

LAPORAN PROYEK

EKSPRESI NOTASI ALGORITMA 2

EKSPRESI,VARIABEL,CARA CETAK



OLEH:

M.Afriansyah

(NISN. -0084156265)

REKAYASA PERANGKAT LUNAK

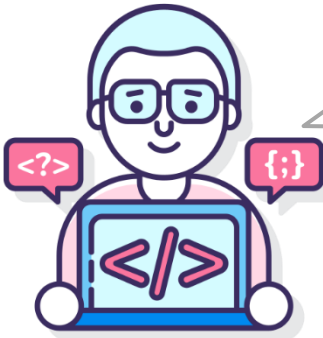
SMK NEGERI 1 KARANG BARU

PEMERINTAH PROVINSI ACEH 2024



MATERI PERTEMUAN 5

Sintaks Dasar PHP yang Wajib Kamu Pahami

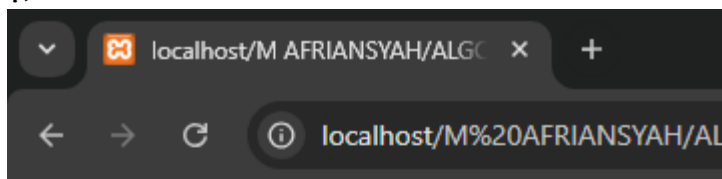


TAHUKAH KAMU...?

- “Bagaimana Struktur Program PHP yang Paling Dasar?”
- “Bagaimana Menulis Kode HTML dan PHP?”
- “Bagaimana Penulisan Statement dan Ekspresi?”
- “Bagaimana Aturan Penulisan Case PHP?”
- “Bagaimana Penulisan Komentar di PHP?”
- “Bagaimana Penulisan Blok Program?”

Kita sudah belajar membuat program **hello_world.php** dengan kode seperti ini:

```
<?php  
echo "Hello World!";  
?>
```



Hello World!
nama saya afriansyah
nama saya afriansyah

Ini adalah contoh program PHP yang paling sederhana.
Program tersebut hanya berfungsi untuk menampilkan teks **Hello World** saja.
Tapi...

Apa maksudnya **<?php** dan **?>**?
Apa itu echo?
...dan kenapa harus ditulis seperti itu?

Apa itu sintak?

Sintak adalah aturan penulisan kode program.

Pada dasarnya setiap bahasa pemrograman itu sama, yang membedakan adalah sintak dan fitur.

Struktur Program PHP yang Paling Dasar

Ini adalah bentuk paling dasar program PHP:

```
<?php  
  
echo "Hello World!";
```

Keterangan:

- **<?php** ini adalah pembuka program PHP. Pembuka ini wajib ada di setiap program PHP.
- **echo "Hello World!";** adalah sebuah statement atau perintah untuk menampilkan teks.

Tunggu dulu...

Kenapa tidak ditutup dengan **?>**?

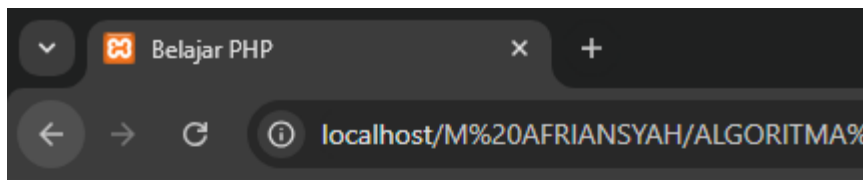
Tutup sebenarnya bersifat opsional. Tutup program dibutuhkan saat kita menggabungkan kode PHP dengan HTML.

Menulis Kode HTML dan PHP

Saat kita menulis kode PHP di dalam HTML, maka wajib hukumnya membuat tutup program.

Contoh:

```
<!DOCTYPE html>  
<html>  
<head>  
<title><?php echo "Belajar PHP" ?></title>  
</head>  
<body>  
<?php  
    echo "kita sedang belajar PHP<br>";  
    echo "<p>Belajar PHP hingga jadi master</p>";  
?>  
</body>  
</html>
```



kita sedang belajar PHP

Belajar PHP hingga jadi master

By : M.Afriansyah

Perhatikan contoh di atas!

Apa yang akan terjadi bila kita menghapus tutup PHP (=)?</p

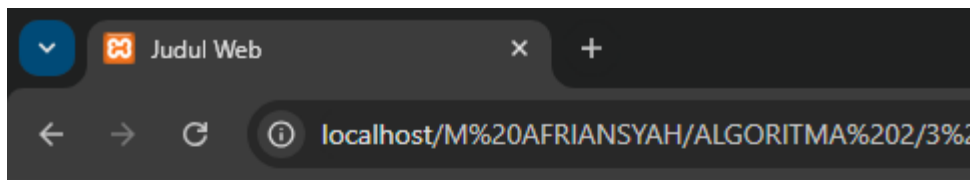
Tentunya program akan error.

Oya, PHP yang ditulis di dalam HTML, filenya harus disimpan dengan ekstensi **.php** bukan **.html** meskipun isinya HTML dan PHP.

Lalu ada juga yang menulis seperti ini:

```
<?php
```

```
echo "<html>";  
echo "<head>";  
echo "<title>Judul Web</title>";  
echo "</head>";  
echo "<body>";  
echo "<h1>Selamat datang</h1>";  
echo "</body>";  
echo "</html>";
```



Selamat datang

di smk 1 karang baru aceh tamiang

By M AFRIANSYAH

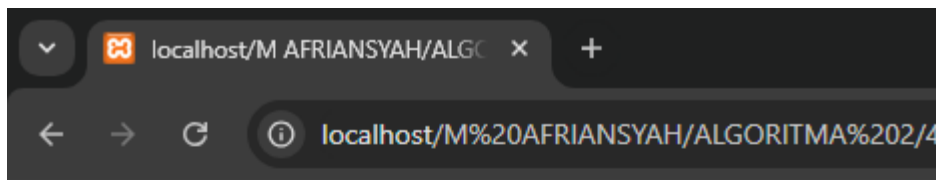
Nah kalau yang ini bisa tidak ditutup, karena kode HTML-nya ditulis di dalam sintak PHP. Masih bingung?

Penulisan Statement dan Ekspresi

Statement dan ekspresi adalah intruksi yang akan diberikan ke komputer. Setiap statement dan ekspresi di PHP harus diakhiri dengan titik koma (;).

Contoh:

```
<?php
echo "ini statement 1";
echo "ini statement 2";
$a = $b + $c;
```



ini statement 1 ini statement 2

ini ekspresi -> $9 = 4 + 5$;

ini ekspresi -> $9 = 4 + 5$;

yaitu : .9

yaitu : .9

ini statement 1

ini statement 2

by **MAFRIANSYAH**

Gimana kalau kita lupa menuliskan titik koma? tentu programnya akan error.

Aturan Penulisan Case PHP

PHP adalah bahasa pemrograman yang bersifat *case sensitive*. Artinya, huruf besar (kapital) dan huruf kecil akan dibedakan.

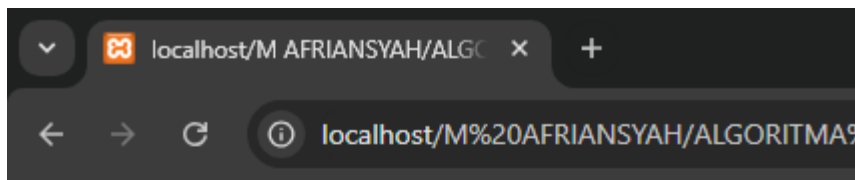
Contoh:

```
<?php
```

```
$nama = "ahmadimuslim";
```

```
$NAMA = "dian";
```

```
?>
```



==== mencetak variabel ====

Nama saya adalah m afriansyah

Nama ketua kelas adalah rusli

Umur saya adalah 16

coba pakai blok program

Nama saya adalah m afriansyah

Nama ketua kelas adalah rusli

Umur saya adalah 16

Variabel **\$nama** dan **\$NAMA** adalah dua variabel yang berbeda. Mereka tidak sama. Penulisan huruf besar dan kecil dalam program harus diperhatikan, karena bisa menyebabkan error bila salah.

Kita sering menemukannya, banyak yang salah tulis.

Contoh:

```
$umur = 19;
```

```
echo "Umur kita adalah $Umur";
```

Pada kode diatas, kita membuat variabel **\$umur** dengan huruf kecil. Lalu saat menggunakan variabel kita menggunakan **\$Umur**.

Jelas sekali ini akan menyebabkan error.

Untuk menghindari ini, kita harus konsisten dalam penamaan variabel dan konstanta.

Gunakan nama variabel dengan huruf kecil saja dan konstanta dengan huruf kapital.

Contoh:

```
const INI_KONSTANTA = 123;
```

```
$ini_variabel = 23;
```

```
$iniJugaVariabel = 49;
```

kok ada huruf kapital di **\$iniJugaVariabel**?

Ini namanya *camelCase*.

Jika variabel terdiri dari dua atau lebih suku kata, maka kita bisa memisahkannya dengan huruf kapital atau bisa juga dengan garis bawah (*underscore*).

Penulisan Komentar di PHP

Komentar adalah bagian yang tidak akan dieksekusi oleh komputer. Biasanya digunakan untuk keterangan, penjelasan, dan dokumentasi kode program.

Komentar di PHP dapat ditulis dengan dua cara:

1. Menggunakan tanda `//` untuk komentar satu baris;

2. Menggunakan tanda `/*` untuk komentar lebih dari satu baris.

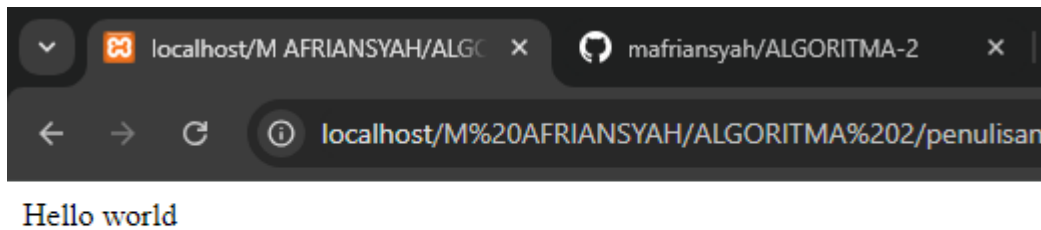
Contoh:

```
<?php

// ini adalah komentar
echo "Hello world";

/*
ini adalah komentar
yang lebih dari satu
baris
*/

?>
```



Penulisan Blok Program

Blok program merupakan kumpulan dari statement dan ekspresi. Blok program di PHP dibungkus dengan kurung kurawal `{ ... }`.

Contoh:

```
if ($umur > 18) {
    echo "Kamu sudah dewasa";
    echo "Selamat datang";
    echo "Kamu boleh minum kopi";
}
```

Program di atas adalah contoh blok kode *if* yang berisi tiga statement.

▪ LATIHAN 5

Kerjakan soal-soal berikut dengan baik dan benar!

- 1) Buatlah contoh bentuk paling sederhana dari program PHP..
- 2) Kapan membuat tutup program PHP, dan kapan tidak harus...
- 3) Apa yang dimaksud Statement dan ekspresi, dan bagaimana cara menuliskannya...
- 4) PHP adalah bahasa pemrograman yang bersifat case sensitive. Apa maksudnya...
- 5) Apa yang dimaksud dengan Komentar di PHP, dan bagaimana cara menuliskannya...
- 6) Apa yang dimaksud dengan Blok program, dan bagaimana cara menuliskannya...



MATERI PERTEMUAN 6

Cara Mencetak, Variabel Dan Tipe Data Pada PHP



TAHUKAH KAMU...?

PHP memiliki beberapa fungsi untuk mencetak teks ke layar:

- fungsi **echo()**;
- fungsi **print()**;
- fungsi **printf()**.

Apa saja perbedaan fungsinya...?

A. Variabel Pada PHP

Apa kamu pernah menemukan **x** dan **y** dalam pelajaran matematika?

Mereka berdua sebenarnya adalah variabel yang menyimpan sesuatu.

Kadang, kita sering diberi tugas untuk mencari tahu isi dari **x** dan **y**.

Contohnya:

jika $x + 3 = 5$, Berapakah **x**?

Variabel dalam pemrograman juga memiliki arti yang sama seperti dalam matematika.

Variabel adalah tempat kita menyimpan nilai sementara.

Variabel akan ada selama program dijalankan. Namun kita juga bisa menghapusnya dari memori.

1. Membuat Variabel di PHP

Pada PHP, kita membuat variabel dengan tanda dolar (\$), lalu diikuti dengan nama variabelnya serta nilai yang ingin kita simpan.

Contoh:

```
<?php
```

```
$harga = 1000;
```

Kita baru saja membuat variabel bernama **\$harga** dengan isi **1000**.

Tanda sama dengan (=) adalah simbol atau operator yang digunakan untuk mengisi nilai ke variabel.

Mudah bukan?

Mari kita coba contoh yang lain:

```
<?php
```

```
$nama_barang = "Kopi C++";
```

```
$harga = 4000;
```

```
$stok = 40;
```

Oya, variabel juga dapat diisi ulang dengan nilai yang baru.

Contoh:

```
<?php

// membuat variabel baru
$stok = 40;

// mengisi ulang variabel dengan nilai baru
$stok = 34;
```

Dalam membuat nama variabel ada 4 hal yang harus diperhatikan:

- 1) Awal dari nama variabel tidak boleh menggunakan angka dan simbol, kecuali underscore.

Contoh: ✗ Salah

```
$!nama = "";
$46rosi = "";
```

Contoh: ✓ □ Benar

```
$nama = "";
$rosi46 = "";
```

- 2) Nama variabel yang terdiri dari dua suku kata, bisa dipisah dengan underscore (_) atau menggunakan style camelCase.

Contoh:

```
$nama_barang = "Buku PHP";
$namaPembeli = "Ahmadi"; // ← menggunakan camelCase
```

- 3) Variabel harus diisi saat pembuatannya. Bila kita tidak ingin mengisi, cukup isi dengan nilai kosong.

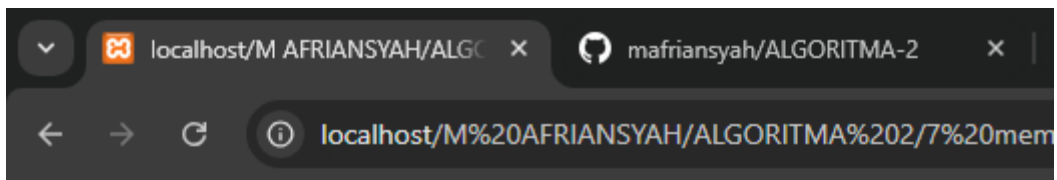
Contoh:

```
$nama_barang = "";
$namaPembeli = "";
$harga = 0;
```

- 4) Nama variabel bersifat *Case Sensitive*, artinya huruf besar dan huruf kecil dibedakan.

Contoh: Tiga variabel ini akan dianggap berbeda.

```
$Belajar = "";
$BELAJAR = "";
$belajar = "";
```



rian

200000000

laptop samsung

HP iphone 26 pro max

200000000

34

rian

rian slebew

laptop samsung

rian slebew

laptop samsung

rian slebew

200000000

PHP

Html

Css

B. Konstanta Pada PHP

Konstanta seperti variabel. Dapat digunakan untuk menyimpan nilai, tapi tidak bisa diubah.

Contoh:

```
// kita punya konstanta dengan nilai 5  
const SEBUAH_NILAI = 5
```

```
// lalu kita ubah menjadi 10  
SEBUAH_NILAI = 10 // <-- maka akan terjadi error di sini
```

Mari kita buktikan:

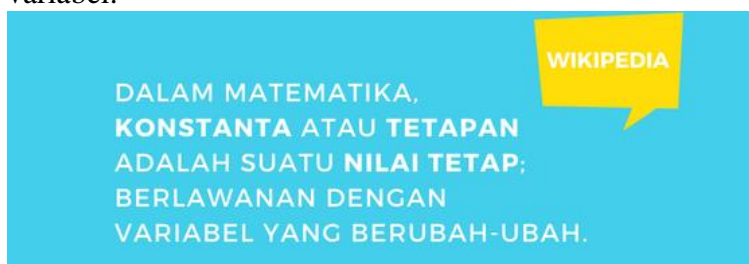
```
File Edit View Bookmarks Settings Help
php > const SITE_NAME = "Petanikode";
php > echo SITE_NAME;
Petanikode
php > SITE_NAME = "Belajar PHP";
PHP Parse error: syntax error, unexpected '=' in php shell code on line 1
php > define('SITE_NAME', 'Belajar PHP');
PHP Notice: Constant SITE_NAME already defined in php shell code on line 1
php > █
```

Pada percobaan tersebut, kita membuat konstanta dengan nama **SITE_NAME** dengan nilai **Ahmadimuslim**.

Lalu kita coba ubah nilainya menjadi **Belajar PHP**, tapi error.

Kenapa bisa error?

Karena sifat konstanta memang begitu. Nilai konstanta tidak bisa kita ubah-ubah seperti variabel.



Kapan Kita Harus Menggunakan Konstanta?

Konstanta biasanya digunakan untuk menyimpan nilai yang tidak pernah berubah.

Contoh:

```
const PHI = 3.14;  
const API_KEY = "182939812739812478u12ehj1h2u3123h12";
```

Pada dunia nyata, nilai **PHI** memang tidak akan pernah berubah. Lalu untuk **API_KEY** biasanya kita gunakan untuk mengakses sebuah web service dan nilai ini tidak akan pernah berubah di dalam program.

Jadi, kita bisa memikirkan...kapan harus menggunakan konstanta dan kapan harus menggunakan variabel.

Intinya:

Jika kita ingin menyimpan nilai yang tak akan pernah berubah, maka gunakanlah konstanta.

Sedangkan apabila nilai tersebut berubah-ubah di dalam program, maka gunakanlah variabel.

Cara Membuat Konstanta dan Contohnya

Pada PHP, kita dapat membuat konstanta dengan dua cara.

1. Menggunakan fungsi **define()**;
2. Menggunakan kata kunci **const**.

Contoh:

```
<?php  
// membuat konstanta dengan fungsi define()  
define('DB_SERVER', 'localhost');  
define('DB_USER', 'ahmadimuslim');  
define('DB_PASS', 'R4Hasia');  
define('DB_NAME', 'belajar');  
  
// membuat konstanta dengan kata kunci const  
const API_KEY = "1234";  
?>
```

Nama konstanta diharuskan menggunakan huruf kapital agar mudah dibedakan dengan variabel. Meskipun di PHP menggunakan simbol dolar (\$) untuk variabel, kita harus mengikuti aturan ini.



Lalu, bagaimana cara mengambil nilai dari konstanta?

Sama seperti variabel, kita tinggal tulis namanya.

Contoh:

```
const SITE_NAME = "Ahmadimuslim";  
  
echo "Nama situs: " . SITE_NAME;
```

Perhatikan!

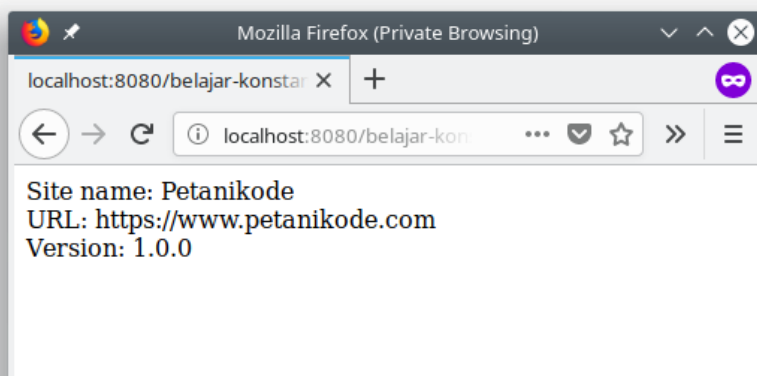
Kita menggunakan titik (.) untuk menggabungkan dua buah string. Karena konstanta tidak menggunakan dolar, kita tidak bisa langsung menulisnya seperti ini:

```
echo "Nama situs: SITE_NAME";
```

Biar makin mantap, coba contoh program berikut:

```
<?php // file: belajar-konstanta.php  
  
// membuat konstanta  
define('VERSION', '1.0.0');  
  
const SITE_NAME = "Ahmadimuslim";  
const BASE_URL = "https://www.ahmadimuslim.com";  
  
// cetak nilai konstanta  
echo "Site name: " . SITE_NAME . "<br/>";  
echo "URL: " . BASE_URL . "<br/>";  
echo "Version: " . VERSION . "<br/>";
```

Hasilnya:



C. Cara Menampilkan/Mencetak Nilai dari Variabel Pada PHP

Setelah kita membuat variabel, biasanya akan kita gunakan pada proses berikutnya dengan mengambil nilainya.

Mengambil nilai dari variabel bisa kita lakukan.

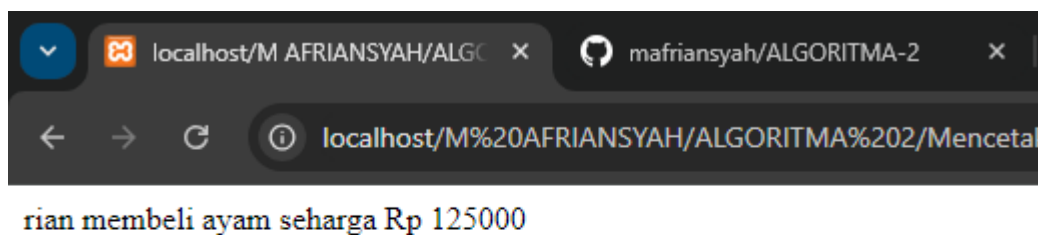
Contoh:

Akan dengan menuliskan namanya dalam perintah **echo** maupun ekspresi yang lain.

<?php

```
// membuat variabel baru
$nama_barang = "Minyak Goreng";
$harga = 15000;
//
$nama_barang . " seharga Rp $harga"; menampilkan isi variabel
echo "Ibu membeli
```

Hasilnya:



Jika kita menggunakan tanda petik ganda (") dalam **echo**, maka kita bisa menuliskan langsung nama variabelnya seperti ini:


```
$judul = "Belajar PHP dari nol sampai expert";  
echo "Judul artikel: $judul";
```

Namun...

Apabila kita menggunakan tanda petik tunggal ('), maka kita harus menggunakan titik untuk menggabungkan teks dengan variabelnya.

Contoh:

```
$judul = "Tutorial PHP untuk Pemula";  
echo 'Judul artikel: '.$judul;
```

Tanda titik pada perintah **echo** berfungsi untuk menggabungkan teks yang ada di dalam variabel dengan teks yang akan kita cetak.

