

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
Проверил преп. каф. ИТАС

А.Р. Шведов
А.Л. Гончаревич

Минск 2022

СОДЕРЖАНИЕ

Введение	3
1 Постановка задачи	4
2 Теоретическая часть	5
3 Ход работы	7
Заключение.....	14

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 Вы делаете 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*. Изучение базового синтаксиса в языке *PHP*, работу с переменными, разными типами данных, операциями сравнения и логическими операциями. Ознакомление с основами серверных технологий.

2 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Следующими по частоте использования, после конструкций условий (условных операторов), находятся циклы. Циклы позволяют повторять определенное (и даже неопределенное, когда работа цикла зависит от условия) количество раз различные операторы. Данные операторы называются телом цикла. Проход цикла называется итерацией. *PHP* поддерживает три вида циклов: *while*, *do..while*, *for*, *foreach*.

При использовании циклов есть возможность использования операторов *break* и *continue*. *Break* прерывает работу всего цикла, а *continue* только текущей итерации.

Циклы *while* являются простейшим видом циклов в *PHP*. Простейшей формой цикла *while* является следующее выражение:

```
while (expr)
{
    statement
}
```

Смысл выражения *while* очень прост. Оно указывает *PHP* выполнять вложенные выражения повторно до тех пор, пока выражение в самом *while* является *true*. Значение выражения *expr* проверяется каждый раз перед началом цикла, поэтому даже если значение выражения изменится в процессе выполнения вложенных выражений в цикле, выполнение не прекратится до конца итерации (каждый раз, когда *PHP* выполняет выражения в цикле – это одна итерация). Если выражение *while* равно *false* с самого начала, вложенные выражения ни разу не будут выполнены.

Цикл *do-while* очень похож на цикл *while*, с тем отличием, что истинность выражения проверяется в конце итерации, а не в начале. Главное отличие от обычного цикла *while* в том, что начальная итерация цикла *do-while* гарантированно выполнится (истинность выражения проверяется в конце итерации), тогда как она может не выполниться в обычном цикле *while* (истинность выражения которого проверяется в начале выполнения каждой итерации, и если изначально имеет значение *false*, то выполнение цикла будет прервано сразу).

Цикл *for* самый сложный цикл в *PHP*. Синтаксис цикла *for* следующий:

```
for (expr1; expr2; expr3){
    statement
}
```

Выражение *expr1* всегда вычисляется (выполняется) только один раз в начале цикла. В начале каждой итерации оценивается выражение *expr2*. Если оно принимает значение *true*, то цикл продолжается и выполняются вложенные операторы. Если оно принимает значение *false*, выполнение цикла заканчивается. В конце каждой итерации выражение *expr3* вычисляется (выполняется). Каждое из выражений может быть пустым или содержать несколько выражений, разделённых запятыми. В *expr2* все выражения, разделённые запятыми, вычисляются, но результат берётся из последнего. Если выражение *expr2* отсутствует, это означает, что цикл будет выполняться бесконечно.

Конструкция *foreach* предоставляет простой способ перебора массивов. *foreach* работает только с массивами и объектами, и будет генерировать ошибку при попытке использования с переменными других типов или неинициализированными переменными. Существует два вида синтаксиса:

```
foreach (iterable_expression as $value)  
statement  
foreach (iterable_expression as $key => $value)  
statement
```

На самом деле массив в PHP – это упорядоченное отображение, которое устанавливает соответствие между значением и ключом. Этот тип оптимизирован в нескольких направлениях, поэтому можно использовать его как собственно массив, список (вектор), хеш-таблицу (являющуюся реализацией карты), словарь, коллекцию, стек, очередь. Так как значением массива может быть другой массив PHP, можно также создавать деревья и многомерные массивы.

Массив (тип *array*) может быть создан языковой конструкцией *array()*. В качестве параметров она принимает любое количество разделённых запятыми пар *key => value* (ключ => значение).

```
array(  
    key => value,  
    key2 => value2,  
    key3 => value3,  
    ...  
)
```

3 ХОД РАБОТЫ

Рассмотрим практическую реализацию операторов, приведенных в цели данной работы. Работа выполняется с помощью редактора кода — *VSCode*.

В задании 1 необходимо было с помощью цикла *while* вывести все числа в интервале $[0..100]$, которые делятся на три без остатка. Результат выполнения задания показан на рисунке 3.1.

