Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерного проектирования

Кафедра проектирования информационно-компьютерных систем

В. Ф. Алексеев, Т. В. Русак, Г. А. Пискун

ОСНОВЫ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

Рекомендовано учебно-методическим объединением по образованию в области приборостроения в качестве пособия для обучающихся по специальности 1-38 80 04 «Технология приборостроения»

Минск БГУИР 2017

УДК 004(076.5)
ББК 32.973.2я73
А47

Р е ц е н з е н т ы:

кафедра экономической информатики учреждения образования «Белорусский государственный экономический университет» (протокол №11 от 25.05.2016);

профессор учреждения образования «Белорусская государственная академия связи», доктор технических наук, профессор В. И. Курмашев

Алексеев, В. Ф.

А47 Основы информационных технологий. Лабораторный практикум : пособие / В. Ф. Алексеев, Т. В. Русак, Г. А. Пискун. – Минск : БГУИР, 2017. – 104 с. : ил.

ISBN 978-985-543-264-8.

Рассмотрены основы разработки web-приложений с использованием языка гипертекстовой разметки HTML, каскадных таблиц стилей CSS, языка программирования PHP и СУБД MySQL.

Предназначено для студентов высших учебных заведений, может быть использовано аспирантами, инженерами, экономистами, занимающимися вопросами анализа данных.

УДК 004(076.5)
ББК 32.973.2я73

ISBN 978-985-543-264-8

СОДЕРЖАНИЕ

ЛАБОРАТОРНАЯ РАБОТА №1
ОСНОВЫ HTML

Цель работы: изучить основы языка разметки гипертекста HTML, научиться создавать HTML-страницы.

Ход выполнения лабораторной работы должен быть отражен в отчете. Отчет должен содержать титульный лист, цель работы, номера заданий, коды программ, скриншот с результатом выполнения программы, выводы.

1.1 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

1.1.1 Основные термины и понятия

Hypertext Markup Language (HTML) – язык разметки гипертекстов одним из языков, используемых для доставки информации через Интернет.

Будучи универсальным средством, HTML вместе с протоколом передачи гипертекста (Hypertext Transfer Protocol – HTTP) позволяет решать задачи перевода документов между разными компьютерами, платформами и сетями, обеспечивая широкую доступность документов.

Основной конструкцией языка HTML является элемент. **Элемент** – это контейнер, содержащий данные и позволяющий отформатировать их определенным образом. Любая web-страница представляет собой ряд элементов. Одна из идей гипертекста – возможность вложения различных элементов.

Начальный или конечный маркер элемента называется **тегом.** Теги определяют границы действия элементов и отделяют элементы друг от друга.

Тег HTML состоит из следующих друг за другом в определенном порядке элементов:

- левой угловой скобки < (такой же, как символ «меньше чем»);
- необязательного слэша /, который означает, что тег является конечным тегом, закрывающим некоторую структуру;
- имени тега, например TITLE или PRE;
- необязательных атрибутов (тег может быть без атрибутов или сопровождаться одним или несколькими атрибутами);
- правой угловой скобки > (такой же, как символ «больше чем»).

Примеры:

```
<H1 ALIGN=LEFT>Заголовок</H1>
<P>Тест</P>
```

Атрибут – параметр или свойства элемента. Это, по сути, переменная, которой может придаваться определенное значение в кавычках. Атрибуты расположены внутри начального тега и отделяются они пробелами.

Спецификация атрибута состоит из расположенных в следующем порядке элементов символов:

- имя атрибута, например WIDTH;
- знак равенства (=);
- значение атрибута, которое задается строкой символов, например "80".

Всегда полезно заключить **значение атрибута** в кавычки, используя либо одинарные ('80'), либо двойные кавычки ("80"). Строка в кавычках не должна содержать такие же кавычки внутри себя.

Можно также опустить кавычки для значений атрибутов, которые состоят только из следующих символов (обратитесь к технической концепции имени):

- букв английского алфавита (A – Z, a – z);
- цифр (0 – 9);
- промежутков времени;
- дефисов (–).

Таким образом, WIDTH=80 и ALIGN=CENTER – разрешенные сокращения для WIDTH="80" и ALIGN="CENTER". Ссылка на URL, например HREF=foo.htm, допустима, однако, когда URL используется с атрибутами, он должен быть заковычен, например HREF="http://www.hut.fi/".

Если значение атрибута такое же, как его имя, может быть использован минимальный синтаксис атрибута. То есть <UL COMPACT="COMPACT"> можно сократить до <UL COMPACT>.

Гиперссылка (hyperlink) – фрагмент гипертекста, который указывает на другой файл или объект.

HTML-файл, или HTML-страница, – документ, созданный в виде гипертекста на основе языка HTML.

Скрипт, или **сценарий** (script), – программа, лежащая в HTML-странице, расширяя ее возможность при помощи средств мультимедиа.

1.1.2 Структура HTML-документа

Каждый HTML-документ, отвечающий спецификации HTML какой-либо версии, обязан начинаться со строки декларации версии HTML !DOCTYPE.

После объявления версии и типа документа необходимо обозначить его начало и конец. Это делается с помощью тега-контейнера <HTML>. Необходимо отметить, что любой HTML-документ открывается тегом <HTML> и им же закрывается. Затем между тегами <HTML> и </HTML> следует разместить заголовок и тело документа. Пример самого короткого HTML-документа приведен ниже.

Пример 1.1

```
<! DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<HTML>
    <HEAD>
        <TITLE>Заголовок документа</TITLE>
    </HEAD>
```

```
<BODY>
        Текст документа
</BODY>
</HTML>
```

Из примера видно, что документ состоит из двух основных блоков – «заголовка» и «тела документа». Заголовок определяется с помощью элемента HEAD, а тело – элементом BODY.

1.1.3 Заголовок HTML-документа

Создается с помощью элемента **HEAD**, между тегами которого размещаются элементы, содержащие техническую информацию о документе. Заголовок обычно располагается до тела документа.

Элементы, относящиеся к заголовку документа:

– **HEAD** определяет начало и конец заголовка документа;

– **TITLE** определяет имя всего документа, которое отображается в заголовке окна браузера;

– **BASE** определяет базовый адрес, от которого отсчитываются относительные линки внутри документа;

– **STYLE** используется для вставки в документ таблицы стилей CSS;

– **LINK** описывает взаимосвязь документа с другими объектами;

– **META** используется для вставки метаданных.

Элемент **TITLE** определяет имя всего документа. Имя, как правило, отображается в заголовке окна браузера. Данный элемент *обязателен* для любого HTML-документа и может быть указан не более одного раза.

Пример 1.2

```
...
<HEAD>
   <TITLE>Руководство по эксплуатации</TITLE>
</HEAD>
...
```

Задание 1.1. *Используя Блокнот создать документ html. Заголовок документа должен содержать следующую информацию:* ***«Прайс-лист компании «Название компании»*** *(название компании указывается любое). Файл сохранить в каталоге D:\Номер_группы\ФИО.*

1.1.4 Тело HTML-документа

Тело документа создается с помощью элемента **BODY**. Именно в теле документа содержится все то, что будет отображено на странице:

– текст, блоки текста;

– гиперссылки;

– списки;

– таблицы;

– объекты, картинки;

– заполняемые формы.

Элемент BODY указывает начало и конец тела HTML-документа. Между начальным и конечным тегами содержится текст документа, изображения и таблицы. Одним словом, все HTML-элементы, отвечают за отображение документа, управление им и гипертекстовые ссылки. Элемент BODY должен встречаться в документе не более одного раза.

Атрибуты элемента BODY

TOPMARGIN – определяет ширину (в пикселах) верхнего и нижнего полей документа.

LEFTMARGIN – определяет ширину (в пикселах) левого и правого полей документа.

BACKGROUND – определяет изображение для «заливки» фона. Значение задается в виде полного URL или имени файла с картинкой в формате GIF или JPG.

BGCOLOR – определяет цвет фона документа.

TEXT – определяет цвет текста в документе.

LINK – определяет цвет гиперссылок в документе.

ALINK – определяет цвет подсветки гиперссылок в момент нажатия.

VLINK – определяет цвет гиперссылок на документы, которые вы уже просмотрели.

Значения атрибутов **BGCOLOR**, **TEXT**, **LINK**, **ALINK** и **VLINK** задаются либо RGB-значением в шестнадцатеричной системе, либо одним из 16 базовых цветов (рисунок 1.1).

	Black		Navy
	Gray		Blue
	Silver		Aqua
	White		Green
	Red		Lime
	Fuchsia		Teal
	Maroon		Yellow
	Purple		Olive

Рисунок 1.1 – Таблица 16 базовых цветов

Пример 1.3

```
<HTML>
<BODY BACKGROUND="images/bricks.jpg" BGCOLOR="#202020"
TEXT="#FFFFFF" LINK="#FF0000" VLINK="#505050" TOPMARGIN="30"
LEFTMARGIN="40">
```

...
Текст документа.
...
</BODY>
</HTML>

Задание 1.2. *В созданном документе необходимо определить цвет фона, цвет текста, цвет уже просмотренных гиперссылок и задать левое и правое поля документа, равные по 20 пикселов каждое.*

1.1.5 Текстовые блоки

В этом пункте описаны элементы, разбивающие текст документа на блоки тем или иным способом. Типичными примерами текстовых блоков являются параграфы, абзацы и главы. Для отделения одной части текста от другой также используются разделительные горизонтальные линии и символы возврата каретки.

Элементы H1,H2,...H6 используются для создания заголовков текста. Существует шесть уровней заголовков, различающихся величиной шрифта. С их помощью можно разбивать текст на смысловые уровни – разделы и подразделы.

Атрибут элементов H1,H2,...H6

ALIGN – определяет способ выравнивания заголовка по горизонтали. Возможные значения: left, right, center. По умолчанию имеет значение left.

Пример 1.4

```
<H1 ALIGN="center">Самый большой заголовок посередине</H1>
<H2>Заголовок поменьше</H2>
...
<H6>Самый маленький заголовок</H6>
```

Элемент P используется для разметки параграфов.

Атрибут элемента P

ALIGN – определяет способ горизонтального выравнивания параграфа. Возможные значения: left, center, right. По умолчанию имеет значение left.

Пример 1.5

```
<P ALIGN="center">Это центрированный параграф.<BR>
Текст располагается в центре окна браузера</P>
<P ALIGN="right">А это параграф, выровненный по правому краю.</P>
```

Элемент DIV используется для логического выделения блока HTML-документа. Элемент группировки, как и элемент **SPAN**, в современном сайтостроении используется как удобный контейнер для объектов страницы, которым легко динамически манипулировать – перемещать, включать/выключать, создавать слои, регулировать отступы и т. п.

В браузеронезависимой верстке элемент DIV обычно используется для выравнивания блока html-кода в окне браузера.

Находящиеся между начальным и конечным тегами текст или HTML-элементы по умолчанию оформляются как отдельный параграф.

Атрибут элемента DIV

ALIGN – определяет выравнивание содержимого элемента DIV. Атрибут может принимать значения left, right, center.

Пример 1.6

```
...Текст документа...
<DIV ALIGN="center">
...Текст, таблицы, изображения. Выравнивание по центру.
</DIV>
...Текст документа...
```

Элемент BR осуществляет перевод строки, т. е. практически аналогичен нажатию Enter в текстовом редакторе. После того как в браузерах появилась возможность обтекания изображения текстом (атрибут ALIGN элемента IMG), понадобился дополнительный атрибут CLEAR. Элемент не имеет конечного тега.

Атрибут элемента BR

CLEAR – указывает на необходимость завершения обтекания изображения текстом. Может принимать следующие значения: **all** – завершить обтекание изображения текстом; **left** – завершить обтекание текстом изображения, выровненного по левому краю; **right** – завершить обтекание текстом изображения, выровненного по правому краю.

Пример 1.7

```
Первое предложение<BR>Второе предложение на следующей строке.
```

Элемент HR вставляет в текст горизонтальную разделительную линию.

Атрибуты элемента HR

WIDTH – определяет длину линии в пикселах или процентах от ширины окна браузера.

SIZE – определяет толщину линии в пикселах.

ALIGN – определяет выравнивание горизонтальной линии. Атрибут может принимать следующие значения: *left* – выравнивание по левому краю документа; *right* – выравнивание по правому краю документа; *center* – выравнивание по центру документа (используется по умолчанию).

NOSHADE – определяет способ закраски линии как сплошной. Атрибут является флагом и не требует указания значения. Без данного атрибута линия отображается объемной.

COLOR – определяет цвет линии. Задается либо RGB-значением в шестнадцатеричной системе, либо одним из 16 базовых цветов.

Пример 1.8

Текст, разделенный <HR NOSHADE WIDTH="50%"> сплошной горизонтальной линией.

Задание 1.3. *Добавить в созданный документ заголовок первого уровня:* ***«Прайс-лист компании «Название компании».*** *Заголовок выровнять по центру.*

Задание 1.4. *Отделить заголовок сверху и снизу горизонтальными разделительными линиями. Длина линий составляет 50 % страницы. Одна линия должна быть выровнена по левому краю и быть объемной, а вторая – по правому и полностью закрашена (не объемная). Для каждой линии задать свой цвет.*

1.1.6 Таблицы

Элемент TABLE используется для создания таблицы, обязательно должен иметь начальный и конечный теги. По умолчанию таблица печатается без рамки, а разметка осуществляется автоматически в зависимости от объема содержащейся в ней информации. Ячейки внутри таблицы создаются с помощью элементов **TR**, **TD**, **TH** и **CAPTION**.

Атрибуты элемента TABLE

ALIGN – определяет способ горизонтального выравнивания таблицы. Возможные значения: left, center, right. Значение по умолчанию – left.

VALIGN – должен определять способ вертикального выравнивания таблицы. Возможные значения: top, bottom, middle.

BORDER – определяет ширину внешней рамки таблицы (в пикселах). При **BORDER="0"** или при отсутствии этого атрибута рамка отображаться не будет.

CELLPADDING – определяет расстояние (в пикселах) между рамкой каждой ячейки таблицы и содержащимся в ней материалом.

CELLSPACING – определяет расстояние (в пикселах) между границами соседних ячеек.

WIDTH – определяет ширину таблицы. Ширина задается либо в пикселах, либо в процентном отношении к ширине окна браузера. По умолчанию этот атрибут определяется автоматически в зависимости от объема содержащегося в таблице материала.

HEIGHT – определяет высоту таблицы. Высота задается либо в пикселах, либо в процентном отношении к высоте окна браузера. По умолчанию этот атрибут определяется автоматически в зависимости от объема содержащегося в таблице материала.

BGCOLOR – определяет цвет фона ячеек таблицы. Задается либо RGB-значением в шестнадцатеричной системе, либо одним из 16 базовых цветов.

BACKGROUND – позволяет заполнить фон таблицы рисунком. В качестве значения необходимо указать URL рисунка.

Элемент CAPTION задает заголовок таблицы. Содержание заголовка должно состоять только из текста. Использование блочных элементов в этом случае недопустимо.

Атрибуты элемента CAPTION

ALIGN – определяет способ вертикального выравнивания заголовка таблицы. Возможные значения: top – помещает заголовок над таблицей (значение по умолчанию); bottom – помещает заголовок под таблицей.

Пример 1.9

```
<TABLE BORDER="1">
   <CAPTION ALIGN="bottom">Заголовок таблицы</CAPTION>
   <TR>
      <TD>Ячейка таблицы</TD>
   </TR>
</TABLE>
```

Элемент TR создает новый ряд (строку) ячеек таблицы. Ячейки в ряду создаются с помощью элементов **TD** и **TH**

Атрибуты элемента TR

ALIGN – определяет способ горизонтального выравнивания содержимого всех ячеек данного ряда. Возможные значения: left, center, right.

VALIGN – определяет способ вертикального выравнивания содержимого всех ячеек данного ряда. Возможные значения: top, bottom, middle.

BGCOLOR – определяет цвет фона для всех ячеек данного ряда. Задается либо RGB-значением в шестнадцатеричной системе, либо одним из 16 базовых цветов.

Элемент TD создает ячейку с данными в текущей строке. **Элемент TH** также создает ячейку, но определяет ее как ячейку-заголовок.

Такое разграничение позволяет браузерам оформлять содержимое ячейки-заголовка и ячеек с данными разными шрифтами. В качестве содержимого ячейки можно использовать другие таблицы.

Атрибуты элементов TD и TH

ALIGN – определяет способ горизонтального выравнивания содержимого ячейки. Возможные значения: left, center, right. По умолчанию способ выравнивания определяется значением атрибута ALIGN элемента TR. Если же и он не задан, то для TD выполняется выравнивание по левому краю, а для TH – центрирование.

VALIGN – определяет способ вертикального выравнивания содержимого ячейки. Возможные значения: top, bottom, middle. По умолчанию происходит выравнивание по центру (VALIGN="middle"), если значение этого атрибута не было задано ранее в элементе TR.

WIDTH – определяет ширину ячейки. Ширина задается в пикселах или в процентном отношении к ширине таблицы.

HEIGHT – определяет высоту ячейки. Высота задается в пикселах или в процентном отношении к высоте таблицы.

COLSPAN – определяет количество столбцов, на которые простирается данная ячейка. По умолчанию имеет значение 1.

ROWSPAN – определяет количество рядов, на которые простирается данная ячейка. По умолчанию имеет значение 1.

NOWRAP – блокирует автоматический перенос слов в пределах текущей ячейки.

BGCOLOR – определяет цвет фона ячейки. Задается либо RGB-значением в шестнадцатеричной системе, либо одним из 16 базовых цветов.

BACKGROUND – заполняет ячейку фоновым рисунком. Необходимо указать URL рисунка.

Пример 1.10

```
<TABLE BORDER>
 <TR>
  <TH ROWSPAN=2>HDD</TH>
  <TD>WD Caviar 3.1Gb</TD><TD ALIGN="right">85$</TD>
 </TR>
 <TR>
  <TD>Quantum FB ST 6.4Gb</TD><TD ALIGN="right">110$</TD>
 </TR>
</TABLE>
```

Результат:

HDD	WD Caviar 3.1Gb	85$
	Quantum FB ST 6.4Gb	110$

Пример 1.11

```
<TABLE BORDER>
  <TR>
    <TH COLSPAN=2>Video</TH>
  </TR>
  <TR>
    <TD>Matrox G400</TD><TD ALIGN="right">115$</TD>
  </TR>
  <TR>
    <TD>Voodoo III</TD><TD ALIGN="right">129.95$</TD>
  </TR>
</TABLE>
```

Результат:

Video	
Matrox G400	115$
Voodoo III	129.95$

Задание 1.5:

1.5.1 Добавить в созданный документ таблицу, содержащую информацию о ценах на продукцию, указанную на рисунке 1.2.

1.5.2 Ширина таблицы должна быть равна 80 % от общей ширины окна браузера.

1.5.3 В заголовке таблицы, размещенном перед таблицей, указать название компании и дату формирования прайс-листа.

1.5.4 Заполнить таблицу произвольными данными (в таблице должно быть не менее 3 категорий товаров, каждая категория должна содержать не менее пяти наименований товара).

1.5.5 Название категорий и название наименований товара должно быть выровнено по левому краю и отцентрировано вертикально. Значение цены товара должно быть расположено в центре ячейки по правому краю. Значение скидки должно располагаться в центре ячейки таблицы.

1.5.6 Для первых двух строк таблицы задать цвет заливки.

1.5.7 Отобразить рамку таблицы.

Прайс-лист			
Наименование товара		Цена	Скидка
Категория товара 1	Наименование 1	34	10 %
	Наименование 2	34	10 %
	Наименование 3	34	10 %
Категория товара 2	Наименование 4	34	10 %
	Наименование 5	34	10 %
	Наименование 6	34	10 %

Рисунок 1.2 – Пример таблицы, содержащей информацию о продукции

1.1.7 Форматирование текста

В этом пункте описаны элементы для оформления и смыслового выделения текста – подчеркивания, изменения шрифта, выделения курсивом, цитирования и т. д.

Элемент BASEFONT не имеет конечного тега. Определяет основной шрифт, которым должен отображаться текст документа. Впоследствии можно легко изменить шрифт в любой части документа, используя элемент **FONT.** Действие элемента **BASEFONT** не распространяется на текст, заключенный в ячейки таблиц.

Атрибуты элемента BASEFONT

SIZE – обязательный атрибут. Определяет базовый размер шрифта. Возможные значения: целые числа от 1 до 7 включительно.

FACE – определяет используемый шрифт (гарнитуру).

Пример 1.12

```
<BODY>
<BASEFONT SIZE="3">
```

...
Текст документа шрифтом 3 размера
...
<FONT SIZE="+1">
Слегка увеличиваем шрифт
</FONT>
...
Продолжаем шрифтом 3 размера
...
</BODY>

Элемент FONT позволяет изменять цвет, размер и тип шрифта текста, находящегося между начальным и конечным тегами. Вне тегов <FONT> и </FONT> используется шрифт, указанный в элементе **BASEFONT**.

Атрибуты элемента FONT

SIZE – определяет размер шрифта. Возможные значения: целое число от 1 до 7; относительный размер с указанием знака вычисляется путем сложения с базовым размером, определенным с помощью атрибута SIZE элемента **BASEFONT**.

COLOR – определяет цвет текста. Задается либо RGB-значением в шестнадцатеричной системе, либо одним из 16 базовых цветов.

FACE – определяет используемый шрифт.

Пример 1.13

<FONT SIZE="+2" COLOR="#AA0000">Увеличенный красный шрифт</FONT>

<FONT SIZE="3" FACE="Courier New" COLOR="Magenta">Моноширинный фиолетовый текст 3 размера</FONT>

Элемент I – текст, заключенный между начальным и конечным тегами, будет выделен курсивом.

Пример 1.14

Текст с <I>курсивом</I>

Элемент B – текст, заключенный между начальным и конечным тегами, будет выделен жирным шрифтом.

Пример 1.15

Текст с <B>выделенным</B> словом

Элемент U отображает помещенный между начальным и конечным тегами текст как подчеркнутый.

Пример 1.16

<U> Подчеркнутый текст </U>

Задание 1.6:

1.6.1 Отформатировать текст таблицы, выделив название столбцов таблицы полужирным, название категорий – курсивом.

1.6.2 Товары с наибольшей скидкой выделить одним цветом шрифта, а товары с наименьшей ценой – другим цветом шрифта.

1.6.3 Заголовок таблицы сделать подчеркнутым.

1.1.8 Списки в HTML-документе

Списки в HTML бывают двух видов: упорядоченные (пронумерованные) и неупорядоченные (непронумерованные). Отличаются они лишь способом оформления. Перед пунктами неупорядоченных списков обычно ставятся символы, например точки, ромбики и т. п., в то время как пунктам упорядоченных списков предшествуют их номера.

Элемент UL создает неупорядоченный список. Между начальным и конечным тегами должны присутствовать один или несколько элементов **LI**, обозначающих отдельные пункты списка.

