
*Primeiros passos*

# Um programa PHP

## Estrutura do código fonte

normalmente um programa php tem a extenção **.php**

O código de um programa escrito em PHP deve iniciar com **<?php**, veja:

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

obs: a finalização da maioria dos comandos se dá por ponto e vírgula (;)

# Configuração

para verificar as Configurações do PHP utilizamoas a função **phpinfo().**

```
<?php
    phpinfo();
```

O arquivo de Configuração do PHP é o php.ini

# Comentários

**para comentar uma única linha:**

```
// echo "a";
# echo "a";
```

**para comentar muitas linhas:**

```
/* echo "a";
   echo "b"; */
```

# Comandos de saída(output)

São comandos para gerar uma saída em uma tela.

# Comandos de saída - echo

É um comando que imprime uma ou mais variáveis.
Exemplo:

```
echo 'a' . '<br>' . PHP_EOL;
echo 'b' . '<br>' . PHP_EOL;
```

obs: PHP_EOL representa uma quebra de linha (end of line)

# Comandos de saída - print

É um comando que imprime uma string.
Exemplo:

```
print 'abc';
```

# Comandos de saída - var_dump

É uma função que imprime o conteúdo de uma variável de forma detalhada, muito comum para se realizar debug.

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

# Comandos de saída - print_r

Imprime de forma detalhada como o var_dump, mas em formato mais legível, mais alinhados e sem tipos de dados

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

# Variáveis

Para criar uma variável em PHP, precisamos atribuir-lhe um nome de identificação, sempre precedido pelo caractere cifrão (**$**).

```
<?php
    $nome = "João";
    $sobrenome = "da Silva";
    echo "$sobrenome, $nome";
```

# Variáveis - Algumas dicas:

- Nunca inicio a nomenclatura de variáveis com números.

- Nunca utilize espaços em brancos no meio do identificador da variável.

- Nunca utiliza caracteres especiais **(! @ # % ^ & * / [] {})** na nomenclatura das variáveis.

# Variáveis - Algumas dicas:

- Evite criar variáveis com mais de 15 caracteres em virtude da clareza do código-fonte.

- Nomes de variáveis devem ser significativos e transmitir a ideia de seu conteúdo dentro do contexto no qual a variável está inserida.

# Variáveis - Algumas dicas:

- Utilize preferencialmente palavras em letras minúsculas separadas por **"_"** ou somente as **primeiras letras em maiúsculas** quando da ocorrência de mais palavras.

```
<?php
   $codigo_cliente
   $codigoCliente
```

# Variáveis - Algumas dicas:

Com **exceção** de nomes de **classes e funções**, o PHP é case sensitive, ou seja, é sensível a letras maiúsculas e minúsculas.

<?php
**$codigo** é diferente de **$Codigo;**

# Variáveis - Algumas dicas:

Com **exceção** de nomes de **classes e funções**, o PHP é case sensitive, ou seja, é sensível a letras maiúsculas e minúsculas.

<?php
**$codigo** é diferente de **$Codigo;**

# Variáveis - Tipo booleano

Um booleano expressa um avlaor lógico que pode ser verdadeiro ou falso. Para especificar um valor booleano, utilize as palavras-chave TRUE ou FALSE.

```
<?php
    $exibir_nome = TRUE // declara a variável com o valor TRUE
    //testa se $exibir_nome é TRUE
    if ($exibir_nome) {
        echo 'José da Silva';
     }
```

# Variáveis - Tipo booleano

No programa que segue criamos uma variável numérica contendo o valor 91. Em seguida testamos se a variável é maior que 90. Tal comparação também retorna um valor booleano.

```
<?php
        $umidade = 91 // declara variável numérica

      //testa se é maior que 90. Retorna um booleano
       $vai_chover = ($umidade > 90);

       //testa se $vai_chover é verdadeiro
       if ($vai_chover) {
           echo 'Vai chover';
       }
```

# Variáveis - Tipo booleano

Também são considerados valores **falsos** em comparações booleanas:

- Inteiro 0.
- Ponto flutuante 0.0.
- Uma **string** vazia "" ou "0".
- Um **array** vazio.
- Um objeto sem elementos.
- Tipo **NULL**.

