Федеральное агентство по образованию
ГОУ ВПО «Уральский государственный технический университет – УПИ
имени первого Президента России Б. Н. Ельцина»

Н. Г. Дружинина, О. Г. Трофимова

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЙ С ПОМОЩЬЮ PHP

Методические указания к лабораторной работе по дисциплине
«Информационное обеспечение систем управления»

Учебное электронное текстовое издание
Подготовлено кафедрой «Автоматика и управление в технических системах»
Научный редактор: доц., канд. техн. наук Ю.Н. Чесноков

Методические указания предназначены для студентов всех форм обучения специальности 220201 «Управление и информатика в технических системах».

В методических указаниях рассматриваются основы языка программирования PHP: синтаксис, стандартные функции языка, регулярные выражения. Приведено описание функций PHP для работы с массивами. Приведен пример создания простых WEB-приложений. Элементарные задания к лабораторной работе позволят создать студентам WEB-сайт с помощью PHP и практически закрепить теоретический материал.

Екатеринбург
2009

Оглавление

Цель работы

Получить практические навыки написания *PHP*-отчетов и Web-программирования. Познакомиться с основами языка программирования *PHP*.

Введение

История *PHP* начинается с 1995 года. В 1997 году было решено, что сокращение *PHP* должно означать не «*Personal Home page*», а «*PHP Hypertext Processor*». *PHP* – язык программирования, предназначенный для создания web-страницы с использованием баз данных.

Одним из главных достоинств *PHP* является тот факт, что он внедряется прямо в HTML-код, поэтому программисту не приходится писать программу с множеством команд для простого вывода HTML. Код HTML и *PHP* можно чередовать по мере необходимости. *PHP* позволяет написать фрагмент следующего вида:

```
<html>
<title><? print «Hello world!»; ?></title>
</html>
```

1. Характеристика языка PHP

Главным фактором при проектировании языка *PHP* является практичность. *PHP* предоставляет программисту средства для быстрого и эффективного решения поставленных задач. Практический характер *PHP* обусловлен пятью важными характеристиками: традиционностью, простотой, эффективностью, безопасностью, гибкостью.

Существует еще одна «характеристика», которая делает *PHP* особенно привлекательным: он распространяется бесплатно.

Для работы с *PHP* необходимо загрузить, установить и настроить PHP и Web-сервер на компьютере. *PHP* совместим с разными Web-серверами. Лучше, что использовать *Apache* – во-первых, это самый популярный Web-сервер на сегодняшний день, во-вторых, он чаще всего работает с *PHP*.

1.1. Переход в PHP

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

1.2. Стандартные теги

Стандартные теги используются программистами *PHP* чаще остальных способов, что объясняется наглядностью и удобством этой формы записи (см. листинг 1):

Листинг 1. Вывод кода HTML средствами *PHP*

```
<?php print «Welcome to the world of PHP!»: ?>
```

Весь текст, расположенный до закрывающего тега ?>, интерпретируется как код *PHP*.

Одной из самых замечательных особенностей HTML является простота использования в сочетании с другими языками – например, HTML и *JavaScript* (см. листинг 2).

Листинг 2. Вывод кода HTML средствами *PHP*

```
<html>
<head>
<title>Basic PHP/HTML integration</title>
</head>
<body>
<?
print «<h3>PHP/HTML integration is cool.</h3>»;
?>
</body>
</html>
```

В листинге 3 продемонстрировано включение динамической информации в *WEB*-страницу на примере вывода текущей даты в заголовке окна.

Листинг 3. Динамический вывод даты

<title>PHP Recipes | <? print (date(«F d, Y»)); ?></title>

PHP также позволяет изменять формат конструкций HTML – для этого соответствующая характеристика тега присваивается переменной, вставляемой в файл. В листинге 4 эта возможность продемонстрирована на примере присваивания характеристики шрифта (*h*3) переменной $*big_font* и ее последующего использования при выводе текста.

Листинг 4. Динамические теги *HTML*

```
<html>
<head>
<title>PHP Recipes | <? print (date(«F d, Y»)); ?></title>
</head>
<?
$big_font = «h3»;
?>
<body>
<? print «<$big_font>PHP Recipes</$big_font>»; ?>
</body>
</html>
```

1.3. Переменные и типы данных

Типы данных составляют основу любого языка программирования и являются средством, с помощью которого программист представляет разные типы информации. В *PHP* поддерживаются шесть основных типов данных: целые числа, вещественные числа, строки, массивы, объекты, логические величины.

Переменная представляет собой именованную область памяти, содержащую данные, с которыми можно выполнять операции во время выполнения программы.

Имена переменных всегда начинаются со знака доллара $. Ниже приведены примеры допустимых имен переменных:

$*color, $operating_system,* $*_some_variable,* $*model.*

Имена переменных должны соответствовать тем же условиям, что и идентификаторы. Другими словами, имя переменной начинается с буквы или символа подчеркивания и состоит из букв, символов подчеркивания, цифр или других *ASCII*-символов в интервале от 127 до 255.

Переменная объявляется при первом ее использовании в программе. Более того, тип переменной косвенно определяется по типу хранящихся в ней данных. Рассмотрим следующий пример:

$*sentence = «This is a sentence»; //* $*sentence* интерпретируется как строка;

$*price* = 42.99: // $*price* интерпретируется как вещественное число;

$*weight* = 185; // $*weight* интерпретируется как целое число.

Область видимости (*scope*) переменных определяется как область доступности переменной в той программе, в которой она была объявлена. В зависимости от области видимости переменные *PHP* делятся на четыре типа: локальные переменные; параметры функций; глобальные переменные; статические переменные.

1.4. Параметры функций

В *PHP* любые параметры, передаваемые функции при вызове, должны быть объявлены в заголовке функции. Хотя параметрам присваиваются аргументы, переданные извне, после выхода из функции они становятся недоступными.

Параметры объявляются в круглых скобках после имени функции. Объявление параметров практически не отличается от объявления типичной переменной: функция умножает переданное значение на 10 и возвращает результат (см. пример 1).

Пример 1.

```
function x10 ($value)
{
$value = $value * 10;
return $value;
}
```

1.5. Глобальные переменные

Глобальные переменные в отличие от локальных доступны в любой точке программы. Но чтобы изменить значение глобальной переменной, необходимо специально объявить ее как глобальную в соответствующей функции. Для этого перед именем переменной ставится ключевое слово *GLOBAL* (см. пример 2).

Пример 2.

```
$somevar = 15;
function addit()
{
GLOBAL $somevar;
$somevar++;
print «Somevar is $somevar»;
}
addit();
```

Будет выведено значение $*somevar*, равное 16.

1.6. Константы

Константой называется именованная величина, которая не изменяется в процессе выполнения программы. Константы особенно удобны при работе с заведомо постоянными величинами – например, числом π (3,141592). В *PHP* константы определяются функцией *define*(). После того как константа будет определена, вы не сможете изменить (или переопределить) ее в этой программе (см. пример 3).

Пример 3.

```
define(«PI», «3.141592»);
```

1.7. Выражения

Выражение описывает некоторое действие, выполняемое в программе. Каждое выражение состоит, по крайней мере, из одного операнда и одного или нескольких операторов.

1.8. Операнды

Операнд представляет собой некоторую величину, обрабатываемую в программе. Примеры операндов:

$*a*++; где *$a* – операнд;

$sum = $val1 + $val2; где $*sum,* $*val1* и $*val2* – операнды.

1.9. Операторы

Оператор представляет собой символическое обозначение некоторого действия, выполняемого с операндами в выражении. В табл. 1 приведен полный список всех операторов, упорядоченных по убыванию приоритета.

Таблица 1

Операторы *PHP*

Оператор	Ассоциативность	Цель
( )	–	Изменение приоритета
new	–	Создание экземпляров объектов
! ~	П	Логическое отрицание, поразрядное отрицание
++ —	П	Инкремент, декремент
@	П	Маскировка ошибок
/ * %	Л	Деление, умножение, остаток
+ —.	Л	Сложение, вычитание, конкатенация
<< >>	Л	Сдвиг влево, сдвиг вправо (поразрядный)
< <= > >=	–	Меньше, меньше или равно, больше, больше или равно
== != === <>	–	Равно, не равно, идентично, не равно
& ^ |	Л	Поразрядные операции AND, XOR и OR
&& ||	Л	Логические операции AND и OR
?:	П	Тернарный оператор
= += *= /= .=	П	Операторы присваивания
%= &= |= ^=		
<<= >>=		
AND XOR OR	Л	Логические операции AND, XOR и OR

Примеры выражений:

$*a = 5*; – присвоить целое число 5 переменной $*a*;

$*a = «5»;* – присвоить строковую величину «5» переменной $*a*.

1.9.1. Математические операторы

Математические операторы (табл. 2) предназначены для выполнения различных математических операций.

Таблица 2

Математические операторы

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*	Остаток	Остаток от деления $*a* на $*b*

1.9.2. Операторы присваивания

Операторы присваивания задают новое значение переменной (табл. 3).

