PHP – PARTE 2

DIOGO CEZAR TEIXEIRA BATISTA
http://inf.cp.utfpr.edu.br/diogo
diogo@diogocezar.com

TIPOS DE DADOS

- o PHP suporta vários tipos de dados:
  - *inteiro* – Números inteiros (isto é, números sem ponto decimal);
  - números de *dupla precisão* – Números reais (isto é, números que contêm um ponto decimal);
  - *string* – Texto entre aspas simples (‘’) ou duplas (“”);
  - *booleanos* – armazenam valores verdadeiros ou falsos, usados em testes de condições.

TIPOS DE DADOS

  - *Array* – Grupo de elementos do mesmo tipo;
  - *Objeto* – Grupo de atributos e métodos;
  - *Recurso* – Uma origem de dados externa;
  - *Nulo* – Nenhum valor.
- o tipo da variável geralmente *não é configurado pelo programador*: isso é decidido em tempo de execução pelo PHP, dependendo do contexto no qual a variável é usada.

EXEMPLO

```
<?php
$a = true;
if ($a){
   echo "Verdadeiro";
}
else {
   echo "Falso";
}
?>
```

Teste o código anterior alterando o valor da variável para False.

TIPOS DE DADOS

- pode-se armazenar valores inteiros, positivos ou negativos. Pode-se usar também valores hexadecimais;

```
<?php
$a = 0x1A; //Coresponde ao decimal 26
$b = -16;
$c = $a + $b;
echo "a + b = $c";
?>
```

TIPOS DE DADOS

- os valores de ponto flutuante são representados através de ponto (.);

```
<?php
$preco = 11.90;
$soma = $preco * 4;
echo "Quatro revistas W custam R$ $soma<br>";
?>
```

TIPOS DE DADOS

- as variáveis do tipo matriz ou Array permitem o armazenamento de diversos elementos:

```
<?php
$frutas = array(1 => "Laranja",
                     2 => "Maçã",
                     3 => "Uva");
echo "<li> $frutas[1]<br>";
echo "<li> $frutas[2]<br>";
echo "<li> $frutas[3]<br>";
?>
```

TIPOS DE DADOS

- para verificar o tipo de dados de uma determinada variável pode-se utilizar as funções: is_{type};

```
<?php
  $i = 5;
  if(is_int($i)){
     echo "É número";
  }
  if(is_string($i)){
     echo "É string";
  }
?>
```

TIPOS DE DADOS

- Para obter uma representação legível do seu tipo de dados utilize a função: gettype()

```
<?php
  $bool = true;
  $str = “foo”;
  echo gettype($bool) // imprime
“boolean”
  echo gettype($str) // imprime
“string”
?>
```

TIPO DE DADOS

- Se você quiser *FORÇAR* a conversão de uma variável para um certo tipo, você pode moldar (casting) a variável ou usar a função settype() nela:

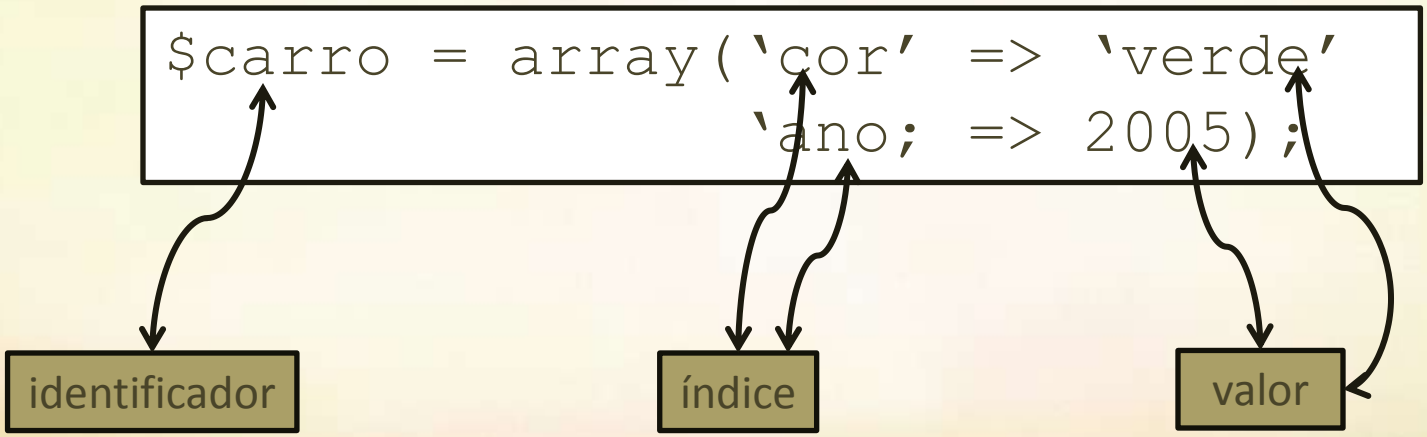
```
<?php
  $foo = "5bar"; // string
  $bar = true;   // boolean

settype($foo, "integer"); // $foo é agora 5   (in
teger)

settype($bar, "string");  // $bar é agora "1" (st
ring)
?>
```

