

MODUL II-STRUKTUR KONTROL

2.1 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Mengerti esensi penggunaan percabangan (pemilihan/kondisional) dalam PHP.
2. Memahami bentuk umum percabangan.
3. Mampu memecahkan masalah sederhana dengan menggunakan percabangan dan mengimplementasikan ke dalam program PHP.
4. Mengerti esensi penggunaan pengulangan (looping) dalam PHP.
5. Memahami bentuk umum pengulangan.
6. Mampu memecahkan masalah sederhana dengan menggunakan pengulangan dan mengimplementasikan ke dalam program PHP.

2.2 Struktur Kontrol

2.2.1 Struktur Kondisional

Pengertian Struktur IF dalam bahasa pemrograman adalah sebuah struktur logika untuk membuat percabangan alur program. Secara sederhananya, dengan menggunakan **struktur IF** kita dapat mengatur apakah sebuah perintah akan dijalankan atau tidak tergantung kepada kondisinya.

2.2.1.1 Aturan Penulisan Struktur IF

Penulisan sederhana dari **struktur if** adalah sebagai berikut:

```
<?php
    if (expression)
        statement;
?>
```

Expression ditulis di dalam tanda kurung, dan tidak diikuti dengan titik koma(;).

Apabila *statement* yang ingin dijalankan terdiri dari 2 baris atau lebih, kita harus memberikan tanda kurung kurawal untuk menandai *statement* yang berhubungan dengan kondisi IF. Berikut contoh strukturnya:

```
<?php
if (expression)
{
    statement1;
    statement1;
}
?>
```

Tanda kurung kurawal menandakan blok perintah yang dijalankan jika *expression* bernilai true. Kita juga bisa membuat beberapa logika IF sekaligus untuk berbagai situasi:

```
<?php
if (expression1)
{
    statement1;
    statement2;
}
if (expression2)
{
    statement3;
    statement4;
}
?>
```

Untuk kasus yang lebih spesifik, kita bisa membuat struktur IF didalam IF, atau dikenal dengan **nested IF**, seperti contoh berikut:

```
if (expression)
{
    statement1;
    if (expression)
    {
        statement1;
    }
}
?>
```

Seberapa banyak kondisi IF didalam IF (*nested*) tidak dibatasi dalam PHP, namun perlu diperhatikan penggunaan tanda kurung kurawal sebagai penanda bagian dari IF. Jika anda membuat **struktur IF** yang kompleks, tanda kurung kurawal ini akan membuat bingung jika tidak dikelola dengan benar. Kesalahan penutupan kurung kurawal akan membuat program tidak berjalan sesuai dengan keinginan.

2.2.1.2 Pengertian Struktur logika Switch

Struktur logika switch adalah sebuah stuktur percabangan yang akan memeriksa suatu *variabel*, lalu menjalankan perintah-perintah yang sesuai dengan kondisi yang mungkin terjadi untuk variabel tersebut. Struktur *switch* ini mirip dengan struktur IF yang ditulis berulang.

Katakan kita ingin membuat sebuah program yang akan menampilkan kata dari angka 0-5, sehingga terdapat 6 kemungkinan yang terjadi. Jika menggunakan struktur IF, maka kita akan membutuhkan 6 perulangan sebagai berikut:

```
1 <?php
2     $a=3;
3     if ($a=="0") {
4         echo "Angka Nol";
5     }
6     elseif ($a=="1") {
7         echo "Angka Satu";
8     }
9     elseif ($a=="2") {
10        echo "Angka Dua";
11    }
12    elseif ($a=="3") {
13        echo "Angka Tiga";
14    }
15    elseif ($a=="4") {
16        echo "Angka Empat";
17    }
18    elseif ($a=="5") {
19        echo "Angka Lima";
20    }
21    else
22        echo "Angka diluar jangkauan";
23 ?>
```

Tidak ada yang salah dari kode program tersebut, namun jika kita menggunakan **switch**, kode tersebut dapat ditulis menjadi:

```

1  <?php
2  $a=3;
3  switch ($a)
4  {
5      case 0 :
6          echo "Angka Nol";
7          break;
8      case 1 :
9          echo "Angka Satu";
10         break;
11     case 2 :
12         echo "Angka Dua";
13         break;
14     case 3 :
15         echo "Angka Tiga";
16         break;
17     case 4 :
18         echo "Angka Empat";
19         break;
20     case 5 :
21         echo "Angka Lima";
22         break;
23     default :
24         echo "Angka diluar jangkauan";
25         break;
26 }
27 ?>

```

Kedua kode program akan menghasilkan output yang sama, namun untuk kondisi logika yang diuji merupakan kondisi sederhana, penulisan dengan switch lebih disarankan dibandingkan IF.