Пример 1.17

```
<UL>
   <LI> Первый пункт списка </LI>
   <LI> Второй пункт списка </LI>
   <LI> Третий пункт списка </LI>
</UL>
```

Элемент OL создает упорядоченный список. Между начальным и конечным тегами должны присутствовать один или несколько элементов **LI**, обозначающих отдельные пункты списка.

Атрибуты элемента OL

START – определяет первое число, с которого начинается нумерация пунктов (только целые числа).

TYPE – определяет стиль нумерации пунктов. Может иметь значения:

– **«A»** – заглавные буквы A, B, C ...;

– **«a»** – строчные буквы a, b, c ...;

– **«I»** – прописные римские числа I, II, III ...;

– **«i»** – строчные римские числа i, ii, iii ...;

– **«1»** – арабские числа 1, 2, 3 ... .

По умолчанию установлено <UL TYPE="1">.

Пример 1.18

```
<OL TYPE="I" START="2">
   <LI> Пункт два </LI>
   <LI> Пункт три </LI>
```

```
   <LI> Пункт четыре </LI>
</OL>
```

Элемент LI создает пункт в списке. Располагается внутри элементов **OL** или **UL**.

Атрибуты элемента LI

VALUE – изменяет порядок нумерации элементов списка. Используется только если элемент **LI** находится внутри элемента **OL**. В качестве значения указывается порядковый номер элемента.

Пример 1.19

```
<OL TYPE="A" START="2">
   <LI> Пункт, озаглавленный буквой B. </LI>
   <LI VALUE="6"> Пункт, озаглавленный буквой F. </LI>
   <LI> Пункт, озаглавленный буквой G. </LI>
</OL>
```

Задание 1.7:

1.7.1 Создать вторую html-страницу, содержащую информацию о компании: название, адрес, контактные телефоны.

1.7.2 Добавить маркированный список, в котором перечислить направления деятельности компании (не менее 3).

1.7.3 Добавить на страницу нумерованный список подразделений компании (не менее 5).

1.1.9 Объекты в HTML-документе

Объекты – это графические и мультимедийные вставки в HTML-документ, такие как картинки, Flash-анимация, Java-апплеты, звуки, музыка, VRML.

Элементы:

– **IMG** используется для вставки в HTML изображений;

– **EMBED** используется для вставки в HTML различных объектов;

– **NOEMBED** используется, если браузер не поддерживает элемент EMBED;

– **APPLET** используется для вставки в HTML Java-апплетов;

– **PARAM** используется для передачи параметров Java-программе.

Элемент IMG используется для вставки изображений в HTML-документ. Элемент допускает вставку изображений в форматах JPEG и GIF. Элемент **IMG** не имеет конечного тега.

Атрибуты элемента IMG

SRC – обязательный атрибут; указывает адрес (URL) файла с изображением.

HEIGHT и **WIDTH** определяют ширину и высоту изображения соответственно. Если указанные значения не совпадают с реальным размером изображения, то оно масштабируется (порой с заметной потерей качества).

HSPACE и **VSPACE** определяют отступ картинки (в пикселах) по горизонтали и вертикали от других объектов документа.

ALIGN – обязательный атрибут; указывает способ выравнивания изображения в документе. Может принимать следующие значения:

– left – выравнивает изображение по левому краю документа; прилегающий текст обтекает изображение справа;

– right – выравнивает изображение по правому краю документа; прилегающий текст обтекает изображение слева;

– top и texttop – выравнивают верхнюю кромку изображения с верхней линией текущей текстовой строки;

– middle – выравнивает базовую линию текущей текстовой строки с центром изображения;

– absmiddle – выравнивает центр текущей текстовой строки с центром изображения;

– bottom и baseline – выравнивают нижнюю кромку изображения с базовой линией текущей текстовой строки;

– absbottom – выравнивает нижнюю кромку изображения с нижней кромкой текущей текстовой строки.

NAME – определяет имя изображения, уникальное для данного документа. Вы можете указать любое имя без пробелов с использованием латинских символов и цифр. Имя необходимо, если вы планируете осуществлять доступ к изображению, например из JavaScript-сценариев.

ALT – определяет текст, отображаемый браузером на месте изображения, если браузер не может найти файл с изображением или включен в текстовый режим. В качестве значения задается текст с описанием изображения.

BORDER – определяет ширину рамки вокруг изображения в пикселах. Рамка возникает, только если изображение является гипертекстовой ссылкой. В таких случаях значение BORDER обычно указывают равным нулю.

Пример 1.20

<IMG src="/img/picture.gif" WIDTH="45" HEIGHT="53" ALT="Рысь" HSPACE="10" ALIGN="left"> Этот текст обтекает картинку справа и находится от нее на расстоянии 10 пикселов.

Пример 1.21

Использование изображения в качестве гиперссылки:

<A HREF="price.html">

<IMG src="/img/button.jpg" WIDTH="70" HEIGHT="30" ALIGN="right" BORDER="0" ALT="Прайс-лист">

</A>

Для просмотра прайс-листа нажмите кнопку справа.

Задание 1.8. *Добавить в две созданные страницы графический файл, содержащий логотип компании. Если у пользователя отключено изображение в браузере, на месте логотипа должна выводиться подпись «Логотип компании». Изображение выровнять по правому краю. Информация о компании должна располагаться рядом с логотипом.*

1.1.10 Гиперссылки в HTML-документе

Ссылки на другие документы в HTML создаются либо с помощью элемента **A**, либо с помощью навигационных карт. Элемент **A** применяется, если ссылкой планируется сделать часть текста или целое изображение. Навигационные карты имеет смысл применять, если ссылкой будет часть изображения.

Элемент A – самый необходимый элемент, без которого Интернет просто немыслим; используется для создания и использования гипертекстовых ссылок. Элемент **A** не может быть вложенным в себе подобные.

Атрибуты элемента A

HREF – определяет находящийся между начальным и конечным тегами текст или изображение как гипертекстовую ссылку (URL) на документ (и/или область документа), указанный в значении данного атрибута. Возможные значения:

– http://... – создает ссылку на www-документ;

– ftp://... – создает ссылку на ftp-сайт или расположенный на нем файл;

– mailto:... – запускает почтовую программу-клиент с заполненным полем имени получателя.

Если тип соединения и адрес машины не указаны, в качестве отправной точки используется адрес текущего документа. Это позволяет использовать относительные ссылки.

Например, ссылка <A HREF="docs/title.html">Документация</A> будет указывать путь к файлу title.html в подкаталоге docs (относительно текущего).

NAME – помечает находящуюся между начальным и конечным тегами область документа как возможный объект для ссылки. В качестве значения нужно латиницей написать любое слово-указатель, уникальное для данного документа.

Например: <A NAME="part">Раздел1</A>. Теперь можно ссылаться на помеченную область простым указанием ее имени после имени документа. Например, линк <A HREF="document.html#part">Раздел1</A> отправит вас в раздел "part" файла document.html, а линк <A HREF="#bottom">В конец документа</A> – в раздел "bottom" текущего документа.

TARGET – определяет окно (фрейм), на которое указывает гипертекстовая ссылка. Этот атрибут используется только совместно с атрибутом HREF. В качестве значения необходимо задать либо имя одного из существующих фреймов, либо одно из следующих зарезервированных имен:

– _self – указывает, что определенный в атрибуте HREF документ должен отображаться в текущем фрейме;

– _parent – указывает, что документ должен отображаться во фрейме-родителе текущего фрейма;

– _top – указывает, что документ должен отображаться в окне-родителе всей текущей фреймовой структуры;

– _blank – указывает, что документ должен отображаться в новом окне.

Пример 1.22

```
<!-- Использование атрибута NAME: -->
<A NAME="history">История бодибилдинга</A>
...
<A NAME="now">Спорт глазами современника</A>
...
Вернуться к разделу<A HREF="#history">истории</A>
```

Пример 1.23

```
<!-- Использование атрибута HREF: -->
<A HREF="ftp://ftp.cdrom.com" TARGET="_blank">FTP-site</A>
<A HREF="http://opengl.rdc.ru">Русский проект по OpenGL</A>
...
```

Пример 1.24

```
<!-- Создадим ссылку для письма -->
<A HREF="mailto:green@igf.ru ">Отправить приглашение</A>.
```

Задание 1.9. *Добавить на странице, содержащей информацию о компании, ссылку на прайс-лист.*

Задание 1.10. *Добавить на странице, содержащей прайс-лист, ссылку на страницу с информацией о компании.*

1.2 ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

1 В HTML-странице создать таблицу, указанную на рисунке 1.3.

Номер группы	ФИО студента	
Название дисциплины		
	1. Лабораторная работа 1. HTML-формы. Обработка данных, введенных в форму: – практическое задание, – индивидуальное задание	
	2. Лабораторная работа 2. Основы PHP: – задание 1, – задание 2	
	…	

Рисунок 1.3 – Пример таблицы

2 Перечень лабораторных работ оформить нумерованным списком.

3 Перечень заданий в каждой лабораторной работе оформить маркированным списком.

4 Создать ссылки на соответствующие задания выполненных лабораторных работ.

5 Применить различные шрифты, цвет фона, цвет текста и другие параметры форматирования для оформления созданной страницы.

1.3 КОНТРОЛЬНЫЕ ВОПРОСЫ

1 HTML – это:

– язык гипертекстовой разметки;

– язык структурной разметки;

– язык редактирования;

– язык программирования;

– нет верных ответов.

2 Тег состоит из ОБЯЗАТЕЛЬНЫХ элементов, следующих друг за другом в определенной последовательности:

– левая угловая скобка (<);

– имя тега;

– слэш (/);

– правая угловая скобка (>);

– атрибут.

3 Гиперссылка – это:

– параметр или свойства элемента HTML, которые расположены внутри начального тега и отделяются пробелами;

– начальный и конечный маркеры элемента HTML, которые определяют границы действия элемента и отделяют их друг от друга;

– фрагмент гипертекста, который указывает на другой файл или объект;

– контейнер, содержащий данные и позволяющий отформатировать их определенным образом;

– нет верных ответов.

4 Атрибут элемента HTML – это:

– контейнер, содержащий данные и позволяющий отформатировать их определенным образом;

– начальный и конечный маркеры элемента HTML, которые определяют границы действия элемента и отделяют их друг от друга;

– параметр или свойства элемента HTML, которые расположены внутри начального тега и отделяются пробелами;

– фрагмент гипертекста, который указывает на другой файл или объект;

– нет верных ответов.

5 Все элементы HTML-разметки заканчиваются тегом конца:

– элементы вида <…/> не имеют тега конца;

– в HTML все элементы имеют тег конца элемента;

– существуют неполные элементы разметки, у которых нет тега конца;
– все ответы верны;
– нет верных ответов.

6 Какие из перечисленных элементов могут быть размещены в заголовке HTML-документа?

– <LINK HREF="…" REL="…">;
– <BODY BGCOLOR="…" TEXT="…">;
– <SCRIPT LANGUAGE="…" SRC="…">;
– <IMG SCR="…" ALT="…">;
– <TITLE>…</TITLE>.

7 Где отображается содержание, указанное в теге TITLE?

– в поле STATUS;
– в рабочей области как дополнительное поле;
– в заголовке окна браузера;
– вообще не отображается;
– нет верных ответов.

8 Какой из приведенных тегов описывает тело HTML-документа?

– <HEAD>;
– <BODY>;
– <META>;
– <NOBR>;
– в приведенном списке нет таких тегов.

9 Какой из приведенных тегов описывает заголовок HTML-документа?

– <HEAD>;
– <BODY>;
– <META>;
– <NOBR>;
– в приведенном списке нет таких тегов.

10 Какой из приведенных примеров задает гипертекстовую ссылку на ДРУГОЙ документ?

– <A HREF=”#m1”>…</A>;
– <A HREF=”/works.shtml#m1”>…</A>;
– <A HREF=”/wks.shtml”>…</A>;
– <A HREF=m1>…</A>;
– нет правильных вариантов.

11 С помощью каких тегов описывается таблица и ее элементы?

– <TABLE>;
– <TR>;
– <OL>;
– <TD>;
–
.

12 Как объединить несколько ячеек таблицы в строке?

– с помощью атрибута ROWSPAN;

– с помощью атрибута COLSPAN;
– с помощью атрибута CELLPADDING;
– с помощью атрибута CELLSPACING;
– объединить несколько ячеек нельзя.

13 Что определяет атрибут BORDER у элемента разметки TABLE?

– расстояние между ячейками;
– расстояние от содержания до границы ячейки;
– ширину ячейки;
– ширину границы;
– вид границы.

ЛАБОРАТОРНАЯ РАБОТА №2
КАСКАДНЫЕ ТАБЛИЦЫ СТИЛЕЙ

Цель работы: изучить основы каскадных таблиц стилей, научиться применять каскадные таблицы стилей для оформления HTML-страниц.

Ход выполнения лабораторной работы должен быть отражен в отчете. Отчет должен содержать титульный лист, номера заданий, коды программ, скриншот с результатом выполнения программы.

2.1 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Каскадные таблицы стилей (Cascading Style Sheets – CSS) – это специальный язык описания стилей, который обладает гораздо более богатым и функциональным, по сравнению с HTML, набором средств форматирования и управления стилями элементов документа. Термин «*каскадный*» означает, что в одной странице HTML могут использоваться разные стили.

Другой аспект каскадирования – *наследование*. Наследование означает, что если не указано иное, то конкретный стиль будет унаследован другими элементами страницы HTML. Например, если применить определенный цвет текста в теге <p>, то все теги внутри этого абзаца наследуют этот цвет, если не оговорено иное.

2.1.1 Правила CSS

Простейшее правило CSS задается следующим образом:

Селектор { свойство CSS: значение }

Селектор это любой из тегов HTML (например BODY, P, H1, LI). Далее в фигурных скобках декларируется значение свойств CSS, определяющих стиль данного элемента в документе.

Пример 2.1

```
H1 { color:red }
```

Для удобства применения можно декларировать в одном правиле несколько свойств CSS для нескольких селекторов.

Пример 2.2

```
BODY {
    background-color:white;
    color:black;
    font-family:Times New Roman;
    font-style:normal;
    font-size:10pt }
```

Пример 2.3

H1, H2, H3, H4, H5, H6 { color:black; font-style:italic }

При задании правил можно использовать принцип наследования, который задается при помощи контекстных селекторов, стиль которых зависит от вложенности в другие элементы. Контекстные селекторы задаются без разделительных запятых:

Основной_селектор Вложенный_селектор { свойство CSS: значение }

Пример 2.4

P STRONG { color:red; font-weight:bold }

В правилах CSS используется понятие класса. Класс элемента задается следующим образом:

Селектор.Класс { свойство CSS: значение }

Пример 2.5

H1.normal { color:black; font-style:normal; font-family:serif }
H1.funny { color:blue;font-style:italic; font-family:fantasy }

Для использования в HTML-документе элемента конкретного класса применяется атрибут CLASS:

<H1 CLASS=normal>Технические параметры устройства</H1>

<H1 CLASS=funny>Новости</H1>

В CSS можно определять классы, не связанные с конкретными элементами:

.Класс { свойство CSS: значение }

Пример 2.6

.red { color:red }

В HTML-документе использование данного стиля выглядит так:

Цена со скидкой 10%. Спешите купить

Элементы SPAN и DIV широко используются вместе с селекторами классов. Элемент SPAN предназначен для работы со строковыми элементами, поэтому используется для изменения стиля встроенных элементов HTML-документов, таких как отдельный символ или их набор. Элемент DIV позволяет применять стиль сразу к целому блоку и часто используется для создания слоев.

2.1.2 Методы применения таблиц стилей

Существует три метода для применения таблицы стилей к документу HTML.

Встроенный. Этот метод позволяет взять любой тег HTML и добавить к нему стиль. Встроенный стиль применяется к любому тегу HTML с помощью атрибута style.

Пример 2.7

<P style="font: 12pt Courier New"> Данный текст отображается шрифтом Courier New размером 12 пт. </P>

Внедренный. Внедренный стиль задает стиль оформления всей HTML-страницы. При использовании элемента STYLE, помещенного внутри раздела HEAD HTML-страницы, в код вставляются детализированные атрибуты стиля, которые будут применяться ко всей странице.

Связанный. Связанная таблица стилей – мощный инструмент, который позволяет создавать образцы стилей, которые можно затем применять ко всему web-сайту. Основной документ таблицы стилей (расширение .css) создается web-дизайнером. Этот документ содержит стили, которые будут едиными для всех страниц web-сайта. Любая страница, связанная с этим документом, будет использовать указанные стили.

Пример 2.8

```
/* CSS документ, сохраненный в файле style-1.css */
p
{       text-indent: 20px;
        text-align:justify;
}
body
{       background-color:#00FF66;
        font-family: "Comic Sans MS", "Century Gothic";
}
.style1
{       font-size: 24px;
        font-weight: bold;
        text-align:center;
}
.menu {background-color: #0000FF;}
a {
        font-size: 18px;
        font-weight: bolder;
        color: #FF0000;
        background-color: #0099FF;
}
a:hover { background-color: #0000FF; }
.term
{       font-size: 12pt;
```

```
        color:6600cc;
        font-family: Courier New;
}
.term2
{       font-size: 14pt;
        color:cc33cc;
        font-family: Courier New;
}
```

Данный документ необходимо сохранить в отдельном файле под именем style-1.css.

С этим документом можно связать любое количество HTML-страниц. Для этого нужно в заголовок HTML-страницы вставить элемент LINK:

```
<LINK REL="stylesheet" TYPE="text/css" HREF="style-1.css">
```

Любая страница, содержащая такую связь, будет оформлена в соответствии со стилями, указанными в файле style_1.css.

Пример 2.9

Код страницы, которая будет оформлена в соответствии со стилями, описанными в файле style_1.css.

```
<HTML>
<HEAD>
        <TITLE>Примеры каскадных таблиц стилей</title>
        <LINK REL="stylesheet" TYPE="text/css" HREF="style.css">
</HEAD>
<BODY>
<H1>Тело документа</H1>
<H2 CLASS=term2>BODY</H2>
<P>Указывает начало и конец тела HTML-документа. </P>
<P>Атрибуты:</P>
<UL>
<LI> <SPAN CLASS=term>BGCOLOR</SPAN> – определяет цвет фона документа. </LI>
<LI> <SPAN CLASS=term>TEXT</SPAN> – определяет цвет текста в документе. </LI>
<LI><SPAN CLASS=term>LINK</SPAN> – определяет цвет гиперссылок в документе. </LI>
<LI><SPAN CLASS=term>ALINK</SPAN> – определяет цвет подсветки гиперссылок в момент нажатия.
<LI> <SPAN CLASS=term>VLINK</SPAN> – определяет цвет гиперссылок на документы, которые вы уже просмотрели.
</UL>
</BODY>
</HTML>
```

2.1.3 Свойства CSS

С помощью CSS можно определять гарнитуру, размер и стиль начертания шрифта (таблица 2.1).

Свойство *font-size* может принимать также значения, которые устанавливают относительные размеры шрифта: *xx-small* | *x-stnall* | *small* | *medium* | *large* | *x-large* | *xx-large*, а также *smaller* | *larger*, где *medium* – размер, установленный в браузере в качестве нормального по умолчанию. Каждый следующий размер отличается от предыдущего в 1,2 раза.

Таблица 2.1 – Свойства шрифта

Свойство	Значение	Описание	Пример
font-family	Гарнитура шрифта	*Гарнитура шрифта*	p {font-family: Axial, sans-serif}
		Определяет список гарнитур шрифта в порядке уменьшения приоритета	
font-style	normal	*Начертание шрифта*	p {font-style: italic}
		Обычный шрифт (по умолчанию)	
	italic	Курсивный шрифт	
	oblique	Косой шрифт	
font-variant	normal	*Регистр шрифта*	p {font-variant: small-caps}
		Нормальный (по умолчанию)	
	small-caps	Малые прописные буквы	
font-weight	normal	*Толщина шрифта*	p {font -weight: bold}
		Нормальная жирность	
	lighter	Светлее	
	bold	Полужирный	
	bolder	Жирнее	
	100 – 900	100 – светлый шрифт, 400 – нормальный шрифт, 700 – полужирный шрифт, 900 – самый жирный	
font-size	normal	*Размер шрифта*	p {font-size: 12pt}
		Нормальный размер	
	pt	Пункты	
	px	Пикселы	
	%	Проценты	

Свойства текста применяются для форматирования текста – выравнивания, разрежения и т. д. Значения свойств приведены в таблице 2.2.

Таблица 2.2 – Свойства оформления текста

Свойство	Значение	Описание	Пример
1	2	3	4
line-height	normal (100 %, по умолчанию) Множитель	Минимальная высота строк (интерлиньяж, межстрочный интервал)	line-height: normal line-height: 1.5 line-height: 12px line-height: 120 %
	Точно (значение в единицах длины)		
	Процент от размера шрифта		

Продолжение таблицы 2.2

1	2	3	4
text-decoration	*Оформление текста*		text-decoration: none
	None	Оформление отсутствует (по умолчанию)	
	Underline	Подчеркивание	
	Overline	Черта над текстом	
	line-through	Перечеркивание	
	blink	Мигание текста	
text-transform	*Регистр текста*		text-transform: capitalize
	none	Регистр не переключается (по умолчанию)	
	capitalize	Начальные буквы слов преобразуются в заглавные	
	uppercase	Все прописные	
	lowercase	Все строчные	
text-align	*Выравнивание текста в блочных элементах*		text-align: justify
	left	Выравнивание влево	
	right	Выравнивание вправо	
	center	Выравнивание по центру	
	justify	Выравнивание по ширине	
letter-spacing	Единицы длины	Расстояние между буквами (для сжатия можно использовать отрицательные значения)	letter-spacing: 5px
word-spacing	Единицы длины	Интервал между словами	word-spacing: 5px
white-space	*Работа с пробелами*		white-space: pre
	normal	Обычный режим – лишние пробелы игнорируются, строки переносятся автоматически	
	pre	Все пробелы учитываются	
	nowrap	Запрещен перенос строк, лишние пробелы игнорируются	
text-indent	*Отступ первой строки*		text-indent: 15px; text-indent: 10 %
	Единицы длины	Отступ в единицах длины	
	%	Отступ в процентах от ширины блока	

Каскадные таблицы стилей имеют несколько свойств для определения цвета текста и фоновых областей HTML-страницы. Эти свойства не только заменяют аналогичные в HTML, но и дают массу новых возможностей. Например, традиционный путь для создания на HTML-странице цветного блока заключается в применении таблиц. Стили позволяют отказаться от использования таблиц, предлагая более простые и удобные варианты управления цветом.

Свойства CSS для управления цветом и фоном приведены в таблице 2.3.

