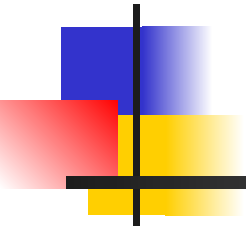
PHP: управляющие конструкции



Краткий обзор

Управляющие конструкции

- Порядок выполнения –

- Управляющие конструкции (control structures) – специальные средства языка программирования, предназначенные для организации алгоритмов нелинейной структуры
- К управляющим конструкциям относят ветвления, циклы, переходы и т.д.

Управляющие конструкции PHP

- Операторы передачи управления
  - goto
  - break | continue
  - include
  - return
- Условные операторы:
  - if | if … else | if … elseif
  - switch
- Циклы
  - while | do … while
  - for
  - foreach

Оператор безусловного перехода **goto**

- **Синтаксис:**
  goto label
- **Значение:**
  передача управления по метке
- **Пример:**

```
goto a;
echo 'я следую за GOTO';
...
a: echo 'я помечен меткой a';
```

Условный оператор **if**

- **Синтаксис:**
  if (expression) statement
- **Значение:**
  если значение выражения expression истинно, будет выполнена инструкция statement
- **Пример:**

```
if ( $a > $b) echo 'a больше b'
...
if ( $a ) {
  echo 'значение a, приведенное к булевому типу - TRUE';
  echo '<br> Тип и значение a:';
  var_dump ($a)
}
```

Вариант условного оператора if - else

- **Синтаксис:**
  if (expression) statement_1 **else** statement_2
- **Значение:**
  если значение выражения expression истинно, будет выполнена инструкция statement_1 , иначе – инструкция statement_2
- **Пример:**

```
if ( $a > $b)
  echo 'a больше b'
else
  echo 'a не больше b';
...
```

вариант условного оператора if - elseif

- синтаксис
  if (выражение_А) блок_А
  elseif (выражение_Б) блок_Б
  elseif (выражение_В) блок_В
  ***
  else (выражение_К)
-
-

Оператор if_elseif: пример 1

```
<?php
$a=5; $b=5;
 if  ($a > $b)
echo “a больше b”;
ifelse ($a < $b)
echo “a меньше b;
else
echo “a равно b”;
?>
```

a равно b

Цикл while

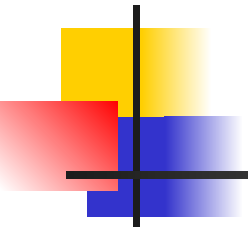
- while (выражение) инструкция
- Повторяет инструкцию, блок инструкций, пока выражение истинно
- $i = 0;
  while ($i < 10) {   echo $i; $i=$i+1; }
- do инструкция while (выражение)
- Выполняет инструкцию, затем проверяет истинность выражения. В случае истинности возвращается к повторению выполнения инструкции
- <?php
  $i = 0;
  do { echo $i;
  } while ($i > 0);
  ?>

Цикл **do - while**

- Синтаксис:
  do инструкция while (выражение)
- Значение:
  циклическое выполнение **инструкции** до тех пор, пока **выражение** истинно
- Пример:
  do echo $i-= , ";" while ($i > 0);

Цикл **for**

- for (expr1; expr2; expr3) statement
- Первое выражение (*expr1*) всегда вычисляется (выполняется) только один раз в начале цикла.
- В начале каждой итерации оценивается выражение *expr2*. Если оно принимает значение ***TRUE***, то цикл продолжается, и вложенные операторы будут выполнены. Если оно принимает значение ***FALSE***, выполнение цикла заканчивается.
- В конце каждой итерации выражение *expr3* вычисляется (выполняется).
- for ($i = 0; $i <= 10; $i++) echo $i,' '



Цикл **foreach**

Директива **break**

- break прерывает выполнение текущей итерации цикла for, foreach, while, do-while
- Используется также применительно конструкции switch.
- break принимает необязательный числовой аргумент, который сообщает ему выполнение какого количества вложенных структур необходимо прервать.

Директива **continue**

- continue используется внутри циклических структур для пропуска оставшейся части текущей итерации цикла и, при соблюдении условий, начала следующей итерации.
- continue принимает необязательный числовой аргумент, который указывает на скольких уровнях вложенных циклов будет пропущена оставшаяся часть итерации. Значением по умолчанию является 1, при которой пропускается оставшаяся часть текущего цикла.

вариант условного оператора switch

- синтаксис
  switch (выражение) {
  case значение_A: блок_A break;
  case значение_B: блок_B break;
  default: блок_0

Оператор switch: пример 1

```
<?php
$a=1;
 switch  ($a)
case 0: echo “a=0”; break;
case 1: echo “a=1”; break;
case 2: echo “a=2”; break;
?>
```

a=1

Оператор switch: пример 2

```
<?php
$fruit=“яблоко”; {
 switch  ($fruit)
case “апельсин”:
     echo “апельсина=0”; break;
case “тыква”:
     echo “тыква: break;
case “яблоко”:
     echo “яблоко”; break;}
?>
```

Директива **include**

- Выражение include включает и выполняет указанный файл.
- Файлы включаются исходя из пути указанного файла, или, если путь не указан, используется путь, указанный в директиве include_path. Если файл не найден в include_path, include попытается проверить директорию, в которой находится текущий включающий скрипт и текущую рабочую директорию перед тем, как выдать ошибку.
- Конструкция include выдаст warning, если не сможет найти файл;

Директива **return**

- return возвращает управление программой в вызывавший модуль. Выполнение возвращается в выражение, следующее после вызова текущего модуля.
- Если вызвано из функции, выражение return немедленно прекращает выполнение текущей функции и возвращает свой аргумент как значение данной функции.
- Если вызывается из глобальной области видимости, выполнение текущего файла скрипта прекращается. Если текущий файл скрипта был подключен с помощью функций include или require, тогда управление возвращается к файлу, который вызывал текущий. Более того, если текущий файл скрипта был подключен с помощью include, тогда значение переданное return будет возвращено в качестве значения вызова include.