На рисунке 3.1 приведен результат работы скрипта. Фрагмент кода:

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

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

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. Результат выполнения задания продемонстрирован на рисунке 3.2. Фрагмент кода:

```
echo "<b> Task 2 </b> <br>";  
$i = 0;  
do {  
    if ($i == 0)  
        echo $i . " - это ноль <br/>";  
    elseif ($i % 2 == 0)  
        echo $i . " - это четное число <br/>";  
}
```

```

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

```

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:

```

echo "<b> Task 3 </b> <br>";
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. Результат выполнения задания показан на рисунке 3.4. Фрагмент кода:

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

Task 4

Московская область

- Москва
- Зеленоград
- Клин

Ленинградская область

- Санкт-Петербург
- Всеволожск
- Павловск
- Кронштадт

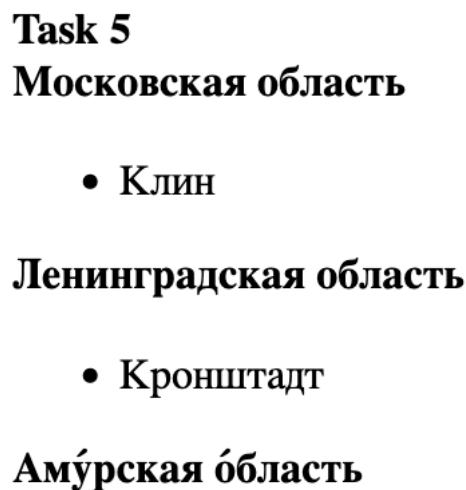
Амурская область

- Белогорск
- город Свободный
- город Шимановск

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. Фрагмент кода:

```
echo "<b> Task 5 </b> <br>";
$obl = array(
    "Московская область" => array("Москва", "Зеленоград", "Клин"),
    "Ленинградская область" => array("Санкт-Петербург",
    "Всеволожск", "Павловск", "Кронштадт"),
    "Амурская область" => array("Белогорск", "город Свободный", "город
Шимановск")
);
foreach ($obl as $i => $cities) {
    echo ("<b>$i</b>" . "<ul>"
);
    foreach ($cities as $city) {
        if (str_starts_with($city, 'K')) {
            echo ("<li>$city</li>");
        }
    }
    echo ("</ul>");
}
```



Task 5

Московская область

- Клин

Ленинградская область

- Кронштадт

Амурская область

Рисунок 3.5 — Задание 5

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*.

Результат выполнения задания показан на рисунке 3.6. 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',  
        'ъ'=>'\', 'ы'=>'y', 'ь'=>'\',  
        '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',  
        'Ъ'=>'\', 'Ы'=>'Y', 'Ь'=>'\',  
        'Э'=>'E', 'Ю'=>'Yu', 'Я'=>'Ya',  
    );  
    return strtr($string, $converter);  
}
```

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

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 строку. 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 строку."));
```

Task 7

Напишите_функцию,_которая_заменяет_в_строке_пробелы_на_подчеркивания_и_возвращает_видоизмененную_строку.

Рисунок 3.7 — Задание 7

В задании 8 необходимо объединить ранее написанные функции в одну, которая получает строку на русском языке, производит транслитерацию и замену пробелов на подчеркивания (аналогичная задача решается при конструировании *url*-адресов на основе названия статьи в блогах). Результат выполнения задания показан на рисунке 3.8. Фрагмент кода:

```
function translit($string) {  
    $converter = array(  
        'a' => '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',
```

'o'=>'o', 'n'=>'p', 'p'=>'r',
 'c'=>'s', 'm'=>'t', 'y'=>'u',
 'ϕ'=>'f', 'x'=>'h', 'u'=>'c',
 'ч'=>'ch', 'u'=>'sh', 'u'=>'sch',
 'b'=>'\\', 'bl'=>'y', 'b'=>'\\',
 'э'=>'e', 'ю'=>'yu', 'я'=>'ya',

'A'=>'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',
 'Б'=>'\\', 'Ы'=>'Y', 'Ь'=>'\\',
 'Э'=>'E', 'Ю'=>'Yu', 'Я'=>'Ya',

' '=>'_'

);

return strtr(\$string, \$converter);

}

print(translit("Объедините две ранее написанные функции в одну"));

Task 8

Ob'edinite_dve_ranee_napisannye_funkcii_v_odnu

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.

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

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

Язык *RHP* 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 *RHP*. Также были изучены основные функции для работы с массивами.