os saltos. Os saltos são informados na ordem da execução, portanto não são ordenados. O programa deve ser encerrado quando não for informado o nome do atleta. A saída do programa deve ser conforme o exemplo abaixo:

Atleta: Rodrigo Curvêllo

Primeiro Salto: 6.5 m
Segundo Salto: 6.1 m
Terceiro Salto: 6.2 m
Quarto Salto: 5.4 m
Quinto Salto: 5.3 m

Melhor salto: 6.5 m
Pior salto: 5.3 m

Média dos demais saltos: 5.9 m

Resultado final:
Rodrigo Curvêllo: 5.9 m

39.Em uma competição de ginástica, cada atleta recebe votos de sete jurados. A melhor e a pior nota são eliminadas. A sua nota fica sendo a média dos votos restantes. Você deve fazer um programa que receba o nome do ginasta e as notas dos sete jurados alcançadas pelo atleta em sua apresentação e depois informe a sua média, conforme a descrição acima informada (retirar o melhor e o pior salto e depois calcular a média com as notas restantes). As notas não são informados ordenadas. Um exemplo de saída do programa deve ser conforme o exemplo abaixo:

Atleta: Aparecido Parente
Nota: 9.9
Nota: 7.5
Nota: 9.5
Nota: 8.5
Nota: 9.0
Nota: 8.5
Nota: 9.7

Resultado final:
Atleta: Aparecido Parente
Melhor nota: 9.9
Pior nota: 7.5
Média: 9,04

40.Faça um programa que peça um numero inteiro positivo e em seguida mostre este numero invertido.

a.Exemplo:

12376489
=> 98467321

41.Faça um programa que mostre os n termos da Série a seguir:

a. S = 1/1 + 2/3 + 3/5 + 4/7 + 5/9 + ... + n/m.
Imprima no final a soma da série.

42.Sendo H= 1 + 1/2 + 1/3 + 1/4 + ... + 1/N, Faça um programa que calcule o valor de H com N termos.

43.Faça um programa que mostre os n termos da Série a seguir:

a. S = 1/1 + 2/3 + 3/5 + 4/7 + 5/9 + ... + n/m.
Imprima no final a soma da série.

- **Soluções:**

01. e 02.

Validando entradas numéricas
Vamos pedir um número ao usuário.

Se for de 0 até 10, ok, dizemos que a nota foi inserida com sucesso.
Se for menos que 0 ou mais que 10, dizemos que houve um erro e exibimos uma mensagem pedindo uma nota correta, essa mensagem só sai se você inserir a nota corretamente:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
 <form action="" method="get">
  Insira uma nota:  <input type="number" name="number" /><br />
  <input type="submit" name="submit" value="Inserir" />
 </form>
 <?php
  $nota=$_GET['number'];

  if($nota<0 || $nota>10)
   echo "Insira uma nota de 0 até 10! <br />";
  else
   echo "Nota inserida com sucesso!";
 ?>
 </body>
</html>
```

Validando Entrada de Strings em PHP

Em muitos cadastros por aí, quando pedem seu login e senha, eles querem que você escolha uma senha que não seja igual ao seu login, por questões de segurança.

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
```

```
 <form action="" method="get">
  Nome: <input type="text" name="name" /><br />
  Senha: <input type="password" name="password" /><br />
  <input type="submit" name="submit" value="Inserir" />
 </form>
 <?php
  $nome=$_GET['name'];
  $senha=$_GET['password'];

  if($nome == $senha)
   echo "Sua senha deve ser diferente do login<br />";
  else
   echo "Cadastro realizado com sucesso";
 ?>
 </body>
</html>
```

Assim, o código de um sistema que não permita que a senha seja igual ao login, seria:

Exercício
Peça o login, a senha e a confirmação da senha.
Confirme se o usuário digitou corretamente a senha.

Código:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <form action=ae method="get">
                Nome: <input type="text" name="name" /><br />
                Senha: <input type="password" name="password" /><br />
                Confirme a senha: <input type="password" name="repassword"
/><br />
                <input type="submit" name="submit" value="Inserir" />
        </form>
        <?php
                $nome=$_GET['name'];
                $senha=$_GET['password'];
                $resenha=$_GET['repassword'];

                if($nome!=$senha && $senha==$resenha)
```

```
                echo "Cadastro realizado com sucesso";
            else
                echo "ERRO. Login igual senha ou senhas não batem!";
        ?>
    </body>
</html>
```

04. e 05.

Seja $A o valor da população da cidade A e $B o da B.
Seja $tA a taxa de crescimento da população da cidade A e $tB o da B.

Primeira coisa que temos que notar é que o usuário vai inserir a taxa em porcentagem, então precisamos dividir por 100 pra ter a taxa correta.

Após o primeiro ano, cada população vai ser:
$A = $A * (1 + $tA);
$B = $B * (1 + $tB);

Isso vai ocorrer enquanto o tamanho de $A seja menor a $B.
Quando for igual ou superior, o laço DO WHILE vai parar, e exibimos o valor da variável de controle, que está contando os anos:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <form action=ae method="get">
                População da cidade A: <input type="number" name="A" /><br />
                População da cidade B: <input type="number" name="B" /><br />
                Taxa de crescimento da cidade A (em %): <input type="text"
name="tA" /><br />
                Taxa de crescimento da cidade B (em %): <input type="text"
name="tB" /><br />
                <input type="submit" name="submit" value="Inserir" />
        </form>
        <?php
                $A=$_GET['A'];
                $B=$_GET['B'];
                $tA=$_GET['tA'];
                $tB=$_GET['tB'];
                $count=1;
```

```
            $tA /= 100;
            $tB /= 100;

            do{
                $A *= (1+$tA);
                $B *= (1+$tB);

                echo "Após $count anos:<br />";
                echo "População de A: $A <br />";
                echo "População de B: $B <br />";

                $count++;
            }while($A<$B);

            echo "Em $count anos a população de A atinge a de B";
        ?>
    </body>
</html>
```

Fizemos com que a cada ano, fosse mostrado o número de habitantes da cidade A e da cidade B, para você ver como só vai ultrapassar no tempo certo.

Isso permite você verificar se a solução está correta.
Deixamos a validação se a entrada está correta ou não com você.
Dica: Cheque se a população é um número positivo de pessoas, e a população de A deve ser menor e ter taxa de crescimento maior, do contrário nunca alcançará B.

07. e 08.

Para descobrirmos o maior número, vamos supor que o maior é o primeiro:

$maior = $n1;

Agora, basta ir comparando com cada um dos outros.
Se algum deles for maior que $maior, fazemos $maior assumir tal valor.

Assim, ao final, teremos comparado o valor de todos os números e o maior número estará armazenado na variável $maior, bastante imprimir.

Para calcular a soma é bem simples e óbvio.

A média, usamos a variável de $soma.

Veja como ficou nosso script que detecta o maior, a soma e a média de uma lista de 5 números.

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <form action=ae method="get">
                Número 1: <input type="number" name="n1" /><br />
                Número 2: <input type="number" name="n2" /><br />
                Número 3: <input type="number" name="n3" /><br />
                Número 4: <input type="number" name="n4" /><br />
                Número 5: <input type="number" name="n5" /><br />
                <input type="submit" name="submit" value="Inserir" />
        </form>
        <?php
                $n1=$_GET['n1'];
                $n2=$_GET['n2'];
                $n3=$_GET['n3'];
                $n4=$_GET['n4'];
                $n5=$_GET['n5'];

                // Maior
                $maior = $n1;
                if($n2>$maior)  $maior=$n2;
                if($n3>$maior)  $maior=$n3;
                if($n4>$maior)  $maior=$n4;
                if($n5>$maior)  $maior=$n5;
                echo "O maior numero é $maior<br />";

                //Soma
                $soma=$n1+$n2+$n3+$n4+$n5;
                echo "Soma: $soma<br />";

                //Média
                $media = $soma/5;
                echo "Média: $media<br />";
        ?>
 </body>
</html>
```

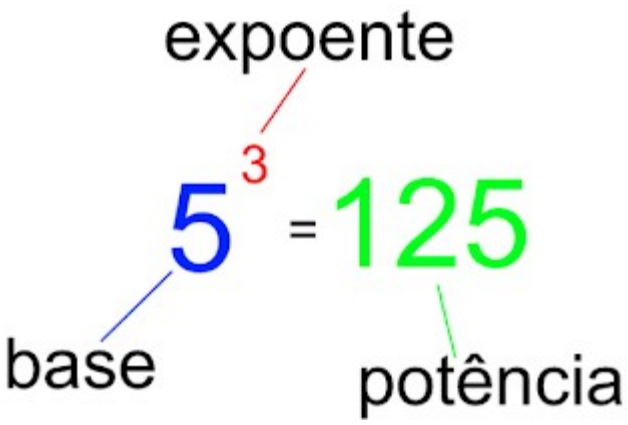
13.

Você deve lembrar das aulas básicas de Matemática, quando aprendemos a potenciação.

Ela é definida por três números:

- •base
- •expoente
- •potência (resultado)

Exemplo:



Por exemplo:
$5^4 = 5 \times 5 \times 5 \times 5$
(leia: 5 vezes o 4, por isso repetimos o 4 cinco vezes)

a^b = a x a x a ... x a (o **a** se repete **b** vezes).

Calcular potência usando laços

Vamos calcular a potência de um número inteiro elevado a outro inteiro, usando apenas laços.

Para isso, note que uma potenciação nada mais é que multiplicar um número por ele, várias vezes.
Por exemplo, em a^b

Multiplicamos a por a por a por a...um total de **b** vezes.

O script que pede uma base e um expoente ao usuário e exibe o resultado, usando laços WHILE, DO WHILE e FOR também:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
```

```
 </head>
 <body>
        <form action=ae method="get">
                Base: <input type="number" name="base" /><br />
                Expoent: <input type="number" name="expoente" /><br />
                <input type="submit" name="submit" value="Calcular" />
        </form>
        <?php
                $base=$_GET['base'];
                $exp=$_GET['expoente'];

                // WHILE
                $resultado = 1;
                $count=1;

                while($count<=$exp){
                        $resultado *= $base;
                        $count++;
                }
                echo "Resultado com WHILE: $resultado<br />";

                // DO WHILE
                $resultado = 1;
                $count=1;

                do{
                        $resultado *= $base;
                        $count++;
                }while($count<=$exp);

                echo "Resultado com DO WHILE: $resultado<br />";

                // FOR
                for($resultado=1, $count=1 ; $count<=$exp ; $count++)
                        $resultado *= $base;

                echo "Resultado com FOR: $resultado<br />";
        ?>
 </body>
</html>
```

Potenciação no PHP

Calcular um inteiro elevado a outro inteiro, já vimos como fazer.

Mas e se for quebrado?
1,2^2,1 ?

Para isso existe a função nativa **pow()**:
$resultado = pow($base, $expoente)

Veja mais detalhes e exemplos em:
http://php.net/manual/pt_BR/function.pow.php

15.

Série de FIbonacci

A série de Fibonacci é aquela composta pelos números 0 e 1, inicialmente.

A partir daí, cada termo é a soma dos dois anteriores.
Vamos formar a série:

Termo 1 = 0
Termo 2 = 1
Termo 3 = 1 + 0 = 1
Termo 4 = 1 + 1 = 2
Termo 5 = 2 + 1 = 3
Termo 6 = 3 + 2 = 5
...

Ela é infinita e possui uma série de funções interessantes, além de aparições curiosas na natureza.
Vale a leitura:
https://pt.wikipedia.org/wiki/Sequ%C3%AAncia_de_Fibonacci

Como Fazer a Série de Fibonacci com Laços

Inicialmente, exibimos os termos 0 e 1. Ou seja, o usuário tem que inserir um número maior que 2, para expressar o número de termos que deseja ver.

Vamos usar três variáveis:
$atual - exibe o número atual, da série
$ultimo - último número da série

$penultimo - penúltimo número da série

Inicialmente:
$ultimo=1;
$penultimo=0;

Assim, o próximo termo da sequência será sempre:
$atual = $ultimo + $penultimo;
Após isso, exibimos o $ultimo com um *echo*.

Agora, temos que atualizar o valor das variáveis, fazer elas darem um pulo pra frente.
O novo valor de $penultimo será $ultimo, e o novo valor de $ultimo vai ser $atual (esse é o passo mais importante para se entender, se não entender tente novamente, até conseguir).

Nossa variável de controle vai de 1 até $n-2 (onde $n é o número de termos que o usuário deseja ver, da série), subtraímos dois pois já exibimos os dois primeiros valores:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <form action="" method="get">
                Exibir até(maior que 2): <input type="number" name="number"
/><br />
                <input type="submit" name="submit" value="Calcular" />
        </form>
        <?php
                $n=$_GET['number'];
                $ultimo=1;
                $penultimo=0;

                echo "0<br />1<br />";
                for($count=1 ; $count<=$n-2 ; $count++){
                        $atual = $ultimo + $penultimo;
                        echo $atual."<br />";

                        $penultimo = $ultimo;
                        $ultimo = $atual;
                }
```

```
    ?>
 </body>
</html>
```

17.

Neste tutorial, vamos aprender como calcular o fatorial de um número usando apenas laços.

Fatorial de um Número

Chamamos fatorial de um número ***n***, representado por n!, o valor:
n! = 1 x 2 x 3 x ...(n-1) x n

Ou seja:
2! = 2 x 1 = 2
3! = 3 x 2 x 1 = 6
4! = 4 x 3 x 2 x 1 = 24
5! = 5 x 4 x 3 x 2 x 1 = 120
...

Ou seja, se o usuário insere um valor *n*, basta multiplicarmos 1 por 2, por 3, por 4....até chegar em *n*.
O fatorial é esse produto.

Script em PHP: Cálculo fatorial
Vamos armazenar em $n o número digitado pelo usuário.

O fatorial, vamos armazenar na variável $fatorial, que inicialmente é 1. Depois, vamos fazer uma variável de controle, a $count, inicie em 1 e vá até $n, incrementando-se unitariamente.

Veja como fazer usando o laço WHILE:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
```

```
        <form action="" method="get">
                Fatorial de: <input type="number" name="number" /><br />
                <input type="submit" name="submit" value="Calcular" />
        </form>
        <?php
                $n=$_GET['number'];
                $fatorial=1;
                $count=1;

                while($count<=$n){
                        $fatorial *= $count;
                        $count++;
                }
                echo $fatorial;
        ?>
 </body>
</html>
```

Veja como fazer usando o laço FOR:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <form action="" method="get">
                Fatorial de: <input type="number" name="number" /><br />
                <input type="submit" name="submit" value="Calcular" />
        </form>
        <?php
                $n=$_GET['number'];
                $fatorial=1;

                for($count=1; $count<=$n ; $count++)
                        $fatorial *= $count;

                echo $fatorial;
        ?>
 </body>
</html>
```

Consegue resolver usando DO WHILE ?

21. e 22.

Números Primos

Dizemos que um número é primo quando ele só é divisível por 1 e por ele mesmo.
Por exemplo, 2 só é divisível por 1 e por 2.

4 é divisível por 1, por 2 e por 4, logo não é primo.

Lista de alguns primos: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211 …

Para sabermos se um número **n** é primo, basta dividirmos ele por 2, por 3, por 4...por (**n-1)**.
Se não for por nenhum deles, é primo.

Começamos por $count=2, daí vamos dividir $n por $count.
Se for divisível, mostramos que aquele número é um divisor e armazenamos o tanto de divisíveis na variável $divisores, se não for, nada ocorre.

Ao final do laço, o valor armazenado em $divisores (que era inicialmente 0), vai nos dizer se tem divisores acima de 1 e abaixo de $n. Se tiver, ele não é primo.
Se não tiver, ele é primo.

Com um IF ELSE simples mostramos o resultado correto ao usuário:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <form action=ae method="get">
                Digite um numero: <input type="number" name="number" /><br />
                <input type="submit" name="submit" value="testar" />
        </form>
        <?php
                $n = $_GET['number'];
                $divisores = 0;
```

```
            for($count=2; $count<$n; $count++)
                if($n % $count == 0){
                    echo "Multiplo de $count<br />";
                    $divisores++;
                }

            if($divisores)  echo "Não é, tem $divisores divisores além de 1 e
ele mesmo";
            else                echo "É primo!";

        ?>
 </body>
</html>
```

42.

