Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет информационных технологий и управления

Кафедра информационных технологий автоматизированных систем

Отчет по лабораторной работе №3 по дисциплине

ТЕХНОЛОГИИ ИНТЕРНЕТ-ПРОГРАММИРОВАНИЯ
на тему
«Циклы и массивы»

Выполнил ст. гр. 820601 А.Р. Шведов
Проверил преп. каф. ИТАС А.Л. Гончаревич

Минск 2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

PHP — язык программирования, который наиболее распространён в сфере веб-разработки. Язык *PHP* работает на удаленном сервере, поэтому он и называется серверный язык программирования.

Любой скрипт *PHP* состоит из последовательности операторов. Оператор может быть присваиванием, вызовом функции, циклом, условным выражением или пустым выражением. Операторы обычно заканчиваются точкой с запятой. Также операторы могут быть объединены в группу заключением группы операторов в фигурные скобки. Группа операторов также является оператором.

Интернет — это множество компьютеров по всему миру, соединённых между собой проводами в единую сеть. Все компьютеры делятся на две большие группы: клиенты и сервера. Клиенты инициируют запросы на сервера, а те, в свою очередь, их принимают, обрабатывают и отправляют клиенту ответ.

PHP позволяет решить множество задач связанных с клиент-серверной архитектурой, например:

1 С помощью *HTML* можно только создать форму. А обработать то, что ввёл пользователь, может лишь *PHP*.

2 Если делать блог на чистом *HTML*, то на каждую статью требуется создавать новый файл. Добавлять и редактировать записи придётся вручную. PHP позволяет обойтись с помощью одного файла, а статьи хранить в базе данных. Благодаря этому, можно сделать админку, из которой можно будет добавлять и редактировать контент.

3 *PHP* позволяет реализовать механизм авторизации на сайте.

1 ПОСТАНОВКА ЗАДАЧИ

Изучить семантику, синтаксис и возможности языка *PHP*. Ознакомиться с *POST* и *GET* запросами протокола *HTTP*, параметрами передаваемыми через *URL* и *HTML* формами.

2 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

HTTP — это протокол, позволяющий получать различные ресурсы, например *HTML*-документы. Протокол *HTTP* лежит в основе обмена данными в Интернете. *HTTP* является протоколом клиент-серверного взаимодействия, что означает инициирование запросов к серверу самим получателем, обычно веб-браузером (*web-browser*). Полученный итоговый документ может состоять из различных поддокументов, являющихся частью итогового документа: например, из отдельно полученного текста, описания структуры документа, изображений, видео-файлов, скриптов и многого другого.

Запрос *GET* передает данные в *URL* в виде пар «имя-значение» (другими словами, через ссылку), а запрос *POST* передает данные в теле запроса. Это различие определяет свойства методов и ситуации, подходящие для использования того или иного *HTTP* метода.

Страница, созданная методом *GET*, может быть открыта повторно множество раз. Такая страница может быть кэширована браузерами, проиндексирована поисковыми системами и добавлена в закладки пользователем. Из этого следует, что метод *GET* следует использовать для получения данных от сервера и не желательно в запросах, предполагающих внесений изменений в ресурс.

Запрос, выполненный методом *POST*, напротив следует использовать в случаях, когда нужно вносить изменение в ресурс (выполнить авторизацию, отправить форму оформления заказа, форму обратной связи, форму онлайн заявки). Повторный переход по конечной ссылке не вызовет повторную обработку запроса, так как не будет содержать переданных ранее параметров. Метод *POST* имеет большую степень защиты данных, чем *GET*: параметры запроса не видны пользователю без использования специального ПО, что дает методу преимущество при пересылке конфиденциальных данных, например в формах авторизации.

В *PHP* параметры, переданные через *URL*, хранятся в системной переменной $_*GET*. Она представляет собой ассоциативный массив (или, по-другому, словарь). Ассоциативный массив (или словарь) – это такая структура данных, которая содержит пары ключ-значение. Ключи словаря $_*GET* – это имена параметров. По аналогии с методом *GET*, параметры, переданные методом *POST*, содержатся в системной переменной $_*POST*. Это также, как и $_*GET*, ассоциативный массив. Ключами в массиве $_*POST* являются значения атрибутов *name* у элементов на форме.

3 ХОД РАБОТЫ

Работа выполняется с помощью редактора кода – *Visual Studio Code*. Работа с программой начинается с создания *PHP* файла, в который будут помещаться скрипты.

