Лабораторная работа 1
Введение

Наберите в программе HateML:

```
<?php
//Это моя первая программа
echo "Hello World!";
?>
```

Сохраните, файл назвав index.php в папке **hello, которая находится в WebServ\wwwroot\htdocs\hello** и запустите этот файл в браузере, набрав **http://localhost/hello/**. В браузере должна появиться надпись "Hello World!". Если нет надписи - значит что-то не то.
Основное применение php является создание текста, но не просто текста, а текста имеющего знакомое название - HTML . Вот более наглядный пример:

```
<html>
<head>
<title>My First Page</title>
</head>
<body>
<?PHP
echo "Hello World!";
?>
</body>
</html>
```

Из чего состоит любой скрипт:

```
<?PHP
//Это моя первая программа
echo "Hello World!";
?>
```

Наша программа начинается с тега *<?php* и заканчивается *?>*.
Все, что помещается между этими двумя тегами - это php код.
Все, что пишут вне этих тегов - обычный html.

После тега *<?php* идет текст, перед которым стоят два слеша: //.
Это комментарий. Комментарий в программировании отличается от обычного тем, что это не просто бессмысленная фраза. Это полезная памятка.

Комментарий это обычный текст, который не обрабатывается интерпретатором.
Комментарии писать не обязательно, но считается хорошим тоном комментировать программу. (*Примечание редактора: читай как обязательная*).
Потом, когда скрипт написан и удачно забыт будет легче вспомнить что к чему.

```
<?PHP

// Это одно-строчный комментарий
# Это тоже одно-строчный комментарий
```

```
/*
 А это комментарий,
 который разбит на несколько строк
*/

echo "Hello World"; //Это тоже комментарий
echo "// А это не комментарий";
echo /* А вот это комментарий*/ "Hello World";

?>
```

echo "зарплата Мужа","зарплата Сына","зарплата Соседа","зарплата Дяди";
Ну и наконец самое главное - точка с запятой **;**
Программирование по сути это набор команд которые компьютер исполняет.
Для того чтобы компьютер отделил одну команду от другой - в конце каждой команды ставится точка с запятой, как в конец каждого предложения точка;

Практика.

Найдите ошибки в коде:
1.

```
<?PHP

echo Hello World;

?>
```

2.

```
<?PHP

echo "Hello World";
/Программа пишет фразу "Здравствуй Мир!"

?>
```

Основы.

Переменные выполняют работу - хранение данных в оперативной памяти. Именно поэтому переменная представлена в виде знака доллара и названия переменной. К примеру: $variable .

Сама по себе переменная существовать не может. Ей нужны её данные. При помощи присваивания мы кладём в переменную данные. Присваивание выглядит следующим образом: $var = что-то;

Правильное имя переменной должно начинаться с буквы или символа подчеркивания с последующими в любом количестве буквами, цифрами или символами подчеркивания.

Данные бывают разных типов: целые числа, дробные числа, наборы букв, наборы других данных и еще несколько.

Целое число (Integer)

Как вам известно, числа бывают положительные, отрицательные и в разных системах исчисления. Чаще всего подразумевается что целое это число из множества Z = {..., -2, -1, 0, 1, 2, ...}.

Предел совершенства наших компьютеров позволяет использовать в качестве целого числа, числа из диапазона Signed: −2,147,483,648 to +2,147,483,647.

```
<?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)

?>
```

Дробные числа (Float)

Числа с плавающей точкой (они же числа двойной точности или действительные числа) отличная кандидатура на хранение к примеру результата деления 7 на 3

```
<?PHP

$a = 1.234;
$b = 1.2e3; //   1.2 x 10^3
$c = 7E-
10; //   Из курса математики напомню что буквой e обозначают экспоненту.

?>
```

Как правило диапазон возможностей здесь немного шире как правило, ~1.8e308 с точностью около 14 десятичных цифр (64-битная система)

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.

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. . ..

Строки (String)

Строка - это набор символов. В PHP символ это то же самое, что и байт, это значит, что возможно ровно 256 различных символов.