Таблица 3

Операторы присваивания

Пример	Название	Результат
$*a = 5;*	Присваивание	Переменная $*a* равна 5
$*a*+= *5;*	Сложение с присваиванием	Переменная $*a* равна сумме $*a* и 5
$*a* *= *5;*	Умножение с присваиванием	Переменная $*a* равна произведению $*a* и 5
$*a*/=*5;*	Деление с присваиванием	Переменная $*a* равна частному отделения $*a* на 5
$*a* .= *5;*	Конкатенация с присваиванием	Переменная *$a* равна конкатенации $a и 5

1.9.3. Логические операторы

Логические операторы (табл. 4) обеспечивают средства для принятия решений в зависимости от значения переменных. Логические операторы позво-

ляют управлять порядком выполнения команд в программе и часто используются в управляющих конструкциях (таких, как условная команда *if*, а также циклы *for* и *while*).

Таблица 4

Логические операторы

Пример	Название	Результат
$*a* && $*b*	Конъюнкция	Истина, если истинны оба операнда
$*a and* $*b*	Конъюнкция	Истина, если истинны оба операнда
$*a OR* $*b*	Дизъюнкция	Истина, если истинен хотя бы один из операндов
!$*a*	Отрицание	Истина, если значение *$a* ложно
NOT !$*a*	Отрицание	Истина, если значение *$a* ложно
$*a XOR* $*b*	Исключающая дизъюнкция	Истина, если истинен только один из операндов

Логические операторы часто используются для проверки результата вызова функций:

file_exists(«filename.txt») OR print «File does not exist!»

1.9.4. Операторы равенства

Операторы равенства (табл. 5) предназначены для сравнения двух величин и проверки их эквивалентности.

Таблица 5

Операторы равенства

Пример	Название	Результат
$*a*==$*b*	Проверка равенства	Истина, если $*a* и $*b* равны
$*a!*=$*b*	Проверка неравенства	Истина, если $*a* и $*b* не равны
$*a*===$*b*	Проверка идентичности	Истина, если $*a* и $*b* равны и имеют одинаковый тип

1.9.5. Операторы сравнения

Операторы сравнения (табл. 6), как и логические операторы, позволяют управлять логикой программы и принимать решения при сравнении двух и более переменных.

Операторы сравнения предназначены для работы только с числовыми значениями.

Таблица 6

Операторы сравнения

Пример	Название	Результат
$*a*<$*b*	Меньше	Истина, если переменная $*a* меньше $*b*
$*a*>$*b*	Больше	Истина, если переменная $*a* больше $*b*
$*a*<= $*b*	Меньше или равно	Истина, если переменная $*a* меньше или равна $*b*
$*a*>= $*b*	Больше или равно	Истина, если переменная $*a* больше или равна $*b*
($a-12)?5:л –1	Тернарный оператор	Если переменная $*a* равна 12, возвращается значение 5, а если не равна – то 1

1.9.6. Операторы выбора: if...else

К операторам выбора относят: условный оператор *(if...else)* и переключатель *(switch)*.

Синтаксис условного оператора: *if(condition) statement* 1 *else statement* 2.

Условие *condition* может быть любым выражением. Если оно истинно, то выполняется оператор *statement* 1. В противном случае – оператор *statement* 2.

1.9.7. Операторы выбора: переключатель switch

Переключатель *switch* является наиболее удобным средством для организации мультиветвления. Синтаксис переключателя таков:

switch(*expression*) – переключающее выражение

{

case value1: – константное выражение 1

statements; – блок операторов

break;

case value2: – константное выражение 2

statements;

break;

```
  default:
    statements;
}
```

1.9.8. Операторы цикла: while

Оператор *while* называется оператором цикла с предусловием. При входе в цикл вычисляется выражение условие, и, если его значение отлично от нуля, выполняется тело цикла. Затем вычисления выражения условия и операторов тела цикла выполняется до тех пор, пока значение выражения условия не станет равным нулю (см. пример 4).

Пример 4.

```
<?
  $var = 5;
  $i = 0;
  while(++$i <= $var)
  {
    echo($i); echo('<br>');
  }
?>
```

1.9.9. Операторы цикла: do…while

Оператор цикла *do...while* называется оператором цикла с постусловием. При входе в цикл в любом случае выполняется тело цикла (т. е. цикл всегда будет выполнен хотя бы один раз), затем вычисляется условие, и если оно не равно 0, вновь выполняется тело цикла (см. пример 5).

Пример 5.

```
<?
  $var = 5;
  $i = 0;
  do
  {
    echo($i); echo('<br>');
```

```
  }
  while(++$i <= $var)
?>
```

Итерационный цикл имеет следующий формат:

```
  for(expression 1;expression 2;expression 3)
  {
    statements;
  }
```

Здесь *expression 1* (инициализация цикла) – 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 (см. пример 6).

Пример 6.

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

1.10. Определение и вызов функций

Функции могут создаваться в любой точке программ *PHP*, однако по соображениям структурной организации кода удобнее разместить все функции, используемые сценарием, в самом начале сценарного файла. Определение функции обычно состоит из трех частей:

- имя функции;

- круглые скобки, в которых перечисляются необязательные входные параметры, разделенные запятыми;
- тело функции, заключенное в фигурные скобки.

Обобщенный синтаксис функций *PHP* выглядит так:

function имя_функции ([$параметр 1, $параметр 2, ... $параметр N])

{

тело функции

}

1.11. Создание массивов

Массив представляет собой совокупность объектов, имеющих одинаковый размер и тип. Каждый объект в массиве называется элементом массива. При объявлении индексируемого массива после имени переменной ставится пара квадратных скобок ([ ]):

$*languages[ ] = «Spanish».*

После этого в массив можно добавлять новые элементы, как показано ниже. Обратите внимание: новые элементы добавляются без явного указания индекса. В этом случае новый элемент добавляется в позицию, равную длине массива плюс 1:

$*languages[ ] = «English»;*

$*languages[ ] = «Gaelic».*

Функция *array*( ) получает ноль или более элементов и возвращает массив, состоящий из указанных элементов. Ее синтаксис:

array array([элемент 1, элемент 2...]).

Ниже показан пример использования *array*( ) для создания индексируемого массива:

$*languages = array («English». «Gaelic». «Spanish»).*

1.11.1. Многомерные массивы

Многомерный массив (массив массивов) – средство для хранения информации, требующее дополнительного структурирования. Создать многомерный

массив несложно – просто добавьте дополнительную пару квадратных скобок, чтобы вывести массив в новое измерение:

$*chessboard[1][4] = «King»* – двухмерный массив;

$*capitals[«USA»][«Ohio»] = «Columbus»* – двухмерный массив;

$*streets[«USA»][«Ohio»][«Columbus»] = «Harrison»* – трехмерный массив.

1.11.2. Сортировка массивов

В *PHP* существуют стандартные функции сортировки (табл. 7).

Таблица 7

Функции сортировки

Функция	Сортировка	Обратный порядок	Сохранение пар «ключ/значение»
sort	Значение	Нет	Нет
rsort	Значение	Да	Нет
asort	Значение	Нет	Да
arsort	Значение	Да	Да
ksort	Ключ	Нет	Да
krsort	Ключ	Да	Да
usort	Значение	?	Нет
uasort	Значение	?	Да
uksort	Ключ	?	Да

1.12. Простые ссылки

По ссылкам пользователь может переходить как на обычные страницы *HTML*, так и на страницы, содержащие код *PHP*:

<View today's date.

Если щелкнуть на ссылке, в браузере будет загружена страница с именем *date.php*.

2. Основные структуры HTML-документа

2.1. Структура документа HTML

Документы в языке HTML начинаются с декларации <!*DOCTYPE*>, затем следует элемент HTML, внутри которого содержатся последовательно элементы *HEAD* и *BODY* (см. листинг 4):

Листинг 4. Структура документа HTML

<HTML>

<HEAD>

<TITLE> Текст заголовка</TITLE>

</HEAD>

<BODY>

тело документа

</BODY>

</HTML>

Минимальный документ HTML выглядит следующим образом (см. листинг 5):

Листинг 5. Пример *hello.html*

<TITLE>Hello</TITLE>

Результат листинга 5: *Hello world*.

2.2. Элемент HEAD и его производная TITLE

<!ELEMENT TITLE – – (#PCDATA) -(%head.misc)>*

Согласно спецификации *HTML*, каждый документ обязан иметь ровно один элемент *TITLE* в поле *HEAD*. С его помощью программе конечного пользователя сообщается название-уведомление данного документа, которое может быть выставлено в заголовке над окном соответствующей программы и т.д. Пример элемента *TITLE*:

<TITLE>Изучение динамики популяции</TITLE>.

2.3. Элемент BODY и его производные

Данный элемент содержит собственно тело (текст) документа. В теле документа может содержаться достаточно большой набор элементов: заголовки (*H*1 – *H*6), элемент *ADDRESS*, блочные элементы, элементы на уровне текста.

