

# DESENVOLVIMENTO DE APLICAÇÕES WEB

## Scripts PHP - Introdução

Os scripts PHP serão utilizados para realizar tarefas necessárias para atender as solicitações dos usuários. Elas são repassadas pelo HTTP server ao script PHP que realiza ações como, por exemplo, capturar os dados digitados pelo usuário, interagir com o banco de dados, realizar cálculos e enviar os resultados para serem vistos através de um navegador. Vamos aos elementos iniciais.

### 1. Generalidades

Os scripts PHP iniciam e terminam da seguinte forma:

```
<?php
.....
?>
```

ou

```
<script language="php">
.....
</script>
```

Os dados manipulados pelos scripts podem ter diversas origens. Por exemplo:

- a) Digitados, escolhidos ou assinalados nos formulários;
- b) Recebidos de consulta a bancos de dados;
- c) Variáveis definidas pelo programador;
- d) Lidos de arquivos;
- e) Obtidos de variáveis de ambientes, através de funções específicas;
- f) Disponíveis em variáveis super-globais.

Aos poucos, veremos algumas dessas formas, no entanto, é bom saber que um mesmo valor, às vezes, pode ser obtido de várias maneiras. Por exemplo, os valores originados dos formulários podem ser obtidos através das seguintes variáveis super-globais, supondo que foi utilizado o método “post”:

- a) \$\_POST
- b) \$\_REQUEST

As variáveis super-globais são aquelas que ficam disponíveis aos scripts, independentemente do escopo, sem necessidade de declaração, atribuição de valores ou passagem de parâmetros. Ou seja, o mecanismo ativador do script torna-as disponíveis. Exemplos:

```
$GLOBALS    // sobre as variáveis globais e conteúdos
$_SERVER    // sobre o HTTP Server e o par (cliente/server)
$_GET       // Dados sobre os campos de entrada do método get
$_POST      // Dados sobre os campos de entrada do método post
$_FILES     // Dados dos arquivos para upload para entrada do tipo file
$_COOKIE    // Dados sobre os cookies
$_SESSION   // Dados sobre as variáveis de sessão
$_REQUEST   // Dados sobre os campos de entrada
$_ENV       //LD_LIBRARY_PATH, PATH, RUNLEVEL, LANG, etc.
```

Estas variáveis armazenam dados sobre o ambiente geral e a comunicação. Os dados podem também ser obtidos também pelo uso de funções específicas. Veja como obter os dados sobre porta e hostname/IP, através dessas funções. Exemplo:

```
$cli_ip = getenv("REMOTE_ADDR");
$cli_prt= getenv("REMOTE_PORT");
$srv_ip = getenv("SERVER_ADDR");
$srv_prt= getenv("SERVER_PORT");
```

## 2. Alternância entre códigos HTML e PHP.

Os scripts PHP podem ser alternados com o código em HTML que será enviado sem nenhuma ação por parte do PHP.

```
<?php
$cli_ip = getenv("REMOTE_ADDR");
?>

<HTML>
<HEAD>
<TITLE> Alternância entre PHP e HTML. </TITLE>

<BODY>

<?php
echo "$cli_ip";
?>

</BODY>
</HTML>
```

No exemplo acima, observa-se que os códigos HTML e PHP se intercalam no texto. Às vezes, a alternância produz códigos pouco legíveis. Veja o exemplo abaixo, extraído de um manual do PHP, como um código “confuso” no qual as partes em PHP estão na cor vermelha.

```

<?php
$i = 0;
if ($i == 0 ) {
    ?>
    <strong>Isso é verdadeiro.</strong>
    <?php
}
else {
    ?>
    <strong>Isto é falso.</strong>
    <?php
}
?>

```

### 3. O mesmo código sem alternância

O mesmo texto de código acima poderia aparecer da seguinte forma:

```

<?php
$i = 0;
if ($i == 0 ) {
    echo "<strong>Isso é verdadeiro.</strong>";
}
else {
    echo "<strong>Isto é falso.</strong>";
}

?>

```

Desta forma, não temos uma alternância de código e, sim, PHP produzindo código HTML.

### 4. Outro código sem alternância

Já o código PHP, do primeiro exemplo, do item 2, poderia ser apresentado da forma abaixo, sem a alternância.

```

<?php
$cli_ip = getenv("REMOTE_ADDR");

echo "<HTML>
<HEAD>
<TITLE> Alternância entre PHP e HTML. </TITLE>
<BODY>";

echo "$cli_ip";

echo "</BODY> </HTML>";

?>

```

A orientação é usarmos as possibilidades disponíveis, mas não produzir códigos confusos ou de leitura difícil e dúbia.

