

Министерство образования Республики Беларусь

Учреждение образования  
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет информационных технологий и управления

Кафедра информационных технологий автоматизированных систем

Отчёт  
по лабораторной работе №1  
«Ветвления и функции»  
по дисциплине «Технологии интернет-программирования»

Выполнил:  
студент гр. 820601  
Шведов А. Р

Проверил:  
А. Л. Гончаревич

Минск 2022

## СОДЕРЖАНИЕ

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

## 1. 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.

Оператор ветвления — оператор или конструкция языка программирования, обеспечивающая выполнение определённой команды (набора команд) только при условии истинности некоторого логического выражения, либо выполнение одной из нескольких команд в зависимости от значения некоторого выражения.

Оператор ветвления применяется в случаях, когда выполнение или невыполнение некоторого набора команд должно зависеть от выполнения или невыполнения некоторого условия.

Данная лабораторная работа ставит своей целью дать понимание работы операторов ветвления, таких как *if* или *switch*, и функций с помощью технологии языка *PHP*.

## 2. ПОСТАНОВКА ЗАДАЧИ

Изучить семантику, синтаксис и возможности языка *PHP*. Изучение операторов ветвления в языке *PHP*, работа с оператором *if*, использование оператора *switch*. Реализация функций в языке *PHP*.

# 1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Оператор — это некоторое законченное предложение на языке программирования. Любое выражение, после которого поставлен символ «;», воспринимается интерпретатором, как отдельный оператор. Таким образом, оператор состоит из одного или целого набора выражений, соединенных операциями.

Условный оператор, или оператор ветвления, позволяет пропустить или выполнить некоторый блок кода в зависимости от результата вычисления указанного выражения — условия. Синтаксис условного оператора: «*if* (условие) выражение\_1 *else* выражение\_2». Условие может быть любым выражением. Если оно истинно, то выполняется оператор «выражение\_1». В противном случае выполняется оператор «выражение\_2».

Если в теле оператора *if* используется всего одна инструкция, то заключать ее в фигурные скобки можно, но не обязательно. Однако, если нужно выполнить в теле оператора *if* не одну инструкцию, а несколько, тогда эти несколько инструкций необходимо заключить в фигурные скобки.

Проверка дополнительных условий возможна при помощи оператора *else if*. Конструкция *else if* должна располагаться после оператора *if* и перед оператором *else*. В одном выражении *if* допустимо несколько операторов *else if*. Выражение *else if* выполнится, если предшествующее выражение *if* и предшествующие выражения *else if* в результате будут эквивалентны *false*, а текущий *else if* равен *true*.

*RHP* предоставляет также возможность альтернативного синтаксиса условного оператора — без фигурных скобок, а с применением оператора *endif*.

Тернарный оператор работает почти так же, как и оператор *if*, но при использовании тернарного оператора, вместо ключевых слов используются символы «?» и «:». Тернарный оператор — единственный оператор в *RHP*, который использует три операнда. С помощью тернарного оператора можно записать условие следующим образом: «условие ? выражение\_1 : выражение\_2». Если условие выполняется успешно, то результатом вычислений будет «выражение\_1», в противном случае результатом будет «выражение\_2».

Оператор *switch* подобен серии операторов *if* с одинаковым условием. Оператор *switch* сравнивает значение условного выражения с несколькими значениями. Как правило, в качестве выражения используется переменная, в

зависимости от значения которой должен быть исполнен тот или иной блок кода. Для сравнения в *switch* используется оператор равенства «==».

Структура *switch* передает управление тому из помеченных *case* операторов, для которого значение константного выражения совпадает со значением переключающего выражения. Если значение переключающего выражения не совпадает ни с одним из константных выражений, то выполняется переход к оператору, помеченному меткой *default*. В каждом переключателе может быть не более одной метки *default*, однако она может отсутствовать вообще.

За ключевым словом *case* каждый раз следует значение, после которого должно обязательно стоять двоеточие. Тип значения, указанного после оператора *case*, должен совпадать с типом значения, возвращаемого условием. Перед началом исполнения тела оператора *switch*, переменная, указанная в скобках, должна быть инициализирована каким-нибудь значением, поскольку это значение будет сравниваться со значениями, указанными после *case*. Инструкции, расположенные после *case*, будут исполняться до тех пор, пока не встретится оператор *break*.

Если должен быть выполнен только один блок кода, соответствующий определенному значению, то в конце этого блока следует вставить ключевое слово *break*. Интерпретатор *PHP*, встретив ключевое слово *break*, завершает работу оператора *switch* и переходит к исполнению инструкции, расположенной после закрывающей фигурной скобки оператора *switch*.

Если значение условного выражения не совпало ни с одним из предложенных значений в секциях *case*, оператор *switch* позволяет выполнить некоторые действия по умолчанию. Для этого используется ключевое слово *default*. Оператор *default* обычно указывается в конце тела *switch*, после всех меток. Это обычное место для него, но на самом деле, оператор *default* может быть расположен в любом месте внутри конструкции *switch*.

