Методические рекомендации по дисциплине "Основы Web - программирования"

Лабораторная работа №1 - 2.

Создание простейшего HTML-документа. Форматирование шрифта и абзаца

Задание на выполнение

1. Создать файл с гипертекстовым документом:

- Запустить редактор Блокнот, ввести текст:

Приветствую Вас на моей первой web-страничке!

- Сохранить файл в созданной папке. При сохранении, в окне диалога **Сохранить как…** в строке **Тип файла:** выбрать вариант **Все файлы (*.*)** , а в строке **Имя файла** задать имя с расширением .**htm**, например **1_name.htm** (где **name** – ваше имя)
- Закрыть документ, найти его пиктограмму в окне **Мой компьютер** или в окне программы **Проводник.**
- Открыть файл. Проанализировать, *с помощью какого приложения отображается файл* и как выглядит введенная фраза.

2. Ввести теги, определяющие структуру html-документа:

- С помощью контекстного меню открыть файл с помощью редактора Блокнот. Ввести приведенные ниже теги, в разделе заголовка документа (между тегами <**TITLE**> </**TITLE**>) указать свою фамилию.

<HTML>

<HEAD> <TITLE> Фамилия </TITLE>

</HEAD>

<BODY>

Приветствую Вас на моей первой web-страничке!

</BODY>

</HTML>

- **Сохранить** документ под тем же именем, обновить его отображение в броузере (выполнить **Вид/Обновить** или нажать кнопку **Обновить** на панели инструментов). Проанализировать произошедшие изменения в отображении документа.

3. Отредактировать документ:

- Вызвать меню броузера **Вид/Просмотр HTML-кода** и добавить после текста **«Приветствую Вас на моей первой web-страничке!»** текст подписи:

Студент группы NNN Фамилия Имя

Сохранить документ (но не закрывать) и обновить его просмотр в броузере.

- Используя одиночный тег **
, отредактировать документ так, чтобы подпись начиналась с новой строки, а **Фамилия Имя – в следующей строке. Просмотреть в броузере новый вариант.

Внимание! ***После каждого изменения документ нужно сохранять, а просмотр в броузере начинать с обновления загрузки документа с помощью кнопки «Обновить» на панели инструментов.***

4. Оформить фрагменты текста с помощью стилей **Заголовков**:

- Первую строку документа оформить стилем **Заголовок 1-го уровня** с помощью парного

тега **<H1> …</H1>**. Вторую строку оформить как **Заголовок 6-го уровня**, а третью как **Заголовок 4-го уровня**.

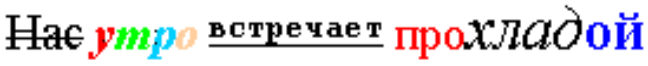
- Просмотреть документ в броузере, изменяя настройку отображения шрифтов (меню **Вид / Размер шрифта / Самый крупный, Средний, Мелкий** и **Самый мелкий**).
- Поменять стиль оформления первой строки на **Заголовок 2 уровня**, второй строки - на **Заголовок 5 уровня**, последней строки - на **Заголовок 3-го уровня**.

5. Выполнить форматирование шрифта:

- После строки **`Фамилия Имя`** добавить еще одну строку текста

`Нас утро встречает прохладой`

- Оформить фразу по приведенному ниже образцу.



В слове УТРО все буквы должны иметь **разные цвета**. В слове ПРОХЛАДОЙ оформить буквы ПРО – **красным** цветом, ОЙ – **синим**.

- Оформить строку с подписью (**`Студент группы NNN Фамилия Имя`**) **курсивом**, размер шрифта задать относительным изменением. Использовать теги **<FONT SIZE=«+2»>** и **<I>**
- Просмотреть полученный документ в броузере.

6. Выполнить форматирование абзацев:

- Создать новый документ **2_name.htm**, сохранить его в той же рабочей папке.
- Ввести текст (использовать копирование текста из документа **1_name.htm**):

<HTML>

<HEAD> <TITLE> **`Фамилия`** **</TITLE>**

</HEAD>

<BODY>

`Приветствую Вас на моей второй web-страничке!` **
** **`Монолог Гамлета`**

</BODY>

</HTML>

- Выровнять текст **по центру**.
- Ввести текст:

Быть иль не быть - вот в чем вопрос. Что благороднее: сносить удары неистовой судьбы - иль против моря невзгод вооружиться, в бой вступить. И все покончить разом...

- Оформить выравнивание абзаца **по ширине**.
- Ограничить абзац горизонтальными разделительными линиями сверху и снизу, используя тег <HR>.
- Скопировать монолог и разбить его на абзацы. Выровнять **по центру**.

Быть иль не быть - вот в чем вопрос.
Что благороднее: сносить удары
Неистовой судьбы - иль против моря
Невзгод вооружиться, в бой вступить

И все покончить разом...

- Сохранить документ.
- Просмотреть документ в окне броузера, изменяя размер окна.

7. Выполнить оформление списков:

- Создать новый документ **3_name.htm**, сохранить его в той же рабочей папке жесткого диска.
- Ввести текст:

```
<HTML>
    <HEAD> <TITLE> Фамилия </TITLE>
    </HEAD>
    <BODY>
Приветствую Вас на моей третьей web-страничке!
    </BODY>
</HTML>
```

- Дополнить текст документа (между тегами <BODY>…</BODY>) следующим текстом:

```
Я знаю как оформлять:
Шрифты,
Заголовки,
Абзацы
```

- Оформить три последние строки как **список нумерованный**. Для этого использовать следующую конструкцию тегов:

```
<OL>
<LI> Шрифты, </LI>
<LI> Заголовки, </LI>
<LI> Абзацы </LI>
</OL>
```

- Поменять оформление списка на **список маркированный**. Использовать теги <UL>, </UL>
- Создать «смешанный» список:

```
Я знаю как оформлять:
   1. Шрифты
      • Размер
      • Цвет
      • Гарнитуру
      • Индексы
   2. Заголовки
      • От 1-го до 6-го уровня
   3. Абзацы
```

- **Выравнивание**
- **Разрыв строк внутри абзаца**
- **С использованием переформатирования.**

8. Предъявить результаты работы преподавателю.

Таблица основных тегов HTML-документа. Теги форматирования шрифта и абзаца

Назначение	Вид тегов	Примечание
Общая структура документа HTML		
Тип документа	<HTML></HTML>	Начало и конец документа
Имя документа	<HEAD></HEAD>	Не отображается броузером
Заголовок	<TITLE></TITLE>	Содержимое строки заголовка окна броузера
Тело документа	<BODY></BODY>	Содержимое WEB-страницы
Структура содержания документа		
Внутренние заголовки различного уровня	<H№> текст </H№>	Где № – номер уровня заголовка (от 1 до 6). Например, <H1>…</H1> - заголовок 1-го уровня.
Заголовок с выравниванием	<H№ ALIGN=”LEFT| CENTER| RIGHT”> текст </H№>	LEFT - по левому краю, CENTER - по центру, RIGHT - по правому краю.
Форматирование абзацев		
Создание абзаца (параграфа)	<P> текст </P>	Абзацы отделяются двойным межстрочным интервалом
Перевод строки внутри абзаца	 	Одиночный тег
Выравнивание абзаца	<P ALIGN=”LEFT”>текст </P> <P ALIGN= ”CENTER”>текст </P> <P ALIGN=”RIGHT”> текст</P> <P ALIGN=”JUSTIFY”> текст </P>	LEFT - по левому краю CENTER - по центру RIGHT - по правому краю JUSTIFY – по ширине
Разделительная горизонтальная линия между абзацами	<HR SIZE=«?»>	Одиночный тег. «?» - толщина линии в пикселях. Толщину линии можно не указывать.
Форматирование шрифта		
Жирный	<B> текст </B>	<B>**Жирный**</B>

Курсив	<I> текст </I>	<I> *Курсив* </I> <U> Подчеркнутый </U> <S> ~~Перечеркнутый~~</S> ^{Верхний индекс} _{Нижний индекс}
Подчеркнутый	<U> текст </U>	
Перечеркнутый	<S> текст </S>	
Увеличенный размер	<BIG> текст </BIG >	
Уменьшенный размер	<SMALL> текст </SMALL>	
Верхний индекс	^{текст}	
Нижний индекс	_{текст}	
Размер шрифта	<FONT SIZE=«?»> текст </FONT>	?- значения от 1 до 7 или относительное изменение (например, +2)
Базовый размер шрифта	<BASEFONT SIZE=«?»>	Одиночный тег ? – размер от 1 до 7; по умолчанию равен 3 и задается для всего документа в целом
Гарнитура шрифта	<FONT FASE=«название1, название2, …»> текст </FONT>	Текст оформляется первым, установленным на компьютере шрифтом из списка названий
Цвет шрифта	<FONT COLOR=”$$$$$$”> текст </FONT>	Цвет задается либо ключевым словом, либо шестнадцатеричным кодом с символом # RED –красный, #FF0000 – шестнадцатеричный код – красного цвета
Создание списков		
Нумерованный	<OL>элементы списка</OL>	<OL> <LI>Элемент списка 1 </LI> <LI>Элемент списка 2</LI> <LI>Элемент списка 3</LI> </OL>
Маркированный	<UL> элементы списка </UL>	
Элемент списка	<LI> элементы списка </LI>	

Таблица основных цветов

Цвет	**Color's name**	**Шестнадцатеричный код цвета**		
		Red	**Green**	**Blue**
Черный	black	00	00	00
Темно-синий	navy	00	00	80
Голубой	blue	00	00	FF
Зеленый	green	00	80	00

Цвет	Color's name	Шестнадцатеричный код цвета		
		Red	Green	Blue
Темно-зеленый	teal	00	80	80
Салатный	lime	00	FF	00
Бледно-голубой	aqua	00	FF	FF
Вишневый	maroon	80	00	00
Фиолетовый	purple	80	00	80
Оливковый	olive	80	80	00
Серый	gray	80	80	80
Светло-серый	silver	C0	C0	C0
Красный	red	FF	00	00
Лиловый	fushsia	FF	00	FF
Желтый	yellow	FF	FF	00
Белый	white	FF	FF	FF

Вставка в HTML-документ рисунков. Создание закладок и гиперссылок

Задание на выполнение

1. Скопировать из Интернета или какой-либо папки в личную папку файлы три графических файла (например, **Arrows1.wmf, Arrows2.wmf, Arrows3.wmf).**

   *Убедиться, что созданные ранее документы **1_name.htm, 2_name.htm и 3_name.htm** также находятся в вашей папке на жёстком диске.*

2. Вставка рисунков в документ.

   - Открыть в Блокноте документ **2_name.htm.**
   - Вставить рисунок **Arrows1.wmf** в начало документа **2_name.htm**. Для вставки использовать тег **IMG** с параметрами **WIDTH** и **HEIGHT** для установки размеров рисунка 50 пикселов по горизонтали и по вертикали.
   - Сохранить документ под именем **4_name.htm**.
   - Просмотреть в броузере полученный результат.
   - Ввести в тег рисунка параметр **ALIGN** для выравнивания рисунка по правому краю. Просмотреть результат в броузере.
   - Вставить рисунок **Arrows2.wmf** в конец документа **4_name.htm** перед, подобрать тип выравнивания рисунка на свое усмотрение. Установить размер рисунка 100 пикселов по горизонтали и по вертикали. С помощью параметра **ALT** создать всплывающую подсказку «**Рисунок 2**», появляющуюся при наведении курсора мыши на рисунок.
   - Просмотреть в броузере полученный результат.

3. Создание гиперссылок и закладок.

   - В документе **3_name.htm** закрепить гиперрсылки за следующими элементами списка:

За словом **Шрифт** – гиперссылка на документ **1_name.htm.**

За словом **Заголовки** – на документ **1_name.htm.**

За словом **Абзацы** - на документ **2_name.htm.**

- Создать закладку в документе **1_name.htm** перед фразой **«Нас утро встречает прохладой»**. Дать ей имя «**Morning**».
- Изменить первую гиперссылку (слово **Шрифт**) так, чтобы она указывала на закладку «**Morning**» в документе **1_name.htm.**
- Создать закладку в начале текущего документа **3_name.htm**.. Присвоить ей имя «**Hello**».
- Изменить вторую гиперссылку (на слове **Заголовки**), определив для неё переход в начало текущего документа на установленную закладку «**Hello**».
- Создать закладку в документе **2_name.htm** перед фрагментом монолога. Присвоить ей имя «**Mono**».
- Установить на слово **переформатирования** гиперссылку на закладку «**Mono**».
- Проверить правильность переходов по всем гиперссылкам.

4. Закрепить гиперссылки за графическими файлами:
   - Отредактировать тег вставки рисунка **Arrows1.wmf,** ввести в тег атрибут **ALT** для отображения текста подсказки **«Вернуться»**. Просмотреть в броузере как реагирует рисунок на наведение курсора мыши.
   - Закрепить за рисунком **Arrows1.wmf** в документе **4_name.htm** гиперссылку на документ **3_name.htm**. Выполнить переход между документами.
5. Предъявить результат преподавателю.

Основные теги вставки рисунков, закладок и гиперссылок

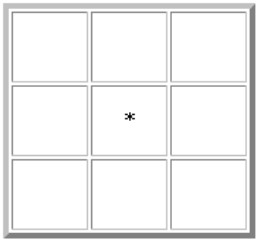
<table>
<tr><th colspan="3">Вставка изображений</th></tr>
<tr><td>Вставка графического файла</td><td><IMG SRC=”файл” WIDTH=”ширина” HEIGHT=”высота” ></td><td rowspan="3">Пример:
<IMG SRC=”grafica.gif”
WIDTH=”550”
HEIGHT=”240”
ALIGN=” right”
ALT=”Графический файл”></td></tr>
<tr><td>Выравнивание картинки относительно текста</td><td><IMG SRC=”файл” ALIGN=”left|right|top|bottom”></td></tr>
<tr><td>Вывод текста всплывающей подсказки при наведении курсора мыши на рисунок</td><td><IMG SRC=”файл” ALT=”текст”></td></tr>
<tr><th colspan="3">Вставка ссылок</th></tr>
<tr><td>Ссылки на другую страницу</td><td><A HREF=” страница”>
текст </A></td><td><A HREF=”str2.htm”>
Ссылка1</A></td></tr>
<tr><td>Ссылка на закладку в другом документе</td><td><A HREF=” страница # имя закладки”>
текст</A></td><td><A HREF=” index.htm #met1”>
На главную страницу </A>”</td></tr>
</table>

Ссылка на закладку в том же документе	<A HREF=”#имя закладки ”> текст </A>	<A HREF=” #metka1”> Ссылка2</A>
Определить закладку	<A NAME=”имя закладки”>текст</A>	<A NAME=”metka1”></A>
Цвет фона, текста и ссылок		
Фоновая картинка	<BODY BACKGROUND=”файл рисунка”>	<BODY BACKGROUND =”grafica.gif” TEXT=”black” (черный) LINK=”#FF0000” (красный) VLINK=”#FFFF00” (желтый) ALINK=”#FFFFFF” (белый) </BODY>
Цвет фона	<BODY BGCOLOR=”#$$$$$$”>	
Цвет текста	<BODY TEXT=”#$$$$$$”>	
Цвет ссылки	<BODY LINK=”#$$$$$$”>	
Цвет пройденной ссылки	<BODY VLINK=”#$$$$$$”>	
Цвет активной ссылки	<BODY ALINK=”#$$$$$$”>	

Создание и форматирование таблиц

Задание на выполнение

1. Создать таблицу по приведенному образцу, сохранить документ под именем **tabl_name.htm.** Сверху над таблицей разместить заголовок **Таблица №1**



При отображении таблицы в броузере должны удовлетворяться следующие условия:

- таблица должна быть выравнена по центру и быть правильной (симметричной) формы;
- в центральной ячейке поместить символ ***** (звездочка), остальные ячейки должны быть пустыми.

Примечание. Для отображения пустых ячеек в них нужно поместить символьный примитив пробела **` `**

2. В этом же документе создать копию таблицы №1, ввести заголовок **Таблица №2** и модифицировать ее:

- В центральной ячейке разместить рисунок **Arrows3.wmf**
- «Раскрасить» все остальные ячейки в различные цвета.

3. Создать еще одну копию таблицы – **Таблица №3** и отредактировать теги таблицы так, чтобы она соответствовала приведенному ниже образцу.

Примечание. Для объединения ячеек в тегах <TD> необходимо использовать параметры **`colspan=`** и **`rowspan=`**

4. Создать новый HTML-документ - **rasp_name.htm** с расписанием занятий.

- Документ должен начинаться заголовком
  Расписание занятий гр. NNN на весенний семестр 2005 г.
- Первая строка таблицы должна быть оформлена как заголовки полей (с использованием тегов **<TH>**).
- Таблица по ширине должна занимать полный размер окна. Ширину отдельных столбцов задать в относительных единицах (в %), с тем, чтобы при изменении ширины окна пропорции таблицы сохранялись.

День недели	Время	Предмет	Преподаватель	Аудитория
Понедельник	8:30-10:05	Математика (лек)	доц. Иванов А.А.	320
	10:15-11:50	Математика (пр)	преп. Петрова И.А.	302
	12:30-14:05	Физика (лаб)	доц. Сидоров О.И.	307
Вторник	8:30-10:05	История (лек)	проф. Громова О.А.	310
	10:15-11:50	История (сем)	преп. Попов М.А.	302
	12:30-14:05	Физика (лаб)	доц. Сидоров О.И.	307
...	...	...	...	...

- Просмотреть созданный документ в броузере при различных размерах окна и различных настройках размера шрифта.

5. Сохранить файл с расписанием под именем **rasp_menu_name.htm** и модифицировать его.

6. После заголовка создать таблицу, состоящую их одной строки меню с названиями дней недели.

Расписание

Понед.	Вторник	Среда	Четверг	Пятн.	Субб.

7. В таблице с расписанием установить закладки на названия дней недели.

8. В таблице меню создать гиперссылки на соответствующие дни недели.

9. Выполнить цветовое оформление каждой ячейки меню.

10. Проверить правильность выполнения переходов по гиперссылкам.

11. Создать группу web-страниц, объединенных меню:

- На рабочем диске создать папку **My_raspisanie** для размещения файлов расписания.
- Поместить расписание на каждый день недели и таблицу с меню в отдельные файлы. Имена файлов: **menu.htm** – для главной страницы, названия дней недели – для остальных. Все документы разместить в папке **My_raspisanie .**
- Отредактировать гиперссылки меню так, чтобы по ним выполнялись переходы на соответствующий документ.
- В конце каждого файла с расписанием на день организовать гиперссылку для возврата в главный документ с меню.

- Оформить фон каждого дня недели собственным цветом, совпадающим с цветом ячейки таблицы меню.

12. Предъявить результат преподавателю.

Теги оформления таблиц

<table>
<tr><td>Определить таблицу</td><td><TABLE></TABLE></td><td rowspan="8">Пример
<TABLE border=”1” align =«CENTER» width=«50%» >
<TR>
<TH >Товар</TH>
<TH>Цена</TH>
</TR>
<TR>
<TD>Радиотелефон</TD>
<TD>2000 </TD>
</TR>
</TABLE>
<table><tr><th>Товар</th><th>Цена</th></tr><tr><td>Радиотелефон</td><td>2000</td></tr></table></td></tr>
<tr><td>Окантовка таблицы</td><td><TABLE BORDER =”?” </TABLE></td></tr>
<tr><td>Строка таблицы</td><td><TR> </TR></td></tr>
<tr><td>Выравнивание</td><td><TR ALIGN=left | right | center | middle | bottom ></td></tr>
<tr><td>Ячейка таблицы</td><td><TD></TD></td></tr>
<tr><td>Выравнивание по горизонтали</td><td><TD ALIGN=LEFT | RIGHT | CEN-TER></td></tr>
<tr><td>Выравнивание по вертикали</td><td><TD VALIGN = TOP|MIDDLE| BOT-TOM></td></tr>
<tr><td>Установка ширины ячейки (в пикселях или %)</td><td><TD WIDTH=«?»></td></tr>
<tr><td>Заливка цветом ячейки</td><td><TD BGCOLOR = «# цвет»> </TD></td><td><TD BGCOLOR = «#FF0000»> </TD> красный цвет</td></tr>
<tr><td>Заголовок столбца или строки</td><td><TH>текст </TH></td><td>Текст в ячейке выравнивается по центру, устанавливается жирный шрифт</td></tr>
</table>

Лабораторная работа № 3-4

Каскадные таблицы стилей (CSS)

Теоретические основы

Расшифровывается CSS (англ. *Cascading Style Sheets)* как каскадные таблицы стилей и является технологией оформления веб-страниц.

Основным понятием CSS является стиль – т. е. набор правил оформления и форматирования, который может быть применен к различным элементам документа. В стандартном HTML для присвоения какому-либо элементу определенных свойств (таких, как цвет, размер, положение на странице и т. п.) приходилось каждый раз описывать эти свойства, увеличивая размер файла и время загрузки на компьютер просматривающего ее пользователя.

CSS действует более удобным и экономичным способом. Для присвоения какому-либо элементу определенных характеристик необходимо один раз описать этот элемент и определить это описание как стиль, а в дальнейшем просто указывать, что элемент, который нужно оформить соответствующим образом, должен принять свойства указанного стиля.

Более того, можно сохранить описание стиля не в тексте кода документа, а в отдельном файле – это позволит использовать описание стиля на любом количестве Web-страниц, а также изменить оформление любого количества страниц, исправив лишь описание стиля в одном (отдельном) файле.

Кроме того, CSS позволяет работать со шрифтовым оформлением страниц на гораздо более высоком уровне, чем стандартный HTML, избегая излишнего утяжеления страниц графикой.

```
<html>
<head>
<style type="text/css">
        .newfont{font-size:24px; color:#CC9933}
</style>
<title>Классы для создания тэгов.</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
</head>
<body>
<blockquote class=" newfont ">Заголовок</blockquote>
</body>
</html>
```

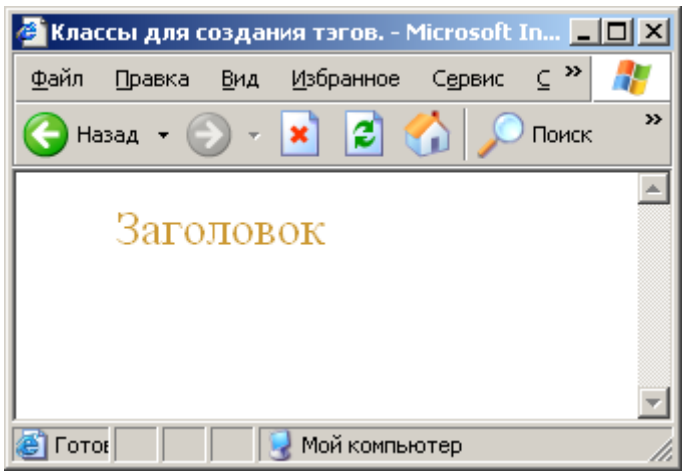


Рисунок 1

Данный пример иллюстрирует вариант объявления нового стиля в документе и потом его использования.

Синтаксис и элементы CSS

1 Добавление стилей CSS в HTML-документ

Существует несколько способов связывания документа и таблицы стилей:

- Связывание - позволяет использовать одну таблицу стилей для форматирования многих страниц HTML
- Внедрение - позволяет задавать все правила таблицы стилей непосредственно в самом документе
- Встраивание в теги документа - позволяет изменять форматирование конкретных элементов страницы
- Импортирование - позволяет встраивать в документ таблицу стилей, расположенную на сервере

Остановимся на каждом из этих способов более подробно.

Связывание. Напомним, что информация о стилях может располагаться либо в отдельном файле, либо непосредственно в коде документа. Расположение описания стилей в отдельном файле целесообразно при применении стилей при количестве страниц более 1. Для этого необходимо создать текстовый файл, описать необходимые стили и в коде документов, которые будут использовать эти стили необходимо создать ссылку на данный файл. Отметим, что данный файл может располагаться где угодно, необходимым условием является только то, чтобы броузер клиента мог его загрузить на свою сторону. Осуществляется это с помощью тега LINK, располагающегося внутри тега HEAD документов:

<LINK REL=STYLESHEET TYPE="text/css" HREF="URL">

Первые два параметра этого тега являются зарезервированными именами, требующимися для того, чтобы сообщить броузеру, что на этой страничке будет использоваться CSS. Третий параметр – HREF= «URL» – указывает на файл, который содержит описания стилей. Этот параметр должен содержать либо относительный путь к файлу – в случае, если он находится на том же сервере, что и документ, из которого к нему обращаются – или полный URL («http://...») в случае, если файл стилей находится на другом сервере.

```
<head>
<title></title>
<meta http-equiv="content-type" content="text/html; charset=windows-1251">
<link rel="stylesheet" href="css/default.css">
```

</head>

*Внедрение.*Второй вариант, при котором описание стилей располагается в коде Web-странички, внутри тега HEAD, в теге <STYLE type="text/css">... </STYLE. В этом случае вы можете использовать эти стили для элементов, располагающихся в пределах странички. Параметр type="text/css" является обязательным и служит для указания броузеру использовать CSS.

```
<head>
<style type="text/css" >
        .el_cl_1{display:inline; z-index:1};
        .el_lst{display:list-item; margin:-1%; background:#ff0000 url("bc.jpg") no-repeat};
</style>
<title></title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
</head>
```

*Встраивание в теги документа.*Данный, третий по счету, метод позволяет располагать описания стилей непосредственно внутри тега элемента, который описывает. Это осуществляется при помощи параметра STYLE, используемого при применении CSS с большинством стандартных тегов HTML. Данный метод нежелателен, т.к. приводит к потере одного из основных преимуществ CSS – возможности разделения информации и описания оформления информации.

```
<blockquote style="color:#CCFF66">Внимание!</blockquote>
```

Импортировании. В теге <STYLE> можно *импортировать* внешнюю таблицу стилей с помощью свойства @import таблицы стилей:

```
@import: url(styles.css);
```

Его следует задавать в начале стилевого блока или связываемой таблицы стилей перед заданием остальных правил. Значение свойства @import является URL файла таблицы стилей.