Aturan Penulisan Struktur Switch dalam PHP

Seperti yang terlihat dalam contoh sebelumnya, struktur **switch** terdiri dari beberapa bagian, berikut format dasar penulisan **switch** dalam PHP:

```

switch ($var)
{
case value1:
    statement1;
    break;
case value2:
    statement2;
    break;
}

```

Setelah kata kunci **switch**, kita harus mencantumkan *variabel* yang akan diperiksa nilainya didalam tanda kurung, lalu memulai block **switch** dengan *kurung kurawal*.

Tiap kondisi yang mungkin terjadi dicantumkan setelah kata kunci **case**, lalu diikuti dengan nilai yang akan dibandingkan dengan nilai *variabel switch*. Jika kondisi sesuai, maka baris

program *statement* akan dijalankan. Kata kunci **break** digunakan untuk keluar dari *switch*, sehingga PHP tidak perlu memeriksa *case* berikutnya.

Alur program untuk **switch** akan dieksekusi dari baris pertama sampai terakhir. Kata kunci **break** memegang peranan penting untuk menghentikan **switch**.

```
1  <?php
2      $a=9;
3      switch ($a)
4      {
5          case 0:
6              echo "Angka Nol ";
7              break;
8          case 1 :
9              echo "Angka Satu ";
10             break;
11         case 2 :
12             echo "Angka Dua ";
13             break;
14         case 3 :
15             echo "Angka Tiga ";
16             break;
17         default :
18             echo "Angka diluar jangkauan";
19             break;
20     }
21  ?>
```

PHP membolehkan kita menjalankan satu *statement* saja untuk *case* yang berlainan, seperti contoh kode PHP berikut ini:

```

1  <?php
2      $a=3;
3      switch ($a)
4      {
5          case 0 :
6          case 1 :
7          case 2 :
8          case 3 :
9              echo "Angka berada di dalam range 0-3";
10             break;
11          case 4 :
12          case 5 :
13          case 6 :
14          case 7 :
15              echo "Angka berada di dalam range 4-7";
16              break;
17          default :
18              echo "Angka diluar jangkauan";
19              break;
20      }
21  ?>

```

Contoh lain penggunaan fungsi IF dan Switch:

Statement kondisional	Contoh
if-else-elseif	<pre> <?php \$nilai = 83; if (\$nilai > 70) { echo "Selamat Anda Lulus Ujian"; } elseif (\$nilai == 70) { echo "Anda harus ujian lagi"; } else { echo "Anda Tidak Lulus"; } ?> </pre>
Switch-case	<pre> <? switch (\$bilangan){ case 28: echo "bilangan genap"; break; case 3: echo "bilangan ganjil"; break; default: echo "bukan bilangan"; break; } ?> </pre>

2.2.2 Struktur Perulangan

Struktur perulangan (atau dalam *bahasa inggris* disebut dengan **loop**) adalah instruksi program yang bertujuan untuk mengulang beberapa baris perintah. Dalam merancang perulangan kode program, kita setidaknya harus mengetahui 3 komponen, yaitu *kondisi awal dari perulangan, perintah program yang akan diulang, serta kondisi akhir dimana perulangan akan berhenti.*

2.2.2.1 Penulisan Struktur Perulangan For

Seperti yang telah kita singgung sebelumnya, untuk kondisi **perulangan for**, kita setidaknya membutuhkan 3 kondisi, yaitu di kondisi **awal perulangan**, kondisi pada **saat perulangan**, dan kondisi yang harus dipenuhi agar **perulangan berhenti**.

