МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
(ДГТУ)

Кафедра «Вычислительные системы и информационная безопасность»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению лабораторных работ по дисциплине
«Интернет-технологии»
для обучающихся 3 курса направления подготовки
10.03.01 «Информационная безопасность»

Ростов-на-Дону
ДГТУ
2019

УДК 681.3

Составители: Н.Д. Панасенко, В.В. Галушка

Методические указания по выполнению лабораторных работ по дисциплине «Интернет-технологии» для обучающихся 3 курса направления подготовки 10.03.01 «Информационная безопасность» – ДГТУ, Ростов-на-Дону, 2019. – 66 с.

Представляют собой комплекс рекомендаций и разъяснений, позволяющих оптимальным образом организовать процесс изучения дисциплины.

УДК 681.3

Печатается по решению редакционно-издательского совета
Донского государственного технического университета

Ответственный за выпуск
зав. кафедрой «Вычислительные системы и информационная безопасность»
д-р техн. наук, профессор В.А. Фатхи

В печать 08.02.2019 г.
Формат 60×84/16. Объем усл. п. л.
Тираж 30 экз. Заказ № 96.

Издательский центр ДГТУ
Адрес университета и полиграфического предприятия:
344000, г. Ростов-на-Дону, пл. Гагарина, 1

Оглавление

Лабораторная работа № 1

Язык гипертекстовой разметки HTML

Цель работы: изучить основные теги и атрибуты языка HTML и методы их применения для форматирования текста.

Краткие теоретические сведения

Электронные документы предназначены для просмотра преимущественно с экрана монитора. Фундаментальная особенность электронных документов состоит в том, что посредством ссылки можно перейти прямо к тому тексту, на который она указывает. Такие ссылки могут указывать не только на тексты, но и на графическую, звуковую, видеоинформацию. Поэтому они получили название *гиперссылок*, а документы, формируемые с помощью языка гипертекстовой разметки HTML (Hypertext Markup Language), стали называть *гипертекстовыми*.

HTML-документы являются текстовыми файлами с расширением htm или html и могут создаваться с помощью любого текстового редактора, например, стандартной программы Windows “Блокнот”. Для просмотра HTML документа в отформатированном виде предназначены Web-браузеры (например, Microsoft Internet Explorer, Netscape Navigator, Opera).

Язык HTML предназначен для описания форматированного электронного документа с помощью специальных управляющих кодов – *тегов* (от англ. tag – ярлык). Язык HTML является нечувствительным к регистру.

Теги бывают двух типов: открывающие и закрывающие. Большинство открывающих тегов имеет парный закрывающий тег. *Тег* состоит из следующих друг за другом в определенном порядке элементов:

– левой угловой скобки <(такого же, как символ "меньше чем");

– слэша / (только для закрывающего тега)*;*

– имени тега, например, **TITLE** или **PRE**;

– необязательных атрибутов. Тег может быть без атрибутов или сопровождаться одним, или несколькими атрибутами, например, *ALIGN*=” **CENTER**”;

– правой угловой скобки> (такой же, как символ "больше чем").

Часть HTML-документа от открывающего тега до парного ему закрывающего тега образуют блок, называемый *контейнером*. Непарные теги, например, **<HR>** – горизонтальная линия, являются контейнерами сами по себе, и не имеют закрывающего тега.

Общая структура HTML–документа имеет вид:

```
<! DOCTYPE html>
     <html><head><title>Мой первый документ</title></head>
     <body>
          Hello, world!
     </body>
     </html>
```

Рассмотрим входящие в этот пример контейнеры. Весь документ является контейнером **<html>…</html>**, в который включены два контейнера **<head>…</head>** – заголовок документа и **<body>…</body>** – тело документа, которое будет отображаться в клиентском окне Web-браузера.

Существуют теги, которые могут использоваться только в заголовке или только в теле документа. Например, контейнер **<title>…</title>** может быть указан только один раз и только внутри контейнера **<head>**, также, как и тег **<meta>**, который не имеет парного закрывающего тега, но может быть указан несколько раз. Тег **<meta>** применяется для указания общей информации о документе (автор, ключевые слова, кодировка, описание документа). Например: <meta name=“author” content=“Ivan Ivanov”>

<meta http-equiv=“Content-Type” content=“text/html; charset=windows-1251”>

Первый тег сообщает о том, что автором документа является Ivan Ivanov, второй сообщает браузеру о том, что документ набран в кодировке, применяемой операционной системой Windows для кириллицы. Из этих примеров видно, что данные теги содержат атрибуты name, *http-equiv* и *content*. Большинство тегов допускает один или несколько атрибутов, однако атрибутов может не быть. Закрывающие теги не имеют атрибутов.

Спецификация атрибута состоит из расположенных в следующем порядке имени атрибута, например, **WIDTH**, знака равенства (=), значения атрибута, которое задается строкой символов, например, “**80”**. Всегда полезно заключить значение атрибута в одинарные ('**80**'), либо двойные кавычки (“**80”**). Строка в кавычках не должна содержать такие же кавычки внутри себя. Так, если дата заключена в двойные кавычки, используйте одинарные кавычки для последующего заключения в кавычки, и наоборот.

Атрибуты тега <body>:

1) bgcolor=“<цвет>” – определяет цвет фона документа. Цвет задается в виде названия или кода (например, “red”, “FF0000”).

2) background=“ <имя файла>” – указывает браузеру местоположения файла с рисунком, который необходимо использовать в качестве фонового. Окно полностью заполняется изображением, начиная с верхнего левого угла;

3) ***text***=“<цвет>” – устанавливает цвет текста;

4) ***alink***=“<цвет>” – определяет цвет активной ссылки;

5) ***link***=“<цвет>” – определяет цвет не просмотренной ссылки;

6) ***vlink***=“<цвет>” – определяет цвет просмотренной ссылки;

7) ***topmargin***=“<расстояние>” – устанавливает границу верхнего поля;

8) ***leftmargin***=“<расстояние>” – граница левого поля.

Цвет задается в виде названия (“**red**”) или шестнадцатеричного числа (“**ff0000**”), а расстояние в пикселах (“**10**”).

Форматирование текста. Для того чтобы в тексте выделить абзац необходимо ограничить его тегами ** **…** **. У этого тега большое количество различных атрибутов, наиболее важным из которых является атрибут *align*. Он

может принимать значения ***left*** (выравнивание по левому краю), ***right*** (по правому краю), ***center*** (по центру), ***justify*** (по ширине).

Теги для форматирования текста:

1) **…** – выделение текста курсивом;

2) **…** – выделение текста полужирным шрифтом;

3) **<u>…</u>** – выделение текста подчеркиванием;

4) **[…]** – текст отображается верхним индексом;

5) **_…** – текст отображается нижним индексом;

6) **
…
** – формирование нумерованного списка, каждый пункт списка заключается в контейнер **
…
**;

7) **
…
** – для создания маркированного списка (вместо **<ol>**);

8) **
** – непарный тег, вставляет "жесткий" разрыв строки;

9) ** … ** – применяется для центрирования блока;

10) **<h1> …</h1>** – форматирует текст как заголовок 1-го уровня (всего 6 уровней **<h2> ...</h2> … <h6> ...</h6>**);

11) **<multicol cols="…"> …</multicol>** – выводит текст в несколько колонок (атрибут **cols** задает их количество);

12) **<pre>…</pre>** – выводит текст моноширинным шрифтом с сохранением форматирования;

13) **<strike>… </ strike>** – перечеркивает текст горизонтальной линией.

Размер шрифта. Язык гипертекстовой разметки позволяет задавать размеры шрифта, которым будет отображаться текст документа. По умолчанию размер шрифта равен "**3**", однако, используя, тег **** с атрибутом ***size*****="<**размер**>"**, его можно переопределить.

Относительно базового размера можно указывать размер шрифта текста, который заключен в контейнере **…**. Для этого необходимо присвоить атрибуту ***size*** одно из следующих значений: (**+1…+7**) – увеличение шрифта на указанное количество единиц относительно базового размера;

(**–1…–7**) – уменьшение шрифта на указанное количество единиц относительно базового размера.

Также могут использоваться теги:

1) **<big>… </big>** – выводит текст шрифтом большего размера;

2) **<small>…</small>** – выводит текст шрифтом меньшего размера.

Таблицы. Таблицы являются не только средством представления данных. Для языка HTML более важно их применение в качестве средства форматирования. Таблицы описываются тегами **<table>…</table>**.

Атрибуты тега **<table>**: ***border*** – толщина рамки таблицы в пикселях (**0** – рамки нет); ***bordercolor*** – цвет рамки; ***bgcolor*** – цвет фона таблицы; ***width*** – ширина таблицы (в пикселах или в процентах от ширины контейнера); ***align*** – выравнивание таблицы внутри родительского контейнера; ***cols*** – количество столбцов; ***rows*** – количество строк в таблице.

Строки таблицы описываются тегами **<tr>…</tr>**, а внутри них располагаются ячейки таблицы, которые ограничиваются тегами **<td>…</td>**. Ячейки

внутри таблицы можно объединять при помощи атрибутов тега **<td>** ***colspan*** и ***rowspan***.

Пример:

```
<table width="750" border=1 bgcolor=#ffffff align=center>
<tr bgcolor=lightgrey align=center>
     <td>Колонка 1</td><td>Колонка 2</td><td>Колонка 3</td>
</tr><tr align=center>
     <td colspan=3 align=center>Широкая ячейка</td>
     </tr>
<tr align=center>
     <td rowspan=2>Высокая<br>ячейка</td>
     <td>Ячейка 11</td>
     <td>Ячейка 12</td>
</tr>
<tr align=center >
     <td>Ячейка 21</td>
     <td>Ячейка 22</td>
</tr>
</table>
```

Часто при форматировании документов при помощи таблиц возникает необходимость, чтобы таблица была расположена вплотную к границам окна браузера. Для этого атрибутам тега **<body>**: ***marginwidth*** и ***marginheight*** для Internet Explorer или ***topmargin*** и ***leftmargin*** для Netscape Navigator присваивается значение "**0**". Обычно в теге указывают все четыре атрибута сразу, для того чтобы форматирование страницы удовлетворяло требованиям всех браузеров.

Гиперссылки. Гиперссылкой в электронном документе может быть любой элемент: текст, рисунок, внедренный объект (например, flash-ролик). Определить гиперссылку в HTML-документе можно с помощью тега **<a *href*="url">…**, где ***href*** – атрибут, значение которого является конечной точкой перехода по этой ссылке. В общем случае этим значением является URL документа (например, http://www.icsc.edu.ru). Текст или изображение, обозначающие ссылку, записываются внутри контейнера. Например:

```
<a href="http://www.icsc.edu.ru">Сайт кафедры «ИиУС»</a>.

<a href="../file.html"><img src="logo.gif" border="0"></a>
```

Атрибуту ***border*** тега **<img>** присвоено значение "**0**" для того, чтобы браузер не отображал рамку вокруг рисунка, так как по умолчанию для рисунка-ссылки толщина рамки вокруг рисунка равна одному пикселу.

Гиперссылка может указывать и на некоторый блок внутри того же документа:

```
<a name="top"> Начало документа
... ... ...
Конец документа
<a href="#top"> Вверх </a>
```

В этом примере **<a *name*="top">** не имеет закрывающего тега и обозначает так называемый якорь, ссылка под названием "Вверх" приведет пользователя к тому месту документа, где он расположен, т. е. к тексту "Начало документа". Конечно, это произойдет, если документ достаточно длинный.

Комментарии начинаются символами <! -- и заканчиваются символами -->. Текст, заключенный в теге комментария, не отображается в браузере.

Задание

Оформить отчёт по лабораторной работе с использованием средств языка HTML.

Отчёт должен содержать:

1) титульный лист;
2) содержание в виде гиперссылок, позволяющих перейти к выбранному пункту;
3) краткие теоретические сведения;
4) описание тегов, их синтаксиса, атрибутов и примеров использования в соответствии с вариантом задания (табл. 1.1);
5) листинг HTML-файлов.

Таблица 1.1

Варианты задания

Вариант	Теги
1	<AREA> / <DIV> / <!DOCTYPE>
2	<BODY> / / <IMG>
3	<HTML> / <NAV> / <TABLE>
4	<EMBED> / <META> / <H1>…<H6>
5	<FIELDSET> / <FORM> / <PRE>
6	<HEAD> / <A> / <INPUT>
7	<HR> / <TEXTAREA> / <TITLE>
8	<P> / <SCRIPT> / <OL>
9	<SUB> / <LINK> / <FOOTER>
10	<SUP> / <XMP> / <MENU>
11	<STYLE> / <WBR> / <DL>
12	<SOURCE> / <SECTION> / <HEADER>
13	<BASE> / <ASIDE> / <HGROUP>
14	<COL> / <INS> / <SECTION>
15	<I> / <DEL> / <ABBR>
16	<UL> / <ADDRESS> / <COLGROUP>
17	<B> / <ARTICLE> / <CAPTION>
18	<KEYGEN> / <ASIDE> / <IFRAME>
19	<PARAM> / <BLOCKQUOTE> / <OBJECT>
20	<TRACK> / <DIR> / <NOSCRIPT>

Контрольные вопросы

1. Для чего предназначен язык HTML?
2. Что такое тег?
3. В чём отличие парных и непарных тегов?
4. Что такое атрибут тега?
5. Структура HTML-докумета.
6. Какой тег используется для создания ссылок?
7. Как создать ссылку, ведущую на ту же страницу, на которой находится сама ссылка?

HTML-код титульного листа:

```
<!DOCTYPE html>
<html>
     <head>
          <meta http-equiv="Content-Type" content="text/html;
charset=utf-8">
          <title>
               Лабораторная работа № 1
          </title>
     </head>
     <body>
     <center><font size="+1">
     МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ<br>
     <b>ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ<br>
     ВЫСШЕГО ОБРАЗОВАНИЯ<br>
     «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» <br>
     (ДГТУ)</b></font><br>
     <br><br><br><br><br><br>
     <font size="+1">Кафедра «Вычислительные системы и информаци-
онная безопасность»</font><br>
     <br><br><br><br>
     <h1>Отчёт</h1><br>
     <font size="+2">по лабораторной работе № 1</font><br>
     <font size="+1"><b>"Язык гипертекстовой разметки
HTML"</b><br>
     по дисциплине "Интернет-технологии"</font><br>
     <br><br><br><br><br><br><br><br>
     <table><tr>
               <td align="right">
                    <font size="+1">выполнил:</font>
               </td><td width="250">
                    <font size="+1">обучающийся гр. ЭИБ3_</font>
               </td><td>
                    <font size="+1">_____________ Фамилия
И.О.</font>
               </td>
          </tr><tr><td></td><td></td><td><br><br><br></td></tr>
          <tr><td align="right">
                    <font size="+1">проверил:</font>
               </td><td>
                    <font size="+1">должность</font>
               </td><td>
                    <font size="+1">_____________ Фамилия
И.О.</font>
               </td></tr>
     </table>
     <br><br><br><br><br><br><br>
     <font size="+1"> Ростов-на-Дону <br>
     201_ г. <br>
     <a href="lab1.html"> Далее</a></font>
</center></body>
</html>
```

Титульный лист должен выглядеть следующим образом (рис. 1.1):

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (ДГТУ)

Кафедра «Вычислительные системы и информационная безопасность»

Отчёт

по лабораторной работе № 1
"Язык гипертекстовой разметки HTML"
по дисциплине "Интернет-технологии"

выполнил: обучающийся гр. ЭИБ3_ _____ Фамилия И.О.

проверил: должность _____ Фамилия И.О.

Ростов-на-Дону
201_ г.
Далее

Рис.1.1. Титульный лист

Пример выполнения работы (рис.1.2)

Содержание

Краткие теоретические сведения

Электронные документы предназначены для просмотра преимущественно с экрана монитора. Фундаментальная особенность электронных документов состоит в том, что посредством ссылки можно перейти прямо к тому тексту, на который она указывает. Такие ссылки могут указывать не только на тексты, но и на графическую, звуковую, видеоинформацию. Поэтому они получили название гиперссылок, а документы, формируемые с помощью языка гипертекстовой разметки HTML (Hypertext Markup Language), стали называть гипертекстовыми.

HTML-документы являются текстовыми файлами с расширением htm или html и могут создаваться с помощью любого текстового редактора, например, стандартной программы Windows "Блокнот". Для просмотра HTML документа в отформатированном виде предназначены Web-браузеры (например, Microsoft Internet Explorer, Netscape Navigator, Opera).

Язык HTML предназначен для описания форматированного электронного документа с помощью специальных управляющих кодов — тегов (от англ. tag — ярлык). Язык HTML является нечувствительным к регистру!

Теги бывают двух типов: открывающие и закрывающие. Большинство открывающих тегов имеют парный закрывающий тег. Тег состоит из следующих друг за другом в определенном порядке элементов:
— левой угловой скобки < (такого же, как символ "меньше чем");
— слеша / (только для закрывающего тега);
— имени тега, например TITLE или PRE;
— необязательных атрибутов. Тег может быть без атрибутов или сопровождаться одним или несколькими атрибутами, например ALIGN="CENTER";
— правой угловой скобки > (такой же, как символ "больше чем").

Часть HTML-документа от открывающего тега до парного ему закрывающего тега образует блок, называемый контейнером. Непарные теги, например <HR> — горизонтальная линия, являются контейнерами сами по себе, и не имеют закрывающего тега.

Задание

Оформить отчёт по лабораторной работе с использованием средств языка HTML. Отчёт должен содержать:

1. титульный лист,
2. содержание в виде гиперссылок, позволяющих перейти к выбранному пункту,
3. краткие теоретические сведения,
4. описание тегов, их синтаксиса, атрибутов и примеров использования в соответствии с вариантом задания,
5. листинг HTML-файлов.

Вариант	Теги
0	<a>, ...

