http://www.intuit.ru/studies/courses/42/42/info

Лабораторная работа №1: Введение в PHP

Первая PHP-программа

Рассмотрим пример.

```
<html>
  <head>
  <title>Пример</title>
  </head>
  <body>
  <?php
  echo "<p>Привет, я - скрипт PHP!</p>";
  ?>
</body>
</html>
```

Пример 1. Простой html-файл со встроенным кодом на PHP

Это простой HTML-файл, в который встроен с помощью специальных тегов код, написанный на языке PHP.

PHP похож на Си и Perl. Однако приведенная здесь программа сильно отличается от аналогичных по смыслу программ на языках Си и Perl. Здесь не нужно писать кучу специальных команд для вывода HTML. Пишется непосредственно код на HTML, в который можно встраивать PHP -код, осуществляющий какие-либо действия (например, выводящий текст на экран, как в нашем примере). Недостатком PHP по сравнению с Си и Perl, несмотря на все усилия разработчиков, все еще является недостаточная быстрота выполнения сложных скриптов.

PHP-скрипты – это программы, которые выполняются и обрабатываются сервером. Так что сравнивать его со скриптовыми языками типа JavaScript невозможно, потому что написанные на них скрипты выполняются на машине клиента. В чем отличие скриптов, выполняемых на сервере и на клиенте? Если скрипт обрабатывается сервером, клиенту посылаются только результаты работы скрипта . Например, если на сервере выполнялся скрипт, подобный приведенному выше, клиент получит сгенерированную HTML-страницу вида:

```
<html>
    <head>
    <title>Пример</title>
    </head>
    <body>
    <p>Привет, я - скрипт PHP!</p>
    </body>
</html>
```

В этом случае клиент не знает, какой код выполняется. Можно даже сконфигурировать свой сервер таким образом, чтобы HTML-файлы обрабатывались процессором PHP, так что клиенты даже не смогут узнать, получают ли они обычный HTML-файл или результат выполнения скрипта. Если же скрипт обрабатывается клиентом (например, это программа на языке JavaScript), то клиент получает страницу, содержащую код скрипта.

Мы отмечали выше, что PHP - скрипты встраиваются в HTML-код. Возникает вопрос, каким образом? Есть несколько способов. Один из них приведен в самом первом примере – с помощью открывающего тега <?php и закрывающего тега ?> . Такого вида специальные теги позволяют переключаться между режимами HTML и PHP. Этот синтаксис наиболее предпочтителен, поскольку позволяет задействовать PHP в XML - совместимых программах (например, написанных на языке XHTML).

Когда PHP обрабатывает файл, он просто передает его текст, пока не встретит один из перечисленных специальных тегов, который сообщает ему о необходимости начать интерпретацию текста как кода PHP. Затем он выполняет весь найденный код до закрывающего тега, говорящего интерпретатору, что далее снова идет просто текст. Этот механизм позволяет внедрять PHP -код в HTML – все за пределами тегов PHP остается неизменным, тогда как внутри интерпретируется как код. Заметим также, что php-файл не похож на CGI-скрипт. Php файл не должен быть исполнимым или еще каким-либо образом помеченным.

Для того чтобы отправить php-файл на обработку серверу, нужно в строке браузера набрать путь к этому файлу на сервере. Скрипты php должны располагаться там, где разрешен доступ через www, например там же, где лежит домашняя страничка. Если php - файл лежит на локальной машине, то его можно обработать с помощью интерпретатора командной строки.

В качестве примера решим задачу создания заготовки электронного письма. Ее смысл заключается в следующем.

Допустим, у вас есть какое-то объявление и несколько разных людей, которым нужно это объявление отправить. Для этого вы делаете заготовку с содержанием объявления, внутри которого есть ряд изменяющихся (в зависимости от потенциального получателя) параметров.

