

LAPORAN PHP DASAR



DI SUSUN OLEH:

UMAR JEKLIN

2143021

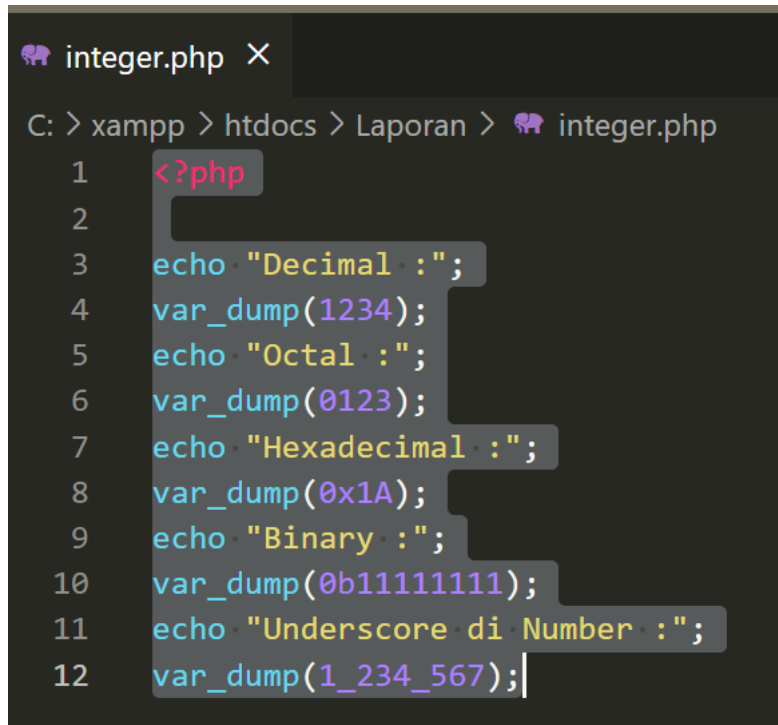
PRODI TEKNIK INFORMATIKA

STMIK WIDYA CIPTA DHARMA

SAMARINDA

2023

1. Tipe Data Number

A screenshot of a code editor window titled 'integer.php'. The code is written in PHP and demonstrates various integer representations. The code is as follows:

```
1 <?php
2
3 echo "Decimal : ";
4 var_dump(1234);
5 echo "Octal : ";
6 var_dump(0123);
7 echo "Hexadecimal : ";
8 var_dump(0x1A);
9 echo "Binary : ";
10 var_dump(0b1111111);
11 echo "Underscore di Number : ";
12 var_dump(1_234_567);
```

Angka 1234 direpresentasikan dalam format desimal. Fungsi `var_dump()` digunakan untuk menampilkan jenis dan nilai variabel. Dalam hal ini, outputnya adalah:

Binary :int(255)

Angka 0123 ditulis dalam format oktal. Dalam PHP, jika angka diawali dengan nol (0), itu diperlakukan sebagai nilai oktal. Output dari kode ini adalah:

Octal :int(83)

Nilai 0x1A mewakili angka dalam format heksadesimal. Dalam PHP, jika angka diawali dengan 0x, itu ditafsirkan sebagai nilai heksadesimal. Output dari kode ini adalah:

Hexadecimal :int(26)

Nilai 0b11111111 adalah representasi biner dari suatu bilangan. Dalam PHP, jika angka diawali dengan 0b, itu dianggap sebagai nilai biner. Output dari kode ini adalah:

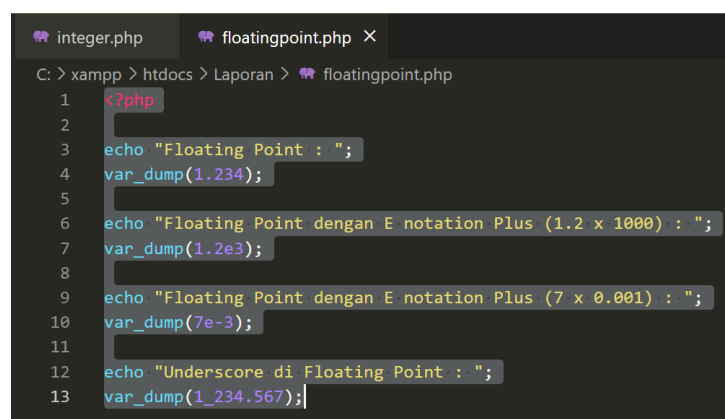
Binary :int(255)

Dalam PHP 7.4 dan versi yang lebih baru, garis bawah (_) dapat digunakan untuk meningkatkan keterbacaan literal numerik dengan memisahkan kelompok digit. Itu tidak mempengaruhi nilai angka. Output dari kode ini adalah:

Underscore di Number :int(1234567)

Singkatnya, kode ini menunjukkan berbagai representasi nilai numerik dalam PHP, termasuk desimal, oktal, heksadesimal, biner, dan penggunaan garis bawah untuk meningkatkan keterbacaan angka besar.