Заметим, что импортирование от связывания отличается тем, что при импортировании можно не только поместить внешнюю таблицу стилей в документ, но и поместить одну внешнюю таблицу стилей в другую.

```
<head>
<title>Untitled </title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251" />
<style type="text/css">
@import url('css/default.css');
</style>
</head>
```

Приведем пример использования свойства текста (рисунке 2.1)

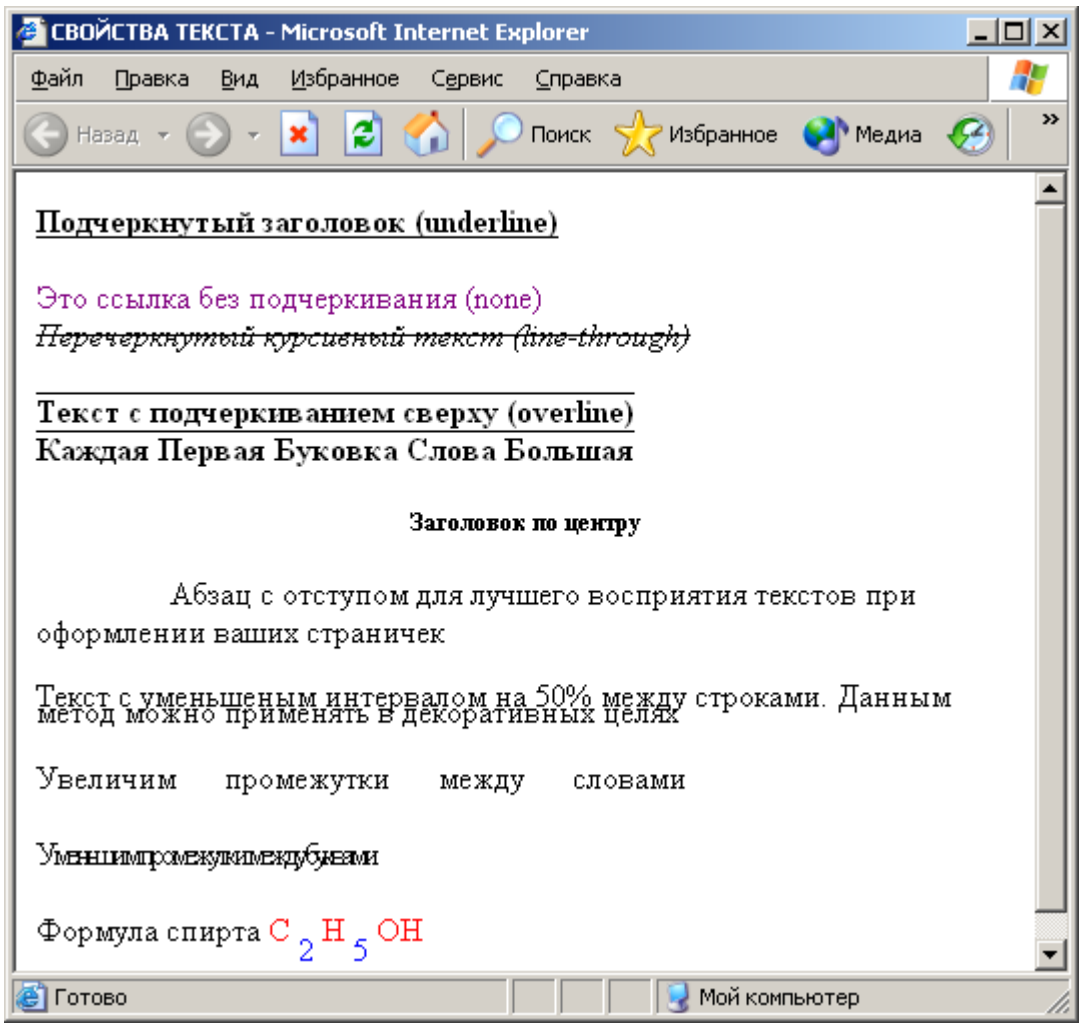


Рисунок 2.1

Ниже приведен код примера.

```
<STYLE type="text/css">
H4 {text-decoration: underline;}
A {text-decoration: none;}
i {text-decoration:line-through;}
b {text-decoration:overline;}
H5 {text-align: center}
b.cap {text-transform:capitalize;}
.otstup {text-indent: 50pt;}
.interval {line-height: 50 %}
</STYLE>
<h4>Подчеркнутый заголовок (underline)</h4>
<a href="/css/003/text.htm">Это ссылка без подчеркивания (none)</a><br>
<i>Перечеркнутый курсивный текст (line-through)</i><p>
<b>Текст с подчеркиванием сверху (overline)</b><br>
<b class=cap>каждая первая буковка слова большая</b>
<h5>Заголовок по центру</h5>
<p class=otstup>Абзац с отступом для лучшего восприятия текстов
при оформлении ваших страничек</p>
<p class=interval>Текст с уменьшеным интервалом на 50% между строками.
Данным метод можно применять в декоративных целях</p>
<p><span style="word-spacing: 15pt">Увеличим промежутки между
```

словами
<p>Уменьшим промежутки между
буквами
<p>Формула спирта
C
2
H
5
OH

Задание:

Получите у преподавателя дизайн HTML-документя и оформите его при помощи CSS. Все стили необходимо вынести в отдельный файл.

Контрольные вопросы

1. Для чего используются каскадные таблицы стилей?
2. Какими способами таблицы стилей связываются с элементами документа?
3. Каковы основные отличия импортирования от связывания?
4. Каким образом сделать так, чтобы изменялся цвет ссылок только внутри тэга <b>?

Лабораторная работа № 5-7

Язык составления сценариев JavaScript

Теоретические основы

JavaScript - это язык для составления сценариев, позволяющих выполнять разные действия непосредственно на машине пользователя. Располагаются данные сценарии внутри HTML документов.

JavaScript применяется для проверки правильности заполнения форм, создания удобной навигации и т.д.

Это язык программирования, который понятен браузеру. Это означает, что браузер умеет выполнять (интерпретировать) команды этого языка.

Программу на JavaScript можно помещать внутрь HTML-кода или держать в отдельном файле. Этот файл браузер прочитает (по специальной команде) во время интерпретации HTML-программы.

Программы на JavaScript (их называют скриптами) не работают самостоятельно. Коды JavaScript дополняют коды HTML и "живут" только вместе с ними. Даже если они расположены в отдельном файле

Размещение JavaScript на HTML-странице

Скрипт размещается между двумя парными тегами <**SCRIPT**>...</**SCRIPT**>. Обычно запись скрипта выглядит так:

<**SCRIPT** language=JavaScript>	Начало скрипта
<**!--**	Скрипт представлен как
...	HTML-комментарий, чтобы
Код на JavaScript	не "смущать" браузеры,
...	которые о скриптах не
//-->	знают.
</**SCRIPT**>	Конец скрипта

<**NOSCRIPT**>	Эта команда --
...	специально для
Для браузеров, которые	пользователей,
не поддерживают JavaScript	у которых
...	браузер не понимает
</**NOSCRIPT**>	скриптов.

Типы данных

JavaScript распознает следующие типы величин:

- числа, типа 42 или 3.14159;
- логические (Булевы), значения true или false;
- строки, типа "Howdy!";
- пустой указатель, специальное ключевое слово, обозначающее нулевое значение.

Преобразование типов данных

Тип переменной зависит от того, какой тип информации в ней хранится. *JavaScript* не является жестко типизированным языком. Это означает, что программист может не определять тип данных переменной, в момент ее создания. Тип переменной присваивается переменной автоматически в течение выполнения. Таким образом можно определить переменную следующим способом:

var answer = 42

А позже, можно присвоить той же переменной, например следующее значение:

answer = "Thanks for all the fish..."

Объявления переменных

Переменная должна быть объявлена до ее использования.

Для объявления используется ключевое слово **var**:

```
var x;      // переменная с именем "x".
var y = 5;  // описание с присвоением числа
var mes = "дядя Федор"; // описание с присвоением строки
```

Оператор цикла

for(нач; усл; прираш) команда

Команда "нач" выполняется один раз перед входом в цикл. Цикл состоит в повторении следующих действий:

- проверка условия "усл";
- выполнение команды "команда";
- выполнение команды "прираш".

Если условие ложно, цикл прекращается ("команда" и "прираш" после отрицательной проверки не работают).

```
// Произведение нечетных чисел массива
var set = new Array(1,2,3,4,5,6,7,8,9);
var p = 1;
for(var i=0; i<set.length; i++)
  if (set[i]%2) p *= set[i];
alert(p);
```

Условная команда

if (условие) команда1; **else** команда2;

или

if (условие) команда1;

Если условие принимает значение **true**, выполняется команда1, иначе команда2. В сокращенной форме ветвь **else** отсутствует.

```
// Абсолютное значение числа
var x = -25.456;
if (x < 0) x = -x;
alert(x);
```

Объектная модель JavaScript

JavaScript основан на простом объектно-ориентированном примере. Объект - это конструкция со свойствоми, которые являются переменными *JavaScript*. Свойства могут быть другими объектами. Функции, связанные с объектом известны как *методы* объекта.

В дополнение к объектам, которые сформированы в Navigator client и LiveWire server, вы можете определять ваши собственные объекты.

Объекты и Свойства

Объект *JavaScript* имеет свойства ассоциированные с ним. Обращаться к свойствам объекта необходимо следующей простой системой обозначений:

objectName.propertyName

И имя объекта и имя свойства чувствительны к регистру.

Например, пусть существует объект, с именем *myCar*. Можно задать свойства, именованные **make** и **year** следующим образом:

```
myCar.make = "Ford"
myCar.year = 69;
```

Также можно обратиться к этим свойствам, используя систему обозначений таблицы следующим образом:

mycar["make"] = "Ford
myCar["year"] = 69;

Функции и Методы

Функции - один из фундаментальных встроенных блоков в *JavaScript*. Функция - *JavaScript* процедура - набор утверждений, которые выполняют определенную задачу.

Определение функции состоит из ключевого слова **function** , сопровождаемого

- Именем функции;
- Списком аргументов функции, приложенной в круглых скобках, и отделяемые запятыми;
- *JavaScript* утверждениями, которые определяют функцию, приложенные в фигурных скобках, {...}.

Можно использовать любые функции, определенные в текущей странице. Лучше всего определять все функции в HEAD страницы. Когда пользователь загружает страницу, сначала загружаются функции.

Утверждения в функциях могут включать другие обращения к функции.

Например, есть функция с именем pretty_print:

function pretty_print(string) { document.write("
" + string) }

Эта функция принимает строку как аргумент, прибавляет некоторые теги HTML, используя оператор суммы (+), затем показывает результат в текущем документу.

Определение функции не выполняет ее. Для этого необходимо *вызвать* функцию, чтобы выполнить ее. Например, можно вызывать функцию pretty_print следующим образом:

< SCRIPT>
pretty_print("This is some text to display")
</ SCRIPT>

Аргументы функции сохраняются в таблице. Внутри функции, вы можете адресовать параметры следующим образом:

functionName.arguments *[i]*

Где *functionName* - имя функции, и *i* - порядковое число аргумента, начинающегося с нуля. Так, первый аргумент в функции, с именем myfunc, будет myfunc.arguments [0]. Общее число аргументов обозначено переменным arguments.length.

Определение Методов

Метод - функция, связанная с объектом. Метод определяется таким же образом, как и стандартная функция. Затем, используется следующий синтаксис, чтобы связать функцию с существующим бъектом:

object.methodname = function_name

Где *object* - существующий объект, *methodname* - имя, которое присвоено методу, и *function_name* - имя функции.

Можно вызывать метод в контексте объекта следующим образом:

object.methodname (params);

Создание Новых Объектов

И клиент и сервер *JavaScript* имеют строки предопределенных объектов. Кроме того, можно создавать собственные объекты. Создание собственного объекта требует двух шагов:

1. Определить тип объекта, написанной функции.
2. Создать образец объекта с **new**.

Чтобы определять тип объекта, необходимо создайть функцию для типа объекта, которая определяет его имя, и его свойства и методы. Например, пусть необходимо создавать тип объекта для автомобилей. Тип объектов будет назван *car*, и необходимо, чтобы он имел свойства для make, model, year, и color. Чтобы сделать это, необходимо написать следующую функцию:

```
function car(make, model, year) {
  this.make = make;
  this.model = model;
  this.year = year;
}
```

Замечание, используйте **this**, чтобы присвоить значения свойствам объекта, основанные на значениях функции.

Теперь можно создавать объект, с именем *mycar* следующим образом:

```
mycar = new car("Eagle", "Talon TSi", 1993);
```

Объект может иметь свойство, которое является самостоятельным другим объектом.

Можно определять методы для типа объекта включением определение метода на определении типа объекта. Например, пусть есть набор файлов изображений GIF, и необходимо определить метод, который показывает информацию для car, наряду с соответствующим изображением. Для этого необходимо определить функцию типа:

```
function displayCar() {
  var result = "A Beautiful " + this.year
          + " " + this.make + " " + this.model;
  pretty_print(result)
}
```

Где *pretty_print* - предопределенная функция, которая показывает строку. Используйте this, чтобы обратиться к объекту, который принадлежит методу.

Далее необходимо определить функцию методом из *car*, прибавляя утверждение

```
This.displayCar = displayCar;
```

к определению объекта. Так, полное определение *car* теперь выглядит так:

```
function car(make, model, year, owner) {
  this.make = make;
  this.model = model;
  this.year = year;
  this.owner = owner;
  this.displayCar = displayCar;
}
```

Новый метод можно вызывать следующим образом:

```
car1.displayCar ()
car2.displayCar ()
```

Использование this для Ссылок Объекта

JavaScript имеет специальное ключевое слово, **this**, которое используется, чтобы обращаться к текущему объекту. Например, есть функция с именем *validate* , которая проверяет правильность свойства значения объекта, данного объект, и high и low значения:

```
function validate(obj, lowval, hival) {
  if ((obj.value < lowval) || (obj.value > hival))
    alert("Invalid Value!")
}
```

Вызывать *validate* можно в каждом элементе формы обработчика событий onChange, используя **this**, как показано в следующем примере:

```
< INPUT TYPE = "text"
      NAME = "age"
      SIZE = 3
      onChange="validate(this, 18, 99)">
```

Вообще, метод **this** обращается к вызывающему объекту.

ОБЪЕКТНАЯ МОДЕЛЬ БРАУЗЕРА

С программистской точки зрения браузер представляет собой следующую иерархию объектов:

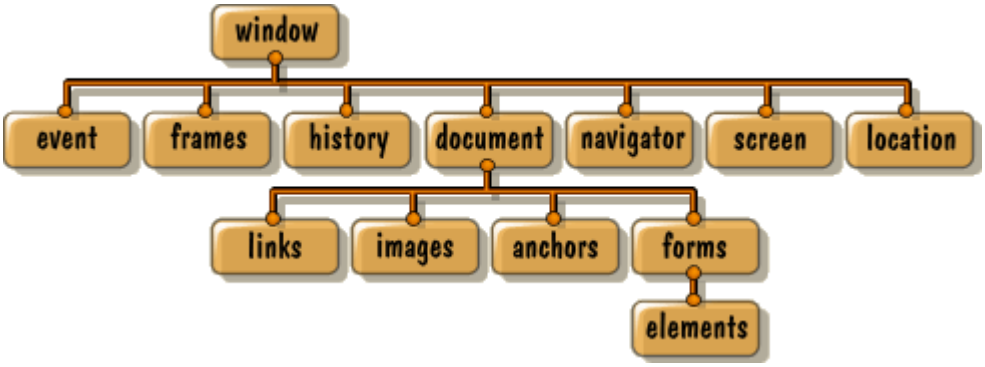


Рисунок 3.1

объект window

Объект window описывает текущее окно браузера и его содержимое.

Таблица 3.1– Свойства объекта

параметр	значение
self	Свойство self является указателем на текущее окно. Пример использования этого свойства показан в разделе перемещение фокуса.
opener	Свойство opener является указателем на окно родителя. Используя этот указатель, можно в созданном окне работать со всем тем, что принадлежит "родителю", например, использовать "родительские" скриптовые функции и переменные.
status	Свойства status и defaultStatus содержат

defaultStatus	соответственно временное и постоянное содержимое статусной строки. Временное содержимое меняется при наезде мышиного курсора на ссылку или при выдаче браузером диагностических сообщений.
dialogArguments	Свойство dialogArguments возвращает аргументы, проходящие через диалоговое окно, как массив. Смотрите пример к методу showModalDialog.
returnValue	Свойство returnValue определяет возвращаемое из окна значение. Смотрите пример к методу showModalDialog.

Таблица 3.2 – Методы объекта

метод	**описание**
open	Открывает новое окно браузера.
close	Закрывает окно браузера.
alert, prompt, confirm	Стандартные диалоговые панели.
showModalDialog	Отображает новое окно как модальную диалоговую панель.
blur	Уводит фокус из окна.
focus	Переводит фокус на окно.
scroll	Показывает документ в окне с заданными абсолютными смещениями от его начала.
scrollBy	Перемещает документ в окне на заданные величины по отношению к текущему положению.
setInterval	Указывает функции выполняться периодически через заданное количество миллисекунд.
clearInterval	Отменяет действие метода setInterval.
setTimeout	Запускает функцию через заданное количество миллисекунд.
clearTimeout	Отменяет действие метода setTimeout.

объект document

Объект document представляет собой модель документа, построенного браузером на экране.

Таблица 3.3– Свойства объекта

параметр	**значение**
title	Название документа, определенное в теге TITLE. В IE это свойство можно читать и менять, в NN -- только читать. <SCRIPT language="javascript"> <!-- var oldTitle=document.title; //--> </SCRIPT> <FORM> <INPUT type=button value=title onclick="alert(document.title)"> <INPUT name=t type=text value="Новое название">

	<INPUT type=button value="Изменить" onclick="document.title=this.form.t.value"> <INPUT type=button value="Восстановить" onclick="document.title=oldTitle"> </FORM>
URL	Адрес страницы. В IE это свойство можно читать и менять, в NN -- только читать. <FORM> <INPUT type=button value="URL" onclick="alert(document.URL)"> <INPUT name=u type="text" value="00.htm"> <INPUT type=button value="Изменить" onclick="document.URL=this.form.u.value"> </FORM>
location	Адрес страницы. Это свойство можно читать и менять как в IE, так и в NN. <FORM> <INPUT type=button value="location" onclick="alert(document.URL)"> <INPUT name=u type="text" value="00.htm"> <INPUT type=button value="Изменить" onclick="document.location=this.form.u.value"> </FORM>
lastModified	Дата последнего изменения документа. <DIV align=center> <P>Этот документ менялся последний раз: <P> <STRONG><FONT color=#FF4900> <SCRIPT language="javascript"> <!-- document.write(document.lastModified); //--> </SCRIPT> </FONT>></STRONG> </DIV>
bgColor fgColor linkColor alinkColor vlinkColor	Цвета фона, текста, не посещенной ссылки, активной ссылки и посещенной ссылки (аналоги атрибутов bgcolor, text, link, alink, vlink тега BODY). Эти свойства доступны для чтения и записи.
readyState	Возвращает значение complete после полной загрузки документа. Свойство поддерживается только в IE. <FORM> <INPUT type=button value="readyState" onclick="alert(document.readyState)"> </FORM>

Таблица 3.4 – Методы объекта

метод	**описание**

open()	Открывает запись в окно браузера. Прежнее содержимое окна очищается.
close()	Закрывает запись в окно браузера.
clear()	Очистка окна браузера.
write(str)	Записывает текст и код HTML, содержащийся в строке str в документ.
writeln(str)	Записывает текст и код HTML, заканчивающийся возвратом каретки (переходом на новую строку). Переход на новую строку на экране браузера будет заметен лишь в том случае, если он выполняется внутри тега PRE.

Методы write и writeln удобно использовать в следующих случаях:

Короткий скрипт заменяет собой длинный HTML-код.

Документ строится, учитывая особенности браузера, разрешения экрана, даты, предпочтения пользователя,...

Создание документов полностью программным путем ("на лету").

пример

Напишем функцию HR, которая будет выводить в документ необычную горизонтальную полоску:

```
<SCRIPT language="javascript">
<!--
 // Перевод числа num в 16-ричную систему счисленИЯ
 function ToHex(num)
 {
  var ret="";
  var s ="0123456789ABCDEF";
  while(num)
  {
    ret = s.charAt(num%16)+ret;
    num = Math.floor(num/16);
  }
  return !ret ? "0":ret;
 }

 // Формирование кода цвета в виде: #rrggbb
 function RGB(r,g,b)
 {
  r=ToHex(r); if (r.length<2) r = "0"+r;
  g=ToHex(g); if (g.length<2) g = "0"+g;
  b=ToHex(b); if (b.length<2) b = "0"+b;
  return "#"+r+g+b;
 }

 // Вывод горизонтальной полосы с растЯжкой серого цвета
 // win -- ссылка на окно, в которое выполняется вывод
 function HR(win)
 {
  var str="<TABLE border=0 cellspacing=0 cellpadding=0><TR>";
  for(var i=80; i<240; i+=4)
    str += "<TD width=12 bgcolor="+RGB(i,i,i)+"> </TD>";
  str += "</TR></TABLE>"
```

```
   win.document.write(str);
  }
HR(this);
//-->
</SCRIPT>
```

Результат

Задание 1

Написать сценарий на языке Javacript, позволяющий для изображения на web-странице менять ширину и высоту изображения, создавать рамку вокруг изображения, менять ее толщину и цвет, задавать альтернативный текст.

Задание 2

Написать сценарий на JavaScript, который реализует обмен рисунков на web-странице. Пусть на web-странице расположено четрые изображения, пронумерованных от 1 до 4. В текстовых полях указываются номера рисунков, которые необходимо поменять местами. Требуется, чтобы после нажатия на кнопку «Поменять местами» изображения переместились на нужные места.

Задание 3

Написать сценарий на JavaScript, который рассчитывает нагрузку преподавателя в часах. В анкете задать поля, в которые вводятся количество часов, отведенных на чтение лекций и проведение практических занятий, а также число студентов. Если по предмету читаются лекции, дополнительно планируется нагрузка: 10% времени от лекционных часов отводится на консультации, для приема экзамена планируется по 30 минут на человека. Если по предмету проводятся практические занятия, предусмотрена контрольная работа из расчета 15 минут на человека, зачет – из расчета 20 минут на человека.

Задание 4

Написать сценарий на JavaScript, который позволяет продемонстрировать, как будет меняться таблица и ее ячейки при изменении значений параметров border, cellspacing, cellpadding.

Задание 5

Приводятся данные о закупках пяти наименований товаров: цена за единицу и количество приобретаемых экземпляров. Напишите сценарий на JavaScript, определяющий сумму, затраченную на приобретениые товары. Определите, имеются ли товары, на которые потрачена одинаковая сумма, и сколько их. Постройте диаграмму, отражающую суммы, затраченные на приобретение разных товаров.

Контрольные вопросы

1. Каким образом JavaScrip добавляется в HTML-документ?
2. Что такое метод?
3. Каким образом объявляются объекты?
4. Назовите основные элементы, входящие в объектную модель броузера?

Лабораторная работа №8 [Знакомство с языком PHP]

Цель работы: Приобретение навыков программирования линейных, ветвящихся и циклических алгоритмов.

Задачи:

Изучить следующие базовые элементы языка:

1. Типы данных, идентификаторы, комментарии, объявление переменных.
2. Управляющие конструкции языка.

План:

1. Запуск Денвера.
2. Создание директории.
3. Выполнение и отладка примеров программ из экспериментальной части.
4. Выполнение заданий для самостоятельной работы.

1. Запустите Денвер. Для запуска используем ярлык Start Denwer. После выполнения лабораторной работы остановите работу Денвера с помощью ярлыка Stop Denwer.
2. Создайте в директории **C:\WebServers\home\localhost\www** папку, например, Ivanov (напишите свою фамилию). Все созданные вами файлы должны быть сохранены в этой директории.

Экспериментальная часть

Создание первой PHP- программы

Для начала напишем Web_страницу, которая будет отображаться в любом существующем браузере. Для этого выполните следующие действия.

1. Откройте программу Блокнот или любой другой доступный текстовый редактор и наберите в нем следующий HTML_код:

```
<html>
<head>
<title>Web-страница</title>
</head>
<body>
Этот текст появляется в окне браузера
</body>
</html>
```

2. Сохраните данный файл с именем simple01.htm.

3. Откройте файл в браузере. Вы получите страничку, в которой будет написано:

```
Этот текст появляется в окне браузера
```

4. В зависимости от инсталляции и настройки, загрузите файл в соответствующий каталог сервера (если Web_сервер работает на локальной машине, то файл следует просто скопировать в каталог, обслуживаемый Web_сервером). В нашем случае этим каталогом будет служить, созданная Вами ранее директория.

Снова откройте файл в браузере, используя HTTP_адрес для локального узла (**http:\\localhost\Ivanov\simple01.htm**). Результат должен выглядеть аналогично.

5. Замените строку, начинающуюся с ‘‘Этот текст ...’’, следующими строками:

```
Этот текст представляет данные, полученные в результате
работы PHP 5: Сегодня
<?php
$todaysdate = date("m",time()) . "-" . date("d",time()) . "-
" . date("Y",time());
echo $todaysdate;
?>
```

6. Сохраните файл, скопируйте его на сервер, если необходимо, и обновите страницу в браузере. Возможно, ничего не изменилось, если расширение файла не было изменено с .htm на .php. Очевидно, что для того чтобы Web_сервер передал данный файл PHP_процессору, во-первых, он (сервер) должен определить, что данный файл является PHP_файлом, а определить это можно по расширению файла. Во_вторых (предположим, что файл имеет соответствующее расширение), PHP_процессор выбирает для обработки разделы PHP_кода путем *синтаксического анализа* этого файла и поиска в нем PHP_тегов (<?php и ?>), а затем *выполняет* PHP_код. Синтаксический анализ означает, что PHP_процессор считывает отдельные команды и проверяет их на синтаксические ошибки. Под выполнением кода следует понимать просто фактическую обработку кода PHP_процессором.

7. Измените расширение файла на .php и введите адрес файла в браузере **http:\\localhost\Ivanov\simple01.php**, указав вместо директории Ivanov ранее Вами созданную. На этот раз код должен работать и в браузере должна появиться текущая дата в конце.

Управляющие конструкции

Примеры объявления переменных:

$a = 1 - целочисленный тип

$b = 2.45 - вещественный тип

$c = “string” - строковый тип

Комментарии:

// однострочный комментарий

это тоже однострочный комментарий

/* многострочный

комментарий */

Для программирования ветвящихся алгоритмов применяются условный оператор и оператор выбора.

Оператор if

```
if (выражение) блок_выполнения
```

Оператор else

```
if (выражение) блок_выполнения_1
else блок_выполнения_2
```

Задние 1. Из двух чисел x , y выбрать наибольшее.

Еще один способ расширения условного оператора if – использование оператора elseif.

Оператор elseif

```
if (выражение_1) блок_выполнения_1
elseif(выражение_2) блок_выполнения_2
...
else блок_выполнения_N
```

Операторов elseif может быть сразу несколько в одном if-блоке. Elseif-утверждение будет выполнено, только если предшествующее if-условие является False, все предшествующие elseif-условия являются False, а данное elseif-условие – True.

Пример 1.

```
<?php
$x=1;
// Используем if-else
if ($x == 0) {
    echo "x=0<br>";
} elseif ($x == 1) {
    echo "x=1<br>";
} elseif ($x == 2) {
    echo "x=2<br>";
}
```

Оператор выбора switch

Оператор выбора позволяет программировать ветвления по многим направлениям. Этот оператор организует переход на одну из нескольких ветвей в зависимости от значения заданного выражения (селектора выбора).

Структура:

```
switch (выражение или переменная)
{
  case значение_1:
    блок_действий_1
  break;
  case значение_2:
    блок_действий_2
  break;
  ...
  default:блок_действий_по_умолчанию
}
```

В отличие от if, здесь значение выражения не приводится к логическому типу, а просто сравнивается со значениями, перечисленными после ключевых слов case (значение_1, значение_2 и т.д.). Если значение выражения совпало с каким-то вариантом, то выполняется соответствующий блок_действий – от двоеточия после совпавшего значения до конца switch или до первого оператора break, если таковой найдется. Если значение выражения не совпало ни с одним из вариантов, то выполняются действия по умолчанию (блок_действий_по_умолчанию), находящиеся после ключевого слова default.

Задание 2. Модифицируйте **пример 1** с помощью оператора switch и сохраните под именем primer2.php

В PHP существует несколько конструкций, позволяющих выполнять повторяющиеся действия в зависимости от условия. Это циклы *while, do..while, foreach* и *for*.

Цикл с предусловием while

Структура:

```
while (выражение) { блок_выполнения }
```

либо

```
while (выражение): блок_выполнения endwhile;
```

Пример 2.

```
<?php
  //эта программа напечатает все четные цифры
  $i = 1;
  while ($i < 10)
  { // печатаем цифру, если она четная
    if ($i % 2 == 0) echo "$i <br>";
    // увеличиваем $i на единицу
    $i++;
  }
```

Цикл с постусловием do... while

Структура:

```
do { блок_выполнения } while (выражение);
```

Циклы do..while очень похожи на циклы while, с той лишь разницей, что истинность выражения проверяется в конце цикла, а не в начале. Благодаря этому блок_выполнения цикл do...while гарантированно выполняется хотя бы один раз.

