Pengantar PHP

Mata Kuliah: Pemograman Web

Materi Praktikum ke: 6

2411102441126
Febri Panji Pratama

FAKULTAS SAINS DAN TEKNOLOGI
PROGRAM STUDI S1 TEKNIK INFORMATIKA
UNIVERSITAS MUHAMMADIYAH KALIMANTAN TIMUR

2411102441170@umkt.ac.id

BAB I PENDAHULUAN

A. Latar Belakang

Dalam dunia pemrograman, pengolahan data menjadi inti dari setiap proses komputasi. Untuk melakukan pengolahan tersebut, diperlukan alat bantu berupa **operator**, yaitu simbol atau tanda yang digunakan untuk melakukan operasi tertentu pada data.
Dua jenis operator yang paling mendasar adalah **operator assignment (penugasan)** dan **operator aritmatika (perhitungan matematis)**. Operator assignment digunakan untuk memberikan nilai kepada variabel, sedangkan operator aritmatika digunakan untuk melakukan operasi matematika seperti penjumlahan, pengurangan, perkalian, pembagian, dan modulus.

Pemahaman terhadap kedua operator ini sangat penting karena hampir semua algoritma dan program komputer menggunakan keduanya. Kesalahan kecil dalam penggunaan operator dapat menyebabkan hasil perhitungan yang salah atau error pada program.

B. Tujuan

- Menjelaskan pengertian dan fungsi dari operator assignment dan operator aritmatika.
- Menunjukkan cara penggunaan kedua operator tersebut dalam bahasa pemrograman.
- Memberikan pemahaman tentang pentingnya operator dalam proses pengolahan data dan pembuatan algoritma.
- Melatih kemampuan logika dasar dalam pemrograman melalui contoh implementasi operator.

C. Tinjauan Pustaka

Menurut *Wahyono (2019)* dalam bukunya Dasar-Dasar Pemrograman Komputer, operator merupakan simbol yang digunakan untuk melakukan operasi pada operand (variabel atau nilai). Operator terbagi dalam beberapa kelompok, di antaranya operator aritmatika, logika, relasional, dan assignment.

1. **Operator Aritmatika**
   Operator ini digunakan untuk melakukan operasi matematis. Contohnya:

- + (penjumlahan)
- - (pengurangan)
- * (perkalian)
- / (pembagian)
- % (modulus/sisa bagi)

Contoh:
$a = 10;
$b = 3;
$hasil = $a + $b; // hasil = 13

2. **Operator Assignment**
   Operator ini digunakan untuk memberikan nilai pada variabel.
   Operator dasar: =
   Operator gabungan: +=, -=, *=, /=, %=
   Contoh:
   $x = 5;
   $x += 3; // sama dengan $x = $x + 3 -> hasilnya 8

Menurut *Siregar (2020)* dalam jurnal **Pembelajaran Dasar Pemrograman Terstruktur**, operator aritmatika dan assignment merupakan fondasi utama untuk memahami logika komputasi dan algoritma karena berfungsi sebagai dasar dalam manipulasi data.

BAB II ALAT DAN BAHAN

A. Alat yang digunakan dalam percobaan atau praktikum ini adalah:

1. Perangkat komputer atau laptop
   Digunakan untuk menjalankan dan menguji program yang berisi operator assignment dan aritmatika.
2. Software editor kode (IDE/Text Editor)
   Contoh: Visual Studio Code, Sublime Text, Notepad++, atau editor bawaan seperti Notepad.
3. Bahasa pemrograman dan interpreter/compiler
   - Untuk PHP: menggunakan XAMPP atau Laragon sebagai server lokal (Apache + PHP).
   - Untuk Python: menggunakan Python Interpreter.
   - Untuk C/C++: menggunakan Code::Blocks atau Dev-C++.
4. Browser (jika menggunakan PHP)
   Contoh: Google Chrome, Mozilla Firefox, atau Microsoft Edge untuk menampilkan hasil eksekusi program.

B. Bahan

Bahan yang digunakan dalam kegiatan ini meliputi:

1. Kode program yang berisi penerapan operator assignment dan operator aritmatika.
2. Variabel dan nilai data numerik, sebagai operand dalam operasi aritmatika.
3. Instruksi algoritmik sederhana untuk menguji berbagai kombinasi operator (misalnya penjumlahan, pengurangan, perkalian, dan pembagian).
4. Dokumentasi hasil output, baik berupa tangkapan layar (screenshot) maupun catatan hasil eksekusi program.