ARRAYS

- um array (vetor) pode armazenar vários valores ao mesmo tempo;
- possui um identificador, e além disso um índice associado (que pode ser número ou texto) e um valor para cada índice.



ARRAYS

- para indicar a indexação do array utilizamos colchetes []:

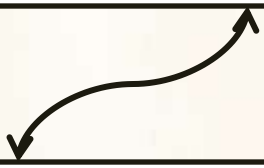
```
<?php
  $arr = array("foo" => "bar",
                        12    =>  true);
  echo $arr["foo"]; // bar
  echo $arr[12]; // 1
?>
```

- os índices podem arrumar valores inteiros ou strings, enquanto que os valores podem assumir qualquer valor.

ARRAYS

- ao omitir o índice quando informar um novo valor, o PHP entenderá que o novo índice é o valor antigo + 1:

```
array(5 => 43, 32, 56, "b" => 12);
```



```
array(5 => 43, 6=> 32, 7 => 56, "b" => 12);
```

- se você especificar uma chave que já possui um valor assumido, então ele será sobrescrito.

ARRAYS

- utilizar true ou false no índice de um array será interpretado como 1 ou 0;
- você pode também modificar um array existente explicitamente assimilando valores nele.

```
$arr = array(5 => 1, 12 => 2);
$arr[] = 56; // isso é o mesmo que $arr[13] = 56;
$arr[“x”] = 42; // isso acrescenta um novo elemento
unset($arr[5]); // isso remove um elemento do array
unset($arr); // isso apaga todo array
```

CONSTANTES

- constantes são identificadores para valores simples;
- o seu conteúdo não muda durante a execução do código;
- elas são criadas com a função define e, por convenção, são escritas com letras MAIÚSCULAS e não usam o cifrão no início.

```
<?php
define("CONSTANTE", "Alô mundo.");
echo CONSTANTE;
?>
```

CONSTANTES

- o PHP implementa algumas constantes, a maioria são matemáticas. O código seguinte demonstra o uso da constante M_PI.

```
<?php
function calculaAreaCirculo($raio){
   return M_PI * pow($raio, 2);
}
$meuRaio = 5;
$area = calculaAreaCirculo($meuRaio);
echo "<b>Raio</b> = $meuRaio<br>";
echo "<b>Área</b> = $area";
?>
```

EXPRESSÕES

- tudo que tem um valor pode ser considerado uma expressão:

```
<?php
$b = ($a = 5);
echo "O valor de 'b' é $b";
?>
```

EXPRESSÕES

- expressões de comparação retornam valores booleanos, sendo vazio (‘’) representando *falso* e um (1) representando *verdadeiro*;
- as expressões de comparação são usadas em declarações condicionais (if) para determinar se um bloco de código será executado ou não.

```
<?php
$valor = (5 < 10);
echo "O valor da expressão '5 > 10' é
$valor";
14
?>
```

OPERADORES ARITMÉTICOS

- lembra-se da aritmética básica da escola? Estes operadores funcionam exatamente como aqueles;

Exemplo	Nome	Resultado
-$a	Negação	Oposto de $a.
$a + $b	Adição	Soma de $a e $b.
$a - $b	Subtração	Diferença entre $a e $b.
$a * $b	Multiplicação	Produto de $a e $b.
$a / $b	Divisão	Quociente de $a por $b.
$a % $b	Módulo	Resto de $a dividido por $b.

