

Лабораторно упражнение № 2

Управляващи структури. Оператори за цикли – while и do...while

1. Оператор if

Структурата на оператора *if*.

Кратка форма на оператора if - Условната конструкция *if* се използва за проверка на истинността на даден израз и задаване на последващо действие.

if (израз)

блок_за_изпълнение;

Пълната форма на оператора if - Когато се налага да се изпълнят две взаимоизключващи се алтернативи, се използва пълната форма на оператора *if*.

if (израз)

блок_за_изпълнение1;

else

блок_за_изпълнение2;

.....

if (израз)

блок_за_изпълнение1;

elseif (израз1)

блок_за_изпълнение2;

...

Стойността на израза се преобразува в логическа. Ако тази стойност е TRUE се изпълнява блок_за_изпълнение, в противен случай се прескача. Ако блок_за_изпълнение съдържа няколко команди, то той се поставя във фигурни скоби {}. Операторът *elseif* е алтернатива на няколко вложени оператора *if*. Той се използва когато в клаузата *else* се налага проверка на друг израз. В един оператор *if* може да има няколко клаузи *elseif*, които се проверяват последователно и се изпълнява първият *elseif*, чиито израз е със стойност TRUE. В PHP този оператор може да се напише и като *else if* (вложен *if*), които имат еквивалентно действие.

Правилата за преобразуване на израза към логически тип:

1. За **FALSE** се считат:
 - Логическа стойност **FALSE**
 - целочислена нула (0)
 - реална нула (0.0)
 - празен стринг и стринг "0"
 - масив без елементи
 - тип **NULL**
2. Всички останали значения се преобразуват в **TRUE**.

Пример1.

```
<?php
$a=10;
$b=10;
if ($a > $b)
    echo "$a е по-голямо от $b";
elseif ($a == $b)
    echo "$a е равно на $b";
else
    echo "$a е по-малко от $b";
?>
```

Резултат:

10 е равно на 10

Алтернативен синтаксис

PHP предлага *алтернативен синтаксис* за някои свои управляващи структури, а именно за *if*, *while*, *for*, *foreach* и *switch*. Във всеки от случаите, отварящата скоба трябва да се замени с двоеточие (:), а затварящата скоба съответно с **endif**;, **endwhile**;; и т.н..

Например, синтаксиса на оператора *if* може да се запише като:

if(израз): блок_за_изпълнение **endif**;

Смисълът е същия: Ако условието, записано в кръглите скоби на оператора *if* е истина, ще се изпълни целия код, от двоеточието «:» до команда **endif**;

Пример 2. Алтернативен синтаксис на if .

```
<?php
$a=10;
$b=10;
if ($a > $b):
    echo "$a е по-голямо от $b";
elseif ($a === $b):
    echo "$a е равно на $b";
else:
    echo "$a е по-малко от $b";
endif;
?>
```

Резултат:

10 е равно на 10

Пример 3. (Date/Time Functions): Един пример с *if* и функция `date("D")`, която връща стринг от 3 букви, форматиран съобразно форматирания спецификатор `D` - за ден от седмицата (`D` - форматиращ спецификатор за ден от седмицата като 3 букви).

Вариант 1.

```
<?php
$d = date("D"); // Функция date("D") връща текущия ден: Mon...Sun
if($d=="Fri")
    echo "Today is $d. Have a nice weekend!";
elseif($d == "Sun")
    echo " Today is $d. Have a nice Sunday!";
else
    echo " Today is $d. Have a nice day"
?>
```

Резултат:

Today is Fri. Have a nice weekend!

Вариант 2 (с форма).

Please enter the day /Mon...Sun/:

The day is not Fry
The day is Mon

```
<!DOCTYPE html>
<html>
<head>
```

```

    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title></title>
  </head>
  <body>
  <form method="post" action="#">
  Please enter the day /Mon...Sun/: <br />
  <input type="text" name="data" />
  <p />
  <input type="submit" name="submit" value="Go" />
  </form>
  <?php
  if (isset($_POST['submit'])){ //Retrieve string from form submission.
    $d = $_POST["data"];
  if ($d==date("D"))
    echo 'The day is '.$d;
  else
    echo 'The actual day is not '.$d.<br>The day is '.date("D");
  }
  ?>
  </body>
  </html>

```

- Функция date() връща текущата дата и има следния общ синтаксис:

string date (string \$format [, int \$timestamp = time()]).

Най-често се използва само форматиращия спецификатор (например F - форматиращ спецификатор за месец във вид на пълен текст, например March).

Пример:

```

<?php
$d = date("D");
echo $d;
echo "<br>";
$m=date("F");
echo $m;
echo "<br>";
$m=date("n");
echo $m;
echo "<br>";
$today = date("F j, Y, g:i a");
echo $today;
echo "<br>";
$today = date("m.d.y");
echo $today;
echo "<br>";
$today=date("d.m.y");
echo $today;
echo "<br>";
?>

```

Резултат:

```

Sun
February
2
February 21, 2016, 10:20 am
02.21.16
21.02.16

```

Забележка: На <http://bg2.php.net/manual/en/ref.datetime.php> може да видите пълният списък на функциите за дати/време.

