# **Лабораторная работа №2. Переменные в PHP**

## **2.1. Общие понятия о переменных в PHP**

Как и в любом другом языке программирования, в PHP существует такое понятие, как переменная.

При программировании на PHP можно не скупиться на объявление новых переменных. Принципы экономии памяти, которые были актуальны несколько лет назад, сегодня в расчет не принимаются. Однако, при хранении в переменных больших объемов памяти, лучше удалять неиспользуемые переменные, используя оператор [**Unset**](http://www.php.su/functions/?unset).

Вообще, **переменная** – это область оперативной памяти, которая хранит в себе данные: числа, строки (текст) и т.д. и доступ к которой осуществляется по имени. У переменной имеется имя (которое вы придумываете сами), обращаясь к переменной по имени, можно получить данные, которые она в себе хранит.

Все данные, с которыми работает программа, хранятся в виде переменных (исключение — [**константа**](http://www.php.su/learnphp/?const), которая, впрочем, может содержать только число или строку). Такого понятия, как указатель (как в Си), в PHP не существует – при присвоении переменная копируется один-в-один, какую бы сложную структуру она ни имела. Тем не менее, в PHP, начиная с версии 4, существует понятие [**ссылок**](http://www.php.su/learnphp/?re) — жестких и символических.

Схема **создания переменной в PHP** (объявления переменной), выглядит следующим образом:

**$имяПеременной;**

Имена переменных в PHP могут состоять из **цифр**, **букв английского алфавита**, знака подчёркивания **\_** и знака доллара **$**.

Имя переменной должно начинаться со знака доллара, после знака доллара должна идти буква или знак подчёркивания.

Имена переменных в PHP **чувствительны к регистру** букв, например, $**имяПеременной** и $**ИМЯПЕРЕМЕННОЙ** – это разные переменные.

В официальной документации PHP указано, что имя переменной может состоять не только из букв «латиницы» и цифр, но также и из любых символов, код ASCII которых старше 127 – в частности, и из символов кириллицы, то есть «русских» букв! Однако не рекомендуется применять кириллицу в именах переменных – хотя бы из-за того, что в различных кодировках ее буквы имеют различные коды. Впрочем, можно поэкспериментировать и делать так, как вам будет удобно.

Можно сказать, что переменные в PHP – это особые объекты, которые могут содержать в буквальном смысле все, что угодно.

Некоторые примеры переменных в PHP:

*<?php  
$var = "Bob";  
$Var = "Joe";  
echo "$var, $Var"; // выведет "Bob, Joe"  
  
$4site = 'not yet'; // неверно; начинается с цифры  
$\_4site = 'not yet'; // верно; начинается с символа подчеркивания  
$tдyte = 'mansikka'; // верно; 'д' это (Дополнительный) ASCII 228.  
?>*

Отличительным преимуществом PHP является то, что в PHP не нужно ни описывать переменные явно, ни указывать их [тип](http://www.php.su/learnphp/vars/?types). Интерпретатор все это делает сам. Однако иногда он может ошибаться (например, если в текстовой строке на самом деле задано десятичное число), поэтому изредка возникает необходимость явно указывать, какой же тип имеет то или иное выражение.  
Чуть чаще возникает потребность узнать тип переменной (например, переданной в параметрах функции) прямо во время выполнения программы.

**Инициализация** – это присвоение начального значения переменной. Присвоить какое-либо значение переменной можно с помощью оператора присваивания, который обозначается символом равно (=):

$str = 'строка1, ';

$Str = 'строка2';

echo $str, $Str;

Переменные в PHP не имеют типа, поэтому переменной может быть присвоено сначала значение одного типа, а затем этой же переменной может быть присвоено значение другого типа:

$x = 10;        *// Сначала присваивается число*

$x = "текст";   *// Затем присваивается строка*

## **2.2 Типы данных (переменных) в PHP**

PHP поддерживает восемь простых типов данных (переменных):

*Четыре скалярных типа:*

[boolean (двоичные данные)](http://www.php.su/learnphp/vars/?types#boolean);

[integer (целые числа)](http://www.php.su/learnphp/vars/?types#integer);

[float](http://www.php.su/learnphp/vars/?types#float) (числа с плавающей точкой или '[double](http://www.php.su/learnphp/vars/?types#float)');

[string (строки)](http://www.php.su/learnphp/vars/?types#string).

