1.6. Работа с функциями

В программировании, как и в математике, **функция** есть отображение множества ее аргументов на множество ее значений. То есть функция для каждого набора значений аргумента **возвращает** какие-то значения, являющиеся **результатом** ее работы.

Как и в любом другом языке программирования, в PHP различают функции, определяемые пользователем (в данном случае программистом), и функции, которые заранее определены в языке программирования. Рассмотрим их по очереди.

Функции, определяемые пользователем, это законченные части программ, которые решают определенную задачу. Пусть, к примеру, необходимо подсчитать факториал какого-либо числа. Факториал числа есть произведение всех чисел от единицы до этого числа. Если брать в качестве числа небольшое, однозначное число, например 4, то задача будет очень простой: 4! = 1*2*3*4 = 24. Но если взять число побольше, например 10, то выписывать все умножения уже несколько утомительно. Функции как раз и предназначены для решения типовой задачи, при этом параметры функции можно менять. Пример алгоритма вычисления факториала числа на PHP приведен ниже.

```
<?php
function fact($n){
  if ($n==0) return 1;
  else return $fact = $n * fact($n-1);
}
echo fact(3);
   // можно было бы написать echo (3*2);
   // но если число большое,
echo fact(50);
   // то удобнее пользоваться функцией,
   // чем писать echo (50*49*48*...*3*2);
?>
```

Описание функции начинается с служебного слова function. В PHP не важно, где именно описывается функция – до или после вызова. После слова function идет название функции, которое должно быть уникальным и содержать символы латин-

ского алфавита. Название не должно начинаться с цифр, служебных символов, пробелов. В имени функции допускается использование нижнего подчеркивания. Если функция содержит входные параметры, то они должны быть определены сразу после имени функции. Внутри функции описываются действия. Если функция должна возвращать какое-то значение, то в теле функции должен быть оператор return, выполняющий эту работу. Пример общего описания функции представлен ниже.

```
function Имя_функции (параметр1, параметр2, ..., параметрN){
    Блок_действий
    return "значение возвращаемое функцией";
}
```

Для того чтобы функция выполнила поставленную ей задачу, ее необходимо вызвать. Вызов функции может осуществляться как до, так и после ее описания. Единственное ограничение в этом случае – описание функций внутри условных конструкций. В случае, когда функция определяется в зависимости от какого-либо условия, например как это показано в двух приведённых ниже примерах, обработка описания функции должна **предшествовать** её вызову (подробнее можно изучить на ресурсе [18]). Вызов функции осуществляется указанием ее имени и значений параметров:

```
<?php
Имя_функции("значение_параметра1","значение_параметра2",...);
?>
```

Аргументы функции бывают двух видов: аргументы-значения и аргументы-переменные.

Аргументы-значения не меняют своего значения в процессе выполнения функции. В этом случае при вызове функции на их месте могут находиться конкретные значения, которые будут использованы в функции. Когда аргумент передается в функцию по значению, изменение значения аргумента внутри функции не влияет на его значение вне функции.

Чтобы позволить функции изменять ее аргументы, их нужно передавать по ссылке (аргументы-переменные). Для этого в определении функции перед именем аргумента следует написать знак амперсанд «&».

```
<?php
//напишем функцию, которая бы добавляла к строке слово se-
lected
function addOption(&$option){
     $option .= "selected";
}
//выводит элемент формы список
$str = "<select>";
echo $str;
// пусть имеется такая строка
$option = "<option ";
echo $option .">Красный</option>";
// выведет "красный", но он не будет выделен
addOption($option);
// вызовем функцию
echo $option .">Синий</option>";
//это выведет в списке выделенный "синий"
$str = "</select>";
echo $str;
?>
```

При работе с функциями в PHP есть замечательная особенность – задавать аргументы функции, используемые по умолчанию. В тех случаях, когда в других языках программирования необходимо дополнительно делать проверки условий существования переданных значений, в PHP можно задавать значения параметров непосредственно в заголовке функции. Само значение по умолчанию должно быть константным выражением, а не переменной и не представителем класса или вызовом другой функции.

Ниже представлена функция, создающая информационное сообщение, подпись к которому меняется в зависимости от значения переданного ей параметра. Если значение параметра не задано, то используется подпись "ЮУрГГПУ".

```
<?php
function message($sign="ЮУрГГПУ."){
// здесь параметр sign имеет по умолчанию значение "ЮУрГГПУ"
echo "Вы приглашены на семинар по веб-технологиям.<br>";
echo "$sign<br>";
}
```

```
message();
//вызываем функцию без параметра.
//В этом случае подпись - это ЮУрГГПУ.
message("С уважением, Иванов В.П.");
// В этом случае подпись "С уважением, Иванов В.П."
?>
```