2.3.1. Заголовки

<!ELEMENT (%heading) – – (%text;)>*

<!ATTLIST (%heading) align (left|center|right) #IMPLIED >

Элементы *H*1, *H*2, *H*3, *H*4, *H*5 и *H*6 используются в документе для разметки заголовков. Всегда нужно указывать как начальный, так и конечный теги. При этом заголовки, размеченные элементами *H*1, главенствуют над заголовками, размеченными элементами *H*2.

2.3.2. Блочные элементы разметки

P – параграфы. Для этого элемента разметки параграфов необходимо указывать начальный тег, но при этом конечный тег может быть всегда опущен.

UL – неупорядоченные списки. В данном случае обязательно указывать как начальный, так и конечный теги, а также один или несколько элементов *LI*, представляющие отдельные пункты списка. В элементах *UL* и *LI* может использоваться атрибут *TYPE*, устанавливающий стиль разметки для данного списка. Неупорядоченные списки имеют вид:

<UL>

<LI> ... первый пункт списка

<LI> ... второй пункт списка

...

</UL>.

OL – упорядоченные (т. е. нумерованные) списки. Здесь также требуется указывать как начальный, так и конечный теги, а также один или несколько элементов *LI*, представляющие в списке отдельные пункты. Упорядоченные (или нумерованные) списки имеют следующий вид:

<OL>

<LI> ... первый пункт списка

<LI> ... второй пункт списка

...

</OL>.

DL – списки определений. Требуется указывать начальный и конечный теги. В таком списке элементом *DT* размечается термин, а элементом *DD* – соответствующее ему определение. Списки определений имеют вид:

<DL>

<DT> название термина

<DD> определение термина

...

</DL>.

Заметим, что элементы *DT* можно использовать только как контейнеры для элементов текстового уровня. Но в то же время внутри *DD* можно использовать блочные элементы (исключение составляют заголовки и элементы *address*). Пример списка определений:

<DL>

<DT>Первый термин<dd>Определение первого термина.

<DT>Второй термин<dd>Определение для второго термина.

</DL>.

В результате должна получиться следующая разметка:

Первый термин

Определение первого термина.

Второй термин

Определение для второго термина.

PRE используется, когда необходимо включить в текст документа уже отформатированный текст. Для данного элемента необходимо указывать начальный и конечный теги. Внутри такого элемента текст печатается шрифтом фиксированной ширины и при этом сохраняется разметка оригинала (пробелы и символы конца строк).

Пример использования элемента *PRE*:

<PRE>

Higher still and higher

From the earth thou springest

Like a cloud of fire;

The blue deep thou wingest,

And singing still dost soar, and soaring ever singest.

</PRE>

будет воспроизведено как:

Higher still and higher

From the earth thou springest

Like a cloud of fire;

The blue deep thou wingest,

And singing still dost soar, and soaring ever singest.

DIV – деление документа на отдельные блоки. Для данного элемента необходимо указывать как начальный, так и конечный теги.

CENTER – выравнивание текста. Для этого элемента необходимо указывать начальный и конечный теги. Используется для центрирования стоящего между ними текста.

FORM – заполняемые формы. Необходимо указывать и начальный, и конечный теги. Используется для создания заполняемой формы, которая будет обрабатываться на HTTP-сервере.

HR – горизонтальные линейки. Не является контейнером, так что использовать конечный тег нельзя.

TABLE – таблица, может быть вложенной.

2.3.3. Таблицы

В общем случае таблицы имеют следующий вид:

<TABLE BORDER=3 CELLSPACING=2 CELLPADDING=2 WIDTH=«80%»>

<CAPTION> ... заголовок таблицы ... </CAPTION>

<TR><TD> первая клетка таблицы <TD> вторая клетка

<TR> ...

</TABLE>

Для элемента *TABLE* всегда необходимо указывать как начальный, так и конечный теги. При этом разрешается использовать следующие атрибуты:

align – данный атрибут принимает одно из следующих значений: *LEFT*, *CENTER* или *RIGHT* (используемый при этом регистр значения не имеет). Ука-

зывает для текущей таблицы, каким образом при разметке осуществляется ее горизонтальное выравнивание;

width – при отсутствии данного атрибута ширина таблицы определяется автоматически;

border – этот атрибут позволяет задавать для таблицы ширину внешней рамки в пикселах (например, *BORDER*=4). Данному атрибуту может быть также присвоено значение «нуль», чтобы полностью отказаться от внешней рамки;

cellspacing – в языке *HTML* каждая ячейка имеет собственную границу, отделенную промежутком от границ соседних ячеек;

cellpadding – атрибут устанавливает для каждой ячейки в таблице расстояние в пикселах между рамкой ячейки и содержащимся в ней материалом.

Элемент *CAPTION* может иметь только один атрибут – *ALIGN*, который может принимать два значения: *ALIGN=TOP* или *ALIGN=BOTTOM*.

Для *TR* – элемента, начинающего новый ряд таблицы, – необходимо указывать начальный тег, но всегда можно опустить конечный тег. Элемент *TR* выступает в роли контейнера для ячеек таблицы и может иметь два атрибута:

align – устанавливает стиль горизонтального выравнивания для содержимого ячейки, который будет использоваться по умолчанию. Атрибут может принимать одно из следующих значений (независимо от используемого регистра): *LEFT*, *CENTER* или *RIGHT* – и выполняет ту же самую роль, что и атрибут *ALIGN* при разметке параграфов;

valign – данный атрибут может использоваться при выборе правила, согласно которому – если нет других указаний – будет осуществляться вертикальное выравнивание во всех ячейках данной строки. Атрибут может принимать одно из следующих значений (независимо от используемого регистра): *TOP*, *BOTTOM* или *MIDDLE*. При этом содержимое ячейки будет выравниваться по ее верхнему или нижнему краю, либо посередине соответственно.

2.4. Элементы текстового уровня

2.4.1. Элементы, задающие шрифт, используемый при разметке документа

Для всех элементов данного класса требуется указывать как начальный, так и конечный тег, например:

<B>жирный текст</B>.

<B>жирный и <I>наклонный текст</I></B>.

Набор шрифтов, оговариваемых в спецификации *HTML*, ограничивается разметкой речи, либо должен применяться в документе для осуществления различных типов выделения:

TT – телетайпный текст или текст фиксированной ширины;

I – стиль с наклонным шрифтом;

B – стиль с жирным шрифтом;

U – стиль с подчеркиванием текста;

STRIKE – стиль с перечеркиванием текста;

BIG – печать текста шрифтом увеличенного размера;

SMALL – печать текста шрифтом уменьшенного размера;

SUB – печать текста со сдвигом вниз (нижний индекс);

SUP – печать текста со сдвигом вверх (верхний индекс).

2.4.2. Элементы разметки фраз

В спецификации *HTML*-документа применяют следующие элементы разметки фраз:

EM – основной элемент, используемый в HTML для выделения текстов, обычно реализуется с привлечением наклонного шрифта;

STRONG – усиленное выделение, обычно реализуемое посредством жирного шрифта;

DFN – дает пример для обсуждаемого термина (понятия);

CODE – используется для выделения программного кода;

SAMP – используется при предоставлении в документе распечаток программ, программных сценариев и т. д.;

KBD – используется для выделения текста, который пользователь должен набрать на клавиатуре;

VAR – используется для обозначения переменных, либо аргументов, передаваемых с командами;

CITE – используется для обозначения цитат или ссылок на другие источники.

3. FORM (форма): заполняемая форма

3.1. Атрибуты формы

Форма используется для таких действий пользователя, как регистрация, упорядочение пользователя или формирование запроса. Основной синтаксис формы и ее возможные атрибуты представлены в табл. 8:

<FORM ACTION=«URL»>

содержание формы, включая элементы *INPUT* и, возможно, элементы *TEXTAREA* и *SELECT </FORM>*.

Таблица 8

Возможные атрибуты формы

Имя атрибута	Возможные значения	Смысл атрибута	Примечания
ACTION	*URL*	адрес сервера, который использует форма	сервер *HTTP* или *URL*
METHOD	*GET, POST*	метод передачи данных, полученных от пользователя, на сервер	по умолчанию – *GET*
ENCTYPE	строка	механизм, используемый для кодирования содержимого формы	по умолчанию – приложение */x-www-form-url-кодирование*

Есть некоторые элементы, которые могут появиться только в пределах элемента *FORM*. В частности:

INPUT – текстовое однострокoвое поле, поля пароля, переключатели, радиокнопки, кнопки установки и перезагрузки, скрытые поля, кнопки выгрузки файла, кнопки изображений и т. д.;

SELECT – меню единичного или множественного выбора;

TEXTAREA – многострочное текстовое поле.