Функция – это блок кода, к которому можно обращаться из разных частей скрипта. Функции могут иметь входные и выходные параметры. Входные параметры могут использоваться в операциях, которые содержит функция. Выходные параметры устанавливаются функцией, а их значения используются после выполнения функции.

Внутри функции можно использовать любой корректный *PHP*-код, в том числе другие функции. Имена функций следуют тем же правилам, что и другие метки в *PHP*. Корректное имя функции начинается с буквы или знака подчёркивания, за которым следует любое количество букв, цифр или знаков подчёркивания.

### 3 ХОД РАБОТЫ

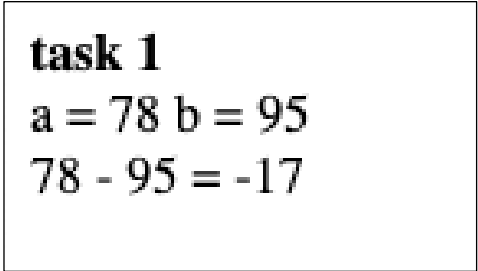
Рассмотрим практическую реализацию операторов, приведенных в цели данной работы. Работа выполняется с помощью редактора кода — *VSCode*.

Объявите две целочисленные переменные *\$a* и *\$b* и зададим им произвольные начальные значения с помощью встроенной функции выбора случайного числа. В зависимости от знаков переменных производятся разные операции. Если оба числа положительные, то выводится их разность, оба отрицательные — произведение, для чисел разных знаков — сумма. На рисунке 3.1 приведен результат работы скрипта. Фрагмент кода:

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

$a = random_int(-100, 100);
$b = random_int(-100, 100);

if ($a >= 0 & $b >= 0) {
    echo "a = $a" . "\t" . "b = $b" . '<br/>';
    echo "$a - $b = " . $a - $b . '<br/>';
} elseif ($a < 0 & $b < 0) {
    echo "a = $a" . "\t" . "b = $b" . '<br/>';
    echo "$a - $b = " . $a * $b . '<br/>';
} elseif (($a < 0 & $b >= 0) / ($a >= 0 & $b < 0)) {
    echo "a = $a" . "\t" . "b = $b" . '<br/>';
    echo "$a - $b = " . $a + $b . '<br/>';
}
```



```
task 1
a = 78 b = 95
78 - 95 = -17
```

Рисунок 3.1 — Вывод переменных и результата операций, указанных в условии

Присвоим переменной *\$a* значение от 0 до 15 и, используя оператор *switch*, выведем значения от *\$a* до 15. На рисунке 3.2 приведён результат. Фрагмент кода:

```
echo "<b> task 2 </b> <br>";

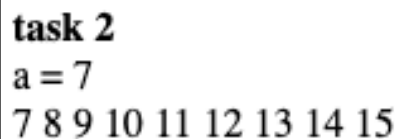
$a = random_int(0, 15);
echo 'a = '. $a . '<br/>';
switch ($a) {
    case 0:
        echo "0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15";
        break;
    case 1:
        echo "1 2 3 4 5 6 7 8 9 10 11 12 13 14 15";
        break;
    case 2:
        echo "2 3 4 5 6 7 8 9 10 11 12 13 14 15";
        break;
    case 3:
        echo "3 4 5 6 7 8 9 10 11 12 13 14 15";
        break;
    case 4:
        echo "4 5 6 7 8 9 10 11 12 13 14 15";
        break;
    case 5:
        echo "5 6 7 8 9 10 11 12 13 14 15";
        break;
    case 6:
        echo "6 7 8 9 10 11 12 13 14 15";
        break;
    case 7:
        echo "7 8 9 10 11 12 13 14 15";
        break;
    case 8:
        echo "8 9 10 11 12 13 14 15";
        break;
    case 9:
        echo "9 10 11 12 13 14 15";
```



```

        break;
    case 10:
        echo "10 11 12 13 14 15";
        break;
    case 11:
        echo "11 12 13 14 15";
        break;
    case 12:
        echo "12 13 14 15";
        break;
    case 13:
        echo "13 14 15";
        break;
    case 14:
        echo "14 15";
        break;
    default:
        echo "15";
        break;
}

```



```

task 2
a = 7
7 8 9 10 11 12 13 14 15

```

Рисунок 3.2 — Вывод значений используя оператор *switch*

В задании 3 необходимо реализовать четыре арифметические операции (сумма, разность, произведение и деление) в виде функций. Функция должна содержать два входных параметра и оператор *return*. Выведем результаты арифметических операций. На рисунке 3.3 отображён вывод. Фрагмент кода для задания 3:

```
echo "<b> task 3 </b> <br>";
```

```
$a = random_int(-100, 100);
```

```

$b = random_int(-100, 100);

function Plus($x, $y)
{
    return ($x + $y);
}
function Minus($x, $y)
{
    return ($x - $y);
}
function Mult($x, $y)
{
    return ($x * $y);
}
function Div($x, $y)
{
    if ($y != 0) {
        return ($x / $y);
    } else {
        return "Devided by 0!!!";
    }
}

echo $a . ' + ' . $b . ' = ' . Plus($a, $b) . '<br/>';
echo $a . ' - ' . $b . ' = ' . Minus($a, $b) . '<br/>';
echo $a . ' * ' . $b . ' = ' . Mult($a, $b) . '<br/>';
echo $a . ' / ' . $b . ' = ' . Div($a, $b) . '<br/>';

echo '<br>';

```

### **task 3**

31 + -86 = -55

31 - -86 = 117

31 \* -86 = -2666

31 / -86 = -0.36046511627907

Рисунок 3.3 — Вывод результата арифметических операций

Далее реализуем функцию *mathOperation* с параметрами *\$arg1*, *\$arg2*, *\$operation*, где *\$arg1*, *\$arg2* — значения аргументов, *\$operation* — строка с названием операции. В зависимости от переданного значения операции требуется выполнять одну из арифметических операций, указанных в задании 3. Выбор функции для переданной операции реализован с помощью оператора *switch*. Параметр *\$operation* передается из *HTML* формы. Результат работы функции показан на рисунке 3.4. Фрагмент кода:

```
// task 4
function mathOperation($arg1, $arg2, $operation)
{
    switch ($operation) {
        case 'plus':
            return Plus($arg1, $arg2);
        case 'minus':
            return Minus($arg1, $arg2);
        case 'mult':
            return Mult($arg1, $arg2);
        case 'div':
            return Div($arg1, $arg2);
        default:
            echo "unknown";
            return ("error");
    }
}

$operation = $_POST['operation'];
echo 'a = ' . $a . '<br/>';
echo 'b = ' . $b . '<br/>';
echo 'operation = ' . $operation . '<br/>';
echo 'answer = ' . mathOperation($a, $b, $operation);
```

<b>task 4</b> a = -76 b = -9 operation = plus answer = -85
--

Рисунок 3.4 — Результат работы функции *mathOperation*

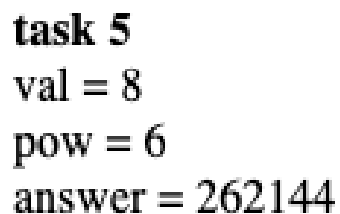
Согласно заданию реализована функция возведения числа в степень с помощью рекурсии. Вывод на экран результата представлен на рисунке 3.5. Фрагмент кода:

```
echo "<b> task 5 </b> <br>";

function power($val, $pow)
{
    if ($pow == 0) {
        return 1;
    }
    if ($pow < 0) {
        return power(1 / $val, -$pow);
    }
    return $val * power($val, $pow - 1);
}

$val = random_int(-10, 10);
$pow = random_int(-10, 10);

echo 'val = ' . $val . '<br/>';
echo 'pow = ' . $pow . '<br/>';
echo 'answer = ' . power($val, $pow);
```



```
task 5
val = 8
pow = 6
answer = 262144
```

Рисунок 3.5 — Вывод возведения в степень

Согласно заданию 6 была реализована функция, которая вычисляет текущее время и возвращает его в формате с правильными склонениями.

Текущее время в языке *PHP* можно узнать с помощью встроенной функции *date*. Вывод результата на экран представлен на рисунке 3.6. Фрагмент кода:

```
$hours = date("H");
$minutes = date("i");

function timeTask($hours, $minutes)
{
    $HourSymbolsCount = $hours % 10;
    if ($HourSymbolsCount == 0 | $HourSymbolsCount >= 5 |
in_array($hours % 100, range(11, 19))) {
        echo $hours . ' часов ';
    } elseif (in_array($HourSymbolsCount, range(2, 4))) {
        echo $hours . ' часа ';
    } elseif ($HourSymbolsCount == 1) {
        echo $hours . ' час ';
    }

    $MinuteSymbolsCount = $minutes % 10;
    if ($MinuteSymbolsCount == 0 | $MinuteSymbolsCount >= 5 |
in_array($minutes % 100, range(11, 19))) {
        echo $minutes . ' минут ';
    } elseif ($MinuteSymbolsCount == 1) {
        echo $minutes . ' минута ';
    } elseif (in_array($MinuteSymbolsCount, range(2, 4))) {
        echo $minutes . ' минуты ';
    }
    return (1);
}

timeTask($hours, $minutes);
```

<b>task 6</b> 19 часов 13 минут
------------------------------------

Рисунок 3.6 — Вывод текущего времени в правильном склонении

## ЗАКЛЮЧЕНИЕ

В результате лабораторной работы мною были изучены операторы ветвления в языке *PHP*. Были рассмотрены особенности работы с условным оператором *if* и с оператором выбора *switch*. Также, был изучен и применен на практике принцип создания функций, вызова функций и передачи параметров в функции в языке *PHP*.