MODUL

PEMOGRAMAN WEB II

Oleh:

CHALIFA CHAZAR

MODUL 5

Modularisasi dan Fungsi

Tujuan:

Mahasiswa memahami penggunaan fungsi untuk menyelesaikan permasalahan tertentu dan untuk meminimalisir penulisan kode PHP dalam bentuk modularisasi.

Pustaka:

Raharjo, B. (2015): ***Mudah Belajar PHP Teknik Penggunaan Fitur-Fitur Baru Dalam PHP 5*****. Informatika. Bandung**

Prasetio, A. (2015): ***Buku Pintar Webmaster*****.**

www.w3schools.com

1. Modularisasi Kode PHP

Pada kenyataannya, aplikasi yang ditulis dengan menggunakan PHP terdiri dari bebeerapa file. Contohnya pada aplikasi web, satu file mewakili satu halaman, atau bisa juga berisi kumpulan fungsi/kelas yang berperan sebagai pustaka pendukung dari aplikasi yang kita kembangkan.

Bahasa pemograman PHP juga memungkin kita untuk menggunakan kode-kode yang ditulis di dalam file lain (tidak satu file dengan kode pemanggil). PHP menyediakan statement **`include`**, **`include_once`**, **`require`** dan **`require_code`**.

1. Untuk memahami cara kerjanya ikuti langkah-langkah berikut ini.
2. Buat dua buah file kode program, beri nama **`modul.php`** dan **`index.php`**.
3. Tulis kode berikut ini pada file **`modul.php`**.

```
<?php
function tambah($a, $b) {
      return $a + $b;
}
function kali($a, $b) {
      return $a * $b;
}
?>
```

4. Tulis kode berikut ini pada file **index.php**.

```
<?php
//menyertakan file modul.php
include "modul.php";

//memamnggil fungsi yang ada dalam modul.php
$x = tambah(10, 8);
$y = kali(5, 20);

echo "Hasil penjumlahan : " . $x;
echo "<br>";
echo "Hasil perkalian : " . $y;
?>
```

5. Buka file index.php pada browser.

Saat **index.php** dieksekusi, maka program akan membaca adanya fungsi yang telah dibuat dalam file **modul.php** dan kemudian dieksekusi. Agar fungsi yang dibuat dalam file **modul.php** dapat dibaca o;eh file **index.php**, maka kita perlu menyertakan file **modul.php** dengan menggunakan kode **include** atau **require**.

```
include "modul.php";

// atau

require "modul.php";
```

2. Fungsi

Fungsi adalah suatu blok kode program yang bertugas menyelesaikan suatu permasalahan spesifik tertentu, misalnya untuk melakukan perhitungan, manipulasi string, koneksi database, dan sebagainya. Dengan adanya fungsi, kode PHP menjadi lebih modular dan mudah dipelihara. Fungsi juga dapat meminimalisir kode yang berulang.

2.1. Jenis Fungsi dalam PHP

Dalam PHP, fungsi dapat dibedakan menjadi dua jenis, yaitu:

- Fungsi tanpa nilai balik
- Fungsi dengan nilai balik

Fungsi tanpa nilai balik adalah fungsi yang tidak menghasilkan nilai. Fungsi ini hanya digunakan untuk melakukan proses tertentu. Misalnya, **`printf()`** adalah fungsi yang hanya digunakan untuk melakukan pencetakan teks dengan format tertentu ke layar.

Fungsi dengan nilai balik adalah fungsi yang ketika dipanggil akan menghasilkan nilai. Misalnya, **`pow(a,b)`** merupakan fungsi yang akan menghitung nilai **a** pangkat **b**.

2.3. Mendefinisikan Fungsi

Bentuk umum pendefinisian fungsi dalam PHP, adalah sebagai berikut:

```
<?php
function NamaFungsi(daftar-parameter) {
  statemen;
  ...
  return [ekspresi];
}
?>
```

Contoh fungsi tanpa nilai balik:

```
<?php
function tulis($s) {
  print $s;
  return; //optional
}
?>
```

Contoh fungsi dengan nilai balik:

```
<?php
function tambah($a, $b) {
  $hasil = $a + $b;
  return $hasil; //wajib disertakan
}
?>
```

2.4. Parameter Fungsi

Parameter adalah suatu nilai (berupa variabel) yang dikirimkan ke dalam file fungsi, yang kemudian akan ikut diproses di dalam badan fungsi. Dengan menggunakan parameter suatu fungsi dapat memberikan hasil yang dinamis atau berubah-ubah setiap fungsi tersebut dipanggil.