Если у функции несколько параметров, то те аргументы, для которых задаются значения по умолчанию, должны быть записаны после всех остальных аргументов в определении функции. В противном случае появится ошибка, если эти аргументы будут опущены при вызове функции. Код функции А

```
<?php
function addArticle($title, $description, $author="Иванов Иван"){
  echo "Заносим в каталог статью: $title,";
  echo "автор $author";
  echo "<br>Краткое описание: ";
  echo "$description <hr>";
}
addArticle("Информатика и мы", "Это статья про информатику...", "Петров Петр");
addArticle("Кто такие продакт-менеджеры", "Это статья про менеджеров проектов...");
?>
```

Пример **неправильной** работы с параметрами функции (значение параметра по умолчанию находится перед остальными параметрами):

```
<?php
function addArticle($author="Иванов Иван", $title, $description){
// ...действия как в предыдущем примере
}
addArticle("Кто такие продакт-менеджеры", "Это статья про менеджеров проектов...");
?>
```

Рассмотрим работу с областью видимости переменных, используемых как в основной программе, так и в области функций. К таким областям видимости относят глобальные и статичные переменные.

Чтобы использовать внутри функции переменные, заданные вне ее, эти переменные нужно объявить, как **глобальные**. Для этого в теле функции следует перечислить их имена после ключевого слова **global.**

Пример использования глобальных переменных

```
<?
$a=1;
function test_g(){
global $a;
  $a = $a*2;
  echo 'в результате работы функции $a=', $a;
}
echo 'вне функции $a=', $a, ', ';
test_g();
echo "<br>";
echo 'вне функции $a=', $a, ', ';
test_g();
?>
```

В результате работы этого примера будет выведено:

```
вне функции $a=1, в результате работы функции $a=2
вне функции $a=2, в результате работы функции $a=4
```

Статические переменные видны только внутри функции и не теряют своего значения, если выполнение программы выходит за пределы функции. Объявление таких переменных производится с помощью ключевого слова **static.**

Пример работы со статическими переменными

```
<?
function test_s(){
static $a = 1;
// нельзя присваивать выражение или ссылку
  $a = $a*2;
  echo $a;
}
test_s(); // выведет 2
echo $a;  // ничего не выведет, так как $a доступна
```

```
            //только внутри функции
test_s(); // внутри функции $a=2, поэтому
          // результатом работы функции
          // будет число 4
?>
```

Если функция в результате работы должна возвращать один результат, то используют оператор **return**. Отметим, что return может возвращать как одно значение, так и несколько значений в виде массива. Тем не менее считается, что функция возвращает один результат – массив.

Рассмотрим пример возвращения функцией количества дней, прошедших между двумя датами. Причем неважно, какая из дат наступила раньше.

```
<?php
function daysBetween($start_data, $end_data)
{
return
abs(strtotime($start_data)-strtotime($end_data))/(60*60*24);
}
$how_days = daysBetween("14.03.2021", "17.03.2021");
echo "Прошло $how_days дней";
?>
```

Здесь мы сталкиваемся с функцией, которая уже определена в php – **strtotime**. Ее задача – перевести строковое значение даты в количество секунд, прошедших с 1 января 1970 года 00:00:00 UTC (так называемое начало эпохи UNIX). Математическая функция **abs** возвращает модуль числа, в этой задаче она служит для вычисления абсолютной величины между двумя датами.

Рассмотрим также и другие популярные функции, уже определенные в PHP. Все функции разбиты по категориям – для работы с массивами, для работы со строками, для работы с датой/временем, для работы с mysql и другие. Большинство функций в PHP имеет особую приставку (префикс) в названии, с помощью которой разработчику легко определить ее применимость и назначение.

Кратко рассмотрим функции для работы с массивами, строками и ряд других, которые будут использованы на лабораторных занятиях в рамках данного курса. Про все неописанные функции можно узнать из источников [18; 20].

Многие функции для работы с массивами имеют префикс «array_». Самыми популярными функциями являются:

- count($array) – возвращает количество элементов массива.
- in_array($elem, $array) – проверяет, содержится ли элемент $elem в массиве $array.
- array_search($elem, $array) – проверяет, содержится ли элемент $elem в массиве $array, и возвращает индекс найденного элемента.
- array_values($array) – возвращает новый массив, содержащий все значения переданного на вход массива.
- array_keys($array) – возвращает новый массив, содержащий все индексы в качестве значений переданного на вход массива.
- sort($array, [$flag]) – сортирует массив $array в зависимости от необязательного параметра $flag. Если значение $flag задать SORT_REGULAR, то произойдет обычное сравнение элементов; если SORT_NUMERIC, то числовое сравнение элементов; если SORT_STRING, то строковое сравнение элементов.
- rsort($array, [$flag]) – выполняет сортировку массива в обратном порядке.
- asort($array, [$flag]) – простая сортировка с сохранением ассоциаций ключей.
- arsort($array, [$flag]) – обратная сортировка с сохранением ассоциаций ключей.
- ksort($array, [$flag]) – сортировка массива по ключам.
- krsort($array, [$flag]) – обратная сортировка массива по ключам.

Функции для работы со строками часто в своем названии имеют приставку str.

