

# **MATERI PEMBEKALAN**

## **JUNIOR WEB PROGRAMMER**



### **HARI KE-3**

<b>J.620100.017.02</b>	<b>Mengimplementasikan pemrograman terstruktur</b>
------------------------	--



JUNIOR WEB  
PROGRAMMER

**J.620100.017.02**

**Mengimplementasikan  
Pemrograman Terstruktur**

# 5

## MENGIMPLEMENTASIKAN PEMROGRAMAN TERSTRUKTUR

### Objektif:

1. Menggunakan tipe data dan Kontrol program
2. Membuat program menggunakan fungsi
3. Membuat program menggunakan array

---

### 1. Tipe Data dan Kontrol Program

Berisi informasi mengenai tipe data dan kontrol program yang ada.

#### 1.1 Tipe Data SQL

Setiap kolom dalam tabel database harus memiliki nama dan tipe data. Pengembang SQL harus memutuskan tipe data apa yang akan disimpan di dalam setiap kolom saat membuat tabel. Tipe data adalah pedoman bagi SQL untuk memahami tipe data apa yang diharapkan di dalam setiap kolom, dan juga mengidentifikasi bagaimana SQL akan berinteraksi dengan data yang disimpan.

##### a. Tipe Data String

Data Type	Description
CHAR(size)	A FIXED length string (can contain letters, numbers, and special characters). The <i>size</i> parameter specifies the column length in characters - can be from 0 to 255. Default is 1
VARCHAR(size)	A VARIABLE length string (can contain letters, numbers, and special characters). The <i>size</i> parameter specifies the maximum column length in characters - can be from 0 to 65535

BINARY(size)	Equal to CHAR(), but stores binary byte strings. The <i>size</i> parameter specifies the column length in bytes. Default is 1
VARBINARY(size)	Equal to VARCHAR(), but stores binary byte strings. The <i>size</i> parameter specifies the maximum column length in bytes.
TINYBLOB	For BLOBs (Binary Large Objects). Max length: 255 bytes
TINYTEXT	Holds a string with a maximum length of 255 characters
TEXT(size)	Holds a string with a maximum length of 65,535 bytes
BLOB(size)	For BLOBs (Binary Large Objects). Holds up to 65,535 bytes of data
MEDIUMTEXT	Holds a string with a maximum length of 16,777,215 characters
MEDIUMBLOB	For BLOBs (Binary Large Objects). Holds up to 16,777,215 bytes of data
LONGTEXT	Holds a string with a maximum length of 4,294,967,295 characters
LOBLOB	For BLOBs (Binary Large Objects). Holds up to 4,294,967,295 bytes of data
ENUM(val1, val2, val3, ...)	A string object that can have only one value, chosen from a list of possible values. You can list up to 65535 values in an ENUM list. If a value is inserted that is not in the list, a blank value will be inserted. The values are sorted in the order you enter them
SET(val1, val2, val3, ...)	A string object that can have 0 or more values, chosen from a list of possible values. You can list up to 64 values in a SET list

## b. Tipe Data Numerik

Data Type	Description
BIT( <i>size</i> )	A bit-value type. The number of bits per value is specified in <i>size</i> . The <i>size</i> parameter can hold a value from 1 to 64. The default value for <i>size</i> is 1.
TINYINT( <i>size</i> )	A very small integer. Signed range is from -128 to 127. Unsigned range is from 0 to 255. The <i>size</i> parameter specifies the maximum display width (which is 255)
BOOL	Zero is considered as false, nonzero values are considered as true.
BOOLEAN	Equal to BOOL
SMALLINT( <i>size</i> )	A small integer. Signed range is from -32768 to 32767. Unsigned range is from 0 to 65535. The <i>size</i> parameter specifies the maximum display width (which is 255)
MEDIUMINT( <i>size</i> )	A medium integer. Signed range is from -8388608 to 8388607. Unsigned range is from 0 to 16777215. The <i>size</i> parameter specifies the maximum display width (which is 255)
INT( <i>size</i> )	A medium integer. Signed range is from -2147483648 to 2147483647. Unsigned range is from 0 to 4294967295. The <i>size</i> parameter specifies the maximum display width (which is 255)
INTEGER( <i>size</i> )	Equal to INT( <i>size</i> )
BIGINT( <i>size</i> )	A large integer. Signed range is from -9223372036854775808 to 9223372036854775807. Unsigned range is from 0 to 18446744073709551615. The <i>size</i> parameter specifies the maximum display width (which is 255)
FLOAT( <i>size</i> , <i>d</i> )	A floating point number. The total number of digits is specified in <i>size</i> . The number of digits after the decimal

