**Лабораторная работа №19**

**Пользовательские функции в PHP**

Сейчас мы с вами научимся создавать свои функции, которые затем можно будет использовать подобно стандартным функциям PHP.

Давайте рассмотрим синтаксис для создания своей функции.

Функция создается с помощью команды function. Далее через пробел следует *имя функции* и *круглые скобки*, в которых пишется какой-то код:

<?php

function func() {

// какой-то код

}

?>

Давайте посмотрим на каком-нибудь примере.

Сделаем функцию с именем func, которая при своем вызове будет выводить на экран восклицательный знак:

<?php

function func() {

echo '!';

}

?>

Давайте теперь вызовем нашу функцию. Для этого необходимо написать ее имя и круглые скобки:

<?php

function func() {

echo '!';

}

// Вызываем нашу функцию:

func(); // выведет '!'

?>

Можно вызвать нашу функцию несколько раз - в этом случае каждый вызов функции будет совершать новый вывод на экран:

<?php

function func() {

echo '!';

}

func(); // выведет '!'

func(); // выведет '!'

func(); // выведет '!'

?>

Функции могут быть вызваны до места своего определения:

<?php

func(); // выведет '!'

function func() {

echo '!';

}

?>

**Задача №1**

Сделайте функцию, выводящую на экран ваше имя.

**Задача №2**

Сделайте функцию, выводящую на экран сумму чисел от 1 до 100.

**Параметры функции в PHP**

При вызове функции мы пишем ее имя и пустые круглые скобки:

<?php

func();

?>

На самом деле скобки не обязательно должны быть пустыми - в них мы можем передавать параметры.

Пусть, к примеру, мы хотим сделать функцию, которая параметром будет принимать число и выводить на экран квадрат этого числа.

Как мы теперь будем вызывать нашу функцию: мы будем писать имя функции, круглые скобки, а в них - какое-то число, квадрат которого мы хотим получить.

К примеру, вот так мы получим квадрат числа 2:

<?php

func(2); // выведет 4

?>

А вот так - квадрат числа 3:

<?php

func(3); // выведет 9

?>

Давайте теперь напишем реализацию нашей функции.

Мы знаем, что функция параметром должна принимать число. Значит, при определении функции в круглых скобках мы должны написать какую-то переменную, в которую будет попадать переданное число.

Название переменной может быть любым, давайте, например, назовем ее $num:

<?php

function func($num) {

}

?>

В эту переменную $num при вызове функции будет попадать указанное в круглых скобках число:

<?php

func(2); // в переменную $num попадет число 2

func(3); // в переменную $num попадет число 3

?>

Давайте теперь сделаем так, чтобы наша функция выводила квадрат переданного числа. Для этого перемножим переменную $num саму на себя и выведем на экран:

<?php

function func($num) {

echo $num \* $num;

}

?>

Проверим работу функции, вызвав ее с различными числами:

<?php

function func($num) {

echo $num \* $num;

}

func(2); // выведет 4

func(3); // выведет 9

?>

**Задача №3**

Сделайте функцию, которая параметром принимает число и выводит на экран куб этого числа.

**Задача №4**

Сделайте функцию, которая параметром принимает число и проверяет, положительное это число или отрицательное. В первом случае пусть функция выводит на экран текст '+++', а во втором '---'.

## Несколько параметров

Функция может принимать не один параметр, а несколько. В этом случае их нужно перечислить через запятую. Давайте для примера сделаем функцию, параметром принимающую два числа и выводящую на экран их сумму:

<?php

function func($num1, $num2) {

echo $num1 + $num2;

}

func(1, 2); // выведет 3

?>

**Задача №5**

Сделайте функцию, которая параметрами принимает 3 числа и выводит на экран сумму этих чисел.

## Параметры-переменные

Давайте рассмотрим следующий код:

<?php

function func($num) {

echo $num \* $num;

}

func(2);

?>

Как вы видите, в при вызове функции в нее передается число 2.

Не обязательно параметром функции передавать именно число - можно передать и переменную, содержащую нужное нам число:

<?php

function func($num) {

echo $num \* $num;

}

$param = 2;

func($param);

?>

**Задача №6**

Сделайте функцию func, которая параметром будет принимать 3 числа и выводить на экран их сумму. С помощью этой функции выведите на экран сумму значений следующих переменных:

<?php

$param1 = 1;

$param2 = 2;

$param3 = 3;

?>

## Необязательные параметры

Пусть у нас есть функция, которая параметром принимает имя и выводит его на экран:

<?php

function func($name) {

echo 'ваше имя: ' . $name;

}

?>

Проверим работу нашей функции:

<?php

func('Вася'); // выведет 'ваше имя: Вася'

?>

Давайте теперь сделаем так, чтобы имя было не обязательным: если оно будет введено, то выведется на экран, а если не введено - то пусть принимает значение 'Аноним'.

