Primeiros passos

Requisição de arquivos

include <arquivo>

A instrução include() inclui e avalia o arquivo informado. Seu código (variáveis, objetos e arrays) entra no escopo do progrma, tornando-se disponível a partir da linha em que a inclusão ocorre. Se o arquivo não existir, produzirá uma mensagem de advertência(warning).

Requisição de arquivos

include <arquivo>

Ex: tools.php

```
<?php
function quadrado($numero) {
    return $numero * $numero;
}
```

teste.php

```
<?php
include 'tools.php'; //carrega arquivo com a função necessária
echo quadrado(4); //imprime o quadrado do número 4;
```

Requisição de arquivos

require <arquivo>
Similar ao include. Difere somente na manipulação de erros. Enquanto o include produz uma mensagem de advertência, o require produz um erro fatal caso o arquivo não exista.

include_once <arquivo>

Funciona da mesma maneira que o comando include, a não ser que o arquivo informado já tenha sido incluído. Neste caso, a operação não é refeita (o arquivo é incluído apenas uma vez). É útil em casos em que o programa passa mais de uma vez pela mesma instrução. Assim evitará sobreposições, redeclarações etc.

Requisição de arquivos

require_once <arquivo>

Funciona da mesma maneira que o comando require, a não ser que o arquivo informado já tenha sido incluído. Neste caso, a operação não é refeita (o arquivo é incluído apenas uma vez). É útil em casos em que o programa passa mais de uma vez pela mesma instrução. Assim podem-se evitar sobreposições, redeclarações etc.

Manipulação de funções

```
<?php
        function nome_da_funcao($arg1, $arg2, $argN) {
            $valor = $arg1 + $arg2 + $argN;
            return $valor;
        }
```

A função recebe dois parâmetros ($peso, $altura) e retorna o valor definido pela fórmula.

```
<?php
        function calcula_imc($peso, $altura) {
            return $peso / ($altura * $altura);
        }

        echo calcula_imc(75, 1.85);
```

Variáveis globais

Todas as variáveis declaradas dentro do escopo de uma função são locais. Para acessar uma variável externa ao contexto de uma função sem passá-la como parâmetro, é necessário declará-la como global. Uma variável global é acessada a partir de qualquer ponto da aplicação.

No exemplo a seguir, a função criada converte quilômetros para milhas, enquanto acumula a quantidade de quilômetros percorridos em uma variável global ($total).

Variáveis globais

```php
<?php
    $total = 0;
    function km2mi($quilometros) {
        global $total;
        $total += $quilometros;
        return $quilometros * 0.6;
    }

    echo 'percorreu ' . km2mi(100) . " milhas <br>\n";
    echo 'percorreu ' . km2mi(200) . " milhas <br>\n";
    echo 'percorreu no total ' . $total . " quilometros <br>\n";
```

Obs: a utilização de variáveis globais **não** é considerada uma boa prática de programação, pois uma variável global pode ser alterada a partir de qualquer parte da aplicação.

Variáveis estáticas

Dentro do escopo de uma função, podemos armazenar variáveis de forma estática. Assim elas mantêm o valor que lhes foi atribuído na última execução. Declaramos uma variável estática com o operador static.

```
<?php function percorre($quilometros) {
        static $total;
        $total += $quilometros;
        echo "percorreu mais de $quilometros do total de $total<br>\n";
    }

    percorre(100);
    percorre(200);
    percorre(50);
```

Resultado:

percorreu mais 100 do total de 100
percorreu mais 200 do total de 300
percorreu mais 50 do total de 350

Manipulação de strings

Para declarar uma string podemos utilizar aspas simples ' ' ou aspas duplas " ".

```
$variavel = 'Isto é um teste';
$variavel = "Isto é um teste";
```

A diferença é que **todo conteúdo dentro de aspas duplas** é avaliado pelo PHP.
Assim, se a string contém uma variável, essa variável será traduzida pelo seu valor.

Manipulação de strings

A diferença é que **todo conteúdo dentro de aspas duplas** é avaliado pelo PHP.
Assim, se a string contém uma variável, essa variável será traduzida pelo seu valor.

```php
<?php
        $fruta = 'maçã';
        print "como $fruta<br>" . PHP_EOL; //resultado: como maçã
        print 'como $fruta<br>' . PHP_EOL; //resultado: como $fruta
```

Concatenação

A diferença é que **todo conteúdo dentro de aspas duplas** é avaliado pelo PHP.
Assim, se a string contém uma variável, essa variável será traduzida pelo seu valor.

```
<?php
        $fruta = 'maçã';
        print "como $fruta<br>" . PHP_EOL; //resultado: como maçã
        print 'como $fruta<br>' . PHP_EOL; //resultado: como $fruta

      //resultados: maça é a fruta de adão.
```

