

## Практическая работа №3 «PHP: ветвления, функции»

**Цель работы:** познакомиться с реализацией бинарных и множественных ветвлений в языке PHP, а также работой с функциями.

### ЗАДАНИЯ

#### А. Подготовка к работе:

1. Запустите менеджер виртуальных машин VMware Player на **реальном** компьютере.
2. В программе VMware Player запустите образ **виртуального** компьютера **WWW** с предустановленным ПО: Apache (HTTP-сервер), PHP (интерпретатор) и MySQL (СУБД).
3. В адресной строке браузера с реального компьютера откройте поочередно ссылки <http://www/>, <http://www/info.php> и <http://www/sqltest.php>. Убедитесь в работоспособности перечисленных выше программных компонентов виртуального веб-сервера.
4. В проводнике на реальном компьютере введите адрес [\\www](http://www), укажите по запросу имя пользователя – Администратор; пароль – 1234. Отобразится список общих папок веб-сервера, в т.ч. папка www – **корневая папка сайта** (локальный путь – C:\wamp\www).

Если проводник сообщает, что нет доступа к ресурсу, то на реальном компьютере (для ОС Windows Vista / 7 / 8.x / 10) откройте «Панель управления» → «Сеть и Интернет» → «Центр управления сетями и общим доступом» → «Изменить дополнительные параметры общего доступа». В нужном профиле выберите «Включить сетевое обнаружение» и «Включить общий доступ, чтобы сетевые пользователи могли читать и записывать файлы в общих папках». Сохраните изменения и повторите попытку.

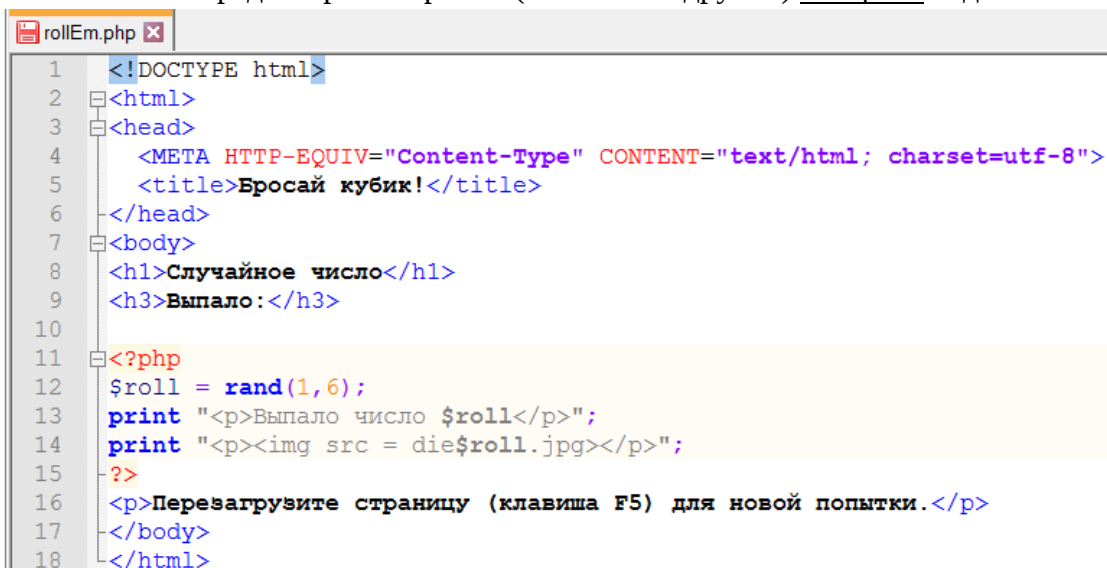
Возможной причиной блокирования также может быть пустой пароль у текущего пользователя реального компьютера.

*Примечание:* если запрещено изменение настроек на реальном компьютере, редактирование файлов сайта в дальнейшем возможно прямо на виртуальном компьютере.