3.2. Способы обработки данных

PHP создает следующие массивы при обработке данных, пришедших из формы:

$_*GET* – содержит *GET*-параметры, пришедшие крипту через переменную окружения *QUERY_STRING*. Например, *$_GET('login')*;

$_*POST* – данные формы, пришедшие методом *POST*;

$_*COOKIE* – все *cookies*, которые прислал браузер;

$_*REQUEST* – объединение всех перечисленных выше массивов. Именно эту переменную рекомендуется использовать в скриптах, потому что таким образом мы не «привязываемся» жестко к типу принимаемых данных (*GET* или *POST*);

$_*SERVER* – содержит переменные окружения, переданные сервером.

При отладке сценария можно использовать переменную $*GLOBALS*, например (см. листинг 6):

Листинг 6. Файл отладки.

```
<!– – Выводит все глобальные переменные – –>
<pre>
<? Print_r ($GLOBALS);
</pre>
```

3.3. Передача данных командной строкой

Пусть на сервере в корневом каталоге есть сценарий *PHP* под названием (см. листинг 7, рис. 1). Сценарий распознает два параметра: *login* и *password*:

Листинг 7. Страница с формой *test_forma.php*.

```
<html><body>
<?php
```

echo '<form action=test_pw.php>

*Логин: <input type=text name=«login» value=«root»>
*

*Пароль: <input type=password name=«password» value=«zz»>
*

<input type=submit value=«Нажмите кнопку»>

</form>';

?>

</body></html>

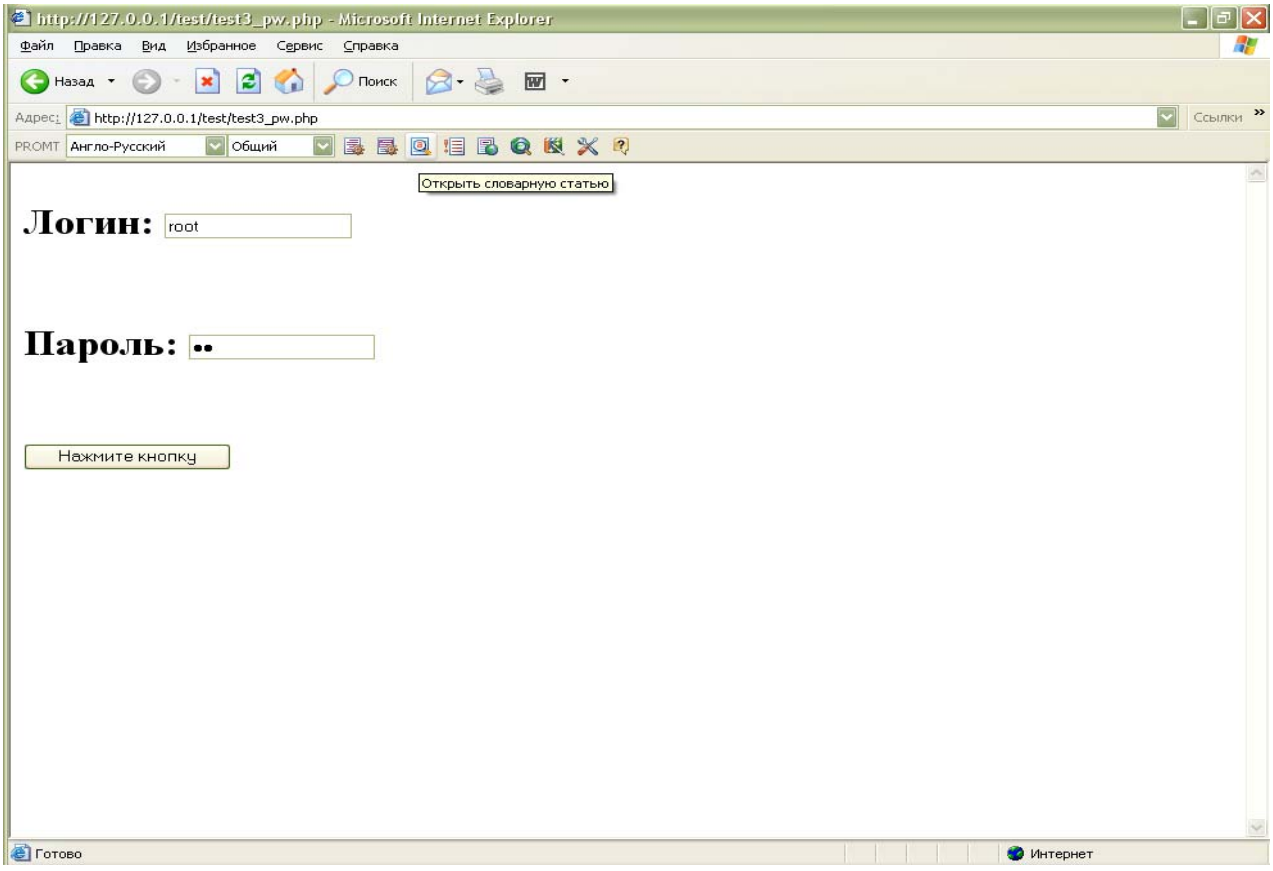

Рис. 1. Результат листинга 7

Если в адресной строке браузера задать:

http://localhost/hello.php?login=root&password=zz,

получается передача параметров в командной строке.

Проанализировать переменную окружения $*QUERY_STRING*, которая доступна в *PHP*, можно под именем $*_SERVER[QUERY_STRING]* (см. листинг 8, рис. 2).

Листинг 8. Файл проверки параметров командной строки test_pw.php.

<html><body>

```
<?php
    Echo '<b><h1>Данные из командной строки:,<br><br>';
    echo $_SERVER[QUERY_STRING],'</h1></b><br>';
?>
</body></html>
```

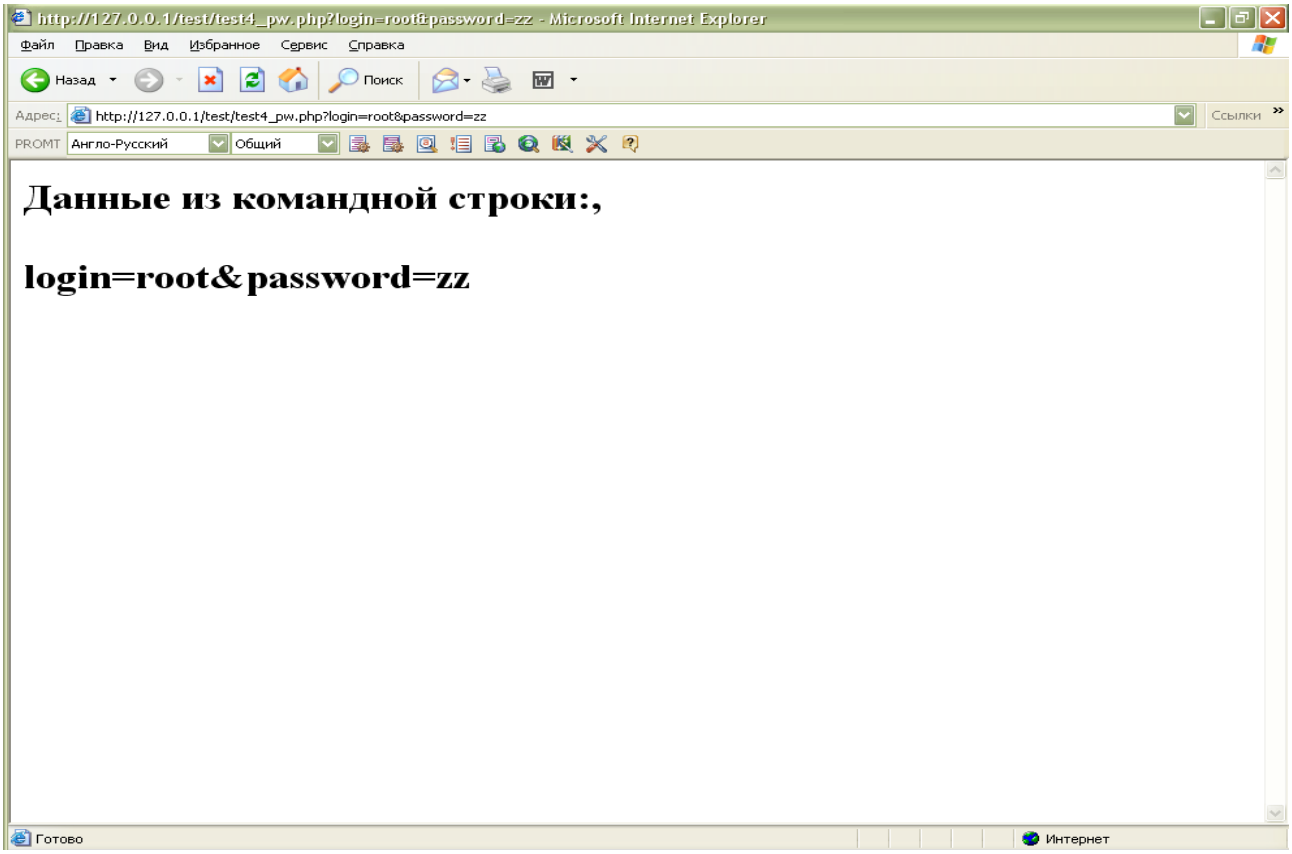

Рис. 2. Результат листинга 8

3.4. Трансляция полей формы

Все данные из полей формы *PHP* помещает в глобальный массив $_*REQUEST*. Значение поля *login* будет храниться в $_*REQUEST*[‘*login’*], а значение поля *password* – в $_*REQUEST*[‘*password*] (см. листинг 9).

