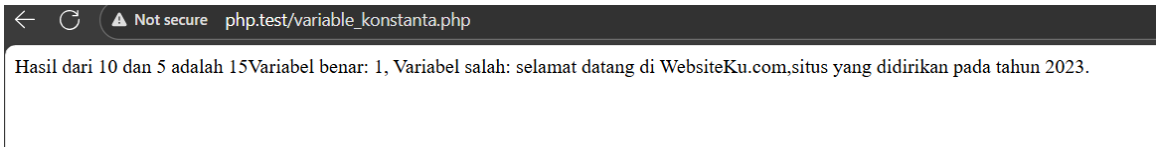
PEMROGRAMAN WEB
Jobsheet 4

Nama : Maulana Ahmad Bukhori
NIM : 244107060133
Kelas : SIB-2C
No.Absen : 13

D IV –SISTEM INFORMASI BISNIS
JURUSAN TEKNIK INFORMATIKA
POLITEKNIK NEGERI MALANG
JL. Soekarno Hatta No.9 Kota Malang Kode Pos: 65141
2025

SOAL 1

Output



Dalam percobaan ini, saya memahami bahwa variabel dalam PHP berfungsi sebagai "wadah" untuk menyimpan berbagai jenis data, seperti angka dan teks. Saya juga belajar cara mendeklarasikan variabel menggunakan simbol $ dan memberikan nilai padanya dengan tanda =. Selain itu, saya melihat bagaimana variabel dapat digunakan kembali dalam perhitungan dan ditampilkan hasilnya menggunakan perintah echo di dalam string. Hal ini menunjukkan fleksibilitas variabel dalam menyimpan dan mengolah informasi secara dinamis. pharafrase kan

SOAL 2

Output

```
Variable a : 10
Variable b : 5
Variable c : 15
Variable d : 55
Variable e : 40
int(40)
Matematika : 5.1
IPA : 6.7
Bahasa Indonesia : 9.3
Rata-rata : 7.0333333333333
float(7.033333333333334)
bool(true)
bool(false)
Nama depan : Ibnu
nama belakang : Jakaria
Ibnu Jakaria
Wahid Abdullah
```

Melalui kode yang dieksekusi, saya dapat mengidentifikasi beberapa jenis tipe data:

- **Integer** (int): Digunakan untuk menyimpan angka bulat, seperti pada variabel $a dan $b. Ini sangat berguna untuk operasi hitungan dasar.
- **Float** (float): Digunakan untuk angka desimal, seperti nilai-nilai mata pelajaran. Tipe ini penting untuk perhitungan yang membutuhkan ketelitian, seperti menghitung rata-rata.
- **Boolean** (bool): Digunakan untuk menyimpan nilai logis **benar** (true) atau **salah** (false). Ini sangat mendasar untuk menentukan kondisi atau membuat keputusan dalam kode.
- **String** (string): Digunakan untuk menyimpan data berupa teks, seperti nama depan atau nama lengkap.
- **Array**: Tipe data ini mampu menampung lebih dari satu nilai dalam satu variabel. Pada kode, $listMahasiswa adalah contohnya, yang menyimpan beberapa nama mahasiswa.

PHP akan secara otomatis menentukan tipe data variabel berdasarkan nilai yang kita berikan. Kemampuan ini membuat program dapat mengolah berbagai jenis data, dari angka, teks, hingga logika, secara fleksibel dan efisien.

SOAL 3.1

Output

Hasil Tambah : 15
Hasil Kurang : 5
Hasil Kali : 50
Hasil Bagi : 2
Sisa Bagi : 0
Pangkat : 100000

Kode untuk melengkapi agar dapat menampilkan hasil

```
echo "Hasil Tambah : $hasilTambah <br>";
echo "Hasil Kurang : $hasilKurang <br>";
echo "Hasil Kali : $hasilKali <br>";
echo "Hasil Bagi : $hasilBagi <br>";
echo "Sisa Bagi : $sisaBagi <br>";
echo "Pangkat : $pangkat <br>";
```

Dari percobaan ini, saya mengamati bahwa **PHP** dapat digunakan untuk melakukan operasi matematika dasar. Setiap operator memiliki fungsinya masing-masing:

- + **(tambah)**: Menjumlahkan dua angka.
- **- (kurang)**: Mengurangi satu angka dengan angka lainnya.
- *** (kali)**: Mengalikan dua angka.
- **/ (bagi)**: Membagi satu angka dengan angka lainnya.
- **% (modulus)**: Menghasilkan sisa dari operasi pembagian.
- **** (pangkat)**: Menghasilkan hasil pemangkatan.