	point is specified in the <i>d</i> parameter. This syntax is deprecated in MySQL 8.0.17, and it will be removed in future MySQL versions
FLOAT( <i>p</i> )	A floating point number. MySQL uses the <i>p</i> value to determine whether to use FLOAT or DOUBLE for the resulting data type. If <i>p</i> is from 0 to 24, the data type becomes FLOAT(). If <i>p</i> is from 25 to 53, the data type becomes DOUBLE()
DOUBLE( <i>size</i> , <i>d</i> )	A normal-size floating point number. The total number of digits is specified in <i>size</i> . The number of digits after the decimal point is specified in the <i>d</i> parameter
DOUBLE PRECISION( <i>size</i> , <i>d</i> )	
DECIMAL( <i>size</i> , <i>d</i> )	An exact fixed-point number. The total number of digits is specified in <i>size</i> . The number of digits after the decimal point is specified in the <i>d</i> parameter. The maximum number for <i>size</i> is 65. The maximum number for <i>d</i> is 30. The default value for <i>size</i> is 10. The default value for <i>d</i> is 0.
DEC( <i>size</i> , <i>d</i> )	Equal to DECIMAL( <i>size</i> , <i>d</i> )

### c. Tipe Data Date and Time

Data Type	Description
DATE	A date. Format: YYYY-MM-DD. The supported range is from '1000-01-01' to '9999-12-31'
DATETIME( <i>fsp</i> )	A date and time combination. Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'. Adding DEFAULT and ON UPDATE in the column definition to get automatic initialization and updating to the current date and time

TIMESTAMP( <i>fsp</i> )	A timestamp. <b>TIMESTAMP</b> values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC. Automatic initialization and updating to the current date and time can be specified using <b>DEFAULT CURRENT_TIMESTAMP</b> and <b>ON UPDATE CURRENT_TIMESTAMP</b> in the column definition
TIME( <i>fsp</i> )	A time. Format: hh:mm:ss. The supported range is from '-838:59:59' to '838:59:59'
YEAR	A year in four-digit format. Values allowed in four-digit format: 1901 to 2155, and 0000. MySQL 8.0 does not support year in two-digit format.

## 1.2 Kontrol Program

Terdapat 2 jenis kontrol program yaitu kontrol kondisional dan kontrol perulangan serta break.

### A. Statement Kondisional

Sangat sering ketika menulis kode, ingin melakukan tindakan yang berbeda untuk kondisi yang berbeda, dapat dilakukan dengan menggunakan statement bersyarat dalam kode untuk melakukan ini. Pada PHP memiliki statement bersyarat berikut.

- a. **if statement** - menjalankan beberapa kode jika satu kondisi benar
- b. **if ... else statement** - menjalankan beberapa kode jika kondisinya benar dan kode lain jika kondisi itu salah
- c. **if ... elseif ... else statement** - menjalankan kode yang berbeda untuk lebih dari dua kondisi
- d. **switch statement** - memilih salah satu dari banyak blok kode untuk dijalankan

### a. Statement if

Menjalankan beberapa kode jika satu kondisi benar.

#### Bentuk Umum:

```
if (condition) {  
    code to be executed if condition is true;  
}
```

#### Contoh:

Output "Semoga harimu menyenangkan!" jika waktu saat ini (JAM) kurang dari 20.

```
$t = date("H");  
if ($t < "20") {  
    echo "Have a good day!";  
}  
?>
```



### b. Statement if...else

Menjalankan beberapa kode jika kondisinya benar dan kode lain jika kondisi itu salah.