Листинг 9. Файл *test_pw.php*. Использование данных форм.

```
<html><body>
<?php
    if ($_REQUEST['login']=='root' && $_REQUEST['password']=='zz') {
        echo «Доступ открыт для пользователя –»;
        echo $_REQUEST['login'];
```

```
    } else {
        echo «Доступ закрыт»;
    }
?>
</body></html>
```

3.5. Трансляция переменных окружения

В переменные преобразуются переменные окружения (см. листинг 10).

Листинг 10. Вывод *IP*-адреса и браузера пользователя.

```
<html><body>
<?PHP
    echo «Ваш IP-адрес:», $_SERVER['REMOTE_ADDR'], «<br>»;
    echo «Ваш браузер:», $_SERVER['HTTP_USER_AGENT'];
?>
</body></html>
```

3.6. Трансляция cookies

Все *cookies*, пришедшие скрипту, передаются в массив (см. листинг 11).

Листинг 11. Демонстрация работы с $_*COOKIES*.

```
<?php
// Вначале счетчик равен нулю
$count=0;
// Если в cookies что-то есть, берем счетчик оттуда.
If (isset ($_COOKIE['count'])){
    $count=($_COOKIE[count]);
    //echo «count==», $count;
    $count++;
    setcookie(«count», $count,0x7FFFFFFF, '/');
}
// выводит счетчик
Echo «КУКА=», $count;
?>
```

3.7. Обработка списков

В *PHP* предусмотрена возможность задавать имена полям формы в виде «массива с индексами» (см. листинг 12, рис. 3):

Листинг 12. Обработка списков.

```
<html>
   <head>
      <title>Пример</title>
   </head>
   <body>
      <?php
      echo '<FORM ACTION=«test6_sel.php» METHOD=POST>
                  Список городов:
                  <SELECT NAME=gorod[]>
                  <OPTION>Екатеринбург
                  <OPTION>Москва
                  <OPTION>Санкт-Петербург
                  <OPTION>Киев
                  </SELECT>
                  <P>
                  <INPUT TYPE=SUBMIT VALUE=«Вывод»>
            </FORM>';
       ?>
        </body>
</html>
```

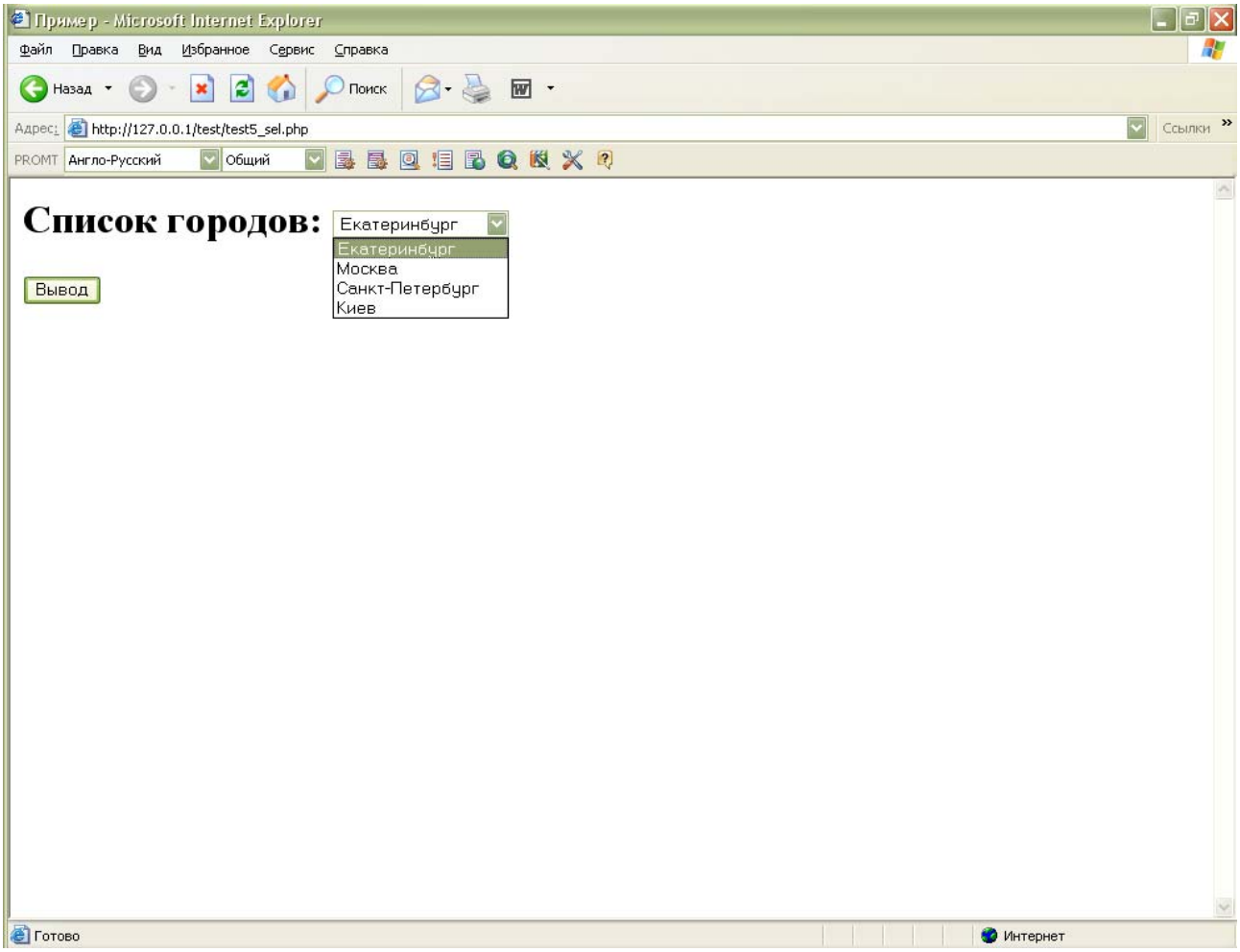

Рис. 3. Результат листинга 12

В результате получим массив в $_*REQUEST* массив массивов (двумерный), доступ к элементам которого можно получить так (см. листинг 13):

Листинг 13. Файл *test6_sel.php.*

```
<html>
   <head>
      <title>Пример</title>
   </head>
   <body>
      <?php
            $gor=$_REQUEST['gorod'][0];
            Echo $gor;  ?>
    </body>
</html
```

Пример ассоциативного массива в форме в файле *test7_sel.php* (см. листинг 14, рис. 4):

Листинг 14. Файл *test7_sel.php*.

```
<html>
  <head>
     <title>Пример</title>
  </head>
  <body>
    <?php
          echo '<FORM ACTION=«test8_sel.php» METHOD=POST>
          Имя <input type=text name=Data[name]><br>
          Адрес <input type=text name=Data[address]><br>
          Город
          <input type=radio name=Data[city] value
=Moscow>Москва<br>
          <input type=radio name=Data[city] value
=Peter>Петербург<br>
          <input type=radio name= Data[city] value=kiev>Киев<br>
          <INPUT TYPE=SUBMIT VALUE=«Вывод»>
          </FORM>';
      ?>
  </body>
</html>
```

В сценарии к отдельным элементам формы можно обратиться при помощи указания ключа массива: например, $*REQUEST[‘Data’][‘city’]* (см. листинг 15):

Листинг 15. Файл *test8_sel.php*.

```
<html>
  <head>
    <title>Пример</title>
```

```
    </head>
    <body>
      <?php
                    $gor=$_REQUEST[Data][city];
                    Echo $gor;
                    echo '<br>', «Имя – »;
                    echo $_REQUEST[Data][name];
       ?>
    </body>
</html>
```

