УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«МОГИЛЕВСКИЙ ГОСУДАРСТВЕННЫЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ»

УТВЕРЖДАЮ
Директор колледжа
_____ С.Н.Козлов
08.12.2025

ВЕБ-ПРОГРАММИРОВАНИЕ НА СТОРОНЕ СЕРВЕРА

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ
ПО ИЗУЧЕНИЮ УЧЕБНОГО ПРЕДМЕТА,
ЗАДАНИЯ НА ДОМАШНЮЮ КОНТРОЛЬНУЮ РАБОТУ
ДЛЯ УЧАЩИХСЯ ЗАОЧНОЙ ФОРМЫ ПОЛУЧЕНИЯ ОБРАЗОВАНИЯ
ПО СПЕЦИАЛЬНОСТИ 5-04-0612-02 «РАЗРАБОТКА И
СОПРОВОЖДЕНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ
ИНФОРМАЦИОННЫХ СИСТЕМ»

2025

Автор: Денисовец Д.А., преподаватель высшей квалификационной категории учреждения образования «Могилевский государственный политехнический колледж»

Рецензент: Карманов А.В., преподаватель первой квалификационной категории учреждения образования «Могилевский государственный политехнический колледж»

Разработано на основе учебной программы учреждения образования по учебному предмету «Веб-программирование на стороне сервера» профессионального компонента учебного плана учреждения образования по специальности 5-04-0612-02 «Разработка и сопровождение программного обеспечения информационных систем» для реализации образовательной программы среднего специального образования, обеспечивающей получение квалификации специалиста со средним специальным образованием, утвержденной директором колледжа, 2025.

Обсуждено и одобрено
на заседании цикловой комиссии
специальностей в области программного
обеспечения информационных систем
Протокол № ___ от_____
Председатель цикловой комиссии
_____Д.А.Денисовец

Пояснительная записка

Учебный предмет «Веб-программирование на стороне сервера» предназначен для реализации образовательной программы среднего специального образования по специальности 5-04-0612-02 «Разработка и сопровождение программного обеспечения информационных систем» и предусматривает изучение языка программирования общего назначения PHP, особенностей создания серверных приложений с использованием платформы Node.js, установки и настройки веб-сервера, работы с основными серверными фреймворками Node.js.

Основной целью изучения учебного предмета «Веб-программирование на стороне сервера» является формирование профессиональной компетентности будущих специалистов в области использования и создания серверных приложений с использованием платформы Node.js, установки и настройки веб-сервера, работы с основными серверными фреймворками Node.js.

В процессе преподавания учебного предмета учитываются межпредметные связи программного учебного материала с такими учебными предметами учебного плана учреждения образования по специальности, как «Основы алгоритмизации и программирования», «Системы управления базами данных», «Конструирование программ и языки программирования», «Программные средства создания Интернет-приложений».

В ходе изложения программного учебного материала необходимо руководствоваться нормативными правовыми актами, техническими нормативными правовыми актами, соблюдать единство терминологии и обозначений, обеспечивать формирование универсальных, профессиональных компетенций, установленных в образовательном стандарте по соответствующей специальности.

В результате изучения учебного предмета «Веб-программирование на стороне сервера» учащиеся должны:

знать на уровне представления:

назначение веб-сервера;

назначение директив конфигурационного файла веб-сервера Apache;

различия веб-программирования на стороне клиента и на стороне сервера;

особенности применения и назначение основных фреймворков;

принципы работы с базами данных;

знать на уровне понимания:

особенности языка PHP, назначение и правила использования основных конструкций языка и стандартных функций;

особенности создания серверных приложений с использованием платформы Node.js;

уметь:

устанавливать и настраивать веб-сервер Apache;

устанавливать и настраивать NPM;

создавать сервер средствами Node.js;

работать с основными серверными фреймворками Node.js;

использовать события Node.js;

разрабатывать серверную часть приложения;

организовывать работу с базой данных.

Учебной программой определены цели изучения каждой темы, спрогнозированы результаты их достижения в соответствии с уровнями усвоения учебного материала.

Для закрепления теоретического материала и формирования у учащихся необходимых умений учебной программой предусмотрено проведение лабораторных работ.

В учебной программе по учебному предмету приведены критерии оценки результатов учебной деятельности учащихся, разработанные в соответствии с Правилами проведения аттестации учащихся, курсантов при освоении содержания образовательных программ среднего специального образования; перечень средств обучения, необходимых для обеспечения образовательного процесса.

В результате изучения учебного предмета учащиеся выполняют 1 домашнюю контрольную работу и 1 обязательную контрольную работу.

Общие методические рекомендации по выполнению домашней контрольной работы №1

Для выполнения домашней контрольной работы необходимо изучить материал курса. Изучение ведется по программе и при использовании источников литературы, приведенных в данных методических рекомендациях.

Основным методом изучения учебного предмета является самостоятельная работа, которая должна проводиться в последовательности, предусмотренной программой учебного предмета, и быть обязательно систематической.

Задания на домашнюю контрольную работу №1 разработаны в количестве 100 вариантов в соответствии с программой курса. Номера заданий выбираются в соответствии с двумя последними цифрами шифра учащегося, на пересечении соответствующей строки с соответствующим столбцом из таблицы 3.

Каждый вариант содержит 4 задания: один теоретический вопрос и три практических задания.

Для выполнения домашней контрольной работы вначале изучается теория и приведенные примеры решения задач.

Первое одно задание относятся к теоретическому содержанию учебного предмета и требуют ознакомления с соответствующими литературными источниками. Требуется раскрыть содержание теоретических вопросов. Привести примеры. Объем – около трех страниц.

Следующие три задания – задачи, в которых нужно составить программу на предложенном языке.

При оформлении домашней контрольной работы следует придерживаться следующих требований:

- работа выполняется в отдельной тетради или на листах А4 (шрифт 12-14, межстрочный интервал - одинарный). Следует пронумеровать страницы и оставить на них поля: справа – не менее 3 см для замечаний преподавателя, остальные поля – 2,5 см;
- на титульном листе указывается учебный предмет и номер работы, номер учебной группы, фамилия, имя, отчество учащегося, шифр;
- ответ следует начинать с номера и полного названия вопроса, он должен содержать текст программы с краткими, но достаточно обоснованными, пояснениям;

− в конце работы следует указать список используемых источников, которым вы пользовались, проставить дату выполнения работы и подпись;

− учащиеся должны соблюдать абзац, всякую новую мысль учащийся должен начинать с новой строки;

− при вставке графических объектов или таблиц они должны быть пронумерованы в каждом вопросе отдельно или допускается сквозная нумерация на протяжении всей домашней контрольной работы. На рисунок или таблицу обязательно должна быть ссылка по тексту ответа на вопрос;

− не допускается использовать в работе авторские обороты;

− к домашней контрольной работе прикладывается диск с файлами, содержащими программы решения задачи;

− если в работе допущены недочеты или ошибки, то учащийся должен выполнить все указания преподавателя, сделанные в рецензии;

− домашняя контрольная работа должна быть выполнена в срок (в соответствии с учебным графиком);

− учащиеся, не имеющие зачета по домашней контрольной работе, к экзамену не допускаются;

− перед экзаменом зачтенные домашние контрольные работы предоставляются преподавателю;

− незачтенные работы исправляются в соответствии с замечаниями преподавателя и повторно сдаются для проверки на заочное отделение.

Требования к оформлению задач:

− для каждого файла задачи привести программный код;

− первой строкой программного кода должен быть комментарий, содержащий имя файла;

− под программным кодом описать все теги, инструкции и функции, используемые в программе: их назначение, результат действия.

К домашней контрольной работе прикладывается диск с выполненными файлами. Каждая задача оформляется в отдельной папке. Имена папкам даются по номеру задачи.

Критерии оценки домашней контрольной работы №1

Домашняя контрольная работа, признанная преподавателем удовлетворительной и содержащая 75% положенного объема, оценивается отметкой «зачтено».

Домашняя контрольная работа оценивается отметкой «не зачтено», если:

- выполнена не в соответствии с вариантом;
- не раскрыто основное содержание теоретического вопроса (15%) и есть незначительные недочеты в практических заданиях (в сумме более 10%);
- не выполнено одно практическое задание (15%) и есть незначительные недочеты в остальных заданиях (в сумме более 10%);
- не выполнено два задания;
- нет диска с рабочими программами;
- есть существенные недочеты в нескольких практических заданиях (в сумме более 25%).

Учебная программа учебного предмета «Веб-программирование на стороне сервера» и методические рекомендации по его изучению

Введение

Цели и задачи учебного предмета «Веб-программирование на стороне сервера», связь с иными учебными предметами, значение в формировании профессиональных компетенций специалиста.

Обзор технологий создания веб-сайтов.

Назначение различных инструментов веб-разработки. Этапы создания веб-сайтов.

Современное состояние и перспективы развития интернет-технологий.

Литература: [1], с.353-356

Вопросы для самоконтроля

1 Назовите цели и задачи учебного предмета «Веб-программирование на стороне сервера».

2 Выскажите общее суждение о связи с другими учебными предметами, значении в формировании профессиональных компетенций специалиста.

3 Назовите этапы создания веб-сайтов.

4 Выскажите общее суждение о современном состоянии и перспективах развития интернет- технологий.

Раздел 1 Основы создания серверных веб-приложений на языке PHP

Тема 1.1 Механизм работы веб-сервера. Назначение протокола HTTP

Понятие веб-сервера. Задачи, принцип работы, механизм работы веб-сервера.

Понятие протокола HTTP. Назначение протокола HTTP. Методы протокола HTTP (get, post, put, delete).

Литература: [2], с.32-48

Вопросы для самоконтроля

1 Дайте определение понятию «Веб-сервер».

2 Опишите задачи, принцип и механизм работы веб-сервера.

3 Дайте определение понятию «протокол HTTP».

4 Объясните назначение и методы протокола HTTP (get, post, put, delete).

Тема 1.2 Установка и настройка веб-сервера

Виды веб-серверов (Apache, Nginx): преимущества и недостатки.

Особенности веб-серверов. Установка и настройка веб-сервера: назначение и использование. Конфигурационные файлы веб-сервера.

Установка и настройка PHP. Конфигурация PHP.ini: назначение и использование.

Литература: [2], с.49-66

Вопросы для самоконтроля

1 Выскажите общее суждение о видах веб-серверов (Apache, Nginx), их преимуществах и недостатках.

2 Опишите особенности, установку и настройку веб-сервера.

3 Объясните назначение и использование конфигурационных файлов веб-сервера, установку и настройку PHP, назначение и использование конфигурации PHP.ini.

Тема 1.3 Язык программирования PHP

Основы синтаксиса PHP: структура PHP-сценария, основные элементы синтаксиса, способы внедрения PHP-сценария в веб-документ.

Передача данных формы PHP-сценарию.

Синтаксис определения и использования функций в PHP.

Литература: [1], с.357-372, 386-388

Вопросы для самоконтроля

1 Объясните структуру PHP-сценария, основные элементы синтаксиса, способы внедрения PHP-сценария в веб-документ, передачу данных формы PHP-сценарию.

2 Опишите синтаксис определения и использования функций в PHP.

Тема 1.4 Работа с массивами и строками

Массивы. Ассоциативные массивы. Многомерные массивы.

Основные операции работы с массивами: создание, доступ к элементам, изменение, удаление элементов, поиск элементов в массиве. Базовые инструкции работы с массивами: list(), array().

Операции над массивами: удаление, слияние, сортировка.

Строки. Основные операции работы со строками: создание, доступ к элементам, изменение, удаление элементов.

Операции над строками: конкатенация, сравнение, замена. Базовые функции работы со строками: strlen(), strpos(), substr(), strcmp(), strcasecmp().