Dalam PHP ada beberapa cara mencetak / menampilkan nilai dari ekspresi/deklarasi variabel sbb:

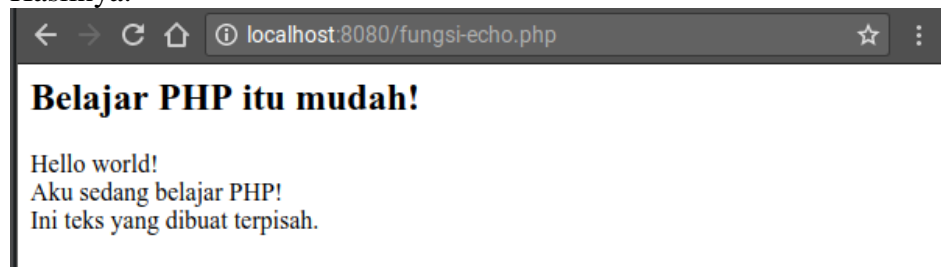
1. Fungsi echo()

Fungsi **echo()** adalah fungsi untuk menampilkan teks ke layar. Fungsi ini dapat digunakan dengan tanda kurung maupun tanpa tanda kurung.

Contoh:

```
<?php  
echo "<h2>Belajar PHP itu mudah!</h2>";  
echo("Hello world!<br>");  
echo "Disedang belajar PHP!<br>";  
echo "Ini ", "teks ", "yang ", "dibuat ", "terpisah."  
?>
```

Hasilnya:



Fungsi **echo()** tidak akan mengembalikan apa-apa setelah dieksekusi. Dia hanya bertugas menampilkan teks saja.

2. Fungsi print()

Fungsi **print()** sama seperti fungsi **echo()**. Dia juga digunakan untuk menampilkan teks ke layar. Fungsi **print()** juga bisa digunakan tanpa tanda kurung.

Perbedaannya dengan **echo()**:

- Fungsi **print()** akan selalu mengembalikan nilai **1** saat dieksekusi, sedangkan **echo()** tidak mengembalikan apa-apa.
- Fungsi **print()** hanya boleh diberikan satu parameter saja, sedangkan **echo()** boleh lebih dari satu.

Contoh:

```
<?php  
print "<h2>Belajar PHP dari Nol!</h2>";  
print "Hello world!<br>";  
print "Belajar mecetak teks di PHP!";  
?>
```

Hasilnya:



Apabila kita membuat variabel lalu mengisinya dengan fungsi **print()** seperti ini:

```
$cetak = print("Hello World!");
```

Maka variabel **\$cetak** akan bernilai **1**.

Apabila kita memberikan dua paramater ke dalam fungsi **print()**, maka akan terjadi error.

```
print("Hello", "World"); // <-- ini akan error
```

Percobaan pada console PHP:

```
petanikode@imajinasi ~  
petanikode@imajinasi ~ $ php -a  
Interactive mode enabled  
  
php > $cetak = print("Hello World!");  
Hello World!  
php > echo $cetak;  
1  
php > print("Hello", "World");  
PHP Parse error: syntax error, unexpected ',' in php shell code on line 1  
php > █
```

Menggabungkan String atau Teks di PHP

Pada fungsi **echo()**, kita menggabungkan teks atau string dengan memberikan sebagai argumen (dipisah dengan tanda koma).

```
echo "Ini ", "teks ", "yang ", "dibuat ", "terpisah.";
```

Selain cara ini, kita juga bisa melakukannya dengan tanda titik (.).

Titik adalah operator untuk menggabungkan dua teks di PHP.

Kenapa tidak menggunakan simbol plus (+)?

Pada PHP, simbol plus digunakan untuk penjumlahan, bukan untuk menggabungkan string.

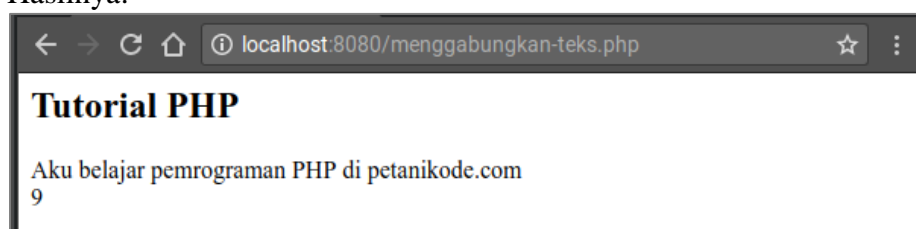
Jadi:

```
echo "1" + "1"; // akan menghasilkan 2  
echo "1" . "1"; // akan menghasilkan 11
```

Contoh:

```
<?php  
$txt1 = "Tutorial PHP";  
$txt2 = "ahmadimuslim";  
$x = 5;  
$y = 4;  
  
echo "<h2>" . $txt1 . "</h2>";  
echo "Dibelajar pemrograman PHP di " . $txt2 . "<br>";  
echo $x + $y;  
?>
```

Hasilnya:



3. Fungsi printf()

Fungsi **printf()** adalah fungsi untuk memformat teks atau string. Fungsi ini akan mengembalikan panjang dari teks saat dieksekusi.

Biasanya saat kita menggunakan fungsi **echo**, kita akan menulis seperti ini:

```
$txt = "ahmadimuslim";  
echo "Dibelajar pemrograman PHP di " . $txt . "<br>";
```

Apabila kita ingin menggunakan fungsi **printf()**, maka kita bisa lakukan seperti ini:

```
$txt = "ahmadimuslim";  
printf("Dibelajar pemrograman PHP di %s<br>", $txt);
```

Simbol **%s** adalah sebuah *placeholder* untuk teks (string). Selain simbol **%s** ada juga simbol:

- **%d** untuk bilangan desimal (integer);
- **%f** untuk pecahan (float);
- **%b** untuk boolean.

Salah satu yang patut kita coba adalah **%f**, karena dengan ini kita bisa mengatur bagaimana bilangan pecahan ditampilkan.

Contoh:

```
// misalkan kita punya bilangan dengan notasi E seperti ini  
$harga = 100000;
```

```
// jika kita cetak dengan echo:  
echo "Harganya adalah Rp $harga";
```

```
// jika kita cetak dengan printf  
printf("Harganya adalah Rp %.2f", $harga);
```

Hasilnya:

```
Harganya adalah Rp 100000  
Harganya adalah Rp 100000.00
```

Simbol **%.2f** artinya kita akan menetak bilangan dengan dua angka di belakang koma.

Mencetak Tanda Kutip

Kadang kita ingin menetak tanda kutip seperti ini:

```
echo 'hari ini hari jum'at';
```

Maka akan terjadi error!

Karena di sana dianggap tutup teksnya berakhir pada teks **jum**.

Lalu bagaimana caranya kita mencetak tanda kutip?

Kita bisa menggunakan tanda *back slash* (****).

Contoh:

```
echo 'hari ini hari jum\'at';
```

Maka akan menghasilkan:

```
hari ini hari jum'at
```

D. Tipe Data Pada PHP

Variabel yang sudah kita buat bisa kita simpan dengan berbagai jenis data.

Jenis-jenis data ini disebut tipe data.

Pada PHP, kita tidak harus mendeklarasikan tipe data secara eksplisit. Karena PHP sudah mampu mengenali tipe data dari nilai yang kita berikan.

Contoh:

```
<?php
```

```
// tipe data char (karakter)  
$jenis_kelamin = 'L';
```

```
// tipe data string (teks)  
$nama_lengkap = "Ahmadi";
```

```
// tipe data integer
```

```

$umur = 20;

// tipe data float
$berat = 48.3;

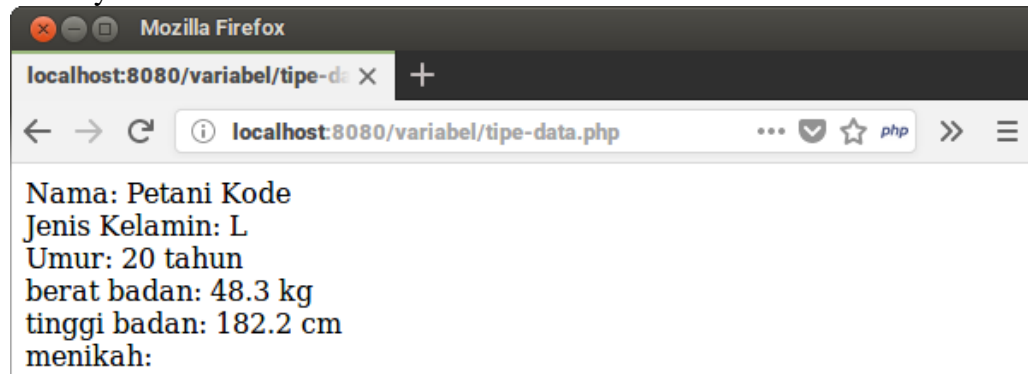
// tipe data float
$tinggi = 182.2;

// tipe data boolean
$menikah = false;

echo "Nama: $nama_lengkap<br>";
echo "Jenis Kelamin: $jenis_kelamin<br>";
echo "Umur: $umur tahun<br>";
echo "berat badan: $berat kg<br>";
echo "tinggi badan: $tinggi cm<br>";
echo "menikah: $menikah";

```

Hasilnya:



Variabel **\$menikah** akan ditampilkan kosong, karena nilai **false** akan dikonversi menjadi kosong dalam string.

Nanti kita akan bahas cara konversi tipe data di PHP.

Sekarang mari kita bahas masing-masing tipe data lebih detail...

Tipe Data Char dan String di PHP

Char adalah tipe data yang terdiri dari karakter. Penulisannya diapit dengan tanda petik satu.

Contoh:

```
$huruf = 'E';
```

Lalu, *String* adalah tipe data yang terdiri dari kumpulan karakter. Penulisannya diapit dengan tanda petik ganda.

Contoh:

```
$alamat = "Jl. Mawar, Jakarta";
```

Tipe Data Integer di PHP

Integer adalah tipe data angka. Penulisannya tidak menggunakan tanda petik.

Contoh:

```
$nilai = 98; // angka positif
```

```
$poin = -31; // angka negatif
```

Tipe Data Float di PHP

Float adalah tipe data bilangan pecahan. Sama seperti integer, tipe data ini ditulis tanpa tanda petik.

Contoh:

```
$panjang = 233.12;
```

```
$kapasitas = 13232.12;
```

Kada juga tipe data float ditulis dalam notasi seperti ini:

```
$jarak = 1.2E-5;
```

E-5 artinya eksponen dari **10**.

Contoh di atas akan sama dengan **1.2×10^{-5}** . Kalau kita jabarkan akan menjadi **0.000012**.

Agar format float tidak tercetak dalam notasi **E**, kita bisa gunakan fungsi **sprintf()**.

Contoh:

```
echo sprintf('%f', $a);  
// batasi angka di belakang koma  
echo sprintf('%.3f', $a);
```

Tipe data Boolean di PHP

Tipe data *boolean* adalah tipe data yang hanya bernilai **true** dan **false**.

Penulisan **true** dan **false** tidak diapit dengan tanda petik.

Contoh:

```
$isActive = false;  
$menikah = true;
```

Tipe Data Array dalam PHP

Array adalah tipe data yang berisi sekumpulan data.

Contoh:

```
$minuman = array("Kopi", "Teh", "Jus Jeruk");  
$makanan = ["Nasi Goreng", "Soto", "Bubur"];
```

Kita akan membahas lebih dalam tentang array di: 7 Hal yang Harus Kamu Ketahui Tentang Array.

Tipe Data Objek di PHP

Tipe data objek adalah tipe data abstrak yang berisi data dan *method*.

Contoh:

```
$user = new User();
```

Tipe data objek lebih sering disebut *instance* dari sebuah class. Pada contoh di atas **User()** adalah class yang di-instance di variabel **\$user**.

Setiap pembuatan *instance* harus didahului dengan kata kunci **new**.

Tipe Data NULL di PHP

Tipe data **NULL** adalah tipe data yang menyatakan kosong.

Artinya: Jika kita mengisi variabel dengan nilai **NULL**, maka variabel tersebut akan dianggap kosong atau tidak punya nilai.

Contoh:

```
$nama = NULL;
```

4. Konversi Tipe data di PHP

Apa yang akan terjadi bila kita melakukan pembagian dengan tipe data teks dengan angka seperti ini:

```
$a = 3;  
$b = "angka 10";
```

```
$c = $b / $a;
```

Tentunya akan terjadi error saat kita ingin mencetak isi variabel **\$c**.

Karena itu, kita harus konversi dulu tipe datanya agar bisa dilakukan operasi yang lain.

Pada PHP konversi tipe data bisa dilakukan dengan operator kali (*).

Contoh:

```
<?php  
$foo = "1"; // mula-mula $foo dalam bentuk string (ASCII 49)  
$foo *= 2; // $foo sekarang adalah integer (2)  
$foo = $foo * 1.3; // $foo sekarang adalah float (2.6)  
$foo = 5 * "10 Little birds"; // $foo sekarang adalah integer (50)  
$foo = 5 * "10 Small birds"; // $foo sekarang adalah integer (50)  
?>
```

Selain cara ini, kita juga bisa melakukannya seperti di bahasa C:

```
<?php
```

```
$a = "32";  
$a = (int) $a; // ubah nilai a menjadi integer  
$a = (float) $a; // ubah nilai a menjadi float  
$a = (string) $a; // ubah nilai a menjadi string
```

```
?>
```

5. Menghapus Variabel dari Memori

Apabila ada variabel yang sudah tidak dibutuhkan lagi, maka kita bisa menghapusnya untuk meningkatkan performa program.

Cara menghapus variabel di PHP dapat menggunakan fungsi **unset()**.

Contoh:

```
// membuat variabel $tmp  
$tmp = 2901;  
  
// menghapus variabel $tmp  
unset($tmp);  
  
// mencoba mengakses variabel $tmp  
echo $tmp;
```

Jika kita eksekusi kode di atas, maka akan terjadi error:

PHP Notice: Undefined variable: tmp

Karena variabel **\$tmp** sudah kita hapus.

Kenapa harus menghapus variabel? nanti juga akan terhapus sendiri saat programnya selesai?

Biasanya saat kita ingin optimasi, variabel-variabel yang tidak terpakai harus dihapus agar tidak membebani memori pada server.

▪ LATIHAN 6

Kerjakan soal-soal berikut dengan baik dan benar!

- 1) Sebutkan 3 fungsi untuk mencetak teks ke layar, dan bagaimana cara menuliskannya....
- 2) Jelaskan perbedaan Fungsi print() dengan echo()....
- 3) Jelaskan Fungsi printf() dalam PHP, dan sebutkan 4 jenis penerapannya (dengan %)...
- 4) Bagaimana jika kita ingin menetak tanda kutip...
- 5) Bagaimana cara membuat variabel bernama harga dengan isi 1000?
- 6) Bagaimana cara mengisi ulang dengan nilai yang baru?
- 7) Jelaskan 4 hal yang harus diperhatikan membuat nama variabel!
- 8) Kapan saatnya Kita Harus Menggunakan Konstanta?
- 9) Jelaskan dua cara membuat konstanta pada PHP!
- 10) Bagaimana cara membuat konstanta bernamasite_name dengan isi Ahmadimuslim;
- 11) Jelaskan Tipe Data berikut ini :
 - a) Char dan String
 - b) Integer
 - c) Float
 - d) Boolean
 - e) Array
 - f) Objek
 - g) NULL
- 12) Mengapa kita harus melakukan konversi tipe data?
- 13) Mengapa kita harus menghapus variabel yang tidak terpakai di PHP? Bagaimana Cara menghapusnya?



MATERI PERTEMUAN 7

7 Jenis Operator dalam PHP



TAHUKAH KAMU...?

Ada 6 Jenis operator dalam pemrograman PHP yang harus kita ketahui:

- Operator Aritmatika;
- Operator Penugasan atau Assignment;
- Operator Increment & Decrement;
- Operator Relasi atau pembandingan;
- Operator Logika;
- Operator Bitwise;
- Operator Ternary.

1. Operator Aritmatika

Operator aritmatika merupakan operator untuk melakukan operasi aritmatika. Operator aritmatika terdiri dari:

| Nama Operator | Simbol |
|----------------|--------|
| 1) Penjumlahan | + |
| 2) Pengurangan | - |
| 3) Perkalian | * |
| 4) Pemangkatan | ** |
| 5) Pembagian | / |
| 6) Sisa Bagi | % |

Contoh:

```
<?php
```

```
$a = 5;
```

```
$b = 2;
```

```
// penjumlahan
```

```
$c = $a + $b;
```

```
echo "$a + $b = $c";
```

```
echo "<hr>";
```

```
// pengurangan
```

```
$c = $a - $b;
```

```
echo "$a - $b = $c";
```

```
echo "<hr>";
```

```
// Perkalian
```

```
$c = $a * $b;
```

```
echo "$a * $b = $c";
```

```
echo "<hr>";
```

```
// Pembagian
```

```
$c = $a / $b;
```

```
echo "$a / $b = $c";
```

```
echo "<hr>";
```

```
// Sisa bagi
```

```
$c = $a % $b;
```

```
echo "$a % $b = $c";
```

```
echo "<hr>";
```

```
// Pangkat
```



```
$c = $a ** $b;
echo "$a ** $b = $c";
echo "<hr>";
```

?>

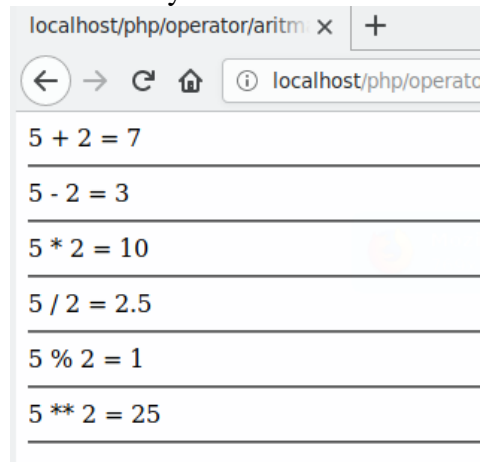
Mula-mula kita punya dua variabel, yaitu **\$a** dan **\$b** dengan nilai awal sebagai berikut:

```
$a = 5;
$b = 2;
```

Kemudian kita menggunakan operator aritmatika untuk melakukan operasi terhadap dua nilai atau variabel tersebut.

Lalu hasilnya disimpan di dalam variabel **\$c**.

Maka hasilnya:



O ya, perhatikan juga simbol-simbol operator yang dipakai.

Pada matematika, perkalian biasanya menggunakan simbol **x**. Namun di dalam pemrograman perkalian menggunakan simbol *****.

```
// salah
$c = 7 x 4;
```

```
// benar
$c = 7 * 4;
```

Lalu untuk operator **%** (modulo), ini adalah operator untuk menghitung sisa bagi.

Misalnya:

```
// kita punya 5
$a = 5;
```

```
// lalu kita bagi dengan 2
$c = $a / 2;
```

Maka variabel **\$c** akan bernilai **1**, karena **5** dibagi **2** sisanya **1**.

Biar lebih mudah, coba bayangkan seperti ini:

Kamu punya permen **5** biji, lalu dibagi berdua dengan adikmu. Biar adil, sama-sama dapat dua biji. Nah, ada sisanya satu yang belum dibagi.

Sisa inilah yang menjadi hasil modulo.

2. Operator Penugasan (Assignment)

Operator berikutnya yang harus kamu ketahui adalah operator penugasan atau *assignment*.

Yap! dari namanya saja sudah bisa ditebak.

Operator ini adalah operator untuk memberikan tugas kepada variabel.

Biasanya digunakan untuk mengisi nilai.

Contoh:

```
$a = 32;
```

Sama dengan **(=)** adalah operator penugasan untuk mengisi nilai.

Selain sama dengan, terdapat juga beberapa operator penugasan seperti:

Nama Operator	Symbol
1) Pengisian Nilai	=
2) Pengisian dan Penambahan	+=
3) Pengisian dan Pengurangan	-=
4) Pengisian dan Perkalian	*=
5) Pengisian dan Pemangkatan	**=
6) Pengisian dan Pembagian	/=
7) Pengisian dan Sisa bagi	%=
8) Pengisian dan Peggabungan (string)	.=

Apa bedanya dengan operator aritmatika?

Operator penugasan digunakan untuk mengisi nilai dan juga menghitung dengan operasi aritmatika. Sedangkan operator aritmatika hanya berfungsi untuk menghitung saja.

Sebagai contoh:

```
$speed = 83;
```

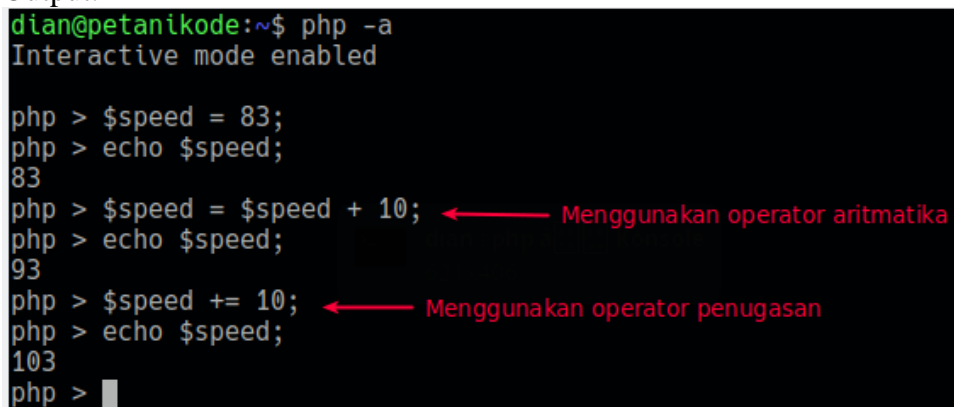
```
// ini operator aritmatika  
$speed = $speed + 10;
```

```
// maka nilai speed akan samadengan 83 + 10 = 93
```

```
// ini operator penugasan  
$speed += 10;
```

```
// sekarang nilai speed akan menjadi 93 + 10 = 103
```

Output:



```
dian@petanikode:~$ php -a  
Interactive mode enabled  
  
php > $speed = 83;  
php > echo $speed;  
83  
php > $speed = $speed + 10; ← Menggunakan operator aritmatika  
php > echo $speed;  
93  
php > $speed += 10; ← Menggunakan operator penugasan  
php > echo $speed;  
103  
php > █
```

Perhatikan operasi ini:

```
$speed = $speed + 10;  
// dan  
$speed += 10;
```

Kedua operasi tersebut merupakan operasi yang sama. Hanya saja yang atas menggunakan operator aritmatika dan yang bawah menggunakan operator penugasan.

