**Лабораторная работа №10**

**Конструкция if-else в PHP**

Сейчас мы с вами изучим специальную конструкцию if-else, которая позволяет выполнять некоторый код в зависимости от выполнения какого-либо условия. Синтаксис конструкции выглядит так:

<?php

if (операция сравнения) {

/\*

расположенный здесь код выполнится,

если результат операция сравнения верный

\*/

} else {

/\*

расположенный здесь код выполнится,

если результат операция сравнения неверный

\*/

}

?>

**Операторы больше и меньше**

Для проверки того, которое из значений больше, а которое меньше, используются операторы больше >, больше или равно >=, меньше <, меньше или равно <=.

Изучим их работу на практическом примере. Пусть у нас есть переменная $test с каким-то значением:

<?php

$test = 1;

?>

Давайте проверим, значение этой переменной больше нуля или нет:

<?php

$test = 1;

if ($test > 0) {

echo '+'; // сработает это

} else {

echo '-';

}

?>

А теперь поменяем значение переменной на отрицательное:

<?php

$test = -1; // поменяем значение переменной

if ($test > 0) {

echo '+';

} else {

echo '-'; // сработает это

}

?>

Пусть теперь значение переменной равно 0. В этом случае мы попадем в блок else, так как в нашем условии написано, что переменная $test должна быть строго больше нуля:

<?php

$test = 0;

if ($test > 0) {

echo '+';

} else {

echo '-'; // сработает это

}

?>

Давайте поменяем условие на больше или равно:

<?php

$test = 0;

if ($test >= 0) {

echo '+'; // сработает это

} else {

echo '-';

}

?>

**Задача №1**

Проверьте то, что переменная $test больше 10.

**Задача №2**

Проверьте то, что переменная $test меньше 10.

**Задача №3**

Проверьте то, что переменная $test больше или равна 10.

**Задача №4**

Проверьте то, что переменная $test меньше или равна 10.

**Проверка на равенство**

Для проверки двух значений на равенство используется оператор ==. Давайте для примера проверим, что переменная $test равна 0:

<?php

$test = 0; // пусть значение переменной равно 0

if ($test == 0) {

echo '+'; // сработает это, так как переменная равна 0

} else {

echo '-';

}

?>

Поменяем значение переменной так, чтобы условие не выполнялось:

<?php

$test = 1; // пусть значение переменной равно 1

if ($test == 0) {

echo '+';

} else {

echo '-'; // сработает это, так как переменная НЕ равна 0

}

?>

**Задача №5**

Проверьте то, что переменная $test равна 10.

**Проверка на неравенство**

Существует также оператор !=, который наоборот, проверяет на неравенство. Давайте для примера проверим, что test переменная не равна 0:

<?php

$test = 1; // пусть значение переменной равно 1

if ($test != 0) {

echo '+'; // сработает это, так как переменная НЕ равна 0

} else {

echo '-';

}

?>

Поменяем значение переменной:

<?php

$test = 0; // пусть значение переменной равно 0

if ($test != 0) {

echo '+';

} else {

echo '-'; // сработает это, так как переменная равна 0

}

?>

**Задача №6**

Проверьте то, что переменная $test не равна 10.

**Сравнение переменных**

В примерах выше мы сравнивали переменную с каким-то числом. Но никто не запрещает нам сравнивать две переменные.

Смотрите пример:

<?php

$test1 = 1;

$test2 = 2;

if ($test2 > $test1) {

echo '+'; // сработает это, так как $test2 больше $test1

} else {

echo '-';

}

?>

**Задача №7**

Даны переменные $test1 и $test2. Проверьте, что значение какой из этих переменных больше и выведите соответствующее сообщение на экран.

**Типы данных и конструкция if-else в PHP**

При использовании конструкции if можно сравнивать и строки. Для примера пусть в переменной $test хранится какая-то строка, например такая:

<?php

$test = 'abc';

?>

Давайте проверим, равно ли содержимое переменной $test строке 'abc':

<?php

$test = 'abc';

if ($test == 'abc') {

echo '+'; // сработает это, так как переменная равна 'abc'

} else {

echo '-';

}

?>

**Числа в кавычках**

Как вы знаете, число в кавычках представляет собой строку. Например, '3' - это строка. Однако, при сравнении таких строк с настоящими числами PHP считает, что строка в кавычках равна такому же числу.