Kode ini juga menunjukkan cara menampilkan teks dan hasil variabel secara bersamaan di web menggunakan perintah echo dan tag HTML
 untuk membuat baris baru, sehingga tampilannya menjadi rapi dan mudah dibaca.

SOAL 3.2

Output

Hasil Sama : bool(false)
Hasil Tidak Sama : bool(true)
Hasil Lebih Kecil : bool(false)
Hasil Lebih Besar : bool(true)
Hasil Lebih Kecil Sama : bool(false)
Hasil Lebih Besar Sama : bool(true)

Kode untuk melengkapi agar dapat menampilkan hasil

```
echo "Hasil Sama : "; var_dump(value: $hasilSama); echo "<br>";
echo "Hasil Tidak Sama : "; var_dump(value: $hasilTidakSama); echo "<br>";
echo "Hasil Lebih Kecil : "; var_dump(value: $hasilLebihKecil); echo "<br>";
echo "Hasil Lebih Besar : "; var_dump(value: $hasilLebihBesar); echo "<br>";
echo "Hasil Lebih Kecil Sama : "; var_dump(value: $hasilLebihKecilSama); echo "<br>";
echo "Hasil Lebih Besar Sama : "; var_dump(value: $hasilLebihBesarSama); echo "<br>";
```

Dari percobaan ini, saya mengamati bahwa **operator perbandingan** di PHP digunakan untuk membandingkan dua nilai dan menghasilkan nilai **boolean**, yaitu true (benar) atau false (salah).

- == (Sama dengan): Membandingkan apakah $a sama dengan $b (10 == 5). Hasilnya **salah**, karena nilainya tidak sama.
- != (Tidak sama dengan): Membandingkan apakah $a tidak sama dengan $b (10 != 5). Hasilnya **benar**, karena 10 memang tidak sama dengan 5.
- < (Lebih kecil dari): Membandingkan apakah $a lebih kecil dari $b (10 < 5). Hasilnya **salah**.
- > (Lebih besar dari): Membandingkan apakah $a lebih besar dari $b (10 > 5). Hasilnya **benar**.
- <= (Lebih kecil dari atau sama dengan): Membandingkan apakah $a lebih kecil dari atau sama dengan $b (10 <= 5). Hasilnya **salah**.
- >= (Lebih besar dari atau sama dengan): Membandingkan apakah $a lebih besar dari atau sama dengan $b (10 >= 5). Hasilnya **benar**.

SOAL 3.3

Output

```
Hasil And : bool(false)
Hasil Or : bool(true)
Hasil Not A : bool(false)
Hasil Not B : bool(false)
```

Kode untuk melengkapi agar dapat menampilkan hasil

```
echo "Hasil And : "; var_dump(value: $hasilAnd); echo "<br>";
echo "Hasil Or : "; var_dump(value: $hasilOr); echo "<br>";
echo "Hasil Not A : "; var_dump(value: $hasilNotA); echo "<br>";
echo "Hasil Not B : "; var_dump(value: $hasilNotB); echo "<br";
```

Dari percobaan ini, dapat di amati cara kerja **operator logika** di PHP, yang membandingkan dua ekspresi atau nilai untuk menghasilkan nilai boolean (true atau false). Dalam kasus ini, nilai non-nol seperti 10 dan 5 dievaluasi sebagai true dalam konteks logis.

- **&& (AND):** Operator ini menghasilkan **true** hanya jika **kedua** nilai yang dibandingkan bernilai true. Karena $a (10) dan $b (5) keduanya dianggap true, maka $hasilAnd bernilai true.
- **|| (OR):** Operator ini menghasilkan **true** jika **salah satu** dari nilai yang dibandingkan bernilai true. Karena $a dan $b keduanya true, maka $hasilOr juga bernilai true.