Bisa dibilang, operator penugasan adalah bentuk yang lebih sederhana dari ekspresi seperti di atas. Penggunaan operator penugasan akan sering kita temukan saat membuat program.

3. Operator Increment & Decrement

Operator *increment* dan *decrement* merupakan operator yang digunakan untuk menambah **+1** (tambah satu) dan mengurangi **-1** (kurangi dengan satu).

Operator *increment* menggunakan simbol **++**, sedangkan *decrement* menggunakan simbol **--**.

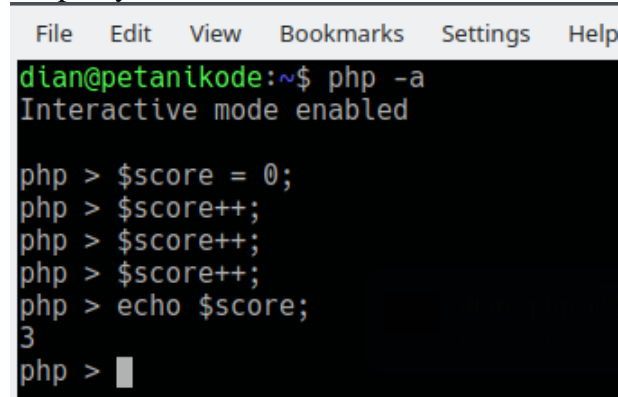
Contoh:

```
$score = 0;

$score++;
$score++;
$score++;
```

```
echo $score;
```

Outputnya:



```
File Edit View Bookmarks Settings Help
dian@petanikode:~$ php -a
Interactive mode enabled

php > $score = 0;
php > $score++;
php > $score++;
php > $score++;
php > echo $score;
3
php > █
```

Nilai **\$score** akan menjadi **3**, karena kita melakukan *increment* sebanyak 3x.

4. Operator Relasi

Operator relasi adalah operator untuk membandingkan dua buah nilai. Hasil operasi dari operator relasi akan menghasilkan nilai dengan tipe data *boolean*, yaitu **true** (benar) dan **false** (salah).

Berikut ini daftar operator relasi:

Nama Operator	Simbol
1) Lebih Besar	>
2) Lebih Kecil	<
3) Sama Dengan	== atau ===
4) Tidak Sama dengan	!= atau !==
5) Lebih Besar Sama dengan	>=
6) Lebih Kecil Sama dengan	<=

Mari kita coba dalam program: **relasi.php**

```
<?php
```

```
$a = 6;
$b = 2;
```

```
// menggunakan operator relasi lebih besar
$c = $a > $b;
echo "$a > $b: $c";
echo "<hr>";
```

```
// menggunakan operator relasi lebih kecil
$c = $a < $b;
echo "$a < $b: $c";
echo "<hr>";
```

```
// menggunakan operator relasi lebih sama dengan
$c = $a == $b;
echo "$a == $b: $c";
echo "<hr>";
```

```
// menggunakan operator relasi lebih tidak sama dengan
$c = $a != $b;
echo "$a != $b: $c";
```

```

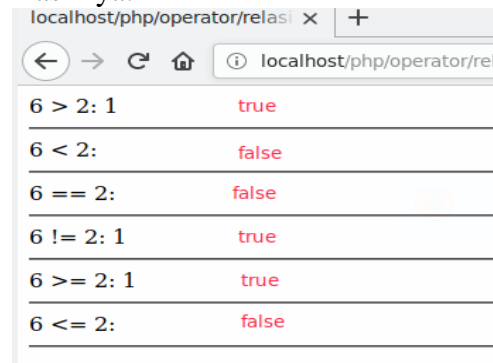
echo "<hr>";

// menggunakan operator relasi lebih besar sama dengan
$c = $a >= $b;
echo "$a >= $b: $c";
echo "<hr>";

// menggunakan operator relasi lebih kecil sama dengan
$c = $a <= $b;
echo "$a <= $b: $c";
echo "<hr>";

```

Hasilnya:



6 > 2: 1	true
6 < 2:	false
6 == 2:	false
6 != 2: 1	true
6 >= 2: 1	true
6 <= 2:	false

Perhatikan!

Di sana kita mendapatkan nilai **1** untuk **true** sedangkan **false** tidak ditampilkan atau **0**. Apakah ini salah?

Tidak, memang seperti itulah sifat dari fungsi **echo** di PHP.

Nilai dengan tipe data boolean biasanya tidak untuk ditampilkan. Biasanya digunakan untuk pembuatan kondisi pada percabangan.

Contohnya seperti ini:

```

<?php
$total_belanja = 150000;

if($total_belanja > 100000){
    echo "Anda dapat hadiah!";
}

```

5. Operator Logika

Jika kamu pernah belajar logika matematika, kamu pasti tidak akan asing dengan operator ini. Operator logika adalah operator untuk melakukan operasi logika seperti AND, OR, dan NOT.

Operator logika terdiri dari:

Nama Operator	Simbol
1) Logika AND	&&
2) Logika OR	
3) Negasi/kebalikan/ NOT	!

Mari kita coba dalam program: **logika.php**

```

<?php

$a = true;
$b = false;

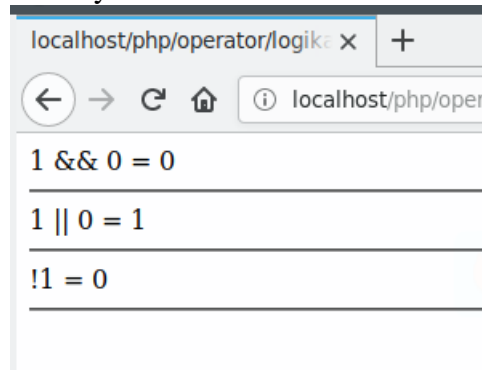
// variabel $c akan bernilai false
$c = $a && $b;
printf("%b && %b = %b", $a,$b,$c);
echo "<hr>";

```

```
// variabel $c akan bernilai true
$c = $a || $b;
printf("%b || %b = %b", $a,$b,$c);
echo "<hr>";

// variabel $c akan bernilai false
$c = !$a;
printf("!%b = %b", $a, $c);
echo "<hr>";
```

Hasilnya:



Perhatikan!

Pada contoh di atas, kita menggunakan fungsi **printf()** untuk mencetak memformat dan mencetak teks.

Namun akan tetap menampilkan **1** untuk **true** dan **0** untuk **false**.

Oprator logika sama seperti operator relasi, ia akan menghasilkan nilai dengan tipe data *boolean*.

Perhatikanlah hasil yang di dapatkan ketika menggunakan operator **&&** (AND), **||** (OR), dan **!** (NOT).

Operator **&&** akan menghasilkan **true** apabila nilai kiri dan kanan bernilai **true**. Sedangkan operator **||** akan menghasilkan **false** saat nilai kiri dan kanan bernilai **false**.

Coba cek kembali hukum logika AND, OR, dan NOT.

AND			OR			NOT	
Hasil			Hasil			Hasil	
true	true	true	true	true	true	true	false
true	false	false	true	false	true	false	true
false	true	false	false	true	true		
false	false	false	false	false	false		

6. Operator Bitwise

Operator bitwise merupakan operator yang digunakan untuk operasi bit (biner).

Operator ini terdiri dari:

Nama	Simbol
AND	&
OR	
XOR	^
Negasi/kebalikan	~
Left Shift	<<
Right Shift	>>

Operator ini berldiuntuk tipe data **int**, **long**, **short**, **char**, dan **byte**.

Operator ini akan menghitung dari bit-ke-bit.

Misalnya, kita punya variabel **a = 60** dan **b = 13**.

Bila dibuat dalam bentuk biner, akan menjadi seperti ini:

a = 00111100

b = 00001101

Kemudian, dilakukan operasi bitwise

Operasi AND

a = 00111100

b = 00001101

a & b = 00001100

Operasi OR

a = 00111100

b = 00001101

a | b = 00111101

Operasi XOR

a = 00111100

b = 00001101

a ^ b = 00110001

Operasi NOT (Negasi/kebalikan)

a = 00111100

~a = 11000011

Konsepnya memang hampir sama dengan operator Logika. Bedanya, Bitwise digunakan untuk biner.

Untuk lebih jelasnya mari kita coba dalam program.

```
<?php
```

```
$a = 60;
```

```
$b = 13;
```

```
// bitwise AND
```

```
$c = $a & $b;
```

```
echo "$a & $b = $c";
```

```
echo "<br>";
```

```
// bitwise OR
```

```
$c = $a | $b;
```

```
echo "$a | $b = $c";
```

```
echo "<br>";
```

```
// bitwise XOR
```

```
$c = $a ^ $b;
```

```
echo "$a ^ $b = $c";
```

```
echo "<br>";
```

```
// Shift Left
```

```
$c = $a << $b;
```

```
echo "$a << $b = $c";
```

```
echo "<br>";
```

```
// Shift Right
```

```
$c = $a >> $b;
```

```
echo "$a >> $b = $c";
```

```
echo "<br>";
```

Hasilnya:

```
localhost/php/operator/bitwise x +
localhost/php
60 & 13 = 12
60 | 13 = 61
60 ^ 13 = 49
60 << 13 = 491520
60 >> 13 = 0
```

7. Operator Ternary

Operator ternary adalah operator untuk membuat sebuah kondisi. Simbol yang digunakan adalah tanda tanya (?) dan titik dua (:).



Pada contoh di atas, “Kamu suka aku” adalah pertanyaan atau kondisi yang akan diperiksa. Kalau jawabannya benar, maka iya. Sebaliknya akan tidak. Untuk lebih jelasnya, mari kita coba...

```
<?php
```

```
$suka = true;
```

```
// menggunakan operator ternary
$jawab = $suka ? "iya" : "tidak";
```

```
// menampilkan jawaban
echo $jawab;
```

Hasilnya:

```
Mozilla Firefox
localhost/php/operator/ternary x +
localhost/php/operator/ternary.php ...
iya
```

Cobalah untuk mengganti nilai variabel **\$suka** menjadi **false**, maka hasil outputnya akan **tidak**.

▪ LATIHAN 7

Kerjakan soal-soal berikut dengan baik dan benar!

- 1) Sebutkan 6 jenis operator aritmatika!
dan Jika diketahui:
\$a = 10;
\$b = 5;
Maka tulikan ekspresi \$c untuk seluruh operasi aritmatika tsb!
- 2) Sebutkan 8 jenis operator penugasan!
dan Jika diketahui:
\$a = 10;
\$b = 5;
Maka tulikan ekspresi \$c untuk seluruh operasi penugasan tsb!
- 3) Jelaskan perbedaan operator penugasan dan operator aritmatika!
- 4) Jelaskan pengertian Operator increment dan decrement
- 5) Jelaskan pengertian Operator relasi.
dan Jika diketahui:
\$a = 10;
\$b = 5;
Maka tulikan ekspresi \$c untuk seluruh 6 operasi relasi tsb!
- 6) Jelaskan pengertian Operator logika.
dan Jika diketahui:
\$a = 10;
\$b = 5;
Maka tulikan ekspresi \$c untuk seluruh 3 operasi logika tsb!
- 7) Jelaskan pengertian Operator bitwise.
dan Jika diketahui:
\$a = 10;
\$b = 5;
Maka tulikan ekspresi \$c untuk seluruh 6 operasi bitwise tsb!



MATERI PERTEMUAN 8

Memahami Percabangan dan Perulangan Pemrograman



TAHUKAH KAMU...?

Ada 6 Jenis percabangan dalam pemrograman PHP yang harus kita ketahui:

- Percabangan If
- Percabangan If/Else
- Percabangan If/Elseif/Else
- Percabangan Switch/Case
- Percabangan dengan Operator Ternary
- Percabangan Bersarang

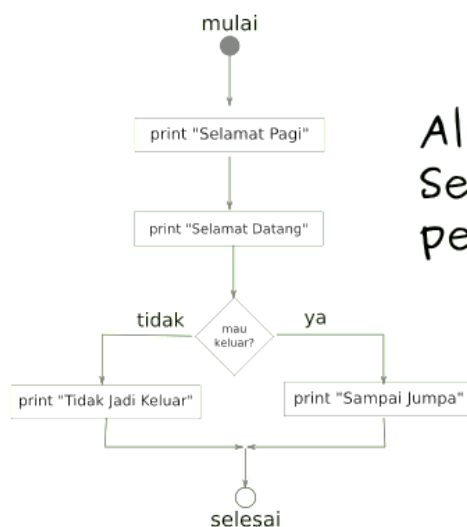
A. Percabangan, Logika Dalam Pemrograman

Percabangan adalah sebutan untuk alur program yang bercabang. Pada diagram alur, kita sering menggambar alur program seperti ini:



Alur Program
Sederhana, tanpa
percabangan

Apabila kita ingin menambahkan percabangan, kita akan membuatnya seperti ini:



Alur Program
Sederhana dengan
percabangan

Pada kesempatan ini, kita akan pelajari tentang percabangan sampai tuntas dan membuat beberapa contoh program.

Percabangan If

Bentuk yang paling sederhana dari percabangan adalah “If” saja. Biasanya digunakan saat hanya ada satu tindakan yang harus dilakukan.

Bentuknya seperti ini:

```
<?php

if (<kondisi>){
    // eksekusi kode ini
}
```

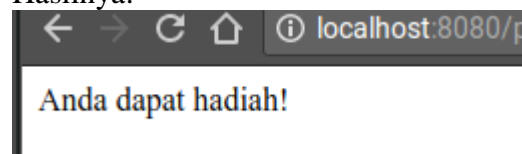
Jika kondisi benar, maka eksekusi kode yang ada di dalamnya. **<kondisi>** bisa kita isi dengan nilai *boolean* atau kita bisa buat pernyataan untuk menghasilkan nilai *boolean*.

Contoh:

```
<?php
$total_belanja = 150000;

if($total_belanja > 100000){
    echo "Anda dapat hadiah!";
}
```

Hasilnya:



Perhatikan contoh di atas!

Teks **Anda dapat hadiah!** hanya akan ditampilkan saat kondisi variabel **\$total_belanja** bernilai di atas **100000**. Kalau di bawah **100000**, tidak akan menampilkan apa-apa.

Kondisi yang digunakan pada contoh adalah:

```
$total_belanja > 100000
```

Kondisi atau pernyataan ini akan bernilai **true** dan **false**. Jika **true** (benar), maka kode yang ada di dalamnya akan dieksekusi. Namun, apabila **false** maka tidak akan mengeksekusinya.

Percabangan If/Else

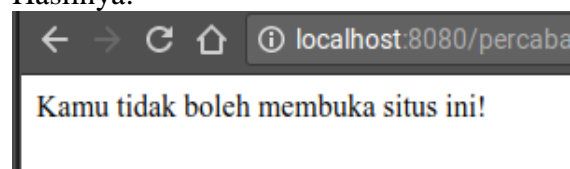
Percabangan If/Else memiliki dua pilihan. Jika **<kondisi>** bernilai **false**, maka blok else akan dikerjakan. Contoh:

```
<?php

$umur = 13;

if ($umur < 18 ){
    echo "Kamu tidak boleh membuka situs ini!";
} else {
    echo "Selamat datang di website kami!";
}
?>
```

Hasilnya:



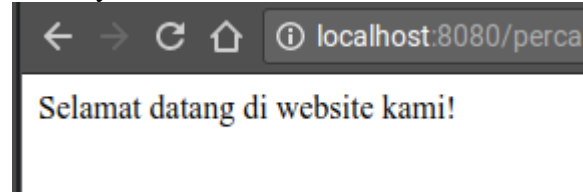
Sekarang coba ubah nilai **\$umur** menjadi **19**:

```
<?php
```

```
$umur = 19;

if ($umur < 18 ){
    echo "Kamu tidak boleh membuka situs ini!";
} else {
    echo "Selamat datang di website kami!";
}
?>
```

Hasilnya:



Percabangan If/Elseif/Else

Percabangan If/Elseif/Else memiliki lebih dari dua pilihan kondisi.

Contoh:

```
<?php
```

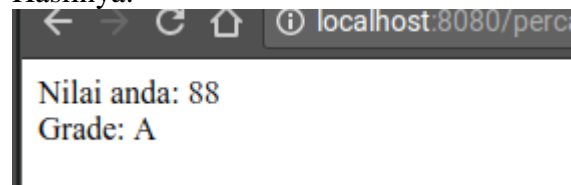
```
$nilai = 88;

if ($nilai > 90) {
    $grade = "A+";
} elseif($nilai > 80){
    $grade = "A";
} elseif($nilai > 70){
    $grade = "B+";
} elseif($nilai > 60){
    $grade = "B";
} elseif($nilai > 50){
    $grade = "C+";
} elseif($nilai > 40){
    $grade = "C";
} elseif($nilai > 30){
    $grade = "D";
} elseif($nilai > 20){
    $grade = "E";
} else {
    $grade = "F";
}

echo "Nilai anda: $nilai<br>";
echo "Grade: $grade";

?>
```

Hasilnya:



Coba ubah variabel **\$nilai** menjadi **54** dan perhatikanlah hasilnya!

Percabangan Switch/Case

Percabangan Switch/Case adalah bentuk lain dari percabangan If/Elseif/Else.

Format penulisannya seperti ini:

```
<?php

switch($variabel){
    case <konidisi>:
        // eksekusi kode ini
        break;
    case <kondisi2>:
        // eksekusi kode ini
        break;
    default:
        // eksekusi kode ini
}

?>
```

Contoh:

```
<?php

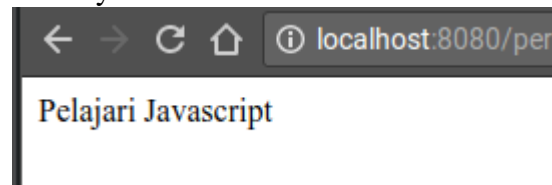
$level = 3;

switch($level){
    case 1:
        echo "Pelajari HTML";
        break;
    case 2:
        echo "Pelajari CSS";
        break;
    case 3:
        echo "Pelajari Javascript";
        break;
    case 4:
        echo "Pelajari PHP";
        break;
    default:
        echo "Kamu bukan programmer!";
}

?>
```

Ada 5 pilihan dalam kondisi di atas. Pilihan **default** akan dipilih apabila nilai variabel **\$level** tidak ada dalam pilihan **case**.

Hasilnya:



Coba ubah nilai variabel **\$level** dan perhatikanlah hasilnya.

Percabangan dengan Operator Ternary

Percabangan menggunakan operator ternary adalah bentuk sederhana dari percabangan If/Else.

Formatnya seperti ini:

```
<?php

<kondisi> ? benar : salah;

?>
```

Contoh:

```
<?php

$suka = true;
$suka ? echo "Dijuga suka kamu": echo "Baiklah!";

?>
```

Atau bisa juga dibuat seperti ini:

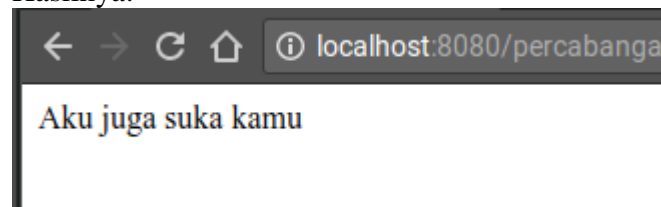
```
<?php

$suka = true;
echo $suka ? "Dijuga suka kamu": "Baiklah!";

?>
```

Artinya: jika variabel **\$suka** bernilai **true** maka cetak "**Dijuga suka kamu**". Tapi kalau bernilai **false**, maka cetak "**Baiklah!**".

Hasilnya:



Percabangan Bersarang

Percabangan bersarang artinya ada percabangan di dalam percabangan (*nested*).

Contoh:

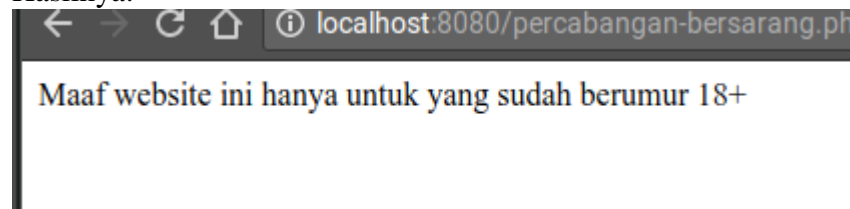
```
<?php

$umur = 17;
$menikah = false;

if($umur > 18){
    if($menikah){
        echo "Selamat datang pak!";
    } else {
        echo "Selamat datang wahai pemuda!";
    }
} else {
    echo "Maaf website ini hanya untuk yang sudah berumur 18+";
}

?>
```

Hasilnya:



B. Perulangan, Iteration dalam Pemrograman

Apa yang akan kamu lakukan bila disuruh membuat daftar judul artikel dengan PHP?

Apakah akan mencetaknya satu per satu dengan perintah **echo** seperti ini:

```
<?php
```

```
echo "<h2>Belajar Pemrograman PHP untuk Pemula</h2>";
echo "<h2>Cara Menggunakan Perulangan di PHP</h2>";
echo "<h2>Memahami Struktur Kondisi IF di PHP</h2>";
echo "<h2>Memahami Perulangan di PHP</h2>";
echo "<h2>Prosedur dan Fungsi di PHP</h2>";
```

```
?>
```

Bisa saja itu dilakukan. Tapi masalahnya, Bagaimana kalau datanya ada 100 atau 1000? Apakah kita mampu mengetik semuanya? Pasti capek. Karena itu, kita harus menggunakan perulangan.

Ada dua jenis perulangan dalam pemrograman: *Counted loop* adalah perulangan yang sudah jelas banyak pengulangannya. Sedangkan *Uncounted loop* tidak pasti berapa kali dia akan mengulang. Manakah diantara keempat perulangan tersebut yang termasuk ke dalam *counted loop* dan *uncounted loop*? Mari kita bahas.

1. Perulangan For

Perulangan *For* adalah perulangan yang termasuk dalam *counted loop*, karena kita bisa menentukan jumlah perulangannya.

Bentuk dasar perulangan for:

```
<?php
```

```
for ($i = 0; $i < 10; $i++){
    // blok kode yang akan diulang di sini!
}
?>
```

Variabel **\$i** dalam perulangan *For* berfungsi sebagai *counter* yang menghitung berapa kali ia akan mengulang.

Hitungan akan dimulai dari nol (0), karena kita memberikan nilai **\$i = 0**.

Lalu, perulangan akan diulang selama nilai **\$i** lebih kecil dari **10**. Artinya, perulangan ini akan mengulang sebanyak **10x**.

