PERTEMUAN 3 PBO

Struktur Kontrol Pemrograman PHP

Selamat datang kembali sudara-saudara.
Pada pertemuan kali ini kita akan masuk ke materi Sturktur Kontrol Pemrograman PHP. Seperti yang sudah pernah kalian pelajari juga pada matakuliah dasar pemgrograman, struktur control akan selalu ada pada setiap materi pemrograman karena sangat penting dan tentu akan sangat sering digunakan.

PEMAHAMAN MATERI INI SANGAT PENTING!!! JANGAN BUKAN UNTUK DIHAFAL

Struktur kontrol pemrograman PHP berkaitan dengan bagaimana sebuah kode program berjalan. Dalam bab ini kita akan mempelajari struktur kontrol if, else if, dan switch. Setelah itu juga akan dibahas tentang perulangan for, while, do while dan foreach.

Struktur Logika IF

Struktur logika if dipakai untuk mengatur kapan sebuah kode program akan dijalankan. Sebagai contoh, saya bisa menampilkan seluruh tabel mahasiswa hanya untuk mahasiswa berdomisili di "bogor". Jika domisili bukan "bogor", data tidak akan ditampilkan. Berikut contoh penulisannya:

```
<?php
if ($domisili == "bogor") {
// tampilkan tabel mahasiswa
}
```

Simple bukan?

Bagian kode `($domisili == "bogor") adalah proses perbandingan dimana outputnya adalah TRUE or FALSE, Silakan buka lagi materi perbandingan sebelumnya apabila lupa.`

Penulisan struktur **if** setidaknya butuh 2 bagian, yakni **kondisi** (atau disebut juga sebagai *expression*) dan **statement**:

```
<?php
if (kondisi) {
statement;
}
```

Statement bisa terdiri dari satu, dua atau ratusan baris perintah PHP. Baris ini hanya akan dijalankan selama **kondisi** bernilai **true**. Jika kondisi menghasilkan nilai **false**, statement tidak akan di proses. Tanda kurung kurawal '{' dan '}' menandakan blok kode program yang akan diproses ketika kondisi terpenuhi (kondisi bernilai true).
Kondisi yang ingin diproses haruslah bertipe boolean atau 'sesuatu' yang menghasilkan boolean true atau false.

Ok kita masuk contoh sederhana buat folder baru dalam htdocs dengan nama pertemuan_2 untuk menyimpan latihan2 yang akan kita buat ke depan:

Buat file dengan nama **if_false.php**

```php
<?php
  $user = "guest";

  if ($user=="admin") {
    echo "Selamat datang Admin!";
  }
```

Apakah kalian bisa membayangkan outputnya apabila kode di atas dipanggil di browser??? Yupp browser tidak akan menampilkan apapun, KENAPA?? Klo sudah bisa jawab berarti kalian sudah paham.

Karena variable **$user** kita beri nilai **guest**, sedangkan pada **if** kondisi yang dilihat apakah nilai **$user = admin**, tentu kondisi ini akan menghasilkan **output false** sehingga statement **echo** tidak akan dijalankan, PAHAM YA GAES???
Ok sekarang kalian buat file baru untuk sebagai pembanding dengan nama **if_true.php**

```php
<?php
  $user="admin";

  if ($user=="admin") {
    echo "Selamat datang Admin!";  // Selamat datang Admin!
  }
```

Nah klo kita lihat program diatas pasti kalian sudah bisa membayangkan apa output pada Browser?? Benar sekali, harusnya akan muncul teks **`Selamat datang Admin!,`** `karena hasil dari kondisi yang dibandingkan adalah $user= admin dimana menghasilkan nilai True dan menyebabkan statement echo akan dijalankan.`

Multiple IF

Kita juga bisa menulis beberapa kondisi `if` untuk membuat percabangan kode program seperti contoh berikut:

```php
<?php
if (kondisi1) {
      statement1;
      statement2;
      }
if (kondisi2) {
      statement3;
      statement4;
      }
```