Тег <a>

Описание

Тег <A> является одним из важных элементов HTML и предназначен для создания ссылок. В зависимости от присутствия параметров name или href тег <A> устанавливает ссылку или якорь. Якорем называется закладка внутри страницы, которую можно указать в качестве цели ссылки. При использовании ссылки, которая указывает на якорь, происходит переход к закладке внутри веб-страницы. Для создания ссылки необходимо сообщить браузеру, что является ссылкой, а также указать адрес документа, на который следует сделать ссылку. В качестве значения параметра href используется адрес документа (URL, Universal Resource Locator, универсальный указатель ресурсов), на который происходит переход. Адрес ссылки может быть абсолютным и относительным. Абсолютные адреса работают везде и всюду независимо от имени сайта или веб-страницы, где прописана ссылка. Относительные ссылки, как следует из их названия, построены относительно текущего документа или корня сайта.

Синтаксис

```
<a href="URL">Ссылка</a>
<a name="идентификатор">Ссылка</a>
```

Параметры

- accesskey — активация ссылки с помощью комбинации клавиш.
- href — задает адрес документа, на который следует перейти.
- name — устанавливает имя якоря внутри документа.
- rel — отношения между ссылаемым и текущим документами.
- tabindex — определяет последовательность перехода между ссылками при нажатии на кнопку "Tab".
- target — имя окна или фрейма, куда браузер будет загружать документ.
- title — добавляет всплывающую подсказку к тексту ссылки.

Пример

Код:

```
<a href="index.html">Перейти на главную страницу</a>
```

Результат:

Перейти на главную страницу

Тег ...

Описание

Устанавливает жирное начертание шрифта. Допустимо использовать этот тег совместно с другими тегами, которые определяют начертание текста.

Синтаксис

```
<b>Текст</b>
```

Пример

Код:

```
Текст текст <b>жирный текст</b> текст текст.
```

Результат:

Текст текст **жирный текст** текст текст.

Исходный код HTML-страниц

index.html

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
        <title>
            Лабораторная работа № 1
        </title>
    </head>

    <body>
    <center>
    <font size="+1">
    МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ<br>
    <br>ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ<br>
    ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ<br>
    «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»<br>
    (ДГТУ)</b></font><br>
    <br><br><br><br><br><br>
    <font size="+1">Кафедра «Вычислительные системы и информационная безопасность»</font><br>
    <br><br><br>
```

Рис.1.2. Пример выполнения

Лабораторная работа № 2

Каскадные таблицы стилей

Цель работы: изучить основные способы использования каскадных таблиц стилей для создания html-страниц.

Краткие теоретические сведения

Каскадные таблицы стилей (Cascading Style Sheets, CSS) – это язык, который содержит набор свойств (*стилевых атрибутов*) для определения внешнего вида документа и является альтернативным по отношению задания стилей с помощью атрибутов HTML. Их основная задача состоит в отделении формата Web-страницы от ее содержимого.

Спецификацией языка предусматривается три способа описания листов стилей в документе.

Первый способ – стиль, встроенный в тег, – описание стиля непосредственно внутри открывающего тега. Например, необходимо текст абзаца отобразить не черным, а синим цветом и шрифтом Arial. Для этого необходимо написать следующее:

```
<p style="color: blue; font-family: Arial"> Текст абзаца</p>
```

Второй способ – лист стилей, встроенный в документ, – позволяет описать стиль для документа с помощью контейнера **<style>…</style>**, расположенного в его заголовке.

```
<html>
<head>
<title>Использование CSS</title>
<style>
#doc_header {font-size: 36px; text-align: center; font-weight:
bold}
h1.header1 {color: red; font-family: Arial;}
p {font-family: Arial}
hyperref {font-family: Times New Roman; color: blue; font-size:
16px; text-decoration: none}
.hyperref: hover {color: green}
</style>
</head>
<body bgcolor=#dddddd>
<h1 id=doc_header> Использование CSS </h1>
<h1 class=header1> Страница с тегом &lt;style&gt;</h1>
<p> Этот текст должен быть отображен шрифтом Arial </p>
<p>A <a href=# class=hyperref>эта ссылка</a> отображается синим
цветом, шрифтом Times, но когда на нее наводят указатель мыши, она
становится зеленой.</p>
</body></html>
```

Первый тег в теле документа имеет идентификатор id=doc_header. На этот тег будет воздействовать стилевой атрибут #doc_header, который относится к любым тегам с ***Id*=“doc_header”**.

Второй тег не имеет идентификатора, но указан стилевой класс: **<h1 *class*="header1">**. На этот тег будет воздействовать стилевой атрибут ***h1.header1***, который относится к тегам <**h1>** со стилевым классом ***header1***.

Далее следуют два абзаца, стиль которых не имеет названия. Описанный в контейнере **<style>** стиль для тега **<p>** будет автоматически применяться ко всем контейнерам этого типа, если для них не будет указан другой стиль в теге.

Стили, встроенные в теги, отменяют установки листов стилей, встроенных в документ, либо дополняют их установки, если они не были определены ранее. Именно это и происходит в контейнере гиперссылки. Атрибут *font-family* встроенного в тег стиля является более важным в иерархии стилевых классов, нежели этот же атрибут встроенного в документ стиля для тега **<p>**. Поэтому ссылка отображается шрифтом Times New Roman, а не Arial.

Изменение цвета гиперссылки реализовано с помощью *псевдостиля*. Все имена псевдостилей заранее определены, причем определены они только для контейнеров гиперссылок.

Их всего четыре:

1. ***hover*** – применяется к ссылке, на которую наведен курсор мыши;
2. ***active*** – применяется к активным гиперссылкам;
3. ***visited*** – применяется ко всем посещенным ссылкам документа;
4. ***link*** – применяется ко всем гиперссылкам этого стиля.

Определять их в листе стилей необходимо в указанном порядке.

Третий способ создания CSS – связанные листы стилей. Чаще всего один и тот же набор стилей целесообразно применять сразу для нескольких страниц одновременно. Повтор одной и той же информации в заголовке каждого документа увеличивает размер документов и усложняет их модификацию. Третий способ позволяет выносить листы стилей в отдельный файл с расширением **css** (тег **<style>** в файле не пишется), а в заголовках документов достаточно указывать лишь один тег:

`<link rel="stylesheet" type="text/css" href="style.css">` (для IE)

или

`<style>@ url("style.css") </style>` (для остальных браузеров).

Связанные листы стилей являются самыми «слабыми» в иерархии стилевых классов. Их настройки отменяются стилевыми классами, встроенными в документ, однако это самый удобный способ описания стилей.

Задание

Используя HTML и CSS создать страницу соответствующую варианту.

Контрольные вопросы

1. Что такое CSS?
2. Какие существуют способы внедрения CSS-кода в HTMLдокумент?
3. Что такое псевдо стиль?
4. В чём отличие правил для тега, класса и идентификатора?
5. Какие существуют способы комбинирования CSS-правил?

Лабораторная работа № 3

Средства разработки web-сайтов. Введение в язык php

Цель работы: изучить принципы работы серверных приложений и основы языка PHP.

Краткие теоретические сведения

Язык PHP является языком сценариев, выполняемых на стороне сервера. Это означает, что он может работать внутри HTML-страниц, наделяя его возможностью генерации содержания по требованию. Web-сайт перестает быть совокупностью статических страниц, а представляет собой набор Web-страниц с внедренными PHP-скриптами, которые генерируют в соответствии с запросом пользователя HTML-страницу.

Текст скрипта можно свободно смешивать с текстом обычного форматированного HTML-документа, заключая его при этом в специальные последовательности символов:

```
<? скрипт; ?>
```

или

```
<?php скрипт; ?>
```

Синтаксис языка практически полностью совпадает с синтаксисом языка Си. Однако при этом он имеет отличия.

Чтобы PHP мог отличать имена переменных от команд или функций, имя переменной должно начинаться со знака доллара ("$") и не должно содержать пробелов, знаков апострофа и некоторых других символов.

Команды PHP обязательно разделяются символом точки с запятой.

В PHP нет необходимости описывать типы переменных. Распределением памяти под них занимается сам интерпретатор, при этом учитывается, какие данные присваиваются переменной – строка (string), целое число (integer) или число с плавающей точкой (double) и т.д.

Тип переменной может быть изменен как явно, так и неявно. Например,

```
$a=1;
$s=”23 Street”;
$b=$a+$s;  //$b=24  // неявное преобразование
$b=(string)$a + $s; //123 Street   //явное преобразование
```

Функции для работы с переменными приведены в табл. 3.1.

Переменные могут быть как глобальными (ключевое слово global), так и локальными.

Часто бывает удобно использовать так называемые «переменные». Это значит, что имя переменной можно изменять. Пример:

```
<?php
$a = "Hello";
$$a = "world!";
echo "$a ${$a}\n";
echo "$a $Hello\n";
?>
```

В третьей строке в качестве имени переменной использовалось содержимое переменной $a. Именно поэтому результатом выполнения операторов в двух последующих строках будет одна и та же строка “Hello world!”.

Таблица 3.1

Функции для работы с переменными

gettype()	Возвращает название типа данных переменной в виде строки	if(gettype($a)==”integer”) …
isset()	Возвращает **true**, если переменной присвоено какое-либо значение	if(!isset($a)) echo “Содержимого нет”;
unset()	Уничтожение переменной	unset($a);
empty()	Возвращает **true** если переменная не установлена, или имеет значение нуль или пустая строка	if(empty($a)) ….
is_*mun*()	Возвращает true, если тип переменной соответствует указанному в функции	if(is_integer($a)) ….
settype()	Устанавливает явным образом тип переменной (array, double, integer, object, string)	settype($a, “integer”);
intval, doubleval, strval	Извлекает значение указанного типа из переменной.	$a = intval(“12.33 kg); В **$a** будет занесено **12**.

Константы. В PHP имеются два вида констант: литеральные и символьные. Под литеральными константами понимаются неизменные величины, ссылка на которые производится без использования идентификатора, например, 3.14, “Hello”, 0.4. Символьные константы представляют собой удобный способоднажды присвоить значение идентификатору, а затем на протяжении всей программы ссылаться на это значение с помощью идентификатора. Например, вместо названия фирмы “Ростовский комбайновый завод” можно использовать константу с именем RSM и значение “Ростовский комбайновый завод”. Такой подход позволит обновить название фирмы только в одном месте.

Для создания констант используется функция define

```
define(“RSM”, “Ростовский комбайновый завод”);
define(“BR”, “<BR>\n ”);
```

Функция `defined()` позволяет определить, существует ли константа. Она возвращает 1, если константа существует, и 0 – в противном случае:

```
if(defined(“YELLOW”))
{
echo (“<BODY BGCOLOR=” . YELLOW . “>\n”);
}
```

Операторы. В языке PHP операторы сходны с операторами языка Си.

Управление ходом выполнения скрипта осуществляется теми же структурами, что и в Си:

```
if (условие)
{
    операции, выполняющиеся в случае, если условие верно;
```

```
}
else
{
    операции, выполняющиеся в случае, если условие неверно;
}

switch (переменная или выражение) {
    case условие:
         команды;
         break;
    case условие:
         команды;
         break;
    default: команды;
}

for (выражение1; выражение2; выражение3)
{
    тело цикла;
}

while (условие)
{
    тело цикла;
}
```

Чтобы пропустить оставшиеся операторы цикла и начать его заново со следующей итерации применяется оператор continue.

PHP позволяет также организовывать функции пользователя:

```
function имя_функции(аргумент1, аргумент2, ...)
{
      тело функции;
}
```

Функции могут возвращать значения, для чего, как обычно, применяется оператор return.

Массивы. Такая необходимая структура данных как массив может быть задана несколькими способами:

```
$fruit[0]=”банан”;
$fruit[1]=”ананас”; // и так далее
```

или

```
$names=array(“Маша”, “Дима”, “Лена”);
$surnames[]=“Иванов”;
$surnames[]=“Петров”;
```

В этих двух случаях каждому элементу массива автоматически присваивается индекс, начиная с 0.

Простейший способ обойти массив в цикле – использование count () для определения числа элементов в массиве, а затем организовать цикл for ():

```
$m = array(“ru”, “de”, “us”);
$n = count($m);
for($i=0; $i<$n; ++$i)
{
echo ($m[$i] . “<BR>”);
}
```

Часто удобнее вместо обычных индексированных массивов использовать так называемые ассоциативные массивы, в которых в качестве индекса служат строки:

```
$computer_parts[“Motherboard”]= “Asus NVidia2”;
$fruit=array(“banana” => “yellow”, “apple” => “red”).
```

Следующий пример показывает, что элементами массива могут быть и сами массивы.

```
$months=array(“autumn” => array(“September”, “October”, “No-
vember”),
“winter” => “December”, “January”, “February”));
```

Функции each() и list() можно использовать для циклического прохода через массив. Ниже приводится пример цикла с перемещением по массиву:

```
reset($m);
while(list($key,$value) = each($m))
{
echo(“Элемент ” . $key . “ = ” . $value  . “<BR>” );
}
```

Функция reset() переводит внутренний указатель на первый элемент массива. А затем для каждого элемента массива название индекса и его значение заносятся соответственно в $key и $value.

Аналогично можно использовать цикл

```
foreach($m as $m_element)
{
echo(“Элемент = ” . $m_element );
}
```

Для перемещения по массиву также можно использовать функции next() и prev(). Функция next перемещает внутренний указатель на один элемент вправо и возвращает значение этого элемента.

```
for(reset($m); $k=key($m); next($m))
{
$val = current($m);
echo(“Элемент ” . $k . “ = ” . $val  . “<BR>”);
}
```

Для массивов также определены различные функции сортировки.

Включение файлов. Для включения файлов в страницы PHP имеется две инструкции – **require** и **include**, которые считывают и выполняют код, представленный в указанном файле. Благодаря этому можно создавать повторно используемые функции, константы и прочий код и хранить их централизованно в файле, который доступен всем остальным сценариям. Инструкция require осуществляет замену только в момент запуска сценария, а include – в момент выполнения.

Ниже приведенный цикл благодаря использованию include, а не require будет подключать 9 различных файлов file1.php … file9.php