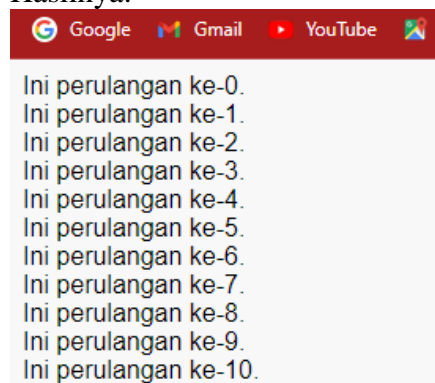
Maksud dari **\$i++** adalah nilai **\$i** akan ditambah **1** disetiap kali melakukan perulangan.

Contoh:

```
<?php
```

```
for($i = 0; $i <= 10; $i++){
    echo "<h2>Ini perulangan ke-$i</h2>";
}
?>
```

Hasilnya:



2. Perulangan While

Perulangan *while* adalah perulangan yang termasuk dalam *uncounted loop*. Karena biasanya digunakan untuk mengulang sesuatu yang belum jelas jumlah pengulangannya.

Namun, perulangan *while* juga bisa digunakan seperti perulangan *for* sebagai *counted loop*.

Bentuk dasarnya:

```
<?php

while (<kondisi>){
    // blok kode yang akan diulang di sini
}
```

```
?>
```

Contoh:

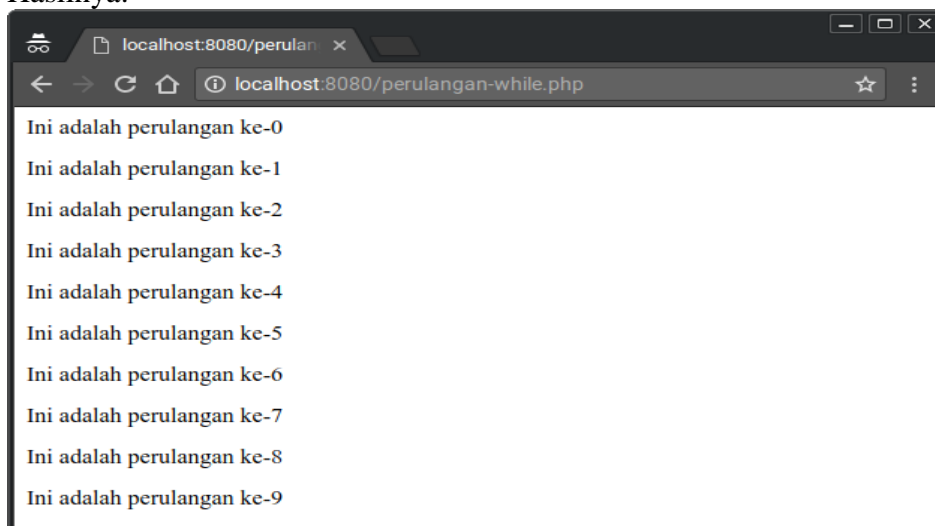
```
<?php

$ulangi = 0;

while($ulangi < 10){
    echo "<p>Ini adalah perulangan ke-$ulangi</p>";
    $ulangi++;
}
```

```
?>
```

Hasilnya:



Perulangan *while* akan terus mengulang selama nilai **\$ulangi** lebih kecil dari **10**.

Lalu di dalam perulangan kita melakukan *increment* nilai **\$ulangi** dengan **\$ulangi++**.

Artinya: Tambah 1 disetiap pengulangan.

Hati-hati, jangan sampai lupa menambahkan *increment*, atau kode yang akan mempengaruhi pengulangan. Karena kalau tidak, pengulangannya tidak akan pernah berhenti dan akan membuat komputer kita *hang*.

3. Perulangan Do/While

Perulangan *Do/While* sama seperti perulangan *while*. Ia juga tergolong dalam *uncounted loop*. Perbedaan *Do/While* dengan *while* terletak pada cara ia memulai pengulangan.

Perulangan *Do/While* akan selalu melakukan pengulangan sebanyak 1 kali, kemudian melakukan pengecekan kondisi.

Sedangkan perulangan *while* akan mengecek kondisi terlebih dahulu, baru melakukan pengulangan.

Bentuk perulangan *Do/While*:

```
<?php
do {
```

```

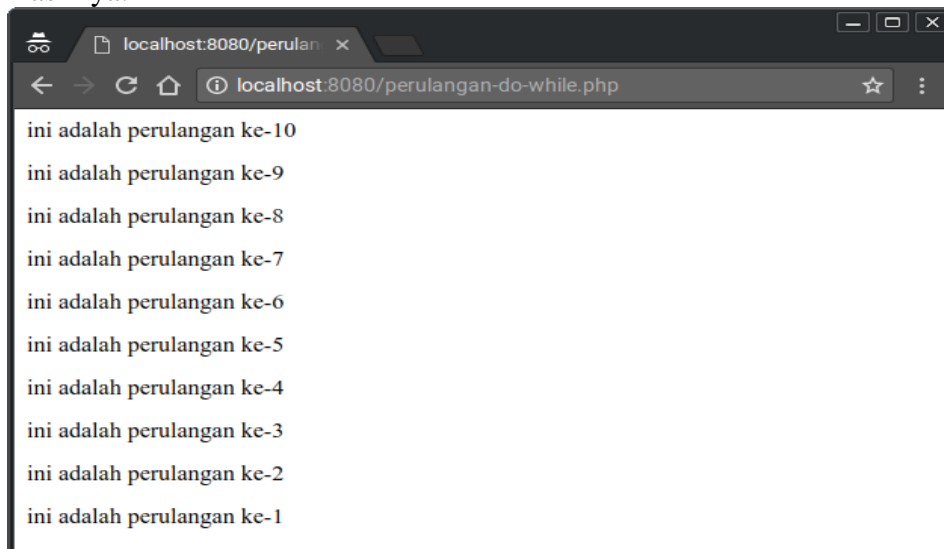
        // blok kode yang akan diulang
    } while (<kondisi>);
?>
Contoh:
<?php

$ulangi = 10;

do {
    echo "<p>ini adalah perulangan ke-$ulangi</p>";
    $ulangi--;
} while ($ulangi > 0);
?>

```

Hasilnya:



4. Perulangan Foreach

Perulangan *foreach* sama seperti perulangan *for*. Namun, ia lebih khusus digunakan untuk mencetak array.

Bentuk perulangan *foreach*:

```

<?php
foreach($array as $data){
    echo $data;
}

```

Contoh:

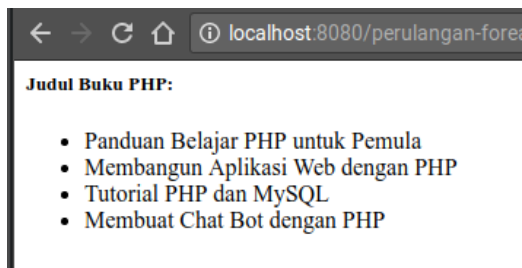
```

<?php
$books = [
    "Panduan Belajar PHP untuk Pemula",
    "Membangun Aplikasi Web dengan PHP",
    "Tutorial PHP dan MySQL",
    "Membuat Chat Bot dengan PHP"
];

echo "<h5>Judul Buku PHP:</h5>";
echo "<ul>";
foreach($books as $buku){
    echo "<li>$buku</li>";
}
echo "</ul>";
?>

```

Hasilnya:



Perulangan Bersarang

Perulangan bersarang adalah istilah untuk menyebut perulangan di dalam perulangan. Dalam bahasa inggris, perulangan bersarang disebut *nested loop*.

Contoh perulangan bersarang:

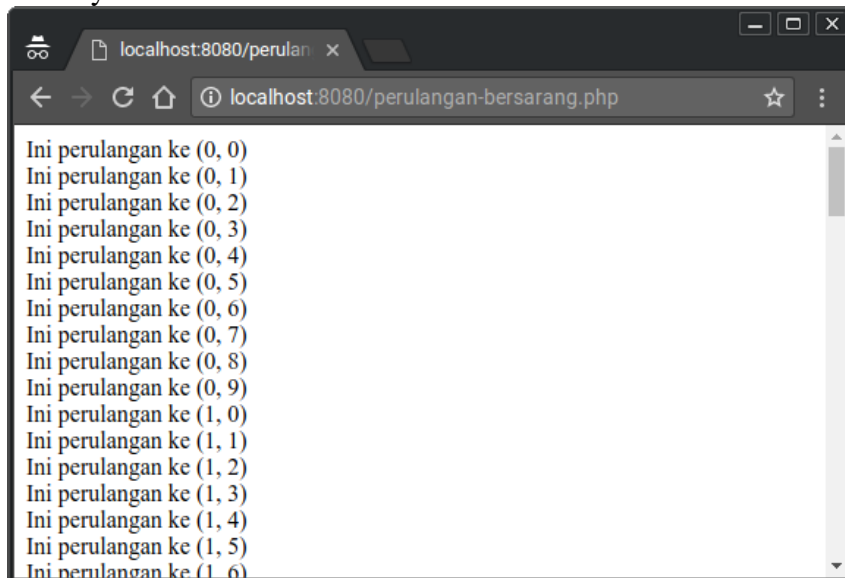
```
<?php
for($i = 0; $i < 5; $i++){
    for($j = 0; $j < 10; $j++){
        echo "Ini perulangan ke ($i, $j)<br>";
    }
}
?>
```

Contoh lain:

```
<?php
$i = 0;
while($i < 10){
    for($j = 0; $j < 10; $j++){
        echo "Ini perulangan ke ($i, $j)<br>";
    }

    $i++;
}
}
```

Hasilnya:



PENALARAN

I. Penulisan For Pada PHP

Perulangan for pada PHP dapat ditulis menggunakan kurung kurawa, colon, atau tanpa keduanya:

```
// Kurung Kurawa, paling umum digunakan
for (ekspresi1; ekspresi2 ; ekspresi3) {
    // kode
}

// Colon
```

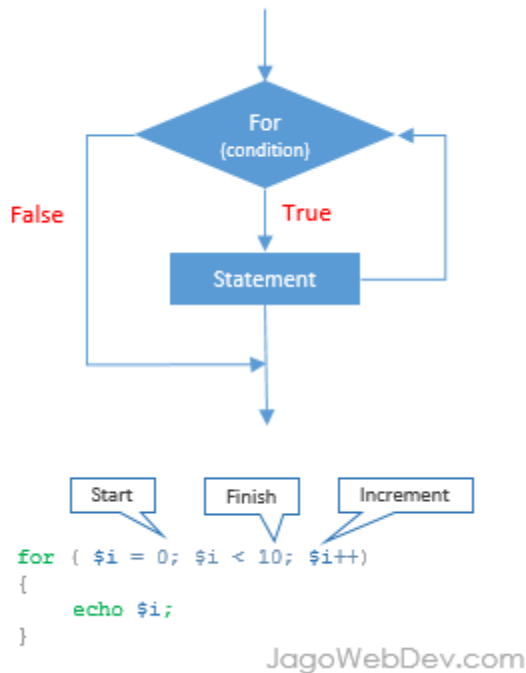
```
for (ekspresi1; ekspresi2 ; ekspresi3) :
    // kode
endfor;
```

```
// Tanpa Keduanya
for (ekspresi1; ekspresi2 ; ekspresi3)
    statement;
```

Dalam menjalankan fungsi loop, PHP akan melakukan eksekusi dengan urutan sebagai berikut:

- PHP akan membaca ekspresi1
- Selanjutnya PHP akan mengevaluasi ekspresi2, jika nilainya TRUE, maka statement di dalam kurung kurawa dijalankan, jika bernilai FALSE maka loop dihentikan.
- Setelah itu PHP akan mengevaluasi atau menjalankan ekspresi3

begitu seterusnya hingga loop selesai, jika digambarkan dalam bentuk flowchart:



Ketentuan mengenai ekspresi

Beberapa ketentuan terkait penulisan ekspresi:

- Semua ekspresi dapat bernilai kosong atau dapat bernilai lebih dari satu dengan pemisah tanda koma
- Semua ekspresi pada ekspresi2 akan di evaluasi, namun untuk menentukan nilai (TRUE atau FALSE – yang menentukan loop berhenti atau berlanjut), digunakan ekspresi yang terakhir.
- Jika ekspresi2 yang bernilai kosong maka loop akan dijalankan terus hingga dihentikan oleh break statemen yang ada di dalam kurung kurawa.

Berikut beberapa contoh penulisan for loop dengan berbagai ekspresi yang semuanya akan mencetak angka 1 s.d 10. (Bentuk 1 merupakan bentuk yang **sering (atau selalu)** dipakai (termasuk saya))

```
<?php
/*
```

Contoh 1, bentuk lengkap, SERING DAN UMUM DIGUNAKAN

```
*/
for ($i = 1; $i <= 10; $i++) {
    echo $i;
}
echo "<hr>";
/*
```

Contoh 2, dengan ekspresi2 kosong, kode dihentikan dengan brak statement

```
*/
for ($i = 1;; $i++) {
    if ($i > 10) {
        break;
    }
}
```

```

        echo $i;
    }
    echo "<hr>";
    /*
Contoh 3, semua ekspresi kosong
    */
    $i = 1;
    for (;;) {
        if ($i > 10) {
            break;
        }
        echo $i;
        $i++;
    }
    echo "<hr>";
    /*
Contoh 4, tanpa statement hanya ekspresi saja
    */
    for ($i = 1; $i <= 10; print $i, $i++);
    echo "<hr>";

```

II. Melompati (skip) For Loop Pada Nilai Tertentu

Pada saat menjalankan loop, terkadang pada kondisi/nilai tertentu, kita ingin melompatinya (skip), untuk keperluan tersebut, kita dapat menggunakan statement continue contoh:

```

<?php
for ($i = 1; $i <= 10; $i++) {
    if ($i == 5) {
        continue;
    }
    echo $i; // 1 2 3 4 6 7 8 9 10
}
?>

```

III. Tips Optimasi Perulangan For Pada PHP

Berikut ini beberapa tips yang dapat kita terapkan agar perulangan for dapat dieksekusi dengan cepat

1. Hindari pemanggilan fungsi dan pendefinisian variabel yang nilainya tetap di dalam loop

Fungsi ini dapat berupa fungsi bawaan PHP seperti count, substr, strlen, dll maupun fungsi yang kita buat sendiri, contoh berikut pengulangan untuk mendapatkan nama bulan:

```

<?php
$bulan = array('1'=>'Januari',
               'Februari',
               'Maret' ,
               'April' ,
               'Mei',
               'Juni',
               'Juli',
               'Agustus',
               'September',
               'Oktober',
               'November',
               'Desember'
               );
$batas_waktu = '2015-11-10';
echo '<table>
    <tr>
        <th>Bulan</th>
        <th>Keterangan</th>
    </tr>';
for ($i = 1; $i <= count($bulan); $i++)
{
    $bln_batas = date("m",strtotime($batas_waktu));
    echo '<tr>

```

```

        <td> ' . strtoupper($bulan[$i]) . ' </td>';

    if ($bln_batas == $i)
        echo '<td>Batas waktu penulisan</td>';
    else
        echo '<td>--</td>';

    echo '</tr>';
}

echo '</table>';
?>

```

output:

Bulan	Keterangan
JANUARI	–
FEBRUARI	–
MARET	–
APRIL	–
MEI	–
JUNI	–
JULI	–
AGUSTUS	–
SEPTEMBER	–
OKTOBER	–
NOVEMBER	Batas waktu penulisan
DESEMBER	–

dari contoh tersebut PHP akan: (1) memanggil fungsi `count($bulan)`, (2) fungsi `date("m",strtotime($batas_waktu))` dan (3) mendefinisikan variabel `$bln_batas` berulang ulang, hal tersebut tidak efisien karena akan memakan waktu dan resource.

Berbeda dengan fungsi `strtoupper` yang memang diperlukan di dalam loop, karena nilainya berubah ubah sesuai nama bulan. Untuk itu, fungsi dan variabel yang nilainya tetap sebaiknya didefinisikan di luar loop, kode dapat ditulis ulang menjadi:

```

<?php
$bln_batas = date("m",strtotime($batas_waktu));
$jml_bln = count($bulan);
for ($i = 1; $i <= $jml_bln, $i++)
{
    // code
}
?>
atau
<?php
$bln_batas = date("m",strtotime($batas_waktu));
for ($i = 1, $jml_bln = count($bulan); $i <= $jml_bln, $i++)
{
    //code
}
?>

```

dari contoh diatas, waktu eksekusi lebih cepat karena fungsi `count` dan `date`, serta pendefinisian variabel `$bln_batas` hanya dijalankan sekali. Dalam kode diatas terdapat `$i++`, kode tersebut merupakan kependekan dari `$i = $i + 1`.

Praktek di lapangan bisa menjadi lebih kompleks, misal dengan contoh diatas, kita akan menambahkan informasi deadline suatu tahapan, misal output yang diinginkan adalah:

Bulan	Deadline	
	Tahapan	Tanggal
JANUARI	Perencanaan	2015-01-31
FEBRUARI	Analisis	2015-02-28
MARET	Perancangan	2015-03-31
APRIL	Penerapan	2015-04-30

MEI	Evaluasi	2015-05-31
JUNI	Penggunaan	2015-06-30
JULI	–	–
AGUSTUS	–	–
SEPTEMBER	–	–
OKTOBER	–	–
NOVEMBER	–	–
DESEMBER	–	–

Kode yang kita gunakan:

```
<?php
$bulan = array('1'=>'Januari',
               'Februari',
               'Maret' ,
               'April' ,
               'Mei',
               'Juni',
               'Juli',
               'Agustus',
               'September',
               'Oktober',
               'November',
               'Desember'
               );

// Informai yang diperoleh dari database
$tahapan = array(
array('tahap' => 'Perencanaan',      'tgl' => '2015-01-31'),
array('tahap' => 'Analisis',        'tgl' => '2015-02-28'),
array('tahap' => 'Perancangan',      'tgl' => '2015-03-31'),
array('tahap' => 'Penerapan',        'tgl' => '2015-04-30'),
array('tahap' => 'Evaluasi',         'tgl' => '2015-05-31'),
array('tahap' => 'Penggunaan',      'tgl' => '2015-06-30')
);

echo '<table>
    <tr>
        <th rowspan="2">Bulan</th>
        <th colspan="2">Deadline</th>
    </tr>
    <tr>
        <th>Tahapan</th>
        <th>Tanggal</th>
    </tr>

    ';

$jumlah_bulan =
for ($i = 1; $i <= count($bulan); $i++)
{
    echo '<tr>
        <td> ' . strtoupper($bulan[$i]) . ' </td>';

        $data_tahapan = false;
        foreach ($tahapan as $tahap)
        {
            $bulan_batas = date("m",strtotime($tahap['tgl']));
            if ($bulan_batas == $i) {
echo '<td>'.$tahap['tahap'].'</td>
        <td>'.$tahap['tgl'].'</td>';
            $data_tahapan = true;
        }
    }
}
```

```

        if (!$data_tahapan)
echo '<td>-</td>'
        <td>-</td>';

        echo '</tr>';
}
echo '</table>';
?>

```

dari data diatas terdapat pengulangan fungsi date yaitu sebanyak 72 kali (12 x 6), dengan struktur data seperti diatas, agak ribet jika harus memenuhi kondisi ideal seperti contoh sebelumnya.

Kondisi tersebut dapat dipenuhi, namun kode yang ditulis bisa jadi menjadi lebih kompleks dan membutuhkan tenaga yang lebih untuk memahaminya (tergantung kondisi lapangan).

Pada kondisi ini, kondisi ideal dapat dilanggar jika kode yang ditulis menjadi lebih sederhana dan mudah dipahami dan performa aplikasi juga tidak terganggu.

Namun jika tidak ada salahnya kita mencobanya. Dengan sedikit perubahan, kode diatas dapat kita tulis kembali menjadi:

```

<?php
$bulan = array('1'=>'Januari',
                'Februari',
                'Maret' ,
                'April' ,
                'Mei',
                'Juni',
                'Juli',
                'Agustus',
                'September',
                'Oktober',
                'November',
                'Desember'
                );

// Informai yang diperoleh dari database
$tahapan = array(
array('tahap' => 'Perencanaan',      'tgl' => '2015-01-31'),
array('tahap' => 'Analisis',         'tgl' => '2015-02-28'),
array('tahap' => 'Perancangan',      'tgl' => '2015-03-31'),
array('tahap' => 'Penerapan',        'tgl' => '2015-04-30'),
array('tahap' => 'Evaluasi',         'tgl' => '2015-05-31'),
array('tahap' => 'Penggunaan',       'tgl' => '2015-06-30')
);

foreach ($tahapan as $key => $tahap)
{
    $bln_batas = date("n",strtotime($tahap['tgl']));
    $ref_tahapan[$bln_batas] = $key;
}
echo '<table>'
    <tr>
        <th rowspan="2">Bulan</th>
        <th colspan="2">Deadline</th>
    </tr>
    <tr>
        <th>Tahapan</th>
        <th>Tanggal</th>
    </tr>

    ';

$jml_bln = count($bulan);
for ($i = 1; $i <= $jml_bln; $i++)
{
    echo '<tr>'
        <td> ' . strtoupper($bulan[$i]) . ' </td>';

```

```

                if (key_exists($i, $ref_tahapan))
                {
echo '<td>'. $tahap[ $ref_tahapan[ $i ] ]['tahap'] . '</td>'
                <td>'. $tahap[ $ref_tahapan[ $i ] ]['tgl'] . '</td>';
                }
                else
                {
echo '<td>-</td>'
                <td>-</td>';
                }

                echo '</tr>';
        }
echo '</table>';
?>

```

pada kode diatas line 26 s.d 30 kita membuat variabel baru bernama \$ref_tahapan yang berbentuk array dengan key bulan dan value index dari array tahapan, contoh \$ref_tahapan[1] = 0 yang berarti bulan 1 merujuk ke \$tahap[0].

Kode diatas tidak terlalu kompleks dan masih wajar untuk digunakan walaupun pengguna kode membuat kita harus menambah tenaga untuk memahami variabel baru \$ref_tahapan

Ketika di tes, kode diatas membutuhkan waktu eksekusi 0.0011389255523682 detik sedangkan sebelumnya 0.014470100402832 detik, tidak terasa perbedaannya oleh karena itu kita dapat menggunakan kedua cara diatas, namun jika kode yang ditulis kompleks, cara kedua bisa dipertimbangkan untuk digunakan.

2Hindari eksekusi perintah SQL di dalam loop

Sebisanya mungkin jangan pernah melakukan eksekusi kode SQL di dalam loop, dengan eksekusi yang berulang ulang maka akan memberatkan server database yang pada akhirnya akan menurunkan performa aplikasi anda.