O comando PHP que você verá com mais frequência é o

```
echo
```

que envia o argumento para a “saída padrão”. Ou seja, se utilizando o browser, é mostrado no navegador. O texto, que se sucede ao comando, deve aparecer entre aspas ou apóstrofos. No primeiro caso, ocorre a substituição de uma variável pelo seu conteúdo, ao contrário do segundo. Quando se deseja que apareça o próprio símbolo “\$” numa cadeia entre aspas, prefixá-lo com a “barra invertida”

## 5. Terminador de comandos.

Da mesma forma como na linguagem C e nos exemplos acima, os comandos devem terminar com o símbolo ponto e vírgula (";"). A única exceção tolerada é o último comando do script PHP, aquele que antecede o símbolo de final de script.

### a) Código correto

```
<?php
$i = 0;
echo "Valor da variável \ $i = $i";
?>
```

### b) Código com erro

```
<?php
$i = 0
echo "Valor da variável \ $i = $i";
?>
```

### c) Código correto

```
<?php
$i = 0;
echo "Valor da variável \ $i = $i"
?>
```

Como regra geral, utilize, sempre, o terminador de comandos.

## 6. Comentários em PHP

Em PHP, os comentários são indicados de forma análoga a outras linguagens de programação, conforme segue

- a) Por linha, como em C. (// ....)
- b) Por blocos, como em C. (/\* .....\*/
- c) Por linha, como em shell Unix. (#.....)

## 7. Variáveis e tipos.

Na linguagem PHP, não somente em questões relativas a tipos e variáveis, as definições têm sido alteradas e incrementadas nas sucessivas versões. Portanto, leve em conta a versão específica em que você está trabalhando.

As variáveis em PHP não são declaradas. Elas existem a partir da primeira utilização e o nome é “case sensitive”. O tipo é determinado pelo valor que recebem. Isto significa que em execuções subsequentes de um mesmo script, uma variável pode conter valores de tipos diferentes, de acordo com as condições de execução. E, dentro de um mesmo script, uma variável pode mudar de tipo de acordo com os valores que ela recebe. Assim, é razoável que, às vezes, você queira saber qual o tipo associado a uma variável. Para isto, utilize as funções:

```
gettype($var)
is_int($var)
is_string($var)
is_float($var)
is_array($var)
```

A primeira função, acima, retorna uma “string”, indicando o tipo; as demais retornam um “boolean” indicando “true” ou “false”.

Como visto, as variáveis PHP são identificadas pelo símbolo “\$” no início do nome.

Existem duas funções muito úteis para depuração de scripts que permitem verificar o conteúdo e outras informações sobre variáveis. Podem ser utilizadas em vetores aninhados e outras estruturas. São elas

```
print_r($variavel); // show variable contents
var_dump($variavel); // more detailed information
```

## 8. Sobre o tipo FLOAT

Da mesma forma como acontece com outras linguagens de programação, a utilização do tipo “float” produz uma imprecisão nos valores que é oriunda da forma como este tipo é representado internamente. Isto é inerente à representação e não à linguagem. Algumas linguagens, quando precisam de aritmética sem estes efeitos, não utilizam o float, mas tipos como NUMERIC, DECIMAL ou equivalente. Quando estiver utilizando ponto flutuante, cuidado com as comparações pela igualdade; considere sempre a imprecisão, mesmo que muito pequena.

Por exemplo, (0 == 0.00000000000000000001) é falso, embora no mundo real a diferença seja desprezível na maioria dos campos de aplicação.

Uma variável é do tipo float pelo valor que ela recebe, na notação em ponto flutuante. Por exemplo,

```
$floating = 1.2345;
```

## 9. Sobre o tipo “boolean”

Com relação a este tipo, o PHP é análogo a outras linguagens de programação.

### Que expressões são considerados FALSE:

- o próprio booleano **FALSE**
- o inteiro 0 (zero)
- o ponto flutuante 0.0 (zero)
- uma string vazia e a string contendo “\0”
- uma matriz sem elementos
- um objeto sem nenhum componente
- o tipo especial NULL (incluindo variáveis não definidas)

Qualquer outro valor é considerado **TRUE**.

