



# Form Submission dan Data Handling

Ikhwan Alfath Nurul Fathony

© Hak Cipta 2025 – Prodi Teknologi Informasi - Fakultas Teknik - Universitas Tidar

The background is a deep blue with a complex, abstract pattern of glowing particles and light trails. A prominent feature is a large, circular ring of bright, yellowish-white particles that forms a frame around the central text. Other smaller, wispy structures of particles are scattered throughout the scene, creating a sense of depth and movement.

# Pengenalan Variable & Operator

Insert the Subtitle of Your Presentation



# Pengenalan Back-End dan Front-End Web

Supermarket

Frontend

+

Backend

Berhubungan  
langsung dengan  
pelanggan

Operasional di  
belakang layar

supermarket



# Pengenalan Back-End dan Front-End Web

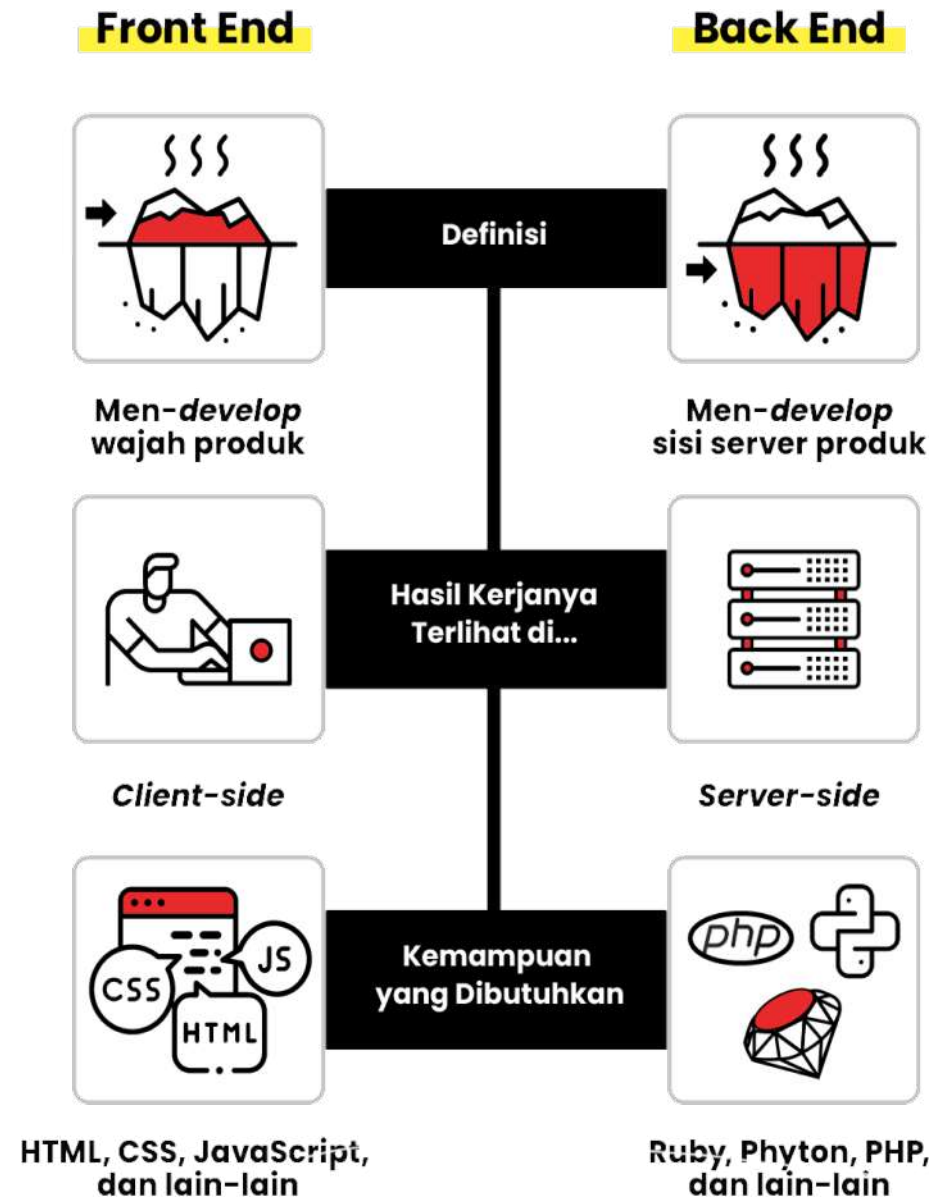
supermarket

## Sistem Supermarket

- Front-end
  - Rak dan produk
  - Etalase
  - Troli belanja
  - Label harga
  - Papan promo
- Back-end
  - Gudang
  - Sistem stok
  - Sistem pembayaran
  - Sistem kasir
  - Sistem pelaporan



# Pengenalan Back-End dan Front-End Web



# Pengenalan Back-End Web

Fokus pada logic  
pengolahan/pemrosesan dan

penyimpanan data

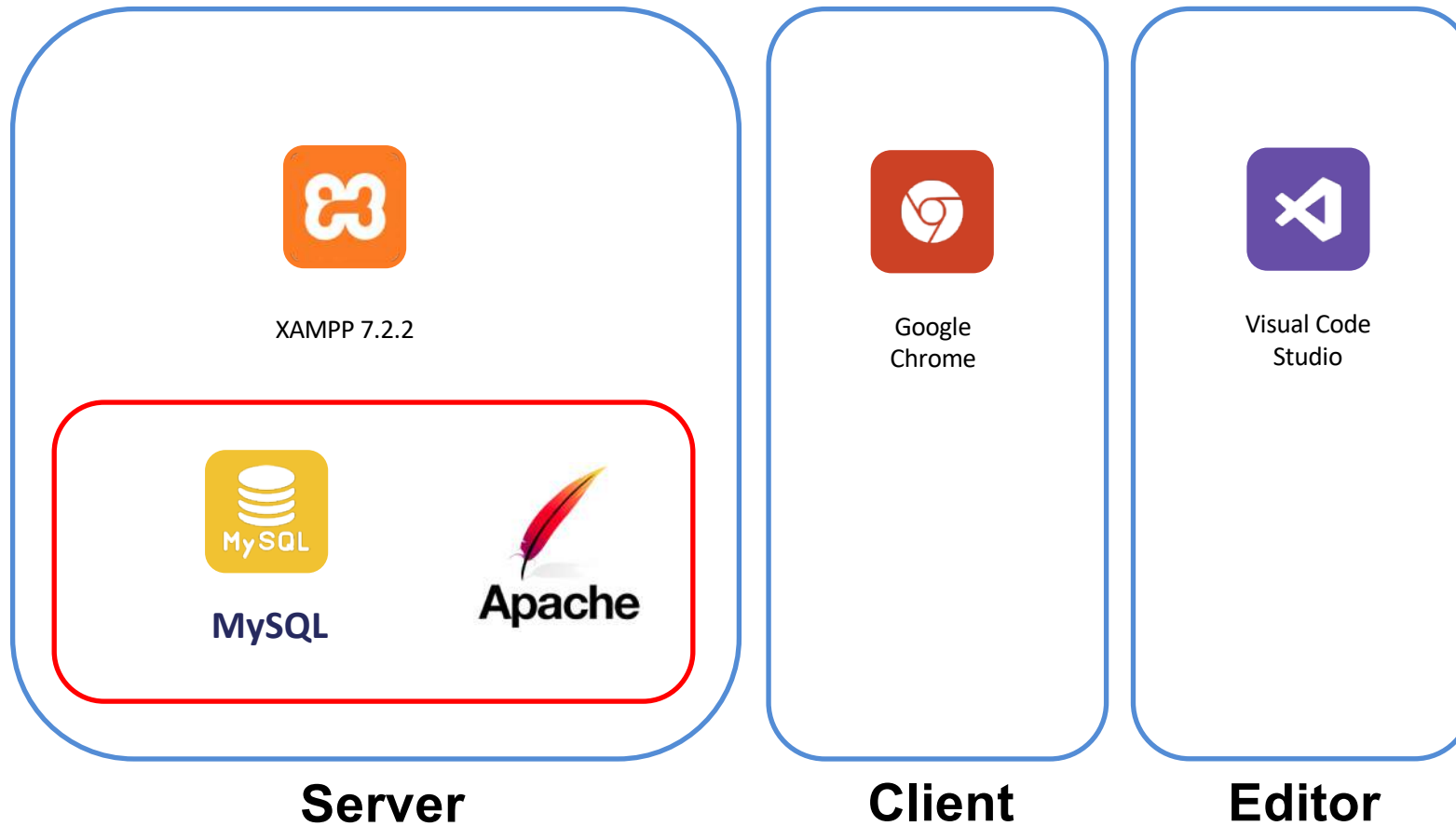


PHP  
Programming

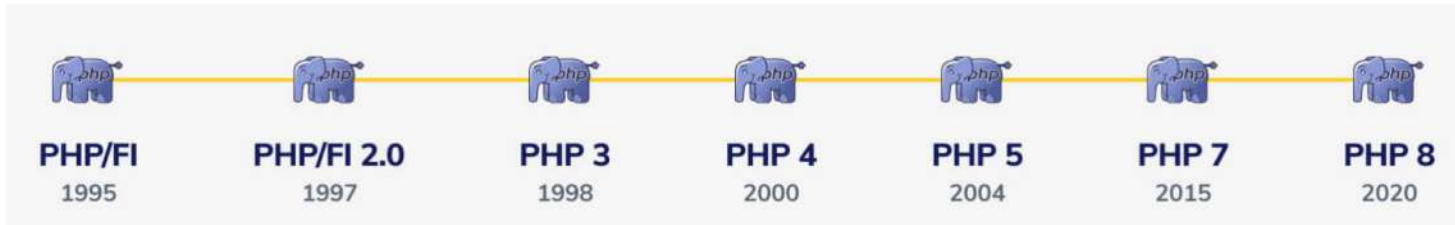


MySQL

# Instalasi PHP dan Tools



# Pengenalan PHP



- Awalnya PHP □ Personal Home Page (Situs personal). Pertama kali dibuat oleh Rasmus Lerdorf tahun 1995. Pada waktu itu PHP masih bernama Form Interpreted (FI)
- Pada November 1997, dirilis PHP/FI 2.0. Pada rilis ini, interpreter PHP sudah diimplementasikan dalam program C.
- Pada Juni 1998, perusahaan Zend merilis interpreter baru untuk PHP dan meresmikan rilis tersebut sebagai PHP 3.0 dan singkatan PHP diubah menjadi akronim berulang PHP: Hypertext Preprocessing.
- Pada pertengahan tahun 1999-2000, Zend merilis interpreter PHP baru dan rilis tersebut dikenal dengan PHP 4.0.
- Pada Juni 2004, Zend merilis PHP 5.0 yang memasukkan model pemrograman berorientasi objek



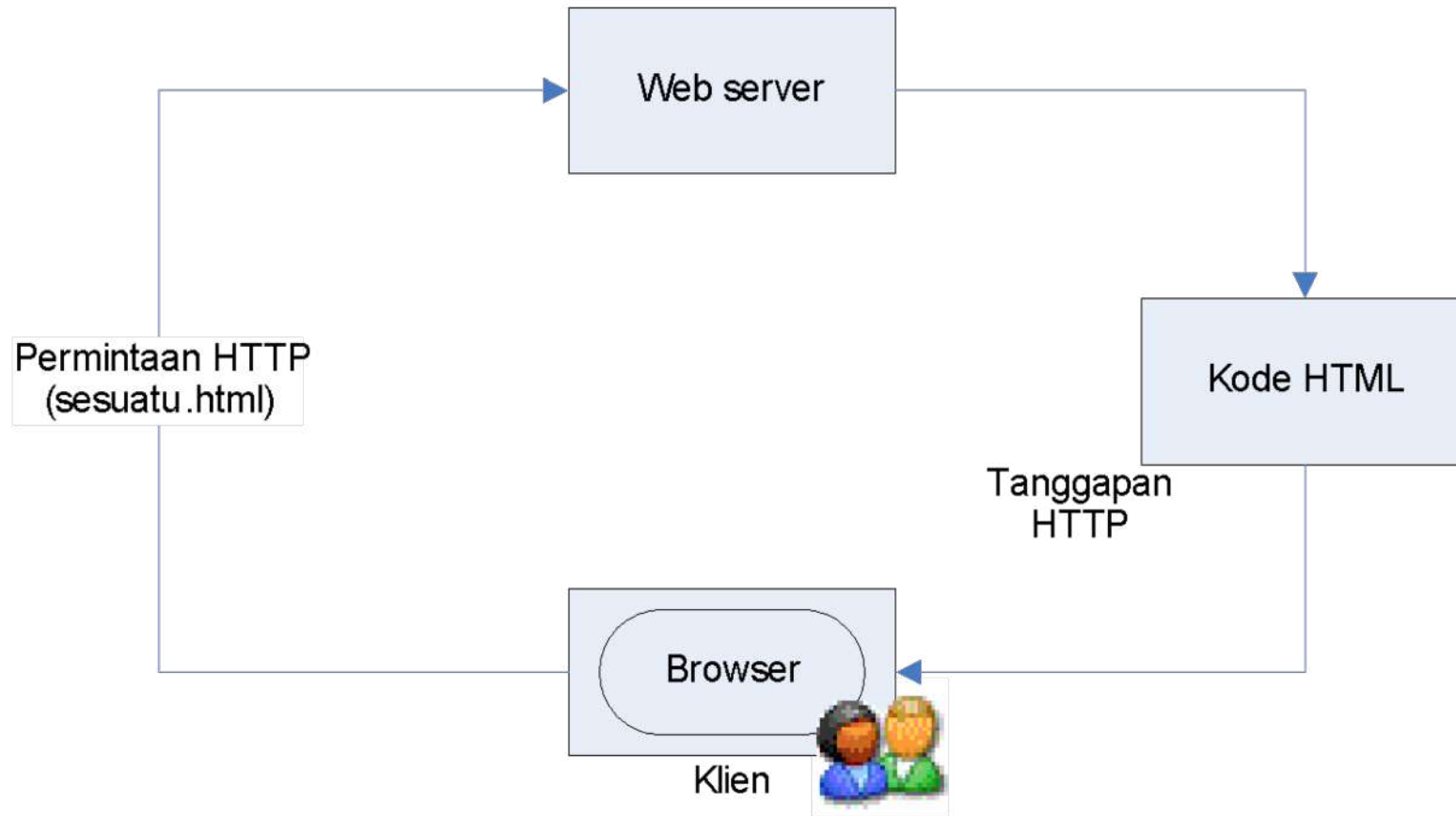


# Pengenalan PHP

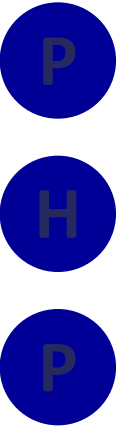
- **Server Side Programming** yang paling populer digunakan untuk membangun aplikasi berbasis web
- Setiap satu statement (perintah) biasanya diakhiri dengan titik-koma (;)
- **CASE SENSITIVE** untuk nama identifier yang dibuat oleh user (variable, konstanta, fungsi dll), namun **TIDAK CASE SENSITIVE** untuk identifier built-in dari PHP



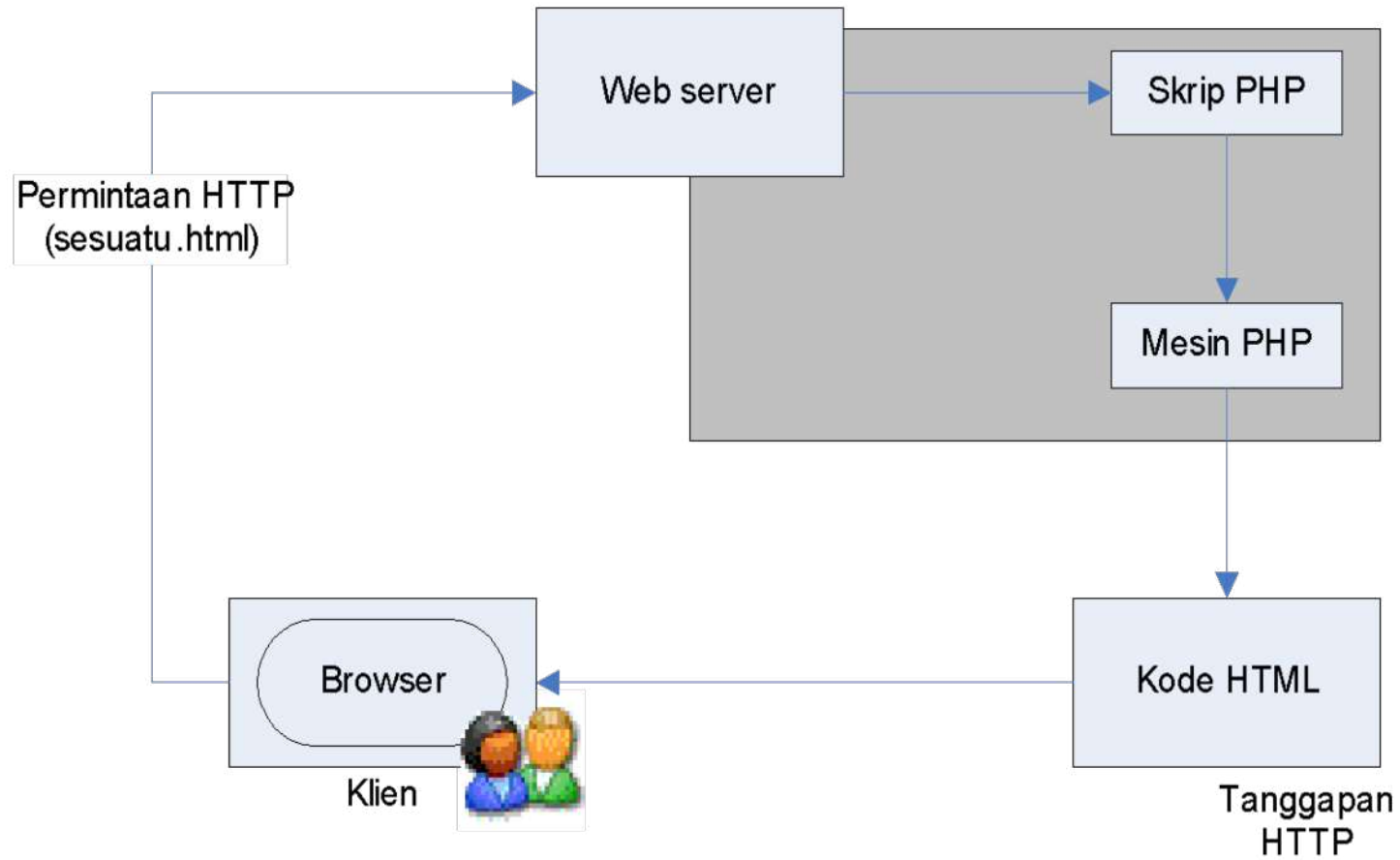
# Pengenalan PHP (Skema Kerja HTML)



**Skema HTML**



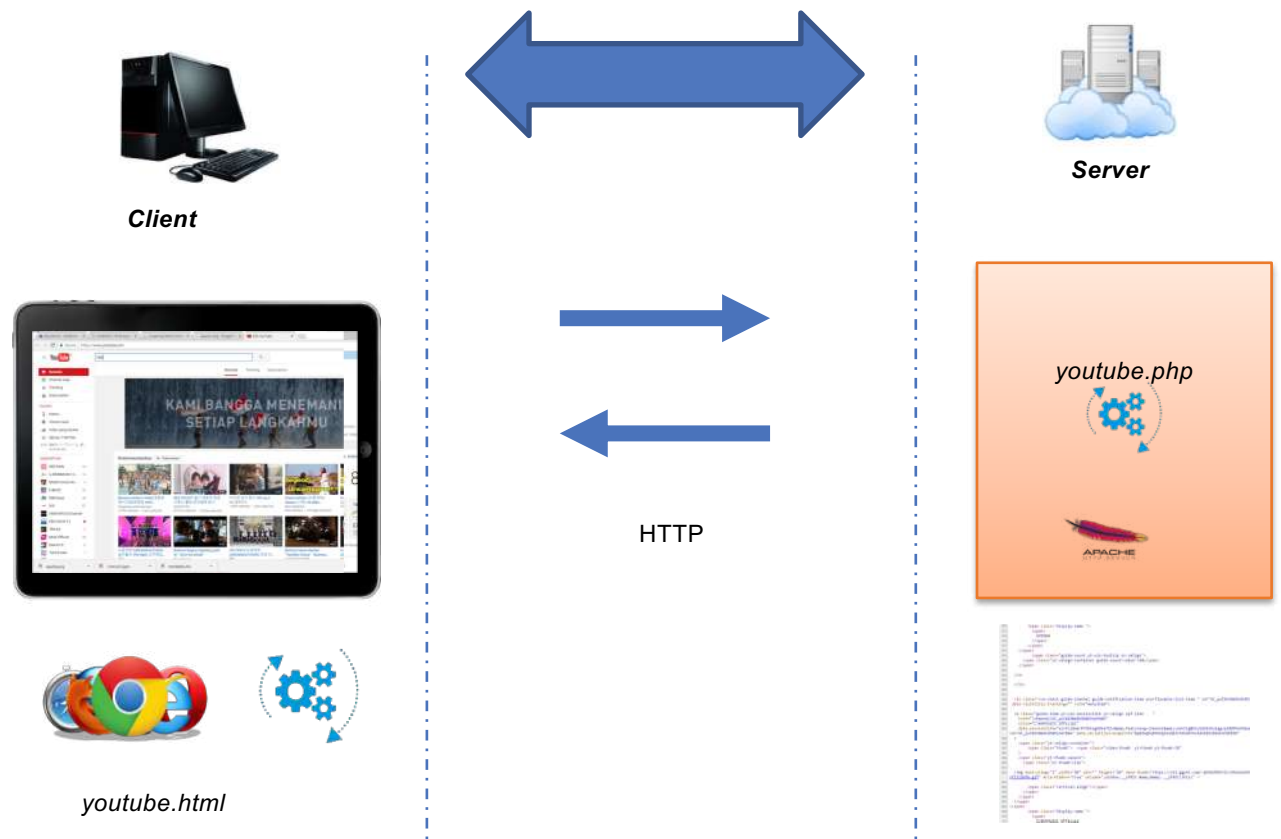
# Pengenalan PHP (Skema Kerja PHP)



