SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER

STMIK AMIKOM YOGYAKARTA

MODUL
PEMROGRAMAN WEB LANJUT

JURUSAN S1 TEKNIK INFORMATIKA
STMIK AMIKOM YOGYAKARTA TAHUN AJARAN 2015/2016

DAFTAR ISI

CONTROL FLOW

Control Flow dalam bahasa Indonesia dapat diartikan sebagai aliran kendali. Yaitu bagaimana urutan eksekusi perintah dalam program. Beberapa contoh control flow dalam PHP adalah:

- Percabangan (Branching)
- Perulangan (Looping)
- Perpindahan (Jumping)

1. Percabangan

Sering juga disebut dengan istilah *dicision-making*.Yang memungkinkan aplikasi untuk memeriksa isi sebuah variabel atau hasil perhitungan dan ekspresi kemudian mengambil tindakan yang sesuai.

Sebagai contoh percabangan dapat ditemukan pada aplikasi login. Dimana hanya user dengan *username* dan *password* yang valid lah yang dapat memasuki sebuah sistem dan menggunakan fasilitas yang memang ditujukan hanya untuk user yang sah.

1.1 Pernyataan `if`

Dalam kondisi sehari hari terdapat beberapa contoh kondisi yang memerlukan tindakan. Diataranya :

- Ketika datang hujan, bawa payung
- Ketika mobil kotor, cuci bersih
- Ketika bensin habis, isi lagi.

Dari contoh diatas, didapatkan sebuah pola umum. Ketika diberikan sebuah kondisi, sebuah aksi akan dilakukan *jika* kondisi yang menjadi prasyarat bernilai benar. Secara umum, bentuk pernyataan *if* dapat dituliskan sebagai berikut:

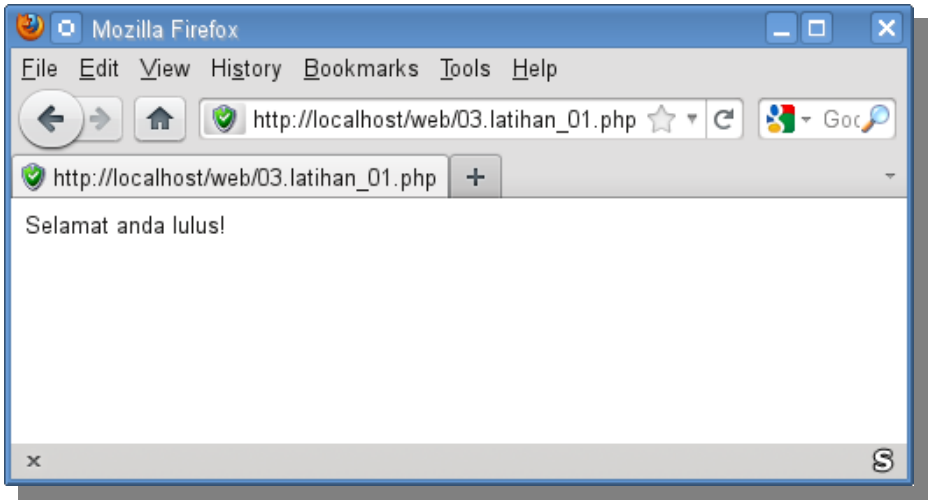
```
if(condition){
   statements
   ...
} else {
   statements
   ...
}
```

Keterangan :

Condition adalah sebuah mekanisme evaluasi terhadap sebuah pernyataan atau ekspresi, yang menentukan apakah ia bernilai benar atau salah. *Statements* mungkin terdiri dari satu atau lebih perintah yang akan dijalankan apabila *condition* bernilai benar. Keyword tambahan `else` memungkinkan kita untuk menjalankan perintah lain jika *condition* bernilai salah.

03.latihan_01.php

```php
<?php

$nilai = 80;

if($nilai >= 60){
   echo "Selamat anda lulus!";
} else {
   echo "Coba lagi semester depan.";
}

?>
```



Program diatas dapat dijelaskan sebagai berikut:

1. Mula-mula variabel `$nilai` diberikan value `80`.
2. Kemudian `if` akan memeriksa kondisi variabel `$nilai`, apakah ia lebih besar atau sama dengan `60` melalui operator perbandingan “>=” (lihat kembali bagian *Operator Relasional* pada modul 2).
3. Jika kondisi bernilai benar, maka program akan menampilkan `"Selamat anda lulus!"`.
4. Jika kondisi bernalai salah, program akan menampilkan `"Coba lagi semester depan."`.

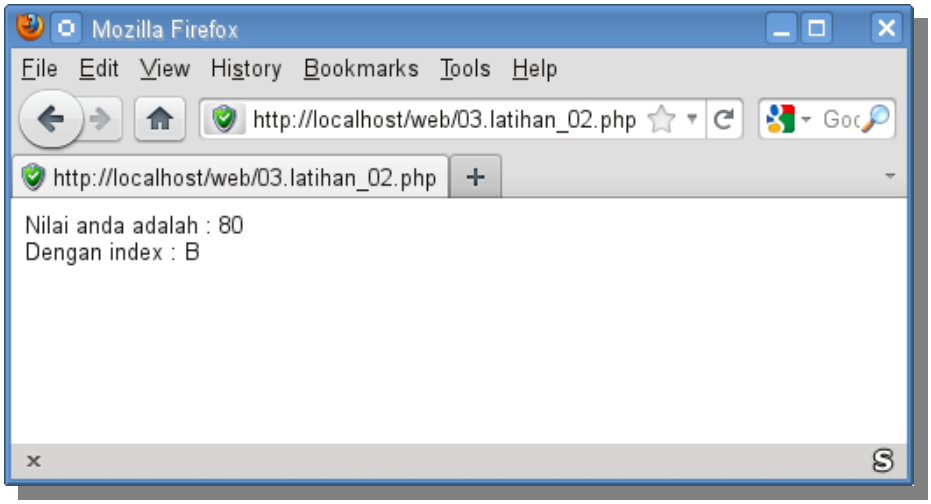
1.2 Pernyataan `else if` dan `else`

Pada beberapa kasus, tidak setiap kondisi dapat ditangani oleh pernyataan kondisi `if-else`. Oleh sebab itu, pernyataan `else if` digunakan untuk merumuskan lebih banyak alternatif.

