БЕЛКООПСОЮЗ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БЕЛОРУССКИЙ ТОРГОВО-ЭКОНОМИЧЕСКИЙ
УНИВЕРСИТЕТ ПОТРЕБИТЕЛЬСКОЙ КООПЕРАЦИИ»

Кафедра информационно-вычислительных систем

ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ

WEB-ПРОГРАММИРОВАНИЕ НА PHP

Практикум
к лабораторным занятиям
для студентов специальности 1-26 03 01
«Управление информационными ресурсами»

Гомель 2010

УДК 004
ББК 32.97
И 74

Авторы-составители: В. В. Бондарева, канд. техн. наук, доцент;
Д. В. Никитенко, ассистент

Рецензенты: О. А. Кравченко, канд. физ.-мат. наук, доцент
кафедры информационных технологий
Гомельского государственного технического
университета им. П. О. Сухого;
С. М. Мовшович, канд. техн. наук, доцент,
зав. кафедрой информационно-вычислительных
систем Белорусского торгово-экономического
университета потребительской кооперации

Рекомендован к изданию научно-методическим советом учреждения образования «Белорусский торгово-экономический университет потребительской кооперации». Протокол № 12 от 11 декабря 2007 г.

И 74 **Информационные** системы и технологии. Web-программирование на PHP : практикум к лабораторным занятиям для студентов специальности 1-26 03 01 «Управление информационными ресурсами» / авт.-сост. : В. В. Бондарева, Д. В. Никитенко. – Гомель : учреждение образования «Белорусский торгово-экономический университет потребительской кооперации», 2010. – 72 с.
ISBN 978-985-461-718-3

УДК 004
ББК 32.97

ISBN 978-985-461-718-3

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

С тех пор как Интернет стал использоваться в целях поддержки и развития бизнеса, основу электронного бизнеса стали составлять сайты организаций. Деятельность современной организации немыслима без использования ресурсов и сервисов Интернета. В связи с этим специалистам экономического профиля необходимы знания, позволяющие успешно внедрять интернет-технологии при решении бизнес-задач. Одним из наиболее популярных инструментов для создания Web-приложений является язык гипертекстовой разметки документов HTML и скриптовый язык PHP.

Изучение современных средств в разработке Web-приложений позволит улучшить интеграцию экономических задач в среде Интернета.

В практикуме рассматривается инструментарий для создания Web-приложений на языке PHP. Данный язык является одним из лучших языков для создания динамических Web-страниц.

PHP (Hypertext Preprocessor) – язык скриптинга (сценариев) общего назначения, который создан специально для Web и который можно внедрять в HTML. Основное назначение языка PHP – это выполнение на сервере сценариев, создающих динамические Web-страницы.

PHP может использоваться на всех крупных операционных системах, включая Linux, многие варианты Unix, Microsoft Windows, Mac OS X, RISC OS и др. PHP имеет поддержку для большинства существующих Web-серверов: Apache, Microsoft Internet Information Server, Personal Web Server, Netscape и iPlanet, Oreilly Website Pro, Caudium, Xitami, OmniHTTPd и многих других.

В PHP нет ограничений в выводе HTML. PHP может выводить изображения, PDF-файлы и даже клипы Flash, генерируемые на лету. Также можно выводить любой текст, включая XHTML, и любой XML-файл. PHP может автоматически генерировать эти файлы и сохранять их в файловой системе.

Одна из наиболее сильных и привлекательных черт PHP – поддержка им большого количества баз данных (БД). В настоящее время поддерживаются следующие БД:

Adabas D	IBM DB2	Oracle (OCI7 и OCI8)
dBase	Informix	Ovrimos
Direc Empresst MS-SQL	Ingres	PostgreSQL
Empress	InterBase	Solid
FilePro (read-only)	mSQL	Sybase
FrontBase	MySQL	Unix dbm
Hyperwave	ODBC	Velocis

PHP поддерживает взаимодействие с другими службами по протоколам IMAP, SNMP, NNTP, POP3, HTTP, COM (под Windows) и множеством других.

PHP имеет предельно удобные возможности для работы с текстом: от POSIX Extended или регулярных выражений Perl до разбора документов XML. При использовании PHP в области электронной коммерции можно использовать функции Cybercash-платежей, CyberMUT, VeriSign Payflow Pro и CCVS для онлайновых платежных программ.

1. ОСНОВЫ PHP

Как работает PHP-сценарий

Прежде чем узнать, как писать PHP-сценарии, важно понять, как выполняется их разработка. Когда клиент (например, Web-браузер) запрашивает документ с Web-сервера, то Web-сервер извлекает документ и отправляет клиенту. В большинстве случаев этот документ представляет собой HTML-файл, графический образ или нечто подобное. Клиент обрабатывает его и отображает в окне браузера.

А при использовании PHP-сценария добавляется еще одна промежуточная стадия – предварительная обработка. На этой стадии интерпретатор PHP обрабатывает запросы PHP-сценария, выполняет код, содержащийся в нем, и посылает вывод обратно Web-серверу, чтобы тот отправил его клиенту. Несмотря на то, что главная цель PHP-сценария состоит в генерировании HTML-содержимого, во время его выполнения может происходить все что угодно – от доступа к базе данных до отправки почтовых сообщений.

Существенное отличие этого языка программирования состоит в том, что весь код выполняется на сервере. Это означает, что для выполнения сценария клиенту не понадобится никаких специальных программ, подключаемых модулей или библиотек. До тех пор, пока клиент может нормально запрашивать документы с Web-сервера, он может пользоваться преимуществами языка сценариев PHP на этом сервере.

Базовый синтаксис PHP

Все PHP-сценарии пишутся в виде блоков кода. Эти блоки могут быть встроены в HTML и определяются с помощью специальных тегов, которые подразделяются на четыре стиля (листинг 1.1).

Листинг 1.1. Теги PHP

```
// 1 - теги в стиле XML
<?php
…
?>

// 2 - теги в стиле SGML
<?
…
?>

// 3 - теги в стиле сценариев HTML
<script language="php">
…
</script>

// 4 - теги в стиле ASP
<%
…
%>
```

В практикуме будет использован стиль XML. Кроме представленных в листинге 1.1 стилей, существуют сокращенные теги: <?=…?>. При таком написании тегов сначала выполняется заключенное в них выражение, а сам тег заменяется результатом выполнения этого выражения, например:

```
пять плюс пять равно <?=5+5?>
```

В результате браузер отобразит следующее:

```
пять плюс пять равно 10
```

Операторы

Каждый оператор в PHP заканчивается разделителем «;». Любое выражение, после которого следует символ «;», воспринимается как отдельный оператор. Операторы могут содержать переводы строк. В PHP используются простые операторы (листинг 1.2) и составные (листинг 1.3). Составные операторы – это последовательность простых операторов, заключенных в фигурные скобки. Составной оператор не оканчивается символом «;».

Листинг 1.2. Простые операторы

```
<?php
echo 5+5;
echo "Hello, world";
?>
```

Функция echo() применяется для вывода различной информации (текста, результата выполнения операции, тегов HTML и т. д.) в браузер.

Листинг 1.3. Составной оператор

```
<?php
{
echo 5+5;
echo "Hello, world";
}
?>
```

Для вывода информации можно использовать также функцию print(). Синтаксис:

```
print(string arg).
```

Функция выводит arg. При успехе возвращается TRUE, а при неудаче – FALSE. В действительности print() – это не функция, а конструкция языка, поэтому использование скобок не обязательно (листинг 1.4).

Листинг 1.4. Функция print()

```
<?php
print("Hello, world");
print "эта функция работает также без скобок";
?>
```

Комментарии

PHP предоставляет несколько способов для вставки комментариев (листинг 1.5). Комментарии PHP действуют только внутри тегов ограничителей PHP, вне тегов-ограничителей они будут отображены браузером.

Листинг 1.5. Комментарии

```
<?php
echo 5+5; // это однострочный комментарий
echo 5-2; # и это однострочный комментарий
echo "Hello, world";
/*
а это многострочный
комментарий
*/
?>
```

Переменные

В PHP имена переменных всегда начинаются с символа «$» и содержат произвольную комбинацию символов при условии, что первый символ после «$» будет буквой или знаком подчеркивания. Переменные в PHP чувствительны к регистру.

Листинг 1.6. Примеры переменных в PHP

```
<?php
$myvar="foo"; // Присвоение строки "foo"
badvar="test"; // Неверно, так как нет символа $
$5php="is wrong"; // Неверно, так как начинается с цифры
?>
```

Внешние переменные

Внешними переменными называются все переменные, которые поступают в программу из окружающего мира (предоставляются браузером и сервером). После того как запрос клиента проанализирован Web-сервером и передан PHP-интерпретатору, последний устанавливает ряд переменных, которые содержат данные, относящиеся к запросу.

Протокол HTTP, лежащий в основе Web, допускает передачу данных с помощью метода GET или POST. По умолчанию используется метод GET (листинг 1.7). Создадим HTML-форму[1] (рис. 1) в файле *index.htm*, которая будет состоять из одного текстового поля T1 и кнопки B1.

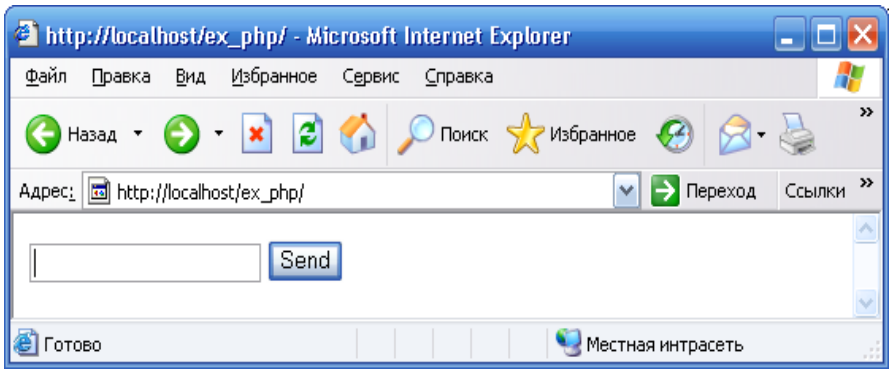

Рис. 1. **HTML-форма**

[1] Создание форм более подробно см. в приложении.

Листинг 1.7. HTML-форма, передающая данные методом GET

```
<form method="GET">
<input type="text" name="T1">
<input type="submit" name="B1" value="Send">
</form>
```

Если ввести в текстовое поле Hello и нажать кнопку Send, никаких видимых изменений не произойдет, но строка адреса в браузере примет вид, показанный на рис. 2.

Так как исходный файл называется *index.htm*, то его название в строке адреса опущено. Таким образом, через строку адреса передаются GET-параметры. Знак вопроса после имени файла обозначает начало передачи переменных методом GET. После него следуют пары *имя=значение*, разделенные символом амперсанда «&». В представленной строке передано два параметра с именами T1 и B1 и значениями Hello и Send соответственно. Получить доступ к GET-параметрам можно через суперглобальный массив $_GET, указав в квадратных скобках после него имя параметра.

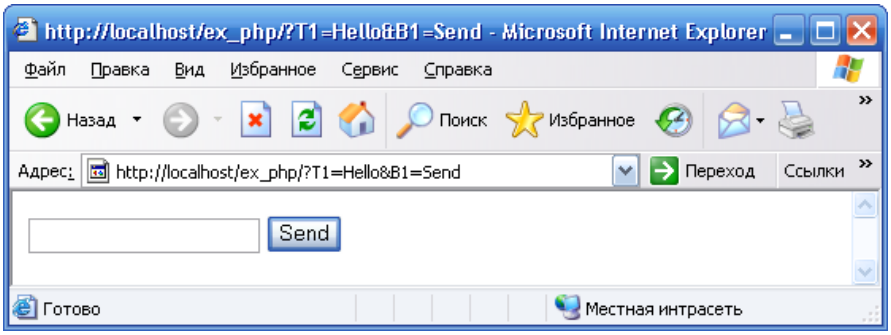

Рис. 2. **Передача данных методом GET**

Изменим файл *index.htm* так, чтобы данные, введенные в поле, выводились сразу после HTML-формы (листинг 1.8).

Листинг 1.8. Вывод GET-параметров

```
<form method="GET">
<input type="text" name="T1">
<input type="submit" name="B1" value="Send">
</form>
<?php
echo $_GET['T1'];
?>
```

Результат работы скрипта из листинга 1.8 представлен на рис. 3.

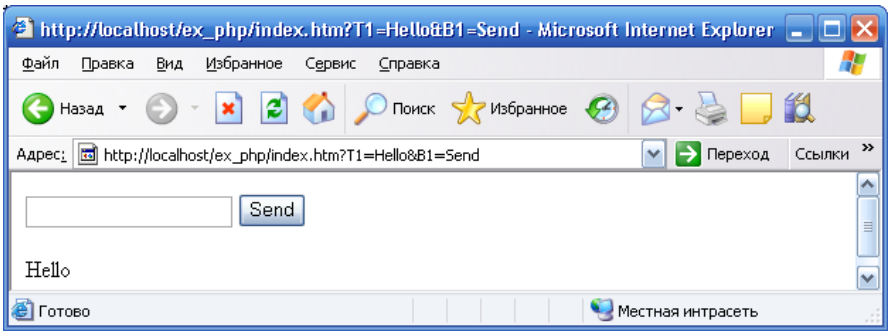

Рис. 3. **Вывод GET-параметра T1**

В рассмотренном примере HTML-форма и обработчик находятся в одном файле. Такой подход является удобным, когда форма и обработчик небольшие. В большинстве случаев удобно размещать обработчик HTML-формы в отдельном файле, для этого в тег <form> необходимо добавить атрибут action, который укажет имя обработчика формы. Пусть в качестве обработчика будет использоваться файл *second.php*.