Пример3.

```
<?php
  // эта программа напечатает число 12,
  // несмотря на то, что условие цикла не выполнено
  $i = 12;
  do
  { // если число четное, то печатаем его
    if ($i % 2 == 0) print $i;
    // увеличиваем число на единицу
    $i++;
  } while ($i<10)
?>
```

Цикл со счетчиком for

Структура:

```
for (выражение_1; выражение_2; выражение_3) {
блок_выполнения }
```

либо

```
for (выражение_1; выражение_2; выражение_3): блок_выполнения
endfor;
```

Пример 4. 1) Выведем все четные цифры с использованием цикла for таким образом:

```
<?php
  for ($i=0; $i<10; $i++)
  { // печатаем четные числа
    if ($i % 2 == 0) echo "$i <br>";
  } ?>
```

Используя оператор break, можно вызвать немедленное завершение цикла, пропуская условное выражение и любой остальной код в теле цикла. Когда программа встречает оператор break внутри цикла, она прекращает выполнение цикла, и управление передается оператору, следующему за циклом.

2) Опустите второе выражение (условие $i<10) и решите такую же задачу, останавливая цикл оператором **break.**

3) Модифицируйте скрипт **Примера 4** согласно данному условию: опустить все три выражения. В этом случае просто не будет задано начальное значение счетчика $i и оно не будет изменяться каждый раз в конце цикла. Все эти действия запишите в виде отдельных команд либо в блоке_выполнения, либо перед циклом.

4)В третье выражение конструкции for можно записывать через запятую сразу несколько простейших команд. Например, если мы хотим просто вывести все цифры, то программу можно записать совсем просто:

```
<?php
  // Если блок_выполнения не содержит команд
  // или содержит только одну команду,
  // фигурные скобки, в которые он заключен,
  // можно опускать
  for ($i=0; $i<10; print $i, $i++)
?>
```

5) Оператор **continue** – прерывает выполнение текущей итерации цикла. Следующий цикл выводит только нечетные числа в диапазоне от 1 до 10 (если $i четная, то мы с помощью continue переходим на следующую итерацию)

```
<?php
  for ($i=0; $i<=10;  $i++)
{ // печатаем нечетные числа
    if ($i % 2 == 0) continue;
 else echo "$i";
}
?>
```

Задания для самостоятельной работы

1. Получить у преподавателя номер варианта задания.
2. Выполнить задания 1, 2, 3 в соответствии со своим вариантом.
3. Смотри приложение №1

Задание 1. Программирование линейных алгоритмов:

1.Вычислить периметр и площадь прямоугольного треугольника по заданным длинам двух катетов a и b.

2. Заданы координаты трех вершин треугольника $(x1, y1)$, $(x2, y2)$, $(x3, y3)$. Найти его периметр и площадь.

3. Вычислить длину окружности и площадь круга одного и того же заданного радиуса R.

4. Вычислить расстояние между двумя точками с данными координатами $(x1, y1)$ и $(x2, y2)$.

5. Даны два действительных числа x и y. Вычислить их сумму, разность, произведение и частное.

6. Дана длина ребра куба. Найти площадь грани, площадь полной поверхности и объем этого куба.

7. Дана сторона равностороннего треугольника. Найти площадь этого треугольника, его высоты, радиусы вписанной и описанной окружностей.

8. Известна длина окружности. Найти площадь круга, ограниченного этой окружностью.

9. Треугольник задан величинами своих углов и радиусом описанной окружности. Найти стороны треугольника.

10. Найти сумму членов арифметической прогрессии, если известны ее первый член, знаменатель и число членов прогрессии.

Задание 2. Программирование ветвящихся алгоритмов:

1. Даны две точки $A(x1, y1)$ и $B(x2, y2)$. Составить алгоритм, определяющий, которая из точек находится ближе к началу координат.

2. Даны целые числа m, n. Если числа не равны, то заменить каждое из них одним и тем же числом, равным большему из исходных, а если равны, то заменить числа нулями.

3. Определить, равен ли квадрат заданного трехзначного числа кубу суммы цифр этого числа.

4. Подсчитать количество отрицательных чисел среди чисел a, b, c.

5. Подсчитать количество целых чисел среди чисел a, b, c.

6. Определить, делителем каких чисел a, b, c является число k.

7. Найти $\max\{\min(a, b), \min(c, d)\}$.

8. Даны действительные числа a, b, c. Удвоить эти числа, если $a < b < c$, и заменить их абсолютными значениями, если это не так.

9. Даны три положительных числа a, b, c. Проверить, могут ли они быть длинами сторон треугольника. Если да, то вычислить площадь этого треугольника.

10. Дан круг радиуса R. Определить, поместится ли правильный треугольник со стороной a в этом круге.

Задание 3. Программирование циклических алгоритмов:

1.Дано действительное число x. Вычислить:

$$y(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} - \frac{x^9}{9!}.$$

2. Даны действительное число a, натуральное число n. Вычислить:

$$P = a\,(a - n)\,(a - 2n) \times \ldots \times (a - n2).$$

3. Дано действительное x. Вычислить:

$$\frac{(x-1)(x-3)(x-7)\times\ldots\times(x-63)}{(x-2)(x-4)(x-8)\times\ldots\times(x-64)}.$$

4. Дано натуральное число n. Найти сумму первой и последней цифры этого числа.

5. Среди всех n-значных чисел указать те, сумма цифр которых равна данному числу k.

6. Дано натуральное число n. Переставить местами первую и последнюю цифры этого числа.

7. Дано натуральное число n. Переставить его цифры так, чтобы образовалось максимальное число, записанное теми же цифрами.

8. Составить программу, которая печатает таблицу умножения.

9. Найти все двузначные числа, сумма квадратов цифр которых кратна M.

10.Найти сумму всех n-значных чисел ($1 \le n \le 4$).

Приложение 1. Математические функции языка PHP

Библиотека математических функций PHP реализует методы для тригонометрических вычислений, числовых преобразований и числовых операций. Тригонометрические функции воспринимают параметры в радианах, но существуют функции преобразования градусов в радианы и наоборот.

Таблица №1. Математические функции языка PHP

Функция	Описание	Применение
abs( )	Абсолютное значение числа.	`echo abs(-0.7); // Выводит 0.7`
acos( )	Арккосинус, выраженный в радианах.	`echo acos(-0.7); // Выводит 2.3461938234056`
asin( )	Арксинус, выраженный в радианах.	`echo asin(-0.7); // Выводит -0.77539749661075`
atan( )	Арктангенс, выраженный в радианах.	`echo atan(-0.7); // Выводит -0.61072596438921`
atan2( )	Арктангенс для координат **x** и **y**, выраженный в радианах. Отличие от выражения **atan(y/x)** состоит в том, что знаки обоих параметров используются для определения квадранта результата.	`echo atan(-0.7/2); // Выводит -0.33667481938673 echo atan2(2, -0.7); // Выводит 1.9074711461816`
base_convert( )	Переводит число из одной системы счисления в другую. ргументы: переводимое число, система счисления, из которой переводят, система счисления, в которую переводят.	`echo base_convert(13, 10, 16); // Выводит d`
bindec( )	Десятичный эквивалент двоичной строки. Наибольшее конвертируемое число содержит 31 разряд, что соответствует 2147483647.	`echo bindec(11101011); // Выводит 235`
ceil( )	Округление числа в большую сторону.	`echo ceil(2.35); // Выводит 3`
cos( )	Косинус аргумента, выраженного в радианах.	`echo cos(2.35); // Выводит -`

Функция	Описание	Применение
		`0.70271307677355`
decbin( )	Двоичный эквивалент десятичного числа. Наибольшее конвертируемое число составляет 2147483647, или 31 разряд.	`echo decbin(235); // Выводит 11101011`
dechex( )	Шестнадцаричный эквивалент десятичного числа. Наибольшее конвертируемое число составляет 2147483647 или 7fffffff в шестнадцаричном выражении.	`echo dechex(235); // Выводит eb`
decoct( )	Восьмеричный эквивалент десятичного числа. Наибольшее конвертируемое число составляет 2147483647 или 17777777777 в восьмеричном выражении.	`echo decoct(235); // Выводит 353`
deg2rad( )	Преобразует градусы в радианы.	`echo deg2rad(90); // Выводит 1.5707963267949`
exp( )	Экспонента числа.	`echo exp(1); // Выводит 2.718281828459`
floor( )	Округление числа в меньшую сторону.	`echo ceil(2.99); // Выводит 2`
getrandmax( )	Максимальное число, которое может быть получено в результате вызова функции **rand( )**.	`echo getrandmax( ); // Выводит 32767`
hexdec( )	Десятичный эквивалент шестнадцаричного числа, представленного строкой. Наибольшее конвертируемое число составляет 7fffffff или 2147483647 в десятичном выражении.	`echo hexdec('eb'); // Выводит 235`
log( )	Натуральный логарифм	`echo log(exp(1)); // Выводит 1 echo log(2.718); // Выводит 0.99989631572895`
log10( )	Десятичный логарифм.	`echo log(1000);`

Функция	Описание	Применение
		`// Выводит 3`

max( )	Наибольшее значение из списка параметров. Возможно сравнение неограниченного кличества значений. В качестве параметра может быть задан массив	`echo max(12.23, 42.554, 58.234, 34.31); // Выводит 58.234`
		`$a = array(12.23, 42.554, 58.234, 34.31);`

min( )	Наибольшее значение из списка параметров. Возможно сравнение неограниченного количества значений. В качестве параметра может быть задан массив.	`echo min(12.23, 42.554, 58.234, 34.31); // Выводит 12.23`
		`$a = array(12.23, 42.554, 58.234, 34.31);`
		`echo min($a);`
		`// Выводит 12.23`

mt_rand( )	Случайное число. Для его получения используется генерация случайных чисел по методу Твистера. Необязательные аргументы указывают диапазон допустимых значений. Перед использованием этой функции необходимо установить начальное число с помощью функции **mt_srand()**.	

mt_srand( )	Устанавливает начальное число для генератора случайных чисел в соответствии с заданным параметром. Это позволяет получать различные результаты при вызове функции **mt_rand( )**.	`mt_srand(9);`
		`for ($i = 1; $i <= 10; $i++)`
		`echo mt_rand(1, 9)." "; //`
		`Выводит 7 3 8 7 1 4 4 9 3 5`

mt_getrandmax( )	Максимальное число, которое может быть получено в результате вызова функции **mt_rand( )**.	`echo mt_getrandmax( );` `// Выводит 2147483647`
number_format( )	Форматирует число. Аргументы: форматируемое число, число знаков после запятой, символ, используемый вместо	`a = 3456787 * log(2.718);` `echo number_format($a,`

Функция	Описание	Применение
	десятичной точки (необязательно), символ разграничения тысяч (необязательно).	`4, ",", "'"); // Выводит 3'456'428,5856`
octdec( )	Десятичный эквивалент восьмеричного числа, представленного строкой. Наибольшее конвертируемое число составляет 17777777777 или 2147483647 в десятичном выражении.	`echo octdec(353); // Выводит 235`
pi( )	Приближенное значение числа π.	`echo pi(); // Выводит 3.1415926535898`
pow( )	Возведение в степень. Аргументы: основание и показатель степени.	`echo pow(2, 3); // Выводит 8`
rad2deg( )	Преобразует радианы в градусы.	`echo rad2deg(pi()/4); // Выводит 45`
rand( )	Псевдослучайное число. Необязательные аргументы указывают диапазон допустимых значений. Если их не задавать, то число выбирается из диапазона от 0 до RAND_MAX. Перед использованием этой функции необходимо установить начальное число с помощью функции **srand( )**.	
round( )	Округление числа до ближайшего целого.	`echo round(6.45656); // Выводит 6`
sin( )	Синус аргумента, выраженного в радианах.	`echo sin(pi()/2); // Выводит 1`
sqrt( )	Квадратный корень числа.	`echo sqrt(121); // Выводит 11`
srand( )	Устанавливает заданное начальное число для генератора псевдослучайных чисел. Это позволяет получать различные результаты при вызове функции **rand( )**.	`srand(9); for ($i = 1; $i <= 10; $i++) echo rand(1, 9)." "; // Выводит 9 1 6 2 3 4 6 3 4 6`

Функция	Описание	Применение
tan( )	тангенс аргумента, выраженного в радианах.	`echo tan(pi()/4); // Выводит 1`

В языке PHP существуют также функции математических вычислений произвольной точности. Особенностью этих функций является параметр разрядности. Разрядность - количество знаков после десятичной точки в операндах и результате. По умолчанию принимается 0.

Таблица №2. Математические функции языка PHP

Функция	Описание	Применение
bcadd( )	Сумма первого и второго аргументов. Третий аргумент - параметр разрядности.	`echo bcadd(4.009, 4.009, 2); // Выводит 8`
bccomp( ) -	Числовое сравнение. Если первый аргумент больше второго, то возвращается +1. Если первый аргумент меньше второго, то возвращается -1. Если аргументы равны, то возвращается 0.	`echo bccomp(4.001, 4.009, 2); // Выводит 0 echo bccomp(4.01, 4.09, 2); // Выводит -1 echo bccomp(4.09, 4.01, 2); // Выводит 1 echo bccomp(4.01, 4.09, 2); // Выводит -1`
bcdiv( )	Частное от деления первого аргумента на второй.	`echo bcdiv(4.001, 2.009, 2); // Выводит 2 echo bcdiv(2.003, 3.009, 2); // Выводит 0.66 echo bcdiv(2.003, 3.009, 2); // Выводит 0.66`
bcmod( )	Остаток от деления первого аргумента на второй.	`echo bcmod(1525, 3); // Выводит 1 echo bcmod(1525.999, 3); // Выводит 1 echo bcmod(1525.999, 3.123); // Выводит 1`
bcmul( )	Произведение первого и второго аргументов. Третий аргумент - параметр разрядности.	`echo bcmul(2.008, 4.009, 2); // Выводит 8.00`
bcpow( )	Возведение в степень. Аргументы: основание, показатель степени, параметр	`echo bcpow(2.005, 3, 2); // Выводит 8.00 echo bcpow(2.25, 3, 2); // Выводит 11.39`

Функция	Описание	Применение
	разрядности. Показатель степени не должен содержать знаков после десятичной точки.	

bcscale( )	Значение параметра разрядности, который будет использоваться по умолчанию.	`echo bcpow(2.25, 3); // Выводит 8`
		`echo bcscale(2);`
		`echo bcpow(2.25, 3); // Выводит 11.39`

bcsqrt( )	Квадратный корень числа.	`echo bcsqrt(121.00, 2); // Выводит 11.00`
		`echo bcsqrt(121.75, 2); // Выводит 11.03`

bcsub( )	Разность. Из первого аргумента вычитается второй.	`echo bcsub(4.001, 2.009, 2); // Выводит 2.00`
		`echo bcsub(2.03, 3.09, 2); // Выводит -1.06`

Лабораторная работа № 9 [Обмен информацией между Web-сервером и клиентом]

Цель работы: Приобретение навыков создания программ для организации обмена информации между WEB-сервером и клиентом

Задачи:

1. Изучить конструкции HTML для организации обмена между сервером и клиентом
2. Разобрать методы передачи данных GET и POST
3. Изучить возможности PHP для получения информации от пользователя

План:

1. Запуск Денвера.
2. Ознакомление с содержимым теоретической части.
3. Выполнение и отладка примеров программ из экспериментальной части.
4. Выполнение заданий для самостоятельной работы.

Теоретическая часть

Получение данных от клиента

Web-сайт — это почти всегда диалог. Конечно, встречаются "односторонние" сайты, авторы которых стремятся только показать, но не услышать отзыв о показанном. Но даже там редко обходится без ссылки на автора: "Все, что вы думаете по этому поводу, пишите сюда".

Но чаще "сайтовладелец" желает получать о своих посетителях гораздо больше информации. Речь пойдет о способах получения информации от

самих пользователей, — например, анкетных данных для вступления в виртуальный клуб или мнений по интересующему вас вопросу.

Как получить данные и передать их для обработки?

Для этой цели используются *формы — это совокупность стандартных HTML-конструкций ввода текстовой и прочей информации и программы-обработчика этой информации, работающей на Web-сервере*. Иными словами, пользовательская форма (или HTML-форма) служит для передачи информационных данных серверу.

Результат конструкций языка разметки HTML интерпретируется браузером, с помощью которого пользователь электронного документа получает информацию. Таким образом, объединив все эти формулировки, можно сказать, что HTML-форма выступает в роли посредника между пользователем и сервером.

Посетитель Web-страницы вводит в HTML-форму определенные данные, которые обрабатываются программой и отсылаются на Web-сервер. Все эти действия укладываются в три стадии:

1. Ввод пользователем информации.
2. Обработка введенной информации программой, установленной на сервере.
3. Получение результата отправления введенной информации на Web-сервер (открытие нового HTML-документа, переадресация на предыдущую страницу и пр.).

В качестве программы-обработчика чаще всего выступает CGI-сценарий (скрипт, который обычно разрабатывается на языке Perl или C/C++ и который взаимодействует со специальным компонентом Web-сервера — Common Gateway Interface) или программы, написанные на основе таких серверных языков программирования, как PHP, ASP, JSP и др.

Значение пользовательских форм трудно переоценить — они являются особым средством HTML, дающим посетителю возможность не только пассивно просматривать информацию, но и быть задействованным в

содержании Web-сайта. Такое свойство принято называть интерактивностью, которая на сегодняшний день встречается практически во всех электронных документах.

Основная схема формы:

Подобно фреймам, таблицам и другим "крупногабаритным" элементам Web-страницы, форма — это блок HTML-кода, образованный специальными элементами HTML. Границами такого блока служат, как легко догадаться, дескрипторы <FORM>:

```
<FORM параметры>
</FORM>
```

Экспериментальная часть

Листинг 1. Добавление формы в документ

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1251">
<title>Формы</title>
</head>
<body>
<form>
 <p>Здесь размещаются элементы формы</p>
</form>
</body>
</html>
```

Внутри формы могут располагаться следующие элементы интерфейса:

- поля ввода;
- скрытые поля ввода;
- кнопки;
- переключатели;
- флажки;
- выпадающие списки.

Для работы в форме необходимо указать два атрибута**:** `action` — пусть к скрипту, который будет обрабатывать данные, и `method` — способ передачи данных.

Как правило, для отправки информации на сервер (т.е. передачи в php-скрипт) пользователь должен нажать кнопку `<input type="submit" value="Текст кнопки" />`.

Пример 1.Рассмотрим типичную форму, пусть это будет форма ввода логина и пароля на сайте:

Листинг 2. Html-форма primer.html

```
<form action="obrabotchik1.php" method="get"><div>
Логин: <input type="text" value="" name="login"/>
Пароль: <input type="password" value=""
name="password"/>
<input type="submit" value="Проверить" name="button"/>
</div></form>
```

Задание 1. Создайте HTML-документ добавьте в него HTML-код, представленный в Листинге 2. Сохраните данный HTML-документ под именем primer.html. Просмотрите результат в окне браузера.

То, какой из доступных в HTML элементов `<input>` будет представлен на странице, определяется атрибутом `type` (по умолчанию он равен text, что означает «поле ввода»).

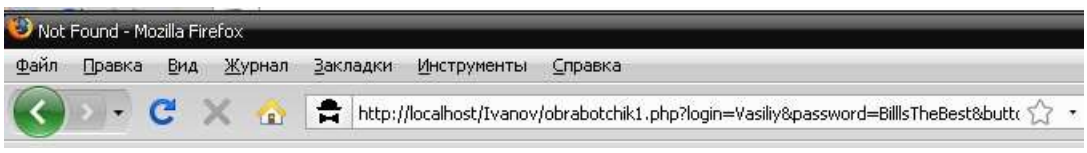
От каждого элемента `<input>`на сервер будут переданы значения двух атрибутов: name (имя элемента) и `value` (значение), т.е. на самом деле на сервер передаются ПЕРЕМЕННЫЕ.

Переменные могут переданы двумя методами: GET и POST. Если метод явно не задан в теге `<form>`, то будет выбран GET.

Метод GET основан на том, что все переменные передаются непосредственно в адресной строке: после полного адреса ставится знак вопроса и перечисляются переменные. Сразу же стоит заметить, что длина URI (адреса) ограничена, а также вся передаваемая информация легко доступна непосредственно в адресной строке.

Задание 2. Используя указанную выше форму, введите в поле `Логин` (name="Login") значение "Vasiliy", а в поле `Пароль` (name="password") значение "BillisTheBest" и нажмите кнопку (name="button" type="submit").

Вы увидете, что браузер автоматически перейдёт на страницу с адресом:



Т.е. серверный сценарий (файл obrabotchik1.php) получит от клиента три переменных:

```
login=Vasiliy
password=BillisTheBest
button=Проверить
```

На самом деле в запросе передаётся намного больше так называемых встроенных переменных, например: информация о браузере пользователя, ip-адрес, предыдущая посещённая страница, протокол используемый клиентом и так далее...

Чтобы получить значение переменных в сценарии PHP нужно обратиться к массиву$_GET["имя переменной"].

Пример 2. Для проверки правильности ввода логина и пароля используем следующий php-код:

Листинг 3. obrabotchik1.php

```
<?
$userLogin = $_GET["login"];
$userPassword = $_GET["password"];
if ($userLogin=="Vasiliy" &&
$userPassword=="BillisTheBest")
echo "Здравствуйте, Василий! Логин и Пароль верны.";
else
echo "Ошибка в вводе Логина или Пароля. Василий вы
забыли пароль?";
?>
```

Это был простейший пример, иллюстрирующий работу метода GET.

Задание 3. Сохраните php-код, представленный в Листинге 3, как obrabotchik1.php**.** Откройте форму primer.html в окне браузера и просмотрите результат работы php-кода obrabotchik1.php.

Метод POST, в отличие от метода GET, передаёт все переменные непосредственно в теле запроса. Это и является его основным отличием от GET. Вы можете передать данные скрытно. Кроме того, метод POST позволяет отправить намного больше информации, не ограничиваясь максимально допустимой длинной адресной строки. Изменим атрибут method тега form на POST и посмотрим что произошло: при нажатии кнопки «Проверить» браузер переходит на страницу с адресом login.php и никаких переменных в адресной строке не передаётся.

В сценариях PHP используем массив $_POST["имя переменной"].

Пример 3. Модифицируем пример php-кода obrabotchik1.php

Листинг 4. obrabotchik2.php

```
<?
$userLogin = $_POST["login"];
$userPassword = $_POST["password"];
if ($userLogin=="Vasiliy" &&
$userPassword=="BillisTheBest")
echo "Здравствуйте, Василий! Логин и Пароль верны.";
else
echo "Ошибка в вводе Логина или Пароля. Василий вы
забыли пароль?";
?>
```

Задание 4. Сохраните php-код, представленный в Листинге 4, как obrabotchik2.php

Задание 5. В созданном ранее HTML-файле primer.html замените

```
action="obrabotchik1.php" method="get"
```

на

```
action="obrabotchik2.php" method="post"
```

и сохраните как primer2.html. Просмотрите результат в окне браузера. Таким образом, рассмотрены два основных метода получения информации от клиента, используемые в PHP-скриптах. Присмотритесь к адресным строкам внимательнее!

Элементы формы

Постановка задачи:

A. Разработать HTML-форму для возможности получения от пользователя следующих данных:
   1. ФИО – текстовое поле
   2. пол (мужской, женский) – радио-кнопки
   3. семейное положение (холост, женат/замужем) – радио-кнопки
   4. образование (среднее, высшее и т.п.) – выпадающий список
   5. хобби (спорт, охота, рыбалка, компьютеры, ТВ и т.п.) – список с множественным выбором
   6. есть ли компьютер – кнопка-переключатель

B. Разработать PHP-скрипт, анализирующий введенные данные и генерирующий страничку вида:

   Иванов Иван Иванович

   Пол: *мужской*

   Семейное положение: *холост*

Образование: *высшее*

Хобби: *охота, рыбалка*

Наличие ПК дома: *есть*

A. Разработка данной HTML-формы:

Выравнивание элементов формы:

С помощью таблиц удобно определять положение полей формы, особенно, когда они перемежаются с текстом.

Таблица в HTML:

<TABLE></TABLE> - тело таблицы

<TR></TR> - строка таблицы

<TD></TD> - ячейка таблицы

Рассмотрим все элементы формы по-порядку:

Элемент INPUT является наиболее употребительным тегом HTML-форм. С помощью этого тега реализуются основные функции формы. Он позволяет создавать внутри формы поля ввода строки текста, имени файла, пароля и.т.д.

Обратите внимание на особенность INPUT - у него нет конечного (завершающего) тега. Атрибуты и особенности использования INPUT зависят от способа его использования. Рассмотрим эти способы.

Текстовое поле (type=text)

Это элемент текстовой строки. Строки для ввода текста на HTML-страницах встречаются постоянно.

Тому, кто хоть раз пользовался поисковым Internet-сервером, не надо объяснять, что это такое: узкий вытянутый прямоугольник, внутри которого можно ввести с клавиатуры одну строку текста

Формат тега INPUT для создания поля ввода текстовой строки:

```
<input type=text name=имя_параметра [value=значение]
[size=размер_поля] [maxlen=длина_поля]>
```

Данный тег создает поле ввода с максимально допустимой длиной текста maxlen и размером в size знакомест. Если указан атрибут value, то в поле будет изначально отображаться значение данного атрибута. В квадратных скобках [] помечены необязательные атрибуты.

Листинг 5.1 Создание текстовых полей

```
<form action="obrabotchik.php" method="post">
  <table>
    <tr>
      <td align="right"><label for="lastname">Фамилия:
</label>
      <input type = “text” name="lastname" size="20"
value=""></td>
    </tr>
    <tr>
      <td align="right"><label for="firstname">Имя:
<input type = “text” name="firstname" size="20"
      value=""></label></td>
    </tr>
    <tr>
      <td align="right"><label
for="patronymic">Отчество: </label><input type = "text"
      name="patronymic" size="20" value=""></td>
    </tr>
```

```
    <tr>
      <td><br>
      </td>
 </tr>
</table>
</form>
```

Задание 6.1.Сохраните данную форму как forma.html и просмотрите результат в окне браузера.

Списки-переключатели (type=radio)

В списках-переключателях слева от пунктов имеются кружки, причем в центре кружка, соответствующего выбранному пункту, появляется жирная точка (radiobutton).

Для создания таких списков используется тот же дескриптор <INPUT>, что и для списка вариантов, но параметру type присваивается значение radio. Как и в списке вариантов, каждому элементу списка-переключателя соответствует отдельный дескриптор <INPUT>. Однако, в отличие от списка вариантов, все элементы списка-переключателя имеют одно имя. Иначе и не может быть: ведь мы хотим получить только одно значение из всего перечня. Наконец, здесь, как и в списке вариантов, для выбора одного из пунктов по умолчанию используется параметр checked — это элемент опции переключения между различными вариантами выбора.

В отличие от CHECKBOX, вариант выбора может быть только один. Также имеет дополнительные параметры NAME, VALUE и CHECKED (показывает, что кнопка помечена), действие которых аналогично опции выбора CHECKBOX (о теге CHECKBOX рассказывается далее).

Листинг 5.2. Создание переключателей radio

```
    <tr>
      <td>Ваш пол:</td>
    </tr>
    <tr>
      <td><input type="RADIO" CHECKED name="pol"
value="Мужской"> Мужской </td>
    </tr>
    <tr>
      <td><input type="RADIO" name="pol"
value="Женский"> Женский </td>
    </tr>
    <tr>
      <td><br>
      <br>
      </td>
    </tr>
    <tr>
      <td>Семейное положение: </td>
    </tr>
    <tr>
      <td><input type="RADIO" CHECKED name="family"
value="1"> Холост (не
      замужем)<br>
      </td>
    </tr>
    <tr>
```

```
      <td><input type="RADIO" name="family"
value="2">Женат (Замужем)<br>
      </td>
    </tr>
    <tr>
      <td><input type="RADIO" name="family" value="3">
Вдовец (вдова)<br>
      </td>
    </tr>
    <tr>
      <td><input type="RADIO" name="family" value="4">
Разведён(а)<br>
      </td>
    </tr>
    <tr>
      <td><br>
      </td>
    </tr>
```

Задание 6.2. Перед тегом </table> html-документа forma.html вставьте html-код, представленный в Листинге 5.2., сохраните и просмотрите результат в окне браузера.

Выпадающий список (select)

Тэг <select> представляет собой выпадающий или раскрытый список, при этом одновременно могут быть выбраны одна или несколько строк. Но будет передано значение последней выбранной кнопке.

Список начинается с парных тегов <select></select>. Теги <option></option> позволяют определить содержимое списка, а параметр value определяет значение строки. Если в теге <option> указан параметр selected, то строка будет изначально выбранной. Параметр size

задает, сколько строк будет занимать список. Если size равен 1, то список будет выпадающим. Если указан атрибут multiple, то разрешено выбирать несколько элементов из списка. Но эта схема практически не используется, а при size = 1 не имеет смысла.

Листинг 5.3. Создание выпадающего списка (select)

```
<tr>
      <td>Образование :<select name="edu" size="1">
        <option selected value="Высшее">Высшее</option>
        <option value="Незаконченное
высшее">Незаконченное высшее</option>
        <option value="Среднее полное">Среднее
полное</option>
        <option value="Среднее неполное">Среднее
неполное</option>
        <option value="Нет образования">Нет</option>
      </select><br>
      </td>
    </tr>
    <tr>
      <td><br>
      </td>
    </tr>
    <tr>
      <td>Хобби :<br>
      <select name="hobby[]" < size="7" multiple>
        <option selected value="Охота">Охота</option>
        <option selected
value="Рыбалка">Рыбалка</option>
        <option value="Спорт">Спорт</option>
        <option value="Компьютеры">Компьютеры</option>
```

```
        <option value="TV и Video">TV и Video</option>
        <option value="Книги">Книги</option>
        <option value="Нет">Нет</option>
      </select> </td>
    </tr>
    <tr>
      <td><br>
      </td>
    </tr>
```

Задание 6.3.Перед тегом </table> html-документа forma.html вставьте html-код, представленный в Листинге 5.3, сохраните и просмотрите результат в окне браузера.

Списки вариантов (type=checkbox)

Обычно пункты списков вариантов снабжены квадратными "окошками", в которых при выборе появляются "птички" (checkbox). Для создания такого списка используется уже знакомый нам дескриптор <INPUT> с параметром type=checkbox

.

Для создания списка вариантов нужно столько дескрипторов <INPUT>, сколько в нем есть вариантов. Однако имена у них должны быть разные — у каждого варианта свое. Зачем это нужно? Дело в том, что, хотя логически список является цельным элементом формы, с точки зрения кода это всего лишь набор разрозненных, несвязанных дескрипторов. У каждого — свое имя и значение. А для того чтобы стало ясно, что именно нам предлагают выбрать, нужно снабдить эти квадратики пояснениями. Пользователь может выбирать несколько вариантов поля CHECKBOX, значение каждого из которых будет передано программой-обработчиком на Web-сервер.

Тег может содержать ряд дополнительных параметров.

Параметр	Описание
NAME	Указание общего для всех вариантов выбора идентификационного имени
VALUE	Определение значения для конкретного варианта выбора (обязательный параметр). Не должен повторяться, т. к. при установке флажка передается на Web-сервер
CHECKED	Данный вариант является выбранным по умолчанию Значений он не имеет.

Листинг 5.4. Создание checkbox.