Latihan 03.latihan_02.php memperlihatkan kepada kita beberapa kemungkinan value variabel `$index` yang akan didapatkan dari hasil perbandingan variabel `$nilai`.

03.latihan_02.php

```php
<?php

$nilai = 80;

if(($nilai >= 85) && ($nilai <= 100)){
   $index = "A";
} else if($nilai >= 70) {
   $index = "B";
} else if($nilai >= 50) {
   $index = "C";
} else if($nilai >= 30){
   $index = "D";
} else {
   $index = "E";
}

echo "Nilai anda adalah : ".$nilai;
echo "<br />";
echo "Dengan index      : ".$index;

?>
```



1.3 Pernyataan `switch`

Pernyataan `switch` adalah alternatif dari bentuk percabangan `if-else if-else`. Seperti yang kita pelajari sebelumnya, bahwa `if-else if-else` sangat ideal untuk membuat keputusan berdasarkan sejumlah kondisi.

Tidak seperti `if`, pernyataan `switch` digunakan *hanya* untuk membandingkan variabel tunggal dengan dengan beberapa kemungkinan nilai-nilai. Bentuk umum dari pernyataan `switch` adalah sebagai berikut:

```
switch($variable){
   case value_1:
      statement 1;
      break;
   case value_2:
      statement 2;
      break;
   case value_3:
      statement 3;
      break;
   default:
      statement 4;
      break;
}
```

Keterangan aliran program:

1. `$variable` akan dibandingkan dengan daftar kemungkinan.
2. Pernyataan dieksekusi
   a. Jika `$variable` == value_1, maka *statement 1* dijalankan kemudian keluar dari dari blok program `switch`.
   b. Jika `$variable` == value_2, maka *statement 2* dijalankan kemudian keluar dari dari blok program `switch`.
   c. Jika `$variable` == value_3, maka *statement 3* dijalankan kemudian keluar dari dari blok program `switch`.
   d. Jika `$variable` tidak memenuhi value_1 - value_3, maka *statement 4* dijalankan.

Keyword `break` menjadi penting dalam setiap bentuk pernyataan `switch`. Fungsinya digunakan sebagai pencegah *fall-through*. Ini karena dalam pernyataan `switch`, **`case`** hanya berfungsi sebagai label. Ketika *statement 1* pada value_1 selesai dijalankan dan tidak diakhiri dengan `break`, ia akan melanjutkan pembandingan kepada value_2 dan seterusnya sampai ditemukannya `break`. Untuk lebih jelasnya perhatikan listing program berikut:

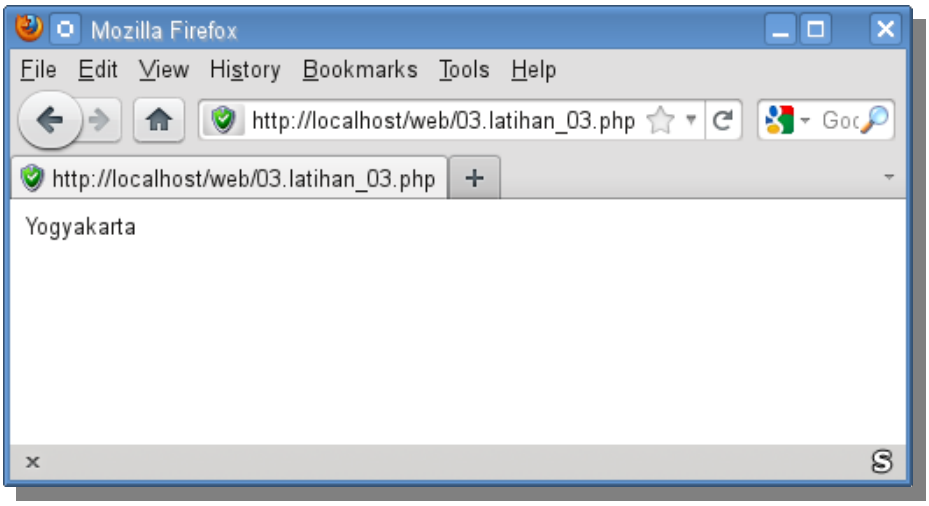
```
switch($variable){
   case value_1:
      statement 1;
   case value_2:
      statement 2;
   case value_3:
      statement 3;
      break;
   default:
      statement 4;
      break;
}
```

Keterangan aliran program:

1. `$variable` akan dibandingkan dengan daftar kemungkinan.
2. Pernyataan dieksekusi
   a. Jika `$variable` == value_1, maka *statement 1* dijalankan.
   b. Jika `$variable` == value_2, maka *statement 2* dijalankan.
   c. Jika `$variable` == value_3, maka *statement 3* dijalankan.
   d. Jika `$variable` tidak memenuhi value_1 - value_3 maka *statement 4* dijalankan.

03.latihan_03.php

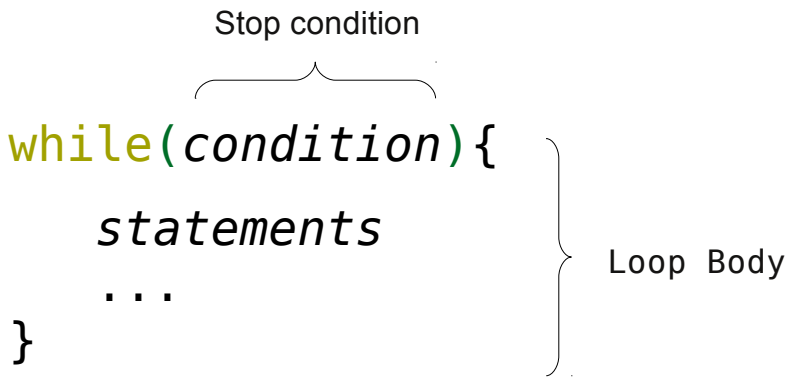
```
<?php

$plat_nomor = "AB";

switch($plat_nomor){
   case "AB":
      echo "Yogyakarta";
      break;
   case "AD":
      echo "Surakarta";
      break;
   case "BE":
      echo "Lampung";
      break;
   case "B":
      echo "Jakarta";
      break;
   default:
      echo "Plat kendaraan tidak diketahui.";
      break;
}

?>
```



2. Perulangan

2.1 Perulangan `while`

Perulangan `while` dikenal juga dengan *indeterminate loop*. Artinya, penentuan jumlah perulangan tidak ditentukan sebelumnya. Perulangan akan dilakukan terus menerus sampai dengan kondisi yang menjadi prasyarat bernilai *false*.



