Министерство сельского хозяйства Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования «Дагестанский государственный аграрный университет имени М.М. Джамбулатова»

Аграрно-экономический техникум

Методические указания к практическим занятиям

ОП.ДЭ.01.01 Web-программирование

для студентов специальности СПО

09.02.07 Информационные системы и программирование

Махачкала 2023

История: методические указания к практическим занятиям для студентов специальности СПО 09.02.07 «Информационные системы и программирование».

СОГЛАСОВАНО:

Директор АЭТ подпись Магомедов Д.А.

Одобрено на заседании ПЦК общепрофессиональных, специальных дисциплин "10" марта 2023 г., протокол № 7.

Председатель ПЦК Касимовская О.О..

(подпись) (инициалы, фамилия)

СОГЛАСОВАНО:

Директор Компании Color- IT, Интернет решения Салихов А.Б.

Ф.И.О.

М.П.

САЛИХОВ АХМЕД БАГАУТДИНОВИЧ

СОДЕРЖАНИЕ

Пояснительная записка

Данные методические указания к выполнению практических работ составлены в соответствии с ФГОС по специальности СПО 09.02.03 «Программирование в компьютерных системах» по междисциплинарному курсу «Web-программирование».

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся в ходе освоения междисциплинарного курса должен:

уметь:

- использовать графические программы для создания чертежей структуры web-сайта;

- использовать графические редакторы для обработки изображений, размещаемых на web-сайте;

- использовать язык гипертекстовой разметки HTML и каскадные таблицы стилей CSS для создания web-страниц;

- создавать динамические web-страницы;

- использовать технологию создания web-приложений ASP.NET.

знать:

- технологии создания web-сайта как статичной информационной системы;

- технологии создания web-сайта как динамичной информационной системы;

- теорию использования графики на web-страницах;

- методы обработки и редактирования цифровых изображений;

- программные средства для создания баз данных;

- программные средства, используемые для размещения и сопровождения web-страниц.

Перечень практических работ

№	Тема практической работы
1	Работа с массивами в PHP
2	Работа с конструкциями if-else, switch-case в PHP
3	Работа с циклами foreach, for, while в PHP
4	Работа с математическими функциями в PHP
5	Работа со строковыми функциями в PHP
6	Работа с пользовательскими функциями в PHP
7	Работа с пользовательскими функциями в PHP
8	Установка и настройка CMS WordPress на локальный сервер

ПРАКТИЧЕСКАЯ РАБОТА № 1
тема: «**Работа с массивами в PHP**»

Цель практической работы:

- освоение принципов работы с массивами в сценариях PHP;

- развитие познавательного интереса;

- формирование универсальных учебных действий, связанных с поиском информации, необходимой для решения поставленной задачи.

Оборудование и программное обеспечение: ПК, ОС MS Windows 7 (10), офисный пакет MS Office, браузер, блокнот.

Количество часов: 2 часа**.**

Задание на практическую работу:

1) Дан массив с элементами 'Привет, ', 'мир' и '!'. Необходимо вывести на экран фразу 'Привет, мир!'.

2) Дан массив с элементами 'Привет, ', 'мир' и '!'. Необходимо записать в переменную $text фразу 'Привет, мир!', а затем вывести на экран содержимое этой переменной.

3) Дан массив ['Привет, ', 'мир', '!']. Необходимо записать в первый элемент этого массива слово 'Здравствуй, '.

4) Дан многомерный массив $arr:

$arr = [

'ru'=['голубой', 'красный', 'зеленый'],

'en'=['blue', 'red', 'green'],

];

Вывести с его помощью слово 'голубой'.

5) Создать массив, заполненный числами от 1 до 100. Вычислить сумму элементов данного массива.

6) Создать массив с числами двумя различными способами:

- проверить, что в нем есть элемент со значением 3,

- найти сумму элементов данного массива,

- найти произведение элементов данного массива.

7) Дан массив $arr. С помощью функций array_sum и count найти среднее арифметическое элементов (сумма элементов делить на их количество) данного массива.

8) Оформить отчет по практической работе.

