**Лабораторная работа 21**

Изучение синтаксиса PHP, работа с массивами и строковым типом данных.

Задание на карточке вариант 15.

Контрольные вопросы:

Операторы языка. Опишите синтаксис следующих операторов языка программирования php. Приведите примеры их использования.

1 условный оператор If/Then/Else

1. Условный оператор If/Then/Else

Синтаксис:

if (условие) {

// код, выполняемый если условие истинно

} elseif (другое\_условие) {

// код, выполняемый если другое условие истинно

} else {

// код, выполняемый если ни одно из условий не истинно

}Пример:

$number = 10;

if ($number > 0) {

echo "Число положительное.";

} elseif ($number < 0) {

echo "Число отрицательное.";

} else {

echo "Число равно нулю.";

}

2 оператор множественного выбора Select/Case

Синтаксис:

switch (выражение) {

case значение1:

// код, выполняемый если выражение равно значение1

break;

case значение2:

// код, выполняемый если выражение равно значение2

break;

default:

// код, выполняемый если ни одно из значений не совпало

}Пример:

$day = 3;

switch ($day) {

case 1:

echo "Понедельник";

break;

case 2:

echo "Вторник";

break;

case 3:

echo "Среда";

break;

default:

echo "Неизвестный день";

}

3 операторы организации циклов (while, do..while, for)

While:

while (условие) {

// код, выполняемый пока условие истинно

}

Пример:

$i = 0;

while ($i < 5) {

echo $i;

$i++;

}Do..While:

do {

// код, выполняемый хотя бы один раз

} while (условие);

Пример:

$i = 0;

do {

echo $i;

$i++;

} while ($i < 5);

For:

for (инициализация; условие; итерация) {

// код, выполняемый в цикле

}

Пример:

for ($i = 0; $i < 5; $i++) {

echo $i;

}

4 операторы передачи управления (break и continue).

break: завершает выполнение цикла или переключателя.

continue: пропускает текущую итерацию цикла и переходит к следующей.

Пример с break:

for ($i = 0; $i < 10; $i++) {

if ($i == 5) {

break; // прерывает цикл, когда i равно 5

}

echo $i . " ";

}Пример с continue:

for ($i = 0; $i < 10; $i++) {

if ($i % 2 == 0) {

continue; // пропускает четные числа

}

echo $i . " ";

}

5 операторы включения require() include() require\_once() include\_once()

require(): включает файл и вызывает ошибку, если файл не найден.

include(): включает файл, но не вызывает ошибку, если файл не найден.

require\_once(): включает файл только один раз, вызывает ошибку, если файл не найден.

include\_once(): включает файл только один раз, но не вызывает ошибку, если файл не найден.

Пример:

require 'header.php'; // Если файл не найден, будет фатальная ошибка

include 'footer.php'; // Если файл не найден, просто предупреждение

require\_once 'config.php'; // Включает файл только один раз

include\_once 'functions.php'; // Включает файл только один раз

Пользовательские функции, математические функции, работа с датой, временем и календарем.

6Для чего предназначены пользовательские функции? Опишите правила синтаксиса пользовательской функции в PHP? Приведите пример пользовательской функции и пример ее вызова из скрипта.

Пользовательские функции позволяют группировать код, который можно повторно использовать в разных частях программы. Это помогает улучшить структуру и читабельность кода, а также уменьшает дублирование.

Синтаксис:

function имя\_функции($аргументы) {

// код функции

return $значение; // возвращает значение

}Пример:

function add($a, $b) {

return $a + $b; // возвращает сумму двух чисел

}

// Вызов функции

$result = add(5, 10);

echo $result; // Вывод: 15

7 Что такое локальные и глобальные переменные? Приведите примеры локальных и глобальных переменных. Для каких целей при работе с пользовательскими функциями используется оператор return?

Локальные переменные: объявляются внутри функции и доступны только в пределах этой функции.

Глобальные переменные: объявляются вне всех функций и могут быть доступны в любой части скрипта. Для доступа к глобальным переменным внутри функции нужно использовать ключевое слово global.

