LAPORAN PROYEK 4

STRUKTUR KONTROL PERULANGAN DAN ARRAY PHP

OLEH:

FIKA LAURA CITRA

(NISN.)

REKAYASA PERANGKAT LUNAK
SMK NEGERI 1 KARANG BARU
PEMERINTAH PROVINSI ACEH
2024

MATERI PERTEMUAN 8
Memahami Perulangan Pemrograman

TAHUKAH KAMU...?

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

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

Perulangan, Iteration dalam Pemrograman

Apa yang akan kamu lakukan bila disuruh membuat daftar judul artikel dengan PHP?
Apakah akan mencetaknya satu per satu dengan perintah **echo** seperti ini:

```
<?php

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

?>
```

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

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

1. Perulangan For

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

```
<?php

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

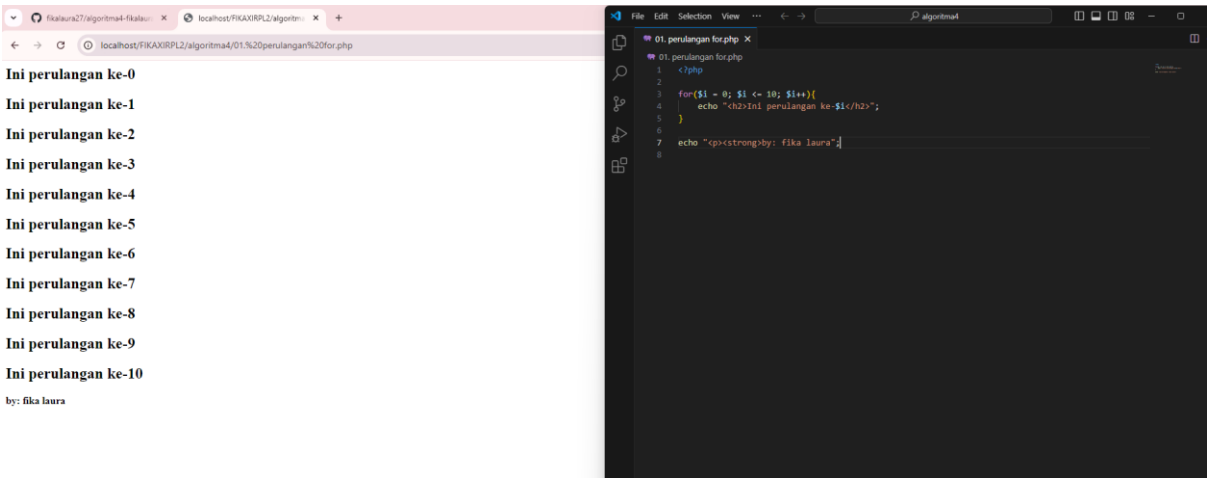
Veriabel **$i** dalam perulangan *For* berfungsi sebagai *counter* yang menghitung berpa kali ia akan mengulang.
Hitungan akan dimulai dari nol (**0**), karena kita memberikan nilai **$i = 0**.
Lalu, perulangan akan diulang selama nilai **$i** lebih kecil dari **10**. Artinya, perulangan ini akan mengulang sebanyak **10x**.
Maksud dari **$i++** adalah nilai **$i** akan ditambah **1** disetiap kali melakukan perulangan.

Contoh:

```
<?php
```

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

Hasilnya:



2. Perulangan While

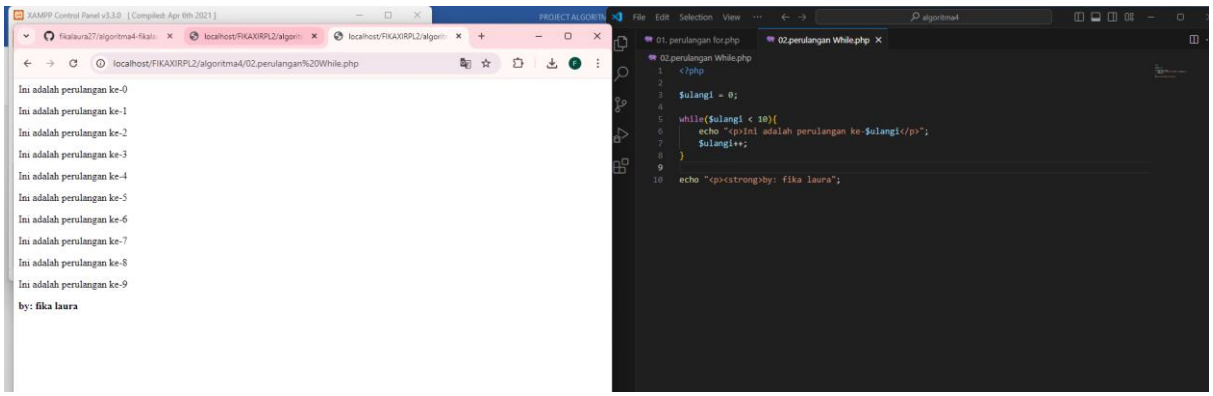
Perulangan *while* adalah perulangan yang termasuk dalam *uncounted loop*. Karena biasanya digunakan untuk mengulang sesuatu yang belum jelas jumlah pengulangannya.
Namun, perulangan *while* juga bisa digunakan seperti perulangan *for* sebagai *counted loop*.
Bentuk dasarnya:

```
<?php

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

?>
```

Contoh:

```
<?php

$ulangi = 0;

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

?>
```

Hasilnya:



Perulangan *while* akan terus mengulang selama nilai **$ulangi** lebih kecil dari **10**.
Lalu di dalam perulangan kita melakukan *increment* nilai **$ulangi** dengan **$ulangi++**.
Artinya: Tambah 1 disetiap pengulangan.
Hati-hati, jangan sampai lupa menambahkan *increment*, atau kode yang akan mempengaruhi pengulangan. Karena kalau tidak, pengulangannya tidak akan pernah berhenti dan akan membuat komputer kita *hang*.

3. Perulangan Do/While

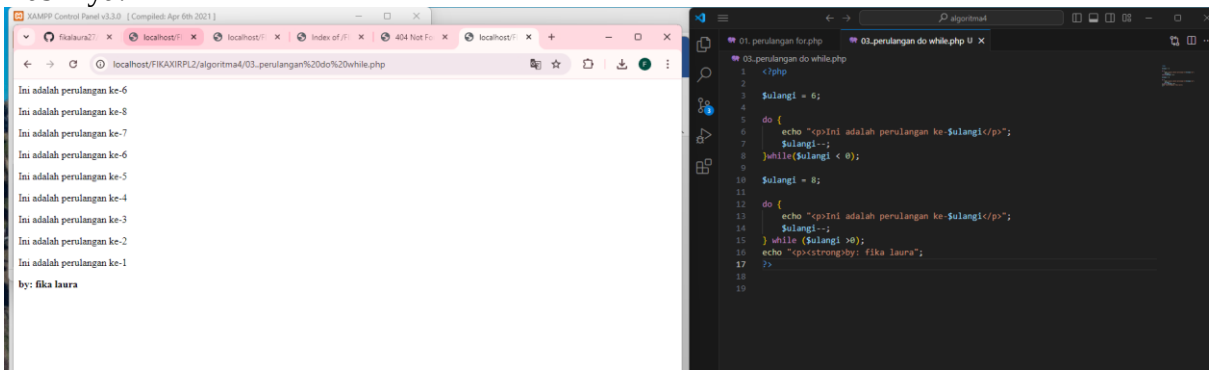
Perulangan *Do/While* sama seperti perulangan *while*. Ia juga tergolong dalam *uncounted loop*.
Perbedaan *Do/While* dengan *while* terletak pada cara iya memulai pengulangan.
Perulangan *Do/While* akan selalu melakukan pengulangan sebanyak 1 kali, kemudian melakukan pengecekan kondisi.
Sedangkan perulangan *while* akan mengecek kondisi terlebih dahulu, baru melakukan pengulangan.
Bentuk perulangan *Do/While*:

```
<?php
do {
    // blok kode yang akan diulang
} while (<kondisi>);
?>
```

Contoh:

```
<?php

$ulangi = 10;

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

Hasilnya:



4. Perulangan Foreach

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

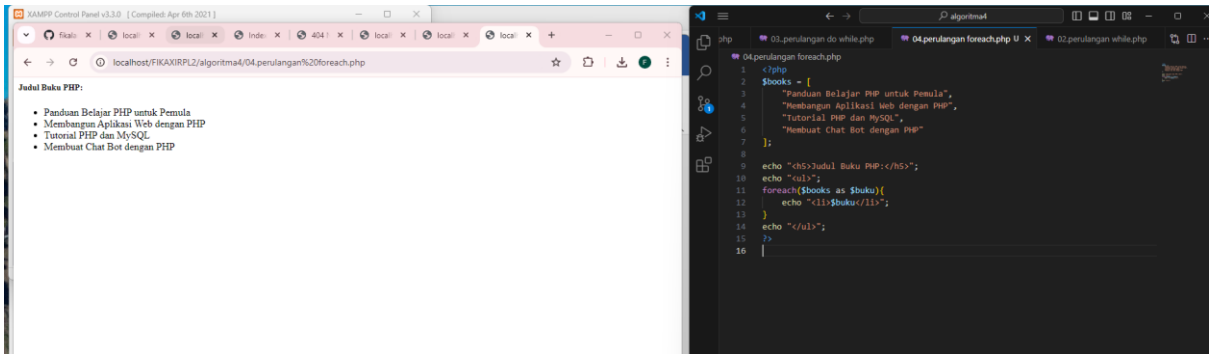
Bentuk perulangan *foreach*:

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

Contoh:

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

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

Hasilnya:



Perulangan Bersarang

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

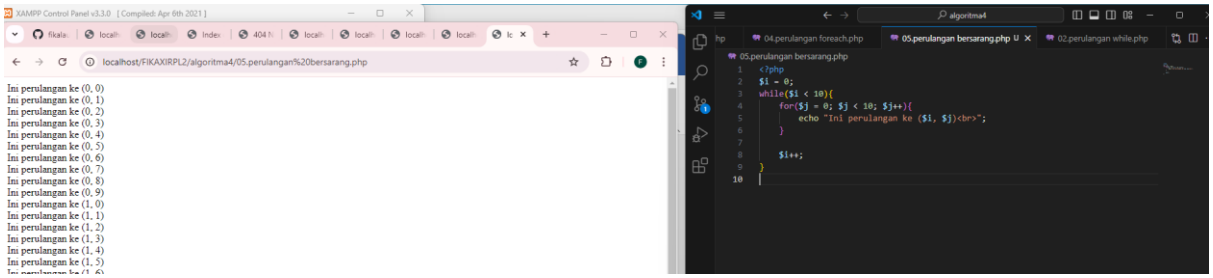
Contoh perulangan bersarang:

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

Contoh lain:

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

```
    $i++;
}
```
Hasilnya:

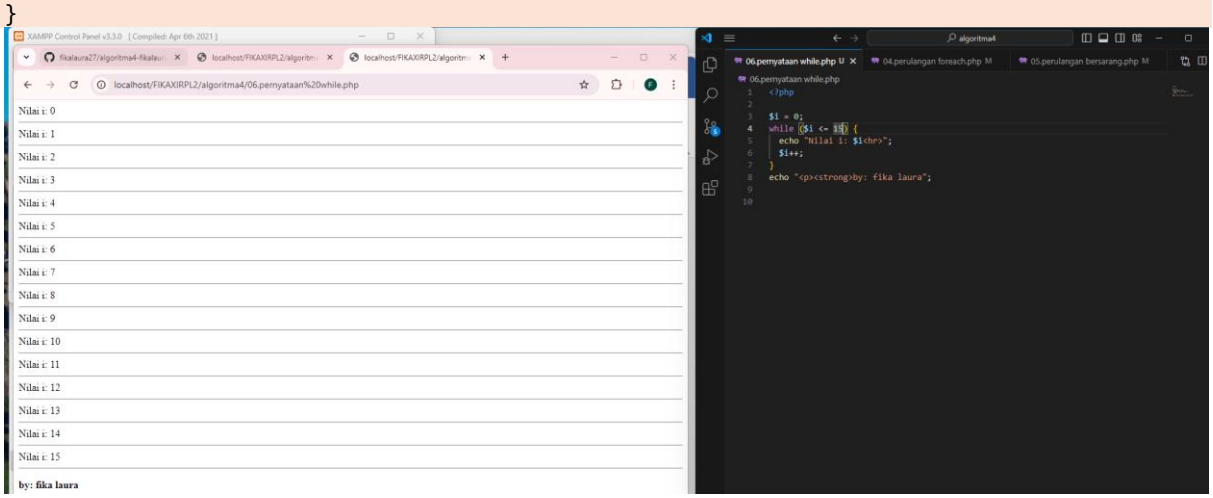


Pernyataan While

while membuat sebuah loop yang menjalankan blok kode selama kondisi tertentu terpenuhi.

```
while ($kondisi) {
  // Kode yang akan dijalankan
}
```
Contoh penggunaan:

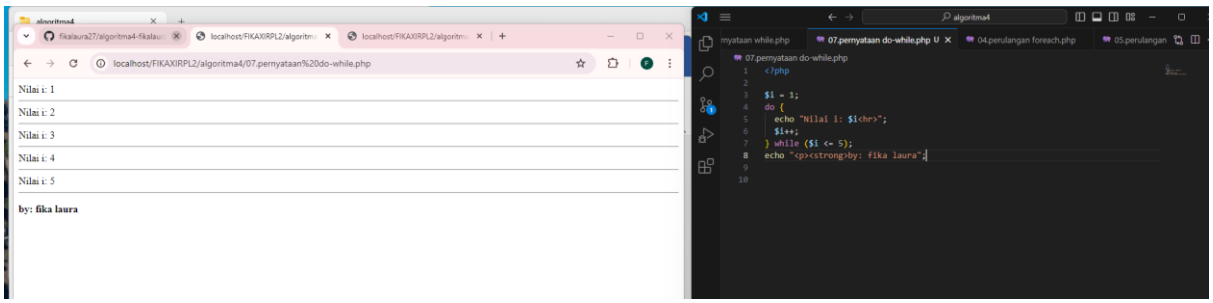
```
$i = 1;
while ($i <= 5) {
  echo "Nilai i: $i";
  $i++;
}
```



Pernyataan Do-While

```
do-while serupa dengan while, tetapi memastikan bahwa kode dijalankan setidaknya satu kali.

do {
  // Kode yang akan dijalankan
} while ($kondisi);
```
Contoh penggunaan:

```
$i = 1;
do {
  echo "Nilai i: $i";
  $i++;
} while ($i <= 5);
```



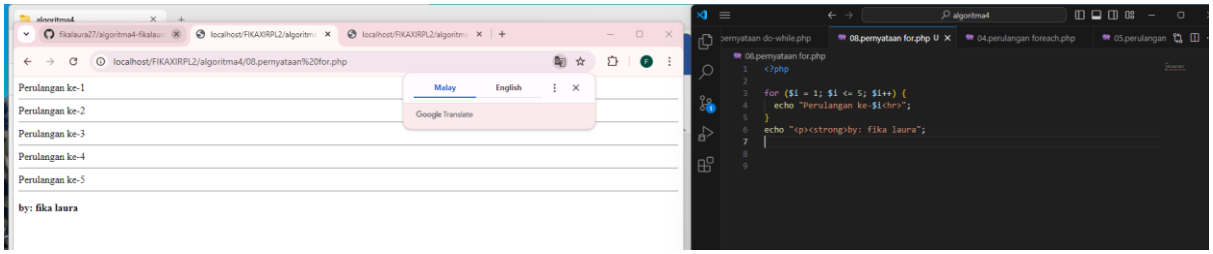
Pernyataan For

for loop digunakan ketika kamu tahu sebelumnya berapa kali kode harus diulang.

```
for (inisialisasi; kondisi; perubahan) {
  // Kode yang akan dijalankan
}
```

Contoh penggunaan:

```
for ($i = 1; $i <= 5; $i++) {
  echo "Perulangan ke-$i";
}
```



Pernyataan Foreach

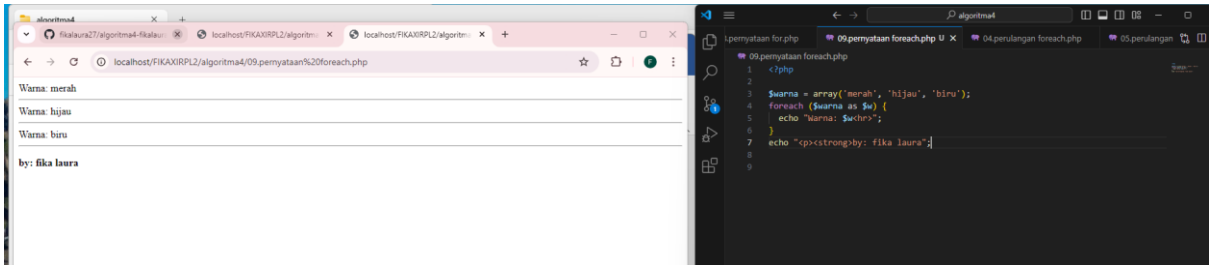
Pada PHP, pernyataan foreach sangat berguna untuk mengulangi semua elemen dalam array.

```
foreach ($array sebagai $nilai) {
  // Kode yang akan dijalankan
}
```

Contoh penggunaan:

```
$warna = array('merah', 'hijau', 'biru');
foreach ($warna as $w) {
  echo "Warna: $w";
}
```

Struktur kontrol adalah kunci untuk membuat kode yang efisien dan mudah dibaca. Dengan menguasai pernyataan kondisional dan perulangan di PHP, kamu akan bisa mengatur bagaimana dan kapan kode dieksekusi dengan lebih tepat.