*Два смешанных типа:*

[array (массивы)](http://www.php.su/learnphp/vars/?types#array);

[object (объекты)](http://www.php.su/learnphp/vars/?types#object).

*И два специальных типа:*

[resource (ресурсы)](http://www.php.su/learnphp/vars/?types#resource);

[NULL ("пустой" тип)](http://www.php.su/learnphp/vars/?types#null).

*Существуют также несколько псевдотипов:*

[mixed (смешанный)](http://www.php.su/learnphp/vars/?types#mixed);

[number (числовой)](http://www.php.su/learnphp/vars/?types#number);

[callback (обратного вызова)](http://www.php.su/learnphp/vars/?types#callback).

Рассмотрим кратко перечисленные типы данных PHP.

### **2.2.1 Тип boolean (двоичные данные)**

Это простейший тип. Он выражает истинность значения – это может быть либо **TRUE**, либо **FALSE**. Булев тип был введен в PHP 4. Чтобы определить булев тип, используйте ключевое слово **TRUE** или **FALSE**. Оба регистро-независимы.

**<?php**$x = **True**; *// присвоить $x значение TRUE***?>**

**Преобразование в тип boolean**

Для преобразования значения в булев тип используйте приведение типа **(bool)** или **(boolean)**. Однако в большинстве случаев вам нет необходимости использовать приведение типа, поскольку значение будет автоматически преобразовано, если оператор, функция или управляющая конструкция требует булев аргумент.

При преобразовании в логический тип, следующие значения рассматриваются как **FALSE**:

* сам булев **FALSE**
* [целое](http://www.php.su/learnphp/datatypes/?integer) 0 (ноль)
* [число с плавающей точкой](http://www.php.su/learnphp/datatypes/?float) 0.0 (ноль)
* пустая [строка](http://www.php.su/learnphp/datatypes/?string) и строка «0»
* [массив](http://www.php.su/learnphp/datatypes/?array) с нулевыми элементами
* [объект](http://www.php.su/learnphp/datatypes/?object) с нулевыми переменными-членами
* специальный тип [NULL](http://www.php.su/learnphp/datatypes/?null) (включая неустановленные переменные)

Все остальные значения рассматриваются как **TRUE** (включая любой [ресурс](http://www.php.su/learnphp/datatypes/?resource)).

Внимание! **-1** считается **TRUE**, как и любое ненулевое (отрицательное или положительное) число!

**<?php***var\_dump*((bool) **""**); *// bool(false)  
var\_dump*((bool) 1); *// bool(true)  
var\_dump*((bool) -2); *// bool(true)  
var\_dump*((bool) **"foo"**); *// bool(true)  
var\_dump*((bool) 2.3e5); *// bool(true)  
var\_dump*((bool) **array**(12)); *// bool(true)  
var\_dump*((bool) **array**()); *// bool(false)  
var\_dump*((bool) **"false"**); *// bool(true)***?>**

### **2.2.2 Тип integer (целые числа)**

Целое – это число из множества Z = {..., -2, -1, 0, 1, 2, ...}, обычно длиной 32 бита (от –2 147 483 648 до 2 147 483 647).

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

Формально возможная структура целых такова:

**десятичные** : [1-9][0-9]\*  
             | 0  
  
**шестнадцатеричные**: 0[xX][0-9a-fA-F]+  
  
**восьмеричные**: 0[0-7]+  
  
**целые:**  
  
[+-]?десятичные  
  
| [+-]?шестнадцатеричные  
  
| [+-]?восьмеричные

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

**Превышение размера целого**

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.

**<?php**$large\_number = 2147483647;  
*var\_dump*($large\_number);  
*// вывод: int(2147483647)*$large\_number = 2147483648;  
*var\_dump*($large\_number);  
*// вывод: float(2147483648)  
  
// это справедливо и для шестнадцатеричных целых:  
var\_dump*( 0x80000000 );  
*// вывод: float(2147483648)*$million = 1000000;  
$large\_number = 50000 \* $million;  
*var\_dump*($large\_number);  
*// вывод: float(50000000000)***?>**

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()**.

