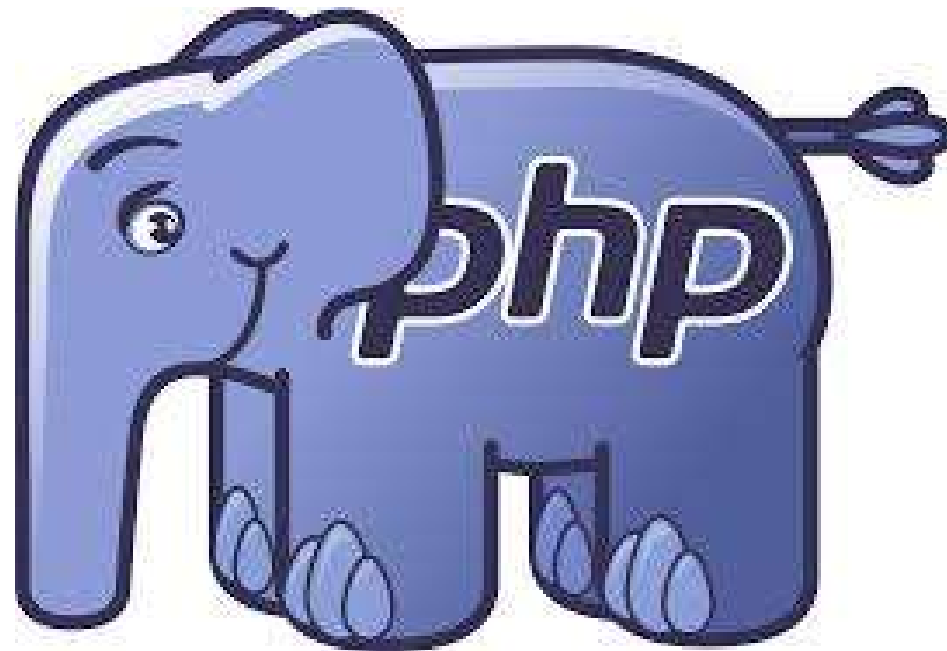




AULA 3

@explicadoresnet



Rotinas



Funções

FORMAL

```
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; 4  
}
```

// programa principal

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

\$A	\$x
3	3

4



Passagem de parâmetro

Por referência

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

REFERÊNCIA

```
// programa principal
```

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

\$x
\$A
34



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); NADA (1, 5, 7)

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)
{
    ...
}
```



\$valor
5 10

\$valor	\$valor
5	10

```
<?php
function dobro ($valor)
{
    $valor = 2 * $valor;
    Echo $valor; 10
}

function duplica (&$valor)
{
    $valor = 2 * $valor;
    Return $valor;
}

$valor = 5;
dobro ($valor);
echo $valor . "<br>"; 5
duplica ($valor);
echo $valor;
10
?>
@explicadoresnet
```



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);
?>
```



\$A
\$x
3

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

Handwritten notes: Red box around 1, red checkmark above ++\$i, red numbers 1 3 5 below echo \$i, red arrow pointing to \$i++.

Handwritten diagram: A box labeled \$i with a red checkmark and a red arrow pointing to it.

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

Handwritten notes: Red checkmark above \$FAB= 20;, red underline under \$b=5;, red numbers 1 5 20 below the code.

Handwritten diagram: A box labeled \$B+ with a red checkmark and a red arrow pointing to it. Below it, a box labeled \$FAB with a red checkmark and a red arrow pointing to it. Below that, a box labeled 5 20.

Handwritten diagram: A box labeled \$B with a red checkmark and a red arrow pointing to it. Below it, a box labeled 5.

Handwritten diagram: A box labeled \$FAB with a red checkmark and a red arrow pointing to it. Below it, a box labeled 20. To the right, a box labeled \$B with a red checkmark and a red arrow pointing to it. Below it, a box labeled 20 5.





