VERSÁTIL
tecnologia em informação

Introdução:

O PHP é uma linguagem de script do servidor e uma ferramenta poderosa para criar páginas da Web dinâmicas e interativas.
O PHP é uma alternativa amplamente utilizada, gratuita e eficiente para concorrentes, como o ASP da Microsoft.

Antes de começar, você deve ter uma compreensão básica de:

- HTML
- CSS
- JavaScript

O que é o PHP?

PHP é um acrônimo para "PHP: Hypertext Preprocessor"

O PHP é uma linguagem de script de código aberto amplamente utilizada para leitura dos scripts executados em um servidor.

O que é um arquivo PHP?

Arquivos PHP podem conter texto, HTML, CSS, JavaScript e código PHP. O código PHP é executado no servidor e o resultado é retornado ao navegador como HTML simples. Arquivos PHP têm extensão ".php". **O que o PHP pode fazer?**

- O PHP pode gerar conteúdo de página dinâmico.
- O PHP pode criar, abrir, ler, escrever, excluir e fechar arquivos no servidor.
- PHP pode coletar dados de formulário.
- O PHP pode enviar e receber cookies(informações).
- PHP pode adicionar, excluir, modificar dados em seu banco de dados.
- PHP pode ser usado para controlar o acesso do usuário.
- PHP pode criptografar dados.

Com PHP você não está limitado a produzir HTML. Você pode produzir imagens, arquivos PDF e até filmes em Flash. Você também pode enviar qualquer texto, como XHTML e XML.

Por que PHP?

O PHP é executado em várias plataformas (Windows, Linux, Unix, Mac OS X, etc.)

O PHP é compatível com quase todos os servidores usados hoje (Apache, IIS, etc.)

O PHP oferece suporte a uma ampla gama de bancos de dados

O PHP é gratuito. Faça o download do recurso PHP oficial: www.php.net

Para começar a usar o PHP você precisa:

Host(plataforma) com suporte PHP e MySQL

Instalar um servidor web(Apache) no seu PC e, em seguida, instale PHP e MySQL

Sintaxe básica do PHP

Um script PHP pode ser colocado em qualquer lugar no documento.
Começa com

<? Php e termina com**?>**
<?php // O código vai aqui ?>

A extensão de arquivo padrão para arquivos PHP é ".php". Um arquivo PHP normalmente contém tags HTML e algum código de script PHP.

Para processar arquivos com extensão .php é necessário copiá-lo para o diretório do servidor. Caso esteja utilizando o **WAMP**, basta clicar no ícone quando estiver verde (ativado todos os serviços) e clicar em **www directory**. Para reproduzir o arquivo acesse o **Localhost** também pelo ícone do **WAMP** ou digite esse comando no navegador. Seu arquivo estará em **Your Projects** na página principal. Organize todos os projetos em pastas!

PHP é uma linguagem de programação bem versátil e muito familiar a linguagem HTML. O código **PHP** está escrito entre as tags <?php?>, como é mostrado na **Atividade 1.**

Atividade 1. Exemplo de uso do **PHP**(index.php).

```
<!DOCTYPE html>
<html>
  <body>
    <h1>Estamos aprendendo PHP!</h1>
    <?php
      echo "Vamos prosseguir aprendendo PHP";
    ?>
  </body>
</html>
```

O arquivo index foi salvo com a extensão .php para mostrarmos ao nosso interpretador que há um código **PHP** a ser interpretado. Além disso, no exemplo usamos a função **echo** para escrever na tela uma mensagem.