**Skema PHP**



# Pengenalan PHP (Skema Kerja PHP: Contoh)

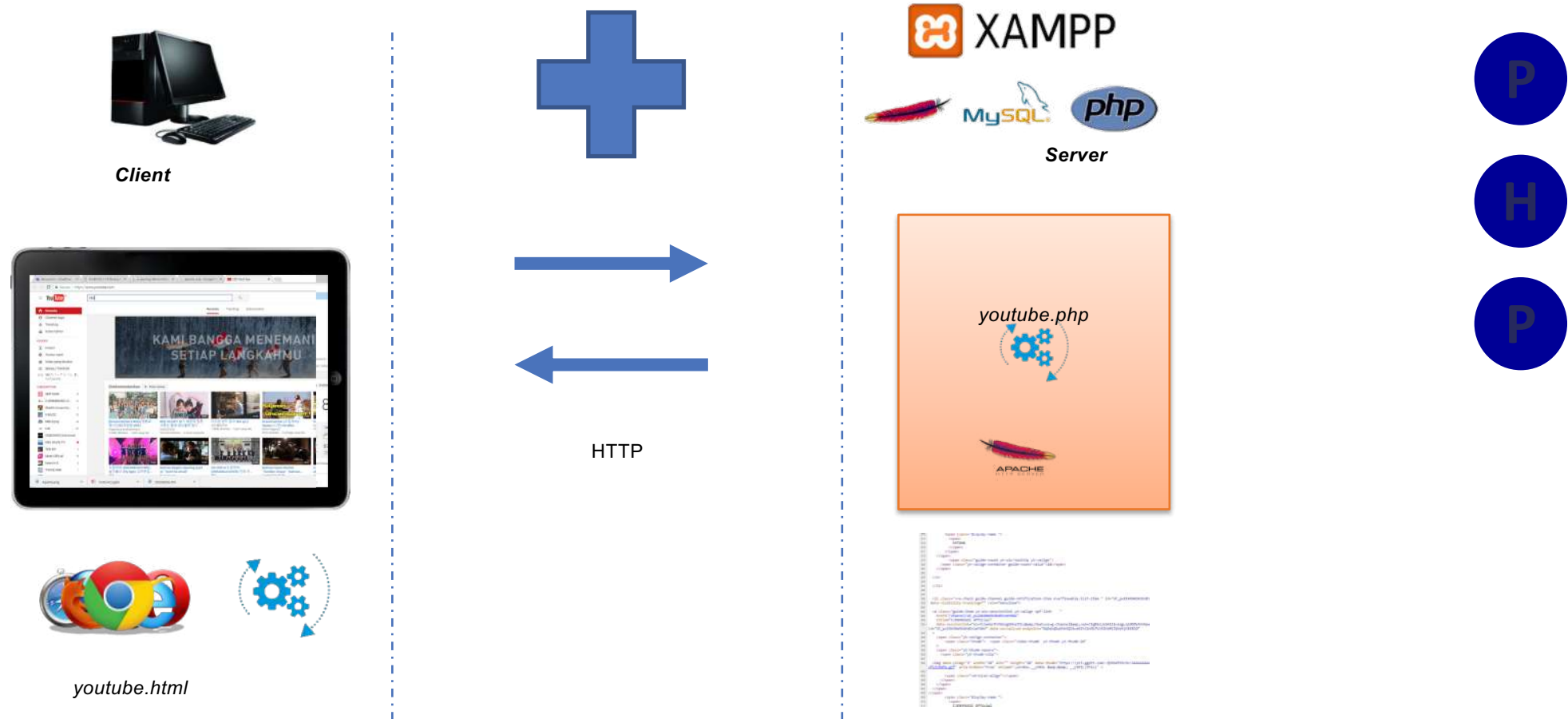


P  
H  
P

Skema PHP



# Pengenalan PHP (Skema Kerja PHP: Server Lokal)



# Struktur PHP

```
<?php  
  
//tuliskan syntax disini  
  
?>
```

```
<?php  
  
echo "Hello World!";
```

- '**<?php**' ini adalah pembuka program PHP. Pembuka ini wajib ada di setiap program PHP.
- '**echo "Hello World!";**' adalah sebuah statement atau perintah untuk menampilkan teks.
- Kenapa tidak ditutup dengan '**?>**' ? Tutup sebenarnya bersifat opsional. Tutup program dibutuhkan saat kita menggabungkan kode PHP dengan HTML.



# Struktur PHP: PHP dan HTML

```
<!DOCTYPE html>
<html>
  <head>
    <title><?php echo "Belajar PHP" ?></title>
  </head>
  <body>
    <?php
      echo "saya sedang belajar PHP<br>";
      echo "<p>Belajar PHP hingga jadi master</p>";
    ?>
  </body>
</html>
```

- Terdapat tag pembuka dan penutup php.
- PHP yang ditulis di dalam HTML, filenya harus disimpan dengan ekstensi .php bukan .html meskipun isinya HTML dan PHP.



# Struktur PHP: HTML dan PHP

```
<?php
echo '<div>';
echo '<p>This is a paragraph.</p>';
echo '<span>This is a span.</span>';
echo '</div>';
?>
```



- Tag-tag HTML bisa dihasilkan atau bisa dikeluarkan dari output PHP
- Seperti contoh di atas, dimana tag HTML dituliskan menggunakan perintah echo d PHP
- Hal ini penting ketika dibutuhkan dari proses PHP di program Anda diperlukan untuk menghasilkan output berupa HTML



# Struktur PHP: Sintaks PHP

```
<?php  
echo "ini statement 1";  
echo "ini statement 2";  
$a = $b + $c;
```

- Setiap statement PHP harus diakhiri tanda ;



# Struktur PHP: Sintaks PHP

```
<?php
```

```
$nama="rudy";
```

```
$NAMA="arie";
```

```
?>
```

```
<?php
```

```
$nama_depan="rudy";
```

```
$namaDepan="arie";
```

```
?>
```

- PHP adalah bahasa pemrograman yang bersifat **case sensitive**. Artinya, huruf besar (kapital) dan huruf kecil akan dibedakan

P

H

P

# Struktur PHP: Sintaks PHP

```
<?php

// ini adalah komentar
echo "Hello world";

/*
ini adalah komentar
yang lebih dari satu
baris
*/

?>
```

Komentar di PHP dapat ditulis dengan dua cara:

- Menggunakan tanda // untuk komentar satu baris;
- Menggunakan tanda /\* untuk komentar lebih dari satu baris.



# Variabel

```
<?php

$nama_barang = "Kopi C++";
$harga = 4000;
$stok = 40;
```

```
<?php

// membuat variabel baru
$stok = 40;

// mengisi ulang variabel dengan nilai baru
$stok = 34;
```

- ❑ Membuat variabel dengan tanda dolar (\$), lalu diikuti dengan nama variabelnya serta nilai yang ingin disimpan.
- ❑ Variabel dapat diisi ulang dengan nilai yang baru.
- ❑ Awal dari nama variabel tidak boleh menggunakan angka dan simbol, kecuali underscore.
- ❑ Nama variabel yang terdiri dari dua suku kata, bisa dipisah dengan underscore (\_) atau menggunakan style camelCase.
- ❑ Nama variabel bersifat Case Sensitive, artinya huruf besar dan huruf kecil dibedakan.
- ❑ Variabel harus diisi saat pembuatannya. Bila tidak ingin diisi, cukup isi dengan nilai kosong.
- ❑ Nama variabel bersifat Case Sensitive, artinya huruf besar dan huruf kecil dibedakan.





# Variabel

```
<?php

// membuat variabel baru
$nama_barang = "Minyak Goreng";
$harga = 15000;

// menampilkan isi variabel
echo "Ibu membeli $nama_barang seharga Rp $harga";
```

Ibu membeli Minyak Goreng seharga Rp 15000

```
$judul = "Belajar PHP dari nol sampai expert";
echo "Judul artikel: $judul";
```

```
$judul = "Tutorial PHP untuk Pemula";
echo 'Judul artikel: '.$judul;
```

**Mengambil nilai dari variabel dapat dilakukan dengan menuliskan namanya dalam perintah echo maupun ekspresi yang lain.**

# Konstanta

Konstanta digunakan untuk menyimpan nilai yang tidak pernah berubah.

Membuat konstanta dengan dua cara.

- Menggunakan fungsi `define()`;
- Menggunakan kata kunci `const`.

```
<?php  
  
define('NAMA_KONSTANTA','NILAI KONSTANTA');  
const KONSTANTA="nilai konstanta";  
  
echo "nilai konstanta: KONSTANTA";  
echo "NILAI KONSTANTA: NAMA_KONSTANTA";  
  
?>
```

# Tipe Data

Ada beberapa macam tipe data yang dapat disimpan dalam variabel:

- Tipe data char (karakter)
- Tipe data string (teks)
- Tipe data integer (angka)
- Tipe data float (pecahan)
- Tipe data boolean
- Tipe data objek
- Tipe data Array
- NULL
- dll.

PHP tidak harus mendeklarasikan tipe datanya secara eksplisit. Karena PHP sudah mampu mengenali tipe data dari nilai yang diberikan.

```
<?php

$jenis_kelamin='L'; // Tipe data Char
$nama_lengkap="Arie Ariyanto";// Tipe Data String
$umur=17; // Tipe data Integer
$berat=65.5; // Tipe data Float
$tinggi=178.5; // Tipe data Float
$menikah=false; // Tipe data boolean

echo "nama lengkap: $nama_lengkap <br>";
echo 'jenis kelamin: '.$jenis_kelamin.'<br>';
echo "umur: ".$umur." tahun<br>";
echo "berat: $berat kg <br>";
echo 'tinggi: '.$tinggi.'cm <br>';
echo "status menikah:$menikah";

?>
```

# Tipe Data

```
$huruf = 'E';
```

```
$alamat = "Jl. Mawar, Jakarta";
```

```
$nilai = 98; // angka positif  
$poin = -31; // angka negatif
```

```
$isActive = false;  
$menikah = true;
```

- Char adalah tipe data yang terdiri dari karakter. Penulisannya diapit dengan tanda petik satu.
- String adalah tipe data yang terdiri dari kumpulan karakter. Penulisannya diapit dengan tanda petik ganda.
- Integer adalah tipe data angka. Penulisannya tidak menggunakan tanda petik.
- Tipe data boolean adalah tipe data yang hanya bernilai true dan false. Penulisan true dan false tidak diapit dengan tanda petik.



# Tipe Data

```
$panjang = 233.12;  
$kapasitas = 13232.12;
```

```
$jarak = 1.2E-5;
```

```
echo sprintf('%f', $a);  
// batasi angka di belakang koma  
echo sprintf('%.3f', $a);
```

- Float adalah tipe data bilangan pecahan ditulis tanpa tanda petik.
- E-5 artinya eksponen dari 10.
- Contoh di atas akan sama dengan  $1.2 \times 10^{-5}$ . Kalau kita jabarkan akan menjadi 0.000012.
- Agar format float tidak tercetak dalam notasi E, kita bisa gunakan fungsi sprintf().

# Tipe Data

```
$minuman = array("Kopi", "Teh", "Jus Jeruk");  
$makanan = ["Nasi Goreng", "Soto", "Bubur"];
```

```
<?php  
  
$data=[  
    "nama"=>"Arie Ariyanto",  
    "usia"=> 17,  
    "berat" =>65.5,  
    "tinggi"=> 178.5,  
    "menikah"=> false  
];  
  
?>
```

Array adalah tipe data yang berisi sekumpulan nilai.

# Operator: Aritmatika

Nama Operator	Simbol
Penjumlahan	`+`
Pengurangan	`-`
Perkalian	`*`
Pemangkatan	`**`
Pembagian	`/`
Sisa Bagi	`%`

```
$a = 5;  
$b = 2;  
  
// penjumlahan  
$c = $a + $b;  
echo "$a + $b = $c";  
echo "<hr>";
```

# Operator: Penugasan

Nama Operator	Sombol
Pengisian Nilai	`=`
Pengisian dan Penambahan	`+=`
Pengisian dan Pengurangan	`-=`
Pengisian dan Perkalian	`*=`
Pengisian dan Pemangkatan	`**=`
Pengisian dan Pembagian	`/=`
Pengisian dan Sisa bagi	`%=`
Pengisian dan Peggabungan (string)	`.=`

```
$speed = 83;  
  
// ini opertor aritmatika  
$speed = $speed + 10;  
  
// maka nilai speed akan samadengan 83 + 10 = 93  
  
// ini operator penugasan  
$speed += 10;  
  
// sekarang nilai speed akan menjadi 93 + 10 = 103
```

# Operator: Increment & Decrement

```
$score = 0;  
  
$score++;  
$score++;  
$score++;  
  
echo $score;
```

Operator increment dan decrement merupakan operator yang digunakan untuk menambah +1 (tambah satu) dan mengurangi -1 (kurangi dengan satu).

- Operator increment menggunakan simbol ++
- Operator decrement menggunakan simbol --.

# Operator: Relasi

Nama Operator	Simbol
Lebih Besar	`>`
Lebih Kecil	`<`
Sama Dengan	`==` atau `===`
Tidak Sama dengan	`!=` atau `!==`
Lebih Besar Sama dengan	`>=`
Lebih Kecil Sama dengan	`<=`

```
<?php
$total_belanja = 150000;

if($total_belanja > 100000){
    echo "Anda dapat hadiah!";
}
```

# Operator: Logika

Nama Operator	Simbol
Logika AND	`&&`
Logika OR	`  `
Negasi/kebalikan/ NOT	`!`

AND		Hasil
true	true	true
true	false	false
false	true	false
false	false	false

OR		Hasil
true	true	true
true	false	true
false	true	true
false	false	false

NOT	Hasil
true	false
false	true

```
<?php

$a = true;
$b = false;

// variabel $c akan bernilai false
$c = $a && $b;
printf("%b && %b = %b", $a,$b,$c);
echo "<hr>";
```



# Operator: Bitwise

Nama	Simbol di Java
AND	`&`
OR	` `
XOR	`^`
Negasi/kebalikan	`~`
Left Shift	`<<`
Right Shift	`>>`

```
<?php

$a = 60;
$b = 13;

// bitwise AND
$c = $a & $b;
echo "$a & $b = $c";
echo "<br>";
```

# Operator: Ternary

Operator Ternary

kamu suka aku ? ya : tidak;

jawaban benar      jawaban salah

```
<?php

$suka = true;

// menggunakan operator ternary
$jawab = $suka ? "iya" : "tidak";

// menampilkan jawaban
echo $jawab;
```



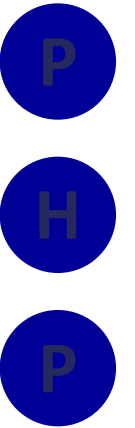
# Percabangan

Insert the Subtitle of Your Presentation

# Struktur Kontrol

Struktur kontrol digunakan untuk mengatur alur logika program agar sesuai dengan kenyataan. Secara mendasar struktur program memiliki kombinasi struktur kontrol sebagai berikut

- Urutan (Sequence)
- Pemilihan / Percabangan (Selection)
- Pengulangan (Iteration)



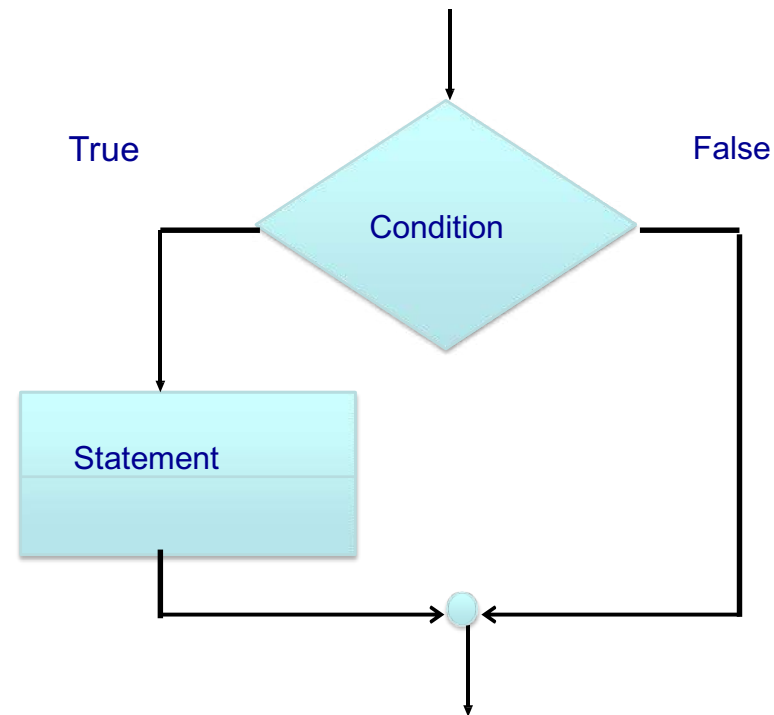
# Struktur Kontrol Percabangan

- Struktur Kontrol If
- Struktur Kontrol If – else
- Struktur Kontrol if – else if – else
- Struktur Kontrol switch – case



# Struktur Kontrol IF

Struktur IF dalam PHP merupakan struktur kontrol pemilihan atau pengambilan keputusan yang digunakan untuk pemeriksaan apakah perintah perintah yang ada di blok if dikerjakan atau tidak.



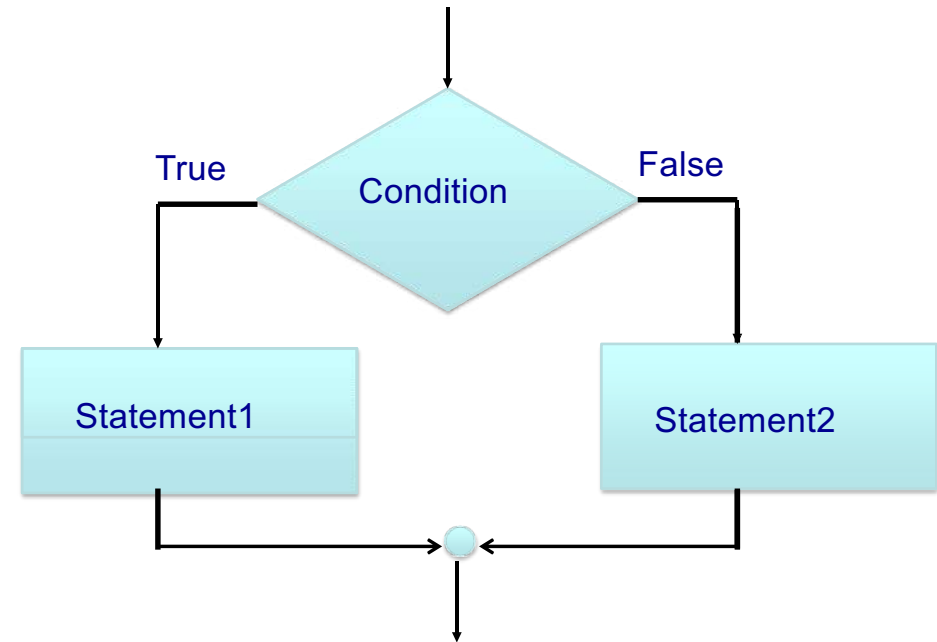
# Struktur Kontrol IF: Contoh

```
<?php
    if(isset($_GET["myNumber"])) {
        $a = $_GET["myNumber"];
        echo "Hasilnya".$a;
        if(a>10)
            echo "Lebih dari 10";
    }
?>
```



# Struktur Kontrol IF – ELSE

- Kondisi IF yang menyertakan ELSE disebut IF – ELSE
- Kondisi ELSE dieksekusi ketika kondisi dalam pernyataan IF bernilai FALSE

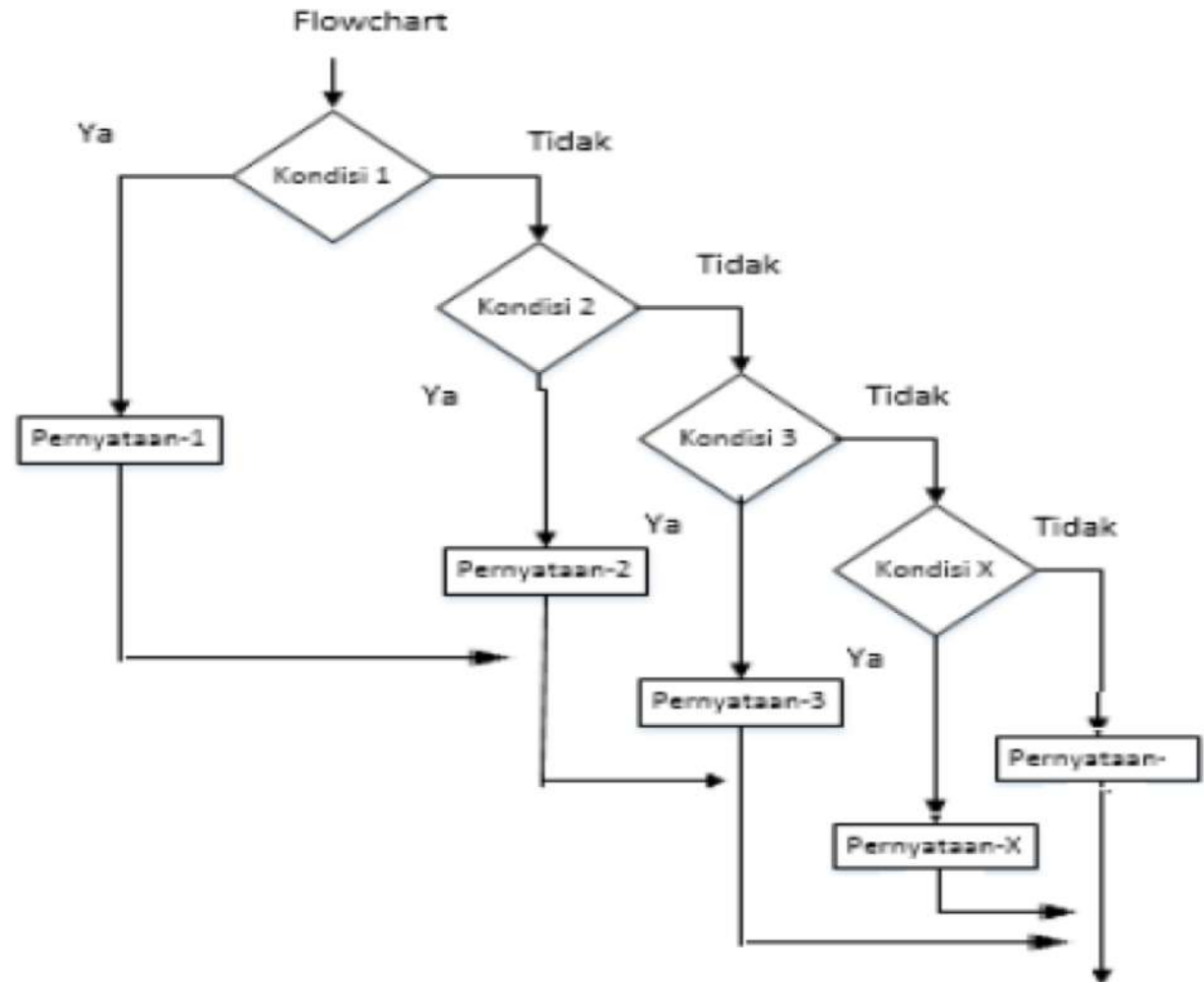


# Struktur Kontrol IF – ELSE : Contoh

```
$a = 1;  
$b = 2;  
if($a > $b)  
    echo "a is greater than b";  
else  
    echo "a is less than or equal to b";
```

# Struktur Kontrol IF – ELSE IF – ELSE

- Struktur ini merupakan pengembangan dari struktur IF – ELSE
- Jumlah ekspresi yang diuji **lebih dari 2**, jadi struktur IF – ELSE IF – ELSE ini mengharuskan proses pemeriksaan kembali ekspresi apabila nilai ekspresi pada if bernilai salah.