Разделение инструкций

Программа на PHP (да и на любом другом языке программирования) – это набор команд (инструкций). Обработчику программы (парсеру) необходимо как-то отличать одну команду от другой. Для этого используются специальные символы – разделители. В PHP инструкции разделяются так же, как и в Си или Perl, – каждое выражение заканчивается точкой с запятой.

Закрывающий тег " ?> " также подразумевает конец инструкции, поэтому перед ним точку с запятой не ставят. Например, следующие фрагменты кода эквивалентны:

```
<?php
echo "Hello, world!"; // точка с запятой
                      // в конце команды
                      // обязательна
?>
<?php
echo "Hello, world!" ?>
<!-- точка с запятой
    опускается из-за "?>" -->
```

Комментарии

Часто при написании программ возникает необходимость делать какие-либо комментарии к коду, которые никак не влияют на сам код, а только поясняют его. Это важно при создании больших программ и в случае, если несколько человек работают над одной программой. При наличии комментариев в программе в ее коде разобраться гораздо проще. Кроме того, если решать задачу по частям, недоделанные части решения также удобно комментировать, чтобы не забыть о них в дальнейшем. Во всех языках программирования предусмотрена возможность включать комментарии в код программы. PHP поддерживает несколько видов комментариев: в стиле Си, C++ и оболочки Unix. Символы // и # обозначают начало однострочных комментариев, /* и */ – соответственно начало и конец многострочных комментариев.

```
<?php
echo "Меня зовут Вася";
  // Это однострочный комментарий
  // в стиле C++
echo "Фамилия моя Петров";
/* Это многострочный комментарий.
Здесь можно написать несколько строк.
При исполнении программы все, что
находится здесь (закомментировано),
будет игнорировано. */
echo "Я изучаю PHP в INTUIT.ru";
  # Это комментарий в стиле
  # оболочки Unix
?>
```

Пример 2. Использование комментариев в PHP

Переменные, константы и операторы

Важным элементом каждого языка являются переменные, константы и операторы, применяемые к этим переменным и константам. Рассмотрим, как выделяются и обрабатываются эти элементы в PHP.

Переменные

Переменная в PHP обозначается знаком доллара, за которым следует ее имя. Например:

```
$my_var
```

Имя переменной чувствительно к регистру, т.е. переменные $my_var и $My_var различны.

Имена переменных соответствуют тем же правилам, что и остальные наименования в PHP: правильное имя переменной должно начинаться с буквы или символа подчеркивания с последующими в любом количестве буквами, цифрами или символами подчеркивания.

В PHP 3 переменные всегда присваивались по значению. То есть когда вы присваиваете выражение переменной, все значения оригинального выражения копируются в эту переменную. Это означает, к примеру, что после присвоения одной переменной значения другой, изменение одной из них не влияет на значение другой.

```
<?php
$first = ' Text '; // Присваиваем $first
                   // значение
```

```
                    // ' Text '
$second = $first; // Присваиваем $second
                  // значение
                  // переменной $first
$first = ' New text '; // Изменяем
                       // значение
                       // $first
                       // на ' New text '
echo "Переменная с именем first равна " .$first <br>";
        // выводим значение $first
echo "Переменная с именем second равна " .$second";
        // выводим значение $second
?>
```

Пример 3. Присваивание по значению

Результат работы этого скрипта будет следующим:

```
Переменная с именем first равна New text
Переменная с именем second равна Text
```

PHP 4, кроме этого, предлагает еще один способ присвоения значений переменным: присвоение по ссылке. Для того, чтобы присвоить значение переменной по ссылке, это значение должно иметь имя, т.е. оно должно быть представлено какой-либо переменной. Чтобы указать, что значение одной переменной присваивается другой переменной по ссылке, нужно перед именем первой переменной поставить знак амперсанд &.