```
<tr>
      <td><input type="CHECKBOX" name="computer"
checked value="ON"> Наличие компьютера </td>
    </tr>
  </table>
```

Задание 6.4. Перед тегом `</table>` html-документа forma.html вставьте html-код, представленный в Листинге 5.4, сохраните и просмотрите результат в окне браузера.

Кнопка отправки формы (type=submit)

Это кнопка отправления пользовательских данных на Web-сервер. При нажатии на нее запускается программа-обработчик, которая анализирует введенные пользователем данные и отправляет результат на сервер.

Для элемента отправления данных на сервер также могут использоваться дополнительные (необязательные) параметры — NAME (имя кнопки отправления, значение которого также будет передано на сервер) и VALUE (присвоение собственного имени кнопке).

Синтаксис кнопки отравки данных:

```
<input type=submit [name=go] value=Отправить>
```

Кнопка сброса формы (type=reset)

Это кнопка сброса введенных пользователем данных HTML-формы. При нажатии на нее восстанавливаются все установленные по умолчанию значения полей формы. Элемент не обязателен, однако в ряде случаев весьма полезен. При работе с многочисленными текстовыми строками и опциями выбора пользователь может, допустив ошибку, пожелать заново заполнить форму, тогда ему придется либо перезагружать страницу, либо вручную удалять текст из каждого поля формы. Кнопка сброса в этом случае позволит добиться желаемого при одном нажатии ранее.

Синтаксис кнопки сброса:

```
<input type=reset value=Сброс>
```

Листинг 5.5. Кнопки отправки и сброса формы.

```
  <div align="center"><center><p><input type="submit"
value="Отправить">
<input type="reset" value="Сброс"> <br>
  </p>
  </center></div>
```

Задание 6.5. После тега `</table>` html-документа forma.html вставьте html-код, представленный в Листинге 5.5, сохраните и просмотрите результат в окне браузера.

В. Разработка php-скрипта, анализирующий введенные данные и генерирующий страничку вида, указанной в постановке задаче.

Листинг 6. PHP-скрипт obrabotchik.php.

```
  <?php
    print "<CENTER><FONT size=10>Получены
данные:</FONT></CENTER><BR><BR><BR><BR>";

     print "<TABLE width=400 align=\"left\" border=1";
     echo  "<TR><TD width=45%>Ф.И.О. :</TD><TD>",
$_POST["lastname"], " ", $_POST["firstname"], " ",
$_POST["patronymic"], "</TD></TR>";
     echo  "<TR><TD>Пол:</TD><TD>",
$pol=$_POST["pol"],"</TD></TR>";
     print  "<TR><TD>Семейное положение:</TD><TD>";
     $family=$_POST["family"];
     if ($family==1) print
($pol=="Мужской")?("Холост"):("Не замужем");
     /*$pol = условие ? true : false ;
    Тоже самое что if () { .. } else {.. }*/
     if ($family==2) print
($pol=="Мужской")?("Женат"):("Замужем");
     if ($family==3) print
($pol=="Мужской")?("Вдовец"):("Вдова");
```

```
     if ($family==4) print
($pol=="Мужской")?("Разведён"):("Разведена");

     echo "</TD></TR><TR><TD>Образование:</TD><TD>",
$_POST["edu"], "</TD>", "</TR>";

     print "<TR><TD>Хобби:</TD><TD>";
     $hobby=$_POST["hobby"];
     if (in_array("Нет", $hobby)) print "Хобби
отсутствует";
      //in_array - возвращает TRUE, если значение
существует в массиве.
        else for ($i=0; $i<count($hobby); $i++) echo
$hobby[$i], "<BR>";

     echo "</TD></TR><TR><TD>Компьютер:</TD><TD>",
isset($_POST["computer"])?("есть"):("нет"),
"</TD></TR>";
     /* Функция isset() используется для определения,
присвоено ли переменной какое-либо значение.
      Если значение присвоено, функция возвращает
true.*/
       print "</TABLE>";

   ?>
```

Задание 7. Проанализируйте данный php-код, представленный в Листинге 6, сохраните как obrabotchik.php и просмотрите результат работы в окне браузера.

Задания для самостоятельной работы

1. Выполнить задания 1, 2 в соответствии со своим вариантом.
2. Смотри приложение №2

Задание 1: Разработать приложение, в котором:

1. Создается форма form.htm для введения пользователем данных:
2. PHP-сценарий HandleForm.php получает данные с формы .
3. Отображает извлеченные из формы данные в окне браузера.
4. Сценарий генерирует отправку на еще один PHP-файл, который отображает персональное приветствие пользователю.

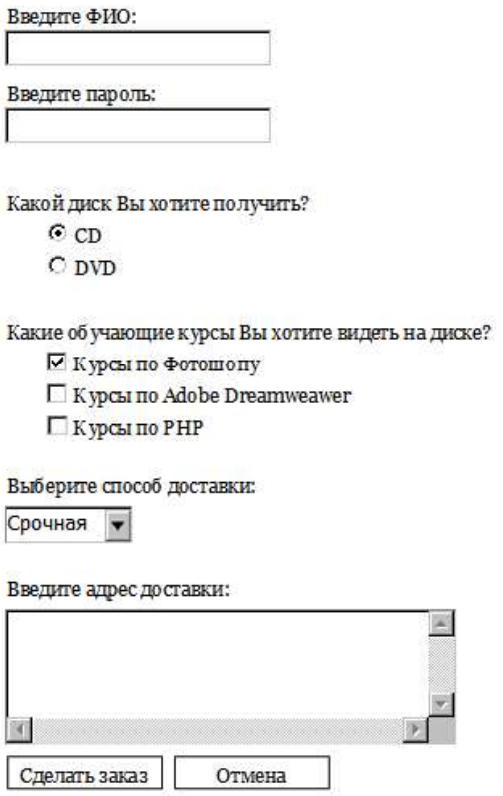
Описание элементов формы таких как:

- кнопка с изображением (type=image),
- поле ввода пароля (type=password),
- скрытое текстовое поле (type=hidden),
- многострочное поле ввода текста (textarea),
- кнопка для загрузки файлов (type=file),

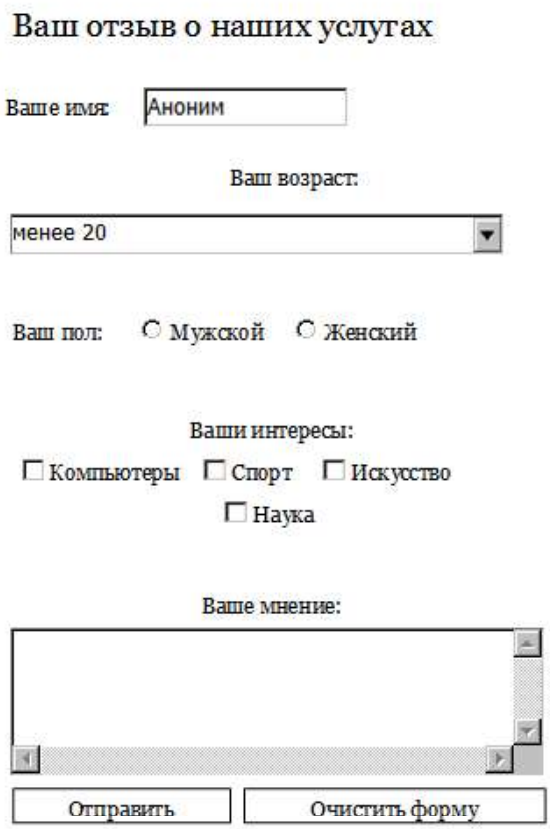
имеются в Приложении №2 данной лабораторной работы.

Обмен информацией между Web-сервером и клиентом

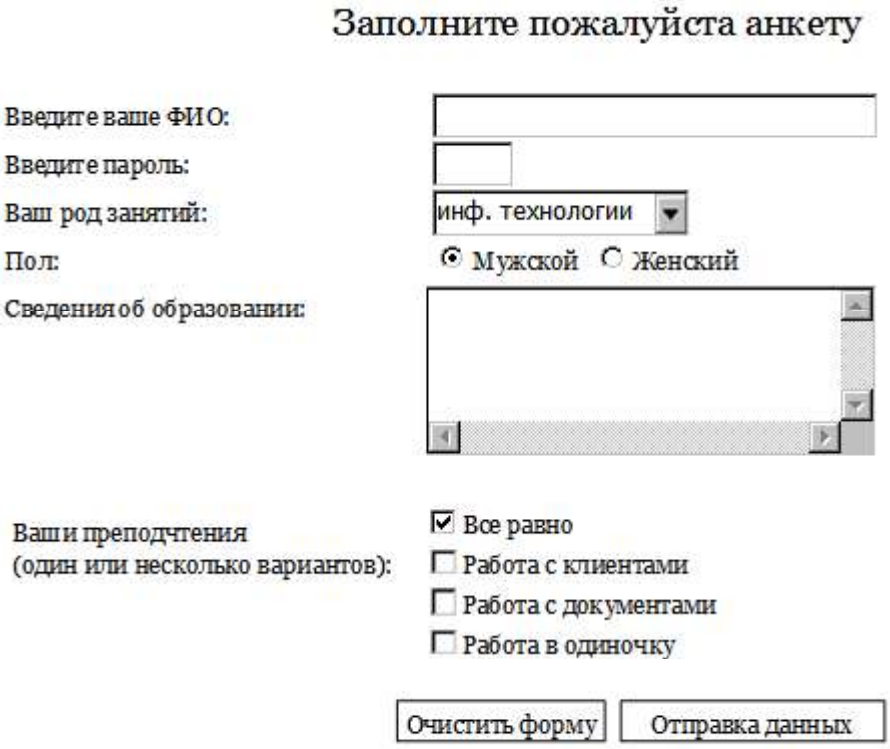
Вариант 1.



Вариант 2



Вариант 3.



Вариант 4.

Дорогие друзья!

Никогда не стоит забывать , что периоды плодотворной работы время от времени должны перемежаться периодами не менее плодотворного отдыха. Пожалуйста заполните нашу анкету. Быть может они поможет Вам настроиться на отпускную волну :-)

Друзья называют Вас:

С кем Вы хотели бы провести отпуск: с попугаем

Куда бы Вы предпочли отправиться в наступающем отпускном сезоне? на Кудыкину гору

Как Вы предполагаете добираться к месту отдыха:

- Кривая вывезет
- Язык до Киева доведет
- Мы поедем, мы помчимся, на оленях утром ранним

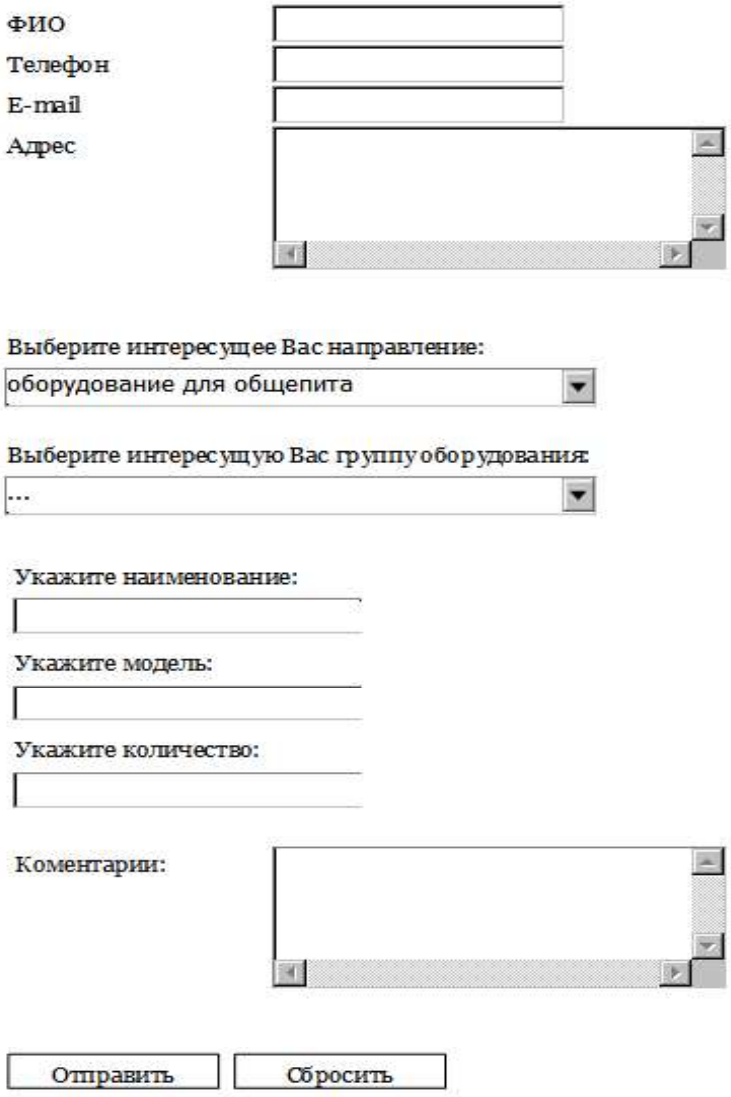
Сколько лет Вы так чудесно отдыхаете?

Ваш E-mail адрес:

Отправить

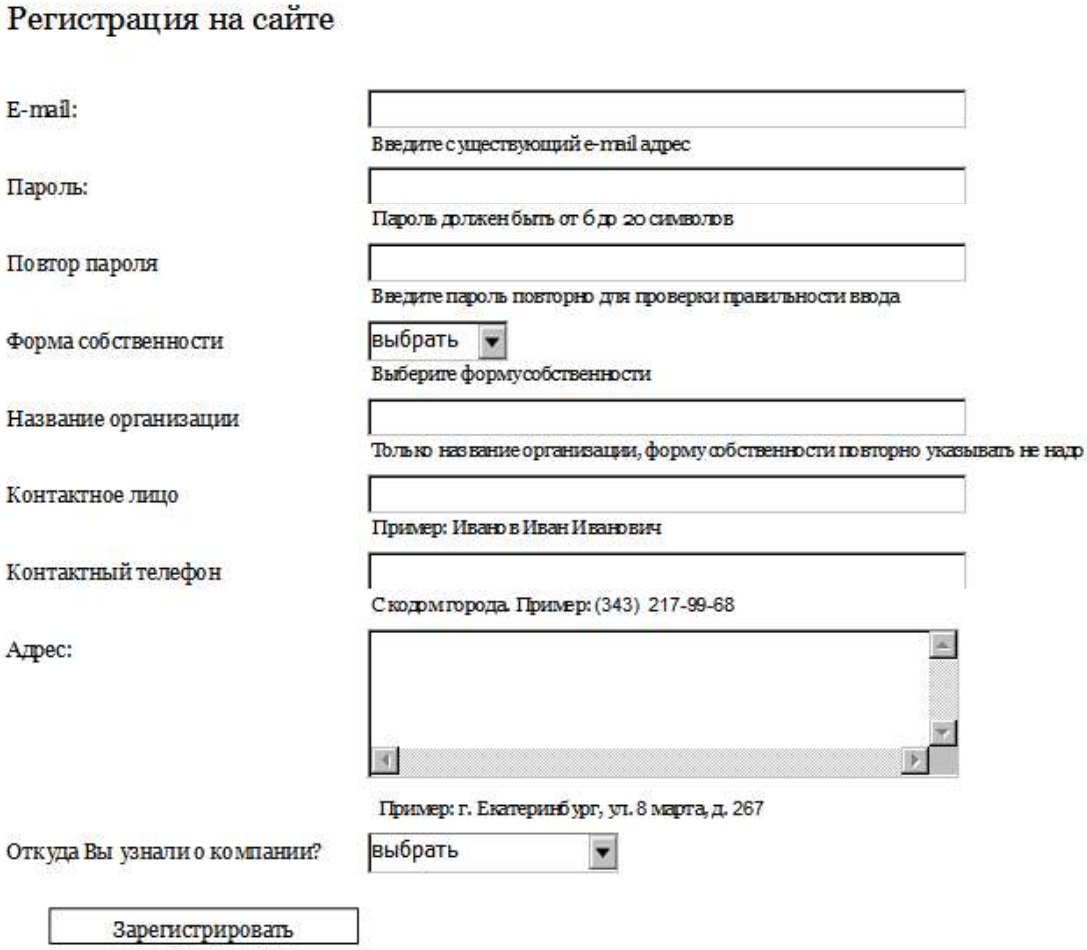
Обмен информацией между Web-сервером и клиентом

Вариант 5



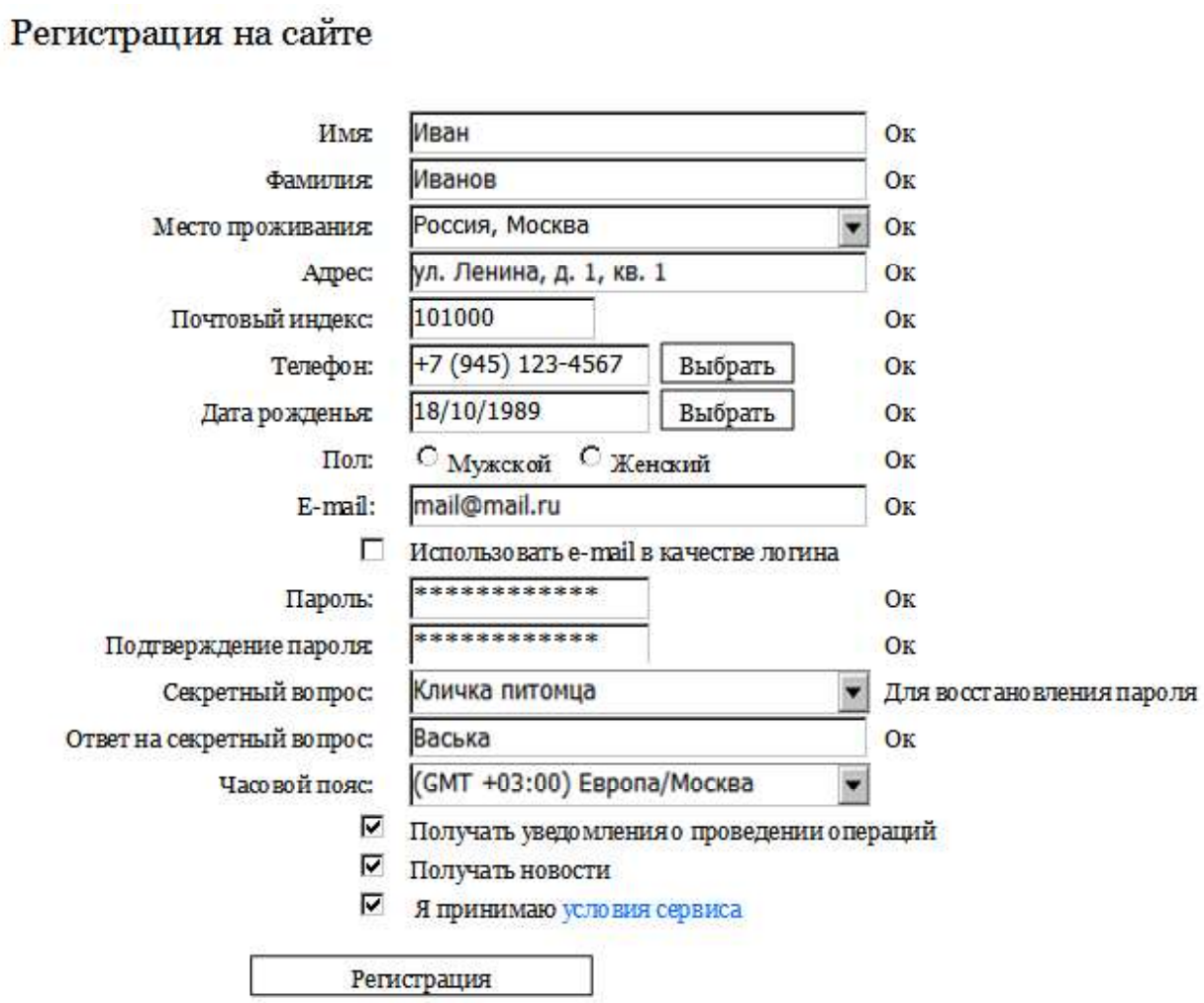
Обмен информацией между Web-сервером и клиентом

Вариант 6.



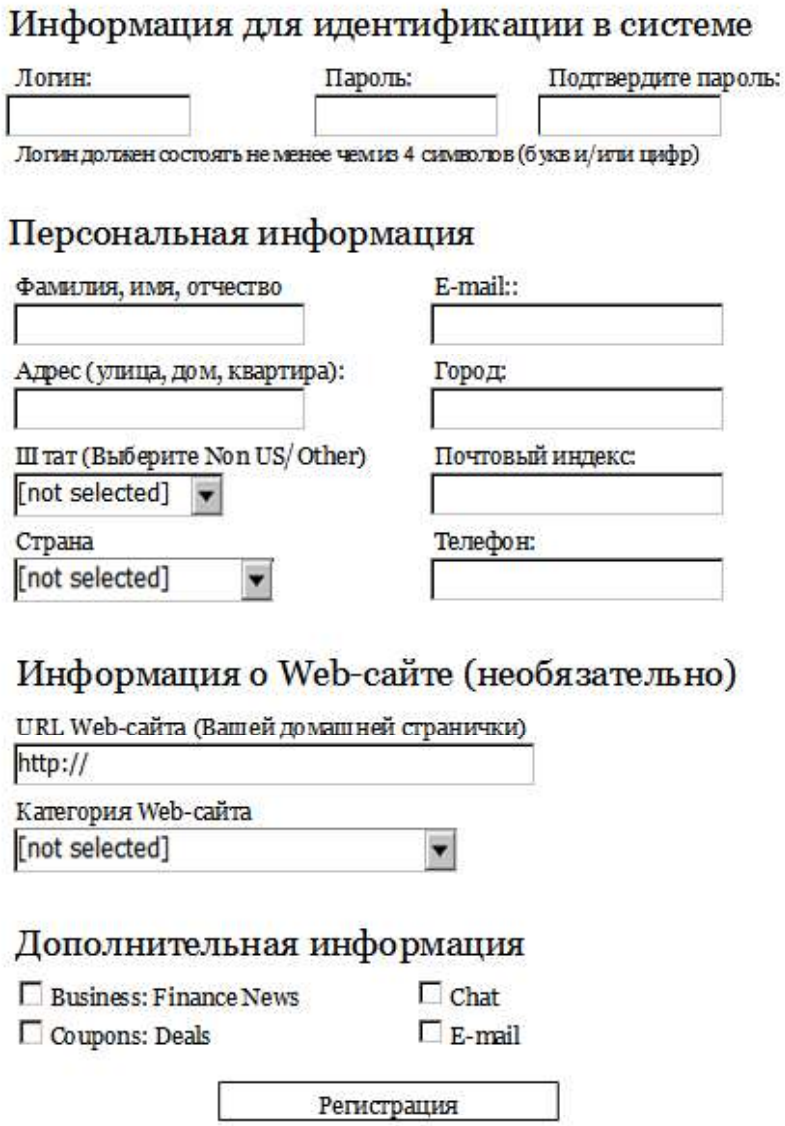
Обмен информацией между Web-сервером и клиентом

Вариант 7.



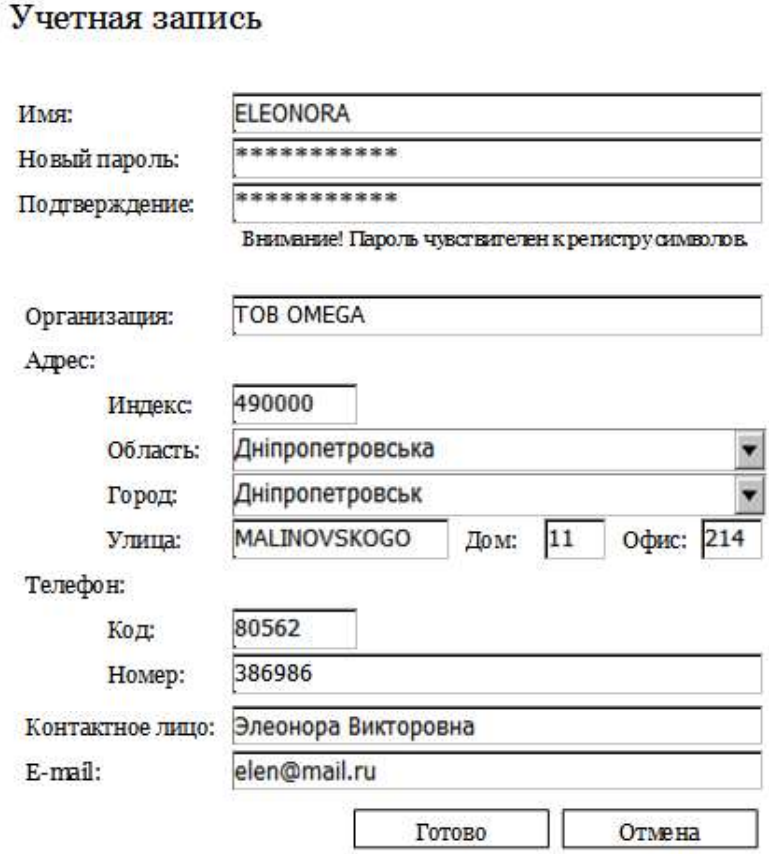
Обмен информацией между Web-сервером и клиентом

Вариант8.

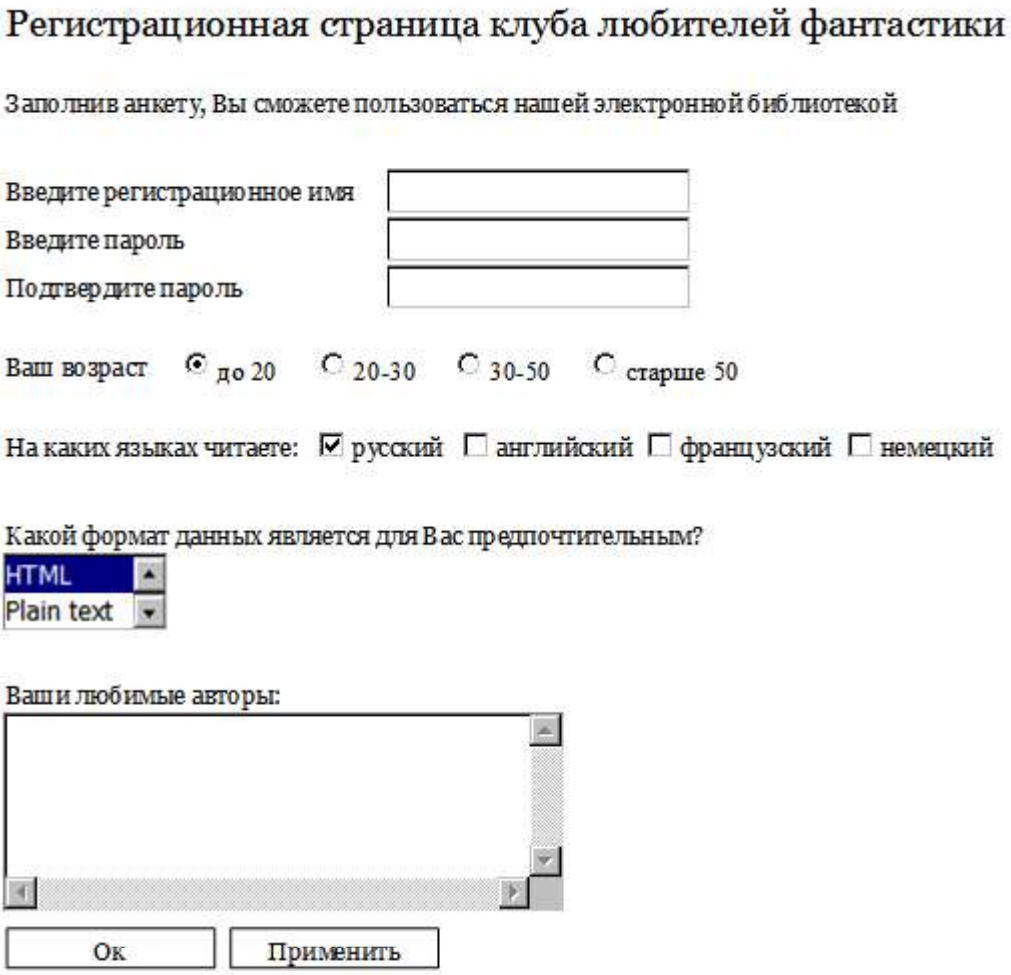


Обмен информацией между Web-сервером и клиентом

Вариант 9



Вариант 10



Задание 2: Решить задания первой лабораторной работы, используя формы

.

Приложение №2 Элементы формы

Кнопка с изображением (type=image)

Вместо кнопки submit можно использовать рисунок для отправки данных. Клик на этом рисунке дает то же самое, что и нажатие на кнопку submit. Однако, кроме этого, сценарию будут переданы координаты места клика на рисунке. Координаты будут переданы в формате имя.x=koop_X, y=koop_Y.

Синтаксис кнопки отправки с рисунком:

```
<input type=image name=имя src=рисунок>
```

Поле пароля (type=password)

Это элемент ввода пользовательского пароля. Ничем не отличается от обыкновенной текстовой строки, за исключением того, что набранный текст отображается в виде звездочек:

```
<INPUT TYPE="password" SIZE="30" NAME="password">
```

Такая мера связана с сохранением конфиденциальности пользовательских данных (однако не стоит забывать, что данные, вводимые в это поле, при использовании типа передачи GET будут отображаться в ссылке запроса браузера).

Скрытое текстовое поле (type=hidden)

Для передачи служебной информации (о которой пользователь даже не должен подозревать) используются скрытые поля. С помощью таких полей, например, могут передаваться параметры настройки.

```
<input type=hidden name=имя_параметра value=значение>
```

Такие поля передаются серверу, но на веб-странице не отображаются.

Текстовые поля

Не всегда текст, который нужно ввести, помещается в одной строке. Бывает, что он растягивается на несколько строк или даже абзацев. Конечно, можно обойтись текстовой строкой "бесконечной" длины (без указания значения параметра maxlength). Однако выглядит такая строка — без начала, без конца — неизящно, а пользоваться ею очень неудобно.

Поэтому для ввода крупных блоков текста предусмотрен другой элемент формы — *поле ввода.*

Для создания текстового поля используется дескриптор <TEXTAREA>. Он создает внутри формы поле для ввода многострочного текста, отображаемое в окне браузера в виде прямоугольной области с горизонтальной и вертикальной полосами прокрутки.

Дескриптор <TEXTAREA> — парный. Внутри него помещается текст, который должен оказаться в поле ввода по умолчанию. Это логичнее, чем делать его значением параметра value: ведь текст может быть довольно длинным.

```
<TEXTAREA COLS="25" ROWS="5" NAME="comment">Ваш
комментарий...</TEXTAREA>
```

Весь текст выводится, как правило, моноширинным шрифтом ("пишущая машинка").

Основные параметры тега <TEXTAREA>

Параметр	Описание
COLS	Определение количества столбцов текстового поля
ROWS	Определение количества рядов текстового поля
NAME	Присвоение уникального имени, необходимого для идентификации программой-обработчиком
READONLY	Позволяет создать элемент, недоступный для редактирования
WRAP	Способ представления текста, вводимого в окно. • Virtual - в окне текст автоматически разбивается на строки, но при передаче эта автоматическая разбивка не сохраняется, если вы ввели все одной строкой, то оно так и будет передано. Используется по умолчанию. • Off - Если мы хотим, чтобы переход на новую строку в окне происходил только когда пользователь нажимает <Enter> • Hard - если мы хотим, чтобы переход на новую строку происходил автоматически, и эта разбивка сохранялась при передаче текста на обработку

Кнопка для загрузки файлов (browse)

Тег INPUT позволяет реализовать еще одну возможность форм, а именно создавать поле выбора файла для его отправки на сервер. Синтаксис следующий:

```
<input type=file name=имя [value=имя_файла]>
```

Лабораторная работа №10[Создание простейших Web-приложений средствами языка php]

Цель работы: Формирование навыков создания простейших web-приложений средствами языка PHP.

Задачи:

1. Научиться применять основные управляющие конструкции языка для создания простейших web-приложений средствами языка php
2. Получить навыки обрабатывать данные, полученные с формы, средствами языка php при создании различных по назначении простейших web-приложений

План:

1. Запуск Денвера.
2. Выполнение заданий в соответствии со своим вариантом.

Варианты. Выполните задания указанные в Вашем варианте. Четные варианты выполняют все задания под римской цифрой - II , нечетные - I

I. 1,3,5,7,8,10,12,13
II. II2,4,5,6,7,11,12,13

1. Создайте веб-приложение, реализующее калькулятор (минимум четыре действия).

Рекомендации: Создайте HTML-документ с формой, содержащей поля ввода для операндов и выпадающий список для выбора действия над операндами, а также кнопку, отсылающую информацию на сервер (не забудьте дать имена элементам формы). После этого создайте HTML-

документ с PHP-скриптом, который будет обрабатывать запрос из формы и выдавать результат. Не забудьте учесть ноль при делении!

2. Создайте веб-приложение, анализирующее введенное в поле ввода число по следующим параметрам: четность, знак (положительное\отрицательное) и количество разрядов. Если введено не число, то сообщить об этом.

Рекомендации: Создайте HTML-документ с формой, содержащей поле ввода для ввода анализируемого числа и кнопку, отсылающую информацию на сервер (не забудьте дать имена элементам формы). После этого создайте HTML-документ с PHP-скриптом, который будет обрабатывать запрос из формы и выдавать результат. Не забудьте про ноль! Помня о том, что PHP не следит за типом данных, число разрядов можно считать как количество символов в строке (если, конечно, пользователь ввёл число).

Для написания этого приложения вам могут пригодиться следующие функции языка PHP:

1. `strlen(string)` — функция, возвращающая длину переданной строки string

Например, выполнение следующего кода выведет на экран число 70:

```
<?
$text = "PHP was written as a set of CGI binaries in the
C programming language";
$length = strlen($text);
echo $length;
?>
```

2. `is_numeric(variable)` - функция, возвращающая TRUE в том случае, если в переменной variable хранится числовое значение.

Например, выполнение следующего кода выведет на экран сообщение «не число»:

```
<?
$variable = "PHP";
if (!is_numeric($variable))
echo "Не число";
else
echo "Число";
?>
```

3.Создайте веб-приложение, позволяющее проверить информацию, введенную в регистрационную форму, состоящую из следующих полей: Имя (максимум 10 символов) и адрес электропочты (наличие обязательного символа @ и минимум двух точек). Если хотя бы одно поле содержит ошибку, то вывести соответствующее сообщение, если же вся информация введена верно, то поздравить с успешной регистрацией.

Рекомендации: Создайте HTML-документ с формой, содержащей нужное количество полей ввода и кнопку, отсылающую информацию на сервер (не забудьте дать имена элементам формы). После этого создайте HTML-документ с PHP-скриптом, который будет обрабатывать запрос из формы и выдавать результат.

Для написания этого приложения вам могут пригодиться следующие функции языка PHP:

1. `strlen(string)` — функция возвращающая длину переданной строки string. Например, выполнение следующего кода выведет на экран число 70:

```
<?
$text = "PHP was written as a set of CGI binaries in the
C programming language";
$length = strlen($text);
echo $length;
?>
```

2. `substr_count(string1, string2)` — функция возвращающая количество вхождений строки string2 в строке string1.

Например, выполнение следующего кода выведет на экран число 2:

```
<?
$text = "abrakadabra";
$quantity = substr_count($text, "bra" );
echo $quantity;
?>
```

4. Создайте веб-приложение, реализующее тест на основе выпадающих списков.

Рекомендации: Создайте HTML-документ с формой, содержащей нужное количество выпадающих списков с вариантами ответов на вопросы и кнопку, отсылающую информацию на сервер (не забудьте дать имена элементам формы). После этого создайте HTML-документ с PHP-скриптом, который будет обрабатывать запрос из формы и выдавать результат.

Для выполнения этого задания вам уже известны все необходимые функции и конструкции.

5. Создайте веб-приложение, позволяющее определить название браузера пользователя. Выведите результат в удобочитаемом виде, например: «Вы используете Apple Safari».

Рекомендации: В этой задаче не нужно отправлять данные на сервер через форму. Создайте HTML-документ с PHP-скриптом, определяющим веб-браузер пользователя по строке User Agent, которая хранится в переменной окружения $_SERVER. $_SERVER — это глобальный ассоциативный массив, содержащий информацию о параметрах, получаемых от сервера, в том числе необходимую строку User Agent. Узнайте, как информация помещается в переменную $_SERVER["HTTP_USER_AGENT"] различными браузерами (Microsoft Internet Explorer, Mozilla Firefox, Opera, Apple Safari).

Пример строки User Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; ru; rv:1.8.1.6) Gecko/20070725 Firefox/2.0.0.6

Для написания этого приложения вам может пригодиться следующая функция языка PHP:

1. `substr_count(string1, string2)` — функция, возвращающая количество вхождений строки string2 в строке string1.

Например, выполнение следующего кода выведет на экран надпись «Володя!»:

```
<?
$text = "Володя сильно обжегся крапивой!";
if substr_count($text, "Володя" )
echo "Тут есть Володя!";
else
echo"Увы, Володи нет!";
?>
```

6. Создайте веб-приложение, позволяющее определить операционную систему, в которой запущен браузер пользователя. Выведите результат в удобочитаемом виде, например: «Вы используете ОС Windows Vista»

Рекомендации: В этой задаче не нужно отправлять данные на сервер через форму. Создайте HTML-документ с PHP-скриптом определяющий операционную систему пользователя по строке User Agent(, которая хранится в переменной окружения $_SERVER. $_SERVER — это глобальный ассоциативный массив (подробнее об ассоциативных массивах:) содержащий информацию о параметрах получаемых от сервера, в том числе необходимую строку (Чаще всего в строку $_SERVER["HTTP_USER_AGENT"] попадают следующие записи, позволяющие определить ОС:

- Windows NT 5.0
- Windows NT 5.1
- Windows NT 6.0
- Macintosh
- Linux

Самостоятельно узнайте, какие операционные системы скрываются за этими строками, а так же постарайтесь протестировать свой скрипт на максимально возможном числе ОС.

Пример строки User Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; ru; rv:1.8.1.6) Gecko/20070725 Firefox/2.0.0.6

Для написания этого приложения вам может пригодиться следующая функция языка PHP:

1.`substr_count(string1, string2)` — функция возвращающая количество

вхождений строки string2 в строке string1.

Например, выполнение следующего кода выведет на экран надпись «Тут есть пиво!»:

```
<?
$text = "Володя сильно обжегся крапивой!";
if substr_count($text, "пиво" )
echo "Тут есть пиво!";
else
echo "Увы, пива нет!";
?>
```

7. Создайте веб-приложение, позволяющее определить, используется ли нв операционной системе пользователя русский язык, а так же предыдущую посещённую страницу. Если используется русский язык, то написать ответ на русском, в противном случае ответить на английском. Например, сообщение о языке может выглядеть следующим образом: «Вы используете русский язык!». Адрес предыдущей страницы оформите в ссылку, а если предыдущая страница не указана, то выведите соответствующее сообщение.

Рекомендации: В этой задаче не нужно отправлять данные на сервер через форму. Создайте HTML-документ с PHP-скриптом определяющий язык используемый в операционной системе пользователя по строке ACCEPT_LANGUAGE, которая хранится в переменной окружения $_SERVER. $_SERVER — это глобальный ассоциативный массив

содержащий информацию о параметрах получаемых от сервера, в том числе необходимую строку о языке $_SERVER["HTTP_ACCEPT_LANGUAGE"]. Самостоятельно узнайте как обозначается русский язык. Адрес предыдущей (ссылающейся на текущую) страницы можно найти в строке $_SERVER["HTTP_REFERER"].
Для написания этого приложения вам может пригодиться следующая особенность языка PHP:
Что бы предотвратить вывод сообщения-ошибки об отсутствии требуемой переменной (так может случиться, если не было предыдущей страницы $_SERVER["HTTP_REFERER"]) используйте следующий синтаксис:
@$_SERVER["HTTP_REFERER"] Символ @ указывает интерпретатору нато, что нужно проигнорировать возможную ошибку.

8. Создайте веб-приложение позволяющее выстроить возрастающую последовательность из чисел, переданных через поля ввода формы.

Рекомендации: Создайте HTML-документ с формой, содержащей нужное количество (например, 10) полей ввода, предназначенных для ввода чисел, и кнопку, отсылающую информацию на сервер (не забудьте дать имена элементам формы). После этого создайте HTML-документ с PHP-скриптом, который будет обрабатывать запрос из формы и выдавать результат. Занесите значения из форм в массив. Реализуйте любой известный вам алгоритм сортировки массивов. Выведите результирующий массив на страницу.

Для написания этого приложения вам может пригодиться следующая информация о синтаксисе PHP:

Создать массив можно непосредственно присваивая значения без дополнительных объявлений, например:

```
$my_array[1]=$_POST["number_1"];
$my_array[2]=$_POST["number_2"];
```

и так далее...

Примечание.

Выясните, как можно передать данные сценарию в виде массива, дав соответствующие имена элементам формы.

9.Создайте веб-приложение, позволяющее выстроить убывающую последовательность из чисел переданных через поля ввода формы.

Рекомендации: Создайте HTML-документ с формой, содержащей нужное количество (например, 10) полей ввода, предназначенных для ввода чисел, и кнопку, отсылающую информацию на сервер (не забудьте дать имена элементам формы). После этого создайте HTML-документ с PHP-скриптом, который будет обрабатывать запрос из формы и выдавать результат. Занесите значения из форм в массив. Реализуйте любой известный вам алгоритм сортировки массивов. Выведите результирующий массив на страницу.

Для написания этого приложения вам может пригодиться следующая информация о синтаксисе PHP:

Создать массив можно непосредственно присваивая значения без дополнительных объявлений, например:

```
$my_array[1]=$_POST["number_1"];
$my_array[2]=$_POST["number_2"];
```

и так далее...

Примечание.

Выясните, как можно передать данные сценарию в виде массива, дав соответствующие имена элементам формы.

10. Создайте веб-приложение, позволяющее вычислить среднее арифметическое,
а так же найти максимальный элемент последовательности чисел,
переданных через поля ввода формы.

Рекомендации: Создайте HTML-документ с формой, содержащей нужное количество (например, 10) полей ввода, предназначенных для ввода чисел, и кнопку, отсылающую информацию на сервер (не забудьте дать имена элементам формы). После этого создайте HTML-документ с PHP-скриптом, который будет обрабатывать запрос из формы и выдавать результат.
Возможно, удобнее всего будет занести значения из форм в массив.

Для написания этого приложения вам может пригодиться следующая информация о синтаксисе PHP:
Создать массив можно непосредственно присваивая значения без дополнительных объявлений, например:

```
$my_array[1]=$_POST["number_1"];

$my_array[2]=$_POST["number_2"];
```

и так далее...

Примечание.

Выясните, как можно передать данные сценарию в виде массива, дав соответствующие имена элементам формы.

11. Создайте веб-приложение, позволяющее вычислить сумму элементов, а так же найти минимальный элемент последовательности чисел переданных через поля ввода формы.

Рекомендации: Создайте HTML-документ с формой, содержащей нужное количество (например, 10) полей ввода, предназначенных для ввода чисел, и кнопку, отсылающую информацию на сервер (не забудьте дать имена элементам формы). После этого создайте HTML-документ с PHP-скриптом, который будет обрабатывать запрос из формы и выдавать результат. Возможно, удобнее всего будет занести значения из форм в массив.

Для написания этого приложения вам может пригодиться следующая информация о синтаксисе PHP:

Создать массив можно непосредственно присваивая значения без дополнительных объявлений, например:

```
$my_array[1]=$_POST["number_1"];

$my_array[2]=$_POST["number_2"];
```

и так далее...

Примечание.

Выясните, как можно передать данные сценарию в виде массива, дав соответствующие имена элементам формы

12. Создайте веб-приложение, реализующее функцию обратной связи для вашегопортфеля.

Рекомендации: Создайте HTML-документ с формой, содержащей поля ввода: Имя, Адрес, Сообщение (многострочному полю ввода соотвествует тег textarea) икнопку, отсылающую информацию на сервер (не забудьте дать имена элементам формы). После этого создайте HTML-документ с PHP-скриптом, который будет обрабатывать запрос из формы и отсылать сообщение по электронной почте через протокол SMTP средствами функции mail. Не забудьте удостовериться в том, что все поля заполнены! Желательно так же реализовать проверку адреса электронной почты на правильность (см. похожее задание выше). После отправки сообщения выведите на страницу соответствующее сообщение и ссылку на другую страницу сайта.

Для написания этого приложения вам необходима следующая функция PHP:

mail(получатель, тема, сообщение, дополнительные заголовки) — функция реализующая отправку сообщения электронной почты.

Удобно каждый из параметров заранее определить в переменные.
В качестве параметра «получатель» (например, переменная $to) укажите ваш адрес электропочты. Тему ($subject) укажите самостоятельно так, что бы вы смогли отличить это письмо из кучи прочих .Сообщение ($message) получите из формы через POST.
Что бы понять от кого это письмо, в «дополнительные заголовки» занесите следующую информацию:
$mail_headers = "From: " // Здесь добавьте информацию из поля формы

«Адрес». Должно получиться что-то вроде:
"From: myaddress@mydomain.com"
После определения всех параметров, просто вызовите функцию
mail($to, $subject, $message, $mail_headers), указав вкачестве параметров необходимые переменные.

13. Создайте веб-приложение, в котором страница с формой и обработчик формы находятся в одном файле. В качестве формы используйте регистрационную форму из нескольких полей (минимум 4) и флажка (checkbox) обозначающего согласие с правилами поведения или лицензионным соглашением. Если все поля заполнены и флажок установлен, то результатом работы скрипта должно быть сообщение об успешной регистрации нового пользователя.

Рекомендации: Идея состоит в том, что HTML-форма и её обработчик (php-скрипт) физически находятся в одном файле, но одновременно никогда не передаются браузеру. PHP-скрипт выбирает что именно передавать основываясь на каком-либо параметре. Чаще всего таким параметром служит значение одного из GET параметров. Будем считать так: если параметр GET получен и равен какому-то определённому значению (пусть в нашем случае сигналом будет GET- переменная status со значением finish), то нужно обработать форму. Если параметр GET не получен, то необходимо вывести форму. Далее необходимо вспомнить о том, что PHP — это вкрапления в HTML. Алгоритмические конструкции (в нашем случае это будет ветвление) могут прерываться HTML-кодом, например страница registration.php:

```
<html>
<head>
<tiltle>Страница, которая сама себя
обрабатывает</title>
</head>
<body>
<?
if ($_GET["status"]!="finish") {
?>
```

```
<form method="POST"
action="registration.php?status=finish">
Ваше имя: <input type="text" name="user_name" />
...
</form>
<? } else {
$user_name = $_POST["user_name"]
....
?>
<h1>Регистрация прошла успешно</h1>
<? } //Закрывающая скобка Else ?>
</body>
</html>
```

Обратите внимание на тег <form> и его атрибут action. Здесь мы самостоятельно определили GET-переменную. Просто повторили синтакисис, который создаёт форма с атрибутом method="GET". В этом веб-приложении мы использовали стразу два способа передачи информации от клиента серверу: и GET, и POST.

Для написания этого приложения вам может пригодиться следующаяособенность языка PHP:

Что бы предотвратить вывод сообщения-ошибки об отсутствии требуемой переменной (так может случиться, если не передан параметр **$_GET["status"]**, т.е. нужно вывести форму, а не запускать обработчик) используйте следующий синтаксис: **@$_GET["status"]** Символ @ указывает интерпретатору на то, что нужно проигнорировать возможную ошибку.

Лабораторная работа №11[Работа с массивами данных]

Цель работы: Изучить основные особенности использования массивов в PHP.

Задачи:

1. Научиться работать с одномерными, многомерными и ассоциативными массивами.
2. Формирование навыков сортировки массивов, работы с элементами массива.

План:

1. Запуск Денвера.
2. Ознакомление с теоретической частью.
3. Выполнение и отладка примеров программ из экспериментальной части.
4. Выполнение заданий для самостоятельной работы.

Теоретическая часть

Массив – упорядоченная совокупность данных. Доступ к элементам массива производится через индекс. Одному индексу соответствует один и только один элемент массива. В PHP индексы также называются ключами.

Существует два типа массивов, различающиеся по способу идентификации элементов.

1. В массивах первого типа элемент определяется индексом в последовательности. Такие массивы называются простыми массивами.

2. Массивы второго типа имеют ассоциативную природу, и для обращения к элементам используются ключи, логически связанные со значениями. Такие массивы называют ассоциативными массивами.

Существует два метода создания массива в PHP:

- Присвоение значений элементам массива
- Использование функции array()

Массивы могут быть как одномерными, так и многомерными.

Простые массивы в PHP

При обращении к элементам простых индексируемых массивов используется целочисленный индекс, определяющий позицию заданного элемента.

Простые одномерные массивы:

1 способ создания массива:

Обобщенный синтаксис элементов простого одномерного массива:

```
$имя[индекс];
```

Массивы, индексами которых являются числа, начинающиеся с нуля - это списки:

Работа с массивами данных

Пример 1.

```
<?php
// Простой способ инициализации массива
$names[0]="Апельсин";
$names[1]="Банан";
$names[2]="Груша";
$names[3]="Помидор";
// Здесь: names - имя массива, а 0, 1, 2, 3 - индексы
массива
?>
```

Доступ к элементам простых массивов (списков) осуществляется следующим образом:

Пример 2.

```
<?php
// Простой способ инициализации массива
$names[0]="Апельсин";
$names[1]="Банан";
$names[2]="Груша";
$names[3]="Помидор";
// Здесь: names - имя массива, а 0, 1, 2, 3 - индексы
массива

// Выводим элементы массивов в браузер:
echo $names[0]; // Вывод элемента массива names с
индексом 0
echo "<br>";
```

```
echo $names[3]; // Вывод элемента массива names с
индексом 3
// Выводит:
// Апельсин
// Помидор
?>
```

С технической точки зрения разницы между простыми массивами и списками нет.

Простые массивы можно создавать, не указывая индекс нового элемента массива, это за вас сделает PHP.

Пример 3.

```
<?php
// Простой способ инициализации массива, без указания
индексов
$names[]="Апельсин";
$names[]="Банан";
$names[]="Груша";
$names[]="Помидор";
// PHP автоматически присвоит индексы элементам
массива, начиная с 0

// Выводим элементы массивов в браузер:
echo $names[0]; // Вывод элемента массива names с
индексом 0
echo "<br>";
```

```
echo $names[3]; // Вывод элемента массива names с
индексом 3
// Выводит:
// Апельсин
// Помидор
?>
```

В рассмотренном примере вы можете добавлять элементы массива names простым способом, то есть не указывая индекс элемента массива:

```
$names[]="Яблоко";
```

Новый элемент простого массива (списка) будет добавлен в конец массива. В дальнейшем, с каждым новым элементом массива, индекс будет увеличиваться на единицу.

2 способ создания массива:

Создать массив можно более простым способом, используя функцию **array( ),** а не оператор[].

Пример 4.