Методика выполнения практической работы

В программировании очень часто возникает задача хранения списка похожих значений, например, всех дней недели или всех месяцев. Можно создать под каждое значение списка

свою переменную, но это очень неудобно и долго - для списка дней недели понадобилось бы 7 переменных, а для месяцев - 12.

Поэтому был изобретен специальный тип данных - массив.

Массив создается с помощью функции [ ]:

```
$a = []; //создание массива $a
?
```

Массив, содержащий дни недели:

```
$a = ['пн', 'вт', 'ср', 'чт', 'пт', 'сб', 'вс'];
?
```

Каждое значение списка, которое записано в массив, называется элементом массива.

Элементы разделяются между собой запятой. После этой запятой можно ставить пробелы, а можно и не ставить.

Названия дней недели представляют собой строки и поэтому взяты в кавычки. Кроме строк в массиве можно хранить числа, без кавычек:

```
//В массиве можно хранить как строки, так и числа:
$a = ['пн', 256, 'ср', 34, 38, 'сб', 95];
?
```

Для того, чтобы PHP вывел все элементы массива, нужно воспользоваться функцией var_dump:

```
$a = ['пн', 'вт', 'ср', 'чт', 'пт', 'сб', 'вс'];
var_dump($a);
?
```

Вывод отдельного элемента массива:

```
$a = ['пн', 'вт', 'ср', 'чт', 'пт', 'сб', 'вс'];
echo $a[2];
?
```

Чтобы обратиться к нужному элементу массива, необходимо указать в квадратных скобках его порядковый номер (нумерация начинается с нуля). Эти порядковые номера называются ключами массива. То есть, значение элемента массива получают по его ключу.

Во всех случаях PHP сам определяет ключи для элементов, но в PHP можно указать ключи в явном виде.

```
$a = [1='пн', 2='вт', 3='ср', 4='чт', 5='пт', 6='сб', 7='вс'];
echo $a[1]; //выведет 'пн' ?
```

Ключи не обязательно должны быть числами, они могут быть и строками.

Пример массива, в котором ключами будут имена работников, а элементами - их зарплаты:

```
//Массив работников:
$a = ['Коля'=200, 'Вася'=300, 'Петя'=400];
?

$a = ['Коля'=200, 'Вася'=300, 'Петя'=400];
echo $a['Вася']; //выведет 300
?
```

Массивы, у которых явно указаны ключи, называются ассоциативными.

На самом деле нет необходимости расставлять ключи всем элементам - достаточно только первому элементу поставить ключ 1. Если у второго элемента не будет ключа, PHP поставит его автоматически, причем следующий по порядку.

```
$a = [1='пн', 'вт', 'ср', 'чт', 'пт', 'сб', 'вс'];
echo $a[3]; //выведет 'ср'?
```

Объявление массива с помощью команды [ ] не является единственным способом его создания.

```
$a[0] = 1;
$a[1] = 2;
$a[2] = 3;
var_dump($a); //с помощью var_dump убеждаемся в том, что $a – это массив
?
```

Ключи могут быть не только числовыми, но и текстовыми:

```
//Ключи могут быть строками:
$a['Коля'] = 100;
$a['Вася'] = 200;
$a['Петя'] = 300;
var_dump($a);
?
```

Кроме того, можно сделать так, что PHP сам добавит ключи (начиная с нуля и так далее). Для этого необходимо оставить квадратные скобки пустыми: $a[] = 1, а PHP сам добавит ключ. Пример:

```
$a[] = 100; //100 будет иметь ключ 0
$a[] = 200; //200 будет иметь ключ 1
$a[] = 300; //300 будет иметь ключ 1
var_dump($a); //массив $a будет иметь вид [0=100, 1=200, 2=300]
?
```

Элементы массива могут быть не только строками и числами, но и массивами. То есть, получится многомерный массив.

Пример массива студентов $students, который будет содержать два подмассива: студенты мужского пола и женского:

```
//Многомерный массив студентов:

$a = [

'boys' = ['Коля', 'Вася', 'Петя'],

'girls' = ['Даша', 'Маша', 'Лена'],

];

?
```

Чтобы вывести какой-либо элемент из многомерного массива следует писать уже не одну пару [ ], а две: $a['boys'][0].