Листинг 1.9. Использование атрибута action

```
<form method="GET" action="second.php">
<input type="text" name="T1">
<input type="submit" name="B1" value="Send">
</form>
```

Создав файл *second.php*, можно добиться того, чтобы данные из формы отправлялись обработчику в файл *second.php* (листинг 1.10).

Листинг 1.10. Обработчик second.php

```
<?php
echo $_GET['T1'];
?>
```

Передача данных методом GET не всегда является удобной по следующим причинам:
- пользователь может видеть значение параметров и легко подделывать их в строке запроса;
- объем передаваемой информации через GET-параметры ограничен (как правило, 8 кбайт).

GET-параметры передаются через HTTP-заголовки. Существует еще один способ передачи данных – через тело документа методом POST. Чтобы передать данные обработчику из формы методом POST, атрибуту method тега <form> необходимо присвоить значение POST (листинг 1.11).

Листинг 1.11. Передача данных методом POST

```
<form method="POST" action="next.php">
<input type="text" name="T1">
<input type="submit" name="B1" value="Send">
</form>
```

Для получения POST-параметров необходимо использовать суперглобальный массив $_POST.

Листинг 1.12. Обработчик next.php

```
<?php
echo $_POST['T1'];
?>
```

Результат работы скриптов из листингов 1.11 и 1.12 представлен на рис. 4.

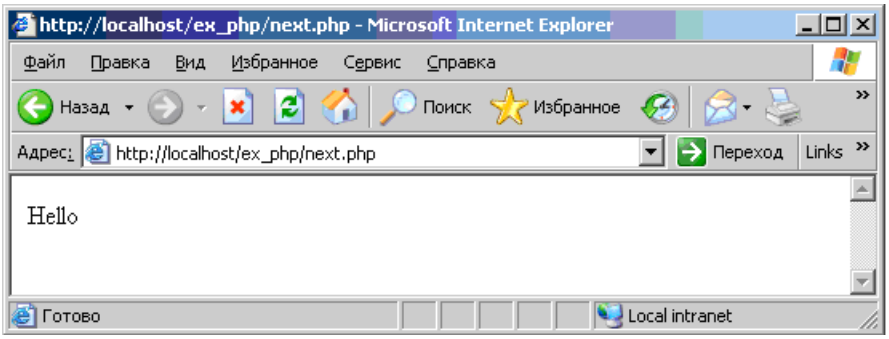

Рис. 4. **Передача данных методом POST**

Как видно из рис. 4, в адресной строке уже нет никаких параметров, но, тем не менее, строка Hello выводится на странице.

Данные, передаваемые методом POST, также ограничены в объеме (как правило, 8 Мбайт).

Типы данных

PHP позволяет не заботиться явно об определении типа переменной. С одной и той же переменной в рамках одной программы можно работать и как со строкой, и как с числом. Однако в PHP существует набор основных типов данных, которые могут явно указываться при работе с переменными:
- integer;
- string;
- float(double, real);
- array;

- boolean;
- object.

Для того чтобы определить тип, который PHP назначил переменной, применяется функция gettype(). Ее единственным параметром является имя переменной.

Операция присваивания

Использование оператора присваивания «=» означает, что значение операнда выражения из правой части нужно присвоить операнду левой части (листинг 1.13).

Листинг 1.13. Оператор присваивания

```
<?php
$var=5;
echo $var; // выведет '5'
?>
```

Операции над числами

Выполнение операций над числами осуществляется с помощью арифметических операторов, которые перечислены в табл. 1.

Таблица 1. **Арифметические операции**

Пример	Название	Результат
$a + $b	Сложение	Сумма $a и $b
$a - $b	Вычитание	Разность $a и $b
$a * $b	Умножение	Произведение $a и $b
$a / $b	Деление	Частное от деления $a на $b
$a % $b	Modulus	Целочисленный остаток от деления $a на $b

Кроме основных операторов, можно использовать сокращенную запись арифметических операторов, используя вместо них операторы присваивания «+=», «-=», «*=». Например, выражение $a=$a+$b в сокращенной форме запишется как $a+=$b и будет означать, что операнду левой части необходимо присвоить сумму значений левого и правого операндов. Аналогичные действия допустимы для всех приведенных ранее операторов.

В зависимости от количества участвующих операндов операции подразделяют на унарные и бинарные. Унарная операция работает с одним операндом, бинарная – с двумя. Все арифметические операции, кроме операций инкремента и декремента, являются бинарными.

Операции инкремента (++) и декремента (--) подразделяют на префиксные и постфиксные. При префиксной операции увеличение или уменьшение операнда на единицу происходит до того, как возвращается значение, а при постфиксной – после. Таким образом, операторы инкремента и декремента можно записывать в двух формах: префиксной и постфиксной (листинг 1.14).

Листинг 1.14. Операция инкремента

```
<?php
$i=1;
// префиксная форма инкремента
echo ++$i; // 2
$i=1;
// постфиксная форма инкремента
echo $i++; // 1
?>
```

В первом случае на выходе получаем значение 2, так как префиксный оператор инкремента сначала выполняет инкрементирование, а затем возвращает полученное значение. Во втором случае в качестве результата получаем 1.

Операции инкремента и декремента могут быть применены не только к переменным типа integer, но и к другим типам переменных.

Листинг 1.15. Применение операции инкремента к переменной типа string

```
<?php
$var="aaa"; // переменная типа string
echo ++$var; // aab
?>
```

Операции сравнения и логические операции

Данные операции сравнивают два значения (таблицы 2 и 3).

Таблица 2. **Операции сравнения**

Пример	Название	Результат
$a == $b	Равно	TRUE, если $a равно $b
$a === $b	Эквивалентно	TRUE, если $a равно $b и они одного типа
$a != $b	Не равно	TRUE, если $a не равно $b
$a <> $b	Не равно	TRUE, если $a не равно $b
$a !== $b	Не эквивалентно	TRUE, если $a не равно $b или они разных типов
$a < $b	Меньше	TRUE, если $a строго меньше $b
$a > $b	Больше	TRUE, если $a строго больше $b
$a <= $b	Меньше или равно	TRUE, если $a меньше или равно $b
$a >= $b	Больше или равно	TRUE, если $a больше или равно $b

Таблица 3. **Логические операции**

Пример	Имя	Результат
$a and $b	and	TRUE, если и $a, и $b TRUE
$a or $b	or	TRUE, если $a или $b TRUE
$a xor $b	xor	TRUE, если $a или $b TRUE, но не оба
! $a	not	TRUE, если $a не TRUE
$a && $b	and	TRUE, если и $a, и $b TRUE
$a || $b	or	TRUE, если $a или $b TRUE

Смысл двух вариантов and и or состоит в том, что они работают с различными приоритетами.

Строковые операции

Имеются две строковые операции (листинг 1.16):
- операция конкатенации «.», которая возвращает объединение из правого и левого аргументов;
- операция присвоения «.=», которая присоединяет правый аргумент к левому.

Листинг 1.16. Строковые операции

```
<?php
$a="Hello";
$b=$a."World!";
echo $b; // теперь $b содержит "Hello World!"
$a="Hello";
$a.="World!";
echo $a; // теперь $a содержит "Hello World!"
?>
```

Приоритет выполнения операций

Приоритет операции специфицирует, какая операция выполняется первой. Например, в выражении 1+5*3 результат будет 16, а не 18, поскольку умножение «*» имеет более высокий приоритет, чем сложение «+». Скобки можно использовать для переопределения приоритетов выполнения, если это необходимо. Например, (1+5)*3 вычисляется в 18.

В табл. 4 дан список приоритетов операций от низшего к высшему.

Таблица 4. **Приоритет выполнения операций**

Ассоциативность	Операции
Левая	,
Левая	or
Левая	xor
Левая	and
Правая	print
Левая	= += -= *= /= .= %= &= |= ^= ~= <<= >>=
Левая	? :
Левая	||
Левая	&&
Левая	|
Левая	^
Левая	&
Не ассоциативная	== != === !==
Не ассоциативная	< <= > >=
Левая	<< >>
Левая	+ - .
Левая	* / %
Правая	! ~ ++ -- (int) (float) (string) (array) (object) @
Правая	[
Не ассоциативная	new

Лабораторная работа 1
Основы PHP

1. Создайте свою папку на учебном сервере. Для этого в папке *Сетевое окружение/Вся сеть/Uch/Uchserv/isit* создайте папку с именем своей фамилии, записанной *английскими буквами* (можно сокращенный вариант).

2. Загрузите редактор языка PHP Expert Editor 3.2.1 (далее – редактор). Окно редактора показано на рис. 5. Документ редактора содержит минимальный набор тегов PHP и HTML (не обязательных для PHP-файла).

3. Наберите код сценария из листинга 1.17. Обратите внимание, что редактор подсвечивает разными цветами теги HTML, PHP, операторы и текст.

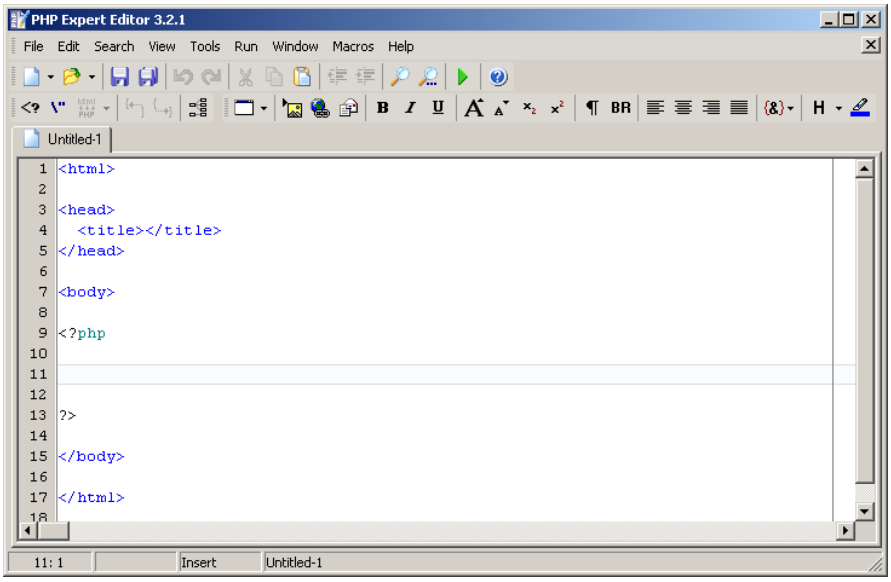

Рис. 5. **Окно редактора PHP Expert Editor**

Листинг 1.17. Файл index.php

```
<html>
<head><title>Первый PHP-сценарий</title></head>
<body>
<?php
echo "Добро пожаловать, пользователь!";
?>
</body>
</html>
```

4. Сохраните файл *index.php* в своей папке на сервере (команда *File/Save* или кнопка *Сохранить*). Путь: *Сетевое окружение/Вся сеть/Uch/Uchserv/isit/вашая папка*.

5. Просмотрите в браузере созданный сценарий. Все PHP-файлы находятся на сервере, поэтому запускать их нужно *только из браузера*:

- Запустите браузер.
- В строке *Адрес* укажите адрес файла *uch.btu/isit/ваша папка/имя файла.* Так как файл из примера называется *index.php*, то имя можно не указывать. Для файлов с другими именами указание имени обязательно.
- Нажмите *Enter*. Браузер выведет результат, показанный на рис. 6.

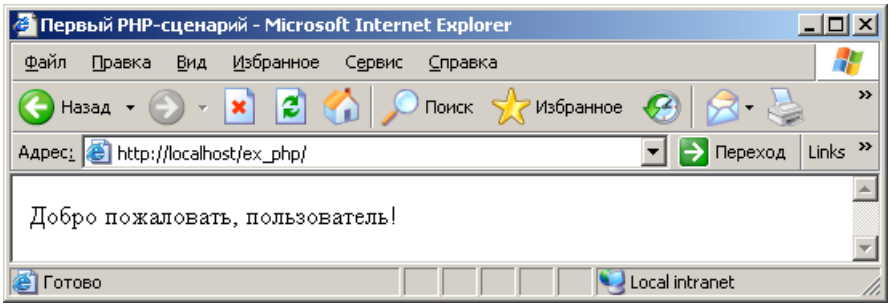

Рис. 6. **Результат работы файла *index.php***

- Просмотрите полученную страницу в режиме HTML (команда *Вид/Просмотр HTML-кода*). Получим результат, показанный на рис. 7.

Как видно из примера, браузер отображает только HTML-код (см. раздел «Как работает PHP-сценарий»). PHP-код может не содержать теги HTML.

- Удалите в файле *index.php* все теги HTML. Сохраните файл с именем *Primer1.php*. Просмотрите полученную страницу в браузере, а затем в режиме HTML. Сравните результаты.