- **! (NOT):** Operator ini membalikkan nilai boolean. Karena $a (10) dianggap true dalam konteks logis, maka !$a (NOT true) menghasilkan **false**. Hal yang sama berlaku untuk $hasilNotB.

SOAL 3.4

Output

```
Hasil a += b : 0
Hasil a -= b : 5
Hasil a *= b : 50
Hasil a /= b : 2
Hasil a %= b : 0
```

Kode untuk melengkapi agar dapat menampilkan hasil

```php
// Agar setiap operasi menunjukkan hasil yang benar,
// nilai $a perlu direset ke 10 sebelum setiap operasi penugasan.
echo "Hasil a += b : $a <br>";
$a = 10; // Reset nilai $a
$a -= $b;
echo "Hasil a -= b : $a <br>";
$a = 10; // Reset nilai $a
$a *= $b;
echo "Hasil a *= b : $a <br>";
$a = 10; // Reset nilai $a
$a /= $b;
echo "Hasil a /= b : $a <br>";
$a = 10; // Reset nilai $a
$a %= $b;
echo "Hasil a %= b : $a <br>";
?>
```

Dari percobaan ini, saya mengamati penggunaan operator penugasan (assignment operators) dalam PHP. Operator ini digunakan untuk melakukan operasi matematika dan langsung menetapkan hasilnya kembali ke variabel yang sama.

- += (Penambahan): Menambahkan nilai $b ke $a dan menyimpannya kembali ke $a. (10 + 5 = 15).
- -= (Pengurangan): Mengurangi nilai $b dari $a dan menyimpannya kembali ke $a. (10 - 5 = 5).
- *= (Perkalian): Mengalikan $a dengan $b dan menyimpannya kembali ke $a. (10 * 5 = 50).
- /= (Pembagian): Membagi $a dengan $b dan menyimpannya kembali ke $a. (10 / 5 = 2).
- %= (Modulus): Menghitung sisa pembagian $a oleh $b dan menyimpannya kembali ke $a. (10 % 5 = 0).

SOAL 3.5

Output

Hasil Identik : bool(false)
Hasil Tidak Identik : bool(true)

Kode untuk melengkapi agar dapat menampilkan hasil

```
echo "Hasil Identik : "; var_dump(value: $hasilIdentik); echo "<br>";
echo "Hasil Tidak Identik : "; var_dump(value: $hasilTidakIdentik); echo "<br>";
```

Dari percobaan ini, saya mengamati cara kerja operator identik (===) dan operator tidak identik (!==).

- === (Identik): Operator ini membandingkan apakah nilai dan tipe data dari kedua variabel sama persis. Karena $a (integer 10) dan $b (integer 5) memiliki nilai yang berbeda, maka hasilnya false.
- !== (Tidak Identik): Operator ini memeriksa apakah nilai atau tipe data dari kedua variabel berbeda. Karena $a dan $b memiliki nilai yang berbeda, maka hasilnya true.

Pengamatan ini menunjukkan bahwa meskipun operator identik lebih ketat dalam membandingkan (mempertimbangkan tipe data), dalam kasus ini hasilnya sama dengan operator perbandingan biasa (== dan !=) karena nilai kedua variabel memang sudah berbeda.

SOAL 3.6

```
//Sebuah restoran memiliki 45 kursi di dalamnya. Pada suatu malam,
// 28 kursi telah ditempati oleh pelanggan. Berapa persen kursi yang masih kosong di restoran tersebut
$totalKursi = 45;
$kursiTerisi = 28;
$kursiKosong = $totalKursi - $kursiTerisi;
$persenKursiKosong = ($kursiKosong / $totalKursi) * 100;

echo "Total kursi: $totalKursi <br>";
echo "Kursi terisi: $kursiTerisi <br>";
echo "Kursi kosong: $kursiKosong <br>";
echo "Persentase kursi yang kosong: " . $persenKursiKosong . "%";
?>
```

Output

Total kursi: 45
Kursi terisi: 28
Kursi kosong: 17
Persentase kursi yang kosong: 37.777777777778%

SOAL 4.1

Output

> Nilai huruf: A

Dari percobaan ini, saya mengamati bagaimana **struktur kontrol if-elseif-else** bekerja untuk membuat keputusan dalam program.

