Базовый курс PHP.
Основные понятия, оператор ветвления, циклы и массивы.

Установка окружения

XAMPP полностью бесплатный и простой в установке дистрибутив Apache, содержащий MariaDB, MySQL, PHP и Perl. XAMPP создан с открытым исходным кодом, чтобы быть невероятно простым в установке и в использовании.

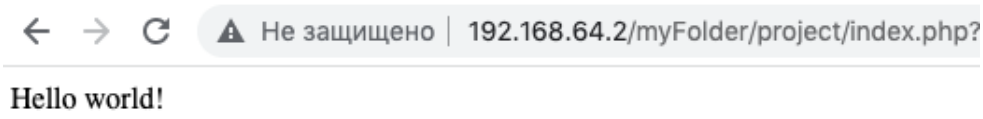
Ссылка для установки: https://www.apachefriends.org/ru/index.html

Начало работы

Давайте напишем самую первую программу на PHP. Для этого нужно зайти в корневую директорию нашего проекта и создать там файл с названием index.php. Напишем в файле следующие строки:

```
<?php
echo "Hello world!";
?>
```

Чтобы посмотреть результат работы кода откроем браузер и наберём в адресной строке http//localhost/папка_проекта (стандартный путь: opt/lampp/htdocs/папка_проекта), чтобы убедиться, что всё отработало как следует. На экране браузера вы увидите примерно следующее:



Любому PHP сценарию для начала работы требуется дескриптор <?php, который даёт понять интерпретатору, что дальше идёт PHP-код. Можно использовать сокращённый синтаксис <?, но, согласно современным стандартам кода, этого делать не рекомендуется.

Далее мы видим команду echo, единственное назначение которой – вывод на экран того, что передали ей в строчке дальше. Для нас это строка, заключённая в кавычки – «Hello, World!». Каждая строка команды заканчивается символом «;», отделяющим смысловой блок команды от других команд и блоков. В нашем примере была только одна инструкция, и точку с запятой можно опустить. Но хорошим стилем в PHP считается всегда её ставить. Это указывает на то, что команда завершена. Завершение программы обозначается с помощью «?>», но это необязательно.

Простейшие операции

Операции со строками

Можно объединять несколько строковых переменных в одну. Такая операция называется конкатенация.

```
<?php
$a = "Hello";
$b = "World";
echo $a . " " . $b;
?>
```

Конкатенация в языке PHP выполняется с помощью символа «.». Кроме того, при выполнении данной операции все переменные других типов, если это возможно, будут приведены к строковому типу. Обратите внимание, что, согласно стандартам при конкатенации двух строк, справа и слева от точки должны быть пробелы.

Математические операции

Вполне резонно, что в PHP существуют стандартные математические операции.

```
$a = 4;
$b = 5;
echo $a + $b . "; // сложение
echo $a * $b . "; // умножение
echo $a - $b . "; // вычитание
echo $a / $b . "; // деление
echo $a % $b . "; // остаток от деления
echo $a ** $b . "; // возведение в степень
```

Операторы if, if-else

Для реализации ветвления в PHP используется оператор if.

```
<?php
if( Условие ) {
Действие;
}
?>
```

Условие – это любое выражение, возвращающее булевское значение (true, false), т.е. такой вопрос, на который ответить можно только двумя способами: либо да, либо нет. Действие выполняется тогда, когда условие истинно (true). Обычно условием является операция сравнения, либо несколько таких операций, объединённых логическими связками (И, ИЛИ). В результате проверки какого-либо условия может выполняться сразу несколько операторов:

```
<?php
if( Условие ) {
Действие 1;
Действие 2;
}
?>
```

Что делать, если одного условия недостаточно? Рассмотрим пример ветвления, где в случае истины мы выполним одно действие, а в случае лжи – другое:

```
<?php
if( Условие ) {
Действие1;
}
else {
Действие2;
}
?>
```

Попробуем реализовать простой пример:

```
<?php
$x = 5;
$y = 42;
if( $x > $y )
echo $x + $y;
else
echo $x * $y;
?>
```

Обратите внимание, если по условию нужно выполнять всего один оператор, фигурные скобки можно не ставить.