### Exemplos:

```
$boolean = false; // Boolean type
$boolean = (1 == 1); // Boolean type
```

## 10. Sobre o tipo inteiro

Os valores inteiros podem ser especificados na forma decimal convencional, como nas formas octal e hexadecimal. Para indicar que o número está em octal, preceda com o algarismo 0 e com 0x para indicar que está em hexadecimal.

```
$int = 0555; // indica octal: equivale ao número decimal 365.
$int = 0x9af; // indica hexadecimal: equivale ao número decimal 2479
$int = 999; // indica decimal
```

## 11. Comando “if”

Não seria necessário, mas somente para registrar, a sintaxe deste comando é a seguinte:

```
if (boolean_condition) {
... // comandos da parte true
}
else{
... // comandos da parte false
}
```

## 12. Casting

Você pode fazer a conversão de tipo do conteúdo de uma variável através do casting. Por exemplo

```
$int = (int) $float;
```

### 13. Formatação de números em cadeias

Você pode apresentar números em vários formatos na saída padrão, utilizando a função abaixo:

```
printf
```

Analogamente, é possível formatá-los numa cadeia de caracteres, em lugar de enviá-los para a saída padrão com a função:

```
sprintf
```

Você encontrará várias maneiras de formatar, mas segue algumas formas:

```
$cadeia = sprintf("O numero 500 escrito em octal: %o.",500);
$cadeial = sprintf("O numero 500 escrito em hexa: %x.",500);

echo "$cadeia <br>";
echo $cadeial;
```

Isto produzirá o seguinte:

```
O numero 500 escrito em octal: 764.
O numero 500 escrito em hexa: 1f4.
```

### 14. Valores ASCII e caracteres correspondentes

A interpretação de um valor como ASCII ou caracter pode ser alternada através das funções:

```
printf("%d <br>",ord("A")); // interpretado como ASCII, na notação
                             // decimal
printf("%c <br>",65);       // O decimal é interpretado como o
                             // caracter cujo ASCII é 65,
                             // pela função printf
                             // através do modificador "c".
printf("%s <br>",chr(65));  // O decimal é interpretado como caracter
                             // e apresentado como "string".
printf("0x%x <br>",ord("z")); // interpretado como ASCII, apresentado
                             // na notação hexadecimal
```

### 15. Alguns especificadores de tipo da função printf e assemelhados (type specifier)

```
%b - Argumento interpretado como inteiro e apresentado em binário
%c - Argumento interpretado como inteiro e apresentado como o caractere ASCII
%d - Argumento interpretado como inteiro e apresentado como um número decimal
%e - Argumento interpretado como número na notação científica
%u - Argumento interpretado como inteiro e apresentado como um número decimal sem sinal
%f - Argumento interpretado como float e apresentado como ponto flutuante
%o - Argumento interpretado como inteiro e apresentado como um número em octal
%s - Argumento interpretado e apresentado como uma cadeia de caracteres (string)
%x - Argumento interpretado como inteiro e apresentado em hexadecimal em letras
    minúsculas
%X - Argumento interpretado como inteiro e apresentado em hexadecimal em letras
    maiúsculas
```

## 16. Strings

Continuando com os assuntos “variáveis” e “tipos”, vamos apresentar algumas características de ‘string’ que armazenam cadeia de caracteres.

- i. Não existe limitação teórica quanto ao tamanho da cadeia.
- ii. Os delimitadores comuns para cadeias constantes de caracteres são as aspas ( " ) e apóstrofes ( ' ). Para expressar o próprio símbolo, prefixe-o com o símbolo de “barra invertida”.
- iii. Existem algumas diferenças na interpretação das cadeias dependendo do delimitador. Por exemplo, as variáveis são substituídas quando utilizadas em cadeias delimitadas por aspas, mas não nas delimitadas por apóstrofes.
- iv. A especificação de cadeias pode se estender por diversas linhas.
- v. Existem situações em que nomes de variáveis se confundem com o texto. Para evitar isto, você deve utilizar o símbolo “abre/fecha chaves” em torno do nome da variável, ou seja, algo semelhante a scripts em shell.

```
${var}
```

No entanto, em PHP, você pode escrever também

```
{ $var } .
```

- vi. O símbolo da operação de concatenação de cadeias é o “.” (ponto).
- vii. As funções *strtoupper* e *strtolower* fazem a alteração de letras minúsculas para maiúsculas e vice-versa. Aplicam sobre strings e retornam strings.
- viii. Os caracteres que compõem uma cadeia podem ser tratados individualmente, indexando-os com índice numérico, iniciando pelo índice zero.

```
$var = "abc";  
echo "$var[0]"; // imprime o primeiro elemento da cadeia
```

- ix. O número de elementos de uma cadeia pode ser obtido com a função

```
strlen($var)
```