Untuk kasus yang lebih spesifik, kita bisa membuat struktur `if` di dalam `if`. Ini dikenal sebagai **nested IF** atau **IF bersarang**. Berikut contohnya:

```php
<?php
if (kondisi1) {
      statement1;
      if (kondisi2) {
            statement1;
```

```
        }
    }
```

Kedua contoh di atas memiliki struktur yang berbeda coba pahami dan resapi, contoh pertama ada 2 buah if yang **berdiri sendiri**, **ARTINYA** apabila kondisi if pertama menghasilkan nilai false maka if kedua akan tetap dijalankan karena berdiri sendiri dan tidak berhubungan dengan if pertama.

pada contoh kedua ada 2 buah if dimana if yang kedua merupakan statement dari if yang pertama, **ARTINYA** apabila kondisi if pertama menghasilkan nilai false maka if kedua **tidak akan** dijalankan karena merupakan bagian dari statement.

Dimana bisa dibedakan??? Perhatikan dengan seksama letak dari kurung kurawal pembatas { } kondisi if, setiap kondisi if akan selalu diawali dengan kurung kurawal buka { dan diakhiri dengan kurung kurawal tutup }, ini harus teliti, karena bila salah peletakan maka akan menyebabkan kesalahan program.

Struktur Logika ELSE dan ELSE IF

Pada dasarnya, struktur logika **else** dan **else if** adalah perpanjangan dari struktur if. Bagian **else** dijalankan ketika kondisi if tidak sesuai atau menghasilkan nilai **false**. Berikut contohnya simpan dengan nama **else_if.php**:

```php
<?php
  $user="guest";

  if ($user=="admin"){
    echo "Selamat datang Admin!";
  }
  else {
    echo "Maaf, anda bukan Admin";
  }
```

Cara baca program diatas adalah jika nilai user sama dengan admin maka tampilkan teks "**selamat datang admin!**" Tapi jika ternyata nilainya bukan admin maka akan tampil "**maaf, anda bukan admin**!"

Jadi apa output program diatas?? Silakan coba di browser.

Multiple ELSE dan ELSE IF

Untuk kode program yang lebih kompleks, kita bisa mengkombinasikan beberapa struktur else if seperti contoh berikut simpan dengan nama **multiple_else.php**:

```php
<?php
  $user="guest";

  if ($user=="admin"){
    echo "Selamat datang Admin!";
  }
  else if ($user=="user"){
    echo "Selamat datang User";
```

```
  }
  else if ($user=="guest"){
    echo "Selamat datang Tamu";
  }
  else {
    echo "Maaf, saya tidak kenal anda";
  }
```

Jadi apabila terdapat beberapa kondisi else kita harus menuliskan if lagi dibelakangnya, else if. Baru pada kondisi yang terakhir kita cukup menuliskan else.

Silakan coba program diatas dan lihat hasilnya di browser.

Latihan IF, ELSE, dan ELSE IF

Latihan 1: Genap atau Ganjil

Buatlah kode program yang bisa membedakan angka genap dengan angka ganjil. Misalkan variabel **$a** berisi 50, maka tampil teks *"50 adalah angka genap"*. Jika **$a** berisi 7, maka tampil teks *"7 adalah angka ganjil"*.
Tips: gunakan operator **modulus (%)** untuk membedakan bilangan genap dan ganjil.

Jawaban Latihan 1

Untuk membedakan antara angka genap dan ganjil, kita bisa menggunakan fakta bahwa bilangan genap adalah bilangan yang habis dibagi 2. Kondisi ini bisa di cek dengan operator **mod** (%). Berikut kode programnya, simpan dengan nama **ganjil_genap.php**:

```
<?php
  $a = 10;
  if ($a % 2 == 0){
    echo "$a adalah angka genap";
  }
  else {
    echo "$a adalah angka ganjil";
  }
```

Silakan coba ganti nilai $a dengan angka ganjil untuk melihat hasil yang lain.

Jika variabel `$a` berisi angka genap (kelipatan 2), maka kondisi `if ($a % 2 == 0)` akan menghasilkan **true**. Dengan demikian, yang dijalankan adalah `echo "$a adalah angka genap"`.
Selain genap, tentu angka ganjil, maka saya tinggal menambahkan blok `else` yang berisi perintah `echo "$a adalah angka ganjil"`.