Рассмотрим тот же пример, что и выше, только будем присваивать значение переменной first переменной second по ссылке:

```
<?php
$first = ' Text '; // Присваиваем $first
                   // значение ' Text '
$second = &$first;
/* Делаем ссылку на $first через $second.
   Теперь значения этих переменных
   будут всегда совпадать */
// Изменим значение $first
// на ' New text '
$first = ' New text ';
echo "Переменная с именем first " .
     "равна .$first <br>";
// выведем значения обеих переменных
echo "Переменная с именем second " .
     "равна $second";
?>
```

Пример 4. Присваивание по ссылке

Этот скрипт выведет следующее:

```
Переменная с именем first равна New text.
Переменная с именем second равна New text.
```

То есть вместе с переменной $first изменилась и переменная $second.

Константы

Для хранения постоянных величин, т.е. таких величин, значение которых не меняется в ходе выполнения скрипта, используются константы. Такими величинами могут быть математические константы, пароли, пути к файлам и т.п. Основное отличие константы от переменной состоит в том, что ей нельзя присвоить значение больше одного раза и ее значение нельзя аннулировать после ее объявления. Кроме того, у константы нет приставки в виде знака доллара и ее нельзя определить простым присваиванием значения. Как же тогда можно определить константу? Для этого существует специальная функция define() . Ее синтаксис таков:

```
define("Имя_константы",
       "Значение_константы",
       [Нечувствительность_к_регистру])
```

По умолчанию имена констант чувствительны к регистру. Для каждой константы это можно изменить, указав в качестве значения аргумента Нечувствительность_к_регистру значение True. Существует соглашение, по которому имена констант всегда пишутся в верхнем регистре.

Получить значение константы можно, указав ее имя. В отличие от переменных, не нужно предварять имя константы символом $. Кроме того, для получения значения константы можно использовать функцию constant() с именем константы в качестве параметра.

```
<?php
// определяем константу
// PASSWORD
define("PASSWORD","qwerty");
// определяем регистронезависимую
// константу PI со значением 3.14
define("PI","3.14", True);
// выведем значение константы PASSWORD,
// т.е. qwerty
echo (PASSWORD);
// тоже выведет qwerty
echo constant("PASSWORD");
echo (password);
/* выведет password и предупреждение,
   поскольку мы ввели регистрозависимую
   константу PASSWORD */
echo pi;
// выведет 3.14, поскольку константа PI
// регистронезависима по определению
?>
```

Пример 5. Константы в PHP

Кроме констант, объявляемых пользователем, о которых мы только что рассказали, в PHP существует ряд констант, определяемых самим интерпретатором. Например, константа __FILE__ хранит имя файла программы (и путь к нему), которая выполняется в данный момент, __FUNCTION__ содержит имя функции, __CLASS__ – имя класса, PHP_VERSION – версия интерпретатора PHP. Полный список предопределенных констант можно получить, прочитав руководство по PHP.

Операторы

Операторы позволяют выполнять различные действия с переменными, константами и выражениями. Мы еще не упоминали о том, что такое выражение. Выражение можно определить как все, что угодно, что имеет значение. Переменные и константы – это основные и наиболее простые формы выражений. Существует множество операций (и соответствующих им операторов), которые можно производить с выражениями. Рассмотрим некоторые из них подробнее.

Таблица 1. Арифметические операторы

Обозначение	Название	Пример
+	Сложение	$a + $b
-	Вычитание	$a - $b
*	Умножение	$a * $b
/	Деление	$a / $b
%	Остаток от деления	$a % $b

Таблица 3. Строковые операторы

Обозначение	Название	Пример
.	Конкатенация (сложение строк)	`$c = $a . $b` (это строка, состоящая из $a и $b)

Таблица 4. Операторы присваивания

Обозначение	Название	Описание	Пример
=	Присваивание	Переменной слева от оператора будет присвоено значение, полученное в результате выполнения каких-либо операций или переменной / константы с правой стороны	`$a = ($b = 4) +5;` ($a будет равна 9, $b будет равна 4)
+=		Сокращение. Прибавляет к переменной число и затем присваивает ей полученное значение	`$a += 5;` (эквивалентно $a = $a + 5;)
.=		Сокращенно обозначает комбинацию операций конкатенации и присваивания	`$b = "Привет "; $b .= "всем";` (эквивалентно