- str_replace($search, $replace, $subject, [$count]) – заменяет все вхождения строки поиска на строку замены. Эта функция возвращает строку или массив, в котором все вхождения $search в $subject заменены на $replace. При этом $search и $replace могут быть как строками, так и массивами строк. Результат функции также может быть строкой или массивом. Необязательный параметр $count может быть использован для возвращения количества произведённых замен.
- substr($string, $offset, [$length]) – возвращает подстроку строки string, которая начинается с номера символа $offset и имеет длину $length символов.

Если $length положительный, возвращаемая строка будет не длиннее $length-символов, начиная с параметра $offset (в зависимости от длины $string). Если $length отрицательный, то будет отброшено указанное этим аргументом число символов с конца строки string (после того как будет вычислена стартовая позиция, если $offset отрицателен). Если при этом позиция начала подстроки, определяемая аргументом $offset, находится в отброшенной части строки или за ней, возвращается false.

- strlen($string) – возвращает длину строки (количество символов в строке).
- explode($separator, $string, [$limit]) – возвращает массив строк, полученных разбиением строки $string с использованием $separator в качестве разделителя. Если аргумент $limit является положительным, возвращаемый массив будет содержать максимум $limit элементов при этом последний элемент будет содержать остаток строки $string. Если параметр $limit отрицателен, то будут возвращены все компоненты, кроме последних $limit. Если $limit равен нулю, то он расценивается как 1.
- implode($separator, $array) – объединяет элементы массива $array в строку с помощью разделителя $separator. В результате возвращает строку.
- strpos($haystack, $needle, [$offset]) – ищет позицию первого вхождения подстроки $needle в строку $haystack. Если $offset указан, то поиск будет начат с указанного количества символов с начала строки. Если задано отрицательное значение, отсчёт позиции начала поиска будет произведён с конца строки.
- addslashes($string) – возвращает строку с обратным слешем перед символами, которые нужно экранировать. Экранируются следующие символы: одинарная кавычка (’); двойная кавычка (”); обратный слеш (\); NUL (байт null).
- trim($string,[$charlist]) – удаляет пробелы из начала и конца строки. Помимо пробелов также может удалять непечатные символы: табуляцию «\t», перевод строки «\n», возврат каретки «\r», NUL-байт «\0», вертикальную табуляцию «\x0B». Можно перечислить удаляемые символы через строку $charlist.
- stripslashes($string) – удаляет экранирующие обратные слэши. (\’ преобразуется в ’, и т.д.). Двойные бэкслэши (\\) преобразуется в одиночные (\).
- htmlspecialchars($string, []) – преобразует специальные символы в HTML-сущности. '&' (амперсанд) преобразуется в '&'; '"' (двойная кавычка) преобразуется в '"'; "' (одиночная кавычка) преобразуется в '''

только в режиме ENT_QUOTES; '<' (знак «меньше, чем») преобразуется в '<'; '>' (знак «больше, чем») преобразуется в '>'.

- md5($string) – возвращает MD5-хеш строки в виде 32-символьного шестнадцатеричного числа. Алгоритм необратим, то есть хеш нельзя дешифровать. Функция используется для сравнения файлов, хранения паролей в закодированном виде и других криптографических задач.

Также в PHP есть функции для работы с переменными – проверка значений переменных

- isset($variable) – определяет, была ли установлена переменная $variable значением отличным от null.
- unset($variable) – удаляет переменную $variable.
- intval($variable) – функция преобразования типов, возвращает целое значение переменной $variable.
- floatval($variable) – функция преобразования типов, возвращает число с плавающей точкой по переменной $variable.

Математические функции

- abs($number) – возвращает абсолютное значение $number.
- round($num, [$precision], [$mode]) – возвращает округлённое значение $num с указанной точностью $precision (количество цифр после запятой). $precision может быть отрицательным или нулём (по умолчанию).
- ceil($num) – возвращает ближайшее большее целое от $num.
- mt_rand([$min], [$max]) – генерирует случайное значение методом с помощью генератора простых чисел. $min – необязательный параметр: минимальное значение случайного числа (по умолчанию: 0). $max – необязательный параметр: максимальное значение случайного числа

Функция для отправки почтового сообщения

- mail($to, $subject, $message, [$additional_headers], [$additional_params]) – отправляет электронную почту на адрес $to, с темой $subject и содержанием письма $message. Необязательный параметр $additional_headers – это строка или массив, которые будут вставлены в конец отправляемых заголовков письма. Параметр $additional_params может быть использован для передачи дополнительных флагов в виде аргументов командной строки для программы, сконфигурированной для отправки писем, указанной директивой sendmail_path. Например, можно установить отправителя письма при использовании sendmail с помощью опции -f.

Здесь приведена лишь малая часть функций, которые определены в PHP. В следующем параграфе мы отдельно рассмотрим функции для работы с файловой системой. Функции для работы с базами данных также рассмотрим в параграфе 2.9. Тем не менее даже такая небольшая часть функций может принести большую пользу разработчику, так как они решают наиболее частые задачи при программировании веб-проектов.