Функция range создает массив с диапазоном элементов. К примеру, можно создать массив, заполненный числами от 1 до 100 или буквами от 'a' до 'z'. Диапазон, который сгенерирует функция, задается параметрами: первый параметр откуда генерировать, а второй - докуда.

Третий необязательный параметр функции задает шаг. К примеру, можно сделать ряд 1, 3, 5, 7, если задать шаг 2, или ряд 1, 4, 7, 10 если задать шаг 3.

```
range(откуда, докуда, [шаг]);
```

Пример массива, заполненного числами от 1 до 5.

```
var_dump(range(1, 5));

?
```

Результат выполнения кода:

```
[1, 2, 3, 4, 5]
```

Пример массива, заполненного числами от 5 до 1.

```
var_dump(range(5, 1));

?
```

Результат выполнения кода:

```
[5, 4, 3, 2, 1]
```

Пример массива, заполненного числами от 0 до 10 с шагом 2.

```
var_dump(range(0, 10, 2));

?
```

Результат выполнения кода:

```
[0, 2, 4, 6, 8, 10]
```

Пример массива, заполненного буквами от 'a' до 'e'.

var_dump(range('a', 'e'));

?

Результат выполнения кода:

['a', 'b', 'c', 'd', 'e']

Функция array_sum вычисляет сумму элементов массива.

array_sum(массив);

Пример вычисления суммы элементов массива.

$arr = [1, 2, 3, 4, 5];

echo array_sum($arr);

?

Результат выполнения кода:

15

Пример вычисления суммы цифр числа.

Для этого необходимо разбить число в массив с помощью str_split и сложить элементы этого массива с помощью array_sum:

$num = 12345;

echo array_sum(str_split($num, 1));

?

Результат выполнения кода:

15

Функция count подсчитывает количество элементов массива.

count(массив);

Пример подсчета количества элементов массива.

$arr = ['a', 'b', 'c', 'd', 'e'];

echo count($arr);

?

Результат выполнения кода:

5

Функция in_array проверяет наличие заданного элемента в массиве. Первым параметром она принимает что искать, а вторым - в каком массиве.

in_array(что искать, в каком массиве);

Пример проверки, есть ли в массиве $arr элемент со значением 3.

$arr = ['a', 'b', 'c', 'd', 'e'];

$result = in_array('c', $arr);

var_dump($result);

?

Результат выполнения кода:

true

Функция str_split разбивает строку в массив. Первым параметром она принимает строку, а вторым - количество символов в элементе массива. К примеру, если второй параметр задать как 3 - функция разобьет строку в массив так, чтобы в каждом элементе массива было по 3 символа.

str_split(строка, количество символов в элементе массива);

Пример разбиения строки по 2 символа в элементе массива.

$str = 'abcde';

$arr = str_split($str, 2);

var_dump($arr);

?

Результат выполнения кода:

['ab', 'cd', 'e'];

Пример разбиения строки по 3 символа в элементе массива.

$str = 'abcdefg';

$arr = str_split($str, 3);

var_dump($arr);

?

Результат выполнения кода:

['abc', 'def', 'g'];

Пример вычисления суммы цифр числа. Для этого необходимо разбить число в массив с помощью str_split и сложить элементы этого массива с помощью array_sum.

$num = 12345;

echo array_sum(str_split($num, 1));?

Результат выполнения кода:

15

Функция array_product вычисляет произведение (умножение) элементов массива.

array_product(массив);

Пример вычисления произведения элементов массива.

```
$arr = [1, 2, 3, 4, 5];
echo array_product($arr);
?
```

Результат выполнения кода:

120

Структура отчета:

1) название, цель практической работы;
2) задание на практическую работу;
3) скрипты решения задач;
4) ответы на контрольные вопросы.

Контрольные вопросы:

1) Что такое массив в PHP?
2) Что такое ассоциативный массив?
3) Перечислить способы объявления массива.
4) Для чего используется функция str_split в PHP?