Давайте для примера сравним строку '3' и число 3:

<?php

if ('3' == 3) {

echo '+'; // сработает это - значения равны

} else {

echo '-';

}

?>

**Равенство по значению и типу**

Пусть вы хотите сравнивать так, чтобы число в кавычках не было равно такому же числу без кавычек. Говоря другими словами, вы хотите сравнивать так, чтобы сравнение было не только по значению, но и по типу данных. Для этого вместо оператора == следует использовать оператор ===.

Смотрите пример:

<?php

if ('3' === 3) {

echo '+';

} else {

echo '-'; // сработает это, тк значения не равны по типу

}

?>

А вот при сравнении двух строк '3' на экран выведется '+':

<?php

if ('3' === '3') {

echo '+'; // сработает это

} else {

echo '-';

}

?>

Так же, как и при сравнении чисел:

<?php

if (3 === 3) {

echo '+'; // сработает это

} else {

echo '-';

}

?>

Разница между двумя операторами проявляется именно тогда, когда значения одинаковые, но разный тип данных. В остальных случаях эти операторы работают одинаково. К примеру, при сравнении разных чисел, конечно же, выведется минус:

<?php

if (2 === 3) {

echo '+';

} else {

echo '-'; // сработает это

}

?>

**Неравенство по значению и типу**

Кроме оператора != существует также оператор !==, учитывающий тип при сравнении. Давайте рассмотрим различия между ними на примерах.

Пусть с помощью оператора != сравниваются два числа 3. Данный оператор сравнивает значения на то, что они НЕ равны. Так как наши значения как раз-таки равны, то на экран выведется '-':

<?php

if (3 != 3) {

echo '+';

} else {

echo '-'; // сработает это, так как значения равны

}

?>

Пусть теперь одно из наших значений будет в кавычках. В этом случае оператор != все равно посчитает их равными (так как совпадает значение, а тип не важен для данного оператора) и опять выведет '-':

<?php

if ('3' != 3) {

echo '+';

} else {

echo '-'; // сработает это, так как значения равны

}

?>

Давайте теперь сравним два числа 3 с помощью оператора !==. Он также посчитает их равными и выведет '-':

<?php

if (3 !== 3) {

echo '+';

} else {

echo '-'; // сработает это, так как значения равны

}

?>

А вот если теперь одну из троек взять в кавычки, то оператор !== посчитает наши тройки неравными, так как, хотя значения их и совпадают, но у них разный тип:

<?php

if ('3' !== 3) {

echo '+'; // сработает это, так как значения НЕ равны

} else {

echo '-';

}

?>

Нюансы

Обратите внимание на следующий код:

<?php

if (0 == '') {

echo '+'; // сработает это

} else {

echo '-';

}

?>

И на следующий:

<?php

if (0 == '0') {

echo '+'; // сработает это

} else {

echo '-';

}

?>

Какое сравнение использовать?

В настоящее время в PHP общепринято везде стараться использовать строгое сравнение. Считается, что в этом случае код менее подвержен ошибкам.

Сложные условия в if-else в PHP

**Сложные условия в if-else в PHP**

Иногда может быть нужно составить какое-то сложное условие. Для этого существуют операторы and (логическое И) и or (логическое ИЛИ).

**Логическое И**

Логическое И позволяет задать одновременность условий. В следующем примере условие выполнится, если переменная $num больше нуля и *одновременно* меньше 10:

<?php

$num = 3;

if ($num > 0 and $num < 10) {

echo '+';

} else {

echo '-';

}

?>

Условия могут налагаться не на одну переменную, а на разные. В следующем примере условие выполнится, если переменная $num1 равна 2 и одновременно переменная $num2 равна 3:

<?php

$num1 = 2;

$num2 = 3;

if ($num1 == 2 and $num2 == 3) {

echo '+';

} else {

echo '-';

}

?>

**Задача №8**

Проверьте то, что переменная $num больше нуля и меньше 5.