PENALARAN

I. Penulisan For Pada PHP

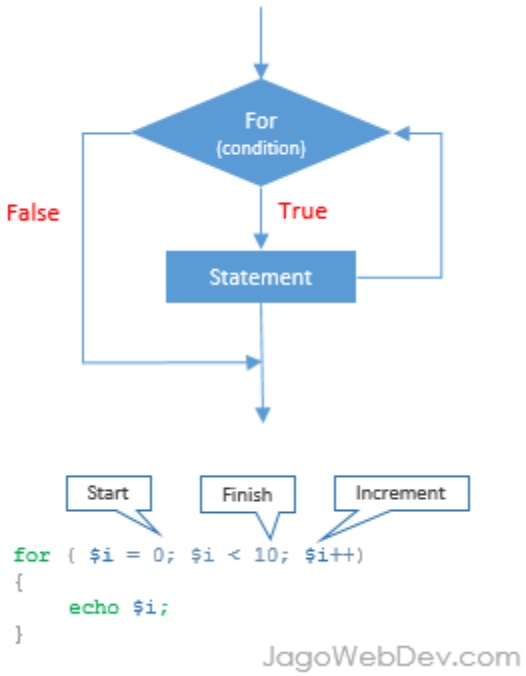
Perulangan for pada PHP dapat ditulis menggunakan kurung kurawa, colon, atau tanpa keduanya:

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

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

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

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

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

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



Ketentuan mengenai ekspresi

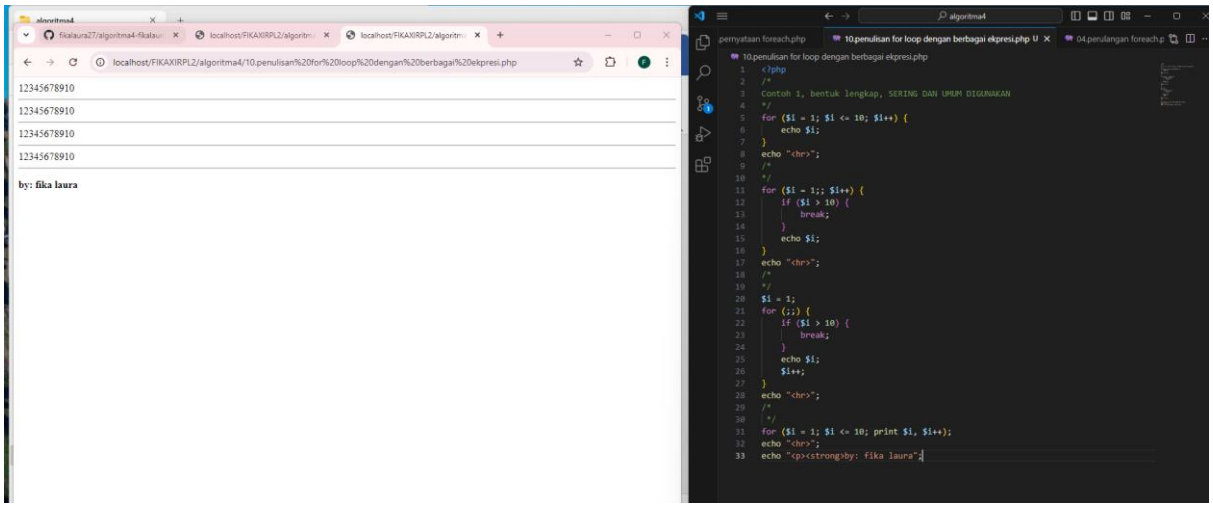
Beberapa ketentuan terkait penulisan ekspresi:

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

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

```
<?php
/*
Contoh 1, bentuk lengkap, SERING DAN UMUM DIGUNAKAN
*/
```

```
for ($i = 1; $i <= 10; $i++) {
    echo $i;
}
echo "<hr>";
/*
Contoh 2, dengan ekspresi2 kosong, kode dihentikan dengan brak statement
*/
for ($i = 1;; $i++) {
    if ($i > 10) {
        break;
    }
    echo $i;
}
echo "<hr>";
/*
Contoh 3, semua ekspresi kosong
*/
$i = 1;
for (;;) {
    if ($i > 10) {
        break;
    }
    echo $i;
    $i++;
}
echo "<hr>";
/*
Contoh 4, tanpa statement hanya ekspresi saja
 */
for ($i = 1; $i <= 10; print $i, $i++);
echo "<hr>";
```

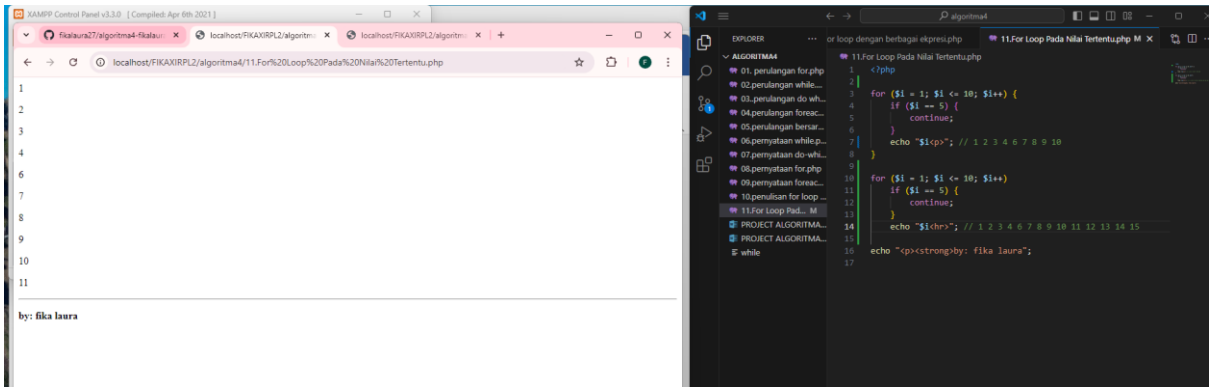


II. Melompati (skip) For Loop Pada Nilai Tertentu

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

```
<?php
```

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

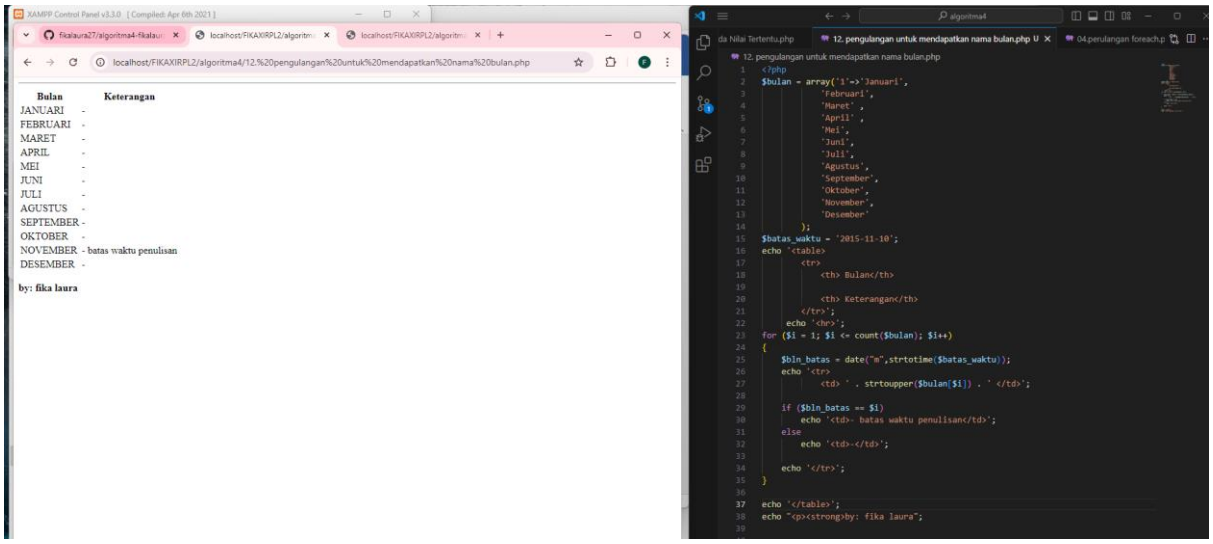


III. Tips Optimasi Perulangan For Pada PHP

Berikut ini beberapa tips yang dapat kita terapkan agar perulangan for dapat dieksekusi dengan cepat
1Hindari pemanggilan fungsi dan pendefinisian variabel yang nilainya tetap di dalam loop
Fungsi ini dapat berupa fungsi bawaan PHP seperti count, substr, strlen, dll maupun fungsi yang kita buat sendiri, contoh berikut pengulangan untuk mendapatkan nama bulan:

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

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

```
        echo '</tr>';
}

echo '</table>';
?>
```



output:

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

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

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

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

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

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

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

Bulan	Deadline	
	Tahapan	Tanggal
JANUARI	Perencanaan	2015-01-31
FEBRUARI	Analisis	2015-02-28
MARET	Perancangan	2015-03-31
APRIL	Penerapan	2015-04-30
MEI	Evaluasi	2015-05-31
JUNI	Penggunaan	2015-06-30
JULI	–	–
AGUSTUS	–	–
SEPTEMBER	–	–
OKTOBER	–	–
NOVEMBER	–	–
DESEMBER	–	–

Kode yang kita gunakan:

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

// Informai yang diperoleh dari database
$tahapan = array(
array('tahap' => 'Perencanaan',     'tgl' => '2015-01-31'),
array('tahap' => 'Analisis',  'tgl' => '2015-02-28'),
array('tahap' => 'Perancangan',     'tgl' => '2015-03-31'),
array('tahap' => 'Penerapan', 'tgl' => '2015-04-30'),
array('tahap' => 'Evaluasi',  'tgl' => '2015-05-31'),
array('tahap' => 'Penggunaan',      'tgl' => '2015-06-30')
                  );
echo '<table>
            <tr>
                  <th rowspan="2">Bulan</th>
                  <th colspan="2">Deadline</th>
            </tr>
            <tr>
                  <th>Tahapan</th>
                  <th>Tanggal</th>
            </tr>
```

```
            ';

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

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

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

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

dari data diatas terdapat pengulangan fungsi date yaitu sebanyak 72 kali (12 x 6), dengan struktur data seperti diatas, agak ribet jika harus memenuhi kondisi ideal seperti contoh sebelumnya.
Kondisi tersebut dapat dipenuhi, namun kode yang ditulis bisa jadi menjadi lebih kompleks dan membutuhkan tenaga yang lebih untuk memahaminya (tergantung kondisi lapangan).
Pada kondisi ini, kondisi ideal dapat dilanggar jika kode yang ditulis menjadi lebih sederhana dan mudah dipahami dan performa aplikasi juga tidak terganggu.
Namun jika tidak ada salahnya kita mencobanya. Dengan sedikit perubahan, kode diatas dapat kita tulis kembali menjadi:

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

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

foreach ($tahapan as $key => $tahap)
{
      $bln_batas = date("n",strtotime($tahap['tgl']));
```

```
	$ref_tahapan[$bln_batas] = $key;
}
echo '<table>
		<tr>
			<th rowspan="2">Bulan</th>
			<th colspan="2">Deadline</th>
		</tr>
		<tr>
			<th>Tahapan</th>
			<th>Tanggal</th>
		</tr>

		';

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

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

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

pada kode diatas line 26 s.d 30 kita membuat variabel baru bernama $ref_tahapan yang berbentuk array dengan key bulan dan value index dari array tahapan, contoh $ref_tahapan[1] = 0 yang berarti bulan 1 merujuk ke $tahapan[0].
Kode diatas tidak terlalu kompleks dan masih wajar untuk digunakan walaupun pengguna kode membuat kita harus menambah tenaga untuk memahami variabel baru $ref_tahapan
Ketika di tes, kode diatas membutuhkan waktu eksekusi 0.0011389255523682 detik sedangkan sebelumnya 0.014470100402832 detik, tidak terasa perbedaannya oleh karena itu kita dapat menggunakan kedua cara diatas, namun jika kode yang ditulis kompleks, cara kedua bisa dipertimbangkan untuk digunakan.