Como comentar o código no PHP

Para comentarmos o nosso código **PHP** usamos duas barras ou # para comentários de uma linha, e para comentários de múltiplas linhas usamos /* */, o mesmo usado em CSS. Observe alguns exemplos na **Atividade 2.**

Atividade 2. Exemplo de comentários no **PHP** utilizando //, # ou /**/

```
<?php
  echo "Oi, Eu serei visto na sua tela";
  // Eu não! Sou apenas um comentário.

  echo "Oi, Eu também serei visto por você";
  # Já eu não serei!

  echo "E eu aqui novamente na sua tela, rs";
  /* Eu não aparecerei na sua tela novamente
  pois sou um comentário */
?>
```

Constantes no PHP

O valor de uma constante jamais poderá ser alterado enquanto estiver sendo executada e para defini-la utilizamos a função *define()*, como mostra a **Atividade 3.**

Atividade 3. Exemplo de Constantes.

```
<?php
  define("PHP", "Linguagem Open - Source");
  echo PHP; // Linguagem Open - Source
?>
```

Utilizando a função define() definimos que a constante com o nome de PHP, terá como valor: Linguagem Open – Source.

Variáveis no PHP

Para criarmos uma variável basta utilizar o sinal de cifrão. Uma variável pode armazenar textos e números. Além disso, a linguagem PHP é case sensitive, então A é diferente de a. Observe um exemplo de uso de variáveis na **Atividade 4.**

Atividade 4. Exemplo de Variáveis.

```
<?php
  $name = "Guilherme";
  $age = 20;

  echo $name; // Guilherme
  echo "</br>";
  echo $age; // 20
?>
```

No exemplo criamos uma variável ($name) e declaramos a ela uma string, sendo assim precisamos colocá-la entre aspas. Já a outra variável ($age) é declarada como inteiro, então não é necessário o uso de aspas. Ao usarmos echo nas variáveis, o resultado impresso é o conteúdo dessa variável.

Curso relacionado: **Curso de PHP**

Para a nomeação de variáveis, as dicas a seguir são necessárias:

- Não inicie o nome de uma variável com números;
- Não utilize espaços em brancos;
- Não utilize caracteres especiais, somente underline;
- Crie variáveis com nomes que ajudarão a identificar melhor a mesma;
- Evite utilizar letras maiúsculas.

Falaremos agora sobre alguns dos tipos de variáveis que existem no **PHP**:

- **Booleanos**: Este é o tipo mais simples, pois só pode expressar apenas dois valores: **TRUE (1)** ou **FALSE (0, null ou uma string vazia);**

- **Integer:** é um número inteiro, podendo ser negativo ou positivo;
- **Float :** também chamado de double ou números reais representados com um ponto para separar os dígitos do valor inteiro dos dígitos do valor das casas decimais.
- **Strings:** é uma palavra ou frase entre aspas simples ou duplas, assim como também pode ser binário, como o conteúdo de um **arquivo MP3 ou JPG**. Veja os exemplos na **Atividade 5.**

Atividade 5. Exemplos de String.

```
<?php
  $a = "mundo!";
  echo "Olá, $a"; // Olá, mundo!
  echo 'Olá, $a'; // Olá, $a
?>
```

Note que quando declaramos no echo "Olá, $a, o PHP interpretou o conteúdo da $a, pois está entre aspas duplas. E quando usamos a mesma forma, só que entre aspas simples (echo 'Olá, $a'), não temos o mesmo resultado. Então quando queremos que o PHP interprete o valor de nossa variável dentro de uma string é necessário o uso de aspas duplas. Fique atento!

Além disso, podemos usar um ponto para concatenar strings, assim como o sinal + para o JavaScript, como mostra o código a seguir:

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

Arrays no PHP

Um *array* que mantém uma série de elementos que podem ter diferentes tipos, como mostra a **Atividade 6.**

Saiba mais sobre **Arrays no PHP**

Atividade 6. Exemplo de arrays.

```
<?php
  $php = array("Zend" => "CERTIFICAÇÃO", 6 => false);
  echo $php["Zend"]; // CERTIFICAÇÃO
  echo $php[6]; // 0

  // Zend é nossa chave e CERTIFICAÇÃO nosso valor
  // 6 é nossa chave e false(0) é nosso valor
?>
```

Note que nossa primeira chave se chama **Zend**, e a outra chama-se 6, mas quanto a nomeação de chaves de array pode ser tanto string ou um integer. Para o valor pode ser qualquer coisa.