**<?php***var\_dump=*(25/7); *// float(3.5714285714286)   
var\_dump=*((int) (25/7)); *// int(3)  
var\_dump=*(*round*(25/7)); *// float(4)***?>**

**Преобразование других типов данных в целое (integer)**

Для несомненного преобразования значения в целое используйте приведение типа **(int)** или **(integer)**. Однако в большинстве случаев вам нет необходимости использовать приведение типа, поскольку значение будет автоматически преобразовано, если оператор, функция или управляющая конструкция требует целый аргумент. Вы также можете преобразовать значение в целое при помощи функции **intval()**.

При преобразовании из числа с плавающей точкой (**float**) в целое, число будет округлено в меньшую сторону.

**<?php**$a=1.55;  
$b=*intval*($a);*//преобразование типа float в integer***echo** $b;*// b будет равно 1*  
**?>**

Если число с плавающей точкой превышает пределы целого (как правило, это *+/- 2.15e+9 = 2^31*), результат будет неопределенным, так как целое не имеет достаточной точности, чтобы вернуть верный результат. В этом случае не будет выведено ни предупреждения, ни даже замечания!

Внимание! Никогда не приводите неизвестную дробь к целому, так как это может иногда дать неожиданные результаты. Пример ошибочного преобразования:

<?php  
echo (int) ( (0.1+0.7) \* 10 ); // выводит 7! (ошибка!)  
?>

При преобразовании переменных типа Booleanв integer **FALSE** преобразуется в **0** (ноль), а **TRUE** – в **1** (единицу).

**<?php**$a=**false**;  
$b=*intval*($a);*//преобразов. типа boolean в integer. Результат будет равен 0***echo** $b;  
**?>**

### **2.2.3 Тип float (числа с плавающей точкой)**

**Float (или double)** – вещественное число довольно большой точности (ее должно хватить для подавляющего большинства математических вычислений).

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;  
**?>**

Формальная структура скалярного типа **float** такая:

LNUM          [0-9]+  
DNUM          ([0-9]\*[.]{LNUM}) | ({LNUM}[.][0-9]\*)  
EXPONENT\_DNUM ( ({LNUM} | {DNUM}) [eE][+-]? {LNUM})

Размер целого зависит от платформы, хотя максимум, как правило, ~1.8e308 с точностью около 14 десятичных цифр (это 64-битный IEEE-формат).

**Точность числа с плавающей точкой**

Довольно часто простые десятичные дроби вроде *0.1* или *0.7* не могут быть преобразованы в свои внутренние двоичные аналоги без небольшой потери точности. Это может привести к неожиданным результатам: например, *floor((0.1+0.7)\*10)* скорее всего возвратит *7* вместо ожидаемой *8* как результат внутреннего представления числа, являющегося в действительности чем-то вроде *7.9999999999...*. (floor – округляет дробь в меньшую сторону).

**<?php**$a=*floor*((0.1+0.7)\*10);  
**echo** $a;*// b будет равно 1***?>**

Это связано с невозможностью точно выразить некоторые дроби в десятичной системе счисления конечным числом цифр. Например, *1/3* в десятичной форме принимает вид *0.3333333. . .*.

Так что никогда не доверяйте точности последних цифр в результатах с числами с плавающей точкой и никогда не проверяйте их на равенство. Если вам действительно необходима высокая точность, вам следует использовать [*математические функции произвольной точности*](http://www.php.su/functions/?cat=bc)или [*gmp*](http://www.php.su/functions/?cat=gmp)-функции.

### **2.2.4 Тип string (строки)**

**Строка** в PHP – это набор символов любой длины. В отличие от Си, строки могут содержать в себе также и нулевые символы, что никак не повлияет на программу. Иными словами, строки можно использовать для хранения бинарных данных. Длина строки ограничена только размером свободой оперативной памяти.

Строка легко может быть обработана при помощи стандартных функций, можно также непосредственно обратиться к любому ее символу.

Пример строковой переменной:

**<?php**$a = **"Это просто текст, записанный в строковую переменную"**;  
**echo** $a; *//Выводит 'Это просто текст, записанный в строковую переменную'***?>**

А теперь подробно разберем синтаксис типа данных **string**.

