DESENVOLVIMENTO DE APLICAÇÕES WEB – DAW (CURSO DE SI)

Funções em PHP

Funções em PHP podem ser definidas de forma semelhante às outras linguagens de programação, mas existem diferenças como veremos a seguir.

1. Um exemplo

Observe, no trecho abaixo, a forma como uma função é definida e invocada. A diferença mais perceptível é a ausência da declaração do tipo das variáveis, mas isto ocorre em todos os scripts PHP, como vimos em aulas anteriore:

```
<?php

// Exemplo de definição de uma função

function f($x) {

   echo "<p>Este é um exemplo de função</p>";
   echo "<p>que imprime o conteúdo de uma variável.</p>";
   echo "<p>Valor da variável \$x = $x</p>";
   return 1000;

}

echo "<p>Executando a função:</P>";

// Exemplo de chamada da função

$retorno = f("teste...");

echo '<br>$retorno = ' . $retorno;

?>
```

2. Passagem de parâmetro por valor

Normalmente, você passará os parâmetros por valor. Observe como os parâmetros são apresentados, no exemplo abaixo.

```php
<?php

// Exemplo de definição de uma função que recebe parâmetro por valor:

function f_valor($inteiro) {

   return ($inteiro = 2 * $inteiro);
}
```

Neste exemplo, o valor retornado pela função é o dobro do valor recebido através do parâmetro, no entanto o valor da variável passado como parâmetro não é alterado no chamador.

3. Passagem de parâmetro por referência

Se você precisa alterar os valores dos parâmetros e manter as alterações após o retorno da função, precisará passá-los por referência.

```php
// Exemplo de definição de uma função que recebe parâmetro por
// referência:

function f_referencia(&$inteiro) {

   return ($inteiro = 2 * $inteiro);

}
```

Neste exemplo, a função retorna o dobro do valor recebido através do parâmetro e a alteração é mantida na volta ao chamador.

4. Parâmetros com valores “default”

Você pode desejar que os parâmetros de uma função assumissem certos valores quando a função for invocada com valores não informados. Neste caso, estes parâmetros precisam ser os últimos. Examine o exemplo a seguir, e observe como os valores “defaults” são informados na definição da função. Se na invocação, o parâmetro real não for informado, o valor “default” será utilizado. No trecho abaixo, se a função for chamada sem valor para “estado_civil”, será assumido “solteiro” e, analogamente, para profissão, será assumido “estudante”.

```
<?php

function exemplo_parm_default(
             $parm1,
             $parm2,
             $estado_civil = "solteiro",
             $profissao = "estudante") {
.
.
}
.
.
.
?>
```

5. Uma questão de escopo.

Analise o script seguinte no que se refere a escopo de variáveis.

```
<?php

$var = 100;

function imprime_var(){

if (is_null($var))
   echo "variável indefinida";
else
   echo "Variavel = $var";
}

imprime_var();

?>
```

A função `imprime_var` utiliza a variável `$var`, sem que a tenha definido. O que você espera que aconteça?

Se você pensou que funciona como na linguagen “C”, não acertou. Em PHP, a variável só existe dentro do seu escopo local, a menos que você expressamente declare que se refere à variável global, da seguinte forma:

```
global $var, $var1, $var2, ...
```

dentro da função.

Ou seja, no exemplo acima, o resultado apresentará a mensagem de que a variável é indefinida.

6. Variáveis locais com cláusula “static”

Às vezes, você gostaria que uma variável local de uma função fosse inicializada apenas na primeira invocação, ou seja, que na segunda e demais execuções, a variável não fosse reinicializada. Isto é conseguido através da declaração “static”.

```
<?php

function f() {

   static $i = 1;
   .
   .
   .
}
```

7. Funções com número variável de parâmetros

Em PHP podemos definir funções com número variável de parâmetros. Neste caso, você deve utilizar as funções:

```
func_num_args() - retorna o número de argumentos.
func_get_arg(n) - retorna o (n+1) ésimo parâmetro, começando com 1.
func_get_args() - retorna um vetor com a lista de parâmetros com
índices numéricos, começando pelo índice zero.
```

8. Comando “switch”

Quando o fluxo de execução de um script depende apenas do conteúdo de uma variável é mais simples utilizar a construção abaixo, em lugar do comando “IF”. Veja a estrutura do comando “switch”.

```
.
.
.
switch ($i) {
    case 0:
        print "i igual a 0";
        break;
    case 1:
        print "i igual a 1";
        break;
    case 2:
        print "i igual a 2";
        break;
    default:
        print "i não é igual a 0, 1 ou 2";
}
.
.
.
?>
```

Atividades de hoje

Atenção: Nos comandos de repetição, não produza ciclos infinitos porque afetará todos os colegas que utilizam o ambiente.

1. Em aulas anteriores, você já estudou alguns comandos de controle de "loop". Com o apoio dos elementos já conhecidos, crie uma função com a definição:

```
function soma($m, $n)
```

que soma iterativamente, sem utilizar a forma da soma de uma PA (progressão aritmética), os números inteiros dentro de um intervalo fechado [$m, $n]. Ou seja, soma os elementos m, m + 1, m + 2, m + 3,...n. Vamos considerar também a possibilidade de se informar os parâmetros em que o segundo deles é menor que o primeiro. Neste caso, a soma deve incluir os elementos n, n + 1, n + 2, n + 3,..., m. A função deve retornar o valor da soma dos elementos. Se os parâmetros tiverem o mesmo valor, deve retornar o valor de um deles.

```
soma1.php
```

e, no mesmo arquivo, chame a função com os valores 5 e 10 para os parâmetros e imprima o resultado. Envie o arquivo para o servidor e verifique se está funcionando.