2Hindari eksekusi perintah SQL di dalam loop
Sebisa mungkin jangan pernah melakukan eksekusi kode SQL di dalam loop, dengan eksekusi yang berulang ulang maka akan memberatkan server database yang pada akhirnya akan menurunkan performa aplikasi anda.
Contoh dibawah ini pengulangan eksekusi MySQL di dalam loop (variabel $bulan menggunakan contoh sebelumnya):

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

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

3Berhati – hati dalam penulisan nested loop
Terkadang kita menuliskan banyak loop di dalam loop, untuk kehati-hatian, gunakan variabel yang mencerminkan kondisi yang ada, tidak sekedar $i, mengingat nilai variabel akan berubah jika kita mendefinisikan dengan nama yang sama (baik sengaja maupun tidak), contoh berikut akan menghasilkan

```
*
**
***
****
*****
```

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

KASUS

Cara Membuat Looping For di dalam Tabel

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

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

Script CSS untuk membuat tampilan table di atas lebih menarik.

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

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

```
<form>
              <table border="1" cellspacing="0">
                     <tr>
<th>NO</th>
<th>BUAH</th>
<th>SAYUR</th>
                     </tr>

                     <?php  for ($no = 1, $i=10, $a=100; $i<=100, $a<=1000  ;
                     $i+=10, $a+=100) { ?>

                     <tr>
<td> <?php echo $no; ?></td>
<td><?php echo $i; ?></td>
<td><?php echo $a; ?></td>
                     </tr>

              <?php $no++; } ?>

              </table>
       </form>
```

Cara Membuat Looping For di dalam Tabel

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

Cara Membuat Tabel dengan perulangan

```
<?php

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

Cara Kerja:

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

PENGAYAAN FOR LOOP

Pelajari cara menggunakan perulangan for di PHP dengan penjelasan dan contoh yang mudah dipahami, cocok untuk pemula dalam pemrograman web.

For loop adalah struktur kontrol yang sangat berguna dalam PHP yang memungkinkan kamu untuk mengeksekusi blok kode berulang kali dengan jumlah tertentu. Instruksi perulangan ini digunakan saat kamu mengetahui sebelumnya berapa kali blok kode harus dijalankan.

Penggunaan For Loop

Perulangan for di PHP memiliki sintaks sebagai berikut:

```
for (inisialisasi; kondisi; increment) {
    // blok kode yang akan diulang
}
```

Inisialisasi: Menetapkan nilai awal untuk variabel loop.
Kondisi: Ekspresi yang dievaluasi sebelum setiap iterasi. Jika kondisi bernilai TRUE, blok kode di dalam loop akan dijalankan. Jika FALSE, loop akan berakhir.
Increment: Digunakan untuk meningkatkan (atau kadang-kadang mengurangi) nilai variabel loop.

Contoh For Loop

Misalkan kita ingin menampilkan angka dari 1 sampai 5. Kita bisa menggunakan for loop seperti ini:

```
for ($i = 1; $i <= 5; $i++) {
    echo $i . " ";
}
```

Output:
1 2 3 4 5
Dalam contoh di atas, $i = 1 adalah inisialisasi, $i <= 5 adalah kondisi yang menyatakan loop akan terus berjalan selama $i kurang dari atau sama dengan 5, dan $i++ adalah increment yang akan menambahkan nilai $i setiap kali loop dijalankan.

Penggunaan Pada Array

For loop juga bisa digunakan untuk mengiterasi elemen-elemen dalam sebuah array. Sebagai contoh:

```
$arr = array("Apple", "Banana", "Cherry");

for ($i = 0; $i < count($arr); $i++) {
    echo $arr[$i] . " ";
}
```

Output:
Apple Banana Cherry
Di sini, count($arr) digunakan untuk mendapatkan jumlah elemen dalam array sehingga kita bisa mengulang hingga elemen terakhir.

For Loop dengan Step yang Berbeda

Kamu juga bisa mengatur increment untuk langkah yang berbeda-beda. Misalnya, jika kamu ingin menampilkan setiap angka genap antara 1 sampai 10:

```
for ($i = 2; $i <= 10; $i += 2) {
    echo $i . " ";
}
```

Output:
2 4 6 8 10
Perulangan for di PHP sangat fleksibel dan dapat digunakan untuk berbagai kasus. Dengan memahami konsep dan sintaks dasarnya, kamu bisa membuat kode yang efisien dan mudah dibaca. Selamat mencoba!

WHILE LOOP

Pelajari cara menggunakan while loop dalam PHP dengan penjelasan dan contoh kode yang mudah dipahami.

While loop adalah struktur kendali yang umum digunakan dalam pemrograman untuk mengeksekusi blok kode berulang kali selama suatu kondisi bernilai true. Dalam PHP, while loop memiliki sintaks

yang sederhana dan mudah diimplementasikan. Artikel ini akan memandu kamu untuk memahami dan menggunakan while loop dalam PHP dengan beberapa contoh praktis.

While Loop dalam PHP Sebuah while loop akan terus menjalankan sebuah blok kode selama kondisi yang ditentukan masih benar. Struktur dasar dari while loop dalam PHP ditunjukkan seperti berikut:

```
while (kondisi) {
    // blok kode yang akan diulang
}
```

Kunci dari loop ini adalah kondisi; jika kondisi berhenti menjadi true, maka loop akan berhenti. Apabila kondisi tidak pernah berubah menjadi false, maka kamu akan mendapatkan apa yang dikenal sebagai loop tak terbatas, yang akan terus berjalan selamanya.

Contoh Penggunaan while Loop

Berikut adalah contoh sederhana penggunaan while loop dalam PHP:

```
<?php
$i = 1;

while ($i <= 5) {
    echo "Nomor: $i <br>";
    $i++;
}
?>
```

Dalam contoh ini, variabel $i diinisialisasi dengan nilai 1. Kemudian, while loop memeriksa apakah $i kurang dari atau sama dengan 5. Jika true, blok kode di dalam loop akan dijalankan, mencetak nomor saat ini serta menambahkan nilai $i dengan 1. Proses ini berulang sampai kondisi menjadi false, yaitu ketika $i lebih besar dari 5.

Menghindari Loop Tak Terbatas

Untuk menghindari terjadinya loop tak terbatas, pastikan bahwa kondisi loop pada suatu titik akan bernilai false. Berikut adalah beberapa cara untuk mencegah loop tak terbatas:

Pastikan bahwa ada perubahan nilai dalam variabel yang menjadi kondisi, seperti menggunakan increment ($i++)
Gunakan kondisi yang pasti akan berubah seiring dengan iterasi loop, misalnya mengecek nilai dari sebuah variabel yang terus berkurang atau bertambah
Pertimbangkan penggunaan break untuk keluar dari loop jika ada kondisi tertentu yang terpenuhi sebelum kondisi utama loop menjadi false

Penanganan Array dengan while Loop

While loop juga bisa digunakan untuk menangani elemen dalam array. Contoh:

```
<?php
$fruits = ["apple", "banana", "cherry"];
$i = 0;

while ($i < count($fruits)) {
    echo $fruits[$i] . "<br>";
    $i++;
}
?>
```

Dalam contoh di atas, while loop digunakan untuk menampilkan tiap elemen dalam array $fruits. Loop akan berjalan selama nilai $i lebih kecil dari jumlah elemen dalam array.

While loop merupakan alat yang penting dan sering digunakan dalam PHP. Dengan memahami cara kerjanya dan bagaimana mengimplementasikannya dengan benar, kamu dapat menghindari masalah umum seperti loop tak terbatas dan menggunakan loop ini untuk mempermudah proses iterasi dalam programmu.

DO WHILE LOOP

Pelajari bagaimana menggunakan do while loop dalam PHP dengan tutorial langkah demi langkah yang mudah dipahami.

Looping adalah konsep penting dalam pemrograman, dan PHP menawarkan beberapa cara untuk melakukan loop. Salah satu struktur loop yang sering digunakan adalah do while. Loop do while di

PHP sangat berguna ketika kamu ingin memastikan bahwa blok kode dijalankan setidaknya sekali, bahkan jika kondisi loop tidak terpenuhi.

Penggunaan Dasar do while
Loop do while menjalankan blok kode sekali, kemudian mengecek kondisi. Jika kondisi bernilai true, maka blok kode akan dijalankan lagi, dan proses ini terus berlanjut sampai kondisi bernilai false.

Sintaks umum dari do while loop adalah sebagai berikut:
```
do {
   // blok kode untuk dijalankan
} while (kondisi);
```
Elemen kunci dari loop ini adalah kondisi yang diperiksa setelah eksekusi blok kode. Ini berarti bahwa blok kode dalam do pasti akan dijalankan setidaknya satu kali, tidak peduli apa kondisi awalnya.

Contoh Sederhana do while Loop
Berikut adalah contoh sederhana penggunaan do while dalam PHP:
```
$hitung = 1;