В задании 1 для начала необходимо создать папку с изображениями и две страницы *index.php* и *photo.php*. На странице *photo.php* необходимо отобразить определённую картинку. Чтобы понять, какую именно картинку показывать, нужно считать определённый параметр – номер изображения – из *URL*-адреса. Фрагмент кода:

```
echo "<b> Task 1 </b> <br>";

$n = 100;
$i = 0;
while ($i <= $n) {
   if ($i % 3 == 0) {
      echo "$i ";
      $i++;
   } else
      $i++;
}
```

Страница *photo.php* показана на рисунке 3.1.

Task 1
0 3 6 9 12 15 18 21 24 27 30 33 36 39 42 45 48 51 54 57 60 63 66 69 72 75 78 81 84 87 90 93 96 99

Рисунок 3.1 — Задание 1

В задании 2 необходимо было с помощью цикла *do…while* написать функцию для вывода чисел от нуля до десяти. Фрагмент кода:

```
echo "<b> Task 2 </b> <br>";
$i = 0;
do {
   if ($i % 2 == 0)
```

```
        echo $i . " - это четное число <br/>";
    elseif ($i %2 != 0)
        echo $i . " - это нечетное число <br/>";
    else
        echo $i . " - это ноль <br/>";
    $i++;
} while ($i <= 10);
```

Результат выполнения программы показан на рисунке рисунке 3.2.

Task 2
0 - это ноль
1 - это нечетное число
2 - это четное число
3 - это нечетное число
4 - это четное число
5 - это нечетное число
6 - это четное число
7 - это нечетное число
8 - это четное число
9 - это нечетное число
10 - это четное число

Рисунок 3.2 — Задание 2

В задании 3 необходимо было вывести с помощью цикла *for* числа от нуля до девяти, не используя тело цикла. Результат выполнения задания продемонстрирован на рисунке 3.3. Фрагмент кода:

```
for ($i = 0; $i < 10; print($i++)) {}
```

Task 3
0123456789

Рисунок 3.3 — Задание 3

В задании 4 необходимо объявить массив, где в качестве ключей будут использоваться названия областей, а в качестве значений – массивы с названиями городов из соответствующей области. Вывести в цикле значения массива. Фрагмент кода:

```
echo "<b> Task 4 </b> <br>";
$obl = array(
   "Московская область" => array("Москва", "Зеленоград", "Клин"),
   "Ленинградская область" => array("Санкт-Петербург", "Всеволожск", "Павловск", "Кронштадт"),
   "Аму́рская о́бласть " => array("Белогорск", "город Свободный", "город Шимановск")
);
foreach ($obl as $i => $cities) {
   echo ("<b>$i</b>" . "<ul>"
   );
   foreach ($cities as $city) {
      echo ("<li>$city</li>");
   }
   echo ("</ul>");
}
```

Результат выполнения задания показан на рисунке 3.4

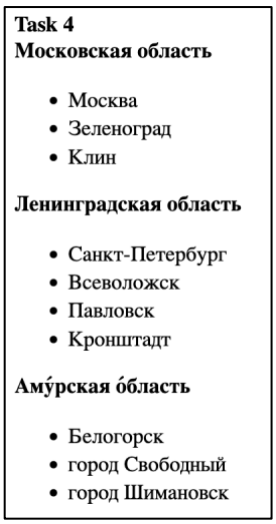

Рисунок 3.4 — Задание 4

В задании 5 необходимо повторите предыдущее задание, но вывести на экран только города, начинающиеся с буквы «К». Результат выполнения задания показан на рисунке 3.5.

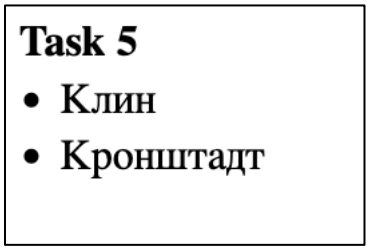

Рисунок 3.5 — Задание 5

Фрагмент кода:

```
$obl = array(
  "Московская область" => array("Москва", "Зеленоград", "Клин"),
  "Ленинградская область" => array("Санкт-Петербург", "Всеволожск", "Павловск", "Кронштадт"),
  "Аму́рская о́бласть " => array("Белогорск", "город Свободный", "город Шимановск")
);
foreach ($obl as $i => $cities) {
  foreach ($cities as $city) {
    if (str_starts_with($city, 'К')) {
      echo ("<li>$city</li>");
    }
  }
  echo ("</ul>");
}
```

В задании 6 необходимо объявить массив, индексами которого являются буквы русского языка, а значениями – соответствующие латинские буквосочетания. Реализовать функцию транслитерации строк.