Penulisan dasar format **perulangan for** PHP adalah sebagai berikut:

```
for (start; condition; increment)
{
    statement;
}
```

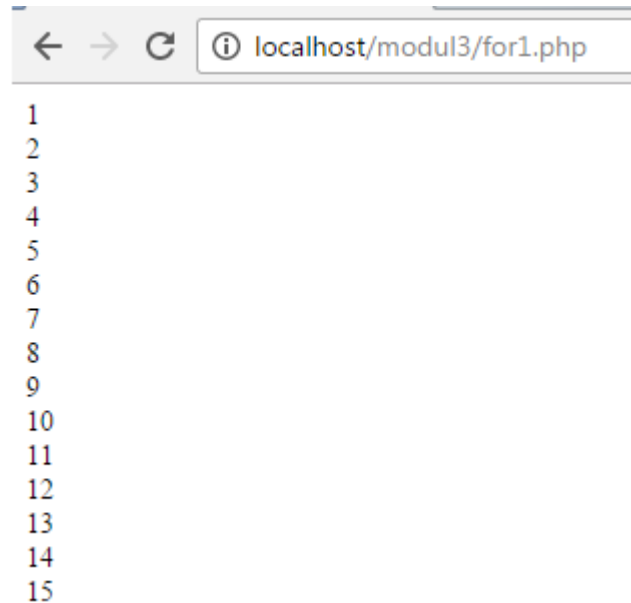
- **Start** adalah kondisi pada saat awal perulangan. Biasanya kondisi awal ini digunakan untuk membuat dan memberikan nilai kepada **variabel counter** yang digunakan untuk mengontrol perulangan. Misalkan, kita akan membuat variabel counter **\$i**, maka untuk kondisi start ini, kita juga harus memberikan nilai awal untuk variabel **\$i**, misalnya dengan **1**, maka **\$i=1**.
- **Condition** adalah kondisi yang harus dipenuhi agar perulangan dijalankan. Selama kondisi ini terpenuhi, maka PHP akan terus melakukan perulangan. Biasanya **variabel counter** digunakan untuk mengatur akhir perulangan. Misalkan kita ingin menghentikan perulangan jika variabel **\$i** telah mencapai nilai 20, maka pada bagian **condition** ini kita membuat perintah **\$i<=20**, yang berarti selama nilai **\$i** kurang atau sama dengan 20, terus lakukan perulangan.
- **Increment** adalah bagian yang digunakan untuk memproses variabel counter agar bisa memenuhi kondisi akhir perulangan. Biasanya, pada bagian inilah kita akan membuat kondisi dari **variabel counter**.
- **Statement** adalah bagian kode program yang akan diproses secara terus-menerus selama proses perulangan berlangsung. Untuk **statement** ini, kita membuat blok program di antara **tanda kurung kurawal** ({ dan }) sebagai penanda bahwa bagian di dalam kurung kurawal inilah yang akan dikenai proses perulangan.

Sebagai contoh, kita akan membuat perulangan untuk menampilkan angka 1-15 kedalam web browser, berikut kode PHP yang digunakan:

```

1 <?php
2 for ($i= 1; $i <= 15; $i++)
3 {
4     echo $i;
5     echo "<br />";
6 }
7 ?>

```



1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

Jika anda menjalankan kode tersebut, maka di dalam web browser akan tampil urutan angka dari 1 sampai dengan 15.

Sebagai kondisi awal dari perulangan tersebut adalah **\$i=1**, dimana kita memberikan nilai 1 kepada variabel **\$i**.

Variabel **\$i** ini lah yang akan menjadi **counter** atau *penghitung* dari **perulangan for**.

Untuk kondisi akhir, kita membuat **\$i <= 15**, jadi selama variabel **\$i** bernilai kurang atau sama dengan 15, maka perulangan akan terus dijalankan.