Lebih Besar, Lebih Kecil atau Sama Dengan

Buatlah sebuah kode program yang terdiri dari 2 variabel: `$a` dan `$b`. Kedua variabel ini berisi angka integer atau float. Tampilan akhir adalah salah satu kondisi berikut:

- `$a lebih kecil dari $b`
- `$a lebih besar dari $b`
- `$a sama dengan $b`

Sebagai contoh, jika saya memberikan nilai `$a=6` dan `$b=10`, tampilan akhir adalah: *"6 lebih kecil dari 10"*. Jika `$a=5` dan `$b=5`, tampilan akhir adalah: *"5 sama dengan 5"*.

Jawaban Latihan 2
Kita harus membuat kondisi untuk 3 kemungkinan: apakah **$a < $b**, atau **$a > $b**, atau **$a==$b**. Berikut kode programnya, simpan dengan nama **besar_kecil.php**:

```php
<?php
  $a = 8;
  $b = 10;
  if ($a < $b){
    echo "$a lebih kecil dari $b";
  }
  else if ($a > $b){
    echo "$a lebih besar dari $b";
  }
  else {
    echo "$a sama dengan $b";
  }
```

Konsep program diatas adalah akan memeriksa seluruh kondisi dari if, apabila ada yang sesuai maka statement akan dijalankan. Saya harap kalian sudah paham akan hal ini karena sudah pernah belajar dasar pemrograman, bedanya disini kita menggunakan Bahasa pemrograman PHP dengan tentunya perbedaan sintax penulisan.

Latihan 3: Username dan Password

Buatlah kode program yang akan memeriksa apakah `$username` berisi string "`admin`" dan `$password` berisi string "`qwerty`". Jika keduanya benar, tampilkan pesan: *"Username dan Password sesuai, hak akses diberikan"*. Jika salah satu salah, tampilkan pesan: *"Username atau Password tidak sesuai!"*.
Tips: *username* dan *password* harus sesuai, jika salah satu atau keduanya tidak cocok maka tampilkan pesan: *"Username atau Password tidak sesuai!"*.

Jawaban Latihan 3
Kata kunci di sini adalah `$username` berisi "`admin`" **dan** `$password` berisi "`qwerty`". Artinya kita perlu gabungan 2 buah kondisi menggunakan operator **AND**. Berikut kode programnya, simpan dalam file **user_password.php**

```php
<?php
  $username = "admin";
  $password = "qwerty";

  if ($username=="admin" AND $password=="qwerty"){
    echo "Username dan password sesuai, hak akses diberikan";
  }
  else {
    echo "Username atau password tidak sesuai!";
  }
```

Perhatikan bagian if condition, disana kita memungkinkan untuk mengecek kondisi lebih dari 1 perbandingan, lihat perbedaan dengan 2 contoh sebelumnya.!!!

Ok contoh untuk logika if else cukup sampai disini.

Latihan silakan masing2 membuat contoh minimal 5 program dengan logika if else. Nama file bebas, simpan ditempat yang sama dengan folder pertemuan_2

Struktur Logika SWITCH

Struktur logika switch adalah sebuah struktur percabangan yang akan memeriksa satu variabel, lalu menjalankan perintah sesuai dengan kondisi. Struktur switch ini mirip dengan struktur if yang ditulis berulang.

struktur **switch** terdiri dari beberapa bagian, berikut format dasar penulisan switch dalam PHP:

```
 switch ($var) {
 case value1:
       statement1;
       break;
 case value2:
       statement2;
       break;
 }
```

Setelah perintah **switch**, kita menulis variabel yang akan diperiksa. Variabel ini ditempatkan dalam tanda kurung. Seluruh *block switch* berada di antara kurung kurawal '{' dan '}'. Tiap kondisi yang mungkin terjadi ditulis pada perintah **case**, lalu diikuti dengan kondisi yang ingin diuji. Jika sesuai, baris statement akan dijalankan. Alur program untuk switch di eksekusi berurutan mulai dari baris pertama sampai terakhir. Kata kunci **break** dipakai untuk keluar dari **switch** begitu kondisi case sudah terpenuhi.

