Министерство образования и науки РФ
Нижневартовский государственный гуманитарный университет
Кафедра информатики и МПИ

Р.Х. Хакимов

СЕРВЕРНОЕ WEB-ПРОГРАММИРОВАНИЕ

Учебное пособие

Нижневартовск
2012

ББК 32.973.202
Х 16

Печатается по постановлению редакционно-издательского совета
Нижневартовского государственного гуманитарного университета

Рецензенты:

доктор технических наук, профессор кафедры АСУ Уфимского государственного авиационного технического университета *В.В.Миронов*;

кандидат физико-математических наук, профессор кафедры естественнонаучных дисциплин Нижневартовского филиала Тюменского государственного нефтегазового университета *В.А.Беляев*

Хакимов Р.Х.

Х 16 **Серверное Web-программирование:** Учебное пособие. — Нижневартовск: Изд-во Нижневарт. гуманит. ун-та, 2012. — 128 с.

ISBN 978–5–89988–949–3

Учебное пособие дает достаточно полное представление о серверном Web-программировании (язык PHP, СУБД MySL, обработка запросов на сервере, взаимодействие PHP и MySL).

Практические задания позволяют получить навыки создания серверных Web-приложений.

Пособие предназначено для студентов, изучающих «Web-программирование», «Web-технологии» и смежные дисциплины, будет полезно и другим категориям читателей, приступающих к работе в области создания Web-приложений.

ББК 32.973.202

Изд. лиц. ЛР № 020742. Подписано в печать 25.09.2012
Формат 60×84/16. Бумага для множительных аппаратов
Гарнитура Times. Усл. печ. листов 8
Тираж 300 экз. Заказ 1329

Отпечатано в Издательстве
Нижневартовского государственного гуманитарного университета
628615, Тюменская область, г.Нижневартовск, ул.Дзержинского, 11
Тел./факс: (3466) 43-75-73, E-mail: izdatelstvo@nggu.ru

ISBN 978–5–89988–949–3

ПРЕДИСЛОВИЕ

Web-программирование является все более широко используемой областью в программировании. Оно берет начало со времени создания «всемирной паутины» — WWW и постоянно развивается и расширяется. Для создания современного сайта требуется использовать все разновидности Web-программирования: HTML-разметку, каскадные таблицы стилей, Web-программирование клиентское и серверное.

Серверное Web-программирование — это технология, позволяющая запускать на web-сервере программы, имеющие возможность получать данные от посетителей сайтов, поддерживаемых этим Web-сервером, и в свою очередь выдавать им обработанные данные в виде Web-страниц или других файлов. Таким образом, серверное Web-программирование — это создание программ, работающих на сервере, а не на клиентской машине.

Хотя на рынке специальной литературы достаточно книг по «сайтостроительству», потребность в таком учебном пособии имеется по нескольким причинам. Одна из них — нет книг по Web-программированию, которые можно использовать на занятиях — для этого их надо очень сильно адаптировать к учебному процессу, «препарировать».

Далее, инструментарий Web-программирования создавался в основном не университетскими профессорами, а действующими высококвалифицированными, талантливыми программистами. Они в первую очередь учитывали интересы производственного процесса. Этот инструментарий ориентирован на квалифицированного пользователя. Интересы процесса обучения, самих обучаемых учитывались разработчиками, конечно, в последнюю очередь.

В инструментарии Web-программирования нет той строгости, однозначности, логичности, которые присущи классическим языкам программирования. Вспомните оператор вывода в Turbo Pascal и сравните его со способами использования оператора вывода `echo` в PHP. В первом случае однозначность, строгость, во втором — «разнобой». Разрешено и то, и другое, и третье. Такая свобода не всегда полезна для дела, особенно при обучении.

Все сказанное говорит в пользу того, что нужны методически выдержанные учебные пособия по Web-программированию.

Учебное пособие в первую очередь предназначено студентам специалитета (ИСиТ, ПОВТ и АС) и будущим бакалаврам, обучающимся по направлениям «Информатика и ВТ», «Информационные системы и технологии», «Прикладная математика и информатика» для приобретения навыков серверного Web-программирования. Но оно может быть полезно и студентам других специальностей при изучении предметов, связанных с Web-технологиями.

Соответствие учебного пособия образовательным стандартам и характеристики дисциплин, обслуживаемых им, представлены в таблице.

№ п/п	Направление подготовки и присваиваемая квалификация	Название дисциплины, позиция дисциплины в учеб. плане, семестр	Соответствие дидактич. единицам и матрице компетенций ФГОС	Трудоемкость дисциплины в часах (зач. ед.)
1	050202.65 — «Информатика», магистр	Web-конструирование, ДВМ.03 семестр 4	+	80
2	230105.65 — «Программное обеспечение вычислительной техники и автоматизированных систем», инженер	Web-технологии, ЕН.Р.01 семестр 8	+	90
3	230201.65 — «Информационные системы и технологии», инженер	Web-технологии, ЕН.Р.01 семестр 8	+	151
4	230100 — «Информатика и ВТ», бакалавр	Основы Web-программирования, Цикл: Б.3. Профессиональный цикл, Вариативная часть В.09 семестр 6	ОК-3, ОК-13, ПК-2, ПК-3, ПК-5, ПК-6,	108 (3)

5	230400 — «Информационные системы и технологии», бакалавр.	Разработка информационных систем на основе Web-проектов, Цикл: Б.3. Профессиональный цикл, Вариативная часть, В.05 семестр 7	ОК-3, ПК-1, ПК-2, ПК-3, ПК-4, ПК-29, ПК-30, ПК-31	108 (3)
6	010400 — «Прикладная математика и информатика», бакалавр	Web-программирование, Цикл: Б.3. Профессиональный цикл, Вариативная часть В.03 семестр 7	ОК-3, ОК-13, ПК-2, ПК-3, ПК-5, ПК-6,	72 (2)

В период рассмотрения данного учебного пособия экспертной комиссией НГГУ пособие обслуживало преподавание по двум дисциплинам (позиции 1, 3 в таблице). Для студентов факультета учебное пособие будет обслуживать 6 дисциплин (6 специальностей и направлений обучения).

Существующая и прогнозируемая динамика использования учебного пособия на факультете в период 2012—2016 гг. представлена в таблице.

год	2012		2013		2014		2015		2016	
группа	Семестр, количество студентов, использующих учебное пособие									
11 ИВТ					6-й, 25 ч.					
12 ИВТ					6-й, 25 ч.					
13 ПМИ						7-й, 25 ч.				
21 ПОВТ					8-й, 25 ч.					
22 ПОВТ					8-й, 25 ч.					
23 ФМО										
31 ПОВТ			8-й, 25 ч.							

32 ИСИТ			8-й, 25 ч.							
33 ФМО										
34 ПМ		7-й, 20 ч.								
41 ИСИТ	8-й, 15 ч.									
51 ПОВТ										
52 ПМ										
МАГ, ИНФ	4-й, 7 ч.									
ПРИЕМ В 2012 г.										
11 ИВТ							6-й, 25 ч.			
12 ИВТ							6-й, 25 ч.			
13 ПМИ								7-й, 25 ч.		
Всего чел.	22	20	50		100	25	50	25		

Пособие состоит из 5 разделов, литературы, обзора терминов и понятий из области серверного Web-программирования и тестовых заданий. В первом разделе рассмотрены разновидности Web-серверов (пассивные и активные), схема взаимодействия клиента с активным сервером; рассмотрен механизм программных расширений сервера — CGI (Common Gates Interface). Обоснован выбор языка PHP в качестве наиболее эффективного инструмента серверного Web-программирования.

Во втором разделе на примере решения конкретной задачи показана технология обработки данных, полученных с HTML-формы, с помощью PHP-программы. Рассматривается процесс установки локального Web-сервера для создания сайтов. Рассмотрены особенности синтаксиса языка PHP, необходимые для решения данной задачи, рассмотрены методы передачи данных с формы серверной программе.

Третий раздел посвящен работе с СУБД MySQL, ее графическим интерфейсом PhpMyAdmin. Кратко изложен материал по структурированному языку запросов SQL. На примерах излагается материал по программированию взаимодействия PHP и MySQL.

В четвертом разделе в объеме, достаточном для решения рассматриваемых в пособии практических задач, изложен теоретический материал по языку серверного Web-программирования PHP.

В пятом разделе даны задания для выполнения трех лабораторных работ. Учитывая, что «серверное Web-программирование» обычно изучается в рамках дисциплин типа: «Web-программирование», «Web-технологии» и т.д. и составляет только часть дисциплины, три лабораторные работы для одного раздела дисциплины можно считать приемлемым числом.

Завершают пособие обзор терминов и понятий из области серверного Web-программирования и тестовые задания, которые предназначены для использования при балльно-рейтинговой системе оценки знаний студентов.

Структуры разделов сильно различаются. Есть раздел с чисто теоретический материалом, и разделы с заданиями, выполняемыми в режиме пошаговых подсказок; заданиями для самостоятельного выполнения и контрольными вопросами.

Изложение материала реализовано по схеме «от практики к теории». Поэтому чисто теоретический материал по языку PHP вынесен в четвертый раздел, а во втором и третьем разделах решаются практические задачи серверного Web-программирования с минимальным привлечением теории.

Автор попытался обеспечить благоприятные условия для усвоения этой области Web-технологий за счет методически правильного выбора материала для первоначального изучения, используемого стиля изложения, выбора примеров и заданий для практического выполнения.

1. ВВЕДЕНИЕ В СЕРВЕРНОЕ WEB-ПРОГРАММИРОВАНИЕ

Web-программирование — это разработка любых программных продуктов, предназначенных для работы на сайтах World Wide Web. Имеются следующие разновидности Web-программирования:

1. Разработка Web-страниц на чистом HTML. Является Web-программированием, потому что при просмотре страницы браузер фактически исполняет код HTML, форматируя текст согласно инструкциям этого языка. На профессиональных сайтах, как правило, для оформления используются каскадные таблицы стилей CSS (cascading style sheets). Поэтому их использование также следует отнести к Web-программированию.

2. Клиентское Web-программирование. Это использование на Web-страницах технологий JavaScript и VBScript для динамического изменения внешнего вида Web-страницы при ее просмотре и выполнение обработки информации, введенной пользователем в формы.

3. Серверное Web-программирование. Это создание CGI-приложений. CGI (сокращение от Common Gateway Interface) — технология, позволяющая запускать на Web-сервере программы, имеющие возможность получать данные от посетителей сайтов, поддерживаемых этим Web-сервером, и в свою очередь выдавать им обработанные данные в виде Web-страниц или других файлов.

Прежде чем перейти к детальному описанию особенностей Web-приложений, вспомним принципы работы Web-сервера.

1.1. Пассивные и активные серверы Web

Web-сервер — это программа, устанавливаемая на узле сети Интернет и выдающая посетителям этого узла Web-страницы по запросам. Также Web-сервером часто называется узел, на котором эта программа запущена, или даже компьютер, являющийся таким узлом.

По данным за 2004 г., в Интернете наиболее часто используются серверы Apache (67,2%) и MS IIS (21,02%) [8].

Различают пассивные и активные серверы Web. Если страницы сервера содержат только статическую текстовую и мультимедийную информацию, а также гипертекстовые ссылки на другие страницы, то сервер называется *пассивным*. Такой сервер способен только выдавать Web-страницы по запросам пользователей.

В отличие от пассивных, *активные* серверы могут:

- вступать в диалог с пользователем, например, запрашивая и принимая от него информацию;
- динамически создавать (не просто извлекать из своей памяти имеющиеся там Web-страницы) Web-страницы для предъявления клиентам;
- при формировании динамических страниц обращаться к базам данных сервера, извлекать оттуда информацию и включать ее в формируемую Web-страницу.

Очевидно, что Web-страницы активного сервера кроме статической текстовой и мультимедийной информации, а также гипертекстовых ссылок должны содержать другие элементы для реализации перечисленных возможностей.

Имеется два варианта реализации активных Web-серверов. Первый из них предполагает применение специальных программных расширений Web-сервера, таких как CGI и ISAPI.

Второй связан с использованием серверных сценариев и технологии активных страниц Active Server Pages (ASP).

Активные серверы могут отличаться друг от друга функциональностью и специфическими ограничениями. Например, Perl присутствует, а PHP нет; PHP есть, но разрешается создавать только одну базу данных.

1.2. Программы CGI, схема их работы

Для того чтобы сервер Web мог вести диалог с пользователем, был разработан механизм программных расширений сервера, называемый *стандартным шлюзовым интерфейсом* Common Gateway Interface (CGI). Это часть Web-сервера, которая может взаимодействовать с другими программами, выполняющимися на этом Web-узле. И в этом смысле CGI является шлюзом для передачи данных, полученных от клиента [9].

Общая схема работы CGI состоит из следующих этапов:

- получение (через протокол HTTP) информации с клиентских машин (от пользователя);
- обработка полученной информации. В некоторых случаях CGI-программа не может обработать информацию (ответить на запрос) самостоятельно. Например, запрос может потребовать обращения к некоторой базе данных, которую CGI-программа читать не умеет. В этом случае CGI-программа на основании полученной информации формирует запрос к компетентной программе, выполняющейся на том же Web-сервере;
- отправка обработанной информации обратно в виде нового документа HTML.

При этом для ввода информации со стороны пользователя в документ HTML встраиваются формы, содержащие такие органы управления, как текстовые поля, списки, переключатели, кнопки и т.д.

Обычно одна из кнопок (типа Submit) предназначена для завершения ввода данных в форму. Когда пользователь заполнит всю форму, он нажимает эту кнопку, и данные из полей формы передаются программе CGI. Обработав данные, программа CGI динамически формирует новый документ HTML с результатами обработки и отправляет его обратно пользователю. При необходимости программа CGI может обращаться к СУБД.

Этот процесс проиллюстрирован на рис. 1.1.

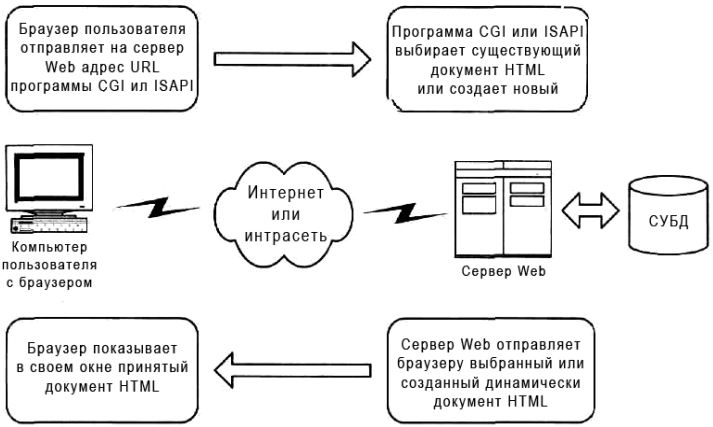


Рис. 1.1. Взаимодействие браузера и сервера Web

Таким образом, программы CGI позволяют Web-серверу вести диалог с пользователем.

Что представляет собой программа CGI?

Это обычное консольное приложение, работающее в среде операционной системы сервера Web и осуществляющее обмен данными через стандартные потоки ввода и вывода [9].

Такое приложение загружается в оперативную память сервера и запускается только по явному запросу пользователя, когда к нему выполняется обращение из документа HTML (с помощью тега типа <form action="obrabotka.php" method = "post">). Здесь CGI-программа — это "obrabotka.php". Окончив обработку запроса пользователя, программа CGI завершает свою работу и выгружается из памяти сервера.

Программа CGI работает на сервере как отдельный процесс. В рамках этого процесса она способна, в частности, обращаться к базам данных для выборки или обновления информации.

1.3. Язык создания CGI-сценариев — PHP

Для использования технологии CGI программа Web-сервер должна удовлетворять определенным критериям — «поддерживать CGI». CGI-программы, запускаемые на Web-сервере, представлены не двоичным кодом (т.е. скомпилированным файлом), а текстовым. Поэтому для их выполнения требуется программа-интерпретатор того языка, на котором написана эта программа. Такой интерпретатор включается в состав Web-сервера и вызывается им при необходимости заняться выполнением программного кода.

CGI-сценарий (CGI-скрипт) — программа (в текстовом виде), предназначенная для исполнения на Web-сервере. Для создания CGI-скриптов можно использовать любой язык программирования — важно лишь, чтобы на том Web-сервере, где предполагается эту программу запускать, имелся интерпретатор этого языка.

Препроцессор — программа (интерпретатор), работающая совместно с Web-сервером, которая просматривает все или некоторые файлы, выдаваемые Web-сервером посетителям, и выполняет над ними определенные действия в зависимости от содержащихся в этих файлах инструкций. Примером препроцессора является PHP (первоначальная расшифровка была Personal Home Page).

От других CGI-языков PHP отличается в лучшую сторону прежде всего своей простотой.

При создании программы на PHP нет необходимости учитывать все те многочисленные мелочи, которые усложняют жизнь программистам на Perl или C++ [9]. Синтаксис языка PHP допускает его легкое освоение как начинающим программистом, так и тем, кто уже использовал ранее какой-либо язык программирования. Можно целиком и полностью сосредоточиться на решаемой задаче и не думать о мелочах.

Именно это и делает PHP наиболее подходящим вариантом для Web-дизайнера.

Одним из достоинств PHP является возможность без особых затруднений работать с серверами баз данных. Благодаря PHP использование базы данных на сайте стало едва ли не элементарным. Для работы с подавляющим большинством типов баз данных в PHP есть встроенные функции, поэтому теперь достаточно лишь установить на Web-сервере программу работы с базой данных (наиболее часто используется бесплатная MySQL — http://www.mysql.com) и включить в текст PHP-сценария команды работы с нею.

Контрольные вопросы

1. Назовите и опишите разновидности Web-программирования.
2. Сформулируйте определение Web-сервера.
3. Покажите различия между пассивными и активными Web-серверами.
4. Назовите способы реализации активных Web-серверов.
5. Расшифруйте аббревиатуру CGI и объясните его природу.
6. Опишите схему работы CGI.
7. Что такое препроцессор?
8. Перечислите преимущества языка PHP перед другими CGI-языками.

2. ОБРАБОТКА НА СЕРВЕРЕ ЗАПРОСОВ С КЛИЕНТСКОЙ МАШИНЫ

2.1. PHP-сценарий обработки на сервере заказа на автозапчасти, отправленного с клиентской машины

Мы знаем и понимаем, что пассивный (статический) сайт, написанный на языке HTML, можно отлаживать на локальном (клиентском) компьютере с помощью браузера. Далее отлаженный в таком режиме сайт можно публиковать (перенести) на сервер выбранного нами провайдера. И даже активный (динамический) сайт, содержащий скрипты на языке JavaScript, можно отлаживать на локальном компьютере с помощью браузера. Потому что интерпретатор скриптов JavaScript содержится в составе самого браузера.

Иначе обстоит дело при серверном Web-программировании, т.е. создании программ, выполняющихся на сервере. Как было сказано ранее, такие программы чаще всего пишут на языке PHP.

Для демонстрации принципов серверного Web-программирования мы создадим сайт, иллюстрирующий работу магазина автозапчастей [8]. Как увидим далее при описании функциональности, сайт наш будет интерактивный (пользователь будет вводить данные, серверная программа на языке PHP эти данные будет обрабатывать и выдавать результат). Следовательно, такой сайт нельзя будет отлаживать на локальном компьютере только с помощью браузера, как отлаживаются статические сайты. Нельзя хотя бы потому, что в составе браузера нет интерпретатора языка PHP (есть и другие причины).

Такой сайт уже на этапе создания и отладки надо поместить на Web-сервер. Мы так и сделаем. Но мы поместим свой сайт не на «настоящем» удаленном Web-сервере, принадлежащем провайдеру, а воспользуемся локальным или, по-другому, виртуальным Web-сервером, имитирующим удаленный сервер. Материал об установке такого Web-сервера дан в разделе 2.3.

На нашем локальном сервере сайт будет представлен двумя файлами:
C:\WebServers\home\autoparts\www\Index.htm, Obrabotka.php.

Обратившись по адресу http://autoparts к сайту магазина, мы будем получать форму для ввода заказа (рис. 2.1). Обратите внимание на то, что мы указываем только часть адреса сайта, остальная часть формируется сервером.

Рис. 2.1. Форма для ввода заказа

Форма должна будет храниться на сервере в папке сайта в файле под именем index.htm. В форме для ввода количества заказываемых товаров используются текстовые поля. Для отправки заполненной формы на сервер служит кнопка **Отправить заказ.**

Для обработки заказа и вывода на экран клиента результатов обработки на сервере в папке сайта должна быть программа обработки заказа, написанная на языке PHP. Назовем ее obrabotka.php.

Код формы (в файле index.htm) будет следующий.

Листинг 2.1

```
<html>
<body>
<h1>Запчасти от Занифа</h1>
<h2>Форма заказа</h2>
<form action="obrabotka.php" method="post">
<table border=2>
<tr bgcolor=#cccccc>
<td width=150>Товар</td>
<td width=100>Количество</td>
</tr>
```

```
<tr>
<td>Шины</td>
<td align="center"><input type="text" name="tireqty" size=3
maxlength=3> </td>
</tr>
<tr>
<td> Масло</td>
<td align="center"> <input type="text" name="oilqty" size=3
maxlength=3></td>
</tr>
<tr>
<td>Свечи зажигания</td>
<td align=center> <input type="text" name="sparkqty" size=3
maxlength=3></td>
</tr>
<tr>
<td colspan=2 align=center> <input type="submit"
value="Отправить заказ"></td>
</tr>
</table>
</form>
</body>
</html>
```

Комментарии к коду формы

Для ввода количества заказываемых товаров используются текстовые поля. Для обеспечения хорошего дизайна они помещены в ячейки таблицы. Таблица несколько усложняет восприятие HTML-кода, но улучшает интерфейс.

Для отправки заполненной формы на сервер служит кнопка Submit.

В теге :

```
<form action="obrabotka.php" method="post">
```

указаны:

- метод передачи данных на сервер — method="post"
- имя серверной PHP-программы для обработки заказа — obrabotka.php

Заполнив заказ, отправляем его на сервер, нажав кнопку Submit.

Необходимо, конечно, понимать метод передачи данных с формы на сервер — post. Этим мы займемся позже.

Пока примем на веру то, что имена текстовых полей для ввода количества заказываемых изделий в коде формы (tireqty, oilqty, sparkqty) должны использоваться и в программе obrabotka.php, обрабатывающей заказ и формирующей Web-страницу для отправки клиенту.

Как только программа obrabotka.php получит значения полей tireqty, oilqty, sparkqty, она начнет работать и, для наших введенных в форму данных, результат обработки вернет на клиентскую машину в виде следующей Web-страницы.