ПРАКТИЧЕСКАЯ РАБОТА № 2
тема: «**Работа с конструкциями if-else, switch-case в PHP**»

Цель практической работы:

- освоение принципов использования управляющих конструкций при составлении PHP-скриптов;

- развитие познавательного интереса;

- формирование универсальных учебных действий, связанных с поиском информации, необходимой для решения поставленной задачи.

Оборудование и программное обеспечение: ПК, ОС MS Windows 7 (10), офисный пакет MS Office, браузер, блокнот.

Количество часов: 2 часа**.**

Задание на практическую работу:

1) Дано число 15. Если оно больше 10, то увеличить его на 100, иначе уменьшить на 30.

2) Дано число. Если оно не меньше 50, то вывести квадрат этого числа, если же это число больше 10 и меньше 30, то вывести нуль, в остальных случаях вывести слово «Ошибка».

3) Переменная $num может принимать одно из значений: 1, 2, 3 или 4. Если она имеет значение '1', то в переменную $result записать 'зима', если имеет значение '2' – 'лето' и так далее. Решить задачу через switch-case.

4) В переменной $day лежит некое число из интервала от 1 до 31. Определить в какую декаду месяца попадает это число (в первую, вторую или третью).

5) Дана строка с символами, например, 'abcde'. Проверить, что первым символом этой строки является буква 'a'. Если это так – вывести 'да', в противном случае вывести 'нет'.

6) Дана строка с цифрами, например, '12345'. Проверить, что первым символом этой строки является цифра 1, 2 или 3. Если это так – вывести 'да', в противном случае вывести 'нет'.

7) Дана строка из 6-ти цифр. Проверить, что сумма первых трех цифр равняется сумме вторых трех цифр. Если это так – вывести 'да', в противном случае вывести 'нет'.

8) Оформить отчет по практической работе.

Методика выполнения практической работы

Любой сценарий PHP сформирован из ряда конструкций. Конструкцией могут быть операторы, функции, циклы, условные операторы. Конструкции обычно заканчиваются точкой с запятой. Кроме того, конструкции могут быть сгруппированы в группу, формируя группу конструкций с изогнутыми фигурными скобками {...}. Группа конструкций - это также отдельная конструкция.

Условные операторы являются, пожалуй, наиболее распространенными конструкциями во всех алгоритмических языках программирования.

1. Конструкция IF

Синтаксис конструкции if аналогичен конструкции if в языке Си:

```
if (логическое выражение) оператор;
?
```

Согласно выражениям PHP, конструкция if содержит логическое выражение. Если логическое выражение истинно (true), то оператор, следующий за конструкцией if будет исполнен, а если логическое выражение ложно (false), то следующий за if оператор исполнен не будет.

```
if ($a  $b) echo "значение a больше, чем b";
?
```

В следующем примере если переменная $a не равна нулю, будет выведена строка "значение a истинно (true)":

```
if ($a) echo "значение a истинно (true) ";
?
```

В следующем примере если переменная $a равна нулю, будет выведена строка "значение a ложно (false)":

```
if (!$a) echo "значение a ложно (false) ";
?
```

Часто будет необходим блок операторов, который будет выполняться при определенном условном критерии, тогда эти операторы необходимо поместить в фигурные скобки {...}

```
if ($a  $b) {
    echo "a больше b";
    $b = $a;}
?
```

Приведенный пример выведет сообщение, "a больше b", если $a $b, а затем переменная $a будет приравнена к переменной $b. Данные операторы выполняются в теле конструкции if.

2. Конструкция ELSE

Часто возникает потребность исполнения операторов не только в теле конструкции if, если выполнено какое-либо условие конструкции if, но и в случае, если условие конструкции if не выполнено. В данной ситуации нельзя обойтись без конструкции else. В целом, такая конструкция будет называться конструкцией if-else.

Синтаксис конструкции if-else такой:

```
if (логическое_выражение)
инструкция_1;
else
инструкция_2;
```

Действие конструкции if-else следующее: если логическое_выражение истинно, то выполняется инструкция_1, а иначе — инструкция_2. Как и в любом другом языке, конструкция else может опускаться, в этом случае при получении должного значения просто ничего не делается.

