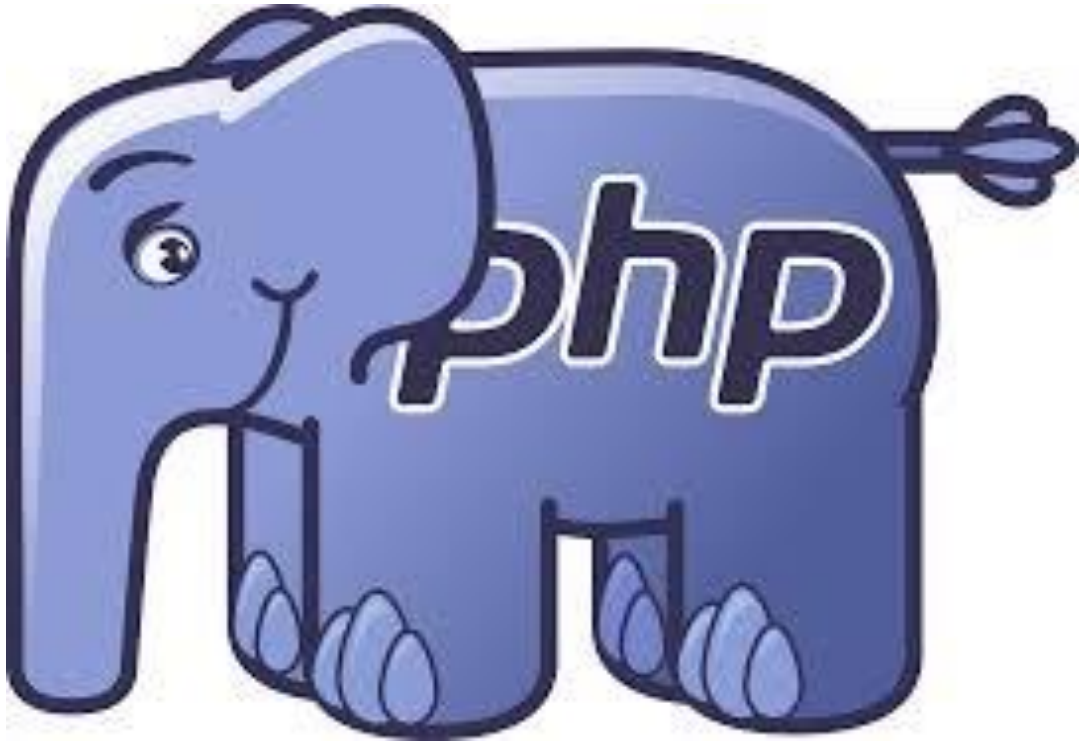

AULA 3

EXPLICADORES.NET

@explicadoresnet

Arrays

As variáveis comuns (também chamadas de variáveis escalares) podem armazenar apenas um valor por vez. Um array (vetor) pode armazenar vários valores ao mesmo tempo, pois se trata de uma estrutura de armazenamento que, assim como as variáveis, possui um identificador, mas além disso há um índice associado (que pode ser um número ou um texto), e cada índice indica uma posição de memória em que fica armazenado um elemento do array. O índice deve aparecer entre colchetes ([]) logo após o identificador do array.

Arrays

```
$aluno [0] = 8.0;
$aluno [1] = 9.0;
$aluno [2] = 2.5;

$aluno [] = 8.7;     // Acrescenta ao próximo espaço livre

$notas = array range (1, 10);

$notas = array (8.5, 7.0, 4.5, 8.0);

$cadastro [“nome”] = “José”;
$cadastro [“idade”] = 23;
$cadastro [“peso”] = 93.5;   // Chave associativa

$pessoa = array (“nome" => “Paulo”, “idade" =>30);
```


```php
$array = array(1, 2, 3);
```

```php
$array = [1, 2, 3];
```

```php
$array = array(
 "chave1" => 1,
 "chave2" => "PHP",
 "chave3" => false
 );
```

$notas = array range (1, 10);

$cadastro ["nome"] = "José";
$cadastro ["idade"] = 23;
$cadastro ["peso"] = 93.5;

$aluno [] = 8.7; // Acrescenta ao próximo espaço livre

$aluno [0] = 8.0;
$aluno [1] = 9.0;
$aluno [2] = 2.5;

Acessando os índices de um array

```php
echo $array[0];
echo $array[1];
echo $array[2];
```

```php
echo $array["chave2"];
```

```php
$array["chave2"] = 2;
```