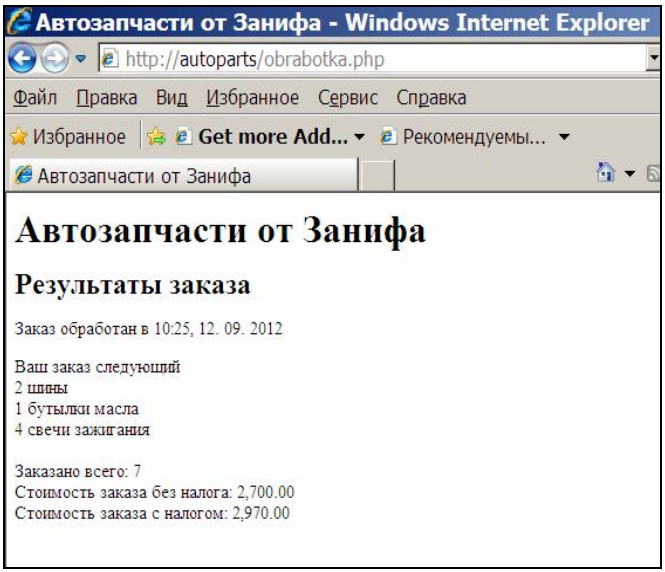


Рис. 2.2. Web-страница с результатом обработки заказа

В случае пустых полей ввода должно быть выдано сообщение: Вы ничего не заказали на предыдущей странице! Цвет текста должен быть красный.

Серверная PHP-программа для такой обработки заказа — obrabotka.php будет иметь следующий код.

Листинг 2.2

```
<html>
<head>
<title>автозапчасти от Занифа</title>
</head>
<body>
<h1 align="center"> Автозапчасти от Занифа</h1>
<h2 align="center"> Результаты заказа</h2>
<?
echo "Заказ обработан в ";
echo date ("H:i.d.m.Y")."<br>";
$totalqty=$tireqty+$oilqty+$sparkqty;
if ($totalqty==0)
{
echo "<font color=red>";
echo "Вы ничего не заказали на предыдущей странице";
echo "</font>";
exit();
}
echo "Ваш заказ следующий:<br>";
echo $tireqty." шины <br>";
echo $oilqty." масло <br>";
echo $sparkqty." свечи <br>";
echo "<br>Заказано всего: ".$totalqty."<br>";
define ("TIREPRICE",1000);
define ("OILPRICE",300);
define ("SPARKPRICE",100);
$totalamount=TIREPRICE*$tireqty+OILPRICE*$oilqty+
SPARKPRICE*$sparkqty;
$totalamountf=number_format($totalamount,2);
echo "<br> Стоимость заказа без налога:".$totalamountf;
$taxrate=0.1;
$totalamount=$totalamount*(1+$taxrate);
$totalamountf=number_format($totalamount,2);
echo "<br> Стоимость заказа с налогом:".$totalamountf;
?>
</body>
```

2.2. Особенности синтаксиса языка PHP

Систематическое и достаточно полное для решения наших задач изложение материала по языку PHP дано в разделе 4 этого пособия. Для получения детальной и исчерпывающей информации, естественно, надо обращаться к специальной литературе, перечень которой приведен в списке литературы.

Чтобы понять эту программу контингенту, для которого предназначено данное пособие, достаточно ознакомиться с особенностями синтаксиса языка PHP и несколькими операторами. Они представлены в следующей таблице.

Таблица 2.1

Действие	Пример и описание
Внедрение PHP-скриптов в HTML-код	Простейший способ такой: <? PHP-скрипт ?>
Комментарий однострочный	// Вывод даты в формате: 23:15, 08.05.2009 или # Однострочный комментарий
Комментарий многострочный	/* Многострочный комментарий*/
Оператор программы завершается символом «;». Символ «;» ставится даже перед «else»	$totalqty=$tireqty+$oilqty+$sparkqty;
Правила конструирования идентификаторов	Первым символом в идентификаторах переменных должен быть символ $. Остальные требования к идентификаторам как в Паскале. Но идентификаторы регистрозависимые (однако, имена встроенных функций и служебные слова не регистрозависимые)
Строковые значения заключаются в кавычки	$name=”Расмус Лердоф”;
Задание констант	define("OILPRICE", 300);
Вывод данных на экран	Оператор echo
Перевод курсора в начало следующей строки	echo " ";

Тег <P> перед выводимой строкой вставляет пустую строку	echo "<p>Ваш заказ следующий";
Вывод строкового значения	echo "Заказ обработан в ";
Вывод значения арифметического выражения	echo (20+12); или Echo 20+12;
Вывод значения переменной	echo $name;
Вывод значения константы	echo CONST; //здесь CONST имя константы
Вывод смешанного значения	echo "$oilqty бутылки масла ";
Конкатенация	echo $oilqty. " бутылки масла ";
Конкатенация и вычисление значения функции	Echo “В массиве ”. count($arr). “элементов”
Вывод имен переменных без интерпретации в виде последовательности символов	Echo ‘$name Расмус Лердоф’; (выведет строку «$name Расмус Лердоф»)
Вывод значения функции	echo date ("H:i, d. m. y.");
Форматирование для вывода числа с двумя десятичными знаками	$totalamountf=number_format ($totalamount,2);
Условный оператор	If (условие) {ветвь 1}; else {ветвь 2};

Обратите внимание на то, что имена текстовых полей для ввода количества заказываемых изделий в коде формы (tireqty, oilqty, sparkqty) с очень небольшим изменением используются и в программе obrabotka.php, обрабатывающей заказ и формирующей Web-страницу для отправки клиенту. Изменение имен полей сводится к добавлению в начало идентификатора символа $.

Возникает вопрос: как программа обработки получает значения этих переменных, введенных в форму пользователем. Ответ на этот вопрос дается в следующем подразделе.

2.3. Методы передачи информации в серверную PHP-программу

В серверном Web-программировании достаточно распространенной задачей является передача данных (параметров) PHP-программе извне (из другой программы, из адресной строки браузера).

В подразделе 2.1 потребовалась передача данных, введенных в форму, и имен этих данных обрабатывающей PHP-программе. Там для этого был использован метод POST.

2.3.1. Метод POST

Работает следующим образом. Данные, необходимые обрабатывающей программе, вводятся пользователем в соответствующие поля HTML-формы (рис. 2.1). В нашей задаче это поля типа Text.

При нажатии кнопки типа Submit (у нас это кнопка **Отправить заказ**) имена полей ввода и введенные в них значения отправляются на сервер блоком данных и там могут быть использованы указанной в теге <FORM> программой (у нас это программа из файла obrabotka.php).

Чтобы обеспечить такую совместимость, в обрабатывающей программе для обозначения получаемых с формы данных надо использовать имена, получаемые путем добавления к именам полей формы символа $ (в нашем примере — $tireqty, $oilqty, $sparkqty).

На некоторых серверах в обрабатывающей программе эти имена и значения переменных можно использовать без всяких подготовительных операций (как в нашей программе):

```
echo $tireqty. " шины <br>";
echo $oilqty. " бутылки масла <br>";
echo $sparkqty. " свечи зажигания <br>";
```

Но эта возможность определяется состоянием директивы register_globals в разделе Data Handling конфигурационного файла php.ini (он находится по адресу c:\webservers\usr\local\php5\php.ini). Если она выключена (по умолчанию это так), т.е. register_globals = Off, такое прямое использование будет запрещено.

Тогда приходится пользоваться так называемым суперглобальным массивом $_POST. Этот ассоциативный массив формируется на сервере из данных, поступающих с формы. В нашем случае он будет иметь вид:

```
$_POST= array (“tireqty” =>2, “oilqty” => 1, “sparkqty” =>4)
```

Здесь показан один из способов формирования ассоциативного массива в PHP.

Имея некоторые знания по работе с обычными одномерными массивами, можно догадаться, что ассоциативный массив отличается от обычного тем, что в нем вместо числовых индексов используются словесные, «смысловые» индексы. Значения элементов массива взяты из нашего примера, из заполненной формы.

Для такого сервера нашу обрабатывающую программу пришлось бы изменить. А именно, значения требуемых переменных пришлось бы брать из суперглобального массива таким образом:

```
$tireqty =$_POST [“tireqty”];
$oilqty =$_POST [“toilqty”];
$sparkqty =$_POST [“sparkqty”];
```

2.3.2. Метод GET

При отправке данных с формы на сервер по методу GET содержимое формы добавляется в адресной строке браузера к URL программы, принимающей эти данные, в следующем виде:

http://action.php?name1=value1&name2=value2&name3=value3.

Здесь **action.php** — это URL-адрес программы, которая должна обрабатывать форму. Обычно это программа, заданная в атрибуте **action** тега **form**. Имена `name1`, `name2`, `name3` соответствуют именам элементов формы, а `value1`, `value2`, `value3` — значениям этих элементов.

В нашем примере после ввода данных в форму и нажатия клавиши **Отправить заказ** в адресную строку браузера занесется следующее значение:

http://autoparts/obrabotka.php?tireqty=2&oilqty=1&sparqqty=4.

Таким образом, передача данных серверной программе по методу GET происходит совсем не так, как при методе POST. А именно, передаваемые данные «проходят» через адресную строку браузера, и они видны пользователю (рис. 2.3).

В принципе, для передачи данных методом GET вводить данные в поля HTML-формы не обязательно. Можно просто добавить в строку URL нужные переменные и их значения.

Это будет выглядеть, как представлено на рис. 2.4. И данные будут переданы в серверную программу, хотя в поля формы данные не введены.

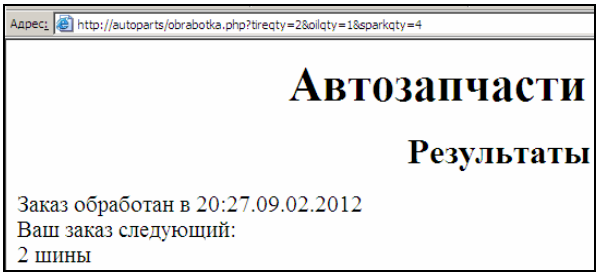


Рис. 2.3. Передача данных с формой по методу GET

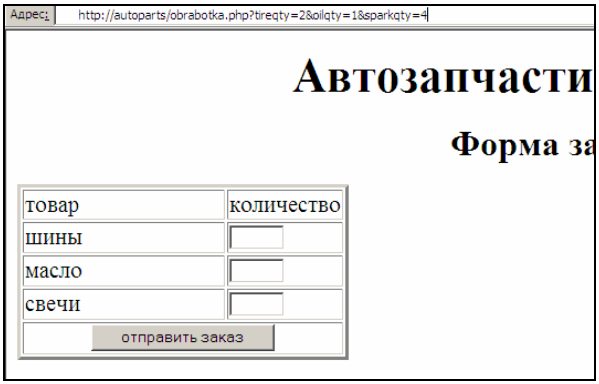


Рис. 2.4. Передача данных с формой без заполнения формы

Итак, из рассмотренного примера понятно, что для полей ввода текста (это элементы **input** с атрибутом `type=text`) передаваемым значением будет то, что введет пользователь. Это же справедливо и для поля ввода пароля (тег **input** с атрибутом `type=password`) **.** Если пользователь ничего не вводит в такое поле, то в строке адреса будет присутствовать элемент `name=(без значения)`, где `name` соответствует имени этого элемента формы.

Доступ к переданным данным в серверной программе происходит по той же схеме, что и в методе POST. Если директива register_globals в конфигурационном файле php.ini включена, переданные из формы значения можно использовать напрямую.

В противном случае данные придется извлекать из суперглобального массива $_GET:

$tireqty =$_GET [“tireqty”];

$oilqty =$_GET [“oilqty”];
$sparkqty =$_GET [“sparkqty”];

После рассмотрения обоих методов передачи данных в серверную программу возникает вопрос: «А нельзя ли обходиться только методом POST?» В основном так и делается. Но использовать метод GET удобно при отладке скриптов (тогда можно видеть имена и значения передаваемых данных) [7].

2.4. Варианты передачи данных серверной программе с разных элементов формы

На практике для передачи данных серверной программе с помощью формы используются не только рассмотренные выше поля типов text и password. Бывает потребность брать данные с полей типов: Radio, CheckBox, Select, Textarea.

Работу с полями этих типов рассмотрим на примере обработки анкетных данных (рис. 2.5).

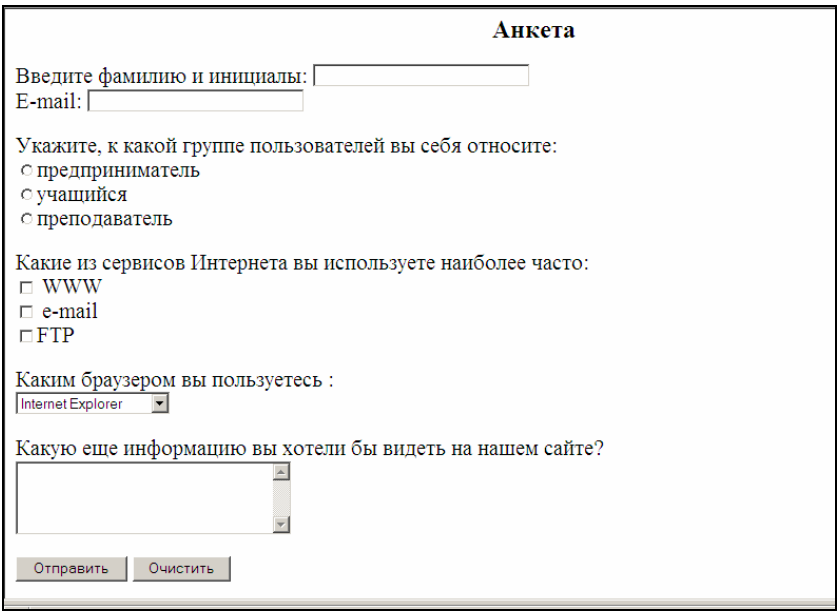


Рис. 2.5. Форма для заполнения анкеты

Такая Web-страница создается следующим HTML-кодом.

Листинг 2.3

```
<h3 align="center">Анкета </h3>
<form action="obrabotka.php" method="post">
Введите фамилию и инициалы:
<input type="text" name="fio" size=30><br>
E-mail: <input type="text" name="email" size=30><p>
Укажите, к какой группе пользователей вы себя относите:<br>
<input type="radio" name="group"
value="предприниматель">предприниматель<br>
<input type="radio" name="group" value="студент">студент<br>
<input type="radio" name="group"
value="преподаватель">преподаватель<p>
Какие из сервисов Интернета вы используете наиболее часто:
<br>
<input type="checkbox" name="service[]" value="www"> WWW
<br>
<input type="checkbox" name="service[]" value="e-mail"> e-mail
<br>
<input type="checkbox" name="service[]" value="ftp">FTP<p>
Каким браузером вы пользуетесь :<br>
<select name="browser[]" multiple>
<option value="Internet Explorer" selected> Internet Explorer
<option value="Netscape Navigator"> Netscape Navigator
<option value="Opera"> Opera
<option value="Mozilla Firefox"> Mozilla Firefox
</select> <p>
Какую еще информацию вы хотели бы видеть на нашем сайте?
<br>
<textarea name="pojelanie" rows=4 cols=30></textarea><p>
<input type="submit" value="Отправить">
<input type="reset" value="Очистить">
```

Чтобы понять работу скрипта, обрабатывающего анкету, следует уяснить следующее.

Имена радиокнопок одинаковые (group), но кнопки отличаются значениями атрибута value.

Имена независимых переключателей (флажков) тоже одинаковые (service). Они тоже отличаются значениями атрибута value. Но в отличие от радиокнопок, в этом случае может быть выбрано любое количество флажков (от 0 до 3 в нашем примере). Поэтому значение атрибута name наших флажков задается в виде массива: name="service[]". А PHP-скрипт можно разработать такой, чтобы, обработав массив, можно было разобраться, что выбрал пользователь. Заметим также, что в материалах по языку HTML возможность задания имени группы флажков в виде массива обычно не упоминается.

Для раскрывающегося списка тоже имеется возможность выбора нескольких опций из списка. Для этого в нашем примере в теге select значение атрибута name задается в виде массива: <select name="browsers[]">.

Надо заметить, что в обоих случаях, если значение атрибута name не задавать в виде массива, выбирать можно будет только одну кнопку (checkbox) и одну опцию (select). Обрабатывать эти поля надо не как массив, а как обычную переменную.

После обработки нашей анкеты пользователь должен получить результат в виде следующей Web-страницы.

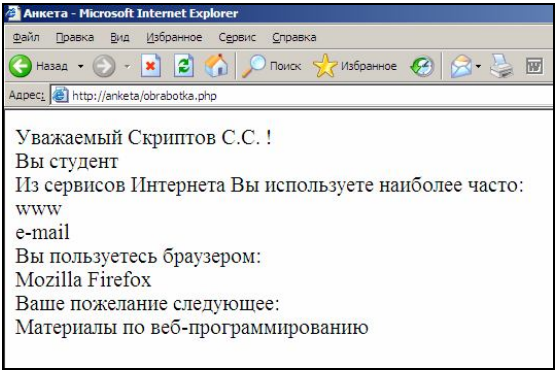


Рис. 2.6. Результат обработки анкеты

Такую обработку анкеты может выполнить следующий скрипт.

```
<body>
<?
$fio=$_POST["'fio'];
$email=$_POST["email"];
```

```
//if (isset($_POST['group']))
$selected_radio = $_POST['group'];

echo "Уважаемый $fio !<br>";
echo "Вы ". $selected_radio." <br>";

echo "Из сервисов Интернета Вы используете наиболее часто:
<br>";
foreach ($_POST["service"]
as $index=>$value)
 {echo "$value<br>";}

echo "Вы пользуетесь браузером: <br>";
foreach ($_POST["browser"]
as $index=>$value)
 {echo "$value<br>";}

$pojelanie = $_POST["pojelanie"];
echo "Ваше пожелание следующее:<br>";
echo $pojelanie;
?>
</body>
</html>
```

Разбор скрипта начнем с 6-й строки: $selected_radio = $_POST['group'].

Так как выбранной может быть только одна радиокнопка, при обработке группы из трех радиокнопок в суперглобальный массив попадает значение атрибута value той радиокнопки, которая была выбрана пользователем. В нашем примере это радиокнопка со значением “студент”. Переменной $selected_radio присваивается именно это значение.

Строки 9—12 обрабатывают группу из трех флажков, соответствующих трем сервисам Интернета.

```
echo "Из сервисов Интернета Вы используете наиболее часто:
<br>";
foreach ($_POST["service"]
as $index=>$value)
 {echo "$value<br>";}
```

В отличие от радиокнопок, в этом случае может быть выбрано любое количество флажков, и их значения помещаются в суперглобальный массив $_POST["service"]. Для выборки значений из этого массива используется функция foreach. Она по очереди просматривает элементы массива, а оператор echo выводит значения элементов на экран.

Строки 13—16 обрабатывают раскрывающееся меню, содержащее 4 значения.

```
echo "Вы пользуетесь браузером: <br>";
foreach ($_POST["browser"]
as $index=>$value)
 {echo "$value<br>";}
```

Хотя в нашей анкете элемент формы select позволяет одновременно выбрать только одну опцию из списка, есть возможность превратить его в элемент с множественным выбором. Для этого в тег select надо было бы добавить атрибут multiple таким образом:

<select name="browser[]" multiple>.

В таком случае может быть выбрано любое количество флажков, и их значения помещаются в суперглобальный массив $_POST["service"]. Для выборки значений из этого массива, как и в предыдущем фрагменте, используется функция foreach. Она по очереди просматривает элементы массива, а оператор echo выводит значения элементов на экран.

Строки 17—19 обрабатывают многострочное текстовое поле с именем pojelanie.

```
$pojelanie = $_POST["pojelanie"];
echo "Ваше пожелание следующее:<br>";
echo $pojelanie;
```

Как видим, многострочное текстовое поле (textarea) обрабатывается точно так же, как и простое текстовое поле (text).

Теперь, после рассмотрения вариантов передачи данных со всех типов элементов ввода в форму, можно вернуться к рассмотрению структуры суперглобального массива $_POST. Для нашей анкеты после его заполнения этот массив будет содержать следующие элементы с их индексами:

- $_POST["fio"] = Скриптов С.С;
- $_POST["email"] =scr12@ mail.ru;
- $_POST["group"] = студент;

- $_POST["service"[0]] = WWW;
- $_POST["service"[1]] = email;
- $_POST["browser"[0]] = Mozilla Firefox;
- $_POST["pojelanie"] = Материалы по Web-программированию.

Еще раз обратим внимание на то, что два элемента суперглобального массива, а именно $_POST["service"[]] и $_POST["browsers"[0]] сами являются массивами. Причем размерность их зависит от того, сколько значений выбрал пользователь.

2.5. Проверка данных, введенных в форму

При передаче данных из формы в серверную программу необходимо бывает организовать контроль:

- полноты введенных данных (все ли поля заполнены);
- корректности введенных данных (например, по типу — буквы или числа);
- корректности введенных данных (например, вхождение значений данных в заданный диапазон).

С этой целью рассмотрим еще один пример.

Составим серверное Web-приложение, выполняющее следующие функции.

1. Вывод формы с запросом на ввод сторон треугольника.
2. После завершения ввода и нажатия кнопки типа Submit проверка на полноту ввода.
3. Проверка того, что в поля ввода введены только числа.
4. Проверка на корректность введенных значений сторон треугольника (должны соблюдаться известные из геометрии соотношения между длинами сторон треугольника).
5. Обеспечение в любом из предыдущих трех случаев неправильного ввода повторного ввода данных в форму.
6. Вычисление площади треугольника по формуле Герона.

Форма для решения задачи может иметь следующий вид (рис. 2.7).

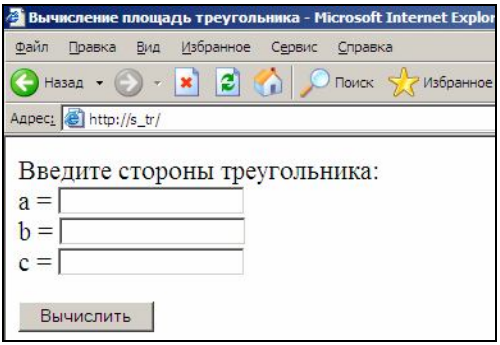


Рис. 2.7. Форма для ввода исходных данных для решения задачи

Для ее реализации потребуется такой HTML-код:

Листинг 2.4

```
<html><title>Вычисление площади треугольника</title>
<body>
<form action="obrabotka.php" method="POST">
Введите стороны треугольника:
<br>a = <input type="text" name="a">
<br>b = <input type="text" name="b">
<br>c = <input type="text" name="c">
<p><input type="submit" value="Вычислить">
</form></body></html>
```

PHP-скрипт для решения нашей задачи будет такой:

Листинг 2.5

```
<html><title>Вычисление площади треугольника</title><body>
<?php
$a=$_POST["a"];
$b=$_POST["b"];
$c=$_POST["c"];
if(!is_numeric($a)||!is_numeric($b)||!is_numeric($c))
{echo "Ошибка! Введены не числа или заполнены не все поля. ";
echo"<a href='index.htm'>На повторный ввод</a>";
exit();}
```

```
if($a+$b<=$c||$a+$c<=$b||$b+$b<=$a)
{echo "Ошибка! Эти значения не могут быть сторонами треугольника. ";
echo"<a href='index.htm'>На повторный ввод</a>";
exit();}

$pp=($a+$b+$c)/2;
$s_tr=sqrt($pp*($pp-$a)*($pp-$b)*($pp-$c));
echo "Площадь треугольника = ".$s_tr;
?>
</body></html>
```

Разбор скрипта следует начать с 6-й строки. Она содержит условие проверки ввода числовых значений во все поля формы. Функция is_numeric($a) возвращает значение true только в том случае, если переменная $a содержит данные, и эти данные являются числом. В остальных случаях (поле не заполнено или в него введены не цифры) результатом выполнения этой функции будет значение false.

