DAFTAR ISI

BAGIAN 1 :
PHP (PHP : HYPERTEXT PREPROCESSOR)

1.1. PENGANTAR PHP

PHP (PHP : Hypertext Preprocessor) adalah bahasa *server-side scripting* yang menyatu dengan HTML untuk membuat halaman web yang dinamis. Karena PHP merupakan *server-side scripting* maka sintaks dan perintah-perintah PHP akan dieksekusi di server kemudian hasilnya dikirimkan ke browser dalam format HTML. Dengan demikian kode program yang ditulis dalam PHP tidak akan terlihat oleh user sehingga keamanan halaman web lebih terjamin. PHP dirancang untuk membentuk halaman web yang dinamis, yaitu halaman web yang dapat membentuk suatu tampilan berdasarkan permintaan terkini, seperti menampilkan isi basis data ke halaman web.

PHP termasuk dalam *Open Source Product*, sehingga *source code* PHP dapat dirubah dan didistribusikan secara bebas. Versi terbaru PHP dapat didownload secara gratis di situs resmi PHP : http://www.php.net. PHP juga dapat berjalan pada berbagai web server seperti IIS (*Internet Information Server*), PWS (*Personal Web Server*), Apache, Xitami, dll. PHP juga mampu lintas *platform*. Artinya PHP dapat berjalan di Sistem Operasi Windows dan beberapa versi Linux, dan PHP dapat dibangun sebagai modul pada *web server* Apache dan sebagai binary yang dapat berjalan sebagai CGI (*Common Gateway Interface*). PHP dapat mengirim HTTP header, dapat mengatur *cookies*, mengatur *authentication* dan *redirect users*.

Salah satu keunggulan yang dimiliki oleh PHP adalah kemampuannya untuk melakukan koneksi ke berbagai macam software basis data, sehingga dapat menciptakan suatu halaman web yang dinamis. PHP mempunyai koneksitas yang baik dengan beberapa basis data antara lain Oracle, Sybase, mSQL, MySQL, Microsoft SQL Server, Solid, PostgreSQL, Adabas, FilePro, Velocis, dBase, Unix dbm, dan tak

terkecuali semua database ber-interface ODBC. PHP juga memiliki integrasi dengan beberapa *library* eksternal yang dapat membuat Anda melakukan segalanya dari dokumen PDF hingga mem-parse XML. PHP mendukung komunikasi dengan layanan lain melalui protokol IMAP, SNMP, NNTP, POP3 atau bahkan HTTP. Bila PHP berada dalam halaman web Anda, maka tidak lagi dibutuhkan pengembangan lingkungan khusus atau direktori khusus. Hampir seluruh aplikasi berbasis web dapat dibuat dengan PHP. Namun kekuatan utama adalah konektivitas basis data dengan web. Dengan kemampuan ini kita akan mempunyai suatu sistem basis data yang dapat diakses dari web.

Tips :

- Sebelum anda mencoba script-script yang ada dihalaman berikut, buatlah sebuah folder kerja anda.
- Folder ini menjadi tempat penyimpanan file-file php anda. Setiap mengetikkan script di tiap sesi simpanlah sesuai dengan nama file yang ada diatas script tersebut.
- Contoh : File : “Lat01.php” berarti anda harus menyimpan file tersebut dengan nama “Lat01.php”.
- Jangan lupa untuk memberi ekstensi *.PHP disetiap nama file yang anda simpan.

1.2. MENGENAL SKRIP PHP

Script PHP termasuk dalam *HTML-embedded*, artinya kode PHP dapat disisipkan pada sebuah halaman HTML. Ada beberapa cara untuk menuliskan *script* PHP, yaitu:

```
1.  <?php
      Script PHP
    ?>

2.  <?
      Script PHP
    ?>
```

```
3.  <script language = ”php”>
      Script PHP Anda
    </script>

4.  <%
      Script PHP Anda
    %>
```

Cara pertama merupakan format yang dianjurkan tetapi mungkin cara kedua akan sering digunakan karena lebih ringkas. Cara yang ketiga digunakan untuk mengantisipasi editor-editor yang tidak dapat menerima kedua cara di atas. Selain itu

kita juga bisa menggunakan cara penulisan ASP, tetapi tentu saja ada beberapa konfigurasi yang perlu dilakukan.

Parser PHP bekerja membaca file HTML sampai ditemukan penanda khusus yang memberitahukan untuk menerjemahkan skrip berikutnya sebagai kode PHP. Parser PHP akan mengeksekusi semua perintah dalam blok kode PHP tersebut. Dengan cara inilah maka kode skrip PHP dapat ditempelkan pada dokumen HTML. Teks lainnya yang berada diluar blok PHP akan dianggap sebagai skrip HTML biasa.

Berikut ini contoh sederhana pemakaian bahasa PHP yang disisipkan dalam halaman HTML:

File : Lat01.php

```
<html>
  <head>
    <title>Skrip PHP pertama</title>
  </head>
  <body>
    <?php
      echo "Ini Skrip PHP pertamaku!";
    ?>
  </body>
</html>
```

Contoh lain :

File : Lat02.php

```
<html>
<head>
<title>Test Penyisipan PHP Pada HTML</title>
</head>
<body>
Kapal Asing, Silakan identifikasikan diri Anda! <br>
<?php
// Berikut ini adalah kode PHP yang disisipkan
echo "<b>Ini adalah kapal Federasi Planet USS Enterprise.<br>";
echo "Saya Piccard, Jean Luc Piccard, kapten kapal.</b>";
?>
</body>
</html>
```

1.3. VARIABEL

Pada setiap bahasa pemrograman pasti akan kita temui konsep variabel. Variabel adalah sebuah tempat di memori untuk menyimpan data yang nilainya dapat berubah-ubah selama program dijalankan. Tetapi tidak seperti pada kebanyakan bahasa pemrograman lain yang mengharuskan kita untuk mendeklarasikan variabel terlebih dahulu, variabel dalam PHP tidak harus dideklarasikan sebelum variabel tersebut digunakan. Varabel diwakili oleh kata tertentu dengan aturan penulisan sebagai berikut :

1. Variabel dimulai dengan tanda ***dollar*** (**$**).
2. Harus dimulai dengan huruf atau ***underscore*** (**_**).
3. Tidak boleh menggunakan tanda baca.
4. *Case sensitive* atau huruf besar dan huruf kecil berbeda.
5. Jangan menggunakan kata yang merupakan fungsi *built-in* PHP.

Contoh-contoh penulisan variabel :

Benar	Salah
$variabel	$var!abel
$_pilih	$-pilih
$ti02	$02ti
$ini_itu	$ini-itu

Contoh penggunaan variabel :

File : Lat03.php

```
<html>
<head>
  <title>Menggunakan variabel</title>
<head>
<body>
  <?php
  $jurusan = "Teknik Informatika";
  print("Jurusan : $jurusan <br>");
  $jurusan = "Sistem Informasi";
  print("Jurusan : $jurusan <br>");
  ?>
</body>
</html>
```

Berikutnya bukalah file Lat01.php lalu rubahlah menjadi berikut :

File : Lat04.php

```
<html>
<head>
  <title>Test Penyisipan PHP Pada HTML</title>
</head>
<body>
  Kapal Asing, Silakan identifikasikan diri Anda! <br>
  <?php
    // Berikut ini adalah inisiasi beberapa variabel
    $namad = "Jean";
    $namat = "Luc";
    $namab = "Piccard";
  ?>
  <b>Ini adalah kapal Federasi Planet USS Enterprise.<br>
  <?php
    echo "Saya $namab, $namad $namat $namab, kapten kapal.</b>";
  ?>
</body>
</html>
```

Dari kedua contoh diatas Lat03.php dan Lat04.php dapat dilihat ada 2 metode yang bisa digunakan untuk menampilkan ouput. Pada file Lat03.php menggunakan fungsi **print** sedangkan pada Lat04.php menggunakan fungsi **echo**. Untuk mengetahui struktur dasar ***echo*** dan ***print*** silahkan lihat sesi **Fungsi Built-In PHP**.

Latihan :

Sekarang coba buka kembali file Lat03 dan rubahlah semua tulisan **print** dengan **echo** lalu simpanlah dengan nama Lat05.php lalu amatilah hasilnya. Apakah ada perbedaan antara output Lat03.php dengan Lat05.php yang barusan anda buat? Dengan cara yang sama bukalah file Lat04.php dan rubahlah scriptnya kemudian beri nama dengan Lat06.php

Berikut ini adalah sebuah contoh mengenai betapa *luwes*nya penggunaan variabel dalam PHP. Simpan kode ini dalam nama Lat07.php.

File : Lat07.php

```
<?php
// Contoh variabel $a
$a = "Testing";
// Kini $a adalah variable jenis String
echo "Nilai a adalah $a (string)<br>";
$a = 55;
// Kini $a adalah variable jenis Integer
echo "Nilai a berubah menjadi $a (Integer)<br>";
$a = 7.5;
// Kini $a adalah variable jenis floating point
echo "Nilai a sekarang menjadi $a (floating point)<br>";
?>
```

1.4. KONSTANTA

Konstanta adalah variabel yang nilainya tetap. Konstanta hanya diberi nilai pada awal program dan nilainya tidak pernah berubah selama program berjalan. PHP telah mendefinisikan beberapa konstanta, misalnya: PHP_VERSION, yaitu konstanta yang memberikan informasi tentang versi PHP yang digunakan. Selain konstanta yang telah disediakan oleh PHP kita juga dapat membuat konstanta sendiri. Aturan penulisan konstanta adalah sebagai berikut :

```
define(“nama_konstanta”,”nilai_konstanta”)
```

Contoh penggunaan konstanta :

File : Lat08.php

```
<html>
<head>
  <title>Menggunakan konstanta</title>
<head>
<body>
  <?php
  define("KAMPUS", "STMIK AMIKOM YOGYAKARTA");
  print(KAMPUS);
  ?>
</body>
</html>
```

1.5. TIPE DATA

PHP mengenal beberapa macam tipe data antara lain : integer, floating point, dan string. Floating point lebih dikenal dengan nama double atau desimal. Penulisan string selalu diawali dengan tanda petik ganda (“) atau tanda petik tunggal (‘).

Contoh-contoh penulisan tipe data yaitu sebagai berikut :

Tipe Data	Contoh	Keterangan
Integer	$jumlah = 10; $nilai = -5;	Bilangan bulat
Doble	$skor = 90.00; $bunga = 12.50;	Bilangan real
String	$kota = “Yogyakarta” $motto = “Nyaman”	Karakter, kalimat

1.6. OPERATOR

Operator adalah simbol yang digunakan untuk memanipulasi data, seperti penambahan, pengurangan, perkalian, perbandingan, atau penugasan. Ada operator yang menggunakan satu operand ada juga yang menggunakan dua operand.

Sedangkan *operand* adalah data yang dioperasikan atau dimanipulasi. *Operand* dan *Operator* bersama-sama membentuk suatu ekspresi (ungkapan).