Contoh dibawah ini pengulangan eksekusi MySQL di dalam loop (variabel \$bulan menggunakan contoh sebelumnya):

```

$jml_bln = count($bulan);
for ($i = 1; $i <= $jml_bln, $i++) {
    $bln = substr('0'.$i, -2);
    $sql = 'SELECT jml_byr FROM penjualan WHERE MONTH(tgl_byr) = $bln;
    $stmt = $pdo->prepare($sql);
    $stmt->execute();
    $penjualan[$bln] = $stmt->fetchAll(PDO::FETCH_ASSOC);
}

```

Contoh diatas akan mengeksekusi perintah SQL sebanyak 12 kali yang tentu saja akan memberatkan, terlebih lagi jika datanya sangat besar.

3Berhati – hati dalam penulisan nested loop

Terkadang kita menuliskan banyak loop di dalam loop, untuk kehati-hatian, gunakan variabel yang mencerminkan kondisi yang ada, tidak sekedar \$i, mengingat nilai variabel akan berubah jika kita mendefinisikan dengan nama yang sama (baik sengaja maupun tidak), contoh berikut akan menghasilkan

```

*
**
***
****
*****
<?php
for ($row = 1; $row <= 5; $row++)
{
    for ($col = 1; $col <= $row; $col++)
    {
        echo '*' . '<br/>';
    }
}
?>

```

KASUS

Cara Membuat Looping For di dalam Tabel

Script HTML di bawah ini digunakan untuk menampilkan bentuk table yang akan kita buat nantinya, dari table sederhana inilah kita akan membuat sebuah looping for.

```
<!DOCTYPE html>
<html>
<head>
    <title>Cara Membuat Looping For di dalam Tabel</title>
</head>
<body>
    <h2>Cara Membuat Looping For di dalam Tabel</h2>
    <form>
        <table border="1" cellspacing="0">
            <tr>
                <th>NO</th>
                <th>BUAH</th>
                <th>SAYUR</th>
            </tr>
            <tr>
                <td></td>
                <td></td>
                <td></td>
            </tr>
        </table>
    </form>
</body>
</html>
```

Script CSS untuk membuat tampilan table di atas lebih menarik.

```
<style>
    table{width:300px; text-align:center; margin:auto;}
    table th { background-color: #95a5a6; }
    h2 {text-align:center; font-style:italic; font-weight:bold;}
</style>
```

Nah pada contoh Ini kita akan memanfaatkan sebuah data di dalam table, langsung saja lihat scriptnya di bawah ini.

```
<form>
    <table border="1" cellspacing="0">
        <tr>
            <th>NO</th>
            <th>BUAH</th>
            <th>SAYUR</th>
        </tr>
        <tr>
            <td><?php echo $no; ?></td>
            <td><?php echo $i; ?></td>
            <td><?php echo $a; ?></td>
        </tr>
    </table>
</form>
```


</form>

Cara Membuat Looping For di dalam Tabel

NO	BUAH	SAYUR
1	10	100
2	20	200
3	30	300
4	40	400
5	50	500
6	60	600
7	70	700
8	80	800
9	90	900
10	100	1000

Cara Membuat Tabel dengan perulangan

```
<?php
echo "<table border=1>";
for($i=1; $i <=3; $i++){
    echo "<tr>";
    for($j=1; $j<=5; $j++){
        echo "<td>";
        echo $i.$j;
        echo "</td>";
    }
}
echo "</table>";
?>
```

Cara Kerja:

Kali ini kita akan membuat table dengan ukuran 3 baris 5 kolom. Di perulangan pertama kita akan membuat kotak baris dengan <tr>. Diperulangan kedua akan kita pecah sebanyak 5 kolom dengan <td>. Di dalam kotak akan berisi nilai dengan skrip \$i.\$j sesuai dengan urutan matrik kotak.

▪ LATIHAN 8

Kerjakan soal-soal berikut dengan baik dan benar!

- 1) Jelaskan pengertian percabangan pemrograman.
dan Jika diketahui:
`$nilai = nilai_akhir;`
`kkm = 76;`
Maka tuliskan sourcecode kondisi kelulusan dengan ekspresi "lulus" dan "mengulang" untuk percabangan berikut!
 - percabangan if
 - percabangan if/else
 - percabangan if,elseif/else
 - percabangan swicth case
 - percabangan bersarang
- 2) Jelaskan pengertian *counted loop* dan *uncounted loop*.
dan Jika diketahui:
`$i = 5;`
ekspresi = "ujian perulangan pak mahdi";
Maka tuliskan sourcecode untuk perulangan berikut!
 - Perulangan *For*
 - Perulangan *while*
 - Perulangan *Do/While*
 - Perulangan *foreach*
 - Perulangan bersarang



MATERI PERTEMUAN 9

Array Dalam Pemrograman PHP



TAHUKAH KAMU...?

Pada kesempatan ini, kita akan membahas:

- Apa itu Array?
- Cara membuat Array di PHP dan Mengisinya
- Cara menampilkan nilai Array
- Cara Menghapus isi Array
- Cara Menambah isi Array
- Array Asosiatif
- Array Multidimensi

Bayangkan sekarang kita sedang membuat aplikasi web, lalu ingin menampilkan daftar nama-nama produk.

Bisa saja kita buat seperti ini:

```
<?php
$produk1 = "Modem";
$produk2 = "Hardisk";
$produk3 = "Flashdisk";

echo "$produk1<br>";
echo "$produk2<br>";
echo "$produk3<br>";
```

Apakah boleh seperti ini?

Boleh-boleh saja. Tapi kurang efektif. Kenapa? Bagaimana kalau ada 100 produk, apakah kita akan membuat variabel sebanyak 100 dan melakukan **echo** sebanyak 100x? oleh karena itu kita akan menggunakan array agar lebih efisien

1. Apa itu Array?

Array adalah salah satu struktur data yang berisi sekumpulan data dan memiliki indeks. Indeks digunakan untuk mengakses nilai array (**Array Indexed**).

Indeks array selalu dimulai dari nol (0).

Contoh:

"Hardisk 2TB"	"Flashdisk 32GB"	"Modem"
0	1	2

Jadi, apabila kita ingin menampilkan "Hardisk 2TB", maka kita harus mengambil indeks yang ke-0.

Array menggunakan nomor sebagai identitasnya (Index) dan dimulai dengan nomor 0.

```
<?php
$nama_variabel=array("isi variabel1","isi variabel2","isi variabel3");
echo "Variabel
".$nama_variabel[0].", ".$nama_variabel[1].", ".$nama_variabel[2]."!";
?>
```

Berikut ini merupakan fungsi - fungsi yang berhubungan dengan array pada bahasa pemrograman PHP :

- **arsort()**. Pengurutan berdasarkan value secara descending.
- **asort()**. Pengurutan berdasarkan value secara ascending.
- **krsort()**. Pengurutan berdasarkan index/key secara descending
- **ksort()**. Pengurutan berdasarkan index/key secara ascending.
- **rsort()**. Pengurutan berdasarkan value secara descending dengan mengubah index/key.
- **sort()**. Pengurutan berdasarkan value secara ascending dengan mengubah index/key.
- **shuffle()**. Random pengurutan array.
- **current()**. Mendapatkan element array yang ditunjuk oleh pointer.
- **end()**. Pointer menunjuk pada element array terakhir.
- **key()**. Mendapatkan key yang ditunjuk oleh pointer.
- **next()**. Pointer menunjuk pada element selanjutnya.
- **prev()**. Pointer menunjuk pada element sebelumnya.
- **reset()**. Memindahkan pointer ke array awal (element pertama).
- **count()**. Menghitung jumlah element array.
- **array_search()**. Mencari posisi key berdasarkan value ke dalam array.
- **array_key_exists()**. memeriksa suatu key didalam array.
- **in_array()**. Memeriksa suatu element kedalam array.

2. Membuat Array di PHP

Array di PHP dapat kita buat dengan 3 bentuk fungsi, dengan kurung biasa **array()** dan tanda kurung kotak **[]**.

Contoh:

- 1) Membuat array kosong

```
$buah = array();  
$hobi = [];
```
- 2) Membuat array sekaligus mengisinya

```
$minuman = array("Kopi", "Teh", "Jus Jeruk");  
$makanan = ["Nasi Goreng", "Soto", "Bubur"];
```
- 3) Membuat array dengan mengisi indeks tertentu

```
$anggota[1] = "Dian";  
$anggota[2] = "Muhar";  
$anggota[0] = "Ahmadi";
```

Cukup mudah bukan.

Oya, array dapat kita isi dengan tipe data apa saja. Bahkan dicampur juga boleh.

Contoh:

```
<?php  
  
$item = ["Bunga", 123, 39.12, true];
```

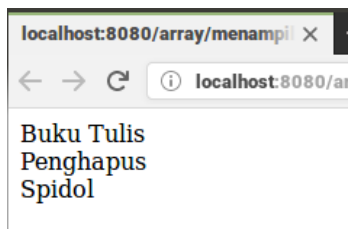
3. Menampilkan isi Array

Untuk menampilkan isi array, kita bisa mengaksesnya melalui indeks.

Contoh:

```
<?php  
// membuat array  
$barang = ["Buku Tulis", "Penghapus", "Spidol"];  
  
// menampilkan isi array  
echo $barang[0]."<br>";  
echo $barang[1]."<br>";  
echo $barang[2]."<br>";
```

Hasilnya:



Tapi cara ini kurang efektif, karena kita mencetak satu per satu. Nanti kalau datanya ada 1000, berarti harus ngetik perintah **echo** sebanyak 1000.

Lalu bagaimana kah?

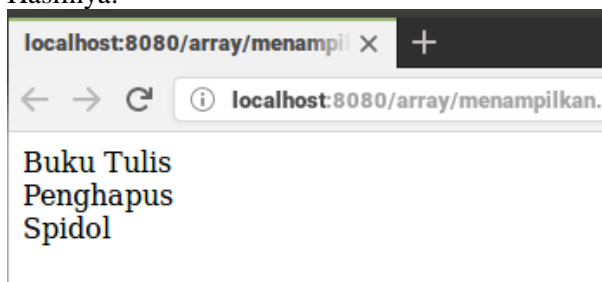
Biasanya kita menggunakan perulangan.

Contoh:

```
<?php
// membuat array
$barang = ["Buku Tulis", "Penghapus", "Spidol"];
// menampilkan isi array dengan perulangan for
for($i=0; $i < count($barang); $i++){
    echo $barang[$i]."<br>";
}
```

Kita bisa menggunakan fungsi **count()** untuk menghitung banyaknya isi array. Pada contoh di atas isi array sebanyak 3, maka perulangan akan dilakukan sebanyak 3x.

Hasilnya:



Selain menggunakan perulangan **for**, kita juga bisa menggunakan perulangan **while** dan **foreach**.

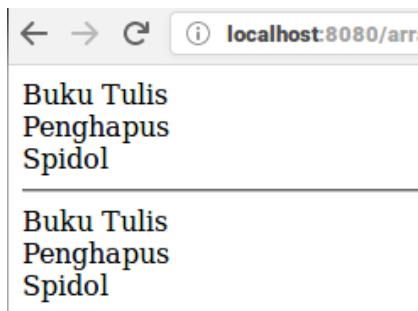
Contoh:

```
<?php

// membuat array
$barang = ["Buku Tulis", "Penghapus", "Spidol"];

// menampilkan isi array dengan perulangan foreach
foreach($barang as $isi){
    echo $isi."<br>";
}
echo "<hr>";
// menampilkan isi array dengan perulangan while
$i = 0;
while($i < count($barang)){
    echo $barang[$i]."<br>";
    $i++;
}
```

Hasilnya:



4. Menghapus isi Array

Untuk menghapus isi array, kita bisa menggunakan fungsi **unset()**. Fungsi ini juga dapat digunakan untuk menghapus variabel.

Contoh:

```
<?php

// membuat array
$hewan = [
    "Burung",
    "Kucing",
    "Ikan"
];

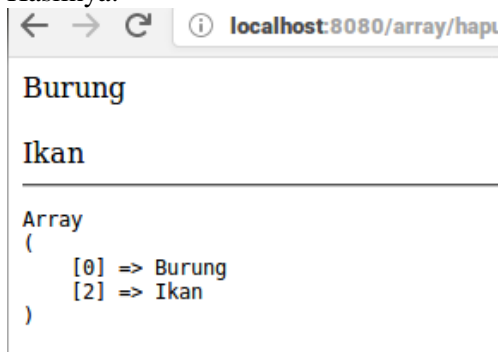
// menghapus kucing
unset($hewan[1]);

echo $hewan[0]."<br>";
echo $hewan[1]."<br>";
echo $hewan[2]."<br>";

echo "<hr>";

echo "<pre>";
print_r($hewan);
echo "</pre>";
```

Hasilnya:



Pada contoh di atas, Kita menggunakan fungsi **print_r()** untuk menampilkan array secara mentah (*raw*). Biasanya fungsi ini digunakan untuk *debugging*.

5. Menambahkan isi Array

Ada dua cara yang bisa dilakukan untuk menambah isi array:

1. Mengisi langsung ke nomer indeks yang ingin ditambahkan
2. Mengisi langsung ke indeks terakhir

Mari kita coba kedua-duanya.

```
<?php
// membuat array
$hobi = [
    "Membaca",
    "Menulis",
```

```

        "Ngeblog"
    ];

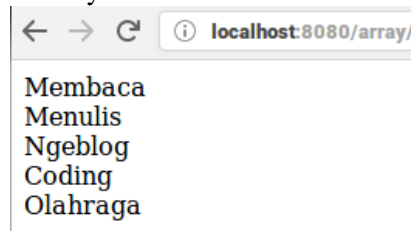
    // menambahkan isi pada idenks ke-3
    $hobi[3] = "Coding";

    // menambahkan isi pada indeks terakhir
    $hobi[] = "Olahraga";

    // cetak array dengan perulangan
    foreach($hobi as $hobiku){
        echo $hobiku."<br>";
    }
    ?>

```

Hasilnya:



Apabila kita menambahkan pada indeks yang sudah memiliki isi, maka isinya akan ditindih dengan yang baru.

Contoh:

```

<?php
// membuat array
$user = [
    "dian",
    "muhar",
    "ahmadimuslim"
];

// mengisi array pada indek ke-1 ("muhar")
$user[1] = "Ahmadi";

// mencetak isi array
echo "<pre>";
print_r($user);
echo "</pre>";
?>

```

Hasilnya:

```

Array
(
    [0] => dian
    [1] =>Ahmadi
    [2] =>ahmadimuslim
)

```

6. Array Asosiatif

Array asosiatif adalah array yang indeksnya tidak menggunakan nomer atau angka. Indeks array asosiatif berbentuk kata kunci.

Contoh:

```

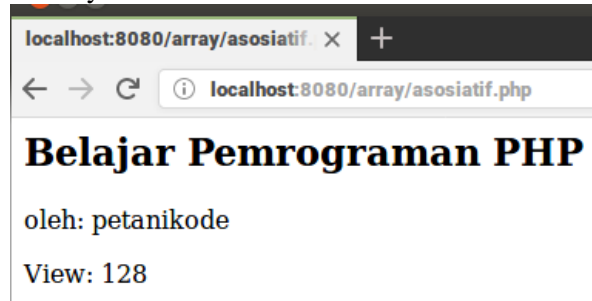
<?php
// membuat array asosiatif
$artikel = [
    "judul" => "Belajar Pemrograman PHP",
    "penulis" => "ahmadimuslim",
    "view" => 128
]

```

```
];

// mencetak isi array asosiatif
echo "<h2>".$artikel["judul"]."</h2>";
echo "<p>oleh: ".$artikel["penulis"]."</p>";
echo "<p>View: ".$artikel["view"]."</p>";
```

Hasilnya:



Pada array asosiatif, kita menggunakan tanda => untuk mengasosiasikan sebuah kata kunci dengan isi array. Selain menggunakan tanda =>, kita juga bisa membuat array asosiatif seperti ini:

```
<?php
$email["subjek"] = "Apa Kabar?";
$email["pengirim"] = "dian@ahmadimuslim.com";
$email["isi"] = "Apa kabar? sudah lama tidak berjumpa";

echo "<pre>";
print_r($email);
echo "</pre>";
```

Hasilnya:

```
Array
(
    [subjek] => Apa Kabar?
    [pengirim] => dian@ahmadimuslim
    [isi] => Apa kabar? sudah lama tidak berjumpa
)
```

Jika array indexed menggunakan nomor sebagai identitasnya dan dimulai dengan nomor 0 maka array ini adalah dengan menggunakan penamaan, untuk lebih jelasnya lihat contoh

```
<?php
$nama_variabel=array("nama1"=>"70","nama2"=>"67","nama3"=>"89");
echo "hasil dari program " . $nama_variabel['nama1'] . " nilai.";
?>
```

Key, Value, Indexed dan Associative Array

Sebelum kita membahas foreach pada PHP, penting untuk memahami tentang key dan value pada array, karena hal tersebut akan lebih mempermudah kita memahami perulangan foreach.

Seperti telah dijelaskan pada artikel Memahami Array Pada PHP, indexed array merupakan array dengan key berupa angka, misal:

```
$bulan = array('Januari', 'Februari', 'Maret')
```

Pada contoh diatas, value dari array adalah Januari, Februari, dan Maret, sedangkan key nya adalah 0, 1, dan 2. Karena key tidak didefinisikan, maka nilai key otomatis mulai dari 0

bentuk diatas sama dengan:

```
array(0=>'Januari',1=>'Februari',3=>'Maret')
```

sedangkan associative array merupakan array dengan key berupa nilai tertentu, misal:

```
array('jenis' => 'Mobil', 'merk' => 'Toyota', 'tipe' => 'Vios')
```

Pada contoh diatas, value dari array adalah Mobil, Toyota, dan Vios. Sedangkan key-nya adalah jenis, merk, dan tipe.

I. Cara Penulisan Foreach Pada PHP

Setelah kita faham tentang key dan value pada array, kita

Pada PHP foreach dapat ditulis dalam dua cara yaitu:

1. Mengabaikan nilai key

Pada cara ini, nilai key akan diabaikan/tidak digunakan:

```
foreach ($array as $value) {  
    statement;  
}
```

Penjelasan:

- \$array adalah nama variabel array yang akan kita gunakan untuk perulangan.
- \$value adalah nama variabel yang mewakili nilai/data dari array yang ada pada variabel \$array. Kita bebas memberi nama variabel ini, umumnya variabel tersebut diberi nama \$value, \$val atau cukup \$v.

Cara ini bermanfaat jika kita hanya ingin menggunakan data value dari array dan mengabaikan data key nya

Contoh penggunaan:

```
$bulan = array ('Januari', 'Februari', 'Maret');  
foreach ($bulan as $nama_bulan) {  
    echo $nama_bulan . '<br/>';  
}
```

Hasil yang kita peroleh:

```
Januari  
Februari  
Maret
```

2. Menyertakan nilai key

Model kedua adalah dengan menyertakan nilai key, cara ini berlaku baik untuk indexed array maupun associative array. Adapun format penulisannya adalah sebagai berikut:

```
foreach ($array as $key => $value) {  
    statement;  
}
```

Penjelasan:

- \$array adalah nama variabel array yang akan kita gunakan untuk perulangan.
- \$key merupakan nama variabel yang mewakili nilai index yang ada di dalam variabel \$array. Kita bebas memberi nama variabel ini, umumnya variabel tersebut diberi nama \$key atau cukup \$k
- Tanda => digunakan untuk menghubungkan antara \$key dan \$value nya
- Seperti pada model pertama, \$value adalah nama variabel yang mewakili data value yang ada di dalam variabel \$array. Kita juga bebas memberi nama variabel ini, umumnya nama yang digunakan adalah \$value, \$val, atau cukup \$v

Contoh penggunaan pada indexed array:

```
$bulan = array ('Januari', 'Februari', 'Maret');  
foreach ($bulan as $index => $nama_bulan) {  
    echo ($index + 1) . '. ' . $nama_bulan . '<br/>';  
}
```

Hasil yang kita peroleh:

```
1. Januari  
2. Februari  
3. Maret
```

Pada contoh diatas terlihat bahwa setiap array pasti memiliki key, sehingga meskipun key tersebut tidak kita tulis, key tersebut tetap ada dan bisa digunakan.

Contoh pada associative array:

```
$kendaraan = array('jenis' => 'Mobil', 'merk' => 'Toyota', 'tipe' => 'Vios');  
foreach ($kendaraan as $key => $val) {  
    echo ucfirst($key) . ': ' . $val . '<br/>';  
}
```

Hasil yang kita peroleh:

```
Jenis: Mobil  
Merk: Toyota  
Tipe: Vios
```

Pada contoh diatas, saya menggunakan fungsi ucfirst untuk membuat huruf pertama dari key menjadi huruf besar.

3. Foreach didalam foreach (nested foreach)

Pada multidimensional array, kita akan sering membuat perulangan foreach didalam foreach.

Hati hati jika membuat perulangan dengan model seperti ini, karena sifat variabel dapat berubah ubah, maka, nilai variabel dari \$key dan \$value juga rawan berubah.

Oleh karena itu, sebisa mungkin memberi nama variabel untuk \$key dan \$value sesuai dengan isi datanya, sehingga memudahkan kita untuk membaca alur dari perulangan.

Contoh:

```
foreach ($array as $key => $val) {  
    statement;  
    foreach ($val as $key => $val) {  
        statement;  
    }  
}
```

Pada contoh tersebut nilai variabel \$key pada foreach yang pertama akan berubah karena ditimpa oleh nilai variabel \$key pada foreach ke dua.

II. Contoh Penggunaan Foreach Pada PHP

Pada PHP, foreach dapat digunakan untuk berbagai keperluan baik untuk backend maupun frontend, pada frontend, foreach digunakan salahsatunya untuk membuat element dropdown select.