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <form action="" method="get">
            Digite um numero: <input type="number" name="number" /><br />
            <input type="submit" name="submit" value="testar" />
        </form>
        <?php
            $n = $_GET['number'];
            $soma = 0;

            for($count=1 ; $count<=$n ; $count++)
                $soma += (1/$count);

            echo "Soma: $soma";
        ?>
 </body>
</html>
```

43.

Cada expressão é na forma:

n/(2n-1), se você for substituindo vai notando isso.

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <form action=ae method="get">
                Digite um numero: <input type="number" name="number" /><br />
                <input type="submit" name="submit" value="testar" />
        </form>
        <?php
                $n = $_GET['number'];
                $soma = 0;

                //exibe
                for($count=1 ; $count<=$n ; $count++)
                        $resultado .= $count."/".(2*$count-1)."+";

                echo $resultado."<br />";

                for($count=1 ; $count<=$n ; $count++)
                        $soma += ($count/(2*$count -1));

                echo "Soma: $soma";
        ?>
 </body>
</html>
```

Funções

Nesta seção de nosso **Curso de PHP,** vamos estudar um dos elementos mais importantes e fundamentais de qualquer linguagem de programação: as funções.

Como o nome sugere, são blocos de código, que podem ser invocados, executados, receber e retornar informações, cujo objetivo é fazer uma tarefa específica.

Todo sistema web tem muuuitas funções, cada uma com seu propósito, fazendo tarefas únicas e objetivas, todas trabalhando em conjunto.

Função em PHP - O que é? Para que servem? Como funcionam ?

Neste tutorial introdutório, falaremos o que são funções em PHP, para que servem, como são usadas e como usar.

Função - O que é?

Função, de uma maneira bem simples, é um trecho de código que cumpre uma tarefa, ou seja, algo que tem uma função, um propósito específico.

Imagine que você precisa calcular o fatorial de um número, vai sempre que usar esse script, copiar e colar diversas vezes? E se em todo seu sistema web, for preciso calcular dezenas de vezes, vai colocar esse pedaço de código toda santa vez?
Muito trabalhoso!

Imagina então um único pedaço de código, que recebe um número pra calcular o fatorial e retorna o fatorial calculado, e sempre que quiser calcular novamente, basta solicitar a esse trecho de código. Pode requisitar uma, duas, 10 milhões de vezes.

Você digitou o código apenas uma vez e vai usar sempre que quiser, quantas vezes quiser.
Se entendeu essa ideia, você entendeu o conceito por trás de uma função.

Vai existir uma função de calcular fatorial, em algum canto do seu sistema, que sempre que você quiser esse cálculo, basta chamar esse trecho específico de código.

Funções - Quais as vantagens? Por que usar ?

Não existe sistemas de grande porte sem uso de funções, de tão importantes que ela são.

Os benefícios são vários, como:

- Escrever menos código
- Diminui o tempo de carregamento do seu sistema, pois escreveu menos

- Reduz o tanto de possibilidades de erro de código
- Auxilia na organização de seu código
- É mais fácil dar manutenção é um código complexo, pois deixa ele mais organizado
- Como cada função é compilada somente uma vez, diminui o tempo de execução de seus sistemas

• Funções - Como funcionam ?

Imagine um carro como um sistema web.

Existe uma parte específica dele que serve para dar partida.
O motor, é outra parte que faz a queima do combustível.

Existem sistemas específicos somente para a parte elétrica e eletrônica do carro.
Ele possui um sistema de amortecimento.

Existem peças responsáveis apenas pela parte de refrigeração do sistema, como o compressor do ar-condicionado.

Imagine cada sistema desse como uma função, pois fazem coisas mais simples, porém específicas.
Quando você une todo esse quebra-cabeça, faz essas funções trabalharem entre si, você tem seu sistema.

É assim na vida real.
Vai ter as funções de login, as de recuperar senha, a de exibir produtos na sua loja virtual, as funções responsáveis pelo sistema de pagamento e por ai vai.

Vamos dividir nosso sistema em partes menores, mais específicas, são as funções.
Funções essas que vão se encaixando e conversando uma com a outra, para montar seu sistema maior.

A partir do próximo tutorial vamos aprender a criar funções e invocar elas (chamar uma função, fazer ela funcionar).

Função - Como invocar e Criar uma função

Neste tutorial de nossa **apostila de PHP**, vamos aprender como invocar, ou seja, como chamar as funções, para fazerem elas serem executadas e vamos aprender a criar funções simples.

Como invocar uma Função

Já usamos, no decorrer de nossa apostila, algumas funções.
A primeira delas foi a print():

```
<?php
        print("Apostila PHP Progressivo");
?>
```

Ela funciona como a *echo*, simplesmente exibe uma string.

Detectamos que é uma função pelo par de parêntesis: print()
As vezes tem algo dentro do parêntesis, as vezes não tem, você entenderam o motivo.

Outra função que vimos, foi a de exponenciação: pow(base, expoente)

```
<?php
  echo "3 elevado a 4: ".pow(3,4);
?>
```

pow() é uma função, que recebe dois valores (3 e 4) e retornou um valor, o 81, com o resultado da operação.

É simplesmente assim que invocamos funções.

Como criar uma Função em PHP

Embora o PHP tenha uma vasta gama de funções prontas, escritas em algum lugar, e podendo ser invocada milhões de vezes que você quiser, muitas vezes vamos precisar criar nossas próprias funções.

A sintaxe para declarar uma função é:

```
function nome_funcao (parameteros) {
  codigo;
```

```
   codigo;

   return valor;
}
```

Ou seja, digitamos *function*, damos um nome que quisermos a função, vem os parêntesis, os parâmetros que queremos passar para a função, e o par de chaves.

Dentro do par de chaves, vem o código, o corpo da função, onde a mágica vai ocorrer.
Por fim, o comando *return* que retorna algum valor.

Os parâmetros e return não são obrigatórios e vamos aprender mais adiante como usar eles.

Vamos criar uma função, chamada olamundo() que simplesmente existe "Olá, mundo":

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <?php
          function olamundo(){
                echo "Olá, mundo";
          }
        ?>
 </body>
</html>
```

Quando rodamos a página, nada acontece.
Nem o Olá, mundo aparece. Por que?

Função é um pedaço de código que fica ali, guardado, na dele.
Só faz algo quando invocamos. Vamos chamar nossa função olamundo():

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
```

```
 </head>
 <body>
      <?php
            olamundo();

            function olamundo(){
              echo "Olá, mundo";
            }
      ?>
 </body>
</html>
```

Agora sim, aparece!
Experimente digitar:
olamundo();
olamundo();
olamundo();

Vai aparecer a mensagem 3x. E não tivemos que dar 3 echo.

Cada vez que invocamos, um bloco específico de código é executado, não precisamos ficar repetindo o mesmo código várias vezes.

É só criar funções bem específicas, com tarefas simples e invocar elas.
Vamos agora criar mais uma, a curso()
Que exibe na tela "Curso PHP Progressvo"

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
      <?php
            olamundo();
            curso();

            function olamundo(){
              echo "Olá, mundo<br />";
            }
            function curso(){
              echo "Curso PHP Progressivo<br />";
            }
```

```
        ?>
 </body>
</html>
```

O resultado são as duas strings, printadas no documento HTML.

Captou a ideia por trás das funções?
Calma que estas aí são bem simples, vamos fazer umas mais interessantes no próximo tutorial.

Função em PHP: Parâmetros, Argumentos e Retorno de uma função

Neste tutorial de nossa **apostila de PHP**, vamos aprender o que são parâmetros, argumentos e retorno de uma função que iremos criar.

Função em PHP: Parâmetro e Argumento

Quando demos o exemplo da print(), inserimos dentre parêntesis uma informação, ou seja, um parâmetro, que no caso, é uma string.

```
<?php
  print("Apostila PHP Progressivo");
?>
```

"Apostila PHP Progressivo" é o argumento que passamos para a função, que recebe como parâmetro uma string, que foi o que passamos.

A função pow() por exemplo, recebe dois números como parâmetros, a base e o expoente.
Exemplo:

```
<?php
  echo "3 elevado a 4: ".pow(3,4);
?>
```

3 e 4 são os argumentos que enviamos pra função.

Retorno de uma função

Como explicamos, as funções são trechos de códigos que fazem coisas específicas e geralmente retornam um valor.

No caso da pow(), ela deve ter uma implementação em algum canto do PHP, com diversas contas matemática dentro dela. Ela recebe dois parâmetros, faz alguns procedimentos e resultado no valor de um número elevado a outro...e retorna esse valor para o local onde a função foi invocada.

No exemplo dado, ela retornou pow(3,4)=81 para a *echo*, que exibe:
"3 elevado a 4: 81"

- **Exemplo de uso de função**

Vamos declarar uma função, a soma() que vai receber dois parâmetros, dois números, vai realizar a soma e retornar essa soma.

Nossa função fica assim:

```
function soma($a, $b){
   $soma = $a + $b;
   return $soma;
}
```

Veja o script:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <form action="" method="get">
                Numero 1: <input type="number" name="a" /><br />
                Numero 2: <input type="number" name="b" /><br />
                <input type="submit" name="submit" value="Calcular" />
        </form>
        <?php
                $a = $_GET['a'];
                $b = $_GET['b'];

                echo "Soma de $a + $b = ".soma($a, $b);
                function soma($a, $b){
                  $soma = $a + $b;
                  return $soma;
                }

        ?>
 </body>
</html>
```

Agora basta o usuário inserir dois valores, passarmos esses valores na forma de argumentos para a função e printar o retorno

Exemplo de uso de função

Agora vamos criar uma função que recebe um número como parâmetro e retorna o fatorial deste número.

O escopo dela é:

```
function fat($n){
   codigo que calcula o fatorial
   return $fatorial;
}
```

Como já ensinamos a você a calcular o fatorial com laços antes, nosso script fica:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <form action=ae method="get">
                Fatorial de: <input type="number" name="number" /><br />
                <input type="submit" name="submit" value="Calcular" />
        </form>
        <?php
                $n = $_GET['number'];

                echo "Fatorial de $n = ".fat($n);

                function fat($n){
                        $fatorial=1;
                        $count=1;

                        while($count<=$n){
                          $fatorial *= $count;
                          $count++;
                        }
                        return $fatorial;
                }

        ?>
 </body>
</html>
```

Passagem por Valor e Passagem por Referência: Escopo de uma função

Neste tutorial de nosso **curso de PHP**, vamos aprender como passar argumentos para funções de duas maneiras distintas: passagem por valor e passagem por referência, além de entendermos o escopo de uma função.

O Escopo de uma Função

Vamos criar um script PHP simples, para te explicar o escopo de uma função,
Vamos definir o valor de uma variável:

$a=1;

Em seguida, invocamos uma função dobra(), que recebe $a como argumento, dobra seu valor internamente.

Em seguida, após chamar a função, exibimos o valor de $a.
Ora, como era 1 e dobramos, deve ser 2, correto?

Teste o script:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <?php
                $a=1;
                dobra($a);
                echo $a;

                function dobra($a){
                        $a *= 2;
                }
        ?>
 </body>
</html>
```

E o resultado é....que runfem os tambores (sempre quis dizer isso)...tcharam...1 !
Isso mesmo, 1!

Ué, mas não tínhamos dobrado?
Na verdade, dobramos OUTRA variável, uma cópia dela...as coisas dentro do escopo de uma função são bem diferentes do que imaginávamos.
Vamos entender melhor.

▪ Passagem por Valor em funções PHP

Quando passamos uma variável como argumento para uma função, a função vai e cria uma **CÓPIA** dela e passa a trabalhar com ela.

Quando passou $a=1, a função criou outra variável, QUE SÓ EXISTE INTERNAMENTE, copiou o valor 1 e passou a trabalhar com ela.

Essa passagem é chamada por valor, pois passamos só o VALOR para a função, não a variável em si.
Para fazer a função se comunicar com o mundo externo, devemos fazer ela dar um *return* com seu resultado final, ao passo que fazemos $a receber o retorno dessa função:

Passagem por Referência
E se quisermos passar a variável em si, para que ela seja alterada dentro do escopo da função?
Tem como fazer? Tem sim, via passagem por referência.

Para passar uma variável por referência, basta colocarmos o símbolo & antes do nome da variável, assim: &$a

Vamos refazer o primeiro exemplo, agora com passagem por referência:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
      <?php
           $a=1;
```

```
        dobra($a);
        echo $a;

        function dobra(&$a){
            $a *= 2;
        }
    ?>
 </body>
</html>
```

Veja que o valor de $a realmente se alterou e agora é 2!

Isso acontece porque passamos uma referência, o endereço de memória da variável $a, assim, todas as alterações feitas nessa variável, serão feitas diretamente na variável. Tudo isso graças ao operador &.

Então resumindo:

1. Passagem por valor: os argumentos não são alterados, dentro da função se trabalha com uma cópia dessas variáveis, só o que importa é o valor delas. Se quiser mudar algo, use *return* para alterar fora da função
2. Passagem por referência: usamos o operador & que dá acesso diretamente a variável, assim a função vai trabalhar com as variáveis diretamente, alterando seu valor. Geralmente, não se usa *return* quando se trabalha assim, pois alterações fora da função já estão sendo feitas

▪ Mais escopo de funções

Vamos criar um script onde definimos uma variável:
$curso="PHP Progressivo";

E vamos invocar a função exibe(), que não recebe argumento nenhum, apenas dá um echo na variável $curso:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
```

```
        <?php
            $curso="PHP Progressivo";

            function exibe(){
                echo $curso;
            }
        ?>
 </body>
</html>
```

Note o que aconteceu: nada.

Isso porque a função é um mundo a parte, ela não sabe nada do que está acontecendo fora dela.
Ela só tem duas maneiras de comunicação:
Via parâmetros, recebendo argumentos, seja por valor ou referência
Enviando informações vi *return*

Se a variável $curso não foi passada para ela, ela não sabe o que é isso, nunca viu, nem ouviu.
A função só tem acesso aquilo que você passar para ela de informação, ok?

Variáveis de Escopo Global: global e $GLOBALS

Neste tutorial de nossa **apostila de PHP**, vamos aprender a usar e criar variáveis de escopo global em nossos scripts PHP.