Para Cada

```
$vetor = array range(1,30);
foreach ($vetor as $valor)
{
    echo “O valor atual é $valor <br>”;
}
```

```
$vetor = array (“nome" => “Gustavo”, “profissão" =>
“Professor”, “idade" => 36);
foreach ($vetor as $chave => $valor)
{
    echo “Campo $chave contém $valor <br>”;
}
```


exemplo5_7.php

```php
<?php
    for($cont=0 ; $cont<10 ; $cont++)
    {
        echo "A variável \$cont vale $cont";
        echo "<br>";
    }
 ?>

<html>
<body>
<?php
    echo "Estou fazendo uma contagem regressiva: <br>";
    for($i=15 ; $i>=0 ; $i-)
    {
        echo $i . ", ";
    }
    echo "... FIM!";
?>
</body>
</html>
```


```php
<?php
    $vetor[0][0]= "elemento00";
    $vetor[0][1]= "elemento01";
    $vetor[1][0]= "elemento10";
    $vetor[1][1]= "elemento11";
    for($i=0 ; $i<2 ; $i++)
    {
        for($k=0 ; $k<2 ; $k++)
        {

            echo "O elemento da posição $i,$k é ";
            echo $vetor [$i][$k] . "<br>";
        }
    }
```


```php
<?php
    for( $i=0, $k=10  ;  $i<10  ;  $i++, $k- )
    {
        echo "\$i vale $i e \$k vale $k";
        if ($i==$k)
            { echo " (os valores são iguais!)"; }
        echo "<br>";
    }
?>
```


Controlando o fluxo de execução

Existem comandos que podem ser usados em conjunto com essas estruturas de controle que acabamos de ver. Esses comandos são o *break* e o *continue*.

Vamos ver qual é a utilidade de cada um deles:

break

O *break* termina a execução do comando atual, que pode ser um *if*, *for*, *while*, ou *switch*. Quando o *break* é encontrado dentro de algumas dessas estruturas, o fluxo de execução passa para o primeiro comando após o término dessa estrutura. É um recurso que podemos utilizar para forçar a saída de um laço ou de um comando condicional. Acompanhe o exemplo a seguir:

Interrompendo o fluxo

```
$vetor = array (1, 4, 6, 2, -1, 8, 7, 5);
foreach ($vetor as $valor)
{
    if ($valor == -1) {
        break;
    }
    echo “O valor atual é $valor <br>”;
}
```


continue

O comando *continue* é utilizado dentro dos comandos de repetição para ignorar o restante das instruções pertencentes ao laço corrente, e ir para a próxima iteração (voltando ao início do laço). Veja o exemplo a seguir:

exemplo5_13.php

```
<?php
    $vetor = array (1, 3, 5, 8, 11, 12, 15, 20);
    for($i=0 ; $i<sizeof($vetor) ; $i++)
    {
        if ($vetor[$i] % 2 != 0)     // é impar
        { continue;  }
        $num_par = $vetor[$i];
        echo "O número $num_par é par. <br>";
    }
?>
```


Modificando o fluxo

```
$vetor = array (1, 4, 6, 2, -1, 8, 7, 5);
for ($i = 0; $i < sizeof($vetor); $i++)
{
    if ($vetor[$i] % 2 == 0) {
        continue;
    }
    echo “O valor atual é $valor <br>”;
}
```


Rotinas

Funções

```
function soma ($a, $b, $c) {
    $s = $a + $b + $c;
    echo “A soma entre $a , $b e $c é igual a $s”;
}
```

```
function soma ($a, $b, $c) {
    $s = $a + $b + $c;
    return $s;
}
```


Passagem de parâmetro

Por valor

```
function incrementa ($x) {
    $x++;
    echo $x;
}

// programa principal

$a = 3;
incrementa ($a);
echo $a;
```


Passagem de parâmetro

Por referência

```
function incrementa (&$x) {
    $x++;
    echo $x;
}

// programa principal

$a = 3;
incrementa ($a);
echo $a;
```


Considere a seguinte função:

```
function nada ($a=10 , $b, $c)
{
    ...
}
```

Essa definição está errada, pois como o envio do parâmetro $a é opcional, poderíamos fazer a seguinte chamada:

```
nada (1, 5);
```