Если условие в строке 6 будет выполнено, значит пользователь ввел неверные данные или что-то не ввел. Об этом ему сообщит строка 7: «Ошибка! Введены не числа или заполнены не все поля».

Строка 8 выведет гиперссылку На повторный ввод. Щелчок по этой ссылке снова вызовет на экран форму для ввода исходных данных.

Вызов функции exit() в строках 9 и 13 нужен для завершения выполнения скрипта после вывода формы для повторного ввода данных.

Строка 10 содержит условие проверки корректности введенных длин сторон треугольника. При некорректности введенных значений строка 11 выводит сообщение: «Ошибка! Эти значения не могут быть сторонами треугольника».

Строка 12 выведет гиперссылку На повторный ввод. Щелчок по этой ссылке снова вызовет на экран форму для ввода исходных данных.

Строки 14—16 выполняются, когда во все поля формы будут введены правильные значения. Здесь вычисляется площадь треугольника по форме Герона и выводится результат.

2.6. Разработка и отладка локального сайта с помощью локального сервера

Написать код формы и код PHP-скрипта можно на локальном компьютере в редакторе Блокнот. Если бы сайт был статический, его можно было бы отладить на локальном компьютере с помощью браузера. Но в случае интерактивного сайта его сначала надо установить (опубликовать) на сервере какого-нибудь провайдера (получить хостинг, бесплатный или платный). Получить хостинг в наше время тоже не проблема. Проблемой становится отладка сайта. Как мы уже знаем по собственному опыту, чтобы сайт заработал, придется устранить немало ошибок.

В случае дистанционного (настоящего) хостинга это означает, что после устранения каждой ошибки надо будет исправленный файл закачивать на удаленный сервер, а это — деньги, время, лишняя работа.

Выходом из этой ситуации является установка виртуального сервера на своей локальной машине. Установив такой сервер, интерактивный сайт можно создать и отладить на своей машине. Далее отлаженный сайт можно выставить в Интернет.

2.6.1. Система Денвер

Есть прекрасные варианты решения этой проблемы. Наиболее известное решение — это использование «джентльменского набора» сайтостроителя — бесплатного пакета Денвер.

Последняя его версия — Денвер 3 (Денвер — 3-2010-11-07) в своем составе содержит:

- web-сервер в лице сервера Apache / 2.2.4 (Win32) mod_ssl/2.2.4 OpenSSL/0.9.8k PHP 5.2.12;
- интерпретатор PHP/ 5.2.12;
- MySQL server 5.1.40-community;
- MySQL client version: 3.2.3;
- графический интерфейс СУБД MySQL — программа phpMyAdmin 3.2.3.

Этот пакет отработан очень хорошо. Установка происходит легко, быстро и без ошибок.

2.6.2. Установка Web-сервера на локальном компьютере

1. Запускаем инсталлятор Base.exe.

2. В процессе инсталляции Денвера нам придется ответить на задаваемые вопросы следующим образом:

- В качестве буквы будущего виртуального диска примем предложенную букву Z;
- Выберем предложенный режим 1;
- Принимаем предложение Создать ярлыки на рабочем столе (Start servers, ReStart servers, Stop servers).

3. После запуска серверов в правой части панели управления появится эмблема Web-сервера Apache — перышко.

2.6.3. Структура папок локального Web-сервера

При установке Denver 3 в папку Webservers3 создаются следующие папки (рис. 2.8—2.14).

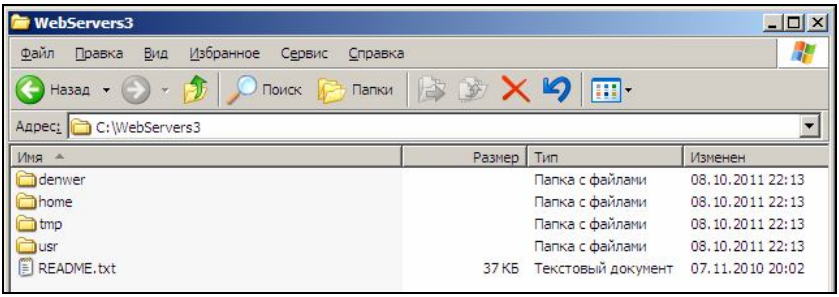


Рис. 2.8. Содержание папки WebServers3

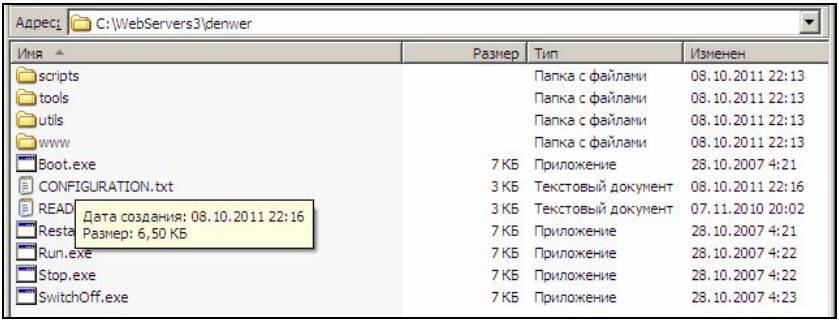


Рис. 2.9. Содержание папки WebServers3\denver

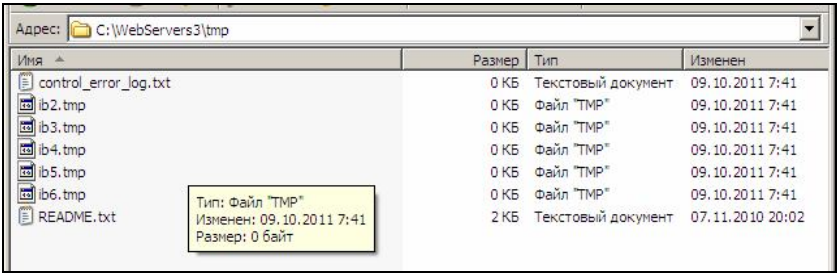


Рис. 2.10. Содержание папки WebServers3\tmp

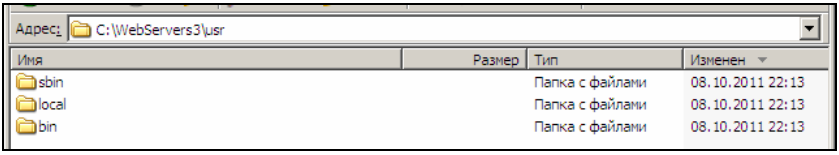


Рис. 2.11. Содержание папки WebServers3\usr

В папке usr\local при дальнейшем погружении можно увидеть:

- Сервер Apache
- СУБД MySQL
- Интерпретатор PHP

Довольно подробное описание системы Денвер 3 дано в файле Readme.txt в папке websrevers3.

HTML-документы должны находиться в директориях /home/<имя_хоста>/www. Содержимое директории home показано на следующем рисунке.

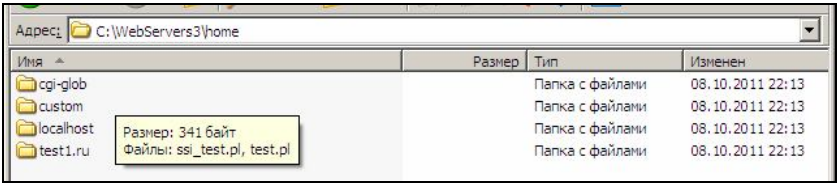


Рис. 2.12. Содержание папки WebServers3\home

При инсталляции Денвера сконфигурированы 3 виртуальных хоста: custom, localhost и test1.ru (cgi_glob не сайт, там хранятся cgi-скрипты).

В директории третьего хоста (/home/test1.ru/www) расположена тестовая страница, содержащая некоторую информацию о настройках PHP, переменных окружения и т.д., выводимая функцией PHP phpinfo().

Наш домен под именем **autoparts** тоже будет создан здесь.

Для создания нового сайта в каталоге **autoparts** надо создать папку WWW. В нее помещаются файлы и папки сайта, в частности, обязательный запускной файл index.htm. Именно он выдается браузеру после указания в адресной строке адреса сайта: http://autoparts.

Требуемое содержимое папки **autoparts** представлено на следующем рисунке.

Рис. 2.13. Содержание папки WebServers3\home

Теперь, если, запустив браузер, в адресной строке набрать адрес http://autoparts и нажать Enter, откроется наша форма заказа из файла index.htm.

Здесь уместно ответить на вопрос, почему, получив адрес сервера, браузер не пытается обратиться к Интернету, а сразу же находит сайт на локальном компьютере. Это обеспечивается благодаря следующему механизму.

Получив адрес сайта, браузер сразу же обращается к файлу c:\windows\system32\drivers\etc\hosts. Там, как видим из следующего рисунка, имеется перечень доменов компьютера.

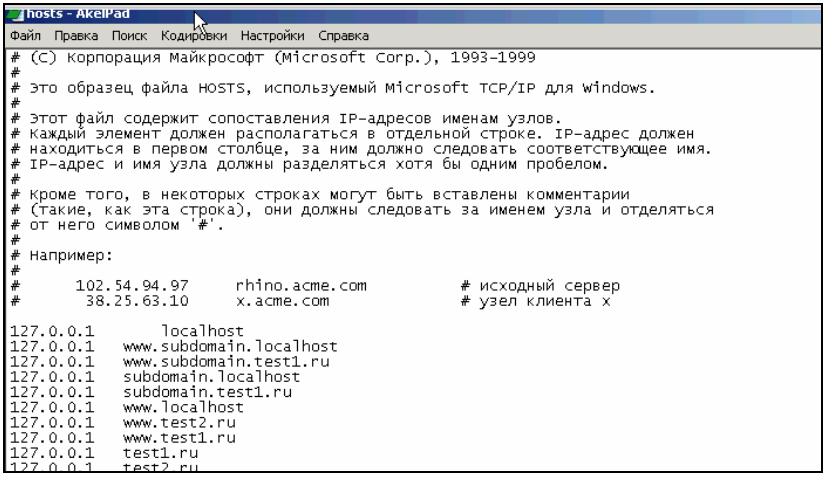


Рис. 2.14. Перечень доменов компьютера

Поддерживаются также виртуальные хосты с доменными именами третьего и выше уровней. Примеры того, как Apache ищет директории документов хостов в этом случае, приведены в следующей таблице.

Таблица 2.2.

Доменное имя	Директория документов
abcd.test1.ru	/home/test1.ru/abcd
ab.cd.test1.ru	/home/test1.ru/ab.cd
test.localhost	/home/localhost/test
ab.cd.localhost	/home/localhost/ab.cd

2.6.4. Порядок создания интерактивного сайта

1. Создать папку **autoparts** по адресу C:\WebServers\home. В этой папке создать папку **www.**

2. Набрать в редакторе Блокнот код Web-страницы с формой заказа автозапчастей. Сохранить страницу в файле index.htm и поместить ее в папку **www**.

3. Набрать код PHP-программы для обработки заказа из формы. Сохранить программу в файле obrabotka.php и поместить ее тоже в папку **www**.

4. Запустить локальный сервер.

5. Набрать в адресной строке браузера адрес: http://autoparts, нажать Enter. Откроется наша форма заказа из файла index.htm.

6. Заполнить заказ, отправить его. Получить результат от сервера. Изучить результат обработки.

7. В случае ошибок отладить форму и/или PHP-программу.

Контрольные вопросы

1. Можно ли отлаживать PHP-скрипты на клиентской машине?
2. Укажите адрес создаваемого нами на виртуальном сервере сайта.
3. Какие файлы будут на нашем сайте и каково их назначение?
4. Воспроизведите форму для ввода заказа на автозапчасти.
5. Опишите синтаксис оператора вывода в PHP.
6. Как определяются константы в PHP?
7. Опишите синтаксис условного оператора в PHP.
8. Опишите метод POST для передачи информации PHP-программе.
9. Опишите метод GET для передачи информации PHP-программе.
10. Объясните назначение системы Денвер.
11. Объясните назначение файла c:\windows\system32\drivers\etc\hosts.
12. Воспроизведите порядок создания сайта autoparts.

3. РАБОТА С СЕРВЕРОМ БАЗ ДАННЫХ MySQL

3.1. Общая характеристика MySQL

Задача длительного хранения информации и оперативной выборки ее для обслуживания запросов пользователей в сети Интернет решается постоянно. В настоящее время для этого чаще всего используется СУБД MySQL. Ее называют также сервером баз данных. Она не предназначена для работы с большими объемами информации, но ее применение отлично подходит для Web-проектов. СУБД MySQL отличается хорошей скоростью работы, надежностью и гибкостью. Поддержка сервера MySQL включена в язык серверного Web-программирования PHP. Кроме всего этого СУБД MySQL является бесплатно распространяемым программным продуктом.

Для работы с этой СУБД при решении наших задач в основном будет достаточно знаний, полученных при изучении СУБД MS ACCESS. При решении простых задач можно обойтись графическим интерфейсом, реализуемым программой phpMyAdmin. Так же как интерфейс MS ACCESS, он позволяет выполнять многие операции с данными из БД, не пользуясь операторами языка SQL. Однако более сложные задачи невозможно решать без использования языка SQL.

Для иллюстрации отличий MySQL от ACCESS, хотя бы поверхностной, приведем наиболее часто используемые типы данных MySQL. В скобках указаны эквивалентные или аналогичные типы данных MS ACCESS.

Таблица 3.1

Тип	Описание
INT (числовой)	Данные целого типа. Диапазон значений: -2 147 483 648 …+2 147 483 647
VARCHAR (N) (текстовый)	Строка переменной длины, N — допустимое число символов (максимум 255)
TEXT (memo)	Текст, максимальный размер — 64 Кб
DATE (дата/время)	Дата (формат ГГГГ-ММ-ДД)
TIME (дата/время)	Время (формат ЧЧ:ММ:СС)
BLOB (OLE)	Бинарные данные, например, изображения. Максимальный размер поля — 64 Кб

3.2. Описание задания по созданию базы данных в MySQL

Для изучения технологии работы с базой данных ее надо создать. Причем, надо рассмотреть 2 способа создания БД:

- с помощью графического интерфейса СУБД MySQL — программы phpMyAdmin 3.2.3 создадим БД **Firstdb** в ручном режиме;
- из PHP-программы, которую мы напишем сами, создадим БД **Seconddb.**

В обеих БД создадим по одной таблице с именем **Staff** (Сотрудники). Для работы с каждым вариантом БД напишем PHP-программы и будем выполнять операции с данными.

Создание БД, как обычно, будет включать следующие этапы:

- создание БД;
- создание таблицы;
- заполнение БД.

Далее для работы с БД надо будет «создать пользователя». Это означает ввод имени пользователя, имени хоста, пароля.

3.3. Методика создания MySQL — БД с помощью программы phpMyAdmin 3.2.3

Пусть структура и начальное содержимое БД будет такое.

Таблица 3.2

id	name	lastname	dol
1	Алексей	Алексеев	директор
2	Иван	Иванов	бухгалтер
3	Александр	Александров	программист

Типы полей подойдут следующие.

Таблица 3.3

Название поля	Тип данных	Длина поля
id	Int	3
name	varchar	30
lastname	varchar	30
dol	varchar	30

Таблица содержит 4 поля:

- поле первичного ключа с именем id, значение которого будет автоматически увеличиваться при добавлении новых записей;
- поле name, содержащее имя пользователя, где будет храниться строка переменной длины типа VARCHAR;
- поле lastname, содержащее фамилию пользователя;
- поле dol, содержащее должность пользователя.

PHPmyAdmin, как было сказано выше, — это графический интерфейс для работы с СУБД MySQL. Для запуска этой программы надо запустить локальный Web-сервер и в адресной строке браузера набрать адрес: http://localhost/tools/phpmyadmin. Появляется начальная страница. Дальнейшие действия следующие.

1. В поле **Create new database** введем имя базы данных **Firstdb** и нажмем кнопку **Create**. В следующем окне увидим сообщение «**Database firstdb has been created**».

2. Введем имя таблицы **Staff,** количество полей 4 и нажимаем кнопку **Go.**

3. Вводим названия полей, типы полей и их длины. Свойство полей «not null» вводится по умолчанию. Для поля **id** выберем свойство **auto_increment (**в поле **A_I**) и укажем, что оно является первичным ключом (выбрав **Primary** в поле **Index**). Нажмем кнопку **Save**.

4. В следующем окне увидим сообщение «**Table 'firstdb'.'staff' has been created**».

5. В поле слева щелкаем ссылку **staff.** В следующем окне нажимаем кнопку **Insert.** Появляется форма для ввода полей первой и второй записей таблицы.

6. После набора полей первой и второй записей (поле id не заполняем, оно заполняется автоматически) выбираем опцию "**Insert another new row**" и нажимаем кнопку **Go.**

7. В следующем окне увидим сообщение «**2 row(s) inserted**» и форму для ввода следующих двух записей.

8. После ввода последней третьей записи выбираем опцию "**Go back to previous page**", нажмем кнопку **Go**. В следующем окне, нажав кнопку **Browse,** увидим заполненную таблицу. Здесь таблицу можно редактировать.

9. До того, как начать соединяться с БД из PHP-скрипта, необходимо создать пользователя, который имеет право работать с

данной БД. Для этого в левой области окна щелкаем гиперссылку **phpMyAdmin**. В открывшемся окне нажимаем кнопку **Priveleges.** Переходим в новое окно, где должна быть информация о пользователях. Щелкаем ссылку **Add a new User.** В открывшемся окне указываем имя пользователя (User name), например, **student**, имя хоста **localhost** и пароль **12345**. Повторяем пароль, набрав его в поле **Re-type**. Даем пользователю все привилегии. Для этого в области **Global priveleges** щелкаем гиперссылку **Check All**. Нажимаем кнопку **Go**. В следующем окне увидим сообщение «**You have added a new user**». Далее этот пользователь сможет соединиться с БД firstbd и работать с ней.

10. В результате выполнения предыдущих 9 пунктов задания в каталоге C:\webservers\usr\local\mysql-5.1\data\firstdb\ будет создана таблица staff в виде четырех файлов: staff.MYI; staff.MYD; staff.frm; db.opt.

3.4. Основные SQL-операторы для работы с БД

Для манипулирования данными в реляционных базах данных, как правило, используется структурированный язык запросов SQL (Structured Query Language). При создании Web-проектов чаще всего используются SQL-операторы для выполнения следующих операций.

1. Создание базы данных.
2. Создание таблицы.
3. Выборка записей.
4. Вставка записей.
5. Обновление записей.
6. Удаление записей.

Рассмотрим эти SQL-операторы применительно к нашей базе данных и таблице staff.

3.4.1. Создание базы данных

Для этого служит оператор.

CREATE DATABASE seconddb;

Здесь seconddb — имя нашей второй базы данных, которую мы будем создавать и заполнять из PHP-программы.

3.4.2. Создание таблицы

Для создания нашей таблицы **staff** понадобится оператор CREATE TABLE.

```
CREATE TABLE staff
(
id              int (3)          autoincrement,
name            varchar(30)      not null,
lastname        varchar(30)      not null,
dol             varchar(30)      not null,
primary key     (id)
);
```

3.4.3. Выборка записей

Для выборки записей используется оператор SELECT. Это самый часто используемый оператор. У него самый сложный синтаксис. Несколько упрощенный синтаксис этого оператора следующий:

SELECT <список полей через запятую>

FROM <имя таблицы>

[WHERE <условие выбора строк>]

[GROUP BY <имя поля для группировки одинаковых значений>]

[HAVING <условие выбора группы строк для выполнения групповых операций>]

[ORDER BY <поле, по значению которого надо сортировать записи> [ASC|DESC]]

[LIMIT смещение, количество] (позволяет задать номер начальной записи, подлежащей выборке, и количество выбираемых из БД записей).

Например, для выборки из нашей таблицы полей «Фамилия» и «Должность» из второй записи потребуется следующий SQL-оператор:

SELECT lastname, dol FROM staff WHERE id=2.

3.4.4. Вставка записей

Синтаксис этого оператора SQL следующий:

INSERT INTO Имя_таблицы [(Список полей)] VALUES (Список значений);

Для добавления в нашу базу данных водителя Андреева оператор должен иметь вид:

INSERT INTO staff (id, name, lastname, dol) VALUES (0, “Андрей”, “Андреев”,”Водитель”);

Запись вставляется (добавляется) под номером, на единицу больше номера последней записи. Причем номера удаленных записей сохраняются и учитываются при добавлении новых записей.

3.4.5. Обновление записей

Обновление записи таблицы означает изменение отдельных полей записи или замену одной записи другой. Измененная запись имеет тот же номер, что и до изменения.

Синтаксис этого оператора SQL следующий:

UPDATE Имя_таблицы SET поле1=’значение’, поле2=’значение’ WHERE условие.

Для замены данных водителя Андреева данными Егорова оператор должен иметь вид:

UPDATE staff SET name=’Егор’, lastname=’Егоров’ WHERE id=4;

3.4.6. Удаление записей

Синтаксис этого оператора SQL следующий:

DELETE FROM Имя_таблицы WHERE условие.

Для удаления из таблицы данных программиста Александрова оператор должен иметь вид:

DELETE FROM staff WHERE id=’3’;

3.5. Функции PHP для работы с СУБД MySQL. Программирование взаимодействия с БД

Система MySQL представляет собой сервер (СУБД), к которому может подключиться PHP-сценарий. Подключившись к серверу, можно выбрать базу данных, с которой надо работать, если у вас есть на это полномочия. Далее можно:

- выводить данные из БД;
- редактировать БД (обновлять);
- вставлять записи;
- удалять записи.

Рассмотрим в требуемой логической последовательности функции PHP для работы с СУБД MySQL:

- подключение к серверу баз данных;
- выбор БД;
- выполнение SQL-запроса;
- обработка результатов запроса;
- завершение MySQL-соединения.

3.5.1. Подключение к серверу баз данных

Перед тем как начинать работать со своей базой данных, нужно подключиться к серверу. Для этого в языке PHP есть функция mysql_connect().