#### Bentuk Umum:

```
if (condition) {  
    code to be executed if condition is true;  
} else {  
    code to be executed if condition is false;  
}
```

#### Contoh:

Output "Semoga harimu menyenangkan!" jika waktu saat ini kurang dari 20, dan "Selamat malam!" jika tidak.



```

<?php
$t = date("H");
if ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}
?>

```

**c. Statement if...elseif...else**

Menjalankan kode yang berbeda untuk lebih dari dua kondisi.

**Bentuk Umum :**

```

if (condition) {
    code to be executed if this condition is true;
} elseif (condition) {
    code to be executed if first condition is false and this
condition is true;
} else {
    code to be executed if all conditions are false;
}

```

**Contoh:**

Output "Selamat pagi!" jika waktu saat ini kurang dari 10, dan "Semoga harimu menyenangkan!" jika waktu saat ini kurang dari 20. Jika tidak, akan muncul "Selamat malam!".

```

<?php
$t = date("H");
if ($t < "10") {
    echo "Have a good morning!";
} elseif ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}
?>

```

#### d. **Statement switch**

Memilih salah satu dari banyak blok kode untuk dijalankan.

##### **Bentuk Umum:**

```
switch (n) {  
    case label1:  
        code to be executed if n=label1;  
        break;  
    case label2:  
        code to be executed if n=label2;  
        break;  
    case label3:  
        code to be executed if n=label3;  
        break;  
    ...  
    default:  
        code to be executed if n is different from all labels;  
}
```

##### **Contoh:**

```
<?php  
$favcolor = "red";  
switch ($favcolor) {  
    case "red":  
        echo "Your favorite color is red!";  
        break;  
    case "blue":  
        echo "Your favorite color is blue!";  
        break;  
    case "green":  
        echo "Your favorite color is green!";  
        break;  
    default:  
        echo "Your favorite color is neither red, blue, nor green!";  
}  
?>
```

## B. Statement Perulangan PHP

Seringkali ketika menulis kode, ingin blok kode yang sama dijalankan berulang kali dalam beberapa kali. Jadi, alih-alih menambahkan beberapa baris kode yang hampir sama dalam skrip, dapat menggunakan perulangan. Perulangan digunakan untuk mengeksekusi blok kode yang sama berulang kali, selama kondisi tertentu benar. Di PHP memiliki jenis perulangan berikut.

- a. **while** - perulangan melalui blok kode selama kondisi yang ditentukan benar
- b. **do ... while** - perulangan melalui blok kode satu kali, dan kemudian mengulangi perulangan selama kondisi yang ditentukan benar
- c. **for** - perulangan melalui blok kode beberapa kali
- d. **foreach** - perulangan melalui blok kode untuk setiap elemen

### a. Perulangan while

Perulangan melalui blok kode selama kondisi yang ditentukan benar.

#### Bentuk Umum:

```
while (condition is true) {  
    code to be executed;  
}
```

#### Contoh:

Menampilkan angka dari 1 hingga 5.

```
<?php  
$x = 1;  
while($x <= 5) {  
    echo "The number is: $x <br>";  
    $x++;  
}  
?>
```

### b. Perulangan do...while

Perulangan melalui blok kode satu kali, dan kemudian mengulangi perulangan selama kondisi yang ditentukan benar.

#### Bentuk Umum:

```
do {  
    code to be executed;  
} while (condition is true);
```

#### Contoh:

Pertama-tama menetapkan variabel \$ x ke 1 (\$ x = 1). Kemudian, perulangan do while akan menulis beberapa output, dan kemudian menaikkan variabel \$ x dengan 1. Kemudian kondisinya dicentang (adalah \$ x kurang dari, atau sama dengan 5?), Dan perulangan akan terus berjalan selama \$ x kurang dari, atau sama dengan 5.

```
<?php  
$x = 1;  
do {  
    echo "The number is: $x <br>";  
    $x++;  
} while ($x <= 5);  
?>
```

### c. Perulangan for

Perulangan melalui blok kode beberapa kali.

#### Bentuk Umum:

```
for (init counter; test counter; increment counter) {  
    code to be executed for each iteration;  
}
```

#### Contoh:

Menampilkan angka dari 0 hingga 10.

```
<?php
for ($x = 0; $x <= 10; $x++) {
    echo "The number is: $x <br>";
} ?>
```

#### d. Perulangan foreach

Perulangan melalui blok kode untuk setiap elemen dalam array.