		(сначала добавляется строка, потом полученная строка записывается в переменную)	$b = $b . "всем";) В результате: $b="Привет всем"

Таблица 5. Логические операторы

Обозначение	Название	Описание	Пример
and	И	$a и $b истинны (True)	$a and $b
&&	И		$a && $b
or	Или	Хотя бы одна из переменных $a или $b истинна (возможно, что и обе)	$a or $b
||	Или		$a || $b
xor	Исключающее или	Одна из переменных истинна. Случай, когда они обе истинны, исключается	$a xor $b
!	Инверсия (NOT)	Если $a=True, то ! $a=False и наоборот	! $a

Таблица 6. Операторы сравнения

Обозначение	Название	Описание	Пример
==	Равенство	Значения переменных равны	$a == $b
===	Эквивалентность	Равны значения и типы переменных	$a === $b
!=	Неравенство	Значения переменных не равны	$a != $b
<>	Неравенство		$a <> $b
!==	Неэквивалентно	Переменные не эквивалентны	$a !== $b

	сть	
<	Меньше	$a < $b
>	Больше	$a > $b
<=	Меньше или равно	$a <= $b
>=	Больше или равно	$a >= $b

Таблица 7. Операторы инкремента и декремента

Обозначение	Название	Описание	Пример

++$a	Пре- инкремент	Увеличивает $a на единицу и возвращает $a	`<?`
			`$a=4;`
			`echo "Должно быть 4:" .$a++;`
			`echo "Должно быть 5:" .++$a;`
			`?>`

$a++	Пост- инкремент	Возвращает $a, затем увеличивает $a на единицу	
--$a	Пре- декремент	Уменьшает $a на единицу и возвращает $a	
$a--	Пост- декремент	Возвращает $a, затем уменьшает $a на единицу	

Типы данных

PHP поддерживает восемь простых типов данных.

Четыре скалярных типа:

- boolean (логический) ;
- integer (целый) ;
- float (с плавающей точкой) ;
- string (строковый).

Два смешанных типа:

- array (массив) ;
- object (объект).

И два специальных типа:

- resource (ресурс) ;
- NULL.

В PHP не принято явное объявление типов переменных. Предпочтительнее, чтобы это делал сам интерпретатор во время выполнения программы в зависимости от контекста, в котором используется переменная. Рассмотрим по порядку все перечисленные типы данных.

Тип boolean (булев или логический тип)

Этот простейший тип выражает истинность значения, то есть переменная этого типа может иметь только два значения – истина TRUE или ложь FALSE .

Чтобы определить булев тип, используют ключевое слово TRUE или FALSE. Оба регистронезависимы.

```
<?php
$test = True;
?>
```

Пример 5. Логический тип

Логические переменные используются в различных управляющих конструкциях (циклах, условиях и т.п., более подробно речь о них пойдет в одной из следующих лекций). Иметь логический тип, т.е. принимать только два значения, истину или ложь, могут также и некоторые операторы (например, оператор равенства). Они также используются в управляющих конструкциях для проверки каких-либо условий. Например, в условной конструкции проверяется истинность значения оператора или переменной и в зависимости от результата проверки выполняются те или иные действия. Здесь условие может быть истинно или ложно, что как раз и отражает переменная и оператор логического типа.

```
<?php
// Оператор '==' проверяет равенство
// и возвращает
// булево значение
if ($know == False) { // если $know
                      // имеет значение
                      // false
echo "Изучай PHP!";
}
if (!$know) { // то же самое, что
              // и выше, т.е. проверка
              // имеет ли $know значение
              // false
echo "Изучай PHP!";
}
/* оператор == проверяет, совпадает ли
   значение переменной $know со строкой
   "Изучай PHP". Если совпадает, то
   возвращает true, иначе – false.
   Если возвращено true, то выполняется
   то, что внутри фигурных скобок */
if ($know == "Изучай PHP")
{ echo "Начал изучать"; }
```

```
?>
```

Пример 6. Использование логического типа

Тип integer (целые)

Этот тип задает число из множества целых чисел Z = {..., -2, -1, 0, 1, 2, ...}. Целые могут быть указаны в десятичной, шестнадцатеричной или восьмеричной системе счисления, по желанию с предшествующим знаком " - " или " + ".