Perbedaan Antara IF dengan Switch

Walaupun memiliki tujuan yang hampir sama, struktur **if** dan **switch** memiliki perbedaan mendasar. Di dalam struktur **switch**, kondisi logika hanya diperiksa satu kali saja, yakni di awal perintah switch. Sedangkan di dalam struktur if, kondisi logika akan selalu diperiksa pada setiap perintah if. Untuk struktur percabangan yang banyak, switch lebih cepat di eksekusi daripada if.
Di sisi lain, switch memiliki keterbatasan karena hanya bisa digunakan untuk tipe data sederhana seperti memeriksa nilai dari sebuah variabel. Struktur switch tidak bisa menangani kondisi yang lebih rumit seperti logika **AND** atau operasi perbandingan. Dengan keterbatasan ini, kita akan lebih sering menggunakan if daripada switch.

Contoh program dengan logika switch, simpan dengan nama **switch_hari.php**.

```php
<?php
  $hari = 4;
  switch ($hari) {
  case 1 :
    echo "Hari Senin";
    break;
  case 2 :
    echo "Hari Selasa";
```

```
    break;
  case 3 :
    echo "Hari Rabu";
    break;
  case 4 :
    echo "Hari Kamis";
    break;
  case 5 :
    echo "Hari Jum'at";
    break;
  case 6 :
    echo "Hari Sabtu";
    break;
  case 7 :
    echo "Hari Minggu";
    break;
  default :
    echo "Nama hari cuma ada 7!";
    break;
  }
```

Inti dari program adalah logika switch akan memeriksa seluruh kondisi case yang sudah didaftarkan sesuai dengan nilai variable yang diinginkan, misal pada contoh diatas akan dicari apakah nilai $hari=4 ada di dalam case. Apabila ditemukan maka akan ditampilkan hasilnya pada contoh di atas Hari kamis. Tetapi apabila kondisi yang diinginkan tidak ditemukan akan ditampilkan nilai default Nama hari Cuma ada 7!!!.

Perintah **break berfungsi untuk menghentikan proses pencarian apabila sudah menemukan kondisi yang diinginkan, bagaimana bila tanpa break??? Silakan dicoba.**

Silakan dicoba dan dipahami.

Struktur Perulangan FOR

Struktur perulangan (atau dalam bahasa inggris disebut dengan **loop**) adalah kode program untuk mengulang beberapa baris perintah. Dalam PHP terdapat beberapa jenis instruksi perulangan: **for**, **while**, **do while** dan **foreach**.
Untuk membuat perulangan, setidaknya harus tersedia 3 komponen, yaitu:

- Kondisi awal perulangan.
- Perintah program yang akan di ulang.
- Kondisi akhir dimana perulangan harus berhenti.

Kita akan bahas struktur perulangan for terlebih dahulu. Berikut struktur dasar perulangan for dalam PHP:

```
for (start; condition; increment) {
      statement;
      statement;
      }
```

Penjelasan dari struktur diatas sangat sederhana

Start adalah kondisi awal perulangan. Bagian ini biasanya berisi sebuah variabel yang berfungsi sebagai **counter** (variabel pengontrol perulangan). Menjadi standar tidak resmi, variabel counter ditulis dengan `$i`. Jika kita ingin mulai dari angka 1, kondisi start ditulis sebagai `$i = 1`.

Condition adalah kondisi yang menentukan kapan perulangan selesai. Dalam setiap perulangan (di kenal juga dengan istilah **iterasi**), *condition* akan terus diperiksa. Selama *condition* bernilai **true**, iterasi akan di ulang terus menerus.
Sebagai contoh, jika saya ingin menjalankan perulangan sebanyak 20 kali, di bagian condition ini bisa ditulis `$i <= 20`. Artinya, selama variabel counter `$i` nilainya kurang dari atau sama dengan 20, terus lakukan perulangan.