Оператор switch

Теперь представим ситуацию, когда нужно разделить программу не на 2 или 3 варианта, а на большее количество. Конечно, можно много раз использовать конструкцию else if, но это способно привести к серьёзному ухудшению читаемости кода. Поэтому существует специальный оператор выбора из нескольких вариантов – switch. Он имеет следующий синтаксис:

```
<?php
switch(переменная) {
case Значение1: Действие1;
break;
case Значение2: Действие2;
break;
default:
Действие3;
}
?>
```

Оператор switch смотрит на значение переменной (вместо неё также может стоять выражение, возвращающее значение) и сравнивает его с предложенными вариантами. В случае совпадения выполняется соответствующий блок кода. Если же после прохода по всем вариантам совпадения так и не обнаружилось, выполняются операторы из блока default. Это необязательный блок, который может отсутствовать.

Давайте создадим функцию, которая будет сравнивать числа:

```
<?php
function compare_numbers($x, $y) {
if ($x > $y)
echo "$x > $y";
else if ($x < $y)
echo "$x < $y";
else
echo "$x = $y";
}
?>
```

Имея такую функцию в арсенале, можно сравнивать сколько угодно пар чисел, не задумываясь о процессе сравнения.

```
<?php
compare_numbers(10, 20);
compare_numbers(20, 10);
compare_numbers(20, 20);
?>
```

Понятие цикла

Цикл – разновидность управляющей конструкции в высокоуровневых языках программирования, предназначенная для организации многократного исполнения набора инструкций. Говоря простым языком, цикл – это конструкция, заставляющая определённую группу действий повторяться до наступления нужного условия.

Каждая итерация состоит из следующих шагов:

1. Инициализация переменных цикла.
2. Проверка условия выхода.
3. Исполнение тела цикла.
4. Обновление счётчика итераций.

Циклы в PHP следуют всем вышеобозначенным правилам и делятся на несколько типов:

1. Цикл с предусловием – цикл, который выполняется, пока истинно некоторое условие, указанное перед его началом. Поскольку условие проверяется до выполнения самой первой итерации, вполне может не выполниться ни одной итерации, если условие изначально ложно.

2. Цикл с постусловием – цикл, в котором условие проверяется после выполнения тела цикла. Отсюда следует, что тело всегда выполняется хотя бы один раз.

3. Цикл со счётчиком – цикл, в котором счётчик итераций изменяет своё значение от заданного начального значения до конечного значения с некоторым шагом. Для каждого значения счётчика тело цикла выполняется один раз.

4. Совместный цикл – цикл, задающий выполнение некоторой операции для объектов из заданного множества без явного указания порядка перечисления этих объектов. Говоря проще, этот цикл проходит по всем элементам переданного множества.

Циклы while и do…while

Цикл while является примером цикла с предусловием. Выглядит он следующим образом:

```
<?php
while( Условие ) {
Операторы;
}
?>
```

Например,

```
<?php
$n = 10;
$i = 1;
while ($i <= $n) {
echo "$i";
$i++;
}
?>
```

Цикл do…while – это уже цикл с постусловием, работающий по алгоритму:

1. Выполнение блока операторов.
2. Проверка условия.
3. Выход, если условие ложно.

```
<?php
$n = 10;
$i = 1;
do {
echo "$i";
$i++;
}
while ($i <= $n)
?>
```

Цикл do...while используют достаточно редко ввиду его громоздкости и плохой читаемости.

Цикл FOR

Цикл for относится к третьему типу циклов – это цикл со счётчиком. Является классическим примером красивой организации кода и имеет наиболее схожий вид в большинстве языков программирования. В PHP он пришёл из языка C. Конструкция позволяет одной строкой полностью определить поведение цикла.

```
<?php
for (выражение1; выражение2; выражение3) {
операторы;
}
?>
```

Выражение1 вычисляется перед началом цикла. Обычно в нем инициализируется управляющая переменная. Выражение2 вычисляется в начале каждой итерации цикла. Это выражение ведет также, как условие цикла while, если значением Выражения2 оказывается true, цикл продолжается, иначе – останавливается. Выражение3 вычисляется в конце каждой итерации и, как правило, используется для изменения значения управляющей переменной цикла.