Contoh kita ingin membuat dropdown nama bulan, dibanding menggunakan cara manual, akan jauh lebih efisien jika menggunakan perulangan foreach:

```
<?php  
$bulan = array (1=>'Januari', 'Februari', 'Maret', 'April', 'Mei', 'Juni',  
'Juli', 'Agustus', 'September', 'Oktober', 'November', 'Desember');  
  
$opsi_bulan = '<select name="bulan">';  
foreach ($bulan as $key => $value) {  
    $opsi_bulan .= '<option value="" . $key . ">' . $value . '</option>'  
    . "\r\n";  
}  
$opsi_bulan .= '</select>';
```

```
echo $opsi_bulan;
```

Kode HTML yang kita peroleh:

```
<select name="bulan">  
<option value="1">Januari</option>  
<option value="2">Februari</option>  
<option value="3">Maret</option>  
<option value="4">April</option>  
<option value="5">Mei</option>  
<option value="6">Juni</option>  
<option value="7">Juli</option>  
<option value="8">Agustus</option>  
<option value="9">September</option>  
<option value="10">Oktober</option>  
<option value="11">November</option>  
<option value="12">Desember</option>  
</select>
```

Output:



Contoh lain adalah untuk membuat tabel HTML

```
$no = 1;  
$tabel = '  
<table>  
    <tr>  
        <th>No</th>  
        <th>Bulan</th>  
        <th>Penjualan</th>  
    </tr>';  
foreach ($sales as $bulan => $nilai) {  
    $tabel .= '
```

```

        <tr>
            <td>' . $no . '</td>
            <td>' . $bulan . '</td>
            <td>' . $nilai . '</td>
        </tr>';
        $no++;
    }
    $tabel .= '</table>';
    echo $tabel;

```

Hasil:

```

<table>
    <tr>
        <th>No</th>
        <th>Bulan</th>
        <th>Penjualan</th>
    </tr>
    <tr>
        <td>1</td>
        <td>Januari</td>
        <td>5.500</td>
    </tr>
    <tr>
        <td>2</td>
        <td>Februari</td>
        <td>7.500</td>
    </tr>
    <tr>
        <td>3</td>
        <td>Maret</td>
        <td>11.500</td>
    </tr>
    <tr>
        <td>4</td>
        <td>April</td>
        <td>8.800</td>
    </tr>
    <tr>
        <td>5</td>
        <td>Mei</td>
        <td>7.500</td>
    </tr>
</table>

```

7. Array Multi Dimensi

Array multi dimensi adalah array yang memiliki dimensi lebih dari satu. Biasanya digunakan untuk membuat matrik, graph, dan stuktur data rumit lainnya.

Contoh:

```

<?php
// ini adalah array dua dimensi
$matrik = [
    [2,3,4],
    [7,5,0],
    [4,3,8],
];

// cara mengakses isinya
echo $matrik[1][0]; //-> output: 7

```

Masi kita coba contoh yang lain:

```

<?php
// membuat array 2 dimensi yang berisi array asosiatif
$artikel = [

```

```

[
    "judul" => "Belajar PHP & MySQL untuk Pemula",
    "penulis" => "ahmadimuslim"
],
[
    "judul" => "Tutorial PHP dari Nol hingga Mahir",
    "penulis" => "ahmadimuslim"
],
[
    "judul" => "Membuat Aplikasi Web dengan PHP",
    "penulis" => "ahmadimuslim"
]
];

// menampilkan array
foreach($artikel as $post){
    echo "<h2>".$post["judul"]."</h2>";
    echo "<p>".$post["penulis"]."<p>";
    echo "<hr>";
}

```

Hasilnya:



PENGAYAAN

Contoh Program PHP Menggunakan Array

Setelah memahami tentang array diatas, sekarang saya akan memberikan contoh program dan penjelasan dari program tersebut menggunakan bahasa pemrograman PHP. Disini tentu saya hanya menunjukkan program yang mudah dipahami saja. Berikut ini adalah contoh program PHP menggunakan array.

Array 1 Dimensi PHP

Array satu dimensi adalah array yang hanya memiliki satu index saja. Tidak berbeda dengan bahasa pemrograman lain, Pada bahasa pemrograman PHP, index dalam array tersebut diinisialisasikan menggunakan tanda kurung besar ([]). Di bahasa pemrograman lainnya ketika kita mendapati sebuah variabel dengan tanda kurung tersebut, lalu terdapat angka didalamnya, itulah yang disebut index.

```

<?php
$test=array("Index pertama","Index Kedua","Index ketiga");
echo "Hasil array : 0." . $test[0] . ", 1." . $test[1] . " dan 2." .
$test[2] . ".";
?>

```

Array 2 Dimensi Pada PHP

Array atau larik dua dimensi ini memiliki dua buah index. Berbeda dengan array satu dimensi yang hanya memiliki satu index. Dalam konteks ini, array dua dimensi memiliki dua index dimana index

pertama melambangkan baris, sedangkan index kedua melambangkan kolom. Sama seperti bahasa pemrograman lain. Pada pembuatan program, array dua dimensi adalah array yang paling sering digunakan karena dapat menyelesaikan masalah lebih banyak dibanding array lainnya.

```
<?php
$nilai=array(
    array(90,65,83),
    array(90,78,97),
    array(78,90,78)
);
echo"output array <br>";
echo $nilai[0] [0]." ".$nilai[0] [1]." ".$nilai[0] [2]."<br>";
echo $nilai[1] [0]." ".$nilai[1] [1]." ".$nilai[1] [2]."<br>";
echo $nilai[2] [0]." ".$nilai[2] [1]." ".$nilai[2] [2]."<br>";
?>
```

Array 3 Dimensi PHP

Untuk array tiga dimensi ini kita bisa sebut sebagai array dalam array. Karena array tiga dimensi ini ibarat tabel, satu array bisa menyimpan banyak tabel. Index pertama pada array tersebut merupakan jumlah array maksimal yang bisa disimpan, index kedua merupakan kolom di tiap array. Sedang index ketiga merupakan baris pada tiap array. Artinya satu buah array tiga dimensi, dapat menyimpan banyak array dua dimensi.

```
<?php
$nilai=array(
    array(
        array(90,65,83),
        array(90,78,97),
        array(78,90,78)
    ),
    array(array(90,65,83),
        array(90,78,97),
        array(78,90,78)
    )
);
echo"output array <br>";
echo $nilai[0] [0] [0]." ".$nilai[0] [0] [1]." ".$nilai[0] [0] [2]."<br>";
echo $nilai[0] [1] [0]." ".$nilai[0] [1] [1]." ".$nilai[0] [1] [2]."<br>";
echo $nilai[0] [2] [0]." ".$nilai[0] [2] [1]." ".$nilai[0] [2] [2]."<br>";
?>
```

PENALARAN

Menggabungkan Array dan Menampilkan dalam tabel HTML

```
<?php
$merk    = array("Oppo", "Samsung", "Vivo", "Xiaomi", "Nokia", "Realme",
"Sonny");
$harga    = array("19000000", "12000000", "16000000", "12000000",
"16000000", "11000000", "19000000");

$totalArray = count($harga);

echo "<table border='1'>";
echo "<tr>";
echo "<th>Merk handphone</th>";
echo "<th>Harga handphone</th>";
echo "</tr>";

for ($i = 0; $i < $totalArray; $i++) {
    echo "<tr>";
    echo "<td>$merk[$i]</td>";
    echo "<td>$harga[$i]</td>";
    echo "</tr>";
}
```

```

}
echo "</table>";
echo "<br>";

```

Hasilnya

Merk handphone	Harga handphone
Oppo	19000000
Samsung	12000000
Vivo	16000000
Xiaomi	12000000
Nokia	16000000
Realme	11000000
Sonny	19000000

Contoh Kasus Array Asosiatif

Dimisalkan kita akan membuat array customer yang menyimpan data nama, alamat, no. tlp, pekerjaan, dan gaji. Kemudian kita akan tampilkan datanya ke dalam bentuk tabel. Array yang kita buat adalah array asosiatif dengan jumlah customers sebanyak 3. Jadi bagaimana kita membuatnya ?

Pertama kita buat array asosiatifnya seperti berikut :

```

1 <?php
2 $customers = [
3
4     [
5         "Nama" => "Andika",
6         "Alamat" => "Diponegoro",
7         "No.Telp" => "081999666777",
8         "Pekerjaan" => "PNS",
9         "Gaji" => "5.000.000"
10    ],
11
12    [
13        "Nama" => "Fahri",
14        "Alamat" => "Penamparan Agung",
15        "No.Telp" => "085222333444",
16        "Pekerjaan" => "Asisten Manager",
17        "Gaji" => "7.000.000"
18    ],
19
20    [
21        "Nama" => "Miranda",
22        "Alamat" => "Gunung Salak",
23        "No.Telp" => "085999634788",
24        "Pekerjaan" => "Wiraswasta",
25        "Gaji" => "15.000.000"
26    ],
27 ];

```

Jadi terdapat 3 data customer yang masing-masing sudah memiliki elemen nama, alamat, no. tlp, pekerjaan, dan gaji. Array yang kita buat sudah menggunakan array asosiatif dengan cara Array di dalam array. Jadi awal array pertama (kurung buka paling atas) diisi dengan elemen array customer pertama, kedua dan ketiga (dengan membuat array data customer didalam array \$customers).

Selanjutnya kita buat code untuk akses setiap elemen di array \$customers dan menampilkannya pada tabel. Seperti berikut :

```

55 <head>
56 <title>Array Asosiatif</title>
57 </head>
58 <body>
59 <table>
60 <tr>
61 <th>Nama</th>
62 <th>Alamat</th>
63 <th>No. Telp</th>
64 <th>Pekerjaan</th>
65 <th>Gaji</th>
66 </tr>
67 <?php foreach ($customers as $customer) { ?>
68 <tr>
69 <td><?php echo $customer["Nama"]; ?></td>
70 <td><?php echo $customer["Alamat"]; ?></td>
71 <td><?php echo $customer["No.Telp"]; ?></td>
72 <td><?php echo $customer["Pekerjaan"]; ?></td>
73 <td><?php echo $customer["Gaji"]; ?></td>
74 </tr>
75 <?php } ?>
76 </table>
77 </body>

```

header tabel

perulangan untuk mengakses setiap elemen di array \$customers

Pada code diatas, kita membuat header tabel pada tag <th>...</th> sedangkan untuk isi tabel di tag <td>...</td>. Kita menggunakan perulangan foreach untuk setiap data array \$customers. Untuk sobat yang ingin mempelajari tabel php bisa mengklik link belajar tabel.

Selanjutnya kita lihat hasilnya pada browser masing-masing, berikut ini adalah contoh hasilnya :

← → ↻ ⓘ localhost/basicphp/latihanarray/latihan3.php akses direktori latihan3.php

Nama	Alamat	No. Telp	Pekerjaan	Gaji
Andika	Diponegoro	081999666777	PNS	5.000.000
Fahri	Penamparan Agung	085222333444	Asisten Manager	7.000.000
Miranda	Gunung Salak	085999634788	Wiraswasta	15.000.000

▪ LATIHAN 9

Kerjakan soal-soal berikut dengan baik dan benar!

- 1) Array di PHP dapat kita buat dengan 3 bentuk fungsi, tuliskan contohnya!
- 2) Tuliskan cara mengakses array berikut melalui melalui indeks.

```
<?php
$stampan = ["ahmadi", "muslim", "bang muslim"];
```
- 3) Perhatikan array berikut!

```
<?php
$stampan = ["ahmadi", "muslim", "bang muslim"];
```

Tuliskan cara mengakses array diatas melalui:

 - perulangan for,
 - perulangan while
 - perulangan foreach.
- 4) Perhatikan array berikut!

```
<?php
$stampan = [
    "ahmadi",
    "muslim",
    "bang muslim"
];
```

Bagaimanakah caramenghapus data “bang muslim”?
- 5) Ada dua cara yang bisa dilakukan untuk menambah isi array, yaitu:
- 6) Perhatikan array berikut!

```
<?php
$user = [
    "ahmadi",
    "muslim",
    "ahmadimuslim"
];
```

Bagaimanakah cara menambahkan index ke 3 dengan isi “bang muslim”?
- 7) Perhatikan array berikut!

```
<?php
$user = [
    "ahmadi",
    "muslim",
    "ahmadimuslim"
];
```

Bagaimanakah cara mengganti index ke 2 dengan isi “bang muslim”?
- 8) Apa yang dimaksud dengan array asosiatif!
- 9) Perhatikan array asosiatif berikut!

```
<?php
$artikel = [
    "nama" => "ahmadi muslim",
    "pekerjaan" => "guru",
    "usia" => 35
];
```

Bagaimanakah cara mengganti index ke 2 dengan isi “36”?

Bagaimana cara menambah index ke 3 dengan nama array “alamat” isi “paya raja”?
- 10) Apa yang dimaksud dengan array satu dimensi, array dua dimensi, dan array tiga dimensi ?



MATERI PERTEMUAN 10

Memahami Prosedur dan Fungsi



TAHUKAH KAMU...?

Pada kesempatan ini, kita akan membahas:

- Fungsi dengan Parameter
- Parameter dengan Nilai Default
- Fungsi yang Mengembalikan Nilai
- Memanggil Fungsi di dalam Fungsi

A. Menulis / Membuat Fungsi Pada PHP

Fungsi adalah sekumpulan intruksi yang dibungkus dalam sebuah blok. Fungsi dapat digunakan ulang tanpa harus menulis ulang instruksi di dalamnya. Fungsi pada PHP dapat dibuat dengan kata kunci **function**, lalu diikuti dengan nama fungsinya.

Membuat fungsi pada PHP dapat dilakukan dengan mudah, yaitu (1) menuliskan keyword **function** (2) kemudian diikuti dengan nama fungsi (3) diikuti dengan tanda kurung **()** sebagai tempat argumen, (4) kemudian diikuti dengan kurung kurawal **{ }** sebagai block statement yang akan dijalankan ketika fungsi dipanggil.

```
Function Name      Arguments
    ^              ^
function cetak ( $text, $callback ) {
    echo $text;
}                  Statement
```

Pada kondisi tertentu nama fungsi ini tidak ditulis, lihat bagian V. Anonymous function atau closure

Penulisan nama fungsi harus mengikuti ketentuan sebagai berikut:

1. Harus diawali huruf atau underscore(_) kemudian dapat diikuti dengan huruf, angka, dan underscore
2. Case in-sensitive (**tidak** membedakan huruf kecil dan besar)

Banyak fungsi *build-in* dari php yang sering kita gunakan, seperti **print()**, **print_r()**, **unset()**, dll. Selain fungsi-fungsi tersebut, kita juga dapat membuat fungsi sendiri sesuai kebutuhan.

Contoh:

```
function namaFungsi() {
    //...
}
```

Kode intruksi dapat di tulis di dalam kurung kurawal (**{...}**).

Contoh:

```
function perkenalan() {
    echo "Assalamulaikmu, ";
    echo "Perkenalkan, nama kita Ahmadi<br/>";
    echo "Senang berkenalan dengan anda<br/>";
}
```

Fungsi yang sudah dibuat tidak akan menghasilkan apapun kalau tidak dipanggil. Kita dapat memanggil fungsi dengan menuliskan namanya.

Contoh:

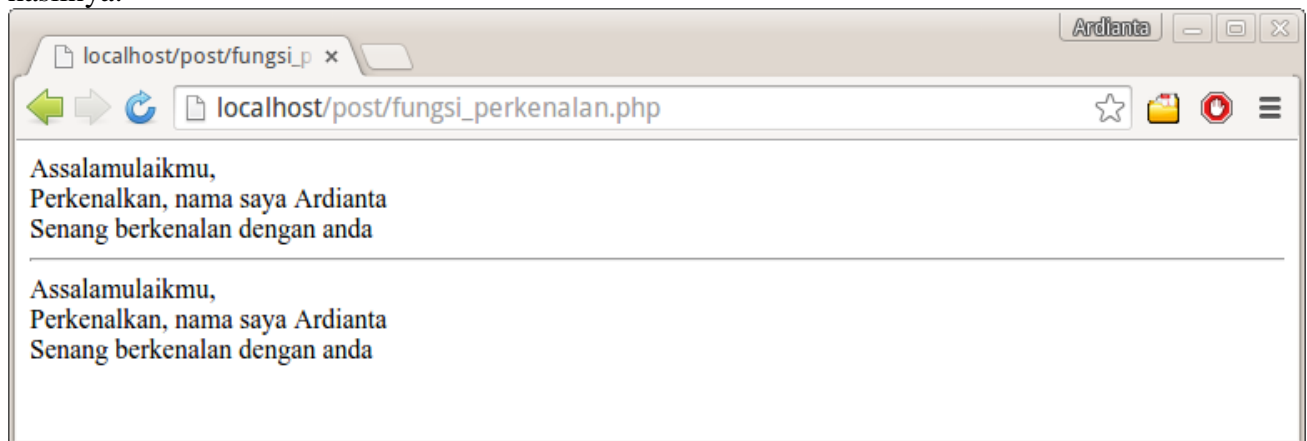
```
perkenalan();
```

Jadi, kode lengkapnya seperti ini:

```
<?php
// mmbuat fungsi
function perkenalan(){
    echo "Assalamulaikum, ";
    echo "Perkenalkan, nama kita Ahmadi<br/>";
    echo "Senang berkenalan dengan anda<br/>";
}
// memanggil fungsi yang sudah dibuat
perkenalan();

echo "<hr>";
// memanggilnya lagi
perkenalan();
?>
```

hasilnya:



1. Argumen Fungsi

Argumen fungsi ditulis dalam tanda kurung dan dapat berupa tipe data apapun baik string, array, object, boolean, dsb., selain itu argumen juga dapat dikosongkan, contoh:

```
<?php
// Tanpa argumen
function nama_bulan() {
    echo 'Agustus';
}
nama_bulan(); // Hasil Agustus
```

Contoh berikutnya kita definisikan argumen, sehingga kita dapat mencetak nama bulan sesuai dengan yang kita inginkan:

```
function nama_bulan($bulan) {
    echo $bulan;
}
nama_bulan('Januari'); // Hasil Januari
```

Lebih lanjut, argumen dari fungsi ini dapat kita definisikan lebih dari satu, caranya, pisahkan argumen dengan tanda koma, contoh:

```
function nama_bulan($bulan, $tahun) {
    echo $bulan . ' ' . $tahun;
}
nama_bulan('Januari', 2016); // Hasil Januari 2016
```

2. Nilai Default Argumen

Kita dapat mendefinisikan nilai default dari argumen, sehingga memudahkan pemanggilan fungsi karena tidak perlu menulis argumen terlalu banyak, contoh:

```
function nama_bulan($bulan, $tahun = 2016) {  
    echo $bulan . ' ' . $tahun;  
}  
nama_bulan('Januari'); // Hasil Januari 2016
```

Nilai default argumen ini bisa kita isi tipe data apa saja seperti boolean (true, false), null, array, object, dll

3. Memanggil Fungsi

Pemanggilan fungsi dilakukan dengan menulis nama fungsi tersebut, seperti pada contoh sebelumnya, kita memanggil fungsi dengan menulis `nama_bulan()` dan `nama_bulan('Januari', 2016)`

Jika fungsi memerlukan argumen, maka kita juga harus menulis argumen tersebut.

Catatan: Ketika melakukan pemanggilan fungsi, maka penulisan argumen harus lengkap, jika fungsi terdiri dari 3 argumen, maka kita harus menuliskan ketiganya, jika tidak maka akan muncul pesan error, KECUALI argumen tersebut memiliki nilai default (dibahas dibawah)

B. Nilai Kembalian – Return Value

1. Menggunakan return

Nilai kembalian ini maksudnya fungsi yang kita panggil tadi akan menghasilkan nilai tertentu, nilai tersebut bisa bertipe apa saja seperti: boolean, float, array, object, dll

Nilai kembalian ini dijalankan dengan menggunakan keyword `return`, contoh:

```
<?php  
function nama_bulan($bulan) {  
    $nama_bulan = array (1 => 'Januari', 2 => 'Februari', 3 => 'Maret');  
    return $nama_bulan[$bulan];  
}  
// date('n') akan menghasilkan bulan sekarang dalam bentuk 1 digit, misal 3  
untuk Januari  
$bulan = nama_bulan(date('n')); // Hasil Maret  
echo $bulan . ' ' . date('Y'); // Hasil Maret 2016
```

Keyword ini dapat diletakkan dimana saja di dalam fungsi dan ketika php menemukan keyword ini, maka seketika pemanggilan fungsi akan dihentikan dan PHP kembali ke baris dimana fungsi tadi dipanggil.

Karakteristik tersebut dapat memudahkan kita mengatur penulisan kode, sehingga, ketika menulis fungsi, kita harus selalu mempertimbangkan kemungkinan penggunaan `return` di tengah code, terutama ketika menggunakan conditional `if`

Contoh:

```
function report($bulan) {  
    if ($bulan < 3) {  
        $status = 'Report belum tersedia';  
    } else {  
        $status = 'Report sudah tersedia';  
    }  
    return $status;  
}  
echo report(2); // Hasil Report belum tersedia;
```

Untuk lebih efisien, kode tersebut dapat diubah menjadi:

```
function report($bulan) {  
    if ($bulan < 3) {  
        return 'Report belum tersedia';  
    } else {  
        return 'Report sudah tersedia';  
    }  
}
```

Pada script pertama, PHP akan membaca seluruh kode pada fungsi, sebaliknya untuk script kedua, ketika sampai `if` maka fungsi berhenti karena bertemu `return`,

Hal ini tentu akan mempercepat proses eksekusi terlebih jika script yang kita tulis panjang.

2. Return value lebih dari satu nilai

Return value HANYA memberikan nilai kembalian sebanyak satu nilai, misal pada contoh diatas hanya menghasilkan nama bulan, jika ingin menghasilkan nilai kembalian lebih dari satu, maka kita gunakan array, contoh:

```
<?php
function nama_bulan($bulan) {
    $nama_bulan = array (1 => 'Januari', 2 => 'Februari', 3 => 'Maret');
    $semester    = $bulan < 7 ? 1 : 2;
    return array('bulan' => $nama_bulan[$bulan], 'semester' => $semester);
}
$bulan = nama_bulan(3);
echo '<pre>'; print_r($bulan);
/* HASIL:
Array
(
    [bulan] => Maret
    [semester] => 1
) */
```

Hasil pengolahan nilai dari fungsi mungkin saja kita butuhkan untuk pemrosesan berikutnya.

Oleh karena itu, kita harus membuat fungsi yang dapat mengembalikan nilai.

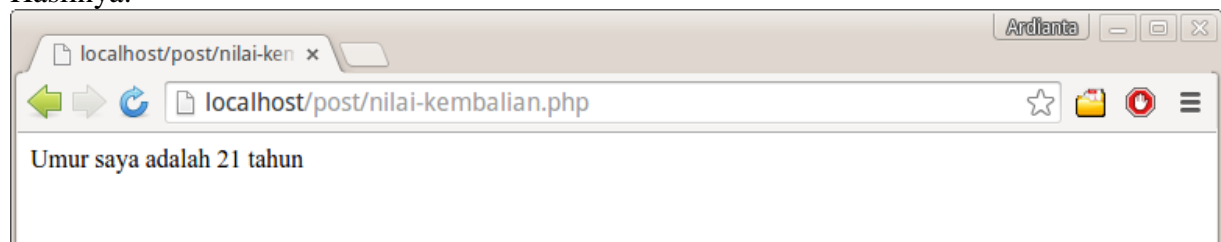
Pengembalian nilai dalam fungsi dapat menggunakan kata kunci **return**.