```
for ($i=1; $i<=9; $i++)
{
include “file$i.php”;
}
```

Инструкция **include** не является оператором в прямом смысле, а напрямую подставляет содержимое файла. Это приведет к тому, что первая строка файла будет в теле цикла, а остальные – вне его. Поэтому необязательные на первый взгляд фигурные скобки, тем не менее, обязательны.

Таким образом, **include** имеет в функциональном отношении некоторые преимущества, связанные с возможностью его использования в циклах. Тем не менее, он является более ресурсоемким и везде по возможности необходимо использовать **require**.

Подключаемый файл предполагается написанным на **HTML**, если только не происходит переключения с **html** на **php** с помощью тегов **<?php .. ?>.** PHP ищет подключаемый файл в директории, который определяется директивой **include_path** в конфигурационном файле **php.ini**.

Также имеются инструкции **require_once** и **include_once**, которые осуществляют подключение файла только в том случае, если до этого затребованный файл не был подключен**.** По возможности везде используйте инструкции с суффиксом **_once**, что позволит сделать программу более масштабируемой.

Задание

На языке PHP разработать программу, вычисляющую и выводящую на экран в виде таблицы значения функции, в соответствии с вариантом задания, на интервале от -10 до 10 с шагом 0.5.

Вариант 1

$$F = \begin{cases} ax^2 + b & \text{при } x < 0 \text{ и } b \neq 0 \\ \dfrac{x-a}{x-c} & \text{при } x > 0 \text{ и } b = 0 \\ \dfrac{x}{c} & \text{в остальных случаях} \end{cases}$$

Вариант 2

$$F = \begin{cases} \dfrac{1}{ax} - b & \text{при } x+5 < 0 \text{ и } c = 0 \\ \dfrac{x-a}{x} & \text{при } x+5 > 0 \text{ и } c \neq 0 \\ \dfrac{10x}{c-4} & \text{в остальных случаях} \end{cases}$$

Вариант 3

$$F = \begin{cases} ax^2 + bx + c & \text{при } a < 0 \text{ и } c \neq 0 \\ \dfrac{-a}{x-c} & \text{при } a > 0 \text{ и } c = 0 \\ a(x+c) & \text{в остальных случаях} \end{cases}$$

Вариант 4

$$F = \begin{cases} -ax\ -c & при\ c<0\ и\ x\neq 0 \\ \dfrac{x-a}{-c} & при\ c>0\ и\ x=0 \\ \dfrac{bx}{c-a} & в\ остальных\ случаях \end{cases}$$

Вариант 5

$$F = \begin{cases} a-\dfrac{x}{10+x} & при\ x<0\ и\ b\neq 0 \\ \dfrac{x-a}{x-c} & при\ x>0\ и\ b=0 \\ 3x+\dfrac{2}{c} & в\ остальных\ случаях \end{cases}$$

Вариант 6

$$F = \begin{cases} ax^2+b^2x & при\ c<0\ и\ b\neq 0 \\ \dfrac{x+a}{x+c} & при\ c>0\ и\ b=0 \\ \dfrac{x}{c} & в\ остальных\ случаях \end{cases}$$

Вариант 7

$$F = \begin{cases} -ax^2-b & при\ x<5\ и\ c\neq 0 \\ \dfrac{x-a}{x} & при\ x>5\ и\ c=0 \\ \dfrac{-x}{c} & в\ остальных\ случаях \end{cases}$$

Вариант 8

$$F = \begin{cases} -ax^2 & при\ c<0\ и\ a\neq 0 \\ \dfrac{a-x}{cx} & при\ c>0\ и\ a=0 \\ \dfrac{x}{c} & в\ остальных\ случаях \end{cases}$$

Вариант 9

$$F = \begin{cases} ax^2+b^2x & при\ a<0\ и\ x\neq 0 \\ x-\dfrac{a}{x-c} & при\ a>0\ и\ x=0 \\ 1+\dfrac{x}{c} & в\ остальных\ случаях \end{cases}$$

Вариант 10

$$F = \begin{cases} ax^2 - bx + c & при\ x < 3\ и\ b \neq 0 \\ \dfrac{x-a}{x-c} & при\ x > 3\ и\ b = 0 \\ \dfrac{x}{c} & в\ остальных\ случаях \end{cases}$$

Вариант 11

$$F = \begin{cases} ax^2 + \dfrac{b}{c} & при\ x < 1\ и\ c \neq 0 \\ \dfrac{x-a}{(x-c)^2} & при\ x > 1{,}5\ и\ c = 0 \\ \dfrac{x^2}{c^2} & в\ остальных\ случаях \end{cases}$$

Вариант 12

$$F = \begin{cases} ax^3 + b^2 + c & при\ x < 0{,}6\ и\ b + c \neq 0 \\ \dfrac{x-a}{x-c} & при\ x > 0{,}6\ и\ b + c = 0 \\ \dfrac{x}{c} + \dfrac{x}{a} & в\ остальных\ случаях \end{cases}$$

Вариант 13

$$F = \begin{cases} ax^2 + b & при\ x - 1 < 0\ и\ b - c \neq 0 \\ \dfrac{x-a}{x} & при\ x - 1 > 0\ и\ b + x = 0 \\ \dfrac{x}{c} & в\ остальных\ случаях \end{cases}$$

Вариант 14

$$F = \begin{cases} -ax^3 - b & при\ x + c < 0\ и\ a \neq 0 \\ \dfrac{x-a}{x-c} & при\ x + c > 0\ и\ a = 0 \\ \dfrac{x}{c} + \dfrac{c}{x} & в\ остальных\ случаях \end{cases}$$

Вариант 15

$$F = \begin{cases} -ax^2 + b & при\ x < 0\ и\ b \neq 0 \\ \dfrac{x}{x-c} + 5{,}5 & при\ x > 0\ и\ b = 0 \\ \dfrac{x}{-c} & в\ остальных\ случаях \end{cases}$$

Вариант 16

$$F = \begin{cases} a(x+c)^2 - b & при\ x = 0\ и\ b \neq 0 \\ \dfrac{x-a}{-c} & при\ x = 0\ и\ b = 0 \\ a + \dfrac{x}{c} & в\ остальных\ случаях \end{cases}$$

Вариант 17

$$F = \begin{cases} ax^2 - cx + b & при\ x + 10 < 0\ и\ b \neq 0 \\ \dfrac{x-a}{x-c} & при\ x + 10 > 0\ и\ b = 0 \\ \dfrac{-x}{a-c} & в\ остальных\ случаях \end{cases}$$

Вариант 18

$$F = \begin{cases} ax^3 + bx^2 & при\ x < 0\ и\ b \neq 0 \\ \dfrac{x-a}{x-c} & при\ x > 0\ и\ b = 0 \\ \dfrac{x+5}{c(x-10)} & в\ остальных\ случаях \end{cases}$$

Вариант 19

$$F = \begin{cases} a(x+7)^2 - b & при\ x < 5\ и\ b \neq 0 \\ \dfrac{x-cd}{ax} & при\ x > 5\ и\ b = 0 \\ \dfrac{x}{c} & в\ остальных\ случаях \end{cases}$$

Вариант 20

$$F = \begin{cases} -\dfrac{2x-c}{cx-a} & при\ x<0\ и\ b\neq 0 \\ \dfrac{x-a}{x-c} & при\ x>0\ и\ b=0 \\ -\dfrac{x}{c}+\dfrac{-c}{2x} & в\ остальных\ случаях \end{cases}$$

Контрольные вопросы

1. Какие средства используются при создании сайтов?
2. Какие функции выполняет Web-сервер?
3. Какие функции выполняет интерпретатор языка программирования?
4. Какие функции выполняет сервер баз данных?
5. Что такое динамическая страница?
6. Какой тег используется для внедрения PHP-кода в HTMLдокумент?
7. В чём особенность описания переменных в PHP?

Лабораторная работа № 4

Передача данных с формы

Цель работы: изучить способы передачи информации в программе на языке php при помощи web-форм.

Краткие теоретические сведения

Формы. *Форма* – это средство, позволяющее получать различные данные от пользователя web-страницы и затем обрабатывать их с помощью JavaScript на машине клиента либо передавать их на сервер, который затем может сформировать страницу специально для этого пользователя, занести эти данные в базу данных и т. д.

Формы создаются при помощи контейнера **<form>…</form>**. В форме размещаются *поля формы* (*элементы управления*). Тег **<form>** имеет следующие атрибуты:

– ***action*** – адрес серверной программы, которая запустится при вызове команды submit;

– ***method*** – метод передачи данных (“**get**” или “**post**”).

Кроме того, как и большинство тегов, тег <**form**> имеет атрибуты ***id***, ***name***, ***class***, ***style***. В теге также можно указать слово атрибут ***disabled***, что сделает все поля формы недоступными для редактирования пользователем.

Свойство ***method*** сообщает браузеру, как передавать данные: включив их в URL (метод “**get**”) или поместив их в пакет данных протокола HTTP, по которому и осуществляется в основном передача данных в Интернет (метод **“post”**). Первый метод удобен тогда, когда нет необходимости передавать большие объ-

емы данных и не нужно скрывать их от глаз пользователей (например, ключевые слова для поисковых серверов или номер товара в базе данных Интернет-магазина). Метод **“post”** не имеет ограничений на объем пересылаемых данных и используется для передачи больших массивов данных и файлов.

Тег **<input>** позволяет организовывать сразу несколько разных элементов пользовательского интерфейса, применяемых в формах:

1. **<input *type*=“text”>** – поле для ввода текста (одна строка). Атрибуты: ***maxlength*** – максимальная длина строки в символах, ***size*** – видимый размер поля ввода, ***value*** – значение по умолчанию.

2. **<input *type*=“password”>** – поле ввода пароля, все символы заменяются “звездочками”. Атрибуты те же, что и для поля ввода текста.

3. **<input *type*=“checkbox”>** – флажок. Атрибуты: ***checked*** – включен по умолчанию, ***value*** — отсылаемое серверу значение в случае включения этого элемента.

4. **<input *type*=“radio”>** – переключатель. Атрибут ***name*** должен быть одинаковым для всей группы переключателей. Другие атрибуты как у флажка.

5. **<input *type*=“hidden”>** – скрытое поле, которое не отображается в форме, но его значение ***value*** отправляется вместе с остальными данными формы.

6. **<input *type*=“file”>** – поле ввода имени файла с кнопкой “Browse” для отправки файла на сервер. Атрибут **v*alue*** – значение по умолчанию.

7. **<input *type*=“button”>** – кнопка. Атрибут ***value*** – текст, отображаемый на кнопке.

8. **<input *type*=“reset”>** – кнопка, нажатие которой приводит к очистке всей формы.

9. **<input *type*=“submit”>** – кнопка отправки данных формы.

10. **<input *type*=“image”>** – по действию аналогичен кнопке “**submit”**, но вместо кнопки используется рисунок (атрибут ***src***).

Контейнер **<textarea>…</textarea>** создает на странице поле редактирования текста, внутри него указывается текст, который будет отображен в этом поле по умолчанию. Атрибуты: ***rows*** и ***cols*** – видимый размер зоны текста в количествах строк и столбцов (в символах), а также ***id***, ***name***, ***style***, ***class***, ***tabindex***, ***disabled***, ***readonly***.

Контейнер **<select>…</select>** позволяет создать выпадающий список. Пункты списка заключаются в теги **<option>**. Атрибуты: ***value*** – значение, отсылаемое в случае выбора этого пункта; ***selected*** – пункт выбран по умолчанию. Если атрибут ***value*** не указан, то отправляется текст пункта.

Элементы формы можно собирать в группы с помощью контейнера **<fieldset>…</fieldset>**. Группа будет выделена рамкой, в верхней части которой может быть отображен заголовок, определенный с помощью тега **<legend>…</legend>**.

Общими для всех типов элементов являются такие атрибуты, как: ***disabled*** – делает элемент недоступным для пользователя, ***readonly*** – элемент только для чтения.

При нажатии на кнопку Submit на форме выполняется действие, указанное в атрибуте ACTION этой формы.

Возможно использовать 2 варианта значения атрибута ACTION:
1) PHP-скрипт, например, ACTION=”2.php”;
2) JavaScript, например, ACTION=”javascript: fun()”.

Использование PHP-скрипта.

Например, пусть имеется страница (рис. 4.1):

```
<FORM ACTION="2.php"  METHOD="POST" >
     <INPUT TYPE=”TEXT“  VALUE="2">
     <INPUT TYPE=”CHECKBOX“  NAME="c" VALUE="flag">Флажок<BR>
     <INPUT TYPE=”RADIO” NAME="r" VALUE="red">Выбор1<BR>
     <INPUT TYPE=”RADIO” NAME="r" VALUE="green">Выбор2<BR>
     <INPUT TYPE=”SUBMIT“ NAME="b1" VALUE="Calc1">
     <INPUT TYPE=”SUBMIT” NAME="b2" VALUE="Calc2">
</FORM>
```

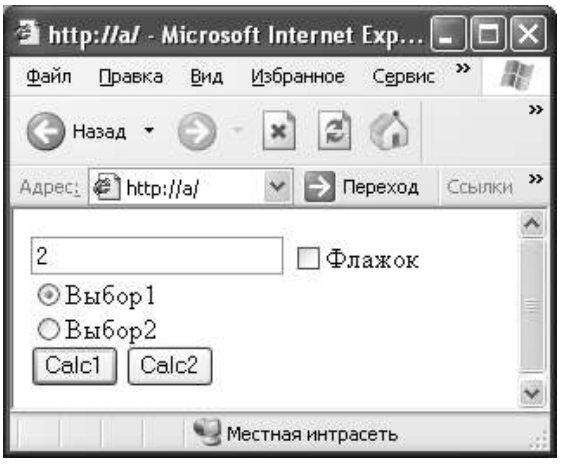

Рис. 4.1. Форма с установленными значениями элементов управления

При нажатии на кнопку типа Calc1 произойдет передача установленных значений всех элементов управления на страницу “2.php” методом POST.

То есть на странице 2.php имеется возможность получить значения с формы страницы “1.php” из элементов суперглобального массива $_POST.

На странице 2.php будут доступны элементы массива $_POST:

```
$_POST[‘t’] = 2;
$_POST[‘r’] = “red”;
$_POST[‘b1’] = “Calc1”.
```

Значение элемента управления “c” типа CHECKBOX и кнопки b2 не будут переданы, поскольку флажок “c” не установлен, а кнопка “b2” не была нажата.

Задание

1. Создать 2 страницы: input.php и output.php:
страница input.php должна содержать форму ввода данных для вычисления значений функции из предыдущей лабораторной, а именно: граничных значений аргумента, шага и коэффициентов (a, b, c) функции. После нажатия на кнопку “посчитать” должна открыться страница output.php с результатами расчета в табличной форме.

2. Если обнаружена ошибка во введенных данных (неверно введено число или отсутствует число, или начальное значение больше конечного), то резуль-

татом выполнения скрипта должна стать HTML-страница с сообщением о том, в каком поле допущена ошибка, и ссылкой на первую страницу.

3. Содержимое созданных страниц необходимо вывести внутри шаблона, разработанного в лабораторной работе № 2.

При нажатии на один из пунктов меню в центральной части страницы должна быть отображена форма, содержащаяся в файле input.php (рис. 4.2)

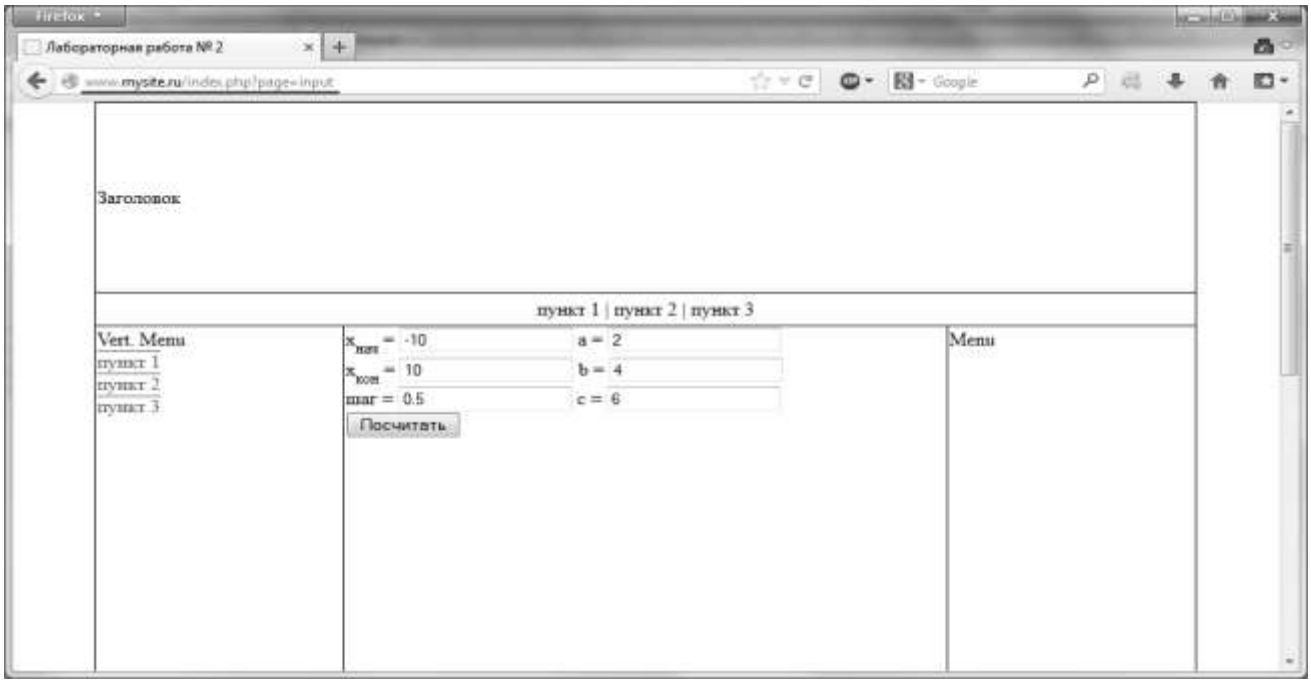

Рис. 4.2. Содержание лабораторной с формой в файле input.php

При нажатии кнопки «Посчитать» на месте формы ввода должен быть отображён результат вычислений (рис. 4.3)

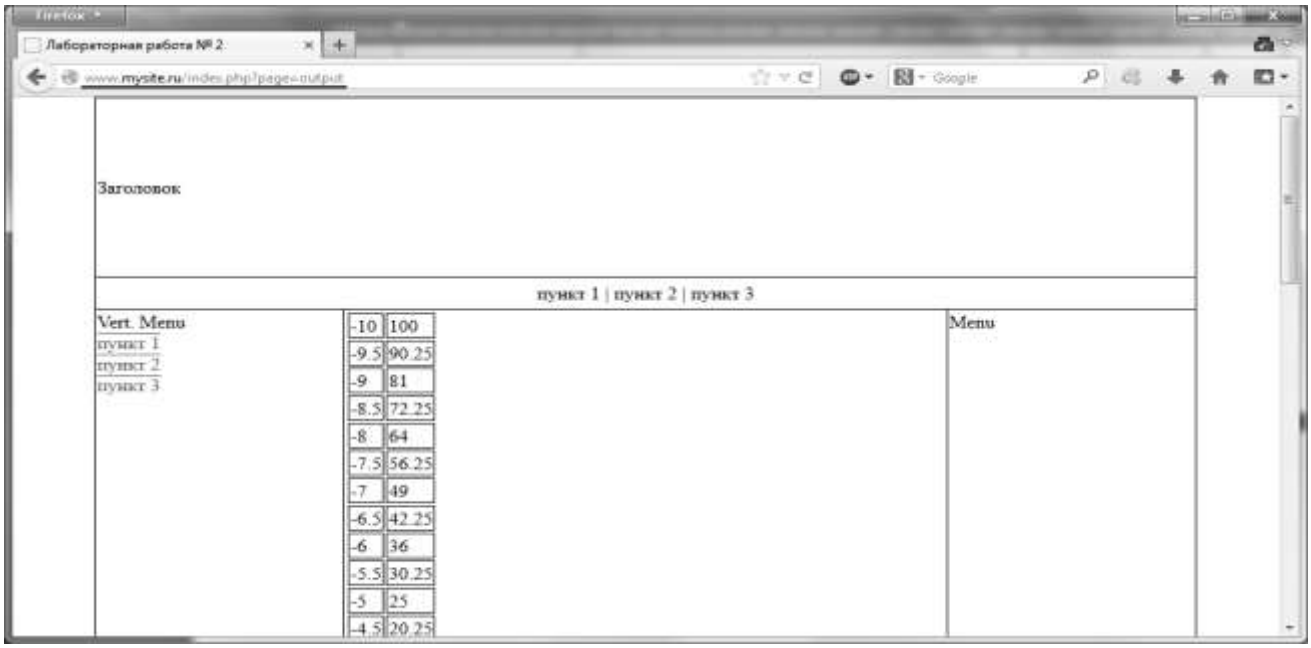

Рис. 4.3. Содержание лабораторной с формой в файле output.php

Контрольные вопросы

1. Что такое форма?
2. За что отвечает атрибут формы ―action‖?
3. Какие элементы могут быть размещены на форме?
4. Какие методы используются для передачи данных с формы? В чём их отличие?
5. Можно ли передать PHP-скрипту данные методом GET без использования формы?

Лабораторная работа № 5

Работа файлами cookie и сессиями

Цель работы: изучить два основные способа сохранения пользовательских переменных – механизм сессий и файлы cookie.

Краткие теоретические сведения

Cookie. Cookie – это некоторый текст, который передается сервером браузеру пользователя. Этот текст сохраняется на компьютере пользователя в виде файла с расширением. cookie в кэше браузера. С помощью этого механизма можно организовывать корзину в Интернет-магазине, настраивать сайты под конкретного пользователя и т.д. В cookie не рекомендуется сохранять важные данные, т.к. пользователь имеет свободный доступ к этим файлам, однако в них можно сохранять, например, некоторый идентификатор, по которому может быть найдена информация о пользователе в базе данных на сервере. Кроме того, при использовании файлов cookie необходимо помнить, что в браузере пользователя может быть отключена возможность работы с ними.

При работе с cookie необходимо помнить, что они передаются клиенту вместе с HTTP-заголовком, а это значит, что их можно устанавливать только до передачи какого-либо текста клиенту. В противном случае клиент получит сообщение об ошибке “*Cannot add header information*”.

Для установки, изменения и удаления переменных cookie применяется одна и та же функция `setcookie()`. Она может принимать до шести параметров (обязательным параметром является только первый), указанных в том же порядке, в котором они перечислены ниже:

– `Name` – имя переменной, которая может быть затем доступна через массив `$_COOKIE` при последующих обращениях пользователя к скриптам сервера.

– `Value` – значение переменной, которое нужно сохранить.

– `Expire` – момент времени, после наступления которого cookie станет недоступным. Если параметр не указан, cookie становится недоступным после закрытия окна браузера.

– `Path` – каталог на сервере, для скриптов которого данный cookie доступен. Если указан слэш (‘/’), то cookie доступен из всех фалов и поддиректорий web-сервера.

– `Domain` – файлы cookie доступны только тем серверам (доменам), которые их установили. Если параметр не указан, то ему присваивается доменное имя сервера, его установившего.

– `Security` – Принимает значения 0 или 1. Если параметр равен 1, то cookie будет передаваться посредством защищенного соединения по протоколу HTTPS.

Пример. Создайте файл с расширением .php, сохраните в нем приведенный ниже скрипт:

```
<?
$cookie_name = "test_cookie";
$cookie_value = "test string!";
$cookie_expire = time()+86400;

setcookie($cookie_name, $cookie_value, $cookie_expire, "/");
setcookie("perem1", $cookie_value);
?>
<HTML>
<HEAD>
<TITLE>Set Test Cookie</TITLE>
</HEAD>
<BODY>
<h1>Mmmmmmmm...cookie!</h1>
</BODY>
</HTML>
```

Необходимо обратить внимание на строку, в которой переменной $cookie_expire присваивается значение. Функция `time()` возвращает целое число секунд, прошедшее до момента ее вызова с 1 января 1970 года (т.н. Unix-формат). К этому значению прибавляется 86400 секунд, а это значит, что действие cookie прекратится через 24 часа после установки.

Удалить cookie можно следующими способами:

```
setcookie($cookie_name, ""); //значение не установлено
setcookie($cookie_name, "", time()-3600); //срок действия прошел
```

После установки cookie она становится доступной на следующей странице:

```
if (isset($_COOKIE[$cookie_name]) &&
$_COOKIE[$cookie_name]!="deleted")
{
echo $_COOKIE[$cookie_name] . "<br>";
}
if (isset($_COOKIE["perem1"]) &&
$_COOKIE["perem1"]!="deleted")
{
echo $_COOKIE["perem1"] . "<br>";
}
```

Можно сохранить в одном cookie одномерный массив (но устанавливать значение каждого элемента нужно по отдельности):

```
setcookie("m[0]", "banan");
setcookie("m[1]", "ananas");
```

Установку cookie с именем m можно проверить на следующей странице:

```
if (isset($_COOKIE['m']))
{
foreach ($_COOKIE['m'] as $name => $value)
{
echo "$name == $value <br>";
}
}
```

Сессии. Сессии в PHP были введены для того, чтобы сделать более удобным процесс передачи данных пользователя от одной страницы сайта к другой. Вообще сессию можно определить, как промежуток времени, в течение которого пользователь находится на сайте. Для программистов же сессия – это некоторый набор переменных всевозможных типов и их значений, который хранится на сервере и имеется доступ к ним с любой страницы текущего пользователя, пока не произойдет закрытие браузера пользователем. Каждый такой набор имеет свой уникальный идентификатор, например, `940f8b05a40d5119c030c9c7745aead9`. Именно такое имя будет иметь файл сессии на сервере. Этот же идентификатор автоматически отсылается пользователю с помощью cookie или непосредственно в URL. В зависимости от настроек PHP, может использоваться любой из этих способов, но наиболее предпочтительно применение cookie (по умолчанию), т.к. иначе даже сам пользователь по недосмотру сможет сам передать его злоумышленнику, которому станет доступна вся информация о нем в отношении этого сайта. Более подробную информацию о настройках сессий можно найти в документации по PHP.

Для того, чтобы начать пользовательскую сессию, необходимо в серверном скрипте вызвать функцию `session_start()`. Эта функция делает много важных вещей. Во-первых, она проверяет, была ли для этого пользователя запущена сессия. Для этого у пользователя должен храниться доступный cookie с идентификатором сессии. Если это не так, то создается новая сессия (генерируется идентификатор и передается пользователю). Затем все переменные сессии, которые хранятся на сервере, инициализируются в массиве $_SESSION.

При работе с сессиями различают следующие этапы:

1. открытие сессии;
2. регистрация переменных сессии и их использование;
3. закрытие сессии.

Рассмотрим пример. Пусть необходимо, чтобы переменная, созданная в скрипте first.php, оставалась видна и в скрипте second.php. Для этого тексты скриптов должны быть примерно следующими:

first.php

```
<?
session_start();
// задаём значение переменной
$a = "Меня задали на index.php";
```

```
     // регистрируем переменную с открытой сессией
     // важно: названия переменных передаются функции
session_register()
     // без знака $
     session_register("a");
     $_SESSION["a"] = $a;
     ?>
     <html>
       <body>
         <p>Начата новая сессия</p>
         <p>Сохранилась ли переменная $a  <a
href="second.php>там</a>?</p>
       </body>
     </html>
```

second.php

```
     <?php
       // открываем сессию
       session_start();
     ?>
     <html>
       <body>
       <?php
         echo $_SESSION[“a”];
       ?>
       </body>
     </html>
```

Другие полезные функции для работы с сессиями:

session_unregister(string) – сессия "забывает" значение заданной глобальной переменной;

session_destroy() – сессия уничтожается (по умолчанию она уничтожается, например, если пользователь покинул систему, нажав кнопку "выход");

session_set_cookie_params(int lifetime [, string path [, string domain]]) – с помощью этой функции можно установить, как долго будет действительна сессия. По умолчанию, сессия действительна до тех пор, пока клиент не закроет окно браузера.

Следующий код позволит осуществить запрет запуска страниц, предназначенных только для администраторов. Код помещается в начало каждой страницы, на которую имеет доступ только администраторы. Если ранее был зарегистрирован администратор, то в сессии зарегистрирована переменная admin. Если переменная admin в сессии отсутствует, то осуществляется переход на главную страницу сайта.

```
     <?
     session_start();
     if (!session_is_registered("admin"))
     header("Location: http://icsc.edu.ru/index.php");
     //Здесь располагается код страницы администрирования
     ?>
```

Естественно, для того, чтобы выше объявленный код работал, необходимо на странице регистрации пользователя в случае входа пользователя из группы администраторы осуществить создание переменной в сессии под именем admin.

Задание

Разработать скрипт, который выдавал бы пользователю приветственную страницу с указанием его имени или при первом входе должно быть предложение указать пользователю имя, настройки фонового цвета, какие-либо другие настройки (определить самостоятельно, всего не менее пяти параметров). После передачи этих сведений на сервер, пользователь должен получить страницу с оформленными по параметрам, указанным пользователем. После того, как пользователь закрыл окно браузера, а затем через, например, 2 минуты вернулся на эту страницу вновь, все его настройки должны быть восстановлены, причем с возможностью повторного изменения настроек.

Контрольные вопросы

1. Что такое cookie-файлы?
2. Какая функция в PHP позволяет установить cookie-файл?
3. В какой момент работы PHP-скрипта необходимо создавать cookie-файлы? Почему?
4. Что такое время жизни cookie-файла?
5. Для чего используются cookie?
6. Что такое сессия?
7. Какая функция в PHP позволяет установить сессию?
8. Алгоритм работы сессии?
9. Как закрывается сессия?

Лабораторная работа № 6
СУБД MySQL

Цель работы: получение практических навыков проектирования структуры базы данных и использования инструментальных средства и языка SQL для манипулирования данными.

Краткие теоретические сведения

База данных – совокупность данных, отражающая состояние объектов и их отношений в конкретной предметной области.

Система управления базами данных (СУБД) – совокупность языковых и программных средств, предназначенных для создания, ведения и совместного использования БД многими пользователями.

Каждая строка в таблице представляет собой один уникальный объект в реальном мире, а каждый атрибут в ней относиться именно к этому объекту. Для логической гарантии уникальности каждой из строк один из атрибутов

назначается первичным ключом – именно по нему обращаются к строкам таблицы команды модификации данных.

База данных включает, как правило, несколько таблиц, которые могут быть связаны друг с другом различными отношениями.

В отношениях "один к одному" каждая строка таблицы связана с единственной строкой в другой таблице.

В отношении “один ко многим” одной записи первой таблицы ставится в соответствие несколько записей второй таблицы, однако, каждая запись второй таблицы не может иметь более одной соответствующей записи в первой таблице.

В отношениях типа “многие ко многим” обе стороны могут быть связаны с множеством элементов противоположной стороны.

Все связи реализуются с помощью внешних ключей. Внешний ключ – столбец таблицы, ссылающийся на первичный ключ другой таблицы.

При разработке базы данных обычно выделяется несколько уровней моделирования, при помощи которых происходит переход от предметной области к конкретной реализации базы данных средствами конкретной СУБД. Можно выделить следующие уровни:

– сама предметная область;
– модель предметной области;
– логическая модель данных;
– физическая модель данных;
– собственно, база данных и приложения.

Предметная область – это часть реального мира, данные о которой мы хотим отразить в базе данных. Например, в качестве предметной области можно выбрать бухгалтерию какого-либо предприятия, отдел кадров, банк, магазин и т.д. Предметная область бесконечна и содержит как существенно важные понятия и данные, так и малозначащие или вообще не значащие данные. Так, если в качестве предметной области выбрать учет товаров на складе, то понятия "накладная" и "счет-фактура" являются существенно важными понятиями, а то, что сотрудница, принимающая накладные, имеет двоих детей – это для учета товаров неважно. Однако с точки зрения отдела кадров данные о наличии детей являются существенно важными. Таким образом, важность данных зависит от выбора предметной области.

Модель предметной области – это наши знания о предметной области. Знания могут быть как в виде неформальных знаний в мозгу эксперта, так и выражены формально при помощи каких-либо средств. В качестве таких средств могут выступать текстовые описания предметной области, наборы должностных инструкций, правила ведения дел в компании и т.п. Опыт показывает, что текстовый способ представления модели предметной области крайне неэффективен. Гораздо более информативными и полезными при разработке баз данных являются описания предметной области, выполненные в виде схем при помощи специализированных графических нотаций.

Логическая модель данных описывает понятия предметной области, их взаимосвязь, а также ограничения на данные, налагаемые предметной областью. Примеры понятий – "сотрудник", "отдел", "проект", "зарплата". Примеры взаимосвязей между понятиями – "сотрудник числится ровно в одном отделе",

"сотрудник может выполнять несколько проектов", "над одним проектом может работать несколько сотрудников". Примеры ограничений – "возраст сотрудника не менее 16 и не более 60 лет".

Логическая модель данных является начальным прототипом будущей базы данных. Логическая модель строится в терминах информационных единиц, но без привязки к конкретной СУБД. Более того, логическая модель данных необязательно должна быть выражена средствами именно реляционной модели данных.

Физическая модель данных описывает данные средствами конкретной СУБД. Отношения, разработанные на стадии формирования логической модели данных, преобразуются в таблицы, атрибуты становятся столбцами таблиц, для ключевых атрибутов создаются уникальные индексы, домены преображаются в типы данных, принятые в конкретной СУБД.

Ограничения, имеющиеся в логической модели данных, реализуются различными средствами СУБД, например, при помощи индексов, декларативных ограничений целостности, триггеров, хранимых процедур.

Концептуальное проектирование базы данных

Концептуальное проектирование базы данных состоит в построении инфологической модели данных (модели **«сущность - связь»** (или ER-модели)).

Главными элементами этой модели данных являются *сущности, их атрибуты* и *типы связей.* Сущности часто представляют в виде *существительных,* а типы связей – в виде *глаголов.*

ER-модель предметной области изображается в виде диаграммы с учетом принятых обозначений для ее элементов (рис. 6.1).

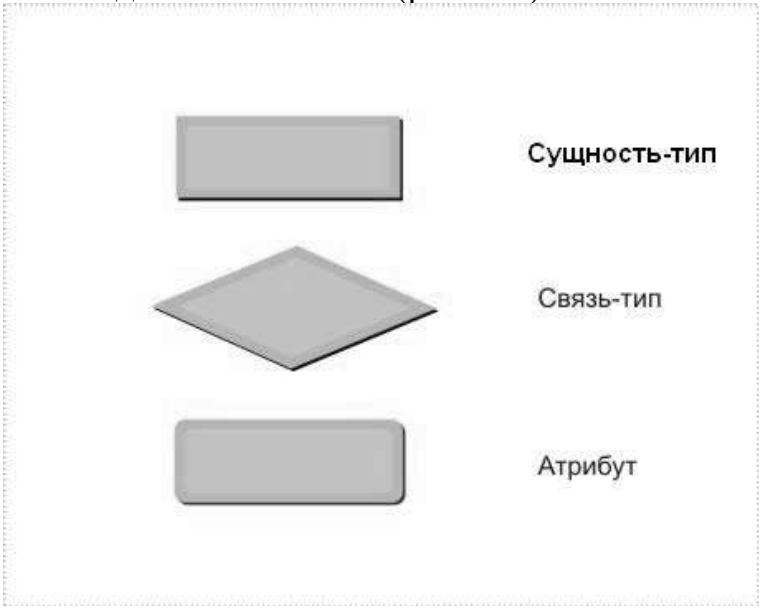

Рис.6.1. Обозначения элементов диаграммы

Сущности

Сущность – это то, о чем накапливается информация в информационной системе и что может быть однозначно идентифицировано.

Сущность – тип (в дальнейшем просто сущность) характеризуется независимым существованием и представляет множество объектов реального мира

с одинаковыми свойствами. Отдельные объекты, которые входят в данный тип, называют экземплярами сущности.

Атрибуты

Атрибут – это поименованная характеристика сущности, с помощью которой моделируется ее свойство. Каждой сущности присущи свои атрибуты. Например, сущность **ТОВАР** должна иметь такие атрибуты: Наименование_товара, Индекс_товара, Цена_товара, Количество. На диаграммах атрибуты сущности соединяются с ней линиями.

Ключи

Среди атрибутов особое положение занимают такие, с помощью которых можно идентифицировать экземпляр сущности. Такие атрибуты называются *ключами.* Атрибут или несколько атрибутов, значения которых уникальным образом идентифицируют каждый экземпляр сущности, являются *потенциальным* ключом данной сущности. Потенциальных ключей может быть несколько.

Один из потенциальных ключей может быть выбран в качестве *первичного* ключа. Обычно в качестве первичного ключа выбирается тот, который имеет наименьшую длину. Остальные потенциальные ключи называются *альтернативными.* Тот факт, что атрибут служит первичным ключом, отмечается его подчеркиванием.

Идентификацию некоторых сущностей иногда приходится осуществлять при помощи *составных* ключей, которые включают несколько атрибутов.

Связи между сущностями

Две сущности могут быть связаны между собой. Подобная связь осуществляется через связь экземпляров одной сущности с экземплярами другой сущности, образуя набор экземпляров связи между двумя сущностями, который называется типом связи.

Показатель кардинальности

Для того чтобы указать количество возможных связей для каждого экземпляра участвующего в связи сущности, используют показатель кардинальности.

Для бинарных связей показатель кардинальности может иметь следующие значения:

"один к одному" (1: 1), "один ко многим"(1: N), "многие ко многим"(M: N).

Если максимальная мощность связи в обоих направлениях равна одному, мы называем ее связью "один к одному" (1: 1).

Каждая сущность имеет имя и изображается на диаграммах в виде прямоугольника, а экземпляр сущности – в виде точки в прямоугольнике данной сущности.

Пример моделирования предметной области

С помощью рассмотренных выше понятий могут быть получены ER- модели для большинства схем баз данных в традиционных административно-управленческих приложениях. Отправными элементами для построения ER-модели предметной области очень часто являются используемые в организации документы (рис. 6.4).

Предположим, что определена предметная область: поставка товаров на склад. Пусть используемая форма поставки имеет вид, как на рис. 6.2. Покажем,

как, используя приведенную форму, можно построить концептуальную модель этой небольшой предметной области.

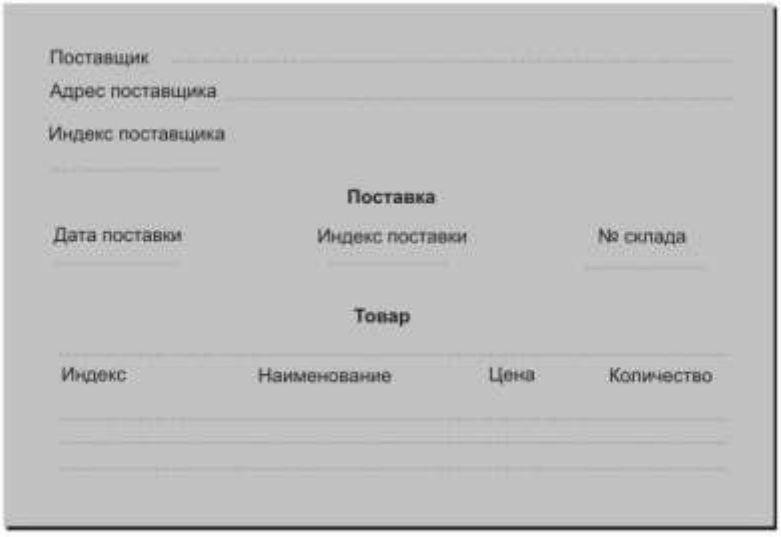

Рис.6.2. Форма поставки

Итак, анализируемая форма содержит следующую информацию: **Поставщик, Индекс поставщика, Адрес поставщика, Товар, Индекс товара, Наименование товара, Цена товара, Количество товара, Поставка, Индекс поставки, Дата поставки** и **Номер склада.**

Выделим две сущности: **ПОСТАВЩИК** и **ТОВАР** (рис. 6.3).

Рис.6.3. Предметная область ПОСТАВКА

Оставшиеся атрибуты характеризуют сущность – **ПОСТАВКА.**

Сформируем ее и установим определенные типы бинарных связей между тремя сущностями, исходя из следующих рассуждений: один и тот же поставщик может осуществить ряд поставок, но каждая поставка осуществляется только одним поставщиком.

Мощность связи между сущностями **ПОСТАВКА** и **ТОВАР** должна быть установлена M:N, так как каждая поставка может содержать несколько товаров, и один и тот же товар может содержаться в нескольких поставках.

Исходя из вышесказанного, диаграмма модели предметной области **ПОСТАВКА** примет такой вид, как на рис. 6.4.

Атрибуты **Индекс поставщика, Индекс поставки** и **Индекс товара** были введены дли однозначной идентификации экземпляров рассматриваемых сущностей, так как ни один из остальных атрибутов не подходит на эту роль. Как уже упоминалось, такие идентификационные атрибуты называются первичными ключами.

При построении концептуальной модели следует избегать избыточности информации. После того, как выделены сущности, ключи, определяют и удаляют имеющиеся избыточные связи. Большое внимание уделяется анализу атрибутов. Забегая вперед, следует указать на то, что в хорошо спроектированной БД должно соблюдаться правило: среди атрибутов сущности должна наблюдаться зависимость описательного атрибута от ключевого, но не должна существовать зависимость между описательными атрибутами.

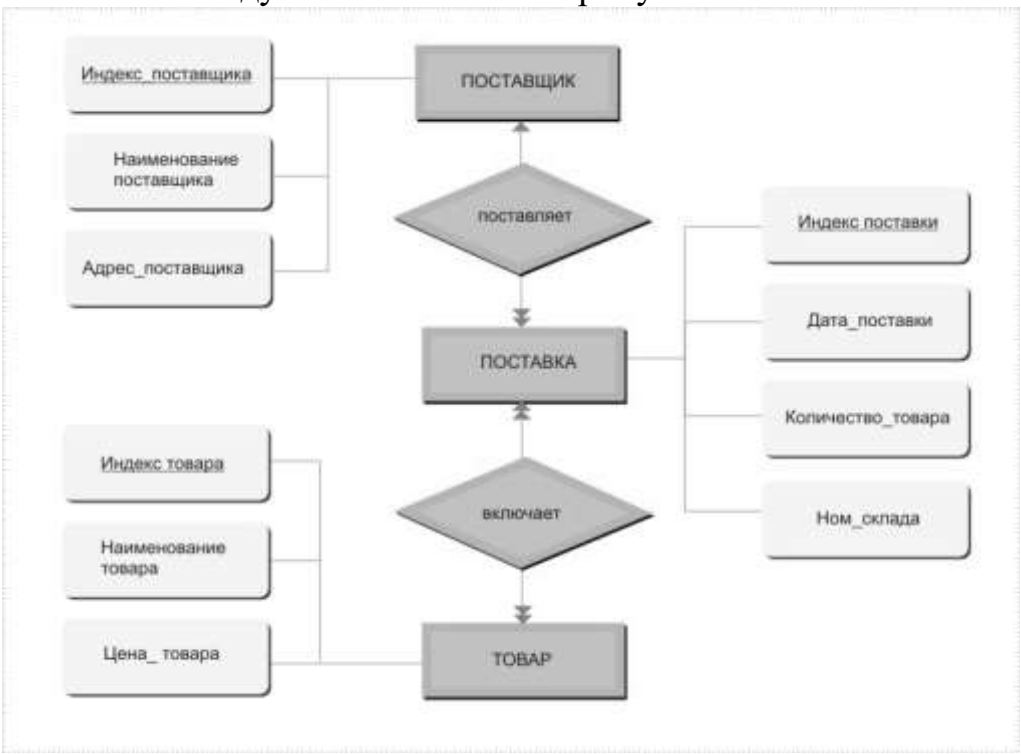

Рис.6.4. Диаграмма модели предметной области ПОСТАВКА

Завершающим этапом построения концептуальной модели исследуемой предметной области является спецификация всех сущностей, входящих в модель. Для данного примера результаты этого шага должны быть сведены к следующему:

1. Спецификация сущностей:

ПОСТАВЩИК: Индекс_поставщика– идентификационный атрибут
Адрес_поставщика – описательный атрибут
Наименование_поставщика – описательный атрибут

ПОСТАВКА: **Индекс_поставки** – идентификационный атрибут
Количество_товара– описательный атрибут
Дата_поставки – описательный атрибут
Номер_склада – описательный атрибут
ТОВАР: **Индекс_товара** – идентификационный атрибут
Наименование_товара – описательный атрибут
Цена_товара – описательный атрибут
2. Спецификация типов связей:
ПОСТАВЛЯЕТ: связь **ПОСТАВЩИК** <-----> **ПОСТАВКА** 1: N
ВКЛЮЧАЕТ: связь **ПОСТАВКА** <-----> **ТОВАР** M: N
3. Спецификация атрибутов табл. 6.1:

Таблица 6.1

Спецификация атрибутов

Название атрибута	Тип данных	Размер	Ограничение на значение
Индекс_поставщика	символьный	6	
Адрес_поставщика	символьный	50	
Цена_товара	денежный		
Количество_товара	числовой, целое число	5	>= 0

Задание

1. Построить концептуальную модель базы данных в виде диаграммы «сущность-связь» для варианта задания (определить самостоятельно и описать в отчете). Определите атрибуты каждой сущности в модели, указав для них тип данных, размер, ограничение на значение. Оформить в отчете, как первое задание.

2. Построить концептуальную модель базы данных для хранения информации о пользователях сайта (вариант для данных лабораторной 7). С использованием phpMyAdmin или любого инструментального программного обеспечения для работы с СУБД MySQL создать разработанную БД. В отчете привести SQL-код для ее создания, а также запросы для добавления и модификации данных.

Контрольные вопросы

1. Какие функции выполняет СУБД?
2. По какому принципу работают клиент-серверные СУБД?
3. В какой форме хранятся данные в базе данных?
4. Что такое запрос?
5. Какие операторы используются для извлечения, добавления, изменения и удаления данных в базе данных?
6. Концептуальная модель?

Лабораторная работа № 7

Организация взаимодействия с СУБД MySQL

Цель работы: получение практических навыков использования баз данных средствами языка программирования php.

Краткие теоретические сведения

Язык PHP имеет большой набор программных интерфейсов для различных СУБД. В список поддерживаемых СУБД входят MS SQL Server, PostgreSQL, MySQL, Oracle (начиная с PHP 4.3.0), ODBC и несколько других. Для каждой из них в PHP имеется достаточно большой набор функций.

Функции PHP для работы с СУБД MySQL.

`resource mysql_connect ([string server [,string username [,string password [,bool new_link [,int client_flags]]]]])` – открывает соединение с сервером MySQL.

В случае успеха возвращает идентификатор соединения, иначе - `False`. Для открытия постоянного соединения используется функция `mysql_pconnect()`, параметры и возвращаемые значения те же, что и для `mysql_connect()`.

`bool mysql_select_db (string database_name[,resource link_identifier])` – выбор базы данных.

`string mysql_error()` – возвращает текст сообщения об ошибке, произошедшей после последней операции MySQL.

`resource mysql_query (string query [,resource link_identifier])` – отправляет запрос к базе данных, которая определяется идентификатором `link_identifier`. Если он не указан, запрос отправляется через последнее установленное соединение. Если установленных соединений нет, функция пытается установить соединение так же, как и функция `mysql_connect()` без параметров. Функция возвращает ресурс только для операторов `SELECT, SHOW, EXPLAIN, DESCRIBE`. Для остальных запросов – `True`; в случае неудачи для всех видов запросов – `False`.

`array mysql_fetch_array(resource result [, int result_type])` – возвращает массив из значений ячеек строки результирующей таблицы. Таблица является результатом SQL-запроса и хранится в ресурсе `result`. Тип возвращаемого массива определяется указанием одной из констант `MYSQL_ASSOC` (ассоциативный), `MYSQL_NUM` (индексированный), `MYSQL_BOTH` (ассоциативный и индексированный одновременно, по умолчанию). Имена полей таблицы чувствительны к регистру.

`int mysql_insert_id([int link_id])` – возвращает последнее число сгенерированное в поле, имеющего свойство Автоинкремента после выполнения запроса INSERT.

`int mysql_affected_rows([int link_id])` – возвращает число измененных записей после выполнения запросов DELETE, INSERT, UPDATE или REPLACE.

Пример. Вывести содержимое таблицы my_table из базы данных my_database в Web-браузер.