Terdapat beberapa bagian penting dalam perulangan `while`, yaitu:

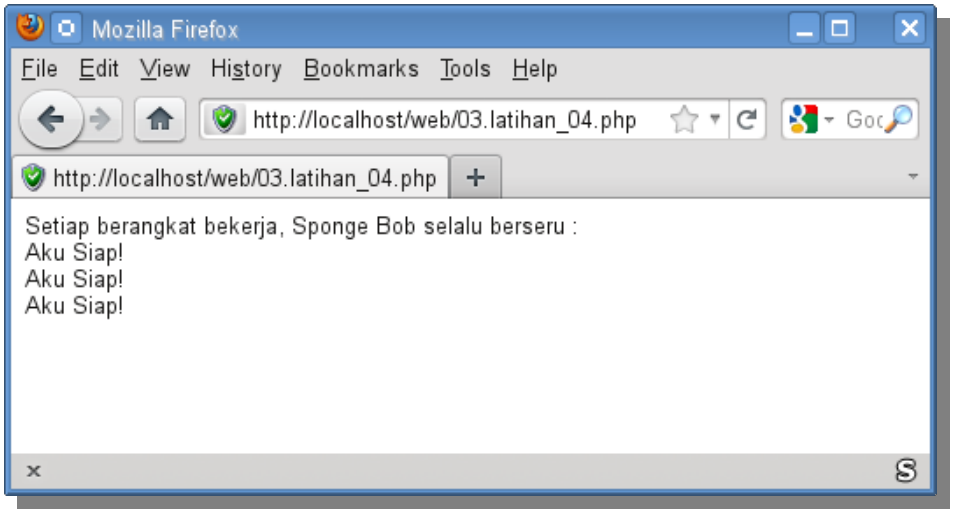
- Stop condition : kondisi yang menjadi syarat terjadinya satu siklus perulangan.
- Loop body : merupakan perintah yang akan dilakukan dalam sebuah perulangan.

Prinsip kerja :

1. Pernyataan while menguji kondisi yang menjadi prasyarat.
2. Jika kondisi bernilai *true*, maka statements didalam *loop body* akan dikerjakan sekali lagi. Jika kondisi bernilai *false*, maka perulangan akan dihentikan.
3. Setiap kali statements pada loop body selesai dikerjakan, kondisi prasyarat akan kembali diperiksa.

03.latihan_04.php

```php
<?php

echo "Setiap berangkat bekerja, Sponge Bob selalu berseru : <br />";

$i = 1;

while($i <= 3){
   echo "Aku Siap!<br />";
   $i++;
}

?>
```

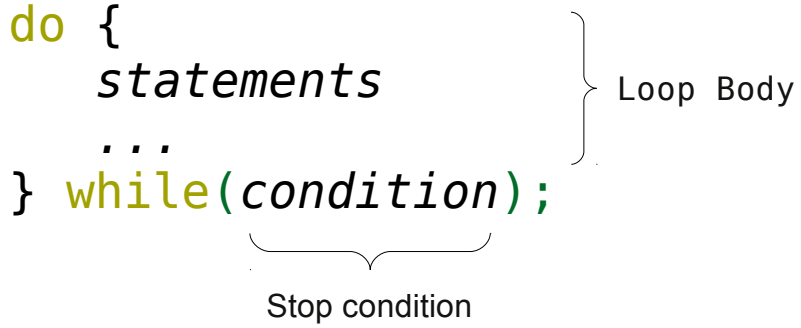


Keterangan aliran program:

1. Program akan mengisialisasi nilai *counter* (penghitung). Pada program diatas ditunjukkan dengan pernyataan `$i = 1;`.
2. Memeriksa kondisi perulangan. Pada program diatas ditunjukkan dengan pernyataan `$i <= 3`. Yaitu program akan memeriksa apakah nilai variabel `$i` masih lebih kecil atau sama dengan `3`.
3. Jika pernyataan bernilai benar, program akan menampilkan `"Aku Siap!<br />"` pada browser.
4. Operasi menaikkan nilai *counter*. Pada program diatas ditunjukkan dengan pernyataan `$i++;` yang ditujukan untuk merubah nilai `$i` setiap kali looping terjadi, sehingga suatu saat nanti perulangan bisa dihentikan.
5. Program kembali kepada langkah no. 2. Yaitu untuk memeriksa kondisi pernyataan perulangan.

2.2 Perulangan `do-while`

Perulangan `do-while` merupakan modifikasi dari perulangan `while`. Perulangan `do-while` memiliki prinsip kerja yang sama dengan perulangan `while`. Hanya saja pemeriksaan kondisi prasyaratnya dilakukan pada akhir perulangan.

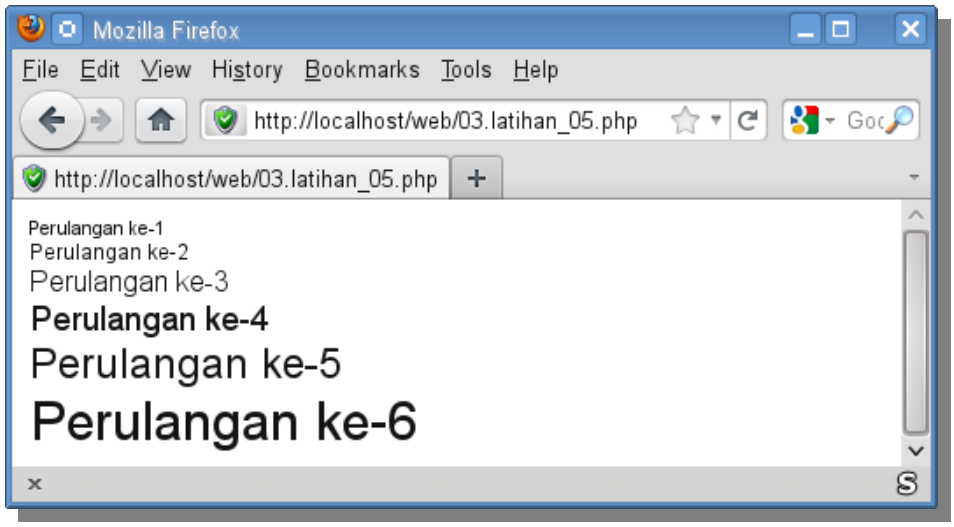


Prinsip kerja :

1. Mula-mula statements akan dikerjakan tanpa melakukan pengujian terhadap kondisi yang menjadi prasyarat.
2. Setelah statements dijalankan, program akan memeriksa kondisi prasyarat perulangan.
3. Jika kondisi bernilai *true*, maka statements didalam *loop body* akan dikerjakan sekali lagi. Jika kondisi bernilai *false*, maka perulangan akan dihentikan.