Conversão de tipos

Os tipos de **variáveis no PHP** são dinâmicos. Para forçarmos os tipos de nossas variáveis utilizamos uma técnica conhecida como type casting, ou simplesmente troca de tipos. Veja na **Atividade 7** alguns exemplos.

Atividade 7. Exemplo de conversão de tipos.

```
<?php
  $var = 100;
  $type_casting = (bool) $var; // torna – se booleano
  $type_casting = (int) $var; // torna – se inteiro
  $type_casting = (float) $var; // torna – se float
  $type_casting = (string) $var; // torna – se string
  $type_casting = (array) $var; // torna – se array
  echo $type_casting = (bool)$var; // 1
?>
```

Veja que transformamos o valor da $var, que antes era um inteiro, para um valor booleano

Vamos conhecer agora **operadores**, que permitem que nós manipulemos o conteúdo de uma ou mais variáveis.

Operadores Aritméticos no PHP

Podemos utilizar operadores matemáticos para efetuar cálculos com os valores de variáveis, como mostra a **Atividade 8**.

Atividade 8. Exemplo de operadores aritméticos.

```
<?php
  $a = 3;
  $b = 3;
  $c = $a * $b; // resultado é 9
  $d = $a + $b; // resultado é 6
  $e = $c - $d; // resultado é 3
?>
```

Criamos as variáveis $a e $b e a partir delas conseguimos fazer vários cálculos matemáticos.

Os operadores matemáticos disponíveis em PHP são:

- Adição: +
- Subtração: -
- Multiplicação: *
- Divisão: /
- Módulo: %

Lembrando que não precisamos especificar os tipos de variáveis no PHP, como nos exemplos a seguir:

```
<?php
  $a = "5"; // string
  echo $a + 2; // 7, integer
  echo $a + '5 carros'; // 10, integer
?>
```

Note que $a é uma string e quando demos um echo nela somando com 2, que é um inteiro, o resultado retornado foi 7. Isso demonstra que nossos tipos de variáveis em PHP são sempre dinâmicos.

Operadores de Atribuição no PHP

Utilizamos os operadores de atribuição para definir variáveis e seus valores, além de usá-los juntamente com os operadores matemáticos, como mostra o exemplo da **Atividade 9**.

Atividade 9. Exemplo de Operadores de Atribuição com Operadores Matemáticos.

```
<?php
  $a = 1; // A variável $a é igual a 1
  $a += 2; // Somamos 2 ao valor da $a;
  echo $a;
?>
```

O resultado acima é 3, pois somamos 2 ao valor da $a, que é 1. A seguir temos mais exemplos:

```
<?php
  $a -= 2; // Subtraímos 2 ao valor da variável $a;
  $a *= 2; // Multiplicamos o valor da variável $a por 2;
  $a /= 2; // Dividimos o valor da variável $a por 2.
?>
```

A sintaxe desses operadores é a mesma do exemplo da soma, pois basta dar um echo depois de ter declarado a variável com seu respectivo operador.

Podemos também incrementar ou decrementar variáveis utilizando os operadores de incrementação, herdados da linguagem C, como nos exemplos a seguir:

```
<?php
```

```
  $a = 1;
  echo ++$a; // Incrementamos 1 e retornamos o valor
  echo $a++; // Retornamos o valor e incrementamos 1
  echo --$a; // Decrementamos 1 e retornamos o valor
  echo $a--; // Retornamos o valor e decrementamos 1
?>
```

Operadores Relacionais

Esses são usados para comparar valores ou expressões, retornando um valor booleano (true ou false):

- Igual: ==
- Idêntico: ===
- Diferente: != ou <>
- Menor que: <
- Maior que: >
- Menor ou igual: <=
- Maior ou igual: >=

É importante lembrar que == não checa o tipo da variável, apenas seu valor. Já o === checa tanto o valor da variável quanto o seu tipo.