```
<?php
    /* Соединение с СУБД и выбор базы данных */
    $link = mysql_connect("localhost", "root ", "")
        or die("Соединение невозможно: " . mysql_error());
    mysql_select_db("my_database") or die("Невозможно выбрать
БД");
    /* Выполнение SQL-запроса */
    $query = "SELECT * FROM my_table";
    $result = mysql_query($query) or die("Не удалось выполнить за-
прос: " . mysql_error());

    /* Вывод результатов в формате HTML */
    echo "<table>\n";
    while ($line = mysql_fetch_array($result, MYSQL_ASSOC)) {
        echo "\t<tr>\n";
        foreach ($line as $col_value)
        {
            echo "\t\t<td>$col_value</td>\n";
        }
        echo "\t</tr>\n";
    }
    echo "</table>\n";

    /* Освобождение памяти, содержащей результат запроса */
    mysql_free_result($result);

    /* Завершение соединения */
    mysql_close($link);
?>
```

Задание

Для базы данных, созданной в предыдущей лабораторной, реализовать на языке php следующие функции:

1. добавление нового пользователя. Для добавления пользователя необходимо создать форму, содержащую поля для информации, необходимой при регистрации пользователя, и файл с программой на php, осуществляющей непосредственное внесение информации в базу данных;
2. вывод списка существующих пользователей;
3. удаление выбранного пользователя из БД;
4. редактирование информации о пользователе.

Выполнение работы

1. Реализация добавления нового пользователя.

В соответствии с заданием необходимо создать форму для ввода данных о пользователе, необходимых при регистрации. Обычно такими данными являются имя пользователя (логин) и пароль. Для ввода логина используется обычное текстовое поле, а для ввода пароля – поле с типом «password», символы в котором отображаются в виде точек или любого другого символа, чтобы исключить возможность подсмотреть пароль, однако при этом пароль не видит и тот человек, который его набирает, поэтому обычно предлагается ввести пароль 2 раза, что уменьшает вероятность возникновения ошибки.

Создадим файл с именем «form_registr.php», который будет содержать описанную форму для регистрации нового пользователя. Ниже на рис. 7.1 приведён его html код и вид в браузере.

Рис.7.1. Файл с именем «form_registr.php»

Атрибут action данной формы указывает файл, которому будут переданы данные, введённые на форме. В качестве такого файла указан «new_user.php», соответственно в нём и должна располагаться программа, добавляющая в базу данных нового пользователя.

Прежде чем выполнять любые действия с базой данных необходимо подключиться к серверу, на котором запущена СУБД MySQL и выбрать нужную базу данных из всех созданных на сервере. Для этого используются функции mysql_connect и mysql_select_db. Так как подключение к базе данных необходимо будет производить для выполнения каждой из функций (добавление, удаление, редактирование и вывод списка всех пользователей), то целесообразно реализовать его в отдельном файле и подключать его, по мере необходимости, к файлам, в которых будут использоваться база данных. Этот файл будет называться «db.php» и содержать следующий код (рис. 7.2).

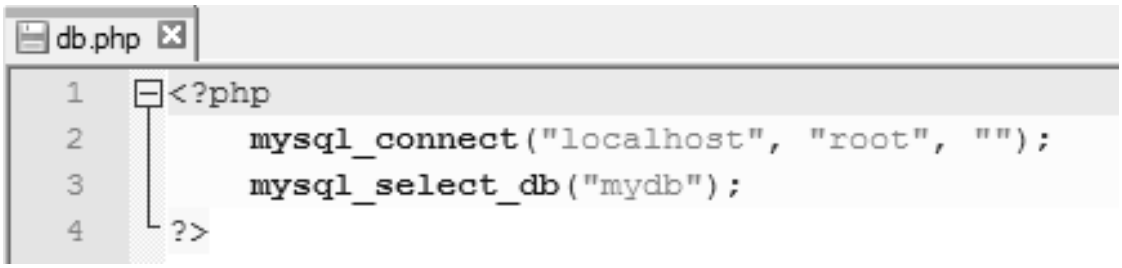

Рис.7.2. Файл с именем «db.php»

В нём у функции `mysql_connect` имеются 3 параметра:

– адрес сервера – localhost означает, что сервер MySQL находится на том же компьютере, на котором будет запущена данная программа;

– логин – root принадлежит пользователю, обладающему полными правами на доступ ко всем базам данных на сервере (аналог администратора);

– последний параметр – пароль оставлен пустым, так как пользователь root изначально не имеет пароля.

Функция `mysql_select_db` имеет в данном случае только один параметр – имя базы данных, которое должно соответствовать реально имеющейся базе данных.

После этого можно создать файл «new_user.php» и подключить к нему «db.php» с помощью функции require_once("db.php"). Прежде чем добавлять пользователя в базу данных необходимо выполнить проверить, заданы ли все необходимые для этого значения, и соответствуют ли пароли и его подтверждение, введённые на форме. Программный код данного файла приведён на рис. 7.3.

new_user.php

```php
<?php
    require_once("db.php");

    if(empty($_POST["username"]) or empty($_POST["password1"]) or empty($_POST["password2"]))
    {
        echo "Заданы не все поля";
    }
    else if($_POST["password1"] != $_POST["password2"])
    {
        echo "Пароли не совпадают";
    }
    else
    {
        $usermane=$_POST["username"];
        $password=$_POST["password1"];
        $query="INSERT INTO users (username, password) VALUES ('$usermane', '$password')";
        if(mysql_query($query)==true)
        {
            echo "Пользователь успешно добавлен";
        }
        else
        {
            echo "Ошибка при добавлении пользователя";
        }
    }
?>
```

Рис.7.3. Файл с именем «new_user.php»

В данном файле использована функция mysql_query, принимающая в качестве параметра текст SQL-запроса на добавление. В случае успешного выполнения такого запроса данная функция возвращает результат true, а в случае

неуспешно – false. Предполагается, что сведения о пользователях хранятся в таблице «users».

Для проверки работоспособности программы добавьте нескольких пользователей с использование формы регистрации и проверьте, появляются ли они в таблице «users» базы данных.

2. Реализация вывода списка существующих пользователей.

Для вывода списка пользователей создадим файл «show_users.php», содержащий следующий код (рис. 7.4)

show_users.php

```
<?php
    require_once("db.php");

    $query="SELECT * FROM users";
    $results=mysql_query($query);

    echo "<table border='1'>";
    while($row=mysql_fetch_assoc($results))
    {
        $username=$row["username"];
        $id=$row["id"];
        echo "<tr><td>$username</td>
            <td><a href='edit_user.php?id=$id'>редактировать</a></td>
            <td><a href='delete_user.php?id=$id'>удалить</a></td></tr>";
    }
    echo "</table>";
?>
```

Рис.7.4. Файл с именем «show_users.php»

В данном файле также используется функция mysql_query, однако, так как ей в качестве параметра передан запрос на извлечение, то результатом её работы будет указатель на результаты запроса, представляющие собой таблицу.

Функция mysql_fetch_assoc возвращает ассоциативный массив, соответствующий очередной строке из результатов запроса, переданных ей в качестве параметра (в данном случае $results) и сдвигает вперед внутренний указатель. То есть при следующем вызове данной функции она вернёт массив, соответствующий следующей строке, поэтому вызов этой функции в цикле позволяет перебрать все строки результата запроса.

При выводе результаты формируются в виде таблицы, в которой в первом столбце указано имя пользователя, а в остальных ссылки, нажатие на которые позволит либо удалить пользователя, либо перейти к форме его редактирования. При этом ко всем ссылкам дописан параметр id, передаваемый методом GET, который будет использован для идентификации пользователя, которого необходимо удалить, либо изменить.

3. Реализация удаления пользователей.

Как видно из предыдущей части ссылка «удалить» ведёт на страницу с адресом «delete_user.php», следовательно, необходимо создать файл с таким именем, содержащий следующий код (рис. 7.5).

delete_user.php

```php
<?php
    require_once("db.php");

    if(empty($_GET["id"]))
    {
        echo "Не указан пользователь";
    }
    else
    {
        $id=$_GET["id"];

