4.1. Pernyataan Kondisi dan Perulangan

Pernyataan kondisi digunakan apabila kita ingin membandingkan atau mengetahui nilai suatu objek.

4.2. Perintah If

Perintah If adalah sesuatu yang paling penting pada bahasa pemrograman umumnya. Perintah If PHP menyerupai bentuk If pada bahasa C:

```
if (ekspresi)
   Perintah
```

ekspresi adalah sesuatu yang dapat dievaluasi menjadi nilai TRUE atau FALSE.
Berikut ini adalah contoh yang akan mencetak 'a lebih besar dari b' jika nilai $a lebih besar dari $b.

```
if ($a > $b)
    print "a is bigger than b";
```

Jika perintah yang akan dijalankan ketika ekspresi true lebih dari satu, maka perintah-perintah tersebut perlu dikelompokan dengan kurung kurawal { dan }.

```
if ($a > $b) {
    print "a is bigger than b";
    $b = $a;
 }
```

Anda juga dapat membentuk if yang bersangkar. (if didalam if)

4.3. If Else

Sering kita perlu menjalankan perintah lain kalau nilai ekspresi adalah FALSE. Untuk keperluan tersebut kita dapat menggunakan perintah Else.

```
if ($a > $b) {
    print "a is bigger than b";
 } else {
    print "a is NOT bigger than b";
 }
```

4.4. Elseif

ELSEIF adalah kombinasi dari suatu if dan else, anda dapat menggunakannya pada suatu pilihan multi kondisi:

```
if ($a > $b) {
    print "a is bigger than b";
 } elseif ($a == $b) {
    print "a is equal to b";
 } else {
    print "a is smaller than b";
 }
```

4.5. Alternative penulisan untuk struktur kontrol

PHP mendukung penulisan struktur kontrol yang lainnya. Dan ini merupakan alternatif bagaimana menuliskan pernyataan kondisi.
Contoh:

```
<?php if ($a==5): ?>
 A is equal to 5
 <?php endif; ?>
```

Program diatas ini akan tampil jika nilai $a == 5, dan di browser akan keluar teks A is equal to 5

Berikut adalah contoh penggabungan fungsi if, else dan elseif

```
if ($a == 5):
    print "a equals 5";
    print "...";
 elseif ($a == 6):
    print "a equals 6";
    print "!!!";
 else:
    print "a is neither 5 nor 6";
 endif;
```

4.6. While

WHILE adalah perulangan yang paling sederhana pada PHP. Bentuk dasar dari perintah WHILE adalah:

WHILE(ekspresi) statement

Statement akan diulang selama ekspresi memiliki nilai TRUE. Jika pada perulangan ternyata ekspresi bernilai FALSE, maka perulangan tidak pernah dilakukan.
Jika perintah yang akan diulang lebih dari satu, maka perintah-perintah tersebut dapat dikelompokan dengan mengetiknya diantara kurung kurawal { dan }, atau mengunakan tata cara penulisan alternatif:
WHILE(expr): statement ... ENDWHILE;
Berikut ini adalah dua contoh yang identik, yaitu akan mencetak nilai 1 s/d 10:

```
/* contoh 1 */
$i = 1;
 while ($i <= 10) {
    print $i++;  /* the printed value would be
                $i before the increment
                (post-increment) */
 }

 /* example 2 */
  $i = 1;
 while ($i <= 10):
    print $i;
    $i++;
 endwhile;
```

4.7. Do .. While

Perulangan DO..WHILE adalah sama saja dengan perulangan While, cuma ekspresi diperiksa pada akhir dari perulangan. Jadi perulangan jenis ini minimal terjadi satu kali.
Tata cara penulisan untuk perulangan DO..WHILE :

```
$i = 0;
do {
   print $i;
} while ($i>0);
```

Dalam C lanjutan, anda mungkin sering menggunakan do..while loop, untuk menghentikan proses ditengah jalan dan biasanya digunakan perintah break.
Lihat contoh berikut:

```
do {
    if ($i < 5) {
        print "i is not big enough";
        break;
    }
    $i *= $factor;
    if ($i < $minimum_limit) {
        break;
    }
    print "i is ok";
    ...process i...
 } while(0);
```

4.8. FOR

Perulangan FOR adalah perulangan yang paling kompleks dalam PHP, dan menyerupai perulangan FOR pada bahasa C. Tata cara penulisan untuk perulangan FOR

```
FOR (ekspresi1; ekspresi2; ekspresi3) statement
```

Ekspresi pertama (ekspresi1) di evaluasi (dieksekusi) secara un-kondisional pada awal perulangan.
Pada awal dari tiap iterasi, ekspresi2 akan di evaluasi. Jika hasil evaluasi adalah TRUE, maka perulangan akan diteruskan dan statement akan dieksekusi. Jika hasil evaluasi FALSE, maka perulangan diakhiri.
Pada akhir dari tiap iterasi, ekspresi3 akan dievaluasi (dieksekusi).
Contoh berikut akan mencetak angka 1 s/d 10:

```
/* example 1 */
  for ($i = 1; $i <= 10; $i++) {
    print $i;
 }

 /* example 2 */

 for ($i = 1;;$i++) {
    if ($i > 10) {
        break;
    }
    print $i;
 }
```

```
 /* example 3 */
 $i = 1;
for (;;) {
    if ($i > 10) {
        break;
    }
    print $i;
    $i++;
}

 /* example 4 */
  for ($i = 1; $i <= 10; print $i, $i++)
```