Если инструкция_1 или инструкция_2 должны состоять из нескольких команд, то они, как всегда, заключаются в фигурные скобки. Например:

```
if ($a  $b) {
    echo "a больше, чем b";
} else {
    echo "a НЕ больше, чем b";}
?
```

3. Конструкция ELSEIF

elseif - это комбинация конструкций if и else. Эта конструкция расширяет условную конструкцию if-else.

Синтаксис конструкции elseif:

```
if (логическое_выражение_1)
оператор_1;
elseif (логическое_выражение_2)
оператор_2;
else
оператор_3;
```

Практический пример использования конструкции elseif:

```
if ($a  $b) {
    echo "a больше, чем b";
} elseif ($a == $b) {
    echo "a равен b";
} else {
    echo "a меньше, чем b";}
?
```

4. Конструкция SWITCH-CASE

Часто вместо нескольких расположенных подряд инструкций if-else целесообразно воспользоваться специальной конструкцией выбора switch-case. Данная конструкция предназначена для выбора действий, в зависимости от значения указанного выражения. Конструкция switch-case чем-то напоминает конструкцию if-else, которая, по сути, является ее аналогом. Конструкцию выбора можно использовать, если предполагаемых вариантов много, например, более 5, и для каждого варианта нужно выполнить специфические действия. В таком случае, использование конструкции if-else становится действительно неудобным.

Синтаксис конструкции switch-case:

```
switch(выражение) {
case значение1: команды1; [break;]
case значение2: команды2; [break;]
. . .
case значениеN: командыN; [break;]
[default: команды_по_умолчанию; [break]]
}
```

Принцип работы конструкции switch-case:

- вычисляется значение выражения;

- просматривается набор значений. Пусть значение1 равно значению выражения, вычисленного на первом шаге. Если не указана конструкция (оператор) break, то будут выполнены команды i, i+1, i+2, ... , N. В противном случае (есть break) будет выполнена только команда с номером i;

- если ни одно значение из набора не совпало со значением выражения, тогда выполняется блок default, если он указан.

Примеры использования конструкции switch-case:

```
$x=1;
// Конструкция if-else
if ($x == 0) {
   echo "x=0
";
} elseif ($x == 1) {
   echo "x=1
";
} elseif ($x == 2) {
   echo "x=2
";
}
// Конструкция  switch-case
switch ($x) {
case 0:
   echo "x=0
";
   break;
case 1:
   echo "x=1
";
   break;
case 2:
   echo "x=2
";
   break;
}
?
```

Рассмотренный сценарий выводит x=1 дважды. Еще пример использования конструкции switch-case:

```
$x="Яблоко";
switch ($x) {
case "Яблоко":
   echo "Это Яблоко";
   break;
case "Груша":
   echo "Это Груша";
   break;
case "Арбуз":
   echo "Это Арбуз";
   break;
}
?
```

Данный скрипт выводит "Это Яблоко".

Конструкция switch выполняется поэтапно. Изначально никакой код не исполнен. Только, когда конструкция case найдена со значением, которое соответствует значению выражения switch, PHP начинает исполнять конструкции. PHP продолжает исполнять конструкции до конца блока switch, пока не встречается оператор break. Если не использовать конструкции (операторы) break, скрипт будет выглядеть так:

```
$x=0;
switch ($x) {
case 0:
   echo "x=0
";
case 1:
   echo "x=1
";
case 2:
   echo "x=2
";
}
// Без использования break выводит
// x=0
// x=1
// x=2
?
```

Операторный список для case может быть также пуст, он просто передает управление в операторный список до следующей конструкции case:

```
switch ($x) {
case 0:
case 1:
case 2:
    echo "x меньше, чем 3, но не отрицателен";
    break;
case 3:
    echo "x=3";
}
?
```

Когда ни одно значение из набора не совпало со значением выражения, тогда выполняется блок default, если он указан, например:

```
$x=3;
switch ($x) {
case 0:
   echo "x=0";
   break;
case 1:
   echo "x=1";
   break;
case 2:
   echo "x=2";
   break;
default:
   echo "x не равен 0, 1 или 2";
}
?
```

Данный скрипт выводит "x не равен 0, 1 или 2", поскольку переменная $x=3.