        $query="DELETE FROM users WHERE id=$id";
        mysql_query($query);

        header("Location:show_users.php");
    }
?>
```

Рис.7.5. Файл с именем «delete_user.php»

В него в начале включается файл «db.php», содержащий все необходимые функции для подключения к базе данных, затем происходит проверка, передан ли id пользователя, которого необходимо удалить и, если он передан, то выполняется запрос на удаление.

Следует обратить внимание на функцию header, расположенную в конце файла, она используется для отправки http-заголовков, предназначенных для передачи служебной информации от сервера к браузеру. Заголовок «Location:» перенаправляет браузер на указанную после него страницу. В данном случае после удаления пользователя происходит переход обратно на страницу со списком всех пользователей. Важно, что перед отправкой заголовка программой не должен осуществлять вывод на страницу чего-либо.

4. Реализация редактирования пользователей.

Как видно из php-кода, реализующего вывод списка пользователей, ссылка «редактировать» ведёт на страницу «edit_user.php», следовательно, необходимо создать файл с таким именем, содержащий следующий код (рис. 7.6)

edit_user.php

```php
<?php
    require_once("db.php");

    if(empty($_GET["id"]))
    {
        echo "Не указан пользователь";
    }
    else
    {
        $id=$_GET["id"];

        $query="SELECT * FROM users WHERE id=$id";
        $results=mysql_query($query);

        $username=mysql_result($results, 0, "username");
        $password=mysql_result($results, 0, "password");

        echo "Информация о пользователе<br>";
        echo "id: $id<br>";
        echo "имя пользователя: $username<br>";
        echo "текущий пароль: $password<br>";
        echo "
            <br>
            Задать новый пароль
            <form method='post' action='update_user.php?id=$id'>
                Пароль: <input type='password' name='password1'><br>
                Пароль еще раз: <input type='password' name='password2'><br>
                <input type='submit' value='Задать'>
            </form>
        ";
    }
?>
```

Рис.7.6. Файл с именем «edit_user.php»

Помимо стандартного подключения файла, отвечающего за соединение с базой данных, проверок переданного значения id пользователя и выполнения запроса с помощью функции mysql_query, здесь также присутствует функция mysql_result, которая возвращает значение одной ячейки результата запроса. В качестве параметров ей передаются:

– указатель на результаты запроса ($result), полученные с помощью функции mysql_query;
– номер строки в результатах запроса (0);
– название столбца (username, а затем password).

Так как при выборке данных в запросе использовался идентификатор пользователя (id), являющийся первичным ключом таблицы, а значит уникальным для каждой строки, то результатом выполнения такого запроса всегда будет только одна строка, поэтому номер строки в данном случае равен 0.

Внизу файла расположена форма для ввода нового пароля и его подтверждения, предназначенная для изменения существующего пароля и в некоторой

части повторяющая форму регистрации. При нажатии на кнопку «Задать» данной формы происходит переход на страницу «update_user.php», что указано в её атрибуте «action».

Создадим файл с названием «update_user.php», содержащий следующий программный код (рис. 7.7)

update_user.php

```php
<?php
    require_once("db.php");

    if(empty($_GET["id"]))
    {
        echo "Не указан пользователь";
    }
    else if($_POST["password1"] != $_POST["password2"])
    {
        echo "Пароли не совпадают";
    }
    else
    {
        $id=$_GET["id"];
        $newpassword=$_POST["password1"];

        $query="UPDATE users SET password='$newpassword' WHERE id=$id";
        if (mysql_query($query)==true)
        {
            echo "Данные изменены";
        }
        else
        {
            echo "Ошибка при изменении данных";
        }
    }
?>
```

Рис.7.7. Файл с именем «update_user.php»

По аналогии с предыдущими фалами, в начале выполняется подключение в базе данных, затем проверка совпадения паролей, а затем выполняется запрос на обновление таблицы пользователей и, в случае его успеха, выводится соответствующее сообщение.

Для проверки работы всех реализованных функций необходимо изменить пароли нескольких пользователей, а затем удалить их.

Задание для самостоятельного выполнения

1. Реализовать проверку при регистрации нового пользователя, совпадает ли его имя с именем уже существующего в базе данных пользователя.

2. При удалении пользователя реализовать проверку, существует ли в базе данных пользователь с id, переданным в качестве параметра. Так как id передаётся в адресной строке, пользователь может с лёгкостью поменять его значение.

3. Добавить базу данных не менее 2-х полей, характеризующих пользователя (например, настоящее имя и возраст) и реализовать возможность их редактирования в форме изменения информации о пользователе (файлы «edit_user.php» и «update_user.php»).

Контрольные вопросы

1. Какие функции PHP используются для взаимодействия с СУБД MySQL?
2. Какие параметры необходимо обязательно указывать при подключении к БД?
3. Какие функции используются для обработки результатов запроса, представляющих собой таблицу, состоящую из множества строк?

Лабораторная работа № 8
Авторизация пользователей сайта

Цель работы: получение практических навыков использования средств языка php и баз данных для авторизации пользователей.

Задание

С использованием базы данных пользователей и php-скриптов, разработанных на предыдущих лабораторных работах, реализовать возможности регистрации и авторизации пользователей сайта. Для администратора обеспечить возможность просмотра и изменения данных всех пользователей.

Выполнение работы

Для оформления работы в виде сайта необходимо использовать одинаковый шаблон для всех страниц. В качестве такого шаблона подойдёт html-страница, разработанная в лабораторной работе № 2, которая будет использоваться в качестве основной страницы создаваемого сайта (рис. 8.1). Для этого необходимо скопировать HTML и CSS файls из 2-ой лабораторной в папку с остальными файлами, разработанными на предыдущей лабораторной работе, и переименовать в «index.php». Файлы с именем «index» открываются автомати-

чески, если пользователь в адресной строке не указал конкретную страницу после названия сайта.

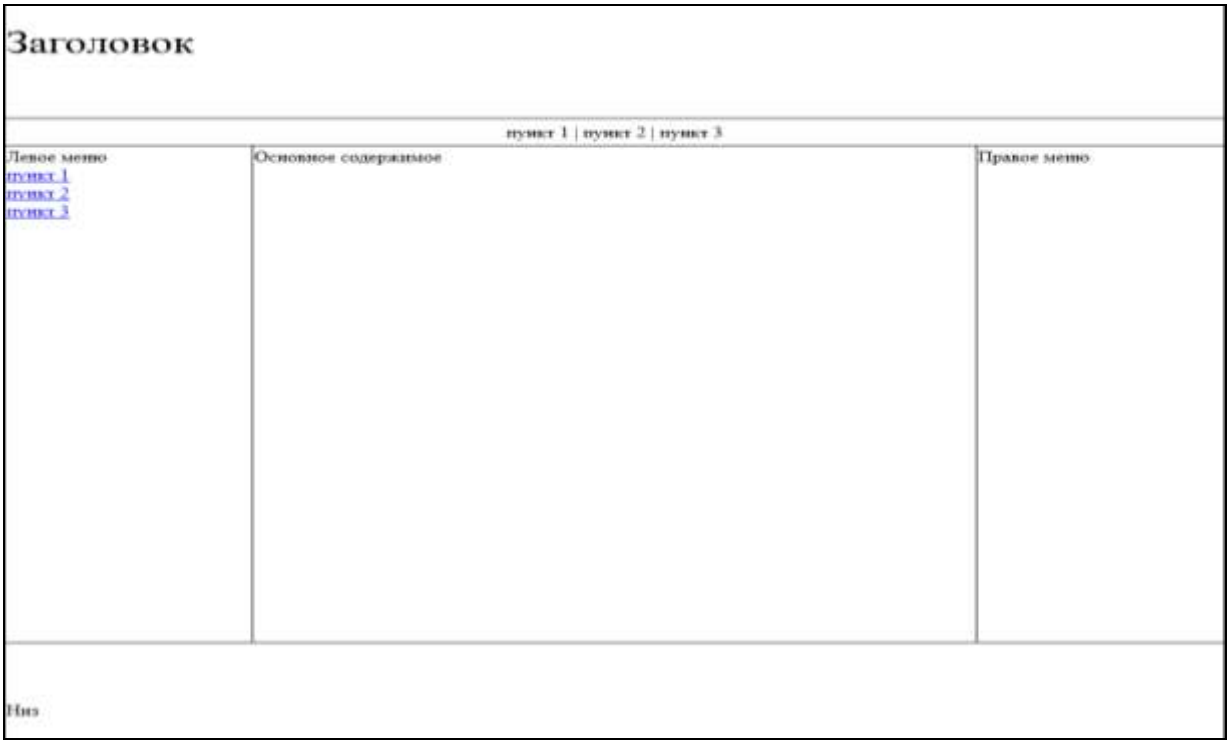

Рис.8.1. Лабораторная работа № 2

Будем использовать ячейку с надписью: «Правое меню» для вывода формы логина (входа на сайт). Для этого создадим файл «form_login.php», со следующим кодом (рис. 8.2).

form_login.php

```
1  <b>Войти на сайт</b>
2  <form method="post" action="login.php">
3      Логин:<br><input type="text" name="login"><br><br>
4      Пароль:<br><input type="password" name="password"><br><br>
5      <input type="submit" value="Войти">
6  </form>
```

Рис.8.2. Файл с именем «form_login.php»

Для отображения этой формы на месте правого меню необходимо подключить файл «form_login.php» в соответствующее место файла «index.php» (рис.8.3) с использованием функции require_once().

index.php

```
18                </td>
19            </tr>
20                <td id="left_menu">
21                    Левое меню<br>
22                    <a href="">пункт 1</a><br>
23                    <a href="">пункт 2</a><br>
24                    <a href="">пункт 3</a>
25                </td>
26                <td id="center">
27                    Основное содержимое
28                </td>
29                <td id="right_menu">
30                    <?php require_once("form_login.php");?>
31                </td>
32            <tr>
33                <td colspan=3 id="footer">
34                    Низ
35                </td>
36            </tr>
37        </table>
```

было

добавили

было

Рис.8.3. Подключение файла «form_login.php»

В результате главная страница будет выглядеть в соответствии с рис. 8.4.

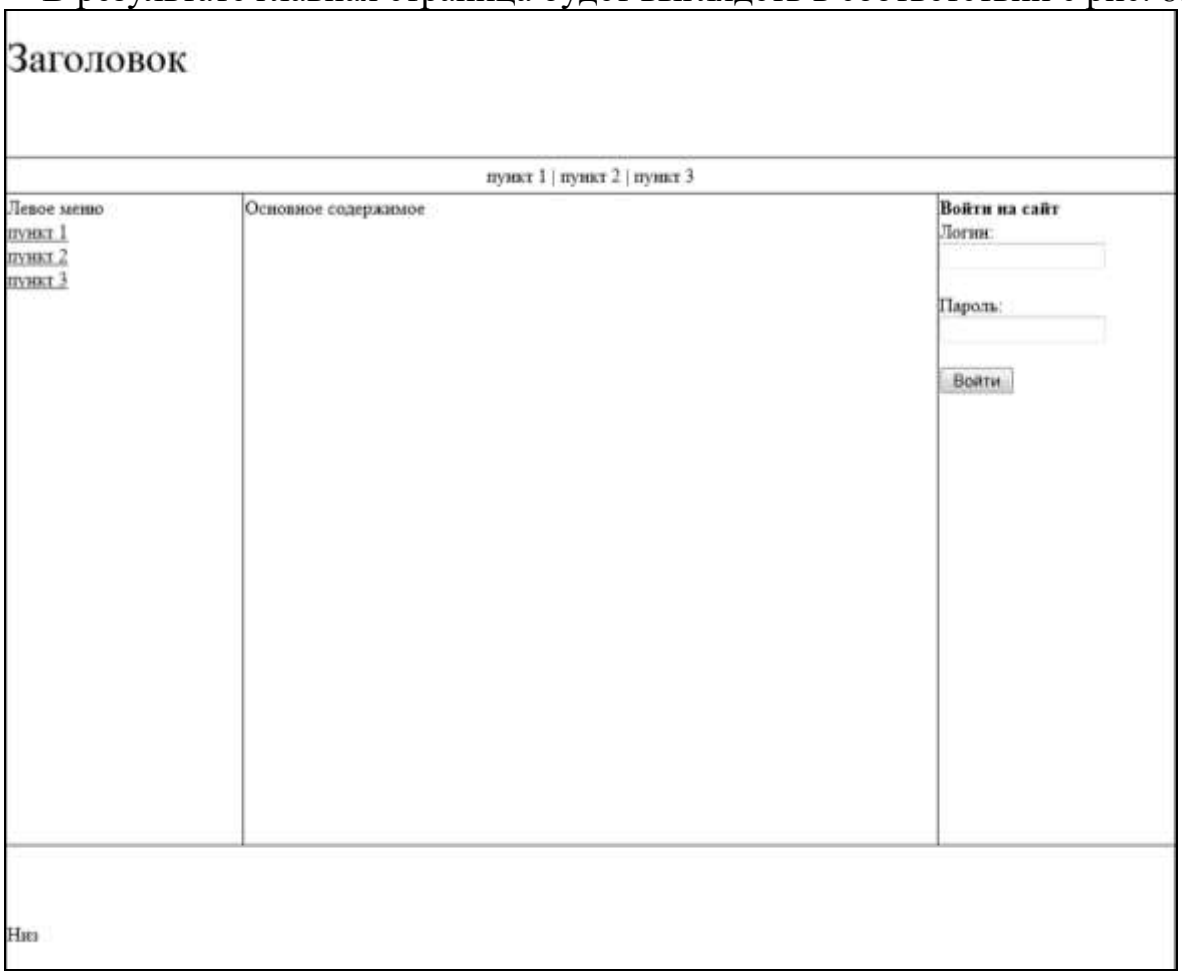

Рис.8.4. Внешний вид главной страницы после изменения

Далее необходимо обеспечить возможность регистрации новых пользователей. Для этого служит форма «form_registr.php», созданная на предыдущей лабораторной работе, однако размещаться она должна в поле, предназначенном для основного содержимого и быть доступной по ссылке.

Создадим ссылку, позволяющую открыть форму регистрации, для этого в правое меню ниже формы входа впишем следующую строку, рис.8.5.

index.php

```
18                </td>
19            </tr>
20                <td id="left_menu">
21                    Левое меню<br>
22                    <a href="">пункт 1</a><br>
23                    <a href="">пункт 2</a><br>
24                    <a href="">пункт 3</a>
25                </td>
26                <td id="center">
27                    Основное содержимое
28                </td>
29                <td id="right_menu">
30                    <?php require_once("form_login.php");
31                          echo "<a href='index.php?action=register'>Зарегистрироваться</a>";
32                    ?>
33                </td>
34            <tr>
35                <td colspan=3 id="footer">
36                    Низ
37                </td>
38            </tr>
39        </table>
```

Рис.8.5. Созданная ссылка, позволяющая открыть форму регистрации

Как видно, эта ссылка ведёт на страницу «index.php» и передаёт ей методом GET параметр action (действие) равный register (регистрация). Обработку этого параметра необходимо реализовать в файле «index.php», для этого в поле для основного содержания впишем следующий код (рис.8.6)

index.php

```
18                </td>
19            </tr>
20                <td id="left_menu">
21                    Левое меню<br>
22                    <a href="">пункт 1</a><br>
23                    <a href="">пункт 2</a><br>
24                    <a href="">пункт 3</a>
25                </td>
26                <td id="center">
27                    <?php
28                        if(!empty($_GET["action"]))
29                        {
30                            if($_GET["action"]=="register")
31                            {
32                                require_once("form_registr.php");
33                            }
34                        }
35                    ?>
36                </td>
37                <td id="right_menu">
38                    <?php require_once("form_login.php");
```

Рис.8.6. Обработка параметра action (действия)

Данный код сначала проверяет, передана ли переменная «action», затем, если она передана, проверяет её значение, и, если оно равно «register», подключает уже имеющуюся форму регистрации.

Теперь при нажатии на ссылку «Зарегистрироваться» будет отображаться страница, приведённая на рис. 8.7.

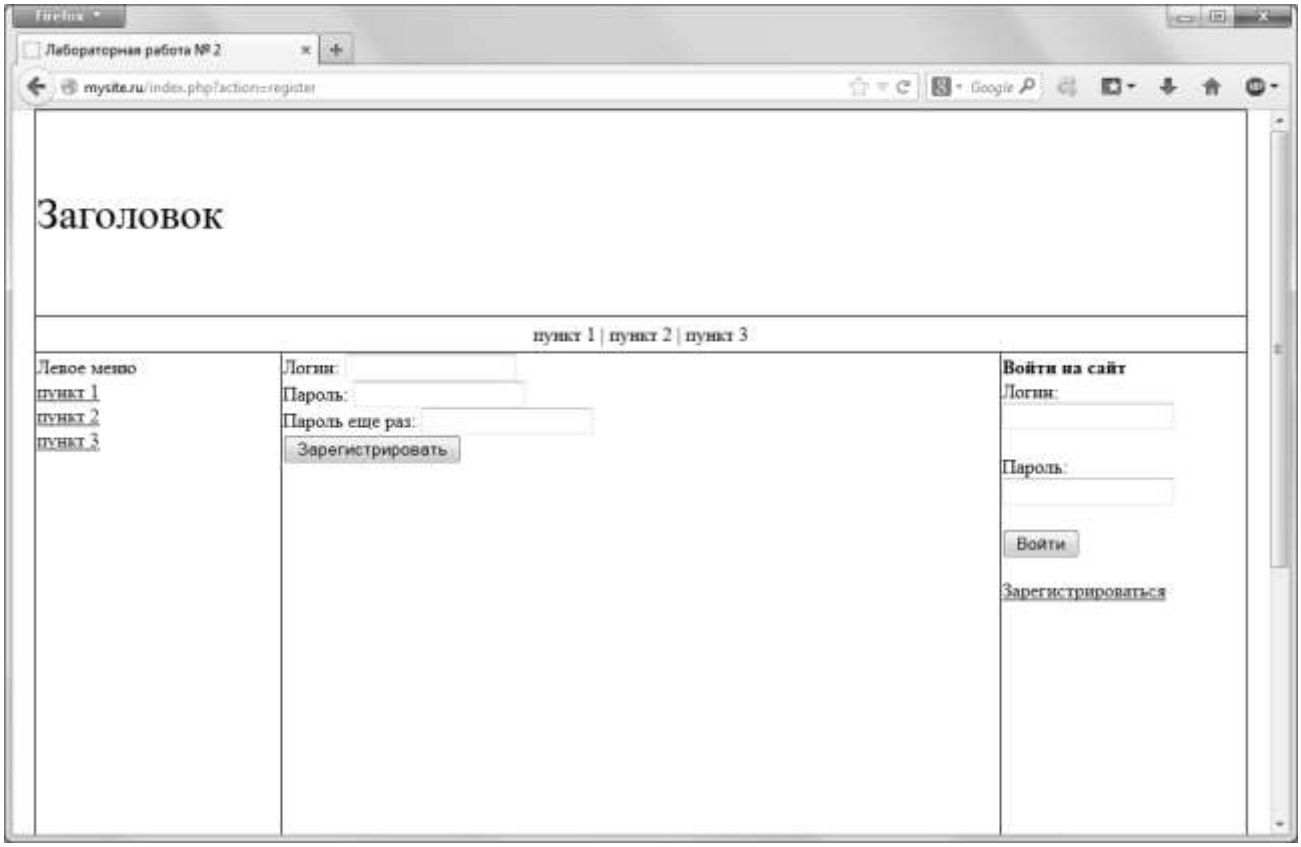

Рис.8.7. Внешний вид главной страницы после изменения

Если у Вас в базе данных нет ни одного пользователя, необходимо его зарегистрировать прежде чем продолжать дальше.

Как видно из атрибута action формы входа, при нажатии на кнопку «Войти» будет происходить переход на страницу с именем «login.php» и ей методом POST будут переданы переменные login и password. Соответственно необходимо создать страницу с таким именем, реализующую обработку данных переменных, и все остальные функции, необходимые для авторизации пользователя. Исходный код страницы «login.php» приведён на рис. 8.8.

login.php