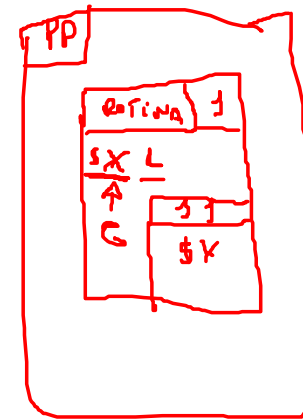
```
<?php
function incrementa(&$x){
    $x++;
    global $a;
    echo "valor de x: $x <br>";

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

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



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



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.

for ()
do

Require: APENAS }

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

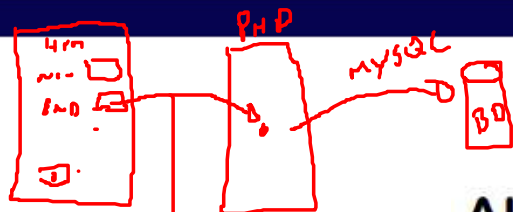
IP do visitante

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

SO do servidor

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





GET
POST
MÉTODOS
DE
ENVIÓ

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()

www.oGlobo.com.br?nome=Alex



Funções de Manipulação

STRINGS



Caractere

SA = "e" + P(i = 4) "ORES" -

✓ **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  
sen  
!!!
```



str_pad - Preenche string com espaços ou caracteres.

```
Echo str_pad ("explicadores", 15, "#"); // explicadores###
```

str_repeat - Repete uma string.

```
Echo str_repeat ("Edu", 3); // EduEduEdu
```

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", "c"); // 5
```

ex:

STR_PAD_RIGHT

STR_PAD_LEFT

explicadores

STR_PAD_BOTH

explicadores



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
```

↑

$\$A = \text{" "};$

$\$A = \text{" "};$



```

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

echo substr($texto, 0, 16); — eu não sou besta
echo "<br>\n";
echo substr($texto, 11); — besta pra tirar onda de herói;
echo "<br>\n";
echo substr($texto, 11, 9); — besta pra
echo "<br>\n";
echo substr($texto, -5); — herói
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";
32 $posicao = strpos($texto, 'http://');
echo substr($texto, $posicao+11);
?>
```

Handwritten annotations:

- A red box highlights the string "http://www.pablo.blog.br" in the variable \$texto.
- A red arrow points to the "http://" part of the string, with the number "32" written above it.
- The number "43" is written below the expression \$posicao+11.



Funções de Manipulação

NÚMEROS



Números

✗ **abs** - valor absoluto de um número

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

base_convert (111, 2, 10)

✗ **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
```

ARREDONDAMENTO



pow - Eleva um valor ao outro (exponenciação).

echo pow(3, 2); // 9

$$3^2 = 9$$

sqrt - Calcula a raiz quadrada.

echo sqrt(9); // 3



Funções de Manipulação

VARIÁVEIS



Variáveis

INTEIRO → INT/INTEGER
 REAL → FLOAT/DOUBLE
 LITERAL → STRING
 LÓGICO → BOOLEANO

\$A = 1.5;
 GETTYPE(\$A);

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);

\$A = 0;
 \$B = floatval(\$A);

\$C = strval(\$A);

\$C = "0";

ISSET(\$A); FALSE
 TRUE

\$A = 0;
 UNSET(\$A);
 ISSET(\$A); FALSE



```
<?php
2   $variavel = 'Diferença';
3
4   if(isset($variavel))
5   {
6       echo 'A variável existe';
7   }
8
9   if(empty($variavel))
10  {
11      x echo 'A variável está vazia';
12  }
13  ?>
```



TRANSFORMA
ARRAY
EM STRING

implode x Explode

TRANSFORMA
UMA STRING
EM ARRAY



STRING
MEU ARRAY em 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
// 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
)

```

Imprimindo
um
array



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.

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

arsort

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

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

ksort

Ordena um array pelas chaves em ordem crescente.

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

krsort

Ordena um array pelas chaves em ordem decrescente.

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

```
$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.

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



FUNÇÃO RAND

```
1 <!--?php
2     $numero = 20;
3     $gera = rand(1,100);
4
5     if($gera == $numero)
6         echo "Você venceu com o número: ".$numero;
7     else
8         echo "Você perdeu, tente novamente.";
9 ?-->
```



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"
}
```