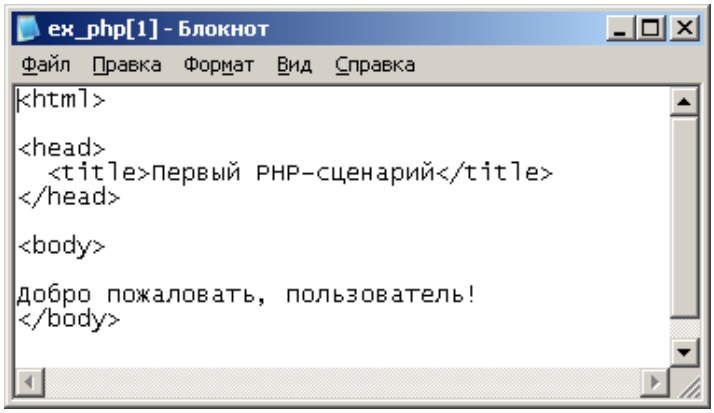

Рис. 7. **Результат работы файла *index.php* в кодировке HTML**

Лабораторная работа 2
Работа с простыми операторами

1. Выполните задания, приведенные в листингах 1.2, 1.3, 1.7–1.14. Сохраните файлы с именами *Primer_№листинга.php*.

Результат работы скрипта из листинга 1.2 представлен на рис. 8.

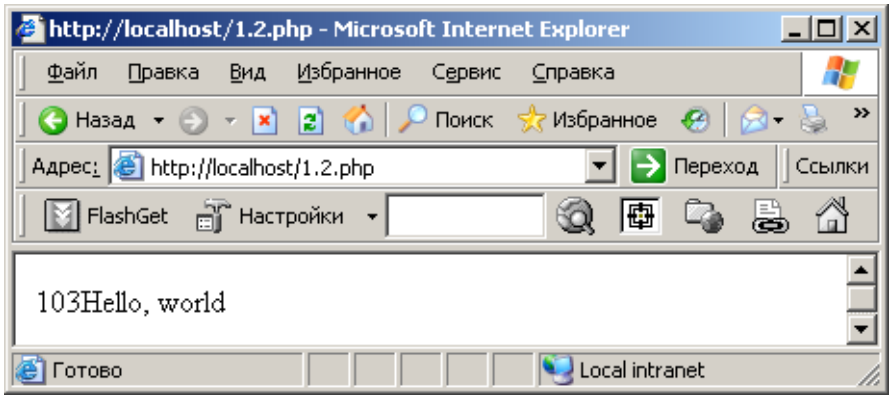

Рис. 8. **Вывод данных на экран в одной строке**

Такой результат является неудовлетворительным, так как в браузере все три строки слились в одну. Чтобы их разделить, можно воспользоваться тегом HTML
, предназначенным для перевода строки (листинг 1.18). Преобразуем листинг 1.2 следующим образом:

***Листинг 1.18. Простые операторы и тег
***

```
<?php
echo 5+5;
echo "<br>";
echo 5-2;
echo "<br>";
echo "Hello, world";
?>
```

Результат работы скрипта из листинга 1.18 представлен на рис. 9.

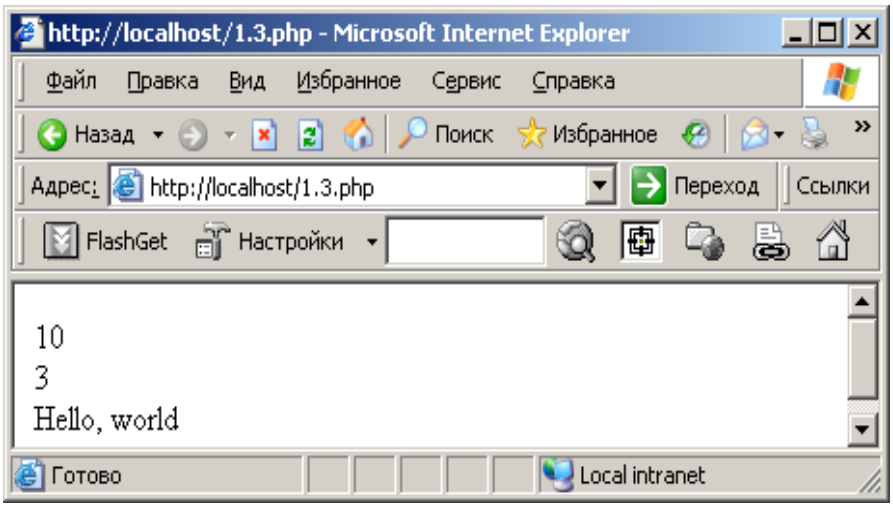

Рис. 9. **Вывод данных на экран в трех строках**

Такой результат вывода данных более удобен.

2. Самостоятельно составьте и выполните скрипт для:

• выведения фразы «Мне … лет», где количество лет вычисляется на основании текущего года и года рождения.

• выведения фразы «Я учусь в БТЭУ … года», где количество лет вычисляется на основании текущего года и года поступления.

3. Результаты работы продемонстрируйте преподавателю.

2. ОПЕРАТОРЫ ВЫБОРА

К операторам выбора относят: условный оператор if … else, переключатель switch и условную операцию.

Условный оператор if

Синтаксис условного оператора:

if(condition) operator1 [else operator2].

В качестве аргумента condition оператор if принимает логическую переменную или выражение, возвращающее логическую переменную. Если оно истинно, то выполняется operator1. В противном случае выполняется operator2.

Если $a больше $b, то код в листинге 2.1 выведет «a больше чем b», в противном случае – «a меньше чем b».

Листинг 2.1. Файл index.php

```
<?php
$a=5;
$b=7;
if($a>$b)
echo "a больше чем b";
else
echo "a меньше чем b";
?>
```

Часто по условию необходимо выполнить не один, а несколько операторов. Для этого можно сгруппировать несколько операторов в блок. Например, код в листинге 2.2 сравнит $a и $b, переменной с большим значением присвоит значение переменной с меньшим значением и выведет значения обеих переменных.

Листинг 2.2. Условный оператор if

```
<?php
$a=5;
$b=7;
if($a>$b)
{
$a=$b;
echo "a равно".$a;
echo "b равно".$b;
}
else
{$b=$a;
echo "a равно".$a;
echo "b равно".$b;
}
?>
```

Операторы if могут вкладываться друг в друга, что дает полную свободу при условном выполнении различных частей программы.

Оператор switch

Оператор switch предоставляет более удобные средства для организации множественного выбора, чем оператор if … else.

Синтаксис оператора:

```
switch(expression) // переключающее выражение
{
  case value1: // константное выражение 1
    statements; // блок операторов
    [break;]
  case value 2: // константное выражение 2
    statements;
    [break;]
  [default:
    statements;]
}.
```

Управляющая структура switch передает управление тому из операторов case, для которого значение константного выражения совпадает со значением переключающего. Сначала анализируется переключающее выражение expression и осуществляется переход к той ветви программы, для которой его значение совпадает с константным. Далее следует выполнение оператора или группы операторов до тех пор, пока не встретится ключевое слово break, которым обозначается выход из переключателя. Если значение переключающего выражения не совпадет ни с одним из константных выражений, то выполнится переход к оператору, помеченному меткой default. В каждом переключателе может быть не более одной метки default.

Рассмотрим пример программы с переключателем. В листинге 2.3 выводится сообщение о курсе обучения студента.

Листинг 2.3. Переключатель switch

```
<?php
$a=2; // номер курса
switch($a)
{case 1:
echo "Вы учитесь на первом курсе";
break;
case 2:
echo "Вы учитесь на втором курсе";
break;
case 3:
echo "Вы учитесь на третьем курсе";
break;
case 4:
echo "Вы учитесь на четвертом курсе";
break;
case 5:
echo "Вы учитесь на пятом курсе";
break;
default:
echo "В нашем вузе обучение пять лет";}
?>
```

Лабораторная работа 3
Работа с операторами выбора

1. Выполните задания, приведенные в листингах 2.1–2.3.
2. Самостоятельно составьте и выполните скрипт для:
- определения названия специальности по буквенному шифру группы для экономического факультета;
- определения по дате рождения, достиг ли человек совершеннолетия.
3. Результаты работы продемонстрируйте преподавателю.

3. СТРОКОВЫЕ ФУНКЦИИ

Кавычки

В языках программирования обычно поддерживаются два варианта кавычек: одинарные и двойные. Одинарные кавычки используют для обрамления двойных (листинг 3.1), а двойные – для обрамления одинарных кавычек (листинг 3.2).

Листинг 3.1. Двойные кавычки

```
<?php
echo "Переменная принимает значение '345' ";
?>
```

Листинг 3.2. Одинарные кавычки

```
<?php
echo 'Проект "Бездна" – самый дорогой проект в истории';
?>
```

В PHP функциональность кавычек была расширена. Если поместить в двойные кавычки переменную, ее значение будет подставлено в тексте, а если в одинарные, будет выведено ее имя.

Листинг 3.3. Интерпретация переменных

```
<?php
$str=123
echo "Значение переменной – $str"; // Значение переменной – 123
echo 'Значение переменной – $str'; // Значение переменной – $str
?>
```

Иногда требуется экранирование знака двойных кавычек или знака $. Если строка заключена в двойные кавычки, PHP распознает большее количество управляющих последовательностей для специальных символов, представленных в табл. 5.

Таблица 5. **Специальные символы и их значения**

Символ	Значение
\n	Новая строка
\r	Возврат каретки
\t	Горизонтальная табуляция
\\	Обратная косая черта
\$	Знак доллара
\"	Двойная кавычка

Листинг 3.4. Экранирование символов

```
<?php
$str=123;
// экранирование знака $
echo "Значение переменной – \$str"; // Значение переменной – $str
// экранирование знака "
echo "Проект \"Бездна\" самый дорогой проект в истории";
// Проект "Бездна" – самый дорогой проект в истории
?>
```

Функция strpos

Функция strpos() возвращает позицию вхождения подстроки в строку. Синтаксис:

strpos(string str, string substr [,int offset]).

Эта функция возвращает позицию в строке str, с которой начинается переданная ей подстрока substr (листинг 3.5). Отсчет позиций начинается с нуля, т. е. первому символу строки соответствует позиция 0, второму – 1 и т. д.

Листинг 3.5. Функция strpos()

```
<?php
$var=strpos("Hello, world!", "world");
echo $var; // выводит число 7
?>
```

Необязательный параметр offset позволяет указать в строке позицию, с которой нужно начать поиск.

Функция strlen

Функция strlen() принимает в качестве аргумента строку и возвращает ее длину (листинг 3.6). Синтаксис:

strlen(string str).

Листинг 3.6. Функция strlen()

```
<?php
$var=strlen("Hello, world!!!");
echo $var; // выводит число 15
echo strlen("Hello, PHP!"); // 15
?>
```

Функция strlen() чувствительна к регистру.

Функция strstr

Функция strstr() предназначена для получения части строки (листинг 3.7). Синтаксис:

strstr(string str, string substr).

Функция возвращает часть строки str, начиная с первого вхождения подстроки substr до конца строки str. Если подстрока substr не найдена в строке str, возвращается FALSE.

Листинг 3.7. Функция strstr()

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

Функция substr

Функция strstr() возвращает часть строки. Синтаксис:

substr(string str, int start [, int length]).

Первый аргумент str – исходная строка, из которой вырезается текст. Второй аргумент start – положение в строке, которую нужно вернуть, с первого символа (отсчет начинается с нуля). Третий аргумент length – длина возвращаемой строки в символах. Если третий аргумент не указан, то возвращается вся оставшаяся длина (часть) строки.

Листинг 3.8. Функция substr()

```
<?php
$str="03.02.2007";
echo "день - ".substr($str, 0,2)."<br>"; // день -03
echo "месяц - ".substr($str, 3,2)."<br>"; // месяц -02
echo "год - ".substr($str, 6)."<br>"; // год -2007
?>
```

Функция str_replace

Функция str_replace() позволяет заменить одну подстроку в тексте на другую (листинг 3.9). Синтаксис:

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

Она заменяет в исходной строке str все вхождения строки from на to и возвращает результат.

Листинг 3.9. Функция str_replace()

```
<?php
// исходная строка
$text="<b>Это</b> очень жирный <b>текст</b>";
echo $text."<br>";
$text=str_replace("b", "i", $text);
echo $text."<br>";
$text=str_replace("очень жирный", "наклонный", $text);
echo $text;
?>
```

Специальным видом замены является удаление подстроки из строки.

Листинг 3.10. Функция str_replace()

```
<?php
// исходная строка
$text="<b>Это</b> очень жирный <b>текст</b>";
echo $text."<br>";
$text=str_replace("<b>", "", $text);
$text=str_replace("</b>", "", $text);
echo $text;
?>
```

Лабораторная работа 4
Работа со строковыми функциями

1. Выполните задания, приведенные в листингах 3.1–3.10.
2. Самостоятельно составьте и выполните скрипт для:
- определения по адресу электронной почты адреса почтового сервера;
- определения длины строки «Вас приветствует БТЭУ»;
- определения номера этажа по номеру аудитории;
- замены в строке «Вас приветствует БТЭУ» слова «БТЭУ» на «Белорусский торгово-экономический университет».

3. Результаты работы продемонстрируйте преподавателю.

4. ОПЕРАТОРЫ ЦИКЛА

Операторы цикла задают многократное исполнение операторов в теле цикла. В PHP определены четыре разных оператора цикла:
- while (цикл с предусловием);
- do … while (цикл с постусловием);
- for (итерационный цикл);
- итерационный цикл foreach.

Три первых оператора берут свое начало от языка C, а последний – от Perl, и предназначен для ассоциативных массивов.