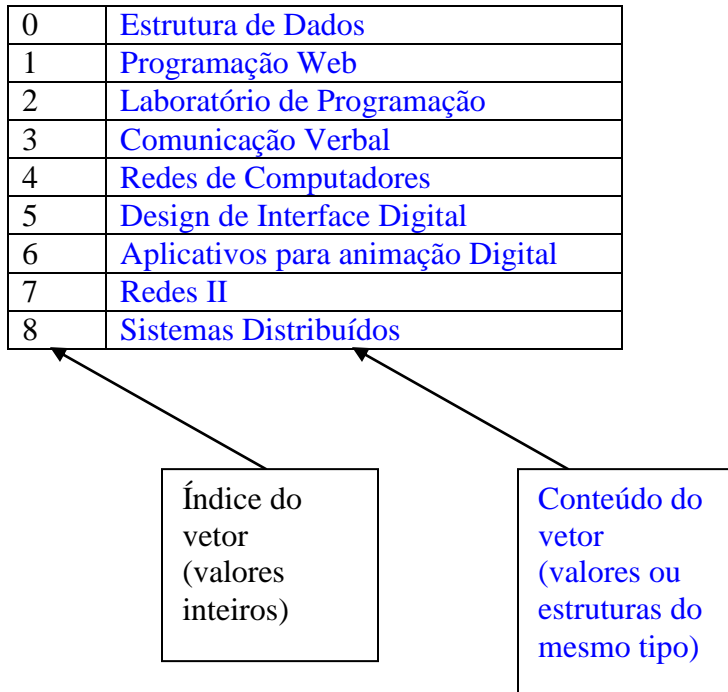
## 17. Arrays

Estamos acostumados a ver “array” como uma sequência de estruturas homogêneas indexadas por números inteiros, isto é, os elementos distintos possuem a mesma estrutura. Por exemplo, um vetor de inteiros, é uma sequência de valores inteiros indexada por números inteiros.

Em PHP, os elementos podem ter estruturas diferentes e, além disso, os índices são sequências de elementos que podem ser inteiros ou strings. Suponha um vetor de nome

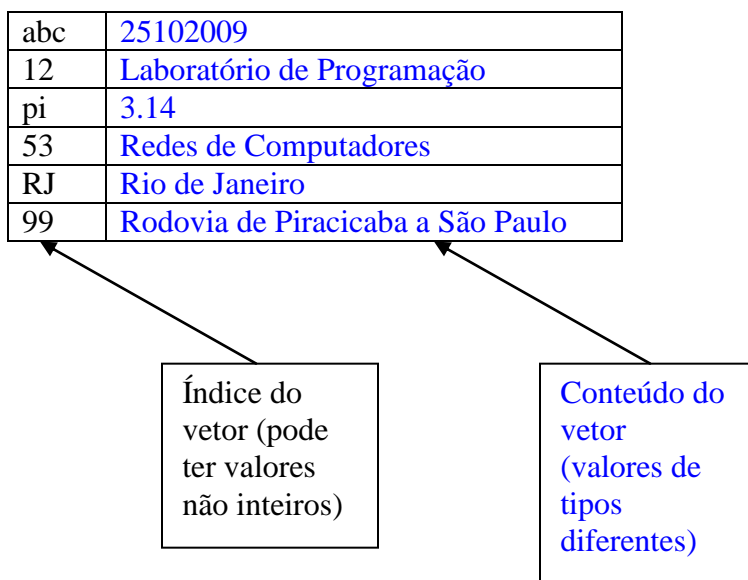


\$disciplinas. Normalmente, os índices são inteiros e os valores contidos são homogêneos, na maioria das linguagens de programação.



De forma diferente, em PHP, os índices não precisam ser inteiros e os conteúdos não precisam ser elementos de um conjunto de valores homogêneos. Os índices inteiros não precisam estar em sequência.

Por exemplo, vejamos a estrutura de um vetor em PHP:



Existem muitas formas de se inicializar uma variável do tipo vetor. Veja a seguir.

a) Exemplo 1

```
$vetor = array("abc" => 12, 12 => "abc", "xyz" => "Fulano", 13 => 1900);
```

b) Exemplo 2

```
$vetor[5] = 20;  
$vetor[10] = 150;  
$vetor["BD"] = 4;
```

c) Exemplo 3

```
$dia_sem = array("Domingo", "Segunda", "Terça", "Quarta",  
                "Quinta", "Sexta", "Sábado");
```

d) Exemplo 4. Para inicializar o vetor a partir do índice 159. Se não especificado, inicia com o índice zero.

```
$meses[159] = "Janeiro";  
$meses[] = "Fevereiro";  
$meses[] = "Março";  
$meses[] = "Abril";  
$meses[] = "Maio";  
$meses[] = "Junho";  
$meses[] = "Julho";  
$meses[] = "Agosto";  
$meses[] = "Setembro";  
$meses[] = "Outubro";  
$meses[] = "Novembro";  
$meses[] = "Dezembro";
```

e) Exemplo 5. Alternativamente, para inicializar o vetor a partir do índice 159 pode ser feito da seguinte forma:

```
$meses = array(159=> "Janeiro", "Fevereiro", "Março", "Abril",  
                "Maio", "Junho", "Julho", "Agosto", "Setembro",  
                "Outubro", "Novembro", "Dezembro");
```