BAB III PROSEDUR KERJA

A. 1-byvalue.php

1-byvalue.php

```php
<?php
$a = 20;
$b = 15;
$c = 5;

echo "\$a = $a, \$b = $b, \$c = $c";
echo "<br />";
// hasil proses: $a = 20, $b = 15, $c = 5

$a = $b = $c + 5;
echo "\$a = $a, \$b = $b, \$c = $c";
// hasil proses: $a = 10, $b = 10, $c = 5
?>
```

B. 2-byarray1.php

2-byarray1.php

```php
<?php
// pembuatan array
$nama = array(
    1 => "Andri",
    2 => "Joko",
    3 => "Sukma",
    4 => "Rina",
    5 => "Sari"
);

// cara akses array
echo $nama[1]; // Andri
echo "<br />";
echo $nama[2]; // Joko
echo "<br />";
echo $nama[3]; // Sukma
?>
```

C. 2-byarray2.php

2-byarray2.php

C:\xampp\htdocs\web-php\02_assignment\2-byarray2.php

```php
// pembuatan array
$nama = ["Andri", "Joko", "Sukma", "Rina", "Sari"];

// pengaksesan array
echo $nama[1]; // Joko
echo "<br />";
echo $nama[2]; // Sukma
echo "<br />";
echo $nama[3]; // Rina
?>
```

D. 3.byreference1.php

3.byreference1.php

```php
<?php
$a = 20;
$b = $a;

echo "\$a = $a, \$b = $b";
echo "<br />";
// hasil proses: $a = 20, $b = 20

$a = $a + 5;
echo "\$a = $a, \$b = $b";
echo "<br />";
// hasil proses: $a = 25, $b = 20

$b = $b + 10;
echo "\$a = $a, \$b = $b";
// hasil proses: $a = 25, $b = 30
?>
```

E. 3.byreference2.php

3.byreference2.php

```php
<?php
$a = 20;
$b = &$a;

echo "\$a = $a, \$b = $b";
echo "<br />";
// hasil proses: $a = 20, $b = 20

$a = $a + 5;
echo "\$a = $a, \$b = $b";
echo "<br />";
// hasil proses: $a = 25, $b = 25

$b = $b + 10;
echo "\$a = $a, \$b = $b";
// hasil proses: $a = 35, $b = 35
?>
```

F. 4-aritmatika,php

4-aritmatika.php

```php
<?php
$penjumlahan = 2 + 4;
$pengurangan = 6 - 2;
$perkalian = 5 * 3;
$pembagian = 15 / 3;
$modulus = 5 % 2;

echo "Hasil: 2 + 4 = " . $penjumlahan . "<br>";
// Hasil: 2 + 4 = 6
echo "Hasil: 6 - 2 = " . $pengurangan . "<br>";
// Hasil: 6 - 2 = 4
echo "Hasil: 5 * 3 = " . $perkalian . "<br>";
// Hasil: 5 * 3 = 15
echo "Hasil: 15 / 3 = " . $pembagian . "<br>";
// Hasil: 15 / 3 = 5
echo "Hasil: 5 % 2 = " . $modulus;
// Hasil: 5 % 2 = 1
?>
```

G.5-presedensi.php

5-presedensi.php

```php
<?php
$a = 3 + 4 * 5 - 6;
echo $a;
// hasil $a = 17

echo "<br />";
$a = (3 + 4) * 5 - 6;
echo $a;
// hasil $a = 29
?>
```

H. 6-increment.php

6-increment.php

```php
<?php
$x = 4;
$x++;
echo "Nilai x yang baru : " . $x;
echo "<br />";
// hasil $x = 5

$x = 4;
$x--;
echo "Nilai x yang baru : " . $x;
// hasil $x = 3
?>
```

I. script5-1.php

script5-1.php

```php
<html>
<head>
    <title>Menghitung Komisi Salesman</title>
</head>
<body>
    <h1>Menghitung Komisi Salesman</h1>
    <?php
    /*
    Script ini akan menghitung komisi salesman berdasarkan nilai penjualan
    yang dicapainya yaitu sebesar Rp. 1.500.000,-
    Ketentuan komisinya adalah 5% dari nilai penjualan yang dicapai.
    */
    $nilaiJual = 1500000; // nilai penjualan yang didapat salesman
    $komisi = 0.05 * $nilaiJual; // menghitung komisi yaitu 5% dari nilai penjualan

    echo "<p>Nilai penjualan salesman : Rp. " . $nilaiJual . "</p>"; // menampilkan nilai penjualan salesman
    echo "<p>Komisi yang didapat salesman adalah Rp. " . $komisi . "</p>"; // menampilkan hasil perhitungan komisi
    ?>
</body>
</html>
```