##### Bentuk Umum:

```
foreach ($array as $value) {
    code to be executed;
}
```

##### Contoh:

Menampilkan nilai dari array yang diberikan (\$ colors).

```
<?php
$colors = array("red", "green", "blue", "yellow");
foreach ($colors as $value) {
    echo "$value <br>";
}
?>
```

#### C. PHP Break

Break digunakan untuk "melompat keluar" dari statement switch. Statement break juga dapat digunakan untuk melompat keluar dari perulangan.

##### Contoh:

Melompat keluar dari perulangan ketika x sama dengan 4.

```
<?php
for ($x = 0; $x < 10; $x++) {
    if ($x == 4) {
        break;
    }
    echo "The number is: $x <br>";
} ?>
```

## 2. PHP Functions

Kekuatan sebenarnya dari PHP berasal dari fungsinya. PHP memiliki lebih dari 1000 fungsi built-in, dan sebagai tambahan dapat membuat fungsi custom sendiri.

### 2.1 Fungsi Bawaan PHP

PHP memiliki lebih dari 1000 fungsi bawaan yang dapat dipanggil secara langsung, dari dalam skrip, untuk melakukan tugas tertentu. Dapat dilihat pada referensi PHP untuk ikhtisar lengkap tentang fungsi bawaan PHP.

### 2.2 Fungsi yang Ditentukan Pengguna PHP

Selain fungsi PHP bawaan, dimungkinkan untuk membuat fungsi sendiri.

- Fungsi adalah sekumpulan statement yang dapat digunakan berulang kali dalam suatu program.
- Suatu fungsi tidak akan dijalankan secara otomatis saat halaman dimuat.
- Sebuah fungsi akan dieksekusi dengan panggilan ke fungsi tersebut.

#### Bentuk Umum:

```
function functionName() {  
    code to be executed;  
}
```

Catatan : Nama fungsi harus dimulai dengan huruf atau garis bawah.  
Nama fungsi TIDAK peka huruf besar / kecil.

#### Contoh:

Pada contoh berikut, dibuat fungsi bernama "writeMsg ()". Tanda kurung kurawal buka ({} ) menunjukkan awal dari kode fungsi, dan tanda

kurung kurawal tutup (}) menunjukkan akhir dari fungsi tersebut. Fungsi ini menghasilkan "Hello world!". Untuk memanggil fungsi tersebut, cukup tulis namanya diikuti dengan tanda kurung ():

```
<?php
function writeMsg() {
    echo "Hello world!";
}
writeMsg(); // call the function
?>
```

### 2.3 Argumen Fungsi PHP

Informasi dapat diteruskan ke fungsi melalui argumen. Argumen itu seperti variabel. Argumen ditentukan setelah nama fungsi, di dalam tanda kurung. Dapat menambahkan argumen sebanyak yang diinginkan, cukup pisahkan dengan koma.

#### Contoh:

Berikut ini fungsi dengan satu argumen (\$ fname). Ketika fungsi familyName () dipanggil, kami juga memberikan nama (misalnya Jani), dan nama tersebut digunakan di dalam fungsi, yang menghasilkan beberapa nama depan yang berbeda, tetapi memiliki nama belakang yang sama:

```
<?php
function familyName($fname) {
    echo "$fname Refsnes.<br>";
}
familyName("Jani");
familyName("Hege");
familyName("Stale");
familyName("Kai Jim");
familyName("Borge");
?>
```

**Contoh:**

Berikut ini fungsi dengan dua argumen (\$ fname dan \$ year):

```
<?php
function familyName($fname, $year) {
    echo "$fname Refsnes. Born in $year <br>";
}
familyName("Hege", "1975");
familyName("Stale", "1978");
familyName("Kai Jim", "1983");
?>
```

**2.4 PHP adalah Bahasa yang Diketik Secara Longgar**

Tidak perlu memberi tahu PHP tipe data variabelnya. PHP secara otomatis mengaitkan tipe data ke variabel, tergantung pada nilainya. Tipe data tidak disetel dalam arti yang ketat, dapat melakukan hal-hal seperti menambahkan string ke integer tanpa menyebabkan kesalahan. Di PHP 7, deklarasi tipe ditambahkan. Ini memberi opsi untuk menentukan tipe data yang diharapkan saat mendeklarasikan fungsi, dan dengan menambahkan deklarasi **strict**, ini akan memunculkan "Fatal Error" jika tipe datanya tidak cocok.