И снова в цикле выведем числа от 1 до 10, но уже с применением цикла for:

```
<?php
$n = 10;
for ($i = 1; $i <= $n; $i++) {
echo "$i";
}
?>
```

Массивы

В общепринятом значении массив – это именованный набор однотипных переменных. Однако, в языке PHP массив устроен чуть сложнее, чем просто набор элементов – это хэш-таблица, хранящая пары ключ-значение.

В PHP в роли ключа элемента массива может храниться не только число, но и строка. Такие массивы называются ассоциативными и являются одной из отличительных черт языка PHP. Массивы в языке PHP создаются следующим образом:

```
<?php
$myArray = [];
?>
```

В языках со строгой типизацией данных мы можем создать массив строк, массив чисел, массив объектов одинакового типа и так далее. PHP – язык с динамической типизацией, поэтому в каждую ячейку массива мы можем записать переменную любого типа. Давайте наполним наш массив.

```
<?php
$myArray = [];
$someVar = true;
$myArray[] = 1;
$myArray[] = 'Hello, world!';
$myArray[] = $someVar;
var_dump($myArray);
?>
```

К конкретному элементу можно обратиться по его индексу (ключу).

```
<?php
$myArray[] = 'Hello, world!';
echo $myArray[1];
?>
```

Разумеется, в ячейки массива можно класть не только простые переменные, но и другие массивы. Создадим индексный массив, содержащий в каждой своей ячейке по ассоциативному массиву.

```
<?php
$users = [];
```

```
$users[0] = [
    'name' => 'Alex',
    'email' => 'alex@example.com'
];
$users[1] = [
    'name' => 'George',
    'email' => 'george@domain.ru',
    'additionalData' => 'Всем привет!'
];
$users[3] = [
    'name' => 'Michael',
    'email' => 'mich@test.com'
];
?>
```

Применение циклов для работы с массивами

Если мы хотим обработать каждый элемент массива, то совершаем так называемый обход массива в цикле. Самый простой способ обойти массив – использовать цикл for:

```
<?php
for ($i = 0; $i < count($arr); $i++) {
…
}
?>
```

Всё очень здорово ровно до тех пор, пока мы не начали работу с ассоциативными массивами. Тогда числовой счётчик становится неприменим. Для обхода массивов произвольных типов существует специальный оператор цикла – foreach

```
<?php
foreach (массив as [$key =>] $value) {
операторы
}
?>
```

Оператор foreach проходит каждый элемент массива по одному разу. В каждом проходе в переменную $key помещается индекс этого элемента, а в переменную $value – значение. Имена этих двух переменных произвольны. Однако элемента $key может и не быть. Иногда достаточно работы только со значением.

```
<?php
foreach ($myArray as $my_key => $my_value) {
echo ("$my_key is $my_value ");
}
?>
```

Задание

1. Установить программное обеспечение: веб-сервер, базу данных, интерпретатор, текстовый редактор и проверить, что всё работает правильно.

2. Реализовать основные 4 арифметические операции в виде функций с двумя параметрами.

3. Реализовать функцию с тремя параметрами: function mathOperation($arg1, $arg2, $operation), где $arg1, $arg2 – значения аргументов, $operation – строка с названием операции. В зависимости от переданного значения операции выполнить одну из арифметических операций (использовать функции из пункта 3) и вернуть полученное значение (использовать switch).

4. С помощью циклов while и for вывести все числа в промежутке от 0 до 100, которые делятся на 3 без остатка.

5. Дан массив с элементами 'html', 'css', 'php', 'js', 'mysql'. С помощью цикла foreach выведите эти слова в столбик.