Таблица 2.3 – Свойства CSS для управления цветом и фоном

Свойство	Значение	Описание	Пример
Color	Цвет	Устанавливает цвет текста	P {color: #999999}
background-color	*Цвет фона*		BODY {background-color: #888888}
	Цвет	Сплошной цвет	
	transparent	Прозрачный фон	
background-image	*Фоновый рисунок*		BODY {background-image: url (pictures/bg.gif)}
	URL	URL рисунка	
	none	Нет рисунка (по умолчанию)	
background-repeat	*Повторяемость фонового рисунка*		{background-repeat: repeat-y}
	repeat	Повторяет рисунок по горизонтали и вертикали	
	repeat-x	Повторяет рисунок только по горизонтали	
	repeat-y	Повторяет рисунок только по вертикали	
background-position	Проценты	Начальное положение фонового рисунка относительно левого верхнего угла блока	BODY {background-position: left top }
	Единицы длины		
	top		
	center		
	bottom		
	left		
	right		

С помощью CSS можно создать нумерованные и ненумерованные списки, причем в качестве маркера может быть использовано любое изображение.

В таблице 2.4 перечислены свойства элементов, предназначенные для форматирования списков.

Таблица 2.4 – Свойства списков

Свойство	Значение	Описание	Пример
1	2	3	4
list-style-type	*Вид маркера для ненумерованного списка*		li {list-style: circle;} li {list-style: upper-alpha;}
	disc	Маркер	
	circle	Окружность	
	square	Квадрат	

Продолжение таблицы 2.2

1	2	3	4
list-style-type	*Вид маркера для ненумерованного списка*		li {list-style: circle;} li {list-style: upper-alpha;}
	disc	Маркер	
	circle	Окружность	
	square	Квадрат	
	Вид маркера для нумерованного списка		—
	decimal	Десятичные (арабские) цифры	
	decimal-leading-zero	Десятичные цифры с добавлением нуля в начале	
	lower- roman	Строчные латинские цифры	
	upper-roman	Прописные латинские цифры	
	lower-alpha	Строчные русские буквы	
	upper-alpha	Прописные русские буквы	
	none	Нумерация отсутствует	
list-style-image	*Устанавливает изображение в качестве символа маркера*		li {list-style-image: url(images/bullet.gif)}
	URL	URL изображения	
	none	Нет изображения (по умолчанию)	
list-style-position	outside	Положение маркера относительно строк текста	li {list-style-position: inside;}
	inside		

Спецификация CSS описывает несколько свойств, с помощью которых можно работать с блочными элементами: создавать границу вокруг различных элементов, отступы и поля. Принцип расположения блочных элементов на web-странице таков. Внутри блока находится контент (чаще всего текст), который отделяется от границ блока отступами (padding). Далее следует граница (border), которая имеет свои параметры. За границей располагается поле (margin), которое отделяет весь блок от других блоков (например, от тела документа, определенного элементом BODY).

Таблица 2.5 – Свойства границ, отступов и полей блочных элементов

Свойство	**Значение**	**Описание**	**Пример**
1	2	3	4
PADDING	Единицы длины или процент от ширины блока	*Отступы от границы элемента до его содержимого*	table {padding: 15px)
padding-top		сверху	
padding-right		справа	
padding-bottom		снизу	
padding-left		слева	
Padding		одинаковые для всех сторон	

Продолжение таблицы 2.5

1	2	3	4
MARGIN	Единицы длины или процент от ширины блока	*Размеры полей*	p {margin-top: 30px; margin-bottom: 5 % }
margin-top		сверху	
margin-right		справа	
margin-bottom		снизу	
margin-left		слева	
margin		одинаковой ширины со всех сторон	
BORDER	—	*Ширина рамки*	P {border-top-width: 5px}
border-top-width		верхней границы	
border-right-width		правой границы	
border-bottom-width		нижней границы	
border-left-width		левой границы	
border-width		всех границ одновременно	
—	thin	Тонкая	
	medium	Средней толщины	
	thick	Толстая	
	единицы длины	—	
border-top-color	Значение цвета	*Цвет рамки для границ*	P {border-color: rgb(90,60,90}
border-right-color			
border-bottom-color			
border-left-color			
border-color			
transparent			
border- style	none	*Стиль рамки*	table {border-style: double}
	dotted		
	dashed		
	solid		
	double		
	groove		
	ridge		
	inset		
	outset		
border-top **border-right** **border-bottom** **border-left**	Ширина, стиль, цвет	*Определяет толщину, стиль и цвет каждой границы*	table {border-top: solid 3px red; border-left: solid 3px green}
border	Ширина, стиль, цвет	*Определяет толщину, стиль и цвет всей рамки*	table {border: solid 3px red}

2.2 ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

1 Создать css-файл, содержащий правила каскадных таблиц стилей, для задания стилей оформления созданных в лабораторной работе №1 HTML-страниц.

2 В каскадной таблице стилей задать свойства для следующих элементов (для каждого элемента задать не менее пяти свойств):

– стиль оформления тела документа;

– стиль оформления таблицы;

– используя классы, описать стиль оформления цены со скидкой и без скидки;

– используя классы, описать стиль оформления номера группы, ФИО студента и названия дисциплины;

– стиль оформления нумерованных и маркированных списков перечня лабораторных работ и заданий.

3 Присоединить к созданным HTML-страницам файл, содержащий каскадные таблицы стилей для оформления созданных страниц.

2.3 КОНТРОЛЬНЫЕ ВОПРОСЫ

1 Могут ли в одной HTML-странице использоваться разные каскадные таблицы стилей?

– да;

– нет;

– зависит от версии web-браузера;

– зависит от версии web-сервера;

– зависит от настроек браузера.

2 Что является селектором (указателем), а что определением в данном примере?

– H1 {color:red;};

– H1 – селектор;

– red – определение;

– {color:red} – селектор;

– {color:red} – определение;

– H1 – определение.

3 Что может использоваться в качестве селектора в каскадных таблицах стилей?

– любой тег HTML;

– только теги форматирования теста;

– имя класса типа .class_name;

– имя класса типа class_name;

– нет верных ответов.

4 Выберите правильные варианты определения стиля элемента:

– P {font-family: helvetica;};

– H1, H2, H3 {color:black; font-style:italic;};

– P STRONG {color:red;};

– H1.nornal {color:blue;};

– .red {color:red;}.

5 Как указывается встроенный стиль?

– в HTML-теге с помощью атрибута STYLE;

– тег <STYLE> помещается внутри раздела <HEAD>;

– стиль хранится в отдельном файле и подключается к документу с помощью тега <LINK>;

– все ответы верны;

– нет верных ответов.

6 Как указывается связанный стиль?

– в HTML-теге с помощью атрибута STYLE;

– стиль хранится в отдельном файле и подключается к документу с помощью тега <LINK>;

– тег <STYLE> помещается внутри раздела <HEAD>;

– все ответы верны;

– нет верных ответов.

7 Как указывается внедренный стиль?

– тег <STYLE> помещается внутри раздела <HEAD>;

– в HTML-теге с помощью атрибута STYLE;

– стиль хранится в отдельном файле и подключается к документу с помощью тега <LINK>;

– все ответы верны;

– нет верных ответов.

ЛАБОРАТОРНАЯ РАБОТА №3
HTML-ФОРМЫ. ОБРАБОТКА ДАННЫХ, ВВЕДЕННЫХ В ФОРМУ

Цель работы: изучить основные элементы, которые могут быть размещены в HTML-форме и научиться обрабатывать данные, введенные в форму с помощью языка программирования PHP.

Ход выполнения лабораторной работы должен быть отражен в отчете. Отчет должен содержать титульный лист, номера заданий, коды программ, скриншот с результатом выполнения программы.

3.1 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Примечание *– Для правильного выполнения PHP-скрипта страница, содержащая такой скрипт, должна быть сохранена с расширением .php в каталоге, который указан в настройках сервера в директиве DocumentRoot. Эта директива определяет местонахождение корневого каталога документов сервера. При установке web-сервера Apache, PHP и MySQL, используя денвер, по умолчанию в директиве DocumentRoot устанавливается значение Z:\home\localhost\www. В этой папке необходимо создать папку с номером вашей группы, а в ней – папку с вашей фамилией (папки должны быть названы латинскими буквами или цифрами без пробелов и специальных символов). Для запуска созданного скрипта необходимо открыть браузер и в адресной строке ввести следующий адрес: http://localhost/Группа/Фамилия/имя_скрипта.php.*

3.1.1 Элемент FORM

Получение и обработка данных, введенных пользователем, стали неотъемлемой частью большинства успешных web-сайтов. Ввод информации в основном реализуется с применением HTML-форм.

При вводе данных в форму используются различные управляющие элементы: поля ввода текста и пароля, скрытые поля, поля со списком, переключатели, флажки, кнопки и т. д.

Одна страница может содержать несколько форм. Обработка элементов формы производится с помощью скриптов.

Одним из распространенных языков программирования для написания скриптов, обрабатывающих данные, введенные в формы, является PHP. В официальной документации язык PHP подается как встраиваемый в HTML скриптовый язык с обработкой на сервере.

Форма в HTML-документе реализуется тегом-контейнером FORM:

<FORM ACTION = "действие" METHOD = "метод">
– элементы формы –
</FORM>

Тег **FORM** имеет следующие атрибуты.

ACTION – адрес сценария (скрипта), которому будет послана форма. Это единственный обязательный атрибут.

METHOD – определяет метод HTTP, используемый для пересылки данных формы от браузера к серверу. Атрибут METHOD может принимать два значения: GET и POST. Если используется метод GET (он же установлен по умолчанию), то все поля, описанные в форме, передаются в строке URL в следующем виде: URL?name=value&name=value. При использовании метода POST поля формы кодируются таким же образом, но передаются через скрытые переменные, не используя строку URL. Метод POST обычно используется при передаче большого объема данных или если необходимо скрыть от пользователя передаваемый набор полей.

Начальный и конечный теги FORM задают границы формы, поэтому их указание является обязательным.

Пример 3.1

Обработка данных, вводимых пользователем на странице form1.html, при помощи скрипта reader.php, расположенного в том же каталоге.

```
<HTML>
    <HEAD>
        <TITLE> Пример HTML-формы </TITLE>
    </HEAD>
    <BODY>
        <H1> Пример HTML-формы </H1>
        <FORM METHOD= "POST" ACTION= "reader.php" >
            Здесь должны быть элементы формы.
        </FORM>
    </BODY>
</HTML>
```

Задание 3.1:

3.1.1 Запустить web-сервер Apache (D:\work\Start servers.lnk).

3.1.2 Проверить работу web-сервера Apache: запустить web-браузер Internet Explorer и в адресной строке ввести адрес локального сервера http://localhost/ (рисунок 3.1).

3.1.3 В корневом каталоге сервера Z:\home\localhost\www создать папку со своей фамилией и номером группы (желательно латинскими буквами без пробелов): Z:\home\localhost\www\171501\Ivanov\.

3.1.4 Создать HTML-документ, содержащий форму и сохранить его под именем form1.php в своем каталоге на диске Z. Заголовок HTML-документа – «Форма ввода персональных данных пользователя».

3.1.2 Текстовое поле

В текстовых полях обычно вводится короткая текстовая информация –

скажем, адрес электронной почты, почтовый адрес или имя. Особой разновидностью текстовых полей является поле ввода паролей и скрытое поле. В поле ввода пароля вводимые символы заменяются звездочками. Скрытые поля не отображаются в браузере и обычно используются для передачи служебной информации между сценариями.

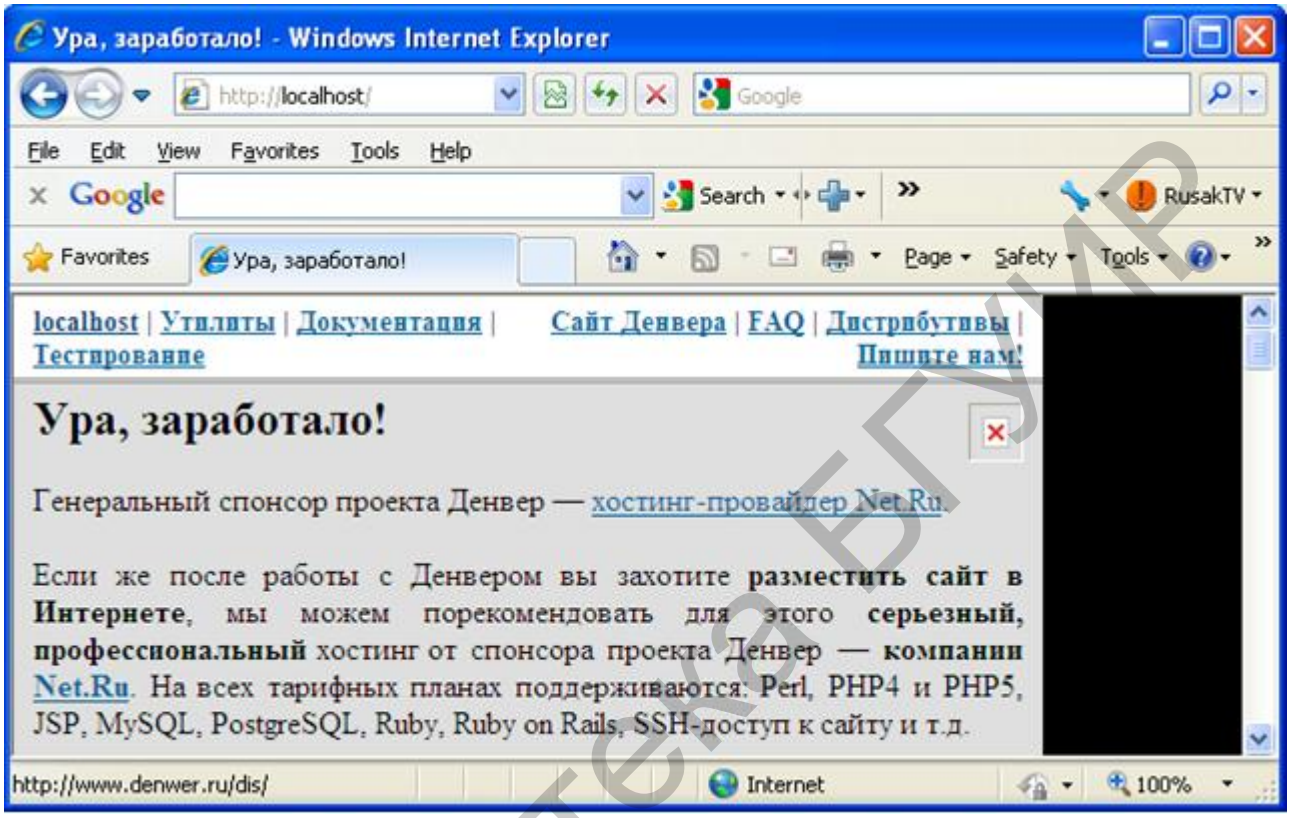

Рисунок 3.1 – Проверка работы локального web-сервера

Синтаксис определения текстового поля:

<INPUT TYPE="тип" NAME="имя_переменной" SIZE="N" MAXLENGTH="N" VALUE="">

Определение текстового поля включает пять атрибутов.

TYPE – тип элемента (для текстовых полей – TEXT, для полей ввода паролей – PASSWORD, для скрытых полей – HIDDEN).

NAME – имя переменной, в которой сохраняются введенные данные.

SIZE – общий размер текстового поля в браузере.

MAXLENGTH – максимальное количество символов, вводимых в текстовом поле.

VALUE – значение, отображаемое в текстовом поле по умолчанию.

Пример 3.2

Создание в форме поля для ввода имени пользователя и пароля. Для удобства форматирования элементы формы можно разместить в таблице.

…

<FORM METHOD= "POST" ACTION= "reader.php" >

```
<TABLE>
<TR><TD>Введите ваше имя:</TD>
      <TD><INPUT NAME ="Name" TYPE="TEXT" > </TD>
</TR>
<TR><TD>Введите ваш пароль:</TD>
      <TD><INPUT NAME ="Password" TYPE="PASSWORD" > </TD>
</TR>
</TABLE>
</FORM>
      …
```

Задание 3.2. *Добавить в созданную форму текстовые поля для ввода ФИО пользователя, логина, пароля и электронного адреса. Все элементы формы разместить в таблице. Таблица располагается по центру, и границы таблицы не отображаются.*

3.1.3 Кнопка отправки данных и кнопка сброса

Кнопка отправки данных инициирует действие, заданное атрибутом ACTION тега <FORM>. Кнопка сброса отменяет все изменения, внесенные в элементы формы. Синтаксис определения кнопок:

<INPUT TYPE="тип" VALUE="текст_на_кнопке">

Определение кнопки включает два атрибута.

TYPE – тип элемента (для кнопки отправки данных – SUBMIT, для кнопки сброса данных – RESET).

VALUE – текст, который будет отображаться на кнопке.

Пример 3.3

Добавление к предыдущему примеру кнопки отправки данных.

```
…
<FORM METHOD= "POST" ACTION= "reader.php" >
<TABLE>
  <TR><TD>…</TD> <TD>… </TD></TR>
  <TR><TD>…</TD> <TD>… </TD></TR>
  <TR><TD COLSPAN=2 >
     <INPUT TYPE="SUBMIT" VALUE="Отправить">
   </TD></TR>
</TABLE>
</FORM>
…
```

Задание 3.3. *Добавить в созданную форму кнопку отправки данных и кнопку сброса.*

3.1.4 Обработка введенных в форму данных

Введенные в форму данные посылаются на обработку после щелчка на кнопке формы отправки данных (SUBMIT). Информация из полей формы передается затем на страницу, определенную атрибутом ACTION элемента FORM. Когда запрос этого URL и соответствующие данные формы приходят на сервер, вызывается указанная страница и данные передаются ей для обработки.

При использовании для обработки данных языка программирования PHP все данные из полей формы помещают в глобальный массив: $_GET, $_POST и $_REQUEST. $_GET[] и $_POST[] – ассоциативные массивы, которые содержат все значения, передаваемые в сценарий с помощью метода формы GET или POST соответственно. $_REQUEST[] – ассоциативный массив, который содержит все значения, передаваемые в сценарий методов POST и GET. Ключами этих массивов являются имена переменных, которые указаны в атрибуте NAME элементов формы, а значениями – введенные или выбранные в элементах формы данные.

Скрипты на PHP легко встраиваются в HTML-код. Для обеспечения необходимой гибкости при построении динамических web-приложений можно внедрить в страницу несколько сценариев PHP. При внедрении нескольких сценариев переменные, значения которых были присвоены в одном сценарии, могут использоваться в другом сценарии той же страницы. Кроме того, код HTML может интегрироваться прямо в команды PHP.

Существуют четыре варианта оформления перехода в PHP:

– стандартные теги: (**<?php** print "Welcome to the world of PHP!"; **?>**);

– короткие теги (**<? print** "Welcome to the world of PHP!"; **?>**);

– теги script;

– теги в стиле ASP (**<%php** print "Welcome!"; **%>**).

Стандартные теги используются программистами PHP чаще остальных способов, что объясняется наглядностью и удобством этой формы записи.

Пример 3.4

Файл reader.php, в котором данные, введенные в форму, содержащуюся в файле form1.html, выводятся на экран (в окно браузера), используя суперглобальный массив $_POST.

```
<HTML>
 <HEAD>
   <TITLE> Обработка данных формы </TITLE>
 </HEAD>
<BODY>
  <CENTER>
      <H1>Вы ввели в форму следующие значения: </H1>
       Ваше имя <?php echo $_POST["Name"]; ?>
  </CENTER>
</BODY>
</HTML>
```

Задание 3.4. *Создать файл reader.php для вывода на экран в табличной форме пользовательских данных, введенных в форму, которая хранится в файле form1.html.*

3.1.5 Флажок

Флажки используются в ситуациях, когда пользователь выбирает один или несколько вариантов из готового набора. Синтаксис определения флажка:

<INPUT TYPE="CHECKBOX" NAME="имя_переменной" VALUE="начальное_значение" [CHECKED]>

Определение флажка включает четыре атрибута.

TYPE – тип элемента (для флажков – CHECKBOX).

NAME – имя переменной, в которой сохраняются введенные данные (в данном случае – состояние элемента).

VALUE – значение, присваиваемое переменной по умолчанию. Если флажок установлен, именно это значение будет присвоено переменной с указанным именем. Если флажок не установлен, значение атрибута VALUE не используется.

CHECKED – если указан этот атрибут, то флажок по умолчанию включен.

Пример 3.5

Создание в форме трех флажков (файл form.html).

```
…
<B>Какие товары вы покупаете в интернет-магазине:</B><BR>
<INPUT TYPE="CHECKBOX" NAME="book" VALUE="Книги">Книги<BR>
<INPUT TYPE="CHECKBOX" NAME="CD" VALUE="CD-диски">CD-диски<BR>
<INPUT TYPE="CHECKBOX" NAME="DVD" VALUE="DVD-диски">DVD-диски<BR>
…
```

Если флажок неактивен, то параметры вообще не будут переданы на сервер и соответствующие глобальные переменные не будут созданы. Следовательно, при попытке обращения к такой переменной будет выведено сообщение о том, что переменная не существует. Поэтому необходимо предварительно проверить, существует переменная или нет, для этого используется функция **isset()**.

Пример 3.6

Вывод на экран значений выбранных флажков (файл reader.php).

```
…
<P>В интернет-магазине вы покупаете: </P>
<?
  if (isset($_POST["book"]))  echo $_POST["book"]."<BR>";
```

```
  if (isset($_POST["CD"]))  echo $_POST["CD"]."<BR>";
  if (isset($_POST["DVD"]))  echo $_POST["DVD"]."<BR>";
?>
…
```

Задание 3.5. *Добавить в форму флажки для выбора пользователем иностранных языков, которыми он свободно владеет. Вывести на экран выбранные значения вместе с остальными данными.*

3.1.6 Переключатель

Переключатель представляет собой разновидность флажка; он работает практически так же, за одним исключением – в любой момент времени в группе может быть установлен лишь один переключатель. Синтаксис определения переключателя:

<INPUT TYPE="RADIO" NAME="имя_переменной"
VALUE="начальное_значение" [CHECKED]>

Как видите, синтаксис почти не отличается от определения флажка.

Определение переключателя поля включает четыре атрибута.

TYPE – тип элемента (для переключателей – RADIO).

NAME – имя переменной, в которой сохраняются введенные данные (в данном случае – состояние элемента).

VALUE – значение, присваиваемое переменной по умолчанию. Если флажок установлен, именно это значение будет присвоено переменной с указанным именем. Если переключатель не включен, значение атрибута VALUE не используется.

CHECKED – если указан этот атрибут, то переключатель по умолчанию активен.