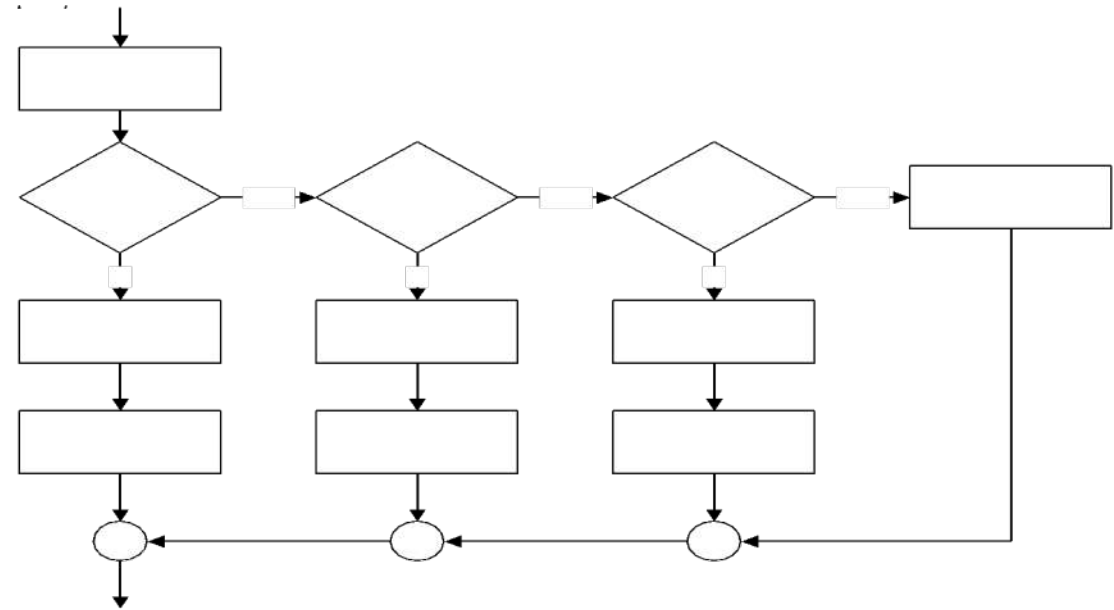
# Struktur Kontrol IF – ELSE IF – ELSE : Contoh

```
$a =19;  
if($a == 1)  
    echo "one";  
elseif($a == 2)  
    echo "two";  
elseif($a == 3)  
    echo "three";  
elseif($a == 4)  
    echo "four";  
else  
    echo "more than four";
```

# Struktur Kontrol Switch – Case

- Struktur ini merupakan pengembangan dari struktur IF – ELSE
- Jumlah ekspresi yang diuji **lebih dari 2**, jadi struktur IF – ELSE IF – ELSE ini mengharuskan proses pemeriksaan kembali ekspresi apabila nilai ekspresi pada if bernilai salah.

```
switch ($variable_name)
{
case valueA:
    statements;
    break; // optional
default:
    statements;
}
```



# Struktur Kontrol Switch – Case

- Digunakan ketika mempunyai **banyak kemungkinan tindakan** yang harus dilakukan pada kondisi yang berbeda-beda.
- Sintaks pemilihan ini akan menjalankan salah satu dari beberapa pernyataan “**case**” sesuai dengan nilai kondisi yang ada di dalam “**switch**”. Selanjutnya proses akan dilanjutkan sampai ditemukan pernyataan “**break**”. Namun, jika tidak ada nilai pada case yang sesuai dengan nilai kondisi, maka proses akan dilanjutkan ke pernyataan yang ada di dalam “**default**”.

# Struktur Kontrol Switch – Case: Contoh

```
$a = 2;  
switch($a){  
    case 0:  
        echo "a equals to 0";  
        break;  
    case 1:  
        echo "a equals to 1";  
        break;  
    default:  
        echo "a is greater than 1";  
}
```



# Ternary Operator

- Pada dasarnya merupakan operator, dan bisa digunakan untuk melakukan seleksi kondisi atau percabangan
- Cocok untuk model percabangan yang sederhana

sintaks (Condition) ? (kondisi jika true) : (kondisi jika false)

```
$nilai = 75;  
$status = $nilai > 60 ? "Lulus" : "Tidak Lulus";
```

P

H

P

# Contoh Studi Kasus (1)

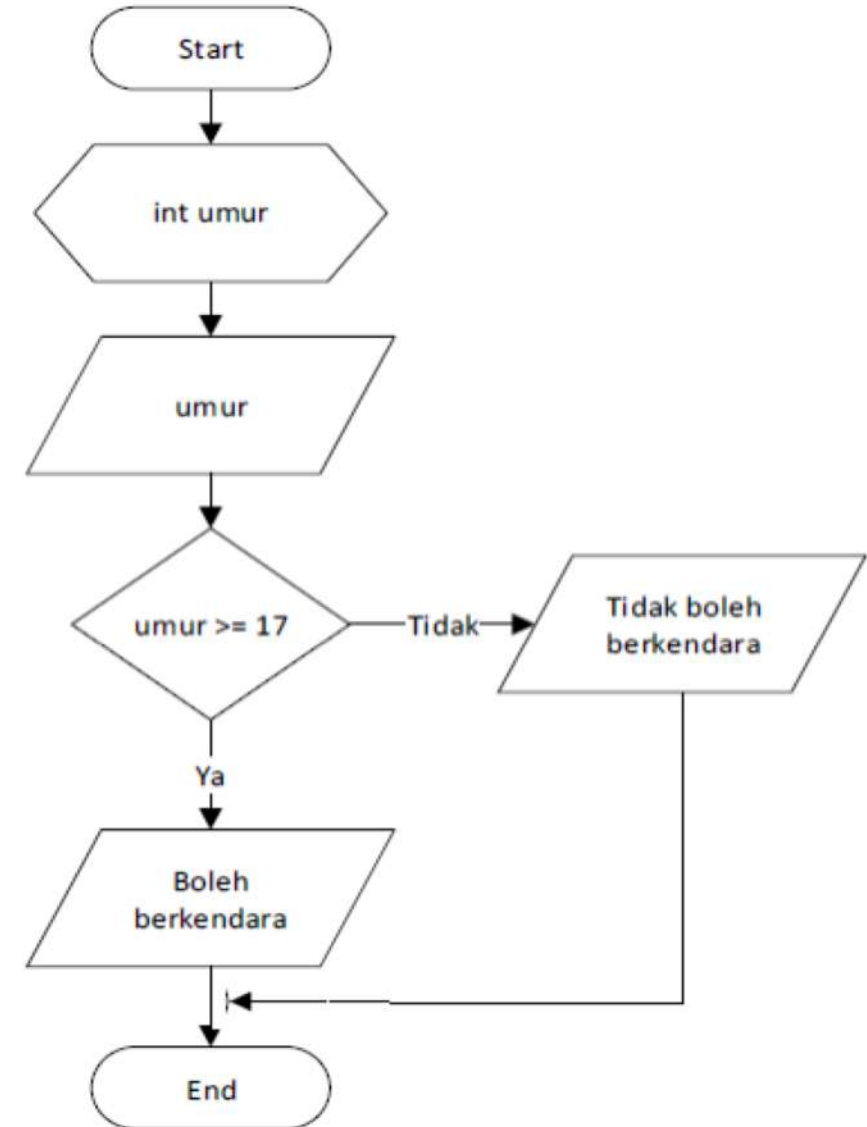
Di dalam aturan tata tertib berkendara kendaraan bermotor maka terdapat aturan dimana orang yang boleh berkendara bermotor yaitu orang yang umurnya minimal 17 tahun

# Contoh Studi Kasus (1)

Tentukan Kondisi : Usia  $> 17$

Jika kondisi “Benar” / “True” Tentukan  
apa yang akan dilakukan ☐ Boleh  
berkendara

Jika kondisi “Salah” / “Benar” Tentukan  
apa yang akan dilakukan ☐ Tidak Boleh  
berkendara

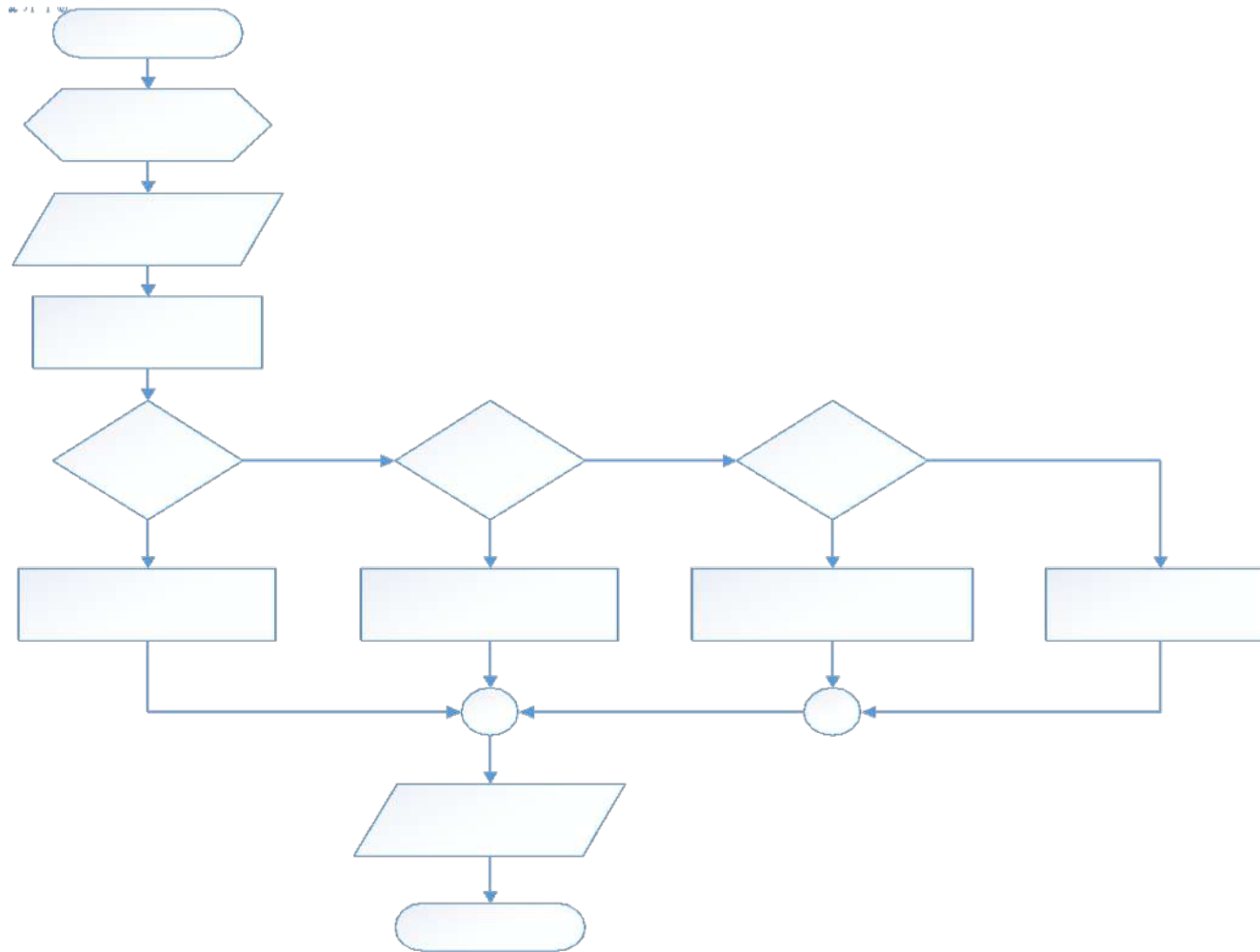


## Contoh Studi Kasus (2)

Salah satu cerminan sikap warga negara yang baik adalah menaati peraturan tata tertib di jalan raya yaitu rambu-rambu lalu lintas. Kamu adalah pengendara sepeda motor yang sedang melintas di jalan raya dan bertemu lampu lalu lintas. Buat flowchart untuk menentukan apa yang harus kamu lakukan untuk setiap kondisi lampu lalu lintas!



# Contoh Studi Kasus (2)



# Percabangan Bersarang

- Pemilihan bersarang (NESTED IF) merupakan jenis pemilihan yang digunakan untuk mengambil keputusan dalam bentuk level (bertingkat)
- Di dalam suatu pernyataan IF (atau IF-ELSE) bisa saja terdapat pernyataan IF (atau IF-ELSE) yang lain

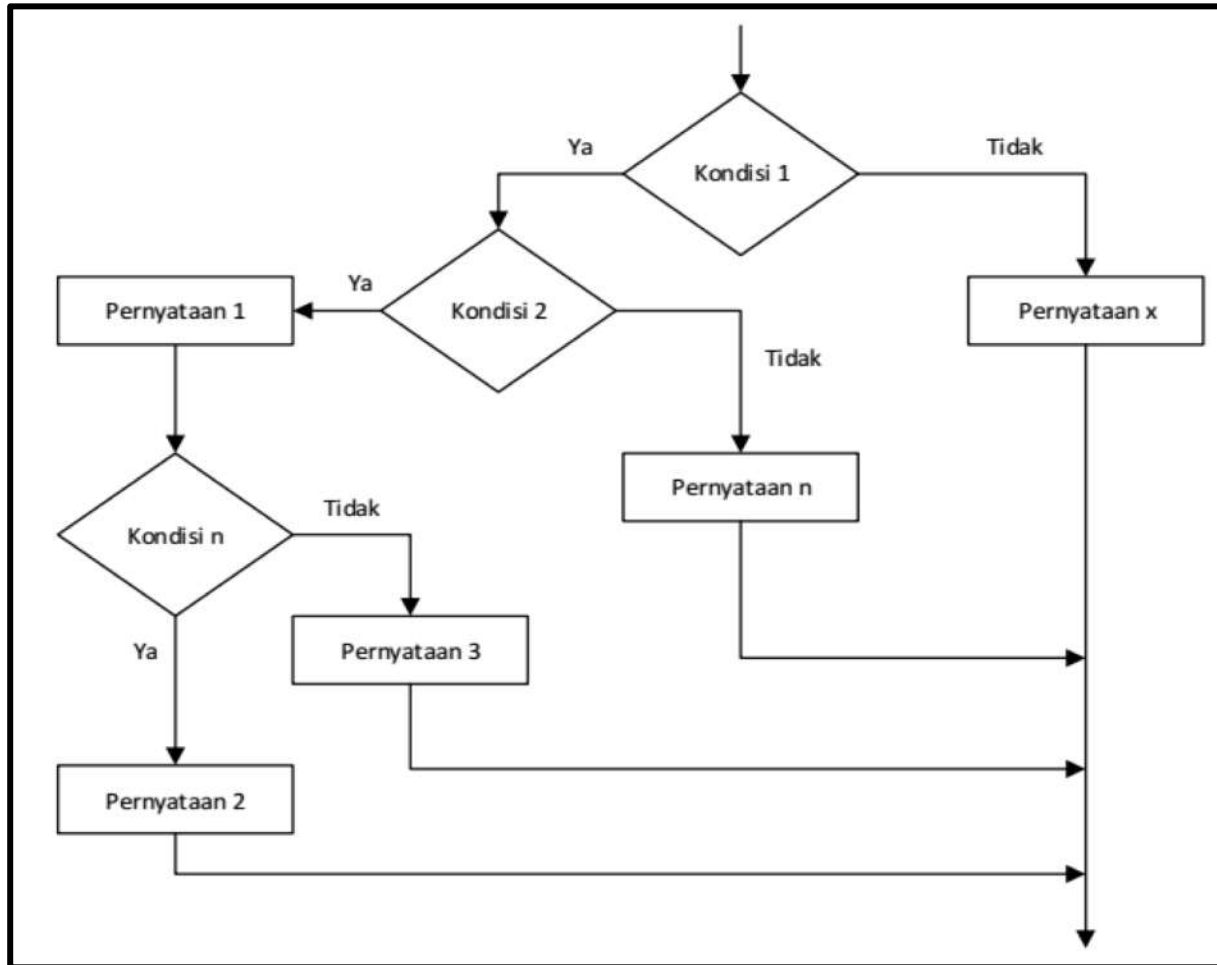
```
if (kondisi 1){  
    if (kondisi 2){  
        pernyataan 1;  
        ...  
        if (kondisi n){  
            pernyataan 2;  
        } else {  
            pernyataan 3;  
        }  
    } else {  
        pernyataan n;  
    }  
} else {  
    pernyataan x;  
}
```

# Percabangan Bersarang

- Kondisi yang akan diseleksi pertama kali adalah kondisi **IF** yang berada di posisi terluar (**kondisi 1**).
- Jika **kondisi 1** bernilai **salah**, maka pernyataan **ELSE** terluar (pasangan dari IF yang bersangkutan) yang akan diproses. Namun, jika pernyataan ELSE (pasangan dari IF) tidak ditulis, maka penyeleksian kondisi akan dihentikan.
- Jika ternyata **kondisi 1** bernilai **benar**, maka kondisi berikutnya yang lebih dalam (**kondisi 2**) akan diseleksi. Jika kondisi 2 bernilai salah, maka pernyataan ELSE (pasangan dari IF yang bersangkutan) yang akan diproses. Namun, jika pernyataan ELSE (pasangan dari IF) tidak ditulis, maka penyeleksian kondisi akan dihentikan.



# Percabangan Bersarang



# Percabangan Bersarang: Contoh

- Ketika seseorang melakukan pembayaran di kasir. Kasir akan memberikan pertanyaan:

**Apakah pelanggan mempunyai kartu anggota?**

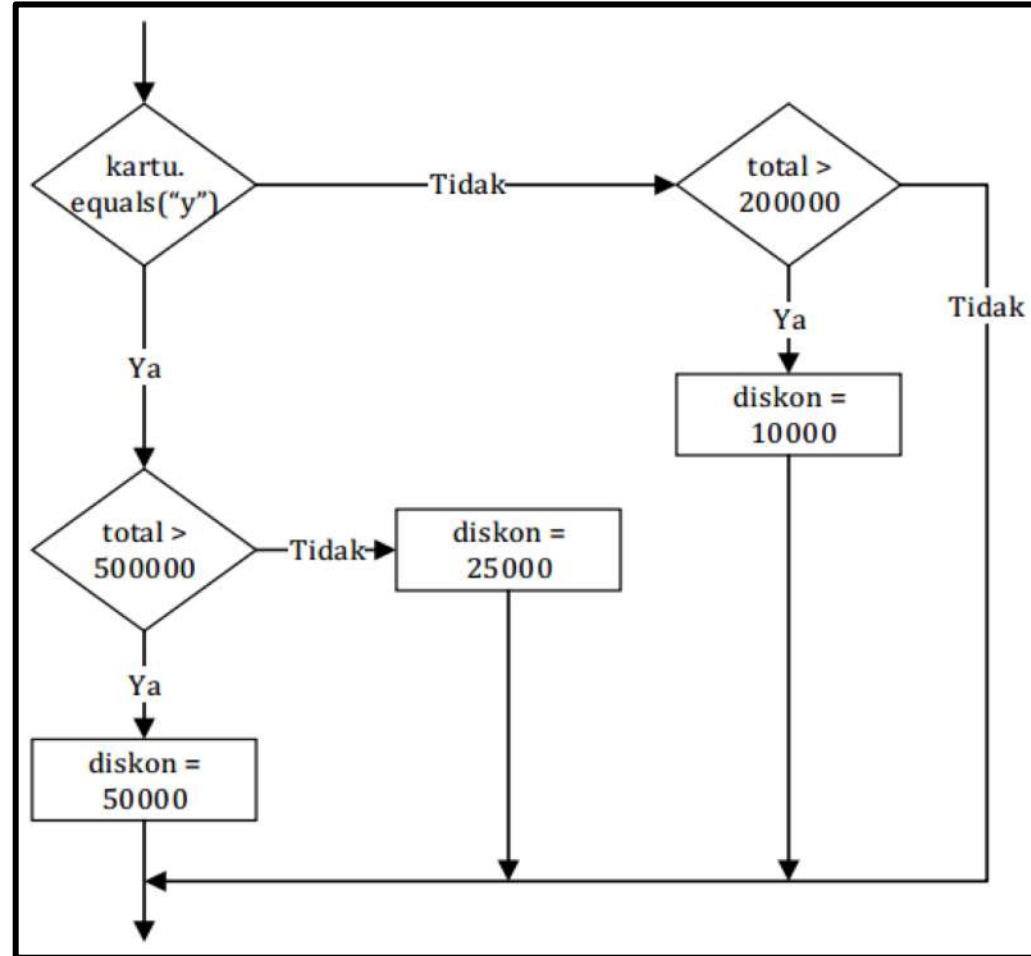
- **TRUE:**

- *Apakah total harga barang belanjaan lebih dari Rp 500.000?*
  - *TRUE: Pelanggan mendapatkan diskon Rp 50.000*
  - *FALSE: Pelanggan mendapatkan diskon Rp 25.000*

- **FALSE:**

- *Apakah total harga barang belanjaan lebih dari Rp 200.000?*
  - *TRUE: Pelanggan mendapatkan diskon Rp 10.000*
  - *FALSE: Pelanggan tidak mendapatkan diskon*