#### Б. Случайные числа в PHP

Создадим программу на угадывание числа, в основе которой – генерация случайных чисел.

5. В текстовом редакторе Notepad++ (или любом другом) наберите код:



```

1 <!DOCTYPE html>
2 <html>
3 <head>
4 <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=utf-8">
5 <title>Бросай кубик!</title>
6 </head>
7 <body>
8 <h1>Случайное число</h1>
9 <h3>Выпало:</h3>
10
11 <?php
12 $roll = rand(1,6);
13 print "<p>Выпало число $roll</p>";
14 print "<p><img src = die$roll.jpg></p>";
15 ?>
16 <p>Перезагрузите страницу (клавиша F5) для новой попытки.</p>
17 </body>
18 </html>
  
```

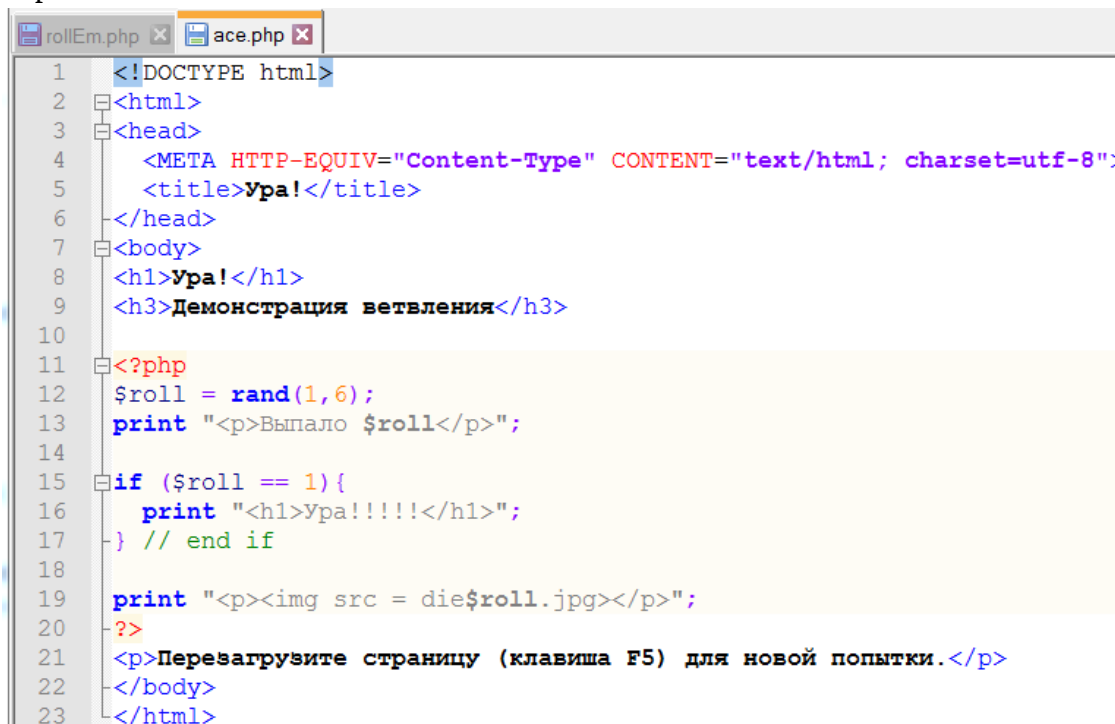
Сохраните код в файле rollEm.php. Скопируйте в папку сайта изображения шести сторон игрового кубика. Откройте программу в браузере, несколько раз обновите содержимое.

Познакомьтесь в справочнике с описанием функции `rand ()`. Какое наибольшее значение может вернуть эта функция?

Обратите внимание, как в скрипте формируется имя файла выводимой картинки.