Operadores Lógicos

Existem também os operadores lógicos para a criação de testes condicionais:

- $a and $b: enquanto A e B forem verdadeiros;
- $a or $b: enquanto A ou B forem verdadeiros;
- $a xor $b: enquanto A ou B forem verdadeiros, mas não os dois;
- !$a: verdadeiro se A for falso;
- $a && $b: enquanto A e B forem verdadeiros;
- $a || $b: enquanto A ou B forem verdadeiros.

Estrutura de Decisão if/else

A condição é avaliada para que, caso algo seja verdadeiro, faça isto, senão, faça aquilo, como mostra a **Atividade 10.**

Atividade 10. Uso de **if/else.**

```
<?php
  $idade = 17;

  if($idade < 18) {
    echo 'Você não pode entrar aqui!';
  } else {
    echo 'Seja bem - vindo';
  }
?>
```

Criamos a variável $idade que guarda um inteiro. Em seguida utilizamos *IF* para verificar se $idade é menor que 18, e caso seja será impresso: Você não pode entra aqui! Depois criamos um *ELSE,* que é o contrário da primeira condição.

Podemos também utilizar os operadores lógicos junto dos operadores relacionais, como mostra a **Atividade 11**.

Atividade 11. Exemplo do uso de **Operadores lógicos com Operadores relacionais**.

```
<?php
$idade = 21;
$identidade = true;

if($idade > 18 && $identidade == true) {
  echo 'Seja bem-vindo!';
}
?>
```

Estruturas de Decisão (elseif/switch)

Podemos utilizar a estrutura ELSEIF quando criamos uma outra condição, além da principal, como mostra o exemplo da **Atividade 12**.

Atividade 12. Exemplo **ELSEIF.**

```
<?php
$nome = 'Till Lindemann';

if($nome == 'Richard Kruspe') {
  echo 'E ae Richard Kruspe!';
```

```
} elseif ($nome == 'Oliver Riedel') {
  echo 'E ae Oliver Riedel!';
} elseif ($nome == 'Till Lindemann') {
  echo 'E ae Till Lindemann!';
} else {
  echo "E ae $nome!";
}
?>
```

Declaramos para $nome uma string e depois fizemos várias condições, onde caso não caia em nenhum elseif, será retornado o valor que estiver dentro da $nome.

O **ELSEIF** pode ser muito útil, mas o mesmo é aconselhável usar apenas quando temos poucas condições. Caso contrário, para não manter um código cheio de ELSEIF's, o mais indicado é usar o **SWITCH**, que permite criarmos infinitas condições de forma organizada. Veja na **Atividade 13.**

Atividade 13. Exemplo **SWITCH**.

```
<?php
  $nome = 'Fulano';

  switch($nome) {
    case 'Fulano':
      echo 'E ai Fulano!';
      break;

    case 'Sicrano':
      echo 'E ai Sicrano!';
      break;

    case 'Beltrano':
      echo 'E ai Beltrano!';
      break;

    default:
      echo 'Qual é o seu nome?';
      break;
  }

  // Resultado é: E ai Fulano!
?>
```

No exemplo foi criada $nome declarando a ela uma string. É verificado a variável passada em switch entre parênteses: caso o valor contido na variável seja o que estiver em "case", será impresso o que conter no echo já pré-definido, e o break para a verificação. Caso não seja na primeira condição, será verificado todas as demais condições até chegar na última, que perguntará o seu nome. Na sintaxe básica do Switch atente-se sempre ao “:” no case e os “;” depois do echo e break.

Operador Ternário no PHP

No PHP existe uma forma mais curta de criar condições através do **Operador Ternário,** como mostra a **Atividade 14**.