do {
   echo "Angka: $hitung <br>";
   $hitung++;
} while ($hitung <= 5);
```
Kode di atas akan mencetak angka 1 sampai 5. Walaupun $hitung dimulai dari 1 dan kondisi adalah $hitung <= 5, loop tetap dijalankan karena pemeriksaan kondisi terjadi setelah eksekusi pertama.

Perbedaan dengan while Loop
Kamu mungkin bertanya-tanya apa perbedaan antara do while dan while. Loop while mengecek kondisi sebelum menjalankan blok kode, jadi jika kondisi awalnya false, blok kode tidak akan pernah dijalankan. Sementara pada do while, blok kode dijamin untuk dijalankan setidaknya sekali.

Contoh while Loop
Untuk memahami perbedaan tersebut, mari kita bandingkan dengan loop while:
```
$hitung = 1;

while ($hitung <= 5) {
   echo "Angka: $hitung <br>";
   $hitung++;
}
```
Hasil dari kode di atas akan sama dengan contoh do while. Namun, jika kita mengubah variabel $hitung menjadi lebih besar dari 5 sejak awal, loop while tidak akan mengeksekusi blok kode sama sekali, sementara do while akan mengeksekusinya sekali sebelum berhenti.

Memahami Kondisi
Kondisi dalam loop do while harus diperhatikan secara seksama karena bisa menyebabkan loop yang tidak pernah berakhir jika kondisi selalu bernilai true. Pastikan kondisi yang kamu gunakan pada do while akan menjadi false pada titik tertentu untuk menghentikan loop.

Contoh Kondisi Loop yang Benar
Pastikan kamu mengupdate variabel yang digunakan dalam kondisi, seperti di contoh berikut:

```
$hitung = 1;

do {
   echo "Angka: $hitung <br>";
   $hitung++;
} while ($hitung <= 5);
```
Dalam kasus ini, $hitung akan terus bertambah hingga tidak lagi memenuhi kondisi $hitung <= 5 dan loop akan berhenti.

Menggunakan loop do while bisa sangat praktis dalam berbagai situasi, terutama ketika kamu membutuhkan setidaknya satu eksekusi dari blok kode, tanpa memandang kondisi awal. Pahami dengan baik cara kerjanya dan implementasikan dengan benar dalam kode PHP kamu.

FOREACH LOOP
Pelajari cara menggunakan foreach loop dalam PHP untuk mengulangi elemen-elemen dalam array dengan tutorial langkah demi langkah ini.

Foreach loop adalah salah satu struktur kontrol yang paling banyak digunakan dalam PHP untuk mengulangi elemen-elemen dalam array. Melalui foreach, kamu bisa mengakses setiap nilai dalam array tanpa perlu mengetahui jumlah elemen yang ada di dalamnya.

Penggunaan Foreach Loop
Foreach loop digunakan untuk menjalankan blok kode untuk setiap elemen dalam array. Ini menjadikannya pilihan yang efisien dan sederhana ketika berurusan dengan array yang berisi kumpulan data.

Sintaks dasar dari foreach loop adalah sebagai berikut:
```
foreach ($array as $value) {
    // Blok kode yang akan dijalankan untuk setiap elemen
}
```
Di dalam foreach loop, variabel $array adalah array yang ingin kamu ulangi, dan $value adalah nama variabel yang akan memegang nilai dari setiap elemen array saat loop sedang berlangsung.

Foreach Loop dengan Key
Kamu juga bisa mendapatkan kunci (key) dari setiap elemen dalam array selama proses iterasi. Ini sangat berguna ketika kamu perlu menggunakan key tersebut dalam operasi.

Berikut adalah sintaks menggunakan key dalam foreach:
```
foreach ($array as $key => $value) {
    // Blok kode dengan $key dan $value
}
```

Contoh Sederhana
Berikut adalah contoh penggunaan foreach loop untuk menampilkan setiap nilai dari array:
```
$buah = array("apel", "pisang", "ceri");

foreach ($buah as $val) {
    echo $val . "<br>";
}
```
Output yang akan dihasilkan dari kode di atas adalah:
apel
pisang
ceri
Contoh dengan Key
Mari kita menggunakan foreach dengan mendapatkan key dari array:

```
$harga = array("apel" => 10000, "pisang" => 5000, "ceri" => 15000);

foreach ($harga as $kunci => $nilai) {
    echo "Harga dari " . $kunci . " adalah " . $nilai . "<br>";
}
```

Contoh di atas akan mengeluarkan output berikut:
Harga dari apel adalah 10000
Harga dari pisang adalah 5000
Harga dari ceri adalah 15000

Penanganan Array Multidimensi
Kadang kita perlu bekerja dengan array yang lebih kompleks, seperti array multidimensi. Dengan mengkombinasikan loop, kamu bisa mengakses elemen-elemen yang terletak lebih dalam.

Contoh penggunaan foreach pada array multidimensi:

```
$matriks = array(
   array(1, 2, 3),
   array(4, 5, 6),
   array(7, 8, 9)
);

foreach ($matriks as $baris) {
    foreach ($baris as $nilai) {
        echo $nilai . " ";
    }
    echo "<br>";
}
```

Output:
1 2 3
4 5 6
7 8 9
Itulah dasar-dasar menggunakan foreach loop di PHP. Dengan memahami konsep ini, kamu akan lebih mudah menangani data dalam array dan mengembangkan aplikasi yang lebih dinamis.

BREAK DAN CONTINUE
Belajar penggunaan perintah break dan continue dalam pengulangan di PHP untuk mengontrol loop secara efektif.

Mengontrol Loop dengan break dan continue di PHP
Dalam pemrograman, terutama saat bekerja dengan loop, terkadang kita perlu menghentikan pengulangan atau melompati iterasi tertentu. PHP menyediakan dua perintah penting, break dan continue, untuk mengatasi hal tersebut.

Penggunaan break
Perintah break digunakan untuk menghentikan eksekusi loop sepenuhnya. Ketika PHP menemui break, ia akan segera keluar dari struktur loop, baik itu for, foreach, while, atau do-while.
Contoh:

```
for ($i = 0; $i < 10; $i++) {
    if ($i === 5) {
        break;
    }
    echo $i . " ";
}
// Output: 0 1 2 3 4
```

Dalam contoh di atas, loop akan berhenti ketika $i bernilai 5.

Penggunaan continue
Berbeda dengan break, perintah continue tidak menghentikan loop sepenuhnya. Sebaliknya, continue akan menghentikan iterasi saat ini dan melanjutkan ke iterasi berikutnya dari loop.
Contoh:

```
for ($i = 0; $i < 10; $i++) {
    if ($i % 2 === 0) {
        continue;
    }
    echo $i . " ";
}
// Output: 1 3 5 7 9
```

Pada contoh di atas, continue digunakan untuk melewatkan iterasi loop bila $i adalah bilangan genap.

break pada Switch
break juga sering digunakan dalam switch untuk menghentikan eksekusi kode setelah sebuah case telah dipenuhi.

Contoh:

```
$buah = "apel";

switch ($buah) {
    case "mangga":
        echo "Ini mangga";
        break;
    case "apel":
        echo "Ini apel";
        // Break disini menghentikan switch
        break;
    case "pisang":
        echo "Ini pisang";
        break;
    default:
        echo "Buah tidak dikenal";
}
```

Bersarang Dengan break dan continue

Ketika bekerja dengan loop bersarang, break dan continue juga dapat mengontrol loop di luar loop saat ini. Namun, kamu perlu memberikan level sebagai argumen kepada mereka.

Contoh dengan continue:

```
for ($i = 0; $i < 5; $i++) {
    echo "i: $i ";
    for ($j = 0; $j < 5; $j++) {
        if ($j == 2) {
            continue 2; // Melanjutkan pada level loop kedua
        }
        echo "j: $j ";
    }
    echo "\n";
}
```

Dalam contoh di atas, ketika $j sama dengan 2, continue akan melewatkan sisa kode di loop luar dan langsung melanjutkan ke iterasi selanjutnya dari loop tersebut.

Perlu diingat, untuk pemula, kesalahan dalam penggunaan break dan continue dapat membingungkan dan seringkali menyebabkan bug. Oleh karena itu, gunakanlah perintah ini dengan bijak dan selalu tes loop kamu untuk memastikan mereka bekerja sesuai yang diharapkan.

MATERI PERTEMUAN 9
Array Dalam Pemrograman PHP

TAHUKAH KAMU...?

Pada kesempatan ini, kita akan membahas:

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

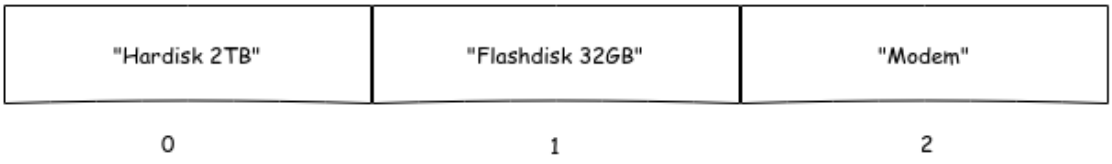
Bayangkan sekarang kita sedang membuat aplikasi web, lalu ingin menampilkan daftar nama-nama produk.
Bisa saja kita buat seperti ini:

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

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

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

1. Apa itu Array?
Array adalah salah satu struktur data yang berisi sekumpulan data dan memiliki indeks. Indeks digunakan untuk mengakses nilai array (**Array Indexed**).
Indeks array selalu dimulai dari nol (**0**).

Contoh:



Jadi, apabila kita ingin menampilkan “Hardisk 2TB”, maka kita harus mengampil indeks yang ke-0.

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

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

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

- **arsort()**. Pengurutan berdasarkan value secara descending.

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

2. Membuat Array di PHP

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

Contoh:

1) Membuat array kosong
```
$buah = array();
$hobi = [];
```
2) Membuat array sekaligus mengisinya
```
$minuman = array("Kopi", "Teh", "Jus Jeruk");
$makanan = ["Nasi Goreng", "Soto", "Bubur"];
```
3) Membuat array dengan mengisi indeks tertentu
```
$anggota[1] = "Dian";
$anggota[2] = "Muhar";
$anggota[0] = "Ahmadi";
```

Cukup mudah bukan.

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

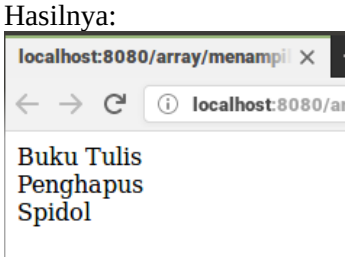
Contoh:
```
<?php

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

3. Menampilkan isi Array

Untuk menampilkan isi array, kita bisa mengaksesnya melalui indeks.

