Loop, Conditional, Array, Function

Pemrograman Web

Topik Bahasan

- If ..... else Statement
- Switch Satement
- For Loop Statement
- While Loop Statement
- For Loop Statement

Conditional Statements

In PHP we have the following conditional statements:

- **if statement** - use this statement to execute some code only if a specified condition is true
- **if...else statement** - use this statement to execute some code if a condition is true and another code if the condition is false
- **if...elseif....else statement** - use this statement to select one of several blocks of code to be executed
- **switch statement** - use this statement to select one of many blocks of code to be executed

- **The if Statement**
  - Use the if statement to execute some code only if a specified condition is true.
  - Syntax:

    if (*condition*) *code to be executed if condition is true;*

  - It will output "Have a nice weekend!" if the current day is Friday

```
<html>
<body>

<?php
$d=date("D");
if ($d=="Fri") echo "Have a nice weekend!";
?>

</body>
</html>
```

- Use **break** to prevent the code from running into the next case automatically. The default statement is used if no match is found.

```
<html>
<body>

<?php
$x=1;
switch ($x)
{
case 1:
  echo "Number 1";
  break;
case 2:
  echo "Number 2";
  break;
case 3:
  echo "Number 3";
  break;
default:
  echo "No number between 1 and 3";
}
?>

</body>
</html>
```

PHP Switch Statement

- Conditional statements are used to perform different actions based on different conditions.
- Use the switch statement to select one of many blocks of code to be executed.

```
switch (n)
{
case label1:
  code to be executed if n=label1;
  break;
case label2:
  code to be executed if n=label2;
  break;
default:
  code to be executed if n is different from both label1 and label2;
}
```

PHP Loop Types

- Loops in PHP are used to execute the same block of code a specified number of times.

- PHP supports following four loop types:
  - **for -** loops through a block of code a specified number of times.
  - **while -** loops through a block of code if and as long as a specified condition is true.
  - **do...while -** loops through a block of code once, and then repeats the loop as long as a special condition is true.
  - **foreach -** loops through a block of code for each element in an array.

The for loop statement

- The for statement is used when you know how many times you want to execute a statement or a block of statements.
- **Syntax**

```
for (initialization; condition; increment)
{
  code to be executed;
}
```

- Example

```
<html>
<body>
<?php
$a = 0;
$b = 0;

for( $i=0; $i<5; $i++ )
{
    $a += 10;
    $b += 5;
}
echo ("At the end of the loop a=$a and b=$b" );
?>
</body>
</html>
```

- Result

```
At the end of the loop a=50 and b=25
```

The while loop statement

- The while statement will execute a block of code if and as long as a test expression is true.

- Syntax

```
while (condition)
{
    code to be executed;
}
```

- Example

```
<html>
<body>
<?php
$i = 0;
$num = 50;

while( $i < 10)
{
   $num--;
   $i++;
}
echo ("Loop stopped at i = $i and num = $num" );
?>
</body>
</html>
```

- Result

```
Loop stopped at i = 10 and num = 40
```

The do...while loop statement

- The do...while statement will execute a block of code at least once - it then will repeat the loop as long as a condition is true.

- Syntax

```
do
{
   code to be executed;
}while (condition);
```

- Example

```
<html>
<body>
<?php
$i = 0;
$num = 0;
do
{
  $i++;
}while( $i < 10 );
echo ("Loop stopped at i = $i" );
?>
</body>
</html>
```

- Result

```
Loop stopped at i = 10
```

The foreach loop statement

- The foreach statement is used to loop through arrays.

- For each pass the value of the current array element is assigned to $value and the array pointer is moved by one and in the next pass next element will be processed.

- Syntax

```
foreach (array as value)
{
    code to be executed;

}
```

- Example

```
<html>
<body>
<?php
$array = array( 1, 2, 3, 4, 5);
foreach( $array as $value )
{
  echo "Value is $value <br />";
}
?>
</body>
</html>
```

- Result

```
Value is 1
Value is 2
Value is 3
Value is 4
Value is 5
```

PHP Arrays

- An array is a special variable, which can store multiple values in one single variable.

```
$cars1="Saab";
$cars2="Volvo";
$cars3="BMW";
```

- There are three kind of arrays:
  - **Numeric array** - An array with a numeric index
  - **Associative array** - An array where each ID key is associated with a value
  - **Multidimensional array** - An array containing one or more arrays

Numeric Arrays

- A numeric array stores each array element with a numeric index.
- There are two methods to create a numeric array.
  - the index are automatically assigned (the index starts at 0).

    $cars=array("Saab","Volvo","BMW","Toyota");
  - assign the index manually.

    $cars[0]="Saab";
    $cars[1]="Volvo";
    $cars[2]="BMW";
    $cars[3]="Toyota";

- Example: access the variable values by referring to the array name and index.

```
$cars[0]="Saab";
$cars[1]="Volvo";
$cars[2]="BMW";
$cars[3]="Toyota";
```

- Output:

  Saab and Volvo are Swedish cars.

Associative Arrays

- An associative array, each ID key is associated with a value.

- Example 1: an array to assign ages to the different persons.

```
$ages = array("Peter"=>32, "Quagmire"=>30, "Joe"=>34);
```

- Example 2: shows a different way of creating the array.

```
$ages['Peter'] = "32";
$ages['Quagmire'] = "30";
$ages['Joe'] = "34";
```

- The ID keys can be used in a script.

```
<?php
$ages['Peter'] = "32";
$ages['Quagmire'] = "30";
$ages['Joe'] = "34";

echo "Peter is " . $ages['Peter'] . " years old.";
?>
```

- Output: Peter is 32 years old.