**Contoh:**

Berikut ini mencoba mengirim angka dan string ke fungsi tanpa menggunakan **strict**.

```
<?php
function addNumbers(int $a, int $b) {
    return $a + $b;
}
echo addNumbers(5, "5 days");
// since strict is NOT enabled "5 days" is changed to int(5), and it
will return 10
?>
```



Untuk menentukan **strict** perlu mengatur **declare(strict\_types = 1);**. Ini harus berada di baris pertama file PHP.

### Contoh:

Berikut ini mencoba mengirim angka dan string ke fungsi, tetapi di sini telah menambahkan deklarasi **strict**.

```
<?php declare(strict_types=1); // strict requirement
function addNumbers(int $a, int $b) {
    return $a + $b;
}
echo addNumbers(5, "5 days");
// since strict is enabled and "5 days" is not an integer, an error will
be thrown
?>
```

Deklarasi **strict** memaksa digunakan dengan cara yang dimaksudkan.

## 2.5 Nilai Argumen Default PHP

Berikut ini menunjukkan cara menggunakan parameter default. Jika memanggil fungsi `setHeight()` tanpa argumen, dibutuhkan nilai default sebagai argumen.

### Contoh:

```
<?php declare(strict_types=1); // strict requirement
function setHeight(int $minheight = 50) {
    echo "The height is : $minheight <br>";
}
setHeight(350);
setHeight(); // will use the default value of 50
setHeight(135);
setHeight(80);
?>
```

## 2.6 Fungsi PHP - Pengembalian nilai

Untuk membiarkan suatu fungsi mengembalikan nilai, gunakan statement return.

### Contoh :

```
<?php declare(strict_types=1); // strict requirement
function sum(int $x, int $y) {
    $z = $x + $y;
    return $z;
}
echo "5 + 10 = " . sum(5, 10) . "<br>";
echo "7 + 13 = " . sum(7, 13) . "<br>";
echo "2 + 4 = " . sum(2, 4);
?>
```

### Deklarasi Jenis Pengembalian PHP

PHP 7 juga mendukung Deklarasi Tipe untuk statement return. Seperti deklarasi tipe untuk argumen fungsi, dengan mengaktifkan persyaratan yang ketat(strict), ini akan memunculkan "Kesalahan Fatal" pada jenis ketidakcocokan. Untuk mendeklarasikan tipe untuk fungsi yang dikembalikan, tambahkan titik dua (:) dan tipe tepat sebelum kurung kurawal buka ({) saat mendeklarasikan fungsi.

### Contoh:

Berikut ini menentukan tipe pengembalian untuk fungsi tersebut.

```
<?php declare(strict_types=1); // strict requirement
function addNumbers(float $a, float $b) : float {
    return $a + $b;
}
echo addNumbers(1.2, 5.2);
?>
```

Bisa menentukan tipe pengembalian yang berbeda, dari tipe argumen, tapi pastikan tipe yang dikembalikan benar.

```
<?php declare(strict_types=1); // strict requirement
function addNumbers(float $a, float $b) : int {
    return (int)($a + $b);
}
echo addNumbers(1.2, 5.2);
?>
```

## 2.7 Meneruskan Argumen dengan Referensi

Dalam PHP, argumen biasanya diteruskan oleh nilai, yang berarti bahwa salinan nilai digunakan dalam fungsi dan variabel yang diteruskan ke fungsi tidak dapat diubah. Ketika argumen fungsi dilewatkan oleh referensi, perubahan pada argumen juga mengubah variabel yang diteruskan.

### Contoh:

Gunakan argumen pass-by-reference untuk memperbarui variabel.

```
<?php
function add_five(&$value) {
    $value += 5;
}
$num = 2;
add_five($num);
echo $num;
?>
```

## 3. PHP Arrays

Sebuah array menyimpan banyak nilai dalam satu variabel.