```php
<?php
    require_once("db.php");

    if(empty($_POST["login"]) or empty($_POST["password"]))
    {
        echo "Указаны не все данные";
    }
    else
    {
        $login=$_POST["login"];
        $password=$_POST["password"];

        $query="SELECT * FROM users WHERE username='$login'";
        $res=mysql_query($query);
        if(mysql_num_rows($res)==0)
        {
            echo "Пользователь не найден";
        }
        else
        {
            if(mysql_result($res, 0, "password")==$password)
            {
                session_start();
                $_SESSION["user_id"]=mysql_result($res, 0, "id");
                $_SESSION["user_name"]=mysql_result($res, 0, "username");
                header("Location:index.php");
            }
            else
            {
                echo "Неправильный пароль";
            }
        }
    }
?>
```

Рис.8.8. Файл с именем «login.php»

В нём описаны следующие основные действия:

– строка 4. Проверка, переданы ли значения переменных login и password.

– строка 13. Запрос, выбирающий из базы данных пользователя, с логином, указанным на форме входа.

– строка 15. Проверка количества срок, выбранных запросом. Если их количество равно 0, то такого пользователя в базе данных нет, о чём выдаётся соответствующее сообщение в строке 17.

– строка 21 выполняется если пользователь найден в базе данных. В ней извлекается пароль данного пользователя, хранимый в базе данных, и сравнивается с паролем, переданным с формы входа. Если пароли совпадают, то со-

здаются переменные сессии, которые позволят идентифицировать этого пользователя на любой странице сайта (строки 24, 25) и происходит переход обратно на главную страницу (строка 26). Если пароль не совпадает, то в строке 30 выдаётся соответствующее сообщение.

Если проверить работу сайта на данном этапе, то можно увидеть, что после выполнения входа сессия действительно создаётся, о чём говорит наличие cookie с именем «PHPSESSID» в браузере и файла с переменными в служебных файлах web-сервера. Причём имя этого файла совпадает со значением идентификатора сессии «PHPSESSID» (рис. 8.9).

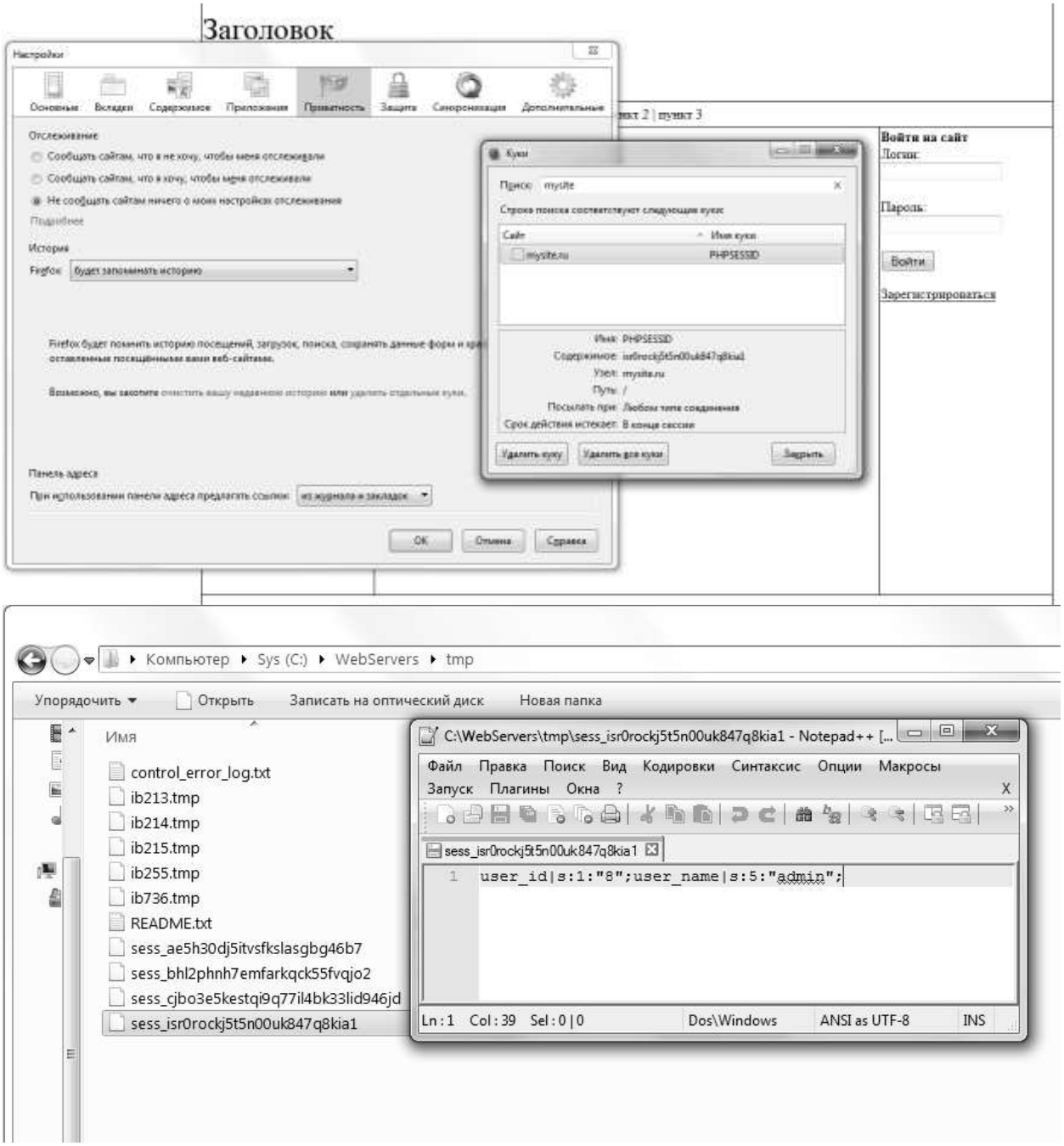

Рис.8.9. Cookie с именем «PHPSESSID» в браузере

Однако визуально, никаких отличий в отображении сайта для авторизованного и неавторизованного пользователя нет. Даже после входа пользователь продолжает видеть форму входа. Вместо неё целесообразно было бы показывать приветствие и ссылки на страницы для управления своим аккаунтом. Для этого создадим файл «user_info.php» со следующим содержанием, рис. 8.10.

user_info.php

```
<?php
    echo "Здравствуйте, ".$_SESSION["user_name"];
    $id=$_SESSION["user_id"];
    echo "<br><a href='index.php?action=edituser&id=$id'>Изменить личные данные</a>";
    echo "<br><a href='logout.php'>Выйти</a>";
?>
```

Рис.8.10. Файл с именем

Так как здесь используются переменные сессии, то для правильной работы приведённого программного кода необходимо получить доступ к этим переменным с помощью функции session_start(), однако данная функция должна выполняться до вывода какой-либо информации на экран, а файл, в котором она используется, включается в «index.php» после вывода части страницы, следовательно использовать функцию session_start() нужно в начале файла «index.php», рис. 8.11.

index.php

```
<?php session_start()?>
<!DOCTYPE HTML PUBLIC ""-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=cp-1251">
        <link rel="stylesheet" type="text/css" href="lab2_.css"/>
        <title>Лабораторная работа № 2</title>
    </head>
    <body>
        <table id="main_table">
```

Рис.8.11. Добавление функции session_start()

В созданном файле «user_info.php» также имеются 2 ссылки:

- ссылка на страницу изменения данных пользователя,
- ссылка на страницу для выхода.

Изменение сведений о пользователе уже было реализовано в предыдущей лабораторной, теперь необходимо реализовать отображение формы редактирования данных пользователя в центральной части страницы. Данная форма содержится в файле «edit_user.php». При нажатии на ссылку «Изменить личные

данные» происходит переход на страницу «index.php», на которую передаётся переменная «action» равная «edituser», следовательно, необходимо реализовать проверку значения данной переменной и, если она содержит значение «edituser», то подключить файл «edit_user.php». Для этого в файл «index.php» добавим код, выделенный на рис. 8.12.

index.php

```
19                  </td>
20              </tr>
21                  <td id="left_menu">
22                      Левое меню<br>
23                      <a href="">пункт 1</a><br>
24                      <a href="">пункт 2</a><br>
25                      <a href="">пункт 3</a>
26                  </td>
27                  <td id="center">
28                      <?php
29                          if(!empty($_GET["action"]))
30                          {
31                              if($_GET["action"]=="register")
32                              {
33                                  require_once("form_registr.php");
34                              }
35                              else if ($_GET["action"]=="edituser")
36                              {
37                                  require_once("edit_user.php");
38                              }
39                          }
40                      ?>
41                  </td>
42                  <td id="right_menu">
43                      <?php
```

Рис.8.12. Проверка значения переменной «edituser»

Теперь при переходе по ссылке «Изменить личные данные» будет открываться страница со сведениями о текущем пользователе и форма для смены пароля. При этом в адресной сроке будет присутствовать запись вида:

«http://mysite.ru/index.php?action=edituser&id=1», где id=1 – идентификатор текущего пользователя.

Если вручную изменить его на идентификатор (id) любого другого, существующего в базе данных, пользователя, то откроется страница с данными этого пользователя, что недопустимо с точки зрения безопасности, следовательно, необходимо реализовать разграничение прав доступа.

Пусть видеть и изменять чужие личные данные могут только администраторы сайта, а остальные пользователи могут видеть только свои собственные данные, для этого необходимо добавить в таблицу users базы данных дополнительный столбец, обозначающий тип пользователя. Этот столбец будет называться type, иметь тип int и содержать 2 значения 0 – администратор, 1 – обычный пользователь. Новым пользователям по умолчанию должно устанавливаться значение 1.

Выбор из базы данных и занесение в сессию данного значения необходимо реализовать самостоятельно в файле «login.php» по аналогии с занесением id и имени пользователя.

Далее необходимо добавить проверку типа пользователя в форму редактирования сведений «edit_user.php». Внесённые изменения выделены на рисунке серым цветом. Если пользователь не является администратором и его id не совпадает с id пользователя, сведения о котором должны быть выведены, то выдаётся сообщение об ошибке, рис. 8.13.

edit_user.php

```
1  <?php
2      require_once("db.php");
3      if(empty($_GET["id"]))
4      {
5          echo "Не указан пользователь";
6      }
7      else if($_GET["id"]!=$_SESSION["user_id"] and $_SESSION["type"]!=0)
8      {
9          echo "Недостаточно прав для просмотра данных другого пользователя";
10     }
11     else
12     {
13         $id=$_GET["id"];
14
15         $query="SELECT * FROM users WHERE id=$id";
16         $results=mysql_query($query);
17
18         $username=mysql_result($results, 0, "username");
19         $password=mysql_result($results, 0, "password");
20
21         echo "Информация о пользователе<br>";
22         echo "id: $id<br>";
23         echo "имя пользователя: $username<br>";
24         echo "текущий пароль: $password<br>";
25         echo "
26             <br>
27             Задать новый пароль
```

Рис.8.13. Добавление проверки типа пользователя в форму редактирования сведений «edit_user.php»

Последним этапом будет реализация возможности выхода, например для смены пользователя. Ссылка «Выйти» ведёт на страницу «logout.php», которую необходимо создать и вписать в неё следующий код, рис. 8.14.

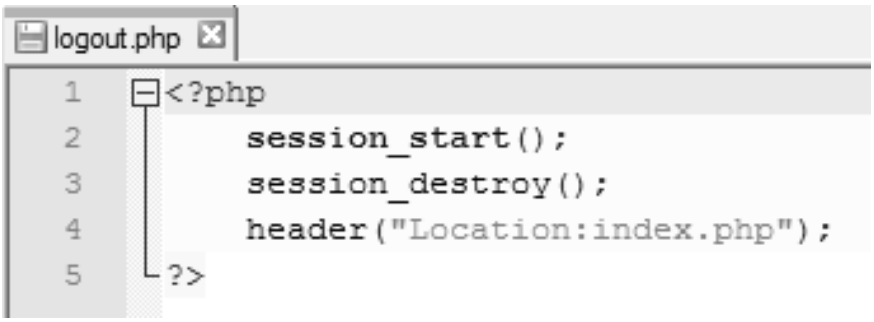

Рис.8.14. Файл с именем «logout.php»

Данный код уничтожает все переменные сессии и перенаправляет браузер на главную страницу.

Задание для самостоятельного выполнения

1. Реализовать вывод списка всех пользователей, разработанного на предыдущей лабораторной, в центральную часть страницы. Данная функция должна быть доступна только администраторам.

2. Реализовать проверку типа пользователя в файле «update_user.php», предназначенном для внесения изменений в базу данных. Внесение изменений должно быть доступно только администраторам.

3. Нарисовать схему переходов по страницам сайта.

Контрольные вопросы

1. Как работает функция require_once()?
2. Как работает функция header()?
3. Что такое сессии в PHP?
4. Как работают сессии?
5. Что такое PHPSESSID?
6. Как задать длительность сессии?
7. Как правильно завершить сессию?
8. Почему функцию session_start() необходимо вызывать в самом начале работы PHP-скрипта?

Лабораторная работа № 9

Обеспечение безопасности сайта

Цель работы: изучение методов защиты от внедрения SQL-кода и методов безопасного хранения паролей в базе данных.

Краткие теоретические сведения

Для защиты от внедрения SQL-кода необходимо фильтровать входные параметры, значения которых будут использованы для построения SQL-запроса.

1. Проверка типов значений

Для проверки переменной на числовое значение используется функция is_numeric(n);, которая вернёт true, если параметр n – число, и false в противном случае.

Так же можно не проверять значение на число, а вручную переопределить тип. Вот пример, переопределяющий значение $id, полученное от $_GET['id'] в значение целочисленного типа (в целое число):

```
$id=(int)$_GET['id'];
```

2. Экранирование ввода

Большинство взломов через SQL происходят по причине нахождения в строках «необезвреженных» кавычек, апострофов и других специальных символов. Для такого обезвреживания нужно использовать функцию addslashes($str);, которая возвращает строку $str с добавленным обратным слешем перед каждым специальным символом. Данный процесс называется экранизацией.

```
$a="пример текста с апострофом ' ";
echo addslashes($a); //будет выведено: пример текста с апо-
строфом '
```

Кроме этого существуют две функции, созданные именно для экранизации строк, используемых в SQL выражениях.