Concatenação

O PHP realiza automaticamente a conversão entre tipos. como neste exemplo de concatenação entre uma string e um número:

```php
<?php
        $a = 1234;
        echo 'O salário é ' . $a . '<br>' . PHP_EOL;
        echo "O salário é $a" . '<br>' . PHP_EOL;

        //resultados:
        O salário é 1234
        O salário é 1234
```

Caracteres de escape

Dentro de aspas duplas "" podemos utilizar caracteres de escape (****) - controles especiais interpretados diferentemente pelo PHP.
Veja a seguir os mais utilizados:

\n - Nova linha (proporciona uma quebra de linha)
**** - Barra invertida "\\" (o mesmo que '\').
\" - Aspas duplas. '
\$ - Símbolo de $.

Caracteres de escape

```php
<?php
        echo "seu nome é \"Paulo\" . <br>";
        echo 'seu nome é "Paulo" . <br>';
        echo 'seu salário é $650,00<br>';
        echo "seu salário é \$650,00<br>";
```

OBS: utilize aspas duplas para declarar strings somente quando for necessário. Avaliar seu conteúdo, evitando, assim, tempo de processamento desnecessário.

Funções para manipulação de strings

strtoupper

Retorna a string com todos os caracteres alfabéticos convertidos para letras maiúsculas
string strtoupper(string $string)

strtoupper

Retorna a string com todos os caracteres alfabéticos convertidos para letras maiúsculas.
string strtoupper(string $string)

Exemplo:

```
<?php
    echo strtoupper('Convertendo para maiusculo') . '<br>';
    echo strtolower('CONVERTENDO PARA MINUSCULO') . '<br>';
```

Funções para manipulação de strings

substr

Retorna um a parte de uma string. O primeiro parâmetro representa a string original, o segundo representa o início do corte, e o terceiro, o tamanho do corte. Se o comprimento (length) for negativo, conta n caracteres antes do final.

```
string substr(string $string, int $start, int $length)
```

Exemplo:

```
<?php
    print substr("Americana", 1) . '<br>'; //mericana
    print substr("Americana", 1, 3) . '<br>'; //mer
    print substr("Americana", 0, -1) . '<br>'; //American
    print substr("Americana", -2) . '<br>'; //na
```

Funções para manipulação de strings

str_repeat

Repete uma string certa quantidade de vezes.

```
string str_repeat(string $input, int $multiplier)
```

Exemplo:

```
<?php
    $txt = ".oO0Oo.";
    print str_repeat($txt, 5) . "\n";

    result: .oO0Oo..oO0Oo..oO0Oo..oO0Oo..oO0Oo.
```

Funções para manipulação de strings

strlen

Retorna o comprimento de uma string.

```
int strlen(string $string)
```

Exemplo:

```
<?php
    $txt = "O Rato roeu a roupa do rei de roma";
    print 'O comprimento é: ' . strlen($txt) . "\n";

    result: O comprimento é: 34
```

Funções para manipulação de strings

str_replace

Substitui uma string (primeiro parâmetro) por outra (segundo parâmetro) dentro de um dado contexto(terceiro parâmetro).

```
str_replace ( mixed $search , mixed $replace ,
mixed $subject [, int &$count ] ) : mixed
```

Funções para manipulação de strings

str_replace

Exemplo:

```
<?php
    print str_replace('Rato', 'Leão', 'O Rato roeu a roupa do rei de roma')
        //result: O Leão Rato roeu a roupa do rei de roma

    print str_replace('Rato', 'Leão', 'O Rato roeu a roupa do rei de roma', $i)
        //$i = 1

    print str_replace('Rato', 'Leão', 'O Rato roeu a roupa do Rato de roma', $i)
        //$i = 2
```

Manipulação de arrays

Para criar um array, pode-se utilizar a função array(chave => valor, ...) ou a sintaxe resumida entre [].

Exemplo:

```php
<?php
    $cores = array(0 => 'vermelho', 1 => 'azul', 2 => 'amarelo');
    ou
    $cores = array('vermelho', 'azul', 'verde', 'amarelo');
    ou
    $cores = ['vermelho', 'azul', 'verde', 'amarelo'];

    outra forma:
        $nomes[] = 'maria';
        $nomes[] = 'joão';
        $nomes[] = 'carlos';
        $nomes[] = 'josé';
```

Manipulação de arrays

Para acessar o array, basta indicar o seu índice entre colchetes:

Exemplo:

```
<?php
    echo $cores[0]; //result = vermelho
    echo $cores[1]; //result = azul
    echo $cores[2]; //result = verde
    echo $cores[3]; //result = amarelo

    echo $nomes[0]; //result = maria
    echo $nomes[1]; //result = joão
    echo $nomes[2]; //result = carlos
    echo $nomes[3]; //result = josé
```

Manipulação de arrays

Arrays associativos

Para criar um array, pode-se utilizar a função array([chave =>] valor, ...).