A. Tipe floating-point



```
integer.php floatingpoint.php X
C: > xampp > htdocs > Laporan > floatingpoint.php
1  <?php
2
3  echo "Floating Point : ";
4  var_dump(1.234);
5
6  echo "Floating Point dengan E notation Plus (1.2 x 1000) : ";
7  var_dump(1.2e3);
8
9  echo "Floating Point dengan E notation Plus (7 x 0.001) : ";
10 var_dump(7e-3);
11
12 echo "Underscore di Floating Point : ";
13 var_dump(1_234.567);
```

Dalam cuplikan kode ini, angka 1,234 direpresentasikan sebagai nilai floating-point. Angka floating-point dalam PHP digunakan untuk mewakili angka dengan bagian pecahan atau desimal. Output dari kode ini adalah:

Floating Point : `float(1.234)`

Fungsi `var_dump()` menampilkan jenis dan nilai variabel.

Di sini, `1,2e3` mewakili nilai 1,2 dikalikan dengan 10 yang dipangkatkan 3. Ini dikenal sebagai notasi ilmiah atau eksponensial. Output dari kode ini adalah:

Floating Point dengan E notation Plus (1.2×1000) : `float(1200)`

Dalam notasi ilmiah, `e3` berarti "kali 10 pangkat 3."

Demikian pula, `7e-3` mewakili nilai 7 dikalikan dengan 10 yang dipangkatkan -3. Output dari kode ini adalah:

Floating Point dengan E notation Plus (7×0.001) : `float(0.007)`

Di sini, `e-3` berarti "kali 10 pangkat -3."

Dalam PHP 7.4 dan versi yang lebih baru, garis bawah (`_`) dapat digunakan untuk meningkatkan keterbacaan literal numerik dengan memisahkan kelompok digit. Itu tidak mempengaruhi nilai angka. Output dari kode ini adalah:

Underscore di Floating Point : `float(1_234.567)`

Garis bawah hanya digunakan sebagai pemisah visual untuk keterbacaan yang lebih baik.

Singkatnya, kode yang disediakan menunjukkan berbagai cara untuk mewakili angka floating-point dalam PHP, termasuk notasi desimal standar, notasi ilmiah

atau eksponensial, dan penggunaan garis bawah untuk meningkatkan keterbacaan angka besar.

B. Integer Overflow

```
C: > xampp > htdocs > Laporan > overflow.php
1  <?php
2
3  echo "Integer Overflow 32 bit : ";
4  var_dump(2147483648);
5
6  echo "Integer Overflow 64 bit : ";
7  var_dump(9223372036854775808);
```

Dalam cuplikan kode ini, angka 2147483648 ditetapkan ke variabel bilangan bulat. Namun, nilai ini melebihi nilai maksimum yang dapat disimpan dalam bilangan bulat 32-bit. Dalam PHP, pada sebagian besar sistem, nilai maksimum untuk bilangan bulat bertanda 32-bit adalah 2147483647. Ketika nilai bilangan bulat melebihi batas ini, itu menyebabkan luapan. Output dari kode ini adalah:

```
Integer Overflow 32 bit : int(2147483648)
```

Perhatikan bahwa tipe output adalah float, bukan int, menunjukkan bahwa PHP secara otomatis mengubah nilai menjadi angka floating-point untuk mengakomodasi besaran yang lebih besar.

Demikian pula, dalam cuplikan kode ini, angka 9223372036854775808 ditetapkan ke variabel bilangan bulat. Nilai ini melebihi nilai maksimum yang dapat disimpan dalam bilangan bulat 64-bit. Pada sebagian besar sistem, nilai maksimum untuk bilangan

bulat bertanda 64-bit adalah 9223372036854775807. Sekali lagi, terjadi luapan. Output dari kode ini adalah:

Integer Overflow 64 bit : float(9.223372036854776E+18)

Jenis output adalah float, dan nilainya direpresentasikan dalam notasi ilmiah atau eksponensial (9,2233720368548E+18) untuk mengakomodasi besaran yang lebih besar.

Dalam PHP, ketika nilai integer meluap, secara otomatis dipromosikan ke angka floating-point untuk mencegah kehilangan data. Angka floating-point memiliki rentang yang lebih besar, tetapi mereka mungkin memiliki keterbatasan presisi dan tunduk pada kebiasaan aritmatika floating-point.

2. Tipe Data Boolean

```
C: > xampp >htdocs > Laporan > 🐘 boolean.php
1  <?php
2
3  echo "Benar : ";
4  var_dump(true);
5
6  echo "Salah: ";
7  var_dump(false);
```

Dalam cuplikan kode ini, nilai true ditetapkan ke variabel boolean. Tipe data boolean dalam PHP mewakili nilai logis, yang dapat memiliki dua kemungkinan status: true atau false. Output dari kode ini adalah:

Benar : bool(true)

Fungsi var_dump() menampilkan jenis dan nilai variabel. Dalam hal ini, jenisnya adalah bool (boolean), dan nilainya benar.

Demikian pula, nilai false ditugaskan ke variabel boolean. Output dari kode ini adalah:

Salah: bool(false)

Di sini, jenisnya adalah bool (boolean), dan nilainya salah.

Nilai Boolean biasanya digunakan untuk perbandingan logis, pernyataan kondisional, dan aliran kontrol dalam PHP. Mereka mewakili kebenaran atau kepalsuan suatu ekspresi. Benar mewakili kondisi benar atau positif, sedangkan salah mewakili kondisi salah atau negatif.

