УВАЖАЕМЫЕ СТУДЕНТЫ! Законспектируйте в своей рабочей тетради по дисциплине приведенную лекцию (объемом 4-5 страницы).

Результаты работы, фотоотчет, предоставить преподавателю на e-mail: igor-gricenko-95@mail.ru в течении ТРЕХ дней.

При возникновении вопросов по приведенному материалу обращаться по следующему номеру телефона: (072)132-63-42

ВНИМАНИЕ!!! При отправке работы, не забывайте указывать ФИО студента, наименование дисциплины, дата проведения занятия (по расписанию).

Лекция № 8

Тема «Язык программирования PHP (типы данных и операторы)»

1. Назначение и история языка PHP

Язык PHP предназначен для написания динамических HTML-страниц. Программа на PHP встраивается в текст (код) HTML-страницы в виде одного или нескольких блоков, начинающихся строкой *<?php* и заканчивающихся строкой *?>*. Схематично структуру HTML-документа со скриптом на PHP можно изобразить так:

```
<HTML>
. . .
. . .

<?php
  //Начало скрипта на PHP
  оператор на PHP
. . .
  оператор на PHP
?>

. . .
. . .

<?php
  //Продолжение скрипта на PHP
  оператор на PHP
. . .
  оператор на PHP
```

```
?>

. . .
. . .

<?php
  //Конец скрипта на PHP
  оператор на PHP
. . .
  оператор на PHP
?>

. . .
. . .
</html>
```

PHP относится к императивным языкам и удобен для обработки текстовой информации. Для этого в нём есть следующие средства:

- тип данных *строка*;
- операция *конкатенация*;
- операция *интерполяция*;
- регулярные *выражения*.

Распространено мнение, что PHP основывается на языке C. Это не совсем так. PHP разрабатывался как альтернатива языку Perl, который также предназначен для обработки текстовой информации. Оба языка относятся к языкам с контекстно зависимым типом данных, в отличие от СИ с его статическим, принудительным назначением типов. Разработчик языка *Perl* Ларри Уолл постарался использовать всё лучшее, что было накоплено в языках программирования на момент появления в 1987 году этого языка, в том числе и многие черты языка C.

К сожалению, многое в PHP не только не лучше, чем в Perl, но и хуже. Вот пример оператора foreach, который служит для перебора всех элементов массива:

```
  Perl                          PHP
foreach $element (@Arr)       foreach ($Arr as $element )
{                   }         {                         }
```

При выполнении обоих операторов выполняются одни и те же действия: перебираются все элементы массива и над каждым элементом производятся операции, указанные в блоке. Оператор, написанный на Perl, читается естественно: "для каждого элемента $element массива @Arr ...". А вот как прочитать тот же оператор на PHP? Не понятно, зачем было переделывать удобный оператор. К сожалению ещё хуже обстоит дело с регулярными выражениями в PHP по сравнению с Perl.

PHP начал разрабатываться в 1994 году. Первая версия (релиз) появилась в 1995 году. Вторая - в 1997. Это были очень слабые версии. Тем не менее, PHP быстро завоевал популярность среди разработчиков, благодаря тому что его транслятор был встроен в самый распространённый веб-сервер Apache. Широко распространился набор LAMP (Linux, Apache, MySQL, PHP). Следующие версии PHP вышли:

- PHP 3.0 - 1998 г.
- PHP 4.0 - 2000 г.
- PHP 5.0 - 2004 г.
- релиз PHP 7.0 - 3.12.2015 г.

Шестая версия разрабатывалась с 2006 года, но в марте 2010 была признана бесперсперктивной. В настоящее время PHP имеет достаточно качественные трансляторы и большое количество модулей, содержащих функции для разработки веб-приложений.

2. Типы данных

PHP является языком с динамической типизацией. Это значит, что тип данных переменной выводится во время выполнения, и в отличии от ряда других языков программирования в PHP не надо указывать перед переменной тип данных.

PHP поддерживает восемь простых типа данных:

- boolean (логический тип)

– integer (целые числа)

– double (дробные числа)

– string (строки)

– array (массивы)

– object (объекты)

– resource (ресурсы)

– NULL

Существуют также несколько псевдотипов:

– mixed (смешанный)

– number (числовой)

– callback (обратного вызова)

Рассмотрим кратко перечисленные типы данных PHP.

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

Обычно вы используете некий оператор, который возвращает логическое выражение, а затем предает его управляющей конструкции.

```php
<?php
// == это оператор, который проверяет
// эквивалентность и возвращает булево значение
if ($action == "показать_версию") {
     echo "Версия 1.23";
}

// это не обязательно...
if ($show_separators == TRUE) {
     echo "<hr>\n";
}

// ...потому что вы можете просто написать
if ($show_separators) {
     echo "<hr>\n";
}
?>
```

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
<?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)
?>
```

Тип float (числа с плавающей точкой)

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

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'
?>
```

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
<?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
?>
```

Ассоциативные массивы

В PHP индексом массива может быть не только число, но и строка. Причем на строку не накладываются ограничения: она может содержать пробелы, специальные символы и быть любой длины.

Массивы, индексами которых являются строки, называются ассоциативными массивами. Индексы ассоциативных массивов называются ключами. Пример ассоциативного массива:

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

Многомерные массивы

Для создания массивов в PHP существует специальная инструкция array(). Ее удобно использовать для создания многомерных массивов. Приведем конкретный пример:

```php
<?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");
?>
```

Многомерные массивы похожи на записи в языке Pascal или структуры в языке C.

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
<?php
class foo
{
     function do_foo()
     {
         echo "Doing foo.";
     }
}

$bar = new foo;
$bar->do_foo();
?>
```

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 смотрите **здесь**.