Пример 3.7

Создание в форме переключателя (файл form.html).

```
…
<B>Ваш пол:</B><BR>
<INPUT TYPE="RADIO" NAME="sex" VALUE="Мужчина"
CHECKED >Мужчина<BR>
<INPUT TYPE="RADIO" NAME="sex"
VALUE="Женщина">Женщина<BR>
…
```

Пример 3.8

Вывод на экран значения переключателя (файл reader.php).

```
…
echo "Ваш пол: ".$_POST["sex"]."<BR>";
…
```

Задание 3.6. *Добавить в форму два переключателя – для выбора пользователем его пола и уровня образования (среднее, среднее специальное, высшее и т. д.). Вывести на экран выбранные значения вместе с остальными данными.*

3.1.7 Раскрывающийся список

Раскрывающиеся списки позволяют выбрать один вариант из множества. Как правило, раскрывающиеся списки применяются при работе с относительно большими наборами данных. Синтаксис определения раскрывающегося списка:

```
<SELECT NAME="имя_переменной">
  <OPTION VALUE="значение_переменной1">Вариант 1</OPTION>
  <OPTION VALUE="значение_переменной2">Вариант 2</OPTION>
  <OPTION VALUE="значение_переменной3">Вариант 3</OPTION>
  <OPTION VALUE="значение_переменной4">Вариант 4</OPTION>
</SELECT>
```

Определение раскрывающегося списка поля включает два атрибута.

NAME – имя переменной, в которой сохраняются введенные данные (в данном случае – строка, выбранная в списке).

VALUE – значение переменной, которое соответствует выбранной строке.

Пример 3.9

Создание в форме раскрывающегося списка (файл form.html).

```
…
<B>Какая пора года вам нравится больше всего?</B><BR>
<SELECT NAME="pora_goda">
  <OPTION VALUE="Зима"> Зима </OPTION>
  <OPTION VALUE="Лето"> Лето </OPTION>
  <OPTION VALUE="Осень"> Осень </OPTION>
  <OPTION VALUE="Весна"> Весна </OPTION>
</SELECT>
…
```

Пример 3.10

Вывод на экран значения, выбранного в раскрывающемся списке (файл reader.php).

```
…
echo "Ваша любимая пора года: ".$_POST["pora_goda"]."<BR>";
…
```

Задание 3.7. *Добавить в форму два раскрывающихся списка – для выбора города проживания и профессии. Вывести на экран выбранные значения вместе с остальными данными.*

3.1.8 Текстовая область

Текстовая область используется для ввода нескольких больших объемов текста, не ограничивающихся простым именем или адресом электронной почты. Синтаксис определения текстовой области:

<TEXTAREA NAME="имя_переменной" ROWS="N" COLS="N" VALUE=""></TEXTAREA>

Определение текстового поля включает четыре атрибута.

NAME – имя переменной, в которой сохраняются введенные данные.

ROWS – количество строк в текстовой области.

COLS – количество столбцов в текстовой области.

VALUE – значение, присваиваемое переменной по умолчанию.

Пример 3.11

Создание в форме текстовой области (файл form.html).

```
…
<B>Дополнительная информация:</B><BR>
<TEXTAREA NAME="comments" ROWS="5" COLS="5" VALUE="Здесь вы можете написать свои комментарии"></TEXTAREA>
…
```

Пример 3.12

Вывод на экран значения текстового поля (файл reader.php).

```
…
echo " Дополнительная информация: ".$_POST["comments"]."<BR>";
…
```

Задание 3.8. *Добавить в форму текстовую область для ввода вопроса пользователя. Вывести на экран введенный вопрос вместе с остальными данными.*

3.1.9 Ввод и обработка данных формы в одном сценарии

При вводе и обработке данных в одном сценарии атрибут ACTION формы должен ссылаться на ту же страницу, в которой определяется сама форма. С помощью условной конструкции if необходимо проверить состояние переменной окружения REQUEST_METHOD, которая хранится в суперглобальном массиве $_SERVER. Переменная окружения REQUEST_METHOD получает свое значение, равное методу передачи информации GET или POST, только после нажатия кнопки отправки данных.

Пример 3.13

Ввод и обработка данных в одном скрипте, который сохранен в файле form.php.

```
<HTML>
<HEAD>
```

```
   <TITLE> Пример HTML-формы </TITLE>
</HEAD>
<BODY>
<?
if ($_SERVER["REQUEST_METHOD"] == "POST")
{
  echo "<h1>Привет, <b>" . $_POST["Name"] . "!</b></h1>";
}
else {
?>
<H1> Пример HTML-формы </H1>
<FORM METHOD= "POST" ACTION= "form.php" >
<TABLE>
<TR><TD>Введите ваше имя:</TD>
        <TD><INPUT NAME ="Name" TYPE="TEXT" > </TD>
</TR>
<TR>
     <TD COLSPAN=2 >
        <INPUT TYPE="SUBMIT" VALUE="Отправить">
     </TD>
  </TR>
</TABLE>
</FORM>
<? }?>
</BODY>
</HTML>
```

3.2 ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

1 Создать один скрит, который бы при первом запуске выводил на экран форму для ввода пользовательских данных (форма должна содержать все возможные элементы управления), а после нажатия кнопки обработки данных выводил бы на экран введенные пользователем данные.

2 Добавить в форму поле для ввода подтверждения пароля. В обработчике проверить правильность ввода пароля (в поле с паролем и в поле с подтверждением пароля должен быть введен один и тот же пароль).

Пример проверки:

```
<? if ($_POST["pass1"]!=$_POST["pass2"])
        echo "Введенные пароли отличаются";
?>
```

3 Добавить в форму поля ФИО, e-mail, пароль пользователя и введенный пользователем вопрос – обязательные поля. Проверить заполнение этих полей, в случае если пользователь не заполнил эти поля, выдать сообщение об ошибке и повторно вывести на экран форму.

Пример проверки:

```
<? if ($_POST["Name"]=="")
        echo "Не введена фамилия";
?>
```

4 Обязательные поля в форме отметить красной звездочкой и написать внизу формы примечание, что эти поля должны быть заполнены обязательно.

3.3 КОНТРОЛЬНЫЕ ВОПРОСЫ

1 Какие элементы используются для задания элементов в форме?

– <INPUT …>;
– <SELECT …>;
– <IMG …>;
– <TEXTAREA …>;
– <SUBMIT … >.

2 Какие теги используются для создания текстовых полей ввода в форме?

– <INPUT TYPE="text">;
– <TEXTAREA>;
– <SELECT>;
– <INPUT TYPE="password">;
– <OPTION>.

3 Какие методы можно применять для отправки данных формы?

– метод GET;
– метод HEAD;
– метод POST;
– метод MAILTO;
– нет верного ответа.

4 Если адрес в браузере имеет вид http://ex.com/script.cgi?time=+5, каким методом передавались данные, введенные в форму?

– метод GET;
– метод HEAD;
– метод POST;
– метод MAILTO;
– нет верного ответа.

5 При каком методе передачи данных формы данные никак не интерпретируются сервером, а пересылаются непосредственно сценарию?

– метод GET;
– метод HEAD;
– метод POST;
– метод MAILTO;
– нет верного ответа.

6 Способ посылки параметров сценарию, когда данные помещаются в командную строку URL, называется:

– метод GET;
– метод HEAD;
– метод POST;
– метод MAILTO;
– нет верного ответа.

7 С помощью какого атрибута элемента FORM указывается адрес, по которому нужно отправлять данные формы?

– ACTION;
– HREF;
– LOCATION;
– METHOD;
– TERGET.

8 С помощью какого атрибута элемента FORM указывается метод, которым будут передаваться данные формы?

– TERGET;
– METHOD;
– LOCATION;
– HREF;
– ACTION.

9 Сколько форм может содержать одна HTML-страница?

– только одну;
– две;
– HTML-страница не может содержать форму;
– три;
– любое количество.

10 Выберите правильные варианты кода для создания скрытого текстового поля:

– <input type="text" name="text" size="5" maxlength="10" value="">;
– <input type="password" name="name">;
– <input type="hidden" name="hidden" value="y">;
– <input type="hidden" name="name" value="y">;
– <textarea name="text" rows="5" cols="5" value=""></textarea>.

11 Выберите правильные варианты кода для создания текстового поля для ввода пароля:

– <input type="text" name="password" size="5" maxlength="10" value="">;
– <input type="password" name="name">;
– <input type="password" name="hidden" value="password">;
– <input type="password" name="password" value="y">;
– <textarea name="text" rows="5" cols="5" value=""></textarea>.

12 Выберите правильные варианты кода для создания текстового поля:

– <input type="text" name="name" size="5" maxlength="10">;

– <input type="text" name="password">;

– <input type="text" name="hidden" value="y">;

– <input type="submit" name="text">;

– <input type="checkbox" name="text">.

13 Выберите правильные варианты кода для создания текстовой области:

– <textarea name="text" rows="5" cols="5" value="">;

– <textarea name="name" rows="5" cols="5" value=""></textarea>;

– <textarea name="text" rows="5" cols="5" value="text"></textarea>;

– <input type="textarea" name="name" rows="5" cols="5" value="">;

– <textarea name="name" rows="5" cols="5" value="">.

14 Выберите правильные варианты кода для создания переключателей:

– <input type="checkbox" name="name" value="Ваш выбор">;

– <input type="checkbox" name="radio" value="Ваш выбор">;

– <input type="radio" name="checkbox" value="Ваш выбор">;

– <input type="radio" name="name" value="Ваш выбор">;

– <radio name="name" value="Ваш выбор"></radio>.

15 Выберите правильные варианты кода для создания флажка:

– <input type="checkbox" name="name" value="Ваш выбор">;

– <input type="checkbox" name="radio" value="Ваш выбор">;

– <input type="radio" name="checkbox" value="Ваш выбор">;

– <input type="radio" name="name" value="Ваш выбор">;

– <checkbox name="name" value="Ваш выбор"></checkbox>.

16 Выберите правильные варианты кода для создания раскрывающего списка:

– <select name="spisok"><option value="Ваш выбор1"><option value="Ваш выбор2"></select>;

– <select name="spisok" option value="Ваш выбор"></select>;

– <input type="select" name="name" value="Ваш выбор">;

– <select name="spisok" option value="Ваш выбор">;

– <select name="spisok">Ваш выбор</select>.

17 Выберите правильные варианты кода для создания кнопки:

– <input type="submit" value="Кнопка">;

– <input type="reset" value="Кнопка">;

– <input type="reset">;

– <input type="Submit">;

– <submit name="Кнопка"></submit>.

ЛАБОРАТОРНАЯ РАБОТА №4
ОСНОВЫ ЯЗЫКА ПРОГРАММИРОВАНИЯ PHP

Цель работы: изучить основы языка программирования PHP.

Ход выполнения лабораторной работы должен быть отражен в отчете. Отчет должен содержать титульный лист, номера заданий, коды программ, скриншот с результатом выполнения программы.

4.1 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

4.1.1 Язык программирования PHP

PHP – скриптовый язык программирования общего назначения, интенсивно применяемый для разработки web-приложений. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков программирования, применяющихся для создания динамических web-сайтов.

В официальной документации язык PHP подается как встраиваемый в HTML скриптовый язык с обработкой на сервере. Это позволяет сразу же иметь в виду следующее:

– обработка PHP-кода производится на стороне сервера еще до того, как web-страница будет передана браузеру; это отличает язык PHP от языка JavaScript;

– PHP-код может быть непосредственно встроен в HTML-код страницы; этим он отличается от Perl.

Кроме того, PHP – язык, который позволяет встраивать в код программы блоки HTML-кода, что позволяет использовать его для написания CGI-сценариев.

4.1.2 Переход в PHP

Механизм лексического анализа должен как-то отличать код PHP от других элементов страницы. Идентификация кода PHP называется «переходом в PHP» (escaping to PHP). Существуют четыре варианта оформления перехода в PHP:

– стандартные теги;
– короткие теги;
– теги script;
– теги в стиле ASP.

Стандартные теги используются программистами PHP чаще остальных способов, что объясняется наглядностью и удобством этой формы записи:

<?php print "Welcome to the world of PHP!": **?>**

У стандартных тегов есть еще одно дополнительное преимущество: за открывающей конструкцией <? следуют символы php, однозначно определяющие

тип дальнейшего кода. Это удобно при использовании в одной странице нескольких технологий, таких как JavaScript, серверные включения и PHP. Весь текст, расположенный до закрывающего тега ?>, интерпретируется как код PHP.

Короткие теги обеспечивают наиболее компактную запись для перехода в PHP:

<? print "Welcome to the world of PHP!"; **?>**

По умолчанию короткие теги не используются, их нужно специально активизировать, включив параметр short_open_tag в файл php.ini.

Теги script. Некоторые текстовые редакторы ошибочно принимают код PHP за код HTML, что нарушает работу над web-страницей. Проблема решается использованием тегов script:

<script language="php">
<?php print "Welcome to the world of PHP!"; **?>**
</script>

Четвертый и последний способ оформления внедренного кода PHP – *теги в стиле ASP* (Active Server Page). Они похожи на короткие теги, описанные выше, однако вместо вопросительного знака используется знак процента (%):

<%php print "Welcome to the world of PHP!"; **%>**

PHP легко встраивается в HTML-код. Для обеспечения необходимой гибкости при построении динамических web-приложений можно внедрить в страницу несколько сценариев PHP. При внедрении нескольких сценариев переменные, значения которых были присвоены в одном сценарии, могут использоваться в другом сценарии той же страницы. Кроме того, код HTML может интегрироваться прямо в команды PHP (пример 4.1).

Пример 4.1

```
<HTML>
<HEAD>
  <TITLE>PHP Recipes | <? print (date("F d, Y")); ?></TITLE>
</HEAD>
  <? $big_font = "H3"; ?>
<BODY>
  <? print "<$big_font>PHP Recipes</$big_font>"; ?>
</BODY>
</HTML>
```

4.1.3 Комментарии в коде PHP

В PHP существуют два формата комментариев:

– *однострочные комментарии* обычно используются для коротких пояснений или примечаний, относящихся к локальному коду;

– *многострочные комментарии* обычно используются при оформлении алгоритмов на псевдокоде и в более подробных объяснениях.

Оба способа в конечном счете приводят к одинаковому результату и со-

вершенно не влияют на общее быстродействие сценария.

При оформлении *однострочных комментариев* используется два стиля комментирования. В одном случае комментарий начинается с двойного символа «косая черта» (//), а в другом – с символа фунта (#). Оба стиля работают абсолютно одинаково (пример 4.2).

Многострочные комментарии оформляются в стиле языка С – их начало и конец обозначаются символами /* и */. Многострочные комментарии особенно удобны для вывода относительно длинной сводной информации обо всем сценарии или его части (см. пример 4.2).

Пример 4.2

```
<?
   echo("Hello"); //это комментарии
   echo("Hello"); #это комментарии
   /* а это многострочные
   комментарии */
?>
```

4.1.4 Типы данных в PHP

Типы данных составляют основу любого языка программирования и являются средством, с помощью которого программист представляет разные типы информации. В PHP поддерживаются шесть основных типов данных:

- целые числа (integer);
- вещественные числа (float, double, real);
- строки (string);
- массивы (array);
- объекты (object);
- логические величины (bool).

Целое число со знаком, обычно длиной 32 бита. В PHP поддерживается запись целых чисел в десятичной (52, 14), восьмеричной (0422, 0524) и шестнадцатеричной (0x3FF, 0x22abc) форме.

Вещественные числа (числа с плавающей точкой) отличаются от целых наличием дробной части. Они используются для представления значений, требующих повышенной точности, например температур или денежных величин. В PHP поддерживаются два вещественных формата: стандартная (12.45, 98.6) и научная (3e8, 5.9736e24) запись.

Строкой (string) называется последовательность символов. Строка легко может быть обработана при помощи стандартных функций, допустимо также непосредственное обращение к любому ее символу. Длина строки ограничена только размером свободной памяти, так что можно прочитать в одну строку целый файл размером около 200–300 Кбайт.

В PHP, как и в большинстве современных языков программирования, строки могут содержать служебные символы (таблица 4.1).

Строки делятся на две категории в зависимости от типа ограничителя – они могут ограничиваться парой кавычек (" ") или апострофов (' '). Между этими категориями существуют два принципиальных различия. Во-первых, имена переменных в строках, заключенных в кавычки, заменяются соответствующими значениями, а строки в апострофах интерпретируются буквально, даже если в них присутствуют имена переменных.

Таблица 4.1 – Служебные символы в PHP

Служебный символ	Назначение служебного символа
\n	Новая строка
\r	Возврат курсора
\t	Горизонтальная табуляция
\\	Обратная косая черта
\$	Знак доллара
\"	Кавычка
\[0-7]{1,3}	Восьмеричная запись числа (в виде регулярного выражения)
\x[0-9A-Fa-f]{l,2}	Шестнадцатеричная запись числа (в виде регулярного выражения)

Два следующих объявления дают одинаковый результат:

```
$food = "meatloaf";
$food = 'meatloaf';
```

Однако результаты следующих объявлений сильно различаются:

```
$sentence = "My favorite food is $food";
$sentence2 = 'My favorite food is $food';
```

Переменной $sentence присваивается строка My favorite food is meatloaf. Переменная $food автоматически интерпретируется. Переменной $sentence2 присваивается строка My favorite food is $food. В отличие от переменной $sentence, в $sentence2 осталась неинтерпретированная переменная $food. Различия обусловлены использованием кавычек и апострофов при присваивании строки переменным $sentence и $sentence2.

Второе принципиальное различие заключается в том, что в строках, заключенных в кавычки, распознаются все существующие служебные символы, а в строках, заключенных в апострофы, – только служебные символы «\\» и «\». Следующий пример наглядно демонстрирует различия между присваиванием строк, заключенных в кавычки и апострофы:

```
$double_list = "item1\nitem2\nitem2";
$single_list = 'item1\nitem2\nitem2';
```

Если вывести обе строки в браузере, окажется, что строка в кавычках содержит внутренние символы новой строки, а в строке в апострофах последовательность \n выводится как обычные символы.

К отдельным символам строки можно обращаться как к элементам массива с последовательной нумерацией (пример 4.3).

Пример 4.3

```
$sequence_number = "04efgh";
$letter = $sequence_number[4];
```

В примере 4.3 переменной $letter будет присвоено значение g, т. к. нумерация элементов массивов начинается с 0.

Массив представляет собой список однотипных элементов. Существует два типа массивов, различающиеся по способу идентификации элементов. В массивах первого типа элемент определяется индексом в последовательности. Массивы второго типа имеют ассоциативную природу, и для обращения к элементам используются ключи, логически связанные со значениями.

Объект представляет собой переменную, экземпляр которой создается по специальному шаблону, называемому классом. Концепции объектов и классов являются неотъемлемой частью парадигмы объектно-ориентированного программирования (ООП).

Логический тип данных принимает всего два значения: истинное (true) и ложное (false). Логические величины создаются двумя способами: при проверке условий и в виде значений переменных.

4.1.5 Переменные в PHP

Переменная – это область оперативной памяти, доступ к которой осуществляется по имени. Все данные, с которыми работает программа, хранятся в виде переменных.

Правила задания переменных в PHP:

– имя переменной должно начинаться со знака доллара $;

– имя переменной не должно содержать никаких других символов, кроме символов латинского алфавита, цифр и знака подчеркивания;

– имена переменных в PHP, как и в C, чувствительны к регистру символов, т. е. переменные $a и $A – это совершенно разные переменные;

– объявлять переменную можно в любом месте программы, но до места первого ее использования. При объявлении переменных тип не указывается. Выбор типа осуществляется самим интерпретатором.

Переменные в PHP могут содержать любую информацию. Исключение составляют только константы, которые могут содержать только число или строку.

4.1.5.1 Функции определения и задания типа переменных

Язык PHP предоставляет много средств для определения типа переменных. Вы можете использовать семь функций для определения типа:

– **is_int($x)** или **is_integer($x)** – возвращает true, если переданная переменная – целое число;

– **is_double($x)** или **is_float($x)** – возвращает true, если переданная переменная – вещественное число;

– **is_string($x)** – возвращает true, если переданная переменная – строка;

– **is_array($x)** – возвращает true, если переданная переменная – массив;

– **is_object($x)** – возвращает true, если переменная является объектом;

– **is_bool($x)** – возвращает true, если переменная объявлена как логическая;

– **gettype(Sx)** – возвращает строки, соответствующие типу переменной: integer, double, string, object, array, bool или unknown type – если невозможно определить тип (когда тип переменной не встраивается в PHP, а добавляется с помощью модулей, расширяющих возможности языка).

Если PHP неправильно определил тип переменной, можно указать его явно. Для этого используется функция **settype($x, $type)**, где **$type** – это одна из строк, возвращаемых функцией gettype() (пример 4.4). Функция **settype($x, $type)** возвращает значение false, если невозможно привести тип переменной **$x** к указанному типу **$type**.

Пример 4.4

```
settype($x,"double");
```

4.1.5.2 Присвоение значений. Оператор присваивания

Оператор присваивания позволяет придать переменной некоторое значение (пример 4.5).

Пример 4.5

```
$x = 4;
$y = $x;
$x=$x+4;
```

Особенности оператора присваивания:

1 Переменной можно присвоить: какое-либо значение; значение, возвращаемое функцией; значение другой переменной; значение выражения; ссылку на другую переменную.

2 В PHP нет указателей, поэтому если в переменную $x поместить файл размером 500 Кбайт, а потом присвоить ее переменной $y, то будет создана точная копия переменной $x, которая тоже будет занимать 500 Кбайт. Итого в памяти будет почти 1 Мбайт информации. Это нужно учитывать при создании так называемых временных переменных. Желательно после окончания работы с такой переменной освободить память, т. е. уничтожить переменную.

3 Интерпретатор сам выполняет преобразование типов, но иногда привести тип переменной к другому просто невозможно.

4 При присваивании создается точная копия переменной, копируется также и тип переменной. Это означает, что если массиву присвоить число, весь массив будет потерян.

4.1.5.3 Проверка существования переменной

Проверка существования переменной – это очень удобная возможность языка программирования. Благодаря возможности проверки существования переменной можно проверить, передан сценарию определенный параметр или нет, и только потом начинать с ним работать. В результате сценарий не прекратит свою работу из-за банальной ошибки пользователя. Проверка существования переменной осуществляется с помощью функции **isset($x)** (пример 4.6).

Пример 4.6

```
<?
  if (isset($name))
       print "Hello, $name";
  else
       print " Вы забыли ввести свое имя";
?>
```

4.1.5.4 Удаление переменных

Чтобы не засорять память, можно удалить ненужные нам переменные. Это можно делать с помощью функции **unset()**. Эта функция удаляет указанную в скобках переменную из памяти, и программа продолжает выполняться как-будто эта переменная вообще не была инициализирована (пример 4.7).

Пример 4.7

```
<?
  $a=читаем_большой_файл;
  //обрабатываем_файл;
  unset($a); // освобождаем память
?>
```

Эта функция особенно полезна и даже необходима при обработке больших объемов данных, чтобы они не оставались в памяти компьютера и не замедляли его работу.

4.1.5.5 Логические переменные и их особенности в PHP

В PHP истиной являются: любое не равное нулю число, любая непустая строка, значение true. Значение false, пустая строка, нуль – это ложь.

В использовании логических переменных в PHP имеется еще одна особенность. Если в операторах сравнения (= =, !=, <,>) один тип является логическим, то и второй также воспринимается как логический (примеры 4.8 и 4.9).