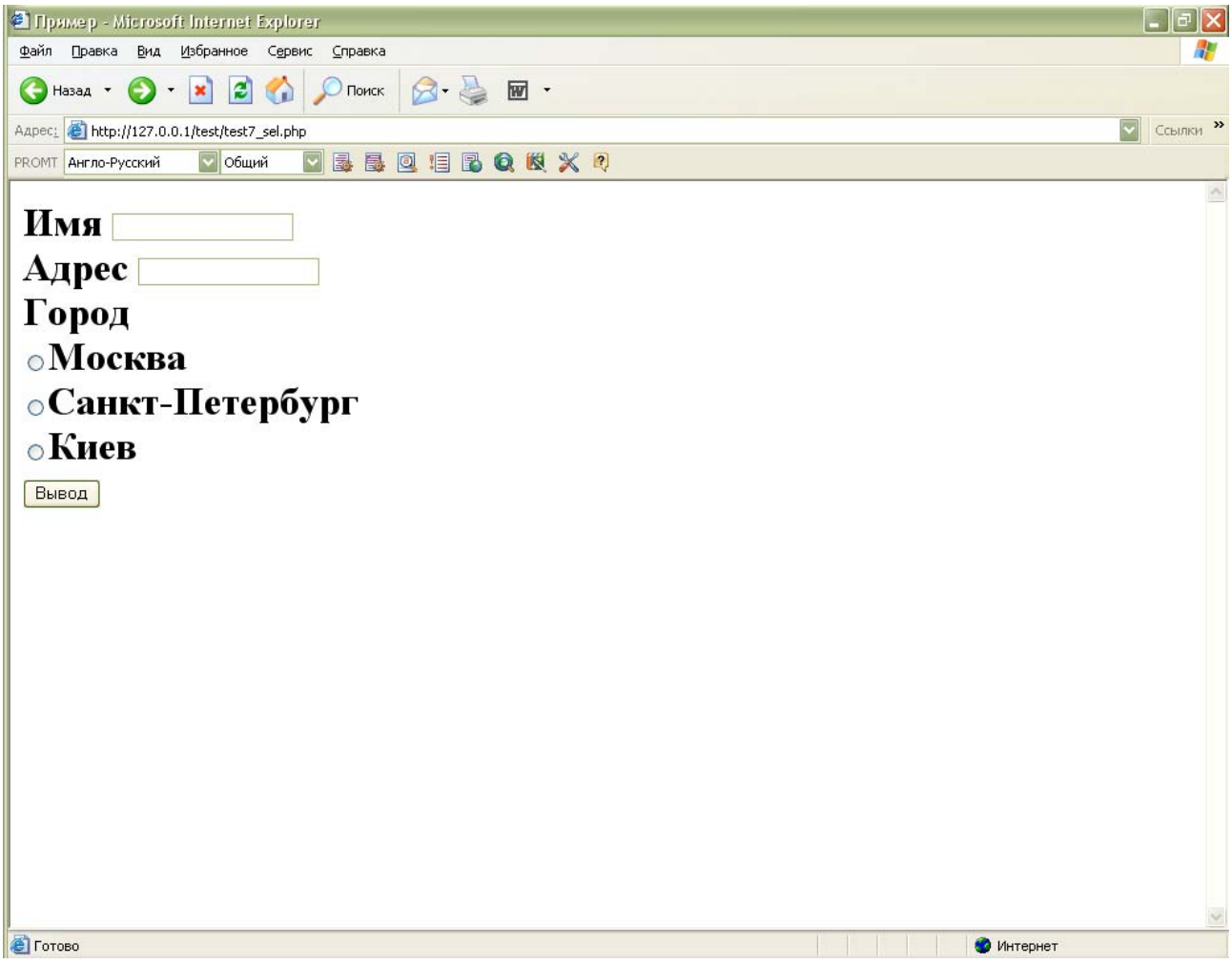

Рис. 4. Результат листинга 14.

3.8. Передача параметров методами GET и POST

При использовании метода *GET* все параметры передаются единой строкой в переменной *QUERY_STRING*. Данные поступят *URL*-кодированными (см. листинг 17).

При использовании метода *POST* параметры передаются сценарию через стандартный поток ввода (см. листинги 12, 14).

3.9. Особенности флажков checkbox

Можно воспользоваться одноименным скрытым полем (hidden) со значением, равным, например, нулю, поместив его перед нужным флажком (см. листинг 16, рис. 5).

Листинг 16. Гарантированный прием значений от флажков в файле *test12_hid.php*.

```
<html>
<head>
   <title>Пример</title>
</head>
<body>
      <?php
         foreach (@$_REQUEST[known] as $k=>$v) {
            If ($v) {echo «Вы знаете язык $k!». «<br>»;}
            else {echo «Вы не знаете язык $k!». «<br>»;}
         }
      echo '<form action=test12_hid.php method=POST>
      какие языки программирования вы знаете?<br>
      <input type=hidden name=«known[PHP]» value=«0»>
      <input type=checkbox name=«known[PHP]» value=«1»>PHP<br>
      <input type=hidden name=«known[Perl]» value=«0»>
      <input type=checkbox name=«known[Perl]» value=«1»>Perl<br>
      <input type=submit name=«doGo» value=«doGo»>
      </form>';
```

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

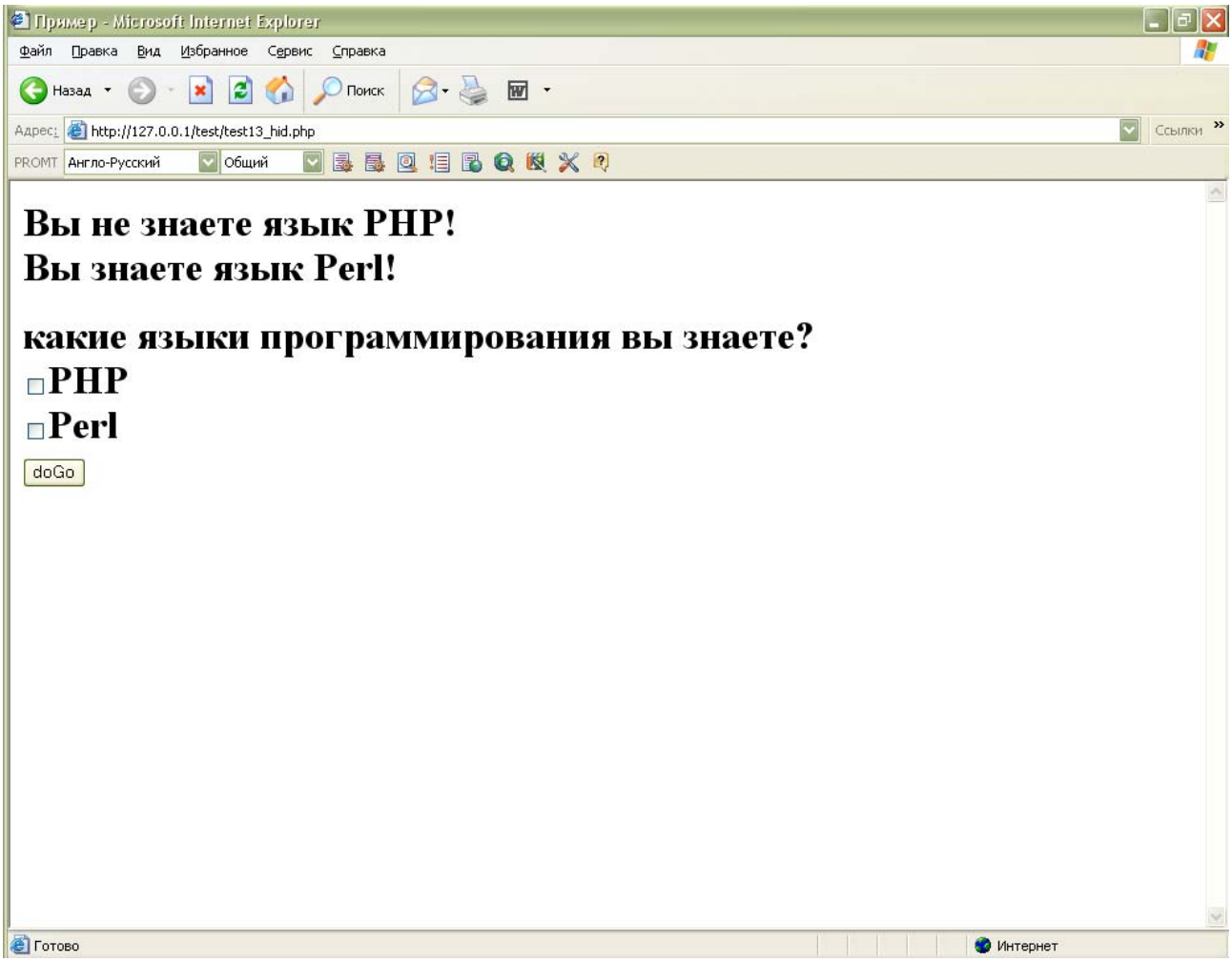

Рис. 5. Результат листинга 16

Если пользователь не выберет никакой флажок, браузер отправит сценарию пару *known*[язык]=0 и в массиве $_*REQUEST*[‘*known’*] создастся соответствующий элемент. Если пользователь выберет флажок, то последует пара *know*[язык]=1.

4. Пример формы

От описания базовых компонентов форм мы переходим к практическому примеру – построению формы для обработки данных, введенных пользователем. Допустим, вы хотите создать форму выбора периода времени (рис. 6).