2. Crie o script de nome

```
soma2.php
```

que calcule também a mesma soma do item anterior, porém utilizando a função

```
function soma($m, $n)
```

construída de forma recursiva.

Crie uma segunda função de nome

```
chama_varias_vezes($p)
```

que abre um loop "for" e chama a função "soma" para os seguintes valores dos parâmetros:

Primeiro parâmetro: 1 (valor fixo)
Segundo parâmetro: 1, 2,..., $p.

No script PHP, chame esta segunda função com o parâmetro 100.

3. Crie o script de nome fat1.php que contenha função

```
function fat($inteiro)
```

que calcula recursivamente o fatorial de um número inteiro maior ou igual a zero. Por definição, o fatorial de 0 é 1.

Crie uma segunda função de nome

```
chama_varias_vezes($n)
```

que abre um loop “for” e chama a função “fat” para os inteiros de 0 a n.

No script PHP, chame esta segunda função com o parâmetro 10. O resultado deve ser semelhante ao seguinte:

```
Fatorial de 0 = 1
Fatorial de 1 = 1
Fatorial de 2 = 2
Fatorial de 3 = 6
```

4. Crie o script de nome fat2.php com a seguinte função:

```
fat($inteiro,&$fat)
```

A função não é recursiva. O primeiro parâmetro é passado por valor e contem o número para o qual o fatorial será calculado. O segundo parâmetro é passado por referência e deve conter o resultado do fatorial. O retorno da função deve ser uma cadeia com os dizeres similares aos seguintes:

```
return 'fatorial inexpressivo'; // Para resultado <= 1000
return 'fatorial expressivo';   // Para resultado >  1000
```

Crie uma segunda função de nome

```
chama_varias_vezes($n)
```

que abre um loop “for” e chama a função “fat” para os primeiros “n” inteiros que recebe como parâmetro. Para cada valor, deve imprimir o valor para o qual o fatorial deve ser calculado, o resultado do fatorial e a cadeia retornada. Algo como o seguinte:

```
Fatorial de 6 = 720 com retorno=fatorial inexpressivo.
Fatorial de 7 = 5040 com retorno=fatorial expressivo.
```

No script PHP, chame esta segunda função com o parâmetro 10.

5. Considere o header da função abaixo:

```
function imprime_argumentos(
             $codigo_produto,
             $nome_produto,
             $tipo = "nacional",
             $unidade = "kilograma")
```

Complete o script no arquivo de nome

```
parm_def.php,
```

de forma que a função apresente na tela os valores dos parâmetros da função.

Chame a função para as três combinações válidas dos valores dos parâmetros.

6. Crie o script de nome

```
escopo.php
```

com o seguinte, supondo um comportamento usual da regra de escopo:

a) No início inicializa a seguinte variável

```
$fator = 10;
```

b) Cria uma funçao

```
function escopo1($i)
```

que retorna o valor do produto entre os valores de $i e $fator

c) Cria uma função de nome

```
function escopo2($i)
```

com o mesmo objetivo, mas define a variável $fator como global.

d) Atribui à variavel $resultado1 o valor retornado pela função escopo1 para o argumento 10.

e) Atribui à variavel $resultado2 o valor retornado pela função escopo2 para o argumento 10.

f) Imprime o conteúdo das variáveis $resultado1 e $resultado2.

g) Execute o script.

7. Crie o script de nome

```
static1.php
```

com o seguinte:

a) Uma função de nome

```
f()
```

que inicializa a variável

```
$i = 1;
```

b) Dentro da função, utilize o comando switch para inspecionar o conteúdo da variável $i e imprime o resultado da inspeçao da seguinte forma:

```
switch ($i) {
   case 1 : echo "<p>Função chamada pela primeira vez.</p>";break;
   case 2 : echo "<p>Função chamada pela segunda vez.</p>";break;
   case 3 : echo "<p>Função chamada pela terceira vez.</p>";break;
   default: echo "<p>Função chamada pela $i-ésima vez.</p>";break;
}
```

c) Antes de sair da função, incrementa o valor da variável $i.

```
$i++;
```

d) Depois, chama a função f(x) dez vezes.

8. Crie o script de nome

```
static2.php
```

com o conteúdo do script static1.php, mas declarando a variável $i como “static” .

9. Crie o script de nome

```
qt_var_parms.php
```

com uma função de nome

```
function qt_parms()
```

que pode ser chamada com uma quantidade variável de matrículas de alunos. A função deve realizar as seguintes tarefas:

a) Imprimir a quantidade de argumentos recebidos;
b) Imprimir o índice e o conteúdo de cada argumento recebido, utilizando uma função própria para obter estes elementos;
c) Salvar a lista de parâmetros no vetor de nome

```
$vetor_args
```

d) Percorrer este vetor e imprimir o índice e o respectivo conteúdo.

No final do script, chamar a função quatro vezes, com quantidades diferentes de parâmetros.

Até a próxima aula!
Prof. Satoshi Nagayama