Пример 4.8

```
<?
   $x=10;
   if ($x= =1) echo "Переменная равна 1\n";
   if ($x= =true) echo "Переменная равна true\n";
```

```
?>
```

В примере 4.8 в первой строке переменной $x было присвоено значение 10. Затем $x сравнивается с 1, и если $x равно 1, то выводится строка «Переменная равна 1». Затем значение переменной $x сравнивается со значением true. И если $x равно true, то выводится строка «Переменная равна true». Исходя из приведенного выше утверждения, должна быть выведенной только вторая строка.

Пример 4.9

```
<?
  $x = 100;
  $y = true;
  echo "x= $x\n";
  echo "y= $y\n";
  if ($x==$y) echo "X=Y";
?>
```

В этом примере сначала программа сообщает, что X = 100, Y = 1, а затем, что X = Y.

4.1.6 Выражения и операции

Выражение – это последовательность знаков операций, операндов и круглых скобок, которая задает вычислительный процесс получения результата определенного типа.

Операции – это специальные комбинации символов, специфицирующих действия по преобразованию различных величин.

В PHP выделяют следующие виды операций:

1) арифметические;
2) строковые;
3) операции присваивания;
4) операции сравнения;
5) логические;
6) побитовые.

4.1.6.1 Арифметические операции

Как и в любом другом языке, в PHP можно использовать арифметические операции:

1) a + b – сложение;
2) a – b – вычитание;
3) a * b – умножение;
4) a / b – деление;
5) a % b – остаток от деления a на b.

Операция деления возвращает целое число (т. е. результат деления нацело), если оба выражения a и b – целого типа (или же строки, выглядящие, как

целые числа), в противном случае результат будет дробным. Операция вычисления остатка от деления % работает только с целыми числами.

4.1.6.2 Строковые операции

Строковых операций в PHP всего две:

1) a.b – слияние строк a и b (пример 4.10);
2) a[n] – символ строки в позиции n.

Пример 4.10

```
$a="100";
$b="200";
echo $a.$b; //выведет 100200
echo $a + $b; //выведет 300
```

Все остальные действия над строками выполняются стандартными функциями.

4.1.6.3 Операции присваивания

В операциях присваивания правое выражение присваивается переменной слева.

1) = – присваивание;
2) +=, -=, .= и т. д. – операции сложного присваивания (пример 4.11);
3) ++, -- – инкремент и декремент. Операции операторов инкремента и декремента не работают с логическими переменными.

Пример 4.11

```
$message="Hello ";
$message.= "World!"; // Теперь в $message "Hello World!"
```

4.1.6.4 Операции сравнения

Независимо от типов своих аргументов, они всегда возвращают одно из двух значений: false или true. Операции сравнения позволяют сравнивать два значения между собой и, если условие выполнено, возвращают true, в противном случае – false. Операции сравнения (пример 4.12):

1) a == b – истина, если a равно b;
2) a != b – истина, если a не равно b;
3) a < b – истина, если a меньше b;
4) a > b – истина, если a больше b;
5) a <= b – истина, если a меньше либо равно b;
6) a >= b – истина, если a больше либо равно b;
7) = = = – истина, если a эквивалентно b;
8) ! = = – истина, если a неэквивалентно b.

Пример 4.12

```
$zero=0; // нуль
$tsss=""; // пустая строка
```

if ($zero==$tsss) echo "Переменные равны";

if ($zero===$tsss) echo "Переменные эквивалентны";

/* в данном примере на экран будет выведена только первая строка, что переменные равны */

В PHP 5 сравнивать на равенство или неравенство можно не только скалярные переменные (т. е. строки и числа), но также массивы и объекты. При сравнении массива сравнивается отдельно каждая пара элементов.

4.1.6.5 Логические операции

Эти операции предназначены исключительно для работы с логическими выражениями и также возвращают false или true:

1) ! a – истина, если a ложно, и наоборот;

2) a && b – истина, если истинны и a, и b;

3) a | | b – истина, если истинны или a, или b, или оба операнда.

Вычисление логических выражений, содержащих такие операции, идет всегда слева направо, при этом если результат уже очевиден, то вычисления обрываются, даже если в выражении присутствуют вызовы функции.

4.1.6.6 Приоритет операций

Приоритет арифметических и логических операций представлен в таблице 4.2. Операции с более высоким приоритетом выполняются в первую очередь. Изменить приоритет операции можно с помощью круглых скобок.

Таблица 4.2 – Приоритет арифметических и логических операций

Приоритет	**Оператор**	**Порядок выполнения**
13	(постфикс)++ (постфикс)--	слева направо
12	(префикс)++ (префикс)--	справа налево
11	* / %	слева направо
10	+ -	
9	<< >>	
8	< <= > >=	
7	= = !=	
6	&	
5	^	
4	|	
3	&&	
2	||	
1	= += -= *= /= %= >>= <<= &= ^= |=	справа налево

4.1.7 Конструкции PHP (операторы выбора, операторы цикла)

4.1.7.1 Условный оператор (if … else)

Формат условного оператора:

```
if (логическое_выражение)
    инструкция_1;
else
    инструкция_2;
```

Если *логическое_выражение* истинно, то выполняется *инструкция_1*, а иначе – *инструкция_2*. Как и в любом другом языке, конструкция else может опускаться. Если *инструкция_1* или *инструкция_2* должны состоять из нескольких команд, то они, как всегда, заключаются в фигурные скобки.

В условном операторе *инструкция_1* и *инструкция_2*, в свою очередь, могут быть условными, что позволяет организовать цепочки проверок любой степени вложенности. Синтаксис языка предполагает, что при вложенных условных операторах каждое *else* соответствует ближайшему *if*.

Существует альтернативная форма конструкции if … else:

```
if (логическое_выражение):
    инструкция_1;
elseif (другое_логическое_выражение):
    инструкция_2;
else:
    инструкция_3;
endif
```

Обратите внимание на расположение двоеточия. Если его пропустить, то будет сгенерировано сообщение об ошибке. Блоки else и elseif можно опускать.

Пример 4.13

В форму вводятся два числа, и скрипт выводит введенные числа, определяет наибольшее из них и выводит их среднее арифметическое.

```
<HTML>
<HEAD>
  <TITLE> Определение наибольшего числа </TITLE>
</HEAD>
<BODY>
<?
if ($_SERVER["REQUEST_METHOD"] == "POST")
{
 echo "Число 1: ".$_POST["chislo1"]."<BR>";
 echo "Число 2: ".$_POST["chislo2"]."<BR>";
 if ($_POST["chislo1"]>$_POST["chislo2"])
   echo "Наибольшее число ".$_POST["chislo1"]."<BR>";
 elseif ($_POST["chislo1"]<$_POST["chislo2"])
   echo "Наибольшее число ".$_POST["chislo2"]."<BR>";
 else
   echo "Числа равны! <BR>";
 $sr=($_POST["chislo1"]+$_POST["chislo2"])/2;
 echo "Среднее арифметическое ".$sr."<BR>";
```

```
}
else {
?>
<H1> Форма для ввода чисел </H1>
<FORM METHOD= "POST" ACTION= "form.php" >
<TABLE>
<TR><TD>Введите первое число:</TD>
        <TD><INPUT NAME ="chislo1" TYPE="TEXT" > </TD>
</TR>
<TR><TD>Введите второе число:</TD>
        <TD><INPUT NAME ="chislo2" TYPE="TEXT" > </TD>
</TR>
<TR>
     <TD COLSPAN=2 >
        <INPUT TYPE="SUBMIT" VALUE="Отправить">
     </TD>
  </TR>
</TABLE>
</FORM>
<? }?>
</BODY>
</HTML>
```

4.1.7.2 Переключатель (switch)

Переключатель *switch* является более удобным средством для организации множественного выбора, чем оператор *if ... else*.

Синтаксис переключателя:

```
switch (expression)  // переключающее выражение
{
  case value1:  //  константное  выражение  1
    statements1;//  блок операторов
    break;
  case value2:  //  константное  выражение  2
    statements2;
    break;
  default:
    statements;
}
```

Управляющая структура *switch* передает управление тому из помеченных операторов *case*, для которого значение константного выражения совпадает со значением переключающего выражения. Сначала анализируется переключающее выражение *expression* и осуществляется переход к той ветви программы, для которой его значение совпадает с константным выражением. Далее следует выполнение оператора или группы операторов до тех пор, пока не встретится

ключевое слово *break*, которым обозначается выход из переключателя. Если же значение переключающего выражения не совпадает ни с одним из константных выражений, то выполняется переход к оператору, помеченному меткой *default*. В каждом переключателе может быть не более одной метки *default*. Ключевые слова *break* и *default* не являются обязательными и их можно опускать.

Пример 4.14

В форме пользователю, к примеру, нужно ввести ответ на вопрос «Продолжить работу программы?». Если пользователь наберет в строке ввода «yes», то работа будет продолжена, если «no», то завершена.

```
<?
switch($answer)
{    case "yes": echo ("Продолжаем работу! ");
  // далее следуют операторы, которые будут выполняться в этом случае
       break;
    case "no": echo ("Завершаем работу");
       break;
    default: echo ("Некорректный ввод");
       break;
} ?>
```

Существует альтернативная форма конструкции *switch*:

```
switch (expression):
  case value1:  statements1;  break;
  case value2:  statements2; break;
  default:  statements;
endswitch;
```

Задание 4.1. *Реализовать калькулятор. Два числа для вычислений вводятся в текстовое поле, с помощью переключателя выбирается необходимая операция (сложение, умножение, деление, вычитание и др.), после нажатия кнопки на экран выводится результат.*

4.1.7.3 Циклы

В PHP определены 4 разных оператора цикла:

- цикл с предусловием (while);
- цикл с постусловием (do … while);
- итерационный цикл (for);
- итерационный цикл (foreach).

Цикл с предусловием while

Синтаксис цикла с предусловием:

```
while (логическое_выражение)
      инструкция;
```

Принцип работы цикла с предусловием:

– вычисляется значение логического выражения;

– если значение истинно, выполняется тело цикла, в противном случае переходим на следующий за циклом оператор.

Альтернативный синтаксис цикла с предусловием:

```
while (логическое_выражение):
    инструкция;
endwhile;
```

Цикл с предусловием do ... while

Синтаксис цикла с постусловием:

```
do {
    команды;
} while (логическое_выражение);
```

Цикл с постусловием отличается от цикла с предусловием тем, что сначала выполняется тело цикла, а только потом проверяется условие. Таким образом, тело цикла хотя бы один раз, но будет обязательно выполнено.

Альтернативного синтаксиса для цикла *do ... while* нет.

Итерационный цикл for

Итерационный цикл (или цикл со счетчиком) используется для выполнения тела цикла определенное количество раз. Синтаксис итерационного цикла *for*:

```
for (инициал-щие_команды; условие_цикла; команды_после_прохода)
    тело_цикла;
```

Оператор *for* начинает свою работу с выполнения команд инициализации. Данные команды выполняются всего лишь один раз. После этого проверяется условие: если оно истинно, выполняется тело цикла. После того как будет выполнен последний оператор тела, выполняются «команды после перехода». Затем снова проверяется условие, в случае, если оно истинно, выполняется тело цикла и поститерационные команды и т. д.

Альтернативный синтаксис итерационного цикла:

```
for (инициал-щие_команды; условие_цикла; команды_после_прохода):
    тело_цикла;
endfor;
```

Итерационный цикл foreach

Синтаксис итерационного цикла *foreach*:

```
foreach (массив as $ключ=>$значение)
    тело_цикла;
```

Здесь *команды* циклически выполняются для каждого элемента массива, при этом очередная пара *ключ=>значение* оказывается в переменных *$ключ* и *$значение*.

Пример 4.15

Вывод всех переменных окружения

```
<? foreach ($_SERVER as $k=>$v)
      echo "<b>$k</b> => <tt>$v</tt><br>\n";
?>
```

У цикла *foreach* имеется и другая форма записи, которую следует применять, когда не интересует значение ключа очередного элемента. В этом случае доступно лишь значение очередного элемента массива, но не его ключ. Выглядит она следующим образом:

```
foreach ($массив as $значение)
      тело_цикла;
```

Задание 4.2. *Добавить в скрипт, созданный в предыдущем примере, и вместе с результатами вычислений вывести содержимое двух глобальных массивов $_POST (содержит данные, переданные из формы) и $_SERVER (содержит переменные окружения, которые установлены web-сервером), используя цикл foreach.*

4.2 ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

1 Разработать скрипт, который будет определять, превысил ли тот или иной покупатель универмага предельный размер кредита на своем расчетном счете. Для каждого покупателя доступна следующая информация (вводится в форму):

а) номер счета;

б) баланс на начало месяца;

в) общая сумма по всем статьям расхода для данного покупателя в этом месяце;

г) общая сумма всех кредитов, отнесенная на счет данного покупателя, в этом месяце;

д) допустимый предельный размер кредита.

После нажатия кнопки на форме на странице должны выводиться: вся введенная информация, новый баланс (= начальный баланс + расходы – кредит) и сообщение о том, превышает новый баланс предельный размер кредита покупателя или нет. Для тех покупателей, чей предельный размер кредита превышен, программа должна отображать на экране номер счета покупателя, предельный размер кредита, новый баланс и сообщение «Предельный размер кредита превышен».

Пример расчета:

Введите номер счета: 100

Введите начальный баланс: 5394.78

Введите общую сумму расходов: 1000.00

Введите общую сумму кредита: 500.00

Введите предельный размер кредита: 5500.00

Счет: 100

Предельный размер кредита: 5500.00
Баланс: 5894.78
Предельный размер кредита превышен.

2 Простые проценты по ссуде рассчитываются по формуле

interest = principal * rate * days / 365

В предыдущей формуле принимается, что **rate** является процентной ставкой за год и поэтому включает деление на 365 (дней). Необходимо разработать скрипт, который на **одной** странице обеспечивает ввод значений **principal, rate и days** и вывод результата вычислений простых процентов по ссуде, используя для этого предыдущую формулу.

Пример расчета:

Введите основную сумму ссуды: 1000.00
Введите процентную ставку: .1
Введите срок ссуды в днях: 365
Выплаты по простым процентам составляют $100.00

4.3 КОНТРОЛЬНЫЕ ВОПРОСЫ

1 Выберите правильные варианты имен переменных в PHP?

– name;
– $ht;
– $45;
– $va;
– 4_name.

2 Необходимо ли явное объявление имени и типа переменных в PHP перед их использованием?

– да;
– нет;
– зависит от версии браузера;
– зависит от версии сервера;
– зависит от настроек сервера и браузера.

3 Какие типы данных поддерживаются в PHP?

– integer;
– string;
– float;
– array;
– boolean.

4 Какая функция позволяет определить тип переменной?

– isset();
– gettype();
– settype();
– define();
– defined().

5 Какая функция позволяет изменить тип переменной?

– isset();

– gettype();

– settype();

– define();

– defined().

6 Какая функция позволяет проверить определена ли переменная?

– isset();

– gettype();

– settype();

– define();

– defined().

7 Что будет выведено на экран в результате работы данного скрипта: <? $f="meatloaf"; $s='My favorite food is $f'; print $s; ?>?

– My favorite food is;

– My favorite food is $f;

– My favorite food is meatloaf;

– meatloaf;

– нет верных ответов.

8 Скрипты PHP:

– скрипт интерпретируется на сервере, и клиенту передается готовая страница;

– сервер передает PHP-скрипт, интерпретирующийся браузером клиента;

– браузер передает PHP-скрипт, интерпретирующийся сервером;

– скрипт интерпретируется в браузере клиента и серверу передается готовая страница;

– нет правильных вариантов.

9 Выберите правильные варианты перехода в PHP:

– <% … %>;

– <? … ?>;

– <?br … ?>;

– <?php … ?>;

– <script language = "php"> … </script>.

10 Выберите правильные варианты написания комментариев в PHP:

– #это комментарии;

– &это комментарии;

– /* а это многострочные комментарии */;

– //это комментарии;

–
 это комментарии</br>.

11 Какие утверждения справедливы для переменных?

– имя переменной должно начинаться со знака доллара $;

– имя переменной при ее объявлении указывается со знаком доллара $, а в дальнейшем можно использовать без знака $;

– имя переменной не должно содержать никаких других символов, кроме

символов латинского алфавита и цифр;

– имя переменной не должно содержать никаких других символов, кроме символов латинского алфавита, цифр и знака подчеркивания;

– имена переменных не чувствительны к регистру;

– имена переменных чувствительны к регистру.

12 Что будет выведено на экран в результате работы данного скрипта: <? $x=10; if ($x= =1) echo " Переменная равна 1."; if ($x= =true) echo "Переменная равна true."; ?>?

– переменная равна true;

– переменная равна 1;

– переменная равна 1. Переменная равна true;

– переменная равна true. Переменная равна 1;

– в результате выполнения фрагмента программы на экран ничего выведено не будет.

13 Какая функция позволяет удалить переменную?

– isset();

– gettype();

– settype();

– unset();

– defined().

14 Что будет выведено на экран в результате работы данного скрипта: <? $a="100"; $b="200"; echo $a.$b; echo $a – $b; ?>?

– 300;

– 100200;

– 100200 300;

– ничего на экран выведено не будет;

– нет верных ответов.

15 Что будет выведено на экран в результате работы данного скрипта: <? $message="Hello "; $message.= "World! "; print $message; ?>?

– Hello World!;

– Hello;

– World!;

– ничего на экран выведено не будет;

– нет верных ответов.

16 Что будет выведено на экран в результате работы данного скрипта: <? $zero=0; $tsss=""; if ($zero==$tsss) echo "Переменные равны"; if ($zero===$tsss) echo "Переменные эквивалентны"; ?>?

– переменные равны;

– переменные эквивалентны;

– переменные равны. Переменные эквивалентны;

– ничего на экран выведено не будет;

– нет верных ответов.

17 Какие циклы существуют в PHP?

– while;
– do…while;
– for;
– foreach (массив as $ключ=>$значение);
– foreach (массив as $значение).

ЛАБОРАТОРНАЯ РАБОТА №5
СОЗДАНИЕ БАЗЫ ДАННЫХ

Цель работы: изучить основы баз данных и научиться создавать базы данных в СУБД MySQL.

Ход выполнения лабораторной работы должен быть отражен в отчете. Отчет должен содержать титульный лист, номера заданий, коды программ, скриншот с результатом выполнения программы.

5.1 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

5.1.1 Варианты хранения информации

Задача длительного хранения информации очень часто встречается в программировании web-приложений: подсчет посетителей в счетчике, хранение сообщений в форуме, удаленное управление содержанием информации на сайте и т. д. Существует ***два*** способа хранения данных: ***в текстовых файлах*** *и в* ***базах данных.***

При написании web-приложений на PHP первый способ настолько же прост, насколько ограничен.

Базы данных:

– сами заботятся о безопасности информации и ее сортировке;

– позволяют извлекать и размещать информацию при помощи одной строки;

– выборка информации из базы данных происходит значительно быстрее, чем из файла;

– код с использованием базы данных получается более компактным, и отлаживать его гораздо легче.

5.1.2 Классификация баз данных

База данных – это файл или набор файлов, хранящий структурированную определенным образом информацию. Для обработки этой информации используются особые программы, называемые ***системами управления базами данных.***

Задача длительного хранения и обработки информации появилась практически сразу с появлением первых компьютеров. Для решения этой задачи в конце 1960-х гг. были разработаны специализированные программы, получившие название *систем управления базами данных* (СУБД). СУБД проделали длительный путь эволюции. В конце 1980-х гг. доминирующей стала *система управления реляционными базами данных* (СУРБД). Для того чтобы унифицировать работу с ними, был разработан *структурированный язык запросов* (SQL), который представляет собой язык управления именно реляционными базами данных.

Взаимодействие с базой данных происходит при помощи СУБД, которая расшифровывает запросы и производит операции с информацией в базе данных. Поэтому более правильно было бы говорить о запросе к СУБД и о взаимодействии с СУБД из web-приложения. Но так как это несколько усложняет восприятие, далее мы будем использовать термин «базы данных» (БД), подразумевая при этом СУБД.

Существуют следующие ***разновидности*** баз данных:

– иерархические;
– реляционные;
– объектно-ориентированные;
– гибридные.

Иерархическая БД основана на древовидной структуре хранения информации. В этом смысле иерархические базы данных очень напоминают файловую систему компьютера.

В ***реляционных*** БД информация собрана в таблицы, состоящие из столбцов и строк, на пересечении которых расположены ячейки. Запросы к таким базам данных возвращают таблицу, которая повторно может участвовать в следующем запросе. Данные в одних таблицах, как правило, связаны с данными других таблиц, откуда и произошло название «реляционные» (от англ. *relational* – родственный).

В ***объектно-ориентированных*** БД данные хранятся в виде объектов. С объектно-ориентированными базами данных удобно работать, применяя объектно-ориентированное программирование. Однако в настоящее время такие базы данных еще не достигли популярности реляционных, поскольку пока значительно уступают им в производительности.

Гибридные СУБД совмещают в себе возможности реляционных и объектно-ориентированных баз данных.

В web-приложениях, как правило, используются ***реляционные базы данных.***

5.1.3 Терминология реляционных баз данных

Кратко особенности реляционной базы данных можно описать следующим образом:

– данные хранятся в таблицах, состоящих из столбцов и строк;
– на пересечении каждого столбца и строки стоит в точности одно значение;
– у каждого столбца есть свое имя, которое служит его названием, и все значения в одном столбце имеют один тип;
– столбцы располагаются в определенном порядке, который определяется при создании таблицы, в отличие от строк, которые располагаются в произвольном порядке. В таблице может не быть ни одной строчки, но обязательно должен быть хотя бы один столбец.

Запросы к базе данных возвращают результат в виде таблиц, которые тоже могут выступать как объект запросов.

Таблицы в реляционных базах данных состоят из записей, а записи, в свою очередь, состоят из полей.

Поле (field) – это базовый элемент данных в БД. Поле имеет имя и тип.

Запись (record) – это набор полей, содержащих связанную информацию.

Таблица (table) – это набор записей одной структуры полей.

База данных (database) – это совокупность связанных таблиц.

Индекс (index) – это отсортированный список значений полей, предназначенный для ускорения поиска в базе данных. Обычно поиск производится по значению одного поля или по значению нескольких полей.

Уникальный индекс (index) – список значений поля, в котором каждое значение уникально, т. е. в одной таблице не должно быть два поля (по которому строился уникальный индекс) с одним и тем же значением.

Первичный ключ (primary key) представляет собой один из примеров уникальных индексов и применяется для уникальной идентификации записей таблицы. Никакие из двух записей таблицы не могут иметь одинаковые значения первичного ключа.

По способу задания первичных ключей различают ***логические*** (естественные) и ***суррогатные*** (искусственные) ключи. Для логического задания первичного ключа нужно выбрать в базе данных то, что естественным образом определяет запись (например, номер паспорта). Если подходящих примеров для естественного задания первичного ключа не находится, пользуются суррогатным ключом. Суррогатный ключ представляет собой дополнительное поле в базе данных, предназначенное для обеспечения записей первичным ключом.