```
mysql_escape_string($str);
mysql_real_escape_string($str);
```

Первая не учитывает кодировку соединения с БД и может быть обойдена, а вот вторая её учитывает и абсолютно безопасна. mysql_real_escape_string($str); возвращает строку $str с добавленным обратным слешем к следующим символам: x00, n, r, , ', " и x1a.

Магические кавычки

Магические кавычки – эффект автоматической замены кавычки на обратный слэш и кавычку при операциях ввода/вывода. В некоторых конфигурациях PHP этот параметр включён, а в некоторых нет. Для того, что бы избежать двойного экранизирования символов и заэкранизировать данные по-нормальному через mysql_real_escape_string($str);, необходимо убрать автоматические проставленные обратные слеши (если магические кавычки включены).

Проверка включённости магических кавычек для данных получаемых из GET, POST или Куков организуется через функцию get_magic_quotes_gpc(); (возвращает 1 – если магические кавычки включены, 0 – если отключены).

Если магические кавычки включены (т.е обратные слеши добавляются) и такое встречается чаще, то их нужно убрать. Это делается через функцию stripslashes($str); (возвращает строку $str без обратных слешей у кавычек и прямых апострофов).

Пример кода с полной экранизацией строк для записи в БД:

```
if(get_magic_quotes_gpc()==1)
{
$element_title=stripslashes(trim($_POST["element_title"]));
$element_text=stripslashes(trim($_POST["element_text"]));
$element_date=stripslashes(trim($_POST["element_date"]));
}
else
{
$element_title=trim($_POST["element_title"]);
$element_text=trim($_POST["element_text"]);
$element_date=trim($_POST["element_date"]);
}

$element_title=mysql_real_escape_string($element_title);
$element_text=mysql_real_escape_string($element_text);
$element_date=mysql_real_escape_string($element_date);
```

Задание

1. Исключить возможность внедрения SQL-кода путём экранирования ввода всех переменных, используемым в запросах.
2. Реализовать хранение паролей в базе данных в виде md5-хешей с дополнительными случайными символами.

Контрольные вопросы

1. Что такое SQL-инъекция?
2. Какие методы используются для защиты от SQL-инъекций?
3. Для чего предназначены функции get_magic_quotes_gpc(), stripslashes() и mysql_real_escape_string()?
4. Что такое md5 и sha1?
5. В чём принципиальное отличие хеш-функций от шифрования?
6. Почему недостаточно хранить пароли только в виде хешфункций?
7. Приведите примеры атак на сайты.
8. Методы хранения паролей в базе данных.

Лабораторная работа № 10
Введение в язык JavaScript

Цель работы изучить способы использования языка программирования JavaScript для динамического изменения содержимого html-страниц.

Краткие теоретически сведения

Язык JavaScript – это язык написания сценариев, используемый для оформления документации HTML. Благодаря использованию сценариев на языке JavaScript, статические Web-страницы приобретают высококачественное анимационное оформление, возможность интерактивной работы с вводом-выводом данных и так далее.

Язык JavaScript является интерпретируемым, т. е. программа выполняется интерпретатором сразу после чтения текста программы в отличие от компилируемых языков, в которых сначала формируется исполняемый код. Этот язык также является объектно-ориентированным, хотя его возможности значительно меньше возможностей C++.

Расположение скрипта в HTML-документе. Программы на JavaScript можно встраивать непосредственно в теги при помощи обработчиков событий или в качестве значения атрибута ***href*** тега **<a>**. Однако чаще всего одного-двух операторов для решения задачи недостаточно, и включение программ в теги неудобно.

Для того чтобы отделить код программы на JavaScript (или любом другом языке скриптов) от остальной части страницы в спецификацию HTML был введен *контейнер* **<script>…</script>**. В него заключается текст программы. Все операторы этой программы будут выполняться сразу после загрузки документа в браузер вне зависимости от того, где расположен контейнер – в заголовке или в теле документа.

Тег **<script>** имеет атрибут ***language***, который для большинства браузеров по умолчанию имеет значение “JavaScript”, однако явное указание этого атрибута является хорошим тоном. Текст скрипта может также находиться и в отдельном файле. Тогда URL файла указывается с помощью атрибута src: <**script *src*=“modul.js”> </script>.**

Комментарии в скрипте. В языке JavaScript используется два вида комментариев: с двойным слешем и сочетанием слеша и звездочки. Например, *// это комментарий до конца строки;*

/ комментарий в строке */.*

Операторы JavaScript. *Операторы* в JavaScript в большинстве совпадают с операторами языка Си (+, –, *, /, %, +=, –=, ==, !=, <, >, <= и т. д.). Кроме того, оператор сложения (+) применяется для конкатенации строк. Например: s = “string1”+“string2”.

Управление потоком вычислений осуществляется набором следующих операторов:

– *условный оператор*

```
if (условие) { ветвь_1 } else { ветвь_2 };
```

– *оператор множественного выбора:*

```
switch (переменная)  {
case значение_1: оператор_1;  break;
case значение_2: оператор_2; break;

[default: оператор_N;]
}
```

– *операторы цикла:*

1) `for (i=0; i<9; i++) { тело цикла };`

2) `for (i in obj) {str = obj[i]}`. Здесь переменная i используется для перебора всех свойств (элементов) объекта (контейнера) obj;

3) `while (условие) { тело цикла };` – цикл с предусловием;

4) `do { тело цикла } while (условие);` – цикл с постусловием;

– *операторы управления выполнением цикла* **break** (досрочный выход из цикла) и **continue** (переход на следующую итерацию цикла, до завершения выполнения всех последующих операторов тела цикла);

– *оператор объявления переменной* **var**: **var** ***a*** =“**str**”**;**. Тип переменной не указывается, а определяется по типу присвоенного ей значения. В данном случае переменная ***a*** строкового типа.

Массивы в JavaScript. *Массивы в JavaScript* определяются при помощи специального литерала массива, представляющего собой заключенный в квадратные скобки список элементов, разделенных запятыми:

```
var array1=[1, 2, , 4].
```

В этом определении массива пропущен элемент с номером **2** (нумерация начинается с нуля), а потому, например, при обращении к этому элементу так: ***a=array1***[**2**]; переменная ***a*** будет пустой.

Функции в JavaScript. JavaScript позволяет создавать *функции* при помощи оператора **function**. Синтаксис описания функции следующий:

```
function <имя функции> (<список формальных аргументов>)
{   тело функции  }
```

Функция может возвращать значение с помощью оператора **return**. Тип возвращаемого значения в объявлении функции не указывается.

Переменные, объявленные вне функции, являются *глобальными* и доступны из любого места программы, в том числе и из функций. Все переменные, определенные внутри функции являются *локальными* и не видны из основной программы.

В JavaScript существуют также следующие *встроенные функции*:

– **escape**(*строка*) – заменяет недопустимые в URL символы на их шестнадцатеричные коды;

– **eval**(*строка*) – вычисляет выражение, находящееся в строке, как если бы оно было написано в коде программы: **eval**(“**2+2**”) вернет **4**;

– **parseFloat**(*строка*) – преобразует строку в число с плавающей точкой. Если строка не может быть преобразована, то возвращает **NaN**.

Объекты JavaScript. *Объект* – сложный тип данных, который включает в себя множество *переменных* (*свойств*) и *функций* (*методов*) для управления этими переменными. Свойства содержат данные, методы их обрабатывают.

В свою очередь каждый объект относится к какому-либо *классу* объектов. Несколько объектов могут относиться к одному и тому же классу. Таким образом, *класс* – это тип объекта, а *объект* – конкретный экземпляр класса, с которым можно работать.

Для того чтобы создать объект с именем ***obj***, принадлежащий некоторому классу ***someClass***, применяется оператор **new**:

var *obj* = **new *someClass***();

Все объекты JavaScript подразделяются на *встроенные*, *пользовательские* (созданные пользователем) и *внешние* (предоставленные другими программами). К последним относятся все объекты, образующие объектную модель документа (DOM).

Пример. Создадим пользовательский класс **circle**. Для этого напишем соответствующий конструктор и метод для этого класса.

```
function circle(x, y, radius) {
     this.x = x || 50;
     this.y = y || 30;
     this.radius = radius || 20;
     this.findSquare = findSquare (this.radius);
}
function findSquare(radius) {
     return 3.14*Math.pow(radius, 2);
}
```

Ключевое слово ***this*** заменяет в конструкторе имя экземпляра объекта. Для того чтобы значения свойств объекта не оставались пустыми в том случае, когда они не указаны при создании нового объекта класса, при помощи логического оператора **ИЛИ** (| |) им эти значения присваиваются. Благодаря тому, что в JavaScript можно свободно присваивать функции переменным и затем через имена этих переменных с параметрами обращаться к той же функции, в классе **circle** функция *findSquare* становится его методом.

Теперь создадим объект класса **circle** и найдем площадь круга:

circle1 = new Circle(10,10,31);

document.write(“Площадь круга: ” + circle1.findSquare);

В функции *findSquare* используется объект с названием **Math**. Это один из встроенных классов JavaScript.

Встроенные классы. Класс Math. Предоставляет набор стандартных математических функций. Методы класса: ceil(*число*) – возвращает ближайшее большее или равное аргументу целое число; floor(*число*) – ближайшее меньшее или равное аргументу целое число; abs(*число*) – абсолютное значение аргумента; sin(*число*) – синус аргумента; cos(*число*) – косинус аргумента; tan(*число*) – тангенс аргумента; exp(*число*) – экспоненту числа; log(*число*) – натуральный логарифм числа; pow(*x*, *y*) – значение *x* возведенное в степень *y*; sqrt(*число*) – квадратный корень от аргумента; max(*список аргументов*) – максимальный из аргументов; min(*список аргументов*) – минимальный из аргументов; random() –

псевдослучайное число в диапазоне (0, 1); round(*число*) – значение аргумента, округленное до ближайшего целого.

Класс Array. В JavaScript массивы можно создавать как объекты класса **Array**. Если конструктору передать только один числовой параметр, то будет создан массив, содержащий соответствующее число неопределенных элементов. Если передается несколько параметров или один нечисловой, то эти параметры будут использованы как элементы массива. Если параметров нет, то создается пустой массив. Для получения размера массива можно использовать свойство ***length***, для доступа к элементам массива – *индексы*.

Методы класса **Array**: **join**(*разделитель*) – возвращает строку, полученную в результате объединения всех элементов массива, разделенных <разделителем>; **pop**() – удаляет из массива и возвращает последний элемент массива. Если массив пустой, то возвращает значение **undefined**; **push**(*список элементов*) – добавляет в массив элементы и возвращает новую длину массива (список может содержать другие массивы); **reverse**() – возвращает массив, порядок элементов в котором изменен на противоположный; **shift**() – удаляет и возвращает первый элемент массива (если массив пустой, то возвращает значение **undefined**).

Пример работы с массивами.

A=new Array (1, 2, 3);

B=new Array (-1, -2, -3);

A.push(B);

document.write (“A= “+ A.join(';'));

В результате в массиве **A** будет четыре элемента A [0] =1, A [1] =2, A [2] =3, A [3] = [-1, -2, -3]. Элемент A [3] также является массивом, состоящим из элементов A[3][0]=-1, A[3][1]=-2, A[3][2]=-3. Таким образом, можно создавать многомерные массивы.

Класс Date. Служит для хранения даты и времени. Если не указан ни один параметр, то создаваемый объект инициализируется текущим временем и датой.

Пример. Реализация часов.

```
<HTML> <HEAD> <TITLE>Часы</TITLE> </HEAD>
<SCRIPT LANGUAGE=javascript FOR=window EVENT=onload>
//выполняется при загрузке окна
window.tmr = setInterval('tick()', 500);
//создаем таймер, записав ссылку на него во вновь созданное поле
tmr;
//каждые 500мс будет вызываться функция tick()
</SCRIPT>
<SCRIPT LANGUAGE=javascript FOR=window EVENT=onunload>
//при выгрузке окна удалим таймер, если он был раннее создан
успешно
     if (window.tmr!=null)   clearInterval(window.tmr);
</SCRIPT>
<BODY >
<P> Текущее время </P>   <P ID=“clock”>  </P>
<SCRIPT>
```

```
function tick()
{     var time = new Date();
      //создаем переменную time и записываем в нее текущее время
      h = time.getHours();  if (h<10) h="0"+h;
      m = time.getMinutes();  if (m<10) m="0"+m;
      s = time.getSeconds();  if (s<10) s="0"+s;
      clock.innerHTML = " " + h + ":" + m + ":" + s;
           //вписывается внутрь тега <P ID="clock">  </P>
      window.status = clock.innerHTML;
           //в статусную строку записываем текущее время
}
</SCRIPT>  </BODY>  </HTML>
```

Задание

Вычислить значения функции в соответствии с вариантом задания в диапазоне [X_{min}, X_{max}] с шагом ***dx***, задаваемыми на странице.

Вариант 1

$$F = \begin{cases} ax^2 + b & при\ x < 0\ и\ b \neq 0 \\ \dfrac{x-a}{x-c} & при\ x > 0\ и\ b = 0 \\ \dfrac{x}{c} & в\ остальных\ случаях \end{cases}$$

Вариант 2

$$F = \begin{cases} \dfrac{1}{ax} - b & при\ x+5 < 0\ и\ c = 0 \\ \dfrac{x-a}{x} & при\ x+5 > 0\ и\ c \neq 0 \\ \dfrac{10x}{c-4} & в\ остальных\ случаях \end{cases}$$

Вариант 3

$$F = \begin{cases} ax^2 + bx + c & при\ a < 0\ и\ c \neq 0 \\ \dfrac{-a}{x-c} & при\ a > 0\ и\ c = 0 \\ a(x+c) & в\ остальных\ случаях \end{cases}$$

Вариант 4

$$F = \begin{cases} -ax\ -c & при\ c < 0\ и\ x \neq 0 \\ \dfrac{x-a}{-c} & при\ c > 0\ и\ x = 0 \\ \dfrac{bx}{c-a} & в\ остальных\ случаях \end{cases}$$

Вариант 5

$$F = \begin{cases} a - \dfrac{x}{10+x} & \text{при } x < 0 \text{ и } b \neq 0 \\ \dfrac{x-a}{x-c} & \text{при } x > 0 \text{ и } b = 0 \\ 3x + \dfrac{2}{c} & \text{в остальных случаях} \end{cases}$$

Вариант 6

$$F = \begin{cases} ax^2 + b^2 x & \text{при } c < 0 \text{ и } b \neq 0 \\ \dfrac{x+a}{x+c} & \text{при } c > 0 \text{ и } b = 0 \\ \dfrac{x}{c} & \text{в остальных случаях} \end{cases}$$

Вариант 7

$$F = \begin{cases} -ax^2 - b & \text{при } x < 5 \text{ и } c \neq 0 \\ \dfrac{x-a}{x} & \text{при } x > 5 \text{ и } c = 0 \\ \dfrac{-x}{c} & \text{в остальных случаях} \end{cases}$$

Вариант 8

$$F = \begin{cases} -ax^2 & \text{при } c < 0 \text{ и } a \neq 0 \\ \dfrac{a-x}{cx} & \text{при } c > 0 \text{ и } a = 0 \\ \dfrac{x}{c} & \text{в остальных случаях} \end{cases}$$

Вариант 9

$$F = \begin{cases} ax^2 + b^2 x & \text{при } a < 0 \text{ и } x \neq 0 \\ x - \dfrac{a}{x-c} & \text{при } a > 0 \text{ и } x = 0 \\ 1 + \dfrac{x}{c} & \text{в остальных случаях} \end{cases}$$

Вариант 10

$$F = \begin{cases} ax^2 - bx + c & при\ x < 3\ и\ b \neq 0 \\ \dfrac{x-a}{x-c} & при\ x > 3\ и\ b = 0 \\ \dfrac{x}{c} & в\ остальных\ случаях \end{cases}$$

Вариант 11

$$F = \begin{cases} ax^2 + \dfrac{b}{c} & при\ x < 1\ и\ c \neq 0 \\ \dfrac{x-a}{(x-c)^2} & при\ x > 1{,}5\ и\ c = 0 \\ \dfrac{x^2}{c^2} & в\ остальных\ случаях \end{cases}$$

Вариант 12

$$F = \begin{cases} ax^3 + b^2 + c & при\ x < 0{,}6\ и\ b + c \neq 0 \\ \dfrac{x-a}{x-c} & при\ x > 0{,}6\ и\ b + c = 0 \\ \dfrac{x}{c} + \dfrac{x}{a} & в\ остальных\ случаях \end{cases}$$

Вариант 13

$$F = \begin{cases} ax^2 + b & при\ x - 1 < 0\ и\ b - c \neq 0 \\ \dfrac{x-a}{x} & при\ x - 1 > 0\ и\ b + x = 0 \\ \dfrac{x}{c} & в\ остальных\ случаях \end{cases}$$

Вариант 14

$$F = \begin{cases} -ax^3 - b & при\ x + c < 0\ и\ a \neq 0 \\ \dfrac{x-a}{x-c} & при\ x + c > 0\ и\ a = 0 \\ \dfrac{x}{c} + \dfrac{c}{x} & в\ остальных\ случаях \end{cases}$$

Вариант 15

$$F = \begin{cases} -ax^2 + b & при\ x < 0\ и\ b \neq 0 \\ \dfrac{x}{x-c} + 5{,}5 & при\ x > 0\ и\ b = 0 \\ \dfrac{x}{-c} & в\ остальных\ случаях \end{cases}$$

Вариант 16

$$F = \begin{cases} a(x+c)^2 - b & при\ x = 0\ и\ b \neq 0 \\ \dfrac{x-a}{-c} & при\ x = 0\ и\ b = 0 \\ a + \dfrac{x}{c} & в\ остальных\ случаях \end{cases}$$

Вариант 17

$$F = \begin{cases} ax^2 - cx + b & при\ x + 10 < 0\ и\ b \neq 0 \\ \dfrac{x-a}{x-c} & при\ x + 10 > 0\ и\ b = 0 \\ \dfrac{-x}{a-c} & в\ остальных\ случаях \end{cases}$$

Вариант 18

$$F = \begin{cases} ax^3 + bx^2 & при\ x < 0\ и\ b \neq 0 \\ \dfrac{x-a}{x-c} & при\ x > 0\ и\ b = 0 \\ \dfrac{x+5}{c(x-10)} & в\ остальных\ случаях \end{cases}$$

Вариант 19

$$F = \begin{cases} a(x+7)^2 - b & при\ x < 5\ и\ b \neq 0 \\ \dfrac{x-cd}{ax} & при\ x > 5\ и\ b = 0 \\ \dfrac{x}{c} & в\ остальных\ случаях \end{cases}$$

Вариант 20

$$F = \begin{cases} -\dfrac{2x-c}{cx-a} & при\ x<0\ и\ b\neq 0 \\ \dfrac{x-a}{x-c} & при\ x>0\ и\ b=0 \\ -\dfrac{x}{c}+\dfrac{-c}{2x} & в\ остальных\ случаях \end{cases}$$

Контрольные вопросы

1. Язык JavaScript – это?
2. Расположение скрипта в HTML-документе.
3. Операторы JavaScript.
4. Функции в JavaScript.
5. Событие – это?
6. Виды событий.
7. Встроенные классы.