```
$list=array("Mandriva", "Ubuntu", "SUSE", "Fedora")
```

Работа с элементами массива:

Добалять элементы в массив можно также с помощью функции **array_push( ):**

```
array_push(массив, элемент1, элемент2, элемент3…. )
```

Пример 5. Функцией array_push( ) добавили 2 элемента “Gentoo”,”Kubuntu” в массив $list:

```
array_push($list, "Gentoo","Kubuntu" )
```

Чтобы удалить последний элемент, можно использовать функцию **array_pop( )**;

```
array_pop(массив)
```

Пример 6. Функцией array_pop( ) удалили последний элемент - ”Kubuntu” из массива $list:

```
array_pop($list)
```

Поиск элемента:

Для проверки наличия элемента в массиве существуют функции:

- **in_array()** - если элемент найден, возвращает `true`, иначе - `false`.
- **array_search()** - если элемент найден, возвращает его ключ, иначе - `false`.

Пример 7. Поиск элемента в массиве

```
<?php
$a = array("первый" => 6, "второй" => 2, "третий" =>
1);
if (in_array (2, $a)) echo "2 нашли!<br>";
echo "ключ найденного элемента - ".array_search(2, $a);
?>
```

Результат Примера 7:

```
2 нашли! ключ найденного элемента - второй
```

Простые многомерные массивы

В качестве элементов массива могут выступать не только скалярные величины, но и сами массивы. В этом случае получаются так называемые многомерные массивы.

Пусть необходимо получить такую таблицу:

Имя	Профессия	Зарплата
Вася	Слесарь	2500
Миша	Строитель	3000
Андрей	Шофер	2700

Для этого создадим двумерный массив:

```
<?php
  $arr = array( array("Вася", "слесарь", 2500 ),
            array("Миша", "строитель", 3000),
            array("Андрей", "шофер", 2700));
?>
```

Теперь таблицу можно вывести при помощи следующего кода:

```
<?php
  for ($i = 0; $i < 3; $i++)
  {
    for ($j=0; $j <3; $j++)
    {
       echo ' | '.$arr[$i][$j];
    }
    echo '<br />';
  }
?>
```

Ассоциативные массивы в PHP

В PHP индексом массива может быть не только число, но и строка. Причем на такую строку не накладываются никакие ограничения: она может содержать пробелы, длина такой строки может быть любой. Ассоциативные массивы особенно удобны в ситуациях, когда элементы массива удобнее связывать со словами, а не с числами. Итак, массивы, индексами которых являются строки, называются ассоциативными массивами.

Одномерные ассоциативные массивы:

Одномерные ассоциативные массивы содержат только один ключ (элемент), соответствующий конкретному индексу ассоциативного массива.

Приведем пример:

```
<?php
// Ассоциативный массив
$names["Иванов"]="Иван";
$names["Сидоров"]="Николай";
$names["Петров"]="Петр";
// В данном примере: фамилии - ключи ассоциативного
массива
// , а имена - элементы массива names
?>
```

Доступ к элементам одномерных ассоциативных массивов осуществляется так же, как и к элементам обыкновенных массивов, и называется доступом по ключу:

```
echo $names["Иванов"];
```

Для создания ассоциативного массива при помощи функции array(), используется знак “=>” для присвоения значений элементам…

```
<?php
 $mass=array('element1' => 'value1', 'element2' =>
'value2', 'element3' => 'value3');
     echo $mass['element2']; // Выводит элемент массива
с индексом 'element2'
     ?>
```

Многомерные ассоциативные массивы:

Многомерные ассоциативные массивы могут содержать несколько ключей, соответствующих конкретному индексу ассоциативного массива.
Рассмотрим пример многомерного ассоциативного массива:

```
<?php
// Многомерный массив
$A["Ivanov"] = array("name"=>"Иванов И.И.",
"age"=>"25", "email"=>"ivanov@mail.ru");
$A["Petrov"] = array("name"=>"Петров П.П.",
"age"=>"34", "email"=>"petrov@mail.ru");
$A["Sidorov"] = array("name"=>"Сидоров С.С.",
"age"=>"47", "email"=>"sidorov@mail.ru");
?>
```

Доступ к элементам многомерного ассоциативного массива осуществляется следующим образом:

```
echo $A["Ivanov"]["name"]; // Выводит Иванов И.И.
echo $A["Petrov"]["email"]; // Выводит petrov@mail.ru
```

Просмотр массива в цикле

К единичным элементам обращаются редко. Чаще требуется вывести элемент в цикле. Для прохода по списку удобнее использовать цикл **for**. Для итерационного просмотра содержимого массива служит функция **foreach**. С ее помощью можно просмотреть и простой (проиндексированный числами) массив, и ассоциативный, и многомерный.

Синтаксис цикла foreach выглядит следующим образом:

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

Здесь команды циклически выполняются для каждого элемента массива, при этом очередная пара ключ=>значение оказывается в переменных $ключ и $значение. Приведем пример работы цикла foreach.

Пример 8. Просмотр массива

```
<?php
$фрукты = array("яблоко", "груша", "слива", "персик",
"груша");
foreach ($фрукты as $фрукт)
  {
  echo  "$фрукт<br>";
  }
?>
```

Рассмотренный сценарий выводит:

```
яблоко
груша
слива
персик
груша
```

Пример 9. Просмотр ассоциативного массива

```
<?php
$names["Иванов"] = "Андрей";
$names["Петров"] = "Борис";
$names["Волков"] = "Сергей";
$names["Макаров"] = "Федор";
foreach ($names as $key => $value) {
echo "<b>$value $key</b><br>";
}
?>
```

Рассмотренный сценарий выводит:

```
Андрей Иванов
Борис Петров
Сергей Волков
Федор Макаров
```

Пример 10. Просмотр двумерного ассоциативного массива

```
<?php
$данные = array (
    "Иванов" => array ("рост" => 174, "вес"  => 68),
    "Петров" => array ("рост" => 181, "вес"  => 90),
    "Сидоров" => array ("рост" => 166, "вес"  => 73));
foreach ($данные as $фамилия => $данные1)
  {
  echo  "<br>$фамилия:<br>";
  foreach ($данные1 as $параметр => $pp)
```

```
    {
    echo  "$параметр = $pp<br>";
    }
  }
?>
```

Рассмотренный сценарий выводит:

```
Иванов:
рост = 174
вес = 68

Петров:
рост = 181
вес = 90

Сидоров:
рост = 166
вес = 73
```

Операции над массивами

Функция count( )

Возвращает количество элементов в массиве. Например:

```
<?php
   $mass=array(1,2,6,8,4,9);
     echo count($mass);
     ?>
```

Функция array_merge( )

Объединяет несколько массивов в один массив, добавляя элементы каждого последующего массива в конец предыдущего и возвращает полученный в результате объединения массив.

```
<?php
   $array1 = array(1, 2, 3, 4);
     $array2 = array(5, 6, 7);
     $array3 = array(8,9);
     $finish_array=array_merge($array1, $array2,
$array3 );
     echo count($finish_array); // выведет 9
     ?>
```

Функция array_unique()

Удаляет дубликаты из массива. Принимает массив 1 и возвращает массив 2, в котором удалены все дубликаты.

Пример:

```
<?php
     $massiv_1 = array(4, 4, 3, 4, 3, 2);
     $massiv_2 = array_unique($massiv_1);
     echo 'Размер массива 1 =
'.count($massiv_1).'<br>'; // выведет 6
     echo 'Размер массива 2 =
'.count($massiv_2).'<br>'; // выведет 3
     ?>
```

Сортировка массивов

Sort( ) – эта функция сортирует массив. Элементы буду упорядочены от низшего к высшему.

Синтаксис:

```
void sort (array arr [, int sort_flags])
```

Пример 11.

```
<?
   $arr = array("2", "1", "4", "3","5");
   sort($arr);
   for($i=0; $i < count($arr); $i++)
   {
      echo ("$i:$arr[$i]  ");
   }
   // выводит "0:1 1:2 2:3 3:4 4:5"
?>
```

Необязательный аргумент sort_flags указывает как именно должны сортироваться элементы (задает флаги сортировки). Допустимыми значениями этого аргумента являются следующие:
SORT_REGULAR – задает нормальное сравнение элементов (сравнивает элементы "как есть")
SORT_NUMERIC - сравнивает элементы как числа
SORT_STRING - сравнивает элементы как строки

Rsort( ) - эта функция сортирует массив. Элементы буду упорядочены от высшего к низшему.

Синтаксис:

```
void rsort(array arr [, int sort_flags])
```

Аналогична функции sort(), только сортирует по убыванию.***ировка ассоциативных массивов:***

Такие массивы можно сортировать как по ключам, так и по значениям.

Asort( ) Сортировка ассоциативного массива по значениям, элементы буду упорядочены от высшего к низшему.

Синтаксис:

```
void asort(array arr [, int sort_flags])
```

Важное отличие этой функции от функции sort( ) состоит в том, что при применении функции asort()сохраняются связи между ключами и соответствующими им значениями, чего нет в функции sort() (там эти связи попросту разрываются).

Пример 12.

```
<?
   $arr = array("a" =>"one","b" => "two","c" =>
"three","d" => "four");
   asort($arr);
   foreach($arr as $key => $val)
   {
      echo (" $key => $val ");
   }
?>
```

Рассмотренный сценарий выводит:

```
d => four a => one c => three b => two
```

Т.е., как видим, связи "ключ-значение" сохранились.

Arsort( ) -сортировка ассоциативного массива по значениям, элементы буду упорядочены от низшего к высшему.

Синтаксис:

```
void arsort(array arr [, int sort_flags])
```

Эта функция аналогична функции asort(), только она упорядочивает массив не по возрастанию, а по убыванию.

Ksort( ) Сортировка ассоциативного массива по ключам , элементы буду упорядочены от высшего к низшему.

Синтаксис:

```
Int  ksort(array arr [, int sort_flags])
```

Krsort( ) Сортировка ассоциативного массива по ключам, элементы буду упорядочены от низшего к высшему.

Синтаксис:

```
int krsort(array arr [, int sort_flags])
```

Сортировка многомерных массивов:

Сортировка массивов, имеющих более одного измерения, или в порядке, отличающемся от алфавитного или цифрового более сложна. В PHP имеется возможность сравнения двух чисел или двух строк, но в многомерном массиве каждый элемент является массивом. В PHP отсутствует возможность

сравнения массивов, поэтому для их сравнения необходимо создать метод. В большенстве случаев порядок слов или номеров очевиден – но в случае сложных объектов выполнение задачи становится более проблематичным.

Пример 13. Дан двумерный массив, в котором хранятся названия товаров, их коды и цены.

```
$products = array( array( "TIR", "Tires", 100 ),
            array( "OIL", "Oil", 10 ),
            array( "SPK", "Spark Plugs", 4 ) ) ;
```

Каков будет порядок значений, если выполнить сортировку в этом массиве? Поскольку известно, что именно представляет содержимое массива, существует, по меньшей мере, два полезных порядка сортировки. Товары можно упорядочить в алфавитном порядке по описаниям или в цифровом порядке по ценам. И то, и другое одинаково возможно, но нужно использовать функцию usort() и указать PHP, как следует сравнивать элементы. Для этого потребуется создать собственную функцию сравнения.

Следующий код выполняет сортировку этого массива в алфавитном порядке по значению второго столбца — описания.

```
function compare($x, $y)
{
if ( $x[1] == $y[1] )
return 0;
else if ( $x[1] < $y[1] )
return -1;
else
return 1;
}
usort($products, compare);
```

До сих пор мы использовали встроенные PHP-функции. Для сортировки этого массива мы определили свою собственную функцию. Функция определяется с помощью ключевого слова function. Функции необходимо присвоить имя. Имена должны быть имеющими смысл, поэтому назовем нашу функцию compare(). Многие функции принимают параметры, или аргументы. Наша функция compare() принимает два аргумента: x и y. Ее назначение — принять два значения и определить их порядок.

Применительно к рассматриваемому примеру параметрами x и y будут два массива внутри основного массива, каждый из которых представляет один из товаров. Чтобы обратиться к элементу Description массива x, необходимо ввести $x[1], поскольку Description — второй элемент в этом массиве, а нумерация начинается с нуля. Для сравнения элементов Description из массивов, переданных в функцию, используются переменные $x[1] и $y[1]. Когда функция завершает свою работу, она может сообщить ответ вызвавшему ее коду. Это значение называется возвращаемым. Для возврата значения в функции используется ключевое слово return. Например, строка return 1; возвращает значение 1 коду, вызвавшему функцию.

Чтобы она могла использоваться функцией usort(), функция compare() должна сравнивать x и y. Она должна возвращать 0, если значение x равно значению y, отрицательное число, если оно меньше, и положительное — если оно больше. В зависимости от значений x и y, наша функция будет возвращать значения 0, 1 или -1.

Заключительная строка кода вызывает встроенную функцию usort() с массивом, в котором нужно выполнить сортировку ($products) и именем нашей функции сравнения (compare()).

Если требуется, чтобы массив был отсортирован в другом порядке, можнопросто создать другую функцию сравнения. Для выполнения

сортировки по ценам необходимо просмотреть третий столбец массива и создать следующую функцию сравнения:

```
function compare($x, $y)
{
if ( $x[2] == $y[2] )
return 0;
else if ( $x[2] < $y[2] )
return -1;
else
return 1;
}
```

При вызове функции usort($products, compare) массив будет упорядочен в порядке возрастания цен.

Символ "и" в имени usort() означает "user" ("пользовательская"), поскольку этой функции требуется определяемая пользователем функция сравнения. Версии uasort() и uksort() функций asort() и ksort() также требуют использования определяемых пользователем функций сравнения.

Аналогично функции asort(), функция uasort() должна использоваться при сортировке ассоциативного массива по значениям. Функцию asort следует использовать, если значения являются простыми числами или текстом. Если же значения являются более сложными объектами, такими как массивы, следует определить функцию сравнения и использовать функцию uasort().

Подобно функции ksort(), функция uksort() должна использоваться при сортировке ассоциативного массива по значениям. Функцию ksort следует использовать, если значения являются простыми числами или текстом. Если

же значения являются более сложными объектами, такими как массивы, следует определить функцию сравнениями воспользоваться функцией uksort().

Задания для самостоятельной работы

1. Создайте ассоциативный многомерный массив, содержащий информацию о пользователях (ФИО, возраст, количество посещений страницы). Выведите всю информацию, начиная с пользователей, у которых количество посещений страницы больше.
2. Создайте массив, содержащий сведения об учениках класса (фамилия, рост, вес, средний балл). Найдите самого высокого ученика и выведите всю информацию о нем.
3. Создайте массив, содержащий сведения о ваших друзьях. Отсортируйте его по фамилиям друзей в алфавитном порядке и выведите всю информацию.
4. Создайте массив, содержащий сведения о ваших друзьях. Отсортируйте его по возрасту друзей и выведите всю информацию.
5. Создайте массив, содержащий сведения о продукции фирмы: номер товара, название, цена. Отсортируйте массив по названиям в алфавитном порядке. Среди товаров с одинаковым названием сначала идут более дешевые.
6. Создайте многомерный массив, содержащий названия музыкальных произведений, организованных по жанрам: ассоциативный массив, в котором имена полей будут разными жанрами («рок», «поп», «джаз» и др.), а элементами — названия песен. Выведите информацию.
7. Описать массив расписание, содержащий
    - день недели;
    - количество пар в этот день;
    - время начала и конца пары;
    - название предмета;

- фамилия преподавателя.

Вывести полную информацию о занятиях, относящихся к предметной области «Информатика».

8. В библиотеке имеются книги, газеты, журналы. Для каждого печатного издания указать
    - название;
    - год выпуска (для книги), дату выпуска (для газет и журналов);
    - автора (для книги), редактора (для газеты), редколлегию (для журнала);
    - объем.

    Вывести информацию об изданиях, вышедших в заданном году.
9. Описать массив экзаменационная ведомость (предмет, номер группы, номер зачетной книжки, фамилия, имя, отчество студента, его оценки по итогам текущей сессии). Определить отличников, хорошистов, троечников и двоечников.
10. Создайте многомерный массив, содержащий названия книг, организованных по жанрам: ассоциативный массив, в котором имена полей будут разными жанрами («детектив», "женский роман", "классика" и др.), а элементами — названия книг. Выведите информацию

Лабораторная работа №13[Работа с файлами. Строковые функции]

Цель работы: Приобретение навыков работы с файлами и строками

Задачи:

1. Формирование навыков создания файла, записи в файл, чтения содержимого файла, удаления содержимого файла.
2. Приобретение навыков работы со строковыми функциями PHP форматирования и манипулирования текстом, использования строковых функций поиска (и замены) слов, выражений или последовательностей внутри строки.

План:

1. Запуск Денвера.
2. Выполнение и отладка примеров программ из экспериментальной части.
3. Выполнение заданий для самостоятельной работы.

Экспериментальная часть

Функции для работы файлами

Пример 1. Создание файла и запись в файл

```
<?php

 $text = "Some Text";

 if( file_exists( "file.txt" ) ) // Проверяем файл на
                                 ///существование
```

```
  {
   if( !is_writeable( "file.txt" ) )// Проверяем файл
                                //на доступность записи
    {  die( "Вы не можете записать в этот файл" );  }
       // Завершаем выполнение скрипты, выводом
           //сообщения об ошибке
  }
 else  // Если файл не существует и нам все-таки надо
его создать.
  {
   if( !touch( "file.txt" ) )
    {  die( "Нельзя создать файл" );  }
       // Завершаем выполнение скрипты, выводом
           //сообщения об ошибке

   $f = fopen( "file.txt" , "w" );//открываем файл для
                                        //записи
   fwrite( $f , $text ); // Пишем в файл содержимое
                                //строки $text;
   fclose($f);
}

?>
```

Пример 2. Чтение содержимого текстового файла целиком

1 способ.

```
<?php
 $f = fopen( "file.txt" , "r" );//открываем файл для
чтения
 $text = fread( $f , filesize("file.txt") );
 print $text;
?>
```

2 способ.

```
<?php
 $text = file_get_contents( "file.txt" );//читает весь
файл в одну строку
 print $text // Выведет содержимое файла.
?>
```

3 способ.

```
<?php

 readfile( "file.txt" );
  // Выведет содержимое файла сразу в буфер.
?>
```

Пример 3. Чтение содержимого текстового файла построчно

```
<?php

 $f_arr = file( "file.txt" );
 // Каждая строка файла - будет элементом массива
$f_arr.
 // Номерация начинается с нуля.

?>
```

Пример 4 Добавление информации в начало файла

```
<?php
 $f = fopen( "file.txt" , "r" );//открытие файла для
чтения
 $text = fread( $f ,filesize("file.txt"));
 fclose($f);

 $text_towr = "Some Text to write in file";

 $f = fopen( "file.txt" , "w" );//открытие файла для
записи
 fwrite( $f , $text_towr . $text );
 fclose($f);
?>
```

Пример 5 Удаление строки из файла

```
<?php

 $f_arr = file( "file.txt" );

 $needle = 2; // Какую строчку нужно удалить.
              // Не забывайте, нумерация начинается с
нуля.
 array_splice( $f_arr , $needle , 1 );

 $f = fopen( "file.txt" , "w" );//открытие файла  для
записи

 for( $i = 0; $i < count( $f_arr ); $i++ )
  {  fwrite( $f , $f_arr[$i] . "\n" );  }

 fclose($f);

?>
```

Для удаления первой или последней строчки воспользуйтесь функциями: array_shift(); и array_push(); вместо array_splice().

Вывод случайной строки из файла:

Обычно такой вопрос задаются те, кто не использует какой-нибудь Базы Данных, тем не менее нуждается в ее функциях...

Вывести случайною строку из файла, например случайный анекдот можно так:

```
<?php

$arr = file( "file.txt" ); // Читаем файл построчно
print $arr[ rand( 0 , count( $arr ) - 1 ) ]; // Выводим
строку со случайным образом.

?>
```

Функция rand(); получает два параметра: первый - 0, второй - количество элементов массива минус 1. Т.е. устанавливается диапазон выбора (минимум - максимум).

Копирование / переименование / удаление файла или директории

```
<?php

unlink( "file.txt" ); // Удаление файла
rmdir( "folder/" ); // Удаление директории. Внимание,
папка должна быть пустой
copy( "otkuda.txt" , "kuda.txt" ); // Копирование.
rename( "chto.txt" , "vo_chto.txt" ); //
Переименование.

?>
```

Права доступа (CHMOD) и их изменение

Права доступа показывают, какие операции (чтение, запись, выполнение) с файлом (директорией) может выполнять пользователь.
Права доступа определяются для 3 пользователей:
1. Хозяина (создавшего файл).
2. Группы, в которую входит хозяин файла.
3. Остальные пользователи.

Права доступа могут быть записанны как в буквенном, так и в символьном варианте.

В буквенном: drwxr-x-r-x (стандартные права для директорий).Первый символ - специальный, показывающий чем этот файл является (в UNIX системах все представленно ввиде файлов, даже директории). d - директория.Затем идут три комбинации, rwx - права для хоязина, r-x - права для группы, r-x - права доступа для отсальных пользователей.
r - пользователь имеет право чтения файла (по сути просто обратиться к нему).
w - пользователь имеет право записать / перезаписать файл.
x- показывает, что файл может быть исполнен (актуально для CGI сценариев).

В числовом варианте права каждого пользователя определяет цифра, которая складывается из суммы:
r - 4, w - 2, x -1. Таким образом, drwxr-xr-x - 755.

Изменить права доступа можно либо с помощь FTP клиента, либо через shell. С помощью PHP права доступа меняются функцией chmod().

```
<?php

chmod( "file.txt" , 0755 ); // 0 - показывает, что это восьмеричная система.

?>
```

В Операционных Системах Windows права доступа всегда 777.

Получение информации о файле

Информацию о файле можно получить с помощью функции:

```
<?php

print_r( stat( "file.txt" ) );

?>
```

0 - dev - устройство
1 - ino - inode - отедьная функция: fileinode();
2 - mode - inode protection mode
3 - nlink - number of links
4 - uid - идентификатор хоязина - отедьная функция: fileowner();
5 - gid - идентификатор группы - отедьная функция: filegroup();
6 - rdev - device type, if inode device *
7 - size - размер - отедьная функция: filesize();
8 - atime - время последнего доступа к файлу (Unix time) - отедьная функция: fileatime();

9 - mtime - время последней модификации файла (Unix time) - отедьная функция: filemtime();

10 - ctime - время создания файла (Unix time) - отедьная функция: filectime();

11 - blksize - blocksize of filesystem IO *

12 - blocks - number of blocks allocated

Аналогом данной функции является fstat(), но работает она с открытым указателем.

Пример 6. Редактировать определенной строки в файле

```
<?php

 $f_arr = file( "file.txt" );

 $needle = 2; // Какую строчку нужно отредактировать
              // Не забывайте, нумерация начинается с
нуля.

 $f_arr[ $needle ] = "Новое значение";

 $f = fopen( "file.txt" , "w" );

 for( $i = 0; $i < count( $f_arr ); $i++ )
  {  fwrite( $f , $f_arr[$i] . "\n" );  }

 close($f);

?>
```

Пример 7. "Обнуление" (очистка) содержимого файла

```
<?php

$f = fopen( "file.txt" , "r" );

ftruncate( $f , 0 );

fclose($f);

?>
```

Функции для работы со строками

Строки в PHP – один из самых универсальных объектов. К строке можно привести любой объект с помощью функции **Serialize( )**

1. Форматирование строк

Часто строки, вводимые пользователем (как правило, из интерфейса HTML-формы), придется приводить в порядок, прежде чем их можно будет использовать.

Некоторые пользователи указывают много пробельных символов (пробелы, символы табуляции) в начале и вконце строки. Для удаления таких символов используется функция **Trim()**. Функция не изменяет исходную строку, а работает с ее копией, поэтому если Вам нужно изменить саму строку, то нужно использовать оператор:

```
$str =trim($str);
```

Функции **Chop( )** и **Ltrim( )** и работают аналогично, но удаляют пробельные символы только в конце и начале строки соответственно.

2. Форматирование строк для печати

До сих пор мы использовали языковую конструкцию echo для вывода строк в окне браузера.

PHP поддерживает также функцию print(), выполняет ту же задачу, что и echo, но поскольку она является функцией, то возвращает значение (0 или 1, в зависимости от успешности выполнения).

Обе эти конструкции выводят строку "как есть". Используя функции printf() и sprintf(), можно применять несколько более сложное форматирование. В основном, эти функции работают практически одинаково, но printf() выводит сформатированную строку в окне браузера, а sprintf() возвращает сформатированную строку.

Например, в конструкции echo мы использовали переменные, которые нужно вывести в строке, подобно следующему:

```
echo "Total amount of order is $Total.";
```

Для получения того же результата с помощью функции printf() следовало бы применить

```
printf ("Total amount of order is %s.", $total);
```

Последовательность %s в строке формата называется спецификацией преобразования. Приведенная спецификация означает "заменить строкой". В данном случае она будет заменена значением переменной $total, интерпретируемым в качестве строки. Если бы значением, хранящимся в

переменной $total, было 12.4, оба приведенные выражения привели к выводу 12.4.

Преимущество использования функции printf() в том, что можно использовать более удобную спецификацию преобразования для указания того, что в действительности $total — число с плавающей точкой и оно должно содержать два знака после десятичной точки, как показано в следующем примере:

```
printf ("Total amount of order is %.2f", $total);
```

Строка формата может содержать несколько спецификаций преобразования. При использовании n спецификаций преобразования после строки формата необходимо указать n аргументов. Каждая спецификация преобразования будет замещена переформатированным аргументом в том порядке, в каком они перечислены.

Пример 8.

```
printf ("Total amount of order is %.2f (with shipping
%.2f) ", $total, $total_shipping);
```

В данном случае первая спецификация преобразования будет использовать переменную $total, а вторая — переменную $total_shipping.

Все спецификации преобразования имеют одинаковый формат

```
%['дополняющяй_символ'][-][ширина][.точность]тип
```

Все спецификации преобразования начинаются с символа %. Если требуется вывести символ %, потребуется применить последовательность %%.

Дополняющий_символ необязателен. Он будет использоваться для дополнения переменной до указанной ширины. Примером может служить добавление ведущих нулей к такому числу, как значение счетчика.

Символ "-" также является необязательным. Он указывает, что данные в поле будут выравниваться по левому краю, а не по правому, как определено по умолчанию. Параметр ширина указывает функции printfQ, сколько места (в символах) необходимо для подстановки значения переменной.

Параметр точность должен начинаться с десятичной точки. Он определяет количество отображаемых десятичных знаков после запятой.

Последним параметром спецификации является код типа. Краткое описание используемых кодов приведено в табл. 3.

Таблица 3. Коды типа спецификации преобразования

Тип	**Описание**
b	Интерпретируется как целое число и выводится как двоичное число.
c	Интерпретируется как целое число и выводится как символ.
d	Интерпретируется как целое число и выводится как десятичное число.
f	Интерпретируется как число двойной точности и выводится как число с плавающей точкой.
o	Интерпретируется как целое число и выводится как восьмеричное число.
s	Интерпретируется как строка и выводится как строка.
x	Интерпретируется как целое число и выводится как шестнадцатеричное число с представлением цифр a-f строчными буквами.
X	Интерпретируется как целое число и выводится как шестнадцатеричное число с представлением цифр A-F прописными буквами.

3. Определение длины строки

Функция **strlen** используется очень часто, поскольку значение длины строки требуется для многих ее преобразований. Использовать ее просто – передайте ей строку, а функция вернет количество символов, например:

```
$str="Hello";
echo strlen($str);//5
```

4. Поиск строки внутри другой строки

Для поиска строки внутри другой строки можно использовать любую из функций s**trstr()**, **strchr()**, **strrchr()** или **stristr()**.

- s**trstr()**

Синтаксис:

```
string strstr(string haystack, string needle)
```

Функция **strstr()** возвращает участок строки, заданной в параметре **haystack**, начиная с первого фрагмента, указанного в параметре **needle** и до конца строки. В случае неудачи функция возвращает **false**.

Пример 9.

```
<?
   $url = "http://www.softtime.ru";
   $www = strstr($url,"w");
   echo ($www);
?>
```

Рассмотренный сценарий выводит:

```
www.softtime.ru
```

Эта функция чувствительна к регистру. Заметим также, что в случае, если **needle** не является строкой, то значение преобразуется в целое и используется как код искомого символа.

- **strchr()**

Синтаксис:

```
string strchr(string haystack, string needle)
```

Данная функция работает абсолютно идентично функции **strstr()**

- **stristr()**

Синтаксис:

```
string stristr(string haystack, string needle)
```

Эта функция работает абсолютно аналогично функции **strstr**(), только является нечувствительной к регистру.

- **strrchr()**

Синтаксис:

```
string strrchr(string haystack, string needle)
```

А эта функция отличается от аналогичных ей тем, что осуществляет поиск последнего вхождения подстроки. Т.е. функция **strrchr**() возвращает участок строки, заданной в параметре **haystack**, начиная с последнего фрагмента,

указанного в параметре **needle** и до конца строки. В случае неудачи возвращает **false**.

Чувствительна к регистру. В случае, если **needle** не является строкой, то значение преобразуется в целое и используется как код искомого символа.

5. Функции работы с блоками текста

- **wordwrap()**

Синтаксис:

```
string wordwrap(string str [, int width [, string break [, int cut]]])
```

Функция **wordwrap()** разбивает исходный текст на строки с определенными завершающими символами. Согласно синтаксису, эта функция разбивает блок текста **str** на несколько строк, которые завершаеются символами **break**, так, чтобы в одной строке было не более **width** букв. Поскольку разбиение происходит по границам слов, текст остается вполне читаемым.

Пример 10.