3. Tipe Data String

```
1  <?php
2
3  echo 'Nama : ';
4  echo 'Eko Kurniawan Kennedy';
5
6
7  echo 'Nama : ';
8  echo 'Eko Kurniawan Kennedy';
9  echo "\n";
10
11 echo "Nama : ";
12 echo "Eko\t Kurniawan\t Kennedy\n";
13
14
15 echo <<<EKO
16 Ini adalah contoh string yang sangat panjang,
17 dan juga gak perlu ngetik ENTER secara manual, "bisa quote" juga
18 Heredoc
19 EKO;
20
21 echo <<<'EKO'
22 Ini adalah contoh string yang sangat panjang,
23 dan juga gak perlu ngetik ENTER secara manual, "bisa quote" juga
24 Nowdoc
25 EKO;
```

Dalam cuplikan kode ini, string 'Eko Kurniawan Kennedy' ditampilkan menggunakan pernyataan echo. Dalam PHP, string digunakan untuk

mewakili urutan karakter. Tanda kutip tunggal (') digunakan untuk membatasi literal string. Output dari kode ini adalah:

```
Nama : Eko Kurniawan Kennedy
```

Tambahan "n" dalam tanda kutip ganda adalah urutan escape yang mewakili karakter baris baru. Ini menyisipkan jeda baris dalam output.

Output dari kode ini adalah:

```
Nama : Eko Kurniawan Kennedy
```

Dalam cuplikan kode ini, tanda kutip ganda (") digunakan untuk membatasi literal string. t dalam tanda kutip ganda mewakili karakter tab, dan "n" mewakili karakter baris baru. Output dari kode ini adalah:

```
Nama : Eko    Kurniawan    Kennedy
```

Di sini, sintaks <<<EKO digunakan untuk menunjukkan string heredoc. Ini memungkinkan string multiline tanpa perlu jeda baris manual atau urutan escape. Output dari kode ini adalah:

```
Kennedy Ini adalah contoh string yang sangat panjang, dan juga gak perlu ngetik ENTER  
secara manual, "bisa quote" juga HeredocIni adalah contoh string yang sangat panjang, dan  
juga gak perlu ngetik ENTER secara manual, "bisa quote" juga Nowdoc
```

String heredoc dimulai setelah <<<EKO dan diakhiri dengan pengidentifikasi yang sama (EKO) di awal baris.

Sintaks <<<'EKO' menunjukkan string nowdoc. Ini berperilaku mirip dengan string heredoc tetapi tidak mengurai variabel atau urutan pelarian di dalamnya. Output dari kode ini adalah:

```
Kennedy Ini adalah contoh string yang sangat panjang, dan juga gak perlu ngetik ENTER  
secara manual, "bisa quote" juga HeredocIni adalah contoh string yang sangat panjang, dan  
juga gak perlu ngetik ENTER secara manual, "bisa quote" juga Nowdoc
```

String Nowdoc dimulai dengan <<<'EKO' dan diakhiri dengan identifier (EKO) di awal baris.

Singkatnya, kode menunjukkan berbagai cara untuk mewakili dan mengeluarkan nilai string dalam PHP menggunakan tanda kutip tunggal, tanda kutip ganda dengan urutan escape, string heredoc, dan string nowdoc. String dalam PHP dapat berisi karakter alfanumerik, karakter khusus, dan dapat menjangkau beberapa baris.

A. String Single Quoted

```
1  <?php
2  echo 'Nama :';
3  echo 'Eko Kurniawan Khannedy';
4
5  ?>
```

Dalam cuplikan kode ini, tanda kutip tunggal (') digunakan untuk membatasi literal string. Saat menggunakan tanda kutip tunggal, string diperlakukan secara harfiah, artinya variabel dan urutan escape dalam string tidak akan ditafsirkan. Output dari kode ini adalah:

Nama :Eko Kurniawan Khannedy

Seperti yang Anda lihat, string adalah output persis seperti yang tertulis, termasuk spasi dan tanda baca.

Tanda kutip tunggal sering digunakan ketika Anda ingin menentukan literal string yang tidak memerlukan interpolasi variabel atau interpretasi urutan escape.

Perhatikan bahwa menggunakan tanda kutip tunggal versus tanda kutip ganda (") adalah masalah preferensi dan tergantung pada kasus penggunaan tertentu. Penting untuk memilih pembatas string yang sesuai berdasarkan kebutuhan Anda.

B. String Double Quoted

```
C: > xampp > htdocs > Laporan > stringdouble.php
1  <?php
2  echo 'Nama : ';
3  echo 'Eko Kurniawan Khannedy';
4  echo "\n";
5
6  echo "Nama :";
7  echo "Eko\t Kurniawan\t Khannedy\n";
8  ?>
```

Dalam cuplikan kode ini, tanda kutip tunggal (') digunakan untuk membatasi literal string. Output dari kode ini adalah:

Nama : Eko Kurniawan Khannedy Nama :Eko Kurniawan Khannedy

String kutipan ganda memungkinkan interpolasi string yang lebih mudah dan interpretasi urutan escape dibandingkan dengan string kutipan tunggal. Mereka biasanya digunakan ketika Anda perlu memasukkan variabel atau urutan escape dalam string.

Singkatnya, kode ini menunjukkan penggunaan string kutipan ganda untuk menghasilkan string dengan interpolasi variabel dan interpretasi urutan escape, menghasilkan lebih banyak fleksibilitas dan generasi string dinamis.