Замечание: Нет никаких проблем, если строка очень велика. Практически не существует ограничений на размер строк, налагаемых PHP, так что нет абсолютно никаких причин беспокоиться об их длине.

Строки в php можно определить тремя способами: одинарными кавычками, двойными кавычками и heredoc синтаксисом.

```php
<?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"';
// Обрати внимание: в строке присутствует одинарная кавычка. Для того, чтобы ph
p
// понял, что это не конец строки, а часть ее, нам нужно ему об этом сообщить.
// Обратный слэш выполняет функцию Экранирования.

// Выведет: Это не вставит: \n новую строку
echo 'Это не вставит: \n новую строку';
// В операционных системах используются особые специальные символы в тексте
// которые помогают системе понять, где находится конец строки и начинается но
вая,
// где находится табуляция и где вообще заканчивается весь текст.
// Символ \n обозначает конец строки и переход на новую.
// Слэш является частью символа, а не экранированием.
// Если бы php всё таки вставил этот символ, то мы бы получили вывод в две стро
ки:
// Это не вставит:
// новую строку

// Выведет: Переменные $expand также $either не подставляются
echo 'Переменные $expand также $either не подставляются';

?>
```

Строка в двойных кавычках

Так же как и в одинарных, текст взятый в двойные кавычки это строка.

$var = "val";

Если строка определяется в двойных кавычках, либо при помощи heredoc, переменные внутри нее обрабатываются.

Если интерпретатор встречает знак доллара ($), он захватывает так много символов, сколько возможно, чтобы сформировать правильное имя переменной. Если вы хотите точно определить конец имени, заключайте имя переменной в фигурные скобки.

```php
<?PHP

$beer = 'Heineken';
echo "$beer's taste is great"; // работает, "'" это неверный символ для имени перемен
```

```
ной
echo "He drank some $beers";   // не работает, 's' это верный символ для имени пере
менной
echo "He drank some ${beer}s"; // работает
echo "He drank some {$beer}s"; // работает

?>
```

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.

Напомню что если в строке определенной кавычками встречается кавычка, нужно сообщить php что это часть строки - экранировать кавычку. Как в примере $var='*I\'m*';

```
<?PHP

echo <<<HEREDOC1
Меня зовут "$name". Я печатаю very$fast.
Теперь я вывожу very{$fast}.
Это должно вывести заглавную букву 'A': \x41
HEREDOC1;

?>
```

В данном примере строка "Меня зовут ... ... букву A" образовывается при помощи синтаксиса HereDoc. Обратите внимание что вместо кавычек в начале и конце строки стоит название строки - HEREDOC1. <<<*HEREDOC1* означает начало строки под названием HEREDOC1, а *HEREDOC1;* её конец.

Массив (Array)

Массив в php это холодильник хранящий набор продуктов.

```
<?PHP

$array = Array('Сыр','Колбаса');

?>
```

В данном примере в нашем холодильнике array находятся два продукта относящиеся к категории Строк (и потому заключены в кавычки).

Необычным тебе может показаться что счёт продуктов в программировании начинается с нуля. Именно поэтому в нашем массиве нулевой элемент будет 'Сыр', а первый элемент 'Колбаса'.

```
<?PHP

$array = Array('Сыр','Колбаса'); // Определяем массив с двумя строками
echo $array[0]; // Обращение к нулевому элементу. Выведет Сыр.
$array[1] = 'Мороженое'; // Присвоение первому элементу. заменяем колбасу на мо
локо
```

```
echo $array[1]; // Выведет Мороженое.

?>
```

Ассоциативные массивы

С таким же успехом как и в предыдущем примере, элементами массива могут быть не только строки, но и другие переменные. $array = Array($var1 , $var2);

У любой переменной есть название и есть значение, а это значит что в массив мы можем также записать пары данных в виде название-значение. Массив такого рода называется ассоциативным благодаря ассоциации названия и значения, а название в таком массиве называется ключом.