1. Program menguji setiap kondisi dari atas ke bawah.
2. Karena variabel $nilaiNumerik disetel ke 92, kondisi pertama yaitu ($nilaiNumerik >= 90 && $nilaiNumerik <= 100) dievaluasi sebagai true.
3. Ketika suatu kondisi terpenuhi (bernilai true), program akan mengeksekusi blok kode di dalam if tersebut, yaitu echo "Nilai huruf: A";.
4. Setelah blok kode di dalam if pertama dieksekusi, program akan keluar dari seluruh struktur if-elseif-else dan tidak akan memeriksa kondisi-kondisi berikutnya (elseif). Ini memastikan bahwa hanya satu hasil yang ditampilkan, sesuai dengan nilai yang paling relevan.

SOAL 4.2

Output

Atlet tersebut memerlukan 17 hari untuk mencapai jarak 5000 kilometer.

Dari percobaan ini, saya mengamati cara kerja **loop while**. Loop ini terus mengeksekusi blok kode di dalamnya selama kondisinya masih bernilai true.

1. Program dimulai dengan $jarakSaatIni sebesar 0.
2. Setiap kali loop berjalan, program:
   - Menambahkan 30 ke $jarakSaatIni ($jarakSaatIni += $peningkatanHarian;).
   - Menambah hitungan hari ($hari++;).
3. Loop terus berulang hingga nilai $jarakSaatIni mencapai atau melampaui $jarakTarget (500).
4. Kondisi ($jarakSaatIni < $jarakTarget) menjadi false ketika $jarakSaatIni mencapai 510 (setelah 17 hari), dan loop pun berhenti.
5. Program kemudian mencetak hasil akhir, yaitu nilai $hari yang sudah dihitung.

Pengamatan ini menunjukkan bahwa loop while sangat efektif untuk melakukan tugas yang berulang-ulang hingga suatu kondisi tertentu terpenuhi, tanpa harus tahu pasti berapa kali loop akan berulang di awal.

SOAL 4.2

Output

Atlet tersebut memerlukan 17 hari untuk mencapai jarak 5000 kilometer.

Dari percobaan ini, saya mengamati cara kerja loop while. Loop ini terus mengeksekusi blok kode di dalamnya selama kondisinya masih bernilai true.

1. Program dimulai dengan $jarakSaatIni sebesar 0.
2. Setiap kali loop berjalan, program:
   - Menambahkan 30 ke $jarakSaatIni ($jarakSaatIni += $peningkatanHarian;).
   - Menambah hitungan hari ($hari++;).
3. Loop terus berulang hingga nilai $jarakSaatIni mencapai atau melampaui $jarakTarget (500).
4. Kondisi ($jarakSaatIni < $jarakTarget) menjadi false ketika $jarakSaatIni mencapai 510 (setelah 17 hari), dan loop pun berhenti.
5. Program kemudian mencetak hasil akhir, yaitu nilai $hari yang sudah dihitung.

Pengamatan ini menunjukkan bahwa loop while sangat efektif untuk melakukan tugas yang berulang-ulang hingga suatu kondisi tertentu terpenuhi, tanpa harus tahu pasti berapa kali loop akan berulang di awal.

SOAL 4.3

Output

Jumlah buah yang dihasilkan dari 10 lahan adalah 500 buah.

Dari percobaan ini, saya mengamati cara kerja loop for, yang digunakan untuk mengulang blok kode dalam jumlah yang telah ditentukan. Loop ini berjalan sebanyak $jumlahLahan (10 kali). Pada setiap iterasi, ia menghitung jumlah buah dari satu lahan ($tanamanPerLahan * $buahPerTanaman) dan menambahkannya ke $jumlahBuah. Proses ini berulang 10 kali sampai semua lahan telah dihitung, menghasilkan total 500 buah.

SOAL 4.4

Output

Total skor ujian adalah 439.

Dari percobaan ini, kita mengamati cara kerja loop foreach yang efektif untuk melakukan iterasi pada elemen-elemen dalam sebuah array.