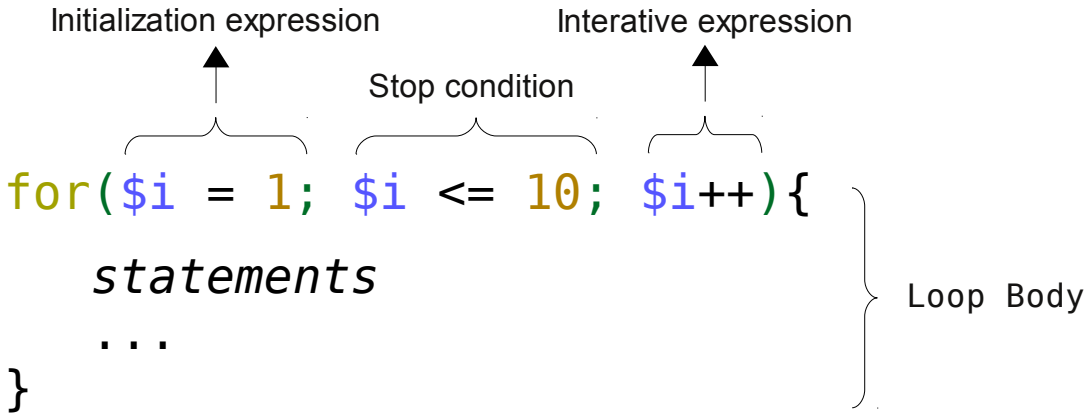
03.latihan_05.php

```
<?php

$i = 1;

do {
   echo "<font size=".$i.">Perulangan ke-".$i."</font>";
   $i++;
} while($i <= 6);

?>
```



2.3 Perulangan `for`

Perulangan `for` disebut juga *determinate loop*. Yaitu jumlah perulangannya (iterasi) telah ditentukan sejak awal pernyataan. Misalnya: kita akan mencetak bilangan 1-10. Dalam hal ini, kita sudah mengetahui bahwasanya perulangan akan dimulai dari bilangan 1 dan terus menerus mencetak selama sebuah bilangan itu masih lebih kecil atau sama dengan 10. Bentuk umum perulangan menggunakan `for` adalah sebagai berikut:

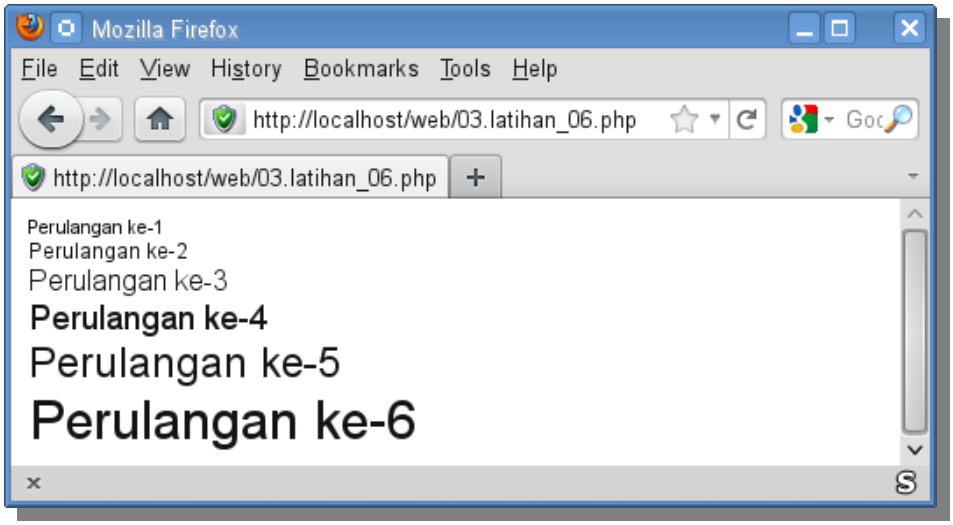


Terdapat beberapa bagian penting dalam perulangan `for`, yaitu:

- Initialization expression : digunakan untuk menentukan nilai awal *counter* (penghitung).
- Stop condition : pemeriksaan nilai *counter*. Jika kondisi ekspresi bernilai *false*, perulangan dihentikan.
- Interative expression : perintah yang dijalankan setiap selesai satu siklus perulangan. Biasanya bagian ini digunakan untuk menambah nilai *counter*.
- Loop body : merupakan perintah yang akan dilakukan dalam sebuah perulangan.

03.latihan_06.php

```php
<?php

for($i=1;$i<=6;$i++){
   echo "<font size=".$i.">Perulangan ke-".$i."</font><br />";
}

?>
```



Keterangan aliran program:

1. Program akan mengisialisasi nilai *counter*. Pada program diatas ditunjukkan dengan pernyataan `$i = 1;`.
2. Program memeriksa kondisi prasyarat perulangan. Pada program diatas ditunjukkan dengan pernyataan `$i <= 6`. Yaitu program akan memeriksa apakah nilai variabel `$i` masih lebih kecil atau sama dengan `6`.
3. Program akan menampilkan `"Perulangan ke-".$i."` pada browser.
4. Operasi menaikkan nilai *counter*. Pada program diatas ditunjukkan dengan pernyataan `$i++;` yang ditujukan untuk merubah nilai `$i` setiap kali looping terjadi, sehingga suatu saat nanti perulangan bisa dihentikan.
5. Program kembali kepada langkah no. 2. Yaitu untuk memeriksa kondisi pernyataan perulangan.