Contoh:

```
<?php
// membuat fungsi
function hitungUmur($thn_lahir, $thn_sekarang){
    $umur = $thn_sekarang - $thn_lahir;
    return $umur;
}

echo "Umur kita adalah ". hitungUmur(1986, 2023) ." tahun";
?>
```

Hasilnya:



C. Fungsi dengan Parameter

Supaya intruksi yang di dalam fungsi lebih dinamis, kita dapat menggunakan parameter untuk memasukkan sebuah nilai ke dalam fungsi. Nilai tersebut akan diolah di dalam fungsi. Misalkan, pada contoh fungsi yang tadi, tidak mungkin nama yang dicetak adalah *Ahmadi* saja dan salam yang dipakai tidak selalu *assalamualaikum*. Maka, kita dapat menambahkan parameter menjadi seperti ini:

```
<?php
// mmbuat fungsi
function perkenalan($nama, $salam){
    echo $salam.", ";
    echo "Perkenalkan, nama kita ".$nama."<br/>";
    echo "Senang berkenalan dengan anda<br/>";
}
```

```
// memanggil fungsi yang sudah dibuat
perkenalan("Muhardian", "Hi");

echo "<hr>";

$saya = "Ahmadi";
$ucapanSalam = "Selamat pagi";
// memanggilnya lagi
perkenalan($saya, $ucapanSalam);
?>
```

Hasilnya:



Parameter dengan Nilai Default

Nilai *default* dapat kita berikan di parameter. Nilai *default* berfungsi untuk mengisi nilai sebuah parameter, kalau parameter tersebut tidak diisi nilainya.

Misalnya: kita lupa mengisi parameter *salam*, maka program akan *error*. Oleh karena itu, kita perlu memberikan nilai *default* supaya tidak *error*.

Contoh:

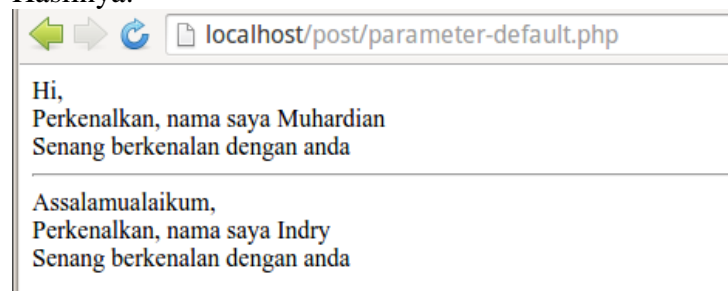
```
<?php
// mmbuat fungsi
function perkenalan($nama, $salam="Assalamualaikum"){
    echo $salam.", ";
    echo "Perkenalkan, nama kita ".$nama."<br/>";
    echo "Senang berkenalan dengan anda<br/>";
}

// memanggil fungsi yang sudah dibuat
perkenalan("Muhardian", "Hi");

echo "<hr>";

$kita = "Indry";
$ucapanSalam = "Selamat pagi";
// memanggilnya lagi tanpa mengisi parameter salam
perkenalan($saya);
?>
```

Hasilnya:



D. Memanggil Fungsi di dalam Fungsi

Fungsi yang sudah kita buat, dapat juga dipanggil di dalam fungsi lain.

Contoh:

```
<?php
// membuat fungsi
function hitungUmur($thn_lahir, $thn_sekarang){
    $umur = $thn_sekarang - $thn_lahir;
    return $umur;
}

function perkenalan($nama, $salam="Assalamualaikum"){
    echo $salam.", ";
    echo "Perkenalkan, nama kita ".$nama."<br/>";
    // memanggil fungsi lain
    echo "Kita berusia ". hitungUmur(1986, 2023) ." tahun<br/>";
    echo "Senang berkenalan dengan anda<br/>";
}
// memanggil fungsi perkenalan
perkenalan("Ahmadi");

?>
```

Hasilnya:



E. Fungsi rekursif

Fungsi rekursif adalah fungsi yang memanggil dirinya sendiri. Fungsi ini biasanya digunakan untuk menyelesaikan masalah seperti faktorial, bilangan fibbonaci, pemrograman dinamis, dll.

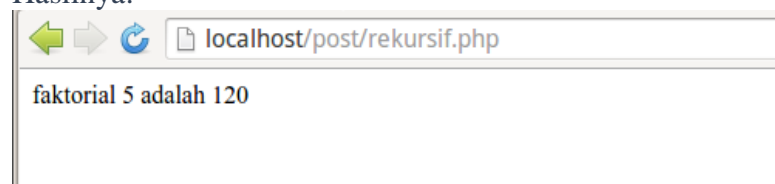
Contoh fungsi rekursif:

```
<?php

function faktorial($angka) {
    if ($angka < 2) {
        return 1;
    } else {
        // memanggil dirinya sendiri
        return ($angka * faktorial($angka-1));
    }
}
// memanggil fungsi
echo "faktorial 5 adalah " . faktorial(5);

?>
```

Hasilnya:



F. Jenis Fungsi Pada PHP

Fungsi pada PHP dibagi menjadi dua yaitu *built-in function* yang merupakan fungsi bawaan PHP dan *user-defined function*, dimana kita membuat fungsi sendiri.

Banyak sekali fungsi yang disediakan php, seperti `substr()`, dll yang dapat kita gunakan langsung, list lengkapnya dapat dilihat pada halaman: PHP: Function and Method Listing – Manual. Terkait fungsi ini ada dua hal pokok yang dapat kita lakukan, yaitu:

1. Mendefinisikan fungsi sendiri, jika fungsi yang kita inginkan belum disediakan oleh PHP
2. Memanggil fungsi, baik *built-in function* dan *user-defined function* cara memanggilmnya sama yaitu menuliskan nama fungsi kemudian diikuti tanda kurung, misal kita memanggil fungsi cetak, maka kita menuliskan: `cetak()`

Kita tidak perlu menghafal semua fungsi tersebut dan ketika memerlukan tidak perlu mencarinya disana, karena tetap akan sulit mencarinya, sebaliknya, gunakan saja google.

G. Fungsi Alias Pada PHP

Untuk alasan tertentu seperti penamaan fungsi yang lebih relevan dengan tugas fungsi tersebut, PHP menyediakan fungsi baru yang tugasnya sama persis dengan fungsi lama, yang disebut fungsi alias (Function Aliases)

Fungsi baru tersebut tidak memiliki kode sendiri, melainkan ketika dipanggil, dia memanggil fungsi yang lama. Contoh dari fungsi alias ini adalah: `die()`, `key_exists()` dan `join()` yang merupakan alias dari fungsi `exit()`, `array_key_exists()` dan `implode()`, contoh penggunaan fungsi ini dapat dibaca disini.

Dari contoh tersebut terlihat bahwa `die()`, `key_exists` dan `join` lebih pendek dan lebih memiliki arti, untuk list lengkap dari alias ini, dapat dilihat pada halaman: PHP: List of Function Aliases – Manual. Beberapa orang memperlmasalahkan performa dari fungsi alias ini, karena fungsinya yang menjalankan fungsi lain, namun sebenarnya tidak masalah menggunakan fungsi ini, karena perbedaan speed nya sangat tidak signifikan.

H. Anonymous Function

Anonymous function atau disebut juga *closure* dapat diartikan fungsi tanpa nama (anonymous). Fungsi ini umumnya digunakan pada fungsi-fungsi yang membutuhkan *callback* (fungsi yang dipanggil oleh fungsi lainnya).

Fungsi yang membutuhkan callback ini bisa *built-in function* seperti `preg_replace_callback`, `array_map`, `array_walk`, dll maupun *user-defined function*

Contoh berikut ini diambil dari tulisan sebelumnya:

```
<?php
$kendaraan      = array('Mobil', 'Motor', 'Sepeda');
$upper          = array_map('toupper', $kendaraan);
function toupper($array_val) {
    return strtoupper($array_val);
}
echo '<pre>'; print_r($upper);
```

Nah, seperti disampaikan sebelumnya, tujuan dibuatnya function adalah agar dapat digunakan kembali (*re-use*), karena fungsi `toupper()` HANYA digunakan sekali, maka fungsi tersebut dapat kita gabungkan ke dalam fungsi `array_map` sehingga bentuknya menjadi anonymous function:

```
<?php
$kendaraan = array('Mobil', 'Motor', 'Sepeda');
$upper = array_map(function($value) {
    return strtoupper($value);
}, $kendaraan);
echo '<pre>'; print_r($upper);
```

Contoh lain pada *user-defined function*:

```
function nama_bulan($bulan, $callback) {
```

```

$list_bulan = array (1 => 'Januari', 2 => 'Februari', 3 => 'Maret');
$nama = $list_bulan[$bulan];

if (is_callable($callback)) {
    return $callback($nama);
}
return $nama;
}

$bulan = nama_bulan(2, function($val) {
    return strtoupper($val);
});
echo $bulan; // Hasil: FEBRUARI

```

▪ LATIHAN 10

Kerjakan soal-soal berikut dengan baik dan benar!

- 1) Jelaskan pengertian fungsi
- 2) Tuliskan bentuk umum sebuah fungsi sederhana dan bagaimana cara memanggilnya?
- 3) Perhatikan contoh fungsi berikut!

```

function perkenalan() {
    echo "Assalamulaikmu, ";
    echo "Perkenalkan, nama kita Ahmadi<br/>";
    echo "Senang berkenalan dengan anda<br/>";
}

```

Bagaimanakah cara memanggil fungsi diatas

- 4) Perhatikan contoh fungsi dengan parameter berikut!

```

<?php
function nama_bulan() {

}
nama_bulan();

```

tulis ulang source code diatas, dan selesaikan cara mengisi **statement** dan **ekspresi** fungsi diatas diatas agar hasilnya :

```

10 januari 2016

```

- 5) Perhatikan contoh fungsi dengan return berikut!

```

<?php
function hitungUmur($thn_lahir, $thn_sekarang) {

}

```

```

echo "Umur saya adalah ". hitungUmur(1986, 2023) ." tahun";

```

tulis ulang source code diatas, dan selesaikan cara mengisi **statement** dan **ekspresi** fungsi diatas diatas agar hasilnya :

```

Nama saya adalah Ahmadi
Umur saya adalah 37 tahun

```

```

int(37)

```


6) Perhatikan contoh fungsi didalam fungsi berikut!

```
Assalamualaikum, Perkenalkan, nama saya Ahmadi  
Kita berusia 37 tahun  
Senang berkenalan dengan anda
```

tulis ulang source code untuk hasil output diatas dengan menggunakan fungsi di dalam fungsi:

```
<?php  
function hitungUmur($thn_lahir, $thn_sekarang)  
{  
}  
  
function perkenalan($nama, $salam = "Assalamualaikum")  
{  
}
```

7) Jelaskan pengertian Fungsi rekursif , buatlah contoh fungsi rekursif untuk faktorial 5!

8) Jelaskan pengertian Fungsi rekursif

```
<?php  
  
function nama_bulan($tanggal = 10, $bulan = "januari", $tahun = 2016)  
{  
    echo $tanggal . ' ' . $bulan . ' ' . $tahun;  
}  
nama_bulan(); // Hasil Januari 2016
```

mana sajakah yang termasuk parameter di dalam fungsi diatas?

9) Perhatikan fungsi berikut ini!

```
<?php;  
"function hitungUmur" ($thn_lahir, $thn_sekarang)  
{  
    $umur = $thn_sekarang - $thn_lahir  
    return $umur  
}  
echo 'Nama saya adalah Ahmadi';  
echo <br>  
echo Umur saya adalah hitungUmur(1986, 2023) " tahun"
```

mana sajakah kesalahan akibat penerapan single/double quote, concatenation dan semicolon?

10) Perhatikan fungsi berikut ini!

```
<?php  
// membuat fungsi  
function hitungUmur($thn_lahir, $thn_sekarang)  
{  
    $umur = $thn_sekarang - $thn_lahir;  
    return $umur;  
}  
echo "Nama saya adalah Ahmadi";  
echo "<br>";  
echo "Umur saya adalah " . hitungUmur(1986, 2023) . " tahun";  
  
echo "<br>";  
  
$ahmadi = hitungUmur(1986, 2023);  
var_dump($ahmadi);  
  
<?php
```

```
function nama_bulan($tanggal = 10, $bulan = "januari", $tahun = 2016)
{
    echo $tanggal . ' ' . $bulan . ' ' . $tahun;
}
nama_bulan(); // Hasil Januari 2016
```

mana sajakah kesalahan akibat penerapan single/double quote, concatenation dan semicolon?



MATERI PERTEMUAN 11

Memahami GET & POST



TAHUKAH KAMU...?

Pada kesempatan ini, kita akan membahas:

- Pengertian GET dan POST pada PHP dan HTTP
- Metode GET
- Method POST
- Kapan Menggunakan Method GET dan POST pada PHP
- \$_REQUEST pada PHP

1. Pengertian GET dan POST pada PHP dan HTTP

Dalam dunia internet, protokol yang umum digunakan adalah protokol HTTP, protokol ini memiliki beberapa metode request (request method) diantaranya adalah dari GET dan POST, jadi GET dan POST ini berdiri sendiri tidak berhubungan dengan bahasa pemrograman seperti PHP dan ASP, sehingga jika kita membicarakan GET dan POST pada PHP, maka sebenarnya kita sedang membicarakan GET dan POST pada HTTP. Dalam berkomunikasi, PHP menggunakan protokol HTTP, oleh karena itu PHP juga menyediakan sarana untuk berinteraksi dengan kedua metode request tersebut yaitu: (1) menyimpan data GET dan POST dan (2) mengirim data GET dan POST.

2. Metode GET

Dalam bahasa Inggris kita akrab dengan istilah GETting, dari istilah tersebut dapat diartikan bahwa metode GET pada HTTP ditujukan untuk mengambil (get) data dari server. Pada metode ini umumnya data berbentuk query string yang dikirim via url, data tersebut berupa pasangan `key=value` yang dipisahkan dengan tanda `&`. Data tersebut digabung dengan url utama yang dipisahkan dengan tanda `?`.

Sebelum dikirim, terlebih dahulu data diproses sehingga memenuhi standar format URL. URL hanya boleh memuat **huruf** (besar dan kecil), **angka**, dan beberapa karakter lain dalam ASCII Character Set seperti (`“-_~`), karakter di luar itu akan diubah ke format tertentu yang diawali tanda `%` kemudian diikuti dengan 2 digit hexadecimal, contoh:

Karakter	URL Encoded
?	%3F
@	%40
=	%3D

Angka pada kolom (URL Encoded) merupakan nilai hexadecimal dari character ASCII, disamping itu URL juga tidak boleh memuat spasi, sehingga spasi akan diubah menjadi tanda `+` atau `%20`. Semua proses tersebut disebut **url encoding**. Metode GET adalah metode yang datanya dikirim melalui URL, data yang dikirim di URL berupa rangkaian pasangan nama dan nilai yang dipisahkan oleh ampersand (`&`). URL dengan data GET akan terlihat sebagai berikut:

```
https://ahmadimuslim.info/data.php?name=alfian&age=21
```

Apabila kita lihat dari URL diatas, disana terdapat nama untuk file PHP yaitu `data.php`. Nah File ini lah yang kita tuju pada Formulir GET. Berikutnya ada pembatas antara File PHP dan juga Variable yang dikirimkan yaitu Tanda Tanya (`?`), dan yang terakhir yaitu Variable yang dikirimkan beserta isi datanya yang dipisah dengan Tanda Ampersand (`&`). Variable yang dikirimkan adalah `name` dan `age`.

Nah, sekarang kita praktekkan bagaimana penggunaan Method GET ini pada Codingan kita, yang kita butuhkan adalah syntax PHP yang berisi Form dan memiliki Method GET.

```
<html lang="en">

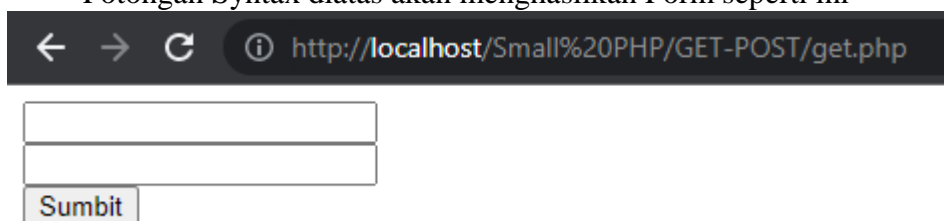
<head>
  <title>Method GET</title>
</head>

<body>
  <form action="" method="GET">
    <input type="text" name="nama"><br />
    <input type="number" name="umur"><br />
    <input type="submit" name="submit" value="Submit">
  </form>

  <?php
  if ($_GET) {
    echo "Nama: " . $_GET["nama"];
    echo "<br/>";
    echo "Umur: " . $_GET["umur"];
  }
  ?>
</body>

</html>
```

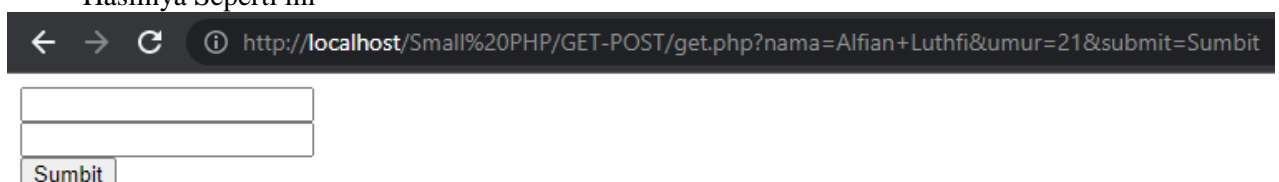
Potongan Syntax diatas akan menghasilkan Form seperti ini



The screenshot shows a web browser address bar with the URL `http://localhost/Small%20PHP/GET-POST/get.php`. Below the address bar is a form with two input fields: a text field for the name and a number field for the age. A "Submit" button is located below the input fields.

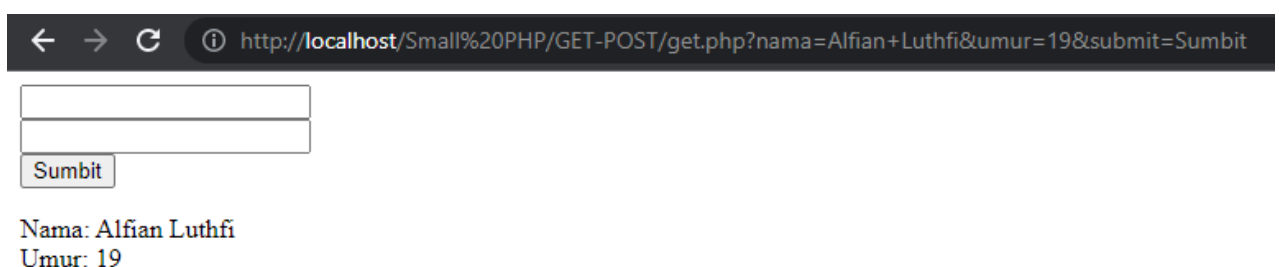
Misalkan kita mengisi Form diatas dengan Nama dan Umur kita. Maka Codingan kita akan mengambil Data yang kita isi menggunakan `$_GET` lalu ditampilkan dengan `echo`.

Hasilnya Seperti ini



The screenshot shows the same web browser as before, but the URL now includes query parameters: `http://localhost/Small%20PHP/GET-POST/get.php?nama=Alfian+Luthfi&umur=21&submit=Submit`. The form fields are empty, and below them, the output is displayed: "Nama: Alfian Luthfi" and "Umur: 21".

Bisa dilihat di URL diatas. Ada tulisan `nama=Alfian+Luthfi&umur=21`. Seperti yang dijelaskan tadi, Method GET ini menggunakan URL, jadi kita bisa mengganti Outputnya dengan mengganti isi URL nya. Misal sekarang kita ganti nilai umur pada URL jadi 19 jadi `nama=Alfian+Luthfi&umur=19`. Maka Outputnya juga akan berubah seperti ini

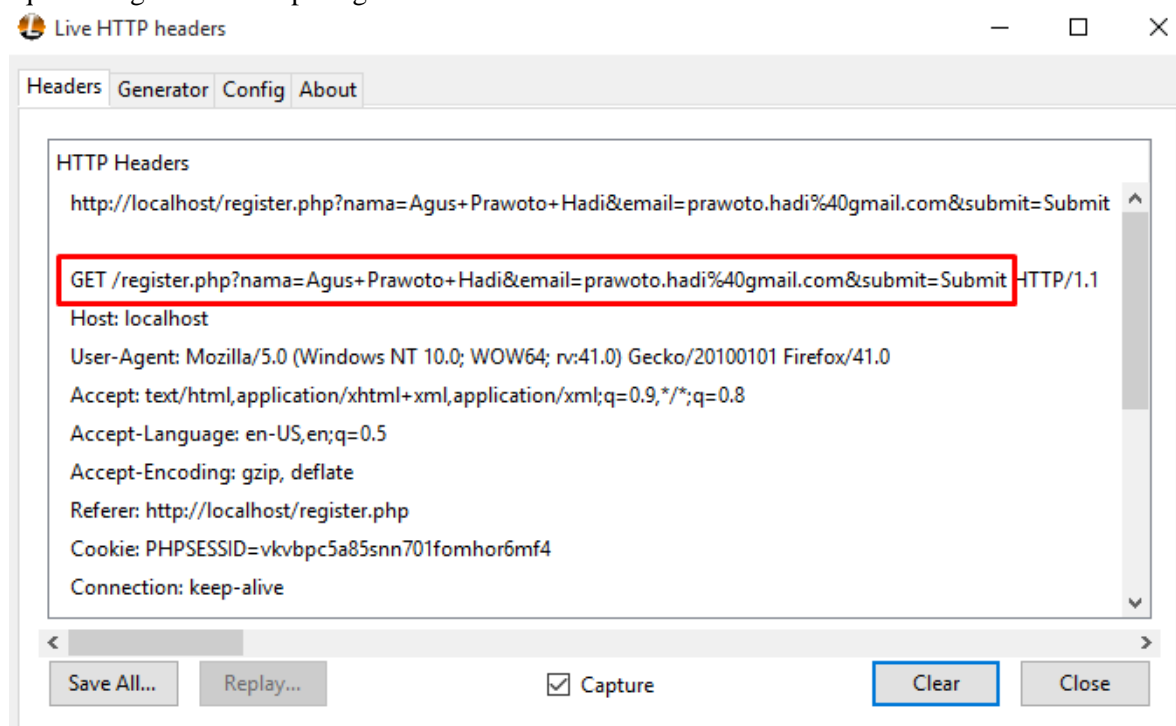


The screenshot shows the same web browser as before, but the URL now has the age changed to 19: `http://localhost/Small%20PHP/GET-POST/get.php?nama=Alfian+Luthfi&umur=19&submit=Submit`. The form fields are empty, and below them, the output is displayed: "Nama: Alfian Luthfi" and "Umur: 19".

