Funções de usuário em PHP

- Em PHP, podemos utilizar funções para desviar o fluxo de execução do script principal;
- O objetivo de se usar uma função é utilizar um mesmo bloco de código para realizar inúmeras vezes uma **mesma tarefa**, que será executada com frequência. Isso permite a reutilização de código;
- O script principal, ao encontrar uma chamada de função, é "congelado". O fluxo de execução é desviado para a função e tudo o que estiver dentro dela é executado. Após isso, o fluxo retorna ao script principal.

Ainda sobre funções

- Variáveis criadas dentro de uma função só podem ser usadas pela função. Elas são destruídas depois que a função termina. **São variáveis locais**;
- Variáveis criadas pelo programa principal estão disponíveis em qualquer lugar do script, menos dentro das funções – **são as variáveis globais**;
- O resultado de uma função é devolvido ao programa que a chamou pelo comando ***return***.

Mais sobre funções

- Uma função recebe um nome, que segue as mesmas regras de criação de nomes de variáveis;
- Uma função também recebe uma lista de variáveis, chamadas de **parâmetros**;
- No programa principal, uma função pode ser chamada como uma expressão dentro de uma estrutura de seleção ou repetição. Ex.:

```
if (Calcula_Media($nota1, $nota2) >= 7)
   echo("Aluno aprovado");
else
   echo("Aluno reprovado");
```

Retornando um array

- Geralmente, uma função retorna um único valor. Para que a função devolva mais de um valor a quem a chamou, fazemos a função criar um vetor com todos os elementos de retorno dentro deste vetor. Exemplo:

```
<?php
function Cria_Retorna_Multiplos_Valores()
   {
   $dias = array('domingo', 'segunda', 'terça');
   return $dias;
   }

//programa principal
$vetor_dias = Cria_Retorna_Multiplos_Valores();
foreach($vetor_dias as $dia)
   echo("Dias da semana: $dia <br />");
?>
```

Passagem de parâmetros

- **Por valor e por referência**;
- *Por valor* – a mudança do valor do parâmetro dentro da função não afeta o valor da variável fora da função;
- *Por referência* – quando usamos este método, qualquer modificação feita sobre os parâmetros dentro da função alterará o valor da correspondente variável fora da função, isto é, dentro do script principal. Usamos, para isto, antes do nome do parâmetro, o símbolo **&**. A variável comporta-se como se fosse global.
- function passarPorValor(idade){ //comandos }
- function passarPorReferencia(**&**idade) { //comandos}

Parâmetros – valores-padrão

- Quando, ao chamarmos uma função, não passamos todos os parâmetros, podemos, na lista de valores, definir um valor-padrão. Para tanto, basta colocar um operador de atribuição no parâmetro seguido do valor-padrão que queremos para ele.
- Exemplo:

```
function soma($a, $b=0)
{ return $a + $b;}
```

- Esta função, no script principal, pode ser invocada com os dois argumentos normalmente. Ou então, com apenas o primeiro argumento. Neste caso, o PHP usa o parâmetro default para o segundo, isto é, $b é zero. Assim:

```
$chamaFunc = soma(valor1);
```

Chamada de uma função

- Ao chamarmos uma função dentro do script principal, podemos ter duas situações:

- **A)**a função deve retornar um ou mais valores - neste caso, a chamada é feita conforme o modelo abaixo:

```
$variavel_retorno = nome_função();
```

- **B)**a função não retorna nenhum valor para o script principal – neste caso, a chamada se torna:

```
nome_função();
```