Importante ressaltar que não existem posições vazias do vetor para índices inferiores a 159.

f) Outras formas de inicialização:

```
$abc=array(9 => "LP", "ED", "ES",  
           "IPT", "COMP", "LP",  
           "abc" => "xyz", "za" => 555);
```

Exemplos de outras operações sobre vetores ou seus elementos.

```
unset($vetor[5]); // remove a estrutura de índice 5 e o valor
correspondente do $vetor.

unset($vetor); // remove todo o $vetor.

print_r($vetor); // imprime o vetor com índice/valor e propriedades.
var_dump($vetor); // imprime o vetor com índice/valor e propriedades.

$n = count($vetor); // retorna o número de elementos do vetor.
```

## 18. Percurso pelos elementos de um vetor

A estrutura mais natural para tratar os elementos de um vetor é fornecida pelos comandos “for” e “foreach”.

- a) Se os índices são inteiros, em sequência, e os limites conhecidos:

```
for ($i = $inicio ; $i <= $fim; $i++ ) {

    // Trata o elemento $array[$i]...

}
```

Para este caso, você pode utilizar a função `count($array)` para obter a quantidade de elementos e calcular o limite superior do índice.

- b) Os índices não são inteiros, ou eles não são sequenciais. O comando **foreach**, para controle de iterações, nos permite uma forma mais genérica de percurso pelo vetor.

```
foreach ( $vetor as $elemento ) {

    // Trata o elemento do vetor que está disponível
    // na variável $elemento

}
```

- c) Caso se queira tratar os elementos do vetor e também os seus respectivos índices:

```
foreach ( $vetor as $indice => $elemento ) {

    // Na variável $indice está o índice corrente.
    // Na variável $elemento está o conteúdo corrente.

}
```

Para mostrar o conteúdo e/ou índice de vetores, percorre-se a sua estrutura e utiliza-se algum comando que imprime na saída padrão, como se faz com qualquer outro tipo de variável.

Para verificar a estrutura e conteúdo dos vetores, utilize as funções:

```
print_r($vetor)
var_dump ($vetor)
```

## 19. Vetores bidimensionais em PHP

Em muitas linguagens de programação, como visto nos vetores simples, as matrizes bidimensionais contém elementos homogêneos. Ou seja, cada elemento é do mesmo tipo. Veja o exemplo abaixo. O vetor bidimensional tem índices inteiros e os seus elementos são todos do mesmo tipo.

	0	1	2	3
0	AA	AB	AC	AD
1	BA	BB	BC	BD
2	CA	CB	CC	CD
3	DA	DB	DC	DD

Agora, veja o seguinte exemplo. Deseja-se tratar os dados dos nomes das pessoas, os estados de residência, peso e altura. Seguem as alternativas de vetor de registros e vetores múltiplos. As cores têm meramente a função de facilitar a distinção dos elementos.

### a) Vetor de registros ou estruturas

Neste caso, cada elemento do vetor é um registro e todos os registros têm a mesma estrutura com quatro campos.

#### Pessoas (Array de registros)

0	Ana	São Paulo
	60	70
1	Jeca	São Paulo
	55	60
2	Cris	Paraná
	49	55
3	Tatu	Minas
	56	65

## b) Múltiplos vetores

Pessoas		Estado		Peso		Altura	
1	Ana	1	São Paulo	1	60	1	70
2	Jeca	2	São Paulo	2	55	2	60
3	Cris	3	Paraná	3	49	3	55
4	Tatu	4	Minas	4	56	4	65

Em PHP, podemos ter vetores bidimensionais com índices não numéricos e conteúdos não homogêneos. Veja o exemplo:

### Pessoas (Array em PHP)

	Nome	Estado	Peso	altura
1	Ana	São Paulo	60	70
2	Jeca	São Paulo	55	60
3	Cris	Paraná	49	55
4	Tatu	Minas	56	65

Cada elemento do vetor pode ser referenciado da seguinte forma:

```
$pessoas[4]["Estado"] = "Minas";
```

Esta notação é muito semelhante à forma “vetor de registros” de outras linguagens nas quais os elementos podem ser referenciados da forma abaixo

```
pessoas[4].Estado = "Minas"
```

A diferença é que o nome do "campo" do registro é constante. No caso do array em PHP, os elementos podem ser percorridos como índices nas duas dimensões.