Оператор return используется для возврата значения из функции. Это позволяет передавать результат выполнения функции обратно в место вызова.

Пример локальной переменной:

function myFunction() {

$localVar = "Я локальная переменная";

echo $localVar;

}

myFunction(); // Вывод: Я локальная переменная

// echo $localVar; // Ошибка: переменная не определена

Пример глобальной переменной:

$globalVar = "Я глобальная переменная";

function myFunction() {

global $globalVar; // Делаем глобальную переменную доступной

echo $globalVar;

}

myFunction(); // Вывод: Я глобальная переменная

8 Укажите назначение известных вам математических функций PHP. Приведите примеры их использования.

PHP предоставляет множество встроенных математических функций, таких как:

* abs($value) — возвращает абсолютное значение.
* round($value, $precision) — округляет число.
* sqrt($value) — возвращает квадратный корень.
* pow($base, $exp) — возводит число в степень.
* rand($min, $max) — генерирует случайное число в заданном диапазоне.

Примеры использования:

echo abs(-5); // Вывод: 5

echo round(3.6); // Вывод: 4

echo sqrt(16); // Вывод: 4

echo pow(2, 3); // Вывод: 8

echo rand(1, 100); // Вывод: случайное число от 1 до 100

9 Укажите назначение функций PHP, предназначенных для работы с датой, временем и календарем. Приведите примеры их использования.

PHP имеет множество функций для работы с датами и временем, такие как:

* date($format) — форматирует текущую дату и время.
* time() — возвращает текущее время в виде отметки времени.
* strtotime($time) — преобразует строку в метку времени.
* mktime($hour, $minute, $second, $month, $day, $year) — возвращает метку времени для заданной даты.

Примеры использования:

echo date('Y-m-d H:i:s'); // Вывод: текущая дата и время

echo time(); // Вывод: текущее время в секундах с 1970-01-01

echo strtotime('2023-10-01'); // Вывод: метка времени для 1 октября 2023

echo mktime(0, 0, 0, 10, 1, 2023); // Вывод: метка времени для 1 октября 2023

Массивы

1 Что понимается под массивом в PHP? Какие виды массивов можно обрабатывать в PHP?

Массив — это структура данных, которая позволяет хранить несколько значений в одной переменной. В PHP можно работать с несколькими видами массивов:

* Индексы массивов (обычные массивы)
* Ассоциативные массивы (с ключами)
* Многомерные массивы (массивы, содержащие другие массивы)

2 Приведите пример инициализации одномерного и двумерного массивов в PHP.

Одномерный массив:

$fruits = array("apple", "banana", "cherry");

Двумерный массив:

$students = array(

array("name" => "John", "age" => 20),

array("name" => "Jane", "age" => 22),

array("name" => "Doe", "age" => 21));

3 Опишите отличительные особенности ассоциативных массивов. Приведите пример инициализации ассоциативного массива в PHP. Опишите назначение оператора PHP foreach. Приведите пример его использования при обращении к ассоциативному массиву.

Ассоциативные массивы в PHP — это массивы, в которых ключи не являются просто числами, как в обычных (индексных) массивах, а могут быть строками. Это позволяет более удобно и наглядно организовывать данные, так как каждый элемент массива можно идентифицировать по уникальному ключу.

Отличительные особенности ассоциативных массивов:

* Ключи-строки: В ассоциативных массивах ключи могут быть строками, что позволяет задавать более понятные и описательные имена для значений.
* Гибкость: Ассоциативные массивы позволяют хранить данные в виде пар "ключ-значение", что делает их идеальными для хранения информации, где важно сопоставление значений с уникальными идентификаторами.
* Нет фиксированного порядка: В отличие от индексных массивов, ассоциативные массивы не имеют фиксированного порядка, что позволяет добавлять и удалять элементы без необходимости заботиться о последовательности индексов.