Isso causaria um erro, porque o PHP interpretará os dois parâmetros passados como $a e $b, e ocorreria um erro de execução devido à falta do parâmetro $c. Portanto, o modo correto de definir a função seria:

```
function nada ($b, $c, $a=10)
{
    ...
}
```


```php
<?php
    function dobro ($valor)
    {
        $valor = 2 * $valor;
    }
    function duplica (&$valor)
    {
        $valor = 2 * $valor;
    }
    $valor = 5;
    dobro ($valor);
    echo $valor . "<br>";
    duplica ($valor);
    echo $valor;
?>
```


Funções recursivas

Chamamos de funções recursivas aquelas funções que chamam a elas mesmas. Veja um exemplo simples desse tipo de função:

exemplo6_10.php

```
<?php
    function teste ($valor)
    {
        if ($valor!=0)
        {
            echo "Foi chamada a função teste passando o valor $valor <br>";
            teste ($valor-1);
        }
    }
    teste (7);
?>
```


```php
<?php
for ($i = 1; $i <= 10; ++$i) {
    echo $i;
    $i++;
}
```

```php
<?php
$Fab= 20;
$b=&$Fab;
$b=5;
echo $Fab;
```


```php
<?php
    function incrementa(&$x){
        $x++;
        //global $a;
        echo "valor de x: $x <br>";

        echo "valor de a: $a <br> ";
    }

    $a=3;
    incrementa($a);
    echo "PP valor de a: $a";
?>
```

valor de x: 4
valor de a:
PP valor de a: 4

```php
<?php
    function incrementa(&$x){
        $x++;
        global $a;
        echo "valor de x: $x <br>";

        echo "valor de a: $a <br> ";
    }

    $a=3;
    incrementa($a);
    echo "PP valor de a: $a";
?>
```


```php
function funcEstudo(){
    $valor1 = 10;
    $valor2 = 16;
    $total = $valor1 + $valor2;
}

funcEstudo();

echo $total;
```

```php
$x = 'Fala'; // global $x
function funcaoX()
{
    global $x;
    $x .= ' galera!!!';
    echo $x;
}

funcaoX();
```


Includes

Includes

Arquivo [cabecalho.php]

```
<?php echo “<h1>Site do Educandus</h1>”;
echo “<img src=‘educanduslogo.png’>”; ?>
```

Arquivo [index.php]

```
<!DOCTYPE html>
<html>
    <head><title>Educandus</title></head>
    <body>
        <?php include “cabecalho.php”; ?>
        <h2>Preparatório EAGS</h2><hr/>
    </body>
</html>
```


Include vs. Require

Include:

Em caso de erro, gera um aviso (warning), mas continua a execução do script.

Require:

Em caso de erro, gera um erro fatal (compile error), e interrompe totalmente a execução do script.

Exemplo include

data.inc

```
<?php
  $meses = array (“”, “Jan”, “Fev”, “Mar”, “Abr”, “Jun”,
“Jul”, “Ago”, “Set”, “Out”, “Nov”, “Dez”);
  $d = date (“d”, time( ));
  $m = date (“m”, time( ));
  $a = date (“Y”, time( ));
  echo “Hoje é dia $dia de ” . $meses[$m] . “ de $ano”;
?>
```

pagina.php

```
<?php include “data.inc”; ?>
```


Variáveis de Ambiente

Variáveis de ambiente

```
$ip = getenv("REMOTE_ADDR");
echo "<h1> Seja bem-vindo ao meu site </h1>";
echo "<p> Prezado visitante, seu IP é $ip </p>";
```

```
$sw = getenv("SERVER_SOFTWARE");
echo "<p> Esse servidor utiliza o software $sw </p>";
```


Alguns exemplos...

REQUEST_METHOD método de requisição usado

SERVER_NAME nome do servidor (em alguns casos, o IP)

SERVER_PROTOCOL protocolo usado (http/1.1)

SCRIPT_NAME nome do script atual

QUERY_STRING tudo aquilo que vier após o ?

CONTENT_TYPE tipo de conteúdo (text/html)

* para uma lista completa dos getenv() suportados, usar o phpinfo()

Funções de Manipulação

STRINGS

Caractere

strtoupper - Toda a string em maiúsculas.

Echo strtoupper (“explicadores”); // EXPLICADORES

strtolower - Toda a string em minúsculas.

Echo strtolower (“EXPLICADORES”); // explicadores

substr - Retorna parte de uma string.

Echo substr (“Explicadores”, 2); // plicadores

Echo substr (“Explicadores”, 2, 5); // plica

Echo substr (“Explicadores”, -3); // res

```
<?php
   $texto = "blog alex - tudo sobre php.";

   echo ucfirst($texto) . "<br />"; //
Resultado: Só a primeira letra do texto em
Maiúscula

   echo ucwords($texto); // Resultado:
Todas as iniciais do texto em Maiúscula
?>
```


str_pad - Preenche string com espaços ou caracteres.