Если вы используете восьмеричную систему счисления, вы должны предварить число 0 (нулем), для использования шестнадцатеричной системы нужно поставить перед числом 0x.

```
<?php
# десятичное число
$a = 1234;
# отрицательное число
$a = -123;
# восьмеричное число (эквивалентно
# 83 в десятичной системе)
$a = 0123;
# шестнадцатеричное число (эквивалентно
# 26 в десятичной системе)
$a = 0x1A;
?>
```

Размер целого зависит от платформы, хотя, как правило, максимальное значение около двух миллиардов (это 32-битное знаковое). Беззнаковые целые PHP не поддерживает.

Если вы определите число, превышающее пределы целого типа, оно будет интерпретировано как число с плавающей точкой. Также если вы используете оператор, результатом работы которого будет число, превышающее пределы целого, вместо него будет возвращено число с плавающей точкой.

В PHP не существует оператора деления целых. Результатом 1/2 будет число с плавающей точкой 0.5. Вы можете привести значение к целому, что всегда округляет его в меньшую сторону, либо использовать функцию round(), округляющую значение по стандартным правилам. Для преобразования переменной к конкретному типу нужно перед переменной указать в скобках нужный тип. Например, для преобразования переменной $a=0.5 к целому типу необходимо написать (integer)(0.5) или (integer) $a или использовать сокращенную запись (int)(0.5). Возможность явного приведения типов по такому принципу существует для всех типов данных (конечно, не всегда значение одного типа можно перевести в другой тип). Мы не будем углубляться во все тонкости приведения типов, поскольку PHP делает это автоматически в зависимости от контекста.

Тип float (числа с плавающей точкой)

Числа с плавающей точкой (они же числа двойной точности или действительные числа) могут быть определены при помощи любого из следующих синтаксисов:

```
<?php
$a = 1.234;
$b = 1.2e3;
$c = 7E-10;
?>
```

Размер числа с плавающей точкой зависит от платформы, хотя максимум, как правило, ~1.8e308 с точностью около 14 десятичных цифр.

Тип string (строки)

Строка – это набор символов. В PHP символ – это то же самое, что байт, это значит, что существует ровно 256 различных символов. Это также означает, что PHP не имеет встроенной поддержки Unicode. В PHP практически не существует ограничений на размер строк, поэтому нет абсолютно никаких причин беспокоиться об их длине.

Строка в PHP может быть определена тремя различными способами:

- с помощью одинарных кавычек ;
- с помощью двойных кавычек ;
- heredoc-синтаксисом.

Одинарные кавычки

Простейший способ определить строку – это заключить ее в одинарные кавычки " ' ". Чтобы использовать одинарную кавычку внутри строки, как и во многих других языках, перед ней необходимо поставить символ обратной косой черты " \ ", т. е. экранировать ее. Если обратная косая черта должна идти перед одинарной кавычкой либо быть в конце строки, необходимо продублировать ее " \\' ".

Если внутри строки, заключенной в одинарные кавычки, обратный слэш " \ " встречается перед любым другим символом (отличным от " \ " и " ' "), то он рассматривается как обычный символ и выводится, как и все остальные. Поэтому обратную косую черту необходимо экранировать, только если она находится в конце строки, перед закрывающей кавычкой.