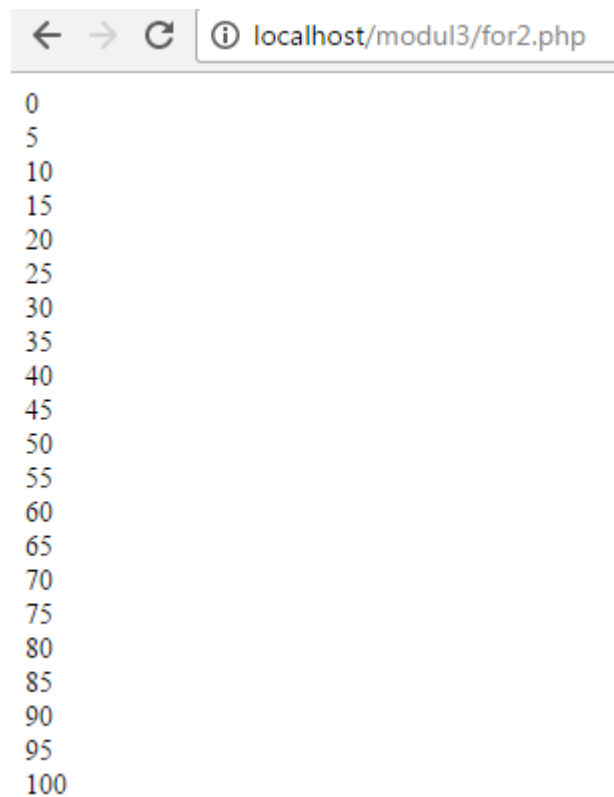
Sebagai **increment**, kita membuat **\$i++**, dimana instruksi ini sama dengan **\$i=\$i+1**. instruksi ini akan dijalankan pada setiap perulangan, sehingga dengan kata lain, setiap proses perulangan, **\$i** akan bertambah 1 angka.

Selain berfungsi sebagai **counter**, variabel **\$i** juga dapat digunakan dalam proses perulangan, sehingga dengan membuat perintah **echo \$i**, maka dalam setiap perulangan, kita bisa menampilkan nilai **\$i** pada saat itu.

Sebagai contoh lain, kita ingin membuat perulangan untuk menampilkan angka 0-100, namun untuk kelipatan 5, seperti: 0.5.10..dst, sampai dengan 100.

Berikut adalah contoh kode PHPnya:

```
1 <?php
2 for ($i= 0; $i <= 100; $i=$i+5)
3 {
4     echo $i;
5     echo "<br />";
6 }
7 ?>
```



← → ↻ ⓘ localhost/modul3/for2.php

0
5
10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100

Perbedaan penulisan **struktur for** diatas dibandingkan contoh sebelumnya adalah pada bagian **increment**, dimana kita membuat kondisi **increment** yang menaik sebanyak 5 angka setiap perulangannya (**$i = i + 5$**). Sehingga **variabel counter**, **i** akan bertambah sebanyak 5 pada setiap perulangan.

Kita juga bisa membuat perulangan dengan kondisi mundur, seperti contoh kode PHP berikut ini:

```

1  <?php
2  for ($i= 20; $i >= 1; $i--)
3  {
4      echo $i;
5      echo "<br />";
6  }
7  ?>

```

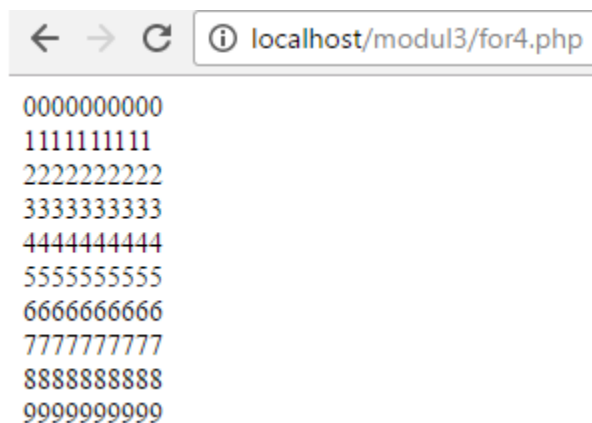
Di dalam kode tersebut, kita memulai nilai awal dari angka **\$i= 20**, membuat perulangan selama **\$i >= 1**, dan pada setiap perulangan, nilai **\$i** akan dikurangi 1 angka (**\$i--**). Dengan kondisi tersebut, maka variabel counter **\$i** akan dikurangi 1 pada setiap perulangan.