C. String Nowdoc

```
C: > xampp > htdocs > Laporan > nowdoc.php
1  <?php
2  echo <<<'EKO'
3  Ini adalah contoh string yang sangat
4  panjang, dan juga gak perlu ngetik ENTER secara
5  manual, "bisa quote" juga
6  EKO;
7  ?>
```

Dalam cuplikan kode ini, sintaks <<<'EKO' digunakan untuk menunjukkan string nowdoc. String Nowdoc adalah cara untuk menentukan string literal dalam PHP tanpa interpolasi variabel atau interpretasi urutan escape. String dimulai setelah <<<'EKO' dan diakhiri dengan pengidentifikasi EKO di awal baris.

Dalam kode yang disediakan, string nowdoc digunakan untuk menghasilkan string multi-baris. String ditampilkan persis seperti yang tertulis, mempertahankan jeda baris dan spasi. Berikut output dari kode:

Ini adalah contoh string yang sangat panjang, dan juga gak perlu ngetik ENTER secara manual, "bisa quote" juga

Singkatnya, sintaks nowdoc memungkinkan Anda untuk mendefinisikan string literal dalam PHP tanpa interpolasi variabel atau interpretasi urutan melarikan diri. Ini berguna untuk mempertahankan pemformatan dan menentukan blok besar teks atau contoh kode.

D. String Heredoc

```
C: > xampp > htdocs > Laporan > heredoc.php
1  <?php
2  echo <<<EKO
3  Ini adalah contoh string yang
4  panjang, dan juga gak perlu ngetik ENTER secara
5  manual, "bisa quote" juga
6  EKO;
7  ?>
```

Dalam cuplikan kode ini, sintaks <<<EKO digunakan untuk menunjukkan string heredoc. String Heredoc adalah cara lain untuk menentukan string literal dalam PHP, mirip dengan string nowdoc. String dimulai setelah <<<EKO dan diakhiri dengan pengidentifikasi EKO di awal baris.

Dalam kode yang disediakan, string heredoc digunakan untuk menghasilkan string multi-baris. String ditampilkan persis seperti yang tertulis, termasuk jeda baris dan lekukan. Berikut output dari kode:

Ini adalah contoh string yang sangat panjang, dan juga gak perlu ngetik ENTER secara manual, "bisa quote" juga

4. Variable

A. Variable

```
C: > xampp > htdocs > Laporan > variable.php
1  <?php
2  $name = "Eko";
3  $age = 30;
4
5  echo "Name : ";
6  echo $name;
7  echo "\n";
8  echo "Age";
9  echo $age;
10 ?>
```

Dalam cuplikan kode ini, dua variabel didefinisikan: \$name dan \$age.

Baris \$name = "Eko"; menetapkan nilai string "Eko" ke variabel \$name. Variabel ini digunakan untuk menyimpan nama seseorang.

Garis `$age = 30;` menetapkan nilai bilangan bulat 30 ke variabel `$age`.
Variabel ini digunakan untuk menyimpan usia seseorang.

Setelah mendefinisikan variabel, kode melanjutkan untuk menampilkan nilainya menggunakan pernyataan `echo`.

Pernyataan `echo` digunakan untuk menampilkan teks atau nilai variabel. Dalam hal ini, kode mengeluarkan string `"Name : "` dan `"Age"` bersama dengan nilai-nilai yang disimpan dalam variabel `$name` dan `$age`.

`\n` escape sequence digunakan untuk menambahkan karakter baris baru, membuat baris baru dalam output.

Jadi, jika kita menjalankan kode ini, outputnya adalah:

`Name : Eko Age30`

Output menampilkan label `"Name : "` diikuti oleh nilai variabel `$name`, yaitu `"Eko"`. Kemudian, pada baris berikutnya, ini menampilkan label `"Umur"` diikuti dengan nilai variabel `$age`, yaitu 30.

Singkatnya, kode menunjukkan penggunaan variabel dalam PHP untuk menyimpan dan memanipulasi data. Nilai variabel dapat diakses dan ditampilkan menggunakan pernyataan `echo`.

B. Variables

```
C: > xampp >htdocs > Laporan > 🐘 variables.php
1  <?php
2  $name = "eko";
3  $$name = "keren";
4
5  echo "\$name";
6  echo $name;
7  echo "\n";
8  echo $eko;
9  echo "\n";|
10 ?>
```

Dua variabel digunakan: \$name dan \$ \$name.

Baris \$name = "eko"; menetapkan nilai string "eko" ke variabel \$name. Variabel ini mudah dan memegang nilai string "eko".

Baris \$\$name = "keren"; memperkenalkan konsep yang disebut variabel variabel. Ini menciptakan variabel baru dengan nama berdasarkan nilai \$name. Dalam hal ini, karena nilai \$name adalah "eko", ia membuat variabel bernama \$eko dan memberikan nilai string "keren" padanya.

Jadi setelah mengeksekusi baris ini, kita memiliki dua variabel: \$name dengan nilai "eko" dan \$eko dengan nilai "keren". Nama variabel kedua ditentukan oleh nilai variabel pertama.

Selanjutnya, kode melanjutkan untuk menampilkan nilai-nilai variabel menggunakan pernyataan echo.

Pernyataan echo pertama, echo "\$name";, menghasilkan string literal "\$name". Garis miring terbalik () digunakan untuk menghindari tanda dolar (\$), sehingga ditampilkan sebagai bagian dari output.

Pernyataan echo kedua, echo \$name;, menghasilkan nilai variabel \$name, yaitu "eko".

Pernyataan echo ketiga, echo \$eko;, menghasilkan nilai variabel \$eko, yaitu "keren".