**Синтаксис типа string (строк)**

Строка может быть определена тремя различными способами.

* [одинарными кавычками](http://www.php.su/learnphp/datatypes/?string#single);
* [двойными кавычками](http://www.php.su/learnphp/datatypes/?string#double);
* [heredoc-синтаксисом](http://www.php.su/learnphp/datatypes/?string#heredoc).

**Определение строк одинарными кавычками**

Простейший способ определить строку – это заключить ее в одинарные кавычки (символ ').

Чтобы использовать одинарную кавычку внутри строки, как и во многих других языках, ее необходимо предварить символом обратной косой черты (\), т. е. экранировать ее. Если обратная косая черта должна идти перед одинарной кавычкой либо быть в конце строки, вам необходимо продублировать ее. Обратите внимание, что если вы попытаетесь экранировать любой другой символ, обратная косая черта также будет напечатана! Так что, как правило, нет необходимости экранировать саму обратную косую черту.

В отличие от двух других синтаксисов, переменные и экранирующие последовательности для специальных символов, встречающиеся в строках, заключенных в одинарные кавычки, **не** обрабатываются.

Приведем пример использования одинарных кавычек:

**<?php  
echo 'это простая строка'**;  
  
**echo 'Также вы можете вставлять в строки  
символ новой строки таким образом,  
поскольку это нормально'**;  
  
*// Выведет: Однажды Арнольд сказал: "I'll be back"***echo 'Однажды Арнольд сказал: "I\'ll be back"'**;  
  
*// Выведет: Вы удалили C:\\*.\*?***echo 'Вы удалили C:\\\*.\*?'**;  
  
*// Выведет: Вы удалили C:\\*.\*?***echo 'Вы удалили 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'**;  
**?>**

**Определение строк двойными кавычками**

Если строка заключена в двойные кавычки (**"**), PHP распознает большее количество управляющих последовательностей для специальных символов:

Таблица 2.1 Таблица управляющих последовательностей

| **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 |
| *\[0-7]{1,3}* | последовательность символов, соответствующая регулярному выражению, символ в восьмеричной системе счисления |
| *\x[0-9A-Fa-f]{1,2}* | последовательность символов, соответствующая регулярному выражению, символ в шестнадцатеричной системе счисления |

Еще раз напомним, что если вы захотите мнемнонизировать любой другой символ, обратная косая черта также будет напечатана!

Самым важным свойством строк в двойных кавычках является обработка переменных. Например, данную особенность можно использовать при выводе значений переменных и т.д.

**<?php**$a = 2;  
$b=10;  
**echo "Переменная a=** $a **, a переменная b=** $b **."**; *//Выводит значения a и b, встроенные в текст***?>**

**Определение строк heredoc-синтаксисом**

Другой способ определения строк – это использование **heredoc-синтаксиса** ("**<<<**"). После *<<<* необходимо указать идентификатор, затем идет строка, а потом этот же идентификатор, закрывающий вставку.

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 должен начинаться с не цифры или знака подчеркивания.

Внимание! Очень важно отметить, что строка с закрывающим идентификатором не содержит других символов, за исключением, возможно, точки с запятой (;). Это означает, что идентификатор не должен вводиться с отступом и что не может быть никаких пробелов или знаков табуляции до или после точки с запятой. Важно также понимать, что первым символом перед закрывающим идентификатором должен быть символ новой строки, определенный в вашей операционной системе. Например, на Windows® это \r.

Если это правило нарушено и закрывающий идентификатор не является «чистым», считается, что закрывающий идентификатор отсутствует и PHP продолжит его поиск дальше. Если в этом случае верный закрывающий идентификатор так и не будет найден, то это вызовет ошибку в обработке с номером строки в конце скрипта.

$str = <<<EOD

Пример строки,

охватывающей несколько строчек,

с использованием heredoc-синтаксиса.

EOD;

/\* Более сложный пример с переменными. \*/

class foo

{

var $foo;

var $bar;

function foo()

{

$this->foo = 'Foo';

$this->bar = array('Bar1', 'Bar2', 'Bar3');

}

}

$foo = new foo();

$name = 'МоеИмя';

echo <<<EOT

Меня зовут "$name". Я печатаю $foo->foo.