**Задача №9**

Проверьте то, что переменная $num больше или равна 10 и меньше или равна 20.

**Задача №10**

Проверьте то, что переменная $num1 равна или меньше 1, а переменная $num2 больше или равна 3.

**Логическое ИЛИ**

Логическое ИЛИ требует выполнения хотя бы одного условия. В следующем примере условие выполнится, если переменная $num1 больше 0 или переменная $num2 больше 0 (достаточно выполнения хотя бы одного из условий):

<?php

$num1 = 10;

$num2 = -5;

if ($num1 > 0 or $num2 > 0) {

echo '+'; // сработает это

} else {

echo '-';

}

?>

**Приоритет операций сравнения**

Операция and имеет приоритет над or. В следующем примере условие сработает, если переменная $num от 0 до 5 ИЛИ от 10 до 20:

<?php

$num = 3;

if ($num > 0 and $num < 5 or $num > 10 and $num < 20) {

echo '+';

} else {

echo '-';

}

?>

**Группировка условий**

Хотя операция and и имеет приоритет над or, часто удобнее использовать группирующие круглые скобки, чтобы явно показать приоритет операций:

<?php

$num = 3;

if ( ($num > 0 and $num < 5) or ($num > 10 and $num < 20) ) {

echo '+';

} else {

echo '-';

}

?>

Конечно же, группировка может быть использована и в том случае, когда вам нужен свой приоритет операций, а не тот, который получается по умолчанию.

**Инвертирование высказываний в if-else**

Рассмотрим следующий код:

<?php

if ($num > 0 and $num < 5) {

echo '+';

} else {

echo '-';

}

?>

Давайте инвертируем условие из приведенного кода, то есть превратим его в его противоположность. Противоположное условие получится таким:

<?php

if ($num <= 0 or $num >= 5) {

echo '+';

} else {

echo '-';

}

?>

Как вы видите, чтобы инвертировать условие, приходится несколько задуматься. Гораздо проще будет использовать оператор !, представляющий собой логическое НЕ. С помощью этого оператора нам достаточно поставить восклицательный знак перед изначальным условием - и оно инвертируется само:

<?php

if ( !($num > 0 and $num < 5) ) {

echo '+';

} else {

echo '-';

}

?>

**Задача №11**

Дан следующий код:

<?php

if ($num1 >= 0 or $num2 <= 10) {

echo '+';

} else {

echo '-';

}

?>

Используя оператор ! инвертируйте приведенное условие.

**Конструкция if-else и булевы значения**

Пусть у нас есть какая-то булева переменная, которая может принимать значения true или false:

<?php

$test = true;

?>

Давайте напишем if, проверяющий нашу переменную на значение true:

<?php

$test = true;

if ($test === true) {

echo '+';

} else {

echo '-';

}

?>

**Задача №12**

Проверьте то, что переменная $test равна true.

**Задача №13**

Проверьте то, что переменная $test равна false.

**Логические значения и двойное равно**

В предыдущем примере я использовал для сравнения оператор ===. В этом случае наша переменная сравнивалась на равенство true как по значению, так и по типу.

В нашей задаче можно использовать и оператор ==. Если в переменной test всегда будет одно из значений true или false, то ничего и не изменится:

<?php

$test = true; // тут пишем или true, или false

if ($test == true) {

echo '+';

} else {

echo '-';

}

?>

А вот если в переменную $test могут попадать любые значения, то все становится намного сложнее. В таком случае, если в переменной будет не булево значение, то это значение вначале будет преобразовано к булеву и только затем будет сравниваться.