## В. Бинарное ветвление в PHP

6. Скопируйте код предыдущего примера в новый файл и отредактируйте его следующим образом:



```

1 <!DOCTYPE html>
2 <html>
3 <head>
4 <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=utf-8">
5 <title>Ура!</title>
6 </head>
7 <body>
8 <h1>Ура!</h1>
9 <h3>Демонстрация ветвления</h3>
10
11 <?php
12 $roll = rand(1,6);
13 print "<p>Выпало $roll</p>";
14
15 if ($roll == 1){
16     print "<h1>Ура!!!!</h1>";
17 } // end if
18
19 print "<p><img src = die$roll.jpg></p>";
20 ?>
21 <p>Перезагрузите страницу (клавиша F5) для новой попытки.</p>
22 </body>
23 </html>

```

Сохраните файл под именем `ase.php`.

Обратите внимание на команду `if`. Это команда *бинарного ветвления* (условный оператор) в языке PHP. Условие размещено в простых скобках, а тело (блок) – в фигурных.

Блок команд (тело, код в фигурных скобках) будет выполнен только если условие окажется истинным (`true`). В нашем примере условие вывода фразы «Ура!!!!» – если значение переменной `$roll` равно 1.

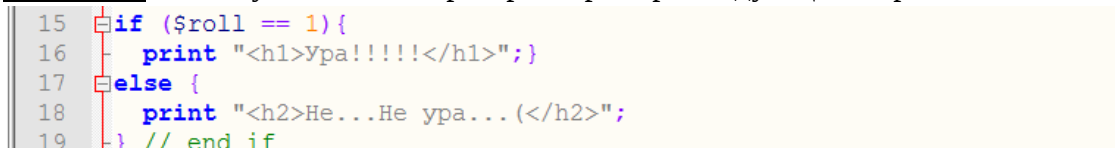
Точки с запятой отделяют команды только внутри тела условного оператора: ни до, ни после фигурной скобки они не ставятся.

В качестве условия может выступать значение логической переменной (типа `boolean`), а также результат, возвращаемый или логической функцией, или оператором сравнения (как в нашем примере).

Операторы простого сравнения (см. табл. 1) могут сравнивать значения даже различных типов (после преобразования к одному), операторы строгого сравнения – только одно-типные значения (в противном случае возвращается значение ложь – `false`).

Откройте документ в браузере. Несколько раз обновите содержимое, пока не выпадет 1.

Исправьте запись условного оператора в примере следующим образом:



```

15 if ($roll == 1){
16     print "<h1>Ура!!!!</h1>";
17 } else {
18     print "<h2>Не...Не ура...(</h2>";
19 } // end if

```

Таблица 1. Операторы сравнения.

| Пример                        | Название              | Результат   |
|-------------------------------|-----------------------|---|
| <code>\$a == \$b</code>       | Равно                 | <b>TRUE</b> если <code>\$a</code> равно <code>\$b</code> после преобразования типов.    |
| <code>\$a === \$b</code>      | Тождественно равно    | <b>TRUE</b> если <code>\$a</code> равно <code>\$b</code> и имеет тот же тип.            |
| <code>\$a != \$b</code>       | Не равно              | <b>TRUE</b> если <code>\$a</code> не равно <code>\$b</code> после преобразования типов. |
| <code>\$a &lt;&gt; \$b</code> | Не равно              | <b>TRUE</b> если <code>\$a</code> не равно <code>\$b</code> после преобразования типов. |
| <code>\$a !== \$b</code>      | Тождественно не равно | <b>TRUE</b> если <code>\$a</code> не равно <code>\$b</code> или они разных типов.       |
| <code>\$a &lt; \$b</code>     | Меньше                | <b>TRUE</b> если <code>\$a</code> строго меньше <code>\$b</code> .                      |
| <code>\$a &gt; \$b</code>     | Больше                | <b>TRUE</b> если <code>\$a</code> строго больше <code>\$b</code> .                      |
| <code>\$a &lt;= \$b</code>    | Меньше или равно      | <b>TRUE</b> если <code>\$a</code> меньше или равно <code>\$b</code> .                   |
| <code>\$a &gt;= \$b</code>    | Больше или равно      | <b>TRUE</b> если <code>\$a</code> больше или равно <code>\$b</code> .                   |

Появляется *вторая ветка* алгоритма – блок **else**, т.е. блок команд, выполняющихся в случае, когда условие ложно. Т.о. во всём условном операторе будет выполнен только один блок команд – либо первый (если условие истинно), либо второй (если условие ложно).