Echo str_pad (“Explicadores”, 15, “#”); // Explicadores###

str_repeat - Repete uma string.

Echo str_repeat (“Expli”, 3); // ExpliExpliExpli

strlen - retorna o tamanho de uma string.

Echo strlen (“Explicadores”); // 12

str_replace - Substitui uma string por outra.

Echo str_replace (“x”, “s”, “Explicadores”); // esplicadores

strpos - Encontra a | **Posição da primeira ocorrência do caracter na string**

Echo strpos (“Explicadores”, “i”); // 4

strstr - Encontra a primeira ocorrência em uma string.

Echo strstr (“Explicadores”, “a”); // adores

empty - verifica se uma string está vazia.

$a = 15;
empty($a); // false

```php
<?php
$texto = "eu não sou besta pra tirar onda de herói";

echo substr($texto, 0, 16);
echo "<br>\\n";
echo substr($texto, 11);
echo "<br>\\n";
echo substr($texto, 11, 9);
echo "<br>\\n";
echo substr($texto, -5);
echo "<br>\\n";
?>
```


Também podemos utilizar a substr() em combinação com a strpos(). A **strpos** detecta a posição que uma string ocorre dentro de uma expressão.

No exemplo seguinte, detectamos onde ocorre a palavra "http://" para retornar somente o domínio presente no texto:

```
<?php
$texto = "retornarei somente o domínio de http://www.pablo.blog.br";
$posicao = strpos($texto, 'http://');
echo substr($texto, $posicao+11);

?>
```


Funções de Manipulação

NÚMEROS

Números

abs - valor absoluto de um número

```
echo abs(-4.5) // 4.5
```

base_convert - Converte valores entre bases.

```
echo base_convert(35, 10, 2); // 100011
```

bindec hexdec octdec decbin dechex decoct - Realiza conversões pontuais.

```
echo bindec (1001); //9
```

ceil - arredonda frações para cima.

```
echo ceil(4.1); //5
```

floor - arredonda frações para baixo.

```
echo floor(4.9); //4
```

round - arredonda frações pela regra de arredondamento.

```
echo round(4.3); //4
```


pow - Eleva um valor ao outro (exponenciação).

```
echo pow(3, 2); // 9
```

sqrt - Calcula a raiz quadrada.

```
echo sqrt(9); // 3
```


Funções de Manipulação

VARIÁVEIS

Variáveis

gettype - Obtém o tipo da variável

```
echo gettype(3); // integer
```

settype - Configura o tipo da variável

```
echo settype($v, “integer”);
```

doubleval floatval - Retorna o valor em ponto flutuante.

intval - Retorna o valor em inteiro.

strval - Retorna o valor como string.

isset - Verifica que a variável foi configurada.

```
echo isset($variavelNaoCriada); // false
```

unset - Exclui uma variável do programa.

```
unset($variavelCriada);
```


```php
<?php
    $variavel = 'Diferença';

    if(isset($variavel))
    {
      echo 'A variável existe';
    }

    if(empty($variavel))
     {
       echo 'A variável está vazia';
     }
?>
```


Inmplode x Explode

```php
<?php
// Um array com 4 chaves (0, 1, 2, 3)
$array = array('Meu', 'Array', 'em', 'PHP');

// Utiliza implode no array dividindo os valores por um espaço
$string = implode(' ', $array);

// Resultado: Meu Array em PHP
echo $string;
?>
```


```php
<?php
// Uma frase
$string = 'Meu Array em PHP';

// Utiliza explode na string, criando um array
$array = explode(' ', $string);

// Exibe a estrutura do array
print_r( $array );

/*
Resultado:

Array
(
    [0] => Meu
    [1] => Array
    [2] => em
    [3] => PHP
)
```