Структура отчета:

1) название, цель практической работы;

2) задание на практическую работу;

3) скрипты решения задач.

ПРАКТИЧЕСКАЯ РАБОТА № 3
тема: «**Работа с циклами foreach, for, while в PHP**»

Цель практической работы:

- освоение принципов использования управляющих конструкций при составлении PHP-скриптов;

- развитие познавательного интереса;

- формирование универсальных учебных действий, связанных с поиском информации, необходимой для решения поставленной задачи.

Оборудование и программное обеспечение: ПК, ОС MS Windows 7 (10), офисный пакет MS Office, браузер, блокнот.

Количество часов: 2 часа**.**

Задание на практическую работу:

1) Дан массив с элементами 1, 2, 3, 4, 5. С помощью цикла foreach вычислить сумму элементов этого массива. Записать ее в переменную $result.

2) Дан массив с элементами 1, 2, 3, 4, 5. С помощью цикла foreach вычислить сумму квадратов элементов этого массива. Результат записать в переменную $result.

3) Дан массив $arr. С помощью цикла foreach отобразить на экране столбец ключей и элементов в формате 'green - зеленый'.

$arr = ['green'='зеленый', 'red'='красный','blue'='голубой'];

4) Дан массив $arr с ключами 'Коля', 'Вася', 'Петя' и с элементами '200', '300', '400'. С помощью цикла foreach отобразить на экране столбец строк такого формата: 'Коля - зарплата 200 долларов.'.

5) Вывести столбец чисел от 11 до 33.

6) С помощью цикла вычислить сумму чисел от 1 до 100.

7) Дан массив с элементами 2, 5, 9, 15, 0, 4. С помощью цикла foreach и оператора if вывести на экран столбец тех элементов массива, которые больше 3-х, но меньше 10.

8) Дан массив с числами. Числа могут быть положительными и отрицательными. Вычислить сумму положительных элементов этого массива.

9) Дан массив с элементами 1, 2, 5, 9, 4, 13, 4, 10. С помощью цикла foreach и оператора if проверить есть ли в массиве элемент со значением, равным 4. Если есть - вывести на экран 'Есть!' и выйти из цикла.

10) Дан массив числами, например: ['10', '20', '30', '50', '235', '3000']. Вывести на экран только те числа из массива, которые начинаются на цифру 1, 2 или 5.

11) Оформить отчет по практической работе.

Методика выполнения практической работы

Циклы позволяют повторять определенное количество раз различные операторы. Данные операторы называются телом цикла. Проход цикла называется итерацией.

PHP поддерживает следующие виды циклов:

1) Цикл с предусловием (while);

2) Цикл с постусловием (do-while);

3) Цикл со счетчиком (for);

4) Специальный цикл перебора массивов (foreach).

При использовании циклов есть возможность использования операторов break и continue. Первый из них прерывает работу всего цикла, а второй - только текущей итерации.

1. Цикл с предусловием while

Цикл с предусловием while работает по следующим принципам:

- вычисляется значение логического выражения;

- если значение истинно, выполняется тело цикла, в противном случае - переходит на следующий за циклом оператор.

Синтаксис цикла с предусловием:

While (логическое_выражение)

инструкция;

В данном случае телом цикла является инструкция. Обычно тело цикла состоит из большого числа операторов. Пример цикла с предусловием while:

$x=0;

while ($x++

?

Сначала проверяется условие, а только потом увеличивается значение переменной. Если поставить операцию инкремента перед переменной (++$x

$x=0;

while ($x

{

$x++; // увеличение счетчика

echo $x;

}

?

Если увеличить счетчик после выполнения оператора echo, будет получена строка 0123456789.