При выборе больше, чем из двух возможных вариантов, может быть задействована конструкция **else if** (*вложенные условия*):

```
if (условие1) {
... }
else if (условие2) {
... }
else if (условие3) {
... }
...
else {
... }
```

Последняя часть **else** без условия задействуется, когда не выполняется ни одно из перечисленных выше условий.

- Используя вложенные условия, напишите скрипт, в котором по введенному номеру года определяется – високосный он или нет. Снабдите вывод подробным описанием самого задания, вводимого значения и результата. Реализуйте ввод через элемент формы.
- В предыдущей работе №2 в заданиях 9 и 11 были созданы файлы `whatsName.html` и `hiUser.php`. В первом файле создавалась форма для ввода строкового значения (имени пользователя), которое затем отправлялось на обработку скрипту второго файла. Реализуем оба этих этапа (ввод данных и обработку) в одном файле. Откройте файл `hiUser.php` и отредактируйте его, приведя к следующему виду (недостающие фрагменты кода возьмите из файла `whatsName.html`):

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title>Привет, посетитель!</title>
5     <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=utf-8">
6 </head>
7 <body>
8     <h1>Привет, посетитель!</h1>
9
10 <?php
11 if (filter_has_var(INPUT_GET, "userName")){
12     //Если форма существует, то работаем с ней:
13     $userName = $_GET ['userName'];
14     print "<p>Привет, $userName!</p>";
15     print "<p>Как мы узнали, что Ваше имя $userName? ";
16     print "Всё очень просто – имя $userName – самое красивое слово
17         в русском языке!..)</p>";
18     print "<p>Увидимся, $userName!..</p>";
19 } else {
20     //Иначе (формы нет) – создаём её:
21     print <<< HERE
22     <form method = "get"
23         action = "">
24     <fieldset>
25         <p>Впишите ваше имя:</p>
26         <input type = "text"
27             name = "userName"
28             value = "">
29         <input type = "submit">
30     </fieldset>
31     </form>
32     HERE;
33 } //Конец if
34 <?>
35
36 </body>
37 </html>

```

В этом варианте алгоритм *разветвляется* – одна ветка формируется, если переменная \$userName уже была сформирована с помощью формы; и вторая ветка – если переменной не существует. В первом случае обрабатывается переданное скрипту значение переменной, во втором случае – создается форма для ввода этого значения.

Изучите с помощью справочника по PHP функцию filter\_has\_var (). Что такое INPUT\_GET? С помощью справочника по HTML определите назначение тега <fieldset>.

Добавьте в скрипт кнопку для вызова скрипта без передачи параметров, чтобы пользователь мог ввести в форму другое значение.

9. Индивидуальное задание (задача №2) на бинарное ветвление (см. по [1]). Снабдите вывод подробным описанием самого задания, переменных и результата. Реализуйте ввод данных через элементы формы.

## Г. Множественное ветвление

10. Наберите код следующего примера:

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4     <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=utf-8">
5     <title>Команда Switch</title>
6 </head>
7