$link = mysql_connect ("localhost", "student", "12345").

Данная функция имеет три аргумента, в которых указывается:

- имя компьютера;
- имя пользователя;
- пароль.

Если опустить эти необязательные аргументы, то функция будет считать, что требуется компьютер localhost, а имя пользователя и его пароль не требуются, т.е. не установлены в таблице mysql-user. Такое подключение можно использовать только для проверки сервера.

Функция mysql_connect () возвращает идентификатор подключения, если все прошло успешно, или false в противном случае. Этот идентификатор можно сохранить в переменной и в дальнейшем пользоваться им для работы с сервером базы данных.

Рассматриваемая функция очень чувствительна к параметрам. В случае ошибки в коде или при сбое сервера базы данных возникает нештатная ситуация. Поэтому все операции с базами данных выполняются вместе с функцией die (). Использование этой функции с контролем ошибки будет иметь следующий вид:

```
$link = mysql_connect ( "localhost", "student", "12345" );
if ( !$link) die ("Couldn't connect to MySQL" ).
```

3.5.2. Выбор базы данных

После того как установлено соединение с сервером MySQL, нужно выбрать базу данных, с которой собираемся работать. Для этого существует функция mysql_select_db().

В следующем примере мы выбираем свою базу данных с именем firstdb:

```
mysql_select_db (“firstdb”, $link).
```

Этой функции нужно передать:

- имя базы данных;
- второй необязательный аргумент — идентификатор подключения к серверу.

Если этот второй аргумент опустить, то по умолчанию будет использован идентификатор последнего полученного подключения. Функция возвращает true, если указанная база данных существует и доступ к ней возможен.

Использование этой функции с контролем ошибки будет иметь следующий вид:

```
mysql_select_db (“firstdb”, $link) or die ("Couldn't open firstdb”).
```

Соединившись с БД и выбрав таблицу из нее, можно манипулировать данными таблицы:

- выбирать данные и выводить их на экран;
- вставлять (добавлять) записи в БД;
- изменять данные;
- удалять записи.

3.5.3. Выборка данных из таблицы

Для выборки данных из таблицы служит функция mysql_query ().

В качестве параметров этой функции надо передать SQL-запрос SELECT с соответствующими параметрами.

PHP-оператор для выборки всей таблицы будет такой:

$result=mysql_query("SELECT * FROM staff ", $link).

При успешном выполнении запроса функция mysql_query () возвращает идентификатор результата запроса, и данный идентификатор можно передавать другим функциям для обработки результата. В этой переменной будет храниться вся таблица в виде матрицы.

Для того, чтобы извлекать из матрицы отдельные записи, служит функция mysql_fetch_array() (fetch — «достать»).

В качестве параметра этой функции надо передать имя $result. PHP-оператор для выборки первой записи из таблицы будет такой:

$myrow=mysql_fetch_array($result).

Первая запись будет выбрана в переменную $myrow в виде ассоциативного массива или в виде false при отсутствии результата. Ассоциативный массив, как мы рассмотрели в п. 2.3.1, — это одномерный массив, в котором вместо числовых индексов элементов используются словесные индексы. Первая строка нашей таблицы при обозначении ее как ассоциативного массива будет иметь вид:

$myrow = array (“id” =>”1”, “name” =>”Алексей”, “lastname” =>”Алексеев”, “dol”=> “директор”).

Далее, рассматривая в логической последовательности операции извлечения данных из нашей таблицы, мы будем их нумеровать. В последующем эти нумерованные действия будут присутствовать в задании для лабораторной работы.

Примеры извлечения данных из БД

1. Для того, чтобы вывести на экран значение поля name первой записи, можно использовать оператор:

Echo $myrow[‘name’].

2. PHP-оператор для выборки не всей таблицы, а только определенных полей всех записей, будет такой:

$result=mysql_query("SELECT lastname, dol FROM staff ", $link).

3. PHP-оператор для выборки не всех записей таблицы, а только определенной записи (например, второй), будет:

$result=mysql_query("SELECT * FROM staff WHERE id=2", $link).

Следовательно, следующие операторы:

$myrow=mysql_fetch_array($result);

Echo $myrow[name]

выведут на экран имя «Иван».

4. А PHP-оператор:

$result=mysql_query("SELECT lastname, dol FROM staff WHERE id='2'", $link)

выберет из таблицы только поля «Фамилия» и «Должность» из второй записи.

5. Для вывода данных всех сотрудников в виде:

Сотрудник № 1
Алексей
Алексеев
директор

Сотрудник № 2
Иван
Иванов
бухгалтер

Сотрудник № 3
Александр
Александров
программист

потребуется следующий циклический PHP-фрагмент.

Листинг 3.1

```
$result=mysql_query("SELECT * FROM staff ", $link);
$myrow=mysql_fetch_array($result); /заносит в $myrow первую запись
do
{
echo "Сотрудник № — " . $myrow['id']. "<BR>";
echo $myrow['name']. "<BR>";
```

echo $myrow[‘lastname’]. “
”;
echo $myrow[‘dol’]. “

”;
}
while ($myrow=mysql_fetch_array($result)); /*заносит в $myrow следующую запись до тех пор, пока не будут выведены все записи из таблицы*/.

Использование сложных условий в SQL-операторе SELECT (and, or)

6. $result=mysql_query("SELECT * FROM staff WHERE id=’2’ and name=’Александр’", $link); //Выборка данных сотрудника с номером 2 и именем Александр

7. $result=mysql_query("SELECT * FROM staff WHERE id=’2’ or name=’Александр’", $link); //Выборка данных сотрудника, номер которого 2 или его имя Александр.

Сортировка и ограничение количества выводимых записей

8. $result=mysql_query("SELECT * FROM staff ORDER BY name", $link); //Выборка данных всех сотрудников с сортировкой их по именам по алфавиту

9. $result=mysql_query("SELECT * FROM staff ORDER BY name DESC", $link); // Выборка данных всех сотрудников с сортировкой их по именам в порядке, обратном алфавитному.

10. $result=mysql_query("SELECT * FROM staff ORDER BY name LIMIT 2", $link); // Выборка данных первых двух сотрудников с сортировкой их по именам по алфавиту.

3.5.4. Отображение данных таблицы БД в виде таблицы на Web-странице

И, наконец, для вывода всех записей нашей таблицы в виде HTML-таблицы можно использовать следующий PHP- сценарий.

Листинг 3.2

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transi-
tional//EN">
<html>
<head>
<title>Работа с БД </title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1251">
</head>
<body>
<?
$link=mysql_connect("localhost",student,12345);
mysql_select_db("firstbd",$link);

$result=mysql_query("SELECT * FROM staff ", $link);

echo "<table border=1 width=100%>
<TR>
<TD>№ п/п</TD>
<TD>Имя</TD>
<TD>Фамилия</TD>
<TD>Должность</TD>
</TR>";
while ($myrow=mysql_fetch_array($result)){
echo "<TR>
<TD>$myrow[id]</TD>
<TD>$myrow[name]</TD>
<TD>$myrow[lastname]</TD>
<TD>$myrow[dol]</TD>
</TR>";
}
echo "</TABLE>";

?>
</body>
</html>
```

3.5.5. Вставка записи в таблицу

Получив доступ к базе данных, мы можем добавлять информацию в ее таблицы. Для добавления данных в таблицу нужно сконструировать и выполнить запрос SQL. Для этого используется функция mysql_query():

$result=mysql_query (“INSERT INTO staff (id, name, lastname, dol) VALUES (0, ‘Андрей’, ‘Андреев’,’Водитель’)”);

Функция возвращает true, в случае успешного выполнения запроса, и false, если в запросе содержится ошибка или если вы не имеет права на выполнение такого запроса. Успешное выполнение запроса не обязательно подразумевает изменение данных в таблице.

Проверить результат вставки записи в таблицу можно следующим образом.

Листинг 3.3

```
If ($result==’true’)
{
echo “Информация в базу добавлена успешно!”;
}
else
{
echo “Информация в базу не добавлена!”;
}
```

Добавлять записи в БД описанным способом можно, но обычно для этого используют форму.

3.5.6. Добавление записей в базу данных через форму

Форма для ввода данных в нашу БД может иметь следующий вид.

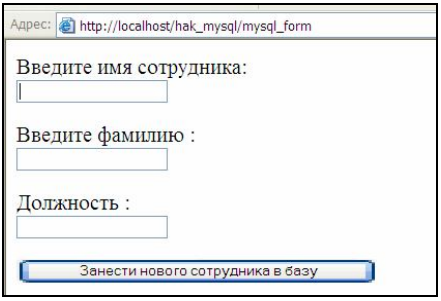


Рис. 3.1. Форма для ввода данных в таблицу staff БД firstdb

Для создания формы надо набрать следующий код страницы.

Листинг 3.4

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transi-
tional//EN">
<html>
<head>
<title>Добавление сотрудника </title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1251">
</head>
<body>
< form action=”mysql_insert.php” method=”post” name=form>
<p> Введите имя сотрудника: <br><input name=”name”
type=”text” size=”20” maxlength=”40”></p>
<p> Введите фамилию: <br><input name=”lastname”
type=”text” size=”20” maxlength=”40”></p>
<p> Должность: <br><input name=”dol” type=”text” size=”20”
maxlength=”40”></p>
<p><input type=”submit” value=”Занести нового сотрудника в
базу”</p>
</form>
</body>
</html>
```

Код обработчика для формы будет следующий.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transi-
tional//EN">
<html>
<head>
<title>Ввод в таблицу через форму </title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1251">
</head>
<body>
<?
if (isset($_POST [‘name’]))
{
$name=$_POST [‘name’]; /Защита от хостингов, не поддержи-
вающих прямую передачу переменных из формы на сервер
}
if (isset($_POST [‘lastname’]))
{
$lastname=$_POST [‘lastname’];
}
if (isset($_POST [‘dol’]))
{
$dol=$_POST [‘dol’];
}
$link=mysql_connect("localhost",’student’,12345);
mysql_select_db("firstdb",$link);

$result=mysql_query (“INSERT INTO staff (id, name, lastname,
dol) VALUES (0, ‘$name’, ‘$lastname’,’$dol’)”);
If ($result==’true’)
{
echo “Информация в базу добавлена успешно!”;
}
else
{
echo “Информация в базу не добавлена!”;
}
```

```
?>

</body>
</html>
```

При выполнении лабораторной работы этот код нужно будет сохранить в файле под именем mysql_insert1.php.

3.5.7. Обновление записей

Для обновления данных в таблице нужно сконструировать и выполнить запрос SQL. Для выполнения запроса используется функция mysql_query().

Заменим программиста Александрова на Егора Егорова:

```
$result=mysql_query (“UPDATE staff SET name=’Егор’, last-name=’Егоров’ WHERE id=’3’”).
```

Функция возвращает значение true в случае успешного выполнения запроса, и false, если в запросе содержится ошибка или если вы не имеет права на выполнение такого запроса. Проверить результат обновления записи в таблице можно следующим образом.

Листинг 3.6

```
If ($result==’true’)
{
echo “Информация в базе обновлена успешно!”;
}
else
{
echo “Информация в базе не обновлена!”;
}
```

3.5.8. Удаление записей

Для удаления записи из таблицы нужно сконструировать и выполнить запрос SQL. Для выполнения запроса, как обычно, используется функция mysql_query().

Для удаления из таблицы данных программиста Егорова PHP-оператор должен иметь вид:

$result=mysql_query (“ DELETE FROM staff WHERE id=’3’”).

Функция возвращает положительное число, в случае успешного выполнения запроса, и false, если в запросе содержится ошибка или если вы не имеет права на выполнение такого запроса. Успешное выполнение запроса не обязательно подразумевает изменение данных в таблице.

Проверить результат удаления записи из таблицы можно следующим образом.

Листинг 3.7

```
If ($result==’true’)
{
echo “Запись из базы удалена!”;
}
else
{
echo “Запись из базы не удалена!”;
}
```

3.5.9. Завершение MySQL-соединения

PHP сам закрывает все открытые MySQL-соединения при завершении сценария. При необходимости закрыть соединение до завершения сценария можно вызвать функцию mysql_close():

mysql_close($link).

3.6. Методика создания MySQL-базы данных путем программирования в PHP

В предыдущих подразделах раздела 3 мы рассмотрели все средства языка запросов SQL и языка серверного программирования PHP, необходимые и достаточные для создания MySQL БД путем программирования в PHP. Сейчас мы выполним эту работу. Создадим базу данных seconddb, а в ней таблицу staff.

Структуру и содержимое БД staff оставим первоначальной.

Для выполнения этой работы необходимо запрограммировать в PHP выполнение следующих действий:

- Выполнение функции `mysql_connect` для соединения с сервером БД;
- Выполнение функции `mysql_query` для создания БД `seconddb`;
- Выполнение функции `mysql_select_db` для выбора БД `seconddb;`
- Выполнение функции `mysql_query` для создания структуры таблицы `staff;`
- Выполнение функции `mysql_query` для заполнения таблицы staff.

Далее приводим листинг скрипта, выполняющего все перечисленные действия:

Листинг 3.8

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Работа с БД </title>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1251">
</head>
<body>
<?
$db = mysql_connect ( "localhost", "student", "12345" );
if ( !$db) die ("Couldn't connect to MySQL" );
$result=mysql_query("CREATE DATABASE `seconddb`", $db);
if (!$result) echo "Ошибка базы данных. MySQL пишет:",
mysql_error();
mysql_select_db ("seconddb", $db) or die ("Couldn't open seconddb");
$result=mysql_query("CREATE TABLE IF NOT EXISTS `staff`
(
```

```
    `id` int(11) NOT NULL AUTO_INCREMENT,
    `name` varchar(30) NOT NULL,
    `lastname` varchar(30) NOT NULL,
    `dol` varchar(30) NOT NULL,
    PRIMARY KEY (`id`)
    )
    ENGINE=InnoDB DEFAULT CHARSET=utf8
AUTO_INCREMENT=7 ", $db);
    if (!$result) echo "Ошибка базы данных. MySQL пишет:",
mysql_error();
    $result=mysql_query("INSERT INTO `staff` (`id`, `name`, `last-
name`, `dol`)
    VALUES (1, 'Алексей', 'Алексеев', 'директор'),
    (2, 'Иван', 'Иванов', 'бухгалтер'),
    (3, 'Александр', 'Александров', 'програмист')", $db);
    if (!$result)
    {
     echo "Ошибка базы данных. MySQL пишет:", mysql_error();
    }
    else
    {
    echo "OK";
    }
    ?>
    </body>
    </html>
```

Такое задание надо будет выполнить в лабораторной работе № 3.

3.7. Экспорт/импорт данных в PhpMyAdmin

В ходе создания и отладки сайта на локальном сервере приходится неоднократно переносить базу данных с одного места на другое с разными целями. Можно выделить следующие варианты переноса:

1. Бывает необходимость переноса дампа (содержимого) БД в другую БД на этом же локальном сервере (например, из БД `firstdb` в БД `thirddb`).

2. Сохранение дампа таблицы из базы данных в виде резервной копии в отдельном файле с расчетом на последующее возвращение данных на сервер. Например, на тот случай, когда БД на сервере испортится, а восстановление вручную трудоемкое. Тогда БД можно восстановить путем экспорта из резервного файла.

3. Перенос дампа БД на другой компьютер (например, с компьютера на работе на домашний компьютер).

4. И, наконец, когда сайт будет готов к эксплуатации, его надо будет перенести на удаленный реальный сервер. Файлы при этом можно копировать с локального на удаленный сервер обычным способом, а вот содержимое БД импортируется не так, как другие файлы сайта.

Надо понимать, что для переноса БД понадобится по отдельности переносить каждую таблицу. Так что дамп БД — это совокупность дампов таблиц, имеющихся в БД.

В действиях по реализации четырех разновидностей переноса БД есть большая общая часть, и есть небольшие различия. Рассмотрим перечисленные варианты экспорта/импорта БД по порядку. Надо заметить, что в PhpMyAdmin процедуры экспорта/импорта достаточно простые, однозначные.

Вариант 1. Перенос содержимого БД «firstdb» в БД «thirddb» на локальном сервере

Шаг 1

Сперва нужно экспортировать базу данных с текущего местонахождения (из БД `firstdb`). В результате мы получим **SQL-запрос**, способный воссоздать нашу базу данных в другом месте. Для этого надо сделать следующее:

1. Зайти на главную страницу PHPMyAdmin, щелкнув на одноименной гиперссылке.

2. Создать базу данных `thirddb`.

3. Выбрать базу данных `firstdb`, щелкнув на соответствующей гиперссылке.

4. В верхнем меню выбрать пункт **Экспорт**. Появится следующее окно (рис. 3.2).

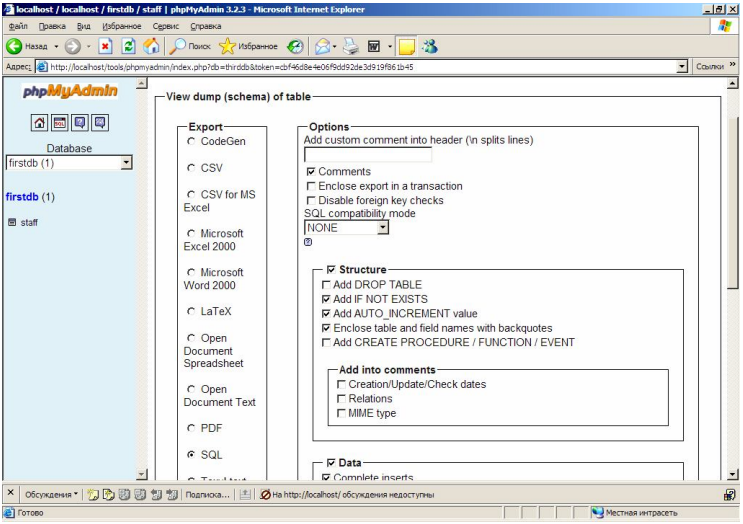


Рис. 3.2. Окно для запуска экспорта БД

5. На открывшейся странице поставим радио-переключатель на **SQL**, затем нажмем кнопку **Go.**

6. На следующей странице (рис. 3.3) нам надо выделить весь SQL-код, который будет сгенерирован, и скопировать его в буфер обмена.

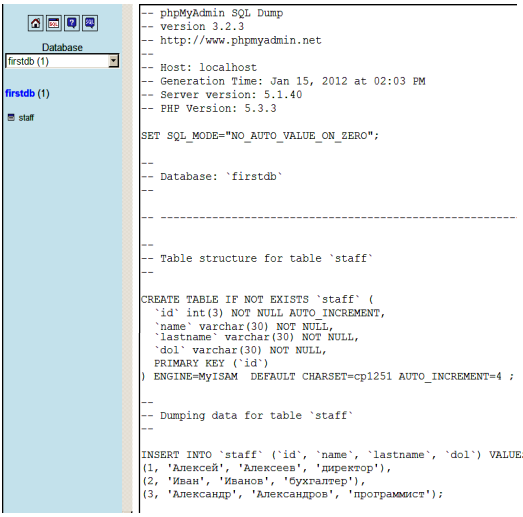


Рис. 3.3. SQL-код запроса для экспорта БД

Шаг 2

Вторым шагом является выполнение скопированного SQL-запроса в PhpMyAdmin. Для этого надо выполнить следующие действия:

1. Зайти на главную страницу PhpMyAdmin.
2. Выбрать базу данных `thirddb`, в которую мы хотим импортировать скопированный SQL-запрос. Обратите внимание, что базу данных предварительно необходимо создать (мы это сделали), а вот таблицу создаст запрос. Это видно из листинга SQL-запроса.
3. В верхнем меню выбрать пункт **SQL**. Появится окно (рис. 3.4).
4. Вставить **SQL-запрос** из буфера обмена в текстовое поле и нажать кнопку "**Go**".

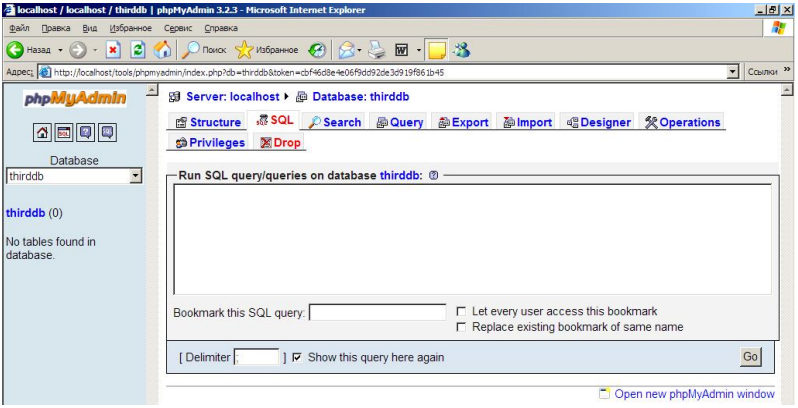


Рис. 3.4. Окно для вставки SQL-кода запроса на экспорт БД

В результате таблица `staff` будет создана в БД `thirddb`.

Вариант 2. Сохранение дампа таблицы БД в виде резервной копии в отдельном файле

Обычно сохранение дампа таблицы из базы данных в виде резервной копии в отдельном файле выполняется с расчетом на последующее возвращение данных на сервер. Поэтому рассмотрим полный цикл: сохранение дампа таблицы в файле и его восстановление в нужном месте. Пусть этим местом будет БД thirddb.

Шаг 1

Сперва нужно экспортировать таблицу `staff` с текущего местонахождения (из БД `firstdb`). В результате мы получим **SQL-запрос**, способный воссоздать нашу базу данных в другом месте. Но в отличие от предыдущего варианта в этом случае **SQL-запрос** надо будет сохранить в файле. Для этого надо сделать следующее:

1. Зайти на главную страницу PHPMyAdmin, щелкнув на одноименной гиперссылке.

2. Создать базу данных `thirddb` (или очистить ее от старого содержимого).

3. Выбрать базу данных `firstdb`, щелкнув на соответствующей гиперссылке.

4. Выбрать таблицу `staff`, щелкнув на соответствующей гиперссылке.

5. В верхнем меню выбрать пункт **Экспорт**. Появится окно (рис. 3.5).