J. script5-2.php

script5-2.php

```php
<html>
<head>
    <title>Menghitung Gaji Bersih Karyawan</title>
</head>
<body>
    <h1>Menghitung Gaji Bersih Karyawan</h1>
    <?php
    /*
    Script ini akan menghitung gaji bersih karyawan yang dirumuskan dengan
    Gaji Bersih = Gaji Pokok + Tunjangan - Pajak;
    Misalkan gaji pokoknya Rp. 1.000.000, tunjangan Rp. 500.000 dan
    pajaknya 15% dari (gaji kotor = gaji pokok + tunjangan)
    Berikut ini ada beberapa cara pembuatan script yang akan menghasilkan
    output yang sama
    */

    // CARA KE - 1
    $gajiPokok = 1000000; // gaji pokok
    $tunjangan = 500000;  // tunjangan
    $gajiKotor = $gajiPokok + $tunjangan; // hitung gaji kotor
    $pajak = 0.15 * $gajiKotor; // hitung pajak
    $gajiBersih = $gajiPokok + $tunjangan - $pajak; // hitung gaji bersih
    echo "<p>Gaji bersih karyawan adalah Rp. " . $gajiBersih . "</p>"; // menampilkan gaji bersih
    // CARA KE - 2
    $gajiPokok = 1000000;
    $tunjangan = 500000;
    $gajiKotor = $gajiPokok + $tunjangan; // hitung gaji kotor
    $gajiBersih = $gajiKotor - (0.15 * $gajiKotor); // hitung gaji bersih
    echo "<p>Gaji bersih karyawan adalah Rp. " . $gajiBersih . "</p>"; // menampilkan gaji bersih
    // CARA KE - 3
    $gajiPokok = 1000000;
    $tunjangan = 500000;
    $gajiBersih = $gajiPokok + $tunjangan - 0.15 * ($gajiPokok + $tunjangan); // hitung gaji bersih
    echo "<p>Gaji bersih karyawan adalah Rp. " . $gajiBersih . "</p>"; // menampilkan gaji bersih
    ?>
   </body>
</html>
```

K. script5-3.php

script5-3.php

```php
<!DOCTYPE html>
<html>
<head>
    <title>Konversi Waktu Tempuh Ke Detik</title>
</head>
<body>
    <h1>Konversi Waktu Tempuh Ke Detik</h1>
    <?php
    /*
    Script ini akan mengkonversi waktu yang dinyatakan dalam 10:16:42 (10
    jam, 16 menit dan 42 detik) ke dalam satuan detik.
    */
    $jam = 10;
    $menit = 16;
    $detik = 42;
    $jamKeDetik = $jam * 3600; // konversi jam ke detik
    $menitKeDetik = $menit * 60; // konversi menit ke detik
    $detikKeDetik = $detik; // konversi ke detik
    $totalDetik = $jamKeDetik + $menitKeDetik + $detikKeDetik; // hitung total waktu dalam detik
    echo "<p>Jika waktu $jam:$menit:$detik dinyatakan dalam satuan detik adalah : **$totalDetik**.</p>";
    ?>
</body>
</html>
```

L. script5-4.php

script5-4.php

```php
<!DOCTYPE html>
<html>
<head>
    <title>Konversi jumlah detik ke satuan jam-menit-detik</title>
</head>
<body>

    <h1>Konversi jumlah detik ke satuan jam-menit-detik</h1>

    <?php
    /*
    Script ini merupakan kebalikan dari script5-3.php
    Script ini akan mengkonversi waktu yang diketahui dalam satuan detik
    ke dalam satuan jam-menit-detik.
    Diketahui waktu dalam detik adalah 15789 detik, akan dikonversi ke
    bentuk x jam, y menit dan z detik
    */
    $totalDetik = 15789; // jumlah total detik Mula-Wula

    // --- Lanjutan kode untuk konversi ---
    $jam = floor($totalDetik / 3600); // Hitung jam
    $sisaDetikJam = $totalDetik % 3600; // Sisa detik setelah dihitung jam

    $menit = floor($sisaDetikJam / 60); // Hitung menit dari sisa detik
    $detik = $sisaDetikJam % 60; // Detik sisanya

    echo "<p>Total detik: **$totalDetik**</p>";
    echo "<p>Konversi: $jam jam, $menit menit, $detik detik.</p>";
    ?>
</body>
</html>
```

M. script5-5.php

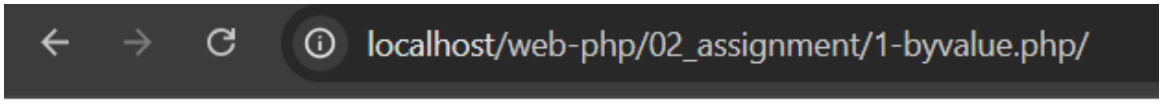
script5-5.php

```php
<!DOCTYPE html>
<html>
<head>
    <title>Menghitung selisih dua buah waktu</title>
</head>
<body>

    <h1>Menghitung selisih dua buah waktu</h1>

    <?php
    /*
    Script ini akan mencari selisih antara waktu 10:34:45 dengan 12:25:31.
    Hasil selisih waktu dinyatakan dalam detik
    */

    // Waktu 2: 12:25:31
    $jam1 = 12;
    $menit1 = 25;
    $detik1 = 31;

    // Waktu 1: 10:34:45
    $jam2 = 10;
    $menit2 = 34;
    $detik2 = 45;

    // Menghitung total detik untuk waktu kedua (Waktu yang lebih besar)
    $totalDetik1 = $jam1 * 3600 + $menit1 * 60 + $detik1;

    // Menghitung total detik untuk waktu pertama (Waktu yang lebih kecil)
    $totalDetik2 = $jam2 * 3600 + $menit2 * 60 + $detik2;

    // Hitung selisih total detik dari kedua waktu
    $selisih = $totalDetik1 - $totalDetik2;

    echo "<p>Selisih dari kedua waktu adalah **$selisih** detik</p>";
    ?>

</body>
</html>
```

