

Практическая работа №5

«PHP: расширенные приёмы работы со структурами данных. Строки»

Цель работы: познакомиться с различными приёмами работы с массивами и строками в языке PHP.

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).

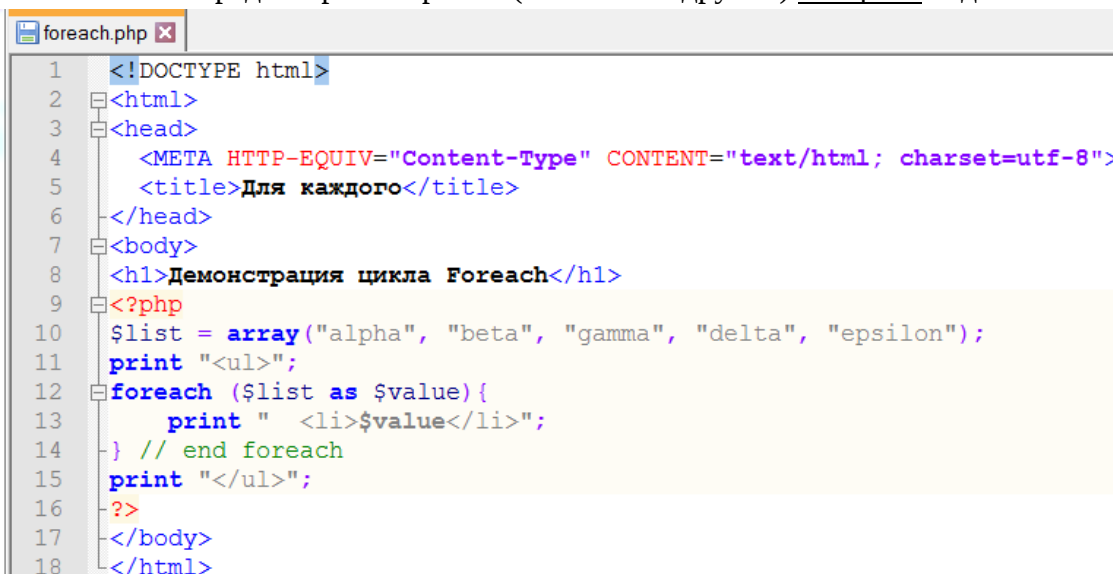
Если проводник сообщает, что нет доступа к ресурсу, то на реальном компьютере (для ОС Windows Vista / 7 / 8.x / 10) откройте «Панель управления» → «Сеть и Интернет» → «Центр управления сетями и общим доступом» → «Изменить дополнительные параметры общего доступа». В нужном профиле выберите «Включить сетевое 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 осуществляйте прямо на виртуальном компьютере.