Jadi, ketika menjalankan kode ini, outputnya adalah:

\$nameeko keren

Singkatnya, kode ini menunjukkan penggunaan variabel variabel dalam PHP, di mana nama variabel ditentukan oleh nilai variabel lain. Ini juga menggambarkan bagaimana menetapkan nilai ke variabel dan mengeluarkan nilainya menggunakan pernyataan echo.

5. Constant

```
C: > xampp >htdocs > Laporan > 🐘 consttant.php
1  <?php
2  define("AUTHOR", "programmer zaman now");
3  define("APP_VERSION", 100);
4
5  echo AUTHOR;
6  echo "\n";
7  echo APP_VERSION;|
8  echo "\n";
9  ?>
```

Dua konstanta didefinisikan menggunakan fungsi define(). Konstanta adalah pengidentifikasi (nama) untuk nilai tetap yang tidak dapat diubah selama eksekusi script.

Baris mendefinisikan ("AUTHOR", "programmer zaman now"); mendefinisikan konstanta bernama "AUTHOR" dengan nilai string "programmer zaman now". Konstanta biasanya ditulis dalam huruf besar untuk membedakannya dari variabel.

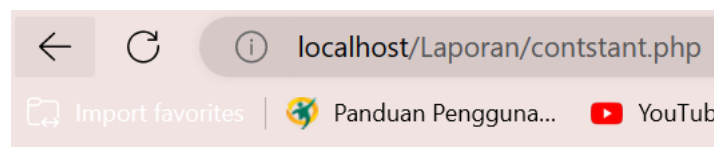
Garis mendefinisikan ("APP_VERSION", 100); mendefinisikan konstanta lain bernama "APP_VERSION" dengan nilai bilangan bulat 100.

Setelah mendefinisikan konstanta, kode melanjutkan untuk menampilkan nilainya menggunakan pernyataan echo.

Pernyataan echo pertama, echo AUTHOR;, menghasilkan nilai konstanta "AUTHOR", yang merupakan string "programmer zaman now".

Pernyataan echo kedua, echo APP_VERSION;, menghasilkan nilai konstanta "APP_VERSION", yang merupakan bilangan bulat 100.

Jadi, ketika menjalankan kode ini, outputnya adalah:



programmer zaman now 100

6. Data NULL

```
C: > xampp > htdocs > Laporan > datanull.php
1  <?php
2  $name = "Eko";
3  $name = NULL;
4
5  $age = null;
6  ?>
```

dua variabel, \$name dan \$age, digunakan untuk menunjukkan konsep null dalam PHP.

Awalnya, variabel \$name diberi nilai string "Eko". Namun, di baris berikutnya, \$name ditetapkan ulang dengan nilai NULL. Nilai NULL mewakili tidak adanya nilai atau tidak adanya data yang berarti. Ini adalah

nilai khusus dalam PHP yang digunakan untuk menunjukkan tidak adanya nilai yang ditetapkan.

Demikian pula, variabel \$age secara langsung diberi nilai null. Baik \$age dan \$name sekarang memegang nilai null.

Dan ada pula cara untuk mengecek NULL yaitu,

```
C: > xampp > htdocs > Laporan > 🐘 ceknull.php
1  <?php
2  $name = "Eko";
3  $name = NULL;
4
5  $isNull = is_null($name);
6  var_dump($isNull);
7  ?>
```

Dalam cuplikan kode ini, variabel \$name awalnya diberi nilai string "Eko". Namun, di baris berikutnya, \$name ditetapkan ulang dengan nilai NULL, mewakili tidak adanya nilai.

Fungsi is_null() kemudian digunakan untuk memeriksa apakah nilai \$name adalah null. Fungsi mengembalikan true jika variabelnya null, dan false sebaliknya.

Hasil is_null (\$name) ditetapkan ke variabel \$isNull. Fungsi var_dump() kemudian digunakan untuk menampilkan nilai \$isNull dan tipe datanya.

Saat menjalankan kode ini, outputnya adalah:

```
bool(true)
```

7. Tipe Data Array

```
C: > xampp > htdocs > Laporan > 🐘 madearray.php
1  <?php
2
3  $values = array(1, 2, 3, 4);
4  var_dump($values)
5
6  $names = ["Eko", "Kurniawan", "Khannedy"];
7  var_dump($names);
8  ?>
```

Dua array, \$values dan \$names, didefinisikan dan nilainya ditampilkan menggunakan fungsi var_dump().

Array dalam PHP adalah struktur data yang memungkinkan Anda menyimpan banyak nilai dalam satu variabel. Ini adalah kumpulan elemen, di mana setiap elemen memiliki indeks atau kunci unik untuk mengakses nilainya.

Pada baris pertama, array \$values didefinisikan menggunakan konstruk array(). Ini berisi empat elemen: 1, 2, 3, dan 4. Konstruk array() adalah fungsi bawaan dalam PHP yang digunakan untuk membuat array.

Baris kedua menggunakan fungsi var_dump() untuk menampilkan konten array \$values. var_dump() adalah fungsi debugging yang berguna yang menampilkan informasi tentang variabel, termasuk jenis dan nilainya.

