Definindo funções

A sintaxe básica para definir uma função é:

```
function nome_da_função([arg1, arg2, arg3]) {
          Comandos;
          ... ;
          [return <valor de retorno>];
}
```

Qualquer código PHP válido pode estar contido no interior de uma função. Como a checagem de tipos em PHP é dinâmica, o tipo de retorno não deve ser declarado, sendo necessário que o programador esteja atento para que a função retorne o tipo desejado. É recomendável que esteja tudo bem documentado para facilitar a leitura e compreensão do código. Para efeito de documentação, utiliza-se o seguinte formato de declaração de função:

```
tipo function nome_da_funcao(tipo arg1, tipo arg2, ...);
```

Este formato só deve ser utilizado na documentação do script, pois o PHP não aceita a declaração de tipos. Isso significa que em muitos casos o programador deve estar atento ao tipos dos valores passados como parâmetros, pois se não for passado o tipo esperado não é emitido nenhum alerta pelo interpretador PHP, já que este não testa os tipos.

Valor de retorno

Toda função pode opcionalmente retornar um valor, ou simplesmente executar os comandos e não retornar valor algum.

Não é possível que uma função retorne mais de um valor, mas é permitido fazer com que uma função retorne um valor composto, como listas ou arrays.

Argumentos

É possível passar argumentos para uma função. Eles devem ser declarados logo após o nome da função, entre parênteses, e tornam-se variáveis pertencentes ao escopo local da função. A declaração do tipo de cada argumento também é utilizada apenas para efeito de documentação.

Exemplo:

```
function imprime($texto){
   echo $texto;
}

imprime("teste de funções");
```

Passagem de parâmetros por referência

Normalmente, a passagem de parâmetros em PHP é feita por valor, ou seja, se o conteúdo da variável for alterado, essa alteração não afeta a variável original.

Exemplo:

```
function mais5($numero) {
   $numero += 5;
}

$a = 3;
mais5($a); //$a continua valendo 3
```

No exemplo acima, como a passagem de parâmetros é por valor, a função mais5 é inútil, já que após a execução sair da função o valor anterior da variável é recuperado. Se a passagem de valor fosse feita por referência, a variável $a teria 8 como valor. O que ocorre normalmente é que ao ser chamada uma função, o interpretador salva todo o escopo atual, ou seja, os conteúdos das variáveis. Se uma dessas variáveis for passada como parâmetro, seu conteúdo fica preservado, pois a função irá trabalhar na verdade com uma cópia da variável. Porém, se a passagem de parâmetros for feita por referência, toda alteração que a função realizar no valor passado como parâmetro afetará a variável que o contém.

Há duas maneiras de fazer com que uma função tenha parâmetros passados por referência: indicando isso na declaração da função, o que faz com que a

passagem de parâmetros sempre seja assim; e também na própria chamada da função. Nos dois casos utiliza-se o modificador "&". Vejamos um exemplo que ilustra os dois casos:

```
function mais5(&$num1, $num2) {
   $num1 += 5;
   $num2 += 5;
}

$a = $b = 1;
mais5($a, $b); /* Neste caso, só $num1 terá seu valor alterado, pois a passagem por referência está definida na declaração da função. */


mais5($a, &$b); /* Aqui as duas variáveis terão seus valores alterados. */
```

Argumentos com valores pré-definidos (default)

Em PHP é possível ter valores *default* para argumentos de funções, ou seja, valores que serão assumidos em caso de nada ser passado no lugar do argumento. Quando algum parâmetro é declarado desta maneira, a passagem do mesmo na chamada da função torna-se opcional.

```
function teste($texto = "testando") {
   echo $texto;
}

teste(); // imprime "testando"
teste("outro teste"); // imprime "outro teste"
```

É bom lembrar que quando a função tem mais de um parâmetro, o que tem valor *default* deve ser declarado por último:

```
function teste($figura = circulo, $cor) {
  echo "a figura é um ", $figura, " de cor " $cor;
}

teste(azul);
/* A função não vai funcionar da maneira esperada, ocorrendo um erro no interpretador. A declaração correta é: */
```

```
function teste2($cor, $figura = circulo) {
  echo "a figura é um ", $figura, " de cor " $cor;
}

teste2(azul);

/* Aqui a função funciona da maneira esperada, ou seja,
imprime o texto: "a figura é um círculo de cor azul" */
```

Contexto

O contexto é o conjunto de variáveis e seus respectivos valores num determinado ponto do programa. Na chamada de uma função, ao iniciar a execução do bloco que contém a implementação da mesma é criado um novo contexto, contendo as variáveis declaradas dentro do bloco, ou seja, todas as variáveis utilizadas dentro daquele bloco serão eliminadas ao término da execução da função.

Escopo

O escopo de uma variável em PHP define a porção do programa onde ela pode ser utilizada. Na maioria dos casos todas as variáveis têm escopo global. Entretanto, em funções definidas pelo usuário um escopo local é criado. Uma variável de escopo global não pode ser utilizada no interior de uma função sem que haja uma declaração.

```
Exemplo:
$texto = "Testando";

function Teste() {
  echo $texto;
}

Teste();
```

O trecho acima não produzirá saída alguma, pois a variável $texto é de escopo global, e não pode ser referida num escopo local, mesmo que não haja outra com nome igual que cubra a sua visibilidade. Para que o script funcione da forma desejada, a variável global a ser utilizada deve ser declarada.

```
Exemplo:
$vivas = “Testando”;

function Teste() {
  global $vivas;
  echo $vivas;
}

Teste();
```

Uma declaração “global” pode conter várias variáveis, separadas por vírgulas. Outra maneira de acessar variáveis de escopo global dentro de uma função é utilizando um array pré-definido pelo PHP cujo nome é $GLOBALS. O índice para a variável referida é o próprio nome da variável, sem o caractere $. O exemplo acima e o abaixo produzem o mesmo resultado:

```
Exemplo:
$vivas = "Testando";

function Teste() {
  echo $GLOBALS["vivas"]; // imprime $vivas
  echo $vivas; // não imprime nada
}

Teste();
```

Exercícios

1. Escreva uma função em PHP para mostrar os números pares de 200 a 300.

2. Altere o exercício anterior, fazendo com que a função receba os valores preenchidos em sua chamada, fixos.

3. Altere o exercício anterior de forma que o usuário possa especificar em um formulário o intervalo que será utilizado (especifica os valores das variáveis).

4. Escreva um programa em PHP utilizando funções para receber 2 números em um formulário e exibir:

   - sua soma

   - sua subtração

   - sua multiplicação

- sua divisão

- o resto de sua divisão

- a média entre eles