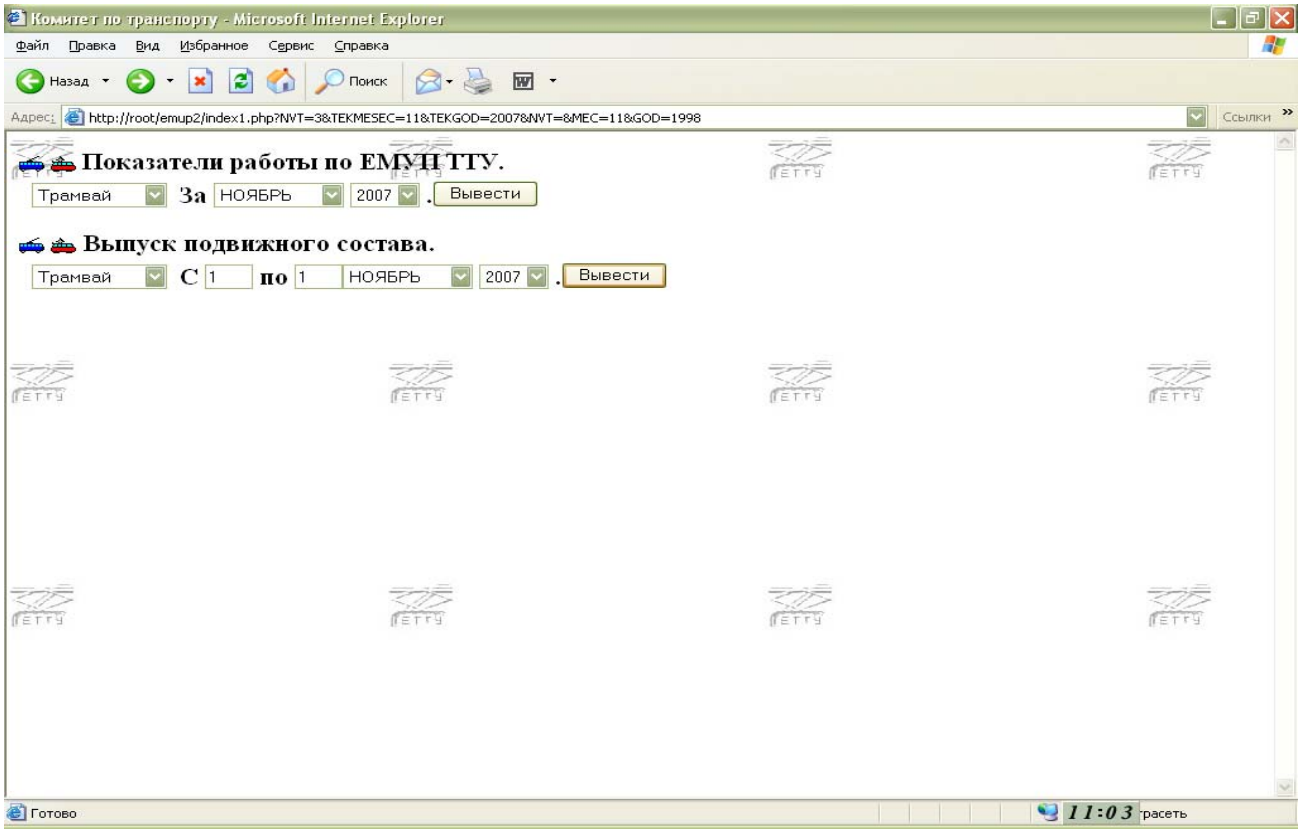

Рис. 6. Результат листинга 17

Программа *«index1.php»* приведена в листинге 17.

Листинг 17. Программа *«index1.php»*

<html>

<head>

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

<title>Комитет по транспорту</title>

</head>

<body bgcolor=«#ffffff» BACKGROUND=«images/fon.jpg»>

<?Php

$DEN=date('d')+0; – объявление значений переменных

$MEC=date('m');

$GOD=date('Y');

Echo '<FORM METHOD=«get» ACTION=«Pokrab/pokrabttu_bas.php» target=«_top»>';

Echo '<IMG SRC=«/image/troll.gif» BORDER=0 ALIGN=CENTER>';

Echo « »;

Echo «<B>Показатели работы по ЕМУП ТТУ.</B>»;

*Echo «
  »;*

include («../vvt2.php»); – список видов транспорта

Echo «<B> За »;

// месяцы года

Echo '<SELECT NAME=«TEKMESEC»>';

Echo '<OPTION';if ($MEC==1) {Echo 'SELECTED';} Echo ' VALUE=«01»> ЯНВАРЬ</OPTION>';

Echo '<OPTION';if ($MEC==2) {Echo 'SELECTED';} Echo ' VALUE=«02»> ФЕВРАЛЬ</OPTION>';

Echo '<OPTION';if ($MEC==3) {Echo 'SELECTED';} Echo ' VALUE=«03»> МАРТ</OPTION>';

Echo '<OPTION';if ($MEC==4) {Echo 'SELECTED';} Echo ' VALUE=«04»> АПРЕЛЬ</OPTION>';

Echo '<OPTION';if ($MEC==5) {Echo 'SELECTED';} Echo ' VALUE=«05»> МАЙ</OPTION>';

Echo '<OPTION';if ($MEC==6) {Echo 'SELECTED';} Echo ' VALUE=«06»> ИЮНЬ</OPTION>';

Echo '<OPTION';if ($MEC==7) {Echo 'SELECTED';} Echo ' VALUE=«07»> ИЮЛЬ</OPTION>';

Echo '<OPTION';if ($MEC==8) {Echo 'SELECTED';} Echo ' VALUE=«08»> АВГУСТ</OPTION>';

Echo '<OPTION';if ($MEC==9) {Echo 'SELECTED';} Echo ' VALUE=«09»> СЕНТЯБРЬ</OPTION>';

Echo '<OPTION';if ($MEC==10) {Echo 'SELECTED';} Echo ' VALUE=«10»> ОКТЯБРЬ</OPTION>';

Echo '<OPTION';if ($MEC==11) {Echo 'SELECTED';} Echo ' VALUE=«11»> НОЯБРЬ</OPTION>';

```
Echo '<OPTION';if ($MEC==12) {Echo 'SELECTED';} Echo ' VALUE=«12»> ДЕКАБРЬ</OPTION>';
Echo '</SELECT> ';
$ii=1999; // годы
Echo '<SELECT NAME=«TEKGOD»>';
$i=$GOD;
while ($i >= $ii) {
Echo '<OPTION ';if ($GOD==$i) {Echo 'SELECTED ';}
Echo 'VALUE=«'.$i.'»>'.$i.'</OPTION>';
$i – –;
}
Echo '</SELECT> ';
Echo '</B><INPUT TYPE=«SUBMIT» VALUE=«Вывести»>';
Echo «</FORM>»;
?>.
```

5. Задание к лабораторной работе

5.1. План выполнения лабораторной работы

Открыть пакет «*XAMPP Control Panel*». Ярлык на рабочем столе или в папке *C:/xampp/xampp-control.exe.* Запустить локальный Web-сервер *Apache* для *Windows* и *MySQL*.

Используя пакет «*NotePad++*», создайте первый сайт с расширением **.php*. Файл сохраните в папке, например *gruppa1* директории *C:/xampp/htdocs/*. Запустите его из адресной строки браузера, например:

по доменному адресу

«http://localhost/gruppa1/primer1.php »

или по IP-адресу

«http://127.0.0.1/gruppa1/primer1.php».

При отладке скрипта в браузере для обновления сайта используйте клавишу F5 или кнопку «Обновить».

Создайте второй сайт из предложенного задания. Все данные из полей формы *PHP* помещайте в глобальный массив $*_REQUEST*.

Требования к отчету: отчет по лабораторной работе должен содержать:

- текст *PHP*-кода программы;
- результат выполнения программы – рисунки двух получившихся сайтов;
- выводы о проделанной работе.

5.2. Варианты заданий

Задание 1

Напишите *PHP*-программу создания формы «Ввод простого текста и его вывод».

На форме расположить:

1) поле ввода (используйте тег <*input*>, параметр *type=text*);

2) кнопку отправки формы *submit* (используйте тег <*input*>, параметр *type=submit* и название «Вывести»).

В поле ввода набивается фамилия, имя и отчество студента. На кнопку «Вывести» наложите еще одну *PHP*-программу, с помощью которой в новой форме выводится:

1) в первой строке фамилия студента «жирным» шрифтом;

2) во второй строке – его имя «курсивным» шрифтом;

3) в третьей строке – фамилия и инициалы студента.

Примечание: функция длины строки *strlen*(*string* $*st*) *int*, где $*st* – последовательность символов. Функция поиска подстроки *strpos*(*string* $*where*, *string* $*what*, *int* $*from*=0) *int*, где в строке $*where*, ищется строка $*what.* В случае успеха возвращает позицию этой подстроки в строке. Функция *substr*(*string* $*str*, *int* $*start* [,*int* $*length*]) string возвращает участок стоки $*str*, начиная с позиции $*start* и длиной $*length*.