$arr = Array('key'=>'val' , 'key2'=>'val2');

Таким образом в массиве нет нулевого и первого элемента, но есть элемент key и элемент key2.

```
<?PHP

$arr = Array('key'=>'val' , 'key2'=>'val2'); // Определяем массив с двумя строками
echo $arr['key']; // Обращение к элементу с ключом key. Выведет val
echo $arr[0]; // Обращение к нулевому элементу. Ошибка. Нет такого элемента
$arr['key2'] = 'Мороженое'; // Присвоение элементу с ключом key2 нового значения
.
echo $arr['key2']; // Выведет Мороженое.

?>
```

Многомерные массивы

Массив может содержать любой тип данных в том числе и другой массив.

Массив массивов называется многомерным массивом.

Всего есть два типа массивов - числовые и ассоциативные.

Первый вид представляет из себя массив с ключами в виде чисел как в самом первом примере. Всё остальное является ассоциативным массивом.

```
<?PHP

$ARRAY = Array // числовой. Ключи 0,1 и 2
(
 Array('Сыр','Колбаса') , // числовой. Ключами являются 0 и 1
 Array('key'=>'val' , 'key2'=>'val2') , // Ассоц. Ключи key и key2
 Array('key3'=>'val3', 'Чипсы') // Тоже Ассоц. Ключи key3 и 0
);

// Для обращения к какому либо элементу многомерного массива, к примеру к val
3
// следует обращаться так:
echo $ARRAY[2]['key3'];
```

```
// В массиве ARRAY обращаемся сначала к элементу номер 2 [ Array('key3'=>'val3'
, 'Чипсы')  ]
// а там к элементу с ключом key3


$arr = array("somearray" => array(6 => 5, 13 => 9, "a" => 42));
echo $arr["somearray"][6];    // 5
echo $arr["somearray"][13];   // 9
echo $arr["somearray"]["a"];  // 42

?>
```

Создание массива и его наполнение

Содержание массива может быть задано как это было в примерах - заранее.

$arr = Array(1,2,3); // ключи 0,1 и 2

Так же как мы присваивали ранее элементу в массиве другое значение можно создать и новый элемент

$arr[3] = 5; // Создаётся элемент с индексом 3 и ему тут же присваивается число (integer) 5.

Попробуй предположить что произойдёт при создании элемента $arr[] = 6;.

В этом случае, создастся элемент в массиве, ему присвоится значение 6, а индекс с которым элемент запишется в массив будет максимальный целочисленные уже существующий индекс + 1.

Таким образом после $arr[3] = 5; максимальный целочисленный индекс стал 3.

При $arr[] = 6; максимальный индекс + 1 будет 4. Таким образом $arr[4] становится 6.

```
<?PHP

$arr = Array('key3'=>'val3', 'Чипсы') ;// напомним индексы key3 и 0
$arr[] = 'Банан'; // Попробуйте сказать какой индекс будет у элемента Банан ?

$arr = Array('key3' => 'val3', 'tasty' => 'Чипсы') ;
$arr[] = 'Банан'; // Попробуйте сказать какой индекс будет у элемента Банан теперь
?

/* ОТВЕТ:
в первом случае максимальный целочисленный индекс 0. 0+1 =1
значит индекс банана будет 1.

Во втором случае максимального целочисленного индекса нет вообще
поэтому новый целочисленный индекс будет 0 .
*/

?>
```

Объекты (Object)

Оглянитесь вокруг. Всё вокруг нас это объекты. Стул, бутылка, веревка, мыло, это всё объекты. У объектов есть свойства и функции.

Объект - Машины. Свойство - Цвет. Значение - белый.

Объект - Машины. Функция - бибикать. Действие - издаёт звук.

```php
<?PHP

// Определяем что такое машины, что она умеет делать и как.
class car
{
   var $color = 'White';
   function tut_tut()    {  echo "BIp BIp";  }
}

$honda = new car; // присваиваем переменной honda объект - Машину
echo $honda->color; // Выводим свойство машины - цвет

?>
```

Ресурс (Resource)

Ресурс представляет из себя указатель, ссылку, на внешний ресурс.