Variáveis Globais

Em nosso tutorial anterior, quando falamos em escopo de uma função, dissemos que elas são escopos fechados, que não tem noção de nada de fora nem atrapalham nada fora do seu contexto, endo possível elas obterem informações via parâmetros e dar saídas via *return*.

Porém existe uma outra maneira de funções verem algo do mundo externo, que é através de variáveis *globais*.

Para usar uma variável como ela se fosse global, por exemplo $a anteriormente declarada, fazemos:
global $a

Essa técnica tem de diferente o fato de ser possível acessar essa variável de qualquer lugar dentro do script, mesmo dentro de funções, sem que as funções recebam elas como argumento.

Imagine que você quer dar um desconto de 10% em sua loja, simplesmente faça;
$desconto = 0.10;

Depois, sempre que quiser usar essa variável, faça:
global $desconto;
E saia sando normalmente, nativamente o PHP vai entender que está falando daquela variável $desconto que usou no começo do script.

E prontinho, não precisa mais sair definindo variáveis de descontos em milhares de páginas, faça apenas uma declaração, use a palavra-chave global e use esse valor onde quiser, sem precisar declarar nunca mais.

▪ Exemplo de uso de Variável Global

Crie um script que pede um radio de uma circunferência ao usuário. Através de uma função, exiba o valor do comprimento da circunferência e através de outra, exiba a área. Uso o valor de pi como 3.14

Nesse caso, nossa variável global vai ser o valor de pi
$pi=3.14;

Agora, sempre que quisermos usar a variável global (e não uma outra, como uma local de uma função de mesmo nome, fazemos):
$global $pi;

Pronto, agora podemos usar essa variável nas funções comprimento() e area().
Veja como ficou nosso código:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
      <form action=ae method="get">
            Raio: <input type="number" name="ratio" /><br />
            <input type="submit" name="submit" value="testar" />
      </form>
      <?php
            $pi = 3.14;
            $raio = $_GET['ratio'];

            echo "Comprimento: ".comprimento($raio)."<br/>";
            echo "Area: ".area($raio)."<br/>";

            function comprimento($raio){
                  global $pi;
                  return (2*$pi*$raio);
            }

            function area($raio){
                  global $pi;
                  return ($pi*$raio*$raio);
            }
      ?>
```

```
 </body>
</html>
```

▪ Variável global com $GLOBALS

A palavra **global** é como se tivesse o poder de transformar qualquer variável em global, o que pode ser arriscado, por isso use sempre com cautela e muita calma, você pode ter acesso ou dar acesso para hackers.

O mesmo podemos fazer usando um array (vamos estudar isso em breve), o $GLOBALS, que tem uma lista de todas variáveis do escopo global do script.

Se tem uma variável de nome $nome, acessamos ela globalmente através de:
$GLOBALS['nome']

Veja o script com uso de $GLOBALS:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
	<form action=ae method="get">
		Raio: <input type="number" name="ratio" /><br />
		<input type="submit" name="submit" value="testar" />
	</form>
	<?php
		$pi = 3.14;
		$raio = $_GET['ratio'];

		echo "Comprimento: ".comprimento($raio)."<br/>";
		echo "Area: ".area($raio)."<br/>";

		function comprimento($raio){
			return (2*$GLOBALS['pi']*$raio);
		}

		function area($raio){
			return ($GLOBALS['pi']*$raio*$raio);
		}
	?>
```

```
 </body>
</html>
```

Mais fonte de estudo:
http://php.net/manual/pt_BR/language.variables.scope.php
http://php.net/manual/pt_BR/reserved.variables.globals.php

Incluindo Arquivos Externos de PHP: include, include_once e require_once

Neste tutorial de **curso de PHP**, vamos aprender como incluir outros arquivos PHP em nosso script, usando a declaração *include, include_once* e *requirce_once*.

Incluir arquivos: include

Fizemos alguns scripts com uma ou duas funções, em nossos tutoriais passados.

Quando você for montar seu sistema web ou criar um jogo, por exemplo, vai facilmente criar dezenas ou centenas de funções.

Imagine só amontoar todas essas funções num bloco <?php ?> ?
Tudo num arquivo .php?
Não dá! Ia ficar muito desorganizado.

Existe então a possibilidade de colocar todas funções em um arquivo separado.
Por exemplo, crie um arquivo chamado "matematica.php"

E insira nele as seguintes funções:

```
<?php
	function soma($a, $b)
	{
		return $a + $b;
	}

	function sub($a, $b)
	{
		return $a - $b;
	}

	function multiplicacao($a, $b)
	{
		return $a * $b;
	}

	function divisao($a, $b)
```

```
	{
		return $a / $b;
	}
?>
```

Agora na sua home.php faça:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
	<form action="" method="get">
		Numero 1: <input type="number" name="num1" /><br />
		Numero 2: <input type="number" name="num2" /><br />
		<input type="submit" name="submit" value="Calcular" />
	</form>
	<?php
		include "matematica.php";
		$a = $_GET['num1'];
		$b = $_GET['num2'];

		echo "Soma: ".soma($a,$b)."<br />";
		echo "Subtracação: ".sub($a,$b)."<br />";
		echo "Multiplicação: ".multiplicacao($a,$b)."<br />";
		echo "Divisao: ".divisao($a,$b)."<br />";
	?>
 </body>
</html>
```

E veja o resultado! Aparece a soma, subtração, multiplicação e divisão!
Veja como nosso código ficou mais organizado.

Imagine agora que você vai criar um jogo.
Talvez as funções de cenários fiquem melhor no "cenario.php"
Os sons do game devem ficar em "sons.php"
As características dos personagens em "personagens.php"

Isso se o arquivo estiver no mesmo diretório de 'home.php'
Se tiver na pasta 'level2112', você deve dar: include "level2112/level.php"

Você pode incluir inclusive arquivos PHP externos de outros sites.

▪ Include, include_onde e require_once

Imagine que tenhamos o arquivo A.php e nele damos include no B.php e no C.php
No B.php damos include no C.php

Note que aí teríamos um problema: o C.php foi incluído duas vezes.
Na primeira quando incluímos o B.php (que já inclui o C) e quando o B novamente inclui o C.php

Isso vai gerar uma série de erros, pois vão existir variáveis e funções duplicadas.
Para evitar isso, ao invés de *include*, use:
include_once

Assim, se tentar incluir um arquivo que já foi incluso, ele será sumariamente desconsiderado.

Mas ainda assim temos um problema.
include e *include_once* apenas tentam incluir um arquivo, se não conseguem, bola pra frente, o script segue e você pode ter diversos erros achando que incluiu um arquivo quando não foi incluso.

Se for absolutamente imprescindível (geralmente é), não inclusa e sim requeira.
Use: *require_once*

Para não ter dúvidas, use sempre *require_once*.

Funções invocando Função

Neste tutorial de nosso **curso de PHP**, vamos ver como é a técnica de uma função invocar outra função dentro dela.

Função invocando Função

Uma das técnicas mais comuns em funções é o ato de uma função invocar outra.
Isso mesmo, dentro de uma função, é normal você invocar outra.

Vamos fazer um script que pede o raio para um usuário e calcula sua área:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
 <form action="" method="get">
  Raio: <input type="number" name="ratio" /><br />
  <input type="submit" name="submit" value="Calcular" />
 </form>
 <?php
  $raio = $_GET['ratio'];
  echo "A área é: ".area($raio);

  function area($raio){
   return pi()*pow($raio,2);
  }
 ?>
 </body>
</html>
```

Note que usamos duas funções
pi() - que retorna o valor matemático de PI
pow() - função de potenciação

Essa é uma técnica bem comum e vamos usar bastante, principalmente no tutorial seguinte, sobre funções recursivas.

▪ Exemplo de função invocando função

Crie um script que receba os coeficientes de uma equação do segundo grau, e diga se ela existe ou não, se tem raízes reais distintas, raízes reais iguais ou raízes imaginárias.

Já fizemos um script que calcula as raízes de uma equação do segundo grau.

Aqui vamos usar duas funções:
equacao() - recebe os 3 coeficientes, a, b e c e diz a classificação dela. Para isso, usa a função seguinte
delta() - função que recebe os 3 coeficientes como argumento e retorna seu valor

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <form action="" method="get">
                Coeficiente a: <input type="number" name="a" /><br />
                Coeficiente b: <input type="number" name="b" /><br />
                Coeficiente c: <input type="number" name="c" /><br />
                <input type="submit" name="submit" value="Calcular" />
        </form>
        <?php
                $a = $_GET['a'];
                $b = $_GET['b'];
                $c = $_GET['c'];

                echo "A equação : ".equacao($a,$b,$c);

                function equacao($a,$b,$c){
                        $delta=delta($a,$b,$c);

                                if($a==0)
                                        return "não existe";
                                elseif($delta==0)
                                        return "duas raízes reais iguais";
                                elseif($delta>0)
                                        return "duas raízes reais distintas";
                                else
                                        return "duas raízes imaginárias";
```

```
        }
        function delta($a,$b,$c){
            return ($b*$b - 4*$a*$c);
        }
    ?>
 </body>
</html>
```

Na echo, invocamos a equacao().
A primeira coisa que esta variável faz é calcular o valor de $delta, usando a função delta().

Depois, faz os testes necessários para saber se a equação existe e como são suas raízes.
Note que a função delta() vem até depois da função equacao(), pois quando você invoca uma função o interpretador sai caçando no script, ou nas páginas extras que você adicionou.

Você pode inclusive invocar várias vezes a delta.
Veja como fica agora sem usar a variável $delta, e sim usando a função delta() dentro dos testes condicionais:

```
<?php
    $a = $_GET['a'];
    $b = $_GET['b'];
    $c = $_GET['c'];

    echo "A equação : ".equacao($a,$b,$c);

    function equacao($a,$b,$c){

            if($a==0)
                return "não existe";
            elseif(delta($a,$b,$c)==0)
                return "duas raízes reais iguais";
            elseif(delta($a,$b,$c)>0)
                return "duas raízes reais distintas";
            else
                return "duas raízes imaginárias";
    }
    function delta($a,$b,$c){
        return ($b*$b - 4*$a*$c);
    }
?>
```

Funções Recursivas - Recursividade em PHP

"Para entender recursividade, tem que saber recursividade".

Ao final deste tutorial de nossa **apostila de PHP**, vamos entender essa piadinha interna entre programadores.

Função Recursiva

Uma função recursiva é nada mais nada menos que uma função que invoca...ela mesma.
Sim, isso mesmo, é uma função que chama ela mesma.

Obviamente se for só o código da função chamando ela mesmo, essa de dentro também vai chamar ela mesma, que vai chamar de novo ela mesma...e você criou um vírus, que vai consumir toda memória ram do computador (você pode upar esse código para seu servidor web e pegar seus amigos).

Para isso, deve necessariamente haver uma condição em que as funções para de se invocar.
É simplesmente um IF e ELSE.

Se atingir determinado valor de argumento, para de se invocar.
Senão, continua se invocando.

Vamos ver como calcular um somatório, um fatorial e a sequência de Fibonacci usando apenas recursividade.

Somatório com Recursão

Seja F(n) uma função que representa o somatório de um número ***n***, ou seja, é a soma:
F(n) = 1 + 2 + 3 + 4 + ...+ (n-1) + n

Por exemplo:
F(4) = 1 + 2 + 3 + 4 = 10
F(5) = 1 + 2 + 3 + 4 + 5 = 15

...
Porém, note uma coisa:
F(5) = (1 + 2 + 3 + 4) + 5 = F(4) + 5

Ou seja:
F(n) = n + F(n-1)

Então vai funcionar assim:
Vamos criar uma função de somatorio() que chega se o argumento for 1, se for, retorna 1 e acabou o programa.

Se não for 1, retorna *n* + somatorio(n-1);

Veja como fica o código:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
 <form action="" method="get">
  Somatório de: <input type="number" name="number" /><br />
  <input type="submit" name="submit" value="Calcular" />
 </form>
 <?php
  $numero = $_GET['number'];
  echo "O somatório de $numero é : ".somatorio($numero);

  function somatorio($numero){
   if($numero==1)
    return 1;
   else
    return $numero + somatorio($numero-1);
  }
 ?>
 </body>
</html>
```

▪ Fatorial com Recursão

Explicamos o que é e como calcular em:
Fatorial com laços em PHP

Veja que:
5! = 5 x 4 x 3 x 2 x 1
6! = 6 x 5 x 4 x 3 x 2 x 1 = 6 x 5!

Ou seja:
n! = n * (n-1)!

Por definição, 0!=1 e esse será nosso ponto de parada.
Nosso código fica:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
 <form action="" method="get">
  Fatorial de de: <input type="number" name="number" /><br />
  <input type="submit" name="submit" value="Calcular" />
 </form>
 <?php
  $numero = $_GET['number'];
  echo "O fatorial de $numero é : ".fat($numero);

  function fat($numero){
   if($numero==0)
    return 1;
   else
    return $numero * fat($numero-1);
  }
 ?>
 </body>
</html>
```

Fibonacci com Recursão

Já falamos sobre isso em:
Sequência de Fibonacci com Laços

Faremos, por definição:
F(0) = 0;
F(1) = 1;

E a fórmula com recursividade é:
F(n) = F(n-1) + F(n-2);

O ponto de parada é quando *n* for 0 ou 1, veja como exibir a sequência de Fibonacci usando recursão:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <form action="" method="get">
                Fatorial de de: <input type="number" name="number" /><br />
                <input type="submit" name="submit" value="Calcular" />
        </form>
        <?php
                $numero = $_GET['number'];
                echo "O termo $numero é: ".fibo($numero);

                function fibo($numero){
                        if($numero==0 || $numero==1)
                                return $numero;
                        else
                                return (fibo($numero-1) + fibo($numero-2));
                }
        ?>
 </body>
</html>
```

Conversão Celsius para Fahrenheit e vice-versa em PHP

Neste **tutorial de PHP**, vamos aprender como fazer a conversão entre Celsius e Fahrenheit, e vice-versa, utilizando nossos conhecimentos de funções em PHP.

Exercício de PHP

Vamos resolver o exercício da lista:

> 1.Escreva um script que pergunta ao usuário se ele deseja converter uma temperatura de grau Celsius para Fahrenheit ou vice-versa. Se ele digitar 1, é de Celsius para Farenheit, se digitar 2 é de Fahrenheit para Celsius, outro valor mostre uma mensagem de erro. Para cada conversão, chame a função correta.

Conversão Celsius e Fahrenheit

Vamos primeiro pedir a opção do usuário.
Se digitar 1, é porque ele vai digitar a temperatura em grau Celsius e quer Fahrenheit.
Se digitar 2, é porque ele vai digitar em Fahrenheit e quer a temperatura em Celsius,

No script PHP, tratamos essa opção.
Se ele escolher 1, chamamos a opção CtoF() e passamos o argumento $temperatura.
Se escolher 2, chamamos a função FtoC() e passamos o argumento $temperatura também.