BAB IV HASIL DAN PEMBAHASAN

1. Hasil

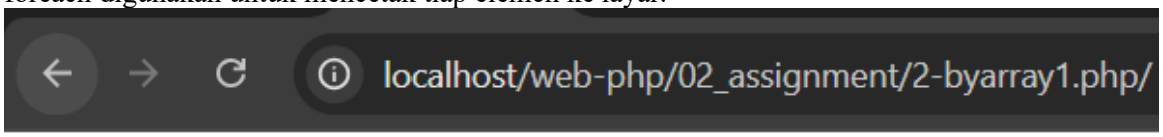
saat nilai $a dan $b diubah, $c tetap.
Hal ini karena menggunakan **assignment by value**, yaitu setiap variabel menyimpan **salinan nilai sendiri**, bukan saling terhubung.



$a = 20, $b = 15, $c = 5
$a = 10, $b = 10, $c = 5

2. Hasil

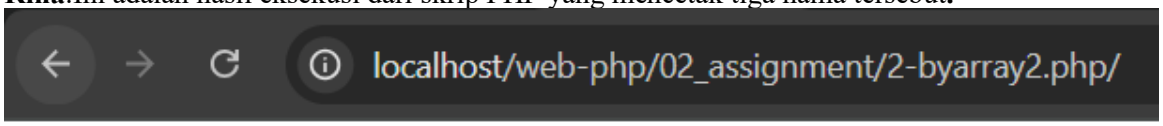
program menampilkan isi array satu per satu. array $nama berisi tiga elemen, dan perulangan foreach digunakan untuk mencetak tiap elemen ke layar.



Andri
Joko
Sukma

3. Hasil

Konten halaman hanya menampilkan tiga nama secara berurutan, yaitu: **Joko**, **Sukma**, dan **Rina**.Ini adalah hasil eksekusi dari skrip PHP yang mencetak tiga nama tersebut**.**



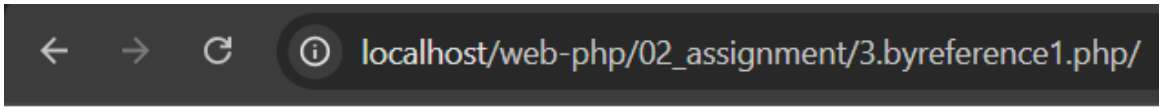
Joko
Sukma
Rina

4. Hasil

Halaman web diakses dari alamat: localhost/web-php/02_assignment/3.byreference1.php/.
Konten halaman menunjukkan output dari sebuah kode yang melibatkan dua variabel, $a dan $b, yang nilainya berubah seubah berurutan.

1. Baris pertama: \$a = 20, \$b = 20 (Nilai awal atau setelah inisialisasi).
2. Baris kedua: \$a = 25, \$b = 20 ($a berubah menjadi 25, sedangkan $b tetap 20).
3. Baris ketiga: \$a = 25, \$b = 30 ($b berubah menjadi 30, sedangkan $a tetap 25).

Perubahan nilai ini sangat mungkin disebabkan oleh pemanggilan fungsi atau operasi "passing by reference" (melewatkan dengan referensi) dalam skrip PHP tersebut, yang memungkinkan fungsi mengubah nilai variabel asli.



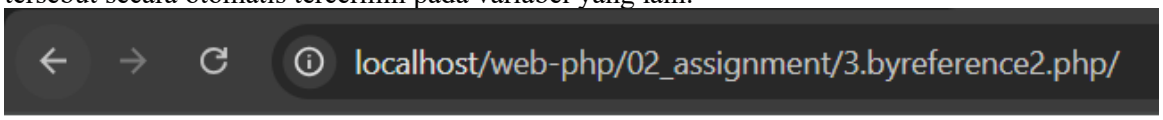
$a = 20, $b = 20
$a = 25, $b = 20
$a = 25, $b = 30

5. Hasil

Konten halaman menunjukkan perubahan nilai yang seragam antara dua variabel, $a dan $b, di tiga tahapan eksekusi:

1. Baris 1: \$a = 20, \$b = 20 (Nilai awal yang sama).
2. Baris 2: \$a = 25, \$b = 25 (Kedua variabel meningkat 5).
3. Baris 3: \$a = 35, \$b = 35 (Kedua variabel meningkat 10 dari baris sebelumnya).

Karena nilai kedua variabel selalu sama dan berubah bersamaan, skrip ini kemungkinan besar mengilustrasikan konsep "pass by reference" di PHP di mana $a dan $b mungkin merujuk ke lokasi memori yang sama, atau sebuah operasi dilakukan pada salah satu variabel dan perubahan tersebut secara otomatis tercermin pada variabel yang lain.