Operator dapat dikelompokkan dalam 4 kategori, yaitu :

1. Operator aritmatika adalah operator yang berhubungan dengan fungsi matematika.
2. Operator logika adalah operator yang membandingkan TRUE dan FALSE.
3. Operator bitwise adalah operator yang membandingkan binary.
4. Ada juga operator yang sering digunakan namun tidak termasuk dalam kelompok di atas, kita dapat mempelajarinya dalam kelompok lain.

1.6.1. Operator Aritmatika

Operator aritmetika merupakan operator yang berhubungan dengan fungsi matematika. Operator ini sering kita gunakan dalam program yang akan kita buat.

Operator	Operasi
+	Penambahan
-	Pengurangan
*	Perkalian
/	Pembagian
%	Sisa pembagian
++	Penambahan dengan 1
--	Pengurangan dengan 1

Listing program berikut ini akan memberikan contoh penggunaan operator aritmatika.

File : Lat09.php

```
<html>
  <head>
    <title>Operator Aritmatika</title>
  </head>
  <body>
  <?php
    $harga = 1500;
    $banyak = 9;
    print("Anda telah menghabiskan $banyak porsi sate<br>\n");
    print("yang tiap porsi seharga Rp $harga<br>");
    print("maka anda harus membayar Rp ");
    print($harga * $banyak);
  ?>
  </body>
</html>
```

Listing berikut adalah bentuk pengembangan dari semua materi yang ada di Lat01.php sampai Lat09.php

File : Lat10.php

```
<?
// inisiasi variable yang digunakan
// nama peralatan
$alat_geordi1 = "Phaser";
$alat_geordi2 = "Tricorder";
$alat_geordi3 = "Visor";
$alat_geordi4 = "Analyzer Photonik";

// harga per unit peralatan
$harga_alat_geordi1 = 7500;
$harga_alat_geordi2 = 12500;
$harga_alat_geordi3 = 16000;
$harga_alat_geordi4 = 2300;

// jumlah peralatan yang ada
$jumlah_alat_geordi1 = 2;
$jumlah_alat_geordi2 = 5;
$jumlah_alat_geordi3 = 1;
$jumlah_alat_geordi4 = 3;

// total harga per jenis peralatan
$total_alat_geordi1 = $jumlah_alat_geordi1 * $harga_alat_geordi1;
$total_alat_geordi2 = $jumlah_alat_geordi2 * $harga_alat_geordi2;
$total_alat_geordi3 = $jumlah_alat_geordi3 * $harga_alat_geordi3;
$total_alat_geordi4 = $jumlah_alat_geordi4 * $harga_alat_geordi4;

// hitung grand total nilai peralatan Geordi
$total_harga = $total_alat_geordi1 + $total_alat_geordi2 +
$total_alat_geordi3 + $total_alat_geordi4;
```

```
// besar diskon untuk Geordi
$diskon = 5;
// jumlah total diskon yang diberikan kepada Geordi
$nilai_diskon = ($diskon * $total_harga)/100;
// jumlah yang harus dibayar Geordi
$total_harga_dibayar = $total_harga - $nilai_diskon;

?>
<html>
<head>
<title>Geordi dan Daftar Peralatan Yang Dibeli</title>
</head>
<body>
<center>
<table border="1" cellspacing="0" cellpadding="3">
<tr>
<td colspan="4" align="center" valign="middle">
<b>Daftar Pemesanan Peralatan Geordi La Forge - NCC1701D</b>
</td>
</tr>
<tr>
<td><b>Nama Peralatan</b></td>
<td><b>Jumlah</b></td>
<td><b>Harga Satuan</b></td>
<td><b>Jumlah Harga</b></td>
</tr>

<?
// Mulai untuk mengisi tabel daftar dengan data yang ada
?>
<tr>
<td align="left"><? echo $alat_geordi1; ?></td>
<td align="right"><? echo $jumlah_alat_geordi1; ?></td>
<td align="right"><? echo $harga_alat_geordi1; ?></td>
<td align="right"><? echo $total_alat_geordi1; ?></td>
</tr>
<tr>
<td align="left"><? echo $alat_geordi2; ?></td>
<td align="right"><? echo $jumlah_alat_geordi2; ?></td>
<td align="right"><? echo $harga_alat_geordi2; ?></td>
<td align="right"><? echo $total_alat_geordi2; ?></td>
</tr>
<tr>
<td align="left"><? echo $alat_geordi3; ?></td>
<td align="right"><? echo $jumlah_alat_geordi3; ?></td>
<td align="right"><? echo $harga_alat_geordi3; ?></td>
<td align="right"><? echo $total_alat_geordi3; ?></td>
</tr>
<tr>
<td align="left"><? echo $alat_geordi4; ?></td>
<td align="right"><? echo $jumlah_alat_geordi4; ?></td>
<td align="right"><? echo $harga_alat_geordi4; ?></td>
<td align="right"><? echo $total_alat_geordi4; ?></td>
</tr>
<tr>
<td colspan="3" align="right">Total Harga</td>
<td align="right"><? echo $total_harga; ?></td>
</tr>
<tr>
<td colspan="3" align="right">
Diskon <? echo "( $diskon % )"; ?></td>
<td align="right"><? echo $nilai_diskon; ?></td>
```

```
</tr>
<tr>
<td colspan="3" align="right">Jumlah harus dibayar</td>
<td align="right"><? echo $total_harga_dibayar; ?></td>
</tr>
</table>
</center>
</body>
</html>
```

Berikut adalah contoh script untuk menggabungkan string :

File : Lat11.php

```
<?php
// inisiasi variabel
$a = "USS Enterprise";
$b = "Menurut catatan kapten";
$c = "Mengunjungi Planet Vulcan";
// alternatif pertama
$alt1 = $a . " " . $c . ", " . $b . ".";
// alternatif kedua
$alt2 = $b . ", " . $a . " " . $c . ".";
?>
<html>
<head>
<title>Menggabungkan String</title>
</head>
<body>
String yang pertama adalah: <br>
<?php echo $alt1; ?>
<br><br>
String yang kedua adalah: <br>
<?php echo $alt2; ?>
</body>
</html>
```

1.6.2. Operator Logika

Operator ini akan membandingkan TRUE atau FALSE. Seperti bahasa C, PHP mendefinisikan *False* dengan 0 dan *True* dengan 1.

Operator	Operasi
==	Sama dengan
!=	Tidak sama dengan
<	Lebih kecil
>	Lebih besar
<=	Lebih kecil sama dengan

>=	Lebih besar sama dengan
AND atau &&	Logika AND
OR atau ||	Logika OR
XOR	Logika XOR
!	Logika NOT

Tabel kebenaran operasi logika :

P	Q	P AND q	p OR q	p XOR q	!p
True	True	True	True	False	False
True	False	False	True	True	False
False	True	False	True	True	True
False	False	False	False	False	True

1.6.3. Operator Bitwise

Operator ini digunakan untuk memanipulasi bit-bit dari nilai data. Operator ini beserta artinya dapat dilihat pada tabel berikut :

Operator	Operasi
>>	Pergeseran bit ke kanan
<<	Pergeseran bit ke kiri
~	Komplemen satu atau NOT
&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR (Exclusive OR)

Hubungan bit dan hasilnya untuk operator bitwise logika :

BIT		AND	OR	XOR
b1	b2			
1	1	1	1	0
1	0	0	1	1
0	1	0	1	1
0	0	0	0	0

Bitwise 12 dan 10:

```
    1   1   0   0     (12)
&   1   0   1   0     (10)
   ---------------------------------
    1   0   0   0     (08)
```

1.6.4. Operator Lain

Selain operator di atas, PHP masih mempunyai operator lain, berikut ini beberapa operator yang sering digunakan dalam *script*.

Operator	Arti
.	Penggabungan string
=	Penugasan/pengisian nilai
$	Mengacu pada variabel

1.7. STRUKTUR KENDALI

Struktur kendali merupakan pengatur aliran program, mempunyai rangkaian perintah yang harus ditulis untuk memenuhi beberapa keadaan, yaitu :

- Mengulang suatu perintah jika terpenuhi suatu kondisi.
- Melanjutkan sebuah pernyataan bila kondisi terpenuhi.
- Memilih sebuah pilihan dari beberapa alternatif bila kondisi terpenuhi.

Struktur kendali dapat dibagi menjadi dua jenis, yaitu struktur kendali percabangan (pengambilan keputusan) dan struktur kendali pengulangan (*looping*).

1.7.1. Struktur Kendali Percabangan

Struktur kendali percabangan (pengambilan keputusan) struktur kendali yang berfungsi untuk melakukan pemilihan atas perintah yang akan dijalankan sesuai dengan kondisi tertentu. Ada empat perintah percabangan dalam PHP, yaitu ***if, if..else, if..elseif***, dan ***switch***.

Perintah if

Perintah ***if*** digunakan untuk menjalankan satu atau lebih perintah berdasarkan suatu kondisi. Sintaks penulisan perintah ***if*** adalah sebagai berikut :

```
if (kondisi)
{
  pernyataan yang akan dijalankan apabila kondisi benar
}
```

Pada bentuk ini, bagian *pernyataan* akan dijalankan hanya kalau bagian *kondisi* bernilai benar. Berikut ini adalah contoh penggunaan struktur kendali ***if*** :

File : Lat12.php

```
<html>
  <head>
    <title>Struktur Kendali IF</title>
  </head>
  <body>
  <?php
    $x = 12;
    if ($x > 10)
    {
      print("Selamat Siang");
    }
  ?>
  </body>
</html>
```

Latihan :

Buatlah script untuk menentukan jenis kelamin. Jika nilainya = M maka Sex = LAKI-LAKI. Simpanlah dengan nama Lat13.php

Perintah if…else

Perintah ***if...else*** digunakan untuk memilih salah satu pernyataan berdasarkan suatu kondisi. Perintah ini akan menjalankan pernyataan tertentu bila kondisi bernilai benar dan akan menjalankan pernyataan yang lain jika kondisi bernilai salah. Sintaks penulisannya adalah sebagai berikut :

```
if (kondisi)
{
     pernyataan_1
}
else
{
     pernyataan_2
}
```

Pada bentuk ini *pernyataan_1* dijalankan kalau kalau *kondisi* bernilai benar, dan *pernyataan_2* dijalankan apablila *kondisi* bernilai salah. Contoh penggunaan pernyataan ***if...else***, sebagai berikut :

File : Lat14.php

```
<html>
  <head>
    <title>Struktur Kendali IF...ELSE</title>
  </head>
  <body>
  <?php
    $x = 9;
    if ($x > 10)
    {
      print("Selamat Siang");
    }
    else
    {
      print("Selamat Pagi");
    }
  ?>
  </body>
</html>
```

Latihan :

Buatlah sebuah script untuk menentukan jenis kelamin. Jika nilainya = M maka Sex = LAKI-LAKI, selain itu Sex = WANITA. Simpanlah dengan nama latihan15.php