В PHP существует ряд комбинаций символов, начинающихся с символа обратной косой черты. Их называют управляющими последовательностями, и они имеют специальные значения, о которых мы расскажем немного позднее. Так вот, в отличие от двух других синтаксисов, переменные и управляющие последовательности для специальных символов, встречающиеся в строках, заключенных в одинарные кавычки, не обрабатываются .

```
<?php
echo 'Также вы можете вставлять в строки
    символ новой строки таким образом,
    поскольку это нормально';

// Выведет: Чтобы вывести ' надо
// перед ней поставить \
echo 'Чтобы вывести \' надо перед ' .
     'ней поставить \\';
// Выведет: Вы хотите удалить C:\*.*?
echo 'Вы хотите удалить C:\\*.*?';
// Выведет: Это не вставит: \n новую строку
echo 'Это не вставит: \n новую строку';
// Выведет: Переменные $expand также
// $either не подставляются
echo 'Переменные $expand также $either' .
     'не подставляются';
?>
```

Пример 7. Использование управляющих последовательностей

Двойные кавычки

Если строка заключена в двойные кавычки " " ", PHP распознает большее количество управляющих последовательностей для специальных символов.

Таблица 8. Управляющие последовательности

Последовательность	Значение
\n	Новая строка (LF или 0x0A (10) в ASCII)
\r	Возврат каретки (CR или 0x0D (13) в ASCII)
\t	Горизонтальная табуляция (HT или 0x09 (9) в ASCII)
\\	Обратная косая черта
\$	Знак доллара
\"	Двойная кавычка

Повторяем, если вы захотите экранировать любой другой символ, обратная косая черта также будет напечатана!

Самым важным свойством строк в двойных кавычках является обработка переменных.

Heredoc

Другой способ определения строк – это использование heredoc-синтаксиса. В этом случае строка должна начинаться с символа <<<, после которого идет идентификатор. Заканчивается строка этим же идентификатором. Закрывающий идентификатор должен начинаться в первом столбце строки. Кроме того, идентификатор должен соответствовать тем же правилам именования, что и все остальные метки в PHP: содержать только буквенно-цифровые символы и знак подчеркивания и начинаться не с цифры или знака подчеркивания.

Heredoc -текст ведет себя так же, как и строка в двойных кавычках, при этом их не имея. Это означает, что вам нет необходимости экранировать кавычки в heredoc, но вы по-прежнему можете использовать перечисленные выше управляющие последовательности. Переменные внутри heredoc тоже обрабатываются.

```
<?php
$str = <<<EOD
Пример строки, охватывающей несколько
строчек, с использованием
heredoc-синтаксиса
EOD;
// Здесь идентификатор – EOD. Ниже
```

```
// идентификатор EOD
$name = 'Вася';
echo <<<EOD
Меня зовут "$name".
EOD;
// это выведет: Меня зовут "Вася".
?>
```

Пример 8. Использование heredoc-синтаксиса

Замечание: Поддержка heredoc была добавлена в PHP 4.

Тип array (массив)

Массив в PHP представляет собой упорядоченную карту – тип, который преобразует значения в ключи. Этот тип оптимизирован в нескольких направлениях, поэтому вы можете использовать его как собственно массив, список (вектор), хеш-таблицу (являющуюся реализацией карты), стек, очередь и т.д. Поскольку вы можете иметь в качестве значения другой массив PHP, можно также легко эмулировать деревья.

Определить массив можно с помощью конструкции array () или непосредственно задавая значения его элементам.