Atividade 14. Utilizando **Operador Ternário.**

```
<?php
  $number1 = 1;
  $number2 = 2;

  if($number2 > $number1) {
    $a = 'Número 2 é maior que número 1';
  } else {
    $b = 'Número 2 não é maior que número 1';
  }

  $ternario = ($number2 > $number1) ? 'Número 2 é maior que número 1' : 'Número 2 não é maior que número 1';

  echo $ternario; // Número 2 é maior que número 1
?>
```

Criamos duas variáveis e a partir delas fizemos verificação com if/else e da forma ternária também. Na $ternario passamos a condição que tem o mesmo valor que o if entre parênteses e o ponto de interrogação faz a pergunta: $number2 é maior $number1?

Caso seja, será impresso o que vier depois desse ponto. Caso $number2 não seja maior que $number1, será impresso o que estiver depois dos ":" que tem o mesmo valor que else. Sendo assim, podemos comparar qual forma é mais simples com esses dois exemplos.

Arrays Associativos

Quando criamos um array, por padrão, ele recebe chaves numéricas incrementadas automaticamente de acordo com novos valores. Contudo, podemos criar chaves que são strings, daí chamamos de array associativo. Para explicar melhor, faremos um exemplo em que mostraremos a temperatura média de alguns meses do ano, como mostra a **Atividade 15.**

Atividade 15. Arrays Associativos.

```
<?php
  $estacao = array('Verao' => 'de 21 de dezembro a 21 de março', 'Outono' => 'de 21 de março a 21 de junho',
    'Inverno' => 'de 21 de junho a 23 de setembro', 'Primavera' => 'de 23 de setembro a 21 de dezembro');
?>
```

Observem que nossos valores agora possuem nomes. Podemos utilizar estes nomes para nos referenciarmos a um valor especifico dentro de um array. Veja como fica na prática o exemplo:

```
<?php
  $estacao = array('Verao' => 'de 21 de dezembro a 21 de março', 'Outono' => 'de 21 de março a 21 de junho',
    'Inverno' => 'de 21 de junho a 23 de setembro', 'Primavera' => 'de 23 de setembro a 21 de dezembro');
  echo "A estação Verão foi: {$estacao['Verao']}";

  // A estação Verão foi: de 21 de dezembro a 21 de março
?>
```

Como sabemos, a sintaxe de um array é chave => valor. Note que demos um echo assim: {$estacao['**Verao**']}, onde $estacao contém um array que tem uma chave chamada Verao e contém o valor: de 21 de dezembro a 21 de março. Sendo assim, será impresso o valor da chave especificada.

Existem maneiras diferentes de utilizar variáveis dentro de strings e vice-versa, concatenadas ou não concatenadas, como mostra a **Atividade 16.**

Atividade 16. Variáveis dentro de strings.

```
<?php
$ensino = 'EAD';
$formacao = array('PHP' => 'Desenvolvedor PHP', 'Infra' => 'SysAdmin Linux');

// Não concatenadas
echo "<p>No $ensino da Versátil você se torna {$formacao['PHP']}";
echo " e pode se tornar também {$formacao['Infra']}.</p>";

// Concatenadas
echo '<p>No ' . $ensino . ' da Versátil você se torna ' . $formacao['PHP'];
echo ' e pode se tornar também '. $formacao['Infra'] . '.</p>';
?>
```

No exemplo o resultado será o mesmo, apenas mostramos a utilização de uma variável e um array com concatenação ou não concatenado para mostrar seus valores. Tente usar sempre a forma não concatenada: você pode notar que nosso código ficou bem mais limpo.