Nested loop adalah istilah pemrograman yang berarti membuat perulangan di dalam perulangan. Perhatikan contoh program berikut:

```

1  <?php
2  for ($i=0; $i <10; $i++)
3  {
4      for ($j=0; $j <10; $j++)
5      {
6          echo $i;
7      }
8      echo "<br />";
9  }
10 ?>

```



← → ↻ ⓘ localhost/modul3/for4.php

0000000000
 1111111111
 2222222222
 3333333333
 4444444444
 5555555555
 6666666666
 7777777777
 8888888888
 9999999999

2.2.2.2 Penulisan Struktur Perulangan While

Seperti terlihat pada contoh program sebelumnya, struktur **while** dalam PHP terdiri dari 2 bagian, yaitu *kondisi yang harus dipenuhi untuk proses perulangan*, dan *baris perintah yang akan diproses secara berulang*.

Struktur dasar **perulangan while** adalah sebagai berikut:

```
while (condition)
{
    statement;
    statement;
}
```

- **Condition** adalah kondisi yang harus dipenuhi agar perulangan berlangsung. Kondisi ini mirip seperti dalam **perulangan for**. Selama **condition** bernilai **TRUE**, maka perulangan akan terus dilakukan. **Condition** ini akan diperiksa pada tiap perulangan, dan hanya jika hasilnya **FALSE**, maka proses perulangan berhenti.
- **Statement** adalah kode program yang akan diulang. Kita bisa membuat beberapa kode program untuk menampilkan perintah seperti *echo*, atau perintah yang lebih kompleks. Namun di dalam bagian ini harus ada baris program yang digunakan sebagai *'penghenti'* perulangan. Misalkan pada bagian **condition** kita menggunakan *variabel counter \$i*, maka di bagian **statement** harus ada baris program yang membuat **condition** bernilai **FALSE**, atau kalau tidak proses perulangan tidak akan pernah berhenti (*infinity loop*).
- Tanda **kurung kurawal** diperlukan untuk membatasi blok program yang akan diulang. Jika **statement** hanya terdiri dari 1 baris, maka tanda kurung kurawal tidak diperlukan.

Sebagai pembahasan, kita akan menampilkan ulang contoh kode program sebelumnya, yakni:

```
1 <?php
2 $i=1;
3 while ($i <= 10)
4 {
5     echo "$i";
6     echo "<br />";
7     $i=$i+1;
8 }
9 ?>
```

Pada baris ke-2 kita membuat sebuah variabel **\$i**, dan memberikan nilai 1. Variabel **\$i** inilah yang akan digunakan sebagai **counter** untuk **kondisi while**.

Setelah penulisan **while**, selanjutnya didalam tanda kurung adalah **condition** yang harus dipenuhi agar perulangan berjalan. Kita membuat kondisi (**`$i <= 10`**) sebagai penanda akhir **while**, yang berarti selama variabel **`$i`** bernilai kurang dari 10, maka lakukan perulangan.

Penting untuk diperhatikan adalah logika pemograman untuk **condition**. **While** (**`$i <= 10`**) juga berarti bahwa jika nilai variabel **`$i = 11`**, maka perulangan akan berhenti. Di dalam kode program, kita harus membuat sebuah baris **statement** yang digunakan untuk terus menambahkan nilai **`$i`** supaya nilai **`$i`** bisa mencapai angka lebih dari 10 untuk menghentikan perulangan.

Setelah membuat beberapa baris kode **echo** untuk menampilkan angka ke web browser pada baris ke-5 dan 6, kita menambahkan kode **`$i=$i+1`** pada baris ke-7 Baris inilah yang akan menambahkan nilai variabel counter **`$i`** sebanyak 1 angka pada tiap perulangan, sehingga pada perulangan ke 10, nilai **`$i`** akan menjadi 11. Dan kondisi **while** akan menghasilkan **FALSE**, sehingga proses perulangan berhenti.