Цикл while

Оператор while называется оператором цикла с предусловием. Синтаксис:

```
while(expr)
{
statement
}.
```

При входе в цикл вычисляется выражение-условие expr, и если его значение истинно (TRUE), выполняется тело цикла. Затем вычисление выражения-условия и операторов тела цикла выполняется до тех пор, пока значение выражения-условия не станет ложным (FALSE).

Тело цикла не обязательно должно быть заключено в фигурные скобки, если требуется выполнить только один оператор, то они могут быть опущены. Далее представлены два примера (листинги 4.1 и 4.2), они идентичны и печатают числа от 1 до 10.

Листинг 4.1. Оператор while

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

Листинг 4.2. Оператор while

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

Цикл do … while

Циклы do … while очень похожи на циклы while, но условное выражение проверяется в конце каждой итерации, а не в начале (листинг 4.3). Синтаксис:

```
do
{
операторы;
}
while(expr).
```

Такой цикл всегда будет выполнен хотя бы один раз. После выполнения тела цикла вычисляется выражение-условие expr, и, если оно истинно (TRUE), вновь выполняется тело цикла.

Листинг 4.3. Оператор do … while

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

В этом случае оператор echo $i выполнится хотя бы один раз.

Цикл for

Синтаксис цикла for:

```
for(expr1; expr2; expr3).
{
statement;
}.
```

Оператор expr1 (инициализация цикла) – это последовательность определений и выражений, разделяемая запятыми. Все выражения, входящие в инициализацию, вычисляются только один раз при входе в цикл. Как правило, здесь устанавливаются начальные значения счетчиков и параметров цикла. Смысл выражения-условия expr2 такой же, как и у циклов с пред- и постусловиями: цикл выполняется до тех пор, пока выражение expr2 истинно, и прекращает работу, когда оно ложно. При отсутствии выражения условия предполагается, что его значение всегда истинно. Выражение expr3 вычисляется в конце каждой итерации после выполнения тела цикла.

Листинг 4.4. Оператор for

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

Листинг 4.5. Оператор for

```
<?php
for($i=5; $i; $i--)
{
   print $i;
}
?>
```

Переменная $i инициируется значением 5, на каждой итерации цикла это значение уменьшается на единицу с помощью операции декремента ($i--). В качестве условия выступает сама переменная $i, как только ее значение достигает 0, цикл прекращает свою работу.

Рассмотренные варианты использования цикла for являются традиционными, но не единственными. Цикл for допускает отсутствие тела цикла, так как все вычисления могут быть выполнены в выражении expr3 (листинг 4.6).

Листинг 4.6. Отсутствие тела цикла в операторе for

```
<?php
for($i=1; $i<=10; print $i, $i++);
?>
```

Здесь вместо оператора $i++ используется последовательность операторов print $i, $i++, разделенных запятой. Такая форма записи допускается синтаксисом PHP, но не приветствуется, так как сильно уменьшает читабельность кода.

Циклы используют увеличение или уменьшение счетчиков не только на единицу, но и допускают использование в качестве шага и другие значения.

Примечание. Использование конструкции echo в цикле for не допускается.

Листинг 4.7. Использование шага, равного 5

```
<?php
for($i=0; $i<=100; $i+=5)
echo $i."<br>";
?>
```

5. РАБОТА С ФАЙЛАМИ
Включение файла в документ

Инструкция include() позволяет включить файл в PHP-скрипт (листинг 5.1). Функция принимает единственный аргумент – путь к включаемому файлу. Результатом ее действия является подстановка содержимого файла в место вызова исходного скрипта. Если в качестве включаемого файла выступает PHP-скрипт, то сначала происходит его подстановка в исходный скрипт, а затем интерпретация результирующего скрипта.

Листинг 5.1. Функция include()

```
<?php
echo "первый<br>";
include("second.php");
echo "третий <br>";
?>
```

Пусть файл *second.php* содержит код, представленный в листинге 5.2.

Листинг 5.2. Файл second.php

```
<?php
echo "второй<br>";
?>
<h3>текст необязательно должен выводиться оператором echo</h3>
```

Результат работы скриптов представлен на рис. 10.

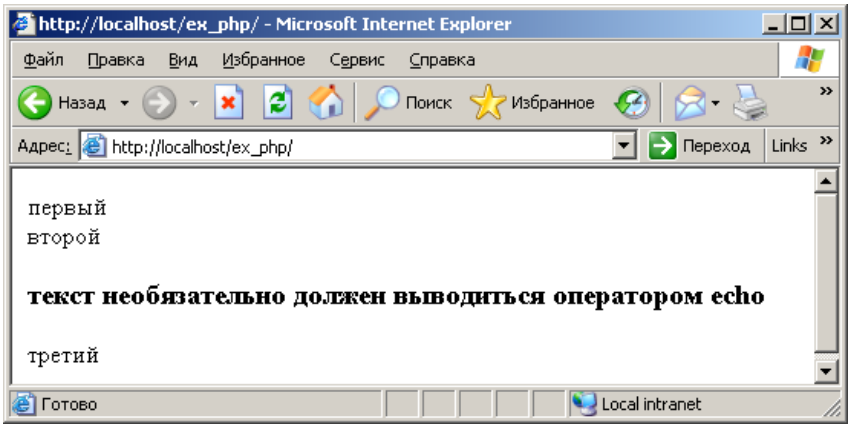

Рис. 10. **Результат работы скрипта с инструкцией include()**

Открытие файла

Любая операция с файлами (создание нового файла, чтение, запись в существующий файл) предваряется операцией открытия файла[2]. Открытие файлов выполняется с помощью функции fopen(). Синтаксис:

fopen(string filename, string mode [, int use_include_path]).

Первый аргумент filename – относительный или абсолютный путь к файлу. Второй аргумент mode задает режим работы с открываемым файлом, может принимать следующие значения:

- r – открыть файл только для чтения, поместить указатель в начало файла;
- r+ – открыть файл для чтения и записи, поместить указатель в начало файла;
- w – создать новый пустой файл только для записи, если файл с таким именем уже есть, то вся информация в нем уничтожается;
- w+ – создать новый пустой файл для чтения и записи, если файл с таким именем уже есть, то вся информация в нем уничтожается;
- a – открыть файл для дозаписи, поместить указатель в конец файла, если файл не существует, делается попытка создать его;
- a+ – открыть файл для дозаписи и чтения, поместить указатель в конец файла, если файл не существует, делается попытка создать его.

Можно использовать необязательный третий параметр и установить в него значение 1, если нужно найти файл также и в каталоге use_include_path.

В случае удачного открытия файла функция fopen() возвращает дескриптор файла, в случае неудачи – FALSE.

Дескриптор файла представляет собой указатель на открытый файл, который используется операционной системой для поддержки операций с этим файлом и представляет собой уникальное число.

Возвращенный функцией дескриптор файла необходимо затем указывать во всех функциях, которые в дальнейшем будут работать с этим файлом.

Листинг 5.3. Создание файла

```
<?php
$fd=fopen("file.txt", "w");
?>
```

После выполнения скрипта в директории, где он расположен, должен появиться файл с именем *file.txt*.

Закрытие файла

После того как работа с файлом закончена, его необходимо закрыть. Закрытие файлов выполняется с помощью функции fclose() (листинг 5.4). Синтаксис:

fclose(int fd).

Аргумент fd представляет собой дескриптор, который необходимо закрыть.

[2] Существует ряд операторов, составляющих исключение.

Листинг 5.4. Функция fclose()

```
<?php
$fd=fopen("file.txt", "w");
fclose($fd);
?>
```

Запись в файл

Запись в файл осуществляется с помощью функции fwrite() (листинг 5.5). Синтаксис:

fwrite(int fd, string str [, int length]).

Первый аргумент fd представляет собой дескриптор файла, который возвращается функцией fopen(). Второй аргумент str представляет собой строку, которая должна быть записана в файл. Третий необязательный аргумент задает количество символов в строке, которые должны быть записаны, если аргумент не указан, записывается вся строка.

Листинг 5.5. Запись в файл

```
<?php
// открываем файл для записи
$fd=fopen("file.txt", "w");
// записываем в файл
fwrite($fd, "Hello, world!");
// закрываем файл
fclose($fd);
?>
```

Чтение из файла

Web-приложения на PHP, работающие с различными данными, для хранения информации используются реляционные базы данных и локальную файловую систему. Сами файлы полезны для хранения простых данных, например, информации о настройках или для различных неструктурированных данных.

Чтение содержимого открытого файла осуществляется с помощью функции fread() (листинг 5.6). Синтаксис:

fread(int fd, int length).

Эта функция возвращает строку длиной length байт. Для чтения файла целиком часто прибегают к функции filesize(), возвращающей число байт, содержащееся в файле, имя которого передано данной функции в качестве аргумента.

Листинг 5.6. Чтение из файла

```
<?php
// имя файла
$filename="file.txt";
// открываем файл для чтения
$fd=fopen($filename, "r");
// читаем содержимое файла в переменную $bufer
$bufer=fread($fd, filesize($filename));
// закрываем файл
fclose($fd);
// выводим содержимое файла в окно браузера
echo $bufer;
?>
```

Для чтения из файла можно также пользоваться функцией fgets(). Синтаксис:

fgets(int file, int length).

Функция читает и возвращает строку длины length. Чтение прекращается, когда достигнута новая строка или конец файла. При достижении конца файла функция возвращает пустую строку.

Листинг 5.7. Построчное чтение файла

```
<?php
$fd=fopen("inputfile.txt", "r");
while(!feof($fd))
{
$buffer=fgets($fd, 4096);
echo $buffer;
}
fclose($fd);
?>
```

Функция feof() проверяет, достигнут ли конец файла. Синтаксис:

```
feof(int fp).
```

Если указатель файла находится в конце файла, функция возвращает TRUE, иначе – FALSE.
Для чтения из файла с удалением из него тегов HTML применяется функция fgetss(). Синтаксис:

```
fgetss(int file, int length [,string allowable_tags]).
```

Необязательный параметр allowable_tags может содержать строку со списком тегов, которые не должны вырезаться. Теги записываются через запятую.

Если необходимо записать содержимое файла в массив, применяется функция file(). Синтаксис:

```
file(int file [, int use_include_path]).
```

Функция считывает файл с именем filename и возвращает массив, каждый элемент которого соответствует строке в прочитанном файле.
Эта функция удобна тем, что с ее помощью можно легко подсчитать количество строк в файле (листинг 5.8).

Листинг 5.8. Подсчет строк в файле

```
<?php
$c=file("file.txt");
echo count($c); // 5
?>
```

Функция count() считает элементы в переменной. Синтаксис:

```
count(mixed var).
```

Эта функция возвращает количество элементов в var, типом которой обычно является array, поскольку все остальные переменные состоят из одного элемента.

Листинг 5.9. Чтение файла, информация из которого выводится в браузер

```
<?php
$c=file("file.txt");
for($i=0; $i<count($c); $i++)
echo $c[$i]."<br>";
?>
```

Содержимое файла *file.txt*	Результат
11111 2222 333 44 5	11111 2222 333 44 5

Для чтения файлов с расширением *.*csv* применяется функция fgetcsv. Синтаксис:

fgetcsv(int file, int length , string delim [, string enclosure]).

Функция читает строку из файла и разбивает ее по символу delim. Строка delim должна состоять из одного символа. Функция возвращает получившийся массив или FALSE, если достигнут конец файла. Пустые строки в файле не игнорируются, а возвращаются как массив из одного элемента – пустой строки. Параметр length задает максимальную длину строки точно так же, как это делается в функции fgets().

Примечание. CSV-файл – это текстовый файл, в котором данные построены следующим образом: строки файла представляют собой строки таблицы, а столбцы таблицы разделены в файле заранее определенным символом-разделителем, например «;».

Листинг 5.10. Чтение и печать всего содержимого CSV-файла

```
<?php
$row=1;
$fp=fopen("test.csv", "r");
while($data=fgetcsv($fp, 1000, ","))
{
$num=count($data);
print "$num элемента в строке $row: <br>";
$row++;
for($c=0; $c<$num; $c++)
print $data[$c]."<br>";
}
fclose($fp);
?>
```

Содержимое файла *test.csv*	Результат
1,1,1,1 2,2,2 3,3 4	4 элемента в строке 1: 1 1 1 1 3 элемента в строке 2: 2 2 2 2 элемента в строке 3: 3 3 1 элемента в строке 4: 4

Примечание. Массивы являются одной из основных и часто встречающихся структур для хранения данных. По определению массив представляет собой индексированную совокупность переменных одного типа. Каждая переменная или элемент массива имеет *индекс*, т. е. все элементы массива последовательно пронумерованы от 0 до N, где N – размер массива. Имена массивов так же, как и переменных, начинаются с символа $.

Для того чтобы обратиться к отдельному элементу массива, необходимо поместить после его имени квадратные скобки, в которых указать индекс элемента массива. Например, обращение к $str[0] запрашивает первый элемент массива $str.

В качестве элементов массива могут выступать другие массивы, в этом случае говорят о многомерных массивах.

Лабораторная работа 5
Работа с файлами

1. Выполните задания, приведенные в листингах 5.1, 5.2, 5.5–5.10.
2. Самостоятельно составьте и выполните скрипт для:
- создания Web-страницы, с помощью функции include() выведите ее содержимое в окно браузера;
- создания текстового файла с помощью функций работы с файлами, запишите в него номер своего телефона и определите по номеру название оператора связи.