Exemplo:

```
<?php
    $cores = array('vermelho' => 'FF0000', 'azul' => '0000FF', 'verde' => '00FF00');
```

Outra forma de criar um array associativo é simplemeste adicionando-lhe valores com a seguinte sintaxe:

```
$pessoa = array();
$pessoa['nome'] = 'Maria da Silva';
$pessoa['rua'] = 'São João';
$pessoa['bairro'] = 'Cidade Alta';
$pessoa['cidade'] = 'Porto Alegre';
```

Manipulação de arrays

Arrays associativos

Para acessar o array, basta indicar a sua chave entre colchetes:

Exemplo:

```
<?php
    echo $cores['vermelho']; //result: FF0000
    echo $cores['azul']; //result: 0000FF
    echo $cores['verde']; //result: 00FF00

    echo $pessoa['nome’]; //result: Maria da Silva
    echo $pessoa['rua’]; //result: São João
    echo $pessoa['bairro’]; //result: CIdade Alta
    echo $pessoa['cidade’]; //result: Porto Alegre
```

obs: a chave pode ser string ou interger não negativo, o valor pode ser de qualquer tipo.

Manipulação de arrays

Iterações

Os arrays podem ser iterados no PHP pelo operador **FOREACH,** que percorre casa uma das posiçoes do array. Exemplo:

```
$frutas = array();
        $frutas['cor'] = 'vermelha';
        $frutas['formato'] = 'redondo';
        $frutas['nome'] = 'maça';
        foreach($frutas as chave => $fruta) {
            echo "$chave => $fruta <br>\n";
        }
```

result:

cor => vermelha
sabor => doce
formato => redondo
nome => maçã

Manipulação de arrays

Acessando arrays

As posições de um array podem ser acessadas a qualquer momento, e sobre elas operações pode ser realizadas.

```
$contato = array();
$contato['nome'] = 'Pablo';
$contato['empresa'] = 'RD';
$contato['peso'] = 73;
//alterações
$contato['nome'] .= 'Nogueira';
$contato['empresa'] .= ' Raiadrogasil';
$contato['peso'] += 2;
//debug
print_r($contato);
```

```
$comidas = array();
$comidas[] = 'Lasanha';
$comidas[] = 'Pizza';
$comidas[] = 'Macarrão';
//alterações
$comidas[1] = 'Pizza Calabresa';
/debug
print_r($comidas);
```

Manipulação de arrays

Arrays multidimensionais

Arrays multidimensionais ou matrizes são arrays nos quais algumas de suas posições podem conter outros arrays de forma recursiva. um array multidimensional pode ser criado pela função array();

```
$carros = array('Palio' => array('cor' => 'azul',
                                 'potência' => '1.0',
                                 'opcionais' => 'Ar Cond.'),
                'Corsa' => array('cor' => 'Cinza',
                                 'potência' => '1.3',
                                 'opcionais' => 'MP3'),
                'Gol'   => array('cor' => 'branco',
                                 'potência' => '1.0',
                                 'opcionais' => 'Metalixa')
echo $carros['Palio']['opcionais']; //result: Ar Cond.
```

Manipulação de arrays

Arrays multidimensionais

Outra forma de criar um array multidimensional é simplesmente atribuindo-lhe valores:

```
$carros = array();
$carros['Palio']['Cor']       =       'azul';
$carros['Palio']['potência']  =  '1.0';
$carros['Palio']['opcionais'] = 'Ar Cond.';
$carros['Corsa']['cor']       =       'cinza';
$carros['Corsa']['potência']  = '1.3';
$carros['Corsa']['opcionais'] = 'MP3';
$carros['Gol']['cor']         =       'branco';
$carros['Gol']['potência']    = '1.0';
 $carros['Gol']['opcionais']   = 'Metalica';

echo $carros['Palio']['opcionais']; //resultado = Ar Cond.
```

Manipulação de arrays

Arrays multidimensionais

Para realizar iterações em um array multidimensional, é preciso observar quantos níveis ele tem.

```
foreach ($carros as $modelo => $caracteristicas) {
        echo "=> modelo $modelo<br>\n";
        foreach ($caracteristicas as $caracteristica => $valor) {
            echo " - característica $caracteristica => $valor<br>\n";
        }
}
```