Anda juga bebas menentukan awal dari variabel **counter** **`$i`**, misalnya untuk mulai dari angka 100 dan mundur ke belakang seperti contoh berikut:

```
1 <?php
2 $i=100;
3 while ($i >= 0)
4 {
5     echo "$i";
6     echo "<br />";
7     $i-=8;
8 }
9 ?>
```

Perulangan while tersebut akan menghasilkan angka menurun dari 100 sampai dengan 0, dimana pada setiap perulangan nilai 100 akan dikurangi dengan 8.

Penulisan Nested Loop untuk While

Walaupun **struktur while** agak jarang digunakan untuk **nested loop**, anda bisa membuat perulangan bersarang dengan **struktur while**, seperti contoh berikut ini:

```

1  <?php
2  $i=0;
3  while ($i < 10)
4  {
5      $j=0;
6      while ($j < 10)
7      {
8          echo $i;
9          $j++;
10     }
11     echo "<br />";
12     $i++;
13 }
14 ?>

```

2.2.3 Fungsi Perintah Break Dalam Perulangan

Ketika proses perulangan berjalan, ada kalanya kita ingin segera keluar dari perulangan jika sebuah kondisi tertentu telah terpenuhi, sehingga sisa proses perulangan tidak perlu dijalankan.

Misalkan kita memiliki nama-nama mahasiswa yang tersimpan di dalam sebuah array atau di dalam database. Proses pencarian sederhana dapat dirancang dengan melakukan pencocokan secara berulang dimulai dari nama pertama, kedua, dan seterusnya. Perulangan ini akan dilakukan sebanyak daftar mahasiswa yang ada.

Akan tetapi, jika nama yang dicari telah ditemukan, proses perulangan seharusnya dapat dihentikan saat itu juga, karena tujuan pencarian nama telah selesai.

Untuk keperluan inilah PHP menyediakan instruksi **break**. **Break** berfungsi sebagai perintah kepada **web server** untuk menghentikan perulangan secara *prematur*, yaitu menghentikan perulangan di luar dari yang direncanakan.

Cara Penulisan Perintah Break

Perintah **break** dapat di letakkan di posisi manapun di dalam perulangan, namun biasanya kita akan membuat **logika IF** untuk menentukan kapan perintah **break** akan dijalankan.

Contoh Penggunaan Break dalam Perulangan For

Berikut adalah contoh program perulangan for dengan menggunakan perintah **break**:

```

1  <?php
2  for ($i=0; $i <100; $i++)
3  {
4      if ($i==13)
5      {
6          break;
7      }
8      echo $i;
9      echo "<br />";
10 }
11 ?>

```

Dalam program diatas, kita membuat perulangan for dari 0 sampai 100, dan dalam keadaan normal, perintah **for** (**\$i=0; \$i <100; \$i++**) akan memproses perulangan sebanyak 100 kali. Namun pada baris ke-4 kita menambahkan sebuah **struktur IF** yang menyatakan bahwa jika nilai variabel **counter** **\$i** sama dengan 13, maka **break**. Perintah **break** akan membuat perulangan **for** langsung dihentikan, dan kita hanya menghasilkan perulangan sampai angka 12. Ini terjadi karena perintah **echo \$i** ditempatkan setelah **break**. Jika perintah **echo \$i** ini anda pindahkan di baris sebelum **break**, akan tampil angka 13 (yang menandakan kalau perulangan sudah masuk ke **\$i = 13**).

2.2.4 Fungsi Perintah Continue dalam PHP

Perintah **continue** juga digunakan untuk men-interupsi perulangan dalam PHP, namun jika perintah **break** digunakan untuk menghentikan perulangan, maka perintah **continue** hanya akan menghentikan perulangan untuk 1 iterasi saja, lalu proses perulangan akan dilanjutkan. Berikut contoh kode PHP penggunaan perintah **continue**:

```

1  <?php
2  for ($i=0; $i <10; $i++)
3  {
4      if ($i==7)
5      {
6          continue;
7      }
8      echo $i;
9      echo "<br />";
10 }
11 ?>

```

Contoh perulangan diatas mirip dengan contoh pada perulangan break. Setelah perintah **for**, kita membuat sebuah kondisi **IF** yang jika **variabel counter \$i** bernilai 7, maka jalankan **continue**.