```

```

8 <body>
9 <h1>Команда Switch</h1>
10 <h3>Демонстрация множественного ветвления</h3>
11
12 <?php
13     $roll = rand(1,6);
14
15     switch ($roll){
16         case 1:
17             $romValue = "I";
18             break;
19         case 2:
20             $romValue = "II";
21             break;
22         case 3:
23             $romValue = "III";
24             break;
25         case 4:
26             $romValue = "IV";
27             break;
28         case 5:
29             $romValue = "V";
30             break;
31         case 6:
32             $romValue = "VI";
33             break;
34         default:
35             print "Некорректные данные!";
36     } // end switch
37
38     print <<< HERE
39     <p>У Вас выпало $roll.</p>
40     <p>
41         <img src = "die$roll.jpg" alt = "die: $roll" />
42     </p>
43     <p>В римской записи это $romValue.</p>
44     HERE;
45     ?>
46
47     <p>Перезагрузите страницу (клавиша F5) для новой попытки.</p>
48 </body>
49 </html>

```

Сохраните набранный код в файле switchDice.php. Проверьте его работу в браузере, несколько раз обновив страницу.

Команда **switch** позволяет реализовать *множественное ветвление*, когда проверяемая переменная может принимать одно из нескольких (более двух) возможных значений, и для каждого значения должен выполняться свой блок команд:

**switch** (переменная) {

**case** значение1:

...

break;

**case** значение2:

...

break;

...

**case** значениеN:

...

```
break;
default:
    ...;
}
```

Последняя часть **default** задействуется, когда проверяемая переменная имеет значение, отличное от перечисленных выше вариантов.

#### Д. Функции

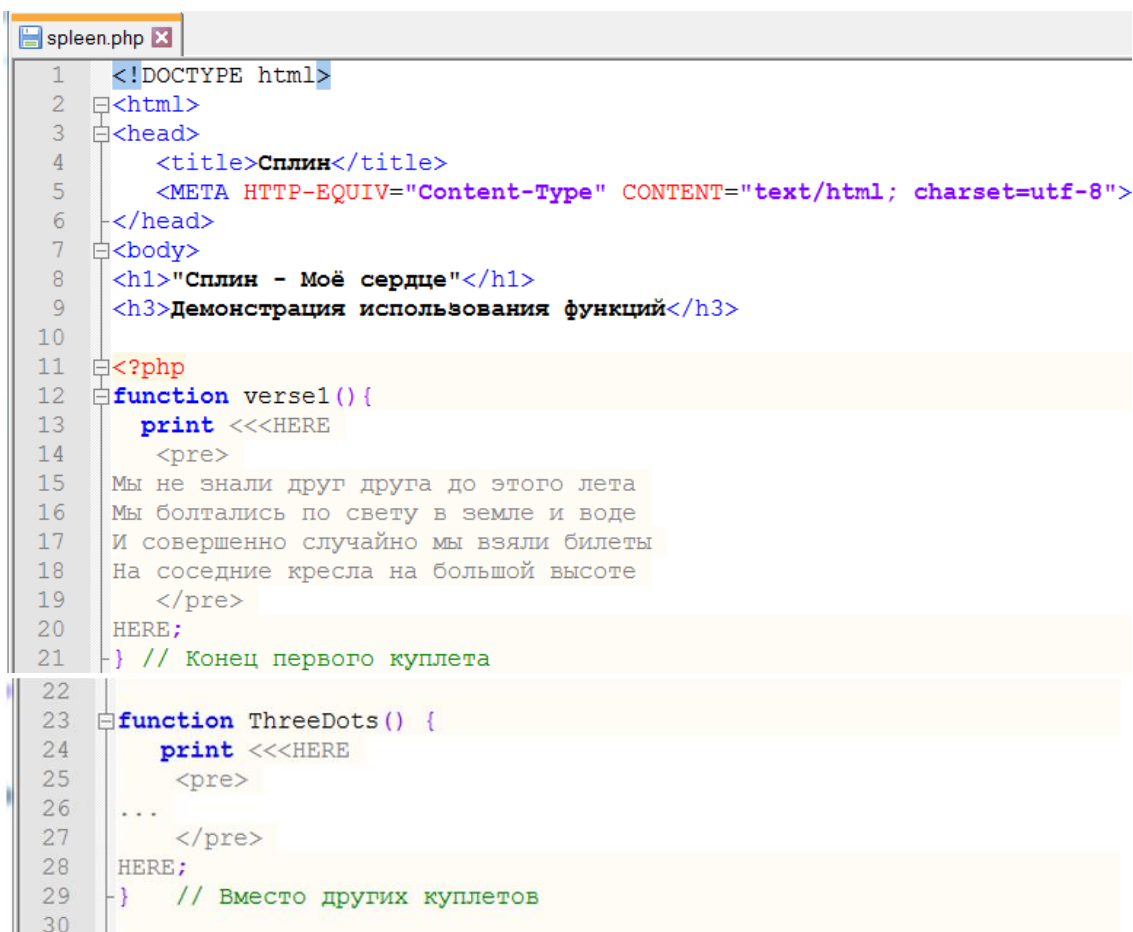
В концепции *структурного программирования* одним из важных понятий является подпрограмма. *Подпрограмма* – это именованный (идентифицируемый) фрагмент программы, содержащий определенный набор действий. Подпрограмма может быть несколько раз вызвана по своему имени при выполнении основной программы.