Outro ponto importante é que, da mesma forma como as variáveis passam a existir quando são utilizadas, novas colunas podem ser criadas no vetor. Por exemplo, na tabela Pessoas não existe a coluna "Cidade\_Natal", mas passará a existir assim que fizer o seguinte:

```
Pessoas[1]["Cidade_Natal"] = 'New York';
```

## 20. Variáveis que referenciam endereço.

A partir da versão 4.0 do PHP, as variáveis podem receber endereços em lugar de valores. Podemos dizer que são duas ou mais variáveis distintas, mas referenciam o mesmo objeto.

```
<?php

$mat1 = "Banco de dados";
$mat2 = &$mat1;
$mat1 = "Bancos de dados distribuídos";

echo "<br>mat1 = $mat1";
echo "<br>mat2 = $mat2";
?>
```

## 21. Constantes

Definição de constante:

```
define("CONST_PI", 3.141592653589793);
```

E você pode utilizar os nomes das constantes nas expressões.

```
<?php
.
.
.
define("CONST_PI", 3.141592653589793);
.
.
.
$area=CONST_PI * pow($raio,2);
.
.
.
?>
```

Observações sobre constantes:

- i. Constantes não possuem o símbolo “\$” no início do nome, ao contrário de variáveis;
- ii. Por convenção, utilizam-se letras maiúsculas para designar constantes;
- iii. Os nomes de constantes são, também, “case sensitive”;
- iv. O valor de uma constante é definido pela função “define”;
- v. Não pode utilizar a operação de atribuição para definir uma constante;
- vi. Constantes não seguem as regras de escopo, ao contrário das variáveis. Elas estão disponíveis em qualquer ponto do código PHP;

- vii. Uma constante não pode ter o seu valor alterado; Se você re-definir a constante, é possível que não receba nenhuma mensagem de erro, mas provavelmente, o seu valor não será efetivamente alterado;
- viii. O PHP define uma série de constantes que você pode utilizar. Elas são reportadas em várias fontes de documentação. Para ver a lista de constantes vigentes na execução do script, utilize a função

```
$constants_list=get_defined_constants();  
var_dump ($constants_list);
```

- ix. Constantes representam somente valores escalares: inteiro, float, boolean e strings.

## Atividades de hoje

Todos os arquivos com os scripts PHP precisam ser enviados ao servidor por “secure ftp”.

Para as atividades, são suficientes as informações acima, no entanto se quiser ver o help do PHP, procure em

[http://172.19.2.126/daw/php\\_help](http://172.19.2.126/daw/php_help)

1. Crie o script simples.php que utilize as seguintes funções:

```
$cli_ip = getenv("REMOTE_ADDR");  
$cli_prt= getenv("REMOTE_PORT");  
$srv_ip = getenv("SERVER_ADDR");  
$srv_prt= getenv("SERVER_PORT");
```

O seu script precisa criar um documento HTML e mostrar o conteúdo das variáveis no browser, um valor por linha. Envie o script ao servidor e carregue-o no navegador.

Execute, seguidas vezes, este script, verifique e analise o comportamento dos valores apresentados.

2. Crie o script de nome float.php que inclua o seguinte conteúdo

```
$float = 0.1 + 0.7;  
$float1 = $float * 10;  
$int = (int) ($float1);
```

O seu script precisa apresentar no browser os valores finais das três variáveis. Envie o arquivo ao servidor e veja como ficou. Explique o resultado.

3. Crie o script “tipo.php”, atribuindo valor inteiro, float e boolean a três variáveis diferentes. Veja o conteúdo com as funções “var\_dump” e “print\_r”.

4. Crie o script inteiro.php que atribui valores inteiros às variáveis de nome

```
$decimal  
$octal  
$hexa
```

Todas as três variáveis devem conter o inteiro decimal de valor 746, mas devem ser expressos, no comando de atribuição, em octal para a variável \$octal e em hexadecimal para a variável \$hexa. O resultado deve aparecer no formato abaixo para as três variáveis quando interpretados como decimais. Se realizado corretamente, deverá aparecer o seguinte:

```
Variavel $decimal = 746  
Variavel $octal = 746  
Variavel $hexadecimal = 746
```

Verifique o resultado no browser.

5. Sobre “casting”. Crie o script casting.php com os seguintes passos:

i. Atribuição de valores:

```
$boolean = false;  
$inteiro = 10;  
$inteiro1 = 0;  
$float = 1.2;  
$cadeia="abcd";  
$cadeial="";
```

ii. Imprime o conteúdo das variáveis no formato:

```
$cadeia = abcd
```

Onde o lado esquerdo é o nome da variável (com o símbolo “\$”) e o lado direito o seu conteúdo.

iii. Verifica cada uma das variáveis com a função var\_dump.

iv. Verifica cada uma das variáveis com casting para ‘boolean’, utilizando a função var\_dump, da seguinte forma:

```
var_dump ( (boolean) $inteiro );
```

Analisar quais valores são considerados falsos ou true.