# Percabangan Bersarang: Contoh



# Ekspresi Logika

- Terdapat 3 jenis operator logika yang digunakan pada pernyataan IF-ELSE, yaitu:
  - && : AND
  - || : OR
  - ! : NOT
- **Ekspresi logika** adalah ekspresi yang menggunakan satu atau lebih operator logika.
- Operator yang diterapkan pada ekspresi logika akan dievaluasi dari **kiri ke kanan**



# Ekspresi Logika

- Ketika mengevaluasi (**e1 && e2**), jika **e1** menghasilkan **FALSE**, maka **e2 tidak akan dievaluasi**. Dengan demikian, nilai seluruh ekspresi (e1 && e2) akan dianggap salah
- Namun, jika **e1** menghasilkan **TRUE**, maka selanjutnya **e2 akan dievaluasi** untuk menentukan nilai seluruh ekspresi
- Contoh:

```
if(kecepatan == 0 && mesinOn == true)  
    echo "Matikan mesin";
```



# Ekspresi Logika

- Ketika mengevaluasi (**e1 || e2**), jika **e1** menghasilkan **TRUE**, maka **e2 tidak akan dievaluasi**. Dengan demikian, nilai seluruh ekspresi (e1 || e2) akan dianggap benar
- Namun, jika **e1** menghasilkan **FALSE**, maka selanjutnya **e2 akan dievaluasi** untuk menentukan nilai seluruh ekspresi
- Contoh:

```
if(kecepatan == 0 || mesinOn == true)
    echo "Matikan mesin";
```

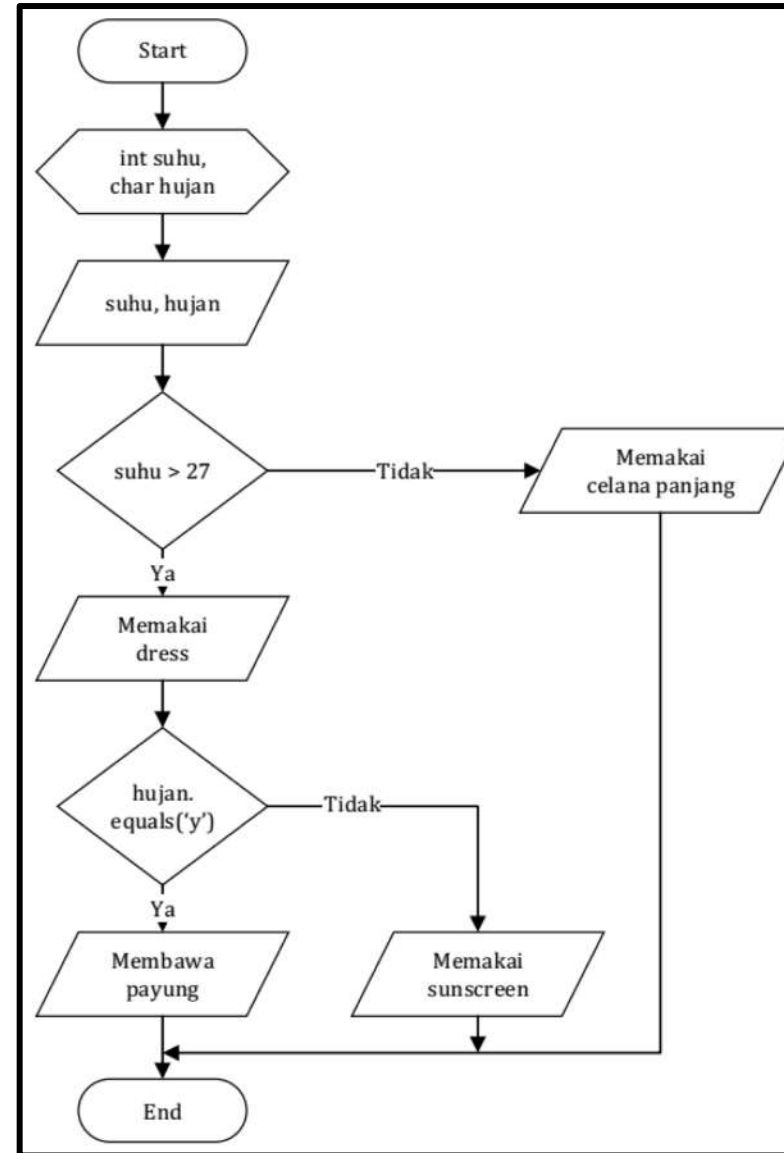


# Contoh Studi Kasus

- Sebuah sistem dibuat untuk menentukan pakaian dan peralatan yang harus dibawa pengguna sesuai dengan kondisi cuaca. Jika suhu lebih dari 27°C, maka pengguna disarankan memakai dress, kemudian dilakukan pengecekan apakah saat ini hujan, jika hujan maka pengguna disarankan untuk membawa payung, sedangkan jika tidak hujan maka pengguna disarankan untuk memakai sunscreen. Namun, jika suhu kurang dari atau sama dengan 27°C, maka pengguna disarankan memakai celana panjang
- Buatlah flowchart untuk sistem tersebut!



# Contoh Studi Kasus







# Perulangan

Insert the Subtitle of Your Presentation

# Struktur Kontrol

- Struktur kontrol digunakan untuk mengatur alur logika program agar sesuai dengan kenyataan. Secara mendasar struktur program memiliki kombinasi struktur kontrol sebagai berikut
- Urutan (Sequence)
- Pemilihan / Percabangan (Selection)
- Perulangan (Iteration)

# Definisi Perulangan

- Perintah perulangan atau iterasi (loop) adalah perintah untuk mengulang satu atau lebih statement sebanyak beberapa kali
- Loop statement digunakan agar kita tidak perlu menuliskan satu/sekumpulan statement berulang-ulang. Dengan begitu maka kesalahan pengetikan bisa dikurangi
- Tipe perulangan:
  - Definite loop
  - Indefinite loop

# Tipe Perulangan – Definite Loop

- Perulangan yang jumlah eksekusinya **telah diketahui sebelumnya**
- Biasanya ditandai dengan “ulangi sebanyak \_\_ kali”
- Contoh:
  - Ulangi pernyataan ini sebanyak n kali
  - Ulangi pernyataan ini untuk setiap bilangan genap antara 8 dan 26

# Tipe Perulangan – Indefinite Loop

- Perulangan yang jumlah eksekusinya **tidak dapat ditentukan sebelum dilakukan**
- Perulangan dieksekusi selama kondisi bernilai benar (TRUE), atau sampai kondisi menjadi salah (FALSE)
- Contoh:
  - Ulangi pernyataan ini selama bilangan n bukan bilangan prima
  - Ulangi pernyataan ini sampai pengguna memasukkan bilangan bulat yang valid

# Struktur Kontrol Perulangan

- Struktur Kontrol For Loop
- Struktur While Loop
- Struktur Kontrol Do...While
- Struktur Kontrol Foreach Loop
- Pernyataan Break
- Pernyataan Continue

# Struktur Kontrol FOR

- Struktur kontrol for adalah salah satu bentuk **perintah pengulangan yang jumlah perulangannya dapat ditentukan** berapa kali melalui bilangan pencacah (counter).

```
<?php
```

```
    for($i = 1; $i < 10; $i++) {  
        echo "The number is $i ";  
    }
```

```
?>
```

# Struktur Kontrol FOR

- FOR umumnya digunakan pada pengulangan yang jumlah perulangannya sudah pasti atau sudah diketahui sebelumnya
- Sintaks FOR

for (*inisialisasi*; *kondisi*; *update*) statement;

atau:

```
for (inisialisasi; kondisi; update){  
    statement1;  
    statement2;  
    .....  
}
```



# Struktur Kontrol FOR

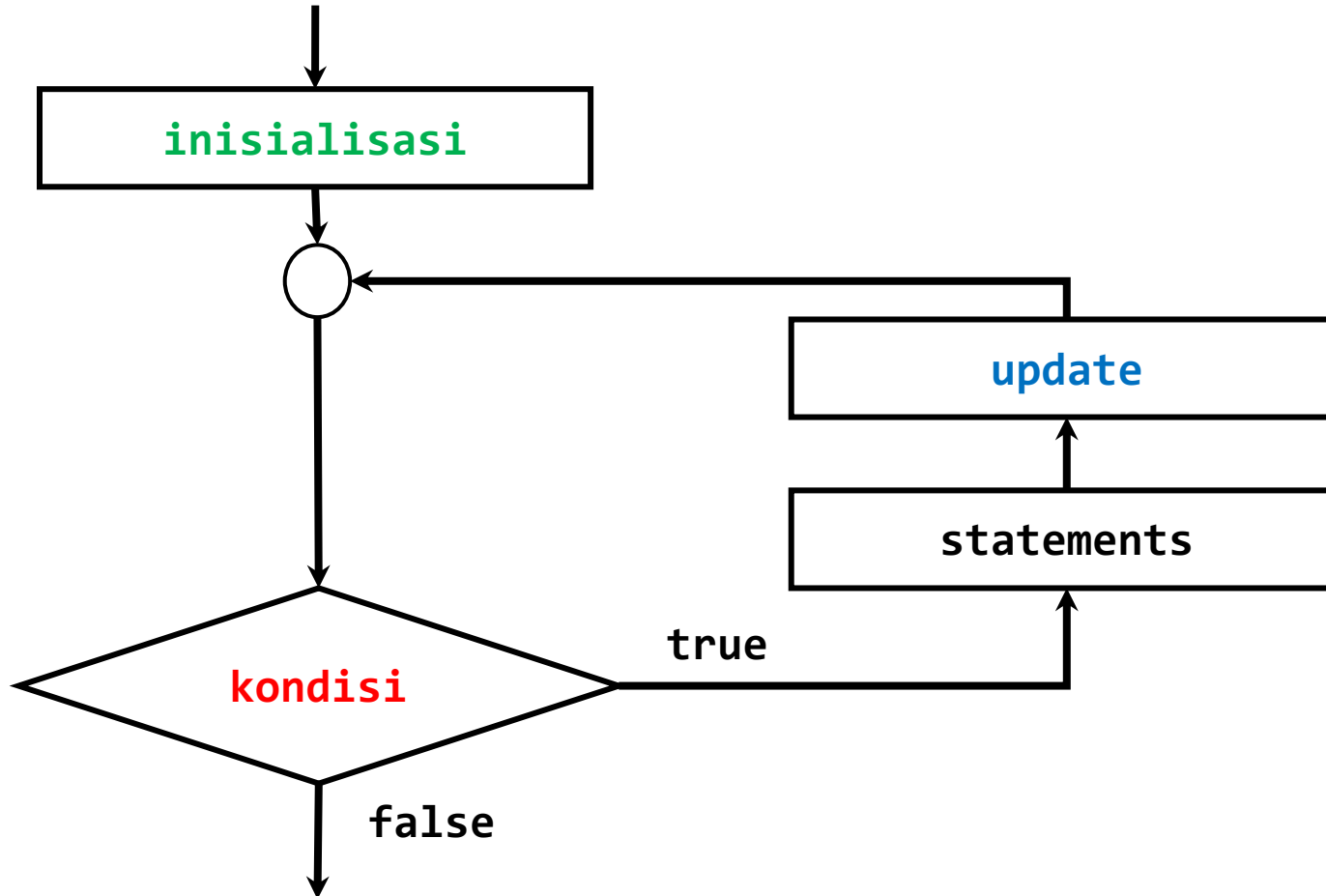
- **inisialisasi**: deklarasi dan inisialisasi variabel counter (variabel pengontrol perulangan)
- **kondisi**: batas atau syarat agar perulangan tetap dieksekusi
- **update**: perubahan nilai variabel counter pada setiap putaran perulangan (increment atau decrement)

**inisialisasi** dan **update** bersifat optional (boleh ada atau tidak)

# Struktur Kontrol FOR

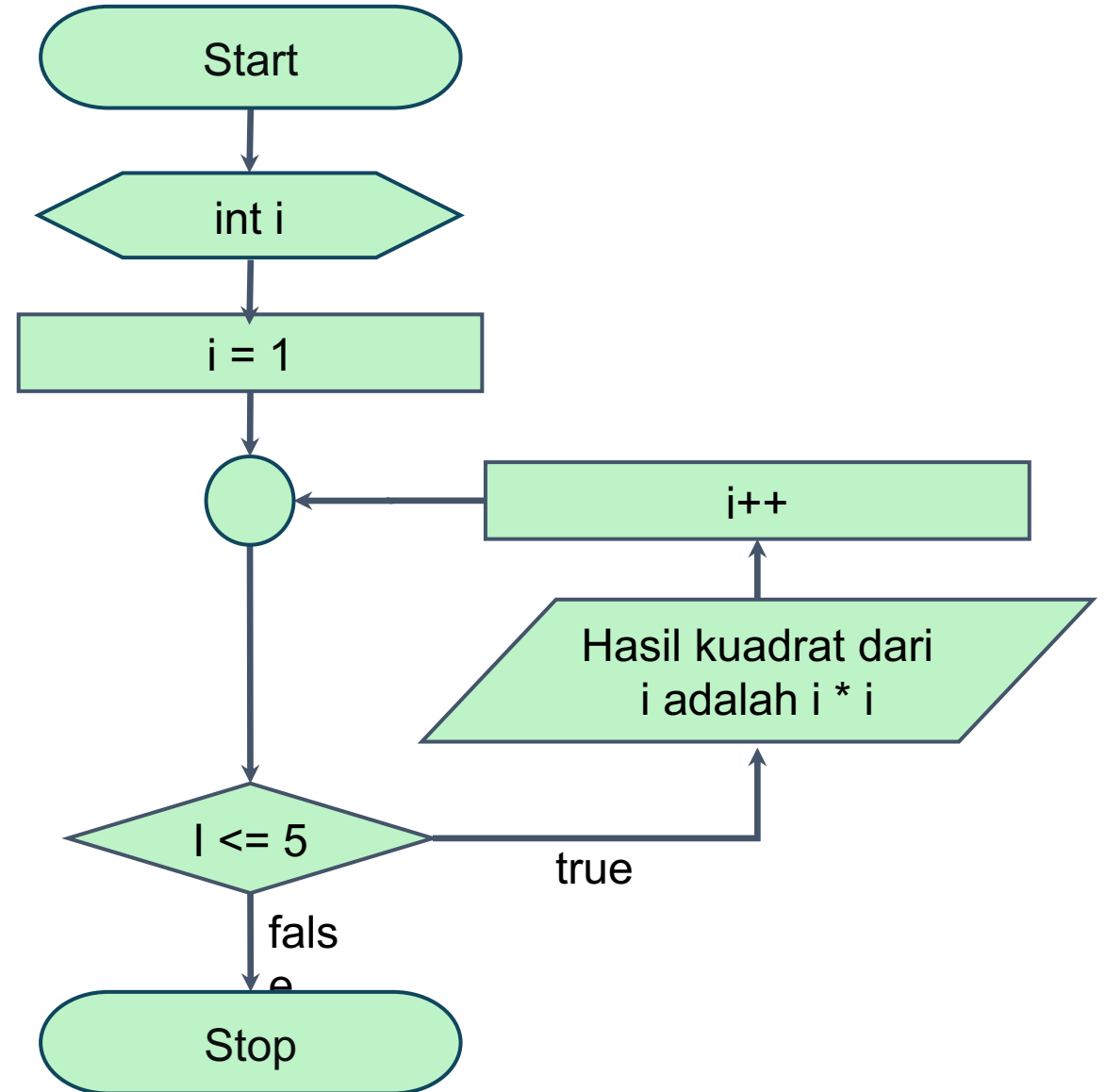
1. Perulangan diawali dengan melakukan **inisialisasi**
2. Evaluasi **kondisi**
  - Jika kondisi bernilai TRUE, eksekusi semua statement di dalam perulangan. Lakukan **update**. Ulangi kembali langkah nomor 2
  - Jika kondisi bernilai FALSE, hentikan perulangan

# Struktur Kontrol FOR



# Contoh Perulangan FOR

Buatlah flowchart dan kode program untuk menampilkan bilangan dan hasil kuadratnya dengan rentang nilai bilangan 1 sampai 5!



# Contoh Perulangan FOR

## Kode Program

```
<?php
for($i = 1; $i <= 5; $i++) {
    echo "Hasil kuadrat dari $i =" . ($i*$i) . "<br>";
}
?>
```

## Output

```
Hasil kuadrat dari 1 adalah 1
Hasil kuadrat dari 2 adalah 4
Hasil kuadrat dari 3 adalah 9
Hasil kuadrat dari 4 adalah 16
Hasil kuadrat dari 5 adalah 25
```

# Variasi Perulangan FOR – Variasi 1

**inisialisasi** dan **update** boleh terdiri dari beberapa ekspresi yang dipisahkan dengan tanda koma

Contoh:

```
for($i = 1, $j=10; $i < $j; $i++, $j--) {  
    echo $i." - ".$j."<br>";  
}
```

Output

1	-	10
2	-	9
3	-	8
4	-	7
5	-	6

# Variasi Perulangan FOR – Variasi 2

- **inisialisasi** dan **update** dapat dikosongkan, sesuai dengan kebutuhan
- Contoh:

```
$bil = 1;  
$kondisi = true;  
for(;$kondisi;){  
    echo $bil."<br>";  
    if($bil==10)  
        $kondisi = false;  
    $bil++;  
}
```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10

# Struktur kontrol WHILE

- While adalah bentuk perulangan yang memungkinkan **blok perintah dikerjakan berulang selama kondisi ekspresi masih benar**.
- Jadi selama kondisi yang ditentukan masih bernilai benar (true) maka perintah tersebut akan terus berulang hingga program mendapati bahwa perintah tersebut telah bernilai salah (false).
- WHILE cocok digunakan untuk perulangan yang **jumlahnya tidak diketahui sebelumnya** (indefinite loop)

```
<?php
```

```
    $x=1;
```

```
    while($x <= 5) {
```

```
        echo "The number is: $x <br>";
```

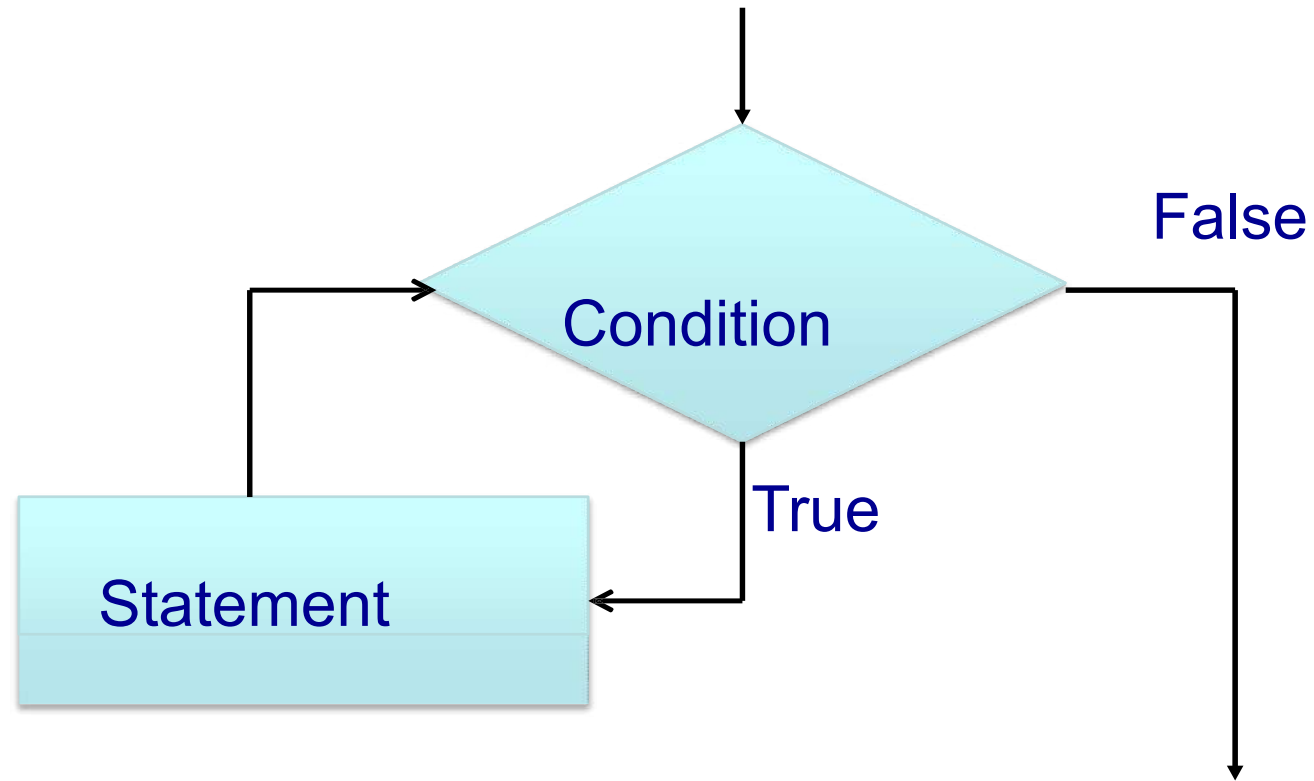
```
        $x++;
```

```
    }
```

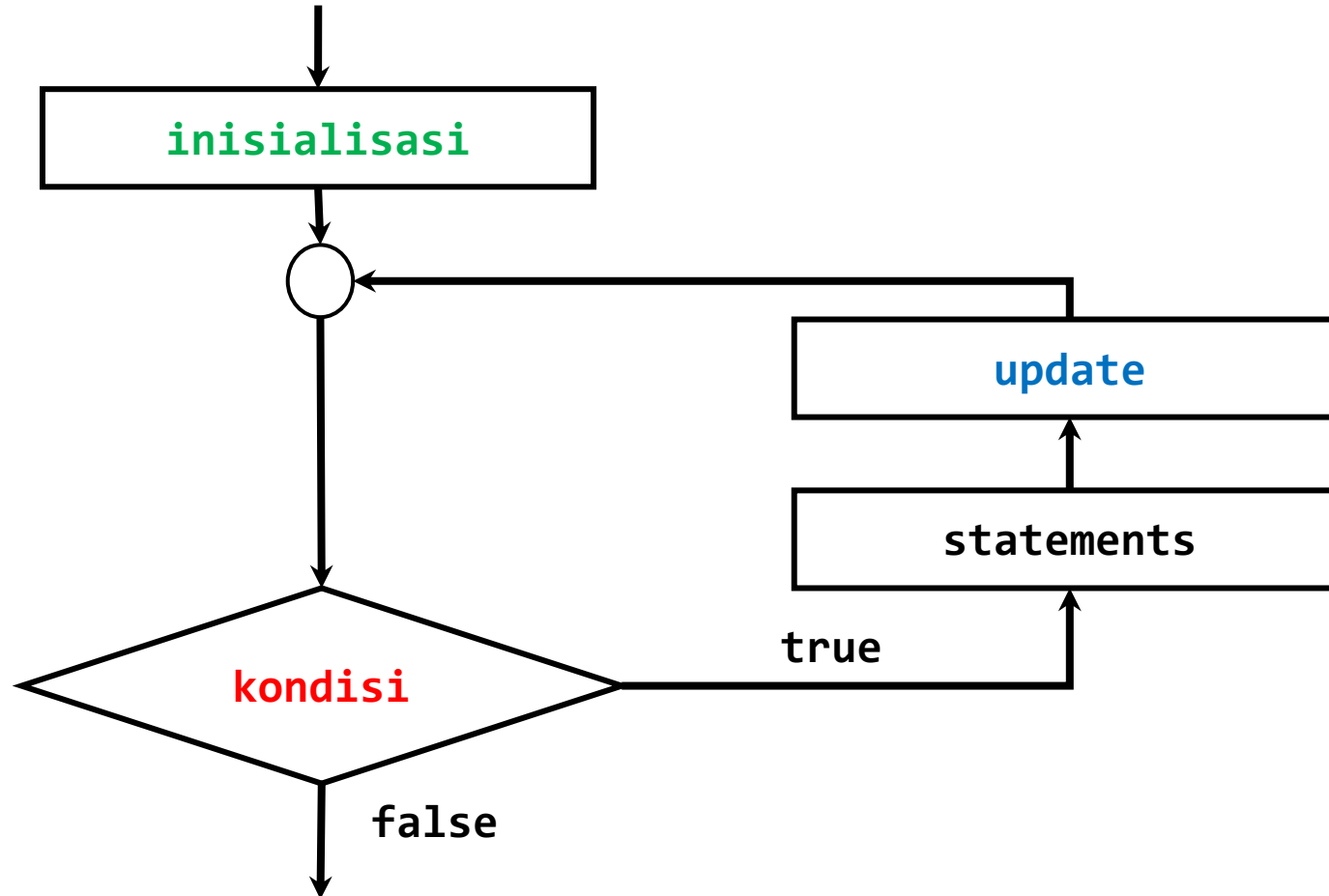
```
?>
```



# Struktur kontrol WHILE



# Flowchart WHILE



# Perbandingan FOR dan WHILE

WHILE      FOR

```
inisialisasi;  
  
while (kondisi) {  
    statement1  
;  
    statement2;  
    ...  
    update  
}
```

setara

```
for (inisialisasi; kondisi; update) {  
    statement1;  
    statement2;  
    ...  
}
```