1. Program menginisialisasi sebuah array bernama $skorUjian dan variabel $totalSkor dengan nilai 0.

2. Loop foreach secara otomatis mengambil setiap nilai dari array $skorUjian dan menetapkannya ke variabel sementara bernama $skor.
3. Setiap kali loop berjalan, program menambahkan nilai $skor saat itu ke $totalSkor.
4. Proses ini berlanjut sampai semua elemen dalam array $skorUjian telah diproses. Setelah iterasi terakhir, loop berhenti.
5. Program kemudian mencetak hasil total skor yang sudah diakumulasikan.

SOAL 4.5

Output

Nilai:85 (Lulus)
Nilai:92 (Lulus)
Nilai: 58 (Tidak Lulus)
Nilai:64 (Lulus)
Nilai:90 (Lulus)
Nilai: 55 (Tidak Lulus)
Nilai:88 (Lulus)
Nilai:79 (Lulus)
Nilai:70 (Lulus)
Nilai:96 (Lulus)

Dari percobaan ini, saya mengamati bagaimana continue memengaruhi alur eksekusi sebuah loop.

1. Loop foreach memproses setiap nilai dalam array $nilaiSiswa satu per satu.
2. Di dalam loop, ada kondisi if ($nilai < 60). Jika kondisi ini true (misalnya, saat nilai 58 atau 55), program akan mencetak "Tidak lulus" dan kemudian menjalankan continue.
3. Perintah continue berfungsi untuk melewatkan sisa kode di dalam iterasi loop saat ini dan segera melompat ke iterasi berikutnya.
4. berarti, ketika nilai siswa adalah 58, program akan mencetak "Nilai: 58 (Tidak lulus)" dan langsung melanjutkan ke nilai berikutnya (64), tanpa menjalankan echo "Nilai: $nilai (Lulus)
"; untuk iterasi itu.
5. Untuk semua nilai lainnya (yang lebih besar atau sama dengan 60), kondisi if tidak terpenuhi, sehingga program melewati blok if dan langsung menjalankan echo "Nilai: $nilai (Lulus)
";.

Pengamatan ini menunjukkan bahwa continue adalah alat yang berguna untuk mengendalikan alur loop, memungkinkan kita untuk melewatkan bagian tertentu dari iterasi berdasarkan kondisi yang telah ditentukan.

SOAL 4.6

Ada soal cerita : Ada seorang guru ingin menghitung total nilai dari 10 siswa dalam ujian matematika. Guru ini ingin mengabaikan dua nilai tertinggi dan dua nilai terendah. Bantu guru ini menghitung total nilai yang akan digunakan untuk menentukan nilai rata-rata setelah mengabaikan nilai tertinggi dan terendah. Berikut daftar nilai dari 10 siswa (85, 92, 78, 64, 90, 75, 88, 79, 70, 96)

Jawab :

Daftar Nilai Siswa awal: 85, 92, 78, 64, 90, 75, 88, 79, 70, 96
Dua nilai terendah: 64, 70
Dua nilai tertinggi: 92, 96
Nilai yang diabaikan: [64, 70, 92, 96]
Total nilai yang digunakan untuk rata-rata adalah: 495
Rata-rata nilai: 82.5

Kode:

```php
// daftar nilai dari 10 siswa
$nilaiSiswa = [85, 92, 78, 64, 90, 75, 88, 79, 70, 96];
echo "Daftar Nilai Siswa awal: " . implode(separator: ", ", array: $nilaiSiswa) . "<br>";

// mengurutkan nilai dari yang terkecil ke terbesar
sort(array: &$nilaiSiswa);

// ambil dua nilai terendah
$nilaiTerendah = [$nilaiSiswa[0], $nilaiSiswa[1]];

// ambil dua nilai tertinggi
$duaTertinggi = [$nilaiSiswa[count(value: $nilaiSiswa)-2], $nilaiSiswa[count(value: $nilaiSiswa)-1]];

echo "Dua nilai terendah: " . implode(separator: ", ", array: $nilaiTerendah) . "<br>";
echo "Dua nilai tertinggi: " . implode(separator: ", ", array: $duaTertinggi) . "<br>";

// mengaabaikan nilai terendah dan tertinggi
$nilaiDiabaikan = array_merge(arrays: $nilaiTerendah, $duaTertinggi);
echo "Nilai yang diabaikan: [" . implode(separator: ", ", array: $nilaiDiabaikan) . "]<br>";

// membuang dua nilai terendah dan dua nilai tertinggi dari daftar nilai
array_shift(array: &$nilaiSiswa); // hapus nilai terendah pertama
array_shift(array: &$nilaiSiswa); // hapus nilai terendah kedua
array_pop(array: &$nilaiSiswa);   // hapus nilai tertinggi pertama
array_pop(array: &$nilaiSiswa);   // hapus nilai tertinggi kedua

// menghitung total dan rata-rata nilai yang tersisa
$totalNilai = array_sum(array: $nilaiSiswa);
$rata = $totalNilai / count(value: $nilaiSiswa);

echo "Total nilai yang digunakan untuk rata-rata adalah: $totalNilai<br>";
echo "Rata-rata nilai: $rata";
?>
```