Increment adalah bagian yang digunakan untuk mengatur penambahan variabel counter pada setiap iterasi. Biasanya kita menggunakan operator increment seperti `$i++`.

Statement adalah bagian kode program yang akan di ulang. Statement harus berada di dalam blok kode program yang ditandai dengan tanda kurung kurawal '{' dan '}', kecuali statement tersebut hanya terdiri dari 1 baris saja.

Sebagai contoh, saya ingin menampilkan 10 baris kalimat "`Saya sedang belajar PHP`". Berikut kode programnya, simpan dengan nama **for_1.php**:

```php
<?php
  for ($i = 1; $i <= 10; $i++)  {
    echo "Saya sedang belajar PHP <br>";
  }
```

Kita lihat perulangan diatas dimulai dari nilai $i=1, dan statement akan terus dijalankan sampai nilai $i<=10, increment $i++ digunakan untuk menambah nilai $i selama iterasi sebanyak 1.

Contoh 2 cetak bilangan genap antara 1-20, bagaimana caranya??? Pertama adalah kita harus bisa mengetahui bagaimana cara menentukan sebuah bilangan genap atau ganjil, dengan apa???? Yup dengan operator modulus, masih ingat ya. Operator modulus akan membagi habis bilangan genap dan akan memberi sisa bagi apabila bilangan itu ganjil

Berikut programmnya, simpan dengan nama **for_2.php**

```php
<?php
for ($i=1; $i <= 20; $i++) {
    if ($i%2==0) {
        echo "$i adalah bilangan genap <br>";
    }
    else
        echo "$i adalah bilangan ganjil <br>";
}
?>
```

Lihat hasil dari program diatas dan pahami, contoh diatas menggabungkan perulangan for dengan kondisi if sebagai statement.

Latihan… buat 3 program menggunakan for simpan pada folder yang sama pertemuan_2

Struktur Perulangan WHILE

Salah satu syarat dari perulangan **for** yang baru saja kita pelajari adalah kondisi akhir perulangan sudah harus diketahui. Misalnya kita harus bisa memastikan perulangan akan dilakukan 10 kali, 100 kali, atau 1000 kali. Tapi bagaimana jika kondisi akhir ini belum bisa dipastikan?
Contohnya, saya ingin membuat program tebak angka. Pengunjung akan menebak 1 angka dari 1 sampai 10. Untuk kondisi ini, kita tidak bisa memastikan berapa kali pengunjung akan mencoba hingga benar. Mungkin butuh 1, 2, 5, atau 10 kali sebelum angka tersebut berhasil di tebak.
Untuk kasus ini, PHP (dan juga bahasa pemrograman lain) menyediakan struktur perulangan **while**. Perulangan while cocok dipakai untuk situasi dimana kondisi akhir belum diketahui pada saat perulangan ditulis.

Berikut struktur dasar perulangan while PHP:

```
<?php
start;
while (condition) {
      statement;
      statement;
      increment;
      }
```

Start adalah kondisi awal perulangan, di sini kita bisa mempersiapkan variabel *counter* yang digunakan untuk mengatur **condition.**

Condition adalah kondisi yang harus dipenuhi agar perulangan berlangsung. Ini mirip seperti dalam perulangan for. Selama *condition* bernilai **true**, perulangan akan terus dilakukan.
Condition diperiksa pada setiap awal perulangan. Jika hasilnya **false**, proses perulangan akan berhenti.

Statement adalah kode program yang akan di ulang. Kita bisa membuat kode program yang sederhana seperti perintah `echo`, maupun kumpulan perintah yang lebih kompleks. Tanda kurung kurawal diperlukan untuk membatasi blok program yang akan di ulang. Jika *statement* hanya terdiri dari 1 baris, tanda kurung kurawal boleh tidak ditulis.

Increment juga sama seperti pada perulangan for, yakni untuk mengatur agar variabel counter naik/turun. Pada perulangan while, bagian *increment* ini juga bisa berupa sebuah statement yang akan mengubah *condition* menjadi **false.** Jika dalam perulangan tidak ada suatu kode program yang bisa mengubah nilai variabel *counter*, proses perulangan tidak akan pernah berhenti (*infinity loop*).