Perintah if...elseif

Perintah ***if...elseif*** digunakan untuk menjalankan suatu pernyataan dengan melibatkan lebih dari satu kondisi. Sintaks penulisannya sebagai berikut :

```
if (kondisi_1)
{
     pernyataan_1
}
elseif (kondisi_2)
{
     pernyataan_2
}
else
{
     pernyataan_3
}
```

Pada bentuk ini *pernyataan_1* dijalankan kalau kalau *kondisi_1* bernilai benar, dan *pernyataan_2* dijalankan apabila *kondisi_2* bernilai benar. Sedangkan *Pernyataan_3* akan dijalankan apabila *kondisi_1* dan *kondisi_2* bernilai salah. Contoh program yang menggunaan perintah *if...elseif* adalah sebagai berikut :

File : Lat16.php

```
<html>
  <head>
    <title>Struktur Kendali IF...ELSEIF</title>
  </head>
  <body>
  <?php
    $waktu = getdate();
    if ($waktu[hours] <= 10)
    {
      print("Selamat Pagi");
    }
    elseif ($waktu[hours] <= 15)
    {
      print("Selamat Siang");
    }
    elseif ($waktu[hours] <= 18)
    {
      print("Selamat Sore");
    }
    else
    {
      print("Selamat Malam");
    }
  ?>
  </body>
```

```
</html>
```

Latihan :

Buatlah script untuk menentukan kelulusan seorang siswa terhadap Mata Kuliah PI dengan kisaran nilai : > 70 nilainya “A”, >= 50 nilainya “B”, >=30 nilainya “C”, >= 10 nilainya “D”, selain itu pesannya tidak Lulus. Simpanlah dengan nama Lat17.php

Perintah switch

Perintah ***switch*** digunakan sebagai alternatif pengganti dari perintah ***if...elseif***. Dengan perintah ini program percabangan akan lebih mudah dibuat dan dipelajari. Sintaks penulisan perintah ***switch*** adalah sebagai berikut :

```
switch (kondisi)
{
     case konstanta_1:
          pernyataan_1;
     break;
     case konstanta_2:
          pernyataan_2;
     break;
     default:
          pernyataan_3;
}
```

Perintah ***switch*** akan menyeleksi kondisi yang diberikan dan membandingkan hasilnya dengan konstanta-konstanta yang berada pada ***case***. Pembandingan akan dimulai dari *konstanta_1* sampai konstanta terakhir. Jika hasil dari kondisi sama dengan nilai konstanta tertentu, maka pernyataan pada konstanta tersebut akan dijalankan sampai ditemukan pernyataan ***break***. Jika hasil dari kondisi tidak ada yang sama dengan konstanta-konstanta yang diberikan, maka pernyataan pada ***default*** yang akan dijalankan. Berikut ini contoh penggunaan struktur kendali dengan perintah ***switch*** :

File : Lat18.php

```
<html>
  <head>
    <title>Struktur Kendali dengan Switch</title>
  </head>
  <body>
  <?php
    $english = date("l");
    switch($english_day)
    {
      case "Monday":
        $indonesian = "Senin";
        break;
      case "Tuesday":
        $indonesian = "Selasa";
        break;
      case "Wednesday":
        $indonesian = "Rabu";
        break;
      case "Thursday":
        $indonesian = "Kamis";
        break;
      case "Friday":
        $indonesian = "Jumat";
        break;
      case "Saturday":
        $indonesian = "Sabtu";
        break;
      default:
        $indonesian = "Minggu";
    }
    print("<h2>Hari ini adalah hari $indonesian</h2>")
  ?>
  </body>
</html>
```

Catatan: nilai pada fungsi **date(“l”)** adalah huruf *l*, bukan angka *1*. Untuk lebih jelasnya lihat pada bagian *Fungsi Tanggal dan Jam*.

Latihan :

Buatlah script untuk menentukan kelulusan seperti pada contoh “Lat17.php” dengan menggunakan ***switch***. Lalu simpan dengan nama file Lat19.php

1.7.2. Struktur kendali pengulangan

Struktur kendali pengulangan digunakan untuk mengulang suatu perintah sebanyak yang diinginkan. Ada tiga jenis perintah pengulangan dalam PHP, yaitu ***for, while,*** **dan** ***do...while***.

Perintah for.

Perintah ***for*** digunakan untuk mengulangi suatu perintah dengan jumlah pengulangan yang sudah diketahui. Pada perintah ini tidak perlu menuliskan suatu kondisi untuk diuji. Kita hanya perlu menuliskan nilai awal dan akhir variabel penghitung. Nilai variabel penghitung akan secara otomatis bertambah atau berkurang tiap kali sebuah pengulangan dilaksanakan. Sintaks penulisan perintah ***for*** adalah sebagai berikut :

```
for (nilai_awal; nilai_akhir; penambahan/pengurangan)
{
     pernyataan yang dijalankan
}
```

Contoh struktur pengulangan dengan ***for*** :

File : Lat20.php

```
<html>
  <head>
    <title>Struktur pengulangan dengan FOR</title>
  </head>
  <body>
  <?php
    for($i=1; $i<=7; $i++)
    {
      print("<h$i>Ini adalah contoh heading $i</h$i>");
    }
  ?>
  </body>
</html>
```

Latihan :

Buatlah script untuk menampilkan bilangan bulat positif dengan pertambahan angkanya = 2 sampai 100. simpanlah dengan nama Lat21.php

Perintah while

Perintah ***while*** digunakan untuk mengulangi suatu perintah sampai jumlah yang belum bisa ditentukan. Pengulangan akan terus berjalan selama kondisi masih bernilai benar. Sintaks penulisan perintah ***while*** adalah sebagai berikut :

```
while (kondisi)
{
   pernyataan yang akan dijalankan
}
```

Contoh struktur pengulangan dengan ***while*** :

File : Lat22.php

```
<html>
  <head>
    <title>Struktur pengulangan dengan WHILE</title>
  </head>
  <body>
  <?php
    $i=1;
    while ($i <= 9)
    {
      print("Ini pengulangan yang ke-$i<br>");
      $i++;
    }
  ?>
  </body>
</html>
```

Perintah do…while

Dengan perintah ***do...while***, proses pengulangan akan terus dikerjakan jika kondisi yang diperiksa di ***while*** masih bernilai benar. Proses pengulangan akan dihentikan jika kondisi sudah bernilai salah. Sintaks penulisannya sebagai berikut :

```
do
{
   pernyataan yang dijalankan
}
while (kondisi);
```

Perbedaan antara perintah ***while*** dengan ***do...while*** adalah terletak dari kondisi yang diperiksa. Pada perintah ***while***, kondisi yang diperiksa terletak di awal perulangan, sehingga sebelum masuk ke dalam perulangan ***while*** kondisi harus bernilai benar. Sedangkan pada perintah ***do...while***, kondisi diperiksa di akhir perulangan. Ini berarti bahwa paling sedikit sebuah perulangan akan dilakukan oleh perintah ***do...while***, karena untuk masuk ke perulangan tidak ada kondisi yang harus dipenuhi. Contoh struktur pengulangan dengan ***do...while*** :

File : Lat23.php

```
<html>
  <head>
    <title>Struktur pengulangan dengan do...while</title>
  </head>
  <body>
  <?php
    $i = 1;
    do
    {
      print("Ini juga pengulangan yang ke-$i<br>");
      $i++;
    }
    while ($i <= 6);

    $j = 5;
    do
    {
      print("<br>Perulangan ini kondisinya tidak terpenuhi<br>");
      $j++;
    }
    while($j <= 3);
  ?>
  </body>
</html>
```

1.8. ARRAY

Array merupakan salah satu fasilitas untuk menyimpan data secara berurutan. Dalam array data tersimpan dengan menggunakan indeks untuk memudahkan pencarian kembali data tersebut. Berbeda dengan variabel yang hanya dapat menyimpan satu nilai, array mampu menampung sejumlah nilai.

1.8.1. Pendeklarasian Array

Data yang terdapat dalam array disebut elemen-elemen array dan letak urutan masing-masing elemen array tersebut ditunjukkan oleh suatu indeks. Array mempunyai batas atas dan batas bawah, dimana data akan tersimpan diantara kedua batas tersebut. Semua elemen array yang tersimpan mempunyai tipe data yang sama. Array dapat berdimensi satu, dua, tiga atau lebih. Array berdimensi satu (*one-dimensional array*) mewakili bentuk suatu vektor. Array berdimensi dua (*two-dimensional array*) mewakili bentuk suatu matrix. Array berdimensi tiga ((*three-dimensional array*) mewakili bentuk suatu ruang.

Beberapa cara mendeklarasikan array :

File : Lat24.php

```
<html>
<head>
  <title>Menggunakan Array</title>
</head>
<body>
<?php
  #Cara pendeklarasian array pertama
  $warna = array("merah","kuning","hijau","biru");

  print("<p>Pendeklarasian array pertama:<br>");
  print($warna[2]);

  #Cara pendeklarasian array kedua
  $kota[] = "Jakarta";
  $kota[] = "Bandung";
  $kota[] = "Surabaya";
  $kota[] = "Solo";
  $kota[] = "Semarang";
  $kota[] = "Jogjakarta";

  print("<p>Pendeklarasian array kedua:<br>");
  print($kota[5]);

  #Cara pendeklarasian array ketiga
  $kota[0] = "Jakarta";
  $kota[5] = "Bandung";
  $kota[2] = "Surabaya";
  $kota[] = "Solo";
  $kota[] = "Semarang";
  $kota[gudeg] = "Jogjakarta";

  print("<p>Pendeklarasian array ketiga:<br>");
  print("$kota[5] <br>");
```

```
  print("$kota[7] <br>");
  print("Saat ini saya sedang kuliah di $kota[gudeg]");
?>
</body>
</html>
```

Contoh penggunaan array multidimensi :

File : Lat25.php

```
<html>
<head>
  <title>Array Multidimensi</title>
</head>
<body>
<?php
  $kota = array(
    "jatim"=>array("Surabaya","Malang","Jember","Bondowoso"),
    "jabar"=>array("Bandung","Bogor","Tangerang"),
    "jogja"=>array("Sleman","Bantul","Kulonprogo","Wates"));

  print("Saya berasal dari ".$kota[jatim][3]."
    dan sekarang saya kuliah di ".$kota[jogja][1]);
?>
</body>
</html>
```

1.9. FUNGSI

Dalam pembuatan program seringkali kita membutuhkan sekumpulan perintah yang akan digunakan berulang kali. Tentunya sangat merepotkan apabila perintah-perintah tersebut harus diketik ulang setiap kali akan digunakan. Hal ini dapat dihindari dengan menggunakan subrutin. Subrutin adalah sekumpulan perintah yang diberi nama dan kemudian dapat kita panggil sewaktu-waktu. Dalam pemrograman dikenal dua macam subrutin, yaitu prosedur dan fungsi.