Contoh:
```
<?php
// membuat array
$barang = ["Buku Tulis", "Penghapus", "Spidol"];

// menampilkan isi array
echo $barang[0]."<br>";
echo $barang[1]."<br>";
echo $barang[2]."<br>";
```

Hasilnya:



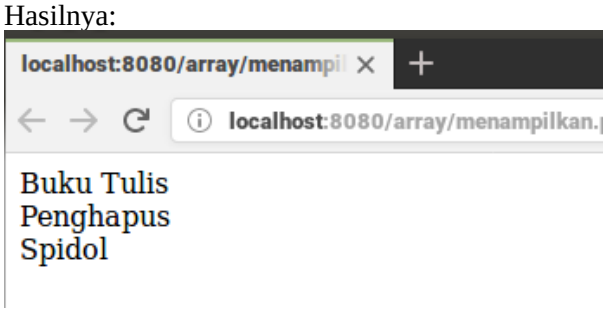
Tapi cara ini kurang efektif, karen akita mencetak satu per satu. Nanti kalau datanya ada 1000, berarti harus ngetik perintah **echo** sebanyak 1000.
Lalu bagaimana kah?

Biasanya kita menggunakan perulangan.
Contoh:

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

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

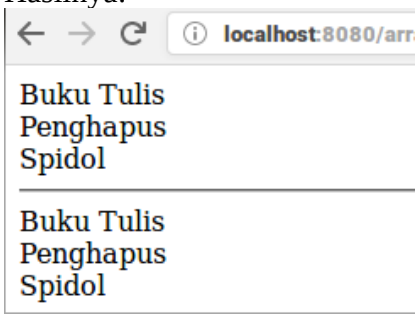
Hasilnya:



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

```
<?php

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

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

Hasilnya:

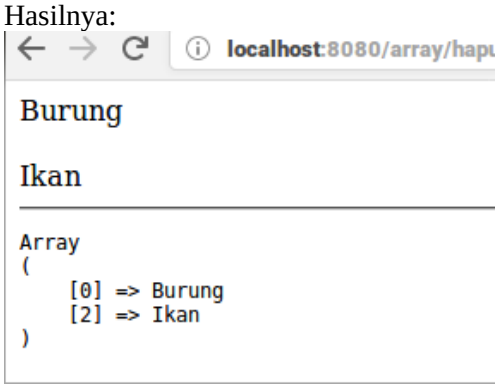


4. Menghapus isi Array
Untuk menghapus isi array, kita bisa menggunakan fungsi **unset()**. Fungsi ini juga dapat digunakan untuk menghapus variabel.
Contoh:

```
<?php
```

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

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

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

echo "<hr>";

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

Hasilnya:



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

5. Menambahkan isi Array

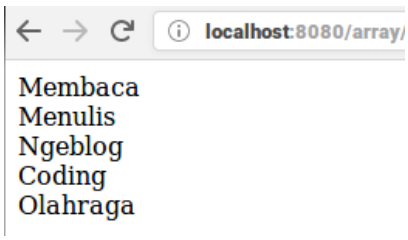
Ada dua cara yang bisa dilakukan untuk menambah isi array:

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

Mari kita coba kedua-duanya.

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

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

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

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

Hasilnya:



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

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

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

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

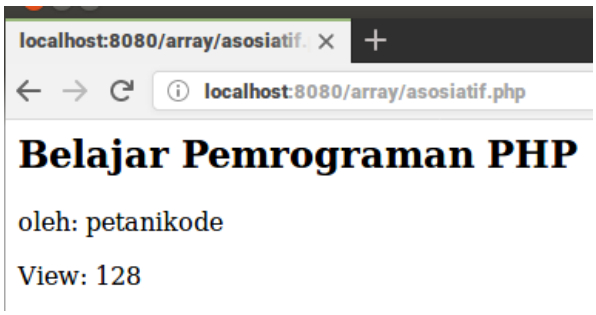
Hasilnya:

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

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

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

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

Hasilnya:



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

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

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

Hasilnya:

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

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

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

Key, Value, Indexed dan Associative Array

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

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

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

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

bentuk diatas sama dengan:

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

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

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

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

I. Cara Penulisan Foreach Pada PHP

Setelah kita faham tentang key dan value pada array, kita

Pada PHP foreach dapat ditulis dalam dua cara yaitu:

1Mengabaikan nilai key

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

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

Penjelasan:

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

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

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

Hasil yang kita peroleh:

```
Januari
Februari
Maret
```

2. Menyertakan nilai key

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

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

Penjelasan:

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

Contoh penggunaan pada indexed array:

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

Hasil yang kita peroleh:

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

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

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

Hasil yang kita peroleh:

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

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

3. Foreach didalam foreach (nested foreach)

Pada multidmensional array, kita akan sering membuat perulangan foreach didalam foreach.
Hati hati jika membuat perulangan dengan model seperti ini, karena sifat variabel dapat berubah ubah, maka, nilai variabel dari $key dan $value juga rawan berubah.
Oleh karena itu, sebisa mungkin memberi nama variabel untuk $key dan $value sesuai dengan isi datanya, sehingga memudahkan kita untuk membaca alur dari perulangan.

Contoh:

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

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

II. Contoh Penggunaan Foreach Pada PHP

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

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

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

$opsi_bulan = '<select name="bulan">';
foreach ($bulan as $key => $value) {
	$opsi_bulan .= '<option value="' . $key . '">' . $value . '</option>'
. "\r\n";
}
$opsi_bulan .= '</select>';

echo $opsi_bulan;
```

Kode HTML yang kita peroleh:

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

Output:

Contoh lain adalah untuk membuat tabel HTML

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

```
echo $tabel;
```
Hasil:
```
<table>
      <tr>
            <th>No</th>
            <th>Bulan</th>
            <th>Penjualan</th>
      </tr>
      <tr>
            <td>1</td>
            <td>Januari</td>
            <td>5.500</td>
      </tr>
      <tr>
            <td>2</td>
            <td>Februari</td>
            <td>7.500</td>
      </tr>
      <tr>
            <td>3</td>
            <td>Maret</td>
            <td>11.500</td>
      </tr>
      <tr>
            <td>4</td>
            <td>April</td>
            <td>8.800</td>
      </tr>
      <tr>
            <td>5</td>
            <td>Mei</td>
            <td>7.500</td>
      </tr>
</table>
```

7. Array Multi Dimensi

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

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

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

Masi kita coba contoh yang lain:
```
<?php
// membuat array 2 dimensi yang berisi array asosiatif
$artikel = [
    [
        "judul" => "Belajar PHP & MySQL untuk Pemula",
        "penulis" => "ahmadimuslim"
    ],
    [
        "judul" => "Tutorial PHP dari Nol hingga Mahir",
        "penulis" => "ahmadimuslim"
    ],
```

```
    [
        "judul" => "Membuat Aplikasi Web dengan PHP",
        "penulis" => "ahmadimuslim"
    ]
];

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

Hasilnya:

PENGAYAAN

Contoh Program PHP Menggunakan Array

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

Array 1 Dimensi PHP

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

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

Array 2 Dimensi Pada PHP

Array atau larik dua dimensi ini memiliki dua buah index. Berbeda dengan array satu dimensi yang hanya memiliki satu index. Dalam konteks ini, array dua dimensi memiliki dua index dimana index pertama melambangkan baris, sedangkan index kedua melambangkan kolom. Sama seperti bahasa pemrograman lain. Pada pembuatan program, array dua dimensi adalah array yang paling sering digunakan karena dapat menyelesaikan masalah lebih banyak dibanding array lainnya.