Литература: [1], с.396-417

Вопросы для самоконтроля

1 Дайте определения понятиям «массив», «ассоциативный массив», «многомерный массив».

2 Опишите основные операции работы с массивами (создание, доступ к элементам изменение, удаление элементов, поиск элементов в массиве), базовые инструкции работы с массивами: list(), array(), основные операции над массивами (удаление, слияние, сортировка).

3 Дайте определение понятию «строк».

4 Опишите основные операции работы со строками (создание, доступ к элементам, изменение, удаление элементов), базовые функции работы со строками: strlen(), strpos(), substr(), strcmp(), strcasecmp(), основные операции над строками (конкатенация, сравнение, замена).

Тема 1.5 Условия, циклы, стандартные функции

Условный оператор. Условные конструкции: операторы сравнения, логические операторы, тернарный оператор.

Циклы. Операторы организации циклов: структура, назначение, правила выполнения и использования операторов организации циклов.

Работа с циклическими операторами в языке PHP.

Функции. Стандартные функции PHP для работы с массивами и строками. Стандартные функции PHP для работы с файлами и каталогами, датой и временем.

Литература: [1], с.372-386

Вопросы для самоконтроля

1 Дайте определение понятию «условный оператор».

2 Опишите условные конструкции: операторы сравнения, логические операторы, тернарный оператор.

3 Дайте определение понятию «цикл».

4 Опишите структуру, назначение, правила выполнения и использования операторов организации циклов, работу с циклическими операторами в языке PHP.

5 Дайте определение понятию «функция».

6 Объясните стандартные функции PHP для работы с массивами и строками, файлами и каталогами, датой и временем.

Тема 1.6 Запросы get и post, обработка данных форм

Понятие запроса. Структура запроса. Методы передачи данных форм в PHP: get и post. Способы обработки данных форм средствами PHP.

Литература: [1], с.388-396

Вопросы для самоконтроля

1 Дайте определение понятию «запрос».

2 Опишите структуру запроса, методы передачи данных форм в PHP (get и post), способы обработки данных форм средствами PHP.

Тема 1.7 Работа с файловой системой с помощью языка PHP

Файловая система. Файлы. Текстовые и бинарные файлы.

Методы работы с файлами: создание, открытие, чтение, запись, удаление.

Каталоги. Методы работы с каталогами: создание, удаление, перемещение, переименование, получение содержимого.

Литература: [1], с.417-439

Вопросы для самоконтроля

1 Дайте определения понятиям «файловая система», «файл», «текстовый и бинарный файлы».

2 Опишите методы работы с файлами: создание, открытие, чтение, запись, удаление.

3 Дайте определение понятию «каталог».

4 Опишите методы работы с каталогами: создание, удаление, перемещение, переименование, получение содержимого.

Тема 1.8 Организация сессии, работа с cookie

Понятие, назначение cookie. Понятие, назначение HTTP-заголовков ответа сервера.

Понятие, назначение кэширования. Управление кэшированием.

Понятие, назначение буферизации. Понятие, назначение хэшировании. HTTP-аутентификация.

Сессии. Управление сессиями в PHP: принципы, основные функции. Протокол OAuth, его назначение.

Литература: [1], с.439-478

Вопросы для самоконтроля

1 Дайте определение понятию «cookie».

2 Опишите назначение cookie, HTTP-заголовков ответа сервера.

3 Дайте определение понятию «кэширование».

4 Объясните управление кэшированием.

5 Дайте определение понятиям «буферизация» и «хэширование».

6 Опишите HTTP-аутентификацию.

7 Дайте определение понятию «сессия».

8 Опишите принципы и основные функции управления сессиями в PHP, назначение протокола OAuth.

Тема 1.9 Функции. Библиотека стандартных функций

Понятие функции. Область видимости переменных. Виды функций: вложенные, условно определяемые. Назначение функций.

Библиотека стандартных функций.

Понятие пользовательской функции. Общий синтаксис написания пользовательских функций.

Особенности применения функций для решения практических задач.

Литература: [2], с.131-142

Вопросы для самоконтроля

1 Дайте определения понятиям «функция», «область видимости переменных».

2 Опишите виды функций: вложенные, условно определяемые, назначение функции.

3 Дайте определение понятию «библиотека стандартных функций».

4 Дайте определение понятию «пользовательская функция».

5 Опишите общий синтаксис написания пользовательских функций, особенности применения функций для решения практических задач.

Тема 1.10 Основы объектно-ориентированного программирования в PHP

Объектно-ориентированное программирование (ООП). Преимущества объектно-ориентированного программирования (ООП).

Основные концепции объектно-ориентированного программирования (ООП): инкапсуляция, наследование, полиморфизм.

Литература: [2], с.142-143

Вопросы для самоконтроля

1 Дайте определение понятию «объектно-ориентированное программирование (ООП).

2 Опишите преимущества объектно-ориентированного программирования (ООП), основные концепции объектно-ориентированного программирования (ООП): инкапсуляция, наследование, полиморфизм.

Тема 1.11 Классы и объекты

Класс. Члены класса. Синтаксис описания класса. Модификаторы доступа к членам класса. Конструкторы и деструкторы: создание и использование. Свойства класса: понятие, типы. Методы класса: понятие, типы (статические и нестатические). Специальные методы (get(),__set(),__call()).

Объект как экземпляр класса. Отношения между классами: наследование, вложение. Создание объектов. Реализация наследования с помощью ключевого слова extends. Переопределение методов и свойств в классах-наследниках.

Литература: [2], с.144-153

Вопросы для самоконтроля

1 Дайте определения понятиям «класс», «член класса».

2 Опишите свойства и методы класса, синтаксис описания класса, модификаторы доступа к членам класса, создание и использование конструкторов и деструкторов, типы свойств, типы методов, специальные методы (__get(),__set(),__call()).

3 Дайте определения понятиям «объект», «наследование», «вложение».

4 Объясните создание объектов, реализацию наследования с помощью ключевого слова extends, переопределение методов и свойств в классах-наследниках.

Тема 1.12 Создание пользовательских функций в JavaScript

Трейт: понятие и назначение. Создание и использование трейтов в PHP. Разрешение конфликтов в трейтах. Функции для работы с трейтами в ООП на PHP.

Интерфейс: понятие и назначение. Реализация интерфейсов в классах. Множественное наследование интерфейсов. Функции для работы с интерфейсами в ООП на PHP.

Пространство имен. Организация кода с помощью пространства имен. Импорты в пространстве имен. Глобальные пространства имен и их использование. Разрешение конфликтов имен в пространстве имен.

Литература: [4], с.135-149

Вопросы для самоконтроля

1 Дайте определение понятию «трейт».

2 Опишите назначение трейта, создание и использование трейтов в PHP, разрешение конфликтов в трейтах, функции для работы с трейтами в ООП на PHP.

3 Дайте определение понятию «интерфейс».

4 Объясните назначение интерфейса, реализацию интерфейсов в классах, множественное наследование интерфейсов, функции для работы с интерфейсами в ООП на PHP.

5 Дайте определение понятию «пространство имен».

6 Опишите организацию кода с помощью пространства имен, импорты в пространстве имен, глобальные пространства имен и их использование, разрешение конфликтов имен.

Тема 1.13 Клиент-серверное взаимодействие

Общие сведения о клиент-серверном взаимодействии. Способы организации клиент-серверного взаимодействия.

Функции PHP для работы с базой данных (БД) MySQL: установка соединения с сервером MySQL, выбор базы данных (БД), обработка ошибок, выполнение запросов к серверу базы данных (БД), обработка результатов запроса, получение информации о результатах SQL-запросов.

Технология AJAX. Асинхронное взаимодействие

Литература: [2], с.202-320

Вопросы для самоконтроля

1 Выскажите общее суждение о клиент-серверном взаимодействии.

2 Опишите способы организации клиент-серверного взаимодействия.

3 Опишите функции PHP для работы с БД MySQL: установка соединения с сервером MySQL, выбор базы данных (БД), обработка ошибок, выполнение запросов к серверу базы данных (БД), обработка результатов запроса, получение информации о результатах SQL-запросов.

4 Выскажите общее суждение о технологии AJAX.

5 Объясните суть асинхронного взаимодействия.

Раздел 2 Основы создания серверных веб-приложений на языке JavaScript

Тема 2.1 Введение в Node.js. Управление зависимостями

Общие сведения о Node.js. Основные концепции Node.js: событийно-ориентированная архитектура, однопоточность, асинхронность.

Модульная система в Node.js.

Зависимости. Управление зависимостями в Node.js

Литература: [3], с.16-35; 76-85

Вопросы для самоконтроля

1 Выскажите общее суждение о Node.js, основных концепциях Node.js: событийно-ориентированная архитектура, однопоточность, асинхронность.

2 Опишите модульную систему в Node.js.

3 Дайте определение понятию «зависимости».

4 Опишите управление зависимостями в Node.js.

Тема 2.2 Введение в NPM-менеджер пакетов для Node.js

Общие сведения о NPM-менеджер пакетов для Node.js. Основные команды NPM: install, uninstall, update, list.

Управление зависимостями в проекте с помощью NPM.

Использование NPM для установки и удаления пакетов. Создание собственных пакетов и публикация их в NPM-реестре.

Литература: [3], с.86-2110

Вопросы для самоконтроля

1 Выскажите общее суждение о NPM-менеджере пакетов для Node.js.

2 Опишите основные команды NPM: install, uninstall, update, list, управление зависимостями в проекте с помощью NPM, использование NPM для установки и удаления пакетов, создание собственных пакетов и публикацию их в NPM-реестре.

Тема 2.3 Цикл событий. События в Node.js

Событие. Обработчик события. Цикл событий.

Основные принципы работы цикла событий. Генерация события и тип EventEmitter. Создание и эмитирование событий. Привязка обработчиков событий. Передача параметров событию.

Работа с событиями ввода-вывода.

Создание и использование собственных событий.

Организация цикла событий с помощью таймеров в Node.js.

Методы для работы с таймерами.

Литература: [3], с.36-55

Вопросы для самоконтроля

1 Выскажите общее суждение о событии, об обработчике события, цикле событий.

2 Опишите основные принципы работы цикла событий, генерацию события и тип EventEmitter, создание и эмитирование событий, привязку обработчиков событий, передачу параметров событию, работу с событиями ввода-вывода, создание и использование собственных событий.

3 Выскажите общее суждение об организации цикла событий с помощью таймеров в Node.js.

4 Опишите методы для работы с таймерами.

Тема 2.4 Асинхронное программирование

Общие сведения об асинхронном программировании.

Преимущества и недостатки асинхронного программирования.

Колбэки (Callbacks) в Node.js и их использование.

Промисы (Promises) в Node.js и их использование.

Преобразование колбэков в промисы с использованием util.promisify.

Асинхронные функции (Async/Await) в Node.js.

Обработка ошибок при работе с асинхронным кодом.

Литература: [3], с. 55-75

Вопросы для самоконтроля

1 Дайте определение понятию «модуль» в языке программирования JavaScript.

1 Выскажите общее суждение об асинхронном программировании.

2 Объясните преимущества и недостатки асинхронного программирования.

3 Дайте определения понятиям «колбэк» и «промис» в Node.js.

4 Объясните использование колбэков и промисов в Node.js, преобразование колбэков в промисы с использованием util.promisify.

5 Опишите асинхронные функции (Async/Await) в Node.js, обработку ошибок при работе с асинхронным кодом.

Тема 2.5 Работа с файловой системой

Файловая система. Основные модули Node.js для работы с файловой системой.

Методы работы с файлами: создание, чтение, запись и удаление файлов.

Методы работы с каталогами: создание, удаление, перемещение, переименование, получение содержимого.