Perbedaan antara kedua macam subrutin ini adalah, fungsi mengembalikan nilai tertentu, sedangkan prosedur tidak. Bahasa PHP hanya mengenal perintah untuk membuat fungsi, yaitu ***function***. Pada sebuah subrutin diperlukan argumen, yaitu nilai-nilai yang harus dimasukkan pada saat subrutin tersebut dipanggil. Tapi tidak

perlu khawatir kita dapat membuat prosedur dengan menggunakan perintah ***function*** tetapi tidak dengan menggunakan perintah ***return***. Perintah ***return*** digunakan untuk mengembalikan nilai tertentu. Standar penulisan fungsi adalah :

```
Function nama_fungsi(argumen)
{
    kode perintah
}
```

Berikut ini contoh penggunaan fungsi baik yang menggunakan perintah ***return*** maupun yang tidak menggunakan perintah ***return***, perhatikan hasilnya.

Contoh program tanpa perintah ***return*** :

File : Lat26.php

```
<html>
  <head>
    <title>Membuat fungsi tanpa perintah return</title>
  </head>
  <body>
  <?php
    function toko()
    {
      print("www.tokobuku-ku.com<br>toko buku online paling
murah");
    }

    toko();
  ?>
  </body>
</html>
```

Contoh program dengan perintah ***return*** :

File : Lat27.php

```
<html>
  <head>
    <title>Membuat fungsi dengan perintah return</title>
  </head>
  <body>
  <?php
    function cetak_tebal($input)
    {
      $teks  = "<b>";
      $teks .= "$input";
      $teks .= "</b>";
      return($teks);
    }
    print("Kalimat ini belum dicetak tebal <br>\n");
    print(cetak_tebal("Kalau yang ini sudah dicetak tebal"));
  ?>
  </body>
</html>
```

1.9.1. Fungsi dengan Argumen

Argumen adalah suatu nilai yang dimasukkan ke dalam sebuah fungsi. Secara default sebuah argumen bersifat *pass by value*, yang berarti hanya nilainya saja yang dibutuhkan sehingga nilai pada variabel tersebut tidak mengalami perubahan setelah fungsi dijalankan. Perhatikan contoh berikut :

File : Lat28.php

```
<html>
  <head>
    <title>Fungsi dengan argumen pass by value</title>
  </head>
  <body>
  <?php
    function luas($panjang, $lebar)
    {
      $luas = $panjang*$lebar;
      return($luas);
    }

    print("Luasnya = ".luas(3,9));
  ?>
  </body>
</html>
```

Selain argumen yang telah dijelaskan di atas, ada juga argumen yang bersifat *pass by reference*, yang digunakan ketika kita ingin mengubah nilai sebuah argumen. Argumen yang bersifat *pass by reference* ini nilainya akan berubah setelah fungsi dijalankan. Untuk membuat argumen *pass by reference* kita akan menggunakan karakter "&" di depan nama argumennya. Untuk lebih jelasnya perhatikan contoh berikut :

File : Lat29.php

```
<html>
  <head>
    <title>Fungsi dengan argumen pass by reference</title>
  </head>
  <body>
  <?php
    function sambung(&$string)
    {
      $string .= "dan tambahannya";
    }
    $str = "Ini sebuah contoh string ";
    sambung($str);
    echo $str;
  ?>
  </body>
</html>
```

1.9.2. Fungsi Built-In

Bagian ini akan membahas fungsi-fungsi yang telah disediakan oleh PHP. Kita juga dapat mempelajari fungsi-fungsi lainnya dalam berbagai manual yang banyak terdapat di internet.

Fungsi-fungsi Umum

echo *string_first,string_second,..string_last*

Fungsi ***echo*** sering digunakan untuk mengirim satu atau lebih parameter yang dipisahkan dengan tanda baca koma (,) ke *browser*. Sebagai contoh :

File : Lat30.php

```
<html>
  <head>
    <title>Fungsi echo</title>
  </head>
  <body>
  <?php
    echo "Aku datang", 1, 2.0, 3.5, " kamu pergi";
  ?>
  </body>
</html>
```

print(*string_output***)**

Fungsi ini hampir sama dengan fungsi ***echo*** yaitu untuk mengirim output ke *browser*.

Sebagai contoh :

File : Lat31.php

```
<html>
  <head>
    <title>Fungsi print</title>
  </head>
  <body>
  <?php
    print("Hai semua, kunjungi www.toko_buku.com ya !!!");
  ?>
  </body>
</html>
```

printf(*string_format, argumen ...***)**

Fungsi ini hampir sama dengan fungsi ***print***, yaitu untuk mengirim output ke *browser*.

Fungsi ini sering digunakan untuk menampilkan hasil yang formatnya bisa diatur.

Format	Keterangan
D	Integer, notasi desimal
B	Integer, notasi binary
O	Integer, notasi oktal
X	Integer, notasi hexadesimal, dinyatakan dalam huruf kecil
X	Integer, notasi hexadesimal, dinyatakan dalam huruf kapital

C	Karakter yang nilai ASCII-nya dinyatakan dalm argumen
S	String
F	Double (bilangan *real*)

Contoh penggunaan fungsi ***printf*** adalah :

File : Lat32.php

```
<html>
  <head>
    <title>Fungsi printf</title>
  </head>
  <body>
  <?php
    printf("%s %d %f <br>\n","string", 10, 3.14);
  ?>
  </body>
</html>
```

copy(*string_source, string_destination***)**

Fungsi ini digunakan untuk menyalin suatu file yang ditentukan dalam argumen *source* (lokasi asal file) menuju ke *destination* (lokasi tujuan). Akan menghasilkan *true* jika berhasil.

File : Lat33.php

```
<html>
  <head>
    <title>Fungsi copy</title>
  </head>
  <body>
  <?php
    if (copy("data.txt","tmp/data.txt"))
    {
      print("data.txt disalin ke /tmp");
    }
    else
    {
      print("data.txt tidak dapat disalin !");
    }
  ?>
  </body>
</html>
```

include(*string_filename***)**

Argumen *filename* merupakan nama file yang akan dijalankan dengan fungsi ***include***. Fungsi ini sering digunakan untuk menyisipkan file yang disebutkan dalam argumen. Amati contoh di bawah ini :

File : Lat34.php

```
<html>
  <head>
    <title>Fungsi include</title>
  </head>
  <body>
  <?php
    include("fungsi_print.php");
  ?>
  </body>
</html>
```

phpinfo()

Dengan fungsi ini akan didapatkan informasi tentang versi PHP, pembuatnya, sistem operasi pada *web server*, konfigurasi variabel, dan sebagainya. Contoh :

File : Lat35.php

```
<html>
  <head>
    <title>Fungsi phpinfo()</title>
  </head>
  <body>
  <?php
    phpinfo();
  ?>
  </body>
</html>
```

phpversion()

Fungsi ini hanya akan memberi informasi mengenai versi PHP yang kita gunakan.

Contoh :

File : Lat36.php

```
<html>
  <head>
    <title>Fungsi phpversion()</title>
  </head>
  <body>
  <?php
    printf("Versi PHP-ku adalah %s", phpversion());
  ?>
  </body>
</html>
```

strlen(*string_text***)**

Fungsi ini akan mengembalikan nilai integer yang merupakan panjang string dalam argumen text. Lihat contoh berikut ini :

File : Lat37.php

```
<html>
  <head>
    <title>Fungsi strlen</title>
  </head>
  <body>
  <?php
    print(strlen("tokobukuku"));
  ?>
  </body>
</html>
```

ord(*string_character***)**

Fungsi ini akan mengembalikan nilai ASCII dari karakter pada argumen. Jadi argumennya hanyalah sebuah karakter. Perhatikan contoh di bawah ini :

File : Lat38.php

```
<html>
  <head>
    <title>Fungsi ord</title>
  </head>
  <body>
  <?php
    print(ord("B"));
  ?>
  </body>
</html>
```

strtolower(*string_text*)

Fungsi ini akan mengubah argumen menjadi huruf kecil semua. Untuk jelasnya Anda dapat mencoba contoh berikut ini :

File : Lat39.php

```
<html>
  <head>
    <title>Fungsi strtolower</title>
  </head>
  <body>
  <?php
    print(strtolower("WWW.TokoBukuKu.Com"));
  ?>
  </body>
</html>
```

strtoupper(*string_text*)

Fungsi ini merupakan kebalikan dari fungsi ***strtolower***. Semua huruf dalam argumen *text* yang bertipe string akan diubah menjadi huruf besar dan menjadi keluaran dari fungsi ini. Perhatikan contoh di bawah ini :

File : Lat40.php

```
<html>
  <head>
    <title>Fungsi strtoupper</title>
  </head>
  <body>
  <?php
    print(strtoupper("WWW.TokoBukuKu.Com"));
  ?>
  </body>
</html>
```

Fungsi Tanggal dan Jam

date(*string_format, integer_timestamp***)**

Fungsi ***date*** mengembalikan tanggal atau waktu tergantung dari argumen yang dimasukkan. Daftar *string_format* terdapat dalam tabel di bawah ini. Argumen *timestamp* bersifat *optional*.

Kode	Keterangan
A	am atau pm
A	AM atau PM
D	Tanggal (*date*)
D	Nama hari (singkatan)
F	Nama bulan
H	Jam 01 sampai 12
H	Jam 00 sampai 23
I	Menit
J	Tanggal
L	Nama hari (lengkap)
M	Bulan (dalam angka)
M	Bulan (singkatan)
Y	Tahun dalam dua digit
Y	Tahun dalam empat digit
Z	Hari ke- dari tahun

Perhatikan contoh :

File : Lat41.php

```
<html>
  <head>
    <title>Fungsi date</title>
  </head>
  <body>
  <?php
    print("Yogyakarta, ".date("d F Y"));
    print("<br>\n");
    print(date("h:i A l"));
  ?>
  </body>
</html>
```

`getdate(`*`integer_timestamp`*`)`

Fungsi ***getdate*** digunakan untuk menghasilkan waktu dengan keluaran bertipe *array*. Argumen *timestamp* bersifat *optional*. Daftar elemen ***getdate*** dapat dilihat dalam tabel berikut :

Elemen	Keterangan
hours	Jam
mday	Hari
minutes	Menit
mon	Bulan dalam digit
month	Bulan
seconds	Detik
wday	Hari dalam digit
weekday	Hari
yday	Hari ke- dari tahun
year	Tahun

Perhatikan contoh berikut :

File : Lat42.php

```
<html>
  <head>
    <title>Ucapan selamat dengan fungsi getdate</title>
  </head>
  <body>
  <?php
    $waktu = getdate();
    if($waktu[hours] <= 9)
    {
      echo "Selamat Pagi";
    }
    elseif($waktu[hours] <= 14)
    {
      echo "Selamat Siang";
    }
    elseif($waktu[hours] <= 18)
    {
      echo "Selamat Sore";
    }
    else
    {
      echo "Selamat Malam";
    }
  ?>
  </body>
</html>
```

Sleep(*integer_seconds***)**

Fungsi ***sleep*** menyebabkan proses tertunda selama nilai pada argumen *seconds*, seperti tampak pada contoh berikut :