Dan ini outpunya :

```
← → 🔄 localhost/PHP%20DASAR/Membuat%20Array.php
array(4) ( [0]=> int(1) [1]=> int(2) [2]=> int(3) [3]=> int(4) ) array(3) ( [0]=> string(3) "Eko" [1]=> string(9) "Kurniawan" [2]=> string(8) "Khannedy" )
```

Array juga memiliki operasi. Berikut contoh kode program operasi array

```
C: > xampp > htdocs > Laporan > 🐘 oparray.php
1  <?php
2  $names = ["Eko", "Kurniawan", "Khannedy"];
3  var_dump($names[0]);
4  $names[0] = "Budi";
5  var_dump($names);
6  unset($names[1]);
7  var_dump($names);
8  $names[] = "Eko";
9  var_dump($names);
10 var_dump(count($names));
11 ?>
```

Array bernama \$names dibuat dan dimanipulasi menggunakan berbagai operasi.

Baris \$names = ["Eko", "Kurniawan", "Khannedy"]; menginisialisasi array \$names dengan tiga elemen: "Eko", "Kurniawan", dan "Khannedy".

var_dump(\$names[0]); pernyataan menghasilkan nilai pada indeks 0 dari array \$names, yaitu "Eko".

Baris \$names[0] = "Budi"; memperbarui nilai pada indeks 0 menjadi "Budi". Oleh karena itu, elemen pertama dari array diubah dari "Eko" menjadi "Budi".

var_dump(\$names); Pernyataan digunakan untuk menampilkan konten array \$names yang diperbarui. Sekarang, itu akan menunjukkan nilai modifikasi ["Budi", "Kurniawan", "Khannedy"].

Yang belum disetel(\$names[1]); Pernyataan menghapus elemen pada indeks 1 dari array \$names. Setelah operasi ini, array akan berisi dua elemen: "Budi" dan "Khannedy".

`var_dump($names);` Pernyataan digunakan lagi untuk menampilkan konten array `$names` yang diperbarui. Sekarang, itu akan menunjukkan nilai `["Budi", "Khannedy"]`.

Baris `$names[] = "Eko";` menambahkan nilai "Eko" ke akhir array `$names`, menghasilkan tiga elemen: "Budi", "Khannedy", dan "Eko".

`var_dump($names);` Pernyataan digunakan untuk menampilkan konten array `$names` yang diperbarui. Sekarang, itu akan menunjukkan nilai `["Budi", "Khannedy", "Eko"]`.

Akhirnya, `var_dump (hitungan ($names));` pernyataan digunakan untuk menampilkan jumlah elemen dalam array `$names`. Ini akan menampilkan nilai 3, menunjukkan bahwa array berisi tiga elemen.

Biasanya di kebanyakan bahasa pemrograman, terdapat tipe data bernama Map, yaitu asosiasi antara key dan value.

Berikut contoh kode programnya.

```
C: > xampp > htdocs > Laporan > map.php
1  <?php
2  $seko = array(
3      "id" => "eko"
4      "name" => "Eko Kurniawan",
5      "age" => 30
6  );
7
8  $budi = [
9      "id" => "budi",
10     "name" => "Budi Nugraha",
11     "age" => 35
12 ];
13 ?>
```

Dua array asosiatif, `$seko` dan `$budi`, didefinisikan.

Array asosiatif dalam PHP adalah array yang menggunakan pasangan kunci-nilai untuk menyimpan data. Tidak seperti array terindeks biasa, di mana kuncinya adalah indeks numerik, array asosiatif memungkinkan Anda menggunakan kunci kustom.

Di blok pertama, array \$eko didefinisikan menggunakan konstruk array(). Ini terdiri dari tiga pasangan kunci-nilai:

Kunci "id" dengan nilai "eko".

Kunci "nama" dengan nilai "Eko Kurniawan".

Kunci "usia" dengan nilai 30.

Di blok kedua, array \$budi didefinisikan menggunakan tanda kurung siku [], yang merupakan sintaks singkatan yang diperkenalkan di PHP 5.4. Ini juga berisi tiga pasangan kunci-nilai:

Kunci "id" dengan nilai "budi".

Kunci "nama" dengan nilai "Budi Nugraha".

Kunci "usia" dengan nilai 35.

Kedua array menunjukkan cara menyimpan data terkait menggunakan kunci yang bermakna. Misalnya, kunci "id" dapat digunakan untuk mengidentifikasi setiap orang secara unik, sedangkan kunci "nama" dan "usia" menyimpan nilai yang sesuai.

Array asosiatif menyediakan cara mudah untuk mengatur dan mengakses data menggunakan kunci deskriptif daripada mengandalkan indeks numerik. Anda dapat mengakses nilai dalam array asosiatif dengan menggunakan kunci seperti yang ditunjukkan di bawah ini:

```
echo $eko["name"]; // Output: Eko Kurniawan
```

```
echo $budi["age"]; // Output: 35
```

Seperti dijelaskan di awal, Array di PHP bisa berisikan data apapun sehingga kita juga bisa membuat array di dalam array jika memang dibutuhkan.

Berikut contoh kode program.

```
C: > xampp > htdocs > Laporan > 🐘 arrayinarray.php
1  <?php
2  $eko = array(
3      "id" => "eko",
4      "name" => "Eko Kurniawan",
5      "age" => 30,
6      "address" => [
7          "city" => "Jakarta",
8          "country" => "Indonesia"
9      ]
10 );
11 ?>
```

Array asosiatif bernama \$eko didefinisikan. Ini berisi berbagai pasangan kunci-nilai, termasuk array bersarang.

Array \$eko terdiri dari pasangan kunci-nilai berikut:

Kunci "id" dengan nilai "eko".