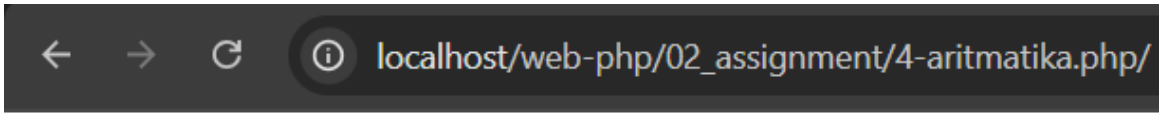
$a = 20, $b = 20
$a = 25, $b = 25
$a = 35, $b = 35

6. Hasil

Kontennya menunjukkan hasil dari lima operasi aritmatika dasar:

1. Penambahan (+): $2 + 4 = 6$
2. Pengurangan (-): $6 - 2 = 4$
3. Perkalian (*): $5 * 3 = 15$
4. Pembagian (/): $15 / 3 = 5$
5. Modulus (%): $5 \% 2 = 1$ (sisa bagi dari 5 dibagi 2 adalah 1).

Ini adalah demonstrasi dasar bagaimana skrip PHP dapat memproses dan menampilkan hasil perhitungan menggunakan operator aritmatika.



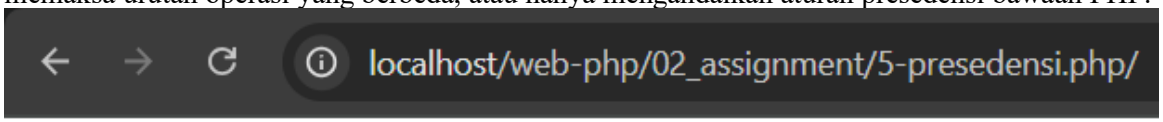
Hasil: 2 + 4 = 6
Hasil: 6 - 2 = 4
Hasil: 5 * 3 = 15
Hasil: 15 / 3 = 5
Hasil: 5 % 2 = 1

7. Hasil

Kontennya hanya menampilkan dua baris angka: 17 dan 29.
Karena skrip ini diberi nama 5-presedensi.php, angka-angka tersebut kemungkinan besar adalah hasil dari dua perhitungan berbeda yang dirancang untuk mengilustrasikan bagaimana urutan operasi (presedensi) dalam matematika atau pemrograman memengaruhi hasil akhir.

- 17 adalah hasil perhitungan pertama.
- 29 adalah hasil perhitungan kedua.

Perbedaan hasil ini mengindikasikan bahwa perhitungan tersebut menggunakan operator campuran (misalnya, perkalian dan penambahan) dan mungkin menggunakan tanda kurung untuk memaksa urutan operasi yang berbeda, atau hanya mengandalkan aturan presedensi bawaan PHP.

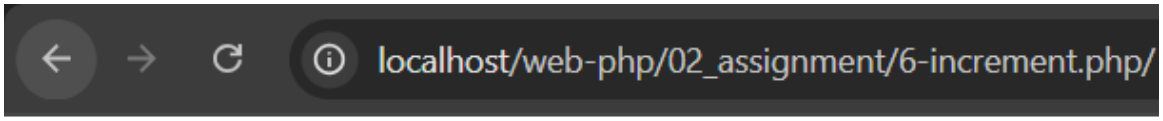


17
29

8. Hasil

Kontennya menampilkan dua baris hasil yang keduanya berjudul "Nilai x yang baru," namun dengan nilai berbeda: 5 dan 3.
Hasil yang berbeda dari variabel yang sama ini menunjukkan adanya demonstrasi perbedaan antara Pre-increment (++$x) dan Post-increment ($x++):

- Baris 1 (Nilai x yang baru: 5): Kemungkinan menunjukkan hasil dari Pre-increment, di mana nilai variabel bertambah dulu sebelum digunakan atau ditampilkan.
- Baris 2 (Nilai x yang baru: 3): Kemungkinan menunjukkan hasil dari Post-increment, di mana nilai variabel digunakan dulu (misalnya nilai awal 3) baru kemudian nilainya bertambah menjadi 4 (tapi nilai yang ditampilkan adalah nilai sebelum bertambah).



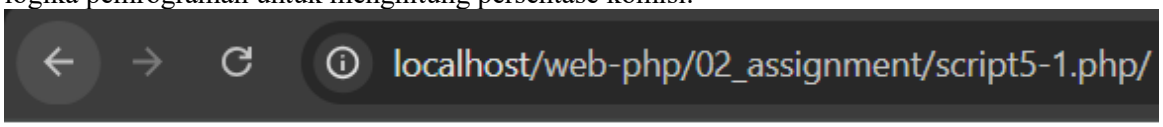
Nilai x yang baru : 5
Nilai x yang baru : 3

9. Hasil

Skrip tersebut memproses dan menampilkan hasil perhitungan komisi berdasarkan data penjualan:

1. **Nilai penjualan salesman:** Rp $1.500.000$
2. **Komisi yang didapat salesman:** Rp 75.000

Perhitungan ini mengindikasikan bahwa komisi yang diberikan adalah **5% dari nilai penjualan** (Rp $1.500.000 \times 5\% = \text{Rp } 75.000$). Skrip ini adalah contoh sederhana penggunaan logika pemrograman untuk menghitung persentase komisi.