File : Lat43.php

```
<html>
  <head>
    <title>Fungsi sleep</title>
  </head>
  <body>
  <?php
    print(microtime());
    sleep(10);
    print("<br>\n");
    print(microtime());
  ?>
  </body>
</html>
```

usleep(*integer_microseconds***)**

Fungsi ini menyebabkan proses tertunda selama nilai pada argumen *microseconds*. Perhatikan contoh berikut ini :

File : Lat44.php

```
<html>
  <head>
    <title>Fungsi usleep</title>
  </head>
  <body>
  <?php
    print(microtime());
    usleep(50);
    print("<br>\n");
    print(microtime());
  ?>
  </body>
</html>
```

Fungsi Matematika

abs(*number_value***)**

Fungsi ini akan menghasilkan nilai mutlak dari nilai pada argumen. Jika nilai pada argumen bernilai positif, maka fungsi ***abs*** akan mengembalikan nilai itu sendiri. Jika nilai argumen bernilai negatif, fungsi ***abs*** akan mengalikan argumen dengan negatif satu (-1). Perhatikan contoh berikut ini :

File : Lat45.php

```
<html>
  <head>
    <title>Fungsi abs</title>
  </head>
  <body>
  <?php
    print(abs(-10));
  ?>
  </body>
</html>
```

cos(*double_angle***)**

Fungsi ini akan mengembalikan nilai cosinus dari nilai sudut (dalam radian). Perhatikan contoh di bawah ini :

File : Lat46.php

```
<html>
  <head>
    <title>Fungsi cos</title>
  </head>
  <body>
  <?php
    print (cos(4 * pi()));
  ?>
  </body>
</html>
```

exp(*double_power***)**

Fungsi ini akan mengembalikan nilai eksponen dari argumen. Lihat contoh berikut ini:

File : Lat47.php

```
<html>
  <head>
    <title>Fungsi exp</title>
  </head>
  <body>
  <?php
    print (exp(1));
  ?>
  </body>
</html>
```

log10(*double_value***)**

Fungsi ini akan mengembalikan nilai log10 dari argumen. Perhatikan contoh berikut :

File : Lat48.php

```
<html>
  <head>
    <title>Fungsi log10</title>
  </head>
  <body>
  <?php
    print (log10(100));
  ?>
  </body>
</html>
```

pi()

Fungsi ini akan memberikan nilai pendekatan dari *pi* (22/7). Perhatikan contoh berikut:

File : Lat49.php

```
<html>
  <head>
    <title>Fungsi pi()</title>
  </head>
  <body>
  <?php
    print (pi());
  ?>
  </body>
</html>
```

pow(*double_base*, *double_power***)**

Fungsi ini digunakan untuk menghasilkan nilai basis pangkat nilai pada power. Perhatikan contoh berikut :

File : Lat50.php

```
<html>
  <head>
    <title>Fungsi pow()</title>
  </head>
  <body>
  <?php
    print (pow(2, 10));
  ?>
  </body>
</html>
```

round(*double_value***)**

Fungsi ini akan mengubah nilai pada argumen (bertipe *double*) menjadi integer yang terdekat (pembulatan). Perhatikan contoh berikut :

File : Lat51.php

```
<html>
  <head>
    <title>Fungsi round()</title>
  </head>
  <body>
  <?php
    print (round(2, 4)."<br>");
    print (round(2, 5)."<br>");
    print (round(2, 6)."<br>");
  ?>
  </body>
</html>
```

sin(*double_single***)**

Fungsi ini menghasilkan nilai sinus dari nilai sudut (dalam radian) pada argumen.

Perhatikan contoh berikut :

File : Lat52.php

```
<html>
  <head>
    <title>Fungsi sin()</title>
  </head>
  <body>
  <?php
    print (sin(2 * pi()));
  ?>
  </body>
</html>
```

sqrt(*double_value***)**

Fungsi ini digunakan untuk mencari nilai akar dari argumen value. Perhatikan contoh berikut :

File : Lat53.php

```
<html>
  <head>
    <title>Fungsi sqrt()</title>
  </head>
  <body>
  <?php
    print (sqrt(100.0));
  ?>
  </body>
</html>
```

tan(*double_angle***)**

Fungsi ini akan menghasilkan nilai tangen dari sudut (dalam radian) pada argumen angle. Perhatikan contoh berikut :

File : Lat54.php

```
<html>
  <head>
    <title>Fungsi tan()</title>
  </head>
  <body>
  <?php
    print (tan(2 * pi()));
  ?>
  </body>
</html>
```

Fungsi Array dan Variabel

define(*string_constant, string_value, boolean non_case_sensitive***)**

Nilai pada argumen bisa bertipe string, integer, atau doble. Argumen *non case sensitive* bersifat opsional. Default fungsi ini adalah *case sensitive*. Contoh implementasinya pada penulisan program adalah :

File : Lat55.php

```
<html>
  <head>
    <title>Fungsi define()</title>
  </head>
  <body>
  <?php
    define("ini", "SAYA SEORANG");
    define("itu", "WEB DEVELOPER");
    print(ini . "<br>\n");
    print(itu . "<br>\n");
  ?>
  </body>
</html>
```

empty(*variable***)**

Fungsi ini akan mengembalikan nilai true jika variabel belum mempunyai nilai, dan sebaliknya false jika variabel telah diberi nilai. Perhatikan contoh berikut :

File : Lat56.php

```
<html>
  <head>
    <title>Fungsi empty()</title>
  </head>
  <body>
  <?php
    if(empty($nama))
    {
      print("Tolong isi nama Anda segera!");
    }
    else
    {
      print("Nama Anda adalah $nama");
    }
  ?>
  </body>
</html>
```

is_array(*expression***)**

Fungsi ini akan mengembalikan nilai true jika ekspresi adalah array, dan sebaliknya false. Perhatikan contoh berikut :

File : Lat57.php

```
<html>
  <head>
    <title>Fungsi is_array()</title>
  </head>
  <body>
  <?php
    $isi = array("buku", "pensil", "pena");
    if(is_array($isi))
    {
      print("Variabel isi adalah variabel array");
    }
  ?>
  </body>
</html>
```

is_double(*expression*)

Fungsi ini akan mengembalikan nilai true jika nilai pada ekspresi bertipe double. Perhatikan contoh berikut :

File : Lat58.php

```
<html>
  <head>
    <title>Fungsi is_double()</title>
  </head>
  <body>
  <?php
    $dollar = 8000.25;
    if(is_double($dollar))
    {
      print("Variabel dollar bertipe double");
    }
  ?>
  </body>
</html>
```

is_integer(*expression*)

Fungsi ini akan mengembalikan nilai true jika nilai pada ekspresi bertipe integer. Perhatikan contoh berikut :

File : Lat59.php

```
<html>
  <head>
    <title>Fungsi is_integer()</title>
  </head>
  <body>
  <?php
    $harga = 8000;
    if(is_integer($harga))
    {
      print("Variabel harga bertipe integer");
    }
  ?>
  </body>
</html>
```

is_string(*expression***)**

Fungsi ini akan mengembalikan nilai true jika nilai pada ekspresi bertipe string Perhatikan contoh berikut :

File : Lat60.php

```
<html>
  <head>
    <title>Fungsi is_string()</title>
  </head>
  <body>
  <?php
    $hero = "Spiderman";
    if(is_string($hero))
    {
      print("Variabel hero bertipe string");
    }
  ?>
  </body>
</html>
```

isset(*variable***)**

Fungsi ini akan mengembalikan nilai true jika variabel telah memiliki nilai, dan sebaliknya false jika variabel belum diberi nilai. Perhatikan contoh berikut :

File : Lat61.php

```
<html>
  <head>
    <title>Fungsi isset()</title>
  </head>
  <body>
  <?php
    if(isset($nama))
    {
      print("Nama Anda adalah $nama");
    }
    else
    {
      print("Tolong isi nama Anda segera!");
    }
  ?>
  </body>
</html>
```

count(*variable_array***)**

Fungsi ini akan mengembalikan nilai berupa jumlah elemen array dari argumen variabel. Jika variabel belum diisi maka fungsi akan mengembalikan nilai 0. Jika variabel bukan array, fungsi akan mengembalikan nilai 1. Perhatikan contoh berikut :

File : Lat62.php

```
<html>
  <head>
    <title>Fungsi count()</title>
  </head>
  <body>
  <?php
    $hero = array("Spiderman", "Wolvrine", "Cyclop");
    print(count($hero));
  ?>
  </body>
</html>
```

1.10. Include dan Require

Apakah *require()* dan *include()* itu benar-benar sama cara kerjanya? Tentu saja tidak, sebab jika sama fungsinya tentu tidak selayaknya dibedakan fungsinya. Perbedaan mendasar antara kedua fungsi ini adalah:

- Fungsi *require()* akan selalu digantikan oleh isi dari file yang ditunjuk dalam fungsi ini dan tidak dapat digunakan dalam percabangan/perkondisian (seperti perkondisian "jika ini maka *require* file anu") , karena file yang ditunjuk akan selalu direferensi tanpa peduli kondisi struktur/ aliran *script*.
- Fungsi *include()* akan mengatur pembacaan file yang ditunjuk dapat sesuai dengan kondisi struktur/ aliran *script*, sehingga fungsi ini dapat digunakan pada percabangan/ perkondisian.

Melihat ciri-ciri di atas, *require()* akan sesuai digunakan untuk mereferensi file yang berisikan variabel dan fungsi-fungsi global yang digunakan pada seluruh bagian dari *script* utama. Sementara *include()* umumnya digunakan untuk menyisipkan kode program/*script* atau tag HTML pada program/*script* utama, misalkan untuk *header* atau *footer* setiap halaman dalam sebuah situs.

Catatan yang penting untuk kedua fungsi ini, parser PHP akan meninggalkan mode PHP dan kembali ke mode HTML standar pada saat membaca file yang ditunjukkan oleh kedua fungsi ini. Itu sebabnya pada contoh di atas, semua file yang ditunjuk oleh fungsi-fungsi ini selalu dimulai dengan tag `<?php` dan diakhiri dengan tag `?>` untuk mengembalikan mode file ke mode *script* PHP.

Berikut adalah contoh penggunaan ***Include*** dan ***Require*** secara bersamaan dalam satu halaman web.

File : Lat63.php

```
<html>
<head>
<title>Persenjataan dan Perlengkapan Perang Enterprise</title>
</head>
<?php
// Standar Senjata Kapal Perang Kelas Galaxy
require("torpedo.php");
require("laser.php");
// Standar Perisai Kapal Perang Kelas Galaxy
include("shielding.php");
// Standar Mesin Penggerak Kapal Perang Kelas Galaxy
include("impuls.php");
include("warp.php");
?>
<body>
LCAR: Cek kesiapan perlengkapan perang USS Enterprise NCC-1701-D
<ol type="1">
<li> Torpedo : <?php echo $torpedo; ?>
<li> Laser : <?php echo $laser; ?>
<li> Perisai : <?php echo $shielding; ?>
<li> Mesin Impuls : <?php echo $impuls; ?>
<li> Mesin Warp : <?php echo $warp; ?>
</ol>
<br>
Commander La Forge, segera laporkan semua sistem persenjataan
telah dicek dan berfungsi dengan baik.
USS Enterprise siap menghadapi Kapal Romulan. <br>
</body>
</html>
```

Selanjutnya coba jalankan script diatas, tentunya masih ada muncul kesalahan. Coba analisa apa penyebab kesalahan dari script diatas? Selanjutnya coba ketikkan script dihalaman berikutnya. Lalu coba jalankan ulang script “Lat63.php”.