Определение при помощи array()

```
array (key => value,
       key1 => value1, ... )
```

Языковая конструкция array () принимает в качестве параметров пары ключ => значение, разделенные запятыми. Символ => устанавливает соответствие между значением и его ключом. Ключ может быть как целым числом, так и строкой, а значение может быть любого имеющегося в PHP типа. Числовой ключ массива часто называют индексом. Индексирование массива в PHP начинается с нуля. Значение элемента массива можно получить, указав после имени массива в квадратных скобках ключ искомого элемента. Если ключ массива представляет собой стандартную запись целого числа, то он рассматривается как число, в противном случае – как строка. Поэтому запись $a["1"] равносильна записи $a[1], так же как и $a["-1"] равносильно $a[-1].

```
<?php
$books = array ("php" =>
                "PHP users guide",
                12 => true);
echo $books["php"];
//выведет "PHP users guide"
echo $books[12];
//выведет 1
?>
```

Пример 9. Массивы в PHP

Если для элемента ключ не задан, то в качестве ключа берется максимальный числовой ключ, увеличенный на единицу. Если указать ключ, которому уже было присвоено какое-то значение, то оно будет перезаписано. Начиная с PHP 4.3.0, если максимальный ключ – отрицательное число, то следующим ключом массива будет ноль (0).

```
<?php
// массивы $arr и $arr1 эквивалентны
$arr = array(5 => 43, 32, 56, "b" => 12);
```

```
$arr1 = array(5 => 43, 6 => 32,
              7 => 56, "b" => 12);
?>
```
Пример 10. Массивы в PHP

Если использовать в качестве ключа TRUE или FALSE, то его значение переводится соответственно в единицу и ноль типа integer. Если использовать NULL, то вместо ключа получим пустую строку. Можно использовать и саму пустую строку в качестве ключа, при этом ее надо брать в кавычки. Так что это не то же самое, что использование пустых квадратных скобок. Нельзя использовать в качестве ключа массивы и объекты.

Определение с помощью синтаксиса квадратных скобок

Создать массив можно, просто записывая в него значения. Как мы уже говорили, значение элемента массива можно получить с помощью квадратных скобок, внутри которых нужно указать его ключ, например, $book["php"]. Если указать новый ключ и новое значение, например, $book["new_key"]="new_value", то в массив добавится новый элемент. Если мы не укажем ключ, а только присвоим значение $book[]="new_value", то новый элемент массива будет иметь числовой ключ, на единицу больший максимального существующего. Если массив, в который мы добавляем значения, еще не существует, то он будет создан.

```
<?
$books["key"]= value; // добавили в массив
                      // $books значение
                      // value с ключом key
$books[] = value1; /* добавили в массив
                      значение value1 с
                      ключом 13, поскольку
                      максимальный ключ у
                      нас был 12 */
?>
```

Для того чтобы изменить конкретный элемент массива, нужно просто присвоить ему с его ключом новое значение. Изменить ключ элемента нельзя, можно только удалить элемент (пару ключ / значение) и добавить новую. Чтобы удалить элемент массива, нужно использовать функцию unset () .

```
<?php
$books = array ("php" =>
                "PHP users guide",
                12 => true);

$books[] =
 "Book about Perl"; // добавили элемент
                    // с ключом (индексом)
                    // 13 это эквивалентно
                    // $books[13] =
                    // "Book about Perl";
$books["lisp"] =
 123456; /* Это добавляет к массиву новый
            элемент с ключом "lisp" и
            значением 123456 */
unset($books[12]); // Это удаляет элемент
                   // с ключом 12 из массива
unset ($books); // удаляет массив полностью
?>
```

Заметим, что, когда используются пустые квадратные скобки, максимальный числовой ключ ищется среди ключей, существующих в массиве с момента последнего переиндексирования. Переиндексировать массив можно с помощью функции array_values ().

```
<?php
$arr =
 array ("a","b","c"); /* Создаем массив
                         со значениями
                         "a", "b" и "c".
                         Поскольку ключи
                         не указаны, они
                         будут 0,1,2
                         соответственно */
print_r($arr); // выводим массив (и ключи,
               // и значения)
unset($arr[0]);
unset($arr[1]);
unset($arr[2]);
         // удаляем из него все значения
print_r($arr);  // выводим массив (и ключи,
                // и значения)
$arr[] = "aa";  // добавляем новый элемент
                // в массив.
                // Его индексом (ключом)
                // будет 3, а не 0
print_r($arr);

$arr =
 array_values($arr); // переиндексируем
                     // массив
$arr[] = "bb"; // ключом этого элемента
               // будет 1
print_r($arr);
?>
```