OPERADORES ARITMÉTICOS

```php
<?php
  $x = 2;
  echo($x + 2);
  echo(5 - $x);
  echo($x * 5);
  echo($x / 5);
  echo($x % 8);
  $x++;
  $x--;
  $x += 10;
  echo($x);
?>
```

OPERADORES DE COMPARAÇÃO

- uma comparação sempre gera um dos dois valores possíveis: *verdadeiro* ou *falso*;
- a próxima tabela mostra os operadores de comparação.

OPERADORES DE COMPARAÇÃO

Exemplo	Nome	Resultado
$a == $b	Igual	Verdadeiro (TRUE) se $a é igual a $b.
$a === $b	Idêntico	Verdadeiro (TRUE) se $a é igual a $b, e eles são do mesmo tipo (introduzido no PHP4).
$a != $b	Diferente	Verdadeiro se $a não é igual a $b.
$a <> $b	Diferente	Verdadeiro se $a não é igual a $b.
$a !== $b	Não idêntico	Verdadeiro de $a não é igual a $b, ou eles não são do mesmo tipo (introduzido no PHP4).
$a < $b	Menor que	Verdadeiro se $a é estritamente menor que $b.
$a > $b	Maior que	Verdadeiro se $a é estritamente maior que $b.
$a <= $b	Menor ou igual	Verdadeiro se $a é menor ou igual a $b.
$a >= $b	Maior ou igual	Verdadeiro se $a é maior ou igual a $b.

OPERADORES DE COMPARAÇÃO

```
<?php
   $x = 5;
   $resultado = ($x == 8);
   if($resultado == 1){
      echo "verdadeiro";
   }
   else {
      echo "falso";
   }
?>
```

OPERADORES LÓGICOS

- existem ainda os operadores lógicos:
  - utilizados para testar se um conjunto de expressões é verdadeiro ou não.

OPERADORES LÓGICOS

Exemplo	Nome	Resultado
$a and $b	E	Verdadeiro (TRUE) se tanto $a quanto $b são verdadeiros.
$a or $b	OU	Verdadeiro se $a ou $b são verdadeiros.
$a xor $b	XOR	Verdadeiro se $a ou $b são verdadeiros, mas não ambos.
! $a	NÃO	Verdadeiro se $a não é verdadeiro.
$a && $b	E	Verdadeiro se tanto $a quanto $b são verdadeiros.
$a || $b	OU	Verdadeiro se $a ou $b são verdadeiros.

OPERADORES LÓGICOS

```php
<?php
  $x = 6;
  $y = 3;
  $resultado = ($x < 10 && $y > 1);
  $resultado = ($x == 5 || $y == 5);
  $resultado = (!($x == $y));
  echo $resultado;
?>
```

OPERADORES DE INCREMENTE/DECREMENTO

- o PHP suporta operadores de pré e pós-incremento e decremento no estilo C;

Exemplo	Nome	Efeito
++$a	Pré-incremento	Incrementa $a em um, e então retorna $a.
$a++	Pós-incremento	Retorna $a, e então incrementa $a em um.
--$a	Pré-decremento	Decrementa $a em um, e então retorna $a.
$a--	Pós-decremento	Retorna $a, e então decrementa $a em um.

Operador TERNÁRIO

- é uma forma abreviada de usar o comando condicional *if*;

```
cond ? exp1 : exp2
```

- se a condição for verdadeira, o valor retornado é o da exp1, caso contrário o valor retornado é o da exp2;

```
$nota = ($frequencia >= 0.75) ? ($nota++) : ($nota--);
```

PRECEDÊNCIA DE OPERADORES

- a precedência de operadores especifica quem tem maior prioridade quando duas delas estão juntas, por exemplo:

```
1 + 5 * 3
```

- a multiplicação tem prioridade sobre a adição, então a resposta é *16* e não 18.