Arrays Multidimensionais

Arrays multidimensionais são, basicamente, array dentro de um array, como mostra a **Atividade 17.**

Atividade 17. Arrays Multidimensionais.

```
<?php
  $temp = array(
    '2010' => array(
    'Outubro' => 25,
    'Novembro' => 23,
    'Dezembro' => 20),
    '2011' => array(
    'Outubro' => 26,
    'Novembro' => 22,
    'Dezembro' => 21),
    '2012' => array(
    'Outubro' => 27,
    'Novembro' => 28,
    'Dezembro' => 19)
  );

  echo "Dezembro de 2012 foi de: {$temp['2012']['Dezembro']} graus";
// Dezembro de 2012 foi de: 19 graus
?>
```

Veja que criamos a $temp que guarda um array. Em nosso exemplo, o array com o nome de 2010 guarda os valores Outubro, Novembro e Dezembro em um outro array, que por sua vez, guarda outros valores, que são a temperatura. Veja que demos um echo em {**$temp**['**2012**']['**Dezembro**']} onde $temp guarda um array 2012 e inicia um outro array que contém a chave dezembro e que tem o valor que queremos.

Criando Arrays de uma forma alternativa

Podemos criar arrays de uma forma simples e rápida utilizando o operador [ ], como mostra a **Atividade 18.**

Atividade 18. Criando **Arrays** de forma alternativa.

```
<?php
  $number1 = array(100, 101, 102);
  $number1[] = 103;
  $number2[] = 104;

  print_r($number1);
  echo '<hr/>';
  print_r($number2);
?>
```

Através dos colchetes conseguimos acrescentar dados a um array. No caso da $number1 temos um array com os valores 100, 101, 102. Quando declaramos $number1[] = 103; estamos dizendo que queremos acrescentar no array $number1 o valor 103.

Podemos também alterar os valores de um array usando colchetes, como no código a seguir:

```
<?php
  $cert = array('EAD' => 'Você terá um certificado ', 'PHP' => 'Linux');

  $cert['PHP'] = 'Zend';
  print_r($cert);
?>
```

Temos a $cert que contém um array com chave => valor. Quando dizemos $cert['PHP'] = 'Zend', estamos dizendo para o PHP que queremos alterar o valor da chave PHP para Zend.

Laços de Repetição

Os loops no PHP são estruturas de controle muitos importantes que permitem efetuar um laço de repetição enquanto uma determinada condição for verdadeira.

Existem quatro tipos de loop no **PHP** são eles: **while**, **do while**, **for** e **foreach**.

While

O while permite que executemos um bloco de código enquanto a expressão passada como parâmetro for verdadeira, como mostra a **Atividade 19.**

Atividade 19. Exemplo **while.**

```
<?php
  $num = 0;

  while($num < 10 ) {
    echo $num++;
  }

  // 0123456789
?>
```

Declaramos a $num o valor 0. Nossa expressão no while então incrementa a essa variável enquanto $num for menor que 10.

Do while

O **do while** tem a mesma ideia que o while: a diferença é que ele avalia a expressão depois de executar algo. Com isso, será garantido que o código será executado mesmo que a expressão seja falsa, como mostra o exemplo da **Atividade 20.**

Atividade 20. Exemplo **Do while.**

```
<?php
  $cont = 2000;

  do{
    $dobro = $cont + $cont;
    echo "O dobro de $cont é $dobro";
    $cont++;
  } while ($cont <= 1999);
?>
```

A sintaxe básica do **DO WHILE** é primeiro executar o que colocamos entre as chaves do **DO**.

FOR

O for é igual ao **while** e ao **do while,** permitindo que executemos três operações em sua condição, separadas por ponto e vírgula.

A primeira é executada ao início do loop, a segunda é a condição (enquanto ela for verdadeira, o loop continuará), e a terceira é executada ao fim de cada repetição, como mostra o exemplo da **Atividade 21.**

Atividade 21. Exemplo **For**.

```
<?php
for($a = 1; $a <= 10; $a++){
  $cubo = $a * $a * $a;
  echo "O cubo de $a é $cubo<br />";
}
?>
```

Entre parênteses temos $a que guarda 1. Depois verificamos se $a for menor ou igual a 10 e, por último, incrementamos $a. Dentro das chaves do **FOR** criamos uma variável que traz o cubo de $a para nós e quando demos um echo será mostrado o cubo dos números de 1 a 10.