Dari praktikum ini, bahwa PHP memiliki serangkaian fungsi built-in yang sangat efektif untuk memanipulasi data dalam bentuk array.

1. Pengurutan Data: Langkah pertama yang sangat penting adalah mengurutkan array $nilaiSiswa menggunakan fungsi sort(). Pengurutan ini untuk memudahkan identifikasi nilai terendah dan tertinggi, yang merupakan syarat utama untuk langkah-langkah berikutnya.

2. Identifikasi dan Pembuangan Nilai: Setelah array terurut, nilai-nilai yang akan diabaikan (dua terendah dan dua tertinggi) dapat diidentifikasi berdasarkan posisi indeks. Nilai terendah berada di awal array ($nilaiSiswa[0] dan $nilaiSiswa[1]), sementara nilai tertinggi berada di akhir ($nilaiSiswa[count() - 2] dan $nilaiSiswa[count() - 1]). Proses penghapusan dilakukan secara efisien dengan fungsi array_shift() untuk menghapus dari awal dan array_pop() untuk menghapus dari akhir.

3. Perhitungan Akhir: Setelah array dimodifikasi, fungsi array_sum() digunakan untuk menjumlahkan semua elemen yang tersisa. Hasilnya kemudian dibagi dengan jumlah elemen yang tersisa, yang didapat dari fungsi count(), untuk mendapatkan nilai rata-rata yang akurat sesuai kriteria.

SOAL 4.7

Ada soal cerita : Seorang pelanggan ingin membeli sebuah produk dengan harga Rp 120.000. Toko tersebut menawarkan diskon sebesar 20% untuk pembelian di atas Rp 100.000. Bantu pelanggan ini untuk menghitung harga yang harus dibayar setelah mendapatkan diskon.

Jawab :

> Harga Awal : Rp 120.000
> Diskon : Rp 12.000
> Harga Akhir yang harus dibayar : Rp 108.000

Kode

```php
//soal 4.7
$harga="120000";

//cek apakah harga lebih dari 100000
if($harga > 100000){
    //jika ya, maka diskon 20%
    $diskon = 0.10 * $harga;
    $hargaAkhir = $harga - $diskon;
}else{
    //jika tidak, maka diskon 0%
    $diskon = 0;
    $hargaAkhir = $harga;
}
echo "Harga Awal : Rp " . number_format(num: $harga, decimals: 0, decimal_separator: ',', thousands…'.') . "<br>";
echo "Diskon     : Rp " . number_format(num: $diskon, decimals: 0, decimal_separator: ',', thousands…'.') . "<br>";
echo "Harga Akhir yang harus dibayar : Rp " . number_format(num: $hargaAkhir, decimals: 0, decimal_separator: ',', thousands…'.');
?>
```

Percobaan ini, membuat program sederhana untuk menghitung harga akhir sebuah produk setelah diberi diskon:

1. Pada awal program ditentukan harga produk sebesar **Rp120.000**.
2. Program kemudian melakukan pengecekan if ($harga > 100000). Karena kondisi benar, pelanggan berhak mendapatkan diskon.
3. Diskon dihitung sebesar **20% dari harga awal**, yaitu **Rp24.000**, sehingga harga akhir menjadi **Rp96.000**.
4. Jika harga produk tidak melebihi Rp100.000, maka diskon tidak diberikan dan harga akhir tetap sama.
5. Fungsi number_format() digunakan agar tampilan angka rupiah terlihat rapi.