Теперь я вывожу {$foo->bar[1]}.

Это должно вывести заглавную букву 'A': \x41

EOT;

**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 вышеперечисленные управляющие последовательности. Переменные обрабатываются, но с применением сложных переменных внутри heredoc нужно быть также внимательным, как и при работе со строками.

### **2.2.5 Тип array (массивы)**

Массив в PHP – это упорядоченный набор данных, в котором установлено соответствие между значением и ключом. Индекс (ключ) служит для однозначной идентификации элемента внутри массива. В одном массиве не может быть двух элементов с одинаковыми индексами.

PHP позволяет создавать массивы любой сложности. Рассмотрим некоторые примеры:

**Простой массив (список)**

Массивы, индексами которых являются числа, начинающиеся с нуля - это списки:

**<?php***// Простой способ инициализации массива*$names[0]=**"Апельсин"**;  
$names[1]=**"Банан"**;  
$names[2]=**"Груша"**;  
$names[3]=**"Помидор"**;  
*// Здесь: names - имя массива, а 0, 1, 2, 3 - индексы массива***echo** $names[1]; *// Выведет на экран Банан***?>**

Отметим, что нумерация в простом массиве может начинаться и не с нуля, например:

**<?php***// Простой способ инициализации массива*$names[1]=**"Апельсин"**;  
$names[2]=**"Банан"**;  
$names[3]=**"Груша"**;  
$names[4]=**"Помидор"**;  
*// Здесь: names - имя массива, а 0, 1, 2, 3 - индексы массива***echo** $names[1]; *// Выведет на экран Апельсин***?>**

Доступ к элементам простых массивов (списков) осуществляется следующим образом (по номеру элемента в простом массиве (списке)):

**<?php***// Простой способ инициализации массива*$names[0]=**"Апельсин"**;  
$names[1]=**"Банан"**;  
$names[2]=**"Груша"**;  
$names[3]=**"Помидор"**;  
*// Здесь: names - имя массива, а 0, 1, 2, 3 - индексы массива  
  
// Выводим элементы массивов в браузер:***echo** $names[0]; *// Вывод элемента массива names с индексом 0***echo "<br>"**;  
**echo** $names[3]; *// Вывод элемента массива names с индексом 3  
// Выводит:  
// Апельсин  
// Помидор***?>**

Отметим, что в данном примере перед выводом элемента массива names с индексом 3 (помидор) для перехода на новую строку используется echo "<br>".

Простые массивы можно создавать, не указывая индекс нового элемента массива, это за вас сделает PHP. Например:

**<?php***// Простой способ инициализации массива, без указания индексов*$names[]=**"Апельсин"**;  
$names[]=**"Банан"**;  
$names[]=**"Груша"**;  
$names[]=**"Помидор"**;  
*// PHP автоматически присвоит индексы элементам массива, начиная с 0  
  
// Выводим элементы массивов в браузер:***echo** $names[0]; *// Вывод элемента массива names с индексом 0***echo "<br>"**;  
**echo** $names[3]; *// Вывод элемента массива names с индексом 3  
// Выводит:  
// Апельсин  
// Помидор***?>**

В рассмотренном примере вы можете добавлять элементы массива **names** простым способом, то есть не указывая индекс элемента массива:

$names[]=**"Яблоко"**;

Новый элемент простого массива (списка) будет добавлен в конец массива. В дальнейшем, с каждым новым элементом массива, индекс будет увеличиваться на единицу.

**Простые многомерные массивы**

Обобщенный синтаксис элементов многомерного простого массива:

$имя[индекс1][индекс2]..[индексN];

Пример простого многомерного массива:

**<?php***// Многомерный простой массив:*$arr[0][0]=**"Овощи"**;  
$arr[0][1]=**"Фрукты"**;  
$arr[1][0]=**"Абрикос"**;  
$arr[1][1]=**"Апельсин"**;  
$arr[1][2]=**"Банан"**;  
$arr[2][0]=**"Огурец"**;  
$arr[2][1]=**"Помидор"**;  
$arr[2][2]=**"Тыква"**;  
  
*// Выводим некоторые элементы массива:***echo "<h3>"**.$arr[1][0].**":</h3>"**;  
  
**echo "<h3>"**.$arr[2][1].**":</h3>"**;  
  
**?>**

**Ассоциативные массивы**

В PHP индексом массива может быть не только число, но и строка. Причем на такую строку не накладываются никакие ограничения: она может содержать пробелы, длина такой строки может быть любой.