Foreach

O Foreach faz o mesmo que as demais estruturas já apresentadas, porém, com ela podemos trabalhar com arrays, como mostra o exemplo da **Atividade 22.**

Atividade 22. Usando **FOREACH**.

```
<?php
  $ead = array('Aqui na Versátil ', 'você se torna um ', 'desenvolvedor PHP');

  foreach($ead as $can){
    echo "$can";
  }

  //Aqui na Versátil você se torna um desenvolvedor PHP
?>
```

No exemplo é criado um array e depois usamos o foreach para ir nesse array e repetir tudo o que conter nele. A sintaxe do foreach é mostrada entre parênteses onde colocamos o nome de nossa variável, e com o termo *as*alteramos o nome dela para $can. Depois, basta dar um echo que tudo que conter em nosso array será mostrado.

A seguir utilizamos o loop em um array que contém chave =>valor.

```
<?php
$temp = array('Outubro' => 27, 'Novembro' => 28, 'Dezembro' => 19);

foreach($temp as $chave => $valor) {
  echo "A temperatura média de $chave foi de $valor graus<br />";
}
?>
```

Quando formos utilizar o foreach em um array que contém chave e valor é necessário a sintaxe que vemos entre parênteses;

Assim como temos os arrays multidimensionais, assim também temos foreachs multidimensionais, que são foreachs dentro de foreachs, como mostra o exemplo da **Atividade 23.**

Atividade 23. Uso do **foreach** com **array multidimensional.**

```
<?php
$temp = array(
  '2010' => array(
  'Outubro' => 25,
  'Novembro' => 23,
  'Dezembro' => 20),
  '2011' => array(
  'Outubro' => 26,
  'Novembro' => 22,
```

```
  'Dezembro' => 21),
  '2012' => array(
  'Outubro' => 27,
  'Novembro' => 28,
  'Dezembro' => 19)
);

foreach($temp as $ano => $meses){
  echo "Temperaturas em $ano<br />";

  foreach($meses as $mes => $temp) {
    echo "$mes: $temp graus<br />";
  }
}
?>
```

Veja que criamos um array e dentro dele vários outros arrays que guardam temperaturas de meses de um determinado ano. Quando utilizamos o foreach iremos mostrar o ano da temperatura e quantos graus no mês especificado.

Quebrando loops

Enquanto estamos dentro de um loop, podemos utilizar duas instruções: *continue* e *break*. Elas permitem que nós quebremos os laços de repetição.

Veja na **Atividade 24** um exemplo da Continue.

Atividade 24. Exemplo *Continue.*

```
<?php
for($a = 1; $a <= 10; $a++){
  if($a == 3) {
    continue;
  }

  $cubo = $a * $a * $a;
  echo "O cubo de $a é $cubo<br />";
}

//O cubo de 1 é 1
//O cubo de 2 é 8
//O cubo de 4 é 64
//O cubo de 5 é 125
//O cubo de 6 é 216
//O cubo de 7 é 343
//O cubo de 8 é 512
//O cubo de 9 é 729
//O cubo de 10 é 1000
?>
```

Note que não aparece o cubo de 3, pois fizemos a seguinte verificação: if($a == 3) {continue;}.

Na **Atividade 25** vemos um exemplo da instrução Break.

Atividade 25. Exemplo Break.

```
<?php
  for($a = 1; $a <= 10; $a++){
    if($a == 3) {
      break;
    }

    $cubo = $a * $a * $a;
    echo "O cubo de $a é $cubo<br />";
  }
  //O cubo de 1 é 1
  //O cubo de 2 é 8
  ?>
```

O break para o loop, então veja que só é mostrado o cubo de 1 e 2. Isso porque colocamos a verificação if($a == 3) {break;}, ou seja, chegou no 2 e deu um break.