Задание 2

Напишите *PHP*-программу создания формы «Ввод пароля и вывод проверки – правильно ли веден пароль».

На форме расположить:

1) поле ввода (используйте тег <*input*>, параметр *type=password*);

2) кнопку отправки формы *submit* (используйте тег <*input*>, параметр *type=submit* и название «ОК».

На кнопку «Вывести» наложите еще одну *PHP*-программу, с помощью которой в новой форме выводится сообщение о проверке правильности введенного пароля. На экране должно выйти сообщение «Пароль введен верно» или «Пароль введен неверно». Сообщение должно выводиться жирным шрифтом.

Задание 3

Напишите *PHP*-программу создания формы «Выбор параметра независимого переключателя и вывод соответствующего значения».

На форме расположить:

1) независимый переключатель (или флажок) (используйте тег <*input*>, параметр *type=checkbox* поле ввода);

2) кнопку отправки формы *submit* (используйте тег <*input*>, параметр *type=submit* и название «Вывести»).

Значения переключателя – «текущее время», «текущая дата», «текущий день», «текущий месяц», «текущий год». На кнопку «Вывести» наложите еще одну *PHP*-программу, с помощью которой в новой форме выведите выбранное значение. Этот параметр определяется по выбору переключателя.

Примечание: функция текущей даты *date*(«*d.m.y*»), функция времени *date*(«*h.i.s*»).

Задание 4

Напишите *PHP*-программу создания формы «Выбор параметра независимого переключателя и вывод соответствующего значения цвета».

На форме расположить:

1) независимый переключатель (или радиокнопки) (используйте тег <*input*>, параметр *type=radio*);

2) кнопку отправки формы *submit* (используйте тег <*input*>, параметр *type=submit* и название «Вывести»).

Значения переключателя – «красный цвет», «желтый цвет», «синий цвет», «зеленый цвет». На кнопку «Вывести» наложите еще одну *PHP*-программу, с помощью которой в новой форме выводится текст названия цвета в соответствующей цветовой гамме. Название цвета выведите столбиком посередине формы.

Примечание: используйте тег *FONT,* который имеет следующий синтаксис *<FONT SIZE=n>текст</FONT>* или *<FONT COLOR=цветовая спецификация>текст</FONT>*. Для переноса строки используется тег <*br*>.

Задание 5

Напишите *PHP*-программу создания формы «Вставка картинки и вывод даты или времени».

На форме расположить два рисунка для отправки формы (*image*) (используйте тег <*input*>, параметр *type=image*).

На первый рисунок наложите программу вывода текущего времени, на второй рисунок программу вывода текущей даты. Перед значением текущей даты (например, 01.05.2008 г.) напишите слово «Сегодня 1 мая 2008 года». Перед текущим временем напишите «Текущее время» в верхнем регистре.

Примечание: функция *strtoupper*(*string* $*st*) *string* переводит строку в верхний регистр.

Задание 6

Напишите *PHP*-программу создания формы «Создание формы выбора даты и вывод этой даты».

На форме расположить:

1) поле ввода значения числа дня (используйте тег <*input*>, параметр *type=text*);

2) раскрывающийся список *select* значения названия месяца (используйте атрибуты *<name>, <option> и value*);

3) кнопку отправки формы *submit* (используйте тег *<input>*, параметр *type=submit* и название «Вывести»).

На кнопку «Вывести» наложите еще одну *PHP*-программу, с помощью которой в новой форме выводится выбранная дата (например, 1 мая 2008 года).

Задание 7

Напишите *PHP*-программу создания формы «Созданные формы выбора натурального числа и вывод квадрата или куба этого числа».

На форме расположить:

1) раскрывающийся список *select* значения названия натурального числа от 1 до 10 (используйте атрибуты *<name>, <option> и value*);

2) независимый переключатель (или радиокнопки) (используйте тег *<input>*, параметр *type=radio*);

3) кнопку отправки формы *submit* (используйте тег *<input>*, параметр *type=submit* и название «Вывести»).

На радиокнопку наложить два параметра: «квадрат», «куб». На кнопку «Вывести» наложите еще одну *PHP*-программу, с помощью которой в новой форме выводится квадрат или куб выбранного натурального числа соответственно выбранному параметру радиокнопки.

Примечание: для возведения в квадрат используйте функцию *pow(float* $*base, float* $*exp) float* возвращает $*base* в степень $*expl*. Для возведения в куб – используйте цикл *for…end*.

Задание 8

Напишите *PHP*-программу создания формы «Ввод интервала простых чисел и вывод случайного числа из этого интервала».

На форме расположить:

1) два поля ввода (используйте тег *<input>*, параметр *type=text*);

2) кнопку отправки формы *submit* (используйте тег *<input>*, параметр *type=submit* и название «Вывести»).

В первом поле ввода набивается начальное значение интервала, во втором поле ввода – конечное значение интервала. На кнопку «Вывести» наложите еще одну *PHP*-программу, с помощью которой в новой форме выводится случайное число.

Примечание: функция возвращения случайных чисел *mt_rand(int* $*min=0, int* $*max=RAND_MAX) int.*

Задание 9

Напишите *PHP*-программу создания формы «Создание массива фамилий и демонстрация работы со списками».

На форме расположить кнопку отправки формы *submit* (используйте тег <*input*>, параметр *type=submit* и название «Вывести»).

На кнопку «Вывести» наложите еще одну *PHP*-программу, с помощью которой в новой форме выводится список фамилий. Список пронумеруйте.

Примечание: для вывода используйте цикл *for*. Количество элементов в массиве определяется функцией *count( )*. Элементы массива пропишите в *PHP*-программе.

Задание 10

Напишите *PHP*-программу создания формы «Ввод интервала простых чисел и вывод списка заданных значений».

На форме расположить:

1) два поля ввода (используйте тег <*input*>, параметр *type=text*);

2) кнопку отправки формы *submit* (используйте тег <*input*>, параметр *type=submit* и название «Вывести»).

В первом поле ввода набивается начальное значение интервала, во втором поле ввода – конечное значение интервала. На кнопку «Вывести» наложите еще одну *PHP*-программу, с помощью которой в новой форме выводится столбиком заданный интервал чисел, начиная со второго значения и заканчивая предпоследним значением.

Примечание: для вывода чисел напишите функцию вывода.

Задание 11

Напишите *PHP*-программу создания формы «Ввод значений любых чисел и вывод значения».

На форме расположить:

1) два поля ввода (используйте тег <*input*>, параметр *type=text*);

2) кнопку отправки формы *submit* (используйте тег <*input*>, параметр *type=submit* и название «Вывести»).

В первом и во втором поле ввода вводится дробное значение. На кнопку «Вывести» наложите еще одну *PHP*-программу, с помощью которой в новой форме выводится результат перемножения этих чисел. А также выведите: ближайшее целое число, наименьшее целое число, максимальное целое число и абсолютное значение.

Примечание: функция *abs(mixed* $*numeric) mixed* возвращает модуль числа. Функция *round(double* $*val) double* округляет $*val* до ближайшего целого числа. Функция *ceil(float* $*number) int* возвращает наименьшее целое число, которое было бы не меньше, чем *number*. Функция *float(float* $*number) int* возвращает максимальное целое число, не превосходящее *number*.

Список литературы

1. **Котеров, Д. В.** PHP 5 / Д. В. Котеров, А. Ф. Костарев. – СПб. : БХВ-Петербург, 2007. – 1120 с.

2. **Грофф, Дж.** Энциклопедия SQL. – 3-е изд. (+ CD) / Дж. Грофф, П. Вайнберг. – СПб. : Питер, 2003. – 896 с.

3. **Шкарина, Л.** Язык SQL : учебный курс / Л. Шкарина. – СПб. : Питер, 2001. – 592 с.

4. **Яргер, Р.** MySQL и mSQL. Базы данных для небольших предприятий и Интернета / Р. Яргер, Дж. Риз, Т. Кинг. – СПб. : Символ-Плюс, 2001. – 560 с.

5. **http://www.mysql.com**

Учебное электронное текстовое издание

Составители: Дружинина Надежда Геннадьевна,
Трофимова Ольга Геннадьевна

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЙ С ПОМОЩЬЮ PHP

Редактор *А. В. Ерофеева*
Компьютерный набор *Н. Г. Дружининой, О. Г. Трофимовой*
Подготовлено к публикации *А. В. Ерофеевой*

Рекомендовано РИС ГОУ ВПО УГТУ–УПИ
Разрешен к публикации 07.05.09
Электронный формат – pdf
Объем 2,15 уч.-изд. л.

Издательство ГОУ-ВПО УГТУ–УПИ
620002, Екатеринбург, ул. Мира, 19

Информационный портал
ГОУ ВПО УГТУ–УПИ
http://www.ustu.ru