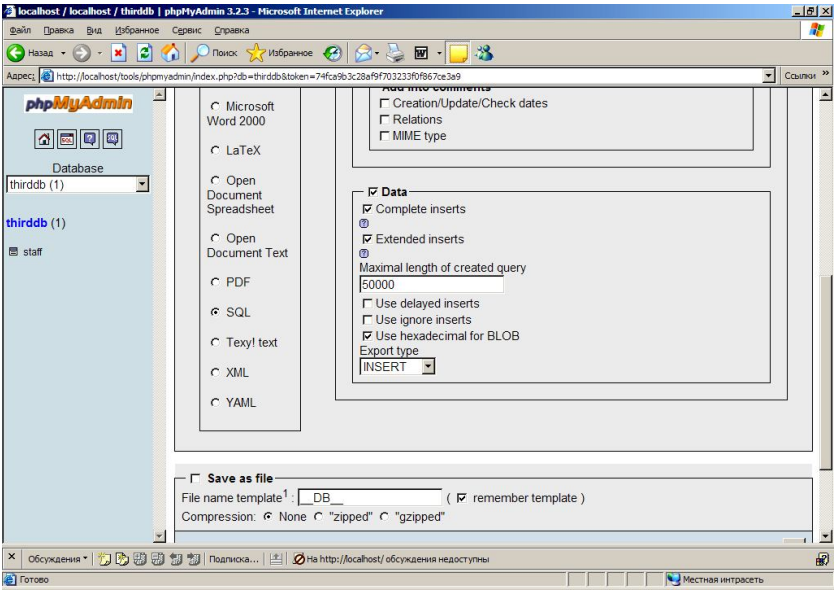


Рис. 3.5. Окно для запуска сохранения SQL-запроса в виде отдельного файла

6. На открывшейся странице поставим радио-переключатель на **SQL** и укажем **Save as file**, затем нажмем кнопку **Go.**

7. В следующем окне необходимо указать **Сохранить** файл (рис. 3.6).

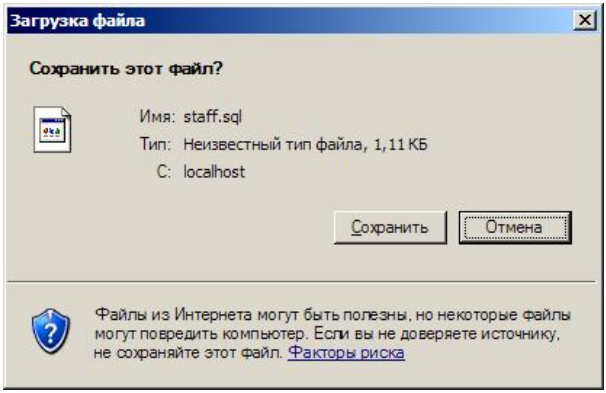


Рис. 3.6. Окно для сохранения SQL-запроса в виде файла

8. В появившемся окне сохранения выбрать место для сохранения. По умолчанию дамп таблицы будет сохранен в файле `staff.sql`.

Шаг 2

Вторым шагом является выполнение скопированного SQL-запроса в PhpMyAdmin. Для этого надо выполнить следующие действия:

1. Открыть файл `firstdb.sql` в редакторе **Блокнот.** Выделить текст дампа, скопировать его в буфер обмена.
2. Зайти на главную страницу PhpMyAdmin.
3. Выбрать базу данных `thirddb`, в которую мы хотим импортировать скопированный SQL-запрос. Обратите внимание, что базу данных предварительно необходимо создать (мы это сделали), а вот таблицу создаст запрос.
4. В верхнем меню выбрать пункт **SQL**. Появится окно (рис. 3.4).
5. Вставить **SQL-запрос** из буфера обмена в текстовое поле и нажать кнопку "**Go**".

Вариант 3. Перенос дампа БД на другую машину (например, с компьютера на работе на домашний компьютер)

Экспорт и импорт дампа таблицы БД выполняется так же, как и в предыдущем варианте.

Вариант 4. Перенос дампа БД с локального на удаленный реальный сервер сайта

Экспорт выполняется как в первом варианте — полностью, а для импорта программу PhpMyAdmin надо запускать на удаленном сервере.

Контрольные вопросы

1. Перечислите достоинства СУБД MySQL в качестве хранилища для Web-серверов.
2. Опишите структуру и состав нашей учебной БД firstdb.
3. Опишите процесс создания MySQL-базы данных в ручном режиме с помощью phpMyAdmin.
4. Каково назначение языка SQL?
5. Воспроизведите синтаксис SQL-оператора для создания базы данных.
6. Воспроизведите синтаксис SQL-оператора для создания таблицы.
7. Воспроизведите синтаксис SQL-оператора для выборки записей из таблицы.
8. Воспроизведите синтаксис SQL-оператора для вставки записи в таблицу.
9. Воспроизведите синтаксис SQL-оператора для обновления записи.
10. Воспроизведите синтаксис SQL-оператора для удаления записи.
11. Воспроизведите функцию PHP для подключения к серверу баз данных MySQL.
12. Воспроизведите функцию PHP для выбора базы данных.

13. Воспроизведите функцию PHP для выборки данных из таблицы.

14. Воспроизведите функцию PHP для извлечения из матрицы отдельных строк (записей).

15. Приведите пример использования сложного условия в SQL-операторе SELECT.

16. Как производится сортировка выводимых из БД данных?

17. Как производится ограничение количества выводимых из БД данных?

18. Перечислите варианты переноса (экспорта/импорта) БД в ходе разработки Web-приложения.

4. ЯЗЫК СЕРВЕРНОГО WEB-ПРОГРАММИРОВАНИЯ PHP

Можно считать, что в самом «первом приближении» материал по PHP был дан в разделе 2.1. То изложение имело целью обеспечить понимание примера серверного PHP-сценария контингентом, знакомым с языками программирования хотя бы на уровне Turbo Pascal.

Материал данного раздела даст возможность понять примеры программ для решения задач из раздела 3 и даст достаточно полное представление об элементах языка PHP, типах данных, операторах и функциях.

PHP в течение последних 10 лет является самым популярным языком серверного Web-программирования. Соответственно, книг, учебников, интернет-ресурсов по нему создано великое множество. Тем не менее, мы считаем, что предложение все же «не покрывает поле спроса». Для конкретной аудитории, конкретной специальности, конкретного объема времени, отведенного на обучение, преподавателям желательно самим «препарировать» имеющийся материал и адаптировать его для своей работы. Чем мы и займемся.

Будем стараться устранить излишнюю детализацию, существующую в одних источниках, чрезмерную краткость или отсутствие материала в других, сопровождать теоретический материал подходящими для нас примерами. И, конечно, все это будет делаться в соответствии со вкусами и предпочтениями автора. Иногда это создает для аудитории более комфортные условия для изучения материала по сравнению с «рысканием» от одного источника к другому.

Для получения исчерпывающе полной информации по языку PHP целесообразно пользоваться источником [9]. Наиболее логичное и компактное изложение материала, по мнению автора, в [7].

Итак, PHP — это работающий на стороне сервера встроенный язык сценариев, позволяющий разработчикам быстро и эффективно разрабатывать динамические web-приложения. С позиций грамматики и синтаксиса PHP напоминает язык программирования C.

Файлы, созданные на PHP, хранятся и обрабатываются на сервере. Когда пользователь запрашивает документ с PHP-кодом,

скрипт обрабатывается не браузером пользователя, как в случае Java Script, а сервером. После такой обработки пользователю передаются уже только результаты работы серверной программы в виде HTML-страницы без скриптов на PHP.

Первое, что нужно знать относительно синтаксиса PHP, — это то, как он встраивается в HTML-код, и как интерпретатор узнает, что это код на языке PHP. Имеется 4 варианта встраивания PHP-скрипта в код Web-страницы. Но достаточно помнить и пользоваться двумя вариантами. Простейший вариант (сокращенный) следующий:

```
<HTML>
……..
<?
Операторы PHP;
?>
…….
</HTML>
```

Но этим вариантом можно пользоваться только тогда, когда в конфигурационном файле PHP php.ini (он находится по адресу c:\webservers\usr\local\php5\php.ini) в разделе Language Options установлена директива, разрешающая короткий открывающий тег <?. В этом случае директива должна иметь вид:

Short_open_tag = On.

Если намечается публикация сайта в Интернете, во избежание “нестыковки” с настройкой конфигурационного файла необходимо использовать второй вариант встраивания PHP-скрипта в код Web-страницы:

```
<HTML>
……..
<? php
Операторы PHP;
?>
…….
</HTML>
```

Этот вариант «сработает» при любой настройке конфигурационного файла PHP.

PHP-скрипт может быть расположен в любом месте HTML-документа, и даже до открывающего тега <HTML>.

Так же как в Pascal, Си, Perl операторы завершаются точкой с запятой, но перед закрывающим тегом «?>» ее можно не ставить (ставить тоже не запрещено).

Забегая вперед, рассмотрим синтаксис оператора (функции) вывода на экран, поскольку он понадобится для иллюстраций при дальнейшем изложении.

Для вывода на экран будем пользоваться оператором echo. Он позволяет выводить значения переменных, констант и смешанные значения. Примеры:

Таблица 4.1

№ п/п	Действие	Пример и описание
1	Вывод строковых значений	echo "Заказ обработан в:";
2	Перевод курсора в начало следующей строки	echo " ";
3	Тег <P> перед выводимой строкой вставляет пустую строку	echo "<p>Ваш заказ следующий:";
4	Вывод значения арифметического выражения	Echo (20+12); или Echo 20+12;
5	Вывод значения переменной	Echo $name; //здесь $name имя переменной
6	Вывод значения константы	Echo CONST; //здесь CONST имя константы
7	Вывод смешанных значений	echo "$oilqty бутылки масла ";
8	Конкатенация	echo $oilqty. " бутылки масла ";
9	Конкатенация и вычисление значения функции	Echo “В массиве ”. count($arr). “элементов”
10	Вывод имен переменных без интерпретации, в виде последовательности символов	Echo ‘$name Расмус Лердоф’; (выведет строку «$name Расмус Лердоф»)
11	Вывод значения функции	echo date ("H: i, d. m. Y.");

Для иллюстраций при дальнейшем изложении понадобятся и комментарии. PHP предоставляет несколько методов для вставки комментариев. Для однострочных комментариев проще всего пользоваться двойным слэш «//». Для многострочных комментариев можно использовать «/*... */» (стиль С).

И, наконец, PHP-скрипты (или сценарии) нужно сохранять в файлах с расширением .php. В именах файлов желательно использовать только строчные латинские буквы и не использовать пробел (это требование ОС Unix).

4.1. Элементы языка (Синтаксис языка) PHP

4.1.1. Алфавит

PHP-скрипт представляет собой последовательность строк, состоящих из символов, образующих алфавит языка. Алфавит PHP включает следующие символы:

- заглавные и строчные латинские буквы и символ “подчеркивание”;
- арабские цифры;
- специальные символы (используются для конструирования знаков операций, выражений, комментариев, а также как синтаксические разделители).

Кроме этих символов в PHP-программах в двух случаях могут использоваться русские буквы:

- в комментариях;
- в качестве значений констант и переменных строкового типа.

4.1.2. Служебные слова

Как и в других языках программирования, в PHP 5 используются служебные (зарезервированные) слова. Это слова, входящие в состав операторов языка, названия операций и предопределенных функций (case, do, or, foreach и т.д.). Зарезервированные слова нежелательно (по правилам «хорошего тона») использовать не по назначению (например, в качестве идентификаторов переменных и констант или имен пользовательских функций).

Служебные слова регистронезависимые. Например: True и true интерпретатором не различаются.

4.1.3. Идентификаторы

Идентификаторы в PHP — это имена констант, переменных, функций. При построении идентификаторов необходимо выполнять следующие требования:

- идентификатор должен начинаться с латинской буквы и может содержать латинские буквы, цифры и знак подчеркивания;
- в идентификаторах переменных и констант строчные и заглавные буквы различаются (регистрозависимые);
- в качестве идентификатора нежелательно использовать служебные слова (хотя наличие префикса $ в качестве первого символа идентификатора делает это возможным).

4.1.4. Данные

Константы

Константы объявляются в PHP при помощи функции **define**():

define(“CONSTANT”, value);

Первый параметр этой функции — имя константы, второй — ее значение. По умолчанию имена констант регистрозависимые (но это свойство можно изменить).

```
define(“CONSTANT1”,15);//целое число
define(“CONSTANT2”,1.5);//вещественное число
define(“CONSTANT3”,true);//логическое значение
define(“CONSTANT4”,"Hello");// строка
echo(“CONSTANT1”);// будет выведено «15»
echo(“CONSTANT2”);// будет выведено «1.5»
echo(“CONSTANT3”);// будет выведено «true»
echo(“CONSTANT4”);// будет выведено «Hello»
```

По традиции имена констант пишут буквами верхнего регистра.

Как видно из примеров, константами могут быть значения целого, вещественного, логического и строкового типов.

Переменные

В PHP идентификаторы переменных должны начинаться со знака доллара ($). Имена переменных в PHP чувствительны к регистру.

При объявлении переменных в PHP не требуется явно указывать тип переменной. При этом одна и та же переменная на протяжении программы может принимать разные типы:

$var=”2”; // строка (ASCII-код 50)

$var=2; // теперь целое число 2

$var=1.3 + $var; /*Теперь вещественное число 3.3. Тип результата определяется типом первого значения, расположенного справа от знака присвоения*/

$var=20 + “12 лет”; /*Опять целое число 32. По предыдущему правилу тип результата должен быть целый. К целому числу 20 прибавляется второе слагаемое 12 после отсечения фрагмента, не удовлетворяющего типу integer*/

Надо заметить, что в PHP не предусмотрена возможность объявления переменной без присвоения ей значения (как, например, в Pascal и JavaScript). Переменная создается (объявляется и инициализируется) в момент присваивания ей значения и существует до тех пор, пока выполняется программа.

4.1.5. Операции

Операции позволяют выполнять различные действия с переменными и константами. Среди операций PHP можно выделить 5 групп: арифметические, строковые, логические, сравнения, инкремента и декремента.

Таблица 4.2

Арифметические операции

Обозначение	Название	Пример
		$a=5; $b=4;
+	Сложение	$a + $b = 9
-	Вычитание	$a - $b =1
*	Умножение	$a * $b =20
/	Деление	$a / $b =1.25
%	Остаток от деления	$a % $b = 1

Таблица 4.3

Строковые операции

Обозначение	Название	Пример
.	Конкатенация (сложение строк)	`$c = $a . $b` (это строка, состоящая из $a и $b)

Таблица 4.4

Логические операции

Обозначение	Название	Пример
and	И	$a and $b
&&	И	$a && $b
or	Или	$a or $b
||	Или	$a || $b
xor	Исключающее или	$a xor $b
!	Инверсия (NOT)	! $a

Таблица 4.5

Операции сравнения (отношения)

Обозначение	Название	Пример
==	Равенство	$a == $b
===	Эквивалентность	$a === $b
!=	Неравенство	$a != $b
<>	Неравенство	$a <> $b
!==	Неэквивалентность	$a !== $b
<	Меньше	$a < $b
>	Больше	$a > $b
<=	Меньше или равно	$a <= $b
>=	Больше или равно	$a >= $b

Таблица 4.6

Операции инкремента и декремента

Обозначение	Название	Описание	Пример

++$a	Пре-инкремент	Увеличивает $a на единицу и возвращает $a	`<?`
			`$a=4;`
			`echo ++$a;`
			`//выведет 5`
			`?>`

$a++	Пост-инкремент	Возвращает $a, затем увеличивает $a на единицу	<?
			$a=4;
			echo $a++;
			//выведет 4
			?>

--$a	Пре-декремент	Уменьшает $a на единицу и возвращает $a	<? $a=4; echo --$a; //выведет 3 ?>
$a--	Пост-декремент	Возвращает $a, затем уменьшает $a на единицу	<? $a=4; echo $a--; //выведет 4 ?>

4.1.6. Выражения

Программа преобразовывает данные (константы и переменные). Действия над данными и последовательность этих действий задаются в программе выражениями. Выражение — это правило получения нового значения.

Выражения входят в состав операторов. Они строятся из констант, переменных, стандартных функций, разделенных знаками операций, круглыми и фигурными скобками. Выражения характеризуются типом. Тип выражения определяется типом данных, входящих в него, и используемыми операциями.

Примеры выражений:

- 0.5 + b

1.25 * sin(x) + s арифметические выражения (могут принимать целые или вещественные значения)

- ‘TURBO’ + ‘PASCAL’ строковое выражение
- IF (C>0.5) and (C< 1.5) логическое выражение

В качестве выражения могут выступать и одиночные константы, и переменные.

S:= 5; S:= F;

4.2. Типы данных

PHP поддерживает восемь типов данных.

Четыре скалярных типа:

- boolean (логический);
- integer (целый);
- float (с плавающей точкой);
- string (строковый).

Два смешанных типа:

- array (массив);
- object (объект).

И два специальных типа:

- resource (ресурс);
- NULL.

В PHP не принято явное объявление типов переменных. Это делает сам интерпретатор во время выполнения программы в зависимости от контекста, в котором используется переменная. Рассмотрим по порядку все перечисленные типы данных.

4.2.1. Тип boolean (логический тип)

Этот тип данных может принимать только два значения: true и false. Используется этот тип данных так же, как и в других языках: в различных управляющих конструкциях (циклах, условиях и т.п.).

Например, в условном операторе проверяется истинность значения выражения (частным случаем является проверка условия) и, в зависимости от результата проверки, выполняются те или иные действия. Надо заметить, что правила приведения выражения к логическому типу будут рассмотрены в разделе 4.3.3.

Пример 1.

```
$know=False;
if ($know == False) {echo "Изучай PHP!"}; /*фигурные скобки обязательны, только если оператор составной*/
```

Здесь операция '==' проверяет условие (равенство) и возвращает булево значение. Если проверяемое условие выполняется, т.е. переменная $know имеет значение false, выводится сообщение «Изучай PHP».

Пример 2.

То же самое по-другому:

```
$know=False;
if (!$know) {echo "Изучай PHP!"};
```

Проверяется, имеет ли переменная $know значение False. Если имеет, выводится сообщение.

Пример 3.

```
$know = "Изучить PHP”;
if ($know == "Изучить PHP"){ echo "Начал изучать"; }
```

Оператор == проверяет, совпадает ли значение переменной $know со строкой "Изучить PHP". Если совпадает, то возвращает true, иначе — false. Если возвращено true, то выполняется то, что внутри фигурных скобок.

4.2.2. Тип integer (целые)

Этот тип задает число из множества целых чисел Z = {..., -2, -1, 0, 1, 2, ...}. Целые могут быть указаны в десятичной, шестнадцатеричной или восьмеричной системе счисления, по желанию с предшествующим знаком «-» или «+».

Если используется восьмеричная система счисления, число надо предварить нулем (0). Для использования шестнадцатеричной системы перед числом нужно поставить 0x.

```
$a = 2012; // положительное десятичное число
$a = -2012; // отрицательное число
$a = 0777; // восьмеричное число
$a = 0x1F; // шестнадцатеричное число
```

Максимальное значение целых чисел — около двух миллиардов (это 32-битное знаковое). Беззнаковые целые PHP не поддерживает.

В PHP результат деления целых чисел — это число вещественное (с плавающей точкой). Результатом 3/2 будет вещественное число 1.5.

4.2.3. Tun float (числа с плавающей точкой)

Числа с плавающей точкой (они же вещественные числа) могут быть представлены в форме с фиксированной точкой и плавающей точкой:

$a = 2.012; // Число в форме с фиксированной точкой

$b = 1.2e3; // число 1200 // Число в форме с плавающей точкой

$c = 7E-3; // число 0.007 // Число в форме с плавающей точкой

Размер числа с плавающей точкой зависит от соответствующей настройки конфигурационного файла PHP, хотя максимум, как правило, ~1.8e308 с точностью 12 десятичных цифр.

4.2.4. Tun string (строка)

Строка — это набор символов. В PHP двоичный код символа — это байт, это значит, что используются 256 различных символов. Это также означает, что PHP не имеет встроенной поддержки двухбайтовой кодировки Unicode.

Строка в PHP может быть определена тремя различными способами:

- с помощью одинарных кавычек;
- с помощью двойных кавычек;
- heredoc-синтаксисом.

Определение строки с помощью одинарных кавычек (апострофов)

Первый способ определить строку — это заключить ее в одинарные кавычки «'». Чтобы использовать саму одинарную кавычку внутри строки, как и во многих других языках, перед ней необходимо поставить обратный слэш «\'», т.е. экранировать ее.

Пример.

Echo 'Синтаксис оператора вывода в Паскале: Writeln(\'s=\',s:7:3);';

Будет выведено:

Синтаксис оператора вывода в Паскале: Writeln('s=',s:7:3);

Если в используемой строке обратный слэш должен идти перед одинарной кавычкой либо быть в конце строки, его необходимо продублировать «\\'».

Если внутри строки, заключенной в одинарные кавычки, обратный слэш «\» встречается перед любым другим символом (отличным от «\» и «'»), то он рассматривается как обычный символ и выводится, как и все остальные. Поэтому обратную косую черту необходимо экранировать, только если она находится в конце строки, перед закрывающей кавычкой.

Идентификаторы переменных, имеющиеся в составе строки, заключенной в одинарные кавычки, не обрабатываются, они выводятся как текст:

```
$year = 2012;
echo ‘На дворе $year год’; // будет выведено «На дворе $year год».
```

Определение строки с помощью двойных кавычек

Если строка заключена в двойные кавычки «"», для использования в ней самого символа «"» его также надо экранировать — «\"». Но отличие от заключения в одинарные кавычки не только в этом. При заключении строки в двойные кавычки PHP распознает и обрабатывает большее количество управляющих последовательностей для специальных символов (не только «\'»). Некоторые из них приведены в таблице.

Таблица 4.7

Управляющие последовательности

Последовательность	Значение
\n	Новая строка (LF или 0x0A (10) в ASCII)
\r	Возврат каретки (CR или 0x0D (13) в ASCII)
\t	Горизонтальная табуляция (HT или 0x09 (9) в ASCII)
\\	Обратная косая черта
\$	Знак доллара
\"	Двойная кавычка

Пример для «\n»:

```
Echo “Фамилия — Иванов\nИмя — Иван\nОтчество — Иванович”;
```

Будет выведено:

```
Фамилия — Иванов
Имя — Иван
Отчество — Иванович
```

Но самым важным, часто используемым свойством строк в двойных кавычках является обработка идентификаторов переменных — выводятся не идентификаторы переменных, как в случае одинарных кавычек, а их значения.

Пример:

```
$year=2012;
echo ‘На дворе $year год’; // будет выведено «На дворе $year год»
echo “На дворе $year год”; // будет выведено «На дворе 2012 год».
```

Определение строки с помощью конструкции Heredoc

Третий способ определения строк — это использование heredoc-синтаксиса. Им целесообразно пользоваться, если выводимый текст большой, например, многострочный, и в нем много апострофов и кавычек, которые при использовании первых двух методов (обрамлении текста одиночными или двойными кавычками) пришлось бы экранировать. Примером такого текста может быть фрагмент HTML-кода, где значения нескольких атрибутов тегов заключены в кавычки.

При данном способе определения строка должна начинаться с символа <<<, после которого идет идентификатор (обычно HERE, но можно любой). Заканчивается строка этим же идентификатором. Закрывающий идентификатор должен начинаться в первом столбце строки. Кроме того, идентификатор должен соответствовать всем правилам именования в PHP.