PRECEDÊNCIA DE OPERADORES

Associação	Operador	Informação adicional
não associativo	clone new	clone e new
esquerda	[	array()
não associativo	++ --	incremento/decremento
não associativo	~ - (int) (float) (string) (array) (object) (bool) @	tipos
não associativo	instanceof	tipos
direita	!	lógico
esquerda	* / %	aritmético
esquerda	+ - .	aritmético e string
esquerda	<< >>	Bit-a-bit
não associativo	< <= > >= <>	comparação
não associativo	== != === !==	comparação
esquerda	&	Bit-a-bit e referências
esquerda	^	Bit-a-bit
esquerda	|	Bit-a-bit
esquerda	&&	lógico
esquerda	||	lógico
esquerda	? :	ternário
direita	= += -= *= /= .= %= &= |= ^= <<= >>=	atribuição
esquerda	and	lógico
esquerda	xor	lógico
esquerda	or	lógico
esquerda	,	muitos usos

PRECEDÊNCIA DE OPERADORES

- associatividade a esquerda significa que a expressão é avaliada da esquerda para direita, associatividade a direita o oposto.

FUNÇÕES ÚTEIS

- Arrays:
  - sort($arr) - Ordena um array;
  - count($arr) - Conta o número de elementos e retorna um inteiro;
  - print_r($arr) - Imprime um array (índices e valores);
  - unset($arr[$i]) – Remove um elemento do array;
  - unset($arr) – Limpa todo o array;
  - shuffle($arr) - Mistura os elementos de um array;
  - in_array("valor", $arr) - Checa se um valor existe em um array.

FUNÇÕES ÚTEIS

- String:
  - substr($str, $start, $length) - Retorna a parte de *string* especificada pelo parâmetro *start* e *length*;
  - strrpos($procurado, $str) - Encontra a ultima ocorrência de um caractere em uma string;
  - strpos($procurado, $str) - Encontra a posição da primeira ocorrência de uma string;
  - trim($str) - Retira espaço no ínicio e final de uma string;
  - str_replace($str, $procurado, $alterar) - Substitui todas as ocorrências da string de $procurado com a string de $alterar
  - str_len($str) – Conta o número de caracteres de $str;
  - strtolower($str) – Transforma todos caracteres em minúsculos;
  - strtoupper($str) – Transforma todos caracteres em maiúsculos.

FUNÇÕES ÚTEIS

- empty($valor) – verifica se determinado valor está vazio;
- explode(“, $string) – recorta a string e a transforma em um array, em todo lugar que o caracter “ for encontrado;
- Implode(“, $array) - Retorna uma string contendo os elementos da matriz na mesma ordem com uma ligação entre cada elemento;
- number_format($valor, $numCasasDecimais, $ponto, $virgula) - Formata um número com os milhares agrupados;
- rand($inicio, $fim) - Gera um inteiro aleatório;
- date(‘d/m/Y - H:i:s’) – Comando para exibição de data.

MANIPULAÇÃO DE DATAS

- para exibir uma data, podemos utilizar o comando date('parametros', [timestamp]);
- o segundo parâmetro é opcional, e representa a data que será exibida. Caso fique em branco, a data atual é escolhida;
- os parâmetros retornam, entre outros:
  - d → dia;
  - m → mês;
  - Y → ano;
  - H → hora;
  - i → minuto;
  - S → segundo.

MANIPULAÇÃO DE DATAS

- o php oferece a comparação nativa de datas no formato timestamp;
- para pegar um timestamp atual utiliza-se a função time();
- para pegar um timestamp específico utiliza-se a função mktime($hora, $minuto, $second $mes, $dia, $ano).

```
$tempo1 = time();
$tempo2 = mktime(20,18,0,2,3,2005);
echo date('d/m/Y', $tempo1); // dia, mês e hora
atual
echo date('d/m/Y', $tempo2); // 03/02/2005
```

ESTRUTURAS DE CONTROLE

- seu uso é fundamental para realizar decisões lógicas;
- veremos os seguintes comandos:
  - comandos condicionais: if e switch;
  - comandos de repetição: while, do...while, for e foreach.

ESTRUTURAS DE CONTROLE

- *if*
  - permite a execução condicional de fragmentos de código;

```
If(expressao){
   instrucoes
}
```

  - o bloco será executado somente se a expressão tiver um retorno TRUE;
  - comandos if podem ser aninhados indefinidamente dentro de outros comandos *if*.