File : torpedo.php

```
<?php
$torpedo = "Four Bays Photon Torpedo";
?>
```

File : laser.php

```
<?php
$laser = "Six Laser Canons";
?>
```

File : shielding.php

```
<?php
$shielding = "EM Polarization Shielding";
?>
```

File : impuls.php

```
<?php
$impuls = "Federation Impulse Power System";
?>
```

File : warp.php

```
<?php
$warp = "Matter/Antimatter Reactor (Warp Core)";
?>
```

1.11. Bekerja dengan Form

Listing berikut merupakan contoh penggunaan form sebagai interface masukan data kemudian akan diproses menggunakan php.

File : Lat64.html

```
<html>
<head>
<basefont face="Arial">
</head>
<body>
<center>
<form method="GET" action="Lat65.php">
<table cellspacing="5" cellpadding="5" border="1">
<tr>
<td colspan="2" align="center">
NCC-1701D USS Enterprise<br>

</td>
</tr>
<tr>
<td>
<font size="-1">Silakan Masukkan Nama Anda
</td>
<td>
<input type="text" name="namaofficer" size="20">
</td>
```

```
</tr>
<tr>
<td colspan="2" align="center">
<input type="submit" name="loginofficer" value="Login">
</td>
</tr>
</table>
</form>
</center>
</body>
</html>
```

File : Lat65.php

```
<html>
<head>
<basefont face="Arial">
</head>
<body>
<center>
<font face="Arial" size="-1">
Selamat Datang, <? echo $namaofficer; ?>
<P>
Anda Berada Di Dunia Lain.
<P>
Apakah Anda siap memasuki Dunia Lain?
</font>
</center>
</body>
</html>
```

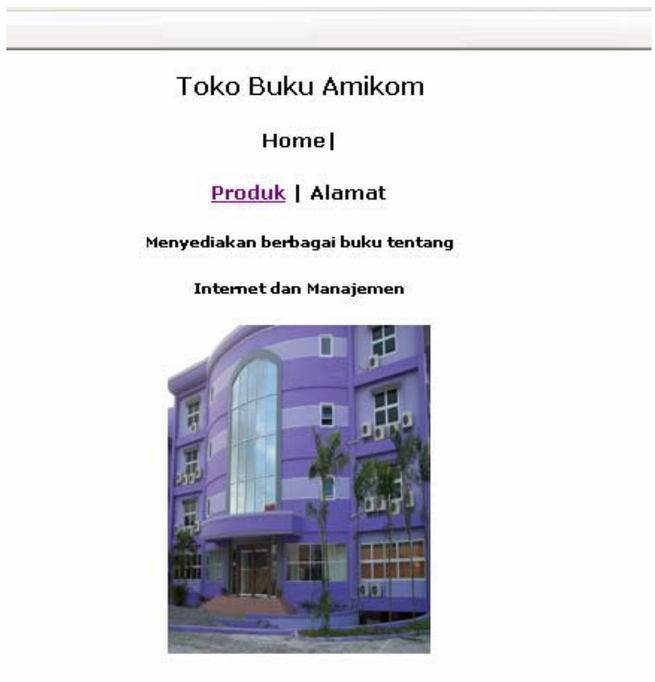
Contoh Lain :

File : home.html

```
<html>
<head>
<title>.::Toko Buku Amikom::.</title>
</head>
<body>
<div align="center"><font size="3" face="Verdana"><b>Toko Buku
Amikom </b></font></div>
<center>
  <p></p>
  <p> </p>
<font face = "verdana" size="-1"><b>
  <td>Home|</p></td>
  <td><a href="produk.htm">Produk</a> |</td>
  <td> Alamat</td>
  </b></font>

  <p><b><font size="1" face="Verdana">Menyediakan
    berbagai buku tentang </font></b></p>
  <p><font size="1"><b><font face="Verdana">Internet
    dan Manajemen</font></b></font></p>
  <p></p>
  <p><img src="amikom%2001.jpg" width="176" height="220"></p>
</center>
</body>
</html>
```

Outputnya :



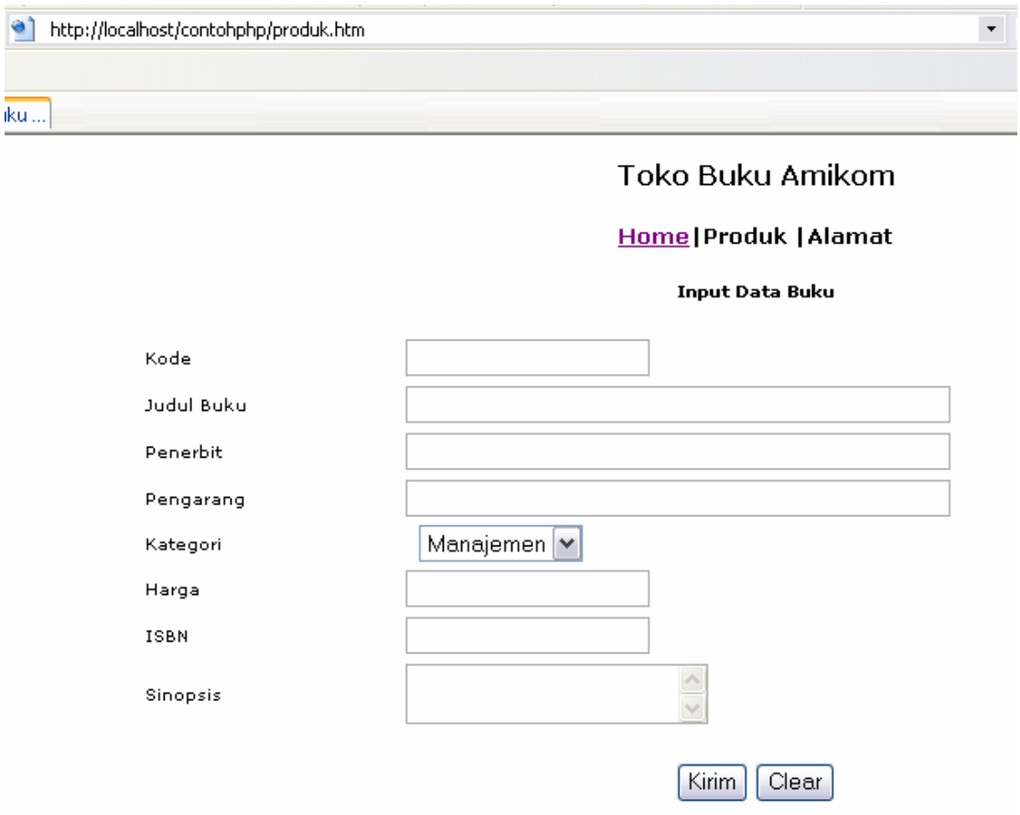
File : produk.html

```
<html>
<head>
<title>.::Toko Buku Amikom::.</title>
</head>
<body>
<div align="center"><font size="3" face="Verdana"><b>Toko Buku
Amikom </b></font></div>
<center>
  <p></p>
  <p> </p>
<font face = "verdana" size="-1"><b>
  <a href="home.htm">Home</a>|Produk |Alamat
  </b></font>

  <p><b><font size="1" face="Verdana">Input
    Data Buku</font></b></p>
     <form method="POST" action="prosesproduk.php">
    <table width="75%" border="0">
      <tr>
        <td width="20%"><font size="1"
face="Verdana">Kode</font></td>
        <td width="77%"><font size="1" face="Verdana">
          <input type="text" name="kodebuku">
          </font></td>
      </tr>
      <tr>
        <td><font size="1" face="Verdana">Judul
          Buku</font></td>
        <td><font size="1" face="Verdana">
          <input name="judulbuku" type="text" size="50">
          </font></td>
      </tr>
      <tr>
        <td><font size="1" face="Verdana">Penerbit</font></td>
        <td><font size="1" face="Verdana">
          <input name="penerbitbuku" type="text" size="50">
          </font></td>
      </tr>
      <tr>
        <td><font size="1" face="Verdana">Pengarang</font></td>
        <td><font size="1" face="Verdana">
          <input name="pengarangbuku" type="text" size="50">
          </font></td>
      </tr>
      <tr>
        <td><font size="1" face="Verdana">Kategori</font></td>
        <td><font size="1" face="Verdana"> 
          <select name="kategori" size="1">
            <option>Internet</option>
            <option selected>Manajemen</option>
          </select>
          </font> <br> </td>
      </tr>
      <tr>
        <td><font size="1" face="Verdana">Harga</font></td>
        <td><font size="1" face="Verdana">
          <input type="text" name="hargabuku">
          </font></td>
      </tr>
      <tr>
```

```
        <td><font size="1" face="Verdana">ISBN</font></td>
        <td><font size="1" face="Verdana">
          <input type="text" name="isbn">
          </font></td>
      </tr>
      <tr>
        <td><font size="1" face="Verdana">Sinopsis</font></td>
        <td><font size="1" face="Verdana">
          <textarea name="sinopsis"></textarea>
          </font></td>
      </tr>
    </table>
    <p>
      <input type="submit" name="Submit" value="Kirim">
      <input type="reset" name="Submit2" value="Clear">
    </p>
  </form>
  <p></p>
  </center>
</body>
</html>
```

Outputnya :



File : prosesproduk.php

```
<html>
<head>
<title>.::Toko Buku Amikom::.</title>
</head>
<body>
<div align="center"><font size="3" face="Verdana"><strong>Toko
Buku Amikom </strong></font></div>
<center>
  <p></p>
  <p> </p>
<font face = "verdana" size="-1"><b>
  <td>Home|</p></td>
  <td><a href="produk.htm">Produk</a> |</td>
  <td> Alamat</td>
  </b></font>

  <p><b><font size="4" face="Verdana">Katalog Buku</font></b></p>
  <p></p>
  <table width="75%" border="1" cellspacing="1" cellpadding="0">
    <tr>
      <td width="24%"><font size="3"
face="Verdana">Kode</font></td>
      <td width="76%"><?php print $kodebuku; ?></td>
    </tr>
    <tr>
      <td><font size="3" face="Verdana">Judul</font></td>
      <td><? echo "$judulbuku"; ?></td>
    </tr>
    <tr>
      <td><font size="3" face="Verdana">Penerbit</font></td>
      <td> <? echo "$penerbitbuku"; ?></td>
    </tr>
    <tr>
      <td><font size="3" face="Verdana">Pengarang</font></td>
      <td><? echo "$pengarangbuku"; ?></td>
    </tr>
    <tr>
      <td><font size="3" face="Verdana">Kategori</font></td>
      <td><? echo "$kategoribuku"; ?></td>
    </tr>
    <tr>
      <td><font size="3" face="Verdana">Harga</font></td>
      <td><? echo "$hargabuku"; ?></td>
    </tr>
    <tr>
      <td><font size="3" face="Verdana">ISBN</font></td>
      <td><? echo "$isbn"; ?></td>
    </tr>
    <tr>
      <td><font size="3" face="Verdana">Sinopsis</font></td>
      <td><? echo "$sinopsis"; ?></td>
    </tr>
  </table>
  <p></p>
  <p></p>
</center>
</body>
</html>
```