Heredoc-текст ведет себя так же, как и строка в двойных кавычках, при этом их не имея. Поэтому нет необходимости экранировать кавычки в heredoc. Использовать перечисленные выше управляющие последовательности в heredoc можно. Переменные внутри heredoc тоже обрабатываются, т.е. будут выводиться не их названия, а их значения.

```
$name = 'Вася';
echo <<<HERE Моя фирма называется ‘Рога и копыта’, а меня зовут "$name".
HERE;
```

Будет выведено: «Моя фирма называется ‘Рога и копыта’, а меня зовут Вася».

4.2.5. Тип array (массив)

Массив — это совокупность упорядоченных данных, объединенных общим именем. Создать массив можно двумя способами: с помощью конструкции array() или непосредственно задавая значения его элементам.

Создание массива при помощи array()

Синтаксис имеет два варианта:

1. $имя_массива=array (“key1” => “value1”, “key2” => “value2”, ...);

2. $имя_массива=array (“value1”, “value2”, ...);

В первом варианте задания массива конструкция array() принимает в качестве параметров пары «ключ => значение», разделенные запятыми. Символ => устанавливает соответствие между значением и его ключом. Ключ может быть как целым числом, так и строкой.

Во втором варианте задания массива конструкция array() принимает в качестве параметров пары «числовой индекс, значение». Числовой ключ массива называют индексом. Индексирование массива в PHP начинается с нуля.

В случае ключей строкового типа массив называют ассоциативным. Значения элементов массива могут быть любого имеющегося в PHP типа. Причем массив в PHP может содержать элементы разных типов:

$book = array ("php" => "PHP users guide", “цена” => 300);

Такой же массив можно задать и по второму варианту:

$book = array ("PHP users guide", 300);

Но этот массив не будет ассоциативным, ключи его будут числовые.

/* Создали массив со значениями " PHP users guide" и "300". Поскольку ключи не указаны, они будут 0,1,2 соответственно */

Значение элемента массива можно получить, указав после имени массива в квадратных скобках ключ искомого элемента. Для первого варианта синтаксиса это будет так:

echo $book["php"]; //выведет «PHP users guide»

echo $book[“цена”]; //выведет «300»

Аналогично для второго варианта синтаксиса:

```
echo $book[0]; //выведет «PHP users guide»
echo $book[1]; //выведет «300»
```

Создание массива с помощью синтаксиса квадратных скобок

Создать массив можно, просто записывая в него значения:

```
$book ["php"] = "PHP users guide";
$book [“цена”] = 300;
```

Возможны вариации этого способа задания массива:

```
$book [0] = "PHP users guide";
$book [1] = 300;
```

или

```
$book [] = "PHP users guide";
$book [] = 300;
```

Но эти массивы не будут ассоциативными.

Если указать новый ключ и новое значение, например, $book["publ_year"]="2011", то в массив добавится новый элемент.

Для того чтобы изменить конкретный элемент массива, нужно просто присвоить ему с его ключом новое значение. Изменить ключ элемента нельзя, можно только удалить элемент (пару ключ/значение) и добавить новую. Чтобы удалить элемент массива, нужно использовать функцию unset().

```
<?php
unset($book["publ_year"]); // Удаляет элемент с ключом "publ_year"
unset ($book); // удаляет массив полностью
?>
```

Заметим, что когда при добавлении элемента в массив используются пустые квадратные скобки, максимальный числовой ключ ищется среди ключей, существующих в массиве с момента последнего переиндексирования. Переиндексировать массив можно с помощью функции array_values().

```
<?php
$arr =array ("a","b","c");
```

```
/* Создали массив со значениями "a", "b" и "c". Поскольку ключи не указаны, они будут 0,1,2 соответственно */
print_r ($arr); // выведет «Array ( [0] => a [1] => b [2] => c )»

// удаляем из него все значения, но не сам массив
unset($arr[0]);
unset($arr[1]);
unset($arr[2]);
print_r($arr); // «выведет Array ( )»

//добавляем новый элемент в массив. Его индексом (ключом) будет 3, а не 0
$arr[] = "aa";
print_r($arr); // выведет «Array ( [3] => aa )»

$arr =array_values($arr); // переиндексируем массив
$arr[] = "bb"; // ключом этого элемента будет 1
print_r($arr); // выведет «Array ( [0] => aa [1] => bb )»

?>
```

Для заполнения массивов часто используют генератор случайных чисел. Приведем пример такого заполнения массива.

```
$arr=array();
for ($i=1; $i<=10; $i++)
$arr[]=mt_rand(1, 100);
```

Очевидно, массив $arr заполняется десятью целыми случайными числами из диапазона [1, 100].

Завершая изложение материала по массивам, необходимо отметить, что массивы в PHP кардинально отличаются от массивов в Turbo Pascal:

- массивы PHP могут быть ассоциативными;
- количество элементов не фиксировано;
- элементы массива могут быть разных типов;
- элементы массива можно удалить;
- элементы массива можно переиндексировать.

4.2.6. Тип object (объекты)

Объекты — тип данных, пришедший из объектно-ориентированного программирования (ООП). Согласно принципам ООП, класс — это набор объектов, обладающих определенными свойствами и методами работы с ним, а объект, соответственно, — экземпляр класса.

Например, программисты — это класс людей, которые пишут программы, изучают компьютерную литературу и, кроме того, как все люди, имеют имя и фамилию. Теперь, если взять одного конкретного программиста, Васю Иванова, то можно сказать, что он является объектом класса программистов, обладает теми же свойствами, что и другие программисты, тоже имеет имя, пишет программы и т.п.

В PHP для доступа к методам объекта используется оператор —>. Для инициализации объекта используется выражение new, создающее в переменной экземпляр объекта.

```
<?php
//создаем класс людей
class Person
{
//создаем метод, который обучает человека PHP
 function know_php()
 {
 echo "Теперь я знаю PHP";
 }
}
$bob = new Person; // создаем объект класса человек
$bob —> know_php(); // обучаем его PHP
?>
```

Пример взят из [8].

4.2.7. Тип resource (ресурсы)

Ресурс — это специальная переменная, содержащая ссылку на внешний ресурс (например, соединение с базой данных). Ресурсы создаются и используются специальными функциями (например, mysql_connect(), pdf_new() и т.п.).

4.2.8. Тип null

Специальное значение NULL говорит о том, что переменная не имеет значения.

Переменная считается NULL, если:

- ей была присвоена константа NULL ($var = NULL);
- ей еще не было присвоено какое-либо значение;
- она была удалена с помощью функции unset().

Существует только одно значение типа NULL — регистронезависимое служебное слово NULL.

4.3. Операторы

Операторы управляют процессом выполнения программы. Набор операторов PHP типичен для языков программирования высокого уровня.

Составные и структурированные операторы

Начиная изложение материала по операторам языка PHP, уместно упомянуть, что, как и в других языках, здесь тоже используются термин «простой оператор» и конструкции, называемые «составной оператор» и «структурный оператор».

Простой оператор не содержит в себе других операторов.

Составной оператор — это последовательность из двух и более операторов, обрамленных фигурными скобками (в PHP).

Структурированный оператор — это оператор, содержащий в себе несколько простых операторов. Например, это условный оператор.

4.3.1. Операторы вывода

Оператор (функция) вывода echo и примеры его использования были рассмотрены в начале этого раздела. Имеются еще две разновидности оператора вывода: **print** и **printf**.

4.3.2. Оператор присвоения

Одной формой оператора присвоения нам уже пришлось многократно воспользоваться при изложении предыдущего материала:

$var=<вычисленное выражение>;

Как и в других языках программирования, при этом переменная слева от знака присвоения «=» получает значение выражения, стоящего справа от знака присвоения.

Кроме этой формы имеются еще другие формы этого оператора, представленные в таблице. Во многих случаях для подготовленного человека они облегчают работу при кодировании программы и несколько уменьшают длину кода.

Таблица 4.3

Обозначение	Название	Описание	Пример
=	Присваивание	Переменной слева от оператора будет присвоено значение, полученное в результате выполнения каких-либо операций или переменной/константы с правой стороны	`$a = ($b = 4) +5;` ($a будет равна 9, $b будет равна 4)
+=		Сокращение. Прибавляет к переменной число и затем присваивает ей полученное значение	`$a += 5;` (эквивалентно $a = $a + 5;)
.=		Сокращенно обозначает комбинацию операций конкатенации и присваивания (сначала добавляется строка, потом полученная строка записывается в переменную)	`$b = "Привет ";` `$b .= "всем";` (эквивалентно $b = $b . "всем";) В результате: $b="Привет всем"

4.3.3. Условный оператор

Оператор if (неполный условный оператор)

Позволяет организовывать выполнение фрагментов кода по условию. Синтаксис оператора следующий:

if (выражение) оператор;

Если оператор составной, его надо заключить в фигурные скобки.

Частным случаем выражения в этом операторе является сравнение, результатом которого всегда является логическое значение.

В общем случае в процессе выполнения оператора выражение преобразуется к логическому типу. Если результат преобразования — истина (true), выполняется **оператор**. В противном случае оператор пропускается.

Правила приведения **выражения** к логическому типу следующие:

1. В FALSE преобразуются следующие значения:
- логическое False;
- целый ноль (0);
- действительный ноль (0.0);
- пустая строка “” и “0”;
- массив без элементов;
- объект без переменных;
- специальный тип NULL.

2. Все остальные значения преобразуются в TRUE.

Примеры.

```
1. if($a>$b){echo "$a больше $b”; $bufer=$a; $a=$b;};
//производится простое сравнение
//Составной оператор заключается в фигурные скобки
2. $result=mysql_query("SELECT * FROM staff ", $link);
while ($myrow=mysql_fetch_array($result))
{
echo "<TR>
<TD>$myrow[id]</TD>
<TD>$myrow[name]</TD>
<TD>$myrow[lastname]</TD>
<TD>$myrow[dol]</TD>
</TR>";
}
```

В этом примере из раздела 3.5.4 переменная $result представляет собой таблицу, а в переменную $myrow функция mysql_fetch_array($result) считывает строки из таблицы.

Пока значение переменной $myrow равно True, на экран выводятся строки таблицы из четырех полей. А когда из переменной $result, представляющей собой таблицу, будут выбраны все строки, значение переменной $myrow становится равным False и цикл while завершается.

Оператор if — else (полный условный оператор)

Синтаксис оператора следующий:

```
if (выражение) {оператор1;} else {оператор2;}
```

Расширяет возможности `IF` по части обработки вариантов `выражения`, когда оно равно `FALSE`. Ветвь после `ELSE` выполняется только если `выражение` равно `FALSE`. Если операторы простые, не составные, фигурные скобки не обязательны. Простые операторы обрамляют фигурными скобками только для удобочитаемости.

Пример:

```
if ($a>$b) {echo "a больше b";} else {echo "a меньше b";}
```

Конструкция elseif

Является комбинацией IF и ELSE. Позволяет в одном if-блоке проверить несколько условий и для каждого выполненного условия выполнить соответствующий оператор (простой или составной). Когда ни одно из проверяемых условий не выполняется, выполняется соответствующий этому случаю оператор.

```
if ($a>$b) {echo "a больше b";}
 elseif ($a==$b) {echo "a равно b";}
 else {echo "a меньше b";}
```

В одном if-блоке может быть несколько конструкций elseif. В последнем примере могли бы быть проверки равенства переменной $a разным значениям типа:

```
elseif ($a==0) {echo "a равно 0";}
```

4.3.4. Оператор switch

Позволяет организовать разветвления в нескольких направлениях. В зависимости от значения выражения (селектора) $a выполняет различные фрагменты кода.

```
switch ($A)
{
 case 0: echo "A равно 0";
 break;
 case 1: echo "A равно 1";
 break;
 case 2: echo "A равно 2";
 break;
 default: echo "A не равно 0, 1, 2";
}
```

default - соответствует любым значениям, которые не удовлетворяют значениям при CASE. Селектором могут быть данные любого скалярного типа, т.е. целые числа или числа с плавающей запятой, и строки.

Как видим, этот оператор похож на оператор if…elseif…else. Однако, здесь не происходит приведения селектора к логическому типу, как это делается в операторе if.

4.3.5. Операторы цикла

Цикл while

Это цикл с предусловием. Синтаксис его такой:

WHILE(выражение) {операторы};

Как и в условном операторе, здесь частным случаем **выражения** может быть проверка условия.

Операторы (тело цикла) выполняются до тех пор, пока вычисление **выражения** дает значение TRUE. Если **выражение** равно FALSE с самого начала, то цикл не выполнится ни разу. Очевидно, для того, чтобы цикл не был бесконечным, в теле цикла должны выполняться действия, влияющие на значение **выражения**.

У этого оператора есть **альтернативный синтаксис:**
WHILE(выражение): операторы; ENDWHILE;

```
$i=1;
while ($i<=5) { echo $i++; }

$i=1;
while ($i<=5): echo $i; $i++; endwhile;
```

Эти два примера выводят числа с 1 до 5.

Пример из раздела 3.5.4 (листинг 3.2).

Следующий скрипт выводит на экран данные сотрудников из БД staff в виде таблицы. Для этого используется цикл while. При каждом проходе цикла выводятся данные одного сотрудника в одну строку таблицы. Сигналом для завершения цикла служит отсутствие массива в переменной $myrow. Как было сказано в разделе 4.3.3, массив без элементов дает значение FALSE.

```
while ($myrow=mysql_fetch_array($result)){
echo "<TR>
<TD>$myrow[id]</TD>
<TD>$myrow[name]</TD>
<TD>$myrow[lastname]</TD>
<TD>$myrow[dol]</TD>
</TR>";
}
```

Цикл do_while

Это цикл с постусловием, значение логического выражения проверяется не до, а после окончания итерации. Основное отличие в том, что цикл хоть один раз, но выполнится.

Синтаксис:

Do {операторы} while (выражение);

Операторы (тело цикла) выполняются до тех пор, пока вычисление **выражения** дает значение TRUE.

Пример

```
$i=12;
```

```
do { if (I % 2 = 0) echo $i; $i++;}
while ($i<10);
```

//цикл выполнится один раз, хотя условие не выполняется с самого начала (выведет «12»)

Пример из раздела 3.5.3 (листинг 3.1.)

Следующий скрипт выводит на экран данные сотрудников из БД staff. Для этого используется цикл do_while. При каждом проходе цикла выводятся данные одного сотрудника. Сигналом для завершения цикла служит отсутствие массива в переменной $myrow. Как было сказано в разделе 4.3.3, массив без элементов дает значение FALSE.

```
$result=mysql_query("SELECT * FROM staff ", $link);
$myrow=mysql_fetch_array($result); /заносит в $myrow первую запись
do
{
echo “Сотрудник № — “ . $myrow[‘id’]. “<BR>”;
echo $myrow[‘name’]. “<BR>”;
echo $myrow[‘lastname’]. “<BR>”;
echo $myrow[‘dol’]. “<BR><BR>”;
}
while ($myrow=mysql_fetch_array($result)); /*заносит в $myrow следующую запись до тех пор, пока не будут выведены все записи из таблицы*/.
```

Цикл for

Это цикл с параметром. Он используется, когда количество повторений цикла известно заранее.

Синтаксис оператора:

FOR (выражение1; выражение2; выражение3) {операторы тела цикла};

Выражение1 — это оператор, устанавливающий начальное значение счетчика цикла. Оно вычисляется в начале цикла.

Выражение2 — это условие выхода из цикла. Оно вычисляется в начале каждой итерации. Если оно равно TRUE, то цикл про-

должается и выполняются операторы тела цикла. Если оно равно FALSE, то цикл заканчивается.

Выражение3 — это приращение счетчика. Оно вычисляется в конце каждой итерации.

Пример 1.

```
for ($i = 1; $i <= 10; $i++) { echo $i;}
```

Цикл выведет: 1 2 3 4 5 6 7 8 9 10

Имеется два специальных оператора, используемые в цикле for:

- Continue — прерывает выполнение текущей итерации цикла;
- Break — прерывает выполнение цикла.

Пример 2.

Для вывода только нечетных чисел фрагмент из примера 1 надо изменить так:

```
for ($i = 1; $i <= 10; $i++)
{
if ($i % 2==0) Continue;
else echo $i;
}
```

Пример 3.

Для вывода только четных цифр можно использовать следующий фрагмент.

```
for ($i = 0;; $i++)
{
if ($i >= 10) break;
if ($i % 2==0) echo $i;
}
```

Как видим, в операторе for опустили условие выхода из цикла. Выход из цикла осуществляет break.

4.3.6. Оператор include (включить)

Вставляет в указанное место в PHP-файле и выполняет содержимое другого заданного файла.

Нетрудно представить случай, когда один и тот же фрагмент PHP-кода используется во многих скриптах в разных PHP-файлах. Такое у нас встречалось в разделе 3.5. Там в файлах, предназначенных для выбора данных из базы, добавления, обновления и удаления данных в базе обязательно присутствует фрагмент, устанавливающий соединение с MySql-сервером и выбирающий нужную БД:

```
$link = mysql_connect ("localhost", "student", "12345");
mysql_select_db ("firstdb", $link);
```

Так вот, вместо этого фрагмента во всех пяти файлах: Mysql_php.php, Mysql_insert_1.php, Mysql_insert_2.php, Mysql_update.php, Mysql_delete.php можно вставить оператор **include “link.php”**. Очевидно, предварительно надо создать этот файл и поместить его в папку с перечисленными файлами. Файл **link.php** должен содержать следующий скрипт:

```
<?
$link = mysql_connect ("localhost", "student", "12345");
mysql_select_db ("firstdb", $link);

?>
```

Теги, обрамляющие PHP-скрипт, здесь обязательны.
Вот еще пример, иллюстрирующий работу оператора **inqlude**:

В файле vars.php содержится следующий скрипт:

```
<?php

$color = 'green';
$fruit = 'apple';

?>
```

В файле vars.php определяются переменные $color и $fruit. Далее этот файл включается в файл test.php и значения указанных переменных используются при выполнении программы из этого файла.

Вот содержимое файла test.php:

```
<?php

echo "A $color $fruit"; // Будет выведено только «A», потому что переменные $color и $fruit пока не определены
include 'vars.php';

echo "A $color $fruit"; // Будет выведено «A green apple», потому что в подключенном файле vars.php переменные $color и $fruit определяются

?>
```

4.4. Функции в PHP

4.4.1. Встроенные функции PHP

PHP имеет огромное количество встроенных функций различного назначения [6, 11]. Вот перечень наиболее часто используемых групп функций:

1. Для работы с переменными.
2. Даты/времени.
3. Для работы с массивами.
4. Математические.
5. Для манипуляций со строками.
6. Для работы с файлами.
7. Обнаружения орфографических ошибок.

Перед использованием встроенных функций в программе с ними приходится знакомиться по справочникам. При этом надо обратить внимание на то, возвращает ли она значение или не возвращает. Потому что от этого зависит схема использования функции в программе (в виде отдельного оператора или в составе выражения или оператора).

```
Sort($arr); // функция Sort() не возвращает значения
$arr=array_unique ($arr); //функция array_unique ();
                                    возвращает значения
```

Некоторые из встроенных функций PHP использовались при описании решения практических задач в разделах 2 и 3. Там не было детального описания этих функций, хотя из контекста было ясно их действие.

Далее мы рассмотрим несколько примеров из первых трех групп функций. При этом будем ссылаться на предыдущее изложение и приведем дополнительные примеры использования рассматриваемых функций.

Пример 1

Функция **isset(переменная)** определяет, была ли установлена переменная значением, отличным от null [11].

Если установлена, возвращается 1, иначе — значение NULL.

Обычно эта функция используется в условном операторе `If (выражение) оператор` как значение «`выражения`». Казалось бы, выражение в условном операторе должно быть логического типа. А функция isset, только что было сказано, возвращает значения 1 или NULL.

Чтобы устранить эту «нестыковку», надо обратиться к разделу 4.3.3. Там сказано, что выражения, используемые в условном операторе, в процессе вычисления приводятся к логическому типу. Так происходит и с выражением в функции isset.

Итак, в процессе выполнения условного оператора `выражение` приводится к логическому типу. Если результат преобразования — истина (true), выполняется **оператор**. В противном случае оператор пропускается.

Казалось бы, эта функция должна быть способна определить, введено ли значение в поле формы, или оно пустое. Например, таким образом:

If (isset($_POST[“имя_поля_формы”]))

Echo “В поле «имя_поля_формы» значение введено”;

Или, по-другому:

Echo isset($_POST[“имя_поля_формы”]); //Будет выведено: 1 или ничего не будет выведено

Однако детальное рассмотрение функции показывает, что такая проверка возможна только для полей типов: radio, checkbox и select. В случае этих полей действительно функция isset() возвращает «1» только тогда, когда выбран хотя бы один вариант.

А в случае полей типов: text, textarea, независимо от того, введено ли значение в такое поле или оно пустое, isset() возвращает значение «1». Так что в данном случае функция проверяет только, существует ли такой элемент в суперглобальном массиве $_POST. А элемент будет существовать всегда, если в форме существует поле с заданным именем «имя_поля_формы».

Для определения, введено ли значение в поля формы типов text и textarea или оно пустое, можно использовать функцию empty().

Пример 2

Функция **empty(var)** возвращает **`FALSE`**, если *`var`* содержит непустое или ненулевое значение. Если переменная var пустая, функция возвращает **`TRUE`.**

Как пустые воспринимаются следующие значения [11]:

- *""* (пустая строка)
- *0* (целое число)
- *0.0* (дробное число)
- *"0"* (строка)
- **`NULL`**
- **`FALSE`**
- *array()* (пустой массив)
- *var $var;* (переменная объявлена, но не имеет значения в классе)

Функцию isset мы использовали в разделе 3.5.6 (листинг 3.4).

```
<?
if (isset($_POST [‘name’]))
{
$name=$_POST [‘name’]; /Защита от хостингов, не поддерживающих прямую передачу переменных из формы на сервер.
}
```

Перед тем как выбирать значение переменной $name из суперглобального массива $_POST, проверяется, существует ли это значение (существует ли на форме поле с именем **name).** Это делается с целью облегчить дальнейшую диагностику ошибки, если она обнаружится.

Функция unset ликвидирует существующую переменную.

```
<?php
 $a = "slovo1";
 $b = "slovo2";

 echo isset ($a); // выведет «1»
 echo isset ($b); // выведет «1»

 unset ($a);

 echo isset ($a); // не выведет ничего, потому что значением isset ($a) становится NULL
 echo isset ($b); // выведет «1»
?>
```

Пример 3

Выводить на экран текущее время и текущую дату в разных форматах позволяет функция **date ().**

В разделе 2.1 в листинге 2.2 используется оператор:

```
echo date ("H:i, d. m. Y"); // Вывод даты в формате: 23:15, 08. 05. 2012
```

Параметры функции имеют следующие значения:

- H — час, 24-часовой формат; т.е. от "00" до "23"
- i — минуты; т.е. от "00" до "59"
- d — день (число) месяца, 2 цифры с ведущим нулем, если необходимо; т.е. от "01" до "31"
- m — месяц; т.е. от "01" до "12"
- Y — год, 4 цифры; например, "1999"

Это был пример вывода в формате: чч:мм, дд.мм.гггг.