Ассоциативные массивы особенно удобны в ситуациях, когда элементы массива удобнее связывать со словами, а не с числами.

Итак, массивы, индексами которых являются строки, называются ассоциативными массивами.

**Одномерные ассоциативные массивы**

Одномерные ассоциативные массивы содержат только один ключ (элемент), соответствующий конкретному индексу ассоциативного массива.

Приведем пример:

**<?php***// Ассоциативный массив*$names[**"Иванов"**]=**"Иван"**;  
$names[**"Сидоров"**]=**"Николай"**;  
$names[**"Петров"**]=**"Петр"**;  
*// В данном примере: фамилии - ключи ассоциативного массива  
// , а имена - элементы массива***echo** $names[**"Сидоров"**]; *//выводит имя Николай***?>**

Доступ к элементам одномерных ассоциативных массивов осуществляется так же, как и к элементам обыкновенных массивов, и называется доступом по ключу. Например:

**echo** $names[**"Сидоров"**]; *//выводит имя Николай*

**Многомерные ассоциативные массивы**

Многомерные ассоциативные массивы могут содержать несколько ключей, соответствующих конкретному индексу ассоциативного массива. Рассмотрим пример многомерного ассоциативного массива:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **name** | **age** | **email** |
| **Ivanov** | **Иванов И.И.** | **25** | **ivanov@mail.ru** |
| **Petrov** | **Петров П.П.** | **34** | **petrov@mail.ru** |
| **Sidorov** | **Сидоров С.С.** | **47** | **sidorov@mail.ru** |

На PHP работа с таким многомерным ассоциативным массивом может выглядеть следующим образом.

**<?php***// Многомерный массив*$A[**"Ivanov"**] = **array**(**"name"**=>**"Иванов И.И."**, **"age"**=>**"25"**, **"email"**=>**"ivanov@mail.ru"**);  
$A[**"Petrov"**] = **array**(**"name"**=>**"Петров П.П."**, **"age"**=>**"34"**, **"email"**=>**"petrov@mail.ru"**);  
$A[**"Sidorov"**] = **array**(**"name"**=>**"Сидоров С.С."**, **"age"**=>**"47"**, **"email"**=>**"sidorov@mail.ru"**);  
  
**echo** $A[**"Petrov"**][**"name"**];*// Выводит Иванов И.И***echo "<br>"**;  
**echo** $A[**"Petrov"**][**"age"**];*// Выводит возраст***echo "<br>"**;  
**echo** $A[**"Petrov"**][**"email"**];*// Выводит e-mail***?>**

Доступ к элементам многомерного ассоциативного массива осуществляется следующим образом:

**echo** $A[**"Petrov"**][**"name"**];*// Выводит Иванов И.И***echo** $A[**"Petrov"**][**"age"**];*// Выводит возраст***echo** $A[**"Petrov"**][**"email"**];*// Выводит e-mail*

Отметим, что для создания многомерного ассоциативного массива была использована специальная функция **array**. Ассоциативные многомерные массивы можно создавать и классическим способом, хотя это не так удобно:

**<?php***// Многомерный ассоциативный массив*$A[**"Ivanov"**][**"name"**]=**"Иванов И.И."**;  
$A[**"Ivanov"**][**"age"**]=**"25"**;  
$A[**"Ivanov"**][**"email"**]=**"ivanov@mail.ru"**;  
  
$A[**"Petrov"**][**"name"**]=**"Петров П.П."**;  
$A[**"Petrov"**][**"age"**]=**"34"**;  
$A[**"Petrov"**][**"email"**]=**"petrov@mail.ru"**;  
  
$A[**"Sidorov"**][**"name"**]=**"Сидоров С.С."**;  
$A[**"Sidorov"**][**"age"**]=**"47"**;  
$A[**"Sidorov"**][**"email"**]=**"sidorov@mail.ru"**;  
  