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().

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

Псевдотип mixed (смешанный)

mixed говорит о том, что параметр может принимать множество (но не обязательно все) типов.

gettype(), например, принимает все типы PHP, тогда как str_replace() принимает строки и массивы.

Псевдотип number (числовой)

number говорит о том, что параметр может быть либо integer, либо float.

Псевдотип callback (обратного вызова)

Некоторые функции, такие как call_user_func() или usort() принимают в качестве параметра определенные пользователем callback-функции. Callback-функции могут быть не только простыми функциями, но также методами объектов, включая статические методы классов.

PHP-функция передается просто как строка ее имени. Вы можете передать любую встроенную или определенную пользователем функцию за исключением array(), echo(), empty(), eval(), exit(), isset(), list(), print() и unset()

.

Приведем примеры callback функций:

```php
<?php

// простой пример callback
function my_callback_function() {
     echo 'hello world!';
}
call_user_func('my_callback_function');

// примеры callback-метода
class MyClass {
     function myCallbackMethod() {
         echo 'Hello World!';
     }
}

// вызов метода статического класса без создания объекта
call_user_func(array('MyClass', 'myCallbackMethod'));

// вызов метода объекта
$obj = new MyClass();
call_user_func(array(&$obj, 'myCallbackMethod'));
?>
```

3. Операторы в php

Оператором называется нечто, принимающее одно или более значений (или выражений, если говорить на жаргоне программирования), и вычисляющее новое значение (таким образом, вся конструкция может рассматриваться как выражение).

Операторы можно сгруппировать по количеству принимаемых ими значений. Унарные операторы принимают только одно значение, например, **!** (оператор логического отрицания) или **++** (инкремент). Бинарные операторы принимают два значения; это, например, знакомые всем арифметические операторы **+** (плюс) и **-** (минус), большинство поддерживаемых в PHP операторов входят именно в эту категорию. Ну и, наконец, есть всего один тернарный оператор, **? :**, принимающий три значения, обычно его так и называют -- "тернарный оператор" (хотя, возможно, более точным названием было бы "условный оператор").

Приоритет оператора определяет, насколько "тесно" он связывает между собой два выражения. Например, выражение 1 + 5 * 3 вычисляется как 16, а не 18, поскольку оператор умножения ("*") имеет более высокий

приоритет, чем оператор сложения ("+"). Круглые скобки могут использоваться для принудительного указания порядка выполнения операторов. Например, выражение (1 + 5) * 3 вычисляется как 18.

Если операторы имеют равный приоритет, то будут ли они выполняться справа налево или слева направо определяется их ассоциативностью. К примеру, "-" является лево-ассоциативным оператором. Следовательно 1 - 2 - 3 сгруппируется как (1 - 2) - 3 и пересчитается в -4. С другой стороны "=" - право-ассоциативный оператор, так что $a = $b = $c сгруппируется как $a = ($b = $c).

Неассоциативные операторы с одинаковым приоритетом не могут использоваться совместно. К примеру 1 < 2 > 1 не будет работать в PHP. Выражение 1 <= 1 == 1, с другой стороны, будет, поскольку == имеет более низкий приоритет чем <=.

Использование скобок, кроме случаев когда они строго необходимы, может улучшить читаемость кода, группируя явно, а не полагаясь на приоритеты и ассоциативность.

В следующей таблице приведён список операторов, отсортированный по убыванию их приоритетов. Операторы, размещённые в одной строке имеют одинаковый приоритет и порядок их выполнения определяется исходя из их ассоциативности.

Порядок выполнения операторов		
Ассоциативность	**Оператор**	**Дополнительная информация**
(н/а)	clone new	clone и new
правая	**	арифметические операторы
(н/а)	++ -- ~ (int) (float) (string) (array) (object) (bool) @	типы и инкремент/декремент
левая	instanceof	типы
(н/а)	!	логические операторы

левая	* / %	арифметические операторы
левая	+ - .	арифметические операторы и строковые операторы
левая	<< >>	побитовые операторы
неассоциативна	< <= > >=	операторы сравнения
неассоциативна	== != === !== <> <=>	операторы сравнения
левая	&	побитовые операторы и ссылки
левая	^	побитовые операторы
левая	|	побитовые операторы
левая	&&	логические операторы
левая	||	логические операторы
правая	??	операторы сравнения с null
левая	? :	тернарный оператор
правая	= += -= *= **= /= .= %= &= |= ^= <<= >>= ??=	операторы присваивания
(н/а)	yield from	yield from
(н/а)	yield	yield
(н/а)	print	print
левая	and	логические операторы
левая	xor	логические операторы
левая	or	логические операторы