As fórmulas são:

$$F = C \cdot \frac{9}{5} + 32$$

$$C = (F - 32) \cdot \frac{5}{9}$$

- **Script em PHP da Conversão**

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <form action=ae method="get">
                Digite 1 para: Celsius para Farenheit<br />
                Digite 2 para: Farenheit para Celsius<br />
                <input type="number" name="opt" /><br />
                Temperatura:<input type="number" name="number" /><br />
                <input type="submit" name="submit" value="Calcular" />
        </form>
        <?php
                $op = $_GET['opt'];
                $temp = $_GET['number'];

                if($op==1)
                        echo "Temperatura em Farenheit: ".CtoF($temp);
                elseif($op==2)
                        echo "Temperatura em Celsius: ".FtoC($temp);
                else
                        echo "Opção inválida";


                function CtoF($temperatura){
                        return ($temperatura*9/5) + 32;
                }
                function FtoC($temperatura){
                        return ($temperatura-32)*5/9;
                }
        ?>
 </body>
</html>
```

Prontinho.
Veja que bacana e organizado nosso script.

Agora você pode até colocar seu site no ar, que faz essa conversão, pois está funcionando perfeita e maravilhosamente bem.

Você pode ir trabalhar numa estação de serviços meteorológicos e não vai mais precisar criar esse código de novo, pois já programou essa função, agora é só guardar esse script e usar quando quiser.

Pode guardar como "conversao.php" e prontinho, sempre que quiser usar é só dar um *require_once "conversao.php"* e fazer suas conversões a vontade

Show esse PHP, não?

Como gerar números aleatórios em PHP: Função rand()

Neste tutorial de nosso **curso de PHP**, vamos aprender a gerar qualquer tipo de número aleatório através da função rand() do PHP.

Números Aleatórios: Função rand()

Você já viu aqueles captchas malucos ou sistemas que geram códigos para você confirmar no celular ou mesmo senhas aleatórias pra você?

Pois é, gerar valores aleatórios é algo muito importante mesmo no mundo da programação, e o lindo e maravilhoso PHP já possui uma função que faz tudo isso por nós, de maneira automática, é a função **rand()**.

Dê um:
echo rand()

E fique atualizando a página.
Várias, várias vezes.

Vai ver que cada vez que atualiza, aparece um número diferente.
Esse número vai de 0 até getrandmax()

Aqui, esse getrandmax() deu 2147483647, e aí?

Gerando números aleatórios em um intervalo

A função rand() é tão versátil que você pode definir intervalos que você deseja gerar.
Por exemplo, para gerar 0 ou 1 para você brincar de par ou ímpar, faça:
rand(0,1)

Se você quiser simular lançamento de dados, faça:
rand(1,6)

Que ela vai gerar os números 1, 2, 3, 4, 5 e 6 aleatoriamente.

E se quiser um número quebrado? Entre 0.0 e 1.0?

Faça: rand(0,1)/10

Entendeu a lógica ?

▪ Exercício: Criando um jogo em PHP

Crie um script de um game. O PHP vai gerar um número entre 1 e 100, e você vai ficar tentando adivinhar que número foi esse.
Se seu palpite for maior, ele avisa. Se for menor, também te avisa.
Quando acertar, te diz em quantas tentativas você levou para conseguir.

No próximo tutorial vamos te ensinar como fazer esse jogo em PHP.

Mais fonte de estudos:
http://php.net/manual/pt_BR/function.rand.php

Jogo em PHP: Adivinhe o Número

Jogo em PHP

Primeiro, começamos pedindo um número ao jogador, que vai ficar armazenado na variável $palpite.

Depois, geramos um número aleatório entre 1 e 10 através da função rand() e armazenamos tal valor na variável $resposta.

Agora é só comprar os valores de $palpite e $resposta.
Se forem iguais, dizemos que acertou!
Se forem diferentes, dizemos que errou e qual foi o número sorteado.

Veja o código do jogo:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
	<form action="" method="get">
		PHP gerou um número entre 1 e 10! Tente adivinhar!<br />
		Palpite:<input type="number" name="number" /><br />
		<input type="submit" name="submit" value="Calcular" />
	</form>
	<?php
		$palpite = $_GET['number'];
		$resposta = rand(1,10);

		if($palpite==$resposta)
			echo "Acertou miseravi! Era $resposta!";
		else
			echo "Errou! Era $resposta !";
	?>
 </body>
</html>
```

Agora é só upar esse jogo pro seu site e desafiar seus colegas pra ver quem acerta de primeira.

Lista de exercícios de Funções

1.Escreva um script que pergunta ao usuário se ele deseja converter uma temperatura de grau Celsius para Farenheit ou vice-versa. Se ele digitar 1, é de Celsius para Farenheit, se digitar 2 é de Farenheit para Celsius, outro valor mostre uma mensagem de erro. Para cada conversão, chame a função correta.

2.Faça um programa, com uma função que necessite de três argumentos, e que forneça a soma desses três argumentos através de uma função. Seu script também deve fornecer a média dos três números, através de uma segunda função que chama a primeira.

3.Faça um programa que recebe três números do usuário, e identifica o maior através de uma função e o menor número através de outra função.

4.A probabilidade de dar um valor em um dado é 1/6 (uma em 6). Faça um script em PHP que simule 1 milhão de lançamentos de dados e mostre a frequência que deu para cada número.

5.A série de Fibonacci é uma sequência de números, cujos dois primeiros são 0 e 1. O termo seguinte da sequência é obtido somando os dois anteriores. Faça uma script em PHP que solicite um inteiro positivo ao usuário, *n.* Então uma função exibe todos os termos da sequência até o *n-ésimo* termo. Use recursividade.

6.Crie uma função que recebe um inteiro positivo e teste para saber se ele é primo ou não. Faça um script que recebe um inteiro *n* e mostra todos os primos, de 1 até n.

7.Um número é dito **perfeito** quando ele é igual a soma de seus fatores. Por exemplo, os fatores de 6 são 1, 2 e 3 (ou seja, podemos dividir 6 por 1, por 2 e por 3) e 6=1+2+3, logo 6 é um número perfeito. Escreva uma função que recebe um inteiro e dizer se é perfeito ou não. Em outra função, peça um inteiro *n* e mostre todos os números perfeitos até *n*.

Números perfeitos em PHP

8.Faça um programa para imprimir:

```
    1
  2   2
```

```
3   3   3
.....
n  n  n  n  n  n ... n
```

para um n informado pelo usuário. Use uma função que receba um valor n inteiro e imprima até a n-ésima linha

•

9.Faça um programa para imprimir:

```
1
1   2
1   2   3
.....
1  2  3  ... n
```

para um n informado pelo usuário. Use uma função que receba um valor n inteiro imprima até a n-ésima linha.

•

10.Faça um programa, com uma função que necessite de um argumento. A função retorna o valor de caractere ‘P’, se seu argumento for positivo, e ‘N’, se seu argumento for zero ou negativo.

11.Faça um programa com uma função chamada somaImposto. A função possui dois parâmetros formais: taxaImposto, que é a quantia de imposto sobre vendas expressa em porcentagem e custo, que é o custo de um item antes do imposto. A função “altera” o valor de custo para incluir o imposto sobre vendas.

12.Faça um programa que converta da notação de 24 horas para a notação de 12 horas. Por exemplo, o programa deve converter 14:25 em 2:25 P.M. A entrada é dada em dois inteiros. Deve haver pelo menos duas funções: uma para fazer a conversão e uma para a saída. Registre a informação A.M./P.M. como um valor ‘A’ para A.M. e ‘P’ para P.M. Assim, a função para efetuar as conversões terá um parâmetro formal para registrar se é A.M. ou P.M. Inclua um loop que permita que o usuário repita esse cálculo para novos valores de entrada todas as vezes que desejar.

13.Faça uma função que informe a quantidade de dígitos de um determinado número inteiro informado.

01.

Já resolvida

07.

Vamos receber um número $n do usuário.

Dentro da função soma(), vamos ter uma variável chamada $divisores, que inicialmente é 0, ela vai receber a soma de todos os divisores.

Depois, com um laço, vamos dividir o número do usuário por 1, por 2, por 3...até por $n-1
Cada vez que for divisor, somamos esse números na $divisores.

Ao final da função, retornamos $divisores, que é a soma dos divisores.

Fora da função, verificamos se $n é igual ao retorno da função, se for, o número é dito perfeito:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
 <form action=ae method="get">
  Numero<input type="number" name="numb" /><br />
  <input type="submit" name="submit" value="Testar" />
 </form>
 <?php
  $n = $_GET['numb'];

  if($n == soma($n))
   echo "É perfeito";
  else
   echo "Não é perfeito";

  function soma($n){
   $divisores=0;
   for($count=1 ; $count<$n ; $count++)
    if($n % $count==0)
     $divisores += $count;
   return $divisores;
  }
 ?>
```

```
 </body>
</html>
```

Teste com alguns números perfeitos:
6, 28, 496 e 8.128

08.

O grande segredo aqui é fazer a divisão correta entre linhas e colunas, o que é cada uma e o que cada uma faz:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <form action=ae method="get">
                Numero<input type="number" name="numb" /><br />
                <input type="submit" name="submit" value="Testar" />
        </form>
        <?php
                $n = $_GET['numb'];

                imprime($n);

                function imprime($n){
                        for($linha=1 ; $linha<=$n ; $linha++){
                                for($coluna=1 ; $coluna<=$linha ; $coluna++)
                                        echo "$linha  ";

                                echo "<br />";
                        }
                }
        ?>
 </body>
</html>
```

09.

Se entendeu bem o anterior, vai entender este:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
```

```
</head>
<body>
    <form action=ae method="get">
        Numero<input type="number" name="numb" /><br />
        <input type="submit" name="submit" value="Testar" />
    </form>
    <?php
        $n = $_GET['numb'];

        imprime($n);

        function imprime($n){
            for($linha=1 ; $linha<=$n ; $linha++){
                for($coluna=1 ; $coluna<=$linha ; $coluna++)
                    echo "$coluna  ";

                echo "<br />";
            }
        }
    ?>
</body>
</html>
```

13.

Quando o usuário digita algo via formulário, ele está introduzindo uma string. Assim, ao invés de digitar o número 2112, ele está digitando a string: "2112"

Então, para calcular o número de dígitos, temos que calcular tamanho de caracteres da string, e para isso usamos a função **strlen()** (*string length*):

```
function tamanho($n){
    return strlen($n);
}
```

Número de dígitos de um número

E se ao invés de uma string for um número?
Simples, transforma em string.

Se $n for um número, transformamos ele em string assim:
$n = (string)$n
Ou assim:
$n .=""
Depois é só aplicar como argumento na função strlen.

Arrays

Agora que já aprendemos a trabalhar com uma ou outra variável, chegou a hora de trabalhar com dezenas, centenas, milhares ou até milhões de informações, tudo ao mesmo tempo, de uma maneira rápida, fácil e organizada.

Tudo isso é possível por conta do estudo dos arrays, estruturas de dados que nos permitem trabalhar com grande quantidade de informações de uma maneira muito prática, que seria impossível de se fazer ‘manualmente’.

Array em PHP - O que é ? Para que serve? Onde se usa?

Neste tutorial de nosso **Curso de PHP**, vamos dar uma introdução sobre o uso dos Arrays, falar o que são, para que servem, como e onde são usados, bem como criar alguns.

Array em PHP - Por que usar?

Até o momento, nossas variáveis serviam para armazenar uma coisa apenas de cada vez.
Uma número, uma string, um float...

As vezes, pedíamos 2, 3 até 5 informações ao usuário, até aí, ok.
Mas e se você for contratado para criar o sistema web de uma faculdade com mil alunos.

Vai criar mil variáveis?
Se cada aluno tiver 10 matérias? Vai criar 10 mil variáveis?

Assim não pode, assim não dá, e é aí que entram os arrays!

Arrays em PHP - O que é?

Veja os arrays como um conjunto ordenado de alguma coisa.
Por exemplo, uma sala de aula é um array de estudantes.

Uma concessionária, é um array de carros.
Pessoas na porta de uma boate, um array de clientes querendo entrar.

Arrays são um bloco de estruturas menores, parecidas entre si, que estão de alguma maneira ordenada, seja numericamente ou por uma chave (key).

Arrays em PHP - Onde são usados ?

Sabe aqueles pães de padaria, assim que saem de forma, todos grudados? São exemplos perfeitos de um array, onde cada pão é um elemento do array.

Então, em vez de trabalhar declarando mil variáveis na sua faculdade, você declara um array com mil elementos, onde cada elemento desse representa a nota de um aluno.

Imagine agora um sistema bancário, com milhões de clientes. Existe um array, onde cada cliente é um bloco desse array.

Um array pode conter um inteiro, um float, uma string, outro array (arrays multidimensionais), objetos de qualquer tipo...arrays são estruturas para se trabalhar com muita informação, de maneira rápida, fácil e organizada, como vamos ver.

Vamos ver como declarar e acessar os elementos de um array no próximo tutorial.

Arrays em PHP: Como criar, Acessar e Printar

Neste tutorial de PHP, vamos finalmente aprender como criar nossos arrays. Também veremos como acessar seus elementos e exibir (printar) um array.

Como criar um Array

Primeiro, vamos te ensinar como criar um array numericamente indexado, que são os mais comuns na maioria das linguagens de programação.

A sintaxe é:
$nome_variavel = array (elemento0, elemento1, elemento2, ...)

Ou seja, basta escolhermos o nome da variável, usar a palavra-chave **array**, abrir parêntesis e entre vírgulas fornecer os elementos de um array.

Vamos criar um array de carros:
$carros = array('gol', 'celta', 'fox', 'corolla', 'civic');

Uma outra maneira de inicializar um array seria:
$carros[]='gol';
$carros[]='celta';
$carros[]='fox';
$carros[]='corolla';
$carros[]='civic';

Note que tem um par de colchetes após o nome da variáveis, ela quer dizer que estamos lidando com um array, e não com uma variável escalar simples. Cada vez que fazemos isso, estamos adicionando um elemento so array $carros.

Como Acessar Elementos de um Array

Para os programadores, a contagem começa do 0, sempre.
Assim, o primeiro elemento, é o elemento 0, e é o 'gol'.
Elemento 0 -> 'gol'
Elemento 1 -> 'celta'
...

Elemento 4 -> 'civic'

Para acessar o primeiro elemento, fazemos:
$carros[0] -> aqui está armazenada a string 'gol'
$carros[1] -> aqui está armazenada a string 'celta'
...
$carros[2] -> aqui está armazenada a string 'corolla'

Como Exibir Elementos de um Array

Imagina que tivéssemos a variável
$meuCarro='gol'

E você quisesse escrever em PHP: "Meu carro é um $meuCarro";
Você entenderia perfeitamente, não é verdade?

Aqui, com arrays, é a mesma coisa, a diferença é em vez de uma variável comum, a do array possui um colchete de um número associado.
Ficaria: "Meu carro é um $carros[0]"

Prontinho.
A diferença é que você tem um array com uma gama enorme de dados.

Como imprimir um Array: print_r

Existe uma função especial, a **print_r()** que serve para imprimir arrays de uma maneira mais bonita e organizada.

Teste: print_r($carros)
O resultado vai ser:
Array ([0] => gol [1] => celta [2] => fox [3] => corolla [4] => civic)

- ## Exercício de Array

Você foi contratado para ser um desenvolvedor web. Com seu salário, vai comprar uma BMW.
Adicione "BMW" ao array de carros e diga que possui uma BMW acessando o elemento do array.

Resposta:
$carros[] = 'BMW';
echo "Eu tenho uma $carros[5]";

Bem simples esses arrays, não?

Arrays Associativos em PHP

Neste tutorial de nossa **apostila de PHP**, vamos aprender o que são, como criar, usar e acessar os arrays associativos em PHP.

Arrays Associativos em PHP

Vimos no tutorial anterior sobre como criar arrays de índices numéricos, como trabalhar com arrays que possuem índices enumerados.

Porém, as vezes fica difícil trabalhar com números, é decorativo pois você precisa lembrar que número é associado a cada elemento e isso também deixa difícil para outros programadores darem manutenção em seu código.

E é aí que entram os arrays associativos, que não precisam de números para associar determinados items, você pode associar qualquer chave (*key*) para relacionar a qualquer valor (*value*).

Como criar um array associativo

Vamos supor que você foi contratado numa concessionária e vai precisar criar um array com os carros da sua loja, mas você vai usar arrays associativos. Nosso array é o $array.

Em vez de números, vamos dar nomes as *keys*:

```
$carros['popular'] = 'celta';
$carros['sedan'] = 'corolla';
$carros['premium'] = 'bmw';
$carros['suv'] = 'hr-v';
```

Pronto.
Quer se referir ao Celta, que é um carro popular? Acesse o item do array:

```
$carros['popular'];
```

Como imprimir arrays associativos

Vamos imprimir esse array:
print_r($carros);

O resultado é:
Array ([popular] => celta [sedan] => corolla [premium] => bmw [suv] => hr-v)

Veja o motivo de ser associativo:
[popular] => celta
[sedan] => corolla
[premium] => bmw
[suv] => hr-v

Os nomes da esquerda são as *keys* e da direita os *values*.
Aí fica mais fácil entender essa 'associação' dos pares chave-valor.

A função range() no PHP

Neste tutorial, vamos aprender a usar a função **range()**, muito útil para se usar com arrays em PHP.

A função range()

Muitas vezes, é necessário preencher um array com números, ordenados. Por exemplo, criar um array de 100 elementos, para representar os funcionários de uma empresa e inicializar eles manualmente se torna uma tarefa enfadonha.

Imagina nas lojas virtuais que você vai criar, com milhares de produtos a venda?

A função range() é perfeita para se trabalhar com arrays e ela possui a seguinte sintaxe:
array range(int menor, int maior, int *pulo*);

Esse *pulo*, é opcional.
Mas é basicamente assim, você vai fornecer um valor mínimo (menor) e um máximo (maior) e ela te retorna um objeto do tipo array

Por exemplo, vamos criar um dado, um array que vai de 1 até 6:
$dado = range(1, 6);

Pronto, $dado já um objeto do tipo array com 6 elementos (de índices que vão de 0 até 5 - sempre se lembre, em computação se começa contando do 0 e não do 1).

Função range() com salto

Vamos supor que você queira um array com elementos de 1 até 100, mas de 2 em 2.
Ou seja, seus elementos vão ser: 1, 3, 5, 7, 9, ..., 99

Basta fazer: $meuArray = range(1, 100, 2);

Primeiro o ponto de partida, depois o ponto máximo que ele pode chegar e o salto que vai ficar dando como terceiro argumento.

Função range() para Letras
É possível usar a função range com letras também.

Para gerar o alfabeto minúsculo:
$alfaminusculo = range('a', 'z');

Para gerar o alfabeto maiúsculo:
$alfamaiusculo = range('A', 'Z');

E prontinho, automaticamente vai ter um array totalmente preenchido pronto para você usar, seja lá qual for sua necessidade, e acredite, várias vezes você vai precisar usar a range.

O Loop FOREACH - Laço de arrays

Neste tutorial de nossa **apostila de PHP**, vamos aprender o que é e como usar o FOREACH, laço especialmente criado para se usar com arrays.

O Laço foreach as ... para arrays numéricos

Como se já não bastasse o número de laços (while, do while e for) existentes na linguagem, os criadores do PHP desenvolveram um especialmente para se usar com arrays, o *foreach as*.

Vamos pegar o array:
$carros = array('gol', 'celta', 'fox', 'corolla', 'civic');

Para exibir esses carros usando foreach, fazemos:

```
<?php
	$carros = array('gol', 'celta', 'fox', 'corolla', 'civic');

	foreach($carros as $car)
		echo "$car <br />";
?>
```

O que ocorre é o seguinte...fazemos com que o array $car assuma o valor de cada elemento do array maior $carros.

Assim, o laço foreach percorre todos os elementos do array $carros, onde podemos trabalhar da maneira que quisermos com cada item.

▪ O Loop foreach as ... para arrays associativos

Embora não sejam estruturas enumeradas e logicamente organizadas, é possível usar o laço foreach com arrays associativos também.

Vamos definir um array de chaves e valores da seguinte forma:
$carros['popular'] = 'celta';
$carros['sedan'] = 'corolla';
$carros['premium'] = 'bmw';
$carros['suv'] = 'hr-v';

Para exibir da chave e cada valor, com foreach, um por linha, fazemos:

```
<?php
    $carros['popular'] = 'celta';
    $carros['sedan'] = 'corolla';
    $carros['premium'] = 'bmw';
    $carros['suv'] = 'hr-v';
    foreach($carros as $chave => $valor)
        echo "$chave : $valor <br />";
?>
```

A variável $chave vai pegar o valor de cada *key* do array $carros e a variável $valor o *value* de cada elemento correspondente, assim podemos acessar e alterar esses elementos da maneira que quisermos.

Note que é possível percorrer arrays gigantescos, medonhos e ir fazendo alterações, copiando, ou o que for, de uma maneira muito simples e totalmente automatizada com o laço *foreach as*...que inglês significa "para cada ... como..."

Ou seja, "para cada elemento do array, trate como"...e o foreach vai sozinho pegando um por um, todos os elementos do array.

Funções de Arrays em PHP

Neste tutorial de nosso **curso de PHP**, vamos estudar diversas funções e operações que são possíveis de se fazer usando arrays.

Operações com Arrays

União: $a + $b
O array $b é anexado ao final do array $a, menos os conflitos de chaves

Igualdade: $a == $b
Verdade se $a e $b contém os mesmos elementos

Identidade: $a === $b
TRUE se $a e $b contém os mesmos elementos e na mesma exata ordem

Desigualdade: $a != $b ou $a <> $b
TRUE se $a e $b não contém os mesmos elementos, ou seja, ao menos um difere

Não-identidade: $a !== $b
Verdadeiro caso $a e $b não contenham os mesmos elementos na mesma ordem

Funções de Adição e remoção de elementos em Arrays

1 Adiciona elementos ao final: array_push()

Sintaxe:
int array_push(array alvo , variáveis);

Seja o array: $carros=array("gol", "celta");
Se fizermos: array_push($carros, "civic", "corolla")
Vai ficar: $carros=array("gol", "celta", "civic", "corolla");

2 Adiciona elementos ao início: array_unshift()

Funciona igual ao array_push(), mas ao invés de adicionar ao final, adiciona no começo.

Seja o array: $carros=array("gol", "celta");
Se fizermos: array_unshift($carros, "civic", "corolla")
Vai ficar: $carros=array("civic", "corolla", "gol", "celta");

3 Retira o último elemento: array_pop()

Retira e retorna o último elemento do array.

Sintaxe:
variavel array_pop(array alvo);

Seja o array: $carros=array("gol", "celta");
Se fizermos: $carro = array_pop($carros)
Temos: $carro = "celta";

4 Retira o primeiro elemento: array_shift()

Funciona igual o array_pop(), mas ao invés de retirar o último elemento, retira o primeiro.

Se estiver usando arrays numéricos, todos vão 'descer' um degrau numérico. Se for em arrays associativos, nada acontece.

Seja o array: $carros=array("gol", "celta");
Se fizermos: $carro = array_shift($carros)
Temos: $carro = "gol";

▪ Localizando Elementos em um Array

5 Localizar se algo está no array: in_array()

Sintaxe:
boolean in_array(algo, $array);
Retorna TRUE se *algo* está no $array e FALSE caso não esteja.

Seja $carros=array("gol", "celta", "civic", "corolla");
Se fizermos:
in_array("civic", $carros) -> retorna verdadeiro
in_array("fox", $carros) -> retorna falso

Localizar key: array_key_existis()
Sintaxe:
boolean array_key_existis(key, $array)

Retorna TRUE se *key* existe no array e FALSE se não existe.

- **Ordenando Arrays**

6 Classificando arrays: sort()

Classifica o array em ordem alfabética ou numérica.

Seja: $numeros = array(2,3,1)
Fazendo: sort($numeros)
Fica: $numeros = arrau(1,2,3)

Seja $estados = array('SP','CE','AL')
Fazendo: sort($estados)
Fica: $estados = array('AL', 'CE', 'SP')

Ou seja, o método sort() classifica alfa ou numericamente os valores de elementos de um array.

Sabe aqueles filtros:
"Ordenar por mais procurado"
"Ordenar por mais barato"

Pois é, é tudo array e tudo usa sort().
Esse PHP é mesmo poderoso, não é?

7 Classificando em ordem contrária: rsort()

Inverso da sort()

8 Classificação nativa: natsort()

Procura fazer uma classificação nativa, como uma pessoa faria.

Seja o array:
$fotos=array("foto2.jpg", "foto3.jpg", "foto1.jpg");

Fazendo: natsotr($fotos);
Fica:$fotos=array("foto1.jpg", "foto2.jpg", "foto3.jpg");

- # Mais funções

9 Combinando keys e values: array_combine()

Sintaxe:
array array_combine(array $key, array $value)

Ele combina um array de que são serão as keys, o $key, com um que são os valores, o $value.
Seja:
$tipo=array('popular', 'sedan');
$modelo=array('gol', 'civic');
Se fizermos: $carros = array_combine($tipo, $modelo);

E dermos um print_r:
Array ([popular] => gol [sedan] => civic)

10 Número aleatório de elementos: array)rand()

Sintaxe:
elementos array_rand($array, numero_aleatorio_elementos);

Ele pega um array e retorna um número aleatório de elementos.
Seja $carros=array("gol", "celta", "civic", "corolla");
Se fizermos:
$aleatorios = array_rand($carros, 2)

Aqui retornou: Array ([0] => 0 [1] => 3)
Ou seja, "gol" e "corolla", ele já dá a resposta em array associativo.

11 Embaralhar elementos: shuffle()

void shuffle(array input_array)

Simplesmente embaralha os elementos de um array.
Por exemplo, se tiver: $carros=array("gol", "celta", "civic", "corolla");
E fizer: shuffle($carros);

O resultado vai ser:
Array ([0] => celta [1] => gol [2] => corolla [3] => civic)

Somando elementos: array_sum()
array_sum($array)

Retorna a soma de todos os items que forem somáveis, ou seja, que foram inteiros ou floats.
Seja:
$meuArray = array(21, "rush", 12);
E fizermos: $soma = array_suma($meuArray);
O resultado séra: $soma=33

Falamos apenas de algumas poucas funções que podemos fazer com arrays, para ver mais, acesse:
https://secure.php.net/manual/pt_BR/ref.array.php

Orientação a Objetos em PHP

Nesta seção, iremos dar início ao estudo de um dos assuntos mais fantásticos e inovadores do PHP: a orientação a objetos.

Iremos aprender sobre os principais conceitos de orientação a objetos, como classes, objetos, herança, polimorfismos, encapsulamento e tudo dessa nova maneira de programar que imita muito o mundo real.

O que são Classes e Objetos

Neste tutorial vamos (tentar) te ensinar os conceitos de Classe e Objetos em PHP, a base fundamental da programação orientada a objetos.

Classes e Objetos

O conceito mais importante, de longe, para você entender, é o de classe e o de objeto. E também é considera o mais difícil, pois a definição é muito técnica, muito abstrata, mas vamos tentar resolver isso depois com exemplo do mundo real.

Classes são, em suma, entidades, com suas características e conceitos próprios, como pessoas, veículos, lugares, coisas abstratas, genéricas. Cada entidade é definida por um conjunto de características e comportamentos específicos daquela classe.

As classes é como se fossem uma 'receita' um 'template' de algo tangível, e esse tangível são os objetos.
Tudo ao seu redor é um objeto. Classes são apenas ideias, generalizações, conceitos, objeto é a parte tangível, existencial da classe.

Calma, você não vai precisar fumar um pra entender essa viagem toda, vamos partir para o mundo real e entender de fato o que são classes e objetos.

A Classe Pessoa

Pessoa é uma classe, é uma generalização.
Pessoa tem cabeça, tem coração, tem cérebro, tem membro, tem órgãos, tem nome...mas você não conhece nenhuma Pessoa.

"Como não, PHP Progressivo, ta louco de pedra, tio? Claro que conheço pessoas"

Não, amigo.
Você conhece sua mãe, seu pai, o João, a Maria.
Você conhece pessoas específicas.

Você não chega em ninguém e essa pessoa diz:
- Oi, sou pessoa. Tenho nome, e tenho emprego. Tchau.

Não, ela diz.
- Sou Maria, sou programadora Web e rica pra caramba.

Pessoa é uma classe, uma generalização, um conceito.
Os objetos é que são reais, a Maria é um objeto, você é um Objeto.

A classe reúne características que todos os objetos tem:
Todo objeto da classe Pessoa tem nome, tem RG, alguns tem profissão outros não, todos tem pais (embora alguns conheçam ou não)...a classe é uma 'receita' de criar objetos. Entendeu?

Vamos pra mais um exemplo?

A Classe Carro

Carro é um belo exemplo de classe.
Ele tem suas funções e características.
Ele serve para se locomover, para refrigerar o ar interno, para prover um somzinho interno, para buzinar, as suas várias partes cumprem cada uma de suas funções...e tem suas características, cada carro tem sua cor, tem seu número de porta, motorização, tipo de câmbio.

Mas você não chega numa concessionária:
- Olá, quero um carro
- Ok, aqui está um carro
- Obrigado, vou levar
- É seu, você comprou
- Vou dirigir meu carro

Não, carro é uma classe, é uma abstração.
Você compra objetos: você compra Gol, Celta, Uno, Civic, Corolla...esses são objetos da classe Carro.
São exemplos específicos.

Chamamos instâncias.

A instância Uno tem 2 ou 4 portas, motor 1.0, é bem básico, não tem muita coisa de fábrica.
Já a instância Corolla, também é um carro, possui motor 2.0 e é mais completo.

Mas todas as instâncias possuem coisas em comum:
Possuem portas
Possuem motor
Possuem cor
Possuem peças fazendo funções

Ou seja, classes são uma espécie de 'receitas' para fazer os objetos.

E aí, deu pra entender melhor ?
Notou que vivemos em um mundo de objetos?

Características da Orientação a Objetos: Polimorfismo, Herança e Encapsulamento

Neste tutorial, vamos estudar algumas vantagens da orientação a objetos, como o polimorfismo e a herança

▪ Polimorfismo em Orientação a objetos

Não se preocupe com a quantidade de teoria dada neste início, vamos aprender tudo depois na prática.

Poli quer dizer várias, morfos maneira. Ou seja, polimorfismos quer dizer várias maneiras de fazer a mesma coisa, e essa é uma característica da orientação a objetos.

Imagine que a classe Automóvel tenha a função ligar().
Ora, a maneira de ligar um carro é diferente da maneira de ligar uma moto, que é diferente da maneira de ligar um triciclo e é diferente da maneira de ligar uma mobilete. Mas todos são automóveis, e todos ligam, e todos tem a função (que em orientação a objetos chamamos de métodos) ligar().

Então, ao executar o método ligar() de um carro ou de uma moto, ele vai ligar, pode ser de uma maneira ou outra, mas vai ligar.
Isso é polimorfismo, é a capacidade de fazer algo de maneiras diferentes.

Se você tem um objeto chamado "moto", para ligar ela, vai acionar o método: moto.ligar()
Se tem uma "BWM", vai chamar o mesmo método: BMW.ligar()

Porém, cada um deles funciona de uma maneira diferente, internamente, embora todos sejam automóveis.
Isso é a ideia básica por trás do polimorfismo. O importante é ligar o automóvel.

Herança em Orientação a objetos

Uma das características mais interessantes da orientação a objetos, é a da herança, que é a capacidade que uma classe tem de herdar informações de outras classes.

Vamos supor a classe Funcionario.
Ela define todos os funcionários de uma empresa.
O que todos funcionários tem em comum?

Ué, salário, horário de trabalho, tarefas a fazer...

Porém, cada funcionário, de cada setor, tem coisas especificas a fazer.

Os funcionários de TI, herdam as características da classe Funcionário: tem salário, horários a cumprir e tarefas a fazer dentro da empresa.
Os funcionários de entrega, também herdam algumas características, como salário, horários a cumprir, mas não ficam dentro da empresa, ficam fazendo entregas fora dela.

O presidente da empresa também é um funcionário, também vai herdar características gerais da classe Funcionário, como salário, mas vai viajar, vai passar dias fora, nem sempre vai estar na empresa.

O que queremos dizer com isso, é que algumas classes podem herdar informações e métodos de outras classes ditas superiores, assim evitamos escrever muito código.

Dizemos que a classe Funcionario é uma superclasse.
A classe TI é uma subclasse, assim como a Entregador é uma subclasse, pois herdam características da classe Funcionario.
Uma subclasse herda atributos e métodos de uma superclasse.

Com o uso da herança, você poupa escrever código ao escrever uma classe que difere de outra apenas em algumas detalhes menores e mais específicos.

- ## Encapsulamento em Programação orientada a objetos

Até o momento, todo nosso código são ‘pedaços’ soltos, uma função aqui, um formulário ali, e as coisas meio que vão se encaixando.
Uma função, por exemplo, ela é do script inteiro, qualquer um pode ter acesso a ela e usar como bem entender, isso pode ser bom ou ruim.

Pode ser bom por facilitar a vida, já que ela deveria ser usada mesmo
Pode ser ruim porque algumas funções não deveriam ser usadas por algumas pessoas.
Por exemplo, a galera da TI não deveria ter acesso as funções da galera do setor da Tesouraria da empresa, concorda?

E é aí que vem o encapsulamento.
Quando criamos uma classe, definimos basicamente duas coisas dentro dela: informações (atributos) e ações (métodos) e estes são próprios.
Somente os próprios objetos tem acesso a seus próprios atributos e métodos.

Se quiser que um objeto do setor de RH converse com um objeto do setor de TI, você vai ter que fazer um método específico em ambos os lados para que ocorra essa conversa.
Um objeto não faz ideia do que ocorre dentro de outro. É um mundo a parte.
É o encapsulamento.

Por exemplo, os atributos e métodos do motor de um carro não são acessíveis pelo objeto ar-condicionado, assim esse não pode se meter naquele e ocasionar problema. É cada um no seu quadrado, entendeu?

Como Criar Classe, Objeto, Atributos e Métodos

Neste tutorial, vamos de parar com a teoria e de fato aprender como criar classes, métodos, atributos e métodos.

Como declarar uma classe: class

A sintaxe para a declaração de uma classe é:

```
class Carro
{
//codigo dos atributos
//codigo dos metodos
}
```

Simplesmente a palavra-chave class, o nome que queremos dar a nossa classe (recomendamos começar com maiúsculo), e o código da classe entre chaves. Prontinho.

Como declarar um objeto: new

Após declarar uma classe, já podemos instanciar (como chamamos criar) nossos objetos. Fazemos isso como se fossemos criar novas variáveis, mas elas vão receber: new e o nome da classe e um par de parêntesis.

Vamos criar alguns carros:

```
$gol = new Carro();
$celta = new Carro();
$corolla = new Carro();
```

Prontinho, agora $gol, $celta e $corolla são instâncias, são objetos e podemos trabalhar com eles diretamente. Mas eles ainda estão vazios, não tem nada dentro deles.
Que tal definimos a motorização deles?

▪ Como criar e acessar atributos

Atributos são características, em outras palavras, são variáveis dentro das classes. Vamos definir uma variável dentro da nossa declaração de classe:

```
class Carro
{
   var $motor;
}
```

Pronto.

Agora toda vez que você for instanciar um objeto dessa classe, essa variável vai junto. Vamos ver?
Vamos instanciar um Honda Fit:
$fit = new Carro();

Para acessar a varável $motor, usamos o operador: ->

Vamos definir como 1.5 o motor do seu Fitoso:
$fit->motor = 1.5;

Agora rode o código:

```php
<?php
	class Carro
	{

		var $motor;
	}

	$fit = new Carro();
	$fit->motor=1.5;

	echo "Motor do meu corolla $fit->motor";

?>
```

Como declarar e acessar métodos

Métodos nada mais são que funções, mas funções que só existem dentro de cada objeto.
Declaramos as funções dentro da classe, e ao instanciarmos os objetos, eles herdam dela.
Vamos criar duas funções, a ligar ligar() e a desliga():

```
<?php
	class Carro
	{
		var $motor;
		function liga()
		{
			echo "Ligando o carro...vruum!<br/>";

		}

		function desliga()
		{
			echo "Desligando o carro...fueeem ! <br />";
		}
	}
?>
```

Para acessar funções, fazemos:
$fit - >liga();
$fi - > desliga();

Teste:

```
$fit = new Carro();
$fit->motor=1.5;
$fit->liga();
echo "Motor do meu corolla $fit->motor<br />";
$fit->desliga();
```

Bem simples e bacana, essa orientação a objetos, não?

Propriedades SET e GET

Propriedades

Nunca é interessante deixar que nossos atributos e métodos fiquem 100% disponíveis e livres para qualquer um usar, acessar ou até mesmo alterar, isso pode ser uma falha e deixar brecha para hackers invadirem seus sistemas.

Então, para acessar e alterar alguns tipos de informações, usaremos alguns métodos especiais, chamados accessors ou mutators, cujo único objetivo é alterar ou acessar determinadas informações, mas sob o 'olhar' vigilante e sanguinário do vigia, digo, do objeto, para saber o que está acontecendo de fato ali.

As funções Get e Set

Set significa...setar, configurar, colocar...quando setamos uma variável, estamos impondo um valor nela.

O método setName($name) por exemplo, vai colocar o valor $name na variável $name do objeto.

Para não confundir as coisas, a variável do objeto é definida pro:
$this->name
E a variável que vem de fora por: $name

Então, um método setName() que seta um nome para a variável do objeto é:

```
public function setName($name) {
$this->name = $name;
}
```

Get significa pegar, acessar...usamos ela para ter controle, pegar alguma informação de uma variável de um objeto, por exemplo, a getName(), retorna o valor da variável $name de um objeto:

```
public function getName() {
return $this->name;
}
```

Note que mais uma vez usamos o $this que serve para se referir ao próprio objeto.

Classe Empregado

Então uma classe Empregado, onde pedimos o nome do usuário, setamos o nome dele e depois exibimos, ficaria:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>

	<form action=ae method="get">
		Digite o nome do funcionario<input type="text" name="name" /><br
/>
		<input type="submit" name="submit" value="Testar" />
	</form>

	<?php

		$nome = $_GET['name'];

	class Empregado {
		private $name;
		// Get
		public function getName() {
			return $this->name;
		}

		// Set

		public function setName($name) {
			$this->name = $name;
		}
	}

	$func = new Empregado();
	$func->setName($nome);
	echo "Funcionário: ".$func->getName();

?>
```

```
 </body>
</html>
```

- ## Uso real de SET e GET

O uso mostrado de set e get, são apenas para enfatizar que você não deve deixar que acessem diretamente os dados de atributos.
O saldo bancário, por exemplo, é um atributo.

O método get para acessar esse atributo, só deve ocorrer depois do usuário fornecer a senha ou digital dele, provando ser ele o titular da conta, só então ele vai poder acessar o atributo, entende?

O mesmo jeito o método set.
Vamos supor que ele tenha mil na conta e queira setar um saque de 1200.
Não dá amigo.

Você precisa fazer uma verificação antes de permitir qualquer set. Pra depositar, ok, pode depositar o que for, agora pra sacar, tem regras, não é só sair sacando, a vida real não é assim? Pois é exatamente assim nos sistemas reais bancários, ok?

Relembre: use métodos para deixarem que acessem suas variáveis,seus sagrados atributos.

Escopo de Atributos e Métodos: public, private, protected, final e static

Neste tutorial, vamos aprender algumas palavras-chaves que permitem que nosso escopo tenha um comportamento especial relativo a segurança dos dados contidos.

Escopo de Atributo Public

Você pode declarar seus atributos com a palavra chave public antes:

```
class Pessoa
{
public $nome;
}
```

Isso significa que ele pode ser acessado e alterado do lado de fora por qualquer pessoa ou entidade:

```
$alguem = new Pessoa();
$alguem->nome = "Neil Peart";
$name = $alguem->nome;
echo "Nome da pessoa: $name";
```

Escopo de Atributo Private

Esse só pode ser acessado de dentro da clase.

```
class Pessoa
{
private $nome;
}
```

Para definirmos um valor para esse atributo, temos que criar um método setter:

```
class Pessoa
{
   private $nome;
```

```
  public function setNome($name) {
    $this->nome = $name;
  }
}
```

- ## Escopo de Atributo Protected

Assim como algumas funções querem que variáveis existam somente dentro de seu escopo, podemos querer o mesmo em classes.
Para isso, usamos a palavra-chave protected:

```
class Pessoa
{
  protected $nome;
}
```

Porém, atributos do tipo protected estão disponíveis para serem acessados e modificados por classes herdadas, o que é uma características não existente no modo private.

- ## Escopo de Atributo Final

Fazendo com que uma variável seja final, você previne que ela seja sobrecarregada por uma subclase, ou seja, que outra classe defina um atributo 'por cima' na hora da herança.

```
class Pessoa
{
  final $nome;
}
```

Discutiremos esse assunto mais em detalhes adiante.

Escopo dos Métodos

Assim como atributos, métodos também possuem seus escopos de níveis de encapsulação.

O public, permite que você invoque ele de qualquer lugar do código, como já fizemos alguma vezes.
Já o privado é mais recluso, não pode ser invocado por subclasses ou objetos instanciados.

O protected pode ser invocado por suas subclasses e em qualquer lugar da classe, sem problemas.

Existe ainda um tipo especial, chamado de abstract.
Ele é declarado na classe pai, porém só é implementado nas classes filhas.

Por exemplo, a classe:

```
class Animal
{
   abstrc som();
}
```

Só isso.

Então a classe filha, que vai herdar a classe pai (ainda vamos estudar), é que implementa essa classe.
Se for um leão, implementa um som.
Se for uma cobra, implementa outro som.

Por fim, o escopo final, que assim como no atributo, evita que ela seja sobrescrita por qualquer classe herdada. Ela é a implementação única e final daquele método.

Método Construtor (Orientação a Objetos em PHP)

Neste tutorial, vamos aprender o que é e como usar o método construtor em orientação a objetos em PHP.

Criação de Objetos

Um objeto é cheio de informação, cheio de atributos, variáveis, funções, faz isso, faz aquilo e se você for lembrar de fazer tudo ao criar cada coisa, ficaria louco.

Imagina que é o chefe de TI de uma grande empresa, cada pessoa que entra você tem que registrar nome, setor, salário, carga horária...não ia dar, ia ser muito trabalho manual, e programador é preguiçoso por natureza.

Existe um método, chamado método construtor, que ele já cria um objeto com diversas informações, automaticamente.
É o método construtor.

Método Construtor

O método construtor é um método como outro qualquer, a primeira diferença é que ele tem o nome da classe. Sempre.
Se sua classe for "Funcionario", seu método vai se chamar "Funcionario".

O método construtor tem a característica de sempre iniciar ao criarmos um objeto.
Sempre que instanciamos um objeto, o método construtor é inicializado.

Teste:

```
<?php
	$nome = $_GET['name'];
	class Empregado {
		var $nome;
		function Empregado()
		{
			echo "Construtor invocado";
		}
```

```
        }

        $func = new Empregado();
?>
```

Vai aparecer "Construtor invocado".
Agora vamos fazer com que este construtor sete a variável $nome da classe.

Para isso, ao instanciarmos o objeto $func, precisamos passar uma variável. Depois, imprimimos a variável $nome do objeto para ver se o método construtor inicializou ela corretamente, veja:

```
<?php
        $nome = $_GET['name'];
        class Empregado {
                var $nome;
                function Empregado($nome)
                {
                        $this->nome=$nome;
                }

        }

        $nome = "João";
        $func = new Empregado($nome);
        echo $func->nome;

?>
```

Data e Tempo em PHP

Já percebeu que ao entrar em seu internet banking, fica um relógio contando quanto tempo você ainda tem pra usar aquela sessão?

E o preço dos sites de compras que você acessa, será que é recente?
A notícia do portal, é de quando?
Já percebeu que tem site que atualiza sua página depois de um tempo?

Pois é, tudo isso está ligado com conceitos de datas e tempo, que iremos aprender nesta seção de nosso **curso de PHP**.

Como checar se uma data é válida em PHP: checkdate()

Neste tutorial, vamos aprender como verificar se uma determinada data é válida (existente) ou não, em PHP, através da função **checkdate()**.

Verificando datas: Função checkdate()

Muitas vezes, durante sua carreira de desenvolvedor web PHP, você vai precisar verificar se uma determinada data é válida, ou seja, se ela realmente existiu ou pode existir, como por exemplo, a data de nascimento de uma pessoa.

Se ela colocou 30/02 de qualquer ano, saiba que é um *bot*, um vírus ou alguém tentando 'trollar' o sistema, pois fevereiro não tem 30 dias.

O PHP possui uma função pré-definida, a **checkdate()**, que determina se uma data é existente ou não e sua sintaxe é:
boolean checkdate(int month, int day, int year)

Ou seja, você fornece três inteiros: mês, dia e ano e ela te retorna TRUE se a data for válida e FALSE se não existir.

Note que primeiro vem o mês. Diferente de nós do Brasil que usamos o dia primeiro, lá fora é o mês que vem antes, ok?