Потоки данных. Применение потоков данных при работе с файловой системой.

Использование модуля path для работы с путями к файлам и директориям.

Операции с путями: объединение, нормализация, разбор.

Обработка ошибок при работе с файлами и каталогами.

Литература: [3], с.156-180

Вопросы для самоконтроля

1 Выскажите общее суждение о файловой системе, основных модулях Node.js для работы с файловой системой.

2 Опишите методы работы с файлами: создание, чтение, запись и удаление файлов, методы работы с каталогами: создание, удаление, перемещение, переименование, получение содержимого.

3 Выскажите общее суждение о потоках данных.

4 Объясните применение потоков данных при работе с файловой системой, использование модуля path для работы с путями к файлам и директориям, операции с путями: объединение, нормализация, разбор, обработку ошибок при работе с файлами и каталогами.

Тема 2.6 Сведения о HTTP-сервере на Node.js

Понятие веб-сервера. Задачи, принцип работы, механизм работы веб-сервера.

Протокол HTTP. Назначение протокола HTTP.

Использование модуля http в Node.js. Настройка и запуск HTTP-сервера.

Маршрутизация в Node.js. Объект запроса HTTP-сервера на Node.js. Обработка различных типов запросов: get, post, put, delete.

Файлы ресурсов в Node.js

Литература: [3], с.135-155

Вопросы для самоконтроля

1 Дайте определение понятию «веб-сервер».

2 Объясните задачи, принцип и механизм работы веб-сервера.

3 Дайте определение понятию «протокол HTTP».

4 Объясните назначение протокола HTTP, использование модуля http в Node.js, настройку и запуск HTTP-сервера, маршрутизацию в Node.js, объект запроса HTTP-сервера на Node.js, обработку различных типов запросов: get, post, put, delete.

5 Опишите файлы ресурсов в Node.js.

Тема 2.7 Серверные фреймворки Koa.js и Express.js

Принципы работы фреймворков Koa.js и Express.js.

Создание и настройка маршрутов в Koa.js и Express.js.

Параметры маршрутов и передача данных через URL. Формирование и отправка ответов на запросы. Post-запросы и отправка форм.

Переадресация.

Понятие Middleware и его роль в обработке запросов. Создание и применение Middleware в Koa.js и Express.js.

Аутентификация и авторизация в Koa.js и Express.js.

Литература: [3], с.226-253

Вопросы для самоконтроля

1 Опишите принципы работы фреймворков Koa.js и Express.js.

2 Объясните создание и настройку маршрутов в Koa.js и Express.js.

3 Опишите параметры маршрутов и передачу данных через URL, формирование и отправку ответов на запросы, post-запросы и отправку форм.

4 Дайте определение понятию «переадресация».

5 Выскажите общее суждение о Middleware.

6 Объясните роль Middleware в обработке запросов, создание и применение Middleware в Koa.js и Express.js, аутентификацию и авторизацию в Koa.js и Express.js.

Тема 2.8 Понятие о CLI-приложениях

Общие сведения о CLI-приложениях: аргументы командной строки, вывод информации и интерактивное взаимодействие с пользователем.

Модули commander и inquirer. Обработка опций и аргументов. Создание пользовательских команд.

Упаковка и распространение CLI-приложений. Создание исполняемых файлов для установки CLI-приложений.

Литература: [3], с. 172-174

Вопросы для самоконтроля

1 Выскажите общее суждение о CLI-приложениях: аргументы командной строки, вывод информации и интерактивное взаимодействие с пользователем.

2 Опишите модули commander и inquirer, обработку опций и аргументов, создание пользовательских команд, упаковку и распространение CLI-приложений, создание исполняемых файлов для установки CLI-приложений.

Тема 2.9 Модули OS, Events, HTTP

Модуль OS. Основные методы модуля OS. Использование модуля OS для получения информации о системе.

Модуль Events. Основные методы модуля Events. Использование модуля Events для создания и обработки пользовательских событий.

Модуль HTTP. Свойства, методы и классы модуля HTTP. Методы для работы с HTTP-заголовками.

Литература: [5], с. 82-90

Вопросы для самоконтроля

1 Выскажите общее суждение о модуле OS.

2 Опишите методы модуля OS, использование модуля OS для получения информации о системе.

3 Выскажите общее суждение о модуле Events.

4 Опишите методы модуля Events, использование модуля Events для создания и обработки пользовательских событий.

5 Выскажите общее суждение о модуле HTTP.

6 Опишите свойства, методы и классы модуля HTTP, методы для работы с HTTP-заголовками.

Тема 2.10 Работа с потоками в Node.js

Потоки. Преимущества использования потоков. Типы потоков в Node.js: Readable, Writable, Duplex, Transform.

Организация работы с потоками в Node.js.

Особенности использования API Node.js для работы с потоками.

Литература: [3], с. 181-200

Вопросы для самоконтроля

1 Дайте определение понятию «поток».

2 Опишите преимущества использования потоков, типы потоков в Node.js: Readable, Writable, Duplex, Transform, организацию работы с потоками в Node.js.

3 Объясните особенности использования API Node.js для работы с потоками.

Тема 2.11 Основы работы с базами данных в Node.js

Системы управления базами данных (далее – СУБД).

Установка и настройка системы управления базами данных (СУБД): MySQL, MongoDB.

Методы для работы с базой данных MySQL и MongoDB в Node.js: установка соединения с сервером, выбор базы данных, обработка ошибок, выполнение запросов к серверу базы данных, обработка результатов запроса, получение информации о результатах SQL-запросов.

Использование фреймворка Koa.js для работы с базой данных.

Литература: [5], с. 150-181

Вопросы для самоконтроля

1 Дайте определение понятию «системы управления базами данных (СУБД)».

2 Опишите установку и настройку системы управления базами данных (СУБД) (MySQL, MongoDB), методы для работы с базой данных MySQL и MongoDB в Node.js: установка соединения с сервером, выбор базы данных, обработка ошибок, выполнение запросов к серверу базы данных, обработка результатов запроса, получение информации о результатах SQL-запросов, использование фреймворка Koa.js для работы с базой данных.

Список используемых источников

Основная литература

1 Брылёва, А.А. Программные средства создания интернет-приложений: учеб. пособие / А.А. Брылёва. – Минск: РИПО, 2022. – 483 с.

2 Никсон, Р. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5. 6-е изд. / Р. Никсон. – СПб.: Питер, 2023. – 832 с.

3 Пауэрс, Ш. Изучаем Node. Переходим на сторону сервера. 2-е изд., доп. и перераб. / Ш. Пауэрс. – СПб.: Питер, 2017. – 304 с.

4 Скляр, Д. Изучаем PHP 7. Руководство по созданию интерактивных веб-сайтов / Д. Скляр. – М. : Вильямс, 2017. - 464 с.

5 Node.js в действии. 2-е изд. / А. Янг [и др.]. – СПб.: Питер, 2018. – 432 с.

Дополнительная литература

1 Аквино, К. Front-end. Клиентская разработка для профессионалов. Node.js, ES6, REST / К. Аквино, Т. Ганди. – СПб.: Питер, 2017. – 512 с.

2 Дакетт, Джон. PHP и MYSQL. Серверная веб-разработка / Джон Дакетт ; [перевод с английского М. А. Райтмана]. – Москва: Эксмо, 2023. – 688 с.

3 Дронов, В. А. Laravel 9. Быстрая разработка веб-сайтов на PHP. / В. А. Дронов. – СПб.: БХВ-Петербург, 2023. – 768 с.

4 Зандстра, М. PHP 8: объекты, шаблоны и методики программирования, 6-е изд. / Мэтт Зандстра; пер. с англ. И.В. Красикова. – Киев: «Диалектика», 2021. – 866 с.

5 Котеров, Д. В. PHP 8 / Д. В. Котеров, И. В. Симдянов. – СПб.: БХВ-Петербург, 2023. – 992 с.

6 Прохоренок, Н.А. JavaScript и Node.js для веб-разработчиков / Н. А. Прохоренок, В.А. Дронов. – СПб.: БХВ-Петербург, 2022. – 768 с.

Задания на домашнюю контрольную работу №1 по учебному предмету «Веб-программирование на стороне сервера»

1-40 Теоретические вопросы

1 Перечислите и опишите способы внедрения PHP-сценария в веб-документ в языке программирования PHP.

2 Приведите синтаксис и опишите алгоритм работы оператора условия if…else в языке программирования PHP.

3 Охарактеризуйте протокол HTTP, его назначение и методы в Node.js.

4 Перечислите и опишите основные команды менеджера пакетов NPM для Node.js.

5 Приведите синтаксис и опишите алгоритм работы оператора цикла while в языке программирования PHP.

6 Охарактеризуйте назначение и принципы использования трейтов в языке программирования PHP.

7 Перечислите и опишите основные методы для работы с таймерами в Node.js.

8 Приведите синтаксис и опишите алгоритм работы оператора цикла do…while в языке программирования PHP.

9 Охарактеризуйте технологию cookie (создание, чтение и удаление cookie) в языке программирования PHP.

10 Перечислите и опишите основные функции модуля fs для работы с файловой системой в Node.js.

11 Приведите синтаксис и опишите способы создания строк в языке программирования PHP.

12 Охарактеризуйте сессии (инициализация сессии, чтение из сессии, удаление переменных из сессии, использование обработчиков сессии, завершение сессии) в языке программирования PHP.

13 Перечислите и опишите методы создания, эмитирования и обработки событий с использованием класса EventEmitter в Node.js.

14 Приведите синтаксис и опишите способы создания одномерных и многомерных массивов в языке программирования PHP.

15 Охарактеризуйте синтаксис объявления класса, процесс создания и использования объектов класса, приведите примеры в языке программирования PHP.

16 Перечислите и опишите методы для работы с файлами и каталогами в языке программирования PHP.

17 Приведите синтаксис и опишите алгоритм работы оператора цикла for в языке программирования PHP.

18 Охарактеризуйте коллбэки и промисы в асинхронном программировании в Node.js.

19 Перечислите и опишите методы работы с массивами в языке программирования PHP.

20 Приведите синтаксис создания и настройки маршрутов в Koa.js и Express.js и опишите принцип их работы.

21 Охарактеризуйте назначение и принципы использования интерфейсов в языке программирования PHP.

22 Перечислите и опишите основные функции для работы с базой данных MySQL в языке программирования PHP.

23 Приведите синтаксис объявления пользовательских функций и опишите принцип их работы в языке программирования PHP.

24 Охарактеризуйте асинхронные функции async/await в асинхронном программировании в Node.js.

25 Опишите типы данных языка программирования PHP. Приведите примеры их использования.

26 Опишите операторы в языке программирования PHP (оператор присваивания, операторы сравнения и логические операторы, инкремент и декремент).

27 Опишите синтаксис и алгоритм работы оператора выбора switch в языке программирования PHP. Приведите примеры использования.

28 Объясните назначение и методы протокола HTTP (get, post, put, delete) в языке программирования PHP.

29 Дайте определение понятиям «Массив», «Ассоциативный массив». Опишите основные операции работы с массивами (создание, доступ к элементам изменение, удаление элементов, поиск элементов в массиве), базовые инструкции работы с массивами: list(), array(), основные операции над массивами (удаление, слияние, сортировка) в языке программирования PHP.

30 Дайте определение понятию «Строка». Описывает основные операции работы со строками (создание, доступ к элементам, изменение, удаление элементов), базовые функции работы со строками: strlen(), strpos(), substr(), strcmp(), strcasecmp(), основные операции над строками (конкатенация, сравнение, замена) в языке программирования PHP.

31 Охарактеризуйте стандартные функции PHP для работы с массивами и строками, файлами и каталогами, датой и временем в языке программирования PHP.