Подобно конструкции условного оператора if, можно группировать операторы внутри тела цикла while, используя следующий альтернативный синтаксис:

while (логическое_выражение):

инструкция;

...

endwhile;

Пример использования альтернативного синтаксиса:

```
$x = 1;
while ($x
echo $x;
$x++;
endwhile;
?
```

2. Цикл с постусловием do while

В отличие от цикла while, этот цикл проверяет значение выражения не до, а после каждого прохода (итерации). Таким образом, тело цикла выполняется хотя бы один раз. Синтаксис цикла с постусловием такой:

```
do
{
тело_цикла;
}
while (логическое_выражение);
```

После очередной итерации проверяется, истинно ли логическое_выражение, и, если это так, управление передается вновь на начало цикла, в противном случае цикл обрывается.

Альтернативного синтаксиса для do-while разработчики PHP не предусмотрели (этот цикл довольно редко используется при программировании web-приложений).

Пример скрипта, показывающего работу цикла с постусловием do-while:

```
$x = 1;
do {
echo $x;
} while ($x++
?
```

Рассмотренный сценарий выводит: 12345678910

3. Цикл со счетчиком for

Цикл со счетчиком используется для выполнения тела цикла определенное число раз.

Синтаксис цикла for:

```
for (инициализирующие_команды; условие_цикла; команды_после_итерации) { тело_цикла; }
```

Цикл for начинает свою работу с выполнения инициализирующих_команд. Данные команды выполняются только один раз. После этого проверяется условие_цикла, если оно истинно (true), то выполняется тело_цикла. После того, как будет выполнен последний оператор тела, выполняются команды_после_итерации. Затем снова проверяется условие_цикла. Если оно истинно (true), выполняется тело_цикла и команды_после_итерации и так далее.

for ($x=0; $x

?

Данный сценарий выводит: 0123456789

Есть вариант вывода строки 12345678910:

for ($x=0; $x++

?

В данном примере обеспечено увеличение счетчика при проверке логического выражения.

Если необходимо указать несколько команд, их можно разделить запятыми, пример:

for ($x=0, $y=0; $x

4. Цикл перебора массивов foreach

В PHP4 появился еще один специальный тип цикла - foreach. Данный цикл предназначен специально для перебора массивов.

Синтаксис цикла foreach выглядит следующим образом:

foreach (массив as $ключ=$значение)

команды;

Здесь команды циклически выполняются для каждого элемента массива, при этом очередная пара ключ=значение оказывается в переменных $ключ и $значение.

Пример работы цикла foreach:

$names["Иванов"] = "Андрей";

$names["Петров"] = "Борис";

$names["Волков"] = "Сергей";

$names["Макаров"] = "Федор";

foreach ($names as $key = $value) {

echo "**$value $key**
";

}

?

Рассмотренный сценарий выводит:

Андрей Иванов

Борис Петров

Сергей Волков

Федор Макаров

У цикла foreach имеется и другая форма записи, которую следует применять, когда не важно значение ключа очередного элемента:

foreach (массив as $значение)

команды;

В этом случае доступно лишь значение очередного элемента массива, но не его ключ. Это может быть полезно, например, для работы с массивами-списками:

$names[] = "Андрей";

$names[] = "Борис";

$names[] = "Сергей";

$names[] = "Федор";

foreach ($names as $value) {

echo "**$value**
";

}

?

Внимание: Цикл foreach оперирует не исходным массивом, а его копией. Это означает, что любые изменения, которые вносятся в массив, не могут быть "видны" из тела цикла. Что позволяет, например, в качестве массива использовать не только переменную, но и результат работы какой-нибудь функции, возвращающей массив (в этом случае функция будет вызвана всего один раз - до начала цикла, а затем работа будет производиться с копией возвращенного значения).

5. Конструкция break

Очень часто для того, чтобы упростить логику какого-нибудь сложного цикла, удобно иметь возможность его прервать в ходе очередной итерации (к примеру, при выполнении какого-нибудь особенного условия). Для этого и существует конструкция break, которая осуществляет немедленный выход из цикла. Она может задаваться с одним необязательным параметром - числом, которое указывает, из какого вложенного цикла должен быть произведен выход. По умолчанию используется 1, т.е. выход из текущего цикла, но иногда применяются и другие значения.