```
<?php
$nilai=array(
        array(90,65,83),
        array(90,78,97),
        array(78,90,78)
```

```
      );
echo"output array <br>";
echo $nilai[0] [0]." ".$nilai[0] [1]." ".$nilai[0] [2]."<br>";
echo $nilai[1] [0]." ".$nilai[1] [1]." ".$nilai[1] [2]."<br>";
echo $nilai[2] [0]." ".$nilai[2] [1]." ".$nilai[2] [2]."<br>";
?>
```

Array 3 Dimensi PHP

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

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

PENALARAN

Menggabungkan Array dan Menampilkan dalam tabel HTML

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

$totalArray = count($harga);

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

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

Hasilnya

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

Contoh Kasus Array Asosiatif

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

Pertama kita buat array asosiatifnya seperti berikut :

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

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

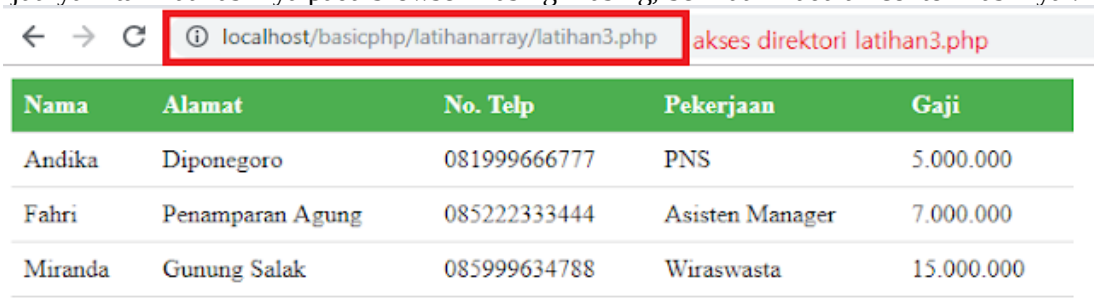
```php
<head>
    <title>Array Asosiatif</title>
</head>
<body>
    <table>
        <tr>
            <th>Nama</th>
            <th>Alamat</th>
            <th>No. Telp</th>
            <th>Pekerjaan</th>
            <th>Gaji</th>
        </tr>
        <?php foreach ($customers as $customer) { ?>
            <tr>
                <td><?php echo $customer["Nama"]; ?></td>
                <td><?php echo $customer["Alamat"]; ?></td>
                <td><?php echo $customer["No.Telp"]; ?></td>
                <td><?php echo $customer["Pekerjaan"]; ?></td>
                <td><?php echo $customer["Gaji"]; ?></td>
            </tr>
        <?php } ?>
    </table>
</body>
```

header tabel

perulangan untuk mengakses setiap elemen di array $customers

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

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



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

PENGAYAAN ARRAY PHP

Pelajari dasar-dasar array di PHP termasuk cara membuat, mengakses, dan memodifikasi array untuk meningkatkan kemampuan pemrograman kamu.

Mengenal Array PHP

PHP adalah bahasa skrip yang populer digunakan dalam pengembangan web. Salah satu fitur fundamental dari PHP yang sering digunakan adalah array. Array memungkinkan kamu menyimpan sejumlah data dalam satu variabel, yang memudahkan pengelolaan dan akses data.

Apa Itu Array?

Array adalah struktur data yang dapat menyimpan banyak nilai di bawah satu nama variabel, dengan setiap nilai dapat diakses melalui indeks atau kunci. Di PHP, ada tiga jenis array:

Indexed arrays - Array dengan indeks numerik.
Associative arrays - Array dengan kunci sebagai string untuk menyimpan data.
Multidimensional arrays - Array yang mengandung satu atau lebih array.

Membuat Array

Untuk membuat array di PHP, kamu bisa menggunakan fungsi array() atau menggunakan tanda kurung siku []. Berikut adalah contoh cara membuat array:

```
// Menggunakan fungsi array()
$buah = array('apel', 'jeruk', 'pisang');

// Menggunakan tanda kurung siku []
$warna = ['merah', 'hijau', 'biru'];
```

Mengakses Data Array

Kamu bisa mengakses data dalam array dengan menunjukkan indeksnya, dimulai dari 0 untuk array berindeks dan dengan nama kunci untuk array asosiatif.

```
echo $buah[1]; // akan menampilkan 'jeruk'
echo $warna[0]; // akan menampilkan 'merah'
```

Untuk array asosiatif, kamu harus menggunakan kunci sebagai indeks:

```
$pengguna = ['nama' => 'Ali', 'umur' => 25, 'pekerjaan' => 'Pengembang Web'];
echo $pengguna['nama']; // akan menampilkan 'Ali'
```

Memodifikasi Array

Kamu bisa mengubah nilai dari elemen array dengan menetapkan nilai baru ke elemen tersebut dengan indeksnya.

```
$buah[1] = 'mangga';
echo $buah[1]; // akan menampilkan 'mangga'
```

Kamu juga bisa menambahkan elemen baru ke array:

```
$buah[] = 'anggur'; // akan menambahkan 'anggur' ke akhir array
```

Menghapus Elemennya

Untuk menghapus elemen, kamu bisa menggunakan fungsi unset().

```
unset($buah[1]); // akan menghapus elemen dengan indeks 1 dari array $buah
```

Menghitung Elemen Array

Fungsi count() digunakan untuk mengetahui berapa banyak elemen yang ada dalam array:

```
echo count($buah); // akan menampilkan jumlah elemen dalam array $buah
```

Looping Melalui Array

Kamu bisa menggunakan foreach untuk beriterasi melalui setiap elemen dalam array.

```
foreach ($warna as $w) {
    echo $w . '<br>';
}
```

Untuk array asosiatif, kamu juga bisa mengakses kuncinya:

```
foreach ($pengguna as $kunci => $nilai) {
    echo $kunci . ': ' . $nilai . '<br>';
}
```

Dengan mengenal array PHP, kamu dapat memanfaatkan fitur ini untuk menyimpan dan mengelola data dengan lebih efisien dalam aplikasi webmu. Mulai praktikkan dengan berbagai contoh di atas untuk meningkatkan pemahaman dan keterampilan pemrograman kamu di PHP.

MULTIDIMENSIONAL ARRAY

Pelajari penggunaan array multidimensi di PHP dengan penjelasan sederhana dan contoh kode untuk memudahkan pemahaman tentang struktur data yang kompleks ini.

Array merupakan struktur data penting yang memungkinkan kamu menyimpan dan mengelola kumpulan nilai dalam satu variabel. Dalam PHP, array bisa lebih kompleks lagi dengan adanya array multidimensi. Array ini memungkinkan kamu untuk menata data dalam bentuk tabel, atau bahkan struktur yang lebih rumit.

Mengenal Array Multidimensi di PHP

Array multidimensi adalah array yang berisi satu atau lebih array di dalamnya. Kamu bisa membayangkannya sebagai tabel dengan baris dan kolom, di mana setiap elemen array bisa lagi berisi array lain.

Contohnya, jika kamu ingin menyimpan informasi tentang film, di mana tiap film terdiri dari judul, tahun, dan direktur, kamu bisa menggunakan array multidimensi seperti berikut:

```
$films = array(
    array("The Shawshank Redemption", 1994, "Frank Darabont"),
    array("The Godfather", 1972, "Francis Ford Coppola"),
    array("The Dark Knight", 2008, "Christopher Nolan")
);
```

Mengakses Nilai dalam Array Multidimensi

Untuk mengakses nilai dalam array multidimensi, kamu perlu menggunakan indeks yang sesuai. Indeks ini umumnya merupakan angka yang menunjukkan posisi dari elemen yang kamu ingin akses.

Misalnya, jika kamu ingin mencetak judul film pertama dari contoh array di atas, kamu akan menuliskan:

```
echo $films[0][0]; // Outputs: The Shawshank Redemption
```

Menambahkan Nilai ke Dalam Array Multidimensi

Untuk menambahkan nilai baru ke dalam array multidimensi, kamu bisa menggunakan fungsi array_push() atau hanya menggunakan kurung siku []. Namun, pastikan kamu mengetahui indeks array yang benar untuk menempatkan data yang baru.

Misalnya, untuk menambahkan film baru ke dalam daftar:

```
$newFilm = array("Inception", 2010, "Christopher Nolan");
$films[] = $newFilm; // Menambahkan di akhir array films
```

Melakukan Loops Pada Array Multidimensi

Ketika bekerja dengan array multidimensi, sangat umum untuk menggunakan perulangan, seperti foreach atau for, untuk mengelola atau menampilkan data.

Contohnya, untuk menampilkan informasi semua film dalam array:

```
foreach ($films as $filmDetails) {
    echo "Judul: " . $filmDetails[0] . "<br>";
    echo "Tahun: " . $filmDetails[1] . "<br>";
    echo "Sutradara: " . $filmDetails[2] . "<br><br>";
}
```

Fungsi Array dalam Array Multidimensi

PHP menawarkan berbagai fungsi untuk bekerja dengan array, dan banyak di antaranya juga bisa digunakan untuk array multidimensi. Sebagai contoh, kamu bisa menggunakan count() untuk mendapatkan jumlah elemen atau sort() untuk mengurutkan elemen dalam array.

Sebagai contoh penggunaan count() pada array multidimensi:

```
$totalFilms = count($films);
echo "Jumlah film: " . $totalFilms; // Menampilkan jumlah film
```

Dengan memahami konsep array multidimensi di PHP, kamu bisa menyusun dan mengelola data yang kompleks dengan lebih efisien. Eksperimen dengan array ini dan fungsi-fungsi yang tersedia akan membantu kamu menjadi programmer PHP yang lebih cakap dalam menyelesaikan berbagai masalah pemrograman.

ASSOCIATIVE ARRAY

Pelajari cara menggunakan associative array di PHP untuk mengelola data dengan pasangan kunci-nilai.

PHP menawarkan berbagai tipe data array, dan salah satu yang paling berguna adalah associative array. Associative array memungkinkan kamu untuk menggunakan kunci yang kamu tentukan sendiri untuk setiap nilai yang disimpan. Ini memudahkan dalam mengakses dan mengelola data pada aplikasi PHP.

Apa itu Associative Array?

Associative array adalah tipe data di PHP yang mengindeks nilai menggunakan kunci yang tidak harus berupa bilangan bulat. Kunci dalam associative array biasanya berupa string, yang memungkinkan kode menjadi lebih membaca dan memudahkan pencarian nilai berdasarkan penanda tertentu.

Contoh sederhana associative array:

```
$orang = array(
   "nama" => "Ayu",
   "umur" => 25,
   "kota" => "Jakarta"
);
```

Cara Membuat Associative Array

Untuk membuat associative array, kamu bisa menggunakan fungsi array() dengan pasangan kunci dan nilai, atau dengan cara langsung menetapkan nilai ke kunci dalam array yang sudah ada.