Date/Time Functions

- [checkdate](#) — Validate a Gregorian date
- [date_add](#) — Alias of `DateTime::add`
- [date_create_from_format](#) — Alias of `DateTime::createFromFormat`
- [date_create](#) — Alias of `DateTime::__construct`
- [date_date_set](#) — Alias of `DateTime::setDate`
- [date_default_timezone_get](#) — Gets the default timezone used by all date/time functions in a script
- [date_default_timezone_set](#) — Sets the default timezone used by all date/time functions in a script
- [date_diff](#) — Alias of `DateTime::diff`
- [date_format](#) — Alias of `DateTime::format`
- [date_get_last_errors](#) — Alias of `DateTime::getLastErrors`
- [date_interval_create_from_date_string](#) — Alias of `DateInterval::createFromDateString`
- [date_interval_format](#) — Alias of `DateInterval::format`
- [date_isodate_set](#) — Alias of `DateTime::setISODate`
- [date_modify](#) — Alias of `DateTime::modify`
- [date_offset_get](#) — Alias of `DateTime::getOffset`
- [date_parse_from_format](#) — Get info about given date formatted according to the specified format
- [date_parse](#) — Returns associative array with detailed info about given date
- [date_sub](#) — Alias of `DateTime::sub`
- [date_sun_info](#) — Returns an array with information about sunset/sunrise and twilight begin/end
- [date_sunrise](#) — Returns time of sunrise for a given day and location
- [date_sunset](#) — Returns time of sunset for a given day and location
- [date_time_set](#) — Alias of `DateTime::setTime`
- [date_timestamp_get](#) — Alias of `DateTime::getTimestamp`
- [date_timestamp_set](#) — Alias of `DateTime::setTimestamp`
- [date_timezone_get](#) — Alias of `DateTime::getTimezone`
- [date_timezone_set](#) — Alias of `DateTime::setTimezone`
- [date](#) — Format a local time/date
- [getdate](#) — Get date/time information
- [gettimeofday](#) — Get current time
- [gmdate](#) — Format a GMT/UTC date/time
- [gmmktime](#) — Get Unix timestamp for a GMT date
- [gmstrftime](#) — Format a GMT/UTC time/date according to locale settings
- [idate](#) — Format a local time/date as integer
- [localtime](#) — Get the local time
- [microtime](#) — Return current Unix timestamp with microseconds
- [mktime](#) — Get Unix timestamp for a date
- [strftime](#) — Format a local time/date according to locale settings
- [strptime](#) — Parse a time/date generated with `strftime`
- [strtotime](#) — Parse about any English textual datetime description into a Unix timestamp
- [time](#) — Return current Unix timestamp
- [timezone_abbreviations_list](#) — Alias of `DateTimeZone::listAbbreviations`
- [timezone_identifiers_list](#) — Alias of `DateTimeZone::listIdentifiers`
- [timezone_location_get](#) — Alias of `DateTimeZone::getLocation`
- [timezone_name_from_abbr](#) — Returns the timezone name from abbreviation
- [timezone_name_get](#) — Alias of `DateTimeZone::getName`
- [timezone_offset_get](#) — Alias of `DateTimeZone::getOffset`
- [timezone_open](#) — Alias of `DateTimeZone::__construct`
- [timezone_transitions_get](#) — Alias of `DateTimeZone::getTransitions`

- `timezone_version_get` — Gets the version of the `timezonedb`

2. Оператор `switch`

Структурата на `switch` може да се запише по следния начин:

```
switch (израз или променлива){
case значение1:
    блок_от_оператори1;
    break;
case значение2:
    блок_от_оператори2;
    break;
...
default:
    блок_от_оператори_по_подразбиране;
}
```

За разлика от `if`, тук стойността на израза не се преобразува към логически тип, а просто се сравнява със значенията след `case` (`значение1`, `значение2` и т.н.). Ако стойността на израза съвпада с някакъв вариант, то се изпълнява съответстващия блок_от_оператори – от двоеточието до края на `switch` или до първото срещане на `break`. Ако значение на израза не съвпада с нито един от вариантите, то изпълняваме `блок_от_оператори_по_подразбиране` след `default`. Израз в `switch` се изчислява само един път, а в оператор `elseif` – всеки път, ако изразът е достатъчно сложен, то `switch` оператор работи по-бързо от `if`.

Пример 4.

```
<?php
    $i=2;
    switch ($i) {
    case 0:
    case 1:
    case 2: echo "$i е по-малко от 3 ";
        break;
    case 3:
        echo " $i е равно на 3";
    default:
        echo " $i е по-голямо от 3";
    }
?>
```

Резултат:

2 е по-малко от 3

3. Цикли

В PHP съществуват няколко конструкции за цикъл: `while`, `do while`, `foreach` и `for`.

`while`

Структура:

```
while (израз) { блок_за_изпълнение; }
```

или алтернативен синтаксис:

```
while (израз): блок_за_изпълнение endwhile;
```

`while` – командите в `блок_за_изпълнение` се изпълняват докато резултата от израз е TRUE. Стойността му се изчислява преди всяко изпълнение на цикъла. Ако тази стойност е FALSE при първото влизане в цикъла, то той няма да се изпълни нито веднъж. Ако тялото на цикъла съдържа два или повече оператора, те се заграждат с `{}`. Значението на израза се проверява всеки път в

началото на цикъла, така че дори и да настъпи промяна на стойността на израза в тялото на цикъла, цикълът няма да спре.

Пример 5. Отпечатва всички четни числа между 1 и 9.

```
<?php
//print all even numbers
$i = 1;
while ($i < 10) {
    if ($i % 2 == 0){
        print $i;
        // print the numner, if even
        print '<br>';
    }
    $i++;
    // increment $i with 1
}
?>
```

do... while

Този оператор винаги се изпълнява поне веднъж.

Структура:

```
do {
блок_за_изпълнение;
}while (израз);
```

Пример 6. Отпечатва всички четни числа между 1 и 9.

```
<?php
$i = 1;
do{
    if ($i % 2 == 0){
        print $i;
        print '<br>';
    }
    $i++;
}while ($i<10);
?>
```