$x=0;

while ($x++

if ($x==3) break;

echo "bИтерация $xbbr";

```
}
// Когда $x равен 3, цикл прерывается
?
```

Рассмотренный сценарий выводит:

Итерация 1

Итерация 2

Примеры решения задач

Задача 1. Дан массив с элементами 'html', 'css', 'php', 'js', 'jq'. С помощью цикла foreach вывести эти слова в столбик.

Решение: Для начала необходимо создать сам массив.

```
php
$arr = ['html', 'css', 'php', 'js', 'jq'];
?
```

Теперь необходимо воспользоваться циклом foreach:

```
php
$arr = ['html', 'css', 'php', 'js', 'jq'];
foreach ($arr as $elem) {
echo $elem.'
';
}
?
```

Задача 2. Дан массив с элементами 10, 20, 15, 17, 24, 35. Вычислить сумму элементов этого массива. Записать ее в переменную $result.

Решение:

```
$arr = [10, 20, 15, 17, 24, 35];
$result = 0;
foreach ($arr as $elem) {
$result = $result + $elem;
}
echo $result;
?
```

Задача 3. Вывести столбец чисел от 1 до 100.

Решение: Задачу можно решить как циклом for, так и циклом while.

Через цикл while:

```
php
$i = 1; //счетчик цикла устанавливается в 1
while ($i
echo $i.'
'; //выведет 1,2... 100
$i++;
}
?
```

Через цикл for:

```
php
for ($i = 1; $i
echo $i.'
'; //выведет 1,2... 100
}
?
```

Структура отчета:

1) название, цель практической работы;

2) задание на практическую работу;

3) скрипты решения задач;

4) ответы на контрольные вопросы.

Контрольные вопросы:

1) Дать определение понятиям «Тело цикла», «Итерация».

2) Какие три вида циклов поддерживает PHP?

3) Записать синтаксис цикла с предусловием.

4) Записать синтаксис цикла с постусловием.

ПРАКТИЧЕСКАЯ РАБОТА № 4
тема: «**Работа с математическими функциями в PHP**»

Цель практической работы:

- освоение принципов использования математических функций при составлении PHP-скриптов;

- развитие познавательного интереса;

- формирование универсальных учебных действий, связанных с поиском информации, необходимой для решения поставленной задачи.

Оборудование и программное обеспечение: ПК, ОС MS Windows 7 (10), офисный пакет MS Office, браузер, блокнот.

Количество часов: 2 часа**.**

Задание на практическую работу:

1) Даны переменные $a и $b. Проверить, что $a делится без остатка на $b. Если это так - вывести 'Делится' и результат деления, иначе вывести 'Делится с остатком' и остаток от деления.

2) Даны переменные $a и $b. Найти модуль разности $a и $b. Проверить работу скрипта для различных $a и $b.

3) Дан массив с числами. Создать из него новый массив так, чтобы отрицательные числа стали положительными.

4) Возвести 2 в 8 степень. Результат записать в переменную $st.

5) Вычислить квадратный корень из 255.

6) Дан массив с элементами 7, 8, 10, 1, 3, 1, 4. Вычислить корень из суммы квадратов его элементов. Для решения использовать цикл foreach.

7) Найти квадратный корень из 427. Результат округлить до целых, до десятых, до сотых.

8) Найти квадратный корень из 348. Округлить результат в большую и меньшую сторону, записать результаты округления в ассоциативный массив с ключами 'floor' и 'ceil'.

9) Даны числа 7, -12, 5, 20, -140, 1, 20. Определить минимальное и максимальное число.

10) Оформить отчет по практической работе.

Методика выполнения практической работы

К основным математическим операторам PHP относятся следующие.

+ – сумма двух чисел;

- – разность двух чисел;

* – произведение двух чисел;

/ – частное от деления двух чисел;

% – остаток от деления одного числа на другое (частное по модулю).

Ниже представлен пример использования математических операторов сложения и вычитания в языке PHP.

Пример использования математических операторов "+" и "-"

Математические операторы "+" и "-"

```
        echo "5 + 2=", 5 + 2, "
";
        echo "5 - 2 =", 5 - 2, "
";
    ?
```