32 Объясните основные принципы управления сессиями в языке программирования PHP. Охарактеризуйте сессии в языке программирования PHP (инициализация сессии, чтение из сессии, удаление переменных из сессии, использование обработчиков сессии, завершение сессии).

33 Опишите структуру запроса, методы передачи данных форм в языке программирования PHP (get и post), способы обработки данных форм средствами PHP.

34 Дайте определение понятию «Объектно-ориентированное программирование» в языке программирования PHP. Объясните основные концепции объектно-ориентированного программирования: инкапсуляция, наследование, полиморфизм в языке программирования PHP.

35 Опишите организацию кода с помощью пространства имен, импорты в пространстве имен, глобальные пространства имен и их использование, разрешение конфликтов имен в языке программирования PHP.

36 Опишите управление зависимостями в Node.js.

37 Опишите основные принципы работы цикла событий, генерацию события и тип EventEmitter, создание и эмитирование событий, привязку обработчиков событий, передачу параметров событию, работу с событиями ввода-вывода, создание и использование собственных событий в Node.js.

38 Выскажите общее суждение о модуле OS. Опишите методы модуля OS, использование модуля OS для получения информации о системе в Node.js.

39 Дайте определение понятию «Поток» в Node.js. Опишите преимущества использования потоков, типы потоков в Node.js: Readable, Writable, Duplex, Transform, организацию работы с потоками в Node.js.

40 Опишите установку и настройку системы управления базами данных (СУБД) (MySQL, MongoDB), методы для работы с базой данных MySQL и MongoDB в Node.js: установка соединения с сервером, выбор базы данных, обработка ошибок, выполнение запросов к серверу базы данных, обработка результатов запроса, получение информации о результатах SQL-запросов.

41-60 Практическое задание 1

Решение задач на языке программирования PHP при помощи циклов

В отчете необходимо представить текст программы с комментариями и скриншотами результата выполнения программы.

41 Напишите программу, которая находит среднее арифметическое всех чисел от 1 до 50 и выводит результат на экран.

42 Напишите программу, которая вычисляет факториал заданного числа и выводит результат на экран.

43 Напишите программу, которая проверяет, является ли заданное число простым, и выводит результат на экран.

44 Напишите программу, которая находит сумму всех чисел, кратных 3 или 5, в заданном диапазоне, и выводит результат на экран.

45 Напишите программу, которая выводит на экран таблицу умножения от 1 до 10.

46 Напишите программу, которая находит наибольшую цифру в заданном числе и выводит результат на экран.

47 Напишите программу, которая проверяет, является ли заданная строка палиндромом (читается одинаково с начала и с конца), и выводит результат на экран.

48 Напишите программу, которая запрашивает у пользователя целое число и определяет, является ли оно числом Армстронга (числом, равным сумме его цифр, возведённых в степень, равную количеству цифр), и выводит результат на экран.

49 Напишите программу, которая проверяет, является ли заданное число совершенным (числом, равным сумме своих делителей, кроме самого числа), и выводит результат на экран.

50 Напишите программу, которая вычисляет сумму всех целых чисел от 1 до заданного пользователем числа n и выводит результат на экран.

51 Напишите программу, которая выводит квадраты первых n натуральных чисел.

52 Напишите программу, которая находит сумму всех простых чисел в заданном диапазоне и выводит результат на экран.

53 Напишите программу, которая определяет количество делителей заданного числа и выводит результат на экран.

54 Напишите программу, которая вычисляет сумму всех чисел Фибоначчи до заданного числа и выводит результат на экран.

55 Напишите программу, которая находит сумму всех простых чисел в указанном пользователем диапазоне и выводит результат на экран.

56 Напишите программу, которая определяет, является ли заданное число простым, и выводит результат на экран.

57 Напишите программу, которая находит наименьший делитель заданного пользователем целого числа n (кроме 1) и выводит результат на экран.

58 Напишите программу, которая выводит цифры заданного положительного числа в обратном порядке.

59 Напишите программу, которая генерирует случайное число от 1 до 100 и предлагает пользователю его угадать, отображая подсказки «больше» или «меньше» до тех пор, пока число не будет угадано, и выводит результат на экран.

60 Напишите программу, которая выводит последовательность из n первых квадратов натуральных чисел и отображает их сумму.

61- 80 Практическое задание 2

Решение задач обработки массивов на языке программирования PHP

В отчете необходимо представить текст программы с комментариями и скриншотами результата выполнения программы.

61 Дана матрица A(3,3). Найдите сумму элементов на главной и побочной диагоналях матрицы. Выведите исходную матрицу и результаты вычислений.

62 Дана матрица A(3,3). Найдите сумму элементов главной диагонали и эту сумму поставьте на место последнего элемента. Выведите исходную матрицу и результаты вычислений.

63 Дана матрица A(3,3). Напишите программу для транспонирования этой матрицы (поменять строки и столбцы местами). Выведите исходную матрицу и результаты вычислений.

64 Дана матрица A(3,3). Вычислите сумму второй строки и произведение первого столбца. Выведите исходную матрицу и результаты вычислений.

65 Дана матрица A(3,3). Найдите сумму элементов, находящихся выше главной диагонали. Выведите исходную матрицу и результаты вычислений.

66 Дана матрица A(3,3). Найдите произведение элементов побочной диагонали, а также среднее арифметическое элементов главной диагонали матрицы. Выведите исходную матрицу и результаты вычислений.

67 Дана матрица A(3,3). Замените все отрицательные элементы на нули, а положительные на единицы (бинаризация матрицы). Выведите исходную матрицу и результаты вычислений.

68 Дана матрица A(3,3). Найдите наименьший элемент в каждой строке матрицы. Выведите исходную матрицу и результаты вычислений.

69 Дана матрица A(3,3). Найдите произведение всех элементов, расположенных ниже главной диагонали. Выведите исходную матрицу и результаты вычислений.

70 Дана матрица A(3,3). Определите количество положительных, отрицательных и нулевых элементов матрицы. Выведите исходную матрицу и результаты вычислений.

71 Дана матрица A(3,3). Поменяйте местами первую и последнюю строки матрицы. Выведите исходную матрицу и результаты вычислений.

72 Дана матрица A(3,3). Поменяйте местами первый и последний столбцы матрицы. Выведите исходную матрицу и результаты вычислений.

73 Дана матрица A(3,3). Найдите сумму всех четных элементов матрицы. Выведите исходную матрицу и результаты вычислений.

74 Дана матрица A(3,3). Замените элементы главной диагонали числом 0. Выведите исходную матрицу и результаты вычислений.

75 Дана матрица A(3,3). Найдите максимальный элемент матрицы и определите его позицию (строку и столбец). Выведите исходную матрицу и результаты вычислений.

76 Дана матрица A(3,3). Найдите сумму всех элементов матрицы, расположенных на границе (периметре) матрицы. Выведите исходную матрицу и результаты вычислений.

77 Дана матрица A(3,3). Умножьте все элементы второй строки на первый элемент третьей строки. Выведите исходную матрицу и результаты вычислений.

78 Дана матрица A(3,3). Найдите сумму элементов каждого столбца и выведите получившийся одномерный массив. Выведите исходную матрицу и результаты вычислений.

79 Дана матрица A(3,3). Замените элементы побочной диагонали числом 1. Выведите исходную матрицу и результаты вычислений.

80 Дана матрица A(3,3). Определите, является ли матрица симметричной относительно главной диагонали. Выведите исходную матрицу и результаты вычислений.

81- 100 Практическое задание 3

Решение задач на использование Node.js для взаимодействия с базой данных MySQL

Разработайте программу на Node.js для доступа к базе данных MySQL, содержащей информацию о книгах (автор, название, изображение обложки, издательство, год выпуска, цена, количество продаж за последний год).

81 В базе данных содержится информация о книгах: автор, название, изображение обложки, издательство, год выпуска, цена, количество продаж за последний год.

Запросы:

а) вывести информацию о книгах, цена которых лежит в заданном диапазоне;

б) вывести информацию о книгах заданного автора.

82 В базе данных содержится информация об автомобилях: модель, изображение автомобиля, год выпуска, тип кузова, мощность двигателя, цвет, цена.

Запросы:

а) вывести информацию об автомобилях, выпущенных в указанном году;

б) вывести информацию об автомобилях, цена которых лежит в заданном диапазоне.

83 В базе данных содержится информация о туристических поездках: страна, город, изображение городской достопримечательности, количество дней, дата поездки, класс отеля, цена.

Запросы:

а) вывести информацию о турах в заданную страну;

б) вывести информацию о турах из заданного диапазона дат.

84 В базе данных содержится информация о журналах: название, изображение обложки, год выпуска, номер, издательство, число страниц, цена.

Запросы:

а) вывести информацию о журналах заданного издательства;

б) вывести информацию о журналах, выпущенные за заданный период времени.

85 В базе данных содержится информация о местах в отеле: название отеля, класс номера, изображение номера, количество мест в номере, цена.

Запросы:

а) вывести информацию об отелях заданного класса;

б) вывести информацию об отелях, стоимость которых не превышает заданную цену.

86 Спроектировать структуру базы данных о студентах, для их распределения по местам практики: фамилия, год рождения, пол, группа, факультет, средний балл, место работы, город.

Запросы:

а) вывести информацию о студентах, распределенных на практику в заданный город;

б) вывести информацию о студентах, средний балл которых попадает в заданный интервал.

87 Спроектировать структуру базы данных об автомобилях: номер, год выпуска, марка, цвет, состояние, фамилия владельца, адрес.

Запросы:

а) вывести информацию об автомобилях заданной марки;

б) вывести информацию об автомобилях, год выпуска которых попадает в заданный интервал.

88 Спроектировать структуру базы данных о квартирах, предназначенных для продажи: район, этаж, площадь, количество комнат, сведения о владельце, цена.

Запросы:

а) вывести информацию о квартирах, расположенные в заданном районе;

б) вывести информацию о квартирах, цена которых не превышает указанную.

89 Спроектировать структуру базы данных о книгах, купленных библиотекой: название, автор, год издания, адрес автора, адрес издательства, цена, книготорговая фирма.

Запросы:

а) вывести информацию о книгах указанной книготорговой фирмы;

б) вывести информацию о книгах, год издания которых попадает в заданный интервал.

90 Спроектировать структуру базы данных о сотрудниках, имеющих компьютер: фамилия, номер комнаты, название отдела, данные о компьютерах, дата приобретения ПК.

Запросы:

а) вывести информацию о сотрудниках из указанного отдела;

б) вывести информацию о сотрудниках, дата приобретения ПК которых попадает в заданный интервал.

91 Спроектировать структуру базы данных о заказах, полученных сотрудниками фирмы: фамилия, сумма заказа, наименование товара, название фирмы - клиента, фамилия заказчика.

Запросы:

а) вывести сведения о заказах, полученных указанным сотрудником фирмы;

б) вывести сведения о заказах, сумма которых находится в заданном интервале.

92 Спроектировать структуру базы данных об оценках, полученных студентами на экзаменах: фамилия, группа, предмет, номер билета, оценка, преподаватель.

Запросы:

а) вывести сведения об экзаменах, которые принимал указанный преподаватель;

б) вывести сведения о студентах, получивших оценки из указанного диапазона.

93 Спроектировать структуру базы данных об авторах web-сайта и их статьях: имя, адрес, учетная запись, пароль, тема, заголовок, дата публикации.

Запросы:

а) вывести информацию о статьях, относящихся к указанной теме;

б) вывести информацию о статьях, опубликованных за определенный период.

94 Спроектировать структуру базы данных о списке рассылки и подписчиках: тема и содержание письма, дата отправки, имена и адреса подписчиков, их учетные записи и пароли.