Для того, чтобы сделать параметр необязательным, для него нужно определить *значение по умолчанию*. Для этого при определении параметра нужно поставить знак = и желаемое значение, вот так:

<?php

function func($name = 'Аноним') {

echo 'ваше имя: ' . $name;

}

?>

Проверим теперь работу нашей функции:

<?php

// Без параметра:

func(); // выведет 'ваше имя: Аноним'

// С параметром:

func('Вася'); // выведет 'ваше имя: Вася'

?>

Пусть теперь наша функция принимает два параметра: имя и возраст:

<?php

function func($name, $age) {

echo 'ваше имя: ' . $name . ' возраст: ' . $age;

}

?>

Давайте сделаем так, чтобы возраст был необязателен:

<?php

function func($name, $age = 'неизвестен') {

echo 'ваше имя: ' . $name . ' возраст: ' . $age;

}

?>

А теперь сделаем так, чтобы и имя, и возраст были необязательны:

<?php

function func($name = 'Аноним', $age = 'неизвестен') {

echo 'ваше имя: ' . $name . ' возраст: ' . $age;

}

?>

А вот сделать так, чтобы имя было необязательным, а возраст - нет, нельзя. Дело в том, что все необязательные параметры следует писать в конце. Смотрите пример:

<?php

// так не будет работать

function func($name = 'Аноним', $age) {

echo 'ваше имя: ' . $name . ' возраст: ' . $age;

}

?>

**Функция return в PHP**

Пусть у нас есть функция, которая выводит на экран квадрат переданного числа:

<?php

function func($num) {

echo $num \* $num;

}

func(3); // выведет 9

?>

Пусть мы хотим не выводить значение на экран, а записать в какую-нибудь переменную, вот так:

<?php

$result = func(3); // в переменной $result теперь 9

?>

Для этого в PHP существует специальная инструкция return, которая позволяет указать значение, которое возвращает функция. Под словом *возвращает* понимают то значение, которое запишется в переменную, если ей присвоить вызванную функцию.

Итак, давайте перепишем нашу функцию так, чтобы она не выводила результат на экран, а возвращала его в переменную:

<?php

function func($num) {

return $num \* $num;

}

$result = func(3); // в переменной $result теперь 9

?>

После того, как данные записаны в переменную, их можно, например, вывести на экран:

<?php

function func($num) {

return $num \* $num;

}

$result = func(3);

echo $result; // выведет 9

?>

А можно сначала как-то изменить эти данные, а затем вывести их на экран:

<?php

function func($num) {

return $num \* $num;

}

$result = func(3);

$result = $result + 1;

echo $result; // выведет 10

?>

Можно сразу выполнять какие-то действия с результатом работы функции перед записью в переменную:

<?php

function func($num) {

return $num \* $num;

}

$result = func(3) + 1;

echo $result; // выведет 10

?>

А можно не записывать результат в переменную, а сразу вывести его на экран:

<?php

function func($num) {

return $num \* $num;

}

echo func(3); // выведет 9

?>

**Задача №7**

Сделайте функцию, которая параметром принимает число, а возвращает куб этого числа. С помощью этой функции найдите куб числа 3 и запишите его в переменную $result.

## Использование функций в выражении

В следующем примере с помощью функции func мы сначала найдем квадрат числа 2, а затем - квадрат числа 3, сложим эти значения и запишем в переменную:

<?php

function func($num) {

return $num \* $num;

}

$result = func(2) + func(3);

echo $result; // выведет 13

?>

**Задача №8**

Сделайте функцию, которая параметром принимает число, а возвращает квадратный корень из этого числа. С помощью этой функции найдите корень числа 3, затем найдите корень числа 4. Просуммируйте полученные результаты и выведите их на экран.

## Функции в функциях

Можно также результат работы одной функции передать параметром в другую, например, вот так мы сначала найдем квадрат числа 2, а затем квадрат результата:

<?php

function func($num) {

return $num \* $num;

}

$result = func(func(2));

echo $result; // выведет 16

?>

Функции, конечно же, не обязательно должны быть одинаковыми.

Пусть, например, у нас есть функция, возвращающая квадрат числа, и функция, возвращающая куб числа:

<?php

function square($num) {

return $num \* $num;

}

function cube($num) {

return $num \* $num \* $num;

}

?>

Давайте с помощью этих функций возведем число 2 в квадрат, а затем результат этой операции возведем в куб:

<?php

$result = cube(square(2));

echo $result;

?>

Пусть теперь у нас есть функция, возвращающая квадрат числа, и функция, находящая сумму двух чисел:

<?php

function square($num) {

return $num \* $num;

}

function sum($num1, $num2) {

return $num1 + $num2;

}

?>

Найдем с помощью этих функций сумму квадрата числа 2 и сумму квадрата числа 3:

<?php

$result = sum(square(2), square(3));

echo $result;

?>

## Тонкое место return

После того, как выполнится инструкция return - функция закончит свою работу. То есть: *после выполнения return больше никакой код не выполнится.*

Смотрите пример:

<?php

function func($num) {

return $num \* $num;

echo '!'; // этот код никогда не выполнится

}

$result = func(3);

?>

Это не значит, что в функции должен быть один return. Но выполнится только один из них.