## 6. Sobre correspondência entre caracteres e os valores ASCII. Crie o script

`intl.php`

que contenha os comandos abaixo.

```
$character='z';  
$decimal=122;  
$character1=ord($character);
```

E, utilizando a função “printf”, apresente:

- a) O valor armazenado pela variável

`$character1`

em decimal;

- b) O valor armazenado pela variável

`$decimal`

como caracter.

- c) O valor armazenado pela variável

`$decimal`

como “string”.

## 7. Cadeias.

Codifique o script PHP de nome

`cadeias.php`

que contenha as seguintes características, apresentando em cada item o conteúdo da variável com o comando “echo”.

- a) Exemplo de uma cadeia que expande a variável com o seu conteúdo;
- b) Exemplo de uma cadeia que não expande a variável;
- c) Exemplo de como isolar o nome de uma variável com o restante do texto que forma a cadeia;
- d) Exemplo de concatenação de cadeias;

- e) Atribua a uma variável uma cadeia de caracteres com quebras em várias linhas e apresente o seu conteúdo com o comando *echo*;
- f) Percorra cada elemento da cadeia e transforme a letra correspondente em maiúscula se a posição for par; em minúscula se for ímpar. Utiliza a função *strlen* para saber a quantidade de caracteres e percorrer a estrutura da cadeia. Apresente o conteúdo da variável.

## 8. Operação unset em vetores.

Na maioria das linguagens, um vetor é indexado por inteiros e você pode manipular, à vontade, os seus valores, incluindo os índices em suas expressões; no entanto, não pode fazer nada com a existência dos elementos. Nessas linguagens, pode-se assumir, por exemplo, que, se o conteúdo do elemento for “-1”, então a célula deve ser ignorada. Em PHP, é possível remover um elemento do vetor, ou seja, o elemento deixa de existir. Isto não se trata de remover apenas o conteúdo.

- a) Construa o vetor abaixo no script `array1.php`:

```
$vetor = array("abc" => 12, 12 => "abc", "xyz" => "Fulano", 13 => 1900);  
  
$vetor[5] = 20;  
$vetor[10] = 150;  
$vetor["BD"] = 4;
```

- b) Veja o conteúdo com a função `var_dump`;
- c) Remova os elementos do vetor dos índices 12 e "xyz";
- d) Veja o conteúdo com a função `var_dump`.

## 9. Vetor indexado sequencialmente por número inteiro

Crie um vetor no arquivo `array2.php`, conforme segue:

- a) Vetor indexado por inteiros, começando no índice 11 e os demais com os números subsequentes. Os elementos devem conter os nomes das disciplinas do curso de Sistemas de Informação. Inclua 5 disciplinas.
- b) Utilize o comando “**for**” para imprimir a posição e o conteúdo de cada célula do vetor. Embora, no exemplo, você tenha incluído 5 elementos, genericamente, este número poderá ser outro. Estabeleça os limites do ciclo, de forma conveniente.

## 10. Vetor com índices não inteiros – Tratamento de conteúdo

Crie o script de nome `array3.php` que faça o seguinte:

a) Inicializa o vetor de \$créditos.

```
$creditos = array( "BD" => 4,  
                  "LP" => 4,  
                  "SO" => 3,  
                  "IC" => 3,  
                  "COMP" => 4,  
                  "ES1" => 2,  
                  "ES2" => 2,  
                  "TG" => 3,  
                  "IPT" => 2,  
                  "EF" => 1,  
                  "FG" => 2,  
                  "IA" => 2,  
                  "TEC1" => 3,  
                  "TEC2" => 3);
```

b) Utiliza o comando

```
foreach ($vetor as $conteudo)
```

para obter a soma de créditos das disciplinas

c) Apresenta o valor final obtido para total de créditos

## 11. Vetor com índices não inteiros – Tratamento de índice e conteúdo

Realize as tarefas abaixo no script array4.php:

RA444444	Capitu
RA555555	Cotrim
RA666666	Quincas Borba
RA777777	Brás Cubas
RA888888	Iaiá Garcia
RA999999	Virgília

Índice do  
vetor é pelo  
registro  
acadêmico

Nome dos  
alunos

a) Inicialize o vetor com o nome \$pessoas, com os valores da figura.