Outputnya :


http://localhost/contohphp/prosesproduk.php
Search
Toko Buku Amikom
Home|
Produk | Alamat
Katalog Buku
Kode 001
Judul Pemrograman Internet Lanjut
Penerbit Amikom Press
Pengarang rudyanto arief
Kategori
Harga 35000
ISBN 100-1234-123456
Sinopsis Buku ini berisikan trik membuat aplikasi internet dengan PHP

BAGIAN 2 :
DATABASE MySQL

2.1. PENGANTAR MySQL

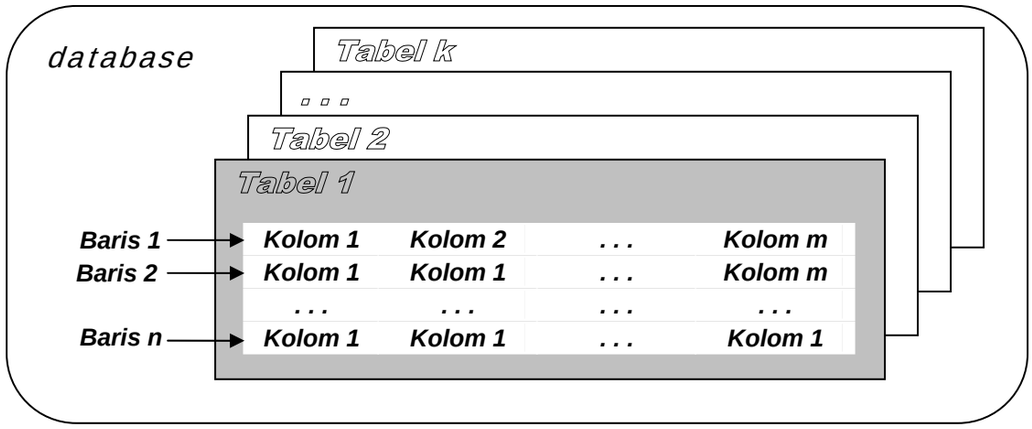
MySQL dikembangkan oleh sebuah perusahaan Swedia bernama *MySQL AB* yang pada saat itu bernama *TcX DataKonsult AB* sekitar tahun 1994-1995, namun cikal bakal kodenya sudah ada sejak 1979. Awalnya TcX membuat MySQL dengan tujuan mengembangkan aplikasi web untuk klien. TcX merupakan perusahaan pengembang software dan konsultan database.

MySQL adalah salah satu jenis database server yang sangat terkenal dan banyak digunakan untuk membangun aplikasi web yang menggunakan database sebagai sumber dan pengelolaan datanya. Kepopuleran MySQL antara lain disebabkan karena MySQL menggunakan SQL sebagai bahasa dasar untuk mengakses databasenya sehingga mudah untuk digunakan, cepat secara kinerja *query*, dan mencukupi untuk kebutuhan database perusahaan-perusahaan skala menengah-kecil. Selain itu mySQL juga bersifat *open source* dan *free* (Anda tidak perlu membayar untuk menggunakannya) pada berbagai *platform* (kecuali pada Windows, yang bersifat *shareware*). MySQL didistribusikan dengan lisensi *open source* GPL (*General Public License*) mulai versi 3.23, pada bulan Juni 2000. Software MySQL bisa di-*download* di *http://www.mysql.org* atau *http://www.mysql.com*.

MySQL merupakan database yang pertama kali didukung oleh bahasa pemrograman *script* untuk internet (PHP dan Perl). MySQL dan PHP dianggap sebagai pasangan software pengembangan aplikasi web yang ideal. MySQL lebih sering digunakan untuk membangun aplikasi berbasis web, umumnya pengembangan aplikasinya menggunakan bahasa pemrograman *script* PHP.

2.1.1. Database, Tabel, Baris dan Kolom

MySQL termasuk RDBMS (*Relational Database Management System*). Itulah sebabnya istilah tabel, baris, dan kolom digunakan pada MySQL. Pada MySQL, sebuah database mengandung satu atau sejumlah tabel. Tabel terdiri atas sejumlah kolom dan baris, dimana setiap kolom berisi sekumpulan data yang memiliki tipe yang sejenis, dan baris merupakan sekumpulan data yang saling berkaitan dan membentuk informasi. Kolom biasanya juga disebut sebagai *field* dan informasi yang tersimpan dalam setiap baris disebut dengan *record*.

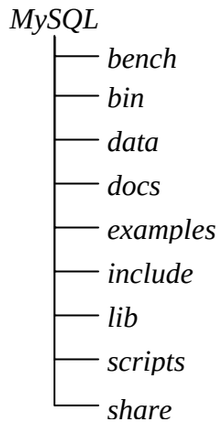


2.2. MENGGUNAKAN MySQL

2.2.1. Struktur Direktori MySQL

Software MySQL secara default akan diletakkan pada direktori *C:\MySQL* jika di-install pada sistem operasi Windows. Direktori ini bisa saja diubah sesuai dengan keinginan pada saat instalasi. Apabila instalasi dilakukan dengan menggunakan software PHPTriad (paket software yang menggabungkan tiga aplikasi: Apache Web Server, PHP, dan MySQL), maka software MySQL terletak di dalam direktori *C:\Apache\MySQL*.

Struktur direktori MySQL:



Direktori yang paling penting dalam struktur direktori MySQL adalah direktori *bin* dan *data*. Sub-direktori *bin* merupakan direktori yang menyimpan semua program database MySQL, sedangkan sub-direktori data digunakan untuk menyimpan data dan file-file yang dibutuhkan oleh MySQL untuk menyimpan database. Setiap database MySQL dibuatkan sebagai sebuah direktori didalam sub-direktori *data* ini.

2.2.2. TIPE DATA PADA MySQL

Berikut ini tabel tipe data yang dapat digunakan untuk field-field tabel pada database MySQL.

TIPE DATA	UKURAN	KETERANGAN
TINYINT	1 byte	Nilai integer yg sangat kecil
SMALLINT	2 bytes	Nilai integer yang kecil
MEDIUMINT	3 bytes	Integer dengan nilai medium
INT	4 bytes	Integer dengan nilai standar
BIGINT	8 bytes	Integer dengan nilai besar
FLOAT	4 bytes	Bilangan desimal dengan single-precission
DOUBLE	8 bytes	Bilangan desimal dengan double-precission
DECIMAL(M,D)	M bytes (D+2, if M < D)	Bilangan float (desimal) yang dinyatakan sebagai string
CHAR(M)	M bytes, 1 <= M <= 255	String karakter dengan panjang yang tetap
VARCHAR(M)	L+1 bytes, L <= M and 1 <= M <= 255	String karakter dengan panjang yang tidak tetap
TINYBLOB	L+1 bytes, L < 2^8	BLOB (Binary Large Object) yang sangat kecil
BLOB	L+1 bytes, L < 2^16	BLOB berukuran kecil
MEDIUMBLOB	L+1 bytes, L < 2^24	BLOB berukuran sedang
LONGBLOB	L+1 bytes, L < 2^32	BLOB berukuran besar
TINYTEXT	L+1 bytes, L < 2^8	String teks yang sangat kecil
TEXT	L+1 bytes, L < 2^16	String teks berukuran kecil
MEDIUMTEXT	L+1 bytes, L < 2^24	String teks berukuran medium
LONGTEXT	L+1 bytes, L < 2^32	String teks berukuran besar
ENUM('v1','v2',...)	1 or 2 bytes, (65535 values max)	Enumerasi, kolom dapat diisi dengan 1 member enumerasi
SET('val1','val2',...)	1, 2, 3, 4 or 8 bytes, (64 max)	Himpunan, kolom dapat diisi dengan beberapa nilai anggota himpunan
DATE	3 bytes	"1000-01-01" sampai "9999-12-31"
TIME	3 bytes	"-832:59:59" sampai "838-59:59"
DATETIME	8 bytes	"1000-01-01 00:00:00" sampai "9999-12-31 23:59:59"
TIMESTAMP	4 bytes	Range: 19700101000000 (suatu nilai tanggal pada tahun 2037
YEAR	1 byte	1901 sampai 2155
NULL		Nilai kosong (hampa)

2.2.3. Mengaktifkan Database MySQL

Program MySQL berjalan pada mode DOS sehingga untuk menggunakan database MySQL terlebih dahulu masuk ke mode DOS. Kita mengasumsikan bahwa program MySQL berada pada direktori *C:\Apache\MySQL*, dan untuk mengaksesnya kita masuk ke direktori MySQL dengan mengetikkan `cd\apache\mysql\bin` pada DOS *prompt*. Setelah berada pada direktori *C:\Apache\MySQL\bin*, langkah selanjutnya adalah mengaktifkan server MySQL (*MySQL-daemon*). Jika server MySQL telah aktif, diteruskan dengan memanggil program MySQL yang berjalan di sisi *client*. Untuk lebih jelasnya ikuti langkah-langkah berikut :

1. Masuk ke mode DOS
2. `cd:\apache\mysql\bin` (enter)
3. `mysqld` (enter)
4. `mysql` (enter)

Apabila langkah-langkah di atas dilakukan dengan benar, kita akan dibawa kedalam lingkungan MySQL client, yang ditandai dengan munculnya tampilan seperti berikut :

```
C:\WINDOWS>cd\apache\mysql\bin

C:\Apache\MySQL\bin>mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2 to server version: 3.23.42

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Pada layar monitor akan ditampilkan versi MySQL yang digunakan (pada contoh ini versi yang digunakan adalah versi 3.23.42), dan adanya *prompt* `mysql>` menunjukkan bahwa program MySQL telah siap untuk menerima masukan berupa perintah-perintah SQL. Kita dapat mengetikkan perintah-perintah SQL pada *prompt* tersebut. Dalam penulisan perintah SQL, satu baris perintah selalu diakhiri dengan tanda titik koma (;). Sedangkan untuk keluar dari lingkungan MySQL dapat dilakukan dengan mengetikkan salah satu perintah : *exit, quit*, atau *\q*.

2.2.4. Operasi-operasi MySQL

Membuat database

Sebelum melakukan proses pembuatan tabel dan manipulasi data, database harus diciptakan terlebih dahulu. Dalam pembuatan database pemberian nama database harus unik. Ini berarti tidak boleh menggunakan nama database yang sudah ada. Untuk mengetahui daftar database yang terdapat pada MySQL, dapat digunakan perintah berikut :