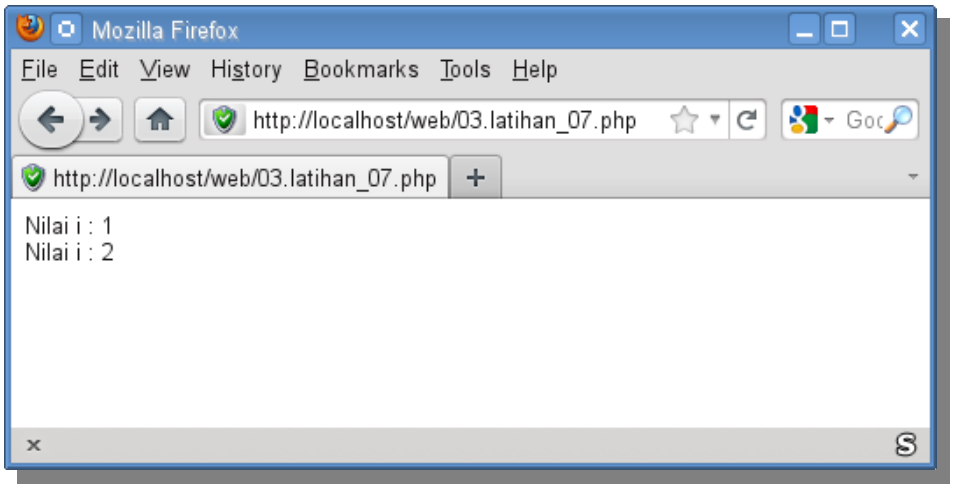
3. Perpindahan

3.1 break

Keyword `break` adalah bagian dari syntax bahasa pemrograman, fungsinya untuk keluar dari sebuah perulangan.

03.latihan_07.php

```
<?php

for($i=1;$i<=6;$i++){
   if($i == 2){
      break;
   }
   echo "Nilai i : ".$i."<br />";
}

?>
```



Keterangan aliran program:

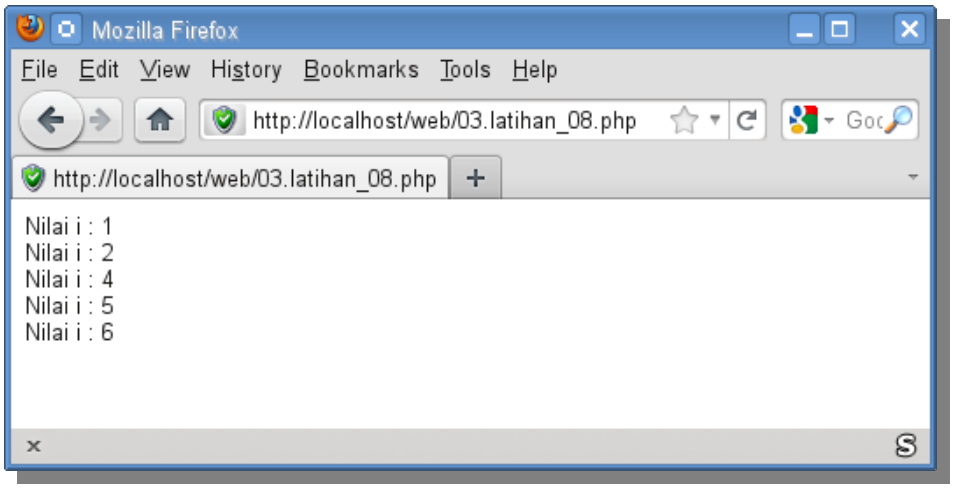
1. Program akan melakukan perulangan dari 1-6. Ini dapat dilihat dari inisialiasi *counter* `$i = 1;` dan pemeriksaan kondisi pada pernyataan `$i <= 6;`.
2. Dalam *loop body* diatas, terdapat pernyataan `if` yang akan memeriksa kondisi variabel `$i`. Dimana jika variabel `$i == 2`, maka perulangan harus diakhiri (`break`).

3.2 continue

Dalam bahasa pemrograman `continue` PHP, digunakan untuk melewati satu siklus perulangan / iterasi.

03.latihan_08.php

```php
<?php

for($i=1;$i<=6;$i++){
   if($i == 3){
      continue;
   }
   echo "Nilai i : ".$i."<br />";
}

?>
```



Dari *running* program diatas dapat kita lihat bahwa perulangan akan dilakukan dari 1-6. Namun, dalam *loop body* terdapat pernyataan `if`, dimana ketika nilai `$i == 3`, maka siklus perulangan pada saat itu diabaikan melalui statements `continue;`. Kemudian program menaikkan counter (`$i++`) untuk kemudian kembali melanjutkan (melompat) siklus perulangan berikutnya.

3.3 `return`

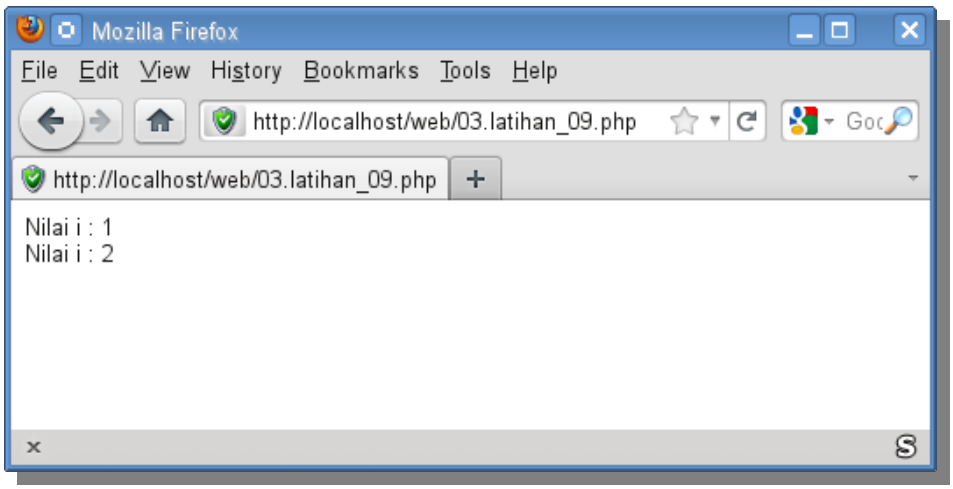
Keyword `return` berguna untuk memerintahkan program keluar dari sebuah fungsi. Keterangan dan penggunaan `return` secafa lebih lanjut akan kita bahas pada bab fungsi.

3.4 `exit`

Keyword `exit` adalah fungsi yang digunakan untuk keluar dari sebuah program. Walaupun terdapat statements-statements lain dibawah baris kode `exit` yang belum dikerjakan.