Contoh:

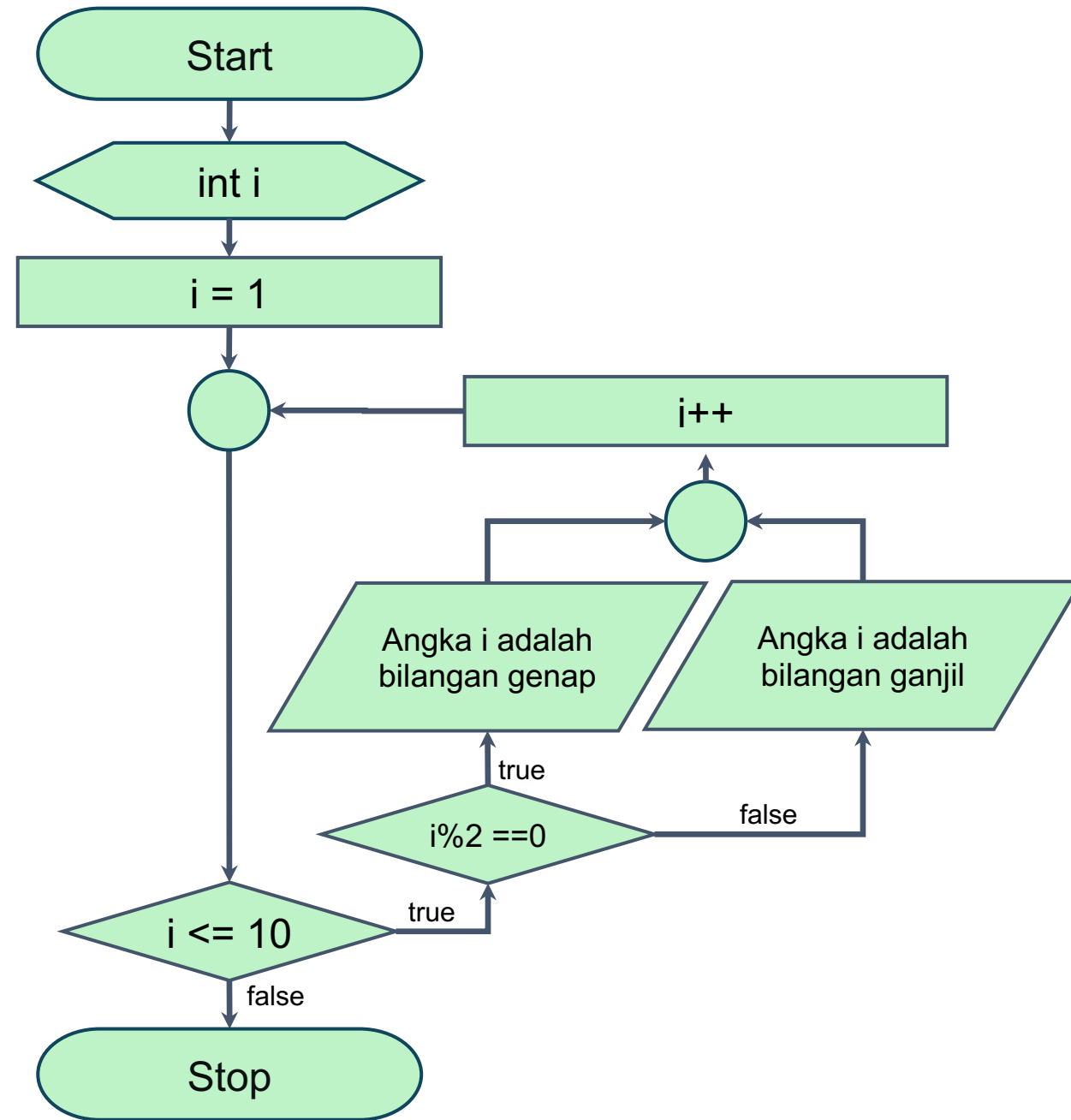
```
$x = 1;  
  
while ($x <= 10) {  
    _____  
    _____  
    $x++;  
}
```

sama

```
for($x = 1 ; $x <= 10 ; $x++){  
    _____  
    _____  
}
```

# Contoh Perulangan WHILE

Buatlah flowchart dan kode program untuk menampilkan keterangan bilangan ganjil dan genap dengan rentang nilai bilangan 1 sampai 10!



# Contoh Perulangan WHILE

## Kode Program

```
<?php
    $x=1;

    while($x <= 10) {
        if($x%2 == 0)
            echo "$x adalah angka genap <br>";
        else
            echo "$x adalah angka ganjil <br>";
        $x++;
    }
?>
```

## Output

```
1 adalah angka ganjil
2 adalah angka genap
3 adalah angka ganjil
4 adalah angka genap
5 adalah angka ganjil
6 adalah angka genap
7 adalah angka ganjil
8 adalah angka genap
9 adalah angka ganjil
10 adalah angka genap
```

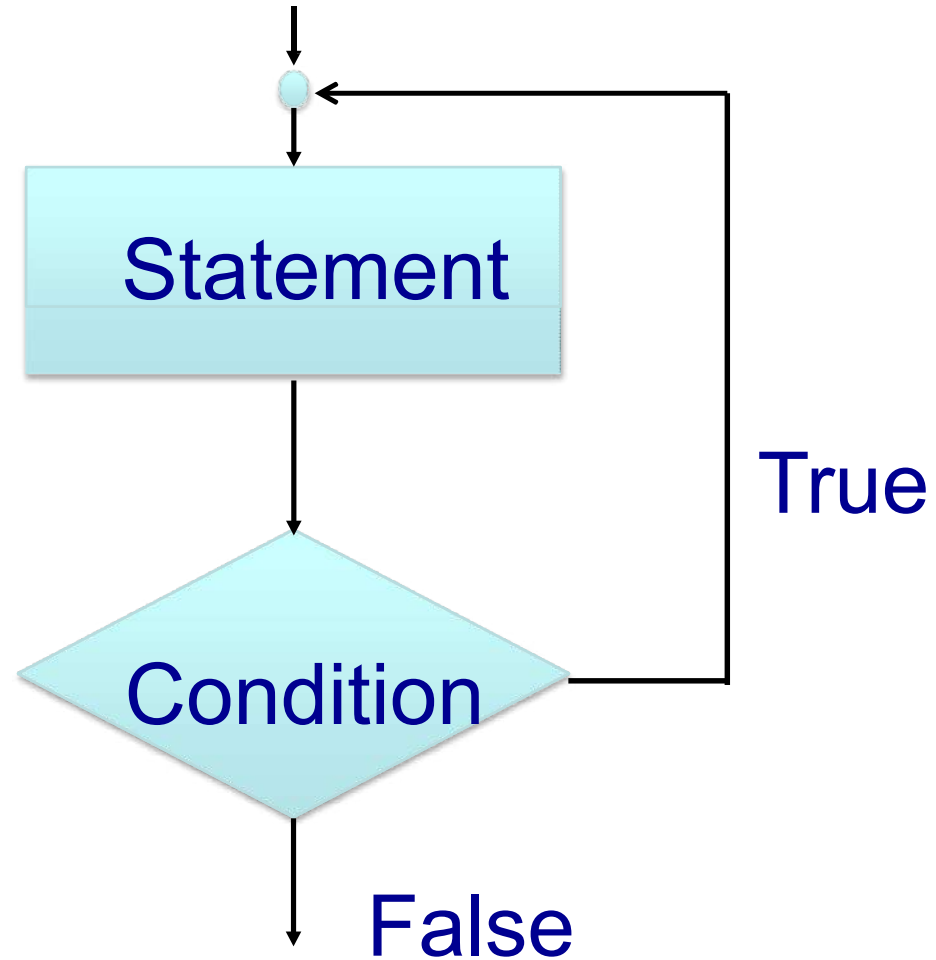
# Struktur Kontrol DO – WHILE

- Pada prinsipnya, perintah DO-WHILE sama dengan perintah WHILE
- Perbedaanya:
  - DO-WHILE mengeksekusi statementnya terlebih dahulu, lalu mengevaluasi kondisi
  - WHILE mengevaluasi kondisi sebelum mengeksekusi statement
  - Oleh karena itu, perintah **DO-WHILE** akan **mengeksekusi block statement minimal 1 kali**, meskipun kondisi tidak terpenuhi
  - Kondisi kebenaran **diperiksa di akhir setiap iterasi**, bukan di awal.

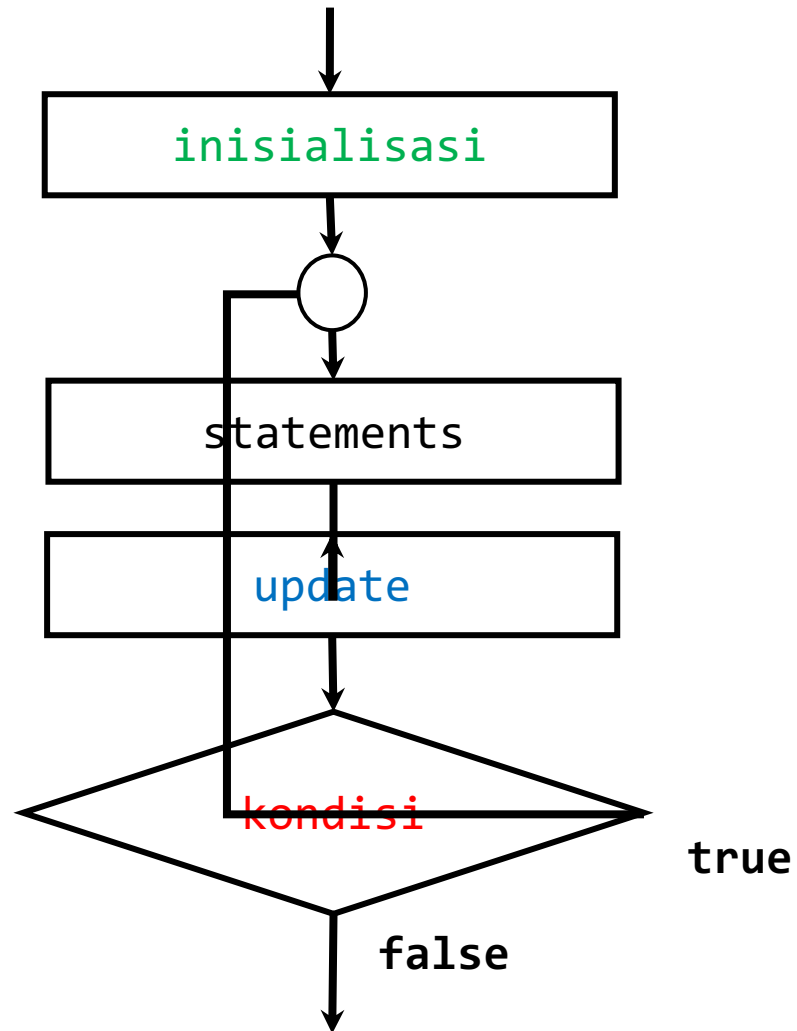
```
<?php
    $x=1;

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

# Struktur kontrol do... while



# Flowchart DO-WHILE





# Struktur kontrol Foreach Loop

- Pernyataan foreach **digunakan untuk loop melalui array**

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

# Perulangan Bersarang (Nested Loop)

- Perulangan bersarang (***nested loop***) adalah
  - struktur perulangan yang berada di dalam perulangan lainnya, ***atau***
  - suatu perulangan yang memiliki perulangan lagi di dalamnya.
- *Loop* terluar dikenal dengan istilah ***outer loop***, sedangkan *loop* yang ada di dalamnya disebut ***inner loop***.
- *Nested loop* bisa lebih dari 2 tingkat/level (*minimal 2 tingkat/level*)

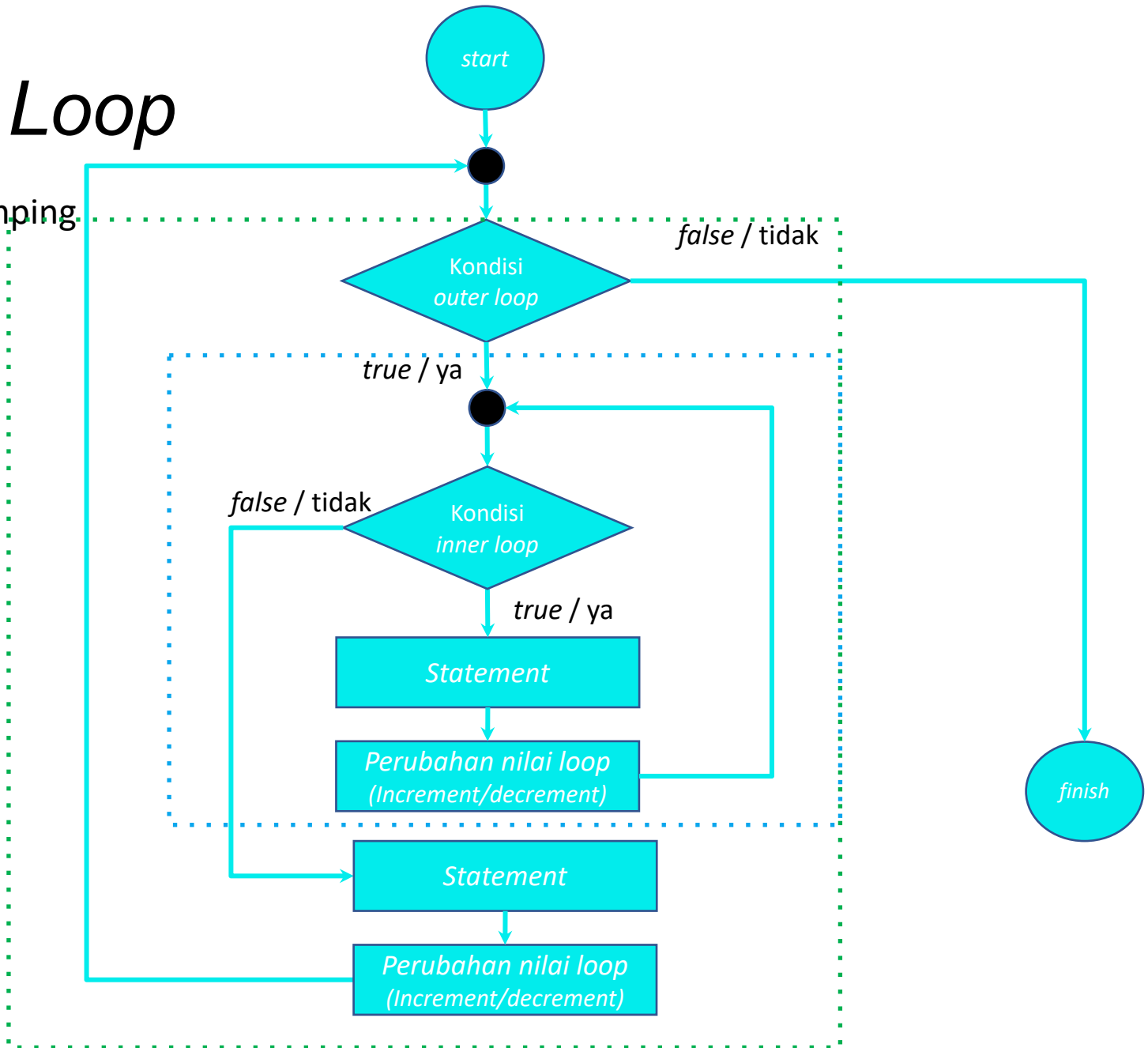
# *Pseudocode Nested Loop*

- *Nested loop* bisa memiliki lebih dari 2 tingkat.
- Secara umum gambaran *nested loop* seperti berikut:

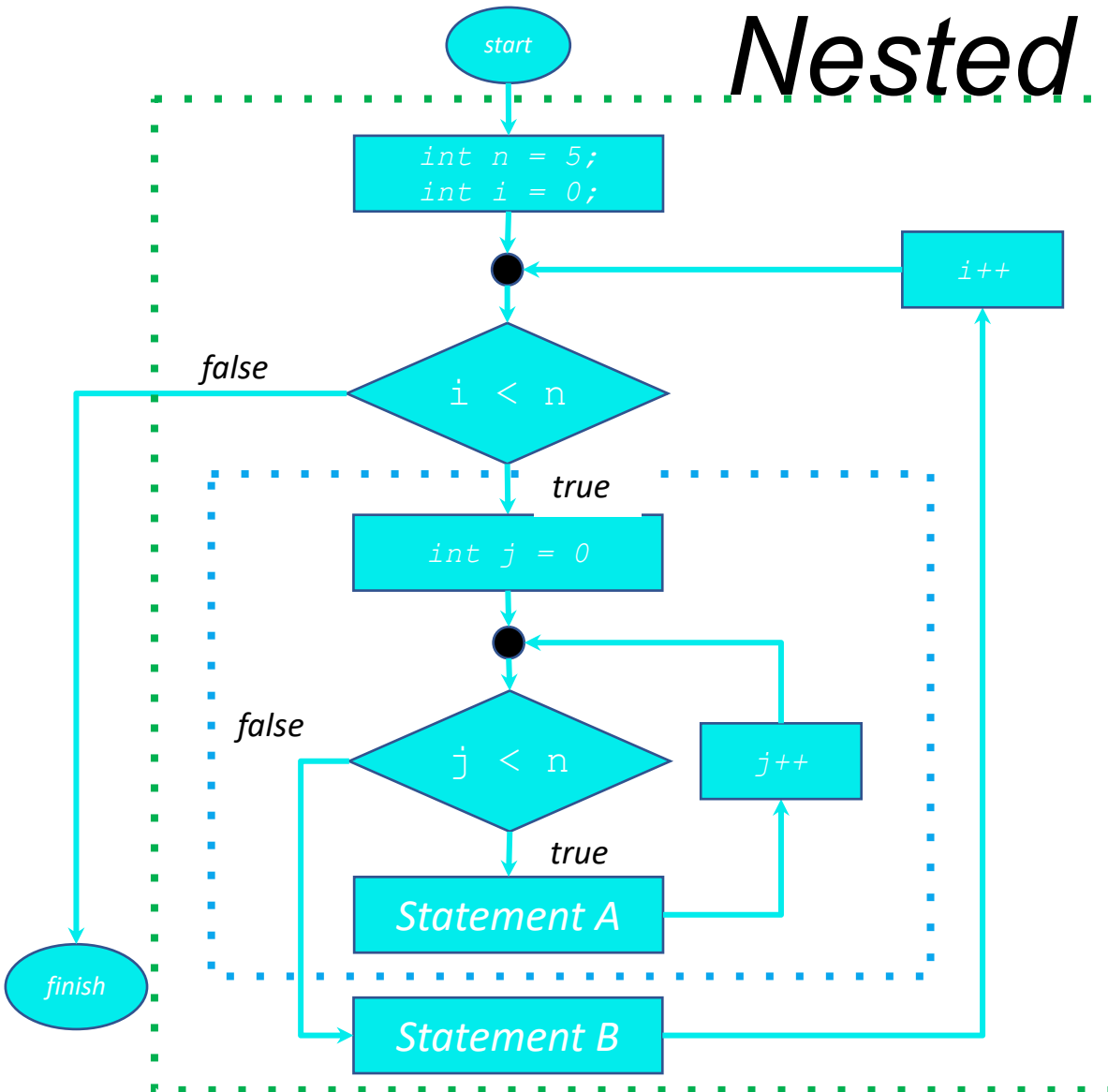
```
1  [ ] loop-level-1 {  
2  [ ]     loop-level-2 {  
3  [ ]         .....  
4  [ ]         loop-level-n {  
5  [ ]             // statement  
6  [ ]         }  
7  [ ]     .....  
8  [ ] }  
9  [ ] }
```

# Flowchart Nested Loop

secara umum *flowchart* untuk *nested loop* seperti pada gambar di samping



# Nested Loop : FOR