Запросы:

а) вывести информацию о рассылках, принадлежащих определенной теме;

б) вывести информацию о рассылках, отправленных в определенный период времени.

95 В базе данных содержится информация о туристических поездках: страна, город, изображение городской достопримечательности, количество дней, дата поездки, класс отеля, цена.

Запросы:

а) вывести информацию о турах на заданную дату;

б) вывести информацию о турах, продолжительность которых находится в заданном диапазоне.

96 В базе данных содержится информация о журналах: название, изображение обложки, год выпуска, номер, издательство, число страниц, цена.

Запросы:

а) вывести информацию о журналах с заданным названием;

б) вывести информацию о журналах, число страниц в которых лежит в заданном диапазоне.

97 Спроектировать структуру базы данных о квартирах, предназначенных для продажи: район, этаж, площадь, количество комнат, сведения о владельце, цена.

Запросы:

а) вывести информацию о квартирах, расположенных на заданном этаже;

б) вывести информацию о квартирах, имеющих площадь из указанного интервала.

98 Спроектировать структуру базы данных об автомобилях: номер, год выпуска, марка, цвет, состояние, цена.

Запросы:

а) вывести информацию об автомобилях указанного года выпуска;

б) вывести информацию об автомобилях, цена которых попадает в заданный интервал.

99 Спроектировать структуру базы данных о списке рассылки и подписчиках: тема и содержание письма, дата отправки, имена и адреса подписчиков, их учетные записи и пароли.

Запросы:

а) вывести информацию о рассылках, принадлежащих указанному подписчику;

б) вывести информацию о рассылках, отправленных в определенный период времени.

100 Спроектировать структуру базы данных о студентах, для их распределения по местам практики: фамилия, год рождения, пол, группа, факультет, средний балл, место работы, город.

Запросы:

а) вывести информацию о студентах указанного факультета;

б) вывести информацию о студентах, год рождения которых лежит в заданном диапазоне.

Методические рекомендации по выполнению домашней контрольной работы №1

Методические рекомендации по выполнению заданий 41-60

Циклы используются для того, чтобы некоторый участок кода выполнился несколько раз подряд. Зачем это нужно - представьте, что нужно возвести в квадрат `100` элементов массива. Если обращаться к каждому элементу отдельно по его ключу — это займет `100` строчек кода, и для того, чтобы написать этого код, нужно будет потратить довольно много времени.

Но это не нужно - есть возможность сделать так, чтобы PHP выполнил некоторую операцию нужное количество раз. Например, возвел все элементы массива в квадрат. Это и делается с помощью циклов.

Цикл `while` будет выполняться до тех пор, пока верно (истинно) выражение, переданное ему параметром. Смотрите синтаксис:

```
<?php
    while ( пока выражение истинно ) {
        выполняем этот код циклически
        в начале каждого цикла проверяем выражение в круглых скобках
    }
?>
```

Цикл закончится, когда выражение перестанет быть истинным. Если оно было ложным изначально - то он не выполнится ни разу.

Для примера последовательно выведем с помощью цикла `while` числа от одного до пяти:

```
<?php
    $i = 1; // задаем какую-нибудь переменную
    while ($i <= 5) {
        echo $i; // выводим содержимое $i в консоль
        $i++;       // увеличиваем $i на единицу при каждом проходе цикла
    }
?>
```

Каждый проход цикла по-научному называется *итерацией* цикла. Можно сказать, что увеличиваем переменную `$i` на единицу в каждой итерации цикла.

Сама переменная `$i` называется *счетчиком цикла*. Счетчики используются для того, чтобы подсчитывать, сколько раз выполнился цикл. Кроме того, они выполняют вспомогательную роль - в задаче использовали счетчик, чтобы вывести цифры от `1` до `5`. Для счетчиков принято использовать буквы `i`, `j` и `k`.

К счетчику не обязательно прибавлять единицу. Для примера выведем столбец четных чисел от `2` до `10`. Для этого начальное значение переменной `$i` зададим как `2` и будем прибавлять двойку:

```
<?php
	$i = 2;
	while ($i <= 10) {
		echo $i;
		$i += 2; // увеличиваем $i на 2 при каждом проходе цикла
	}
?>
```

Счетчик не обязательно должен увеличиваться в цикле. Бывают и обратные ситуации, когда счетчик, наоборот, уменьшается. Для примера выведем столбец чисел от `10` до `1`:

```
<?php
	$i = 10; // начальное значение 10
	while ($i >= 1) { // пока $i больше 1
		echo $i;
		$i--; // уменьшаем $i на единицу
	}
?>
```

Пусть есть вот такой цикл, выводящий числа от `1` до `10`:

```
<?php
	$i = 1;
	while ($i <= 10) {
		echo $i;
		$i++;
	}
?>
```

Представим теперь, что программист забыл сделать увеличение счетчика в цикле:

```
<?php
	$i = 1;
	while ($i <= 10) {
```

```
            echo $i;
        }
    ?>
```

В этом случае цикл будет выполнятся бесконечно, так как условие, при котором цикл закончится, никогда не будет достигнуто.

Цикл do..while похож на цикл while, только теперь выполняется блок цикла, и только потом выполняется проверка условия. То есть даже если условие ложно, то блок цикла выполнится как минимум один раз:

```
<?php
$counter = 1;
do
{
   echo $counter * $counter . "<br />";
   $counter++;
}
while($counter<10)
?>
```

Цикл `for` является альтернативой `while`. Он более сложен для понимания, но чаще всего его любят больше за то, что он занимает меньше строчек.

Вот его синтаксис:

```
<?php
    for ( начальные команды; условие окончания; команды после прохода ) {
            тело цикла
    }
?>
```

Начальные команды — это то, что выполнится перед стартом цикла. Они выполнятся только один раз. Обычно там размещают начальные значения счетчиков. *Условие окончания* — это условие, при котором цикл будет крутится, пока оно истинное. *Команды после прохода* — это команды, которые будут выполнятся каждый раз при окончании прохода цикла. Обычно там увеличивают счетчики.

С помощью цикла `for` выведем последовательно числа от `1` до `9`:

```
<?php
    /*
        В начале цикла $i будет равно нулю,
        цикл будет выполнятся пока $i <= 9,
        после каждого прохода к $i прибавляется единица:
```

```
        */
        for ($i = 0; $i <= 9; $i++) {
            echo $i; // выведет 1, 2... 9
        }
?>
```

С помощью цикла найдем сумму целых чисел от `1` до `100`. Это решение заключается в том, что циклом перебираются числа и их сумма последовательно записывается в какую-то переменную:

```
<?php
    $result = 0;
    for ($i = 1; $i <= 100; $i++) {
        $result = $result + $i;
    }
    echo $result; // искомая сумма
?>
```

Как это работает: изначально переменная `$result` имеет значение `0`, затем при первом проходе цикла в нее записывается ее текущее содержимое плюс значение счетчика. Получится, что каждую итерацию переменная будет расти, постепенно накапливая в себе результат.

Рассмотрим сокращенный синтаксис циклов. Речь идет о том, что в циклах, подобно конструкциям `if`, фигурные скобки не обязательны. Если их опустить, то цикл выполнит только одну строку под ним.

Рассмотрим пример. Пусть есть некоторый цикл с фигурными скобками:

```
<?php
    for ($i = 0; $i <= 9; $i++) {
        echo $i; // выведет числа от 0 до 9
    }
?>
```

Опустим фигурные скобки - и результат от этого не изменится:

```
<?php
    for ($i = 0; $i <= 9; $i++)
        echo $i; // выведет числа от 0 до 9
?>
```

Внесу маленькое исправление в приведенный выше код (найдите какое) - и он перестанет работать:

```
<?php
    for ($i = 0; $i <= 9; $i++);
```

```
        echo $i; // выдаст 10
?>
```

Проблема возникла из-за того, что я поставил точку с запятой после скобки) от цикла. В таком случае получится так называемый цикл без тела: он просто прокрутится внутри, а следующая строчка уже не будет к нему относится. Поэтому, во избежание проблем, я всегда рекомендую ставить фигурные скобки в циклах.

Синтаксис цикла `for`:

```
<?php
    for (начальные команды; условие окончания; команды после прохода) {
        тело цикла
    }
?>
```

На самом деле начальные команды и команды после прохода цикла могут состоять не из одной, а из нескольких команд, разделяемых запятыми.

Для примера сделаем два счетчика: первый пусть каждую итерацию цикла увеличивается на единицу, а второй - на двойку:

```
<?php
    for ($i = 0, $j = 0; $i <= 9; $i++, $j += 2) {
        echo $i . ' ' . $j . '<br>';
    }
?>
```

Пусть есть вот такой цикл, выводящий на экран элементы массива:

```
<?php
    $arr = [1, 2, 3, 4, 5];
    foreach ($arr as $elem) {
        echo $elem;
    }
?>
```

Пусть стоит задача определить, есть ли в массиве число `3`. Если есть - выведем на экран слово `'есть'` (а если нет - ничего не будем делать).

Решим задачу:

```
<?php
    $arr = [1, 2, 3, 4, 5];
    foreach ($arr as $elem) {
```

```
        if ($elem == 3) {
            echo 'есть';
        }
    }
?>
```

Задача решена, однако, есть проблема: после того, как число `3` уже найдено, массив все равно продолжает бессмысленно перебираться дальше, тратя ценные ресурсы процессора и замедляя работу скрипта.

Оптимальнее было бы сразу после нахождения числа завершить работу цикла. Это можно сделать с помощью специально инструкции `break`, позволяющей досрочно завершить работу цикла.

Итак, завершим цикл, как только встретится число `3`:

```
<?php
    $arr = [1, 2, 3, 4, 5];
    foreach ($arr as $elem) {
        if ($elem == 3) {
            echo 'есть';
            break; // выйдем из цикла
        }
    }
?>
```

Инструкция break может завершать любые циклы: foreach, for, while.

Помимо инструкции `break`, завершающей работу цикла, существует также инструкция `continue`, запускающая новую итерацию цикла. Данная инструкция иногда может быть полезна для упрощения кода, хотя практически всегда задачу можно решить и без нее. Рассмотрим на практическом примере.

Пусть дан массив с числами. Переберем его циклом и числа, которые делятся на `2`, возведем в квадрат и выведем на экран, а числа, которые делятся на `3`, возведем в куб и выведем на экран.

Вот решение описанной задачи:

```
<?php
    $arr = [1, 2, 3, 4, 5, 6, 7, 8, 9];
    foreach ($arr as $elem) {
        if ($elem % 2 === 0) {
            $res = $elem * $elem;
            echo $res;
```

```
        } elseif ($elem % 3 === 0) {
            $res = $elem * $elem * $elem;
            echo $res;
        }
    }
?>
```

Как видно, строчка `echo $res` повторяется два раза. Вынесем ее за `if`, вот так:

```
<?php
    $arr = [1, 2, 3, 4, 5, 6, 7, 8, 9];
    foreach ($arr as $elem) {
        if ($elem % 2 === 0) {
            $res = $elem * $elem;
        } elseif ($elem % 3 === 0) {
            $res = $elem * $elem * $elem;
        }
        echo $res; // вынесли вывод за условие
    }
?>
```

Скрипт, однако, работает немного не так: получится, что и для обычных элементов, не обработанных через `if`, будет выполнятся вывод переменной `$res` на экран, что по условию задачи не нужно.

Поправим проблему, добавив к `if` еще условие `else`, которое будет срабатывать для элементов, не делящихся на `2` и `3`, и вызовем там инструкцию `continue`, которая сразу же будет перебрасывать на новую итерацию цикла:

```
<?php
    $arr = [1, 2, 3, 4, 5, 6, 7, 8, 9];
    foreach ($arr as $elem) {
        if ($elem % 2 === 0) {
            $res = $elem * $elem;
        } elseif ($elem % 3 === 0) {
            $res = $elem * $elem * $elem;
        } else {
            continue; // перейдем на новую итерацию цикла
        }
        echo $res; // выполнится, если делится на 2 или 3
    }
?>
```

Пример:

Условие задачи. Напишите программу, которая выводит все числа Армстронга в указанном пользователем диапазоне.