В следующем примере в зависимости от значения параметра выполнится либо первый, либо второй return:

<?php

function func($num) {

if ($num >= 0) {

return '+';

} else {

return '-';

}

}

echo func( 3); // выведет '+'

echo func(-3); // выведет '-'

?>

## Применение return в цикле

То, что return расположен внутри цикла, не всегда может быть ошибкой.

В следующем примере сделана функция, которая определяет, сколько первых элементов массива нужно сложить, чтобы сумма стала больше или равна 10:

<?php

function func($arr) {

$length = count($arr);

$sum = 0;

for ($i = 0; $i < $length; $i++) {

$sum += $arr[$i];

// Если сумма больше или равна 10:

if ($sum >= 10) {

return $i + 1; // выходим из цикла и из функции

}

}

}

$result = func([1, 2, 3, 4, 5]);

echo $result;

?>

А в следующем примере сделана функция, которая вычисляет, сколько целых чисел, начиная с 1, нужно сложить, чтобы результат был больше 100:

<?php

function func() {

$sum = 0;

$i = 1;

while (true) { // бесконечный цикл

$sum += $i;

if ($sum >= 100) {

return $i; // цикл крутится пока не выйдет тут

}

$i++;

}

}

echo func( );

?>

**Задача №9**

Напишите функцию, которая параметром будет принимать число и делить его на 2 столько раз, пока результат не станет меньше 10. Пусть функция возвращает количество итераций, которое потребовалось для достижения результата.

## Приемы работы с return

Существуют некоторые приемы работы с return, упрощающие код.

Рассмотрим, к примеру, следующий код:

<?php

function func($num) {

if ($num >= 0) {

$result = sqrt($num);

} else {

$result = 0;

}

return $result;

}

echo func(3);

?>

Как вы видите, в данном коде в зависимости от условия в переменную $result будет попадать либо одно, либо другое значение. А последней строкой функции содержимое этой переменной возвращается через return.

Давайте перепишем этот код в более сокращенной форме, избавившись от ненужной здесь переменной $result:

<?php

function func($num) {

if ($num >= 0) {

return sqrt($num);

} else {

return 0;

}

}

echo func(3);

?>

**Задача №10**

Дана следующая функция:

<?php

function func($num1, $num2) {

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

$result = $num1 \* $num2;

} else {

$result = $num1 - $num2;

}

return $result;

}

echo func(3, 4);

?>

Перепишите ее в сокращенной форме согласно изученной теории.

**Практические задания по созданию функций**

Далее мы с вами будем отрабатывать полученные ранее знания по пользовательским функциям на практических задачах.

Пусть у нас дан какой-то произвольный массив с числами, например, такой:

<?php

$arr = [1, 2, 3, 4];

?>

Пусть теперь перед нами, к примеру, стоит задача найти сумму квадратов элементов этого массива. Пока не будем использовать пользовательские функции, чтобы показать, какие проблемы возникнут в этом случае.

Давайте напишем код, реализующий нашу задачу:

<?php

$arr = [1, 2, 3, 4];

// Код, находящий сумму:

$sum = 0;

foreach ($arr as $elem) {

$sum += $elem \* $elem;

}

echo $sum;

?>

Пусть теперь у нас даны два массива:

<?php

$arr1 = [1, 2, 3, 4];

$arr2 = [5, 6, 7, 8];

?>

Пусть теперь перед нами стоит задача найти сумму квадратов элементов и одного массива, и второго. В этом случае нам придется код, находящий сумму, повторить два раза: для первого и для второго массива, вот так:

<?php

$arr1 = [1, 2, 3, 4];

$arr2 = [5, 6, 7, 8];

// Код, находящий сумму для первого массива:

$sum1 = 0;

foreach ($arr1 as $elem) {

$sum1 += $elem;

}

// Код, находящий сумму для второго массива:

$sum2 = 0;

foreach ($arr2 as $elem) {

$sum2 += $elem;

}

// Код, выводящий найденные суммы:

echo $sum1;

echo $sum2;

?>

Как мы видим, у нас возникает дублирование кода. Давайте избавимся от него, создав свою функцию, находящую сумму элементов переданного параметром массива.

Вот код этой функции:

<?php

function getSquareSum($arr) {

$sum = 0;

foreach ($arr as $elem) {

$sum += $elem;

}

return $sum;

}

?>

Используем теперь нашу функцию для нахождения суммы одного и второго массива:

<?php

function getSquareSum($arr) {

$sum = 0;

foreach ($arr as $elem) {

$sum += $elem;

}

return $sum;

}

$arr1 = [1, 2, 3, 4];

$arr2 = [5, 6, 7, 8];

echo getSquareSum($arr1);

echo getSquareSum($arr2);

?>

**Задача №11**

Напишите функцию, которая будет находить сумму квадратных корней элементов массива.

**Задача №12**

Реализуйте функцию getDivisors, которая параметром будет принимать число и возвращать массив его делителей, то есть целых чисел, на которое делится наше число.