Б. Цикл foreach

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>Для каждого</title>
6 </head>
7 <body>
8 <h1>Демонстрация цикла Foreach</h1>
9 <?php
10 $list = array("alpha", "beta", "gamma", "delta", "epsilon");
11 print "<ul>";
12 foreach ($list as $value){
13     print " <li>$value</li>";
14 } // end foreach
15 print "</ul>";
16 ?>
17 </body>
18 </html>
  
```

Сохраните код в файле foreach.php, проверьте его в браузере.

В данном примере все выводимые в окно обозревателя в виде списка значения сначала заведены с помощью функции `array()` в массив `$list`.

Цикл **foreach** работает аналогично циклу **for**, только заголовок проще: в круглых скобках служебное слово **as** разделяет имя массива (`$list`), из которого последовательно для каждой следующей итерации будет извлекаться значение очередного его элемента, и имя переменной (`$value`), в которой это очередное значение сохраняется на время итерации.

Цикл `foreach`, таким образом, «пройдёт» через массив `$list` столько раз, сколько элементов в нём содержится (в этом примере – пять). В отличие от цикла `for`, здесь нет необходимости в переменной-счётчике, которая показывала бы *индекс (ключ)* очередного элемента в массиве. Индекс текущего элемента, если он необходим, возвращает функция `key()`.

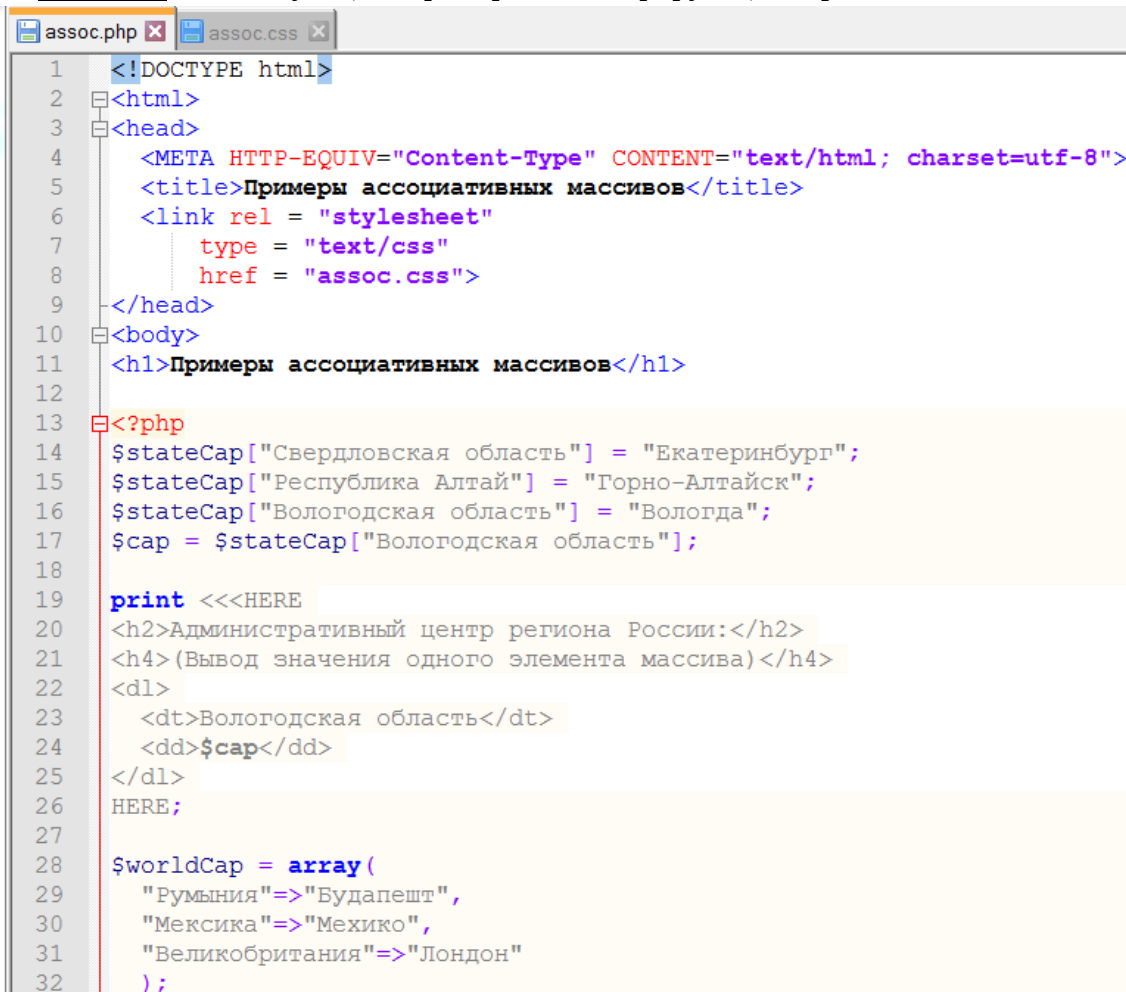
Таким образом, используйте цикл `foreach` в случаях, когда необходимо обработать каждый элемент массива без исключения.

6. С помощью справочника по PHP изучите описание и синтаксис функции `key()`.
7. Отредактируйте код предыдущего индивидуального задания (задача №5 на массив) так, чтобы в нём использовался цикл `foreach`.

В. Ассоциативный массив

В отличие от обычных массивов, которые были рассмотрены в предыдущей работе, в *ассоциативном* массиве индексами (именами, ключами) элементов выступают не целочисленные, а *строковые* значения.

8. Наберите код следующего примера, иллюстрирующего применение этого типа данных:



```

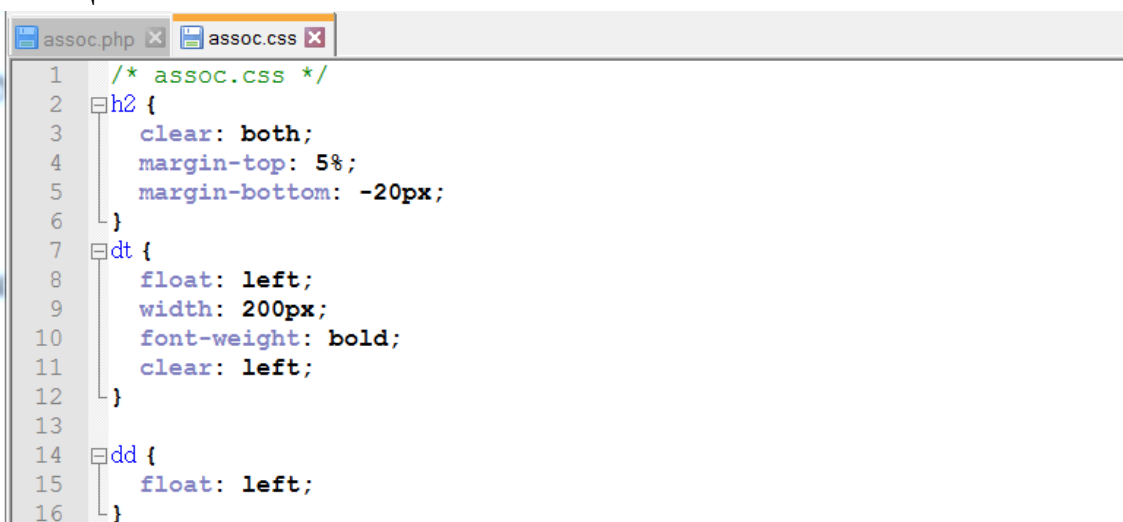
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=utf-8">
5 <title>Примеры ассоциативных массивов</title>
6 <link rel = "stylesheet"
7       type = "text/css"
8       href = "assoc.css">
9 </head>
10 <body>
11 <h1>Примеры ассоциативных массивов</h1>
12
13 <?php
14 $stateCap["Свердловская область"] = "Екатеринбург";
15 $stateCap["Республика Алтай"] = "Горно-Алтайск";
16 $stateCap["Вологодская область"] = "Вологда";
17 $cap = $stateCap["Вологодская область"];
18
19 print <<<HERE
20 <h2>Административный центр региона России:</h2>
21 <h4>(Вывод значения одного элемента массива)</h4>
22 <dl>
23 <dt>Вологодская область</dt>
24 <dd>$cap</dd>
25 </dl>
26 HERE;
27
28 $worldCap = array(
29     "Румыния"=>"Будапешт",
30     "Мексика"=>"Мехико",
31     "Великобритания"=>"Лондон"
32 );
  
```

```

33
34 $cap = $worldCap["Мексика"];
35
36 print <<<HERE
37 <h2>Столица государства:</h2>
38 <h4>(Вывод значения одного элемента массива)</h4>
39 <dl>
40 <dt>Мексика</dt>
41 <dd>$cap</dd>
42 </dl>
43
44 HERE;
45
46 print "<h2>Столицы государств:</h2>";
47 print "<h4>(Вывод значений всех элементов массива)</h4>";
48 print "<dl>";
49
50 foreach ($worldCap as $country => $capital){
51     print <<<HERE
52     <dt>$country</dt>
53     <dd>$capital</dd>
54     HERE;
55 } // end foreach
56
57 print "</dl>";
58 ?>
59 </body>
60 </html>

```

Сохраните код в файле assoc.php. Создайте также файл assoc.css со следующими стилевыми таблицами:



```

1  /* assoc.css */
2  h2 {
3      clear: both;
4      margin-top: 5%;
5      margin-bottom: -20px;
6  }
7  dt {
8      float: left;
9      width: 200px;
10     font-weight: bold;
11     clear: left;
12 }
13
14 dd {
15     float: left;
16 }

```

Проверьте работу кода в браузере.

Здесь на примере двух массивов – \$stateCap и \$worldCap – иллюстрируется: а) создание ассоциативных массивов двумя способами, б) выбор значения определенного элемента по строковому ключу (имени, индексу) и в) вывод (как одного определенного значения, так и всего массива в цикле).

Т.о. по строковому ключу (наименованию региона или страны) становится легко найти соответствующее значение (имя города).

В дальнейшем ассоциативный массив используется также, как и обычный.

На самом деле, вы уже многократно использовали ассоциативные массивы – это *суперглобальные массивы* \$_GET и \$_POST при отправке и получении данных из формы. Каждый элемент такого массива соответствует элементу формы.

При передаче HTTP-запроса (GET или POST) PHP создаёт также ассоциативный суперглобальный массив \$_REQUEST. Он объединяет в себе все переменные массивов \$_GET, \$_POST, \$_SESSION и \$_COOKIE. Обращаться в скрипте к переданным значениям полей формы можно, соответственно, и через этот массив.

9. С помощью справочника по языку PHP изучите описание суперглобальных массивов \$_SESSION и \$_COOKIE.

Г. Двумерный массив

Многомерным называется массив, элементами которого являются массивы. В наиболее простой версии – *двумерный* массив – такая структура данных используется для представления табличной информации.

В качестве иллюстрации работы с двумерными массивами создадим скрипт, вычисляющий расстояния между парами городов.

10. Наберите код HTML-документа с формой для ввода и отправки данных скрипту:

```

basicMultiArray.html
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=utf-8">
5      <title>Двумерный массив</title>
6  </head>
7  <body>
8      <h1>Двумерный массив</h1>
9
10     <form action = "basicMultiArray.php"
11         method = "post">
12     <table>
13     <tr>
14         <th>Первый город</th>
15         <th>Второй город</th>
16     </tr>
17
18     <tr>
19         <td>
20             <select name = "cityA">
21                 <option value = "0">Индианаполис</option>
22                 <option value = "1">Нью-Йорк</option>
23                 <option value = "2">Токио</option>
24                 <option value = "3">Лондон</option>
25             </select>
26         </td>
27
28         <td>
29             <select name = "cityB">
30                 <option value = "0">Индианаполис</option>
31                 <option value = "1">Нью-Йорк</option>
32                 <option value = "2">Токио</option>
33                 <option value = "3">Лондон</option>
34             </select>
35         </td>
36     </tr>

```