3. Результаты работы продемонстрируйте преподавателю.

6. ЭЛЕКТРОННАЯ ПОЧТА

Преобразование кодировок. Функция convert_cyr_string

По историческим причинам в русскоязычном секторе Интернета используется большое число кодировок. Для преобразования строк из одной кодировки в другую предназначена функция convert_cyr_string(). Синтаксис:

convert_cyr_string(string str, string from, string to).

Функция convert_cyr_string() преобразует строку str из кодировки from в кодировку to (листинг 6.1). Аргументы from и to – это одиночные символы, определяющие кодировку:

- k – koi8-r;
- w – windows-1251;
- i – iso8859-5;
- a – x-cp866;
- d – x-cp866;
- m – x-mac-cyrillic.

Листинг 6.1. Функция convert_cyr_string()

```
<?php
$str="Hello, world!";
echo " '$str' в koi8-r является ' ".convert_cyr_string($str, "w", "k")." '<br>";
?>
```

Функция date

Часто в письмо требуется вставить дату или время отправки сообщения. Функция date() позволяет форматировать дату (листинг 6.2). Синтаксис:

date(string format [, int timestamp]).

Параметр format определяет строку форматирования, в которой символы интерпретируются в соответствии с табл. 6. Необязательный параметр timestamp позволяет задать время в секундах.

Таблица 6. **Некоторые символы форматирования функции date()**

d	День (число) месяца – 2 цифры с ведущим нулем, если необходимо (от "01" до "31")
D	День недели буквенный – 3 буквы (например, "Fri")
F	Месяц буквенный – long (например, "January")
g	Час – 12-часовой формат без ведущих нулей (от "1" до "12")
G	Час – 24-часовой формат без ведущих нулей (от "0" до "23")
i	Минуты (от "00" до "59")
m	Месяц (от "01" до "12")
M	Месяц буквенный – 3 буквы (например, "Jan")
s	Секунды (от "00" до "59")
Y	Год – 4 цифры (например, "1999")

Листинг 6.2. Функция date()

```
<?php
echo date("d.m.Y G:i:s"); // 04.02.2007 15:47:20
?>
```

Функция mail

Отправка почты средствами PHP осуществляется с помощью функции mail(). Синтаксис:

mail(string to, string subject, string message [, string headers [, string parameters]]).

Аргументы функции:

- to – адрес электронной почты;
- subject – тема сообщения;

• message – текст сообщения;
• headers – дополнительные заголовки, которые можно задать в сообщении;
• parameters – дополнительные параметры, которые можно задать в сообщении.

Очень часто требуется изменить формат письма с обычного текста (text/plain) на HTML-формат (text/html) или указать кодировку сообщения. Установка формата письма и кодировки осуществляется с помощью почтовых заголовков Content-Type и charset соответственно:

```
Content-Type: text/html; charset=KOI8-R\r\n.
```

Для отправки почтового сообщения в кодировке cp1251 вместо KOI8-R следует прописать Windows-1251 (листинг 6.3):

Листинг 6.3. Отправка почтового сообщения с помощью функции mail()

```
<?php
$to=316m1@uch.btu;
$subject="Отчет";
$subject=convert_cyr_string($subject, "w", "k");
$message="<html>
    <head></head>
    <body>
    Письмо отправлено – ".date("d.m.Y ")."<br>
    Отправитель 316m2@uch.btu
    </body>
    </html>";
$message=convert_cyr_string($message, "w", "k");
$headers="Content-Type: text/html; charset=KOI8-R\r\n";
mail($to, $subject, $message, $headers);
?>
```

При получении письма в качестве отправителя будет подставлен адрес сервера. Для того чтобы этого избежать, в качестве обратного адреса можно назначить произвольный адрес с помощью почтового заголовка From:

```
From: name<e-mail>.
```

В качестве name указывается имя, которое будет отображаться клиентским браузером как имя отправителя, а e-mail содержит обратный почтовый адрес. Строки формирования переменной $headers могут выглядеть следующим образом (листинг 6.4):

Листинг 6.4. Формирование почтовых заголовков

```
<?php
$headers="Content-Type: text/html; charset=KOI8-R\r\n";
$headers.="From: server <320m1@uch.btu>\r\n";
?>
```

При отправке электронного письма, снабженного почтовыми заголовками из листинга 6.4, оно будет представлено как письмо от пользователя server с электронным адресом 320m1@uch.btu.

Лабораторная работа 6
Электронная почта

1. Выполните задания, приведенные в листингах 6.1–6.3. В листинге 6.3 в адресе получателя и отправителя письма укажите свою аудиторию.

2. Самостоятельно составьте и выполните скрипт для:

• определения с помощью функции data() текущего времени, в зависимости от результата выведите в окно браузера одно из сообщений: «Доброе утро», «Добрый день», «Добрый вечер», «Доброй ночи»;

• создания текстового файла с помощью функций работы с файлами, запишите в него текст приглашения на семинар (тему семинара и имя приглашенного введите как значения переменных), разошлите приглашения.

3. Результаты работы продемонстрируйте преподавателю.

7. ПРАКТИКА
Лабораторная работа 7
Анкета студента

Требуется разработать Web-сайт, который должен выполнять следующие функции:

1. Сбор информации о студенте:
- фамилия, имя, отчество;
- дата рождения;
- пол;
- факультет, на котором обучается студент;
- номер группы;
- адрес электронной почты.

После ввода данных необходимо подтверждение правильности ввода.

2. Проведение проверки полноты внесенной информации по следующим критериям:
- все поля должны быть заполнены;
- адрес электронной почты должен принадлежать учебному серверу (например, stud@ush.btu).

По результатам проверки должно выдаваться сообщение, подтверждающее правильность ввода информации и предлагающее перейти к просмотру введенной информации либо об ошибках ввода и предложением вернуться к анкете.

3. Запись данных о студенте в файл с расширением **.csv*.

4. Вывод данных по всем студентам из файла в браузер в виде таблицы.

Логическая схема Web-сайта показана на рис. 11.

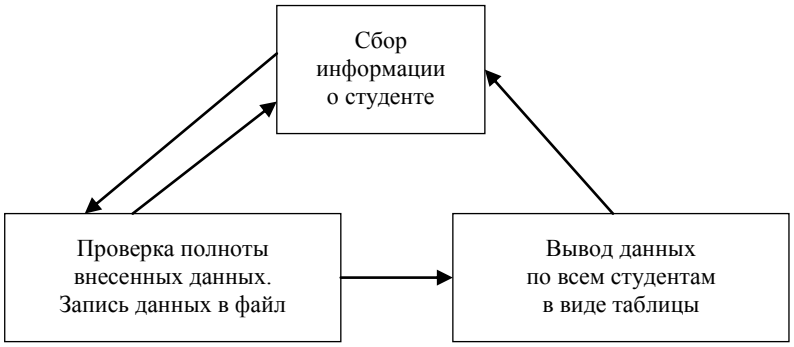

Рис. 11. **Логическая схема Web-сайта**

Из представленной схемы видно, что сайт состоит из трех файлов:

Первый файл: HTML-форма, содержащая необходимые поля для ввода информации о студенте и передающая данные полей PHP-скрипту.

Второй файл: PHP-скрипт, проверяющий правильность и полноту записей в форме и осуществляющий запись данных в файл.

Третий файл: PHP-скрипт, осуществляющий вывод данных на экран в виде таблицы.

Указания по выполнению

1. Спроектируйте HTML-форму (файл *anketa.htm*), проще всего в любом HTML-редакторе (например, Front Page). В качестве обработчика формы установите PHP-скрипт, который будет проверять правильность и полноту заполнения формы.

Код первого файла представлен в листинге 7.1.

Листинг 7.1. Файл anketa.htm

```
<html>
<head>
<title>Анкета</title>
</head>
<body>
<h1 align=center>Анкета студентов БТЭУ</h1>
<form action="executor.php" method="POST">
<table border="0">
<tr>
```

```
<td align=right>Имя</td>
<td ><input name="name"></td>
</tr>
<tr>
<td align=right>Отчество</td>
<td ><input name="sername"></td>
</tr>
<tr>
<td align=right>Фамилия</td>
<td ><input name="famil"></td>
</tr>
<tr>
<td align=right>Дата</td>
<td ><input size="10" name="date"></td>
</tr>
<tr>
<td align=right>Факультет:</td>
<td >
<select size="1" name="fakult">
<option>Учетно-финансовый</option>
<option>Экономики и управления</option>
<option>Коммерческий</option>
<option>Не знаю</option>
</select>
</td>
</tr>
<tr>
<td align=right>Пол:</td>
<td ><input name="gender" value="men" type="radio">Мужской<input name="gender" value="female" type="radio">Женский</td>
</tr>
<tr>
<td align=right>Номер группы:</td>
<td ><input size="4" name="group"></td>
</tr>
<tr>
<td align=right>Почтовый адрес:</td>
<td ><input size="25" name="email"></td>
</tr>
<tr>
<td align=right>подтверждение введенных данных</td>
</tr>
<tr>
<td align=right>
<input value="Отправить" name="B1" type="submit">
</td>
<td ><input name="b2" value="Сброс" type="reset"></td>
</tr>
</table>
<br>
</form>
</body>
</html>
```

Результат отображен на рис. 12.

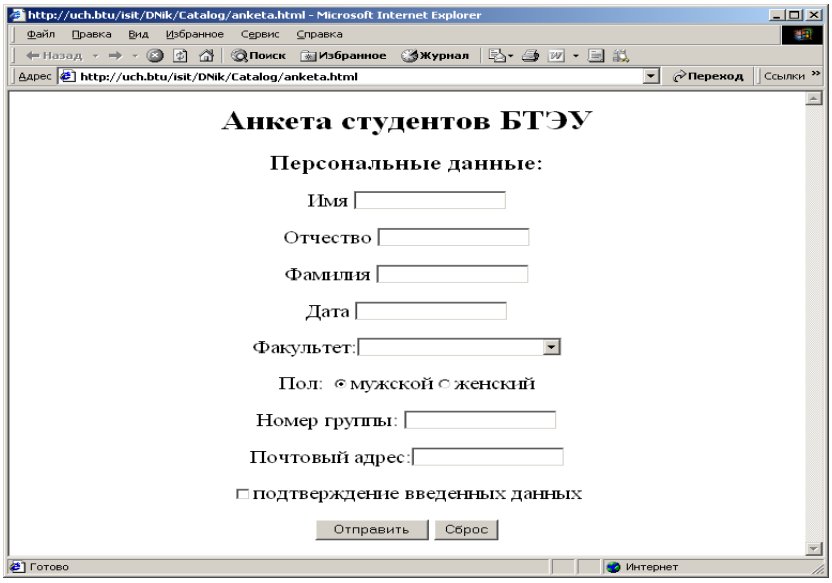

Рис. 12. **Пример оформления анкеты студента**

2. Напишите скрипт (файл *executor.php*), который принимает данные, введенные в форму, и проверяет правильность заполнения формы:

- все ли поля заполнены;
- принадлежит ли адрес электронной почты серверу *uch.btu*.

Сделать проверку заполнения всех полей проще всего при помощи функции strlen(). Чтобы не пришлось проверять значение функции strlen() для каждой переменной, содержащей значение поля, можно просто найти произведение этих значений. Если хотя бы для одной переменной значение функции strlen() будет равно нулю, то и все произведение также будет равно нулю.

Для проверки, находится ли адрес электронной почты на сервере *uch.btu*, можно использовать команду strops(). Функция вернет числовую позицию первого вхождения подстроки «@uch.btu» в строке, содержащейся в переменной $email. Если искомая подстрока в строке не встречается, функция вернет нулевое значение.

Если вся информация в форме введена корректно, то данные о студенте записываются в файл *base.cvs*. Данные по одному студенту записываются одной строкой.

Код второго файла представлен в листинге 7.2.

Листинг 7.2. Файл executor.php

```
<?php
$name=$_POST['name'];
$sername=$_POST['sername'];
$famil=$_POST['famil'];
$date=$_POST['date'];
$gender=$_POST['gender'];
$fakult=$_POST['fakult'];
$email=$_POST['email'];
$verno=$_POST['verno'];
$nowdate=date("d.m.y - H:i:s");
$base="base.cvs";
$filform=strlen($name)*strlen($sername)*strlen($famil)*strlen($gender)
*strlen($fakult)*strlen($email)*strlen($verno);
$fillmail=strpos($email, "@uch.btu");
if($fillform<1)
print("Вы заполнили не все поля<br>вернитесь <a href='Anketa.htm'>назад</a><br>");
elseif($fillmail<1)
print("Не верный адрес электронной почты<br>вернитесь <a href='Anketa.htm'>назад</a><br>");
else
{
$text=$name ."^^". $sername ."^^". $famil ."^^". $date ."^^". $gender ."^^". $fakult ."^^". $email .
"^^". $nowdate ."\n";
$fp=fopen($base,"a");
fputs($fp, $text);
fclose($fp);
}
?>
```

Результат работы скрипта показан на рис. 13.