```
$n = 5;  
for ($i = 0; $i < $n; $i++) { //Loop Level 1  
    for ($j = 0; $j < $n; $j++) { //Loop Level 2  
        //STATEMENT A  
    }  
    //STATEMENT B  
}
```

Outer  
loop

Inner  
loop

Outer loop

Inner loop

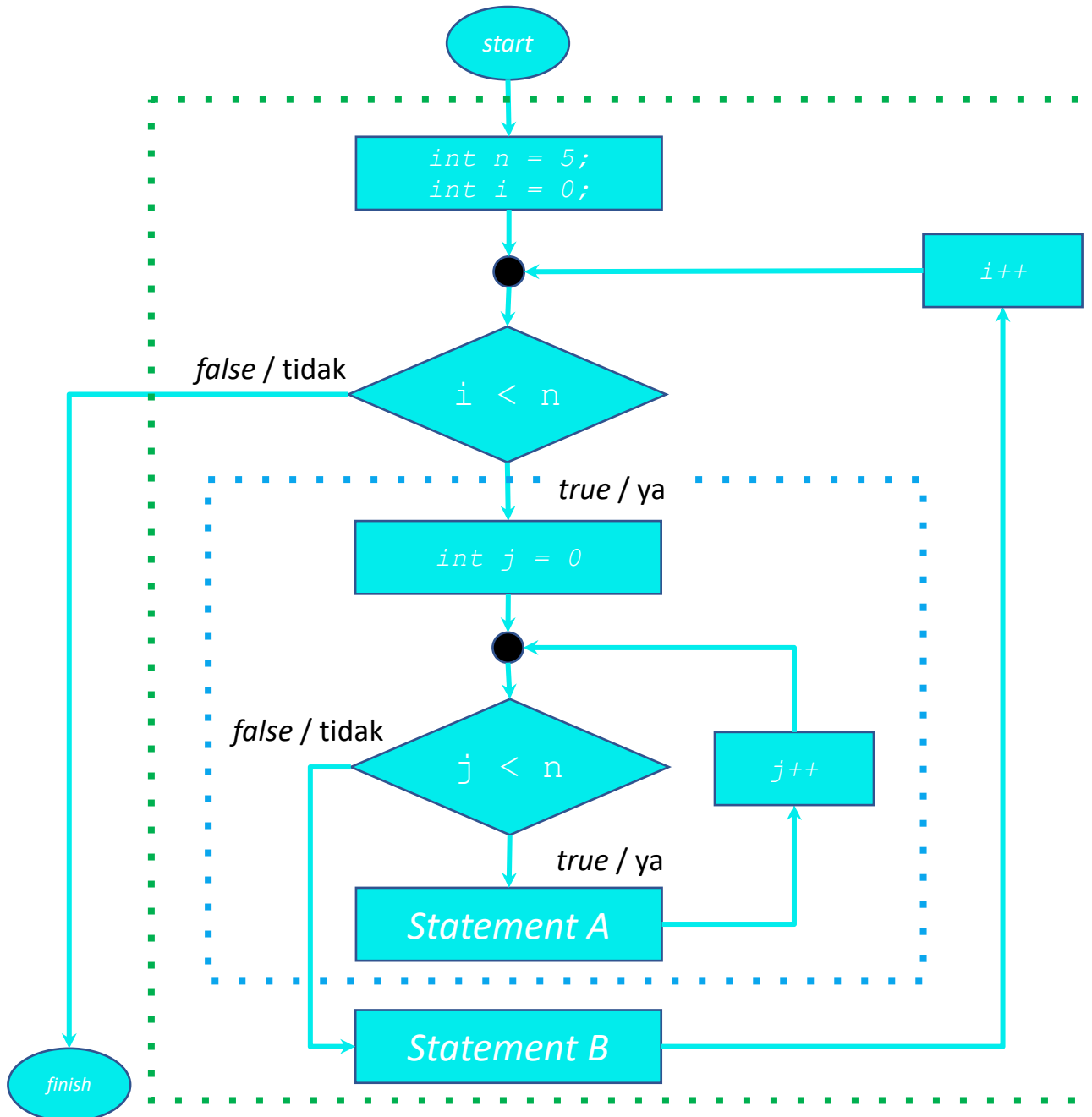
# *Nested Loop : FOR (lebih dari 2 level)*

*Outer loop*

```
for($i=0;$i<$n;$i++){ //Loop Level 1
    for($j=0;$j<$n;$j++){ //Loop Level 2
        for($k=0;$k<$n;$k++){ //Loop Level 3
            //STATEMENT
        }
    }
    for($l=0;$l<$n;$l++){ //Loop Level 2
        //STATEMENT
    }
}
```

*Inner loop*

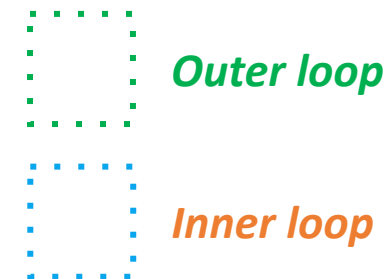
# Nested Loop : While



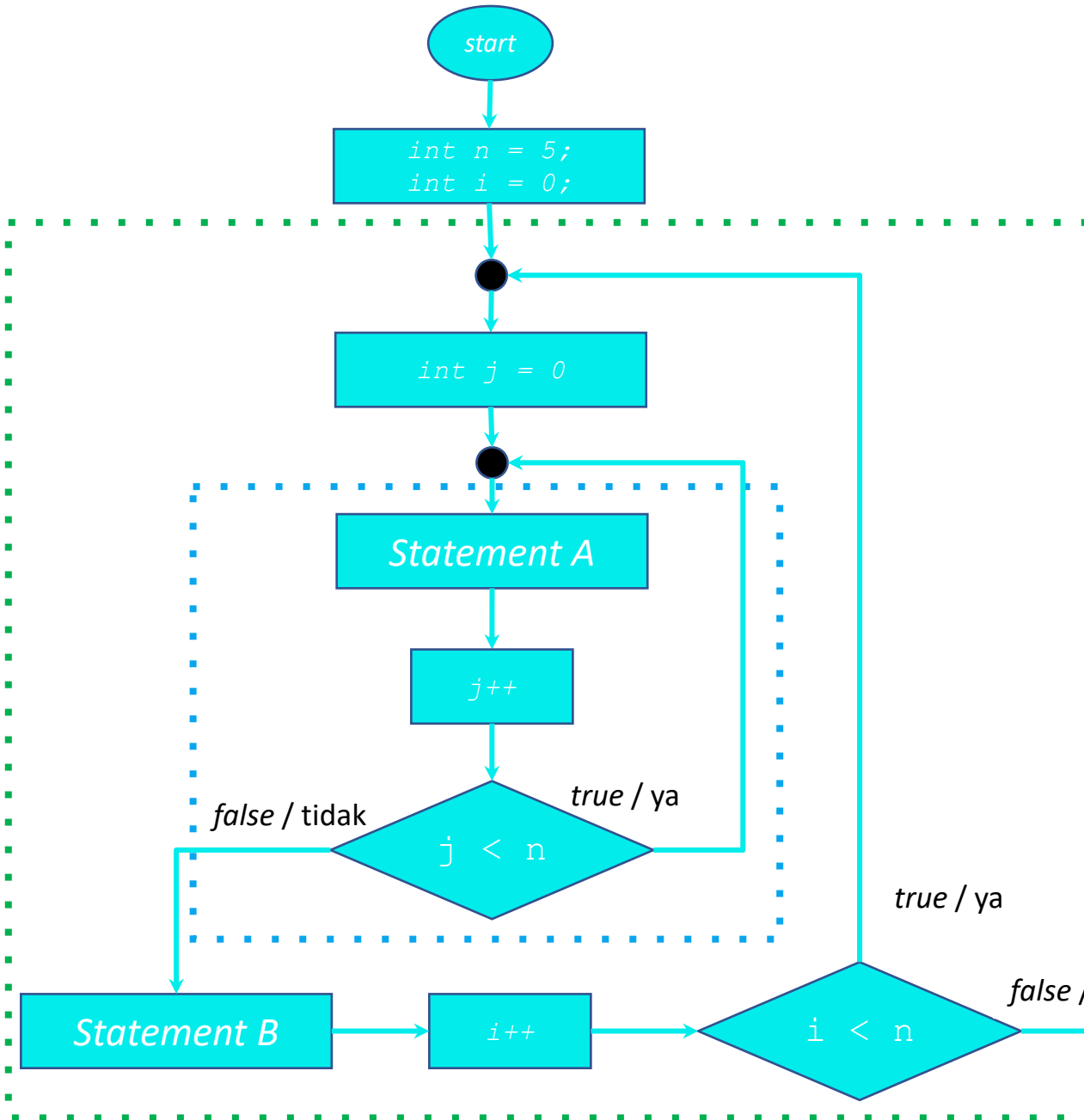
```
$n = 5;  
$i = 0;  
while($i < $n){ //Loop Level 1  
    $j = 0;  
    while($j < $n){ //Loop Level 1  
        //Statement A  
        $j++;  
    }  
    //Statement B  
    $i++;  
}
```

Outer loop

Inner loop



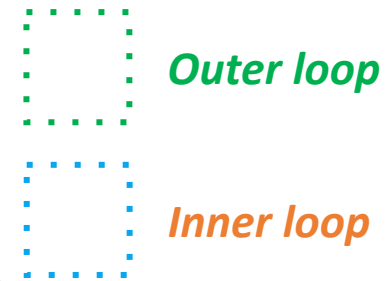
# Nested Loop : Do-While



```
$n = 5;
$i = 0;
do{ //Loop Level 1
    $j = 0;
    do{ //Loop Level 1
        //Statement A
        $j++;
    }while($j < $n);
    //Statement B
    $i++;
}while($i < $n);
```

Outer loop

Inner loop

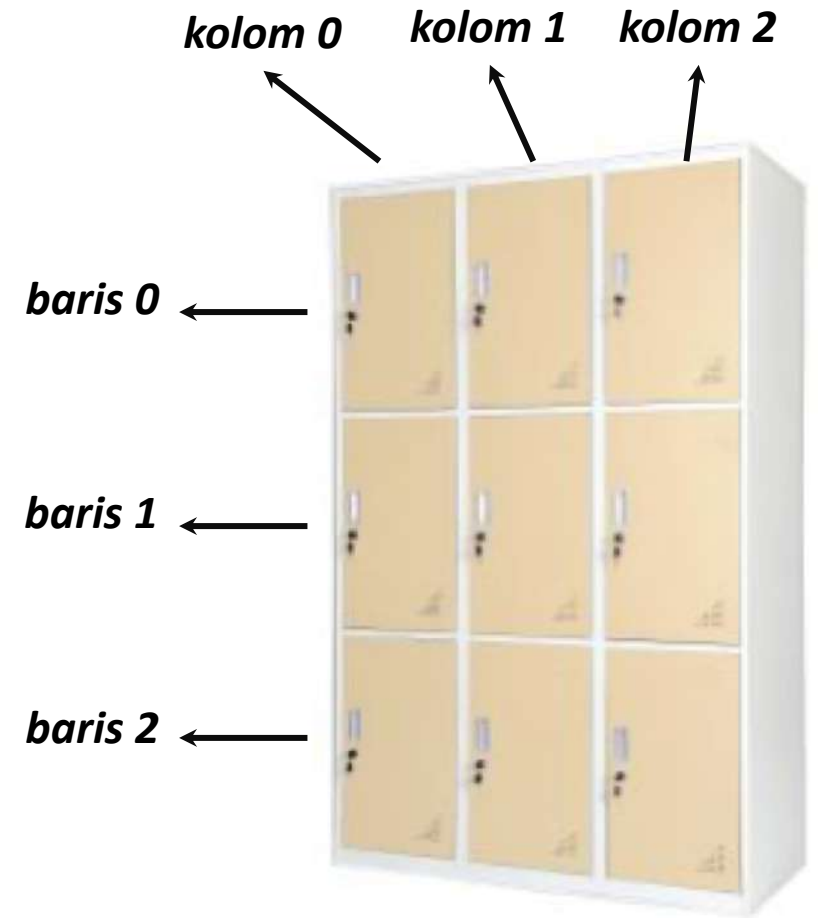




# Logika Rak/Loker

- ***Nested loop*** dengan 2 tingkat/level, ibarat seperti loker.
- Dimana ***outer loop*** kita identifikasi sebagai penunjuk **baris** dan ***inner loop*** kita identifikasi sebagai penunjuk **kolom**.

```
for($baris=0;$baris<3;$baris++){  
    for($kolom=0;$kolom<3;$kolom++){  
        //statement  
    }  
}
```



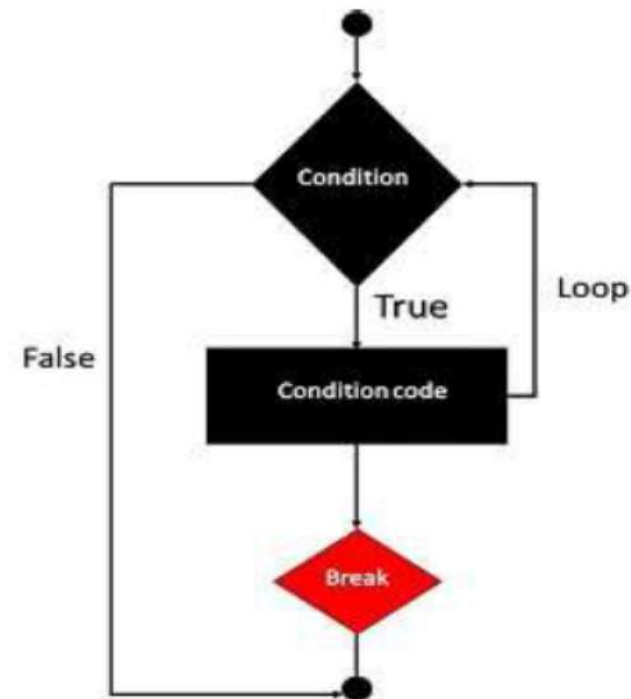
# Logika Rak/Loker (cont.)

```
for($baris=0;$baris<3;$baris++){  
    for($kolom=0;$kolom<3;$kolom++){  
        echo "Baris [$baris] Kolom [$kolom] - ";  
    }  
    echo "<br>";  
}
```



# Break

- Pernyataan `break` pada PHP digunakan untuk mengakhiri eksekusi loop secara dini



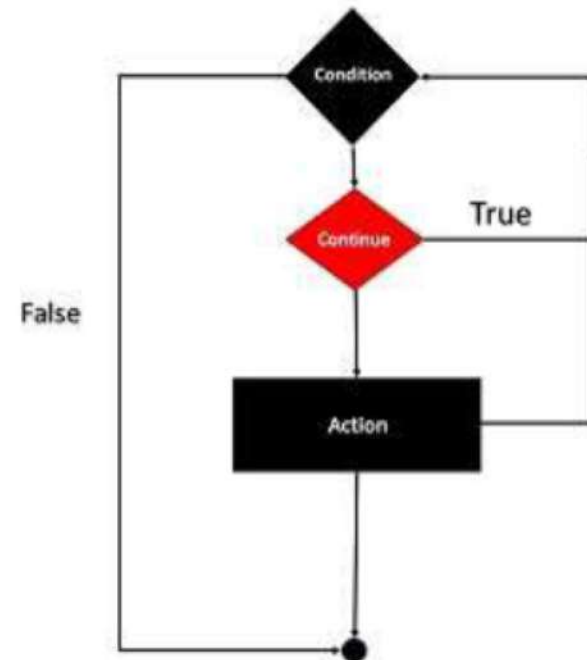
# Contoh Break

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

**Output >>** The number is: 0  
The number is: 1  
The number is: 2  
The number is: 3

# Continue

- Pernyataan continue digunakan untuk menghentikan 1 iterasi yang sedang berjalan dari suatu loop tetapi tidak mengakhiri loop tersebut.



# Contoh Continue

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

## Output >>

The number is: 0  
The number is: 1  
The number is: 2  
The number is: 3  
The number is: 5  
The number is: 6  
The number is: 7  
The number is: 8  
The number is: 9

The background is a deep blue with a complex, abstract pattern. A prominent feature is a glowing, circular ring composed of many small, bright blue particles or dots, which appears to be in motion or vibrating. Surrounding this ring are various other patterns of particles and light trails, some forming larger, more diffuse shapes. The overall effect is one of dynamic energy and digital complexity.

# Array

Insert the Subtitle of Your Presentation

# Pengertian Array

- Array merupakan variable special yang bisa digunakan untuk menyimpan banyak data atau nilai dalam 1 nama variable.
- Karena banyak nilai yang terdapat dalam 1 array, maka tiap nilai akan bisa diidentifikasi menggunakan nomor index masing-masing atau nama key masing-masing
- Tipe Array:
  - Indexed Array
  - Associative Array
  - Multidimensional Array
- Di PHP, array bisa digunakan untuk menyimpan banyak nilai dengan jenis data yang berbeda



# Pengertian Array

- Di PHP, array bisa digunakan untuk menyimpan banyak nilai dengan jenis data yang berbeda, bisa berupa jenis data primitif, objek, function, atau bahkan array yang lain.

```
$myArr = array("Volvo", 15, ["apples", "bananas"], myFunction);
```

- Di PHP terdapat built-in function yang bisa memudahkan mengelola array. Contohnya function `count` di bawah ini

```
$cars = array("Volvo", "BMW", "Toyota");  
echo count($cars);
```

# Tipe Array

## 1. Array Numerik

- Array yang menggunakan indeks numerik.
- Contoh: `$angka = [1, 2, 3, 4];`

## 2. Array Asosiatif

- Array yang menggunakan kunci (**key**) sebagai indeks.
- Contoh: `$mahasiswa = ["nama" => "Budi", "umur" => 20];`

## 3. Array Multidimensi

- Array yang mengandung satu atau lebih array di dalamnya.
- Contoh: `$matrix = [[1, 2], [3, 4], [5, 6]];`

# Deklarasi Array

- Deklarasi array menggunakan fungsi `array()`

```
$cities = array("Malang", "Surabaya", "Jakarta");
```

- Deklarasi array menggunakan tanda `[]`

```
$cities = ["Malang", "Surabaya", "Jakarta"];
```

- Deklarasi associative array

```
$city = [  
    "nama" => "Malang",  
    "luas" => 12345,  
    "penduduk" => 6789  
];
```

# Deklarasi Array

- Membuat Array kosong (cara sama baik indexed maupun associative array)

```
$cities = [];
```

- Dalam satu array, bisa memiliki indeks campuran (baik numerik maupun string)

```
$cities = [];
```

```
$cities[0] = "Malang";
```

```
$cities[1] = "Surabaya";
```

```
$cities["ibukota"] = "Jakarta";
```

# Mengakses Elemen Array

- Dilakukan dengan menggunakan nomor indeks (untuk Indexed atau Numerik array), dan nama key (untuk Associative array)

```
$cities = array("Malang", "Surabaya", "Jakarta");
```

```
echo $cities[0];
```

```
$city = array("nama" => "Malang", "luas" => 12345, "penduduk"  
=> 6789);
```

```
echo $city["nama"];
```

- Ketika mengakses associative array, nama key bisa diapit double quote `echo $city["nama"];` atau single quote `echo $city['nama'];`

# Mengakses Elemen Array

- Mengakses elemen array yang berupa function:

```
function myFunction() {  
    echo "I come from a function!";  
}  
$myArr = array("Volvo", 15, "myFunction");  
$myArr[2]();
```

- Atau dalam associative array:

```
function myFunction() {  
    echo "I come from a function!";  
}  
$myArr = array("car" => "Volvo", "age" => 15, "message" => "myFunction");  
$myArr["message"]();
```

# Mengakses Elemen Array

- Menggunakan looping untuk mengakses elemen Numerik/Indexed array

```
$cities = array("Malang", "Surabaya", "Jakarta");  
foreach ($cities as $x) {  
    echo "$x <br>";  
}
```

- Atau dalam associative array:

```
$city = array("nama" => "Malang", "luas" => 12345, "penduduk" => 6789);  
foreach ($city as $x => $y) {  
    echo "$x: $y <br>";  
}
```

# Mengupdate Elemen Array

- Update elemen array bisa dengan memasukkan nilai baru pada nomor indeks (Numerik/indexed array) atau nama key (Associative array) yang ingin di-update

- Update elemen array pada Indexed array

```
$cities = array("Malang", "Surabaya", "Jakarta");  
$cities[1] = "Lumajang";
```

- Update elemen array pada Associative array

```
$city = array("nama" => "Malang", "luas" => 12345, "penduduk" =>  
6789);  
$city["nama"] = "Lumajang";
```



# Mengupdate Elemen Array

- Jika dibutuhkan, proses update elemen array juga bisa dilakukan melalui perulangan foreach (khususnya apabila nilai baru yang akan dimasukkan bisa dipolakan dalam perulangan)

```
$cities = array("Malang", "Surabaya", "Jakarta");  
foreach ($cities as &$amp;x) {  
    $x = "Lumajang";  
}  
unset($x);  
var_dump($cities);
```

- **Catatan:** Perlu ditambahkan fungsi `unset` setelah looping `foreach`, jika tidak maka variable `$x` akan tetap menjadi reference untuk elemen array terakhir. Sehingga kalau ada perubahan pada `$x` akan ikut mengubah elemen terakhir array.

# Menambahkan Elemen Array

- Untuk menambahkan elemen array bisa menggunakan tanda []

```
$fruits = array("Apple", "Banana", "Cherry");
```

```
$fruits[] = "Orange";
```

- Menambahkan elemen array untuk array associative

```
$city = array("nama" => "Malang", "luas" => 12345);
```

```
$city["jenis"] = "Kotamadya";
```

- Menambahkan multiple elemen, bisa digunakan fungsi array\_push

```
$fruits = array("Apple", "Banana", "Cherry");
```

```
array_push($fruits, "Orange", "Kiwi", "Lemon");
```

# Menambahkan Elemen Array

- Menambahkan multiple elemen pada array associative menggunakan operator +=

```
$city = array("nama" => "Malang", "luas" => 12345);
```

```
$city += ["jenis" => "Kotamadya", "jum_penduduk" => 6789];
```

# Menghapus Elemen Array

- Menggunakan fungsi `array_splice()`, kita bisa menghapus mulai **dari mana** dan **berapa data** yang mau dihapus. **Setelah dihapus**, maka **indeks array akan disesuaikan (disusun ulang)** secara otomatis, mulai dari indeks 0.