```

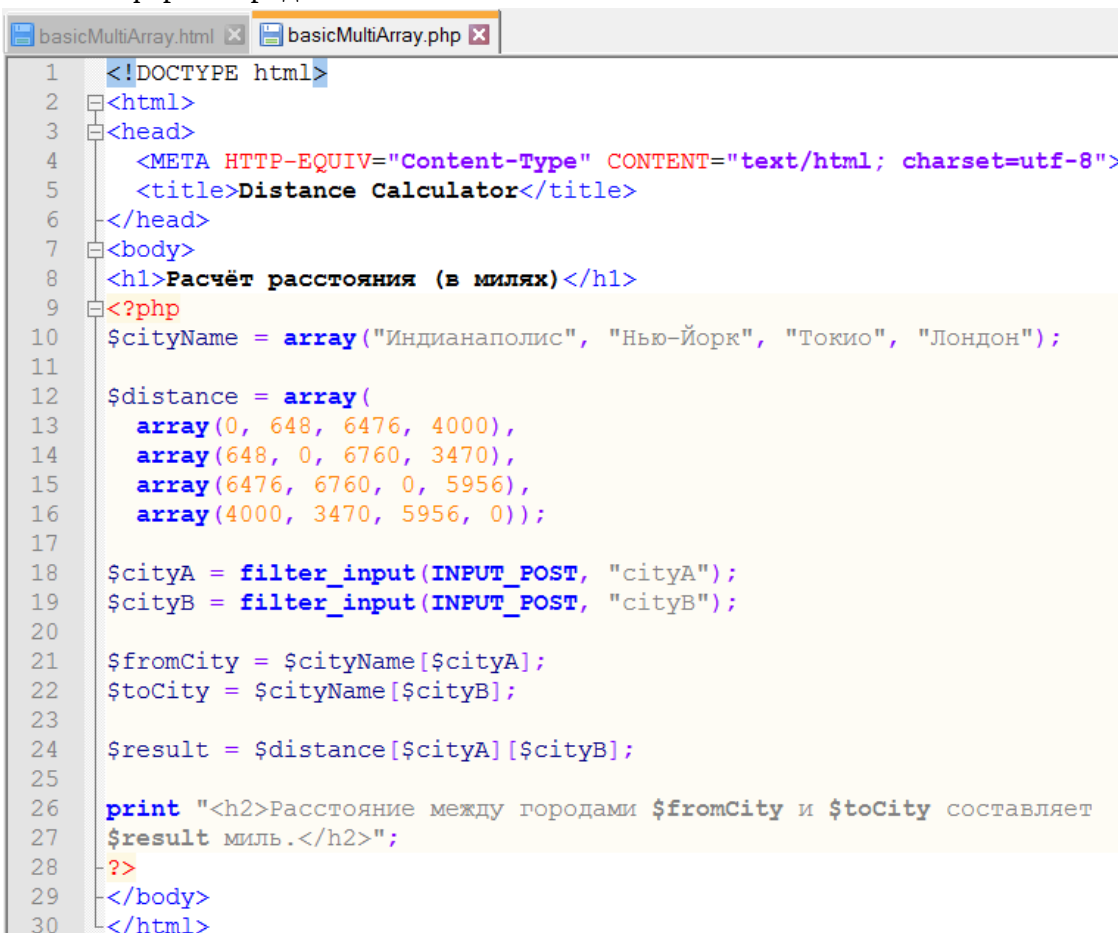
37
38 <tr>
39 <td colspan = "2">
40 <input type = "submit"
41     value = "Рассчитать" />
42 </td>
43 </tr>
44 </table>
45 </form>
46
47 </body>
48 </html>

```

Сохраните код в файле basicMultiArray.html, проверьте отображение элементов формы.

По нажатию кнопки «Рассчитать» в переменной cityA будет храниться числовой индекс, соответствующий названию первого города, а в переменной cityB – второго.

11. Наберите код скрипта, в котором осуществляется расчет расстояния между выбранными на форме городами:



```

1 <!DOCTYPE html>
2 <html>
3 <head>
4 <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=utf-8">
5 <title>Distance Calculator</title>
6 </head>
7 <body>
8 <h1>Расчёт расстояния (в милях)</h1>
9 <?php
10 $cityName = array("Индианаполис", "Нью-Йорк", "Токио", "Лондон");
11
12 $distance = array(
13     array(0, 648, 6476, 4000),
14     array(648, 0, 6760, 3470),
15     array(6476, 6760, 0, 5956),
16     array(4000, 3470, 5956, 0));
17
18 $cityA = filter_input(INPUT_POST, "cityA");
19 $cityB = filter_input(INPUT_POST, "cityB");
20
21 $fromCity = $cityName[$cityA];
22 $toCity = $cityName[$cityB];
23
24 $result = $distance[$cityA][$cityB];
25
26 print "<h2>Расстояние между городами $fromCity и $toCity составляет
27 $result миль.</h2>";
28 ?>
29 </body>
30 </html>

```

Сохраните его в файл basicMultiArray.php. Проверьте работу скрипта.

Здесь массив \$cityName – простой массив строковых значений – имён городов. В переменных \$cityA и \$cityB хранятся индексы выбранных на форме в HTML-документе городов.

Массив \$distance состоит из одномерных массивов, каждый из которых соответствует строке таблицы расстояний. Каждая строка содержит расстояния между определенным городом и другими городами. Тогда расстояние выбирается по двум значениям – номеру строки (индекс исходного города) и столбца (индекс конечного города).

12. Отредактируйте код HTML-документа basicMultiArray.html, заменив числовые обозначения полей списков строковыми:

```

19 <td>
20 <select name = "cityA">
21 <option value = "Indianapolis">Индианаполис</option>
22 <option value = "New York">Нью-Йорк</option>
23 <option value = "Tokio">Токио</option>
24 <option value = "London">Лондон</option>
25 </select>
26 </td>
27
28 <td>
29 <select name = "cityB">
30 <option value = "Indianapolis">Индианаполис</option>
31 <option value = "New York">Нью-Йорк</option>
32 <option value = "Tokio">Токио</option>
33 <option value = "London">Лондон</option>
34 </select>
35 </td>

```

Сохраните измененный документ под именем MultiArray.html.

Тогда код основного скрипта будет выглядеть следующим образом:

```

MultiArray.php
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=utf-8">
5 <title>Distance Calculator</title>
6 </head>
7 <body>
8 <h1>Расчёт расстояния (в милях)</h1>
9 <?php
10 //создание массивов
11 $cityName = array(
12     "Indianapolis" => "Индианаполис",
13     "New York" => "Нью-Йорк",
14     "Tokyo" => "Токио",
15     "London" => "Лондон" );
16 $indy = array (
17     "Indianapolis" => 0,
18     "New York" => 648,
19     "Tokyo" => 6476,
20     "London" => 4000 );
21 $ny = array (
22     "Indianapolis" => 648,
23     "New York" => 0,
24     "Tokyo" => 6760,
25     "London" => 3470 );
26 $tokyo = array (
27     "Indianapolis" => 6476,
28     "New York" => 6760,
29     "Tokyo" => 0,
30     "London" => 5956 );
31 $london = array (
32     "Indianapolis" => 4000,
33     "New York" => 3470,
34     "Tokyo" => 5956,
35     "London" => 0 );
36 $distance = array (
37     "Indianapolis" => $indy,
38     "New York" => $ny,
39     "Tokyo" => $tokyo,
40     "London" => $london );

```

```

41
42 $cityA = filter_input(INPUT_POST, "cityA");
43 $cityB = filter_input(INPUT_POST, "cityB");
44
45 $fromCity = $cityName[$cityA];
46 $toCity = $cityName[$cityB];
47
48 $result = $distance[$cityA][$cityB];
49
50 print "<h2>Расстояние между городами $fromCity и $toCity составляет
51 $result миль.</h2>";
52 -?>
53 </body>
54 </html>

```

Сохраните этот код в файле MultiArray.php. Исправьте в HTML-коде формы имя файла-назначения при отправке и проверьте работу кода.

Массив \$distance здесь – двумерный ассоциативный массив, элементами которого также являются ассоциативные массивы. Английские названия городов используются в качестве ключей (имён, индексов) элементов.

Код в случае использования ассоциативного массива получился большим, однако, он нагляднее и более понятен.

Дополнительного кода требует преобразование английских названий городов в именах элементов массивов в русские названия при выводе результата в окно браузера. Без этого код скрипта получился бы значительно короче (отпала бы необходимость в массиве \$cityName, переменных \$fromCity и \$toCity).

Д. Строковые значения в PHP

По сути, *строка* – это одномерный массив символов, к каждому из которых можно обращаться по его индексу (начиная с нулевого). Для строкового типа данных в PHP имеется большое количество стандартных функций.

13. Наберите следующий код и сохраните его в файле pigify.php:

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4 <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=utf-8">
5 <title>Игра слов</title>
6 </head>
7 <body>
8 <h1>Игра слов</h1>
9 <?php
10 if (!filter_has_var(INPUT_POST, "inputString")){
11     //форма ввода значения
12     print <<<HERE
13     <form action = ""
14         method = "post">
15         <fieldset>
16             <textarea name = "inputString"
17                 rows = "20"
18                 cols = "40"></textarea>
19             <input type = "submit"
20                 value = "Отправить" />
21         </fieldset>
22     </form>
23     HERE;