Berikut ini merupakan contoh penggunaan metode GET pada form. Kita buat file registrasi.php dan tuliskan kode HTML berikut ini:

```
<html>
<body>
    <form method="GET" action="">
        <input type="text" name="nama"><br>
        <input type="text" name="email"><br>
        <input type="submit" name="submit" value="Submit">
    </form>
</body>
</html>
```

setelah itu simpan dan jalankan pada browser, pada field yang muncul, misal kita isikan nama: Ahmadi, email: bangmuslim@gmail.com, jika kita klik Submit, maka browser akan mengirim request dengan bentuk seperti gambar berikut ini:



dari gambar diatas terlihat bahwa metode yang kita gunakan adalah GET disertai query string nya, disamping itu url pada browser juga akan berubah menjadi:

<https://jagowebdev.com/?nama=Agus+Prawoto+Hadi&email=prawoto%40gmail.com&submit=submit>

dari data diatas terlihat bahwa <spasi> baik pada query string maupun pada url diencode menjadi tanda + dan @ menjadi %40. Ketika kita ambil data tersebut dengan PHP maka otomatis data pada url akan didecode sehingga kita dapatkan bentuk asli dari data yang kita kirim, mari kita tambahkan kode pada file registrasi.php, sehingga menjadi:

```
<html>
<body>
    <form method="GET" action="">
        <input type="text" name="nama"><br>
        <input type="text" name="email"><br>
        <input type="submit" name="submit" value="Submit">
    </form>

    <?php
    if ($_GET)
    {
```

```

        echo 'Nama: ' . $_GET['nama'];
        echo '<br>';
        echo 'Email: ' . $_GET['email'];
    }
    ?>
</body>
</html>

```

ketika kita refresh browser maka akan kita dapatkan hasil:

```

Nama: Ahmadi
Email: bangmuslim@gmail.com

```

Variabel `$_GET` pada PHP berbentuk associative array. Variabel ini bentuknya sama seperti variabel pada umumnya, bedanya `$_GET` ini merupakan variabel global sehingga bisa diakses dimana saja.

Karena bentuknya sama dengan yang lain, variabel ini dapat kita manipulasi sebagaimana kita memanipulasi variabel array lainnya, misal dengan menambahkan nilainya:

```
$_GET['status'] = 'aktif' atau menghapusnya unset($_GET['nama'])
```

Kelebihan dan Kekurangan Method Get

Terdapat beberapa **kelebihan** penggunaan metode GET, diantaranya adalah:

1. Sempel, dan data mudah diedit, misal untuk menuju halaman 5 dari suatu website, kita tinggal mengganti urlnya.
2. Halaman dapat dibookmark dan disimpan pada history browser sehingga mudah untuk diakses kembali.
3. Dapat kembali ke halaman sebelumnya dengan mudah (dengan mengklik tombol Back pada browser).
4. Dapat direfresh dengan mudah.
5. Dapat di distribusikan/dishare.

Meskipun banyak kelebihannya, penggunaan metode ini memiliki beberapa kelemahan yaitu:

1. Panjang data terbatas hanya 2kb – 8kb (tergantung browsernya), jika melebihi batas tersebut akan muncul pesan error 414 Request-URI Too Long, sehingga tidak dapat digunakan untuk mengirim data dalam jumlah besar.
2. Hanya dapat mengirim data jenis teks, jenis lainnya seperti: gambar, file zip, dll tidak dapat dikirim.
3. Karena data dikirim via URL, data tersebut mudah terekspose.

Seperti itulah penggunaan Method GET pada PHP. Ada yang sering mengatakan Methode ini tidak aman karena Data yang kita inputkan terlihat di URL dan bisa diganti-ganti secara asal.

3. Method POST

Kita sering mendengar istilah post, yang biasanya terkait dengan POSTing ke sosial media, dimana pada kegiatan tersebut kita mengirim data berupa tulisan atau gambar untuk disimpan di server sosial media tersebut. Begitu juga dengan istilah POST pada HTTP, POST digunakan untuk mengirim data yang biasanya di gunakan untuk menambah/merubah data pada server.

Pada protokol HTTP, metode POST dapat dikirim baik melalui query string maupun body, seperti pada GET, data yang dikirim melalui query string akan ditampilkan pada URL dan sedangkan yang dikirim melalui body tidak terlihat oleh user.

Pada PHP, data POST yang dikirim melalui query string disimpan pada variabel `$_GET` (seperti metode GET) sedangkan yang dikirim melalui body disimpan pada variabel `$_POST`. Sama seperti `$_GET`, variabel `$_POST` juga berbentuk associative array dan bersifat global yang artinya dapat diakses dimana saja, selain itu juga dapat dilakukan manipulasi sebagaimana variabel array lainnya.

Method POST adalah metode pengiriman data yang Datanya tidak disimpan pada URL. Data pada method POST ini tetap dikirimkan akan tetapi tidak ditampilkan pada URL seperti GET. Method POST ini biasanya digunakan saat registrasi yang membutuhkan input email dan password yang seharusnya tidak muncul di URL.

Method POST ini dirasa lebih aman daripada method GET, bahkan Method ini juga bisa mengirimkan File seperti gambar dan dokumen, tidak hanya Text saja. Bagaimana cara penggunaan dan Contohnya? Mari kita lihat.

```
<html lang="en">

<head>
  <title>Method POST</title>
</head>

<body>
  <form action="" method="POST">
    <input type="text" name="nama"><br />
    <input type="number" name="umur"><br />
    <input type="submit" name="submit" value="Sumbit">
  </form>

  <?php
  if ($_POST) {
    echo "Nama: " . $_POST["nama"];
    echo "<br/>";
    echo "Umur: " . $_POST["umur"];
  }
  ?>
</body>

</html>
```

Potongan syntax diatas akan menghasilkan Hasil yang sama seperti yang ada di Method GET. Tapi bedanya, saat kita isi kita tidak bisa melihat yang kita inputkan di URL. Beginilah hasilnya



← → ↻ ⓘ http://localhost/Small%20PHP/GET-POST/post.php

Nama: Alfian Luthfi
Umur: 21

Bisa dilihat pada gambar diatas. Output yang keluar sama, akan tetapi di URL tidak muncul data seperti pada method GET. Pada akhirnya kita tidak bisa mengganti Data yang kita inputkan dan tidak bisa melihat data tersebut. Membuat penginputan data lebih *Secure*.

Penggunaan metode POST sering kita jumpai terutama pada saat pengiriman data menggunakan form html. Misal: meneruskan contoh sebelumnya, pada file registrasi.php kita ganti method pada bagian form dari `get` menjadi `post`

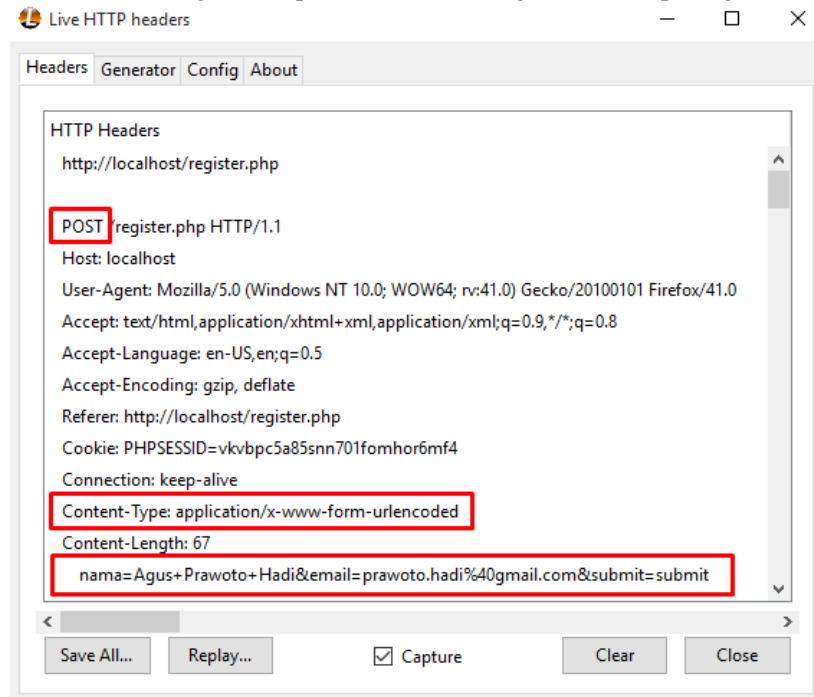
```
<html>
<body>
  <form method="POST" action="">
    <input type="text" name="nama"><br>
    <input type="text" name="email"><br>
    <input type="submit" name="submit" value="submit">
  </form>
```

```

<?php
if ($_POST)
{
    echo 'Nama: ' . $_POST['nama'];
    echo '<br>';
    echo 'Email: ' . $_POST['email'];
}
?>
</body>
</html>

```

misal field kita isi dengan nama: Ahmadi dan Email: bangmuslim@gmail.com, ketika kita klik submit, maka browser akan mengirim request ke server dengan bentuk gambar seperti gambar dibawah ini:



dari gambar diatas terlihat data dikirim pada bagian body (kotak merah paling bawah) dan data yang dikirim diencode dengan sistem sama dengan GET (url encode) yang memang secara default cara pengiriman data pada POST adalah menggunakan `application/x-www-form-urlencoded`, disamping itu juga ada `multipart/form-data` yang digunakan untuk pengiriman data berupa file/binary.

POST yang dikirim via query string dan HTTP body

Sebelumnya telah kita singgung mengenai pengiriman POST melalui query string yang artinya akan ditampilkan pada URL. Sering kita fahami bahwa semua yang ada di URL adalah GET, karena kita sering berinteraksi dengan variabel `$_GET` dan dalam praktek hal tersebut tidak pernah menjadi masalah, tidak akan ada error yang muncul. Namun demikian sebenarnya yang terjadi tidaklah demikian, coba kita buka kembali file `register.php` dan kita ubah menjadi:

```

<html>
<body>
    <form method="POST" action="?action=edit">
        <input type="text" name="nama"><br>
        <input type="text" name="email"><br>
        <input type="submit" name="submit" value="submit">
    </form>

    <?php
    if ($_POST)
    {
        echo '<pre>';
        print_r($_GET);
    }

```



```

        print_r($_POST);
    }
    ?>
</body>
</html>

```

misal kita isi dengan data yang sama dengan contoh sebelumnya, maka ketika kita klik submit, hasil yang kita peroleh adalah:

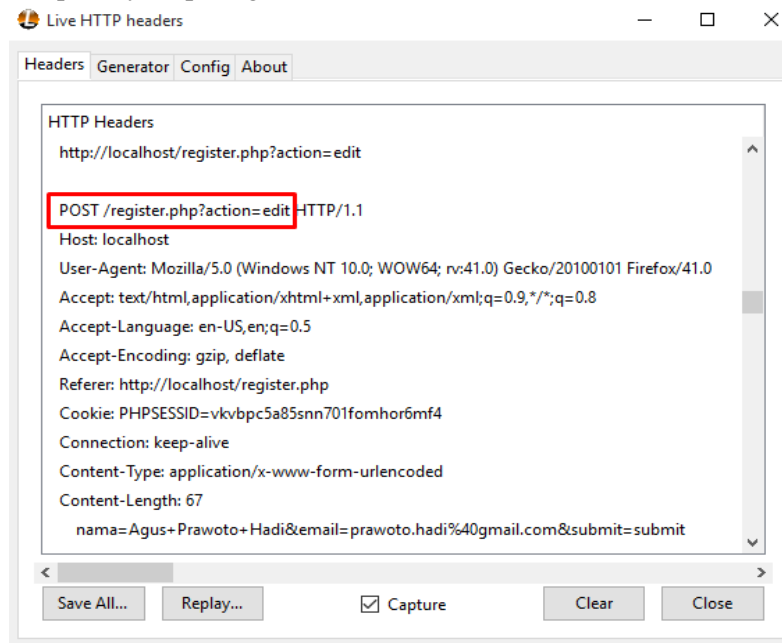
```

// $_GET
Array
(
    [action] => edit
)

// $_POST
Array
(
    [nama] => Ahmadi
    [email] =>ahmadimuslim@gmail.com
    [submit] => submit
)

```

dan bentuk HTTP Requestnya seperti gambar dibawah ini:



dari gambar diatas terlihat bahwa metode yang kita gunakan adalah POST dan data dikirim dengan dua cara baik melalui query string maupun HTTP body. query string disimpan pada variabel \$_GET dan HTTP body disimpan pada variabel \$_POST

Kelebihan dan kekurangan Method Post

Pengiriman data menggunakan metode POST memiliki beberapa kelebihan diantaranya:

1. Lebih aman dari pada metode GET karena data yang dikirim tidak terlihat, serta parameter yang dikirim tidak disimpan pada history browser/log browser.
2. Dapat mengirim data dalam jumlah besar.
3. Dapat mengirim berbagai jenis data seperti gambar, file, dll, tidak harus teks.

Meskipun terdapat kelebihan, penggunaan metode ini juga memiliki beberapa kelemahan, walaupun sebenarnya bukan kelemahan melainkan memang menjadi karakteristik dari metode ini:

1. Data tidak disimpan pada history browser.
2. Data tidak dapat dibookmark.

Karena dianggap sebagai data sensitif, maka ketika kita mererefresh browser, akan muncul konfirmasi pengiriman ulang data, demikian juga ketika kita tekan tombol back.

4. Kapan Menggunakan Method GET dan POST pada PHP

Untuk menentukan apakah kita akan menggunakan Method GET atau POST pada PHP, kita harus memikirkan terlebih dahulu: Apakah terjadi perubahan pada data di Server kita?. Apabila terjadi perubahan, jelas kita harus menggunakan Method POST, akan tetapi kalau tidak terjadi perubahan pada Server, kita bisa menggunakan Method GET.

Contohnya, misal kita membuat pagination pada PHP, lebih baik jelas menggunakan method GET.

POST digunakan saat ada Data Sensitif yang dikirimkan ke Database. Misal kan ID dari suatu hal, atau email dan password, tentunya kita harus menggunakan POST. Akan cukup bahaya apabila kita menggunakan GET, karena nanti semua orang bisa melihat Data Sensitif yang kita kirimkan.

Untuk memutuskan apakah kita akan menggunakan metode GET atau POST, kita harus selalu mengajukan pertanyaan: **apakah akan terjadi perubahan pada server?** jika ya, maka kita gunakan POST, jika tidak maka kita gunakan GET.

Contohnya adalah ketika membuat pagination, kita cukup menggunakan GET tidak perlu POST, bisa dibayangkan ketika kita sudah sampai halaman 5 suatu artikel, dan kita mau membookmarknya untuk dibaca di lain waktu, dan ternyata tidak bisa karena halaman tersebut menggunakan POST, dengan terpaksa kita harus membuka halaman 1 lagi!, Begitu juga dengan sistem pencarian seperti pada google, bisa dibayangkan jika metode yang digunakan adalah POST.

Contoh lainnya adalah penggunaan ID untuk menghapus data, perhatikan url berikut: <http://www.contoh.com/user.php?action=delete&id=1>, url ini akan berbahaya jika hal ini benar-benar terjadi. Kenapa? Karena data user akan terhapus dengan begitu mudahnya, lain halnya jika ID tersebut digunakan hanya untuk menampilkan data seperti <http://www.contoh.com/user.php?action=view&id=1>.

singkatnya GET untuk READ data, POST untuk CREATE, UPDATE, DELETE data. Pertanyaan kedua adalah **apakah ada data sensitif yang dikirim?** jika ya maka kita gunakan POST.

Contohnya adalah pengiriman username dan password ketika login, atau data keuangan seperti kartu kredit. Jika data tersebut dikirim via url maka jelas data tersebut akan terekspose kemana-mana terutama jika ter index oleh google. (Kecuali pada penggunaan AJAX). GET dan POST merupakan metode yang digunakan protokol HTTP untuk pertukaran data.

Pada PHP, data yang dikirim menggunakan metode GET akan disimpan dalam variabel \$_GET, sedangkan POST akan disimpan pada variabel \$_POST (untuk data yang dikirim via url disimpan pada variabel \$_GET). Masing masing metode memiliki kelebihan dan kekurangan dan memang sebenarnya kedua metode tersebut ditujukan untuk keperluan berbeda, sehingga kita harus tahu kapan menggunakan GET dan kapan menggunakan POST.

PHP juga menyediakan variabel \$_REQUEST yang merupakan gabungan dari variabel \$_GET, \$_POST dan \$_COOKIE. Pada variabel ini tidak terlihat dari mana datangnya data apakah dari GET atau POST sehingga sebaiknya lebih berhati-hati ketika menggunakan variabel ini.

5. \$_REQUEST pada PHP

PHP menyediakan variabel global bernama \$_REQUEST, variabel ini merupakan gabungan dari tiga variabel yaitu: \$_GET, \$_POST dan \$_COOKIE. Karakteristik variabel ini sama dengan \$_GET dan \$_POST yaitu berbentuk associative array yang bersifat global dan dapat kita manipulasi seperti variabel array lainnya.

Dalam pembentukan variabel \$_REQUEST, variabel yang menjadi prioritas adalah \$_POST, sehingga jika antara \$_GET dan \$_POST terdapat **key** yang sama, maka yang digunakan adalah data pada variabel \$_POST, contoh mari kita ubah file register.php dengan menambahkan satu hidden field bernama action:

```
<html>
<body>
    <form method="POST" action="?action=edit">
        <input type="text" name="nama"><br>
        <input type="text" name="email"><br>
        <input type="hidden" name="action" value="insert">
        <input type="submit" name="submit" value="submit">
    </form>

    <?php
    if ($_POST)
```

```

        {
            echo '<pre>';
            print_r($_GET);
            print_r($_POST);
            print_r($_REQUEST);
        }
    ?>
</body>
</html>

```

Seperti pada contoh sebelumnya, kita isikan nama: Ahmadi dan email: bangmuslim@gmail.com, ketika kita submit maka kita dapatkan hasil:

```

// $_GET
Array
(
    [action] => edit
)

// $_POST
Array
(
    [nama] => Ahmadi
    [email] =>ahmadimuslim@gmail.com
    [action] => insert
    [submit] => submit
)

// $_REQUEST
Array
(
    [action] => insert
    [nama] => Ahmadi
    [email] => prawoto.hadi@gmail.com
    [submit] => submit
)

```

Dari contoh diatas terlihat bahwa nilai action yang digunakan adalah insert yang terdapat pada variabel \$_POST. Kita pribadi jarang dan mungkin tidak pernah menggunakan variabel \$_REQUEST karena akan menyulitkan dan membuat ambigu, tidak jelas dari mana datangnya data apakah dari inputan user atau dari url, terlebih jika program yang kita buat kompleks, dengan menggunakan \$_GET atau \$_POST masalah tersebut tidak akan terjadi.

▪ LATIHAN 11

Kerjakan soal-soal berikut dengan baik dan benar!

- 1) Sebutkan kelebihan dan Kekurangan Method Get?
- 2) Sebutkan kelebihan dan kekurangan Method Post?
- 3) Kapan Kita Menggunakan Method GET dan POST pada PHP?
- 4) Perhatikan contoh methodGet berikut!

```

<body>
    <form action="" method="GET">
        <input type="text" name="nama"><br />
        <input type="number" name="umur"><br />
        <input type="submit" name="submit" value="Sumbit">
    </form>

</body>

```

tulis ulang source code diatas, dan selesaikan cara mengisi **statement** dan **ekspresi** fungsi diatas diatas agar hasilnya :

Nama: ahmadi muslim

Umur: 14

5) Perhatikan contoh methodPost berikut!

```
<body>
  <form action="" method="POST">
    <input type="text" name="nama"><br />
    <input type="number" name="umur"><br />
    <input type="submit" name="submit" value="Sumbit">
  </form>
```

```
</body>
```

tulis ulang source code diatas, dan selesaikan cara mengisi **statement** dan **ekspresi** fungsi diatas diatas agar hasilnya :

Nama: ahmadi muslim

Email: ahmadimuslim55@guru.smk.belajar.id

6) Perhatikan contoh beberapa method berikut!

```
<body>
  <form method="POST" action="?action=edit">
    <input type="text" name="nama"><br>
    <input type="text" name="email"><br>
    <input type="hidden" name="action" value="insert">
    <input type="submit" name="submit" value="submit">
  </form>
```

```
</body>
```

tulis ulang source code diatas, dan selesaikan cara mengisi **statement** dan **ekspresi** fungsi diatas diatas agar hasilnya :

```
Array
(
    [action] => edit
)
Array
(
    [nama] => ahmadi muslim
    [email] => ahmadimuslim55@guru.smk.belajar.id
    [action] => insert
    [submit] => submit
)
Array
(
    [action] => insert
    [nama] => ahmadi muslim
    [email] => ahmadimuslim55@guru.smk.belajar.id
    [submit] => submit
)
```

PENGAYAAN

Buatlah sebuah program hitung yang menggunakan method POST dan GET, dengan menerapkan penggunaan fungsi, operator, perulangan atau percabangan.!

Contoh:

```
<!DOCTYPE html>
<html>

<head>
<title>WAKTU TEMPUH</title>
</head>

<body>

<form action="" method="post">
<h1>Program Hitung Waktu Tempuh</h1>
    Jarak Tempuh (km) : <br>
<input type="text" name="jarak"><br><br>
    Kecepatan (km/jam) : <br>
<input type="text" name="cepat"><br><br>
<input type="submit" value="Hitung Waktu">
</form>

</body>

</html>

<?php
if (isset($_POST['jarak']) && isset($_POST['cepat'])) {

    $jarak = $_POST['jarak'];
    $cepat = $_POST['cepat'];

    $waktu = $jarak / $cepat;

    echo

    "
<h1>Program Hitung Waktu Tempuh</h1>
<p>Jarak : <b>$jarak</b> km</p>
<p>Kecepatan : <b>$cepat</b> km/jam</p>
<h3>Waktu Tempuh : $waktu jam</h3>
    ";
}
?>
```

Output:

Program Hitung Waktu Tempuh

Jarak Tempuh (km) :

Kecepatan (km/jam):

Hitung Waktu

Program Hitung Waktu Tempuh

Jarak : 100 km

Kecepatan : 50 km/jam

Waktu Tempuh : 2 jam