03.latihan_09.php

```php
<?php

for($i=1;$i<=6;$i++){
   if($i == 3){
      exit();
   }
   echo "Nilai i : ".$i."<br />";
}

// statement dibawah ini tidak akan dijalankan
echo "STMIK AMIKOM Yogyakarta";
echo "Tempat kuliah orang berdasi";

?>
```



Dari *running* program diatas dapat kita lihat bahwa eksekusi program akan berhenti bilamana dalam perulangan nilai variabel `$i == 3`. Hal ini juga mengakibatkan kode program dibawahnya yaitu :

```php
echo "STMIK AMIKOM Yogyakarta";
echo "Tempat kuliah orang berdasi";
```

juga tidak akan diproses lebih lanjut oleh PHP interpreter.

ARRAY

Array adalah kumpulan beberapa data yang disimpan dalam sebuah variabel. Masing-masing data yang disimpan dalam array tersebut mempunyai *index* sebagai sebuah pengenal. Setiap anggota dari array disebut sebagai *elemen*. Berbeda dengan tipe data sebelumnya yang hanya dapat menampung sebuah nilai, array dapat menampung lebih dari satu nilai.

Sebagai contoh, kita ingin menyimpan data teman-teman kita dalam sebuah variabel `$teman`. Ada tiga orang yang akan kita masukkan dalam variabel tersebut. Secara visual, variabel `$teman` dapat kita gambarkan sebagai berikut:

Yudistira	Bima	Arjuna

`$teman`

Setiap nilai dalam elemen variabel `$teman` diatas dapat diakses / dipanggil menggunakan index. Di dalam penggunaannya, PHP terdapat dua macam index yaitu:

1. Index Numerik. Yaitu menggunakan angka untuk menandai sebuah elemen array.
2. Index Asosiatif. Yaitu memberikan nama sebagai penanda sebuah elemen. Penggunaan index secara detail akan dibahas dibawah.

1. Inisialiasi Array

1.1 Index Numerik

Inisialiasi (pengisian) sebuah array dengan index numerik dapat dilakukan dengan cara-cara sebagai berikut :

```
$teman = array("Yudistira", "Bima", "Arjuna");
```

Hal yang sama juga dapat dilakukan dengan cara sebagai berikut :

```
$teman[] = "Yudistira";
$teman[] = "Bima";
$teman[] = "Arjuna";
```

Berbeda dengan inisialiasi dengan cara diatas dimana index didefinisikan secara otomatis oleh program. Cara inisialiasi dibawah ini memungkinkan kita untuk mendefinisikan index secara manual.

```
$teman[0] = "Yudistira";
$teman[1] = "Bima";
$teman[2] = "Arjuna";
```

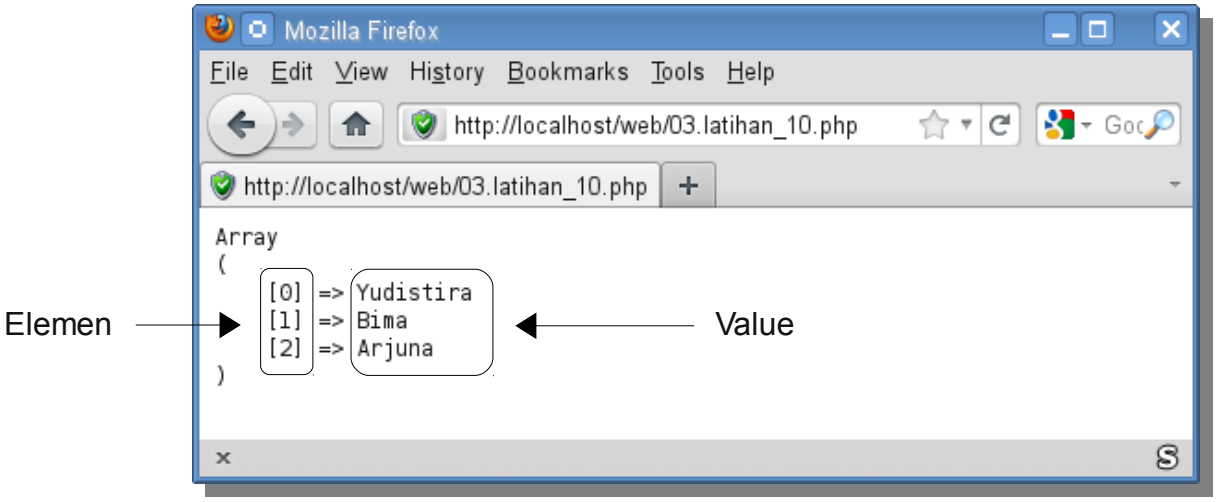
Hasil dari inisialiasi array dengan index numeric diatas, maka array `$teman` dapat digambarkan sebagai berikut:

Value	Yudistira	Bima	Arjuna
Index	0	1	2

Dalam PHP, untuk menampilkan semua elemen dan *value* yang telah didefinisikan dalam sebuah array, digunakan fungsi `print_r()`. Untuk lebih jelasnya, lihat contoh dibawah ini.

03.latihan_10.php

```
<?php

$teman[] = "Yudistira";
$teman[] = "Bima";
$teman[] = "Arjuna";

echo "<pre>";
print_r($teman);
echo "</pre>";

?>
```



1.2 Index Asosiatif

Untuk melakukan inisialisasi dengan menggunakan index asosiatif membutuhkan sebuah label sebagai pengenal sebuah elemen. Inisialiasi (pengisian) sebuah array dengan index asosiatif dapat dilakukan dengan cara-cara sebagai berikut :

Yudistira	Bima	Arjuna
wayang_1	wayang_2	wayang_3

Didalam PHP inisialiasi array dengan index asosiatif dapat dilakukan dengan cara-cara sebagai berikut :

```
$teman = array("wayang_1" => "Yudistira",
               "wayang_2" => "Bima",
               "wayang_3" => "Arjuna");
```

Inisialiasi dengan cara diatas juga dapat dilakukan dengan cara sebagai berikut :

```
$teman['wayang_1'] = "Yudistira";
$teman['wayang_2'] = "Bima";
$teman['wayang_3'] = "Arjuna";
```