```
// Menggunakan fungsi array()
$buah = array(
   "apel" => "hijau",
   "strawberry" => "merah",
   "pisang" => "kuning"
);

// Menetapkan langsung
$mobil['Toyota'] = "Corolla";
$mobil['Honda'] = "Civic";
$mobil['Suzuki'] = "Swift";
```

Mengakses Data dari Associative Array

Untuk mendapatkan nilai dari associative array, kamu bisa menggunakan kunci yang bersangkutan.

```
echo $buah["apel"]; // Output: hijau
echo $mobil['Honda']; // Output: Civic
```

Menambahkan dan Mengubah Data

Associative array fleksibel, kamu bisa menambahkan atau mengubah data dengan mudah.

```
// Menambahkan data baru
$buah["mangga"] = "kuning";

// Mengubah data yang ada
$buah["apel"] = "merah";
```

Looping Melalui Associative Array

Kamu bisa melakukan looping melalui associative array dengan menggunakan foreach. Ini memungkinkan kamu memproses setiap pasangan kunci-nilai.

```
foreach ($buah as $kunci => $nilai) {
   echo "Kunci: $kunci, Nilai: $nilai<br>";
}
```

Beberapa Fungsi Bermanfaat untuk Associative Arrays

PHP menyediakan berbagai fungsi yang dapat membantu kamu dalam mengelola associative arrays.

Menghitung Elemen

```
echo count($buah); // Menampilkan jumlah elemen di dalam array buah
```

Memeriksa Kunci Tertentu

```
if (array_key_exists("apel", $buah)) {
   echo "Buah apel tersedia!";
}
```

Mendapatkan Semua Kunci atau Nilai

```
// Semua kunci
$kunci = array_keys($buah);

// Semua nilai
$nilai = array_values($buah);
```

Mengurutkan Associative Array

Kamu dapat mengurutkan associative array baik berdasarkan kunci maupun nilai.

```
// Berdasarkan kunci
ksort($buah);

// Berdasarkan nilai
asort($buah);
```

Associative array di PHP sangat berguna untuk mengelola data yang membutuhkan identifikasi yang jelas antara kunci dan nilai. Dengan fitur ini, PHP memudahkan pengembang untuk menulis kode yang rapi dan mudah dipelihara.

METODE METODE ARRAY

Pelajari berbagai metode array pada PHP yang paling umum digunakan, termasuk contoh penggunaan dan penjelasan sederhana untuk pemula.

PHP adalah bahasa pemrograman server-side yang populer dan sering digunakan untuk pengembangan web. Salah satu fitur yang kuat dalam PHP adalah manipulasi array. Array adalah struktur data yang memungkinkan kamu menyimpan banyak nilai dalam satu variabel. Berikut ini adalah beberapa metode array yang sering digunakan dalam PHP untuk mempermudah pengelolaan data.

array_push()

Metode array_push() digunakan untuk menambahkan satu atau lebih elemen ke akhir array. Elemen yang ditambahkan akan menjadi elemen terakhir dalam array.

```
$buah = ["apel", "jeruk"];
array_push($buah, "pisang", "mangga");
print_r($buah);
```

Hasilnya akan menjadi:

```
Array
(
    [0] => apel
    [1] => jeruk
    [2] => pisang
    [3] => mangga
)
```

array_pop()

array_pop() digunakan untuk menghapus elemen terakhir dari array. Fungsi ini akan mengurangi panjang array dan mengembalikan nilai yang dihapus.

```
$buah = ["apel", "jeruk", "pisang"];
$buahTerakhir = array_pop($buah);
echo $buahTerakhir;
```

Output:

```
pisang
```

array_shift()

Fungsi array_shift() akan menghapus elemen pertama dari array dan menggeser semua elemen berikutnya ke posisi sebelumnya, sehingga indeks dari setiap elemen akan berkurang satu.

```
$buah = ["apel", "jeruk", "pisang"];
$buahPertama = array_shift($buah);
echo $buahPertama;
```

Output:

```
apel
```

array_unshift()

Kebalikan dari array_shift(), array_unshift() menambahkan satu atau lebih elemen ke awal array dan mengembalikan jumlah elemen baru dalam array.

```
$buah = ["apel", "jeruk"];
array_unshift($buah, "pisang", "mangga");
print_r($buah);
```

Output:

```
Array
(
    [0] => pisang
    [1] => mangga
    [2] => apel
    [3] => jeruk
)
```

array_merge()

Fungsi array_merge() menggabungkan dua atau lebih array menjadi satu array. Jika ada elemen dengan kunci yang sama, elemen dari array terakhir yang disediakan akan mengambil alih.

```
$array1 = ["warna" => "merah", 2, 4];
$array2 = ["a", "b", "warna" => "hijau", "bentuk" => "trapesium", 4];
$result = array_merge($array1, $array2);
print_r($result);
```

Output:

```
Array
(
    [warna] => hijau
    [0] => 2
    [1] => 4
    [2] => a
```

```
    [3] => b
    [bentuk] => trapesium
    [4] => 4
)
```

array_slice()

array_slice() digunakan untuk mendapatkan sebuah sub-array dari array. Kamu bisa menentukan indeks awal dan panjang sub-array yang diinginkan.

```
$buah = ["apel", "jeruk", "pisang", "mangga"];
$subBuah = array_slice($buah, 1, 2);
print_r($subBuah);
```

Output:

```
Array
(
    [0] => jeruk
    [1] => pisang
)
```

array_splice()

Fungsi array_splice() menghapus elemen tertentu dari array dan bisa juga menggantikannya dengan nilai lain. Fungsi ini sangat berguna untuk manipulasi array yang kompleks.

```
$buah = ["apel", "jeruk", "pisang", "mangga"];
array_splice($buah, 2, 1, "kiwi");
print_r($buah);
```

Output:

```
Array
(
    [0] => apel
    [1] => jeruk
    [2] => kiwi
    [3] => mangga
)
```

Itulah beberapa metode array pada PHP yang paling sering digunakan. Dengan memahami metode-metode ini, kamu akan lebih mudah untuk melakukan operasi terhadap array saat mengembangkan aplikasi web dengan PHP.

CRUD ARRAY

Panduan untuk melakukan operasi CRUD (Create, Read, Update, Delete) pada array dalam PHP dengan langkah-langkah sederhana dan contoh kode yang mudah dipahami.

CRUD adalah singkatan dari Create, Read, Update, dan Delete. Operasi ini sangat dasar dalam pemrograman karena melibatkan manipulasi data. Dalam PHP, CRUD sering digunakan untuk mengelola array. Array adalah struktur data yang bisa menyimpan banyak nilai sekaligus. Berikut adalah cara melakukan operasi CRUD pada array dalam PHP.

Membuat Array (Create)

Untuk membuat array, kamu bisa menggunakan fungsi array() atau menggunakan square brackets []. Berikut contoh cara membuat array:

```
$buah = array("apel", "mangga", "pisang");
// atau
$buah = ["apel", "mangga", "pisang"];
```

Kamu juga bisa membuat array asosiatif, di mana setiap elemen mempunyai key sebagai identifiernya:

```
$umur = array("Ali" => 20, "Budi" => 25, "Citra" => 22);
```

Membaca Array (Read)

Untuk membaca atau mengakses isi dari array, kamu bisa menggunakan key atau index dari elemen tersebut. Pada array numerik, index dimulai dari 0.

```
echo $buah[0]; // menampilkan "apel"
echo $umur["Ali"]; // menampilkan 20
```

Untuk membaca seluruh isi array, loop seperti foreach bisa digunakan:

```
foreach ($buah as $b) {
    echo $b . " "; // menampilkan "apel mangga pisang "
}
```

Memperbarui Array (Update)

Untuk mengubah nilai pada array, kamu cukup menetapkan nilai baru ke key atau index tertentu.

```
$buah[0] = "jeruk"; // mengubah nilai elemen pertama menjadi "jeruk"
$umur["Ali"] = 21; // mengubah umur Ali menjadi 21
```

Untuk menambah elemen baru ke dalam array, kamu bisa menggunakan fungsi array_push() atau langsung membuat key atau index baru.

```
array_push($buah, "durian"); // menambah "durian" ke dalam array $buah
$umur["Dina"] = 18; // menambah key "Dina" dengan nilai 18 ke dalam array $umur
```

Menghapus Array (Delete)

Untuk menghapus elemen dari array, kamu bisa menggunakan fungsi unset().

```
unset($buah[2]); // menghapus elemen dengan index 2 dari array $buah
unset($umur["Budi"]); // menghapus elemen dengan key "Budi" dari array $umur
```

Setelah elemen dihapus, index atau key tidak akan digunakan kembali. Jika kamu ingin menghapus semua elemen array, cukup unset() seluruh array.

```
unset($buah); // menghapus keseluruhan array $buah
```

Melakukan operasi CRUD pada array di PHP cukup mudah dan intuitif. Dengan mengikuti langkah-langkah di atas, kamu bisa melakukan manipulasi data pada array dengan lancar. Ingatlah untuk selalu menggunakan struktur dan fungsi yang tepat untuk memenuhi kebutuhan programmu.