```
SHOW DATABASES;
```

Sedangkan perintah untuk membuat database adalah sebagai berikut :

```
CREATE DATABASE nama_database;
```

Agar dapat melakukan operasi-operasi terhadap suatu database, harus dilakukan koneksi ke database tersebut terlebih dahulu. Koneksi ke database dapat dilakukan dengan menggunakan perintah *use*, sebagai berikut :

```
USE nama_database;
```

Jika koneksi ke database berhasil akan muncul tanggapan dari MySQL, sebagai berikut :

```
Database changed
```

Untuk menghapus database dapat dilakukan dengan perintah :

```
DROP nama_database;
```

Membuat tabel

Suatu database ditempati oleh tabel-tabel yang merupakan tempat untuk menyimpan data. Untuk membuat tabel di dalam database MySQL dapat dilakukan dengan mengetikkan perintah SQL berikut :

```
CREATE TABLE nama_tabel(

nama_field1 tipe_data1[(ukuran/nilai) atribut],
nama_field2 tipe_data2[(ukuran/nilai) atribut],
...);
```

Mengubah struktur tabel

Menambah field :

```
ALTER TABLE 'nama_tabel' ADD 'field' tipe_data[(ukuran/nilai)
atribut];
ALTER TABLE 'anama_tabel' ADD 'field' tipe_data FIRST;
ALTER TABLE 'nama_tabel' ADD 'field' tipe_data AFTER field_posisi;
```

Merubah nama field :

```
ALTER TABLE 'nama_tabel' CHANGE 'field_lama' 'field_baru'
tipe_data[(ukuran/nilai) atribut];
```

Menghapus field :

```
ALTER TABLE 'nama_tabel' DROP `field`;
```

Menentukan primary key :

```
ALTER TABLE 'nama_tabel' ADD PRIMARY KEY (field_kunci);
```

Merubah primary key :

```
ALTER TABLE 'nama_tabel' DROP PRIMARY KEY, ADD PRIMARY KEY
(field_kunci);
```

Menghapus primary key :

```
ALTER TABLE 'nama_tabel' DROP PRIMARY KEY;
```

Memasukkan data

Untuk memasukkan data pada suatu tabel yang telah dibuat, digunakan pernyataan *insert*, dengan sintaks penulisan sebagai berikut :

```
INSERT INTO nama_tabel (field1, field2,...) VALUES
(nilai_field1,nilai_field2,...);

INSERT INTO nama_tabel VALUES (nilai_field1,nilai_field2,...);
```

Menampilkan data

Perintah yang digunakan untuk menampilkan data dari tabel adalah *select*. Sintaks penulisannya sebagai berikut :

```
SELECT field1,field2,... FROM nama_tabel;
```

Perintah di atas akan menampilkan semua data yang terdapat pada field yang ditentukan. Untuk melihat semua data yang ada pada suatu tabel, sintak penulisannya adalah :

```
SELECT * FROM nama_tabel;
```

Untuk menyaring hasil seleksi data digunakan kata kunci *where* sehingga record yang ditampilkan hanyalah record yang sesuai dengan kriteria yang diinginkan. Sintaks penulisannya sebagai berikut :

```
SELECT field1,field2,... FROM nama_tabel WHERE kriteria;
```

Mengurutkan data

Hasil query dapat diurutkan sesuai kebutuhan dengan menggunakan kata kunci *order by*. Sintak penulisannya sebagai berikut :

```
SELECT field1,field2,... FROM nama_tabel ORDER BY kriteria;
```

Untuk mengurutkan data dengan urutan terbalik (besar ke kecil) dapat digunakan kunci tambahan *desc*, dengan sintaks penulisan sebagai berikut :

```
SELECT field1,field2,... FROM nama_tabel ORDER BY kriteria DESC;
```

Mengubah data

Pengubahan data pada tabel berfungsi untuk memodifikasi nilai kolom (*field)* dari suatu record. Perintah SQL yang digunakan untuk mengubah data adalah sebagai berikut :

```
UPDATE nama_tabel SET nama_field1=nilai_baru1,
nama_field2=nilaibaru2, ... WHERE kriteria;
```

Untuk memodifikasi nilai suatu kolom secara keseluruhan, digunakan perintah *update* tanpa menentukan kriterianya :

```
UPDATE nama_tabel SET nama_field1=nilai_baru1,
nama_field2=nilaibaru2, ...;
```

Menghapus data

Penghapusan data pada tabel berfungsi untuk menghapus suatu *record* dengan kriteria tertentu. Perintah SQL untuk menghapus data pada tabel adalah *delete*, dengan sintaks penulisan sebagai berikut :

```
DELETE FROM nama_tabel WHERE kriteria;
```

Untuk menghapus seluruh *record* pada suatu tabel, digunakan perintah *delete* tanpa menentukan kriterianya :

```
DELETE FROM nama_tabel;
```

BAGIAN 3 :

KONEKSI PHP-MySQL

PHP memiliki fungsi-fungsi yang digunakan untuk mengakses database MySQL. Fungsi-fungsi tersebut berguna untuk melakukan koneksi dan manipulasi database MySQL melaui program PHP.

mysql_connect

Fungsi :	Membuka koneksi ke server database MySQL
Penulisan :	`resource mysql_connect ([string hostname [:port] [:/path/to/socket] [, string username [, string password]]])`
Contoh :	`mysql_connect("localhost","root","");`

mysql_close

Fungsi :	Memutuskan koneksi dengan server database MySQL
Penulisan :	`Bool mysql_close ([resource link_identifier])`
Contoh :	`$koneksi=mysql_connect("localhost","root","");` `mysql_close($koneksi);`

mysql_select_db

Fungsi :	Memilih database MySQL
Penulisan :	`bool mysql_select_db (string database_name [, resource link_identifier])`
Contoh :	

```
<?php
$link = mysql_connect("localhost","root","")
  or die("Gagal Koneksi ke Server Database!");
mysql_select_db("test",$link)
  or die("Gagal Memilih Database!");
?>
```

mysql_query

Fungsi : Mengeksekusi perintah query

Penulisan : `resource mysql_query (string query [, resource link_identifier])`

mysql_num_rows

Fungsi : Menampilkan jumlah baris yang dihasilkan oleh perintah *mysql_query* yang menggunakan SELECT.

Penulisan : `int mysql_num_rows (resource result)`

Contoh :

```
<?php
$link = mysql_connect("localhost", "username", "password");
mysql_select_db("database", $link);
$result = mysql_query("SELECT * FROM table1", $link);
$num_rows = mysql_num_rows($result);
print("$num_rows Rows");
?>
```

mysql_num_fields

Fungsi : Menampilkan jumlah field yang dihasilkan oleh perintah *mysql_query* yang menggunakan SELECT

Penulisan : `int mysql_num_fields (resource result)`

mysql_affected_rows

Fungsi : Menampilkan jumlah baris yang terpengaruh oleh operasi SQL sebelumnya (INSERT, DELETE, dan UPDATE).

Penulisan : `int mysql_num_rows (resource result)`

resource_result menyatakan variabel yang digunakan pada pemanggilan fungsi **mysql_connect**.

mysql_fetch_row

Fungsi : Menghasilkan *array* yang berisi seluruh kolom dari sebuah baris pada suatu himpunan hasil. Fungsi ini bersifat membaca baris berikutnya dalam suatu himpunan hasil.

Penulisan : `array mysql_fetch_row (resource result)`

mysql_fetch_array

Fungsi : Sama seperti *mysql_fetch_row*. Hanya saja setiap kolom akan disimpan dua kali pada *array* hasil. Yang pertama memiliki indeks angka yang dimulai dari nol, dan yang kedua memiliki indeks berdasarkan nama *field*.

Penulisan : `array mysql_fetch_array (resource result [, int result_type]`

mysql_fetch_field

Fungsi : Memperoleh informasi suatu kolom.

Penulisan : `object mysql_fetch_field (resource result [, int field_offset])`

field_offset menyatakan *nomor_kolom* menyatakan nomor kolom yang ingin didapatkan. Penomoran ini dimulai dari nol dan sifatnya opsional. Jika argumen ini tidak disebutkan, maka kolom berikutnya yang akan didapatkan.

Contoh :

```
<?php
$link = mysql_connect("localhost","root","");
mysql_select_db("workshop ",$link);

$result = mysql_query ("select * from staff");
$i = 0;
while ($i < mysql_num_fields ($result))
{
  echo "Information for column $i:<BR>\n";
  $meta = mysql_fetch_field ($result);
  if (!$meta)
  {
    print("No information available<BR>\n");
  }
  print("<PRE>
    blob:         $meta->blob
    max_length:   $meta->max_length
    multiple_key: $meta->multiple_key
    name:         $meta->name
    not_null:     $meta->not_null
    numeric:      $meta->numeric
    primary_key:  $meta->primary_key
    table:        $meta->table
    type:         $meta->type
    unique_key:   $meta->unique_key
    unsigned:     $meta->unsigned
    zerofill:     $meta->zerofill
    </PRE>");
  $i++;
}
mysql_free_result ($result);
?>
```

mysql_data_seek

Fungsi : Memindahkan pointer pada suatu himpunan hasil supaya menunjuk ke baris tertentu.

Penulisan :
```
bool mysql_data_seek (resource result_identifier, int row_number)
```

mysql_create_db

Fungsi : Menciptakan database baru.

Penulisan :
```
int mysql_create_db (string database name [, resource link_identifier])
```

mysql_drop_db

Fungsi : Menghapus database

Penulisan :
```
Bool mysql_drop_db (string database_name [, resource link_identifier])
```

mysql_list_dbs

Fungsi : Menghasilkan daftar database yang ada di server

Penulisan :
```
resource mysql_list_dbs ([resource link_identifier])
```

Contoh :

```
<?php
$link = mysql_connect("localhost","root","");
$db_list = mysql_list_dbs($link);
while ($row = mysql_fetch_object($db_list))
{
  print($row->Database . "<br>\n");
}
?>
```

mysql_list_tables

Fungsi : Memperoleh daftar nama tabel dalam suatu database MySQL

Penulisan : `resource **mysql_list_tables** (string database [, resource link_identifier])`

Contoh :

```
<?php
$koneksi = mysql_connect("localhost","root","");
$DB = "database";
mysql_select_db($DB);
$tables=mysql_list_tables($DB);
while (list($bla)=mysql_fetch_array($tables))
{
  print($bla." are the tablenames<br>");
}
?>
```

mysql_list_fields

Fungsi : Memperoleh daftar nama kolom.

Penulisan : `resource **mysql_list_fields** (string database_name, string table_name [, resource link_identifier])`

Contoh :

```
<?php
$koneksi = mysql_connect("localhost","root","");
mysql_select_db("workshop",$koneksi);
$fields=mysql_list_fields("workshop","staff",$koneksi);
$columns = mysql_num_fields($fields);
for ($i = 0; $i < $columns; $i++)
{ echo mysql_field_name($fields, $i) . "<br>\n"; }
?>
```