Заметим, что функция *strtr($from, $to)* возвращает копию *string*, в которой все вхождения каждого символа (однобайтного) из *from* были заменены на соответствующий символ в параметре *to. Фрагмент кода:*

```
function translit($string) {
```

```
    $converter = array(
        'а' => 'a',   'б' => 'b',   'в' => 'v',
        'г' => 'g',   'д' => 'd',   'е' => 'e',
        'ё' => 'e',   'ж' => 'zh',  'з' => 'z',
        'и' => 'i',   'й' => 'y',   'к' => 'k',
        'л' => 'l',   'м' => 'm',   'н' => 'n',
        'о' => 'o',   'п' => 'p',   'р' => 'r',
        'с' => 's',   'т' => 't',   'у' => 'u',
        'ф' => 'f',   'х' => 'h',   'ц' => 'c',
        'ч' => 'ch',  'ш' => 'sh',  'щ' => 'sch',
        'ь' => '\'',  'ы' => 'y',   'ъ' => '\'',
        'э' => 'e',   'ю' => 'yu',  'я' => 'ya',

        'А' => 'A',   'Б' => 'B',   'В' => 'V',
        'Г' => 'G',   'Д' => 'D',   'Е' => 'E',
        'Ё' => 'E',   'Ж' => 'Zh',  'З' => 'Z',
        'И' => 'I',   'Й' => 'Y',   'К' => 'K',
        'Л' => 'L',   'М' => 'M',   'Н' => 'N',
        'О' => 'O',   'П' => 'P',   'Р' => 'R',
        'С' => 'S',   'Т' => 'T',   'У' => 'U',
        'Ф' => 'F',   'Х' => 'H',   'Ц' => 'C',
        'Ч' => 'Ch',  'Ш' => 'Sh',  'Щ' => 'Sch',
        'Ь' => '\'',  'Ы' => 'Y',   'Ъ' => '\'',
        'Э' => 'E',   'Ю' => 'Yu',  'Я' => 'Ya',
    );
    return strtr($string, $converter);
}
```

print(translit("Объявите массив, индексами которого являются буквы русского языка"));

Результат выполнения задания показан на рисунке 3.6.

Task 6
Ob'yavite massiv, indeksami kotorogo yavlyayutsya bukvy russkogo yazyka

Рисунок 3.6 — Задание 6

В задании 7 необходимо написать функцию, которая заменяет в строке пробелы на подчеркивания и возвращает видоизмененную строчку. Результат выполнения задания показан на рисунке 3.7. Фрагмент кода:

```
function replace($string) {
   $conv= array(' ' => '_');
   return strtr($string, $conv);
}
print(replace("Напишите функцию, которая заменяет в строке пробелы на подчеркивания и возвращает видоизмененную строчку."));
```

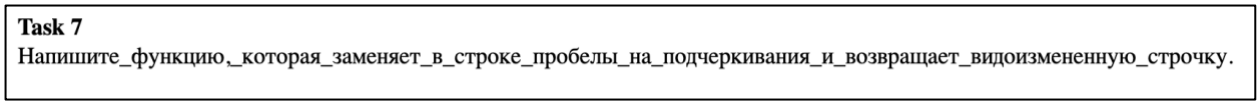

Рисунок 3.7 — Задание 7

В задании 8 необходимо объединить ранее написанные функции в одну, которая получает строк Результат выполнения задания продемонстрирован на рисунке 3.6. На русском языке производит транслитерацию и замену пробелов на подчеркивания (аналогичная задача решается при конструировании *url*-адресов на основе названия статьи в блогах). Фрагмент кода:

```
function task8($string) {
   return replace(translit($string));
}
```

Результат выполнения задания показан на рисунке 3.8.

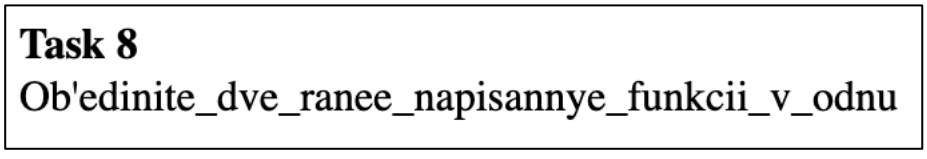

Рисунок 3.8 — Задание 8

ЗАКЛЮЧЕНИЕ

HTTP — лёгкий в использовании расширяемый протокол. Структура клиент-сервера, вместе со способностью к простому добавлению заголовков, позволяет *HTTP* протоколу продвигаться вместе с расширяющимися возможностями Сети.

PHP позволяет создавать качественные *web*-приложения за очень короткие сроки, получая продукты, легко модифицируемые и поддерживаемые в будущем.

PHP прост для освоения, и вместе с тем способен удовлетворить запросы профессиональных программистов.

Язык *PHP* постоянно совершенствуется, и ему наверняка обеспечено долгое доминирование в области языков *web*-программирования, по крайней мере, в ближайшее время.

Мною были изучены циклы и массивы в языке *PHP*. Также были изучены основные функции для работы с массивами.