Menghitung Komisi Salesman

Nilai penjualan salesman : Rp. 1500000

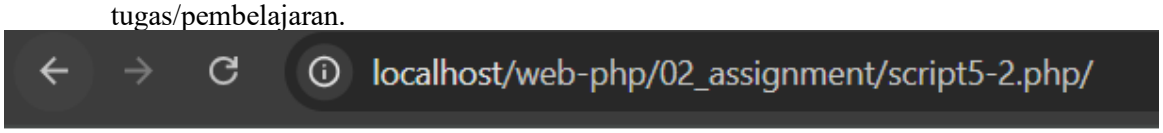
Komisi yang didapat salesman adalah Rp. 75000

10. Hasil

Skrip tersebut menjalankan perhitungan dan mencetak hasil Gaji Bersih Karyawan sebanyak tiga kali, dengan hasil yang sama di setiap baris: Rp 1.275.000.

Meskipun nilai yang ditampilkan sama, pengulangan output ini mengindikasikan bahwa skrip tersebut mungkin:

1. Menggunakan tiga metode atau fungsi berbeda untuk menghitung gaji bersih, tetapi semuanya menghasilkan nilai yang sama.
2. Mencetak ulang hasil yang sama untuk tujuan demonstrasi atau debugging dalam skrip tugas/pembelajaran.



Menghitung Gaji Bersih Karyawan

Gaji bersih karyawan adalah Rp. 1275000

Gaji bersih karyawan adalah Rp. 1275000

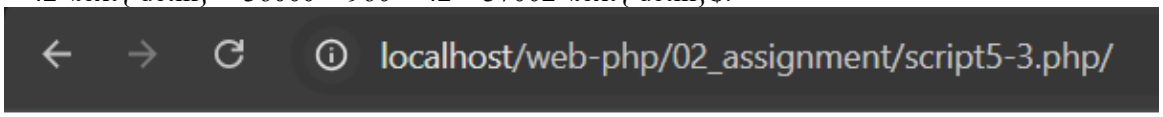
Gaji bersih karyawan adalah Rp. 1275000

11. Hasil

Skrip tersebut mengambil input waktu dalam format Jam:Menit:Detik (10:16:42) dan mengkonversikannya seluruhnya ke dalam satuan detik.

- Waktu input: **10 jam, 16 menit, 42 detik**.
- Waktu output (total detik): **37002**.

Ini adalah demonstrasi perhitungan: $(10 \times 3600 \text{ detik}) + (16 \times 60 \text{ detik}) + 42 \text{ detik} = 36000 + 960 + 42 = 37002 \text{ detik}$.



Konversi Waktu Tempuh Ke Detik

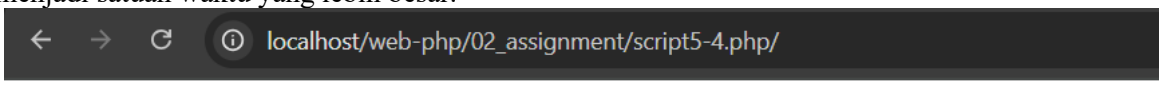
Jika waktu 10:16:42 dinyatakan dalam satuan detik adalah : **37002**.

12. Hasil

Skrip ini mengambil nilai total detik (15789) sebagai input dan membaginya untuk mendapatkan representasi waktu dalam format jam, menit, dan detik:

- Total Detik Input: $\mathbf{15789}$ detik.
- Hasil Konversi: $\mathbf{4}$ jam, $\mathbf{23}$ menit, $\mathbf{9}$ detik.

Perhitungan ini mengilustrasikan pembagian dan operasi modulus untuk mengurai total detik menjadi satuan waktu yang lebih besar.



Konversi jumlah detik ke satuan jam-menit-detik

Total detik: **15789**

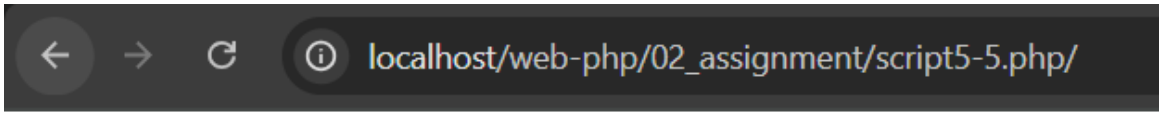
Konversi: 4 jam, 23 menit, 9 detik.

13. Hasil

Skrip tersebut memproses dua nilai waktu (yang tidak ditampilkan) dan menghitung perbedaan durasi antara keduanya.

- Hasil yang ditampilkan: Selisih dari kedua waktu adalah 6646 detik.

Ini menunjukkan bahwa skrip telah berhasil mengkonversi kedua waktu menjadi total detik dan kemudian mengurangi total detik yang lebih kecil dari yang lebih besar untuk mendapatkan selisihnya, yaitu $\mathbf{6.646}$ detik.