Представим автосервис с большим количеством машин, которым заливают по несколько литров масла в двигатель. Со свистом тормозов из подворотни вылетает феррари и паркуется на очередной сервисной парковке. Хозяин сервиса уже кричит рабочему, залить 5 литров масла и тычет большим пальцем в красную феррари.

Наш рабочий получает в данном случае два типа данных - число (литров масла) и указатель на машину, то-есть определение - какому именно объекту из всех вокруг нужно подлить чего-нибудь.

Подмечу, что рабочий получает от босса вовсе не саму машину, а лишь дескриптор (указатель) машины с которой предстоит работать. В php этим дескриптором является тип данных resource.

Null

Чтобы понять что такое Null нужно понять как хранит php переменные в памяти. Компьютерная память состоит из большого количества маленьких ячеек. В эти ячейки и записываются твои данные. Обращение к той или иной ячейке происходит при помощи идентификатора ячейки в памяти, её места.

На деле интерпретатор хранит связь между названием переменной и идентификатором ячейки в памяти. Мы же в свою очередь работаем с названием, вместо того чтобы самим запоминать идентификаторы ячеек, которые и без того, известны одному интерпретатору.

Когда мы присваиваем переменной значение в первый раз, интерпретатор выбирает ячейку в памяти, заносит туда наше значение и привязывает название переменной к определенному идентификатору.

Когда к названию переменной соответствует ячейка в памяти, переменная считается определенной. Пустая переменная означает что содержимое ячейки является числом ноль, строкой нулевой длины или вовсе отсутствует.

Значение Null сбрасывает привязку названия переменной к ячейке в памяти и переменной больше не соответствует никакая ячейка. Переменная просто никуда не указывает и ни к чему не ведет. В этом случае переменная считается неопределенной, так как название есть а привязка к ячейке отсутствует.

Присвоение переменной значения null можно делать так: $a = Null;

Заметьте что слово null не взято в кавычки, потому что это не строка с текстом.

Булев тип (Boolean)

Тип данных который включает в себя всего два значения
Правда и неправда. True и False. Используется в основном в условиях.
к примеру выражение ($a > $b) может быть правдой или не правдой.

Что это за странные типы данных которые я вижу в документации ?

mixed - обозначает что функция возвращает разные типы данных, в зависимости от обстоятельств

number - означает что возвращаемый результат представляет из себя integer или float

callback - в php функция вызывается как её название и скобочки. К примеру print().

Некоторые функции в php (к примеру call_user_func()) принимают в качестве параметров названия функций с которыми они будут работать. в таком случае параметр формата callback которые они требуют представляют из себя строку с названием функции без скобочек. К примеру call_user_func('increment', $a);

Возможно данный аспект будет подробнее рассмотрен в разделе работы с функциями.

Вопросы и задания на усвоение материала:

1. $a = '22'; Какого типа является сейчас переменная $a ?

2. $b = $a = 7; чему равно $b ? не забудьте что читается в лева на право.

3. $a = '46'; $b = '32'; Что больше, а или б ?

4. Какой тип переменной лучше всего подойдёт для хранения результата деления 2 на 4?

5. $a = 5; $b=2.5; Какого типа данных будет результат деления А на Б ?

6. Попробуйте присвоить переменной значение +21474836490000000000000000.

7. Выясните сколько это 2B в шестнадцатеричной системе ? (Присвойте переменной значение и выведите).

8. Можно ли переменной присвоить переменную ? Если да - покажите как. Если нет, объясните почему нет.

9. $a='Вася'; $b = 'Петя'; Поменяйте местами значения переменных .

10. Создайте одномерный числовой массив с одним элементом.
Добавьте туда еще один элемент.
Добавьте туда третий элемент представляющий из себя ассоциативный массив с одним элементом.
Добавьте в этот ассоц. массив еще один элемент.
Замените первый элемент в родительском массиве на второй элемент из ассоц. массива.

11. Создайте переменную типа Resource