ESTRUTURAS DE CONTROLE

- *else*
  - estende um comando if para executar uma instrução caso a expressão no comando if seja avaliada como FALSE;

```
If(expressao){
    instrucoes
}
else{
    instrucoes
}
```

ESTRUTURAS DE CONTROLE

- *elseif*
  - como o nome sugere, é uma combinação de if e else;
  - estende um comando *if*, para executar uma instrução diferente, caso o *if* original seja avaliado como FALSE.

```
If(expressao1){
   instrucoes
}
elseif(expressao 2){
   instrucoes
}
else{
   instrucoes
}
```

ESTRUTURAS DE CONTROLE

- *switch*
  - similar a uma série de instruções if's seguidas.

```
switch(valor){
   case 0: instrucoes; break;
   case 'carro': instrucoes;
break;
   ...
   default: instrucoes;
}
```

  - a opção default será acionada quando nenhum valor corresponder ao testado;
  - os cases podem testar o valor com uma string;
  - sem o comando break, o próximo case também é executado.

ESTRUTURAS DE CONTROLE

- *while*
  - utilizado para criar um 'loop';

```
while(expressao){
    instrucoes
}
```

  - executa as instruções enquanto a expressão for verdadeira (TRUE).

ESTRUTURAS DE CONTROLE

- *dowhile*
  - similar ao while, exceto pelo fato de que a condição é verificada no final de cada iteração, assim sendo, o bloco será executado pelo menos 1 vez.

```
do{
    instrucoes
} while(expressao)
```

ESTRUTURAS DE CONTROLE

- *for*
  - usado quando queremos executar um conjunto de instruções, por um número determinado de vezes;

```
for(inicializacao; expressao;
operador){
   instrucoes
}
```

  - inicialização: usado para atribuir o valor inicial das variáveis do bloco for;
  - expressão: condição de parada do laço;
  - operador: incremento das variáveis do bloco for.

- *foreach*
  - usado para navegar entre os elementos de um array, apresenta 2 sintaxes possíveis:

```
foreach($nome_array as $elemento){
   instrucoes
}
```

ou

```
foreach($nome_array as $chave =>
$valor){
   instrucoes
}
```

ESTRUTURAS DE CONTROLE

- a primeira forma vai do primeiro ao último índice do array, e a cada iteração o valor do elemento corrente é atribuído à variável $elemento;
- a segunda forma faz a mesma coisa, mas disponibiliza ainda o índice do array em $chave.

ESTRUTURAS DE CONTROLE

```
<?php
  $vetor = array(1, 2, 3, 4);
  foreach($vetor as $e){
    echo $e;
  }
  $cores = array("um" => "azul",
                          "dois" =>
  "verde",
                          "tres" =>
  "preto");
  foreach($cores as $indice => $valor){
     echo "A cor ".$indice." é ".$valor;
  }
?>
```

ESTRUTURAS DE CONTROLE

- *break*
  - cancela a execução do comando for, while, do..while ou switch atual;
  - aceita um argumento numérico opcional que diz a ele quantas estruturas aninhadas devem ser quebradas.

```
while(++$i){
   switch($i){
      case 5: break 1; /* Sai somente
do switch */
      case 10: break 2; /* Sai do
switch e do while */
   }
}
```

Estruturas de controle

- *continue*
  - é usado dentro de estruturas de loops para saltar o resto da iteração do loop atual e continuar a execução no início da próxima iteração;

```
while($i < 10){
    if(!($i % 2)){ // pula itens pares
        continue;
    }
}
```

ATIVIDADES

- faça um script que saúde o usuário de acordo com o horário em que a página foi acessada, utilize a função $hora = date('H'); para capturar a hora, e informe:
  - boa madrugada! Vai dormir. → hora entre 00:00 e 06:00;
  - bom dia! Acordou cedo, hein? → hora entre 06:00 e 12:00;
  - boa tarde! Você já almoçou? → entre 12:00 e 19:00;
  - boa noite! Você já jantou? → entre 19:00 e 00:00.

ATIVIDADES

- faça um programa que sorteie os 6 números da loteria;
  - Os números sorteados devem estar entre 1 e 80.

ATIVIDADES

- faça um script que sorteie 2 números de 1 a 28;
  - imprima a soma entre os números;
  - imprima a multiplicação entre os números;
  - imprima a divisão entre os números;
  - imprima a subtração entre os números;
  - imprima o resto da divisão entre os números.

ATIVIDADES

- um conteúdo em html poderá ser exibido somente até o dia 25/03/2009 as 15:00;
  - faça um script que verifique se a data e hora atual é menor que a data e hora limite, em caso positivo exiba o conteúdo.