- ## 12. Vetores bi-dimensionais

A matriz acima, de nome \$pessoas, possui linhas indexadas por inteiros. As colunas são indexadas pelos valores (Matricula, Nome, Salario e Novo\_Salario). Crie o script no arquivo array5.php que realize o seguinte:

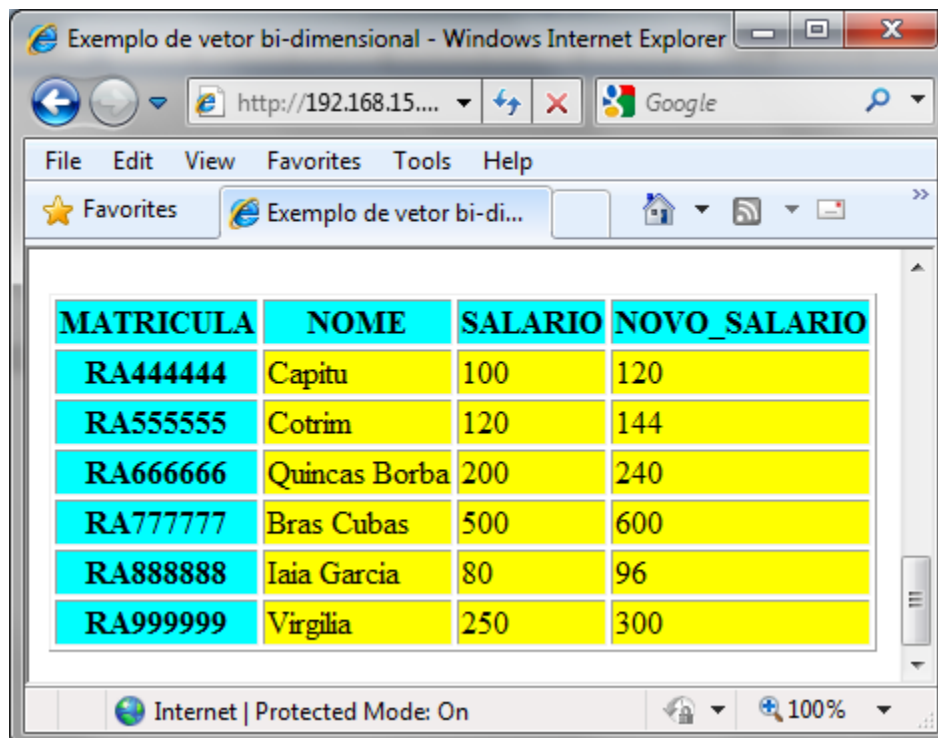
- ```
$pessoas[1][MATRICULA] = "RA4444444";
$pessoas[2][MATRICULA] = "RA5555555";
$pessoas[3][MATRICULA] = "RA6666666";
.
.
.
$pessoas[1]["NOME"] = "Capitu";
$pessoas[2]["NOME"] = "Cotrim";
$pessoas[3]["NOME"] = "Quincas Borba";
.
.
.
$pessoas[1]["SALARIO"] = 100;
```

```
$pessoas[2]["SALARIO"] = 120;  
$pessoas[3]["SALARIO"] = 200;  
.  
.  
.
```

- c) Apresente o vetor com a função `var_dump`;
- d) Atualize a coluna `Novo_Salario` com valor do `Salario` multiplicado pela constante de nome “FATOR”. Esta coluna não existe, mas assim que for atualizada, ela será criada. Para cada índice do array `pessoas`, faça algo como segue:

```
$pessoas[$indice]["NOVO_SALARIO"] =  
    FATOR * $pessoas[$indice]["SALARIO"];
```

- e) Apresente o resultado final com a função `var_dump`. Veja como é apresentada a composição de um vetor, dentro de outro vetor.
- f) Suponha que você não saiba quantas linhas o vetor `$pessoas` possui e nem mesmo quais são as colunas, porque poderiam existir mais do que aparecem na figura. E os valores armazenados no vetor, também podem ser alterados durante a execução do script. Acrescente trecho de código PHP que produza código HTML para apresentar no browser o conteúdo do vetor em formato tabular. Será necessário utilizar o comando “**foreach**” em dois níveis e incluir uma lógica para tratar, de forma diferente, os valores das células e o índice não numérico que aparecerá no formato tabular. Considerando o exemplo, a tabela deve contemplar os seguintes valores, sem levar em conta a aparência.



| MATRICULA | NOME          | SALARIO | NOVO_SALARIO |
|-----------|---------------|---------|--------------|
| RA444444  | Capitu        | 100     | 120          |
| RA555555  | Cotrim        | 120     | 144          |
| RA666666  | Quincas Borba | 200     | 240          |
| RA777777  | Bras Cubas    | 500     | 600          |
| RA888888  | Iaia Garcia   | 80      | 96           |
| RA999999  | Virgilia      | 250     | 300          |

### 13. Variáveis que referenciam endereço.

Crie o script de nome referencia.php que faz o seguinte:

- a) Atribui dois valores inteiros distintos às variáveis

`$x` e `$x_ref`

- b) Mostra o conteúdo das duas variáveis;
- c) Atribua à variável `$x_ref` o endereço da variável `$x`;
- d) Atribua um valor diferente à variável `$x`;
- e) Mostra o conteúdo das duas variáveis.

Até a próxima aula!

Prof. Satoshi Nagayama