*//Получаем доступ к ключам многомерного ассоциативного массива***echo** $A[**"Petrov"**][**"name"**];*// Выводит Иванов И.И***echo "<br>"**;  
**echo** $A[**"Petrov"**][**"age"**];*// Выводит возраст***echo "<br>"**;  
**echo** $A[**"Petrov"**][**"email"**];*// Выводит e-mail***?>**

### **2.2.6 Тип object (объекты)**

Объект является одним из базовых понятий объектно-ориентированного программирования. Внутренняя структура объекта похожа на хэш, за исключением того, что для доступа к отдельным элементам и функциям используется оператор **->**, а не квадратные скобки.

Для инициализации объекта используется выражение *new*, создающее в переменной экземпляр объекта.

**<?php  
class** foo  
{  
 **function** do\_foo()  
 {  
 **echo "Doing foo."**;  
 }  
}  
  
$bar = **new** foo;  
$bar->do\_foo();  
**?>**

Более подробное рассмотрение объектов будет производится в лабораторных, посвященных [**PHP и ООП**](http://www.php.su/learnphp/phpoo/).

### **2.2.7 Тип resource (ресурсы)**

**Ресурс** – это специальная переменная, содержащая ссылку на внешний ресурс. Ресурсы создаются и используются специальными функциями. Полный перечень этих функций и соответствующих типов ресурсов представлен в Приложении 1.

**Преобразование в ресурс**

Поскольку тип ресурс содержит специальные указатели на открытые файлы, соединения с базой данных, область изображения и тому подобное, вы не можете преобразовать какое-либо значение в ресурс.

**Освобождение ресурсов**

В связи с системой подсчета ссылок, введенной в движке Zend PHP 4 автоматически определяется, что ресурс больше никуда не ссылается (как в Java). Когда это происходит, все ресурсы, которые использовались для данного ресурса, освобождаются сборщиком мусора. По этой причине маловероятно, что когда-либо будет необходимо освобождать память вручную, используя какую-нибудь free\_result функцию.

### **2.2.8 Тип NULL («пустой» тип)**

Специальное значение NULL говорит о том, что эта переменная не имеет значения. NULL – это единственно возможное значение типа NULL (пустой тип).

Переменная считается NULL если:

* ей была присвоена константа NULL;
* ей еще не было присвоено какое-либо значение;
* она была удалена с помощью [unset()](http://www.php.su/functions/?unset).

**<?php**$var = **NULL**;  
**?>**

### **2.2.9 Псевдотип mixed (смешанный)**

Псевдотип *mixed* говорит о том, что параметр может принимать множество (но не обязательно все) типов. Так, например, **gettype()** принимает все типы PHP, тогда как **str\_replace()** принимает строки и массивы.

### **2.2.10 Псевдотип number (числовой)**

Псевдотип *number* говорит о том, что параметр может быть либо [**integer**](http://www.php.su/learnphp/vars/?types#integer), либо [**float**](http://www.php.su/learnphp/vars/?types#float).

### **2.2.11 Псевдотип callback (обратного вызова)**

Некоторые функции, такие как [call\_user\_func()](http://www.php.su/functions/?call_user_func) или [usort()](http://www.php.su/functions/?usort) принимают в качестве параметра определенные пользователем callback-функции. Callback-функции могут быть не только простыми функциями, но также методами[объектов](http://www.php.su/learnphp/vars/?types#object), включая статические методы [классов](http://www.php.su/learnphp/phpoo/).

PHP-функция передается просто как строка ее имени. Вы можете передать любую встроенную или определенную пользователем функцию за исключением [array()](http://www.php.su/functions/?array), [echo()](http://www.php.su/functions/?echo), [empty()](http://www.php.su/functions/?empty), [eval()](http://www.php.su/functions/?eval), [exit()](http://www.php.su/functions/?exit), [isset()](http://www.php.su/functions/?isset), [list()](http://www.php.su/functions/?list), [print()](http://www.php.su/functions/?print) и [unset()](http://www.php.su/functions/?unset).

# **Задание**

Написать скрипты (программы), аналогичные приведенным примерам, которые бы использовали следующие типы данных: boolean, integer, float, string, array.