Пример инициализации ассоциативного массива в PHP:

$person = array(

"name" => "Alice",

"age" => 30,

"city" => "New York");

В этом примере мы создали ассоциативный массив $person, где ключи — это строки ("name", "age", "city"), а соответствующие значения — это данные о человеке.

Назначение оператора foreach в PHP

Оператор foreach предназначен для перебора элементов массива. Он позволяет легко итерироваться по массивам (как индексным, так и ассоциативным) без необходимости вручную управлять индексами. Это делает код более читаемым и понятным.

Пример использования foreach при обращении к ассоциативному массиву:

$person = array(

"name" => "Alice",

"age" => 30,

"city" => "New York");

foreach ($person as $key => $value) {

echo $key . ": " . $value . "\n";}

Вывод:

name: Alice

age: 30

city: New York

4 Укажите назначение известных вам функций PHP, предназначенных для обработки массивов. Приведите примеры их использования.

1. array\_push()

Назначение: Добавляет один или несколько элементов в конец массива. Пример:

$fruits = array("apple", "banana");

array\_push($fruits, "orange", "grape");

print\_r($fruits);

// Вывод: Array ( [0] => apple [1] => banana [2] => orange [3] => grape )

2. array\_pop()

Назначение: Удаляет последний элемент из массива и возвращает его. Пример:

$fruits = array("apple", "banana", "orange");

$lastFruit = array\_pop($fruits);

echo $lastFruit; // Вывод: orange

print\_r($fruits);

// Вывод: Array ( [0] => apple [1] => banana )

3. array\_shift()

Назначение: Удаляет первый элемент из массива и возвращает его. Пример:

$fruits = array("apple", "banana", "orange");

$firstFruit = array\_shift($fruits);

echo $firstFruit; // Вывод: apple

print\_r($fruits);

// Вывод: Array ( [0] => banana [1] => orange )

4. array\_unshift()

Назначение: Добавляет один или несколько элементов в начало массива. Пример:

$fruits = array("banana", "orange");

array\_unshift($fruits, "apple");

print\_r($fruits);

// Вывод: Array ( [0] => apple [1] => banana [2] => orange )

5. array\_merge()

Назначение: Объединяет два или более массивов в один. Пример:

$array1 = array("apple", "banana");

$array2 = array("orange", "grape");

$merged = array\_merge($array1, $array2);

print\_r($merged);

// Вывод: Array ( [0] => apple [1] => banana [2] => orange [3] => grape )

6. array\_slice()

Назначение: Извлекает часть массива и возвращает её как новый массив. Пример:

$fruits = array("apple", "banana", "orange", "grape");

$sliced = array\_slice($fruits, 1, 2); // Начинаем с индекса 1 и берем 2 элемента

print\_r($sliced);

// Вывод: Array ( [0] => banana [1] => orange )

7. array\_map()

Назначение: Применяет заданную функцию к каждому элементу массива и возвращает новый массив с результатами. Пример:

$numbers = array(1, 2, 3, 4);

$squared = array\_map(function($n) {

return $n \* $n;

}, $numbers);

print\_r($squared);

// Вывод: Array ( [0] => 1 [1] => 4 [2] => 9 [3] => 16 )

8. array\_filter()

Назначение: Фильтрует элементы массива, используя заданную функцию, и возвращает новый массив с элементами, которые прошли фильтр. Пример:

$numbers = array(1, 2, 3, 4, 5);

$evens = array\_filter($numbers, function($n) {

return $n % 2 == 0;

});

print\_r($evens);

// Вывод: Array ( [1] => 2 [3] => 4 )

9. array\_reduce()

Назначение: Применяет функцию к элементам массива, сводя их к одному значению. Пример:

$numbers = array(1, 2, 3, 4);

$sum = array\_reduce($numbers, function($carry, $item) {

return $carry + $item;

}, 0);

echo $sum; // Вывод: 10

10. in\_array()

Назначение: Проверяет, существует ли значение в массиве. Пример:

$fruits = array("apple", "banana", "orange");

if (in\_array("banana", $fruits)) {

echo "Banana is in the array."; // Вывод: Banana is in the array.