Multidimensional Arrays

- In a multidimensional array, each element in the main array can also be an array. And each element in the sub-array can be an array, and so on.
- Create a multidimensional array, with automatically assigned ID keys.

```
$families = array
   (
   "Griffin"=>array
   (
   "Peter",
   "Lois",
   "Megan"
   ),
   "Quagmire"=>array
   (
   "Glenn"
   ),
   "Brown"=>array
   (
   "Cleveland",
   "Loretta",
   "Junior"
   )
   );
```

- The array above is similar to

```
Array
(
[Griffin] => Array
   (
   [0] => Peter
   [1] => Lois
   [2] => Megan
   )
[Quagmire] => Array
   (
   [0] => Glenn
   )
[Brown] => Array
   (
   [0] => Cleveland
   [1] => Loretta
   [2] => Junior
   )
)
```

- Example: Lets try displaying a single value from the array above.

```
echo "Is " . $families['Griffin'][2] .
" a part of the Griffin family?";
```

- Output: Is Megan a part of the Griffin family?

Get The Length of an Array - The count() Function

```php
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo count($cars);
?>
```

Loop Through an Indexed Array

```php
<?php
$cars = array("Volvo", "BMW", "Toyota");
$arrlength = count($cars);

for($x = 0; $x < $arrlength; $x++) {
    echo $cars[$x];
    echo "<br>";
}
?>
```

Loop Through an Associative Array

- To loop through and print all the values of an associative array:

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

foreach($age as $x => $x_value) {
    echo "Key=" . $x . ", Value=" . $x_value;
    echo "<br>";
}
?>
```

PHP - Sort Functions For Arrays

- sort() - sort arrays in ascending order
- rsort() - sort arrays in descending order
- asort() - sort associative arrays in ascending order, according to the value
- ksort() - sort associative arrays in ascending order, according to the key
- arsort() - sort associative arrays in descending order, according to the value
- krsort() - sort associative arrays in descending order, according to the key

Sort Array in Ascending Order - sort()

- example sorts the elements of the $cars array in ascending alphabetical order:
  - <?php
    $cars = array("Volvo", "BMW", "Toyota");
    sort($cars);
    ?>

sorts the elements of the $numbers array in ascending numerical order

```php
<?php
$numbers = array(4, 6, 2, 22, 11);
sort($numbers);
?>
```

Sort Array in Descending Order - rsort()

```php
<?php
$cars = array("Volvo", "BMW", "Toyota");
rsort($cars);
?>
```

sorts the elements of the $numbers array in descending numerical order

```php
<?php
$numbers = array(4, 6, 2, 22, 11);
rsort($numbers);
?>
```

Sort Array (Ascending Order), According to Value - asort()

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

Sort Array (Ascending Order), According to Key - ksort()

- <?php
  $age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
  ksort($age);
  ?>

Sort Array (Descending Order), According to Value - arsort()

- <?php
  $age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
  arsort($age);
  ?>

Sort Array (Descending Order), According to Key - krsort()

- <?php
  $age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
  krsort($age);
  ?>

PHP User Defined Functions

- Besides the built-in PHP functions, we can create our own functions.
  - A function is a block of statements that can be used repeatedly in a program.
  - A function will not execute immediately when a page loads.
  - A function will be executed by a call to the function.
- **Syntax**

```
function functionName() {
    code to be executed;
}
```

Contoh function

```
<?php
function writeMsg() {
    echo "Hello world!";
}

writeMsg(); // call the function
?>
```

PHP Function Arguments

- Information can be passed to functions through arguments.
- An argument is just like a **variable**.
- Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

Example:

- The following example has a function with one argument ($fname). When the familyName() function is called, we also pass along a name (e.g. Jani), and the name is used inside the function, which outputs several different first names, but an equal last name:

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

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

The following example has a function with two arguments ($fname and $year):

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

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

PHP Default Argument Value

- If we call the function setHeight() without arguments it takes the default value as argument:

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

setHeight(350);
setHeight(); // will use the default value of 50
setHeight(135);
setHeight(80);
?>
```

PHP Functions - Returning values

```php
<?php
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);
?>
```

1. Soal array

Tampilkan nama & nrp mahasiswa ke layar, dengan kondisi seperti berikut:

1. IPK mahasiswa > 3.00 dan < 3.25
2. Mahasiswa berambut hitam dan lurus
3. Mahasiswa memiliki berat badan 50 kg dan tinggi badan 160 cm
4. Mahasiswa tidak sedang menjalani cuti

- NRP 1 – 5: if-else
- NRP 6-10: switch-case
- NRP 11-15: do-while
- NRP 16-20: for-loop
- NRP 21-30:while

2. Soal sorting

Dari soal nomer 1 diatas, tampilkan nama & nrp mahasiswa dengan kondisi berikut:

a. Simpan dalam suatu function

b. Function memiliki return value

c. Format menampilkan nama & nrp sebagai berikut:

i. Nama sort ascending

ii. NRP sort descending

3. Soal function

Tampilkan nama mahasiswa ke layar, dengan kondisi seperti berikut:

1. Kelas mahasiswa : D4-A dan D4- B
2. Dosen wali mahasiswa: Desy Intan Permatasari
3. Nilai mata kuliah konsep pemrograman: A
4. Mahasiswa aktif di HIMIT

Kondisi khusus:

1. Buat dalam suatu function, dengan parameter dari user : NRP
2. Return value : nama mahasiswa
3. Sorting nama ASCENDING
4. Template kalimat yang akan tampil di layar:

   namaMhs adalah mahasiswa kelas **namaKelasMhs** yang aktif di HIMIT

namaMhs dan **namaKelasMhs** sifatnya data yang ada di program