```
<?
   $str = "Вставай, страна огромная";
   $mod_str = wordwrap($str,5,"\t");
   echo($mod_str);
?>
```

- **str_replace()**

Синтаксис:

```
string str_replace(string from, string to, string str)
```

Функция **str_replace()** заменяет в исходной строке str одни подстроки на другие. Т.е. функция заменяет в строке **str** все вхождения подстроки **from** на **to** и возвращает результат. Эта функция может работать с двоичными строками.

К примеру, если Вы пишите что-то типа гостевой книги, форума, и хотите, чтобы в форме ввода для выделения теста можно было пользоваться стандарными тегами HTML, Вы можете с помощью этой функции заменить символы, которые Вы выбрали для форматирования на стандартные теги HTML. К примеру:

```
$txt = str_replace("[B]","<B>",$txt);
```

Т.е. если Вы используете для отображения текста полужирным шрифтом символы "[B]", Вы должны их заменить на символ "<B>", используемые в HTML.

6. Сравнение строк

Функции **strcmp( )** и **strcasecmp( )** используются для сравнения строк. Первая функция осуществляет сравнения строк с учетом регистра символов, а вторая – без учета. Функции возвращают следующие значения:

- 0, если лексикографически строки равны;
- -1,если первая строка лексикографически меньше, чем вторая (например строка «Арбуз» лексикографически меньше, чем строка «Борис»);

- 1,если первая строка лексикографически больше, чем вторая (например строка «Борис2», лексикографически больше, чем «Борис2»).

7. Работа с подстроками

Функция **substr ( )** возвращает подстроку, рассмотрим ее прототип:

```
string substr (string строка, int начало [, int длина]
```

Первый параметр – это исходная строка. Второй параметр – начало подстроки, а третий параметр (он необязательный) – длина подстроки. Если третий параметр не задан, то подстрока возвращается до конца исходной строки.

```
$sub=substr("машина",4);//возвращает "на"
$sub=substr("машина",2,4);//возвращает "шина"
```

Очень мощной функцией явнляется функция **explode()**, позволяющая извлечь все подстроки, разделенные разделителем. Предположим, что у нас есть такая строка:

```
Директива1; Директива2; Директива3
```

```
$str="Директива1; Директива2; Директива3"
```

При вызове **explode()** мы должны указать два параметра: разделитель и исходная строка.

```
$A= explode(";",$str)
```

Задания для самостоятельной работы

1. Выбросить из текста, находящегося в файле, заданный знак, где бы он не встречался.
2. Удалить из текста, находящегося в файле, каждое четное предложение.
3. В тексте перед каждым предложением, в котором встречается заданное слово, поставить восклицательный знак "!"
4. Заменить заданное слово в тексте, находящемся в заданном файле, на другое слово.
5. Напечатать самое длинное слово, найденное в тексте, находящемся в заданном текстовом файле.
6. Перед каждым символом поставить порядковый номер в тексте. Текст находится в заданном текстовом файле. Если строка начинается с цифры, то перед ней поставить левую круглую скобку, а в конец строки – правую круглую скобку. Текст находится в заданном текстовом файле.
7. Напишите программу, которая дописывает в находящийся на диске файл имя, фамилию и номер телефона, например, вашего товарища. Если файла на диске нет, то программа должна создать его. В файле каждый элемент данных (имя, фамилия, телефон) должен находиться в отдельной строке.
8. Напишите программу, которая дописывает в находящийся на диске файл имя, фамилию и номер телефона, например, вашего товарища. Если файла на диске нет, то программа должна создать его. В файле все записи должны находиться последовательно в одной строке.
9. Определить, какая буква чаще всего встречается в тексте, находящемся в заданном текстовом файле.
10. Определить, встречается ли заданное слово в текстовом файле. Если да, то сколько раз.

.

Лабораторная работа № 14-16 [Работа с базой данных MySQL средствами PHP]

Цель работы: Приобретение навыков работы с БД MySQL средствами PHP.

Задачи:

1. Изученить функции, позволяющие работать с базой данных MySQL средствами PHP. Соединение с сервером и его разрыв.
2. Формирование навыков создания и выбора базы данных. Доступ к отдельному полю записи.
3. Формирование навыков создания запросов, отправляемые серверу MySQL.

План:

1. Запуск Денвера.
2. Ознакомление с теоретической частью.
3. Выполнение и отладка примеров программ из экспериментальной части.
4. Выполните задания 1,2,3 для самостоятельной работы в соответствии со своим вариантом.

Выполнение заданий для самостоятельной работы

Теоретическая часть

Язык PHP позволяет работать практически со всеми известными СУБД. Для этого необходимо знать способ подключения к этой СУБД и стандартные функции PHP для работы. Рассмотрим работу в PHP с СУБД MySQL.

MySQL - быстрый многопоточный, многопользовательский надежный SQL-сервер баз данных. (SQL - язык структурированных запросов). Сервер

MySQL предназначен для разнообразных систем - от маленьких сайтов до крупных интернет-проектов. Сейчас это самый распространенный сервер баз данных в сети Интернет. MySQL - это система управления реляционными базами данных. В реляционной базе данных данные хранятся в отдельных таблицах, благодаря чему достигается выигрыш в скорости и гибкости. Таблицы связываются между собой, есть возможность объединять при выполнении запроса данные из нескольких таблиц.

Для выполнения упражнений, приведенных в данной лабораторной работе, потребуется доступ к MySQL-серверу.

В качестве интерфейса для MySQL используется phpMyAdmin – это веб-приложение позволяющее администрировать базы данных MySQL через удобный веб-интерфейс.

MySql базы данных – основной инструмент для создания динамических сайтов.

Принцип работы заключается в следующем: создается HTML-каркас сайта и в определенные места каркаса (например, в область основного содержимого) посредством PHP-скриптов из базы данных выводится информация, которая и формирует контент сайта.

Экспериментальная часть

Учебная база данных

В приводимых в лабораторной работе примерах построения SQL-запросов и упражнениях для самостоятельной работы используется база данных, состоящая из следующих таблиц

Таблица 1. STUDENT (Студент)

STUD_ID	SURNAME	NAME	STIPEND	KURS	CITY	BIRTH DAY	UNIV_ID
1	Иванов	Иван	150	1	Орел	3/12/1982	10
3	Петров	Петр	200	3	Курск	1/12/1980	10
6	Сидоров	Вадим	150	4	Москва	7/06/1979	22
10	Кузнецов	Борис	0	2	Брянск	8/12/1981	10
12	Зайцева	Ольга	250	2	Липецк	1/05/1981	10
265	Павлов	Андрей	0	3	Воронеж	5/11/1979	10
32	Котов	Павел	150	5	Белгород	NULL	14

STUD_ID – числовой код, идентифицирующий студента,

SURNAME – фамилия студента,

NAME – имя студента,

STIPEND – стипендия, которую получает студент,

KURS – курс, на котором учится студент,

CITY – город, в котором живет студент,

BIRTHDAY – дата рождения студента,

UNIV_ID – числовой код, идентифицирующий университет, в котором учится студент.

Таблица 2. LECTURER (Преподаватель)

SUBJ_ID	SUBJ_NAME	HOUR	SEMESTER
10	Информатика	56	1
22	Физика	34	1
43	Математика	56	2
56	История	34	43
94	Английский	56	5

Таблица 3. SUBJECT (Предмет обучения)

SUBJ_ID	SUBJ_NAME	HOUR	SEMESTER
10	Информатика	56	1
22	Физика	34	1
43	Математика	56	2
56	История	34	4
94	Английский	56	3
73	Физкультура	34	5
……	……	……	……

SUBJ_ID – идентификатор предмета обучения,

SUBJ_NAME – наименование предмета обучения,

HOUR – количество часов, отводимых на изучение предмета,

SEMESTER – семестр, в котором изучается данный предмет.

Таблица 4. UNIVERSITY (Университеты)

UNIV_ID	UNIV_NAME	RATING	CITY
22	МГУ	606	Москва
10	ВГУ	296	Воронеж
11	НГУ	345	Новосибирск
32	РГУ	416	Ростов
14	БГУ	326	Белгород
15	ТГУ	368	Томск
18	ВГМА	327	Воронеж
……	……	……	……

UNIV_ID – идентификатор университета,

UNIV_NAME – название университета,

RATING – рейтинг университета,

CITY – город, в котором расположен университет.

Таблица 5. EXAM_MARKS (Экзаменационные оценки)

EXAM_ID	STUD_ID	SUBJ_ID	MARK	EXAM_DATE
145	12	10	5	12/01/2000
34	32	10	4	23/01/2000
75	55	10	5	05/01/2000
238	12	22	3	17/06/1999
639	55	22	**NULL**	22/06/1999
43	6	22	4	18/01/2000

EXAM_ID – идентификатор экзамена,

STUDENT_ID – идентификатор студента,

SUBJ_ID – идентификатор предмета обучения,

MARK – экзаменационная оценка,

EXAM_DATE – дата экзамена.

Таблица 6. *SUBJ_LECT*

(Учебные дисциплины преподавателей)

LECTURER _ID	SUBJ_ID
24	10
46	22
74	43
108	56
276	94
328	73
……	……

LECTURER_ID – идентификатор преподавателя,

SUBJ_ID – идентификатор предмета обучения.

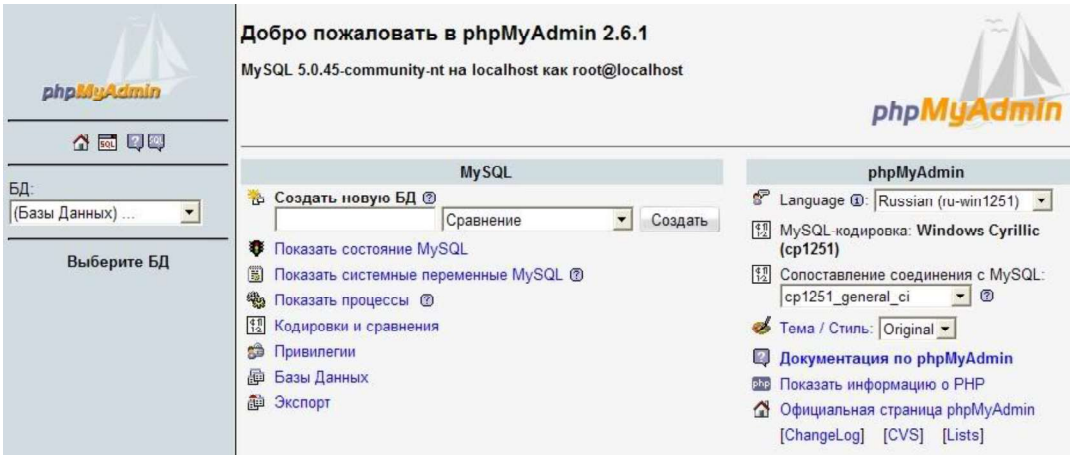
Создание БД и таблицы

«Добро пожаловать в phpMyAdmin»

Запускаем Денвер. Для запуска используем ярлык Start Denwer. В адресной строке браузера набираем http://localhost. Прокручиваем загрузившуюся страницу до списка ссылок.

URL	Описание
https://subdomain.localhost/ssl.php	Проверка SSL
http://subdomain.localhost/	Проверка "не-Интернет" доменов второго уровня, а также SSI
http://test1.ru/	Проверка "Интернет"-доменов второго уровня: test1.ru (вначале отключите прокси-сервер!)
http://subdomain.test1.ru/	Проверка "Интернет"-доменов третьего уровня
http://localhost/Tests/phpnotice/index.php	Проверка перехвата PHP Notice в Денвере
http://localhost/Tests/PHP5/index.php5	PHP5 information
http://localhost/Tools/phpMyAdmin	Проверка MySQL и phpMyAdmin
http://custom-host:8648	Проверка хоста с другим IP-адресом и портом (127.0.0.2:8648) *В Windows XP SP2 имеется ошибка, из-за которой данный хост может не работать. Официальную "заплатку" от Microsoft качайте здесь.*
http://localhost/Tests/sendmail/index.php	Проверка отладочной заглушки для sendmail

Нас интересует ссылка **http://localhost/Tools/phpMyAdmin.** Кликнув по ней, загрузится phpMyAdmin.



Создание пользователя для базы данных

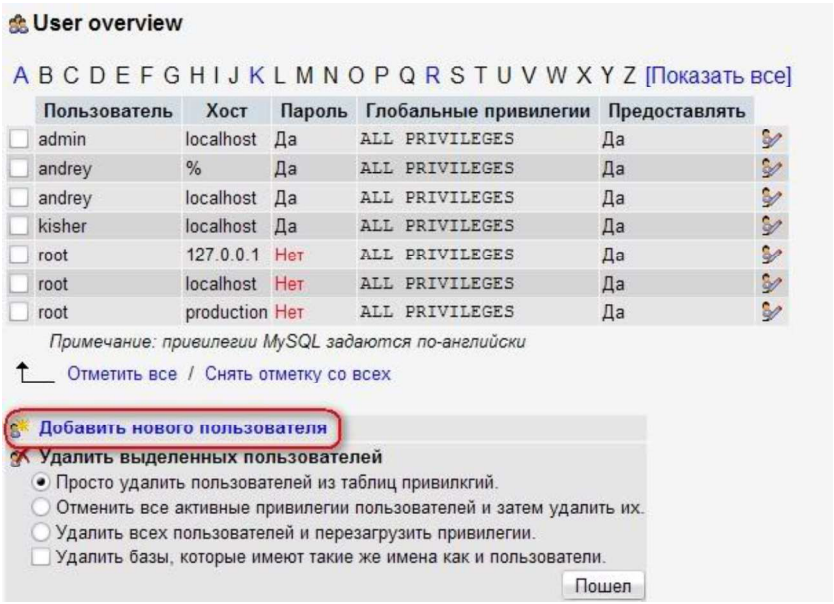
Задание 1. Создайте пользователя для БД.

Пользователь имеет определенные привилегии (может редактировать, удалять, создавать новые таблицы и позиции в базе данных) и авторизуется по заданному логину и паролю.

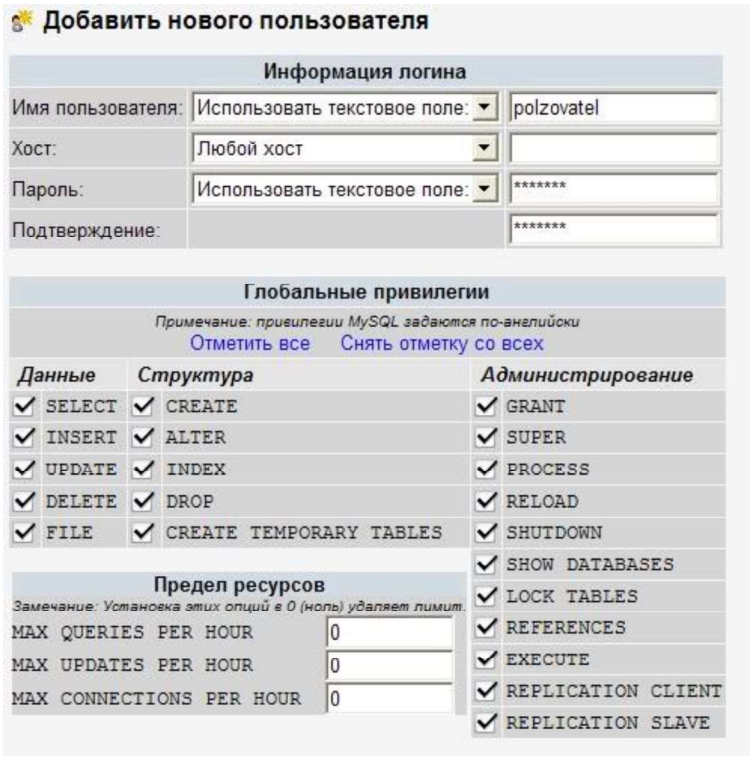
При покупке хостинга у большинства хостинг-провайдеров для вас автоматически создается база данных и аккаунт пользователя с логином и паролем, имеющий все необходимые привилегии. На нормальном хостинге есть точно такой же phpMyAdmin посредством которого вы сможете управлять таблицами и их содержимым, а также импортировать данные из локальной базы данных в базу данных хостинга. Логин и пароль для доступа к базе данных, а также ссылка на phpMyAdmin высылается хостинг-провайдером на ваш e-mail.

На локальном же компьютере нам придется самим создать пользователя.

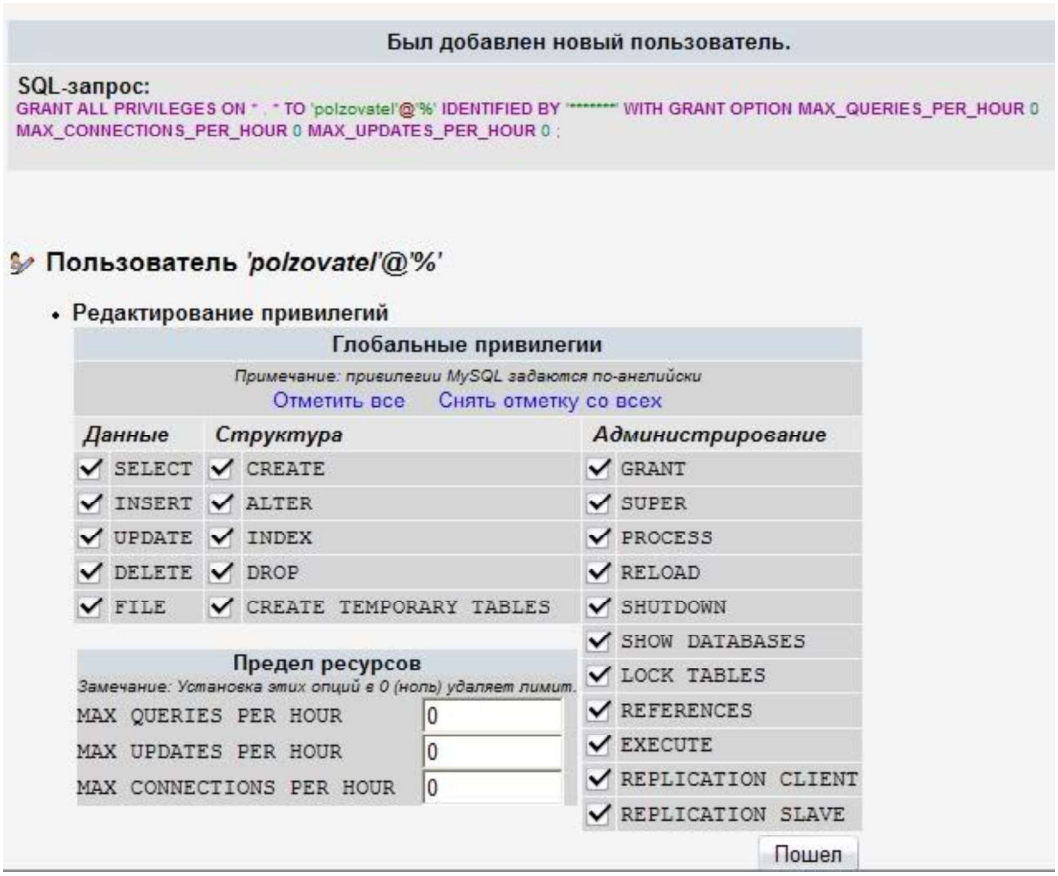
Кликаем по ссылке «Привилегии» в окне phpMyAdmin и попадаем на страницу с перечислением всех аккаунтов пользователей баз данных.На странице «Привилегии» кликаем по ссылке «Добавить нового пользователя».



На странице добавления пользователя вводим логин и пароль и выбираем все привилегии.



Нажимаем кнопку «Пошел» и если все ОК, то мы видим подтверждение создания нового пользователя.



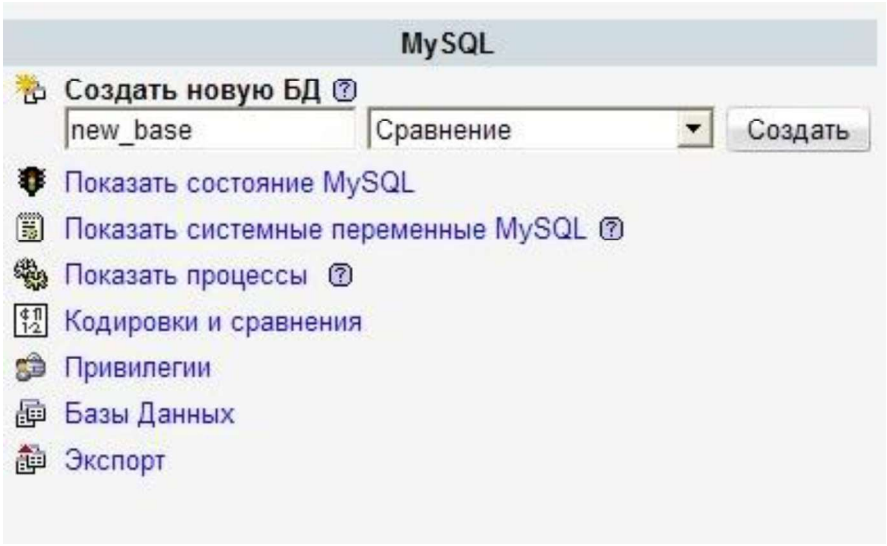
Теперь, когда мы будем работать с php-скриптом и подключаться при помощи него к базе данных, нужно будет использовать логин и пароль пользователя, указанные при его создании (главное не забыть, что мы там написали при создании пользователя).

Создадие БД

Задание 2. Создайте БД

Переходим на главную страницу phpMyAdmin-а

Указываете имя базы данных на латинице, в качестве “Сравнения” указываете utf8_unicode_ci и нажимаем кнопку создать.



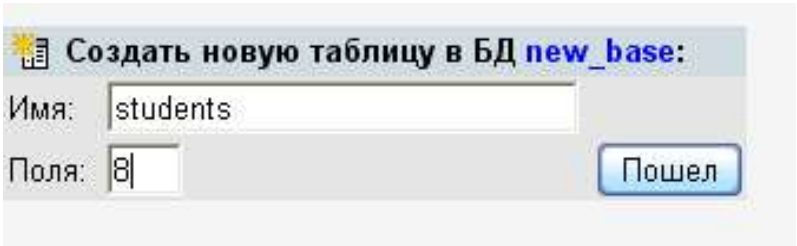
Вот и все, база создана.

Создадие таблиц в БД.

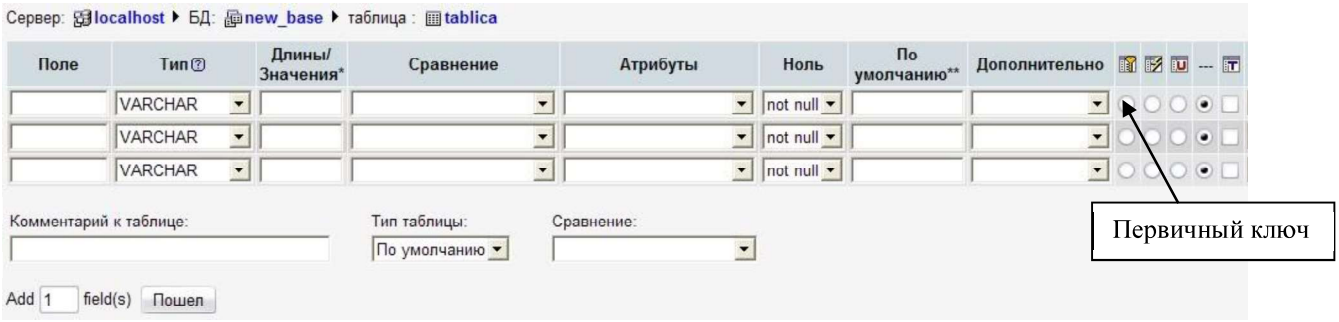
Задание 3. Создайте таблицу student в БД

Информация в базе данных хранится не просто так, а в таблицах. Поэтому нам нужно создать хотя бы одну таблицу с некоторым количеством полей. Для этого воспользуемся полем «Создать новую таблицу в БД».

Задаем таблице название и указываем кол-во полей в этой таблице.



После нажатия кнопки «Пошел» загружается страница создания полей в базе данных.



Заполните следующие столбцы: «Поле», «Тип», «Длины Значения». **В любой таблице обязательно должно присутствовать служебное поле называемое полем первичного ключа.** Это поле позволяет нам нумеровать строки в таблице и потом обращаться к определенной строке по ее значению в поле первичного ключа (в нашем случае (таблица student) роль поля первичного ключа играет поле stud_id).

Просмотреть информацию, записанную в таблицу, можно кликнув по вкладке Обзор.

Стандартные функции PHP для работы с MySQL

Алгоритм взаимодействия PHP с MySQL имеет следующий вид:

1. Установить соединение с сервером MySQL. Если попытка завершается неудачей, вывести соответствующее сообщение и завершить процесс.
2. Выбрать базу данных сервера MySQL. Если попытка выбора завершается неудачей, вывести соответствующее сообщение и завершить процесс. Допускается одновременное открытие нескольких баз данных для обработки запросов.

3. Обработать запросы к выбранной базе (или базам).

4. После завершения обработки запросов закрыть соединение с сервером баз данных.

Подключения к серверу MySQL:

Функция mysql_connect() устанавливает связь с сервером MySQL.

Синтаксис функции mysql_connect( ):

```
int mysql_connect ([string хост   [:порт]
[:/путь//к/сокету] [, string имя   пользователя]
[,
string пароль])
```

В параметре «хост» передается имя хостового компьютера, указанное в таблицах привилегий сервера MySQL. Это имя используется для перенаправления запросов на web-сервер, на котором работает MySQL, поскольку к серверу MySQL можно подключаться в удаленом режиме. Наряду с именем хоста могут указываться необязательные параметры – номер порта, а также путь к сокету (для локального хоста). Параметры «имя_пользователя» и «пароль» должны соответствовать имени пользователя и паролю, заданным в таблицах привилегий MySQL. Если параметр «хост» не задан, mysql_connect( ) пытается установить связь с локальным хостом.

Пример 3.1. Открытие соединения с MySQL:

```
<?php
$link = mysql_connect("localhost", "root", "pass")
or die("Не могу подключиться");
print ("Соединение выполнено");
?>
```

В данном примере localhost – имя компьютера, root-имя пользователя, а pass–пароль.

Выбираем БД:

После успешного соединения с MySQL необходимо выбрать базу данных, находящуюся на сервере. Для этого используется функция mysql_select_db( ).

Синтаксис этой функции:

```
int mysql_select_db (string имя_базы_данных  [, int идентификатор_соединения])
```

Параметр «имя_базы_данных» определяет выбираемую базу данных, идентификатор которой возвращается функцией mysql_select_db( ). При наличии нескольких открытых соединений этот параметр должен указываться.

Пример 3.2 Выбор базы данных функцией mysql_select_db( ):

```
<?php
$link = mysql_connect("localhost", "root", "pass")
or die("Не могу подключиться" );
// сделать new_base текущей базой данных
mysql_select_db('new_base', $link) or die ('Не могу выбрать БД');
?>
```

Если в программе выбирается только одна база данных, сохранять ее идентификатор не обязательно. Однако при выборе нескольких баз

данных возвращаемые идентификаторы сохраняются, чтобы вы могли сослаться на нужную базу при обработке запроса. Если идентификатор не указан, используется последняя выбранная база данных.

Работа с запросами (рассмотрим далее в пункте 4 “Основы работы с БД”).

Закрытие соединения с сервером:

Функция mysql_close( ) закрывает соединение, определяемое необязательным параметром. Если параметр незадан, функция mysql_close( ) закрывает последнее открытое соединение.

Синтаксис функции mysql_close( ):

```
int mysql_close ([int идентификатор_соединения])
```

Задание 3. Создайте файл с расширением .php и наберите в нем следующий текст, заменяя необходимые параметры (пользователь, пароль, название БД), своими:

```
<html>
   <body>
      <?php
         $db = mysql_connect("localhost",
"root","12345") or die("Не могу подключиться" );
         mysql_select_db("mydb",$db) or die ('Не могу
выбрать БД');
         mysql_close($db);
      ?>
   </body>
</html>
```

Теперь рассмотрим построчно, что происходит в этой программе.

Функция mysql_connect() открывает связь с сервером баз данных MySQL. В качестве параметров мы указываем ей имя узла (localhost), на котором находится база данных, имя пользователя (root), под которым мы будем с ней работать, и пароль.

Имя узла localhost означает, что сервер MySQL находится на той же машине, что и сам Web-сервер с PHP - движком. В принципе ничто не мешает вам (имея права) обратиться к серверу MySQL, который находится на соседней машине или вообще на другом конце земного шара.

В результате выполнения этой функции получаем некое значение, которое присваиваем переменной $db. Эта переменная называется идентификатором соединения.

Соединившись с сервером выбираем базу данных, с которой будем работать (ведь на одном и том же сервере могут "крутиться" несколько баз данных). Это делается с помощью функции mysql_select_db(). В качестве параметров мы передаем название нужной нам базы данных и идентификатор соединения, полученный нами при выполнении предыдущей команды.

В результате выполнения функции mysql_select_db() мы получаем значение true или false. Если соединение с базой данных произошло успешно – true, если нет – false. Для того, чтобы наша программа-страница работала более интеллектуально, мы можем при желании проанализировать возвращаемое значение и если оно будет false, вывести хорошее информативное сообщение об ошибке.

Функция mysql_close($db) закрывает соединение.

Основы работы с БД

Операторы и команды MySQL

Структурированный язык запросов SQL позволяет производить различные операции с базами данных:

Далее мы последовательно рассмотрим все эти операторы.

Оператор выборки данных - SELECT

Пример 4.1 Вывод таблицы 1. STUDENTS в цикле.