Manipulação de arrays

Funções para manipulação de arrays

array_unshift

Adiciona elemento(s) ao início de um array
int array_unshift(array $array, mixed $var)

array_push

Adiciona elemento(s) ao final de um array. Tem o mesmo efeito que $array[] = $valor.
int array_push(array $array, mixed $var)

Manipulação de arrays

Funções para manipulação de arrays

array_unshift - array_push

Exemplo:

```
$a = array("verde", "azul", "vermelho");
array_unshift($a, "ciano");
array_push($a, "amarelo");
print_r($a);
```

Manipulação de arrays

Funções para manipulação de arrays

array_shift

Remove um elemento do início de um array.
mixed array_shift(array $array)

array_pop

Remove um valor do final de um array
mixed array_pop(array $array)

Manipulação de arrays

Funções para manipulação de arrays

array_shift - array_pop

Exemplo:

```
$a = array("ciano", "verde", "azul", "vermelho", "amarelo");
array_shift($a);
array_pop($a);
print_r($a);
```

Manipulação de arrays

Funções para manipulação de arrays

Array_reverse

Reverse um array e retorna-o na ordem inversa.
array array_reverse(array $array, bool $preserve_keys)

Exemplo:

```
$a[0] = 'verde';
$a[1] = 'amarelo';
$a[2] = 'vermelho';
$a[3] = 'azul';
$b = array_reverse($a, true);
print_r($b);
```

Manipulação de arrays

Funções para manipulação de arrays

array_merge

Mescla dois ou mais arrays. Um array é adicionado ao final do outro. O resultado é um novo array. Se ambos os arrays tiverem conteúdo indexado pela mesma chave, o segundo irá se sobrepor ao primeiro.

array array_merge(array nome_array1, array nome_array2, array...)

Exemplo:

```
$a = array("verde", "azul");
$b = array("vermelho", "amarelo");
$c = array_merge($a, $b);]
print_r($c);
```

Manipulação de arrays

Funções para manipulação de arrays

array_keys

Retorna as chaves (índices) de um array. Se o segundo parâmetro for indicado, a função retornará apenas índices que apontam para um conteúdo igual ao parâmetro.

array array_keys(array $input)

array_values

Retorna somente os valores de um array, ignorando suas chaves.

array array_values(array $input)

count

Retorna a quantidade de elementos de um array.

int count(array nome_array)

Manipulação de arrays

Funções para manipulação de arrays

array_keys – array_values - count

Exemplo:

```
$exemplo = array('cor' => 'vermelho', 'volume' => 5, 'animal' => 'cachorro');
print_r(array_keys($exemplo));
print_r(array_values($exemplo));
print 'Quantidade: ' . count($exemplo);
```

Manipulação de arrays

Funções para manipulação de arrays

in_array

Verifica se um array contém um determinado valor.
bool in_array(mixed $agulha, array $palheiro)

Exemplo:

```
$a = array('refrigerante', 'cerveja', 'vodca', 'suco natural');
if (in_array('suco natural', $a)) {
        echo 'suco natural encontrado';
}
```

Manipulação de arrays

Funções para manipulação de arrays

sort

Ordena um array pelo seu valor, sem manter a associação de índices.
Para ordem reversa, utilize rsort().

```
sort(array $array)
```

Exemplo:

```
$a = array('refrigerante', 'cerveja', 'vodca', 'suco natural');
sort($a);
print_r($a);
```

Manipulação de arrays

Funções para manipulação de arrays

asort

Ordena um array pelo seu valor, mantendo a associação de índices. Para ordernar de forma reversa, use arsort().

```
asort(array $array)
```

Exemplo:

```
$a[0] = 'verde';
$a[1] = 'amarelo';
$a[2] = 'vermelho';
$a[3] = 'azul';
asort($a);
print_r($a);
```

Manipulação de arrays

Funções para manipulação de arrays

ksort

Ordena um array pelos seus índices. Para ordem reversa, utilize krdort().

```
ksort(array $array);
```

Exemplo:

```
$carro['potência']  = '1.0';
$carro['cor']  = 'branco';
$carro['modelo']  = 'celta';
$carro['opcionais'] = 'ar quente';
ksort($carro);
print_r($carro);
```

Manipulação de arrays

Funções para manipulação de arrays

explode

Converte uma string em um array, quebrando os elementos por meio de um separador(delimiter).

array explode(string $delimiter, string $string)

implode

Converte um array em uma string, agrupando os elementos do array por meio de um elemento cola(glue).

string implode(string $glue, array $pieces)

Manipulação de arrays

Funções para manipulação de arrays

Explode - implode

Exemplo:

```
$string = "10/05/2015";
print_r(explode("/", $string));
$array = [10, 05, 2015];
print implode('/', $array);
```