```
$cities = array("Malang", "Surabaya", "Jakarta");  
array_splice($cities, 1, 1)
```

- Menggunakan fungsi `unset()`, setelah penghapusan maka indeks array yang dihapus **akan hilang** dan **tidak disesuaikan ulang**

```
$cities = array("Malang", "Surabaya", "Jakarta");  
unset($cities[1]);
```

# Menghapus Multiple Elemen Array

- `Array_splice()` bisa digunakan juga untuk menghapus banyak elemen array.

```
$cities = array("Malang", "Surabaya", "Jakarta");  
array_splice($cities, 1, 2)
```

- Menghapus banyak elemen array menggunakan `unset`

```
$cities = array("Malang", "Surabaya", "Jakarta");  
unset($cities[0], $cities[1]);
```

# Menghapus Elemen Associative Array

- Menggunakan unset.

```
$city = array("nama" => "Malang", "luas" => 1234, "jenis" => "Kotamadya");  
unset($city["jenis"]);
```

- Menggunakan fungsi array\_diff. Array\_diff menghasilkan array baru

```
$city = array("nama" => "Malang", "luas" => 1234, "jenis" => "Kotamadya");  
$newarray = array_diff($city, ["Kotamadya", 1234]);
```

# Menghapus Elemen Pertama dan Terakhir Array

- Menghapus elemen terakhir dari array.

```
$cities = array("Malang", "Surabaya", "Jakarta");  
array_pop($cities);
```

- Menghapus elemen pertama array.

```
$cities = array("Malang", "Surabaya", "Jakarta");  
array_shift($cities);
```

# Mengurutkan Array

- Beberapa fungsi untuk pengurutan array:
  - `sort()`: pengurutan elemen array secara ascending
  - `rsort()`: pengurutan elemen array secara descending
  - `asort()`: pengurutan associative array ascending, berdasar value
  - `ksort()`: pengurutan associative array ascending, berdasar key
  - `arsort()`: pengurutan associative array descending, berdasar value
  - `krsort()`: pengurutan associative array descending, berdasar key



# Mengurutkan Array

- Contoh:

```
$numbers = array(4, 6, 2, 22, 11);
```

```
sort($numbers); //ascending
```

```
rsort($numbers); //descending
```

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

```
asort($age); //ascending according to value
```

```
ksort($age); //ascending according to key
```

```
arsort($age); //descending according to value
```

```
krsort($age); //descending according to key
```

# Array Multidimensi

- Array multidimensi merupakan array yang di dalamnya terdapat 1 atau lebih array
- PHP mendukung array 2, 3, 4 atau lebih dimensi
- Tetapi biasanya yang umum dikelola maksimal adalah 3 dimensi. Di atas 3 dimensi akan sulit untuk dikelola untuk kebanyakan orang

# Array 2 Dimensi

Merek	Penjualan Januari 2010	Penjualan Februari 2010
Toyota	20803	21800
Daihatsu	8302	7518
Honda	3755	4431
Mitsubishi	6855	8246

- Deklarasi array

```
$penjualan = array (  
    array("Toyota", 20803, 21800),  
    array("Daihatsu", 8302, 7518),  
    array("Honda", 3755, 4431),  
    array("Mitsubishi", 6855, 8246)  
);
```

# Array 2 Dimensi

- Deklarasi array

```
$penjualan = array (  
    array("Toyota", 20803, 21800),  
    array("Daihatsu", 8302, 7518),  
    array("Honda", 3755, 4431),  
    array("Mitsubishi", 6855, 8246)  
);
```

- Dari deklarasi di atas, terlihat array 2 dimensi \$penjualan memiliki 4 array dan masing-masing merepresentasikan baris dan kolom

# Array 2 Dimensi

- Untuk mengakses array 2 dimensi, perlu menggunakan **2 indeks**, yang masing-masing angka indeks merepresentasikan indeks **baris** dan indeks **kolom**
- Misalkan: `$penjualan[0][0]` untuk mengakses elemen array dengan indeks baris 0 dan indeks kolom 0

```
echo $penjualan[0][0].": Jan: ".$penjualan[0][1].", Feb: ".$penjualan[0][2].".<br>;
```

```
echo $penjualan[1][0].": Jan: ".$penjualan[1][1].", Feb: ".$penjualan[1][2].".<br>;
```

```
echo $penjualan[2][0].": Jan: ".$penjualan[2][1].", Feb: ".$penjualan[2][2].".<br>;
```

```
echo $penjualan[3][0].": Jan: ".$penjualan[3][1].", Feb: ".$penjualan[3][2].".<br>;
```

```
Toyota: Jan: 20803, Feb: 21800.  
Daihatsu: Jan: 8302, Feb: 7518.  
Honda: Jan: 3755, Feb: 4431.  
Mitsubishi: Jan: 6855, Feb: 8246.
```

# Array 2 Dimensi

- Mengakses menggunakan perulangan for

```
for ($row = 0; $row < 4; $row++) {  
    echo "<p><b>Baris ke-$row</b></p>";  
    echo "<ul>";  
    for ($col = 0; $col < 3; $col++) {  
        echo "<li>".$penjualan[$row][$col]."</li>";  
    }  
    echo "</ul>";  
}
```

**Baris ke-0**

- Toyota
- 20803
- 21800

**Baris ke-1**

- Daihatsu
- 8302
- 7518

**Baris ke-2**

- Honda
- 3755
- 4431

**Baris ke-3**

- Mitsubishi
- 6855
- 8246

# Fungsi di Array

- `array_push()`: Menambahkan satu atau lebih elemen ke akhir array.

```
$buah = ["Apel", "Jeruk"];
```

```
array_push($buah, "Mangga", "Durian");
```

```
print_r($buah);
```

```
// Output: Array ( [0] => Apel [1] => Jeruk [2] => Mangga [3] => Durian )
```

- `array_pop()`: Menghapus elemen terakhir dari array

```
$buah = ["Apel", "Jeruk", "Mangga"];
```

```
array_pop($buah);
```

```
print_r($buah);
```

```
// Output: Array ( [0] => Apel [1] => Jeruk )
```

# Fungsi di Array

- `array_shift()`: Menghapus elemen pertama dari array..

```
$buah = ["Apel", "Jeruk", "Mangga"];
```

```
array_shift($buah);
```

```
print_r($buah);
```

```
// Output: Array ( [0] => Jeruk [1] => Mangga )
```

- `array_unshift()`: Menambahkan satu atau lebih elemen ke awal array

```
$buah = ["Jeruk", "Mangga"];
```

```
array_unshift($buah, "Apel");
```

```
print_r($buah);
```

```
// Output: Array ( [0] => Apel [1] => Jeruk [2] => Mangga )
```



# Fungsi di Array

- `array_merge()`: Menggabungkan dua atau lebih array menjadi satu

```
$array1 = ["Apel", "Jeruk"];
```

```
$array2 = ["Mangga", "Durian"];
```

```
$result = array_merge($array1, $array2);
```

```
print_r($result);
```

```
// Output: Array ( [0] => Apel [1] => Jeruk [2] => Mangga [3] => Durian )
```

- `array_keys()`: Mengembalikan semua kunci dari array

```
$harga = ["Apel" => 3000, "Jeruk" => 2000, "Mangga" => 5000];
```

```
$keys = array_keys($harga);
```

```
print_r($keys);
```

```
// Output: Array ( [0] => Apel [1] => Jeruk [2] => Mangga )
```

# Fungsi di Array

- `array_values()`: Mengembalikan semua nilai dari array.

```
$harga = ["Apel" => 3000, "Jeruk" => 2000, "Mangga" => 5000];  
$values = array_values($harga);  
print_r($values);
```

```
// Output: Array ( [0] => 3000 [1] => 2000 [2] => 5000 )
```

- `in_array()`: Memeriksa apakah suatu nilai ada di dalam array

```
$buah = ["Apel", "Jeruk", "Mangga"];  
if (in_array("Jeruk", $buah)) {  
    echo "Jeruk ditemukan!";  
}
```

```
// Output: Jeruk ditemukan!
```

# Fungsi di Array

- `array_search()`: Mencari nilai dalam array dan mengembalikan kunci atau indeks dari nilai tersebut.

```
$buah = ["Apel", "Jeruk", "Mangga"];
```

```
$key = array_search("Jeruk", $buah);
```

```
echo $key;
```

```
// Output: 1
```

- `array_reverse()`: Membalik urutan elemen dalam array

```
$buah = ["Apel", "Jeruk", "Mangga"];
```

```
$reversed = array_reverse($buah);
```

```
print_r($reversed);
```

```
// Output: Array ( [0] => Mangga [1] => Jeruk [2] => Apel )
```

# Fungsi di Array

- `array_slice()`: Mengambil bagian dari array.

```
$buah = ["Apel", "Jeruk", "Mangga", "Durian"];  
$slice = array_slice($buah, 1, 2);  
print_r($slice);  
// Output: Array ( [0] => Jeruk [1] => Mangga )
```

- `array_splice()`: Menghapus bagian dari array dan menggantinya dengan elemen baru

```
$buah = ["Apel", "Jeruk", "Mangga", "Durian"];  
array_splice($buah, 1, 2, ["Pisang", "Anggur"]);  
print_r($buah);  
// Output: Array ( [0] => Apel [1] => Pisang [2] => Anggur [3] => Durian )
```

# Fungsi di Array

- `array_unique()`: Menghapus nilai duplikat dari array.

```
$buah = ["Apel", "Jeruk", "Mangga", "Apel"];
```

```
$unique = array_unique($buah);
```

```
print_r($unique);
```

```
// Output: Array ( [0] => Apel [1] => Jeruk [2] => Mangga )
```

- `count()`: Menghitung jumlah elemen dalam array

```
$buah = ["Apel", "Jeruk", "Mangga"];
```

```
$jumlah = count($buah);
```

```
echo $jumlah;
```

```
// Output: 3
```

# Fungsi di Array

- `sort()`: Mengurutkan elemen array secara ascending

```
$buah = ["Jeruk", "Apel", "Mangga"];
```

```
sort($buah);
```

```
print_r($buah);
```

```
// Output: Array ( [0] => Apel [1] => Jeruk [2] => Mangga
```



# Class Object & Function

Insert the Subtitle of Your Presentation

# Pengenalan OOP

- OOP (Object Oriented Programming) merupakan pendekatan atau paradigma untuk pengembangan perangkat lunak berbasis **objek**
- **Objek** menjadi **fokus utama** dalam merancang perangkat lunak
- Pendekatan yang relatif baru dibandingkan dengan pendekatan Structural Programming
- Jika **Structural/Procedural Programming** menulis program berorientasi pada **prosedur/fungsi** yang melakukan operasi terhadap data, **OOP** lebih menitikberatkan pada pembuatan **objek** yang di dalamnya terdiri dari **data** dan **function** dalam menyusun suatu program.



# Objek Oriented vs Struktural

- Struktural
  - Program dipecah kedalam fungsi
  - Perubahan fitur □ kemungkinan mengganggu keseluruhan program
- Object Oriented
  - Program dipecah kedalam object
  - Didalamnya terdapat state/property/data dan behavior/fungsi
  - Perubahan fitur □ tidak mengganggu keseluruhan program

# Objek Oriented vs Struktural

- Contoh:
- Kita akan membuat program game simulasi sepeda, didalamnya ada karakter sepeda yang memiliki kecepatan, gear dan merk.
- Bagaimana membangun game tersebut dengan metode **konvensional**?
  - Langkah pertama kita buat variabelnya, misal kecepatan, gear, merk
  - Langkah berikutnya kita buat fungsi-fungsinya, tambah kecepatan, kurangi kecepatan.
  - Langkah berikutnya kita coba mengoperasikan sepeda tersebut secara sederhana, yaitu memanipulasi kecepatan, gear, merk nya, didalam fungsi main, kemudian kita cetak ke layer.
- Kode program □...

```
public class SepedaStruktural
{
    public static void main(String[] args)
    {
        String merek;
        int kecepatan, gear;

        merek = "Poligone";
        kecepatan = 10;
        gear = 1;

        kecepatan = tambahKecepatan(kecepatan, 10);

        System.out.println("Merek: " + merek);
        System.out.println("Kecepatan: " + kecepatan);
    }

    public static int tambahKecepatan(int kecepatan, int increment)
    {
        kecepatan += increment;

        return kecepatan;
    }

    public static int kurangiKecepatan(int kecepatan, int decrement)
    {
        kecepatan -= decrement;

        return kecepatan;
    }
}
```

# Objek Oriented vs Struktural

- Bagaimana jika ada dua sepeda di game?
  - Tambahkan variabel merek2, kecepatan2, gear2
  - Coba manipulasi nilai-nilai variabelnya kemudian tampilkan ke layar
- Kode program ☐ ...

```
public class SepedaStruktural
{
    public static void main(String[] args)
    {
        String merek, merek2;
        int kecepatan, kecepatan2, gear, gear2;

        merek = "Poligone";
        kecepatan = 10;
        gear = 1;

        merek2 = "Wiim Cycle";
        kecepatan2 = 15;
        gear2 = 3;

        kecepatan = tambahKecepatan(kecepatan, 10);
        kecepatan2 = tambahKecepatan(kecepatan2, 5);

        System.out.println("Merek: " + merek);
        System.out.println("Kecepatan: " + kecepatan);

        System.out.println("Merek: " + merek2);
        System.out.println("Kecepatan: " + kecepatan2);
    }

    public static int tambahKecepatan(int kecepatan, int increment)
    {
        kecepatan += increment;

        return kecepatan;
    }

    public static int kurangiKecepatan(int kecepatan, int decrement)
    {
        kecepatan -= decrement;

        return kecepatan;
    }
}
```

# Objek Oriented vs Struktural

- Bagaimana jika ada **sepuluh** sepeda?
  - Tambahkan variabel merek3, kecepatan3, gear3 .....  
merek9, kecepatan9, gear9
  - Cukup melelahkan...
- Bagaimana dengan object oriented?
  - Buat sebuah class Sepeda yang memiliki atribut merek, kecepatan, gear.
  - Buat 10 object sepeda
- Kode program □ ...

```

public class Sepeda
{
    private String merek;
    private int kecepatan;
    private int gear;

    public Sepeda(String newMerek, int newKecepatan, int newGear)
    {
        merek = newMerek;
        kecepatan = newKecepatan;
        gear = newGear;
    }

    public void tambahKecepatan(int increment)
    {
        kecepatan += increment;
    }

    public void kurangiKecepatan(int decrement)
    {
        kecepatan -= decrement;
    }

    public void cetakStatus()
    {
        System.out.println("Merek: " + merek);
        System.out.println("Kecepatan: " + kecepatan);
        System.out.println("Gear: " + gear);
    }
}

```

```

public class SepedaOOP
{
    public static void main(String[] args)
    {
        Sepeda spd1 = new Sepeda("Poligone", 10, 1);
        Sepeda spd2 = new Sepeda("Wiim Cycle", 15, 3);

        spd1.tambahKecepatan(10);
        spd2.tambahKecepatan(5);

        spd1.cetakStatus();
        spd2.cetakStatus();
    }
}

```

# Objek Oriented vs Struktural

- Dapat kita lihat bahwa dengan OOP, untuk membuat banyak sepeda, kita tidak perlu tuliskan berulang-ulang variabel merek, kecepatan, dan gear.
- Kita cukup buat banyak objek sepeda saja, dari sebuah class sepeda yang sudah kita buat.



# Kelebihan OOP

- Pemrograman Berorientasi Objek (OOP) lebih cepat dan mudah dieksekusi
- OOP menyediakan struktur yang jelas untuk program.
- OOP membantu menjaga kode PHP tetap DRY (Don't Repeat Yourself) dan membuat kode lebih mudah untuk dipelihara, diubah, dan debug.
- OOP memungkinkan pembuatan aplikasi yang sepenuhnya dapat digunakan kembali dengan lebih sedikit kode dan waktu pengembangan yang lebih singkat.

**Tips:** Prinsip "Don't Repeat Yourself" (DRY) adalah tentang mengurangi pengulangan kode. Kode-kode yang umum untuk aplikasi, akan ditempatkan tersendiri di satu tempat dan menggunakannya kembali alih-alih mengulangi untuk menulisnya.

# Konsep OOP

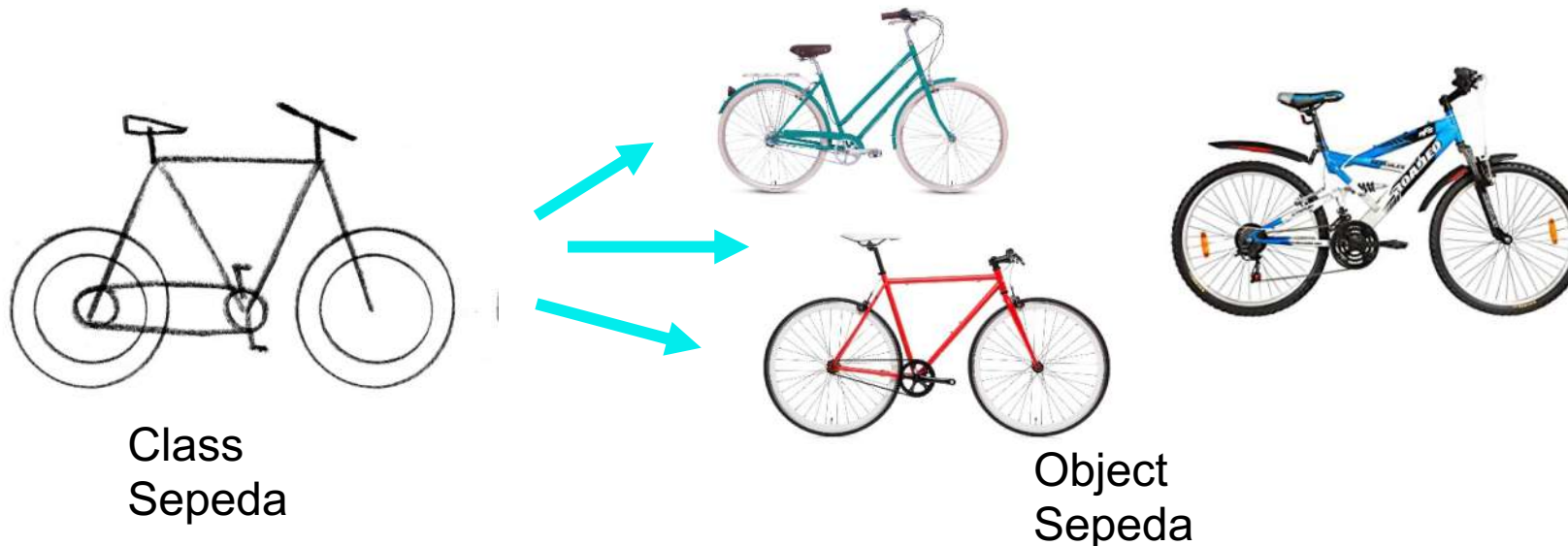
- Beberapa aspek dalam OOP:
  - Object
  - Class
  - Enkapsulasi
  - Inheritance

# Object

- Object adalah suatu rangkaian dalam program yang terdiri dari **state** dan **behaviour**.
- Object pada software dimodelkan sedemikian rupa sehingga mirip dengan objek yang ada di dunia nyata.
- Objek memiliki state dan behaviour.
- State adalah ciri-ciri atau atribut dari objek tersebut.
  - Misal objek Sepeda, memiliki state **merek, kecepatan, gear** dan sebagainya.
- Behaviour adalah perilaku yang dapat dilakukan objek tersebut.
  - Misal pada Sepeda, behaviournya antara lain, **tambah kecepatan, pindah gear, kurangi kecepatan, belok**, dan sebagainya.

# Class

- Class adalah blueprint atau prototype dari objek.
- Ambil contoh objek sepeda.
  - Terdapat berbagai macam sepeda di dunia, dari berbagai merk dan model.
  - Namun semua sepeda dibangun berdasarkan blueprint yang sama, sehingga tiap sepeda memiliki komponen dan karakteristik yang sama.
- Sepeda yang anda miliki dirumah, adalah hasil **instansiasi** dari **class** sepeda.

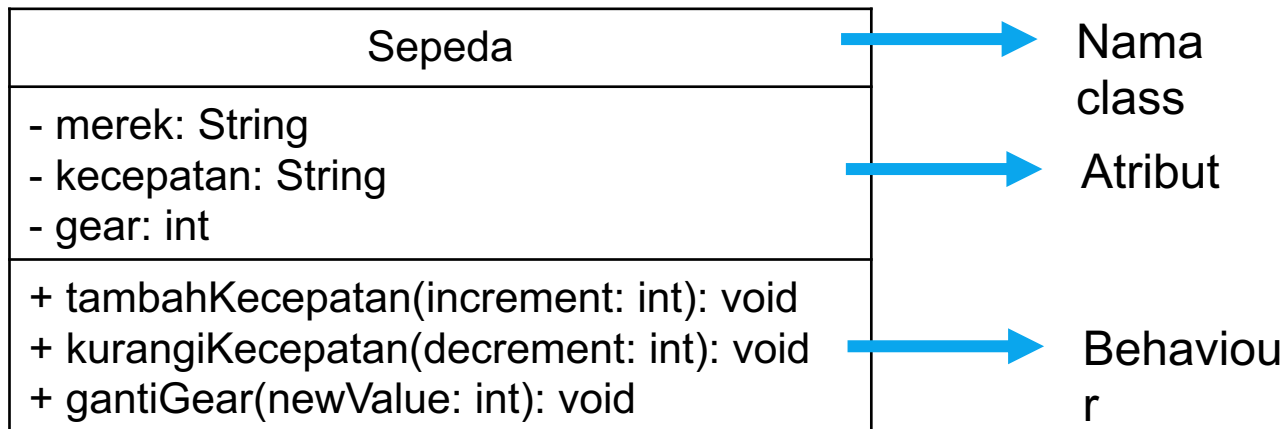


# Inheritance

- Memungkinkan kita untuk mengorganisir struktur program dengan natural.
- Memperluas fungsionalitas program tanpa harus mengubah banyak bagian program.
- Contoh di dunia nyata:
  - Objek sepeda dapat diturunkan lagi ke model yang lebih luas, misal sepeda gunung (mountain bike) dan city bike.
  - Masing-masing dapat memiliki komponen/fitur tambahan, misal sepeda gunung memiliki **suspensi**, yang tidak dimiliki sepeda biasa. Dan city bike memiliki keranjang di bagian depannya.
  - Dalam hal ini, objek mountain bike dan road bike **mewarisi** objek sepeda.

# UML Class Diagram

- Dalam pemrograman berorientasi objek, rancangan class digambarkan dengan UML Class Diagram
  - UML adalah singkatan dari Unified Modelling Language
- Misal class Sepeda, yang memiliki state **merek**, **kecepatan**, **gear** dan behavior **tambahKecepatan**, **kurangiKecepatan**, **gantiGear** digambarkan dengan class diagram sebagai berikut:



# Class dan Object

- Class dan Object merupakan 2 aspek utama dalam OOP
- **Class** merupakan **rancangan/template/blueprint** untuk membuat objek, sedangkan **objek** adalah **bentuk nyatanya (instance)** yang dibuat dari suatu class
- Ketika suatu objek dibuat dari suatu class, maka objek tersebut akan **memiliki semua method** dan **semua atribut** yang sudah dirancang di class.
- **Instansiasi** merupakan proses pembuatan objek dari class

# Class dan Object

Class	Object
Buah	Apel
	Pisang
	Mangga
Mobil	Toyota
	Daihatsu
	Mitsubishi

Misalkan ada sebuah class bernama **Buah**. Sebuah **Buah** dapat memiliki **properti** seperti **nama**, **warna**, **berat**, dll. Kita dapat mendefinisikan variabel seperti **\$nama**, **\$warna**, dan **\$berat** untuk menyimpan nilai-nilai dari properti ini.

Ketika **objek-objek** individu (**apel**, **pisang**, dll.) dibuat, mereka mewarisi semua properti dan fungsi dari class, tetapi setiap objek akan **memiliki nilai yang berbeda untuk propertinya**.



# Class

- Dideklarasikan dengan kata kunci **class**
- Di dalam class dideklarasikan semua atribut/properti dan fungsi/method yang dimiliki oleh class tersebut