}

Строки

1 Что понимается под строкой в PHP? Приведите примеры задания строковых данных 3 различными способами:

1.одинарными кавычками

2. двойными кавычками

3. heredoc-синтаксисом

В чем состоит отличие каждого из них?

В PHP строка — это последовательность символов, заключенная в одинарные (') или двойные (") кавычки, а также может быть определена с помощью heredoc-синтаксиса. Строки могут содержать текст, числа и специальные символы.

Примеры задания строковых данных:

1. Одинарные кавычки:

$singleQuoteString = 'Это строка в одинарных кавычках.';

echo $singleQuoteString; // Вывод: Это строка в одинарных кавычках.

2. Двойные кавычки:

$name = "Alice";

$doubleQuoteString = "Привет, $name!";

echo $doubleQuoteString; // Вывод: Привет, Alice!

3. Heredoc-синтаксис:

$heredocString = <<<EOD

Это строка, написанная с помощью heredoc-синтаксиса.

Она может занимать несколько строк и поддерживает переменные, такие как $name.

EOD;

echo $heredocString;

// Вывод: Это строка, написанная с помощью heredoc-синтаксиса.

// Она может занимать несколько строк и поддерживает переменные, такие как Alice.

Отличия:

* Одинарные кавычки: Строки заключенные в одинарные кавычки не интерпретируют специальные символы (кроме \\ и \'). Переменные внутри таких строк не будут заменены на их значения.
* Двойные кавычки: Строки в двойных кавычках интерпретируют специальные символы и позволяют использовать переменные, которые будут заменены на их значения.
* Heredoc-синтаксис: Позволяет задавать многострочные строки. Переменные внутри heredoc также интерпретируются. Заканчивается на строке, содержащей только идентификатор (в данном случае EOD).

2 Что понимается под строкой в PHP? Как обратиться к отдельному символу в строке? Приведите пример. Какие операции определены над строками в PHP? Приведите примеры.

В PHP можно обратиться к отдельному символу строки, используя квадратные скобки или функцию mb\_substr() для многобайтовых строк.

Пример:

$string = "Привет";

$firstChar = $string[0]; // Обращение к первому символу

echo $firstChar; // Вывод: П

// Использование функции mb\_substr для многобайтовых строк

$secondChar = mb\_substr($string, 1, 1);

echo $secondChar; // Вывод: р

Операции, определенные над строками в PHP:

Конкатенация: Объединение строк с помощью оператора ..

$str1 = "Hello, ";

$str2 = "world!";

$result = $str1 . $str2;

echo $result; // Вывод: Hello, world!

Длина строки: Использование функции strlen().

$string = "Привет";

echo strlen($string); // Вывод: 6 (для многобайтовых строк используйте mb\_strlen)

Поиск подстроки: Использование функции strpos().

$string = "Hello, world!";

$position = strpos($string, "world");

echo $position; // Вывод: 7

Замена подстроки: Использование функции str\_replace().

$string = "Hello, world!";

$newString = str\_replace("world", "PHP", $string);

echo $newString; // Вывод: Hello, PHP!

3 Укажите назначение известных вам функций PHP, предназначенных для обработки строк. Приведите примеры их использования.

Вот некоторые известные функции для работы со строками и примеры их использования:

* strlen(): Возвращает длину строки.

$string = "Hello, world!";

echo strlen($string); // Вывод: 13

* strpos(): Находит позицию первого вхождения подстроки.

$string = "Hello, world!";

$position = strpos($string, "world");

echo $position; // Вывод: 7

* trim(): Удаляет пробелы (или другие символы) из начала и конца строки.

$string = " Hello, World! ";

$trimmedString = trim($string);

echo $trimmedString; // Вывод: Hello, World!

* substr(): Извлекает подстроку из строки.

$string = "Hello, World!";

$substring = substr($string, 7, 5); // Извлекаем 5 символов, начиная с позиции 7

echo $substring; // Вывод: World