4.9. BREAK

BREAK berguna untuk keluar dari perulangan yang sekarang.

```
$i = 0;
$i = 0;
 while ($i < 10) {
    if ($arr[$i] == "stop") {
        break;
    }
    $i++;
}
```

4.10. CONTINUE

CONTINUE digunakan didalam looping untuk keluar dari perulangan dan kemudian melanjutkan eksekusi ke perintah berikutnya.

```
while (list($key,$value) = each($arr))
{
   if ($key % 2)
   {                            // loncati anggota genap
      continue;
   }
   KerjakanYangGanjil ($value);
}
```

4.11. SWITCH

Perintah SWITCH menyerupai sejumlah perintah IF dengan ekspresi yang sama. Sering kali anda ingin membandingkan sejumlah variabel (atau ekspresi) dengan sejumlah nilai dan menjalankan perintah tertentu untuk masing-masing nilai.

```
/* contoh 1 */
if ($i == 0) {
   print "i sama dengan 0";
}
if ($i == 1) {
   print "i sama dengan 1";
}
if ($i == 2) {
   print "i sama dengan 2";
}
```

```
/* contoh 2 */
switch ($i) {
   case 0:
      print "i sama dengan 0";
      break;
   case 1:
      print "i sama dengan 1";
      break;
   case 2:
      print "i sama dengan 2";
      break;
}
```

Perintah SWITCH dieksekusi secara baris per-baris (aktualnya, perintah per-perintah). Pada awalnya tidak ada kode yang dijalankan. Hanya sesaat ketika suatu perintah CASE ditemukan dengan nilai yang sesuai dengan nilai ekspresi pada SWITCH, PHP menjalankan perintah. PHP melanjutkan eksekusi sampai akhir dari blok SWITCH, atau pertama kali menemukan suatu perintah BREAK. Jika anda tidak menulis suatu perintah BREAK pada akhir dari daftar perintah pada suatu case, PHP akan melanjutkan eksekusi untuk case selanjutnya. Sebagai contoh:

```
/* contoh 3 */

switch ($i) {
   case 0:
      print "i sama dengan 0";
   case 1:
      print "i sama dengan 1";
   case 2:
      print "i sama dengan 2";
}
```

Disini, jika $i sama dengan 0, PHP akan mengeksekusikan seluruh perintah print ! Jika $i sama dengan 1, PHP akan mengeksekusikan dua perintah print terakhir, jika dan hanya jika $i sama dengan 2, tercetak 'i equals 2'. Jadi, adalah sangat penting untuk tidak melupakan perintah BREAK.

Suatu case yang khusus adalah case default. Case ini akan sama dengan segala sesuatu yang tidak dapat disamakan dengan case-case sebelumnya. Sebagai contoh:

```
/* contoh 4 */

switch ($i) {
   case 0:
      print "i sama dengan 0";
      break;
   case 1:
      print "i sama dengan 1";
      break;
   case 2:
```

```
        print "i sama dengan 2";
        break;
    default:
        print "i tidak sama dengan 0, 1 atau 2";
}
```

Kenyataaan lain yang perlu diperhatikan adalah ekspresi pada CASE dapat mengevaluasikan segala nilai tipe skalar, yang mana berupa integer, atau real dan string. Array atau objek tidak akan mengakibatkan crash pada PHP, tetapi tidak memiliki arti dalam hal ini.

4.12. REQUIRE

Perintah REQUIRE menganti akan perintah tersebut dengan file tertentu, dan hampir menyerupai prerocessor #include pada bahasa C.
Hal ini berarti anda tidak dapat meletakkan perintah required() dalam suatu struktur perulangan dan mengharapkannya untuk mengikutsertakan file yang berbeda pada tiap iterasi. Untuk melakukan hal tersebut anda harus menggunakan perintah INCLUDE.

```
require ('header.inc');
```

4.13. INCLUDE

Perintah INCLUDE akan mengikutsertakan dan mengevaluasi file tertentu.
Hal ini terjadi setiap kali perintah INCLUDE ditemukan, jadi anda dapat menggunakan perintah INCLUDE diantara suatu struktur perulangan untuk mengikutsertakan sejumlah file yang berbeda.

```
$files = array ('first.inc', 'second.inc', 'third.inc');
for ($i = 0; $i < count($files); $i++) {
    include($files[$i]);
}
```

include() berbeda dengan require() pada perintah include evaluasi dilakukan kembali setiap kali perintah tersebut ditemukan (dan hanya ketika hal tersebut dijalankan), sedangkan perintah require() akan diganti dengan file yang diperlukan ketika pertama kali ditemukan, terserah apakah file tersebut akan dievaluasi atau tidak (sebagai contoh, jika berada dalam suatu perintah if yang memiliki hasil evaluasi false)
Karena include() adalah suatu kontruksi bahasa khusus, anda harus mengapitnya diantara blok, jika perintah tersebut dalam suatu blok kondisi.

```
/* This is WRONG and will not work as desired. */
  if ($condition)
     include($file);
 else
     include($other);

 /* This is CORRECT. */
  if ($condition) {
     include($file);
 } else {
     include($other);
 }
```

Ketika file tersebut dievaluasi, pemisah antara "modus HTML" yang mana akan dicetak sampai tag (<) PHP pertama kali ditemukan.