Напишем функцию для вывода только даты в формате «дд.мм.гг».

```
echo date ("d. m. y.");
```

Пример 4

Рассмотрим две функции, работающие с массивами.

Функция **count(Имя массива)** подсчитывает количество элементов массива.

```
$car=array (“жигули”,”волга”,”touareg ”,”antara”);
$n=count($car);
echo “количество моделей авто — $n”// выведет «количество моделей авто — 4»
```

Функция **foreach** служит для вывода содержимого массива. Синтаксис ее такой:

```
foreach (array as [$key =>] $value )
{
операторы;
}
```

при обходе элементов массива в переменную $key помещаются индексы элементов, а в переменную $value — значения элементов. Использование этой функции для нашего массива $car будет выглядеть так:

```
<?
foreach ($car as $index => $value )
{
echo “$index — $value <br>”;
}
?>
```

Результат:

0 — жигули
1 — волга
2 — touareg
3 — antara

Как видим, выдаются индексы элементов массива и значения элементов.

Пример 5

Функция **die (“сообщение”)** прекращает выполнение скрипта и выводит сообщение. Эта функция, как правило, используется

при работе с базами данных. Функции работы с БД, рассмотренные в разделе 3.5 (mysql_connect, mysql_select_db, mysql_query) очень критичны к параметрам. При любой ошибке в коде или сбое сервера возникает нештатная ситуация. Восстановление системы после этого может быть трудоемким процессом. Поэтому после выполнения этих функций желательно проверять результ и останавливать выполнение скрипта в случае ошибки. При этом есть возможность вывода сообщения, указанного в качестве параметра функции.

Вот примеры такого контроля ошибок, рекомендуемые в разделах 3.5.1 и 3.5.2:

```
$link = mysql_connect ("localhost", "alex", "12345");
if ( !$link) die ("Couldn't connect to MySQL");

mysql_select_db (“firstdb”, $link) or die ("Couldn't open firstdb");
```

Однако можно вывести на экран браузера более подробное сообщение об ошибке, которое может пригодиться при отладке программы. При любом неудачном завершении операции MySQL устанавливает номер ошибки и строку с ее описанием. Номер ошибки можно получить, используя функцию mysql_errno (), а строку с описанием — с помощью функции mysql_error().

```
<?php
$link = mysql_connect ("localhost", “student”, “12345”);
if ( ! $link ) die ("Couldn't connect to MySQL");
echo "Successfully connected to server";
mysql_select_db (“firstdb”, $link) or die ("Couldn't open $db: " .
mysql_error() );
print "Successfully selected database "firstdb";
mysql_close( $link );
?>
```

4.4.2. Пользовательские функции

В программах часто возникает необходимость повторять одни и те же вычисления или действия при различных значениях параметров. Для уменьшения размера программы эти вычисления

выделяются в отдельную подпрограмму. В основной программе при необходимости осуществляются обращение к подпрограмме. В языке Паскаль, как мы помним, в качестве подпрограмм используются функции и процедуры. Функции вычисляют одно единственное значение и присваивают его имени функции. Процедура может вычислить одно или несколько значений или выполнить одно или несколько действий. Причем процедура может заменить функцию всегда, но это часто нежелательно по нескольким причинам. Функция процедуру может заменить не всегда.

В PHP подпрограммы реализуются только в виде функций. Поэтому функции PHP кардинально отличаются от функций Паскаля.

Функция в PHP может быть объявлена и до обращения к ней, и после (в PHP 5).

Синтаксис объявления функции следующий:

```
Function имя_функции ([параметр1, параметр2, ])
{
тело функции

}
```

Здесь:

- Имя функции — правильный идентификатор, в этом случае он регистронезависимый.
- Параметр1, параметр2 — это формальные параметры, которые при обращении к функции заменяются фактическими параметрами (их также называют аргументами).
- В заголовке функции параметры могут отсутствовать.
- Тело функции — это любой верный код PHP.
- Тело функции может завершаться служебным словом return, если она должна вернуть вычисленное значение.

Разновидности объявления и использования функций рассмотрим на примере. Действия, выполняемые в теле функции, принципиального значения для нас не имеют. Поэтому рассмотрим функцию, выполняющую простейшее вычисление — сложение двух чисел и простейшее действие — вывод результата на Web-страницу.

Вариант 1.

Функция не возвращает значение. При обращении к функции вместо формальных параметров $a и $b указываются фактические параметры. Для обращения к этой функции ее надо записать как отдельный оператор.

```
<?
 function summa1($a, $b)
 {
 $s = $a + $b;
 echo $s;
 }
echo “Сумма чисел 5 и 3 будет:”;
summa1(5, 3); // выведет: «Сумма чисел 5 и 3 будет: 8»
?>
```

Вариант 2.

Функция не возвращает значение. Здесь до обращения к функции ее параметрам присвоим нужные значения, а при обращении к ней в качестве параметров используются не фактические, а формальные параметры.

Для обращения к этой функции ее надо записать как отдельный оператор.

```
<?
function summa2($a, $b)
{
 $s = $a + $b;
 echo $s;
}
$a=5;
$b=3;
echo “Сумма чисел $a и $b будет:”;
summa2($a, $b); // выведет: «Сумма чисел 5 и 3 будет: 8»
?>
```

Вариант 3.

Переделаем предыдущий вариант таким образом, чтобы функция только возвращала вычисленный результат. Для вывода результата тогда придется использовать оператор вывода вне функции.

```
<?
function summa3($a, $b)
```

```
{
 $s = $a + $b;
 return $s;
}
$a=5;
$b=3;
echo “Сумма чисел $a и $b будет:”;
echo(summa3($a, $b)); // выведет: «Сумма чисел 5 и 3 будет: 8»
?>
```

Вариант 4.

В объявлении функции (следовательно, и при обращении к ней) параметры могут отсутствовать. Это возможно, если не требуется передавать функции значения параметров из основной программы. Рассмотрим такой пример.

```
<?
function summa4()
{
 $a = 5;
 $b = 10;
 $s = $a + $b;
 return $s;
}
echo(summa4()); // выводит 15
?>
```

Вариант 5.

Функция не выполняет вычисления, а производит действие — выводит на Web-страницу гиперссылку. Это будет аналог процедуры Паскаля.

```
<?
Function href ($url, $pointer)
{
echo “<a href=’url’> $pointer </a>”;
}
href (‘http://kakoitosait.com’,’Нам нужно сюда ’); // В Web-страницу будет выведена гиперссылка ’Нам нужно сюда.
?>
```

Функции PHP — это очень мощный механизм. Остальные возможности функций мы оставим не рассмотренными, поскольку их количество достигает нескольких сотен.

5. ЛАБОРАТОРНЫЕ РАБОТЫ

Лабораторная работа № 1
Получение и обработка данных из форм клиента с помощью PHP-скриптов

Задание 1
Создание серверного Web-приложения «Обработка заказа на автозапчасти, полученного с формы клиента»

Изучив теоретический материал из раздела 2, создайте требуемое серверное Web-приложение.

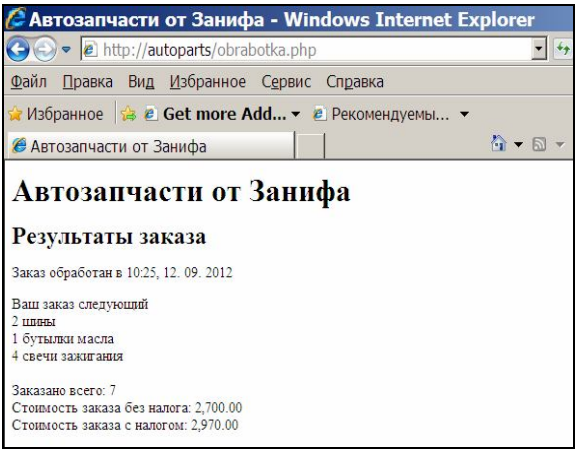
Обратившись по адресу http://autoparts к сайту магазина, мы получим форму для ввода заказа.

Запчасти от Занифа

Форма заказа

Товар	Количество
Шины	2
Масло	1
Свечи зажигания	4
Отправить заказ	

После обработки введенных нами в форму данных PHP-программа должна вернуть результат обработки на клиентскую машину в виде следующей Web-страницы.



Последовательность выполнения работы

1. Создание сайта начните с создания каталогов:

C:\webservers\home**familia_auto\www\.**

В каталог **www** в дальнейшем поместите созданные вами в редакторе Блокнот файлы:

- Index.htm — файл с формой заказа;
- Obrabotka.php — php-программа для обработки заказа.

2. Для вызова формы заказа:

- запустите сервер (ярлык StartServers на рабочем столе);
- запустите браузер;
- в адресной строке браузера наберите адрес: **http://familia_auto.** Откроется наша форма заказа из файла index.htm.

3. Заполните заказ, отправьте его. Изучите результат обработки, полученный от сервера.

4. В случае ошибок отладьте форму и/или PHP-программу.

5. «Подправьте» форму: слово «Товар» выровняйте по центру, а слово «Шины» — по левому краю.

6. Измените программу для случая, когда на сервере запрещено прямое использование данных, полученных с формы. Используйте суперглобальный массив $_POST (подраздел 2.2).

7. Усовершенствуйте программу для вывода результатов ее работы с единицами измерения выводимых величин:

Заказано всего: 7 единиц товара.

Стоимость заказа без налога: 2,700.00 руб.

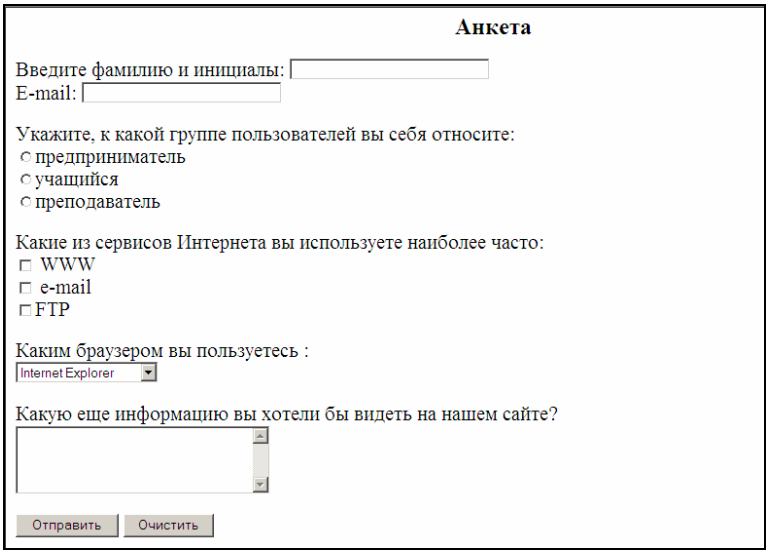
Стоимость заказа с налогом: 2,970.60 руб.

8. Усовершенствуйте программу для того, чтобы при пустых полях на форме в ответ выдавалось только сообщение «Вы ничего не заказали на предыдущей странице» (цвет текста должен быть красный).

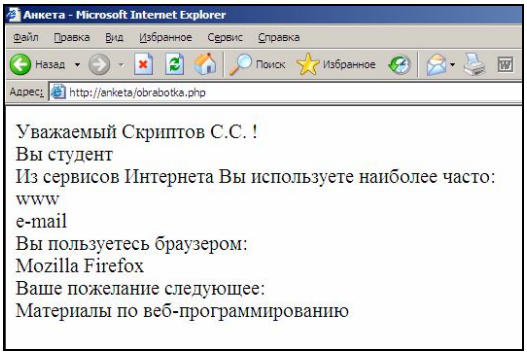
Задание 2
Передача данных серверной программе с разных элементов формы

На практике для передачи данных серверной программе с помощью формы используются не только рассмотренные в предыдущем задании поля типа text. Бывает потребность брать данные с полей типов: Radio, CheckBox, Select, Textarea.

Изучив теоретический материал из раздела 2.5, создайте серверное Web-приложение, выполняющее ввод и обработку анкетных данных.



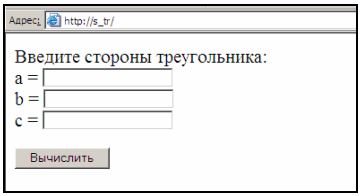
После обработки анкеты пользователь должен получить результат в виде следующей Web-страницы.



Задание 3
Проверка данных, введенных в форму

Изучив теоретический материал из раздела 2.5, создайте серверное Web-приложение, выполняющее следующие функции.

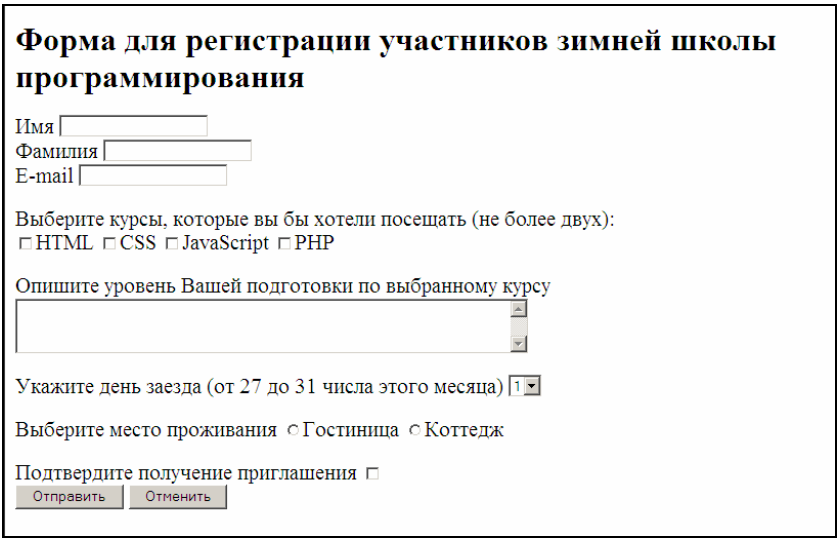
1. Вывод формы с запросом на ввод сторон треугольника.



2. После завершения ввода и нажатия кнопки типа Submit проверка на полноту ввода.
3. Проверка того, что в поля ввода введены только числа.
4. Проверка на корректность введенных значений сторон треугольника (должны соблюдаться известные из геометрии соотношения между длинами сторон треугольника).
5. В любом из предыдущих трех случаев неправильного ввода обеспечить повторный ввод данных в форму.
6. Вычисление площади треугольника по формуле Герона.

Задание 4

1. Создайте форму для регистрации участников зимней школы программирования.



2. Напишите программу-обработчик, которая выдаст информацию зарегистрированным участникам в следующем виде:

Уважаемый [Имя] [Фамилия]!
Ваш E-mail: [электронный адрес].
Вы выбрали для изучения курсы по: [названия/название курса].
Занятия по курсу [название курса] начнутся (01.01.2012, 02.01.2012, 03.01.2012, 04.01.2012 для HTML, CSS, JavaScript, PHP соответственно).
Уровень Вашей подготовки по выбранному курсу: [текст]
День Вашего заезда: [день].
Вы хотите жить в: [вариант места проживания].
Вы подтвердили / не подтвердили получение приглашения.

3. Усовершенствуйте программу для того, чтобы она проверяла, все ли необходимые данные введены в форму. При отсутствии данных надо выдать соответствующее сообщение красным цветом и вывести форму для повторного ввода данных.

4. Попробуйте устранить недостатки в дизайне формы. Старайтесь не увеличивать высоту формы.

Лабораторная работа № 2
Работа с базой данных MySQL в ручном режиме

Задание 1
Создание MySQL - БД с помощью программы PhpMyAdmin 2.3.0

Изучите теоретический материал из подразделов 3.1, 3.2, 3.3, 3.4.. С помощью графического интерфейса СУБД MySQL — программы PhpMyAdmin создайте БД **Firstdb** и таблицу с именем **Staff** (Сотрудники).

Пусть структура и начальное содержание БД будет такое.

id	name	lastname	dol
1	Алексей	Алексеев	директор
2	Иван	Иванов	бухгалтер
3	Александр	Александров	программист

Для облегчения и ускорения работы можно взять у преподавателя распечатку с инструкцией по созданию БД в PhpMyAdmin.

Задание 2
Выполнение операций по обработке данных из MySQL — базы данных в ручном режиме с помощью программы PhpMyAdmin 2.3.0

В PhpMyAdmin в режимах QBE (можно выполнить не все операции) и SQL выполните следующие 13 операций с данными из своей БД **Firstdb.**

Для удобства можно взять у преподавателя распечатку с пунктами задания.

1. Выведите на экран имя первого сотрудника.
2. Выведите на экран фамилию и должность первого сотрудника.
3. Выведите на экран имя второго сотрудника.
4. Выведите на экран фамилию и должность второго сотрудника.

5. Выведите на экран данные всех сотрудников.

6. Выведите на экран должность сотрудника с номером 2 и именем Александр.

7. Выведите на экран фамилию и должность Александра.

8. Выведите на экран данные всех сотрудников, отсортировав их по именам в алфавитном порядке.

9. Выведите на экран данные всех сотрудников, отсортировав их по именам в порядке, обратном алфавитному.

10. Выведите на экран данные первых двух сотрудников, отсортировав их по именам в алфавитном порядке.

11. Вставьте в таблицу данные Андрея Андреева, водителя.

12. Замените программиста Александрова на Егора Егорова.

13. Удалите из БД данных программиста Егорова.

14. Приведите таблицу staff в исходное состояние. Это нужно для выполнения следующего задания.

Задание 3
Экспорт/импорт данных в PhpMyAdmin

В ходе создания и отладки сайта на локальном сервере приходится неоднократно переносить базу данных с одного места в другое.

Изучите теоретический материал из подраздела 3.7 и выполните следующие задания по экспорту и импорту данных.

1. Перенесите дамп (содержимое) БД «firstdb» в БД «thirddb» на вашем локальном сервере. Проверьте содержимое полученной БД.

2. Сохраните дамп таблицы staff из БД «firstdb» в виде резервной копии в файле «staff.sql». Импортируйте таблицу из файла «staff.sql» в БД «thirddb», предварительно удалив данные в последней.

3. Попробуйте сохранить дамп таблицы staff из БД «firstdb» в виде резервной копии в файле «staff1.sql». Импортируйте таблицу из файла «staff1.sql» в БД «thirddb».

4. Попробуйте сохранить дамп таблицы staff из БД «firstdb» в виде резервной копии в файле «staff1.sql». Импортируйте таблицу из файла «staff1.sql» в БД «thirddb», предварительно удалив данные в последней.

5. Перенесите дамп БД «firstdb» на другую машину в БД «foreigndb».

6. Пользуясь командой **Operations** в верхнем меню phpMyAdmin, в БД firstdb создайте копию таблицы staff под именем staff1 и сохраните ее в БД firstdb.

7. Определите назначение других операций по работе с БД и таблицей, выполняемых в пункте верхнего меню **Operations.**

Лабораторная работа № 3
Программирование на PHP взаимодействия с СУБД MySQL

Задание 1

В этом задании надо будет выполнить все 13 операций по обработке данных из БД Firstdb (см. задание 2 к предыдущей лабораторной работе) с помощью PHP-скриптов.

Для выполнения пунктов задания необходимо пользоваться теоретическим материалом из разделов 3.4, 3.5.

Чтобы скрипт не получился очень большим, следовательно, плохо обозримым и трудно отлаживаемым, для выполнения операций по обработке данных из БД Firstdb мы напишем несколько скриптов и сохраним их в соответствующих файлах.

Перед запуском программ не забудьте запустить сервер.

Для запуска программ в адресной строке браузера надо будет набрать адрес: **localhost/familia_mysql/имя_файла.**

Порядок работы

1. Создайте каталог: C:\webservers\home\localhost\www**familia_mysql\.**

В этот каталог в дальнейшем будете помещать создаваемые вами в редакторе Блокнот файлы:

1) Mysql_php.php — php-программа (полигон) для экспериментов по выборке данных из БД **firstdb;**

2) Mysql_table.php — сценарий для отображения содержимого таблицы staff в виде HTML-таблицы;

3) Mysql_insert_1.php — php-программа для экспериментов по вставке данных в таблицу;

4) Mysql_form.php — форма для ввода данных в таблицу staff;

5) Mysql_insert_2.php — php-программа для ввода данных в таблицу через форму;

6) Mysql_update.php — php — сценарий для обновления данных в таблице;

7) Mysql_delete.php — php — сценарий для удаления записей из БД.

2. Создайте в редакторе Блокнот файл-полигон под именем Mysql_php.php для экспериментов по выборке данных из таблицы **staff** БД **firstdb** (подключение к серверу, выбор БД, выборка данных из таблицы, вывод данных на экран).

Шаблон для этого и остальных 6 файлов с PHP-сценариями может быть следующий:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Работа с БД </title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
</head>
<body>
<?