Funções de Manipulação

DATAS

Data e Hora

date_default_timezone_set - Configura a área local.

```
date_default_timezone_set("America/Sao_Paulo");
```

checkdate - Diz se realmente uma data existiu.

```
echo checkdate(02, 31, 2014); // false
```

date - Formata a data ou hora atuais.

```
date("d/m/Y H:i:s A"); // 17/03/2014 12:00 PM
```

getdate - Retorna informações sobre data/hora.

```
$hoje = getdate( );
print_r ($hoje);
```

gettimeofday - Pega a hora atual.

```
print_r (gettimeofday( ));
```

localtime - Pega a hora atual.

```
print_r (localtime(time(), true));
```


Funções de Manipulação

ARRAYS

Array

var_dump - Exibe um array formatado.

```
var_dump($v);
```

print_r - Exibe um array formatado.

```
print_r($v);
```

array_push - Adiciona um item ao array.

```
array_push($v, 3);
```

array_pop - Remove o último elemento de um array.

```
array_pop($v);
```

array_shift - Remove um elemento no início de um array.

```
array_shift($v);
```

array_unshift - Adiciona elemento no início de um array.

```
array_unshift($v, 9);
```


array_pad - Preenche um vetor com valores repetidos.

```
array_pad($v, 5, "X");
```

array_reverse - Cria um vetor, com o inverso do primeiro.

```
$v2 = array_reverse($v, false);
```

array_merge - Une dois arrays em um terceiro.

```
$v3 = array_merge($v1, $v2);
```

count - Retorna a quantidade de elementos de um array.

```
count($v);
```


ORDENAÇÃO DE VETORES

asort

Ordena um array pelos valores em ordem crescente, os índices são mantidos.

```php
$frutas = array("d" => "limao", "a" => "laranja", "b" => "banana", "c" => "melancia");

asort($frutas);
// Resultado: Array ( [b] => banana [a] => laranja [d] => limao [c] => melancia )
```

arsort

Ordena um array pelos valores em ordem decrescente, os índices são mantidos.

```php
$frutas = array("d" => "limao", "a" => "laranja", "b" => "banana", "c" => "melancia");

arsort($frutas);
// Resultado: Array ( [c] => melancia [d] => limao [a] => laranja [b] => banana )
```

ksort

Ordena um array pelas chaves em ordem crescente.

```php
$frutas = array("d" => "limao", "a" => "laranja", "b" => "banana", "c" => "melancia");

ksort($frutas);
// Resultado: Array ( [a] => laranja [b] => banana [c] => melancia [d] => limao )
```

krsort

Ordena um array pelas chaves em ordem decrescente.

```php
$frutas = array("d" => "limao", "a" => "laranja", "b" => "banana", "c" => "melancia");

krsort($frutas);
// Resultado: Array ( [d] => limao [c] => melancia [b] => banana [a] => laranja )
```


sort

Semelhante a função `asort`, porém essa função ordena um array mas zera as chaves.

```php
$frutas = array("d" => "limao", "a" => "laranja", "b" => "banana", "c" => "melancia");

sort($frutas);
// Resultado: Array ( [0] => banana [1] => laranja [2] => limao [3] => melancia )
```

rsort

Semelhante a função `sort`, porém essa função ordena em ordem decrescente zerando as chaves.

```php
$frutas = array("d" => "limao", "a" => "laranja", "b" => "banana", "c" => "melancia");

rsort($frutas);
// Resultado: Array ( [0] => melancia [1] => limao [2] => laranja [3] => banana )
```


FUNÇÃO EMAIL

```
$to = "email@gmail.com";
$dados .= "Nome: $nome";
$dados .= "Assunto: $assunto";
$header = "From: $mail";
$msg = $menssagem;

mail($to;$dados;$msg;$header);
```


FUNÇÃO RAND

```php
<!--?php
    $numero = 20;
    $gera = rand(1,100);

    if($gera == $numero)
        echo "Você venceu com o número: ".$numero;
    else
        echo "Você perdeu, tente novamente.";
?-->
```


Comandos para impressão no PHP

print

Usado apenas para impressão de String. Exemplo:

```
<?php
print('abc'); // abc
?>
```

Há também o comando *var_dump* (usado para imprimir a especificação da variável) e o *print_r* que nada mais é do que listar o conteúdo da variável ou array de forma também explicativa, porém mais legível para o desenvolvedor. Eu costumo usar muito o *print_r* para identificar os dados de um array antes de trabalhar com eles.

Exemplo de uso do **var_dump**:

```
<?php
$vetor = array('pera', 'uva', 'maca', 'salada mista');
print_r($vetor);
?>
```

O resultado será o seguinte:

```
array(4) {
  [0] => string(4) "pera"
  [1] => string(2) "uva"
  [2] => string(4) "maca"
  [3] => string(12) "salada mista"
}
```

Exemplo da impressão com *print_r($variável)*:

```
array {
  [0] => "pera"
  [1] => "uva"
  [2] => "maca"
  [3] => "salada mista"
}
```