# Variáveis - Tipo numérico

```php
<?php
   $a = 1234; // número decimal
   $a = -123; // número negativo
   $a = 1.234; // ponto flutuante
```

# Variáveis - Tipo string

Uma string é uma cadeia de caracteres alfanuméricos. Para declará-la, podemos utilizar aspas simples (**' '**) ou aspas duplas (**" "**).

```
<?php
   $variavel = 'Isto é um teste';
   $variavel = "Isto é um teste";
```

# Variáveis - Tipo array

Array é uma lista de valores armazenados na memória que podem ser de tipos diferentes (números, strings, objetos).

```php
<?php
    $carros = array('Palio', 'Corsa', 'Gol');
    echo $carros[1]; // resultado = 'Corsa'
```

# Variáveis - Tipo string

Uma string é uma cadeia de caracteres alfanuméricos. Para declará-la, podemos utilizar aspas simples (**' '**) ou aspas duplas (**" "**).

```
<?php
   $variavel = 'Isto é um teste';
   $variavel = "Isto é um teste";
```

# Variáveis - Tipo objeto

Um objeto é uma entidade com um determinado comportamento definido por seus métodos(ações) e propriedade(dados). Para criar um objeto deve-se utilizar o operador **new**.

```
<?php
  $carro = new stdClass;
  $carro->modelo = 'Palio';
  $carro→ano = 2002;
  $carro->cor = 'Azul';
  print_r($carro);
  echo $carro->modelo . ' ' . $carro->ano . ' ' . $carro->cor;
```

# Variáveis - Tipo NULL

A utilização do valor especial **NULL** significa que a variável não tem valor. **NULL** é o único valor possível do tipo **NULL.**

# Constantes

Uma constante é um valor que não sobre modificações durante a execução do programa. As regras de nomenclatura de constantes seguem as mesmas regras das variáveis, com exceção de que as constantes **não são** precedidas pelo sinal de (**$**) e geralmente utilizam nomes em letras maiúsculas.

Podemos definir uma constante utilizando a função **define()**.

```
<?php
    define('MAXIMO_CLIENTES', 100);
    echo MAXIMO_CLIENTES;
```

# Operadores - Atribuição

```php
<?php
    $var = 100;
    $var += 5; //Soma 5 em $var;
    $var -= 5; //Subtrai 5 em $var;
    $var *= 5; //Multiplica $var por 5;
    $var /= 5; //Divide $var por 5;
    echo $var; //resultado: 100;
```

# Operadores - Atribuição

**++$var** - ***Pré-incremento***. Incrementa **$var** em um e, então, retorna **$var**.

**$var++** - ***Pós-incremento***. Retorna **$var** e, então, incrementa **$var** em um.

**--$var** - ***Pré-decremento***. Decrementa **$var** em um e, então, retorna **$var**.

**$var**-- - ***Pós-decremento***. Retorna **$var** e, então, decrementa **$var** em um.

# Operadores - Atribuição

```php
<?php
    $var = 100;
    echo $var++ . ' '; // retorna 100 e incrementa para 101

    echo ++$var . ' '; // incrementa para 102 e retorna

    echo $var-- . ' '; // retorna 102 e decrementa para 101

    echo --$var . ' '; // decrementa para 100 e retorna
```

# Operadores - Aritméticos

Operadores aritméticos são utilizados para realização de cálculos matemáticos.

**+** - Adição
**-** - Subtração
***** - Multiplicação
***/*** - Divisão
**%** - Módulo (resto da divisão)

# Operadores - Aritméticos

Em cálculos complexos, procure utilizar parênteses, sempre observando as prioridades aritméticas. Por exemplo:

```php
<?php
  $a = 2;
  $b = 4;
  echo $a+3*4+5*$b . ' '; // resultado = 34
  echo ($a+3)*4+(5*$b) . ' '; // resultado = 40
```

# Operadores - Aritméticos

O PHP realiza automaticamente a conversão de tipos em operações:

```php
<?php // declaração de uma string conteudo 10
    $a = '10';
    // soma + 5
    echo $a + 5;
```

# Operadores - Relacionais

Operadores relacionais são utilizados para realizar comparações entre valores ou expressões, resultado sempre um valor booleano (TRUE ou FALSE).

**==** - Igual. Resuta verdadeiro (TRUE) se as expressões forem iguais.
**===** - Idêntico. Resulta verdadeiro (TRUE) se as expressões forem iguais e do mesmo tipo de dados.
**!=** ou **<>** - Diferente. Resulta verdadeiro (TRUE) se as variáveis forem diferentes.
**<** - Menor
**>** - Maior que.
**<=** - Menos ou igual
**>=** - Maior ou igual

# Operadores - Relacionais

exemplos:

uso do errado do =

```
<?php
 if ($a = 5) {
     echo 'essa operação atribui 5 à variável $a';
 }
```

# Operadores - Relacionais

exemplos:

uso do **==** e **!=**

```php
<?php
    $a = 1234;
    $b = '1234';

    if ($a == $b) {
        echo '$a e %b são iguais';
    } elseif ($a != $b) {
        echo '$a e $b são diferentes';
    }
    //$a e $b são iguais
```