### Contoh:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . "."; ?>
```

Array adalah variabel khusus, yang dapat menampung lebih dari satu nilai dalam satu waktu. Jika memiliki daftar item (daftar nama mobil, misalnya), menyimpan mobil dalam variabel tunggal akan terlihat seperti ini:

```
$cars1 = "Volvo";  
$cars2 = "BMW";  
$cars3 = "Toyota";
```

Bagaimana jika ingin melewati mobil dan menemukan yang spesifik dan bagaimana jika tidak memiliki 3 mobil, tetapi 300. Solusinya adalah membuat array, sebuah array dapat menampung banyak nilai dengan satu nama, dan dapat mengakses nilai-nilai tersebut dengan merujuk ke nomor indeks.

### Array pada PHP

Pada PHP, fungsi `array ()` digunakan untuk membuat array.

```
array();
```

Pada PHP, ada tiga jenis array, yaitu:

- **Array yang diindeks** - Array dengan indeks numerik
- **Array asosiatif** - Array dengan kunci bernama
- **Array multidimensi** - Array yang berisi satu atau lebih array

### Panjang Array - Fungsi `Count ()`

Fungsi **`count ()`** digunakan untuk mengembalikan panjang (jumlah elemen) dari sebuah array:

```
<? php  
$ cars = array ("Volvo", "BMW", "Toyota");  
hitungannya ($ mobil); ?>
```

### 3.1 Array Terindeks PHP

Ada dua cara untuk membuat array terindeks, pertama Indeks dapat ditetapkan secara otomatis (indeks selalu dimulai dari 0), kedua indeks dapat ditetapkan secara manual.

Otomatis : `$cars = array("Volvo", "BMW", "Toyota");`

Manual : `$cars[0] = "Volvo";`  
`$cars[1] = "BMW";`  
`$cars[2] = "Toyota";`

#### Contoh:

Berikut ini membuat array terindeks bernama \$ cars, menetapkan tiga elemen ke dalamnya, lalu mencetak teks yang berisi nilai array:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

#### Loop Melalui Array Terindeks

Untuk mengulang dan mencetak semua nilai dari array yang diindeks, dapat menggunakan for loop.

#### Contoh :

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
$arlength = count($cars);
for($x = 0; $x < $arlength; $x++) {
    echo $cars[$x];
    echo "<br>";
}
?>
```

### 3.2 Array Asosiatif PHP

Array asosiatif adalah larik yang menggunakan kunci bernama yang ditetapkan padanya. Ada dua cara untuk membuat array asosiatif.

Pertama:

```
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
```

Kedua:

```
$age['Peter'] = "35";
```

```
$age['Ben'] = "37";
```

```
$age['Joe'] = "43";
```

Kunci bernama kemudian dapat digunakan dalam skrip.

**Contoh :**

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
?>
```

### Loop Melalui Array Asosiatif

Untuk mengulang dan mencetak semua nilai dari sebuah array asosiatif, dapat menggunakan foreach loop.

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
foreach($age as $x => $x_value) {
    echo "Key=" . $x . ", Value=" . $x_value;
    echo "<br>";
}
?>
```

### 3.3 Array Multidimensi PHP

Di halaman sebelumnya, telah dijelaskan array yang merupakan satu daftar pasangan kunci/nilai. Terkadang ingin menyimpan nilai dengan lebih dari satu kunci, untuk ini berarti memiliki array multidimensi.

## PHP - Array Multidimensi

Array multidimensi adalah larik yang berisi satu atau lebih larik. PHP mendukung array multidimensi yang memiliki kedalaman dua, tiga, empat, lima, atau lebih. Array yang lebih dari tiga level sulit untuk dikelola bagi kebanyakan orang.

Dimensi array menunjukkan jumlah indeks yang diperlukan untuk memilih elemen.

- Untuk array dua dimensi, memerlukan dua indeks untuk memilih elemen.
- Untuk array tiga dimensi, memerlukan tiga indeks untuk memilih elemen.