Dari hasil percobaan, bisa disimpulkan bahwa program berjalan sesuai harapan dan cukup membantu dalam menghitung harga setelah diskon secara otomatis.

SOAL 4.8

Jika skor pemain >= 500

Total skor pemain adalah 650 poin.
Apakah pemain mendapatkan hadiah? YA

Jika skor pemain <= 500

Total skor pemain adalah 450 poin.
Apakah pemain mendapatkan hadiah? TIDAK

Kode

```php
//soal 4.8
//skor awal pemain
$poin = 450;//poin awal dan ubah sesuai output yang diinginkan

echo "Total skor pemain adalah $poin poin.<br>";

//tenary operator
$hadiah = ($poin >= 500) ? " YA " : " TIDAK ";
echo "Apakah pemain mendapatkan hadiah? $hadiah<br>";
?>
```

Penjelasan Setelah Percobaan

1. Program mendefinisikan variabel $poin yang menyimpan skor pemain, misalnya 650.
2. Baris pertama program menampilkan total skor dengan echo.
3. Pada baris berikutnya digunakan operator ternary ($poin > 500) ? "YA" : "TIDAK".
   - Jika skor lebih dari 500, variabel $hadiah berisi "YA".
   - Jika skor kurang atau sama dengan 500, maka $hadiah berisi "TIDAK".
4. Hasil akhir menampilkan apakah pemain berhak mendapat hadiah tambahan atau tidak.
5. Dari percobaan ini terlihat bahwa operator ternary sangat berguna untuk menyingkat kondisi sederhana dalam satu baris kode.

SOAL 5.1

Output

Daftar nilai siswa yang lulus: 85, 92, 78, 90, 88, 79, 70, 96

Dari percobaan ini, Dapat digamati bagaimana loop foreach dan struktur kontrol if dapat bekerja sama untuk memproses data dari sebuah array dan menyimpannya ke array baru.

1. Program menginisialisasi dua array: $nilaiSiswa yang berisi semua nilai, dan $nilaiLulus yang masih kosong.
2. Loop foreach kemudian mengiterasi setiap elemen dalam $nilaiSiswa.

3. Di dalam loop, kondisi if ($nilai >= 70) memeriksa apakah nilai siswa saat ini sama dengan atau lebih besar dari 70.
4. Jika kondisi ini terpenuhi (bernilai true), nilai tersebut ditambahkan ke array $nilaiLulus menggunakan sintaks $nilaiLulus[] = $nilai;.
5. Setelah loop selesai memproses semua nilai, program mencetak array $nilaiLulus yang berisi hanya nilai-nilai yang memenuhi syarat kelulusan.

Pengamatan ini menunjukkan bahwa kombinasi loop dan kondisi adalah cara yang efektif untuk memfilter dan memanipulasi data di dalam array. Ini memungkinkan kita untuk mengambil subset data yang spesifik dari sebuah array yang lebih besar.

SOAL 5.2

Output

Daftar karyawan dengan penglaman kerja lebih dari 5 tahun: Alice, Charlie, Eve

Dari percobaan ini, saya mengamati bagaimana loop foreach dapat digunakan untuk memproses array multidimensi dan menyaring data berdasarkan kondisi tertentu.

1. Program menginisialisasi array $daftarKaryawan, di mana setiap elemennya adalah array lain yang berisi nama karyawan dan tahun pengalaman kerja.
2. Loop foreach mengiterasi setiap elemen array utama, dan setiap sub-array (['Alice', 7], dll.) ditetapkan ke variabel $karyawan untuk sementara.
3. Kondisi if ($karyawan[1] > 5) memeriksa nilai pengalaman kerja, yang disimpan di indeks 1 dari setiap sub-array.
4. Jika kondisi terpenuhi, nama karyawan ($karyawan[0]) ditambahkan ke array baru $karyawanPengalamanLimaTahun.

Pengamatan ini menunjukkan bahwa PHP sangat fleksibel dalam mengolah struktur data yang kompleks. Dengan mengakses indeks yang tepat di dalam loop, kita dapat dengan mudah mengekstrak informasi spesifik dari data yang terstruktur.