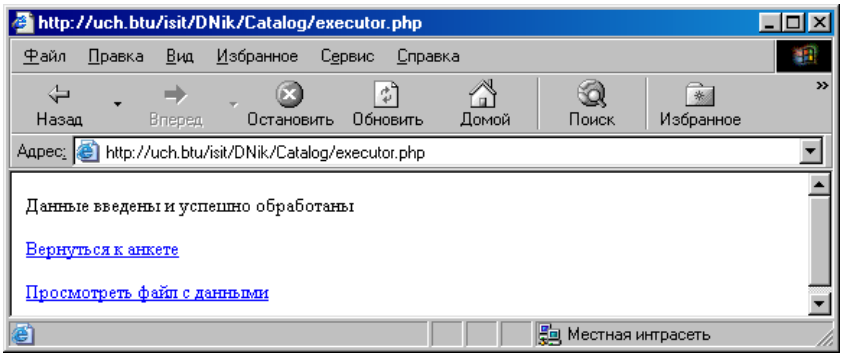

Рис. 13. **Проверка полноты внесенных данных**

3. Напишите скрипт (файл *vuver.php*), который извлекает данные из файла *base.cvs* и выводит их на экран в виде таблицы. Количество строк в таблице определяется как количество строк в файле *base.cvs*.

Код третьего файла представлен в листинге 7.3.

Листинг 7.3. Файл vuver.php

```
<?php
$base="base.cvs";
$file=fopen($base,"r");
echo("<table>");
while($line=fgetcsv($file,1000, "^^")):
$num=count($line);
echo("<tr>");
for($c=0; $c<$num; $c++)
{
print "<td>" . $line[$c] . "</td>";
}
echo("</tr>");
endwhile;
echo("</table>");
fclose($file);
?>
```

Результат работы скрипта показан на рис. 14.

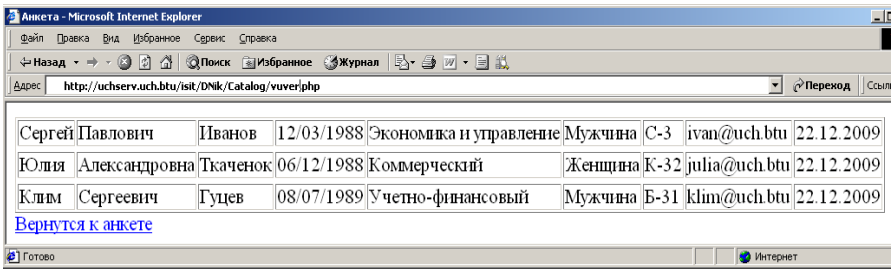

Рис. 14. **Пример вывода файла в окне браузера**

Задания для самостоятельного выполнения

Задание 1. Сделайте гиперссылку для перехода со страницы *vuver.php* на страницу *anketa.htm*.
Задание 2. На странице *vuver.php* добавьте заголовок к таблице. Заголовок разместите по центру.
Задание 3. Определите количество студентов, обучающихся на учетно-финансовом факультете.
Результаты работы продемонстрируйте преподавателю.

Лабораторная работа 8
Электронный магазин

Необходимо создать электронный магазин-витрину по продаже товаров через Интернет. В рассмотренном примере в магазине продается бытовая техника: утюги, тостеры и часы.

Магазин будет состоять из:

- стартовой страницы с названием магазина и местом для рекламы других магазинов;
- каталога товаров;
- блока по обработке заказа.

При создании электронного магазина необходимо предусмотреть обработку некорректно введенных данных. После подтверждения заказа покупатель и менеджер магазина должны получить письма по электронной почте с уведомлением о заказе. Данные о всех заказах должны накапливаться в текстовом файле.

Логическая схема Web-узла представлена на рис. 15.

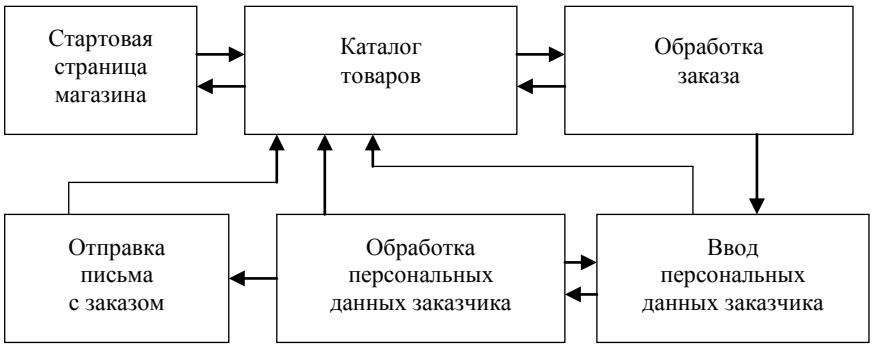

Рис. 15. **Логическая схема Web-узла электронного магазина**

Указания по выполнению

1. В редакторе FrontPage 2003 создайте стартовую страницу (файл *index.htm*) своего магазина (рис. 16 и листинг 7.4). Название магазина, оформление и ассортимент товаров определите самостоятельно.

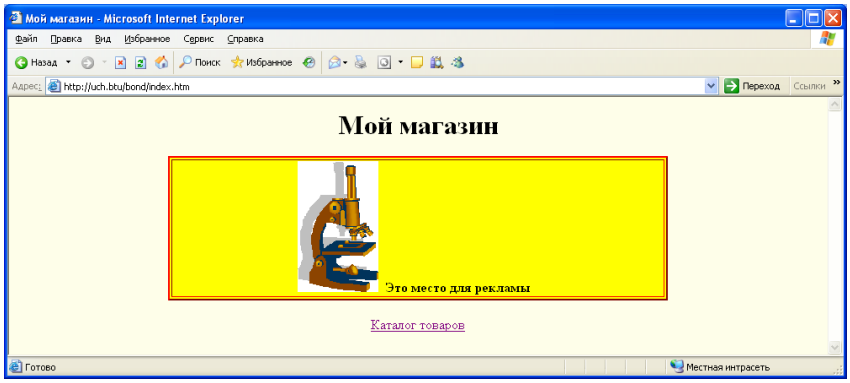

Рис. 16. **Стартовая страница магазина**

Листинг 7.4. Файл index.htm

```
<html>
<head>
<title>Мой магазин</title>
</head>
<body style="text-align: center" bgcolor="#FEFEE9">
<h1 align="center">Мой магазин</h1>
<table border="2" id="table1" bordercolorlight="#FF0000" bordercolordark="#800000" bordercolor="#00FF00"
bgcolor="#FFFF00" height="64" width="594">
<tr>
<td align="center">
<p align="center"><b>
<img border="0" src="j0305257.gif" width="96" height="155">
Это место для рекламы</b>
</td>
</tr>
</table>
<p align="center"><span lang="ru"><a href="catalog.htm">Каталог товаров</a></span></p>
</body>
</html>
```

2. В редакторе FrontPage создайте каталог товаров магазина (файл *catalog.htm*). Для этого выполните следующие действия:

- Создайте форму с двумя кнопками *Заказать* и *Сброс*.

• Внутри формы поместите таблицу из пяти столбцов и четырех строк.
• Заполните таблицу с данными (рис. 17 и листинг 7.5), в последнем столбце во вторую, третью и четвертую строки вставьте по одному текстовому полю.

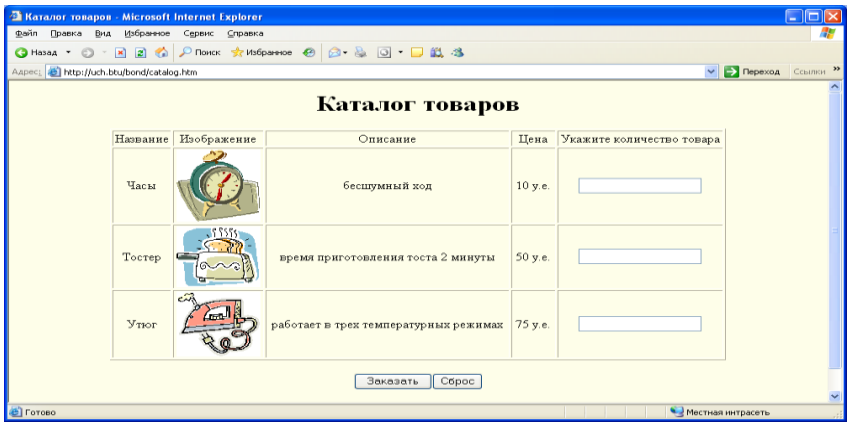

Рис. 17. **Каталог товаров магазина**

Листинг 7.5. Файл catalog.htm

```
<html>
<head>
<title>Каталог товаров</title>
</head>
<body style="text-align: center" bgcolor="#FEFEE9">
<h1 align="center">Каталог товаров</h1>
<form method="POST" action="zakaz.php">
<table border="1" width="732">
<tr>
<td align="center">Название</td>
<td align="center">Изображение</td>
<td align="center"><p align="center">Описание</td>
<td align="center" nowrap width="49">Цена</td>
<td align="center" width="193">Укажите количество товара</td>
</tr>
<tr>
<td align="center">Часы</td>
<td align="center"><img border="0" src="j0234131.gif" width="100" height="106"></td>
<td align="center">бесшумный ход</td>
<td align="center" nowrap width="49">10 у.е.</td>
<td align="center" width="193"><input type="text" name="T1" size="20"></td>
</tr>
<tr>
<td align="center">Тостер</td>
<td align="center"><img border="0" src="j0199283.gif" width="100" height="89"></td>
<td align="center">время приготовления тоста 2 минуты</td>
<td align="center" nowrap width="49">50 у.е.</td>
<td align="center" width="193"><input type="text" name="T2" size="20"></td>
</tr>
<tr>
<td align="center">Утюг</td>
<td align="center"><img border="0" src="j0234266.gif" width="100" height="95"></td>
<td align="center">работает в трех температурных режимах</td>
<td align="center" nowrap width="49">75 у.е.</td>
<td align="center" width="193"><input type="text" name="T3" size="20"></td>
</tr>
</table>
<p><input type="submit" value="Заказать" name="B1">
<input type="reset" value="Сброс" name="B2"></p>
</form>
<p><a href="index.htm">вернуться на главную страницу</a></p>
</body>
</html>
```

• Просмотрите полученную страницу в режиме *С разделением*, запишите в тетрадь названия переменных в текстовых полях, отвечающих за количество заказанного товара (T1, T2, T3) и метод передачи данных (POST).

• Создайте гиперссылку для перехода к стартовой странице магазина.

• Создайте гиперссылку для перехода от стартовой страницы магазина к каталогу товаров.

В редакторе PHP Expert Editor наберите код из листинга 7.6 (файл *zakaz.php*), который обрабатывает данные, поступающие из файла *catalog.htm*. Проверьте корректность выполнения скрипта.

Листинг 7.6. Файл zakaz.php

```
<html>
<head>
<title>Заказ</title>
</head>
<body bgcolor="#FEFEE9">
<div align=center>
<h2>Состояние заказа</h2>
<?php
$z1=$_POST["T1"]; // количество часов
$z2=$_POST["T2"]; // количество тостеров
$z3=$_POST["T3"]; // количество утюгов
$sum_zak=$z1*10+$z2*50+$z3*75; // сумма заказа
if($sum_zak<=0)
{
echo "<h3 align=center>Заказ не сделан</h3><br>
<p align=center><a href='index.htm'>на главную страницу</a></p>
<p align=center><a href='catalog.htm'>каталог товаров</a></p>";
}
else
{
$zak='Вы заказали: <br>'; // состояние заказа
if($z1>0) $zak.="Часы $z1 шт.<br>";
if($z2>0) $zak.="Тостер $z2 шт.<br>";
if($z3>0) $zak.="Утюг $z3 шт.<br>";
echo $zak;
echo "Сумма заказа без учета доставки $sum_zak у.е.<br>";
?>
<form method="POST" action="Oformlen_zak.php">
<br>
<hr>
<input type="hidden" name="sum_zak" value="<?=$sum_zak ?>">
<input type="hidden" name="zak" value="<?=$zak?>">
<p><input type="submit" value="Оформить заказ" name="B1"></p>
</form>
Для изменения заказа вернитесь <a href='catalog.htm'>в магазин</a>
<?php
}
?>
</div>
</body>
</html>
```

Результаты работы скрипта: в случае, если заказ не оформлен, см. рис. 18, если оформлен – рис. 19.

3. Когда заказ сделан, нужно заполнить форму по оформлению заказа (файл *Oformlen_zak.php*). Для этого выполните следующие действия:

• в редакторе FrontPage 2003 создайте форму (рисунки 20 и 21) с двумя кнопками *Принять* и *Сброс*;

• запишите в тетрадь названия переменных всех текстовых полей и радиокнопок (Имя – T1, Пол – R1, Электронный адрес – T3, Город – D1, Адрес доставки – T4, Способ доставки – R2, Способ оплаты – R3).