Даже если в базе данных содержится естественный первичный ключ, лучше использовать суррогатные ключи, поскольку их применение позволяет абстрагировать первичный ключ от реальных данных. В этом случае облегчается работа с таблицами, поскольку суррогатные ключи не связаны ни с какими фактическими данными этой таблицы.

Запрос (query) – оператор, выбирающий записи и поля из одной или нескольких таблиц. При этом выбираемые записи и поля должны соответствовать условию, заданному оператором.

Схемой БД называется структура связей между полями и таблицами.

Нормализация схемы БД – это процедура, производимая над базой данных с целью удаления в ней избыточности. В нормализованной БД уменьшается вероятность возникновения ошибок, она занимает меньше места на жестком диске и т. д.

5.1.4 Настольные и сетевые реляционные СУБД

Как правило, любая СУБД состоит из двух частей. Первая часть – это та программа, с которой работает пользователь, – *клиент данных*. Вторая же часть непосредственно занимается базой данных: принимает от клиента запросы на выборку и изменение данных, выполняет их и возвращает клиенту. Это так называемый *процессор данных*. Можно сказать, что клиент данных занимается приемом запросов от пользователя и выводом результатов, а процессор – собственно обработкой данных.

И в зависимости от того, как реализованы клиент и процессор данных, СУБД делятся на две большие группы: ***настольные*** и ***клиент-серверные***.

Настольная СУБД реализована в виде одной-единственной программы; и клиент, и процессор данных слиты воедино в одном исполняемом файле. Это первое отличие. Второе отличие – настольная СУБД работает непосредственно с файлами баз данных точно так же, как Microsoft Word работает с файлами документов.

Серверная СУБД – это процессор данных, оформленный в виде отдельной программы и работающий на специально выделенном для этого серверном компьютере. Как и любой другой сервер, он принимает от клиентов запросы, считывает из файла базы данных, обрабатывает их и пересылает результаты обработки клиентам.

Преимущества настольных СУБД:

– исключительная легкость установки и использования;

– нетребовательность к дополнительному программному обеспечению.

Недостатки настольных СУБД:

– невысокое быстродействие при многопользовательском доступе к базе данных по сети;

– недостаточная надежность и защищенность.

Поэтому настольные СУБД используются для ведения персональных баз данных и для создания совсем небольших, как правило, несетевых систем обработки данных.

Преимущества серверных СУБД:

– большая производительность (поскольку по сети пересылаются только запросы и ответы, которые меньше по размерам, чем фрагменты файлов);

– большая надежность и защищенность.

Недостатки серверных СУБД: сложность установки, настройки и сопровождения.

Настольные СУБД – это Microsoft Access, Corel Paradox, Borland dBase, Microsoft FoxPro и др.

К серверным СУБД относятся Borland InterBase, MySQL, Microsoft SQL Server, PostgreSQL, Informix, Oracle, Sybase, IBM DB2 и многое другое.

5.1.5 Архитектура web-баз данных

Базовая архитектура web-баз данных включает в себя web-браузер, web-сервер, сценарный механизм и сервер баз данных (рисунок 5.1).

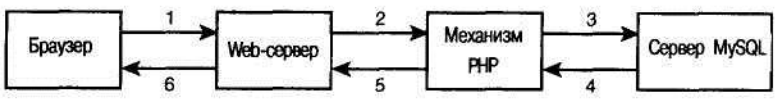

Рисунок 5.1 – Архитектура web-баз данных

Типичная транзакция web-базы данных состоит из этапов, обозначенных цифрами на рисунке 5.1.

1 Web-браузер пользователя отправляет HTTP-запрос определенной web-страницы. Например, поиск в магазине всех книг, написанных определенным автором, используя HTML-форму. Страница с результатами поиска называется results.php.

2 Web-сервер принимает запрос на results.php, получает файл и передает его механизму PHP на обработку.

3 Механизм PHP начинает синтаксический анализ сценария. В сценарии присутствует команда подключения к базе данных и выполнения запроса в ней (поиск книг). PHP открывает соединение с сервером MySQL и отправляет необходимый запрос.

4 Сервер MySQL принимает запрос в базу данных, обрабатывает его, а затем отправляет результаты (в данном случае – список книг) обратно в механизм PHP.

5 Механизм PHP завершает выполнение сценария, форматируя результаты запроса в виде HTML, после чего отправляет результаты в HTML-формате web-серверу.

6 Web-сервер пересылает HTML в браузер, с помощью которого пользователь просматривает список необходимых книг.

Процесс этот, как правило, протекает вне зависимости от того, какой сценарный механизм и какой сервер баз данных используется. Зачастую программное обеспечение web-сервера, механизм PHP и сервер баз данных находятся в одной машине. Правда, не менее часто сервер базы данных работает на другой машине. Это делается из соображений безопасности, увеличения объема или разделения потока. С точки зрения перспектив развития, в работе оба варианта одинаковы, однако в плане производительности второй вариант может оказаться более предпочтительным.

5.1.6 СУБД MySQL

Одна из самых популярных СУБД, которые используются в web-программировании, – MySQL. Она предназначена для создания сравнительно небольших (в районе 100 Мбайт) баз данных и поддерживает некоторое подмножество языка запросов SQL.

Достоинства MySQL:

– MySQL – весьма быстрый и нетребовательный к ресурсам компьютера сервер данных;

– возможностей MySQL более чем хватает для создания сравнительно небольших web-сайтов;

– MySQL распространяется бесплатно, более того – его исходные тексты открыты для изучения и доработки;

– MySQL прекрасно работает в связке с PHP, технологией создания активных серверных web-страниц;

MySQL – это программа-сервер, постоянно работающая на компьютере. Клиентские программы (например, сценарии) посылают ей специальные запросы через механизм сокетов (т. е. при помощи сетевых средств), она их обрабатывает и запоминает результат. Затем также по специальному запросу клиента весь этот результат или его часть передается обратно.

Один сервер MySQL может поддерживать сразу несколько баз данных, доступ к которым может разграничиваться именем пользователя и паролем.

5.1.7 Типы данных, используемые в базе данных MySQL

В MySQL определены три базовых типа столбцов: числовой, дата и время и строковый. Каждая из этих категорий подразделяется на множество типов. Каждый из типов обладает различной емкостью. Выбирая тип столбца, главное – выбрать наименьший, в котором помещаются данные.

5.1.7.1 Целые числа

Общий вид указания типа данных:

***префикс*INT [(M)] [UNSIGNED]**

Необязательный флаг UNSIGNED указывает, что будет создано поле для хранения беззнаковых чисел (больших или равных 0). При объявлении целого типа можно указать максимальную ширину отображения. В приведенных ниже таблицах для типов данных этот параметр обозначается как ***M***. Если для данного типа он не обязателен, то он приводится в квадратных скобках. Максимальным значением ***M*** является 255, минимальным – 1. Параметр ***M*** определяет только количество цифр, которое будет выводиться при запросе в клиентах типа MySQL, и не влияет на размер памяти, отводимой для хранения значения (таблица 5.1).

Таблица 5.1 – Целые типы данных

Название типа	Диапазон значений	Размер
TINYINT	от -128 до 127 (от -2^7 до 2^7-1), от 0 до 255 (от 0 до 2^8-1)	1 байт
SMALLINT	от -2^{15} до $2^{15}-1$, от 0 до $2^{16}-1$	2 байта
MEDIUMINT	от -2^{23} до $2^{23}-1$, от 0 до $2^{24}-1$	3 байта
INT	от -2^{23} до $2^{23}-1$, от 0 до $2^{32}-1$	4 байта
BIGINT	от -2^{63} до $2^{63}-1$, от 0 до $2^{64}-1$	8 байт

5.1.7.2 Дробные числа

Точно так же, как целые числа подразделяются в MySQL на несколько разновидностей, MySQL поддерживает и несколько типов дробных чисел (таблица 5.2). В общем виде они записываются так:

***ИмяТипа*[(length, decimals)] [UNSIGNED]**

где *length* – количество знакомест (ширина поля), в которых будет размещено дробное число при его передаче;

decimals – количество знаков после десятичной точки, которые будут учитываться;

UNSIGNED – задает беззнаковые числа.

Таблица 5.2 – Дробные типы данных

Название типа	Описание	Размер
FLOAT	Число с плавающей точкой небольшой точности	4 байта
DOUBLE	Число с плавающей точкой двойной точности	8 байт
REAL	Синоним для DOUBLE	8 байт
DECIMAL	Дробное число, хранящееся в виде строки	length + 2 байта
NUMERIC	Синоним для DECIMAL	length + 2 байта

5.1.7.3 Строки

Строки представляют собой массивы символов. Строковые типы разделяются на три группы.

Первая группа простые старые строки, которые представляют собой короткие фрагменты текста (таблица 5.3).

Таблица 5.3 – Строковые типы данных (первая группа)

Название типа	Описание	Размер
CHAR (M) [BINARY]	Символы с фиксированной длиной. От 1 до 255 символов	Объем памяти M байт. Максимальный размер M байт
VARCHAR(M) [BINARY]	Символы с произвольной длиной. От 1 до 255 символов	Объем памяти L+1 байт. Максимальный размер M байт

Если указан флаг BINARY, то при запросе SELECT строка будет сравниваться с учетом регистра.

Ширину каждого из них можно регулировать. Столбцы с типом **CHAR** будут дополняться пробелами до максимальной ширины, независимо от размеров данных, в то время как в столбцах с типом **VARCHAR** ширина зависит от размеров данных. MySQL усекает пробелы в конце текстовых строк у **CHAR** во время извлечения и у **VARCHAR** во время сохранения.

Вторая группа – это типы **TEXT** и **BLOB** (таблица 5.4). Их размеры могут быть разными. Первый тип данных предназначен для более длинных текстовых фрагментов, второй – для двоичных. Поля типа BLOB (Binary Large Object – большой двоичный объект) могут содержать любые данные, в том числе звуки и изображения. На практике столбцы **TEXT** и **BLOB** одинаковы, за исключением того, что **TEXT** чувствителен к регистру, а **BLOB** – нет.

Таблица 5.4 – Строковые типы данных (вторая группа)

Название типа	Описание
TINYTEXT	Может хранить не более 255 символов
TINYBLOB	Может хранить не более 255 символов
TEXT	Может хранить не более 65 535 символов
BLOB	Может хранить не более 65 535 символов
MEDIUMTEXT	Может хранить не более 16 777 215 символов
MEDIUMBLOB	Может хранить не более 16 777 215 символов
LONGTEXT	Может хранить не более 4 294 967 295 символов
LONGBLOB	Может хранить не более 4 294 967 295 символов

К *третьей группе* принадлежат два специальных типа **SET** и **ENUM**. Тип **SET** предназначен для того, чтобы определять, что значения в данном столбце принадлежат конкретному набору фиксированных значений. Значения столбца могут содержать несколько значений из набора. В определении набора можно задать вплоть до 64 элементов.

ENUM представляет собой перечисление. Этот тип очень похож на SET, но столбцы этого типа могут иметь всего лишь одно из фиксированных значений или NULL, а максимальное количество элементов в перечислении составляет 65 535.

ENUM(‘value1’, ‘value2’, …)
SET(‘value1’, ‘value2’, …)

5.1.7.4 Дата и время

MySQL поддерживает несколько типов полей, специально приспособленных для хранения дат и времени в различных форматах (таблица 5.5).

Таблица 5.5 – Типы данных даты и времени

Название типа	Описание	Размер
DATE	Дата в формате ГГГГ-ММ-ДД	3 байта
TIME	Время в формате ЧЧ:ММ:СС	3 байта
DATETIME	Дата и время в формате ГГГГ-ММ-ДД ЧЧ:ММ:СС	8 байт
TIMESTAMP [(M)]	Дата и время в формате timestamp. Метка времени, полезная для отчетов по транзакциям. Формат отображения зависит от значения M	4 байта
YEAR[(2|4)]	Год. Можно определить двух- или четырехциферный формат	–

5.1.8 Команды MySQL

Сервер MySQL поддерживает как инструкции SQL, так и служебные команды MySQL, предназначенные для администрирования и использования таблиц в базах данных MySQL.

Язык структурированных запросов SQL (Structure Query Language) позволяет выполнять различные операции с базами данных:

- создавать базы данных и таблицы;
- добавлять информацию в таблицы;
- удалять информацию;
- модифицировать информацию;
- получать нужные данные.

Команда **CREATE DATABASE** создает новую базу данных:

CREATE DATABASE [IF NOT EXISTS] *db_name*;

Здесь *db_name* является именем создаваемой базы данных. Не обязательная ключевая фраза IF NOT EXISTS сообщает, что базу данных следует создавать, только если база данных с таким именем отсутствует, что позволяет предотвратить завершение запроса ошибкой. Особенно это актуально при использовании SQL в PHP-скриптах.

Пример 5.1

Создание базы данных forum:

```
CREATE DATABASE db_test;
```

Для **удаления** базы данных используется команда **DROP DATABASE**. Синтаксис:

```
DROP DATABASE database_name;
```

где *database_name* – задает имя базы данных, которую необходимо удалить.

Пример 5.2

Удаление базы данных db_test:

```
DROP DATABASE db_test;
```

Задание 5.1:

5.1.1 Запустить web-сервер Apache.

5.1.2 Запустить браузер и в адресной строке ввести следующий адрес: http://localhost/Tools/phpMyAdmin/.

phpMyAdmin *– web-приложение с открытым кодом, написанное на языке PHP и представляющее собой web-интерфейс для администрирования СУБД MySQL. phpMyAdmin позволяет через браузер осуществлять администрирование сервера MySQL, запускать команды SQL и просматривать содержимое таблиц и баз данных.*

5.1.3 Создать базу данных, которая хранит сообщения форума. Структура форума может быть следующей. Имеется список разделов, переход по которым приводит посетителя к списку тем раздела. При переходе по теме посетитель приходит к обсуждению этой темы, состоящему из сообщений других посетителей.

Для создания базы данных в phpMyAdmin необходимо ввести имя базы данных, выбрать кодировку и нажать кнопку создать (рисунок 5.2).

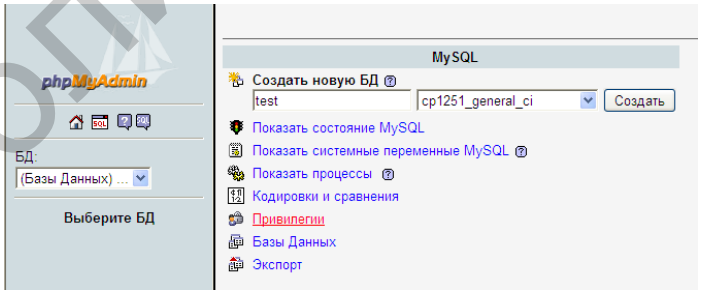

Рисунок 5.2 – Создание базы данных в phpMyAdmin

Созданная база данных отобразится в списке (рисунок 5.3).

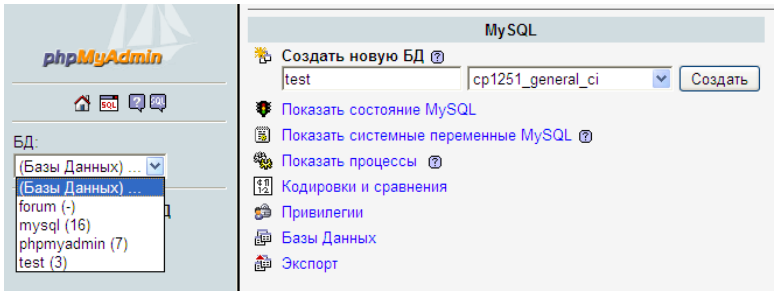

Рисунок 5.3 – Просмотр списка созданных баз данных в phpMyAdmin

5.1.4 Сделать созданную базу данных активной, выбрав ее имя в списке баз данных. Перейти на закладку SQL. Все дальнейшие задания будут выполняться путем ввода SQL команд (рисунок 5.4).

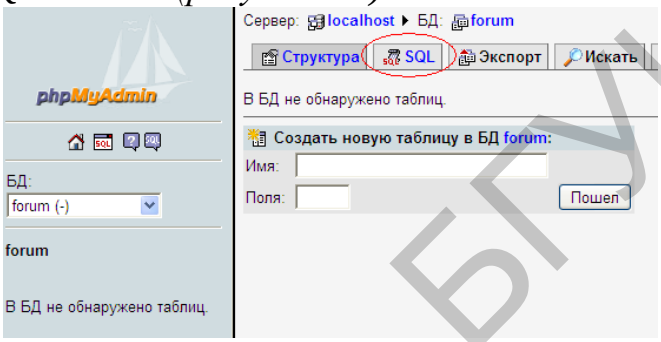

Рисунок 5.4 – Выбор меню для ввода SQL-запросов в phpMyAdmin

Команда **CREATE TABLE** создает новую таблицу в выбранной базе данных. В простейшем случае команда имеет следующий синтаксис:

CREATE TABLE *table_name*(*column_name1 type, column_name2 type*,...)
[table_options];

Здесь *table_name* – имя новой таблицы; *column_name* – имена колонок (полей), которые будут присутствовать в создаваемой таблице; *type* – определяет тип создаваемой колонки; *table_options* – необязательное указание типа таблицы, например TYPE = MyISAM.

Пример 5.3

Создание таблицы телефонных номеров друзей, которая будет состоять из трех столбцов: ФИО друга, адрес и телефон:

CREATE TABLE tel_numb(fio text, address text, tel text);

К типу данных можно присоединить модификаторы, которые задают его свойства и те операции, которые можно (или, наоборот, запрещено) выполнять с соответствующими столбцами.

Not null – означает, что поле не может содержать неопределенное значение, т. е. поле обязательно должно быть инициализировано при вставке новой записи в таблицу (если не задано значение по умолчанию).

Пример 5.4

Для таблицы с телефонами нужно указать, что поле с ФИО друга (поле fio) и его телефоном (поле tel) не может иметь неопределенное значение:

```
CREATE TABLE tel_numb(fio text NOT NULL, address text, tel text NOT NULL);
```

Primary key – отражает, что поле является первичным ключом, т. е. идентификатором записи, на который можно ссылаться.

Пример 5.5

```
CREATE TABLE tel_numb(fio text, address text, tel text, PRIMARY KEY (fio));
```

Auto_increment – при вставке новой записи поле получит уникальное значение, так что в таблице никогда не будут существовать два поля с одинаковыми номерами.

Пример 5.6

```
CREATE TABLE tel_numb(id int AUTO_INCREMENT, fio text, tel text);
```

Default – задает значение по умолчанию для поля, которое будет использовано, если при вставке записи для этого поля не было явно указано значение.

Пример 5.7

```
CREATE TABLE tel_numb(fio text, address text DEFAULT 'Не указан', tel text)
```

Задание 5.2:

5.2.1 Создать первую таблицу базы данных forum, которая называется authors и содержит различные данные о зарегистрированных посетителях форума: имя (name), пароль (passw), e-mail (email), web-адрес сайта посетителя (url), номер ICQ (icq), сведения о посетителе (about), строку, содержащую путь к файлу фотографии посетителя (photo), время добавления запроса (time), последнее время посещения форума (lasttime), статус посетителя – является ли он модератором, администратором или обычным посетителем (statususer). Кроме перечисленных полей в таблице имеется поле id_author, являющееся первичным ключом таблицы.

```
CREATE TABLE authors (
    id_author INT(6) NOT NULL AUTO_INCREMENT,
    name TINYTEXT,
    passw TINYTEXT,
    email TINYTEXT,
    url TINYTEXT,
    icq TINYTEXT,
    about TINYTEXT,
    photo TINYTEXT,
    time DATETIME DEFAULT NULL,
```

```
    last_time DATETIME DEFAULT NULL,
    themes INT(10) DEFAULT NULL,
    statususer INT(2) DEFAULT NULL,
    PRIMARY KEY (id_author)
) TYPE=MyISAM;
```

5.2.2 Создать вторую таблицу базы данных forum, которая содержит данные о разделе форума и называется forums. В таблице forums присутствуют следующие поля: первичный ключ (idforum), название раздела (name), правила форума (rule), краткое описание форума (logo), порядковый номер (pos), флаг, принимающий значение 1, если форум скрытый, и 0, если общедоступный (hide).

```
CREATE TABLE forums (
    id_forum INT(6) NOT NULL AUTO_INCREMENT,
    name TINYTEXT,
    rule TEXT,
    logo TEXT,
    pos INT(6) DEFAULT NULL,
    hide TINYINT(1) DEFAULT NULL,
    PRIMARY KEY (id_forum)
) TYPE=MyISAM;
```

5.2.3 Создать таблицу themes, содержащую список тем. В таблице присутствуют следующие поля: первичный ключ (id_theme); название темы (name); автор темы (author); внешний ключ к таблице authors (id_author); флаг, принимающий значение 1, если тема скрытая, и 0, если тема отображается (hide), – это поле необходимо для моделирования; время добавления темы (time); внешний ключ к таблице форумов (id_forum), чтобы определить, к какому разделу форума относится данная тема.

В таблице themes нормализация проведена частично, она содержит два внешних ключа: id_author и id_forum – для таблиц посетителей и списка форумов, в то же время в ней дублируется имя автора author, присутствующее в таблице посетителей authors под именем name. Этот случай служит примером денормализации, предназначенной для того, чтобы не запрашивать каждый раз имена из таблицы authors при выводе списка тем и их авторов и обеспечить приемлемую скорость работы форума.

```
CREATE TABLE themes (
    id_theme INT(11) NOT NULL AUTO_INCREMENT,
    name TEXT,
    author TEXT,
    id_author INT(6) DEFAULT NULL,
    hide TINYINT(1) DEFAULT NULL,
    time DATETIME DEFAULT NULL,
    id_forum TINYINT(2) DEFAULT NULL,
    PRIMARY KEY (id_theme)
) TYPE=MyISAM;
```

5.2.4 Создать таблицу posts, в которой хранятся сообщения посетителей. В таблице присутствуют следующие поля: первичный ключ (id_post); тело сообщения (name); необязательная ссылка на ресурс, которую автор сообщения может ввести при добавлении сообщения (url); путь к файлу, прикрепляемому к сообщению (file); имя автора (author); внешний ключ к таблице authors (id_author); флаг (hide), принимающий значение 1, если сообщение отмечено как скрытое, и 0, если оно отображается, – необходим для моделирования; время добавления сообщения (time); сообщение, ответом на которое является данное сообщение (parent_post), это поле равно 0, если данное сообщение – первое по этой теме; внешний ключ к таблице тем (id_theme), для того чтобы определить, к какой теме относится сообщение.

```
CREATE TABLE posts (
    id_post INT(11) NOT NULL AUTO_INCREMENT,
    name TEXT,
    url TINYTEXT,
    file TINYTEXT,
    author TINYTEXT,
    id_author INT(6) DEFAULT NULL,
    hide TINYINT(1) DEFAULT NULL,
    time DATETIME DEFAULT NULL,
    parent_post INT(11) DEFAULT NULL,
    id_theme int(11) DEFAULT NULL,
    PRIMARY KEY (id_post)
) TYPE=MyISAM;
```

Команда **SHOW TABLES** предназначена для просмотра списка таблиц в текущей базе данных.