Подпрограммы реализуются по типу *процедур* – определенного набора действий, не требующего возврата какого-либо значения, и *функций* – подпрограмм с возвратом результата – значения определенного типа.

В функциях языка PHP объединены понятия процедуры и функции. Функции в PHP создаются для многократного вызова как с возвратом, так и без возврата результирующих значений.

11. Пример использования функций по типу процедур (без возврата значений).

Наберите следующий код (можно использовать другую полюбившуюся вам песню):



```
spleen.php
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title>Сплин</title>
5     <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=utf-8">
6 </head>
7 <body>
8     <h1>"Сплин - Моё сердце"</h1>
9     <h3>Демонстрация использования функций</h3>
10
11 <?php
12 function versel() {
13     print <<<HERE
14     <pre>
15     Мы не знали друг друга до этого лета
16     Мы болтались по свету в земле и воде
17     И совершенно случайно мы взяли билеты
18     На соседние кресла на большой высоте
19     </pre>
20     HERE;
21 } // Конец первого куплета
22
23 function ThreeDots() {
24     print <<<HERE
25     <pre>
26     ...
27     </pre>
28     HERE;
29 } // Вместо других куплетов
30
```

```

31 function chorus() {
32     print <<<HERE
33     <pre>
34     Припев:
35     И мое сердце остановилось
36     Мое сердце замерло
37     Мое сердце остановилось
38     Мое сердце замерло
39     </pre>
40     HERE;
41 } // Конец припева
42
43 versel();
44 chorus();
45 ThreeDots();
46 chorus();
47 ThreeDots();
48 chorus();
49 ThreeDots();
50 chorus();
51 chorus();
52 ?>
53 </body>
54 </html>

```

Сохраните код в файле spleen.php, проверьте его работу в браузере.

В примере часто повторяющиеся действия (вывод текста) описаны в виде функций в начале скрипта, и, затем, в основной части скрипта эти функции вызываются по своим именам в необходимом порядке.

Функции в PHP бывают *встроенными* (они уже известны интерпретатору) и *пользовательскими* (их создает и описывает программист для своих нужд). Последние могут объединяться в *библиотеки* функций, что избавляет программиста от напрасного труда «изобретения велосипеда», т.е. написания кода, который уже многократно создавался другими программистами. Внешние библиотеки подключаются к скрипту и используются как часть самого скрипта.

При создании своих функций следует избегать имен уже существующих встроенных функций, т.к. это приведет к некорректной работе скрипта. В случае сомнений необходимо обращаться к справочнику.

В описании функции присутствует *заголовок*, начинающийся со служебного слова **function**. В заголовке указывается имя функции и список *параметров*, передаваемых функции для обработки (в нашем примере мы используем функции без параметров). *Тело* функции – это последовательность команд в фигурных скобках.