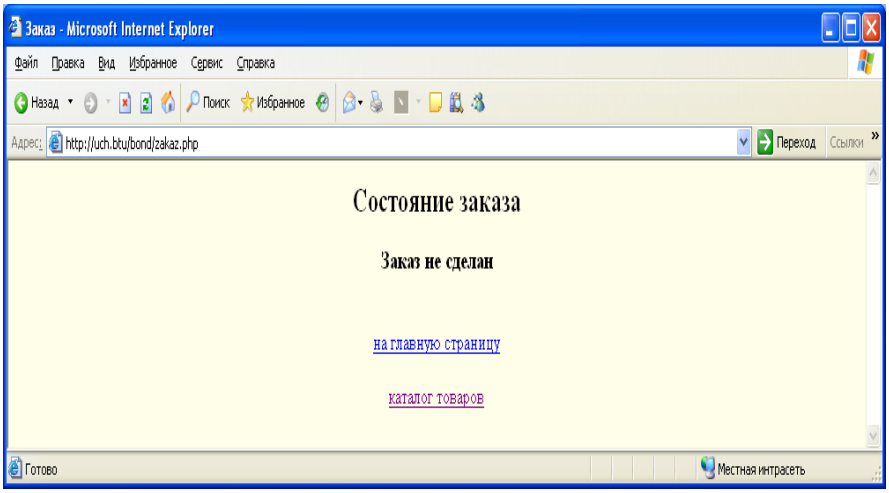

Рис. 18. **Заказ не сделан**

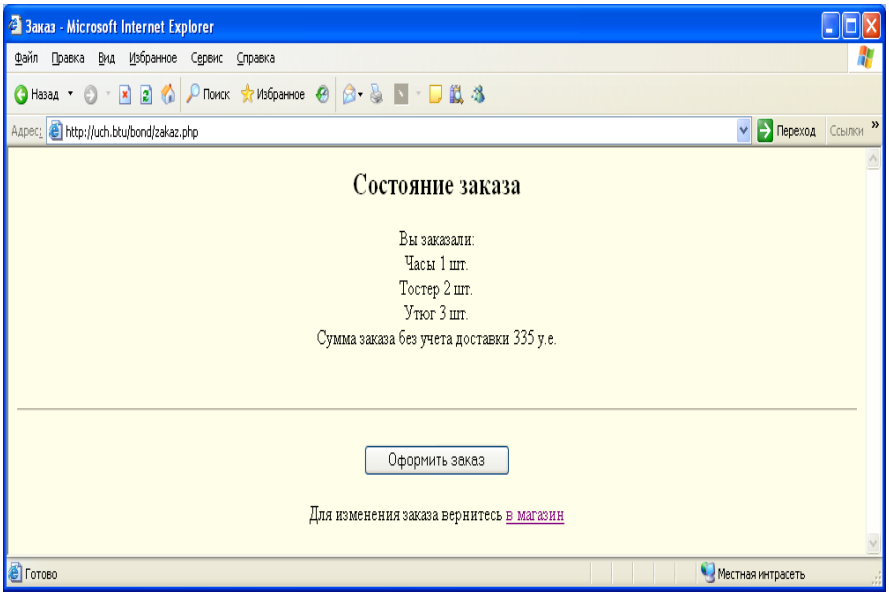

Рис. 19. **Заказ сделан**

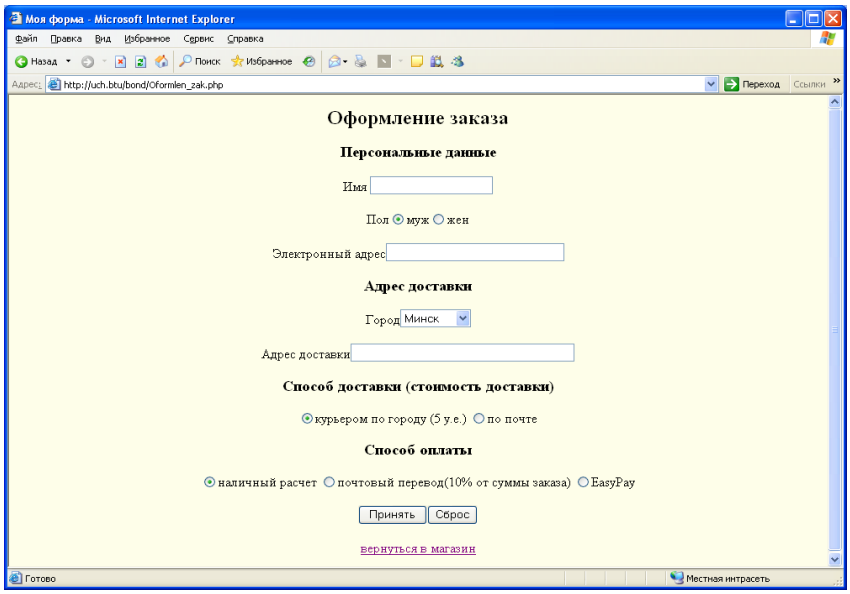

Рис. 20. **Форма оформления заказа**

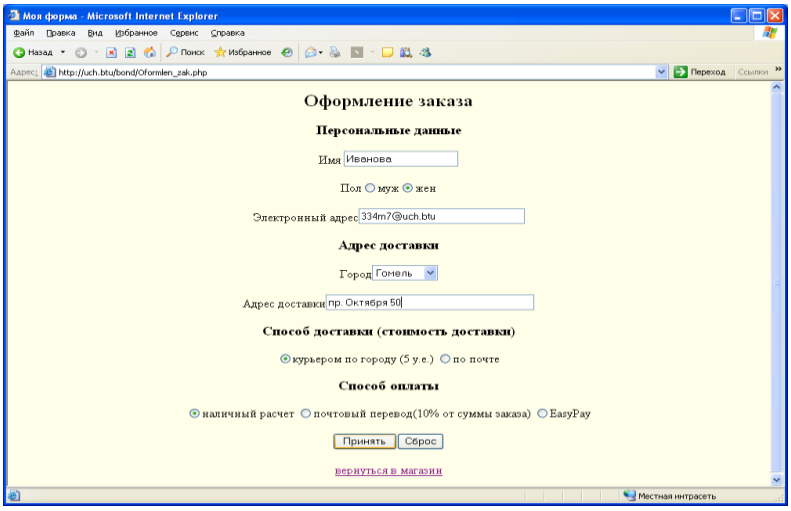

Рис. 21. **Заполненная форма оформления заказа**

Листинг 7.7. Файл Oformlen_zak.php

```
<html>
<head>
<title>Моя форма</title>
</head>
<?php
$sum_zak=$_POST["sum_zak"]; // сумма заказа
$zak=$_POST["zak"]; // заказ
?>
<body bgcolor="#FEFEE9">
<h2 align="center">Оформление заказа</h2>
<form method="POST" action="Form.php">
<h3 align="center">Персональные данные</h3>
<p align="center">Имя <input type="text" name="T1" size="20"></p>
<p align="center">Пол<input type="radio" value="Дорогой" checked name="R1">муж<input type="radio" name="R1" value="Дорогая">жен</p>
<p align="center">Электронный адрес<input type="text" name="T3" size="31"></p>
<h3 align="center">Адрес доставки</h3>
<p align="center">Город<select size="1" name="D1">
<option selected>Минск</option>
<option>Гомель</option>
<option>Гродно</option>
<option>Брест</option>
<option>Могилев</option>
<option>Витебск</option>
</select></p>
<p align=center>Адрес доставки<input type="text" name="T4" size="40"></p>
<h3 align="center">Способ доставки (стоимость доставки)</h3>
<p align="center"><input type="radio" value="1" checked name="R2">курьером по городу (5 у.е.)
<input type="radio" value="2" name="R2">по почте</p>
<h3 align="center">Способ оплаты</h3>
<p align="center"><input type="radio" value="1" checked name="R3">наличный расчет
<input type="radio" value="2" name="R3">почтовый перевод (10% от суммы заказа)
<input type="radio" value="2" name="R3">EasyPay</p>
<input type="hidden" name="sum_zak" value="<?=$sum_zak?>">
<input type="hidden" name="zak" value="<?=$zak?>">
<p align="center"><input type="submit" value="Принять" name="B1"><input type="reset" value="Сброс" name="B2">
</form>
<p align="center"><a href="catalog.htm">вернуться в магазин</a></p>
</body>
</html>
```

4. В редакторе PHP Expert Editor наберите код из листинга 7.7 (имя файла *Form.php*). Код обрабатывает данные, поступающие из файла *Oformlen_zak.php*. Проверьте корректность выполнения скрипта.

Листинг 7.8. Файл Form.php

```
<html>
<body bgcolor="#FEFEE9">
<h2 align=center>Персональные данные</h2>
<hr>
<div align=center>
<?php
// обратите внимание на имена передаваемых переменных!!!!!
$sum_zak=$_POST["sum_zak"]; // сумма заказа
$zak=$_POST["zak"]; // заказ
$pol=$_POST["R1"]; // пол
$name=$_POST["T1"]; // имя
$city=$_POST["D1"]; // город
$adres=$_POST["T4"]; // адрес
$to=$_POST["T3"]; // электронный адрес
$dost=$_POST["R2"]; // способ доставки
// проверяем имя
$str_name=strlen($name);
if($str_name<=0)
print("Не указано имя<br> вернитесь <a href='Oformlen_zak.php'>назад</a><br>");
else
{
// начинаем формировать сообщение электронного письма
$message="$pol $name<br>";
// проверяем адрес почты, принимаются только адреса внутренней сети
$pos1=strstr($to, 'm');
$pos2=strpos($pos1, "@uch.btu");
if($pos2===false)
print("Электронный адрес введен не верно<br> вернитесь <a href='Oformlen_zak.php'>назад</a><br>");
else
{
// проверяем адрес доставки
$str_adr=strlen($adres);
if($str_adr<=0)
print("Не указан адрес доставки товара<br> вернитесь <a href='Oformlen_zak.php'>назад</a><br>");
else
{
// продолжаем формировать сообщение электронного письма
$message.="Ваш заказ будет доставлен по адресу: г.$city $adres<br>";
if($dost=="1")
$summ=$sum_zak+5;
else
$summ=$sum_zak;
// заканчиваем формировать сообщение электронного письма
$message.=$zak."Общая сумма заказа равна $summ<br>";
print $message;
/*создаем форму со скрытыми полями для передачи переменных $message и $to в файл picmo.php*/
?>
<form method="POST" action="picmo.php">
<br>
<hr>
<input type="hidden" name="message" value="<?=$message?>">
<input type="hidden" name="to" value="<?=$to?>">
<p><input type="submit" value="Подтвердить заказ" name="B1"></p>
</form>
<?php
}
}
}
?>
<p align="center"><a href="catalog.htm">вернуться в магазин</a></p>
</div> </body>
</html>
```

Результаты работы скрипта: в случае, если не указано имя, см. рис. 22, если все данные были введены корректно – рис. 23.

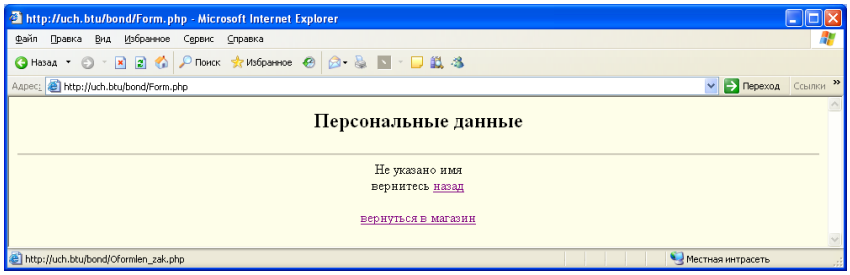

Рис. 22. **В форме было не указано имя**

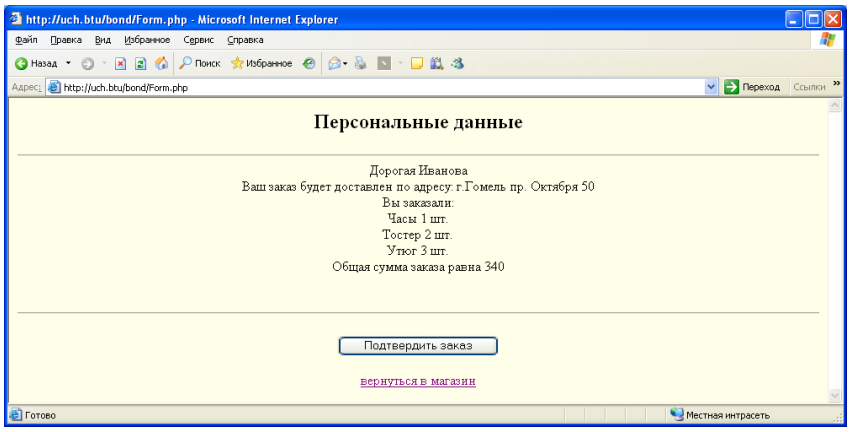

Рис. 23. **В форме все данные были введены корректно**

5. В редакторе PHP Expert Editor наберите код из листинга 7.8 (имя файла *picmo.php*). Код обрабатывает данные, поступающие из файла *Form.php*, и отправляет письмо заказчику товара. Проверьте корректность выполнения скрипта.

Листинг 7.9. Файл picmo.php

```
<html>
<head>
<title></title>
</head>
<body bgcolor="#FEFEE9">
<h2 align="center">Ваш заказ принят</h2><br>
<h3 align="center">Удачных покупок</h2><br>
<?php
$to=$_POST["to"]; // электронный адрес
$message=$_POST["message"]; // сообщение
$subject="Мой магазин"; // тема письма
$today=date("j.m.Y"); // дата заказа
$message.="Дата заказа: $today<br>Удачных покупок<br>";
// заменяем в переменной $message тег <br> на перевод строки \n и каретки \r
$message=str_replace("<br>", "\r\n", $message);
// кодировка письма
$headers="Content-Type: text/html; charset=Windows-1251\r\n";
// определим отправителя письма
$headers.="From: Ivanov <320m1@uch.btu>\r\n\r\n";
// теперь отправим письмо
mail($to, $subject, $message, $headers);
?>
<p align="center"><a href="catalog.htm">вернуться в магазин</a></p>
</body>
</html>
```

Результат работы скрипта представлен на рис. 24.