Задание 5.3. *Убедитесь, что все таблицы успешно созданы, выполнив команду SHOW TABLES.*

Команда **DESCRIBE** показывает структуру созданных таблиц и имеет следующий синтаксис:

```
DESCRIBE table_name;
```

где *table_name* – имя таблицы, структура которой запрашивается.

Команда DESCRIBE не входит в стандарт SQL и является внутренней командой СУБД MySQL.

Пример 5.8

Просмотр структуры таблицы tel_numb с выполнением команды:

```
DESCRIBE tel_numb;
```

Задание 5.4. *Проверьте правильность создания таблиц, посмотрев структуру каждой таблицы (выполнять каждую команду необходимо по отдельности):*

DESCRIBE authors;
DESCRIBE forums;
DESCRIBE posts;
DESCRIBE themes;

Переименование таблицы (ALTER TABLE RENAME). Переименование таблицы можно сделать при помощи следующей конструкции:

ALTER TABLE *table_name_old* RENAME *table_name_new;*

где *table_name_old* – старое имя таблицы, которое нам нужно переименовать; *table_name_new* – новое имя таблицы.

Пример 5.9

Переименование таблицы *search* в *search_en*:

ALTER TABLE search RENAME search_en;

Вставка столбцов (ALTER TABLE ADD). Вставку нового столбца можно осуществить при помощи следующей конструкции:

ALTER TABLE *table_name* ADD *field_name parameters;*

где *table_name* – имя таблицы, в которой будет вставлен новый столбец; *field_name* – имя вставляемого столбца; *parametrs* – параметры, описывающие вставляемый столбец. Обязательным параметром является указание типа данных.

Пример 5.10

Вставка в таблицу *my_friends* столбца под названием *adress_2*, который будет содержать текстовые значения:

ALTER TABLE my_friends ADD adress_2 TEXT;

По умолчанию новый столбец вставляется в конец таблицы. Если необходимо, чтобы столбец встал в начало таблицы, нужно после параметров вставляемого столбца написать ключевое слово **FIRST**:

ALTER TABLE my_friends ADD adress_2 TEXT FIRST;

Если необходимо, чтобы столбец был вставлен не в начале таблицы и не в конце, а после определенного столбца, то нужно применить ключевое слово **AFTER** *имя столбца*, после которого будет установлен новый столбец:

ALTER TABLE my_friends ADD adress_2 TEXT AFTER adress_1;

В этом примере новый столбец *adress_2* будет установлен после столбца *adress_1*.

Если нужно дописать к таблице не один, а несколько столбцов, то для каждого столбца нужно **ADD** *field_name parametrs* записать через запятую:

ALTER TABLE my_friends ADD adress_2 TEXT,
ADD adress_3 TEXT, ADD adress_4 TEXT;

В случае если надо дописать два столбца внутри таблицы, можно поступить следующим образом:

ALTER TABLE my_friends ADD adress_2 TEXT AFTER adress_1,
ADD adress_3 TEXT AFTER adress_2;

т. е. первый вставляемый столбец записываем после *adress_1*, а второй – после первого.

Изменение свойств столбца (ALTER TABLE CHANGE). Изменить свойства одного или нескольких столбцов можно при помощи следующей конструкции:

ALTER TABLE *table_name* CHANGE *field_name_old field_name_new parametrs;*

где *table_name* – имя таблицы, в которой находится изменяемый столбец; *field_name_old* – имя столбца изменяемого столбца; *field_name_new* – новое имя изменяемого столбца (должно равняться *field_name_old*, если мы не хотим поменять имя столбца); *parametrs* – новые параметры столбца.

Пример 5.11

Установление типа строки field_1 как текст:

ALTER TABLE my_table CHANGE field_1 field_1 TEXT;

А если необходимо при этом еще и переименовать столбец в field_2, то получится так:

ALTER TABLE my_table CHANGE field_1 field_2 TEXT;

В случае, если надо изменить свойства сразу нескольких столбцов, то конструкцию **CHANGE** *field_name_old field_name_new parametrs* повторяем через запятую для каждого столбца:

ALTER TABLE my_table CHANGE field_1 field_2 TEXT,
CHANGE field_3 field_3 TEXT";

Удаление столбцов (ALTER TABLE DROP). Удаление столбца можно сделать при помощи следующей конструкции:

ALTER TABLE table_name DROP field_name

где *table_name* – имя таблицы, в которой будет удален столбец; *field_name* – имя удаляемого столбца.

Пример 5.12

ALTER TABLE search DROP id_num;

Если мы хотим удалить сразу несколько полей, то надо через запятую повторить **DROP** *field_name* для каждого столбца:

ALTER TABLE search DROP id_1, DROP id_2, DROP id_3;

Задание 5.5:

5.5.1 Добавить в таблицу forums новый столбец test, разместив его после столбца name.

ALTER TABLE forums ADD test INT(10) AFTER name;

5.5.2 Посмотреть структуру измененной таблицы forums.

DESCRIBE forums;

5.5.3 Переименовать созданный столбец test в текстовый столбец new_test.

ALTER TABLE forums CHANGE test new_test TEXT;

5.5.4 Посмотреть структуру измененной таблицы forums.

DESCRIBE forums;

5.5.5 Изменить тип столбца new_test на целочисленный. Установить, что значение этого столбца не может быть пустым. При изменении только типа столбца, а не его имени, указание имени необходимо, хотя в этом случае оно будет фактически повторяться.

ALTER TABLE forums CHANGE new_test new_test INT(5) NOT NULL;

5.5.6 Посмотреть структуру измененной таблицы forums.

DESCRIBE forums;

5.5.7 Удалить созданный столбец new_test. Посмотреть структуру измененной таблицы.

ALTER TABLE forums DROP new_test;

5.5.8 Посмотреть структуру измененной таблицы forums.

DESCRIBE forums;

Команда **INSERT INTO .. VALUES** вставляет новые записи в существующую таблицу. Синтаксис команды:

INSERT INTO table_name(field_name1, field_name2,...) values('content1', 'content2',...)

Данная команда добавляет в таблицу *table_name* запись, у которой поля, обозначенные как *field_nameN*, установлены в значение *contentN*.

Пример 5.13

Добавляем записи в таблицу адресов и телефонов (ФИО, адрес, телефон):

INSERT INTO tel_numb(fio, address, tel)
values('Вася Пупкин', 'ул.Горького, д.18', '23-23-23')

Те поля, которые не были перечислены в команде вставки, получают «неопределенные» значения (неопределенное значение – это не пустая строка, а просто признак, который «говорит» MySQL, что у данного поля нет никакого значения).

Если при создании таблицы поле было отмечено флажком NOT NULL и оно при вставке записи получило неопределенное значение, то MySQL возвратит ошибку.

Команда **DELETE** предназначена для удаления записей из таблицы:

DELETE FROM *table_name* WHERE (*выражение*);

Данная команда удаляет из таблицы *table_name* все записи, для которых выполнено *выражение*. *Выражение* – это просто логическое выражение.

Пример 5.14

Удаление записи из таблицы, содержащей ФИО, адрес и телефон:

DELETE FROM tel_numb WHERE (fio='Вася Пупкин');

или удаление по нескольким параметрам

DELETE FROM tel_numb WHERE (fio='Вася Пупкин' AND tel='23-45-45');

В выражении помимо имен полей, констант и операторов могут также встречаться простейшие вычисляемые части, например (id<10+4*5).

Обновление записи осуществляется командой **UPDATE**:

UPDATE *table_name* SET *field_name1='var1', field_name2='var2',...* [WHERE (*выражение*)] [LIMIT *rows*)];

Данная команда для всех записей в таблице *table_name*, удовлетворяющих выражению *выражение*, устанавливает указанные поля *field_nameN* в значении *varN*. В выражении WHERE, если оно присутствует, задается, какие строки подлежат обновлению. В остальных случаях обновляются все строки. Ключевое слово LIMIT позволяет ограничить число обновляемых строк.

Эту команду удобно применять, если не требуется обновлять не все поля какой-то записи, а нужно обновить только некоторые.

Пример 5.15

Если необходимо, для записей таблицы, у которых поле C_NO = 1 – это код клиента Иванова, установить значение CITY равным "Псков"

UPDATE CLIENTS SET CITY = 'Псков' WHERE C_NO = 1;

Выборка данных. Оператор **SELECT** является краеугольным камнем всего языка SQL. Он используется, чтобы выполнить запросы к базе данных. Это действительно основа языка SQL. Синтаксис оператора:

```
SELECT [STRAIGHT_JOIN] [DISTINCT | ALL] select_expression,.
  [FROM tables... [WHERE where_definition] [GROUP BY column,...]
  [ORDER BY column [ASC | DESC], ...]
```

Как видно из вышеприведенного, вместе с командой SELECT используются ключевые слова, использование которых очень влияет на ответ сервера. Рассмотрим каждое из них.

DISTINCT. Пропускает строки, в которых все выбранные поля идентичны, т. е. устраняет дублирование данных.

WHERE. Предложение команды SELECT, которое позволяет устанавливать предикаты, условие которых может быть верным или неверным для любой строки таблицы. Извлекаются только те строки, для которых такое утверждение верно.

Пример 5.16

Запрос выводит колонки u_id и lname из таблицы publishers, для которых значение в столбце city – New York. Это дает возможность сделать запрос более конкретным.

SELECT u_id, lname FROM publishers WHERE city ='New York';

IN. Оператор IN определяет набор значений, в котором данное значение может или не может быть включено.

Пример 5.17

Например, запрос

```
SELECT * FROM Salespeople WHERE city = 'Barcelona' OR city = 'London';
```

может быть переписан более просто:

```
SELECT * FROM Salespeople WHERE city IN ( 'Barcelona', 'London' );
```

IN определяет набор значений с помощью имен членов набора, заключенных в круглые скобки и отделенных запятыми. Затем он проверяет различные значения указанного, пытаясь найти совпадение со значениями из набора. Если это случается, то предикат верен. Когда набор содержит значения номеров, а не символов, одиночные кавычки опускаются.

Символ * в операторе SELECT предписывает, что из отобранных записей следует извлечь *все* поля, когда будет выполнена команда получения выборки. С другой стороны, вместо звездочки можно через запятую непосредственно перечислить имена полей, которые требуют извлечения.

BETWEEN. Оператор BETWEEN похож на оператор IN. В отличие от определения по номерам из набора, как это делает IN, BETWEEN определяет диапазон, значения которого должны уменьшаться, что делает предикат верным. Вы должны ввести ключевое слово BETWEEN с начальным значением, ключевое AND и конечное значение. В отличие от IN, BETWEEN чувствителен к порядку, и первое значение в предложении должно быть первым по алфавитному или числовому порядку.

Пример 5.18

```
SELECT * FROM Salespeople WHERE comm BETWEEN .10 AND .12;
SELECT * FROM Salespeople WHERE city BETWEEN 'Berlin' AND 'London';
```

LIKE. LIKE применим только к полям типа CHAR или VARCHAR, с которыми он используется чтобы находить подстроки. То есть он ищет поле символа, чтобы видеть, совпадает ли с условием часть его строки. В качестве условия он использует групповые символы (wildkards) – специальные символы, которые могут соответствовать чему-нибудь. Имеются два типа групповых символов, используемых с LIKE:

- символ подчеркивания (_) замещает любой одиночный символ;
- знак '%', замещающий любое количество символов.

Пример 5.19

Если заданы следующие условия:

```
SELECT * FROM Customers WHERE fname LIKE 'J%';
```

то будут выбраны все заказчики, чьи имена начинаются на J: John, Jerry, James и т. д.

COUNT. Агрегатная функция, производит подсчет значений в столбце или числа строк в таблице. При работе со столбцом использует DISTINCT в качестве аргумента.

Пример 5.20

```
SELECT COUNT ( DISTINCT snum ) FROM Orders;
```

Пример 5.21

При подсчете строк имеет синтаксис:

```
SELECT COUNT (*) FROM Customers;
```

GROUP BY. Предложение GROUP BY позволяет определять подмножество значений в особом поле в терминах другого поля и применять функцию агрегата к подмножеству. Это дает возможность объединять поля и агрегатные функции в едином предложении SELECT.

Пример 5.22

Запрос позволяет найти наибольшую сумму приобретений, полученную каждым продавцом.

```
SELECT snum, MAX (amt) FROM Orders GROUP BY snum;
```

В предложении GROUP могут быть использованы следующие функции:

- AVG() – среднее для группы GROUP;
- SUM() – сумма элементов GROUP;
- COUNT() – число элементов в GROUP;
- MIN () – минимальный элемент в GROUP;
- MAX() – максимальный элемент в GROUP.

Здесь MIN() и MAX() могут принимать строку или число в качестве аргумента. Эти функции не могут использоваться в выражении, хотя их параметр может быть выражением.

ORDER BY. Эта команда упорядочивает вывод запроса согласно значениям в том или ином количестве выбранных столбцов. Многочисленные столбцы упорядочиваются один внутри другого так же, как с GROUP BY.

5.2 ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

Создать базу данных для книжного магазина BookBy. В базе данных необходимо хранить данные о клиентах, о продаваемых книгах и об особенностях заказов.

Исходя из этого, делаем вывод, что в базе данных необходимы, как минимум, три таблицы: Customers (Клиенты), Orders (Заказы) и Books (Книги). Схема приложения имеет следующий вид (*далее приведен пример записей, которые необходимо добавить в базу данных, что поможет определить тип каждого поля*):

Customers (CustomerlD, Name, Address, City)
Orders (OrderlD, CustomerlD, Amount, Date)
Books (ISBN, Author, Title, Price)
Order_ltems (OrderlD, ISBN, Quantity)
Book_Reviews (ISBN, Reviews)
Первичные ключи подчеркивают обычной линией.

Добавить в базу данных ***bookby*** по 2-3 записи в каждую таблицу, используя командную строку. Например:

```
INSERT INTO customers VALUES
    (NULL, "Иванов И.И.", "пр. Пушкина 32", "Минск"),
    (NULL, "Петров И.В.", "пр. Рокоссовского 10", "Минск"),
    (NULL, "Сидоров А.В.", "ул. Цветочная 18", "Пинск");
INSERT INTO orders VALUES
    (NULL, 3, 69.98, "02-Apr-2006"),
    (NULL, 1, 49.99, "15-Apr-2006"),
    (NULL, 2, 74.98, "19-Apr-2006"),
    (NULL, 3, 24.99, "01-May-2006");
INSERT INTO books VALUES
    ("0-672-31697-8", "Толстой", "Война и мир", 34.99),
    ("0-672-31745-1", "Булгаков", "Мастер и Маргарита", 24.99),
    ("0-672-31509-2", "Лермонтов", "Избранные произведения", 24.99),
    ("0-672-31769-9", "Пушкин", "Евгений Онегин", 49.99);
INSERT INTO order_items VALUES
    (1, "0-672-31697-8", 2),
    (2, "0-672-31769-9", 1),
    (3, "0-672-31769-9", 1),
    (3, "0-672-31509-2", 1),
    (4, "0-672-31745-1", 3);
INSERT INTO book_reviews VALUES
    ("0-672-31697-8", "Это замечательная книга, ее должен каждый прочитать.");
```

5.3 КОНТРОЛЬНЫЕ ВОПРОСЫ

1 В чем преимущество использования баз данных для хранения данных?

– базы данных сами заботятся о безопасности информации и ее сортировке;

– позволяют извлекать и размещать информацию при помощи одной строки;

– выборка информации из базы данных происходит значительно быстрее, чем из файла;

– код с использованием базы данных получается более компактным;

– обработка баз данных сводиться к последовательной обработке.

2 К какому типу СУБД относится MySQL?

– настольная реляционная СУБД;

– сетевая реляционная СУБД;

– сетевая иерархическая СУБД;

– настольная иерархическая СУБД;

– объектно-ориентированная реляционная СУБД.

3 Что такое клиент данных?

– часть СУБД, которая занимается приемом запросов от пользователей и выводом результатов;

– часть СУБД, с которой работает клиент;

– часть СУБД, которая выполняет только запросы на выборку и изменение;

– часть СУБД, которая непосредственно занимается взаимодействием с базой данных и собственно обработкой данных;

– часть СУБД, которая только выводит пользователю результаты запросов.

4 Что такое процессор данных?

– часть СУБД, которая занимается приемом запросов от пользователей и выводом результатов;

– часть СУБД, с которой работает клиент;

– часть СУБД, которая непосредственно занимается взаимодействием с базой данных и собственно обработкой данных;

– часть СУБД, которая выполняет только запросы на выборку и изменение;

– часть СУБД, которая только выводит пользователю результаты запросов.

5 Какие утверждения справедливы для серверных СУБД?

– нетребовательность к дополнительному программному обеспечению;

– невысокое быстродействие при многопользовательском доступе к базе данных по сети;

– большая производительность;

– недостаточная надежность и защищенность;

– сложность установки, настройки и сопровождения.

6 Какие утверждения справедливы для настольных СУБД?

– большая производительность;

– нетребовательность к дополнительному программному обеспечению;

– невысокое быстродействие при многопользовательском доступе к базе данных по сети;

– большая надежность и защищенность;

– сложность установки, настройки и сопровождения.

7 Какие утверждения справедливы для СУБД MySQL?

– MySQL – это программа-сервер;

– MySQL обрабатывает запросы и запоминает результат и по специальному запросу передает результат или его часть клиентским программам;

– MySQL – это клиентская программа;

– один сервер MySQL может поддерживать сразу несколько баз данных;

– один сервер MySQL может поддерживать только одну базу данных.

8 Какие типы данных поддерживает СУБД MySQL?

– ARRAY;
– BIGINT;
– DOUBLE;
– ENUM;
– BLOB.

9 Какие атрибуты могут быть заданы столбцу вместе с типом данных?

– AUTO_INCREMENT;
– NOT NULL;
– DEFAULT;
– PRIMARY KEY;
– TYPE.

10 Какая команда MySQL позволяет посмотреть список существующих баз данных?

– SHOW DATABASES;
– SHOW TABLES;
– SHOW db_name;
– SHOW DATABASES db_name;
– нет верных ответов.

11 Какая команда MySQL позволяет просмотреть список существующих в базе данных таблиц?

– SHOW DATABASES;
– SHOW TABLES;
– SHOW db_name;
– SHOW DATABASES db_name;
– нет верных ответов.

12 Какая команда MySQL выбирает активную базу данных?

– USE db_name;
– USE ;
– USE DATABASE db_name;
– USE TABLES db_name;
– нет верных ответов.

13 Какая команда MySQL предназначена для создания базы данных?

– CREATE DATABASE db_name;
– CREATE db_name;
– CREATE DATABASES;
– CREATE TABLES;
– нет верных ответов.

14 Какая команда MySQL предназначена для создания таблиц в базе данных?

– CREATE TABLE table_name (col_name type);
– CREATE TABLES;
– CREATE table_name (col_name type);

– CREATE TABLES (col_name type);

– нет верных ответов.

15 Выберите правильные варианты написания команды создания таблицы в MySQL:

– CREATE TABLE tel (fio text, adreess text, tel text);

– CREATE TABLE tel (fio text NOT NULL, adreess text, tel text);

– CREATE TABLE tel (fio text, adreess text, tel text, PRIMARY KEY(fio));

– CREATE TABLE tel (adreess text, tel text, PRIMARY KEY(fio));

– CREATE TABLE tel (fio, adreess, tel).

16 Какая команда MySQL предназначена для удаления баз данных?

– DROP DATABASE db_name;

– DROP DATABASES;

– DROP DATABASE;

– DROP DATABASES db_name;

– DROP db_name.

17 Какая команда MySQL предназначена для удаления таблицы из базы данных?

– DROP TABLE table_name;

– DROP TABLE;

– DROP TABLE;

– DROP TABLE table_name;

– DROP table_name.

18 Выберите правильные варианты написания команды добавления столбца в таблицу в MySQL:

– ALTER TABLE ADD adress AFTER;

– ALTER TABLE ADD adress text FIRST;

– ALTER TABLE my_friends ADD adress text ADD fio text;

– ALTER TABLE my_friends ADD adress text FIRST;

– ALTER TABLE my_friends ADD adress text AFTER fio.

19 Что делает команда ALTER TABLE table_name CHANGE … в MySQL?

– изменяет свойства одного или нескольких столбцов и при необходимости переименовывает их;

– изменяет свойства одного столбца;

– переименовывает столбец в таблице;

– изменяет свойства базы данных;

– изменяет и переименовывает свойства строки таблицы.

20 Что делает команда ALTER TABLE table_name DROP… в MySQL?

– изменяет свойство одного или нескольких столбцов и при необходимости переименовывает его;

– переименовывает один или несколько столбцов;

– изменяет свойства столбца;

– удаляет столбец;

– добавляет новую запись в таблицу.

21 С помощью какой команды MySQL можно добавить новую запись в таблицу?

– INSERT INTO;
– DELETE FROM;
– CREATE;
– ALTER TABLE ADD;
– ALTER TABLE CHANGE.

22 Что делает команда DELETE FROM … в MySQL?

– удаляет все записи, для которых выполняется условное выражение;
– удаляет все записи из таблицы;
– вычисляет условное выражение;
– выбирает записи, соответствующие условному выражению;
– удаляет все столбцы из таблицы.

23 Для чего предназначена команда UPDATE в MySQL?

– обновляет все записи в таблице;
– обновляет все столбцы в таблице;
– обновляет записи, для которых выполняется условное выражение;
– обновляет все записи и столбцы в таблице.

ЛАБОРАТОРНАЯ РАБОТА №6
ВЗАИМОДЕЙСТВИЕ PHP И MYSQL

Цель работы: изучить основные команды PHP для соединения с сервером MySQL, отправки SQL-запросов на сервер базы данных и обработки результатов SQL-запросов.

Ход выполнения лабораторной работы должен быть отражен в отчете. Отчет должен содержать титульный лист, номера заданий, коды программ, скриншот с результатом выполнения программы.

6.1 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

SQL может применяться в прикладных программах двумя способами: в виде встроенного SQL и интерфейса программирования приложений (Application Program Interface, API). Первый способ напоминает использование PHP – инструкции SQL размещаются среди кода прикладной программы. В настоящий момент такой стиль не поддерживают ни MySQL, ни PHP. Второй подход заключается в том, что программа взаимодействует с СУБД посредством совокупности функций. Именно такой подход используется при взаимодействии PHP и MySQL.