Решение:

Число Армстронга — это число, равное сумме своих цифр, возведённых в степень количества цифр (например, $153 = 1^3 + 5^3 + 3^3$).

Используйте циклы для перебора диапазона.

Код программы:

```
<?php

$start = intval(readline("Введите начало диапазона: "));
$end = intval(readline("Введите конец диапазона: "));

echo "Числа Армстронга в диапазоне от $start до $end:\n";

for ($num = $start; $num <= $end; $num++) {
        // Преобразуем число в строку, чтобы удобно работать с цифрами
    $strNum = strval($num);
    $length = strlen($strNum); // количество цифр

    $sum = 0;

    // Перебираем каждую цифру числа
    for ($i = 0; $i < $length; $i++) {
       $digit = intval($strNum[$i]);
          $sum += pow($digit, $length); // возводим цифру в степень количества цифр
    }

    // Проверяем, является ли число числом Армстронга
    if ($sum == $num) {
       echo $num . "\n";
    }
}

?>
```

Результат выполнения для диапазона от 1 до 500:

```
1
2
3
4
5
6
7
8
9
153
370
371
407
```

Методические рекомендации по выполнению заданий 61-80

Массив представляет собой переменную, в которой в упорядоченном виде можно хранить целый набор каких-то значений.

Для создания массива используются квадратные скобки:

```
<?php
    $arr = []; // создаем массив $arr
?>
```

Пока созданный массив не содержит никаких данных. Заполним его названиями дней недели:

```
<?php
    $arr = ['пн', 'вт', 'ср', 'чт', 'пт', 'сб', 'вс'];
?>
```

Каждое значение списка, который записали в массив, каждый день недели, называется *элементом* массива. Элементы разделяются между собой запятой. После этой запятой можно ставить пробелы, а можно и не ставить (более принято ставить, ставьте).

Обратите внимание на то, что названия дней недели представляют собой строки и поэтому взяты в кавычки. Кроме строк в массиве можно хранить числа, и их в кавычки не берем:

```
<?php
    $arr = [1, 2, 3];
?>
```

Кроме строк и чисел в массиве можно хранить все допустимые типы данных PHP, а также смешивать их между собой в одном массиве, пример:

```
<?php
    $arr = [1, 2, 'a', 'b', null, true, false];
?>
```

Для того, чтобы PHP вывел все элементы массива, нужно воспользоваться функцией `var_dump`:

```
$a = [1, 2, 3];
var_dump($a);
```

Обращение к элементам массива осуществляется подобно обращениям к символам строки: первый элемент имеет номер `0`, второй - номер `1` и так далее. Эти номера называются *ключами* элементов массива. Рассмотрим пример:

```
<?php
    $arr = ['пн', 'вт', 'ср', 'чт', 'пт', 'сб', 'вс'];
    echo $arr[0]; // выведет 'пн'
    echo $arr[1]; // выведет 'вт'
    echo $arr[2]; // выведет 'ср'
?>
```

Рассмотрим следующий массив:

```
<?php
    $arr = ['пн', 'вт', 'ср', 'чт', 'пт', 'сб', 'вс'];
?>
```

Чтобы обратиться к нужному элементу этого массива, нужно написать в квадратных скобках ключ этого элемента. В массивах PHP сам определяет ключи для элементов — это их порядковые номера. Но иногда это может оказаться неудобным: например, нужно вывести на экран название первого дня недели, а должны писать в квадратных скобках цифру `0`.

Логичнее и удобнее было бы все-таки для первого дня недели писать ключ `1`, как привыкли в жизни. Для этого используются *ассоциативные* массивы. Они имеют следующий синтаксис: имя ключа, затем идет стрелка =>, а потом значение. Укажем явные ключи для массива дней:

```
<?php
    $arr = [1 => 'пн', 2 => 'вт', 3 => 'ср', 4 => 'чт', 5 => 'пт', 6 => 'сб',
7 => 'вс'];
?>
```

После добавления ключей обратиться к понедельнику можно уже по ключу `1`, а не `0`. Сделаем это:

```
<?php
    echo $arr[1]; // выведет 'пн'
?>
```

Не очень удобно расставлять ключи всем элементам для того, чтобы нумерация началась не с нуля, а с единицы. К счастью, на самом деле достаточно первому элементу поставить ключ `1` и дальше PHP сам автоматически расставит ключи по порядку.

Попробуем:

```
<?php
    $arr = [1 => 'пн', 'вт', 'ср', 'чт', 'пт', 'сб', 'вс'];
    echo $arr[1]; // выведет 'пн'
    echo $arr[2]; // выведет 'вт'
    echo $arr[3]; // выведет 'ср'
?>
```

Длина массива находится с помощью функции `count`:

```
<?php
    $arr = [1, 2, 3];
    echo count($arr); // выведет 3
?>
```

Элементы в массив не обязательно добавлять сразу в момент объявления этого массива. Можно вначале объявить этот массив пустым, а затем добавить в него необходимые элементы, вот так:

```
<?php
    $arr = [];     // создаем пустой массив
    $arr[] = 'a';  // элемент добавится в ключ 0
    $arr[] = 'b';  // элемент добавится в ключ 1
    $arr[] = 'c';  // элемент добавится в ключ 2
    var_dump($arr); // выведет ['a', 'b', 'c']
?>
```

Массив не обязательно должен быть изначально пустым - там уже что-то может быть, но все равно можно добавлять новые элементы:

```
<?php
    $arr = ['a', 'b', 'c']; // объявляем массив с элементами
    $arr[] = 'd'; // элемент добавится в ключ 3
    $arr[] = 'e'; // элемент добавится в ключ 4
    var_dump($arr); // выведет ['a', 'b', 'c', 'd', 'e']
?>
```

Элементы массива могут быть не только строками и числами, но и массивами. В этом случае получится массив массивов или *многомерный массив*. В следующем примере массив `$arr` состоит из трех элементов, в свою очередь являющихся массивами:

```
<?php
    $arr = [['a', 'b', 'c'], ['d', 'e', 'f'], ['g', 'h', 'i']];
?>
```

Перепишем в более понятном виде:

```
<?php
    $arr = [
        ['a', 'b', 'c'],
        ['d', 'e', 'f'],
        ['g', 'h', 'i'],
    ];
?>
```

В зависимости от уровня вложенности массивы могут быть двухмерными - массив массивов, трехмерными - массив массивов (ну и так далее - четырехмерными, пятимерными и тп).

Приведенный выше массив является двухмерным, так как внутри одного массива расположены другие подмассивы и уже в этих подмассивах нет других массивов. Чтобы вывести какой-либо элемент из двухмерного массива следует писать уже не одну пару квадратных скобок, а две:

```
<?php
    $arr = [
        ['a', 'b', 'c'],
        ['d', 'e', 'f'],
        ['g', 'h', 'i'],
    ];
    echo $arr[0][1]; // выведет 'b'
    echo $arr[1][2]; // выведет 'f'
?>
```

Многомерные массивы также могут быть ассоциативными, например, вот так:

```
<?php
    $arr = [
        'user1' => [
            'name' => 'name1',
            'age'  => 31,
```

```
        ],
        'user2' => [
            'name' => 'name2',
            'age'  => 32,
        ],
    ];
?>
```

С помощью этого массива выведем на экран, к примеру, имя второго юзера:

```
<?php
    echo $arr['user2']['name']; // выведет 'name2'
?>
```

Пусть дан следующий массив:

```
<?php
    $arr = [
        ['a', 'b', 'c'],
        ['d', 'e', 'f'],
        ['g', 'h', 'i'],
    ];
?>
```

Этот массив двухмерный, а это значит, что для его перебора нужно два вложенных цикла `foreach`:

```
<?php
    foreach ($arr as $sub) {
        foreach ($sub as $elem) {
            echo $elem;
        }
    }
?>
```

Пусть теперь нужно в цикле создать какой-нибудь многомерный массив с числами.

Например, вот такой двухмерный массив:

```
<?php
    [[1, 2, 3], [1, 2, 3], [1, 2, 3]]
?>
```

Решим поставленную задачу, применив два вложенных цикла. Внешний цикл будет создавать подмассивы, а внутренний - заполнять эти подмассивы числами:

```php
<?php
    $arr = [];
    for ($i = 0; $i < 3; $i++) {
        for ($j = 0; $j < 3; $j++) {
            $arr[$i][$j] = $j + 1; // заполняем подмассив числами
        }
    }
    var_dump($arr);
?>
```

Функция in_array проверяет наличие заданного элемента в массиве. Первым параметром она принимает что искать, а вторым - в каком массиве.

Синтаксис

in_array(что искать, в каком массиве);

Пример

Проверим, есть ли в массиве **$arr** элемент со значением 3:

```php
<?php
    $arr = ['a', 'b', 'c', 'd', 'e'];
    $result = in_array('c', $arr);
    var_dump($result);
?>
```

Результат выполнения кода:

true

Функция array_sum вычисляет сумму элементов массива.

Синтаксис

array_sum(массив);

Пример

Найдем сумму элементов массива:

```php
<?php
    $arr = [1, 2, 3, 4, 5];
    echo array_sum($arr);
?>
```

Результат выполнения кода:

15

Функция array_product вычисляет произведение (умножение) элементов массива.

Синтаксис

array_product(массив);

Пример

Найдем произведение элементов массива:

```
<?php
    $arr = [1, 2, 3, 4, 5];
    echo array_product($arr);
?>
```

Результат выполнения кода:

120

Функция range создает массив с диапазоном элементов. К примеру, можно создать массив, заполненный числами от 1 до 100 или буквами от 'a' до 'z'. Диапазон, который сгенерирует функция, задается параметрами: первый параметр откуда генерировать, а второй - докуда.

Третий необязательный параметр функции задает шаг. К примеру, можно сделать ряд 1, 3, 5, 7, если задать шаг 2, или ряд 1, 4, 7, 10 если задать шаг 3.

Синтаксис

```
range(откуда, докуда, [шаг]);
```

Пример

Создадим массив, заполненный числами от 1 до 5:

```
<?php
    var_dump(range(1, 5));
?>
```

Результат выполнения кода:

[1, 2, 3, 4, 5]

Функция array_merge сливает два и более массивов вместе. Если в сливаемых массивах встречаются одинаковые ключи - останется только один из таких элементов. Если нужно, чтобы остались все элементы с одинаковыми ключами - используйте функцию array_merge_recursive.

Синтаксис

```
array_merge(первый массив, второй массив...);
```

Пример

Сольем два массива вместе:

```
<?php
    $arr1 = ['a', 'b', 'c', 'd', 'e'];
    $arr2 = [1, 2, 3, 4, 5];
    $result = array_merge($arr1, $arr2);
    var_dump($result);
?>
```

Результат выполнения кода:

['a', 'b', 'c', 'd', 'e', 1, 2, 3, 4, 5]

Функция array_slice отрезает и возвращает часть массива. При этом сам массив не меняется. Первым параметром указывается массив для разрезания. Вторым параметром указывается, с какого элемента начинать отрезание, а третьим - сколько элементов отрезать. Второй параметр может быть отрицательным - в этом случае отсчет начнется с конца (-1 - последний элемент, -2 - предпоследний и так далее). Третий параметр можно вообще не указывать - в этом случае массив отрежется до самого конца.

Последний необязательный параметр регулирует сохранять ли ключи при отрезании, true - сохранять, false (по умолчанию) - не сохранять. Строковые ключи сохраняются, независимо от значения этого параметра.