## PHP - Array Dua Dimensi

Array dua dimensi adalah larik larik (larik tiga dimensi adalah larik larik larik). Pertama, lihat tabel berikut:

Tabel 1. Tabel Dua Dimensi

Name	Stock	Sold
Volvo	22	18
BMW	15	13
Saab	5	2
Land Rover	17	15

Dapat menyimpan data dari tabel di atas dalam array dua dimensi, seperti berikut.

```
$cars = array (  
    array("Volvo",22,18),  
    array("BMW",15,13),  
    array("Saab",5,2),  
    array("Land Rover",17,15)  
);
```

Sekarang array dua dimensi \$cars berisi empat array, dan ini memiliki dua indeks: baris dan kolom. Untuk mendapatkan akses ke elemen array \$cars harus menunjuk ke dua indeks (baris dan kolom).

#### Contoh :

```
<?php
echo $cars[0][0].": In stock: ".$cars[0][1].", sold: ".$cars[0][2]."<br>";
echo $cars[1][0].": In stock: ".$cars[1][1].", sold: ".$cars[1][2]."<br>";
echo $cars[2][0].": In stock: ".$cars[2][1].", sold: ".$cars[2][2]."<br>";
echo $cars[3][0].": In stock: ".$cars[3][1].", sold: ".$cars[3][2]."<br>";
?>
```

Bisa meletakkan perulangan for di dalam perulangan for lainnya untuk mendapatkan elemen dari array \$ cars (masih harus menunjuk ke dua indeks).

```
<?php
for ($row = 0; $row < 4; $row++) {
    echo "<p><b>Row number $row</b></p>";
    echo "<ul>";
    for ($col = 0; $col < 3; $col++) {
        echo "<li>".$cars[$row][$col]."</li>";
    }
    echo "</ul>";
}
?>
```

### 3.4 Array Penyortiran PHP

PHP Sorting Array Elemen-elemen dalam sebuah array dapat diurutkan dalam urutan abjad atau numerik, turun atau naik. Fungsi sortir array PHP berikut.



- **sort ()** - mengurutkan array dalam urutan menaik
- **rsort ()** - mengurutkan array dalam urutan menurun
- **asort ()** - mengurutkan array asosiatif dalam urutan naik, sesuai dengan nilainya
- **ksort ()** - mengurutkan array asosiatif dalam urutan naik, menurut kuncinya
- **arsort ()** - mengurutkan array asosiatif dalam urutan menurun, sesuai dengan nilainya
- **krsort ()** - mengurutkan array asosiatif dalam urutan menurun, menurut kuncinya

### Urutkan Array dalam Ascending Order - Sort ()

Contoh berikut mengurutkan elemen dari array \$ cars dalam urutan abjad.

```
<?php
$scars = array("Volvo", "BMW", "Toyota");
sort($cars);
?>
```

Contoh berikut mengurutkan elemen dari array \$ numbers dalam urutan numerik menaik.

```
<?php
$numbers = array(4, 6, 2, 22, 11);
sort($numbers);
?>
```

### Urutkan Array dalam Urutan Descending - Rsort ()

Contoh berikut mengurutkan elemen dari array \$ cars dalam urutan abjad.

```
<?php
$scars = array("Volvo", "BMW", "Toyota");
rsort($cars);
?>
```

Contoh berikut mengurutkan elemen dari array \$ numbers dalam urutan numerik menurun.

```
<?php
$numbers = array(4, 6, 2, 22, 11);
rsort($numbers);
?>
```

### **Sortir Array (Ascending Order), Menurut Nilai - Asort ()**

Contoh berikut mengurutkan array asosiatif dalam urutan menaik, sesuai dengan nilainya.

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
asort($age);
?>
```

### **Sortir Array (Ascending Order), Menurut Key - Ksort ()**

Contoh berikut mengurutkan array asosiatif dalam urutan menaik, menurut kuncinya.

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
ksort($age);
?>
```

### **Sort Array (Descending Order), Menurut Nilai - Arsort ()**

Contoh berikut mengurutkan array asosiatif dalam urutan menurun, sesuai dengan nilainya.

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
arsort($age);
?>
```

### **Sortir Array (Urutan Descending), Menurut Key - Krsort ()**

Contoh berikut mengurutkan array asosiatif dalam urutan menurun, sesuai dengan kuncinya.

```
<?php  
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");  
krsort($age);  
?>
```