Пример 11. Переиндексация массива

Результатом работы этого скрипта будет:

```
Array ( [0] => a [1] => b [2] => c )
Array ( )
Array ( [3] => aa )
Array ( [0] => aa [1] => bb )
```

Тип object (объекты)

Объекты – тип данных, пришедший из объектно-ориентированного программирования (ООП). Согласно принципам ООП, класс – это набор объектов, обладающих определенными свойствами и методами работы с ним, а объект соответственно – экземпляр класса. Например, программисты – это класс людей, которые пишут программы, изучают компьютерную литературу и, кроме того, как все люди, имеют имя и фамилию. Теперь, если взять одного конкретного программиста, Васю Иванова, то можно сказать, что он является объектом класса программистов, обладает теми же свойствами, что и другие программисты, тоже имеет имя, пишет программы и т.п.

В PHP для доступа к методам объекта используется оператор ->. Для инициализации объекта используется выражение new, создающее в переменной экземпляр объекта.

```
<?php
//создаем класс людей
class Person
{
// метод, который обучает человека PHP
    function know_php()
        {
            echo "Теперь я знаю PHP";
        }
}
$bob = new Person; // создаем объект
                   // класса человек
$bob -> know_php(); // обучаем его PHP
?>
```

Пример 12. Объекты в PHP

Более подробно реализацию принципов ООП в языке PHP мы рассмотрим в следующих лекциях.

Тип resource (ресурсы)

Ресурс – это специальная переменная, содержащая ссылку на внешний ресурс (например, соединение с базой данных). Ресурсы создаются и используются специальными функциями (например, mysql_connect(), pdf_new() и т.п.).

Тип Null

Специальное значение NULL говорит о том, что переменная не имеет значения.

Переменная считается NULL, если:

- ей была присвоена константа NULL ($var = NULL);
- ей еще не было присвоено какое-либо значение;
- она была удалена с помощью unset ().

Существует только одно значение типа NULL – регистронезависимое ключевое слово NULL.

Решение задачи

Теперь вернемся к задаче, которую мы поставили в самом начале лекции. Напомним, что она состояла в составлении письма разным людям по поводу разных событий. Попытаемся использовать для решения этой задачи изученные средства – переменные, операторы, константы, строки и массивы. В зависимости от получателя изменяется событие и обращение, указанные в письме, поэтому естественно вынести эти величины в переменные. Более того, поскольку событий и людей много, удобно использовать переменные типа массив. Подпись в письме остается постоянной всегда, поэтому логично задать ее как константу. Чтобы не писать слишком длинные и громоздкие строки, используем оператор конкатенации. Итак, вот что получилось:

```
<?
// пусть наша подпись
// будет константой
define("SIGN","С уважением, Вася");
```

```
// зададим массивы людей и событий
$names = array("Иван Иванович",
               "Петр Петрович",
               "Семен Семенович");
$events = array(
    "f" => "день открытых дверей",
    "o" => "открытие выставки",
    "p" => "бал выпускников");

// составим текст приглашения
$str = "Уважаемый (ая), $names[0].";
$str .= "<br>Приглашаем Вас на ".
        $events["f"];
$str .= "<br>" . SIGN;
echo $str; // выведем текст на экран
?>
```

Дополнительное задание:

Составьте код своего письма, используя числовые, строчные типы данных, а также ассоциативные массивы и константы. Загрузите скрипт в корневую директорию веб-сервера и проверьте его работоспособность.

Критерии оценивания:

3: предоставлен отчет по всем примерам лабораторной работы с кодом и индивидуальными скриншотами выполнения скриптов.

4-5: выполнено дополнительное задание, в отчете представлена блок-схема алгоритма и скриншоты этого задания