Синтаксис

array_slice(массив, откуда отрезать, [сколько], [сохранять ключи = true]);

Пример

Вырежем элементы с первого (имеет номер 0), 3 штуки:

```
<?php
    $arr = ['a', 'b', 'c', 'd', 'e'];
    $result = array_slice($arr, 0, 3);
    var_dump($result);
?>
```

Результат выполнения кода:

['a', 'b', 'c']

Функция array_splice отрезает и возвращает часть массива. При этом отрезанная часть исчезает из массива. Вместо отрезанной части можно вставлять новые элементы.

Первым параметром указывается массив для разрезания. Вторым параметром указывается, с какого элемента начинать отрезание, а третьим - сколько элементов отрезать. Третий параметр может быть отрицательным - в этом случае отсчет начнется с конца (-1 - последний элемент, -2 - предпоследний и так далее). Третий параметр можно вообще не указывать - в этом случае массив отрежется до самого конца.

В последнем необязательным параметре можно задавать массив элементов, которые будут вставлены взамен удаленных.

Синтаксис

array_splice(массив, откуда отрезать, [сколько], [вставить взамен]);

Пример

Вырежем элементы с первого (имеет номер 0), 3 штуки:

```
<?php
    $arr = ['a', 'b', 'c', 'd', 'e'];
    $result = array_splice($arr, 0, 3);
    var_dump($result);
?>
```

Результат выполнения кода:

['a', 'b', 'c']

При этом массив $arr станет выглядеть так:

['d', 'e']

Функция array_keys получает ключи массива и записывает их в новый массив.

Синтаксис

array_keys(массив);

Пример

Получим ключи из массива:

```
<?php
    $arr = ['a'=>1, 'b'=>2, 'c'=>3, 'd'=>4, 'e'=>5];
    $result = array_keys($arr);
    var_dump($result);
?>
```

Результат выполнения кода:

['a', 'b', 'c', 'd', 'e']

Функция array_values выбирает все значения из массива.

Синтаксис

array_values(массив);

Пример

Получим все значения массива:

```
<?php
    $arr = ['a'=>1, 'b'=>2, 'c'=>3, 'd'=>4, 'e'=>5];
    var_dump(array_values($arr));
?>
```

Результат выполнения кода:

[1, 2, 3, 4, 5]

Функция array_combine осуществляет слияние двух массивов в один ассоциативный. Первым параметром функция принимает массив будущих ключей, а вторым - массив будущих значений.

Синтаксис

array_combine(массив ключей, массив значений);

Пример

Сольем два массива в один ассоциативный. При этом соответствующие элементы из первого массива станут ключами элементов из второго массива:

```
<?php
    $keys = ['a' , 'b', 'c', 'd', 'e'];
    $elems = [1, 2, 3, 4, 5];
    $result = array_combine($keys, $elems);
    var_dump($result);
?>
```

Результат выполнения кода:

['a'=>1, 'b'=>2, 'c'=>3, 'd'=>4, 'e'=>5]

Функция array_flip производит обмен местами ключей и значений массива.

Синтаксис

array_flip(массив);

Пример

Поменяем местами ключи и значения массива:

```
<?php
    $arr = ['a'=>1, 'b'=>2, 'c'=>3, 'd'=>4, 'e'=>5];
    var_dump(array_flip($arr));
?>
```

Результат выполнения кода:

[1=>'a', 2=>'b', 3=>'c', 4=>'d', 5=>'e']

Функция array_reverse переворачивает массив в обратном порядке. Первым параметром передается массив, а вторым - сохранять ключи при перестановке элементов или нет (true - да, false - нет). Второй параметр указывать необязательно. В таком случае по умолчанию вторым параметром является false. Строковые ключи всегда сохраняются, независимо от значения этого параметра.

Синтаксис

array_reverse(массив, [сохранять ли ключи]);

Пример

Перевернем массив:

```
<?php
    $arr = ['a', 'b', 'c', 'd', 'e'];
    var_dump(array_reverse($arr));
?>
```

Результат выполнения кода:

['e', 'd', 'c', 'b', 'a']

Функция array_search осуществляет поиск значения в массиве и возвращает ключ первого найденного элемента. Если такой элемент не найдет - вернет false. Третий параметр задает строгое сравнение по типу (как по ===). Если поставить true - он будет сравнивать строго, а если false (по умолчанию) - то нет.

Синтаксис

array_search(что ищем, где ищем, [сравнивать по типу = false]);

Пример

Найдем в массиве элемент со значением 'c' - в результате получим его ключ (он равен 2):

```
<?php
    $arr = ['a', 'b', 'c', 'd', 'e'];
    echo array_search('c', $arr);
?>
```

Результат выполнения кода:

2

Функция array_replace заменяет значения первого массива значениями с такими же ключами из других переданных массивов. Если ключ из первого массива присутствует во втором массиве, его значение заменяется на значение из второго массива. Если ключ есть во втором массиве, но отсутствует в первом - он будет создан в первом массиве. Если ключ присутствует только в первом массиве, то сохранится как есть.

Если для замены передано несколько массивов, они будут обработаны в порядке передачи и более поздние массивы будут перезаписывать значения из предыдущих.

Синтаксис

array_replace(массив, массив, массив...);

Пример

Заменим элемент с ключом 0 на '!', а элемент с ключом 2 - на '!!':

```
<?php
    $arr = ['a', 'b', 'c', 'd', 'e'];
    $result = array_replace($arr, [0=>'!', 2=>'!!']);
```

```
    var_dump($result);
?>
```

Результат выполнения кода:

['!', 'b', '!!', 'd', 'e']

Для сортировки массивов в PHP существует несколько функций: sort - по возрастанию элементов, rsort - по убыванию элементов, asort - по возрастанию элементов с сохранением ключей, arsort - по убыванию элементов с сохранением ключей, ksort - по возрастанию ключей, krsort - по убыванию ключей, usort - по функции по элементам, uasort - по функции по элементам с сохранением ключей, uksort - по функции по ключам, natsort - натуральная сортировка.

Все эти функции изменяют сам массив — это значит, что результат не нужно никуда присваивать: поменяется сам массив.

Синтаксис

```
sort(массив);
```

Пример

Отсортируем массив по возрастанию элементов:

```
<?php
    $arr = [1, 3, 2, 5, 4];
    sort($arr);
    var_dump($arr);
?>
```

Результат выполнения кода:

[1, 2, 3, 4, 5]

Пример:

Условие задачи. Напишите программу, которая вычисляет произведение элементов побочной диагонали матрицы A(3×3) и заменяет значение первого элемента матрицы этим произведением. Выведите исходную матрицу и результаты вычислений.

Решение:

Код программы:

```
<?php

// Исходная матрица 3×3
$A = [
   [2, 4, 6],
   [1, 5, 9],
```

```
    [7, 3, 8]
];

echo "Исходная матрица:\n";
for ($i = 0; $i < 3; $i++) {
    for ($j = 0; $j < 3; $j++) {
        echo $A[$i][$j] . " ";
    }
    echo "\n";
}

echo "\n";

// Вычисляем произведение элементов побочной диагонали
// Побочная диагональ — элементы A[0][2], A[1][1], A[2][0]

$prod = 1;

for ($i = 0; $i < 3; $i++) {
    $prod *= $A[$i][2 - $i];
}

echo "Произведение элементов побочной диагонали: $prod\n";

// Заменяем первый элемент матрицы (A[0][0])
$A[0][0] = $prod;

echo "\nМатрица после замены первого элемента:\n";
for ($i = 0; $i < 3; $i++) {
    for ($j = 0; $j < 3; $j++) {
        echo $A[$i][$j] . " ";
    }
    echo "\n";
}
?>
```

Результат выполнения:

Методические рекомендации по выполнению заданий 81-100

Для работы с сервером MySQL в Node.js можно использовать ряд драйверов. Самые популярные из них mysql и mysql2. По большей части они совместимы. В данном случае мы будем использовать mysql2, так как, судя по ряду тестов, он предоставляет большую производительность.

Итак, установим пакет mysql2:

npm install --save mysql2

Для создания подключения применяется метод createConnection(), который в качестве параметра принимает настройки подключения и возвращает объект, представляющий подключение.

```
const mysql = require("mysql2");

const connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  database: "usersdb",
  password: "пароль_от_сервера"
});
```

Передаваемые в метод настройки конфигурации могут содержать ряд параметров. Наиболее используемые из них:

- host: хост, на котором запущен сервер mysql. По умолчанию имеет значение "localhost";
- port: номер порта, на котором запущен сервер mysql. По умолчанию имеет значение "3306";
- user: пользователь MySQL, который используется для подключения;
- password: пароль для пользователя MySQL;

- database: имя базы данных, к которой идет подключение. Необязательный параметр. Если он не указан, то подключение идет в целом к северу;
- charset: кодировка для подключения, например, по умолчанию используется "UTF8_GENERAL_CI";
- timezone: часовой пояс сервера MySQL. This is used to type cast server date/time values to JavaScript. По умолчанию имеет значение "local".

Для установки подключения мы можем использовать метод connect() объекта connection:

```
const mysql = require("mysql2");

const connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  database: "usersdb",
  password: "пароль_от_сервера"
});
 connection.connect(function(err){
    if (err) {
      return console.error("Ошибка: " + err.message);
    }
    else{
            console.log("Подключение к серверу MySQL успешно установлено");
    }
 });
```

Метод connect() принимает функцию, параметр которой содержит ошибка, которая возникла при подключении.

Для закрытия подключения применяется метод end():

```
const mysql = require("mysql2");

const connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  database: "usersdb",
  password: "пароль_от_сервера"
});
// тестирование подключения
```

```
connection.connect(function(err){
  if (err) {
    return console.error("Ошибка: " + err.message);
  }
  else{
        console.log("Подключение к серверу MySQL успешно установлено");
  }
});
// закрытие подключения
connection.end(function(err) {
 if (err) {
   return console.log("Ошибка: " + err.message);
 }
 console.log("Подключение закрыто");
});
```

При запуске приложения и удачном подключении-закрытии подключения мы увидим на консоли:

Подключение к серверу MySQL успешно установлено

Подключение закрыто

Метод end() гарантирует, что перед закрытием подключения к бд будут выполнены все оставшиеся запросы, которые не завершились к моменту вызова метода.

Если мы не вызовем этот метод, то подключение будет оставаться активным, и приложение Node.js продолжит свою работу, пока сервер MySQL не закроет подключение.

Если же нам надо немедленно закрыть подключение, не дожидаясь выполнения оставшихся запросов, то в этом случае можно применить метод destroy():

```
connection.destroy()
```

Создадим базу данных на сервере MySQL через Node.js:

```
const mysql = require("mysql2");

const connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "123456"
});
```

```
connection.query("CREATE DATABASE usersdb2",
  function(err, results) {
    if(err) console.log(err);
    else console.log("База данных создана");
});

connection.end();
```

В данном случае посредство команды CREATE DATABASE создается база данных usersdb2.

Теперь добавим в выше созданную базу данных usersdb2 таблицу:

```
const mysql = require("mysql2");

const connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  database: "usersdb2",
  password: "123456"
});

const sql = `create table if not exists users(
  id int primary key auto_increment,
  name varchar(255) not null,
  age int not null
)`;

connection.query(sql, function(err, results) {
    if(err) console.log(err);
    else console.log("Таблица создана");
});
connection.end();
```

Для создания таблицы применяется команда CREATE TABLE, которая создается таблицу users с тремя столбцами - id, name и age.