Пусть, например, в нашей переменной будет число 1. В этом случае оно вначале преобразуется к логическому типу, то есть к true. И затем будет выполнено сравнение:

<?php

$test = 1;

if ($test == true) {

echo '+'; // сработает это, тк 1 == true - это верно

} else {

echo '-';

}

?>

А вот, к примеру, число 0 преобразуется к false. И наше условие в результате будет ложным:

<?php

$test = 0;

if ($test == true) {

echo '+';

} else {

echo '-'; // сработает это, тк 0 == true - это НЕверно

}

?>

**Сокращенный иф**

Пусть, к примеру, мы хотим узнать, равна ли переменная $test значению true. В этом случае конструкцию if можно записать так:

<?php

$test = true;

if ($test == true) {

echo '+';

} else {

echo '-';

}

?>

При программировании такие проверки требуются очень часто, поэтому для них существует более изящная сокращенная форма: вместо if ($test == true) можно написать просто if ($test).

Давайте перепишем наш код в сокращенной форме:

<?php

$test = true;

if ($test) { // эквивалентно if ($test == true)

echo '+';

} else {

echo '-';

}

?>

Пусть теперь мы проверяем на то, что переменная $test не равна true:

<?php

$test = true;

if ($test != true) {

echo '+';

} else {

echo '-';

}

?>

В этом случае сокращенный синтаксис будет выглядеть так:

<?php

$test = true;

if (!$test) { // используем логическое НЕ

echo '+';

} else {

echo '-';

}

?>

Аналогичное сокращение существует и для проверки на false. Пусть дан вот такой код:

<?php

$test = true;

if ($test == false) {

echo '+';

} else {

echo '-';

}

?>

Условие $test == false на самом деле все равно, что $test != true:

<?php

$test = true;

if ($test != true) { // эквивалентно if ($test == false)

echo '+';

} else {

echo '-';

}

?>

Ну, а такое условие мы уже научились сокращать в предыдущем примере. Сократим:

<?php

$test = true;

if (!$test) {

echo '+';

} else {

echo '-';

}

?>

**Задача №15**

Перепишите следующий код с использованием сокращенного сравнения:

<?php

$test = true;

if ($test == true) {

echo '+';

} else {

echo '-';

}

?>

**Нюанс**

Учтите, что в сокращенных операциях сравнение идет по ==, а не по ===.

**Задача №16**

Дано число 15. Если оно больше 10, то увеличьте его на 100, иначе уменьшите на 30.

**Задача №17**

Дано натуральное число 8. Если оно четное, то уменьшите его в 2 раза, иначе

увеличьте в 3 раза.

**Задача №18**

Дано число. Если оно не меньше 50, то выведите квадрат этого числа, если же это число больше 10 и меньше 30, то выведите ноль, в остальных случаях выведите слово "Ошибка"

**Задача №19**

Дано два числа $a = 15, $b = 4. Вывести наибольшее из них. Вывести 'Да', если они отличаются на 100, иначе вывести

**Задача №20**

Дано два натуральных числа. Вывести 'Да', если они отличаются не более чемна 20, иначе вывести 'Нет'.

**Задача №21**

В переменной $min лежит число от 0 до 59. Определите в какую четверть часа попадает это число (в первую, вторую, третью или четвертую).

**Задача №22**

В переменной $day лежит какое-то число из интервала от 1 до 31. Определите в какую декаду месяца попадает это число (в первую, вторую или третью).

**Задача №23**

В переменной $year хранится год. Определите, является ли он високосным (в таком году есть 29 февраля). Год будет високосным в двух случаях: либо он делится на 4, но при этом не делится на 100, либо делится на 400. Так, годы 1700, 1800 и 1900 не являются високосными, так как они делятся на 100 и не делятся на 400. Годы 1600 и 2000 - високосные, так как они делятся на 400.

**Задача №24**

Дана строка с символами, например, 'abcde'. Проверьте, что первым символом этой строки является буква 'a'. Если это так - выведите 'да', в противном случае выведите 'нет'.

**Задача №25**

Дана строка с цифрами, например, '12345'. Проверьте, что первым символом этой строки является цифра 1, 2 или 3. Если это так - выведите 'да', в противном случае выведите 'нет'.

**Задача №26**

Дана строка из 3-х цифр. Найдите сумму этих цифр. То есть сложите как числа первый символ строки, второй и третий.

**Задача №27**

Дана строка из 6-ти цифр. Проверьте, что сумма первых трех цифр равняется сумме вторых трех цифр. Если это так - выведите 'да', в противном случае выведите 'нет'.