Создайте полный текст песни, воспользовавшись любым поисковиком. При этом каждый куплет оформите в виде отдельной функции.

В задании 8 уже использовался вызов функции (встроенной) `filter_has_var()` с двумя параметрами – типом проверяемого значения и именем `userName` элемента массива `$_GET`.

12. Отредактируйте предыдущий код, при этом вместо нескольких функций, каждая из которых выводит на экран свой куплет и припев, опишите одну единственную функцию с входным параметром и одним возвращаемым значением:



```

12 function verse($stanza){
13     switch ($stanza){
14         case 1: //Куплет 1
15             $output = <<<HERE
16             <pre>
17             Мы не знали друг друга до этого лета,
18             Мы болтались по свету в земле и воде.
19             И совершенно случайно мы взяли билеты
20             На соседние кресла на большой высоте.
21             </pre>
22             HERE;
23             break;
24         case 2: //Куплет 2
25             $output = <<<HERE
26             <pre>
27             ...
28             </pre>
29             HERE;
30             break;
31         case 3: //Куплет 3
32             $output = <<<HERE
33             <pre>
34             ...
35             </pre>
36             HERE;
37             break;
38         default: //Припев
39             $output = <<<HERE
40             <pre>
41             Припев:
42             И моё сердце остановилось,
43             Моё сердце замерло.
44             Моё сердце остановилось,
45             Моё сердце замерло.
46             </pre>
47             HERE;
48     } // end switch
49
50     return $output; //Возврат значения
51 } // Конец функции

```

В основной части скрипта достаточно вызывать функцию `verse ()` с соответствующими номерами куплетов (любое значение отличное от 1, 2 и 3 здесь – припев):

```

61 //Основная программа
62 print verse(1);
63 print verse(5);
64 print verse(2);
65 print verse(5);
66 print verse(3);
67 print verse(5);
68 print verse(5);

```

Внесите соответствующие исправления, проверьте работоспособность кода в браузере.

Параметры и при объявлении функции перечисляются через запятую в скобках после имени этой функции. Соответствующие переменные – *локальные*, после выполнения тела функции они будут уничтожены. При вызове функции указываются *аргументы* – соответствующие значения параметров, которые будут использованы при выполнении тела функции.

Команда **return** в теле функции возвращает указанное значение (переменная `$output`) как результат выполнения функции. В нашем примере эти результаты используются в командах **print** в основной части скрипта.



13. Изучите возможность использования глобальных переменных в теле функции.
14. Индивидуальное задание (задача №3) на функции (см. по [1]). Снабдите вывод подробным описанием самого задания, переменных и результата. Реализуйте по возможности ввод данных через элементы формы.

**Литература:**

1. Абрамов С.А. и др. Задачи по программированию, 1988.
2. Дронов В. PHP 5-6, MySQL 5-6 и Dreamweaver CS4. Разработка интерактивных Web-сайтов, 2009.
3. Колисниченко Д. PHP и MySQL. Разработка Web-приложений. 4-е изд., 2013.
4. Кузнецов М. PHP на примерах. 2-е изд., 2012.
5. Маклафлин Б. PHP и MySQL. Исчерпывающее руководство, 2013.
6. Прохоренок Н.А. HTML, Javascript, PHP и MySQL. Джентльменский набор Web-мастера. - 4-е изд., 2015.
7. Суэринг С. и др. PHP 6 и MySQL 6. Библия программиста, 2010.
8. Янк К. PHP и MySQL. От новичка к профессионалу, 2013.
9. PHP: Справочник языка – Manual [Электронный ресурс]. URL: <http://php.net/manual/ru/langref.php> (дата обращения 25.03.2016).
10. Уроки PHP [Электронный ресурс]. URL: <http://www.site-do.ru/php/> (дата обращения 23.03.2016).
11. PHP.RU — Сообщество PHP-Программистов [Электронный ресурс]. URL: <https://php.ru/> (дата обращения 23.03.2016).