Sebagai contoh pertama, saya ingin menampilkan string "*Saya sedang belajar PHP*" sebanyak 10 kali menggunakan perulangan while:

Simpan dengan nama **while_1.php**

```php
<?php
  $i = 1;
  while ($i <= 10) {
    echo "Saya sedang belajar PHP<br>";
    $i++;
  }
```

Perhatikan perbedaan cara penulisannya dengan perulangan for.

Di awal program saya menyiapkan variabel `$i` dan memberikan nilai awal 1. Variabel `$i` inilah yang akan berfungsi sebagai variabel counter dalam perulangan while. Setelah penulisan perintah `while`, di dalam tanda kurung terdapat *condition* yang harus dipenuhi agar perulangan berjalan. Saya membuat kondisi `($i <= 10)`, yang berarti selama nilai variabel `$i` kurang dari 10, terus lakukan perulangan.

Penting untuk diperhatikan adalah logika pemrograman untuk *condition*. Perintah `while ($i <= 10)` berarti bahwa jika nilai variabel `$i = 11`, hasilnya akan **false** dan perulangan berhenti.

Di dalam kode program, kita harus membuat sebuah baris statement yang dipakai untuk mengubah nilai `$i` agar bisa mencapai angka lebih dari 10 supaya perulangan berhenti. Untuk ini, saya menggunakan operator increment `$i++`. Baris inilah yang akan menambah nilai variabel counter `$i` sebanyak 1 angka pada tiap perulangan. Proses *looping* akan terus berjalan hingga pada perulangan ke 10, nilai `$i` akan menjadi 11. Hal ini menyebabkan kondisi `while ($i <= 10)` bernilai **false** dan perulangan berhenti. Kesalahan dalam memahami logika while sering menghasilkan perulangan yang akan diproses secara terus menerus (*infinity loop*).

Latihan… buat 3 program menggunakan while simpan pada folder yang sama pertemuan_2

Struktur Perulangan DO WHILE

Perulangan **while** dan **do while** pada dasarnya hampir sama. Perbedaan terletak pada lokasi pengecekan kondisi akhir perulangan.
Dalam struktur **while**, pengecekan kondisi akhir di lakukan di awal. Jika kondisi tidak terpenuhi, perulangan tidak akan pernah dijalankan. Namun pada perulangan **do while**, pengecekan kondisi dilakukan di akhir, sehingga walaupun kondisi akhir ini menghasilkan *false*, perulangan akan tetap berjalan minimal 1 kali.

Berikut adalah format dasar struktur penulisan do while dalam PHP:

```
start;
do {
      statement;
      statement;
      increment;
} while (condition);
```

Sebagai contoh, perhatikan perulangan while berikut, simpan dengan nama **while_tampil.php**:

```php
<?php
  $i = 1000;
  while ($i <= 10) {
    echo "$i";
    echo "Tidak akan tampil di browser <br>";
    $i++;
  }
```

Lalu buat program berikut untuk contoh while do, simpan dengan nama **while_do.php**

```php
<?php
  $i = 1000;
  do {
    echo "$i ";
    echo "Akan tampil di browser<br>";
    $i++;
  } while ($i <= 10);
```

Silakan cek hasil kedua program di atas lalu lihat perbedaannya.

Kode program pertama while_tampil di atas tidak akan menampilkan apa-apa karena kondisi `while ($i<=10)` sudah langsung menghasilkan nilai **false** (karena saya mendefinisikan nilai `$i=1000`, yang jelas lebih besar dari 10).

Namun apabila perulangan tersebut dijalankan dengan struktur **do while**, hasilnya akan berbeda, Program while_do.php di atas akan menampilkan teks "`1000 Akan tampil di browser`". Ini karena pada struktur do while, perulangan program akan tampil setidaknya 1 kali walaupun kondisi while menghasilkan nilai **false**.

Silakan pahami terlebih dahulu.

Next FUNCTION!!!