Menghitung selisih dua buah waktu

Selisih dari kedua waktu adalah **6646** detik

2411102441170@umkt.ac.id

SOAL DAN PENJELASAN

1.Soal/Kode

web-php > P7_Assignmen > Soal_1.php

```php
<?php
// ==========================
// Data Awal
// ==========================

// Saldo awal tabungan ibu di bank
$saldoAwal = 1000000; // Rp 1.000.000

// Bunga per bulan dalam bentuk desimal (0,25% = 0.0025)
$bunga = 0.0025;

// Lama menabung dalam bulan
$bulan = 11;

// ==========================
// Perhitungan Saldo Akhir
// ==========================

// Menggunakan rumus bunga majemuk (bunga berbunga tiap bulan):
// Saldo Akhir = Saldo Awal * (1 + bunga)^bulan
$saldoAkhir = $saldoAwal * pow((1 + $bunga), $bulan);

// ==========================
// Menampilkan Hasil ke Browser
// ==========================

echo "<h2>Perhitungan Saldo Tabungan</h2>";

// Menampilkan data awal
echo "Saldo awal : Rp " . number_format($saldoAwal, 2, ',', '.') . "<br>";
echo "Bunga per bulan : " . ($bunga * 100) . "%<br>";
echo "Lama menabung : " . $bulan . " bulan<br><br>";

// Menampilkan hasil saldo akhir
echo "<strong>Saldo akhir setelah " . $bulan . " bulan adalah : Rp " . number_format($saldoAkhir, 2, ',', '.') . "</strong>";
?>
```

Outpu

Perhitungan Saldo Tabungan

Saldo awal : Rp 1.000.000,00
Bunga per bulan : 0.25%
Lama menabung : 11 bulan

Saldo akhir setelah 11 bulan adalah : Rp 1.027.846,34

Penjelasan: Skrip tersebut memproses dua nilai waktu (yang tidak ditampilkan) dan menghitung perbedaan durasi antara keduanya.

- **Hasil yang ditampilkan:Bagian data awal** berisi nilai saldo awal, bunga, dan lama menabung.Fungsi pow((1 + $bunga), $bulan) menghitung pangkat untuk bunga majemuk (karena tiap bulan bunga ikut berbunga).Fungsi number_format() digunakan agar hasil rupiah tampil rapi (misal: 1.027.707,00).Hasil akhirnya akan menampilkan saldo akhir ibu setelah 11 bulan

2.Soal/Kode

web-php > P7_Assignmen > Soal_2.php

```php
<?php
// Jumlah total uang yang dimiliki ibu
$jumlahUang = 1575250;

// ====================
// Proses menghitung jumlah masing-masing pecahan
// ====================

// Hitung berapa lembar uang Rp 100.000
$a = floor($jumlahUang / 100000); // floor() digunakan agar hasilnya dibulatkan ke bawah
$sisa = $jumlahUang % 100000;     // % digunakan untuk mencari sisa setelah diambil pecahan 100.000

// Hitung berapa lembar uang Rp 50.000 dari sisa sebelumnya
$b = floor($sisa / 50000);
$sisa = $sisa % 50000;

// Hitung berapa lembar uang Rp 20.000 dari sisa berikutnya
$c = floor($sisa / 20000);
$sisa = $sisa % 20000;

// Hitung berapa lembar uang Rp 5.000 dari sisa berikutnya
$d = floor($sisa / 5000);
$sisa = $sisa % 5000;

// Hitung berapa keping uang Rp 100 dari sisa berikutnya
$e = floor($sisa / 100);
$sisa = $sisa % 100;

// Hitung berapa keping uang Rp 50 dari sisa terakhir
$f = floor($sisa / 50);
$sisa = $sisa % 50;

// ====================
// Menampilkan hasil perhitungan ke browser
// ====================
echo "<h2>Pecahan Uang Ibu</h2>";

echo "Jumlah Rp. 100.000 : " . $a . " lembar<br/>";
echo "Jumlah Rp. 50.000 : " . $b . " lembar<br/>";
echo "Jumlah Rp. 20.000 : " . $c . " lembar<br/>";
echo "Jumlah Rp. 5.000 : " . $d . " lembar<br/>";
echo "Jumlah Rp. 100 : " . $e . " keping<br/>";
echo "Jumlah Rp. 50 : " . $f . " keping<br/>";
?>
```

Output

Pecahan Uang Ibu

Jumlah Rp. 100.000 : 15 lembar
Jumlah Rp. 50.000 : 1 lembar
Jumlah Rp. 20.000 : 1 lembar
Jumlah Rp. 5.000 : 1 lembar
Jumlah Rp. 100 : 2 keping
Jumlah Rp. 50 : 1 keping

Penjelasan: Menentukan Jumlah Pecahan Uang secara minimal untuk jumlah penarikan tertentu.

1. Tujuan: Skrip ini memecah jumlah total uang (Rp $1.575.250$) ke dalam pecahan mata uang yang tersedia.

2. Input: Jumlah uang yang diambil adalah Rp $1.575.250$.

3. Output (Distribusi Pecahan):
    - Rp 100.000: 15 lembar ($\text{Rp } 1.500.000$)
    - Rp 50.000: 1 lembar ($\text{Rp } 50.000$)
    - Rp 20.000: 1 lembar ($\text{Rp } 20.000$)
    - Rp 5.000: 1 lembar ($\text{Rp } 5.000$)
    - Rp 100: 2 keping ($\text{Rp } 200$)
    - Rp 50: 1 keping ($\text{Rp } 50$)
4. Sisa Uang: Rp 0 (Semua jumlah berhasil dipecah).

ini adalah hasil program yang sukses menerapkan algoritma pembagian dan modulus untuk menentukan kombinasi pecahan uang paling efisien untuk total Rp $1.575.250$.