```
<html>
   <body>
      <?php
$db = mysql_connect("localhost", "root","12345") or
die("Не могу подключиться" );
```

```
mysql_select_db("mydb",$db) or die ('Не могу выбрать
БД');
$result=mysql_query("SELECT * FROM students",$db);//
выбор всех записей из таблицы
$myrow= mysql_fetch_array($result);
do
{
echo "Студент N - ".$myrow['stud_id']."<br>";
echo  $myrow['surname']."<br>";
echo  $myrow['name']."<br>";
echo  $myrow['stipend']."<br>";
echo  $myrow['kurs']."<br>";
echo  $myrow['city']."<br>";
echo  $myrow['birthday']."<br>";
echo  $myrow['univ_id']."<br><br>";
}
while($myrow=mysql_fetch_array($result));
mysql_close($db);
      ?>
   </body>
</html>
```

Пояснение к данной программе:

Запрос формируется с помощью функции mysql_query (“Запрос”, идентификатор_соединения). Выполнение функции mysql_query дает в результате массив, который хранится в переменной $result.

Функции mysql_fetch_array в качестве параметра подается массив $result. Функция выбирает из него строку, которую мы записали в переменную $myrow и автоматически переходит на следующую строку. Вызвав снова

mysql_fetch_array, выберем следующую строку из массива, и так далее до тех пор, пока не достигнем конца массива. В этом случае mysql_fetch_array вернет значение false, которое послужит сигналом, что все записи выбраны и можно завершить цикл.

Теперь стоит задача как-то вывести в теле цикла полученную запись. Выбранный ряд хранится в переменной $myrow. Она также, как и $result, является массивом, только одномерным.

К каждому столбцу в массиве $myrow мы можем обратиться по его ключу, который заключается в квадратные скобки. Например, при первом проходе цикла, $myrow[‘surname’] равно "Иванов", при втором $myrow[‘city’] равно "Курск".

Пример 4.1 Вывод таблицы 1. STUDENTS в цикле, используя функцию printf( ).

```
<html>
   <body>
      <?php
         $db = mysql_connect("localhost",
"root","12345") or die("Не могу подключиться" );
         mysql_select_db("new_base",$db) or die ('Не
могу выбрать БД');
        $result=mysql_query(“SELECT * FROM
STUDENTS”,$db);
        $myrow= mysql_fetch_array($result);
do
{
printf("Студент - N %s<br>%s<br>%s<br><br>",
$myrow['stud_id'],$myrow['surname'],$myrow['name']);
}
```

```
while(mysql_fetch_array($result));
 mysql_close($db);
      ?>
   </body>
</html>
```

Примечание к данной программе:

Функция printf — Возвращает отформатированную строку.

Создание условий при запросе к базе

Пример 4.2 Создание условия при запросе к базе.

```
<?php
$db=mysql_connect("localhost","root","12345");
mysql_select_db("new_base",$db);
$result=mysql_query("SELECT * FROM students WHERE
stud_id='3' OR name='Вадим'",$db);//выбор из табл.
записей, где stud_id='3' или name='Вадим'
?>
```

Замечание. В качестве условия в разделе WHERE можно использовать сложные логические выражения, использующие поля таблиц, константы, сравнения (>, <, = и т.д.), скобки, союзы AND и OR.

Сортировка при запросе

Порядок сортировки выводимых записей можно задавать при помощи оператора ORDER BY:

Пример 4.3 Сортировка выводимых записей.

```
<?php
$db=mysql_connect("localhost","root","12345");
mysql_select_db("new_base",$db);
$result=mysql_query("SELECT * FROM students ORDER  BY
name",$db);//сортировка по имени
?>
```

Для того чтобы сортировка производилась в обратном порядке, в утверждении ORDER BY к имени заданного столбца, в котором производится сортировка, следует добавить ключевое слово DESC (убывающий).

Пример 4.4 Сортировка выводимых записей в обратном порядке.

```
<?php
$db=mysql_connect("localhost","root","12345");
mysql_select_db("new_base",$db);
$result=mysql_query("SELECT * FROM students ORDER  BY
name DESC",$db);//сортировка по имени в обратном
порядке
?>
```

Пример 4.5 Ограничение в выборке на количество выводимых записей – LIMIT.

```
<?php
$db=mysql_connect("localhost","root","12345");
mysql_select_db("new_base",$db);
```

```
$result=mysql_query("SELECT * FROM students ORDER  BY
name LIMIT 2 ",$db);//выбор из таблицы 2 первых записи,
табл. отсортирована по имени
?>
```

4.2 Оператор вставки данных - INSERT

INSERT INTO таблица (поле1, поле2) VALUES(‘значение1’,’значение2’)

Пример 4.7 Добавление информации в таблицу student

```
<html>
   <body>
      <?php
         $db = mysql_connect("localhost",
"root","12345",) or die("Не могу подключиться" );
         mysql_select_db("mydb",$db) or die ('Не могу
выбрать БД');
$result=mysql_query("INSERT INTO
students(stud_id,surname,name,stipend,kurs,city,birthda
y,univ_id) VALUES
(12,'Зайцева','Ольга',250,2,'Липецк','1.05.1981',10)",$
db);
mysql_close($db);
      ?>
   </body>
</html>
```

Пример 4.8 Добавление информации в таблицу student через форму:

Создадим html-форму – form_insert.php

```
<html>
<body>
<form action="obrabotchik.php" method="post"
name="form">
<p>Числовой код, идент-ий студента: <br> <input
name="stud_id" type="text" size="20"
maxlength="40"></p>
<p>Введите фамилию:<br> <input name="surname"
type="text" size="20" maxlength="40"></p>
<p>Имя:<br> <input name="name" type="text" size="20"
maxlength="40"></p>
<p>Стипендия: <br><input name="stipend" type="text"
size="20" maxlength="40"></p>
<p>Курс:<br> <input name="kurs" type="text" size="20"
maxlength="40"></p>
<p>Город:<br> <input name="city" type="text" size="20"
maxlength="40"></p>
<p>День рождения: <br><input name="birthday"
type="text" size="20" maxlength="40"></p>
<p>Числовой код, идент-ий студента:<br> <input
name="univ_id" type="text" size="20"
maxlength="40"></p>
<input name="submit" type="submit" value="занести
нового студента в базу">
</form>
</body>
</html>
```

PHP-скрипт, обрабатывающий данную форму (**obrabotchik.php):**

```
<html>
<body>
<?php
if(isset($_POST['stud_id']))
{
$stud_id=$_POST['stud_id'];
}
if(isset($_POST['surname']))
{
$surname=$_POST['surname'];
}
if(isset($_POST['name']))
{
$name=$_POST['name'];
}
if(isset($_POST['stipend']))
{
$stipend=$_POST['stipend'];
}
if(isset($_POST['kurs']))
{
$kurs=$_POST['kurs'];
}
if(isset($_POST['city']))
{
$city=$_POST['city'];
}
if(isset($_POST['birthday']))
{
```

```
$birthday=$_POST['birthday'];
}
if(isset($_POST['univ_id']))
{
$univ_id=$_POST['univ_id'];
}
$db=mysql_connect("localhost","kseniya","12345");
mysql_select_db("new_base",$db);
$result=mysql_query("insert into
students(stud_id,surname,name,stipend,kurs,city,birthda
y,univ_id)
 VALUES
('$stud_id','$surname','$name','$stipend','$kurs','$cit
y','$birthday','$univ_id')");
if ($result=='true')
{
echo"Информация в базу успешно добавлена";
}
else
{
echo"Информация в базу не добавлена";
}
?>
</body>
</html
```

4.3 Оператор изменения данных – UPDATE

UPDATE таблица SET поле1=’значение1’, поле2=’значение2’ WHERE условие

Пример 4.9 Редактирование данных таблицы student

```
<html>
   <body>
      <?php
         $db = mysql_connect("localhost",
"root","12345",) or die("Не могу подключиться" );
         mysql_select_db("mydb",$db) or die ('Не могу
выбрать БД');
$result=mysql_query("UPDATE students SET
surname='Егоров', name='Егор' WHERE stud_id=3");
mysql_close($db);
      ?>
   </body>
</html>
```

4.4 Оператор удаления данных – DELETE

DELETE FROM таблица WHERE условие

Пример 4.10 Удаление данных из таблицы

```
<html>
   <body>
      <?php
         $db = mysql_connect("localhost",
"root","12345",) or die("Не могу подключиться" );
         mysql_select_db("mydb",$db) or die ('Не могу
выбрать БД');
```

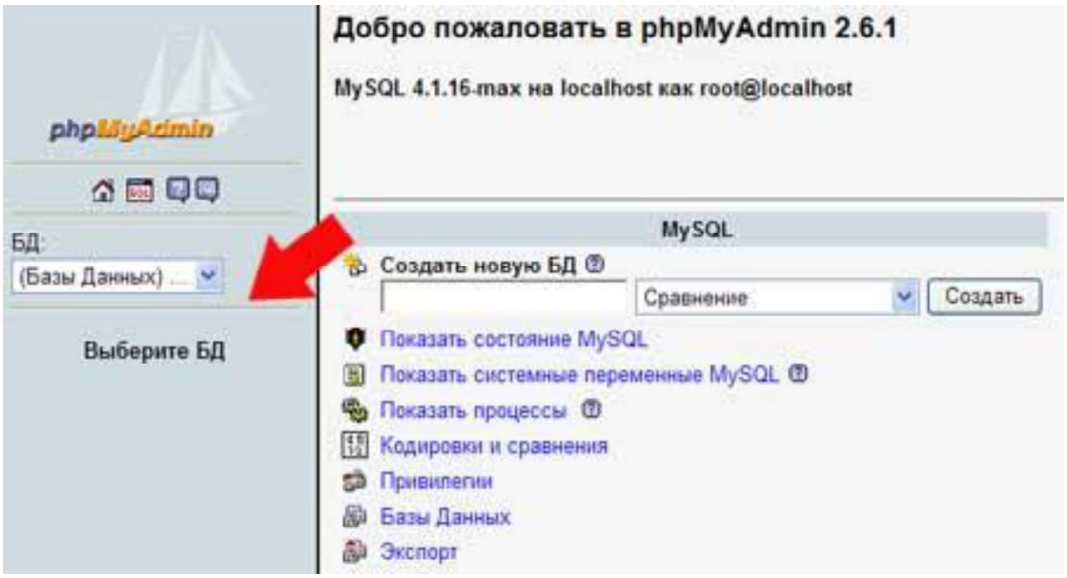
```
$result=mysql_query("DELETE FROM students WHERE
stud_id=1",$db );//из таблицы students удаляется
строка, где stud_id=1.
mysql_close($db);
      ?>
   </body>
</html>
```

Экспорт и импорт данных в phpMyAdmin

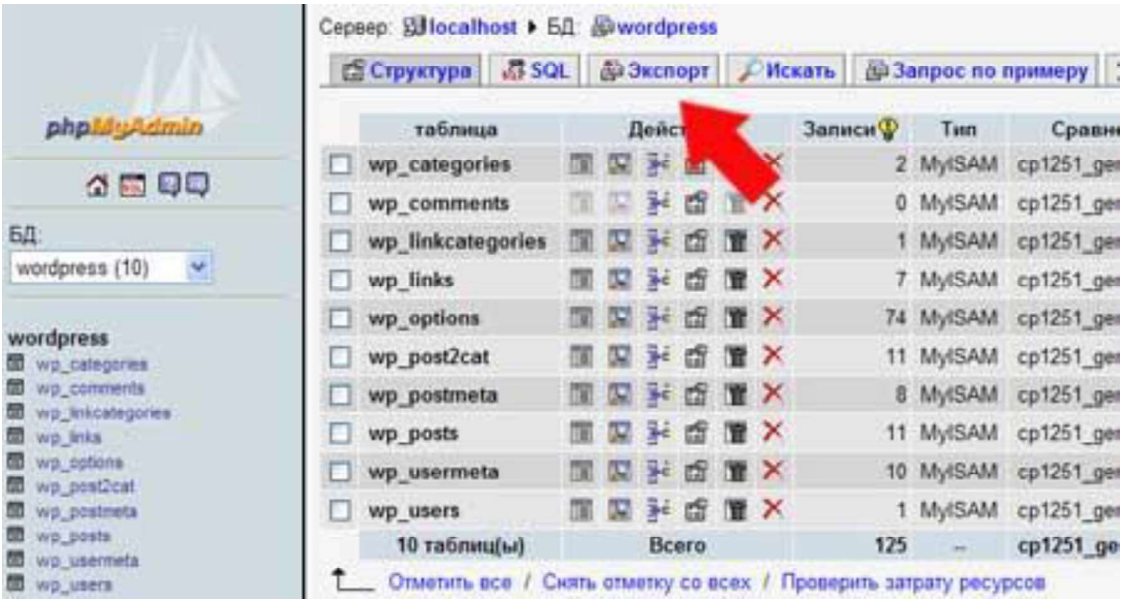
Создание **резервной копии базы данных** (экспорт) и её восстановлении (импорт) на примере **phpMyAdmin** будет полезно новичкам. Процесс это абсолютно не сложный, но очень полезный, особенно когда хостинг не поддерживает резервное копирование и всё приходится делать вручную.

Экспорт БД:

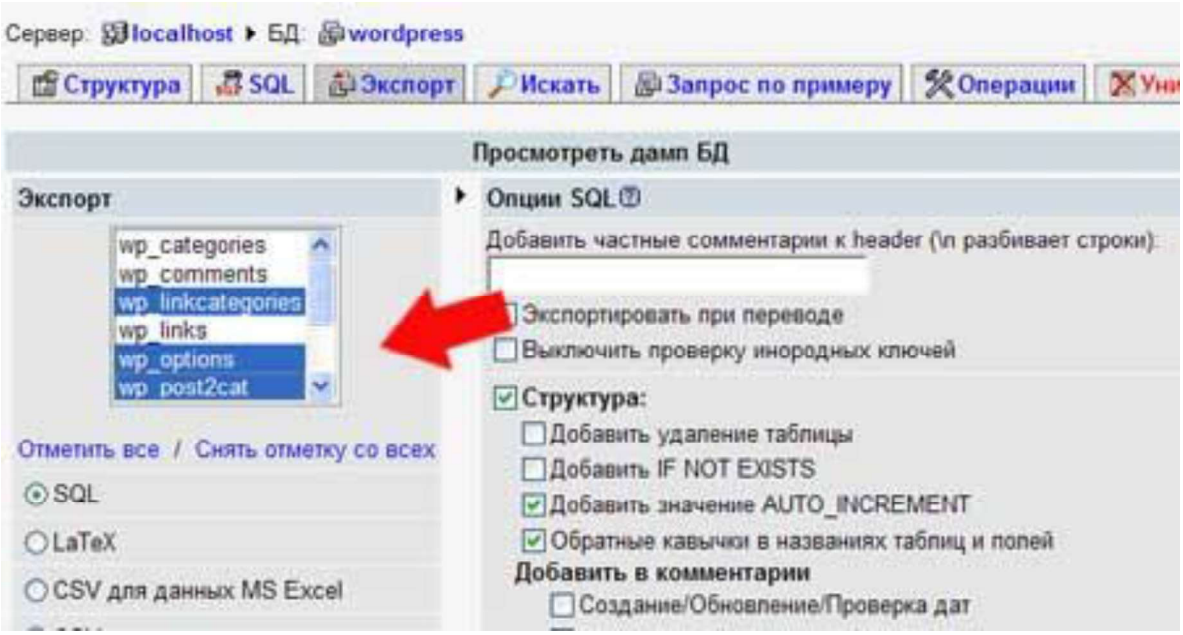
1. Заходим в phpMyAdmin и слева выбираем БД, которую нам предстоит экспортировать.



2. Выбираем “*Экспорт*”.

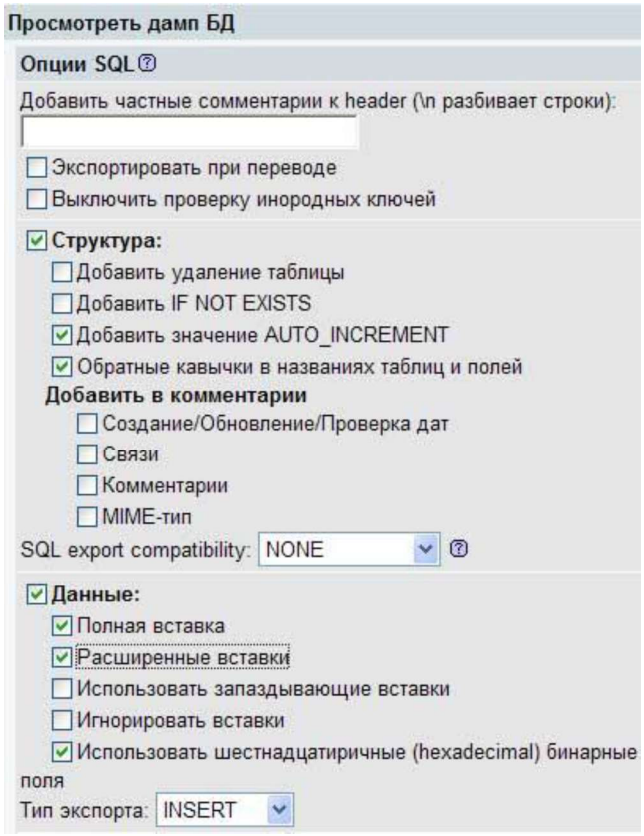


3. Отмечаем те таблицы, которые собрались экспортировать. Для выборочного выделения удерживаем Ctrl и кликаем левой кнопкой мыши по названиям таблиц. Если экспортируем всю БД полностью, просто нажимаем “*Отметить все*”. Убедимся, что данные экспортируются в виде SQL. Настройки, выставленные в phpMyAdmin по умолчанию, в большинстве случаев менять необходимости нет. Но могут быть варианты.

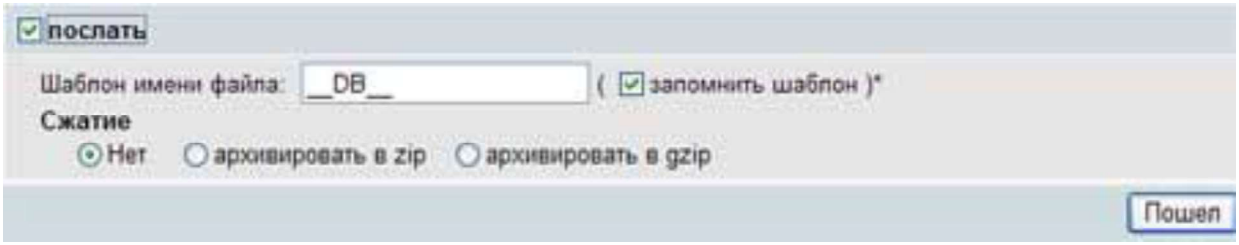


4. Более подробно остановимся на правой части открытой перед вами страницы.

- Если чекбокс “*Структура”* активирован, то в БД, в которую производится импорт, создаются таблицы, описанные в созданном вами дампе.
- В блоке “*Данные*” отмечаем чекбокс “*Данные*“. Этим действием мы указываем скрипту на необходимость создания дампа вместе с данными, содержащимися в БД. Не забудьте отметить чекбоксы “*Полная вставка*” и “*Расширенные вставки*“, иначе при импорте дампа будет создана таблица без данных.
- Тип экспорта: *INSERT* - если данные заносятся в БД впервые; *UPDATE* - производится обновление данных; *REPLACE* - производится замена данных.



5. Отмечаем чекбокс “*Послать*” и нажимаем “*Пошел*“.

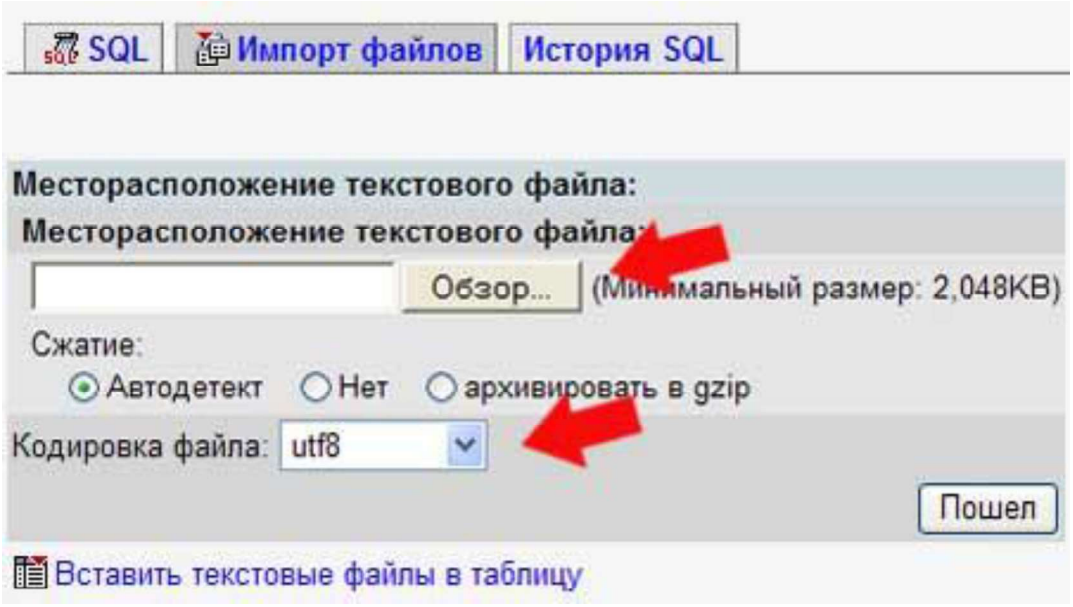


1. Сохраняем файл.

Импорт дампа (копии):

1. . Выбираем иконку *SQL*.

2. В появившемся окошке выбираем “*Импорт файлов*“, указываем кодировку импортируемого дампа, нажимаем кнопку “*Обзор*” и выбираем сохраненный ранее файл, нажимаем кнопку “*Пошел*“.



Если структура и синтаксис не имеют ошибок, процедуру экспорта/импорта можно считать завершенной.

Задания для самостоятельной работы

Задание 1.

1. Написать запрос, позволяющий вывести все строки таблицы EXAM_MARKS, в которых предмет обучения имеет номер (SUBJ_ID), равный 12.
2. Написать запрос, выбирающий все данные из таблицы STUDENT, расположив столбцы таблицы в следующем порядке: KURS, SURNAME, NAME, STIPEND.
3. Написать запрос SELECT, который выполняет вывод наименований предметов обучения (SUBJ_NAME) и следом за ним количества часов (HOUR) для каждого предмета обучения (SUBJECT) в 4-м семестре (SEMESTER).
4. Написать запрос, позволяющий получить из таблицы EXAM_MARKS

значения столбца MARK (экзаменационная оценка) для всех студентов, исключив из списка повторение одинаковых строк.

5. Написать запрос, который выполняет вывод списка фамилий студентов, обучающихся на третьем и более старших курсах.
6. Написать запрос, выбирающий данные о фамилии, имени и номере курса для студентов, получающих стипендию больше 140.
7. Написать запрос, выполняющий выборку из таблицы SUBJECT названий всех предметов обучения, на которые отводится более 30 часов.
8. Написать запрос, который выполняет вывод списка университетов, рейтинг которых превышает 300 баллов.
9. Написать запрос к таблице STUDENT для вывода списка фамилий (SURNAME), имен (NAME) и номера курса (KURS) всех студентов со стипендией большей или равной 100, живущих в Воронеже.
10. Получить список студентов старше 25 лет, обучающихся на 1-м курсе.

Задание 2.

1. Получить список предметов, изучаемых в 1-м семестре более 100 часов.
2. Получить список преподавателей, живущих в Воронеже.
3. Получить список университетов, расположенных в Москве и имеющих рейтинг меньший, чем у ВГУ. Константу в ограничении на рейтинг можно определить по этой же таблице.
4. Получить список студентов, проживающих в Воронеже и не получающих стипендию.

5. Получить список студентов моложе 20 лет.
6. Получить список студентов без определенного места жительства.
7. Написать запрос, который по таблице EXAM_MARKS позволяет найти:

   а) максимальные и б) минимальные оценки каждого студента – и выводит их вместе с идентификатором студента.
8. Написать запрос, выполняющий вывод списка предметов обучения в порядке:

   а) убывания семестров и б) возрастания отводимых на предмет часов.
9. Поле семестра в выходных данных должно быть первым, за ним должны следовать имя предмета обучения и идентификатор предмета.
10. Написать запрос, который выполняет вывод суммы баллов всех студентов для каждой даты сдачи экзаменов и представляет результаты в поряд- ке убывания этих сумм.

Задание 3.

1. Вывести список студентов, получающих максимальную стипендию, отсортировав его в алфавитном порядке по фамилиям.
2. Вывести список студентов, получающих стипендию, превышающую среднее значение стипендии.
3. Получить список студентов, учащихся в Воронеже, отсортировать по идентификаторам университетов и курсам.
4. Получить список предметов, на изучение которых отведено максимальное количество часов.
5. Написать запрос, выполняющий вывод имен и фамилий студентов, место проживания которых не совпадает с городом, в котором

находится их университет.

6. Получить список университетов, расположенных в Москве и имеющих рейтинг меньший, чем у ВГУ.
7. Получить список студентов, которые учатся в своем городе.
8. Получить список иногородних студентов (учащихся не в своем городе), отсортировать его по идентификаторам университетов и курсам.
9. Получить список преподавателей, работающих не в своем городе, отсортировать по идентификаторам университетов и городам проживания преподавателей.
10. Получить список предметов, на изучение которых отведено максимальное количество часов среди всех предметов, изучаемых в том же семестре. Список упорядочить по семестрам.

Итоговое проектное задание по PHP [Создание Web-приложения средствами языка PHP]

Задание. Создать Web-приложение средствами языка PHP, реализующее проект по данной теме.

Разработка проекта. Выбрать проект (либо из списка; либо предложить инициативный проект). Разработать структуру БД. Создать и заполнить БД. Разработать программное обеспечение проекта, обеспечивающее работу с информацией из БД как на стороне клиента, так и владельца ресурса.

Примерная тематика:

1. Гостевая книга.
2. Сборник задач.
3. Магазин.
4. Каталог книг.
5. Голосование.
6. Каталог продукции.
7. Сборник тестов.
8. Статистика посещения страниц сайта.
9. Каталог фотографий.
10. Каталог музыки.

Форма отчетности:

1. Описание проекта. Структура.
2. Локальная версия.
3. Скриншоты хода выполнения задания.