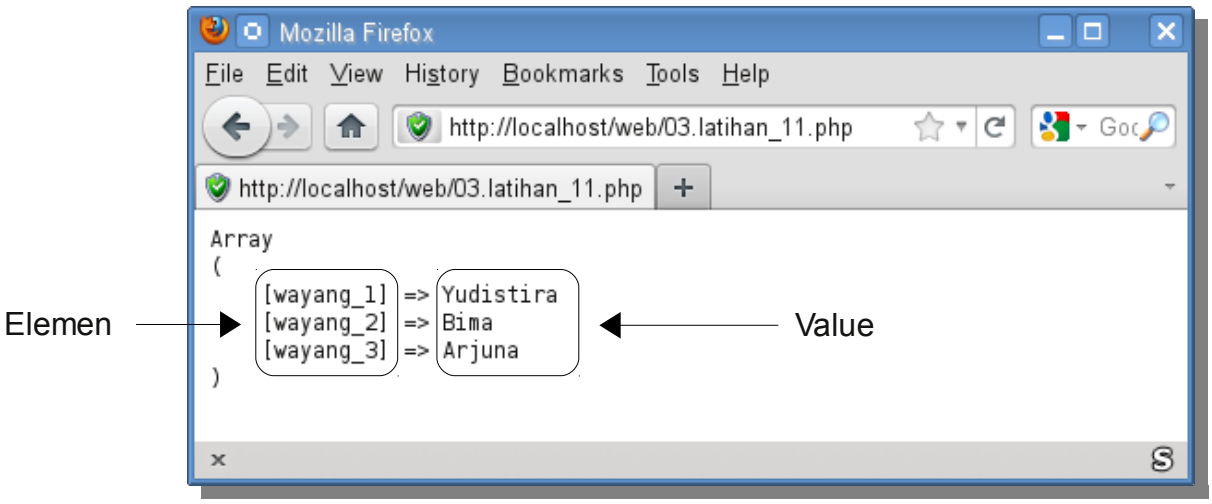
Hasil dari inisialiasi array dengan index asosiatif diatas dapat digambarkan sbb:

Value	Yudistira	Bima	Arjuna
Index	wayang_1	wayang_2	wayang_3

Sama halnya dengan baris kode pada array dengan index numeric, fungsi `print_r()` juga digunakan untuk menampilkan semua elemen dan *value* yang telah didefinisikan dalam array dengan index asosiatif.

03.latihan_11.php

```
<?php

$teman['wayang_1'] = "Yudistira";
$teman['wayang_2'] = "Bima";
$teman['wayang_3'] = "Arjuna";

echo "<pre>";
print_r($teman);
echo "</pre>";

?>
```



2. Navigasi (Mengakses Nilai) Array

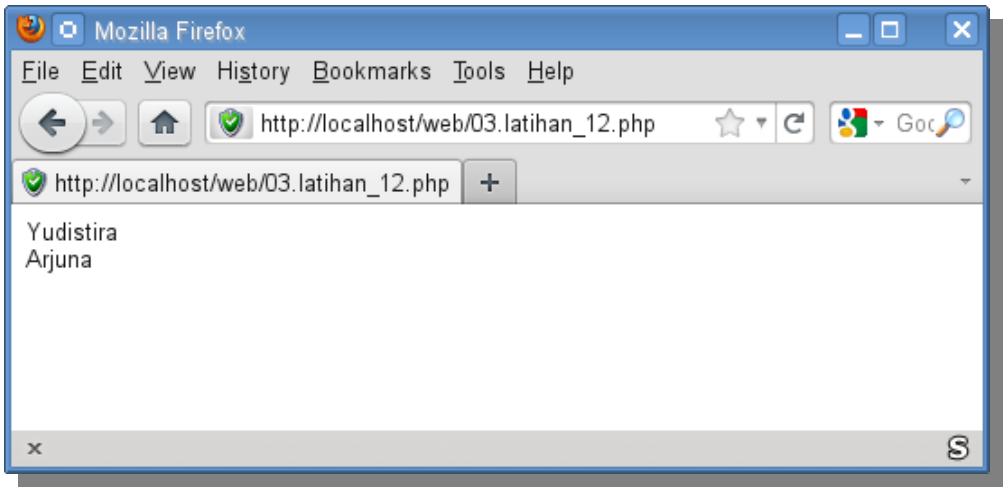
2.1 Index Numerik

Terdapat dua cara yang dapat dilakukan untuk melakukan navigasi (pengaksesan nilai) array dengan index numeric. Yaitu:

a. Pengaksesan Langsung. Pengaksesan array secara langsung dapat dilakukan dengan cara menuliskan nama array berserta indexnya.

03.latihan_12.php

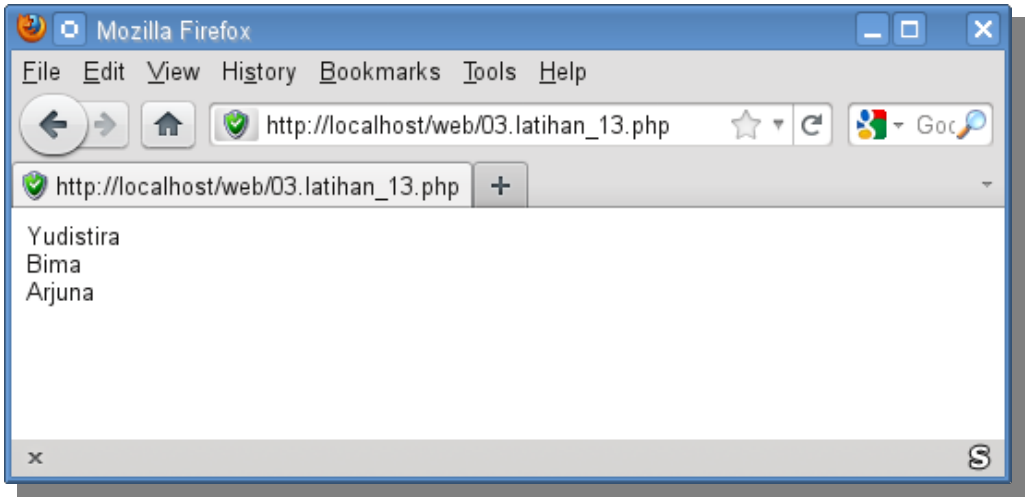
```php
<?php

$teman[] = "Yudistira";
$teman[] = "Bima";
$teman[] = "Arjuna";

echo $teman[0];
echo "<br />";
echo $teman[2];

?>
```



b. Melalui Perulangan. Seperti dijelaskan diatas, pengaksesan array dengan index numeric dapat dilakukan dengan menuliskan nama array diikuti oleh index elemen. Maka pada pemanggilan array melalui perulangan, index elemen dapat digantikan dengan variabel *counter*. Untuk lebih jelasnya, perhatikan contoh program 03.latihan_13.php dibawah ini.