# Operadores - Relacionais

exemplos:

uso do **===** e **!==**

```php
<?php
    $c = 1234;
    $d = '1234';

    if ($c == $d) {
        echo '$c e $d são iguais e do mesmo tipo';
    } elseif ($c != $d) {
        echo '$c e $d são de tipos diferentes';
    }
//$c e $d são de tipos diferentes
```

# Operadores - Relacionais

O PHP considera o valor zero como sendo falso em comparações lógicas.

```php
<?php
$e = 0;
  // testa a variável é FALSE
  if ($e == FALSE) {
       echo '$e é falso ';
  }
```

# Operadores - Relacionais

O PHP considera o valor zero como sendo falso em comparações lógicas.

```
<?php
// testa se a variável é um FALSE e do tipo booleano
   if ($e === FALSE) {
        echo '$e é FALSE e do tipo boolean ';
   }
```

# Operadores - Relacionais

O PHP considera o valor zero como sendo falso em comparações lógicas.

```php
<?php
// testa se $e é igual a zero e do mesmo tipo que zero
    if ($e === 0) {
        echo '$e é zero mesmo';
    }

    //RESULTADO: $e é falso $e é zero mesmo
```

# Operadores - Lógicos

Operadores lógicos são utilizados para combinar expressões lógicas entre si, agrupando testes condicionais.

(**$a and $b)** - E: Verdadeiro (TRUE) se tanto $a quanto $b forem verdadeiros.
**($a or $b)** - OU: Verdadeiro (TRUE) se $a ou $b forem verdadeiros.
**($a xor $b)** - XOR: Verdadeiro (TRUE) se $a ou $b forem Variáveis, de forma exclusiva.
**(! $a)** - NOT: Verdadeiro (TRUE) se $a for FALSE.
**($a && $b)** - E: Verdadeiro (TRUE) se tanto $a quanto $b forem verdadeiros.
**($a || $b)** - OU: Verdadeiro (TRUE) se $a ou $b forem verdadeiros.

**obs:** or e and têm precedência **menor** que && ou ||.

# Estrutura de Controle - if

```php
<?php
    $a = 1;
    if ($a ==5) {
        echo "é igual";
    } else {
         echo "não é igual";
    }
```

# Estrutura de Controle - if

Quando não explicitamos o operador lógico em testes por meio do If, o comportamento-padrão do PHP é retornar TRUE sempre que a variável tiver conteúdo válido.

```
<?php
        $a = 'conteudo';
        if ($a) {
            echo '$a tem conteúdo';
        }
        if ($b) {
            echo '$b tem conteúdo';
        }
```

**Obs**: atualmente se o PHP estiver com o nível de erro NOTICE ligado, o teste if ($b) emitirá a mensagem de erro **"Notice: undefined variable: b"**. A forma mais correta para testar se uma variavel está definida é uilizar a função if (**isset($b)**).

# Operadores - Lógicos

Operadores lógicos são utilizados para combinar expressões lógicas entre si, agrupando testes condicionais.

(**$a and $b)** - E: Verdadeiro (TRUE) se tanto $a quanto $b forem verdadeiros.
**($a or $b)** - OU: Verdadeiro (TRUE) se $a ou $b forem verdadeiros.
**($a xor $b)** - XOR: Verdadeiro (TRUE) se $a ou $b forem Variáveis, de forma exclusiva.
**(! $a)** - NOT: Verdadeiro (TRUE) se $a for FALSE.
**($a && $b)** - E: Verdadeiro (TRUE) se tanto $a quanto $b forem verdadeiros.
**($a || $b)** - OU: Verdadeiro (TRUE) se $a ou $b forem verdadeiros.

**obs:** or e and têm precedência **menor** que && ou ||.

# Estrutura de Controle - if

Para realizar testes encadeados, basta colocar um If dentro do outro ou utilizar o operador AND da seguinte forma:

```
<?php
        $salario = 1020;
        $tempo_servico = 12;
        $tem_reclamacoes = false;
        if ($salario > 1000) {
            if ($tempo_servico >= 12) {
                if ($tem_reclamacoes != true) {
                    echo 'parabéns, você foi promovido<br>' . PHP_EOL;
                }
            }
        }
```

# Estrutura de Controle - if

Para realizar testes encadeados, basta colocar um If dentro do outro ou utilizar o operador AND da seguinte forma:

```
<?php
        if ($salario > 1000) and ($tempo_servico >= 12) and ($tem_reclamacoes != true)
        {
           echo 'parabéns, você foi promovido<br>' . PHP_EOL;
        }
```

# Estrutura de Controle - Operador Ternário

**If tradicional:**

```
if ($valor_venda > 100) {
        $resultado = 'muito caro';
} else {
        $resultado = 'pode comprar';
}
```

# Estrutura de Controle - Operador Ternário

**Operador Ternário:**

```
resultado = ($valor_venda > 100) ? 'muito caro' : 'pode comprar';
```

**A primeira** expressão é a condição a ser avaliada;
**a segunda** é o valor atribuído caso ela seja verdadeira;
**e a terceira** é o valor atribuído caso ela seja falsa.