Для добавления применяется SQL-команда INSERT. Добавим данные в ранее созданную таблицу users:

```
const mysql = require("mysql2");

const connection = mysql.createConnection({
  host: "localhost",
  user: "root",
```

```
  database: "usersdb2",
  password: "123456"
});

const sql = `INSERT INTO users(name, age) VALUES('Sam', 31)`;

connection.query(sql, function(err, results) {
    if(err) console.log(err);
    console.log(results);
});

connection.end();
```

В данном случае в таблицу добавляется одна строка, где столбец name имеет значение "Sam", столбец age - значение 31. С помощью параметра results в функции обратного вызова мы можем получить результаты операции. Например, в моем случае консольный вызов будет следующим:

```
C:\node\mysqlapp> node app.js
ResultSetHeader {
  fieldCount: 0,
  affectedRows: 1,
  insertId: 1,
  info: '',
  serverStatus: 2,
  warningStatus: 0 }
```

В данном случае мы видим, что возвращается объект, где можно выделить ряд свойств. Прежде всего, это affectedRows - количество затронутых операцией строк (в данном случае количество добавленных строк) и insertId - идентификатор (значение поля id) добавленной записи. Соответственно, если бы нам потребовалось получить id добавленной строки, то мы могли бы написать так:

```
connection.query(sql, function(err, results) {
    if(err) console.log(err);
    console.log(results.insertId);
});
```

Добавим сразу несколько значений:

```
const mysql = require("mysql2");
```

```
const connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  database: "usersdb2",
  password: "123456"
});

const users = [
  ["Bob", 22],
  ["Alice", 25],
  ["Kate", 28]
];
const sql = `INSERT INTO users(name, age) VALUES ?`;

connection.query(sql, [users], function(err, results) {
    if(err) console.log(err);
    console.log(results);
});

connection.end();
```

При добавлении множества объектов в sql-запросе после VALUES указывается один вопросительный знак.

И при успешном добавлении свойство results.affectedRows укажет, то добавлено три строки:

```
C:\node\mysqlapp> node app.js
ResultSetHeader {
  fieldCount: 0,
  affectedRows: 3,
  insertId: 3,
  info: 'Records: 3  Duplicates: 0  Warnings: 0',
  serverStatus: 2,
  warningStatus: 0 }
```

Однако в этом случае следует учитывать, что мы не сможем получить id всех добавленных строк.

Для получения данных применяется sql-команда SELECT. Например, получим все данные из таблицы users:

```
const mysql = require("mysql2");

const connection = mysql.createConnection({
```

```
  host: "localhost",
  user: "root",
  database: "usersdb2",
  password: "123456"
});

const sql = `SELECT * FROM users`;

connection.query(sql, function(err, results) {
    if(err) console.log(err);
    console.log(results);
});

connection.end();
```

Объект results в функции обратного вызова будет представлять массив полученных из БД данных:

```
C:\node\mysqlapp> node app.js
[ TextRow { id: 1, name: 'Sam', age: 31 },
  TextRow { id: 2, name: 'Tom', age: 29 },
  TextRow { id: 3, name: 'Bob', age: 22 },
  TextRow { id: 4, name: 'Alice', age: 25 },
  TextRow { id: 5, name: 'Kate', age: 28 },
  TextRow { id: 6, name: 'Tim', age: 22 },
  TextRow { id: 7, name: 'Tom', age: 25 }]
```

Соответственно после получения мы сможем работать с этими данными как с обычным массивом объектов. Например, выведем только имя для каждого пользователя из базы данных:

```
const mysql = require("mysql2");

const connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  database: "usersdb2",
  password: "123456"
});

const sql = "SELECT * FROM users";
connection.query(sql,  function(err, results) {
    if(err) console.log(err);
```

```
    const users = results;
    for(let i=0; i < users.length; i++){
      console.log(users[i].name);
    }
});

connection.end();
```

Консольный вывод:

```
C:\node\mysqlapp> node app.js
Sam
Tom
Bob
Alice
Kate
Tim
Tom
```

Выполним фильтрацию данных с применением выражения WHERE:

```
const mysql = require("mysql2");

const connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  database: "usersdb2",
  password: "123456"
});

const sql = `SELECT * FROM users WHERE name=? AND age=?`;
const filter = ["Tom", 29];
connection.query(sql, filter, function(err, results) {
    if(err) console.log(err);
    console.log(results);
});

connection.end();
```

Здесь запрос фактически будет выглядеть как SELECT * FROM users WHERE name="Tom" AND age=29, и в прицнипе мы могли бы напрямую ввести данные в запрос. Однако чтобы избежать

sql-инъекций при передаче в запрос данных извне рекомендуется использовать параметризацию.

Для обновления данных применяется sql-команда UPDATE:

```
const mysql = require("mysql2");

const connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  database: "usersdb2",
  password: "123456"
});

const sql = `UPDATE users SET age=? WHERE name=?`;
const data = [34, "Tom"];
connection.query(sql, data, function(err, results) {
    if(err) console.log(err);
    console.log(results);
});

connection.end();
```

Консольный вывод:

```
C:\node\mysqlapp> node app.js
ResultSetHeader {
  fieldCount: 0,
  affectedRows: 2,
  insertId: 0,
  info: 'Rows matched: 2  Changed: 2  Warnings: 0',
  serverStatus: 34,
  warningStatus: 0,
  changedRows: 2 }
```

С помощью свойства affectedRows объекта results мы можем проверить, сколько строк было обновлено.

Для удаления применяется sql-команда DELETE:

```
const mysql = require("mysql2");

const connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  database: "usersdb2",
```

```
  password: "123456"
});

const sql = "DELETE FROM users WHERE name=?";
const data = ["Sam"]; // удаляем пользователей с именем Sam
connection.query(sql, data, function(err, results) {
    if(err) console.log(err);
    console.log(results);
});

connection.end();
```

Консольный вывод:

```
C:\node\mysqlapp> node app.js
ResultSetHeader {
  fieldCount: 0,
  affectedRows: 1,
  insertId: 0,
  info: '',
  serverStatus: 34,
  warningStatus: 0 }
```

Пример:

Условие задачи. В базе данных содержится информация о книгах: автор, название, изображение обложки, издательство, год выпуска, цена, количество продаж за последний год.

Запросы:

а) вывести информацию о книгах, цена которых лежит в заданном диапазоне;

б) вывести информацию о книгах заданного автора.

Решение:

Используем пакет mysql2 для подключения к MySQL и readline для ввода данных пользователем.

Код программы:

```
// Установка: npm install mysql2
const mysql = require('mysql2');
const readline = require('readline');

// Создание интерфейса для ввода данных
const rl = readline.createInterface({
    input: process.stdin,
```

```
    output: process.stdout
});

// Подключение к базе данных MySQL
const connection = mysql.createConnection({
    host: 'localhost',
    user: 'root',
    password: 'your_password',
    database: 'books_db'
});

// Функция для запроса книг по цене
function getBooksByPrice(min, max) {
    connection.query(
        'SELECT * FROM books WHERE price BETWEEN ? AND ?',
        [min, max],
        (err, results) => {
            if (err) throw err;
            console.log(`Книги с ценой от ${min} до ${max}:`);
            console.table(results);
        }
    );
}

// Функция для запроса книг по автору
function getBooksByAuthor(author) {
    connection.query(
        'SELECT * FROM books WHERE author = ?',
        [author],
        (err, results) => {
            if (err) throw err;
            console.log(`Книги автора ${author}:`);
            console.table(results);
        }
    );
}

// Пример использования
```

```
rl.question('Выберите тип запроса (1 - по цене, 2 - по автору): ', (choice) => {
    if (choice === '1') {
        rl.question('Введите минимальную цену: ', (min) => {
            rl.question('Введите максимальную цену: ', (max) => {
                getBooksByPrice(min, max);
                rl.close();
            });
        });
    } else if (choice === '2') {
        rl.question('Введите автора: ', (author) => {
            getBooksByAuthor(author);
            rl.close();
        });
    } else {
        console.log('Неверный выбор');
        rl.close();
    }
});
```

Результат выполнения:

id	author	title	cover	publisher	year	price	sales
1	Иванов И.И.	Математика для всех	img1.jpg	Изд-во А	2020	25	100
2	Петров П.П.	PHP с нуля	img2.jpg	Изд-во Б	2019	40	50
3	Иванов И.И.	Алгебра	img3.jpg	Изд-во А	2021	35	80
4	Сидоров С.С.	JavaScript для всех	img4.jpg	Изд-во В	2022	30	120

```
Выберите тип запроса (1 - по цене, 2 - по автору): 1
Введите минимальную цену: 30
Введите максимальную цену: 40
```

Результат:

id	author	title	cover	publisher	year	price	sales
2	Петров П.П.	PHP с нуля	img2.jpg	Изд-во Б	2019	40	50
3	Иванов И.И.	Алгебра	img3.jpg	Изд-во А	2021	35	80
4	Сидоров С.С.	JavaScript...	img4.jpg	Изд-во В	2022	30	120

```
Выберите тип запроса (1 - по цене, 2 - по автору): 2
Введите автора: Иванов И.И.
```

Результат:

id	author	title	cover	publisher	year	price	sales
1	Иванов И.И.	Математика	img1.jpg	Изд-во А	2020	25	100
3	Иванов И.И.	Алгебра	img3.jpg	Изд-во А	2021	35	80

Таблица 1 – Варианты заданий на домашнюю контрольную работу по учебному предмету «Веб-программирование на стороне сервера»

Предпоследняя цифра номера шифра	Последняя цифра номера шифра									
	0	1	2	3	4	5	6	7	8	9
0	1,42, 65,81	2,41, 66,82	3,43, 67,83	4,44, 68,84	5,45, 69,85	6,46, 70,86	7,47, 71,87	8,48, 72,88	9,49, 73,89	10,50, 74,90
1	21,51, 75,91	22,52, 76,92	23,53, 77,93	24,54, 78,94	25,55, 79,95	26,56, 80,96	27,57, 61,97	28,58, 62,98	29,59, 63,99	30,60, 64,100
2	11,40, 70,85	12,41, 72,86	13,42, 73,87	14,43, 74,88	15,44, 75,89	16,45, 76,90	17,46, 77,91	18,47, 78,92	19,48, 79,93	20,49, 80,94
3	31,51, 69,95	32,52, 68,96	33,53, 67,97	34,54, 66,98	35,55, 65,99	36,56, 64,100	37,57, 63,86	38,58, 62,87	39,59, 61,88	40,60, 79,89
4	1,60, 78,90	2,59, 77,91	3,58, 76,92	4,57, 75,93	5,56, 74,94	6,55, 73,95	7,54, 72,96	8,53, 71,97	9,52, 70,98	10,51, 69,99
5	21,50, 68,100	22,49, 67,91	23,48, 66,92	24,47, 65,93	25,46, 64,94	26,45, 63,95	27,44, 62,96	28,43, 61,97	29,42, 80,98	30,41, 78,99
6	11,41, 76,81	12,42, 74,82	13,43, 72,83	14,44, 70,84	15,45, 68,85	16,46, 66,86	17,47, 64,87	18,48, 62,88	19,49, 79,89	20,50, 77,90
7	31,51, 75,91	32,52, 68,92	33,53, 66,93	34,54, 61,94	35,55, 74,95	36,56, 72,96	37,57, 80,97	38,58, 64,98	39,59, 65,99	40,60, 80,100
8	1,50, 61,8,	2,51, 62,85	3,52, 63,86	4,53, 64,87	5,54, 65,88	6,55, 66,89	7,56, 67,90	8,57, 68,91	9,58, 69,92	10,59, 70,93
9	21,41, 71,94	22,42, 72,95	23,43, 73,96	24,44, 74,98	25,45, 75,97	26,46, 76,99	27,47, 77,81	28,48, 78,82	29,49, 79,83	30,50, 80,84