?>
</body>
</html>
```

PHP-операторы надо будет вставлять в тег <? ?>

Создайте и сохраните все 7 файлов-шаблонов в своем каталоге по адресу:

C:\webservers\home\localhost\www\familia_mysql\.

3. Впишите в файл Mysql_php.php PHP-оператор для выполнения подключения к серверу MySql (п. 3.5.1).

4. Впишите туда же оператор для выбора БД (п. 3.5.2).

5. Программируйте выборку данных из БД и их вывод на экран по п. 3.5.3. Проверяйте работу программы после каждой модификации.

5.1. Выведите на экран имя первого сотрудника.

Для выполнения этой операции вы должны написать скрипт:

```
<?
$db=mysql_connect("localhost",'student',12345);
mysql_select_db("firstdb",$db);
$result=mysql_query("SELECT id, name FROM staff WHERE id='1' ", $db);//1
echo '1. Выведите на экран имя первого сотрудника'.'<BR>';
$myrow=mysql_fetch_array($result);
echo $myrow['id']."<BR>";
echo $myrow['name']."<p>";
?>
```

5.2. Выведите на экран фамилию и должность первого сотрудника.

5.3. Выведите на экран имя второго сотрудника.

5.4. Выведите на экран фамилию и должность второго сотрудника.

5.5. Выведите на экран данные всех сотрудников.

5.6. Выведите на экран должность сотрудника с номером 2 и именем Александр.

5.7. Выведите на экран фамилию и должность Александра.

5.8. Выведите на экран данные всех сотрудников, отсортировав их по именам в алфавитном порядке.

5.9. Выведите на экран данные всех сотрудников, отсортировав их по именам в порядке, обратном алфавитному.

5.10. Выведите на экран данные первых двух сотрудников, отсортировав их по именам в алфавитном порядке.

6. Запрограммируйте отображение данных из БД в виде таблицы (п. 3.5.4). Сохраните файл под именем Mysql_table.php. Отобразите содержимое таблицы **staff.**

7. Запрограммируйте вставку записи в таблицу (п. 3.5.5). Вставьте в таблицу данные Андрея Андреева, водителя. Сохраните файл под именем Mysql_insert_1.php. Проверьте результат путем отображения на экране содержимого таблицы staff с помощью программы Mysql_table.php и программы PhpMyAdmin.

8. Запрограммируйте ввод данных в таблицу через форму (п. 3.5.6). Код формы сохраните в файле под именем Mysql_form.php. Код обработчика формы сохраните в файле Mysql_insert_2.php.

Используя созданную форму, добавьте в таблицу данные двух произвольных сотрудников и отобразите содержимое таблицы с помощью программы Mysql_table.php.

9. Запрограммируйте замену программиста Александрова на Егора Егорова. Сохраните файл под именем Mysql_update.php. Проверьте результат путем отображения на экране содержимого таблицы staff с помощью программы Mysql_table.php.

10. Запрограммируйте удаление из БД данных программиста Егорова. Сохраните файл под именем Mysql_deletete.php. Проверьте результат путем отображения на экране содержимого таблицы staff с помощью программы Mysql_table.php.

11. Пользуясь материалом из пунктов 3.5.1 и 3.5.2, используйте функции mysql_connect и mysql_select_db с контролем ошибки. Проверьте работу программ mysql_table.php и mysql_php.php при наличии в них ошибок в имени базы данных.

Задание 2
Создание MySQL - БД путем программирования в PHP

Изучите теоретический материал из подраздела 3.6 и запрограммируйте создание БД **seconddb** и таблицы с именем **Staff** (разделы 3.2, 3.3). То есть базу данных firstdb, которую в лабораторной работе № 2 создавали вручную в PhpMyAdmin, создадим путем программирования в PHP.

Задание 3
Создание PHP+MySql-приложения «Библиотека»

Вы владелец Internet-библиотеки, содержащей 10 книг. Для каждой книги указаны название, автор, издательство, серия и уникальный код (ISBN). В приложении необходимо обеспечить:

- возможность добавления и удаления книг в библиотеке (для администратора);
- возможность поиска по всем перечисленным атрибутам книги (для пользователя).

В качестве хранилища данных о книгах использовать базу данных MySQL.

ЗАКЛЮЧЕНИЕ

Завершив изложение материала учебного пособия, оценим, насколько результат отличается от предварительного плана.

Ответ на этот вопрос надо начать с констатации того, что инструментарий современного Web-программирования, в том числе и серверного Web-программирования, стал очень громоздким. Этот инструментарий повышает производительность труда Web-разработчика, позволяет повысить качество Web-продукта, но не уменьшает трудоемкость изучения этого инструментария.

Чтобы устранить противоречие между объемом материала, подлежащего изложению, с одной стороны, и возможностями студента и отводимым данной дисциплине временем, с другой стороны, автор использовал так называемый «закон Парето»: «Чтобы решить 80% задач, встречающихся на практике, достаточно знать 20% инструментов».

В результате получилось учебное пособие, которое можно успешно, эффективно использовать на лабораторных занятиях и при подготовке к экзамену. Это показал опыт использования электронного варианта учебного пособия (неполного) в течение двух семестров.

Однако надо признать, что для выполнения дипломного проектирования, тем более для профессиональной работы, требуется знать и уметь значительно больше, чем дается в этом пособии.

Но надо признать и то, что студент, изучивший серверное Web-программирование по этому учебному пособию, сможет без лишних усилий и стрессов изучать материал, предназначенный для профессиональной работы в области серверного Web-программирования. Автор частично в этом уже убедился.

ОБЗОР ТЕРМИНОВ И ПОНЯТИЙ ИЗ ОБЛАСТИ СЕРВЕРНОГО WEB-ПРОГРАММИРОВАНИЯ

Введение в Web-программирование

Web-программирование — разработка любых программных продуктов, предназначенных для работы в World Wide Web.

Разновидности Web-программирования — разработка Web-страниц на чистом HTML, использование на Web-страницах технологий JavaScript и VBScript, создание CGI-приложений.

Разработка web-страниц на чистом HTML — разновидность web-программирования, поскольку при просмотре страницы браузер фактически исполняет код HTML, форматируя текст согласно инструкциям этого языка. В профессиональных сайтах, как правило, для оформления используются каскадные таблицы стилей (CSS cascading style sheets). Поэтому их использование тоже надо отнести к Web-программированию.

Создание CGI-приложений (CGI — сокращение от Common Gateway Interface) — технология, позволяющая запускать на web-сервере программы, имеющие возможность получать данные от посетителей сайтов, поддерживаемых этим Web-сервером, и в свою очередь выдавать им обработанные данные в виде Web-страниц или других файлов.

Web-сервер — программа, устанавливаемая на узле сети Интернет и выдающая посетителям этого узла Web-страницы по запросам. Также Web-сервером часто называется узел, на котором эта программа запущена, или даже компьютер, являющийся таким узлом.

Web-страница — документ, хранящий информацию в виде текстовых файлов. Помимо текста, такие страницы могут содержать ссылки на другие страницы (расположенные на том же самом или другом сервере Web), ссылки на графические изображения, аудио и видеоинформацию, различные объекты ввода данных

(поля, кнопки на формах), а также другие объекты. Фактически страницы Web представляют собой контейнер для объектов различных типов. Их проектируют с применением специального языка разметки гипертекстов Hyper Text Markup Language, или сокращенно — HTML.

Сайт — совокупность Web-страниц, связанных между собой гиперссылками, находящихся в одном каталоге и принадлежащих одному пользователю или относящихся к одной теме. Одна из страниц сайта назначается главной (титульной). Это обычно осуществляется путем присвоения файлу с этой страницей имени **index.htm**.

Язык серверного Web-программирования PHP

Синтаксис языка PHP — правила конструирования операторов программы и требования к оформлению программ. Как и в других языках программирования, синтаксис языка PHP включает:

- алфавит используемых символов;
- ключевые (служебные) слова;
- правила построения идентификаторов;
- используемые в языке операции над данными;
- правила построения выражений;
- правила построения и включения в скрипт комментариев.

Типы данных PHP

PHP поддерживает восемь типов данных.

Четыре скалярных типа:

- boolean (логический);
- integer (целый);
- float (с плавающей точкой);
- string (строковый).

Два смешанных типа:

- array (массив);
- object (объект).

И два специальных типа:

- resource (ресурс);
- NULL.

PHP-операторы — действия, которые должен выполнить PHP-интерпретатор. PHP-операторы помещаются между открывающим и закрывающим дескрипторами.

Функция — фрагмент исходного текста, оформленный по соответствующим правилам и который можно вызывать по мере необходимости из различных мест сценария PHP.

Массив — именованный участок памяти, в которой хранится набор значений, что позволяет группировать обычные скалярные значения.

Ассоциативные массивы — это массивы, в которых в качестве индексов могут использоваться практически любые значения, но, как правило, таковыми являются строки.

Инициализация массива — формирование массива путем ввода определенным способом его элементов.

Система управления базами данных MySQL

Сервер баз данных — программа, обслуживающая процессы работы с базой данных.

Реляционная база данных — база данных, в которой данные представлены в виде одной или многих таблиц.

Индекс — отсортированный список значений одного поля таблицы, предназначенный для ускорения поиска в БД.

Уникальный индекс — список значений, в котором каждое значение уникально.

Первичный ключ — уникальный индекс, используемый для однозначной идентификации записей таблицы.

Логический (естественный) ключ — то поле таблицы, которое естественным образом определяет записи.

Суррогатный ключ — дополнительное поле, вводимое в таблицу с данными для однозначной идентификации записей таблицы.

Нормализация БД — устранение избыточности в таблицах путем выделения из нее нескольких таблиц, соответствующих отдельным сущностям.

Программирование взаимодействия PHP и MySQL

Сетевая база данных — это БД, хранящаяся на сервере. По запросам клиентов серверные приложения выбирают из БД необходимые данные, вставляют их в Web-страницы и отправляют на клиентскую машину.

Сеанс обмена PHP-программы с сервером баз данных — процесс, включающий следующие этапы, реализуемые встроенными функциями серверного языка программирования PHP:

- установление соединения с сервером MySQL;
- выбор базы данных;
- формирование и отправка серверу запросов;
- помещение содержимого таблицы базы данных в матрицу;
- выборка из матрицы строк (записей) таблицы;
- доступ к отдельному полю записи;
- разрыв соединения с сервером.

ТЕСТОВЫЕ ЗАДАНИЯ

Фонд тестовых заданий для оценки знаний по дисциплине (КИМ по дисциплине)

Контрольно-тестирующий комплекс содержит 40 тестовых заданий по всем разделам курса и обеспечивает проведение следующих видов контроля знаний студентов:

- определение исходного уровня подготовки (тестирование по 40 заданиям перед началом преподавания дисциплины);
- самоконтроль;
- проверка готовности перехода на другую тему (тестирование на заданиях из этого раздела после завершения работы с очередным разделом курса или несколькими маленькими разделами);
- промежуточный и итоговый контроль.

Критерии для итогового контроля

Количество баллов	Оценка
< 25	2
=> 25	3
=> 30	4
=> 35	5

КОНТРОЛИРУЮЩИЙ КУРС/ТЕСТ ПО РАЗДЕЛУ
"Серверное Web-программирование"
(кафедра И и МПИ НГГУ)
Разработчик Хакимов Р.Х.

Структура аттестационного педагогического измерительного материала (АПИМ):

Номер раз-дела	Наименование темы	Коли-чество ТЗ по теме	Критерий освоения раздела
1	Введение в Web-программирование	3	3
3	Работа с сервером баз данных MySQL	6	6
4	Серверное Web-программирование. Язык серверного Web-программирования PHP		
4.1	Элементы языка (Синтаксис языка) PHP	9	8
4.2	Типы данных	5	5
4.3	Операторы	6	6
4.4	Функции в PHP	5	5
	Итого	40	

Тестовые задания

1. Введение в Web-программирование

1. Web-программирование это:
☐ разработка любых программных продуктов, предназначенных для работы на сайтах World Wide Web
☐ создание Web-страниц на языке HTML
☐ использование на Web-страницах технологии CSS

2. Разновидности Web-программирования это:
☐ разработка Web-страниц на чистом HTML
☐ использование на web-страницах технологий JavaScript и VBScript
☐ создание CGI-приложений
☐ создание BGI-приложений

3. Web-сервер это:

☐ программа, устанавливаемая на узле сети Интернет и выдающая посетителям этого узла Web-страницы по запросам

☐ узел, на котором работает программа, выдающая клиентам Web-страницы по запросам

☐ компьютер, на котором работает программа, выдающая клиентам Web-страницы по запросам

3. Работа с СУБД MySQL

1. Сервер баз данных MySQL обслуживает базы данных типа:

☐ иерархический

☐ реляционный

☐ сетевой

☐ объектно-ориентированный

2. Установите соответствие между терминами и их содержанием

Индекс	Отсортированный список значений поля таблицы, предназначенный для ускорения поиска в БД
Уникальный индекс	Список значений, в котором каждое значение уникально
Первичный ключ	Уникальный индекс, используемый для уникальной идентификации записей таблицы
Логический (естественный) ключ	То поле таблицы, которое естественным образом определяет записи
Суррогатный ключ	Дополнительное поле, вводимое в таблицу с данными для уникальной идентификации записей таблицы
Нормализация БД	Устранение избыточности в генеральной таблице путем выделения из нее нескольких таблиц, соответствующих отдельным сущностям

3. Сервер баз данных MySQL более других СУБД подходит для работы с Web-приложениями по причине:

☐ быстродействия

☐ приспособленности для работы с очень большими БД

☐ отсутствия транзакций

4. Установите соответствие между обозначениями типов полей БД и их описанием

INT	Целые числа
DATE	Дата в формате ГГГГ-ММ-ДД
TIME	Время в формате ЧЧ-ММ-СС
CHAR	Строки с длиной до 255 символов
TEXT	Строки с длиной до 65 535
BLOB	Двоичные данные длиной до 65 535 кБит

5. Установите соответствие между названиями команд MySQL и их действиями

CREATE DATABASE	Создает новую БД
CREATE TABLE	Создает новую таблицу в выбранной БД
DESCRIBE	Показывает структуру БД
ALTER TABLE	Позволяет изменить структуру БД
DROP TABLE	Удаляет одну или несколько таблиц
DROP DATABASE	Удаляет БД со всеми таблицами
INSERT INTO…VALUES	Вставляет новые записи в существующую таблицу
DELETE	Удаляет из таблицы записи, удовлетворяющие заданным параметрам
SELECT	Извлекает строки данных из одной или нескольких таблиц
UPDATE	Обновляет столбцы таблицы
SHOW	Показывает, какие базы данных имеются в системе

6. Установите соответствие между названиями функций MySQL и их действиями

mysql_connect (server, username, password);	для установления соединения с сервером MySQL
mysql_select_db (dbname, $db);	выбор базы данных
mysql_query (query)	для отправки серверу запросов
mysql_fetch_array ($result)	для вывода записей из базы данных
mysql_close ($db);	разрыв соединения с сервером

4. Серверное Web-программирование. Язык серверного Web-программирования PHP

4.1. Синтаксис языка

1. Чему равен результат операции 5%2?
###

2. В результате выполнения скрипта на экран будет выведено ###.
```
<?
$var=1;
echo(++$var);
?>
```

3. В результате выполнения скрипта на экран будет выведено ###.
```
<?
$var=1;
echo($var++);
?>
```

4. В результат выполнения скрипта на экран будет выведено ###.
```
<?
$var=”aaa”;
echo($var++);
?>
```

5. echo (4<<2) = ###

6. echo (5>>1) = ###

7. echo (6|5) = ###

8. echo (6&5) = ###

9. Правильно ли оформлен комментарий?
```
<?
echo(“Hello”); //Это комментарий
?>
```
☐ да
☐ нет

4.2. Типы данных

1. Отметьте типы данных PHP:
☐ integer
☐ string
☐ float
☐ array
☐ boolean
☐ object

2. В результате выполнения скрипта на экран будет выведено ###.
```
<?
$var_str=”5”;
echo(gettype ($var_str));
?>
```

3. В результате выполнения скрипта на экран будет выведено ###.
```
<?
$var_num=5;
echo(gettype ($var_num));
?>
```

4. В результате выполнения скрипта на экран будет выведено ###.
```
<?
$var=”5”;
settype ($var, integer);
echo (gettype ($var));
?>
```

5. В результате выполнения скрипта на экран будет выведено ###.
```
<?
$var=”5”;
$var = (int) $var;
echo (gettype ($var));
?>
```

6. Структурами для хранения данных являются:
☐ массивы

☐ стек
☐ очередь
☐ связный список
☐ дек

7. Стек работает по принципу:
☐ LIFO
☐ FIFO

8. Очередь работает по принципу:
☐ LIFO
☐ FIFO

9. Результатом выполнения скрипта будет ###.
```
<?
$car[] = “автобус”;
$car[] = “грузовик”;
echo ($car[1]);
?>
```

10. Результатом выполнения скрипта будет ###.
```
<?
$car = array(“автобус”, “грузовик”);
echo ($car[1]);
?>
```

11. Результатом выполнения скрипта будет ###.
```
<?
$car = array(“pc”=>“автобус”, “lc”=>“грузовик”);
echo ($car[“lr”]);
?>
```

4.3. Операторы

1. В результат выполнения скрипта
```
<?
$flag1 = true;
$flag2= true;
```

```
$flag3= false;
if ($flag1 && $flag2 || $flag3)
{
 echo “<p>Условие истинно</p>”;
}
else
{
 echo “<p>Условие ложно </p>”;
}
?>
```

на экран будет выведено:
☐ Условие истинно
☐ Условие ложно

2. Результат условной операции x <0 ? —x: x;
☐ x
☐ -x
☐ абсолютное значение x

3. Правильно ли дана структура оператора Switch?

```
Switch (expression)
{
 case value1:
 statements;
 break;
 case value2:
 statements;
 break;
 default:
 statements;
}
```

☐ да
☐ нет

4. В результате работы скрипта будет выведено ### чисел.

```
<?
$a=0;
```

```
while ($a<=10)
{
echo "$a<br>";
$a=$a+1;
}
?>
```

5. В результате работы скрипта будет выведено число ###.

```
<?
$sum=0;
$i=1;
do
{
$sum=$sum+$i;
$i=$i+1;
}
while($i<=5);
echo $sum;
?>
```

6. В результате работы скрипта будет выведено ### чисел.

```
<?
$var = 5;
for ($i = 0; $i<=$var; $i++)
{
echo ($i);
echo(‘<br>’);
}
?>
```

4.4. Функции

1. В результате работы скрипта будет выведено число ###.

```
<?
Function get_sum()
{
$var = 5;
$var1 = 10;
```

```
$sum = $var + $var1;
return $sum;
}
echo (get_sum();
?>
```

2. В результате работы скрипта будет выведено число ###.

```
<?
Function get_sum()
{
$var = 5;
$var1 = 10;
$sum = $var + $var1;
echo ($sum);
}
get_sum();
?>
```

3. В результате работы скрипта будет выведено число ###.

```
<?
Function get_sum($var, $var1)
{
$sum = $var + $var1;
echo ($sum);
}
get_sum(5, 2);
?>
```

4. В результате работы скрипта будет выведено число ###.

```
<?
Function get_sum()
{
$var = 5;
echo ($var);
}
$var =10;
get_sum();
?>
```

5. В результате работы скрипта будет выведено число ###.

```
<?
Function get_sum()
{
global $var;
$var = 5;
echo ($var);
}
$var =10;
get_sum();
?>
```

ЛИТЕРАТУРА

1. Когзолл Дж. PHP5. Полное руководство. М., 2006.
2. Колисниченко Д.Н. Современный сайт на PHP b JavaScript. СПб., 2009.
3. Кравец О.Я., Солдатов Е.А., Селиванова М.В. Практикум по разработке Интернет-приложений: Учеб. пособие. Изд. 2-е, перераб. и доп. Уфа, 2008.
4. Кузнецов М.В., Симдянов И.В. Самоучитель PHP 5. СПб., 2004.
5. Кухарчик А. PHP: обучение на примерах. Минск, 2004.
6. Мазуркевич А., Еловой Д. PHP: настольная книга программиста. М., 2004.
7. Руководство по PHP. URL: http://www.php.net/manual/ru/
8. Савельева Н.В. Основы программирования на PHP: Учеб. пособие для студ. вузов. М., 2005.
9. Томсон Л., Веллинг Л. Разработка Web-приложений на PHP и MySQL. 2-е изд., испр. СПб., 2003.
10. Фролов А.В., Фролов Г.В. Базы данных в Интернете: практическое руководство по созданию Web-приложений с базами данных. Изд. 2-е, испр. М., 2000.
11. Шапошников И.В. PHP 5.1. Учебный курс. СПб., 2007.

Интернет-ресурсы

1. PHP (www.php.net)
2. Web-сервер Apache (www.apache.org)
3. Сервер баз данных MySQL (www.mysql.com)
4. Денвер (denwer.ru)

ОГЛАВЛЕНИЕ

Предисловие ... 3

1. Введение в серверное Web-программирование ... 8
   1.1. Пассивные и активные серверы Web ... 8
   1.2. Программы CGI, схема их работы ... 9
   1.3. Язык создания CGI-сценариев — PHP ... 11

2. Обработка на сервере запросов с клиентской машины ... 13
   2.1. PHP-сценарий обработки на сервере заказа на автозапчасти, отправленного с клиентской машины ... 13
   2.2. Особенности синтаксиса языка PHP ... 18
   2.3. Методы передачи информации в серверную PHP-программу ... 19
      2.3.1. Метод POST ... 20
      2.3.2. Метод GET ... 21
   2.4. Варианты передачи данных серверной программе с разных элементов формы ... 23
   2.5. Проверка данных, введенных в форму ... 28
   2.6. Разработка и отладка локального сайта с помощью локального сервера ... 31
      2.6.1. Система Денвер ... 31
      2.6.2. Установка Web-сервера на локальном компьютере ... 32
      2.6.3. Структура папок локального Web-сервера ... 32
      2.6.4. Порядок создания интерактивного сайта ... 35

3. Работа с сервером баз данных MySQL ... 37
   3.1. Общая характеристика MySQL ... 37
   3.2. Описание задания по созданию базы данных в MySQL ... 38
   3.3. Методика создания MySQL — БД с помощью программы phpMyAdmin 3.2.3 ... 38
   3.4. Основные SQL-операторы для работы с БД ... 40
      3.4.1. Создание базы данных ... 40
      3.4.2. Создание таблицы ... 41
      3.4.3. Выборка записей ... 41
      3.4.4. Вставка записей ... 42
      3.4.5. Обновление записей ... 42
      3.4.6. Удаление записей ... 42

3.5. Функции PHP для работы с СУБД MySQL. Программирование взаимодействия с БД 43
3.5.1. Подключение к серверу баз данных 43
3.5.2. Выбор базы данных 44
3.5.3. Выборка данных из таблицы 45
3.5.4. Отображение данных таблицы БД в виде таблицы на Web-странице 47
3.5.5. Вставка записи в таблицу 49
3.5.6. Добавление записей в базу данных через форму 49
3.5.7. Обновление записей 52
3.5.8. Удаление записей 52
3.5.9. Завершение MySQL-соединения 53
3.6. Методика создания MySQL-базы данных путем программирования в PHP 53
3.7. Экспорт/импорт данных в PhpMyAdmin 55

4. Язык серверного Web-программирования PHP 63
4.1. Элементы языка (Синтаксис языка) PHP 66
4.1.1. Алфавит 66
4.1.2. Служебные слова 66
4.1.3. Идентификаторы 67
4.1.4. Данные 67
4.1.5. Операции 68
4.1.6. Выражения 70
4.2. Типы данных 71
4.2.1. Тип boolean (логический тип) 71
4.2.2. Тип integer (целые) 72
4.2.3. Тип float (числа с плавающей точкой) 73
4.2.4. Тип string (строка) 73
4.2.5. Тип array (массив) 76
4.2.6. Тип object (объекты) 79
4.2.7. Тип resource (ресурсы) 79
4.2.8. Тип null 80
4.3. Операторы 80
4.3.1. Операторы вывода 80
4.3.2. Оператор присвоения 80
4.3.3. Условный оператор 81
4.3.4. Оператор switch 84

4.3.5. Операторы цикла 84
4.3.6. Оператор include (включить) 87
4.4. Функции в PHP..... 89
4.4.1. Встроенные функции PHP..... 89
4.4.2. Пользовательские функции 94

5. Лабораторные работы 98
Лабораторная работа № 1. Получение и обработка данных из форм клиента с помощью PHP-скриптов..... 98
Лабораторная работа № 2. Работа с базой данных MySQL в ручном режиме.....103
Лабораторная работа № 3. Программирование на PHP взаимодействия с СУБД MySQL.....105

Заключение 109

Обзор терминов и понятий из области серверного Web-программирования.....110

Тестовые задания.....114

Литература 125