```
<?php
    class Fruit {
        // Properties
        public $name;
        public $color;

        // Methods/function
        function set_name($name) {
            $this->name = $name;
        }
        function get_name() {
            return $this->name;
        }
        function print(){
            echo ("Nama buah = ".$this->name."<br>");
            echo ("Warna = ".$this->color."<br>");
        }
    }
?>
```

# Class

- Deklarasi atribut/property sama dengan deklarasi variable, dan bisa ditambahkan dengan access modifier.
- Deklarasi method/function sama dengan deklarasi fungsi/prosedur

```
<?php
    class Fruit {
        // Properties
        public $name;
        public $color;

        // Methods/function
        function set_name($name) {
            $this->name = $name;
        }
        function get_name() {
            return $this->name;
        }
        function print(){
            echo ("Nama buah = ".$this->name."<br>");
            echo ("Warna = ".$this->color."<br>");
        }
    }
?>
```

# Class

- Kata kunci **this** mengacu pada atribut/properti yang dimiliki oleh objek atau yang biasa disebut sebagai atribut instansiasi

```
<?php
    class Fruit {
        // Properties
        public $name;
        public $color;

        // Methods/function
        function set_name($name) {
            $this->name = $name;
        }
        function get_name() {
            return $this->name;
        }
        function print(){
            echo ("Nama buah = ".$this->name."<br>");
            echo ("Warna = ".$this->color."<br>");
        }
    }
?>
```

# Deklarasi Function/Method

- Fungsi ada blok statement yang bisa digunakan berulang kali dalam program,
- Fungsi tidak otomatis dijalankan program di-load
- Fungsi akan jalan ketika ia dipanggil
- Dideklarasikan menggunakan kata kunci **function**

```
function myMessage() {  
    echo "Hello world!";  
}
```

# Pemanggilan Fungsi

- Untuk memanggil fungsi secara simple bisa dilakukan dengan **memanggil Namanya**

```
function myMessage() {  
    echo "Hello world!";  
}  
myMessage();
```

- Jika **fungsi** tersebut berada di dalam **class** maka pemanggilan fungsi harus melalui **nama\_objeknya->nama\_fungsi();**

```
class Hello{  
    function myMessage() {  
        echo "Hello world!";  
    }  
}  
$hello = new Hello();  
$hello->myMessage();
```

# Fungsi dengan Parameter

- Parameter digunakan untuk melewati nilai dari luar fungsi untuk diolah di dalam fungsi
- Fungsi bisa tidak memiliki parameter, bisa memiliki 1 parameter atau lebih
- Ketika dibutuhkan suatu fungsi untuk mendapatkan nilai dari luar fungsi, maka perlu dibuat parameternya
- Proses melewati nilai ke dalam fungsi bisa dilakukan saat pemanggilan fungsi

```
function familyName($fname) {  
    echo "$fname Refsnes.<br>";  
}
```

```
familyName("Jani");  
familyName("Hege");  
familyName("Stale");  
familyName("Kai Jim");  
familyName("Borge");
```

```
function familyName($fname, $year) {  
    echo "$fname Refsnes. Born in $year <br>";  
}
```

```
familyName("Hege", "1975");  
familyName("Stale", "1978");  
familyName("Kai Jim", "1983");
```

# Nilai Default Parameter

- Parameter dari suatu fungsi bisa dibuat memiliki nilai default
- Jika pada saat pemanggilan fungsi, tanpa melewatkan nilai parameter, maka nilai parameter tersebut dianggap default

```
function setHeight($minheight = 50) {  
    echo "The height is : $minheight <br>";  
}
```

```
setHeight(350);
```

```
setHeight(); // will use the default value of 50
```

```
setHeight(135);
```

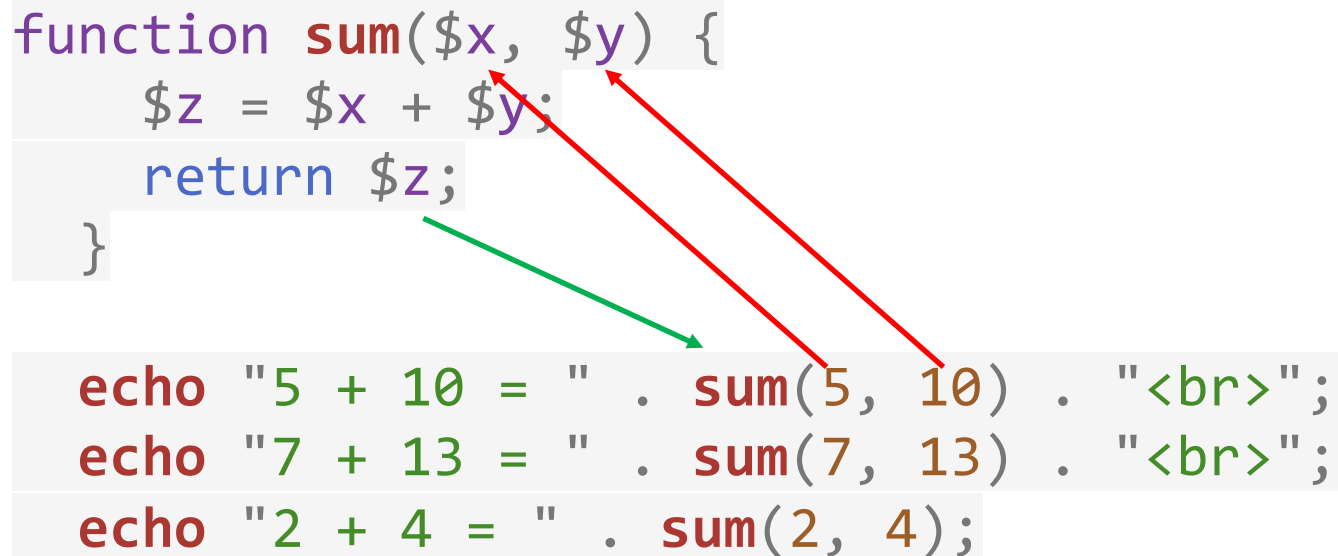
```
setHeight(80);
```

# Fungsi dengan Nilai Kembalian

- Adakalanya suatu fungsi perlu mengembalikan suatu nilai
- Nilai kembalian ini akan diberikan kepada pemangglinya (dimana ia dipanggil)
- Menggunakan statement `return`

```
function sum($x, $y) {  
    $z = $x + $y;  
    return $z;  
}
```

```
echo "5 + 10 = " . sum(5, 10) . "<br>";  
echo "7 + 13 = " . sum(7, 13) . "<br>";  
echo "2 + 4 = " . sum(2, 4);
```

A diagram illustrating the flow of data in PHP. It shows a function definition 'function sum(\$x, \$y) {' with two parameters, \$x and \$y. Below the function definition, there are three function calls: 'sum(5, 10)', 'sum(7, 13)', and 'sum(2, 4)'. Red arrows point from the parameters 5 and 10 in the first call to the \$x and \$y parameters in the function definition. Another red arrow points from the parameters 7 and 13 in the second call to the \$x and \$y parameters. A green arrow points from the 'return \$z;' statement in the function definition to the first function call 'sum(5, 10)', indicating the return value being passed back.



# Objek

- Class tidak bisa diapakan tanpa dibuat objek
- **Class** masih berupa rancangan
- Dari 1 class bisa dibuat **banyak objek**
- Objek memiliki semua atribut dan fungsi yang ada di class
- Objek dibuat menggunakan kata kunci **new** (proses instansiasi)

```
<?php
    class Fruit {
        // Properties
        public $name;
        public $color;
```

```
        // Methods/function
        function set_name($name) {
            $this->name = $name;
        }
        function get_name() {
            return $this->name;
        }
        function print(){
            echo ("Nama buah = ".$this->name."<br>");
            echo ("Warna = ".$this->color."<br>");
        }
    }

    $apple = new Fruit();
    $banana = new Fruit();
    $apple->set_name('Apple');
    $banana->set_name('Banana');
    $apple->print();
    $banana->print();
?>
```

# Objek

- Setelah dibuat objek, class baru dapat digunakan
- Setelah dibuat objek, baru bisa diakses atribut/property dan method/fungsi, dengan cara:

nama\_objek -> atribut atau  
nama\_objek -> fungsi()

```
<?php
class Fruit {
    // Properties
    public $name;
    public $color;

    // Methods/function
    function set_name($name) {
        $this->name = $name;
    }
    function get_name() {
        return $this->name;
    }
    function print(){
        echo ("Nama buah = ".$this->name."<br>");
        echo ("Warna = ".$this->color."<br>");
    }
}

$apple = new Fruit();
$banana = new Fruit();
$apple->set_name('Apple');
$banana->set_name('Banana');
$apple->print();
$banana->print();
?>
```

instansiasi

Pemanggilan method

# Objek

- Setelah dibuat objek, class baru dapat digunakan
- Setelah dibuat objek, baru bisa diakses atribut/property dan method/fungsi, dengan cara:

nama\_objek -> atribut atau  
nama\_objek -> fungsi()

```
<?php
class Fruit {
    // Properties
    public $name;
    public $color;

    // Methods/function
    function set_name($name) {
        $this->name = $name;
    }
    function get_name() {
        return $this->name;
    }
    function print(){
        echo ("Nama buah = ".$this->name."<br>");
        echo ("Warna = ".$this->color."<br>");
    }
}

$apple = new Fruit();
$banana = new Fruit();
$apple->set_name('Apple');
$banana->set_name('Banana');
$apple->print();
$banana->print();
?>
```

instansiasi

Pemanggilan method

# Objek

- Di program tersebut untuk men-set nilai \$name digunakan method set\_name
- Bisa juga dengan dilakukan dari luar class yaitu dengan mengakses objek dan atributnya

```
$apple->name = "Apple";
```

```
<?php
    class Fruit {
        // Properties
        public $name;
        public $color;

        // Methods/function
        function set_name($name) {
            $this->name = $name;
        }
        function get_name() {
            return $this->name;
        }
        function print(){
            echo ("Nama buah = ".$this->name."<br>");
            echo ("Warna = ".$this->color."<br>");
        }
    }

    $apple = new Fruit();
    $banana = new Fruit();
    $apple->set_name('Apple');
    $banana->set_name('Banana');
    $apple->print();
    $banana->print();
?>
```

instansiasi

Pemanggilan method

# Objek

- Operator instanceof digunakan untuk mengecek apakah suatu objek merupakan hasil instansiasi dari suatu class tertentu
- Menghasilkan true/false

```
<?php  
$apple = new Fruit();  
var_dump($apple instanceof Fruit); //menghasilkan TRUE  
?>
```

# Konstruktor

- **Method/fungsi** khusus yang dijalankan saat **pembuatan objek**
- Di konstruktor biasanya diletakkan proses **inisialisasi nilai property/atribut dari suatu objek**, mengingat ini akan dijalankan saat pertama kali objek dibuat
- Otomatis jalan saat dilakukan instansiasi objek
- Format `__construct(){.....}` bisa memiliki parameter atau tidak
- Terlihat dari kode di samping, bahwa dengan ada konstruktor, method setter bisa tidak dituliskan

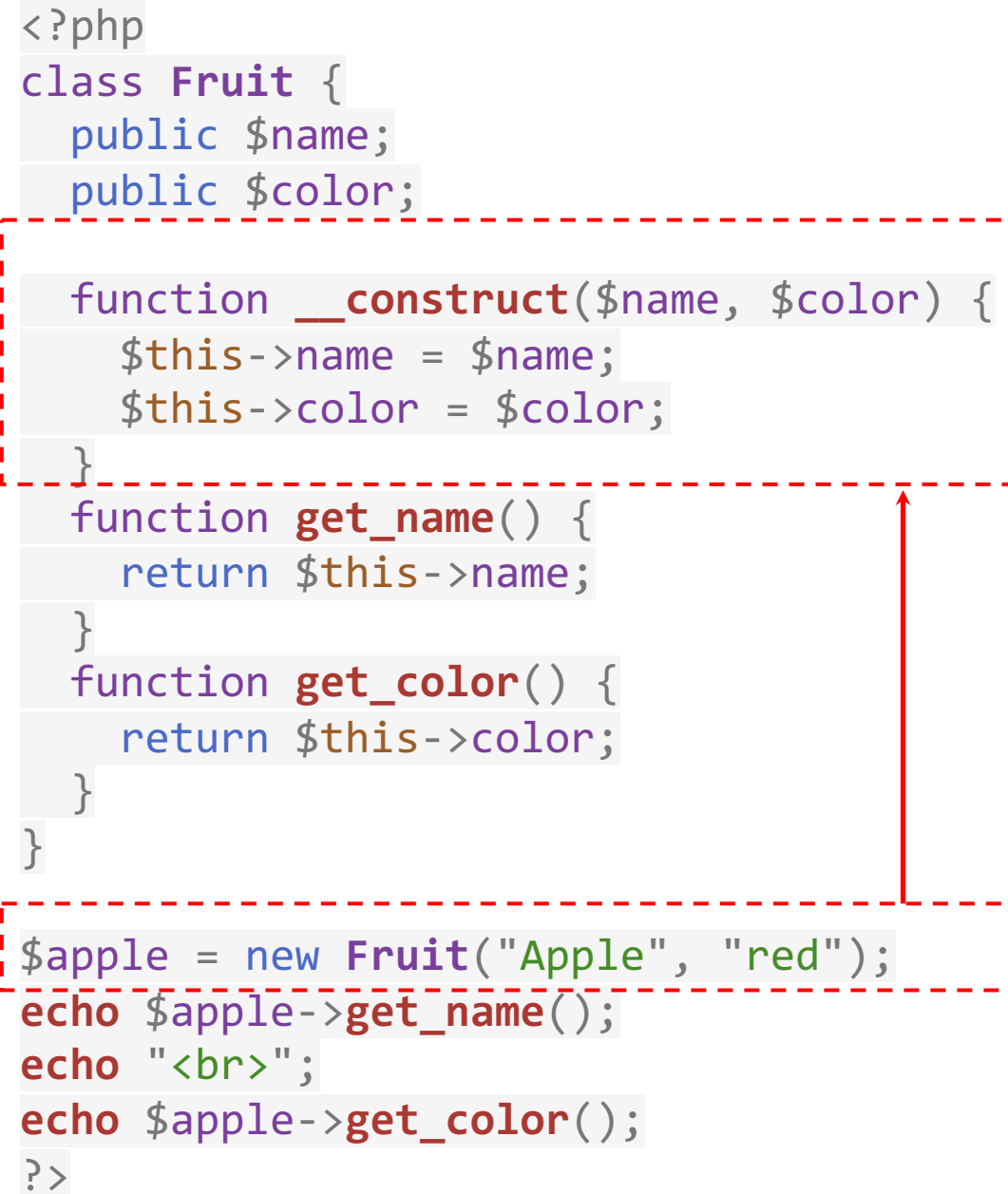
```
<?php
class Fruit {
    public $name;
    public $color;

    function __construct($name, $color) {
        $this->name = $name;
        $this->color = $color;
    }

    function get_name() {
        return $this->name;
    }

    function get_color() {
        return $this->color;
    }
}

$apple = new Fruit("Apple", "red");
echo $apple->get_name();
echo "<br>";
echo $apple->get_color();
?>
```

A red dashed box encloses the constructor and the two getter methods. A red arrow points from the constructor to the instance methods, indicating that the constructor initializes the object's state, which is then accessed by the getters.

# Destruktor

- Destruktor merupakan method khusus yang akan otomatis dijalankan saat objek dihancurkan atau ketika script program dihentikan atau keluar
- Output **The fruit is Apple.**  
Didapatkan Ketika program tersebut dijalankan dan berhenti

```
<?php
class Fruit {
    public $name;
    public $color;

    function __construct($name) {
        $this->name = $name;
    }
    function __destruct() {
        echo "The fruit is {$this->name}.";
    }
}

$apple = new Fruit("Apple");
?>
```

# Access Modifiers

- Properti dan fungsi yang ada di dalam class dapat memiliki **modifier akses** yang mengontrol di mana mereka dapat diakses.
- Ada tiga modifier akses:
  - public - properti atau fungsi dapat diakses dari mana saja. Ini adalah default
  - protected - properti atau fungsi dapat diakses di dalam kelas dan oleh kelas-kelas yang diturunkan dari kelas tersebut
  - private - properti atau fungsi hanya dapat diakses di dalam kelas



# Access Modifiers

- Dalam contoh berikut, ditambahkan tiga modifier akses yang berbeda pada tiga properti (**nama**, **warna**, dan **berat**). Di sini, jika coba diatur properti **nama**, itu akan berfungsi dengan baik (karena properti **nama** adalah **public**, dan dapat diakses dari mana saja). Namun, jika coba diakses properti **warna** atau **berat**, itu akan menghasilkan **kesalahan fatal** (karena properti **warna** dan **berat** adalah **protected** dan **private**)

```
<?php
class Fruit {
    public $name;
    protected $color;
    private $weight;
}

$mango = new Fruit();
$mango->name = 'Mango'; // OK
$mango->color = 'Yellow'; // ERROR
$mango->weight = '300'; // ERROR
?>
```

```

<?php
class Fruit {
    public $name;
    public $color;
    public $weight;

    function set_name($n) { // a public function
        (default)
        $this->name = $n;
    }
    protected function set_color($n) { // a
        protected function
        $this->color = $n;
    }
    private function set_weight($n) { // a private
        function
        $this->weight = $n;
    }
}

$mango = new Fruit();
$mango->set_name('Mango'); // OK
$mango->set_color('Yellow'); // ERROR
$mango->set_weight('300'); // ERROR
?>

```

# Access Modifiers

- Dalam contoh berikut, ditambahkan tiga modifier akses yang berbeda pada tiga properti (**nama**, **warna**, dan **berat**). Di sini, jika coba diatur properti **nama**, itu akan berfungsi dengan baik (karena properti **nama** adalah **public**, dan dapat diakses dari mana saja). Namun, jika coba diakses properti **warna** atau **berat**, itu akan menghasilkan **kesalahan fatal** (karena properti **warna** dan **berat** adalah **protected** dan **private**)

# Inheritance

- Suatu class bisa menurun dari class lain
- Class yang menurun akan mewarisi semua atribut dan fungsi dengan access modifier public dan protected dari parent class nya
- Class turunan juga bisa memiliki atribut dan fungsi tambahan yang spesifik untuknya
- Kata kunci `extends`

# Inheritance

- Class **Strawberry** adalah turunan dari **Fruit**
- Ia mewarisi atribut **name**, **color** dan juga method **intro**, **\_\_construct** Ia juga mempunyai method tambahan yang ia miliki sendiri yaitu method **message**

Am I a fruit or a berry? The fruit is Strawberry and the color is red.

```
<?php
class Fruit {
    public $name;
    public $color;
    public function __construct($name, $color) {
        $this->name = $name;
        $this->color = $color;
    }
    public function intro() {
        echo "The fruit is {$this->name} and the color
is {$this->color}.";
    }
}

// Strawberry is inherited from Fruit
class Strawberry extends Fruit {
    public function message() {
        echo "Am I a fruit or a berry? ";
    }
}

$strawberry = new Strawberry("Strawberry", "red");
$strawberry->message();
$strawberry->intro();
?>
```

# Overriding Method

```
<?php
class Fruit {
    public $name;
    public $color;
    public function __construct($name, $color) {
        $this->name = $name;
        $this->color = $color;
    }
    public function intro() {
        echo "The fruit is {$this->name} and the color is {$this->color}.";
    }
}
```

```
class Strawberry extends Fruit {
    public $weight;
    public function __construct($name, $color, $weight) {
        $this->name = $name;
        $this->color = $color;
        $this->weight = $weight;
    }
    public function intro() {
        echo "The fruit is {$this->name}, the color is {$this->color}, and the weight is {$this->weight} gram.";
    }
}
```

```
$strawberry = new Strawberry("Strawberry", "red", 50);
$strawberry->intro();
?>
```

# Overriding Method

- Method yang sudah diwarisi dari parent class bisa disesuaikan lagi di class turunannya, agar isinya bisa lebih tepat dengan kebutuhan class turunannya
- Fungsi yang **diwarisi** dapat **di-override** dengan mendefinisikan ulang metode tersebut (menggunakan nama yang sama) di kelas turunan.
- Fungsi **\_\_construct()** dan **intro()** di kelas turunan (**Strawberry**) akan menempa metode **\_\_construct()** dan **intro()** di kelas induk (Fruit):

The background is a deep blue with a complex, abstract pattern. A prominent feature is a glowing, circular ring composed of many small, bright blue particles or dots, which appears to be in motion or vibrating. Surrounding this ring are various other patterns of light blue and white particles, some forming wavy, ribbon-like shapes and others appearing as scattered dust or sparks. The overall effect is one of dynamic energy and digital complexity.

# PHP

Form Submission dan data Handling



# Get Request API

Application Programming Interface programming, is the process of using APIs to build software applications.



## API SOAP

API ini menggunakan Simple Object Access Protocol. Klien dan server saling bertukar pesan menggunakan XML. API yang kurang fleksibel ini populer di masa lalu.



# THANK YOU

Universitas Tidar