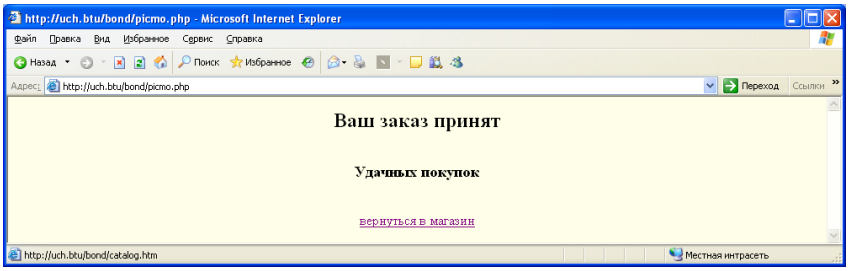

Рис. 24. **Окончание заказа**

6. Самостоятельно подготовьте и отправьте письмо о заказе менеджеру магазина (себе).
7. Разместите на своей стартовой странице рекламу двух других магазинов в виде баннеров.
8. Создайте текстовый файл, в котором будут накапливаться данные о заказах.

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

Кузнецов, М. В. Самоучитель PHP 5. / М. В. Кузнецов, И. В. Симдянов. – 2-е изд., перераб. и доп. – СПб. : БХВ-Петербург, 2006. – 608 с.

Мархвида, И. В. Создание Web-страниц: HTML, CSS, JavaScript / И. В. Мархвида. – Минск : Новое знание, 2002. – 352 с.

Коггзолл, Дж. PHP 5: полное руководство : [пер. с англ.] / Дж. Коггзолл. – М. : ИД «Вильямс», 2006. – 752 с.

ПРИЛОЖЕНИЕ

ОСНОВЫ HTML-ФОРМ

Общее описание

Форма – это раздел HTML-документа, который наряду с обычным содержанием содержит *управляющие элементы*: поля ввода, кнопки, флажки, переключатели, меню и т. д. Пользователь заполняет форму, изменяя ее управляющие элементы (например, путем ввода текста или выбора пункта меню), а затем пересылает заполненную форму для обработки серверу.

Таким образом, центральной частью любой формы являются ее управляющие элементы. Каждый из них имеет имя, заданное атрибутом NAME и действующее в пределах текущей формы. Кроме того, он имеет начальное и текущее значения, которые являются символьными строками. Начальное значение элемента задается его атрибутами и становится текущим при первом отображении формы или при ее повторной инициализации. В дальнейшем пользователь может изменить текущее значение элемента. Когда заполненная форма пересылается для обработки, пересылается и часть ее управляющих элементов в виде пар «имя – значение». Такие управляющие элементы называются *успешными*.

Создание форм

Формы создаются при помощи тега <FORM>…</FORM>. Этот тег определяет в HTML-документе раздел, в котором находятся все элементы управления формы. Тег <FORM> имеет набор необязательных атрибутов, которые определяют его поведение при отправке формы (табл. П1).

Таблица П1. **Атрибуты дескриптора <FORM>**

Атрибут	Описание
ACTION	URL-адрес, по которому отправляется форма
METHOD	Метод отправки формы (GET или POST)
ENCTYPE	Используемый тип кодировки

Элементы текстового поля и поля ввода пароля

Текстовое поле представляет собой однострочное поле ввода, определяется тегом <INPUT>, а также установкой значения его атрибута TYPE, равным TEXT. В табл. П2 перечислены допустимые атрибуты для текстовых полей и их значения.

Таблица П2. **Атрибуты текстового поля**

Атрибут	Описание
NAME	Имя, присвоенное текстовому полю
SIZE	Размер текстового поля в браузере (в символах)
MAXLENGTH	Максимальное количество символов, принимаемое текстовым полем
VALUE	Значение по умолчанию для текстового поля

Листинг П1. Создание текстового поля в HTML

```
<INPUT TYPE="text" NAME="mytextfield"
VALUE="Значение по умолчанию" SIZE=30 MAXLENGTH=30>
```

Поле ввода пароля позволяет организовать такой же однострочный ввод, но, в отличие от текстового поля, поле ввода пароля маскирует вводимые символы таким образом, чтобы они не могли быть прочитаны с экрана. Для создания поля ввода пароля установите значение атрибута TYPE равным PASSWORD. Текстовые поля и поля ввода пароля принимают один и тот же набор атрибутов.

Листинг П2. Создание поля ввода пароля в HTML

```
<INPUT TYPE="password" NAME="mypassword"
VALUE="Вы не сможете прочесть это в браузере">
```

Элементы переключателя и флажка

Использование переключателей позволяет выбирать один пункт из списка допустимых. В HTML переключатель может быть создан установкой значения атрибута TYPE тега <INPUT>, равного RADIO. Элемент переключателя допускает только три атрибута: NAME, VALUE и CHECKED. При работе с переключателями необходимо соблюдать следующие правила:

- чтобы набор переключателей работал правильно, как единая группа (т. е. только один из них мог быть отмечен), каждый переключатель в наборе должен иметь одно и то же значение атрибута NAME;
- атрибуту CHECKED не присваивается значение, и только один переключатель в группе может иметь этот атрибут (листинг П3);
- атрибут VALUE не отображается в браузере, а передается как значение при отправке данных формы.

Листинг П3. Создание группы переключателей в HTML

```
Любимый вид спорта
<INPUT TYPE="radio" NAME="myradio"
CHECKED VALUE="1">Американский футбол< BR >
<INPUT TYPE="radio" NAME="myradio" VALUE="2">Футбол<BR>
<INPUT TYPE="radio" NAME="myradio" VALUE="3">Хоккей<BR>
<INPUT TYPE="radio" NAME="myradio" VALUE="4">Бейсбол<BR>
```

Флажки подобны переключателям, но позволяют отмечать любое количество из представленных позиций. Флажок создается установкой значения атрибута TYPE в теге <INPUT>, равного CHECKBOX. В отличие от переключателей, здесь требуется устанавливать различные значения атрибута NAME для всей группы.

Флажки имеют тот же набор атрибутов, что и переключатели (NAME, VALUE и CHECKED).

Листинг П4. Создание группы флажков в HTML

```
Просмотренные телевизионные передачи
<IMPUT TYPE="checkbox" NAME="mycheckbox1"
VALUE="1">Американский футбол<BR>
<INPUT TYPE="checkbox" NAME="mycheckbox2"
CHECKED VALUE="2">Футбол<BR>
<INPUT TYPE="checkbox" NAME="mycheckbox3"
CHECKED VALUE="3">Хоккей<BR>
<INPUT TYPE="checkbox" NAME="mycheckbox4"
VALUE="4">Бейсбол<BR>
```

Списки и выпадающие списки

Список может быть представлен в виде одной строки, в которой пользователь щелкает по стрелке, чтобы просмотреть все возможные варианты выбора (выпадающий список), или может иметь вид стандартного прокручиваемого списка, в котором выбирается один или более элементов. Это обеспечивается двумя HTML-тегами: тегом <SELECT>…</SELECT>, определяющим список (табл. П3), и тегом <OPTION>…</OPTION>, используемым для определения элементов списка (табл. П4).

Таблица П3. **Атрибуты HTML-тега <SELECT>**

Атрибут	Описание
NAME	Имя, присвоенное списку
SIZE	Количество элементов для одновременного отображения в списке (значение 1 обозначает выпадающий список)
MULTIPLE	Флаг, указывающий на то, что можно выбирать одновременно несколько элементов

Таблица П4. **Атрибуты HTML-тега <OPTION>**

Атрибут	Описание
VALUE	Значение, которое отправляется, если элемент выбран
SELECTED	Флаг, указывающий, что данный элемент выбран по умолчанию

Листинг П5. Использование списков в HTML

```
Выбор любимого цвета
<SELECT NAME="цвета" SIZE=1>
<OPTION VALUE="красный">Мне нравится красный</OPTION>
<OPTION VALUE="синий">Мне нравится синий</OPTION>
< OPTION VALUE="зеленый">Мне нравится зеленый</OPTION>
</SELECT><BR>
выбор одного или более любимых блюд
<SELECT NAME="Блюда" SIZE=4 MULTIPLE>
<OPTION VALUE="Китайские">Мне нравятся китайские блюда</OPTION>
<OPTION VALUE="Русские">Мне нравятся русские блюда </OPTION>
<OPTION VALUE="Итальянские">Мне нравятся итальянские блюда</OPTION>
<OPTION VALUE="никакие">Мне не нравятся никакие из этих блюд </OPTION>
</SELECT>
```

Многострочные текстовые поля

Чтобы дать возможность вводить множество строк, нужно использовать элемент <TEXTAREA>…</TEXTAREA> (табл. П5).

Таблица П5. **Атрибуты HTML-тега <TEXTAREA>**

Атрибут	Описание
NAME	Имя элемента
COLS	Ширина текстового поля в символах
ROWS	Высота текстового поля в строках
WRAP	Определяет, как текст передается относительно того, как он печатается в поле

Использование атрибутов COLS, ROWS и NAME вполне очевидно. Атрибут WRAP принимает одно из трех значений: off, soft и hard. Эти значения определяют, как текст будет передаваться относительно того, как печатается:

- off – текстовое поле не заворачивает свое содержимое (ввод продолжается за границей текстового поля), и текст передается в точности так, как он напечатан;
- soft – текст будет заворачиваться в пределах текстового поля, однако он будет передаваться так, как был напечатан;
- hard – текст будет заворачиваться в пределах текстового поля и передаваться вместе с включенными символами новой строки.

Листинг П6. Использование элемента <TEXTAREA>

```
<TEXTAREA R0WS="5" COLS="30" WRAP="hard">
Это моя текстовая область</TEXTAREA>
```

Скрытые значения формы

HTML-формы позволяют отправлять нередактируемые данные с помощью скрытых значений формы (листинг П7). Такие значения создаются путем установки значения атрибута TEXT тега <INPUT>, равного HIDDEN. Скрытые (hidden) элементы формы никогда не отображаются в браузере клиента, они только передаются на сервер при отправке формы. Скрытые элементы формы очень часто используются при взаимодействии со сценариями. Атрибуты скрытого элемента NAME и VALUE представляют его имя и значение.

Листинг П7. Использование скрытых элементов форм

```
<INPUT TYPE="HIDDEN" NAME="myvalue" VALUE="foo">
```

Кнопки

Все данные формы, которые нужно передать из клиентского браузера на сервер, должны быть отправлены. Для этого используется элемент *кнопка*, который выполняют такую передачу в результате щелчка по ней. Тег <INPUT> позволяет создавать кнопки четырех видов, каждый из которых имеет свое значение атрибута TYPE:

1. SUBMIT – это элемент отправки формы, которая имеет два атрибута: NAME и VALUE. Если необходимость в идентификации, какая именно кнопка использовалась для отправки формы, отсутствует, то атрибут NAME может быть опущен. Атрибут VALUE используется для отображения надписи на кнопке (например, «Отправить форму»).
2. Элемент IMAGE ведет себя аналогично стандартному элементу отправки, однако вместо кнопки он отображает указанное графическое изображение. При использовании такого элемента отправки доступны все атрибуты HTML-тега <IMG>.
3. Элемент RESET предназначен для сброса значений формы, т. е. очистки. Надпись на кнопке задается также с помощью атрибута VALUE. Применение рассмотренных элементов отправки формы продемонстрировано в листинге П8.

Листинг П8. Использование элементов submit и image

```
<INPUT TYPE="submit" NAME="mysubmit"
VALUE="Это кнопка отправки по умолчанию">
<INPUT TYPE="image" NAME="myimagesubmit"
SRC="/images/mybutton.gif">
<INPUT TYPE="reset" NAME="myreset" VALUE="Очистить">
```

4. Элемент BUTTON выглядит и функционирует так же, как и элемент отправки формы, описанный выше. Однако, в отличие от него, кнопка не имеет ассоциированного с ней действия по умолчанию (листинг П9). Вместо этого ей должно быть назначено действие, запрограммированное на сценарном языке клиентской стороны, таком как JavaScript. Поскольку тема JavaScript по объему занимает целую книгу, подробности работы этого элемента формы здесь не рассматриваются, а его упоминание объясняется только целью полноты изложения.

Листинг П9. Использование кнопки

```
<INPUT TYFE="button"
VALUE="Щелкни!" onClick="alert('Вы щелкнули по кнопке');">
```

СОДЕРЖАНИЕ

Учебное издание

ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ

WEB-ПРОГРАММИРОВАНИЕ НА PHP

Практикум
к лабораторным занятиям
для студентов специальности 1-26 03 01
«Управление информационными ресурсами»

Авторы-составители:
Бондарева Валентина Викторовна
Никитенко Дмитрий Владимирович

Редактор М. П. Любошенко
Технический редактор И. А. Козлова
Компьютерная верстка Н. Н. Короедова

Подписано в печать 15.01.10. Бумага типографская № 1.
Формат 60 × 84 $^{1}/_{16}$. Гарнитура Таймс. Ризография.
Усл. печ. л. 4,18. Уч.-изд. л. 4,00. Тираж 100 экз.
Заказ №

Учреждение образования
«Белорусский торгово-экономический университет
потребительской кооперации».
246029, г. Гомель, просп. Октября, 50.
ЛИ № 02330/0494302 от 04.03.2009 г.

Отпечатано в учреждении образования
«Белорусский торгово-экономический университет
потребительской кооперации».
246029, г. Гомель, просп. Октября, 50.