Arti dari **continue** ini adalah sebuah instruksi kepada PHP untuk melewati sisa perintah dalam perulangan, dan langsung lompat ke nilai counter berikutnya, yakni 8. Dari hasil program, anda tidak akan melihat angka 7 ditampilkan.

Pengertian Perulangan Foreach dalam PHP

Array merupakan tipe data yang sering digunakan dalam membuat program menggunakan PHP. Kemampuan **array** dalam menyimpan banyak data dalam satu variabel akan sangat berguna untuk menyederhanakan dan menghemat penggunaan variabel.

Cara Penulisan Perulangan Foreach dalam PHP

Perulangan foreach merupakan perulangan khusus untuk pembacaan nilai **array**. Seperti yang telah kita bahas pada tutorial tentang tipe data array: Menenal Tipe Data Array dan Cara Penulisan Array dalam PHP, setiap array memiliki pasangan **key** dan **value**. **Key** adalah 'posisi' dari **array**, dan **value** adalah 'isi' dari **array**.

Format dasar perulangan **foreach** adalah:

```
foreach ($nama_array as $value)
{
    statement (...$value...)
}
```

- **\$nama_array** adalah nama dari **array** yang telah didefenisikan sebelumnya.
- **\$value** adalah nama 'variabel perantara' yang berisi data array pada perulangan tersebut. Anda bebas memberikan nama untuk variabel perantara ini, walaupun pada umumnya banyak programmer menggunakan **\$value**, atau **\$val** saja.

Berikut adalah contoh perulangan **foreach**:

```
1 <?php
2 $nama = array("Andri", "Joko", "Sukma", "Rina", "Sari");
3
4 foreach ($nama as $val)
5 {
6     echo "$value";
7     echo "<br />";
8 }
9 ?>
```

Pada contoh diatas, kita mendefenisikan variabel array **\$nama** dengan format singkat, dan tanpa mendefenisikan **key** secara tertulis. Variabel **\$val** merupakan *variabel perantara* dalam contoh diatas. Perulangan tersebut akan diulang sebanyak data yang terdapat di dalam array, sehingga kita tidak perlu harus menghitung seberapa banyak perulangan yang harus dilakukan.

Jika anda membutuhkan nilai **key** dari **array** untuk dapat diproses, maka PHP menyediakan bentuk kedua dari perulangan **foreach**, dengan format dasar penulisan sebagai berikut:

```
foreach ($nama_array as $key => $value)
{
    statement ($key...$value...)
}
```

Perbedaan dengan format sebelumnya, disini PHP menyediakan variabel perantara kedua, yaitu variabel **\$key**. Variabel **\$key** ini menampung nilai **key** dari array.

Berikut adalah contoh penggunaannya:

```
1 <?php
2 $nama = array(
3     1=>"Andri",
4     6=>"Joko",
5     12=>"Sukma",
6     45=>"Rina",
7     55=>"Sari");
8
9 foreach ($nama as $kunci =>$isi)
10 {
11     echo "Urutan ke-$kunci adalah $isi";
12     echo "<br />";
13 }
14 ?>
```

← → ↻ ⓘ localhost/modul3/for_array3.php

Urutan ke-1 adalah Andri
Urutan ke-6 adalah Joko
Urutan ke-12 adalah Sukma
Urutan ke-45 adalah Rina
Urutan ke-55 adalah Sari

Variabel array **\$nama** kita defenisikan menggunakan key yang berbeda-beda. Pada perulangan **foreach**, kita membuat variabel perantara **\$kunci =>\$isi**, sehingga didalam perulangan, variabel **\$kunci** akan berisi **key** dari array, dan variabel **\$isi** akan berisi **nilai** dari array.

Proses menampilkan dan memproses **array** akan lebih mudah dengan menggunakan [perulangan foreach](#) dibandingkan perulangan dasar seperti **for**. Terlebih lagi kita tidak perlu mencari tau seberapa banyak perulangan harus dilakukan, karena perulangan **foreach** akan otomatis berhenti pada data terakhir dari array.