Por exemplo:
checkdate(5,5,1988)
retorna TRUE

checkdate(2, 29, 2019)
retorna FALSE, pois 2019 não é bissexto

checkdate(2,29,2020)
retorna TRUE, pois 2020 é ano bissexto

- **Exemplo de checkdate**

O script abaixo pede dia, mês e ano ao usuário e verificar se tal data é válida ou não:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <form action="" method="get">
                Dia<input type="text" name="day" /><br />
                Mês<input type="text" name="month" /><br />
                Ano<input type="text" name="year" /><br />
                <input type="submit" name="submit" value="Testar" />
        </form>
        <?php
                $dia = $_GET['day'];
                $mes = $_GET['month'];
                $ano = $_GET['year'];

                if(checkdate($mes, $dia, $ano))
                        echo 'Data válida';
                else
                        echo 'Data inválida';
        ?>
 </body>
</html>
```

Como exibir Datas em PHP: date()

Neste tutorial, vamos aprender como exibir todo e qualquer tipo de data em PHP, usando a função PHP e seus infinitos atributos.

Exibir datas em PHP

Uma coisa obrigatória que você vai precisa fazer em sua carreira de desenvolvedor web, é exibir datas,

Exibir dia da semana, do mês, do ano, minutos do dia, quanto segundos faltam pra uma sessão encerrar, quanto tempo o usuário vai estar na sua página e por ai vai.

Porém, o PHP fornece uma função fantasticamente maravilhosa, a **date()** que possui trocentos parâmetros, para você exibir datas praticamente do jeito que você quiser:

- Só os dias
- Dia da semana
- Nome do mês
- Abreviação do mês
- Semana do ano
- Segundos
- Minutos
- Até diferença pro meridiano de Greenwich

É uma função poderosíssima.

Sua sintaxe é:
string date (string parametros)

▪ Parâmetros da função date()

Parâmetros	Descrição	Exemplo
a	Antes e pós meridiano, em minúscuo	am or pm
A	Antes e pós meridiano, em maiúsculo	AM or PM
d	Dia do mês, com 0 na frente	01 to 31
D	Dia da semana representado por três letras	Mon, Sun, etc
F	Completa representação do mês	January até December
g	Horário no formato de 12 horas, sem 0	1 até 12
G	Horário no formato de 24 horas, sem 0	1 até 24
h	Horário no formato de 12 horas, com 0	01 até 24
H	Horário no formato de 24 horas, com 0	01 até 24
i	Minutos, com zero	01 até 60
I	Horário de verão	0 se não, 1 se sim
j	Dia o mês, sem o zero	1 até 31
l	Texto representando o dia da semana	Monday até Sunday
L	Ano bissexto	0 se não, 1 se sim
m	Representação numérica do mês, com 0	01 até 12
M	Três letras represetando o mês	Jan até Dec
n	Representação numérica do mês, sem o 0	1 até 12
O	Diferença pro meridiano de Greenwich (GMT)	–0500
r	Data formatada de acordo com o RFC 2822	Tue, 19 Apr 2005 22:37:00 –0500
s	Segundos, com zeros	01 até 59
S	Sufixo ordinário do dia	st, nd, rd, th
t	Números de dias de um mês	28 até 31
T	Definição de fuso horário da máquina executora	PST, MST, CST, EST, etc.
U	Segundos desde a época do Unix	1114646885
w	Representação numérica do dia da semana	0 para domingo e 6 para sábado
W	ISO-8601 numero da semana do ano	1 até 53
Y	Representação do ano em 4 dígitos	1901 até 2038 (Unix) 1970 atẽ 2038 (Windows)
z	Dia do ano	0 até 365
Z	Deslocamento do fuso horário em segundos	–43200 through 43200

- ## Exemplos de uso:

```
echo "Hoje é ".date("d m, Y");
// Hoje é 23 ,02, 2019

echo "Hoje é ".date("F d, Y");
//Hoje é February 23, 2019
```

Você pode ainda formatar do jeito que quiser:

```
$diadasemana = date("l");
$numerodia = date("dS");
$mesano = date("F Y");
```

Fazendo:

```
printf("Hoje é %s o %s dia de %s", $diadasemana, $numerodia, $mesano);
```

O resultado é:
Hoje é Saturday o 23rd dia de February 2019

- ## Trabalhando com horários

Para exibir um horário, faça:

```
echo "A hora exata é ".date("h:i:s");
```

Resultado:
A hora exata é 01:42:27

Notou o poder e versatilidade da função **date()?**

A Unix Timestamp e Calculando Segundos com a função mktime()

Neste tutorial de nossa **apostila de PHP**, vamos aprender o que é a famosa *Unix Timestamp* bem como vamos aprender a calcular segundos a partir dela com a função mktime.

A Unix Timestamp - A Marca Temporal Universal

Imagine que seu chefe te peça um serviço web onde o usuário digita duas datas completas, com dia, mês, ano e horário, e você tenha que dizer o tempo entre as duas.

Por exemplo:

- 5 de abril, 2015, 18h45min e 34s
- 4 de novembro, 1987, 19h00min e 12s

Seria um trabalho altamente maçante e chato de fazer, não concorda?
A solução para isso é simples:
O primeiro passo é definir uma marca temporal, é a do Unix, a timestamp, e ela é a data 1 de janeiro de 1970, as 00:00:00 UTC (Coordinated Universal Time).

Depois, basta calcular quantos segundos ocorreram desde a timestamp até hoje e lidar apenas com esse número, fica mais fácil trabalhar com essa formalização.

A função mktime()

A função mktime() recebe a data completa e retorna os segundos transcorridos desde a marca temporal do Unix, sua sintaxe é:
int mktime(int hora, int minuto, int segundos, int mes, int dia, int ano);

Vamos usar nossos exemplos:
5 de abril, 2015, 18h45min e 34s: echo mktime(18, 45, 34, 4, 5, 2015)
Retorna: 1428270334

4 de novembro, 1987, 19h00min e 12s: echo mktime(19,0,12,11,4,1987)
Retorna: 563058012

Diferença temporal: 1428270334 - 563058012 = 865212322s
A partir daí fica mais fácil trabalhar e transformar em minutos, horas, dias...

▪ Exercício com a mktime()

Crie um script que pede o dia, mês, ano e horário de nascimento de uma pessoa e exiba quantos segundos se passaram da timesamp do Unix até a data de nascimento dela.

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <form action="" method="get">
                Dia<input type="text" name="day" /><br />
                Mês<input type="text" name="month" /><br />
                Ano<input type="text" name="year" /><br />
                Hora<input type="text" name="hour" /><br />
                Minuto<input type="text" name="min" /><br />
                Segundo<input type="text" name="sec" /><br />
                <input type="submit" name="submit" value="Testar" />
        </form>
        <?php
                $dia = $_GET['day'];
                $mes = $_GET['month'];
                $ano = $_GET['year'];
                $hora = $_GET['hour'];
                $minuto = $_GET['min'];
                $segundo = $_GET['sec'];

                echo "Se passaram: ".mktime($hora,$minuto,$segundo,$mes,$dia,
$ano)."
                        segundos desde que você nasceu";

        ?>
 </body>
</html>
```

▪ Exercício com a mktime()

Agora faça um script que peça a data de aniversário completa do usuário e calcule o tanto de segundos que ele viveu.
Dica: use os parâmetros da função Date()

Solução
O grande segredo é usar *date("U")*, que já retorna os segundos do momento atual até a timestamp.

Nosso código fica assim:

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
        <form action=ae method="get">
                Dia<input type="text" name="day" /><br />
                Mês<input type="text" name="month" /><br />
                Ano<input type="text" name="year" /><br />
                Hora<input type="text" name="hour" /><br />
                Minuto<input type="text" name="min" /><br />
                Segundo<input type="text" name="sec" /><br />
                <input type="submit" name="submit" value="Testar" />
        </form>
        <?php
                $dia = $_GET['day'];
                $mes = $_GET['month'];
                $ano = $_GET['year'];
                $hora = $_GET['hour'];
                $minuto = $_GET['min'];
                $segundo = $_GET['sec'];

                $data_nascimento=mktime($hora,$minuto,$segundo,$mes,$dia,
$ano);
                $data_atual=date("U");

                echo "Você viveu: ".($data_atual-$data_nascimento).
                        " segundos de vida até o momento.";

        ?>
 </body>
</html>
```

Obtendo datas e horários em PHP: Função getdate()

Neste tutorial de nosso **curso de PHP**, vamos aprender a usar a função *getdate()* que nos fornece de maneira completa datas e horários no tempo.

A função getdate()

A sintaxe da função é:
array getdate (int timestamp)

Ou seja, devemos fornecer um *timestamp* apenas e ela nos retorna um array completo, de 11 elementos:

1. hours: Representação numérica da hora do dia, de 0 até 23
2. mday: Representação numérica do dia do mês, de 1 até 31
3. minutes: Representação numérica dos minutos, de 0 até 59
4. mon: Representação numérica do mês, de 1 até 12
5. month: Nome do mês, escrito, por exemplo Maio
6. seconds: Representação numérica dos segundos, de 0 até 59
7. wday: Representação numérica do número da semana. Por exemplo, 0 é domingo
8. weekday: Nome do dia da semana, por exemplo, Segunda-feira
9. yday: Representação numérica do dia do ano, de 0 até 365
10. year: Representação numérica do ano, em 4 dígitos, como 2020
11. 0: Timestamp, número de segundos desde a Unix epoch

Bem completa essa getdate(), não?

Exemplo de uso da getdate()

No momento que vos escrevo este tutorial, a atual timestamp é:

- 1550931619

Vamos criar um array com esta data:
$data = getdate(1550931619);

Agora vamos printar este array:

print_r ($data)

(Lembre-se: use print_r para printar arrays)

O resultado é:
Array (
[seconds] => 19
[minutes] => 20
[hours] => 11
[mday] => 23
[wday] => 6
[mon] => 2
[year] => 2019
[yday] => 53
[weekday] => Saturday
[month] => February
[0] => 1550931619)

- ## Exemplo de uso da getdate()

Crie um script que pega o timestamp ATUAL, desse exato momento (use o parâmetro date correto), mande pra getdate() e imprima o array.

Mostre o resultado nos comentários.

Um código mais fácil de entender:
$data = date("U");
print_r (getdate($data);

Ou simplesmente:
print_r(getdate(date("U")));

Como obter tempo atual: Função gettimeofday() do PHP

Neste tutorial de nosso **curso de PHP**, vamos aprender para que serve e como usar a função gettimeofday(), para pegar as informações do momento atual do sistema.

Obtendo informações atuais: gettimeofday()

Para obtermos algumas informações do exato instante que o sistema PHP está rodando, usamos a função pré-definida gettimeofday().

Quando simplesmente invocamos ela: gettimeofday()
Ela retorna um array, com os seguintes elementos:

- [sec] - Número de segundos desde a *Unix Epoch*
- [usec] - microssegundos atual
- [minuteswest] - a quantos minutos você está a oeste do meridiano de Greenwich
- [dsttime] - um inteiro, que representa o horário de verão:

0 - Sem horário de verão
1 - Horário dos EUA
2 - Horário da Australia
3 - Horário do oeste europeu
4 - Horário da europa meridional
5 - Horário do leste europeu
6 - Horário do Canadá
7 - Horário de Grã-Bretanha e Irlanta
8 - Horário da Romênia
9 - Horário da Turquia
10- Horário da Austrália, variação de 1986

Quando eu dou: print_r(gettimeofday());
O que aparece atualmente para mim é:
Array (
[sec] => 1551025438
[usec] => 880119
[minuteswest] => 180

[dsttime] => 0)

E para vocês? Deixem nos comentários.

- ## Como pegar exato instante atual: gettimeofday(TRUE)

A função gettimeofday()) também aceita como parâmetro o booleano TRUE. Quando usamos ele, ela retorna apenas um float, com o instante exato do momento atual, com segundos da *Unix Epoch* e mais 4 casas decimais de microssegundos.

Fazendo: print_r(gettimeofday(TRUE));
Obti: 1551025529.2385

E vocês?

- ## Exercício: Tempo de execução de um script

Uma das funcionalidades da gettimeofday() é descobrir o tempo de execução de um script em PHP.

Primeiro, pegamos o tempo inicial:
$d1 = gettimeofday(TRUE);

Fazemos nosso scripts.

Depois, pegamos o tempo final:
$d2 = gettimeofday(TRUE);

Prontinho, basta subtrair esses valores e teremos o tempo de execução de um script.

Vamos testar com um "Olá, mundo":

```
<?php
	$d1 = gettimeofday(TRUE);
	echo "Olá mundo<br />";
	$d2 = gettimeofday(TRUE);

	echo "Tempo de execução: ".($d2-$d1);
?>
```

O resultado aqui foi:
Tempo de execução: 9.5367431640625E-7

Ou seja: 0,000000954s

Bem rápido esse PHP, não?
Não é por menos que é uma das linguagens mais utilizadas e amadas do mundo. E você está se tornando profissional nela.

Usando as funções time() e date()

Neste tutorial de nossa **apostila de PHP**, vamos aprender a usar a função time() junto da função date() e aprender poderosas técnicas de manipulações de datas.

A função time()

A função time é super mega hiper simples.
Basta invocar: time()

Ela retorna um inteiro: o Unix timestamp atual.
Prontinho. Só isso.

Por exemplo, eu rodando aqui agora: echo time();
O resultado é: 1551028815

Mas convenhamos, esse tanto de segundo, não é muito útil, não quer dizer nada, não dá pra dizer quantos dias, meses ou anos tem aí, não é muito 'legível' para humanos'.

Concorda?

Função date() com função time()

Lembra da função date() ? Responsável por exibir datas e horários de tudo quanto é jeito que você imaginar?

Pois é, podemos combinar ela com a time e exibir datas de maneira que humanos consigam entender e captar de maneira mais interessante.

Por exemplo, para exibirmos uma data no formato: dia-mes-ano
Fazemos: echo date("d/F/Y", 1551028815);

Note que passamos uma *timestamp* que obtivemos com a *time()*.

Podemos ir mais além e mostrar hora:minuto:segundo:
echo date("d/F/Y - h:i:s", 1551028815);

Ou seja, pegamos o *timestamp* atual com a função time() e a função date() com seus inúmeros parâmetros, transforma esse número medonho bizarro em datas bonitas e prazerosas.

Exemplo de uso de time()

Crie um script que exibe o horário daqui exatamente uma semana.

Vamos pegar a timestamp atual:
$agora = time();

Agora vamos somar os segundos de uma semana:
$agora += 7*24*60*60;

O script que exibe o dia, mês, ano e horário atual e o exatamente daqui uma semana é:

```
<?php
	$proxsemana = time()+(7 * 24 * 60 * 60);
	echo "Atual:" .date("d/F/Y - h:i:s", time())."<br/>";
	echo "Semana que vem: ".date("d/F/Y - h:i:s", $proxsemana)."<br/>";
?>
```

Você pode usar isso para fazer promoções em sites, por exemplo "Você tem apenas uma semana para aproveitar a queima de estoque..."

Data e hora em Português no PHP: setlocale() e strftime()

Neste tutorial de nosso **Curso de PHP**, vamos aprender a usar datas e horários locais, no formato e escrita de nossa língua, através das funções setlocale() e strftime().

Formatos locais: setlocale()

Até o momento, todas nossas datas e horários tem o formato americano, ou seja:
12/23/2019

Primeiro o mês, depois o dia, depois o ano.
Sendo que no Brasil primeiro usamos o dia primeiro, só depois o mês.

Quando fizemos:
echo date("d/F/Y", 1551028815);

O resultado foi: 24/February/2019
Mas não queremos 'February', somos brasileiros!

Queremos é Fevereiro!
A resposta para isso é setar o local, com a *setlocale*, basta fazer:
setlocale(LC_ALL, 'pt_BR');

Experimente:

```
<?php
	setlocale(LC_ALL, 'pt_BR');
	echo strftime('%d/%B/%G');
?>
```

Se não funcionar, experimente:

```
<?php
	setlocale(LC_ALL, 'pt_BR.utf8');
	echo strftime('%d/%B/%G');
?>
```

Se vai ser pt_BR ou não, vai depender do que tem instalado em seu servidor.
LC_ALL é uma constante que serve para definir várias coisas a respeito das datas locais que você vai usar.

Note que não usamos date(), e sim strftime()

A função strftime()

A função date() serve apenas para exibir as coisas em inglês padrão.
Se quisermos usar outra língua para trabalharmos com data e tempo, precisamos da strftime(), mas ela é bem semelhante a date(), mas usa outros parâmetros, veja:

%a - dia da semana abreviado
%A - dia da semana completo
%b - dia do mês abreviado
%B - dia do mês completo
%c - data e hora padrão 04/26/05 21:40:46
%C - Número do século 21
%d - Dia numérico do mês, com 0 na frente
%D - Equivalente a %m/%d/%y
%e - Dia numérico do mês, sem o 0 na frente
%g - Exibe o ano, mas sem o século (tipo 19 ao invés de 2019)
%G - Ano numérico
%h - Mesmo que %b
%H - Horário numérico de 24 horas com 0 na frente
%I - Horário numérico num relógio de 12 horas, com 0 na frente
%j - Número do dia do ano de 001 até 366
%m - Número do mês, com 0 na frente
%M - Número do minuto,com 0 na frente, de 00 até 59
%n - Caractere de quebra de linha \n
%p - Antes e pós meridiano AM, PM
%r - Antes e pós meridianos com ponto A.M., P.M.
%R - Notação de 24 horas 00:01:00 até 23:59:59
%S - Número de segundos com zero, de 00 até 59
%t - Caractere de tab \t
%T - Equivalente a %H:%M:%S 12:23:41
%u - Dia numérico da semana, onde 1 = domingo
%U - Número numérico da semana, em que o primeiro domingo é primeiro dia da primeira semana
%V - Número numérico da semana, em que semana 1=primeira semana com>=4 dias, vai de 01 a 53

%W - Número numérico da semana, em que a primeira segunda-feira é primeiro dia da primeira semana
%w - Número do dia da semana, onde 0 = domingo
%x - Padrão de data local d:m:a
%X - Padrão de tempo local h:m:s
%y - Número do ano sem o século
%Y - Número do ano, com o século
%Z ou %z - Fuso horário do horário de verão oriental
%% - Caractere de porcentagem %

Última Modificação de uma Página: getlastmod()

Quase sempre, é muito importante saber qual foi a última vez que uma página foi modificada pela última vez.

Imagine que você tenha um site de promoções e vendas, é essencial exibir a data e o horário nele, pois os usuários precisam saber se ele está sendo atualizado, se as informações estão 'fresquinhas'.

- ## Última Alteração da Página: Função getlastmod()

O PHP possui uma função pré-definida muito útil, que exibe a última alteração de uma página, é a getlastmod().

Teste:

```
<?php
	echo getlastmod();
?>
```

Resultado aqui:
1551106973

Ou seja, ele vai te retornar a *timestamp*.

Mas você não vai exibir isso pro usuário:
Última atualização: 1551106973 segundos desde 1 de janeiro de 1970.

Vamos ter que usar a strftime() junto com a getlasmod() para formatar corretamente (ou a date() caso prefira exibir os dados em inglês).

Veja como fica nosso código:

```
<?php
	setlocale(LC_ALL, 'pt_BR.utf8');
	echo "Última atualização: ".strftime('%d/%B/%G as %H:%M:%S ',
getlastmod() );
?>
```

Resultado:

Última atualização: 25/fevereiro/2019 as 12:08:28

Bem mais bonito, não ?

Determinar Número de Dias em um mês no PHP

Uma grande utilidade de se trabalhar com datas, é saber quantos dias algum mês possui.

Neste tutorial vamos te mostrar scripts que te ensinam a fazer isso de maneira bem simples.

Número de Dias no mês atual

O código date('t') diz quantos dias o mês possui.
O mês em questão é representado por: date('F')

Veja: A função date() e seus parâmetros no PHP

Assim, um script que mostra o número de dias do mês atual, seria:

```
<?php
	echo "Há ".date('t')." dias no mês ".date('F');
?>
```

A saída é: Há 28 dias no mês February

Se preferir 'mês 02' ao invés de 'mês February', use:

```
<?php
	echo "Há ".date('t')." dias no mês ".date('m');
?>
```

Mas se preferir: "O mês de Fevereiro tem 28 dias", basta fazer:

```
<?php
	setlocale(LC_ALL, 'pt_BR.utf8');
	echo "O mês ".strftime('%B')." tem ".date('t')." dias";
?>
```

- ## Número de Dias de Qualquer mês de qualquer ano

E quantos dias teve fevereiro de 1988?
Para cálculos assim, usamos a mktime(), lembra dela?
mktime($hora, $minuto, $segundo, $mes, $dia, $ano);

Basta fazer:
$ultimo = date('d', mktime(0, 0, 0, 3, 0, 1988));

Ou seja, você vai pegar o dia '0' do mês '3'. É o mesmo que pegar o último dia do mês 2, entendeu?

Prontinho, a variável $ultimo vai ter o número de dias que teve fevereiro de 1988.

Calculando Datas no Futuro e no Passado: strtotime()

Muitas vezes, é necessário que saibamos a data exata para daqui algum tempo.

Por exemplo, qual a data daqui 40 dias?
Que data foi 20 dias atrás?

Obviamente, não vamos ficar fazendo contas nas mãos.
Para isso, temos a função strotime()

Calculando datas no futuro: strotime

Ela é uma função nativa do GNU Linux e possui uma sintaxe bem simples e acessível.
Para exibir uma data daqui 40 dias:

```
<?php
	$futuro = strtotime("40 days");
	echo date("d/F/Y", $futuro);
?>
```

Para calcular daqui 3 semanas:

```
<?php
	$futuro = strtotime("3 weeks");
	echo date("d/F/Y", $futuro);
?>
```

2 semanas e 4 dias:

```
<?php
	$futuro = strtotime("2 weeks 4 days");
	echo date("d/F/Y", $futuro);
?>
```

▪ Calculando datas no passado: strotime

A sintaxe é a mesma, mas vamos usar valores negativos na função strotime().
Para saber que dia foi 20 dias atrás:

```
<?php
	$passado = strtotime("-20 days");
	echo date("d/F/Y", $futuro);
?>
```

Uma semana atrás:

```
<?php
	$passado = strtotime("-1 week");
	echo date("d/F/Y", $passado);
?>
```

A Classe DateTime - Trabalhando com Datas e Times em Orientação a Objetos

Neste tutorial, vamos aprender como trabalhar com datas e tempo através da programação orientada a objetos em PHP.

Criando objetos da classe DateTime

A classe DataTime provém uma série de métodos, atributos e possibilidades que nos permitem trabalhar de maneira muito mais fácil com datas e tempo, usando orientação a objetos.

Vamos criar um objeto desta classe:
$date = new DateTime();

Se dermos um print_c($date);, veremos que no resultado é o atual, do momento da criação do objeto:
DateTime Object (
[date] => 2019-02-28 11:06:53.684238
[timezone_type] => 3
[timezone] => America/Sao_Paulo)

É um objeto do momento atual, no momento que estou programando este tutorial.

Construtor da classe DateTime

Podemos, porém, já inicializar com uma data, no formato: ano-mes-dia, para criar um objeto com uma data específica.

Vamos escolher uma data aleatória:
$date = new DateTime("2020-02-28");

O resultado da print_r($date); é:
DateTime Object (
[date] => 2019-02-28 00:00:00.000000
[timezone_type] => 3
[timezone] => America/Sao_Paulo)

▪ Formatando datas e horas: Método format()

A classe DateTime possui o método format() que usa os mesmos parâmetros da função date(), assim podemo exibir o formato de data e hora do jeito que quisermos.

Por exemplo, para exibirmos no formato brasileiro de: dd/mm/YYYY Hora:minuto:segundo, basta fazer:

```
$date = new DateTime();
echo $date->format( "d/m/Y H:i:s" );
```

▪ Somando e Subtraindo datas

Que tal saber qual foi a data de 35 dias atrás?
Basta fazer:

```
$date = new DateTime('-35 days');
echo $date->format( "d/m/Y H:i:s" );
```

E qual data vai ser daqui 2 semanas e 10 dias?

```
$date = new DateTime('+2 weeks 10 days');
echo $date->format( "d/m/Y H:i:s" );
```

▪ Diferença entre datas: Método diff()

Imagina calcular, de maneira instantânea a diferença de anos, meses, dias, horas, minutos e até segundos daqui até uma outra data?

Vamos pegar a data atual:

```
$hoje = new DateTime();
```

Agora uma futura qualquer:

```
$futuro = new DateTime('2020-02-27');
```

O array $diff vai armazenar a diferença entre $futuro e $hoje, assim:

```
$diff = $hoje->diff($futuro);
```

Para exibir o resultado, faça:

```
<?php
    $hoje = new DateTime();
```

```
        $futuro = new DateTime('2020-02-27');

        $diff = $hoje->diff($futuro);

        echo "Diferença de: <br/>"
                .$diff->days . " dias <br/>"
                .$diff->m . " meses <br/>"
                .$diff->y . " anos <br/>"
                .$diff->h . " horas <br/>"
                .$diff->i . " minutos <br/>"
                .$diff->s . " segundos<br/>";
    ?>
```

O resultado aqui foi:
Diferença de:
363 dias
11 meses
0 anos
12 horas
28 minutos
28 segundos

Bem preciso esse PHP, não ?

- ## Modificando Data e Hora: setDate() e setTime()

Após criar e definir seu objeto do tipo DateTime, é totalmente possível mudar seus atributos, como ano, dia, hora ou o que for, através dos *setters*.

Vamos definir a data atual e exibir ela:

```
$data = new DateTime();
echo $data->format( "d/m/Y H:i" )."<br />";
```

O resultado é: 28/02/2019 11:37 (sim, tá quase na hora do almoço)

Agora vamos mudar a data pra 01/03/2019, amanhã e o horário para 12:00. Nosso código fica assim:

```
$data->setDate(2019,03,01);
$data->setTime(12,0,0);
echo $data->format( "d/m/Y H:i" )."<br />";
```

E o resultado assim: 01/03/2019 12:00

Bonitinho, não?

Comparando Datas: var_dump

Muitas vezes é preciso saber se uma data é igual a outra, se é anterior, posterior...
Vamos criar uma variável que representa hoje e outra amanhã:

```
$hoje = new DateTime();
$amanha = new DateTime('2019-03-01');
```

Vamos comparar se elas são iguais:

```
var_dump($hoje == $amanha);
```

O resultado vai ser: bool(false)

Agora vamos comparar se amanhã é maior que hoje:

```
var_dump($amanha > $hoje);
```

O resultado vai ser: bool(true)

Que bom que deu verdade, né?

É possível ainda trabalhar com fuso-horário, somar intervalos de tempo, subtrair, definir língua e local que vai usar e uma porção de outros métodos para você usar, para aprender mais, acesse a documentação oficial do PHP: http://php.net/manual/pt_BR/class.datetime.php

Tutoriais

Script PHP: Informa o IP do usuário

IP (*Internet Protocol*) é um número que todo dispositivo que está conectado a uma rede recebe, de modo a identificá-lo unicamente. Internet que usa o protocolo do tipo IP, claro.

Costumava ser um número de 32 bits (IPv4), mas com o crescimento da grande rede, teve que ser alterado para 128 bits (IPv6).

Para melhor entendimento de humanos, são divididos em 4 blocos e são números do tipo:
http://192.168.1.2

- ### IP do usuário (Script em PHP)

```php
<?php
 //verifica se o IP é do tipo correto
 if (!empty($_SERVER['HTTP_CLIENT_IP']))
   {
  $ip_address = $_SERVER['HTTP_CLIENT_IP'];
   }
 //verifica se o IP é de algum proxy
 elseif (!empty($_SERVER['HTTP_X_FORWARDED_FOR']))
   {
  $ip_address = $_SERVER['HTTP_X_FORWARDED_FOR'];
   }
 //verifica se o IP é de um endereço remoto
 else
   {
  $ip_address = $_SERVER['REMOTE_ADDR'];
   }
 echo $ip_address;
?>
```

Como Exibir Informações do Browser do Usuário (Script PHP)

Browser é o programa que usamos para acessar a internet, como Chrome, Firefox, Opera, Internet Explorer entre outros.

Ao acessarmos uma URL, essas URL's solicitam ao browser informações do usuário.
É possível exibir para o usuário as informações dele, de *user agent*, ou seja, o agente do usuário, por exemplo.

Nessa informação, é dito o navegador do usuário, seu sistema operacional, de quantos bits é seu SO, WebKit e outras informações.

Tudo isso está armazenado em uma simples variável, a:
$_SERVER['HTTP_USER_AGENT']

Veja como usar:

- **Script em PHP: Informações do navegador do cliente**

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
 <?php
  echo "Informações do browser (User agente):" . $_SERVER
['HTTP_USER_AGENT'];
 ?>
 </body>
</html>
```

Script PHP que detecta se o HTTPS ou HTTP estão ativados

Nos últimos anos, nos deparamos com diversas mensagens de erros, em diversos browsers, ao entrar em sites HTTP.

O HTTP (Hypertext Transfer Protocol) é um protocolo de aplicação, responsável por trabalhar com hypertextos, como páginas HTML.

Já HTTPs é uma extensão do HTTP, onde o *s* é de *secure*, ou seja, é mais voltada para a segurança de comunicação numa rede de internet e está sendo vastamente utilizada e é altamente recomendada, tendo várias vantagens, como a encriptação de dados.

O script que mostra se o usuário está numa conexão HTTP ou HTTPS é bem simples, e basta verificar a variável $_SERVER['HTTPS'].

- ## Script PHP: Exibe se a Página é HTTPS ou HTTP

```php
<?php
 if (!empty($_SERVER['HTTPS']))
   echo 'HTTPS está ativo';
 else
  echo 'HTTP está ativo'."\n";
?>
```

Como redirecionar para outra página em PHP

Se em algum momento de seu sistema web você precisar redirecionar o usuário para a página atual para outra, é muito simples, bastando usar a função **header()**, que lida com esse tipo de assunto.

Como redirecionar em PHP

No exemplo abaixo, redirecionamos o usuário para a home do PHP Progressivo:

```
<?php
 header('Location: https://www.phpprogressivo.net/');
 die();
?>
```

Esse comando é muito utilizado para redirecionar o usuário para uma outra página do site, após ele fazer uma compra, por exemplo.

O die() é importante para terminar a execução do script, para se assegurar que nada mais depois do redirecionamento seja executado.

Mais informações:
http://php.net/manual/pt_BR/function.header.php
http://php.net/manual/pt_BR/function.die.php

Como Atrasar a Execução por Alguns Segundos (Script em PHP)

A função **sleep()** recebe como argumento um número de segundos, e 'trava' o script, faz sua página 'dormir', sem que nada aconteça por aquele determinado período de tempo. Veja.

- ## Como Atrasar a Execução de um Script

```
<html>
 <head>
  <title>Apostila PHP Progressivo</title>
 </head>
 <body>
 Atrasando 3 segundos...<br />
 <?php
   // dorme por 3 segundos
   sleep(3);
   // acorda
   echo "Obrigado por esperar!"
 ?>
 </body>
</html>
```