6.1.1 Функции управления соединением с сервером MySQL

Для установки соединения с сервером MySQL используется функция mysql_connect(). Синтаксис функции:

resource **mysql_connect** ([string server [, string username [, string password]]])

Эта функция устанавливает соединение с сервером **server** MySQL и возвращает дескриптор соединения с базой данных, по которому все другие функции, принимающие этот дескриптор в качестве аргумента, будут однозначно определять выбранную базу данных. Вторым и третьим аргументами этой функции являются имя пользователя **username** и его пароль **password** соответственно (пример 6.1).

Пример 6.1

```
<?php
$dblocation = "localhost"; // Имя сервера
$dbuser = "root";          // Имя пользователя
$dbpasswd = "";            // Пароль
$dbcnx = @mysql_connect($dblocation,$dbuser,$dbpasswd);
if (!$dbcnx) // Если дескриптор равен 0, соединение не установлено
{
  echo("<P>В настоящий момент сервер базы данных не доступен, поэтому
        корректное отображение страницы невозможно.</P>");
```

```
  exit();
}?>
```

В примере 6.1 переменные **$dblocation**, **$dbuser** и **$dbpasswd** хранят имя сервера, имя пользователя и пароль и, как правило, прописываются в отдельном файле (к примеру, **config.php**), который потом вставляется в каждый PHP-файл, имеющий код для работы с MySQL.

Для закрытия открытого ранее соединения можно воспользоваться функцией mysql_close(). Открытое соединение закрывается автоматически по завершении работы сценария. Синтаксис функции:

bool **mysql_close** ([resource conn_id])

Эта функция разрывает соединение с сервером MySQL и возвращает true при успешном выполнении операции и false в противном случае. Функция принимает в качестве аргумента дескриптор соединения с базой данных, возвращаемый функцией mysql_connect() (пример 6.2).

Пример 6.2

```
<? // устанавливаем соединение с базой данных
$dbcnx = @mysql_connect($dblocation,$dbuser,$dbpasswd);
if (!$dbcnx)
{
  // Выводим предупреждение
  echo ("<P>В настоящий момент сервер базы данных не доступен, поэтому
  корректное отображение страницы невозможно.</P>");
  // Завершаем работу в случае неудачи
  exit();
}
  // Выполняем запросы
if(mysql_close($dbcnx)) // разрываем соединение
{
  echo("Соединение с базой данных прекращено");
}
else
{
  echo("Не удалось завершить соединение");
} ?>
```

6.1.2 Функции состояния и диагностики ошибок

Функции mysql_errno() и mysql_error() возвращают числовой код и информационное сообщение об ошибке в работе связанных с MySQL функций PHP. Однако эти функции не могут использоваться для получения информации о результатах безуспешного завершения вызовов функций mysql_connect(), поскольку идентификатор связи становится доступным только после успешного установления соединения.

Синтаксис функции:

int **mysql_errno** ([resource conn_id);

Функция возвращает код ошибки для связанной с заданным соединением MySQL функции, которая последней возвращала значение состояния. Нулевой код указывает на отсутствие ошибки.

Синтаксис функции:

string **mysql_error** ([resource conn_id]) ;

Функция возвращает строку с сообщением об ошибке для связанной с заданным соединением MySQL функции, которая последней возвращала значение состояния. Пустая строка свидетельствует об отсутствии ошибки.

6.1.3 Функции построения и выполнения запросов

Функции построения и выполнения запросов используются для создания и отсылки запросов на сервер MySQL. Строки запроса должны состоять из одного оператора SQL и не должны заканчиваться символом «точка с запятой» (";") или последовательностью "\g". Окончания «;» и «\g» являются соглашениями клиентской программы MySQL и не используются при выводе запросов через PHP.

Выборка указанной базы данных, для того чтобы сделать ее текущей базой данных для заданного соединения, выполняется функцией mysql_select_db(). Синтаксис функции:

bool **mysql_select_db**(string db_name [, resource conn_id]);

Таблицы, на которые будут делаться ссылки в последующих запросах, по умолчанию будут принадлежать этой базе данных, если не будет указана другая база данных. Функция возвращает значение «true» при успешном завершении и «false», если появилась ошибка.

Пример 6.3

Переменные $dblocation, $dbuser и $dbpasswd хранят имя сервера, имя пользователя и пароль и, как правило, прописываются в отдельном файле (к примеру, config.php) вместе с функциями для соединения и выбора базы данных, который потом вставляется в каждый PHP-файл, где имеется код для работы с MySQL.

```
<?php
$dblocation = "localhost";
$dbname = "softtime";
$dbuser = "root";
$dbpasswd = "";
$dbcnx = @mysql_connect($dblocation,$dbuser,$dbpasswd);
if (!$dbcnx)
{
  echo( "<P>В настоящий момент сервер базы данных не доступен, поэтому
        корректное отображение страницы невозможно.</P>" );
  exit();
}
```

```
if (!@mysql_select_db($dbname, $dbcnx))
{
  echo( "<P>В настоящий момент база данных не доступна, поэтому
        корректное отображение страницы невозможно.</P>" );
  exit(); } ?>
```

Задание 6.1. *Создать файл* ***config.php*** *для подключения к базе данных* ***bookby*** *(созданной в лабораторной работе №5).*

Для отправки строки к серверу MySQL для заданного соединения используется функция mysql_query(). Синтаксис функции:

int **mysql_query**(string query [, resource conn_id]) ;

Для операторов DELETE, INSERT, REPLACE и UPDATE функция mysql_query () возвращает значение «true» в случае успешного выполнения и значение «false», если имела место ошибка. После успешного выполнения запроса можно вызвать функцию mysql_affected_rows () , чтобы узнать число измененных строк.

Для операторов SELECT функция mysql_query () возвращает положительный идентификатор результирующего набора в случае успешного выполнения и значение «false», если имела место ошибка. Возвращаемый после успешного выполнения запроса идентификатор результата может использоваться самыми разными функциями обработки результирующих наборов, считывающими аргумент result_id. Для освобождения занятых результирующим набором ресурсов системы этот идентификатор необходимо передать функции mysql_free_result ().

6.1.4 Функции обработки результирующих наборов

Описанные в данном разделе функции используются для выборки результатов выполнения запросов, а также для предоставления информации о результатах, например количества измененных строк.

Синтаксис функции:

array **mysql_fetch_array** (resource result_id [, int result_type]) ;

Функция возвращает следующую строку данного результирующего набора в виде массива. Возвращает значение «false», если строк больше не осталось. Возвращаемый массив содержит значения, идентифицируемые как по числовым индексам столбцов, так и по ключам имен столбцов. Другими словами, доступ к значению каждого столбца можно получить как по числовому индексу, так и по названию столбца. Учитывается регистр символов при написании ассоциативных индексов, поэтому их следует записывать в том же регистре, что и имена столбцов в запросе.

Предположим, например, что отправлен следующий запрос:

SELECT last_name, first_name FROM president

Если строки извлекаются из результирующего набора в массив с именем $row, доступ к его элементам можно получить следующим образом:

$row [0] или $row ["last_name"] Содержит значение столбца last_name.

$row [1] или $row ["first_name"] Содержит значение столбца first_name.

Параметр result_type может иметь значения MYSQL_ASSOC (возвращаются значения только по индексам имен), MYSQL_NUM (возвращаются значения только по числовым индексам) или MYSQL_BOTH (возвращаются значения по индексам обоих типов). Если этот аргумент отсутствует, его значение по умолчанию устанавливается равным MYSQL_BOTH.

Вызов mysql_fetch_array () с параметром result_type, имеющим значение MYSQL_ASSOC или MYSQL_NUM, эквивалентно вызову mysql_fetch_assoc () или mysql_f etch_row ().

Пример 6.4

Из таблицы authors базы данных forums выбираются все записи, содержащие информацию об авторах, и выводятся на экран в табличной форме. Результат работы скрипта показан на рисунке 6.1.

```
<?php
include "config.php";
$ath = mysql_query("select * from authors;");
if($ath)
{
 // Определяем таблицу и заголовок
 echo "<table border=1>";
 echo "<tr><td>имя</td><td>пароль</td><td>e-mail</td><td>url</td></tr>";
 // Так как запрос возвращает несколько строк, применяем цикл
 while($author = mysql_fetch_array($ath))
 {
  echo "<tr><td>".$author['name']." </td><td>".$author['passw']."
    </td><td>".$author['email']." </td><td>".
  $author['url']."] </td></tr>";
 }
 echo "</table>";
}
else
{
 echo "<p><b>Error: ".mysql_error()."</b><p>";
 exit();
}
?>
```

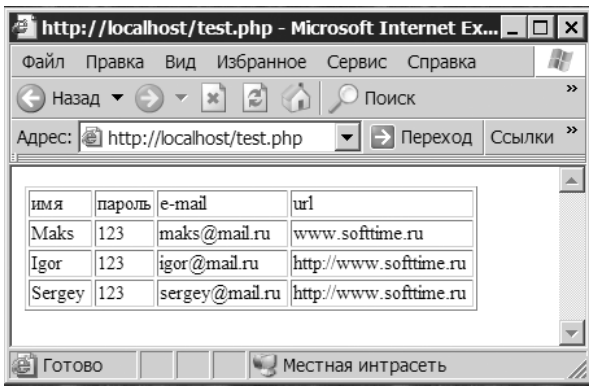

Рисунок 6.1 – Результат выполнения примера 6.4

Задание 6.2. *Создать форму для поиска книг в базе данных bookby* ***(poisk.php).*** *На странице должно быть поле со списком, в котором пользователь может выбрать критерий поиска (по ISBN, или по автору, или по заголовку книги), текстовое поле для ввода искомого значения и кнопка отправки данных. Например:*

```
<HTML> <HEAD> <TITLE>Поиск книги</TITLE> </HEAD>
<BODY>
<H1>Поиск книги в каталоге</H1>
<FORM ACTION="results.php" METHOD="POST">
    Выберите параметр поиска:<BR>
    <SELECT NAME="searchtype">
        <OPTION VALUE="author">Автор
        < OPTION VALUE ="title">Заголовок
        < OPTION VALUE ="isbn">ISBN
    </SELECT>
    <BR>
    Введите параметры поиска:<BR>
    <INPUT NAME="searchterm" TYPE="text">
    <BR>
    <INPUT TYPE="submit" VALUE="Поиск">
</FORM>
</BODY>
</HTML>
```

Задание 6.3. *Вывести на экран результаты поиска results.php по критериям, введенным в форму, созданную в задании 6.2. После нажатия кнопки в форме в переменной searchtype будет храниться имя столбца, по которому будет осуществляться поиск в таблице books, а искомое значение будет храниться в переменной searchterm. Пример файла results.php:*

```
<HTML>
 <HEAD> <TITLE>Результаты поиска книги</TITLE> </HEAD>
<BODY>
<H1> Результаты поиска книги </H1>
```

```
<?
trim($_POST["searchterm"]) ; //убираем лишние пробелы по краям слова
if (!$_POST["searchtype"] || !$_POST["searchterm"])
{
  echo "Вы не ввели критерии поиска. Вернитесь назад и попробуйте еще раз";
  exit();
}
// добавляет символы косой черты перед символами,
// которые могут быть интерпретированы как управляющие
$_POST["searchtype"] = addslashes($_POST["searchtype"]);
$_POST["searchterm"] = addslashes($_POST["searchterm"]);
include "config.php"; // соединяемся с сервером MySQL и выбираем БД
// составляем запрос
$query = "select * from books where ".$_POST["searchtype"]." like
          '%".$_POST["searchterm"]."%'";
$result = mysql_query($query); // выполняем запрос
// определяем количество строк в запросе
$num_results = mysql_num_rows ($result);
echo "<P>Количество найденных книг: ".$num_results."</P>";
for  ($i=0; $i <$num_results; $i++) {
      $row = mysql_fetch_array($result); // функция берет каждую строку
//из списка результата и возвращает ее в виде ассоциативного массива
      echo "<P><STRONG>". ($i+1) .". Название: ";
      echo $row["title"];
      echo "</STRONG><BR>Автор: " ;
      echo $row["author"];
      echo "<BR>ISBN: ";
      echo $row["isbn"];
      echo "<BR>Цена: ";
      echo $row["price"];
      echo "</P>";
}
?>
</BODY> </HTML>
```

Функция **mysql_fetch_assoc** () возвращает следующую строку данного результирующего набора в виде ассоциативного массива. Синтаксис функции:

array **mysql_fetch_assoc** (resource result_id);

Функция возвращает значение «false», если строк больше не осталось. Значения столбцов доступны с помощью ассоциативных имен, соответствующих именам столбцов, выбранных запросом.

Вызов mysql_fetch_assoc () подобен вызову mysql_fetch_array() со вторым аргументом MYSQL_ASSOC. Значения столбцов с цифровыми индексам не возвращаются.

Функция **mysql_fetch_row**() возвращает следующую строку заданного результирующего набора в виде массива либо значение «false», если строк больше не осталось. Синтаксис функции:

array **mysql_fetch_row** (resource result_id) ;

Доступ к значениям столбцов можно получать через элементы массива, используя индексы столбцов в диапазоне от 0 до mysql_num_fields ()-1.

Вызов функции mysql_fetch_row() аналогичен вызову функции mysql_fetch_array() со вторым аргументом, равным MYSQL_NUM. Значения столбцов с ассоциативными индексами не возвращаются.

Функция **mysql_affected_rows** () возвращает количество строк, измененных последним запросом DELETE, INSERT, REPLACE или UPDATE в заданном соединении. Синтаксис функции:

int **mysql_affected_rows** ([resource conn_id]) ;

Функция mysql_affected_rows () возвращает значение 0, если ни одна строка изменена не была, и значение -1, если имела место ошибка.

Вызванная после оператора SELECT функция mysql_affected_rows () возвращает число строк, полученных в результате выборки. Хотя обычно для операторов SELECT используется функция mysql_num_rows ().

Функция **mysql_num_fields** () возвращает число столбцов в заданном результирующем наборе. Синтаксис функции:

int **mysql_num_fields** (resource result_id) ;

Функция **mysql_num_rows** () возвращает число строк в заданном результирующем наборе. Синтаксис функции:

int **mysql_num_rows** (resource result_id) ;

Функция **mysql_insert_id** () возвращает значение AUTO_INCREMENT, сгенерированное последним запросом заданного соединения. Синтаксис функции:

int **mysql_insert_id** ([resource conn_id]) ;

Если в процессе всего соединения такое значение создано не было, эта функция возвращает нуль. В общем, функцию mysql_insert_id () следует вызывать сразу после запроса, который сгенерирует значение AUTO_INCREMENT. Если между этим запросом и вызовом функции будет выполнен промежуточный запрос, возвращаемое функцией значение может быть сброшено до нуля.

Задание 6.4. *Создать файл* ***show.php****, который будет выводить на экран список всех книг.*

Задание 6.5. *Создать форму для ввода новой книги в БД.*

6.2 ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

1 Создать базу данных employee

```
create database employee;
use employee;

create table department
(
departmentID int not null auto_increment primary key,
name varchar(30)
) type=InnoDB;

create table employee
(
employeeID int not null auto_increment primary key,
name varchar(80),
job varchar(30),
departmentID int not null references department (departmentID)
) type=InnoDB;

create table employeeSkills
(
employeeID int not null references employee(employeeID),
skill varchar(15) not null,
primary key(employeeID, skill)
) type=InnoDB;

create table client
(
clientID int not null auto_increment primary key,
name varchar(40),
address varchar(100),
contactPerson varchar(80),
contactNumber char(12)
) type=InnoDB;

create table assignment
(
clientID int not null references client(clientID),
employeeID int not null references employee(employeeID),
workdate date not null,
hours float,
primary key(clientID, employeeID, workdate)
) type=InnoDB;
```

```
insert into department values
(42, 'Финансовый отдел'),
(128, 'Отдел проектирования'),
(null, 'Отдел кадров'),
(null, 'Отдел маркетинга');
insert from employee values
(7513, 'Нора Эдвардс', 'Программист', 128),
(9842, 'Бен Смит', 'Администратор БД' ,42),
(6651, 'Аджай Пател', 'Программист' ,128),
(9006, 'Кэнди Барнетт', 'Системный администратор' ,128);

insert into employeeskills values
(7513, 'C'),
(7513, 'Perl'),
(7513, 'Java'),
(9842, 'DB2'),
(6651, 'VB'),
(6651, 'Java'),
(9006, 'NT'),
(9006, 'Linux');

insert into client values
(Null, 'TelcoInc', '1 Collins St Melbourne', 'Fred Smith', '95551234'),
(Null, 'The Bank', '100 Bourke St Melbourne', 'Jan Tristan', '95559876');

insert into assignment values
(1, 7513, ' 2003-01-20', 8.5);
```

2 Написать скрипты, которые выполняют нижеперечисленные запросы и выводят на экран результаты их работы.

а) Выбрать из таблицы всех работников, вывести на экран их имена и их табельный номер.

```
select name, employeeID from employee;
```

б) Выбрать список всех работников, которые работают программистами:

```
select employeeID, name
from employee
where job=’Программист’;
```

в) Выбрать все рабочие дни работника с табельным номером 6651, в которые он работал более 8 ч.

```
select * from assignment
where employeeID=6651 and hourse > 8;
```

г) Подсчитать количество различных должностей в таблице работников.

```
select count(distinct job) from employee;
```

д) Подсчитать количество различных должностей в таблице работников.

```
select count(*), job
from employee
group by job;
```

е) Выбрать все записи из таблицы работников, которые единственные работают в своей должности.

```
select count(*), job
from employee
group by job
having count(*)=1;
```

ж) Выбрать все записи из таблицы работников и отсортировать их по должности в алфавитном порядке, а по имени в обратном порядке.

```
select *
from employee
order by job asc, name desc;
```

з) Выбрать первые пять записей из таблицы employeeskills

```
select *
from employeeskills
limit 5;
```

6.3 КОНТРОЛЬНЫЕ ВОПРОСЫ

1 Какие функции возвращают числовой код и информационное сообщение об ошибке?

– mysql_error();
– mysql_connect();
– mysql_errno();
– mysql_select_db();
– mysql_affected_row().

2 Какая команда позволяет установить текущую базу данных для заданного соединения?

– mysql_connect();
– mysql_select_db();
– mysql_affected_row();
– mysql_query();
– mysql_fetch_array().

3 Для чего предназначена функция mysql_fetch_array()?

– возвращает количество строк, измененных последним запросом;
– возвращает следующую строку данного результирующего набора в виде массива;
– выбирает текущую базу данных;
– отправляет строку запроса к серверу;
– возвращает сообщение об ошибке.

4 Какая команда возвращает значение AUTO_INCREMENT, сгенерированное последним запросом заданного соединения?

– mysql_select_db();
– mysql_affected_row();
– mysql_insert_id();
– mysql_free_result();
– mysql_fetch_array().

5 Какие функции возвращают следующую строку данного результирующего набора в виде ассоциативного массива?

– mysql_fetch_assoc();
– mysql_fetch_array() с параметром MYSQL_ASSOC;
– mysql_fetch_array() с параметром MYSQL_NUM;
– mysql_affected_row();
– mysql_num_rows().

6 Какая функция возвращает число строк в заданном результирующем наборе?

– mysql_insert_id();
– mysql_num_rows();
– mysql_affected_row();
– mysql_free_result();
– mysql_fetch_array().

7 Какая команда открывает соединение с сервером MySQL?

– mysql_select_db();
– mysql_connect();
– mysql_affected_row();
– mysql_errno();
– mysql_error().

8 Какая команда отправляет строку запроса к серверу MySQL?

– mysql_connect();
– mysql_select_db();
– mysql_affected_row();
– mysql_query();
– mysql_fetch_array().

9 Для чего предназначена функция mysql_affected_row()?

– возвращает количество строк, измененных последним запросом;
– выбирает текущую базу данных;
– отправляет строку запроса к серверу;
– возвращает сообщение об ошибке;
– возвращает количество строк в результирующем запросе.

10 В виде какого массива может вернуть функция mysql_fetch_array() следующую строку результирующего набора?

– в виде ассоциативного (по индексам имен);
– в виде индексного (по числовым индексам);
– по индексам обоих типов;

– нет верных ответов;

– эта функция не возвращает никакого массива.

11 Какие функции возвращают следующую строку данного результирующего набора в виде массива с числовыми индексами?

– mysql_fetch_row();

– mysql_fetch_array() с параметром MYSQL_NUM;

– mysql_fetch_array() с параметром MYSQL_ASSOC;

– mysql_affected_row();

– mysql_num_rows().

ЛИТЕРАТУРА

1 Васкевич, Д. Стратегия клиент/сервер. Руководство по выживанию для специалистов по реорганизации бизнеса / Д. Васкевич. – Киев : Диалектика, 1996.

2 Дилип, Н. Стандарты и протоколы Интернета / Н. Дилип. – М. : Русская редакция, 1999.

3 Дюбуа, П. MySQL / П. Дюбуа. – М. : Изд. дом «Вильямс», 2001.

4 Жданов, А. Dreamweaver 3: краткий курс / А. Жданов, Б. Карпов, М. Левченко. – СПб. : Питер, 2001.

5 Кожемякин, А. А. HTML и CSS в примерах. Создание web-страниц / А. А. Кожемякин. – М. : Альтекс-А, 2004.

6 Котеров, Д. PHP 5 / Д. Котеров, А. Костарев. – СПб. : БХВ-Петербург, 2004.

7 Кузнецов, М. В. PHP 5. Практика разработки web-сайта / М. В. Кузнецов, И. В. Симдянов. – СПб. : БХВ-Петербург, 2005.

8 Кузнецов, М. В. Самоучитель PHP 5 / М. В. Кузнецов, И. В. Симдянов. – СПб. : БХВ-Петербург, 2004.

9 Мещеряков, Е. Публикация баз данных в Интернете / Е. Мещеряков. – СПб. : БХВ, 2001.

10 Орлов, Л. В. Web-сайт без секретов / Л. В. Орлов. – М. : Новый издательский дом, 2004.

11 Томсон, Л. Разработка web-приложений на PHP и MySQL / Л. Томсон. – СПб. : ДиаСофтЮП, 2003.

12 Хоумер, А. Dynamic HTML : справочник / А. Хоумер, К. Улмен. – СПб. : Питер, 2000.

13 Шарма, В. Разработка web-серверов для электронной коммерции / В. Шарма, Р. Шарма. – М. : Изд. дом «Вильямс», 2001.

Св. план 2016, поз. 9

Учебное издание

Алексеев Виктор Федорович
Русак Татьяна Вячеславовна
Пискун Геннадий Адамович

ОСНОВЫ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

ПОСОБИЕ

Редактор *Е. И. Герман*
Компьютерная правка, оригинал-макет *А. В. Бас*

Подписано в печать 09.02.2017. Формат 60x84 1/16. Бумага офсетная. Гарнитура «Таймс».
Отпечатано на ризографе. Усл. печ. л. 3,14. Уч.-изд. л. 6,5. Тираж 100 экз. Заказ 250.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий №1/238 от 24.03.2014,
№2/113 от 07.04.2014, №3/615 от 07.04.2014.
ЛП №02330/264 от 14.04.2014.
220013, Минск, П. Бровки, 6