```



```

24 } else {
25     //обрабатываем введенное значение:
26     $inputString = filter_input(INPUT_POST, "inputString");
27     $newPhrase = "";
28     //записываем слова в массив
29     $words = explode(" ", $inputString);
30     foreach ($words as $theWord){
31         $theWord = rtrim($theWord);
32         $firstLetter = $theWord[0];
33         if (strstr("aeiouAEIOU", $firstLetter)){
34             //первая буква - гласная
35             $newWord = $theWord . "way";
36         } else {
37             //первая буква - согласная
38             $restOfWord = substr($theWord, 1, strlen($theWord)-1);
39             $newWord = $restOfWord . $firstLetter . "ay";
40         } // end if
41         $newPhrase = $newPhrase . $newWord . " ";
42     } // end foreach
43     print "<p>$newPhrase</p> \n";
44 } // end if
45 ?>
46 </body>
47 </html>

```

Проверьте работоспособность кода на примере слов только из латинских букв.

Это программа-игра, «коверкающая» слова – первая буква каждого слова (если она согласная) перемещается в конец слова с добавлением «ay»; если первая буква гласная, то в конец слова просто дописывается «way».

Здесь в условии первого оператора if с помощью переменной \$inputString проверяется – первый раз открыта страница или нет (введено ли значение в текстовое поле). Если пользователь впервые открыл страницу – создаётся форма ввода, иначе (значение отправлено в тот же скрипт) – обрабатывается введенное в текстовое поле значение и выводится результат (переменная \$newPhrase).

Функция explode() разбивает строку (указана вторым аргументом) на слова по заданному разделителю (первый аргумент) и помещает их в массив \$words. Каждое слово (в цикле foreach) «очищается» с помощью функции rtrim() от ненужных символов табуляции, перевода каретки и пр.

Переменная \$firstLetter (первая буква слова) получает своим значением начальный (нулевой) символ строки (по сути, используется обращение к строке как к массиву).

Функция substr() – вырезка из строки подстроки заданной длины – позволяет получить остаток слова без первой буквы (переменная \$restOfWord).

Переменные \$newWord и \$newPhrase содержат измененные слова и всю фразу соответственно. Само изменение происходит с помощью конкатенации (слияния) строковых значений. В PHP команда конкатенации обозначается символом точки.

Примечание: этот код будет корректно работать только с латинскими символами в однобайтовой кодировке ASCII. При использовании русских символов будет задействована многобайтная (от 1 до 6 байт на символ) кодировка [UTF-8](#), с которой большинство обычных строковых функций в PHP правильно работать не будет (за редкими исключениями). Для обработки строк в многобайтных кодировках в PHP предусмотрен специальный набор функций (так называемые mb_функции), ознакомиться которыми можно, например, [здесь](#).

Другим допустимым способом может быть преобразование строки специальными функциями PHP из многобайтной кодировки UTF-8 в однобайтную (например, Windows-1251).

14. Изучите основные mb_функции языка PHP и реализуйте в предыдущем скрипте поддержку русских букв в многобайтной кодировке UTF-8.

15. Индивидуальное задание (задача №6) на строки (см. по [1]). Реализуйте ввод данных через элементы формы. Снабдите ввод и вывод подробным описанием самого задания, переменных и результата. Обеспечьте поддержку строк в кодировке UTF-8.

Литература:

1. Абрамов С.А. и др. Задачи по программированию, 1988.
2. Маклафлин Б. PHP и MySQL. Исчерпывающее руководство, 2013.
3. Суэринг С. и др. PHP 6 и MySQL 6. Библия программиста, 2010.
4. Янк К. PHP и MySQL. От новичка к профессионалу, 2013.
5. PHP: Справочник языка – Manual [Электронный ресурс]. URL: <http://php.net/manual/ru/langref.php> (дата обращения 25.03.2016).