ULBRA

SISTEMAS DE INFORMAÇÃO

LINGUAGEM DE PROGRAMAÇÃO WEB

Prof. Taciano Balardin
taciano@ulbra.edu.br

E-MAIL:
taciano@ulbra.edu.br

SITE DA DISCIPLINA:
http://www.taciano.pro.br/

SENHA:
@lpweb

Introdução ao PHP

Estruturas de Controle

LINGUAGEM DE PROGRAMAÇÃO WEB
AULA 06

Prof. Taciano Balardin

Delimitadores de Código

```
<?php
    //código ;
    //código ;
?>
```

ou com o **short_open_tag** ativo no **php.ini**:

```
<?
    //código ;
    //código ;
    //código ;
?>
```

Todos os comandos são delimitados por **;**

Comentários

- Para comentar uma única linha de código:

```
// echo “a” ;
# echo “a” ;
```

- Para comentar várias linhas de código

```
/*  echo “a” ;
    echo “b” ;  */
```

Comandos de Saída (output)

- **echo** → imprime variáveis, caracteres ou strings:

```
echo 'a', 'b', 'c';  echo 'ABC';
```

- **print** → imprime variáveis, caracteres ou strings:

```
print('Frase impressa com o comando print');
```

- **var_dump** → imprime o conteúdo de uma variável, normalmente usado para debug:

```
$vetor = array('Palio','Gol','Fiesta','Corsa');
var_dump($vetor);
```

- **print_r** → imprime o conteúdo de uma variável, normalmente usado para debug:

```
print_r($vetor);
```

echo

```php
<?
  // Minhas primeiras linhas em php!
  $nome = 'Taciano';
  echo '<h1>Hello World!</h1>';
  echo "Meu nome é: {$nome}";
?>
```

print

```php
<?
  // Minhas primeiras linhas em php!
  print('<h1>Hello Word!</h1>');
?>
```

var_dump

```php
<?
  //Imprimindo um vetor usando var_dump
  $vetor = array('Palio','Gol','Corsa');
  var_dump($vetor);
?>
```

print_r

```
<?
  //Imprimindo um vetor usando print_r
  $vetor = array('Palio','Gol','Corsa');
  print_r($vetor);
?>
```

Variáveis

- Identificadores utilizados para valores mutáveis e voláteis;
- Sempre iniciam com **$**;
- Não precisa ser declarado o tipo :-)

```
<?
  // Define o nome da variável
  $cidade = 'Cachoeira do Sul';
  echo "Eu moro em {$cidade}.";
?>
```

Regras e boas práticas

- Nunca inicie o nome de variáveis com número;
- Nunca utilize espaços no meio do identificador da variável;
- Nunca utilize caracteres especiais:

  ! @ # $ % ^ & * / [ ] { };

- PHP é case-sensitive, ou seja, distingue maiúsculas de minúsculas;
- Nomes das variáveis devem ser significativos e transmitir a idéia de seu conteúdo;
- Utilize preferencialmente palavras em minúsculo (separadas por _).

São variáveis válidas?

$5cliente

$_123

$computador

$#estabilizador

$cod_Cliente

$_cliente

$computador_sem_estabilizador_de_rede

$idPessoa

Comandos condicionais

Comandos de repetição

ESTRUTURAS DE CONTROLE

Comandos condicionais

if

switch

if

- Avalia uma expressão e dependendo do resultado é executado um conjunto diferente de instruções:

```
<?
    if(exp1)  {
      bloco1
    }
    elseif(exp2)  {
      bloco2
    }
    else  {
      bloco3
    }
?>
```

if

- Somente um dos blocos será executado e após a execução continuará depois dos comandos;
- Pode aparecer diversos **elseif**;
- Caso o bloco só tenha uma linha as chaves **{ }** são dispensáveis;
- Não é obrigatório o uso do **elseif** ou **else**. O **if** isoladamente também pode ser usado.

Exercício if

- Criar um algoritmo que atribua valor a duas variáveis (g1 e g2) e calcule a média ponderada do aluno. No final, o algoritmo deve apresentar as notas, a média final e o texto “Aprovado” ou “Reprovado”, considerando as médias da ULBRA.