03.latihan_13.php

```
<?php

$teman[] = "Yudistira";
$teman[] = "Bima";
$teman[] = "Arjuna";

for($i = 0; $i < count($teman); $i++){
   echo $teman[$i];
   echo "<br />";
}

?>
```



Pada contoh program diatas, index dapat digantikan oleh variabel counter `$i`. Fungsi `count()` digunakan untuk menghitung jumlah elemen yang terdapat pada array `$teman` yang nantinya digunakan sebagai stop condition dari perulangan.

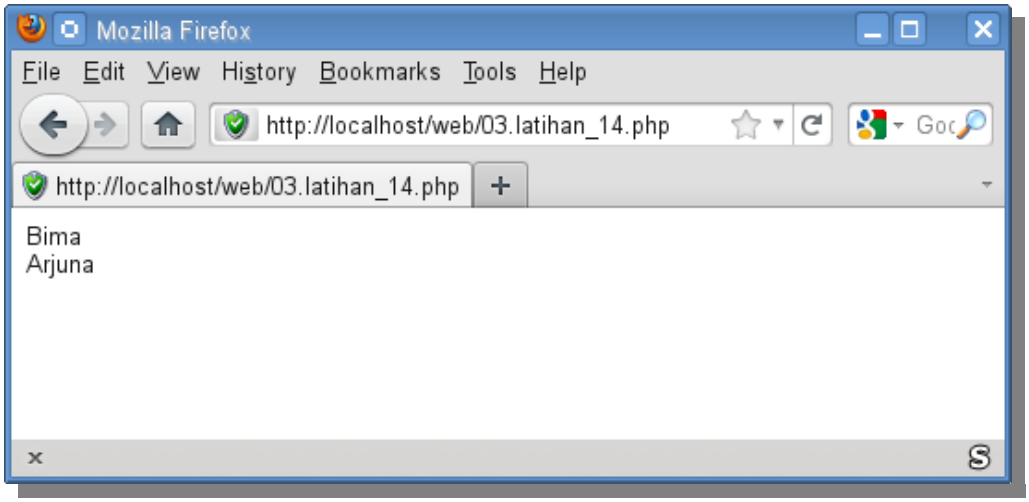
2.2 Index Asosiatif

Seperti halnya pada array dengan index numeric, terdapat dua cara pengaksesan array menggunakan index asosiatif. Yaitu:

a. Pengaksesan Langsung. Pengaksesan array secara langsung dapat dilakukan dengan cara menuliskan nama array berserta label index asosiatifnya.

03.latihan_14.php

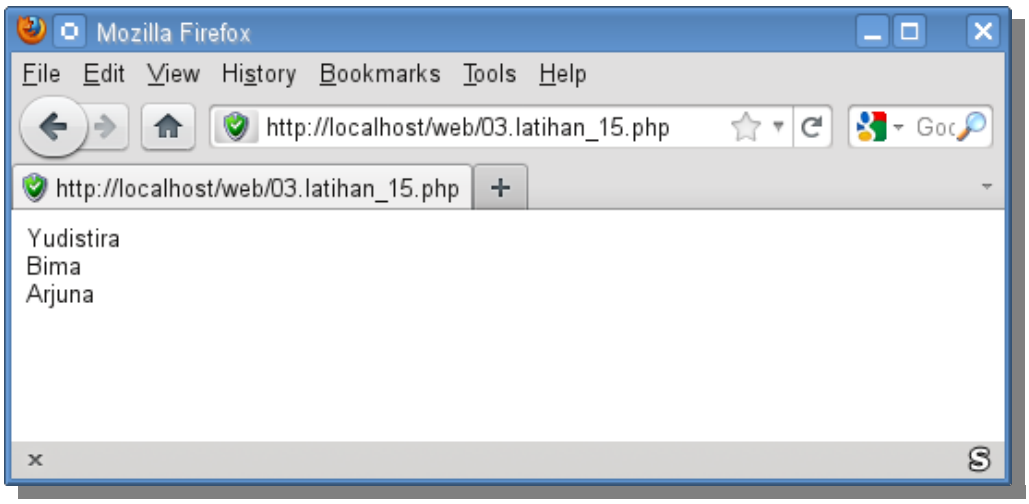
```php
<?php

$teman['wayang_1'] = "Yudistira";
$teman['wayang_2'] = "Bima";
$teman['wayang_3'] = "Arjuna";

echo $teman['wayang_2'];
echo "<br />";
echo $teman['wayang_3'];

?>
```



b. Melalui perulangan. Karena index yang digunakan bukanlah numeric, maka perulangan menggunakan `for` tidak dapat digunakan. Sebagai gantinya kita menggunakan perulangan `foreach`.

03.latihan_14.php

```php
<?php

$teman['wayang_1'] = "Yudistira";
$teman['wayang_2'] = "Bima";
$teman['wayang_3'] = "Arjuna";

foreach($teman as $key => $tmp){
   echo $tmp;
   echo "<br />";
}

?>
```



Pada contoh program diatas, perulangan menggunakan `foreach` secara otomatis akan menampilkan semua *value* pada array tanpa harus menyebutkan index asosiatifnya.

Untuk menampilkan *value* menggunakan perulangan `foreach`, terlebih dahulu array `$teman` akan diubah menjadi variabel sementara `$tmp`, yang mengacu pada elemen array yang sedang diakses saat itu. Sedangkan varibel `$key` mengacu pada index asosiatif elemen array yang saat itu sedang diakses.

DAFTAR PUSTAKA

1. Buzton, Toby. 2002. *PHP By Example*. Indianapolis, Indiana: Que.
2. Choi, Wankyu., Dkk. 2000. *Beginning PHP 4*. Birmingham: Wrox Press.
3. Yuliano,Triswansyah. 2007. *Pengenalan PHP*. IlmuKomputer.Com.
4. Muhardin, Endy. 2003. *PHP Programming Fundamental dan MySQL Fundamental*. IlmuKomputer.Com