Contoh:

```
<?php

//memanggil fungsi sqrt() untuk menghitung akar kuadrat
$x = sqrt(4);
$y = sqrt (25);

//menampilkan hasil
echo "\$x = $x\n";
echo "\$y = $y\n";

?>
```

Fungsi **`sqrt()`** merupakan fungsi yang ada pada pustaka PHP untuk menghitung akar kuadrat dari sebuah nilai. Fungsi **`sqrt()`** akan memberikan hasil yang berbeda sesuai dengan nilai parameter yang dilewatkan ke dalam fungsi.

2.5. Parameter Formal dan Parameter Aktual

Parameter formal adalah parameter-parameter yang terdapat pada bagian definisi fungsi. Perhatikan contoh di bawah ini.

```
<?php
function tambah($a, $b) {
  $hasil = $a + $b;
  return $hasil;
}
?>
```

Catatan:

Pada contoh di atas, **$a** dan **$b** adalah parameter formal.

Parameter aktual adalah parameter-parameter yang dikirimkan atau dilewatkan pada saat pemanggilan fungsi. Perhatikan contoh dibawah ini.

```
<?php
$bil1 = 10;
$bil2 = 20;

//memanggil fungsi kali()
$x = kali($bill1, $bil2);
?>
```

Catatan:

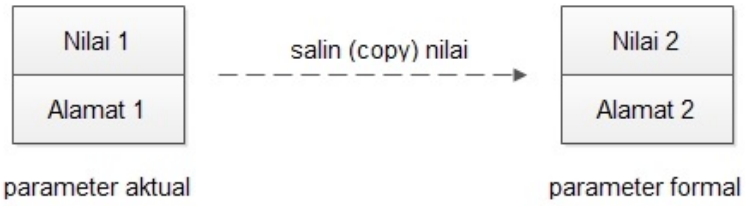
Pada contoh di atas, **$bil1** dan **$bil2** adalah parameter aktual. Parameter aktual juga dikenal dengan sebutan argumen.

2.6. Pass-by-Value dan Pass-by-Reference

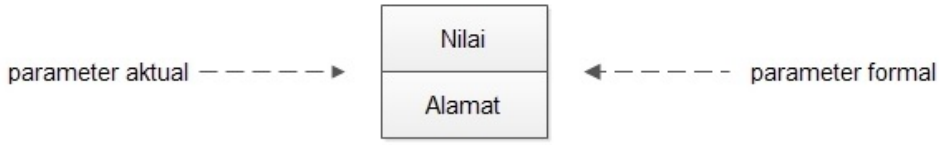
Dalam pemograman, parameter dapat dilewatkan ke dalam suatu fungsi menggunakan dua cara, yaitu:

- Pass-by-value (melewatkan parameter berdasarkan nilai)
- Pass-by-reference (melewatkan parameter berdasarkan alamat atau referensinya)

Pada mekanisme pass-by-value akan terdapat proses penyalinan nilai dari parameter aktual ke parameter formal. Karena antara parameter aktual dan parameter formal merupakan dua variabel yang berbeda dan masing-masing menempati alamat memori komputer yang berbeda pula, maka perubahan nilai yang terjadi pada parameter formal tidak akan berpengaruh terhadap nilai parameter aktual.



Berbeda dengan pass-by-reference, parameter aktual dan parameter formal akan merujuk ke alamat memori komputer yang sama. Dengan demikian, jika nilai dalam alamat memori tersebut diubah, maka keduanya akan ikut berubah. Dengan kata lain, parameter formal hanya merupakan alias (nama lain) atau referensi dari parameter aktual.



Contoh pass-by-value:

```
<?php
function kali($a, $b) {
      $hasil = $a * $b;
      return $hasil;
}

$x = 100;
$y = 200;

echo "menampilkan nilai sebelum pass-by-value" . '<br>';
echo "x adalah " . $x;
echo '<br>';
echo "y adalah " . $y;
echo '<br>';

//memanggil fungsi kali()
kali($x, $y);

echo "menampilkan nilai setelah pass-by-value" . '<br>';
echo "x adalah " . $x;
echo '<br>';
echo "y adalah " . $y;
echo '<br>';
?>
```

Catatan:

Fungsi **`kali()`** tidak akan merubah nilai variabel **`$x`** dan **`$y.`**

Contoh pass-by-reference:

```
<?php
function kali($a, $b, &$hasil) {
      $hasil = $a * $b;
      return $hasil;
}

$x = 100;
$y = 200;
$hasil = 0;

//memanggil fungsi kali()
kali($x, $y, $hasil);

//menampilkan hasil
echo "hasil perkalian:  " . $hasil;
?>
```

Catatan:

Variabel **$hasil** merupakan variabel referensi yang ada di dalam fungsi **kali()**. Untuk mendefinisikan variabel pass-by-reference digunakan tanda "**&**" sebelum variabel.

Tugas.
Buat sebuah program perhitungan matematika yang dibuat kedalam 10 buah fungsi yang berbeda-beda. Buatlah fungsi sendiri (tidak menggunakan pustaka yang ada dalam PHP).