Nota G1: 6

Nota G2: 7

Média Final: 6.66

Aluno aprovado!

switch

- Avalia o valor de uma expressão para escolher o que vai ser executado:

```
<?
    switch(operador) {
        case valor1:
                <comandos>
                break;
        case valor2:
                <comandos>
                break;
        case valorN:
                <comandos>
                break;
        default:
                <comandos>
                break;
    }
?>
```

Depois de cada bloco de comandos, deve ser usado o comando **break** para que o **switch** seja encerrado. Caso não seja usado o PHP continuará executando o **switch**.

if x switch

```php
<?
    $numero = 2;
    if($numero==0) {
        echo "O n° é 0";
    }
    elseif($numero==1) {
        echo "O n° é 1";
    }
    elseif($numero==2) {
        echo "O n° é 2";
    }
?>
```

```php
<?
    $numero = 2;
    switch($numero) {
        case 0:
            echo "O n° é 0";
            break;
        case 1:
            echo "O n° é 1";
            break;
        case 2:
            echo "O n° é 2";
            break;
    }
?>
```

Comandos de repetição

while

do ... while

for

foreach

while

- Traduzido para o português significa **enquanto**;
- O comando avalia a expressão, e **enquanto** essa expressão retornar o valor **verdadeiro**, a execução do conjunto de comandos **será repetida**. Caso seja **falsa** o bloco **encerra** a execução do bloco;
- Tomar cuidado para não criar expressões que **nunca se tornam falsas** pois teríamos um **loop** infinito.

while

- Sintaxe:

```
<?
   while(exp)
   {
      <comandos>
   }
?>
```

Exercício while

- Crie um algoritmo que armazene um número inteiro aleatório entre 1 e 20 em uma variável **$teste** e imprima todos os números anteriores até ele, informando se são **par** ou **ímpar**.

Ex:

$teste = 3;

Número **1** é ímpar;

Número **2** é par;

Número **3** é ímpar;

do ... while

- A única diferença entre o **while** e o **do** ... **while** é que o **while** avalia a expressão no **início** do laço e o **do** ... **while** ao **final**.
- Sintaxe:

```
<?
    do
    {
        <comandos>
    } while(exp);

?>
```

do ... while

```php
<?
   $cont = 1;
    do
    {
      echo "O valor atual do contador é {$cont} <br/>";
      $cont++;
    } while($cont < 15);
?>
```

for

- Usado quando queremos executar um conjunto de instruções por quantidade especifica de vezes;
- Pode ser usado para imprimir os elementos de um *array* ou todos os resultados de uma consulta no banco de dados.
- Sintaxe:

```
<?
    for(inicialização; condição; operador)
    {
        <comandos>
    }

?>
```

for

- Com inicialização iniciamos o valor inicial da variável que controlará o loop:

```
$cont = 0;
```

- Na condição devemos colocar a condição para que o loop continue a ser executado. Quando a condição retornar um valor falso o loop parará:

```
$cont < 20;
```

- O operador é usado para atualizar o valor da variável de controle, fazendo um incremento ou decremento ao final de cada iteração do loop:

```
$cont++;
```

for

```
<?
   for($cont=0; $cont<20; $cont++)
   {
      echo "O valor de cont é {$cont}<br/>";
   }
?>
```

Exercício for

- Crie um algoritmo que armazene um número aleatório inteiro entre 1 e 20 em uma variável **$teste** e faça uma contagem progressiva e outra regressiva até aquele número.

$teste = 3;

Contagem Progressiva:

1

2

3

Contagem Regressiva:

3

2

1

foreach

- Oferece uma maneira mais fácil de “navegar” entre os elementos de um *array*:

```
foreach($nome_array as $elemento)
{
    <comandos>
}
```

- Todos os itens de **$nome_array** serão visitados. A cada iteração o item da vez será armazenado em **$elemento**. Assim é possível trabalhar todos os elementos usando somente uma variável

foreach

- Essa segunda sintaxe funciona da mesma forma, porém enquanto o elemento é adicionado a **$valor**, o índice atual é atribuído a **$chave**

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

Exercício foreach

- Crie um *array* com números ou strings e imprima primeiramente seus valores e depois suas chaves e valores.

Ex:

```
$teste = array('Palio','Gol','Corsa');
```

Palio

Gol

Corsa

0 => Palio

1 => Gol

2 => Corsa