Kunci "nama" dengan nilai "Eko Kurniawan".

Kunci "usia" dengan nilai 30.

Kunci "alamat" dengan nilai array asosiatif lain.

Array bertumpuk yang ditetapkan ke kunci "alamat" berisi pasangan kunci-nilainya sendiri:

Kunci "kota" dengan nilai "Jakarta".

Kunci "negara" dengan nilai "Indonesia".

Dengan menggunakan array bertumpuk, Anda dapat mengatur dan mewakili struktur data yang lebih kompleks. Dalam hal ini, kunci "alamat" menyimpan informasi tentang kota dan negara tempat tinggal seseorang.

Untuk mengakses nilai dalam array \$eko, Anda dapat menggunakan kunci masing-masing:

```
echo $eko["name"];           // Output: Eko Kurniawan
```

```
echo $eko["age"];           // Output: 30
```

```
echo $eko["address"]["city"]; // Output: Jakarta
```

```
echo $eko["address"]["country"]; // Output: Indonesia
```

8. Operator Aritmatika

Berikut adalah 7 jenis operator aritmatika pada php:

1. Penambahan

Jenis pertama adalah penambahan yang disimbolkan (+) dalam bahasa pemrograman. Penambahan berfungsi jika ingin menambahkan satu operan dengan operan yang lain.

2. Pengurangan

Dalam bahasa pemrograman, pengurangan disimbolkan dengan tanda huruf (-) yang berfungsi untuk mengurangi suatu operan satu dengan lainnya.

3. Perkalian

Operator perkalian pada php digunakan untuk mengalikan bilangan, baik dalam bentuk tipe data atau variabel. Untuk membuat operasi perkalian di php harus menggunakan tanda bintang *, bukan huruf x. Misal \$x * \$y.

4. Pembagian

Terakhir ada jenis operator aritmatika yaitu pembagian yang dalam bahasa pemrograman disimbolkan dengan tanda huruf garis miring (/). Fungsi pembagian adalah membagi suatu operan dengan operan yang lain.

Berikut contoh kode program.

```
C: > xampp > htdocs > Laporan > aritmatika.php
1  <?php
2  $result = 10 + 10;
3  var_dump($result);
4
5  $result = 100 % 3;
6  var_dump($result);
7
8  ?>
```

Dua operasi aritmatika dilakukan dan hasilnya ditampilkan menggunakan `var_dump()`.

`$result = 10 + 10;` melakukan penjumlahan antara nilai 10 dan 10.

Operator `+` digunakan untuk penambahan dalam PHP. Hasil dari operasi ini adalah 20, yang ditugaskan ke variabel `$result`.

`var_dump($result);` Kemudian menghasilkan nilai `$result`, yaitu 20.

`$result = 100 % 3;` Menghitung modulus (sisanya) membagi 100 dengan 3.

Operator `%` digunakan untuk operasi modulus dalam PHP. Sisanya ketika 100 dibagi dengan 3 adalah 1, sehingga nilai 1 ditugaskan ke variabel `$result`.

`var_dump($result);` Kemudian menghasilkan nilai `$result`, yaitu 1.

kode menunjukkan operasi aritmatika dasar dalam PHP. Operasi penjumlahan `+` digunakan untuk menambahkan dua angka, sedangkan operasi modulus `%` menghitung sisa pembagian satu angka dengan angka lainnya. Hasil operasi ini disimpan dalam variabel `$result` dan kemudian ditampilkan menggunakan `var_dump()`.

9. Operator Penugasan

Berikut contoh kode program.

```
C: > xampp > htdocs > Laporan > penaritmatika.php
1  <?php
2  $total = 0;
3
4  $fruit = 10000;
5  $chicken = 35000;
6  $orangejuice = 1000;
7
8  $total += $fruit;
9  $total += $chicken;
10 $total += $orangejuice;
11
12 var_dump($total);
13 >
```

Variabel bernama \$total diinisialisasi dengan nilai 0. Variabel ini akan digunakan untuk menghitung total biaya berbagai item.

Tiga variabel tambahan didefinisikan:

\$fruit dengan nilai 10000, mewakili biaya buah-buahan.

\$chicken dengan nilai 35000, mewakili biaya ayam.

\$orangejuice dengan nilai 1000, mewakili biaya jus jeruk.

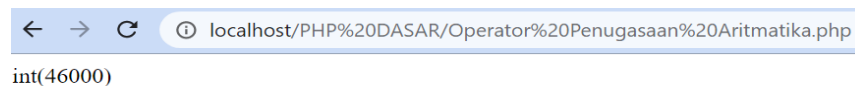
Untuk menghitung biaya total, nilai \$fruit, \$chicken, dan \$orangejuice ditambahkan ke variabel \$total menggunakan operator +=. Operator ini menambahkan operan kanan ke operan sebelah kiri dan menetapkan hasilnya kembali ke operan sebelah kiri.

Pernyataan \$total += \$fruit;, \$total += \$chicken;, dan \$total += \$orangejuice; Akumulasi biaya setiap item ke dalam variabel \$total.

Akhirnya, var_dump(\$total); digunakan untuk menampilkan nilai variabel \$total. Fungsi var_dump() menghasilkan tipe data dan nilai variabel.

Ketika Anda menjalankan kode ini, itu akan menghasilkan total biaya dengan menambahkan nilai \$fruit, \$chicken, dan \$orangejuice bersama-sama.