SOAL 5.3

Output

Daftar nilai mahasiswa dalam mata kuliah Fisika:
Nama: Alice, Nilai: 90
Nama: Bob, Nilai: 88
Nama: Charlie, Nilai: 75

Dari percobaan ini, Dapat diketahui cara kerja array multidimensi dan loop foreach yang digunakan untuk mengaksesnya. Array $daftarNilai adalah array multidimensi yang menggunakan nama mata kuliah sebagai kunci utama. Setiap kunci utama menampung array asosiatif lain yang berisi nama siswa dan nilainya.

1. Program ini dengan cerdas memilih sub-array yang ingin ditampilkan dengan mengakses $daftarNilai[$mataKuliah]. Karena $mataKuliah disetel ke "Fisika", program hanya akan memproses data yang ada di bawah kunci "Fisika".
2. Loop foreach kemudian beroperasi pada sub-array tersebut, menetapkan nama siswa ke variabel $nama dan nilainya ke $nilai di setiap iterasi.
3. Dengan cara ini, program bisa menampilkan daftar nama dan nilai siswa secara spesifik untuk mata kuliah Fisika.

SOAL 5.4

Ada soal cerita : Seorang guru ingin mencetak daftar nilai siswa dalam ujian matematika. Guru tersebut memiliki data setiap siswa terdrir dari nama dan nilai. Bantu guru ini mencetak daftar nilai siswa yang mencapai nilai di atas rata-rata kelas. Dengan ketentuan nama dan nilai siswa Alice dapat 85, Bob dapat 92, Charlie dapat 78, David dapat 64, Eva dapat 90

Jawab :

Rata-rata kelas adalah: 81.80
Daftar siswa dengan nilai di atas rata-rata:
Nama: Alice, Nilai: 85
Nama: Bob, Nilai: 92
Nama: Eva, Nilai: 90

Kode:

```php
//soal 5.4
//Membuat array dua dimensi untuk data siswa
$daftarSiswa = [
    ["nama" => "Alice", "nilai" => 85],
    ["nama" => "Bob", "nilai" => 92],
    ["nama" => "Charlie", "nilai" => 78],
    ["nama" => "David", "nilai" => 64],
    ["nama" => "Eva", "nilai" => 90]
];

//Hitung total nilai untuk menemukan rata-rata
$totalNilai = 0;
foreach ($daftarSiswa as $siswa) {
    $totalNilai += $siswa["nilai"];
}

$rataRataKelas = $totalNilai / count(value: $daftarSiswa);
echo "Rata-rata kelas adalah: " . number_format(num: $rataRataKelas, decimals: 2) . "<br>";

//Filter dan tampilkan siswa yang nilainya di atas rata-rata
echo "Daftar siswa dengan nilai di atas rata-rata:<br>";
foreach ($daftarSiswa as $siswa) {
    if ($siswa["nilai"] > $rataRataKelas) {
        echo "Nama: " . $siswa["nama"] . ", Nilai: " . $siswa["nilai"] . "<br>";
    }
}
```

Dari percobaan ini, Dapat diamati bagaimana array dua dimensi dan loop foreach dapat digunakan untuk menyelesaikan masalah yang lebih kompleks yang melibatkan beberapa langkah pemrosesan data.

1. Struktur Data: Array dua dimensi sangat efektif untuk menyimpan data yang terstruktur, seperti data siswa yang memiliki lebih dari satu properti (nama dan nilai).

2. Menghitung Rata-rata: Langkah pertama yang penting adalah menghitung rata-rata dari semua nilai siswa. Ini dicapai dengan menggunakan satu loop foreach untuk menjumlahkan semua nilai, kemudian membaginya dengan jumlah total siswa.
3. Memfilter Data: Setelah rata-rata diketahui, loop foreach kedua digunakan untuk memproses kembali data siswa. Kondisi if ($siswa["nilai"] > $rataRataKelas) memungkinkan program untuk secara cerdas memfilter dan hanya menampilkan nama serta nilai siswa yang memenuhi kriteria yang ditentukan.

Pengamatan ini menunjukkan bahwa dengan menggabungkan berbagai konsep PHP—seperti array, loop, dan kondisi—kita bisa memanipulasi data untuk mendapatkan informasi yang spesifik dan relevan dari kumpulan data yang lebih besar.