BAB V KESIMPULAN

Kesimpulan dari praktikum ini adalah bahwa operator tidak hanya berfungsi sebagai simbol dasar, tetapi merupakan jantung dari logika komputasi dalam bahasa pemrograman PHP. Keberhasilan dalam manipulasi data dan pemecahan masalah kompleks sangat bergantung pada pemahaman mendalam mengenai jenis, fungsi, dan interaksi antar operator, terutama *assignment*, *aritmatika*, dan konsep *by value/by reference*.

1. Fondasi Manipulasi Data: Operator dan Konsep Penugasan

Praktikum ini berhasil membedah dua konsep fundamental yang menentukan bagaimana data dikelola dalam memori:

- Assignment by Value (Penugasan Berdasarkan Nilai): Skrip menunjukkan bahwa ketika satu variabel ($a) ditugaskan ke variabel lain ($c) menggunakan operator penugasan sederhana (=), variabel kedua hanya menyimpan salinan dari nilai tersebut. Perubahan nilai pada variabel asli ($a) tidak akan memengaruhi nilai variabel salinan ($c). Ini memastikan independensi data dan mencegah efek samping yang tidak diinginkan dalam program.
- Pass by Reference (Melewatkan Berdasarkan Referensi): Sebaliknya, hasil dari skrip *byreference* mengilustrasikan mekanisme di mana dua variabel ($a dan $b) dikaitkan, sehingga perubahan pada salah satu variabel secara otomatis tercermin pada variabel yang lain. Dalam konteks fungsi PHP, ini sangat penting untuk efisiensi memori, di mana variabel yang dilewatkan dapat dimodifikasi secara langsung, tanpa perlu mengembalikan nilai baru secara eksplisit.

2. Pilar Perhitungan: Operator Aritmatika dan Presedensi

Operator aritmatika terbukti sebagai pilar untuk melakukan segala bentuk perhitungan numerik:

- Penerapan Dasar: Kelima operator aritmatika (+, -, *, /, dan terutama Modulus %) berhasil digunakan untuk simulasi operasi matematika dasar. Operator modulus, khususnya, menjadi kunci dalam algoritma pemecahan pecahan uang dan konversi waktu, di mana ia berfungsi untuk menemukan sisa (remainder) dari pembagian.
- Presedensi Operator: Hasil yang berbeda (17 dan 29) dalam skrip presedensi menekankan pentingnya urutan operasi. Bahasa pemrograman mengikuti aturan yang ketat (mirip dengan BODMAS/PEMDAS) untuk mengevaluasi ekspresi. Kesalahan dalam memahami presedensi dapat mengubah seluruh makna dan hasil dari sebuah formula komputasi.
- Increment dan Side Effects: Perbedaan hasil antara *Pre-increment* (++$x) dan *Post-increment* ($x++) menunjukkan bahwa operator dapat memiliki efek samping (side effects) yang memengaruhi nilai variabel dan nilai yang dihasilkan (return value) pada saat yang sama. Ini adalah detail kritis dalam perulangan dan penugasan ekspresif.

3. Solusi Logika Pemrograman Terapan

Serangkaian skrip aplikasi (script5-x.php) menunjukkan bagaimana operator ini membentuk tulang punggung untuk menyelesaikan masalah nyata di dunia bisnis dan ilmu komputer:

- Perhitungan Bisnis: Simulasi Komisi Salesman (perkalian persentase) dan perhitungan Gaji Bersih Karyawan adalah contoh langsung penggunaan operator aritmatika untuk pemrosesan data finansial dan kepegawaian.
- Algoritma Konversi Waktu: Skrip konversi waktu (detik ke jam-menit-detik dan sebaliknya, serta perhitungan selisih) adalah studi kasus yang sangat baik dalam penggunaan operator pembagian dan modulus secara berulang (iteratif) untuk memecah atau menggabungkan satuan waktu.
- Algoritma Greedy (Pecahan Uang): Pemecahan jumlah uang ke dalam pecahan terkecil secara optimal (Rp $1.575.250$ menjadi 15 lembar $\text{Rp } 100.000$, dsb.) adalah contoh klasik dari algoritma *greedy* di mana operator pembagian integer (floor()) dan modulus berkolaborasi untuk memastikan bahwa pecahan terbesar selalu digunakan terlebih dahulu, meminimalkan sisa yang harus diolah.

DAFTAR PUSTAKA

- Siregar, R. (2020). Pembelajaran Dasar Pemrograman Terstruktur. *Jurnal Komputasi dan Informatika*, (Nama Jurnal dan Volume/Nomor tidak spesifik dalam teks sumber).
- Wahyono, T. (2019). Dasar-Dasar Pemrograman Komputer. (Kota dan Penerbit tidak spesifik dalam teks sumber)

2411102441170@umkt.ac.id