# Estrutura de Controle - WHILE

O WHILE é uma estrutura de controle similar ao If. Da mesma forma, contém uma condição para executar um bloco de comandos. A diferença primordial é que o WHILE estabelece um laço de repetição, ou seja, o bloco de comandos será executado repetitivamente enquanto a condição de entrada dada pela expressão for verdadeira. Esse comando pode ser interpretado como "ENQUANTO (expressão) FAÇA {comandos...}".

```
<?php
    $a = 1;
    while ($a < 5) {
        echo $a;
        $a++;
    }
```

# Estrutura de Controle - FOR

O FOR é uma estrutura de controle que estabelece um laço de repetição baseado em um contador; é muito similar ao comando WHILE. O FOR é controlado por um bloco de três comandos que estabelecem uma contagem, ou seja, o cloco de comandos será executado um certo número de vezes.

```
for(expr1; expr2; expr3) {
      comandos
}
```

**expr1** - Valor inicial da variável contadora.
**expr2** - Condição de execução. Enquanto TRUE, o bloco de comandos será executado.
**expr3** - Valor a ser incrementado após cada execução

# Estrutura de Controle - FOR

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

Procure utilizar nomes sugestivos para variáveis, mas, em alguns casos específicos como em contadores, permita-se utilizar variáveis de apenas uma letra, como no exemplo a seguir:

```
<php
    for ($i = 0; $i < 5; $i++) {
        for ($j = 0; $j < 4; $j++) {
            for ($k = 0; $k < 3; $k++) {
                //comandos...
             }
         }
    }
```

# Estrutura de Controle - FOR

Evite laços de repetição com muitos nós de iteração. Como o próprio Linus Torvalds já disse certa vez, se você está utilizando três níveis encadeados ou mais, considere a possibilidade de revisar a lógica de seu programa.

## Estrutura de Controle - SWITCH

```php
<?php
        switch ($variavel) {
            case valor1:
                //Comandos
                break;
            case valor2:
                //Comandos
                break;
            default:
                //Comandos
        }
```

# Estrutura de Controle - SWITCH

```php
<?php
    switch ($variavel) {
        case valor1:
            //Comandos
            break;
        case valor2:
            //Comandos
            break;
         default:
              //Comandos
    }
```

# Estrutura de Controle - SWITCH

```php
<?php //Primeiro um if
        $i = 1;
        if ($i == 0) {
            echo "i é igual a 0";
        } elseif ($i == 1) {
            echo "i é igual a 1";
        } elseif ($i == 2) {
            echo "i é igual a 2";
        } else {
            print "i não é igual a 0, 1 ou 2";
        }
```

# Estrutura de Controle - SWITCH

O switch executa linha por linha até encontrar a ocorrência de break. Por isso a importância do comando break para evitar os blocos de comando seguinte sejam executados por engano. A cláusula default será executada caso nenhuma das expressões anteriores tenha sido verificadas.

# Estrutura de Controle - SWITCH

```
$i = 1;
switch ($i) {
    case 0:
        echo "i é igual a 0";
        break;
    case 1:
        echo "i é igual a 1";
        break;
    case 2:
        echo "i é igual a 2";
        break;
    default:
        echo "i não é igual a 0, 1 ou 2";
}
```

# Estrutura de Controle – FOREACH

O **foreach** é um laço de repetição para iterações em arrays ou matrizes. É um FOR simplificado que decompõe u vetor ou uma matriz em cada um de seus elementos por meios de sua cláusula "**as**".

```
foreach ($array as $valor) {
        //intruções
}
```

# Estrutura de Controle – FOREACH

```
<?php
        $fruta = array("maçã", "laranja", "pera", "banana");
         foreach ($fruta as $valor) {
            echo "$valor - ";
         }
```

RESULTADO: maça - laranja - pera - banana -

# Estrutura de Controle – FOREACH

```
<?php
        $fruta = array("maçã", "laranja", "pera", "banana");
         foreach ($fruta as $valor) {
            echo "$valor - ";
         }
```

RESULTADO: maça - laranja - pera - banana -

# Estrutura de Controle – CONTINUE

A instrução continue, quando executada em um bloco de comandos for/while, ignora as instruções restantes até o fechamento em }. Dessa forma o programa segue para a próxima verificação da condição de entrada do laço de repetição.

# Estrutura de Controle – BREAK

O comando break aborta a execução de blocos de comandos, como If, While, for.

Quando estamos em uma execução com muitos níveis de iteração e desejamos abortar n níveis, a sintaxe é a seguinte:

```
while...
        for...
            break <quantidade de níveis>
```