Misalnya, jika Anda menjalankan cuplikan kode ini, outputnya adalah:



← → ↻ ⓘ localhost/PHP%20DASAR/Operator%20Penugasaan%20Aritmatika.php
int(46000)

10. Operator Perbandingan

Operator perbandingan biasanya disebut dengan operator relasi, dipakai untuk membandingkan dua buah nilai. Hasil perhitungan menggunakan operator ini adalah True yang disimbolkan nilai 1 dan False dengan simbol 0. Berikut ini jenis-jenis operator perbandingan:

== sama dengan (bukan pemberi nilai)

!= tidak sama dengan

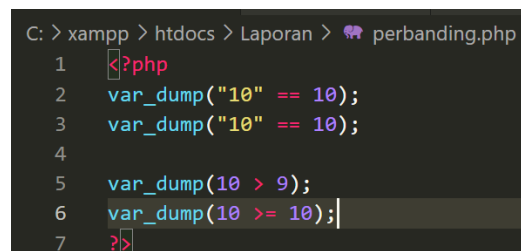
> lebih dari

< kurang dari

>= lebih dari sama dengan

<= kurang dari sama dengan

Berikut contoh kode program.



```
C: > xampp > htdocs > Laporan > perbanding.php
1  <?php
2  var_dump("10" == 10);
3  var_dump("10" == 10);
4
5  var_dump(10 > 9);
6  var_dump(10 >= 10);
7  ?>
```

Singkatnya, kode menunjukkan operasi perbandingan yang berbeda dalam PHP. Operator kesetaraan == membandingkan nilai tanpa mempertimbangkan tipe data, sedangkan operator identitas === membandingkan nilai dan tipe datanya. Operator yang lebih besar dari >

memeriksa apakah satu nilai lebih besar dari yang lain, dan operator yang lebih besar dari atau sama dengan `>=` memeriksa apakah satu nilai lebih besar dari atau sama dengan yang lain. Fungsi `var_dump()` digunakan untuk menampilkan hasil boolean dari perbandingan ini.

11. Operator Logika

```
C: > xampp > htdocs > Laporan > logika.php
1  <?php
2  var_dump(true && true);
3  var_dump(true && false);
4  var_dump(true || false);
5  var_dump(true xor true);
6  var_dump(!true);
7  ?>
```

Beberapa operasi logis dilakukan menggunakan fungsi `var_dump()` untuk menampilkan hasilnya.

`var_dump(benar && benar);` menggunakan operator AND logis `&&` untuk memeriksa apakah kedua operan benar. Karena kedua operan itu benar, hasil dari operasi ini benar.

Outputnya akan menjadi bool (true), menunjukkan bahwa hasil operasi AND logis antara true dan true adalah true.

`var_dump(benar && salah);` menggunakan operator AND logis `&&` untuk memeriksa apakah kedua operan benar. Dalam hal ini, satu operan benar dan yang lainnya salah. Operator AND logis mengharuskan kedua operan benar agar hasilnya benar.

Outputnya adalah bool (false), menunjukkan bahwa hasil operasi AND logis antara true dan false adalah false.

`var_dump(benar || salah);` menggunakan operator OR logis `||` untuk memeriksa apakah setidaknya salah satu operan itu benar. Karena satu operan benar, hasil operasi ini benar.

Outputnya akan bool(true), menunjukkan bahwa hasil operasi OR logis antara true dan false adalah true.

`var_dump(benar xor benar);` menggunakan operator XOR logis (OR eksklusif) xor untuk memeriksa apakah salah satu operan benar. Dalam hal ini, kedua operan itu benar, jadi hasil operasi XOR salah.

Outputnya akan menjadi bool (false), menunjukkan bahwa hasil operasi XOR logis antara true dan true adalah false.

`var_dump(!true);` menggunakan operator NOT logis! untuk meniadakan nilai operan. Karena operan benar, meniadakannya menggunakan operator NOT logis menghasilkan false.

Outputnya akan bool(false), menunjukkan bahwa hasil operasi NOT logis pada true adalah false.

12. Increment dan Decrement

```
C: > xampp > htdocs > Laporan > incredecre.php
1  <?php
2  $a = 10;
3  $b = ++$a;
4
5  var_dump($b);
6  var_dump($a);
7  ?>
```

Dalam cuplikan kode ini, tindakan berikut dilakukan:

`$a = 10;` menetapkan nilai 10 ke variabel \$a. Jadi, \$a sekarang memegang nilai 10.

`$b = ++$a;` Menambah nilai \$a sebesar 1 menggunakan operator pra-kenaikan ++ dan menetapkan nilai yang dihasilkan ke \$b. Operator pra-

kenaikan meningkatkan nilai variabel sebelum digunakan dalam ekspresi. Dalam hal ini, nilai \$a bertambah menjadi 11, dan nilai baru tersebut ditetapkan ke \$b.

`var_dump($b);` Menampilkan nilai \$b menggunakan fungsi `var_dump()`. Karena \$b diberi nilai \$a setelah bertambah, nilai \$b akan menjadi 11.

`var_dump($a);` Menampilkan nilai \$a menggunakan fungsi `var_dump()`. Setelah operasi pra-kenaikan di langkah 2, nilai \$a bertambah menjadi 11. Oleh karena itu, nilai \$a juga akan menjadi 11.

Output dari kode tersebut adalah:

```
int(11) int(11)
```