ЛАБОРАТОРНАЯ РАБОТА №4
ОСНОВЫ ЯЗЫКА ПРОГРАММИРОВАНИЯ PHP

Цель работы: изучить основы языка программирования PHP.

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

4.1 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

4.1.1 Язык программирования PHP

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

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

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

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

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

4.1.2 Переход в PHP

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

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

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

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

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

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

Короткие теги обеспечивают наиболее компактную запись для перехода в PHP:

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

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

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

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

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

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

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

Пример 4.1

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

4.1.3 Комментарии в коде PHP

В PHP существуют два формата комментариев:

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

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

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

вершенно не влияют на общее быстродействие сценария.

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

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

Пример 4.2

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

4.1.4 Типы данных в PHP

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

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

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

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

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

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

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

Таблица 4.1 – Служебные символы в PHP

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

Два следующих объявления дают одинаковый результат:

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

Однако результаты следующих объявлений сильно различаются:

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

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

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

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

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

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

Пример 4.3

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

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

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

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

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

4.1.5 Переменные в PHP

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

Правила задания переменных в PHP:

– имя переменной должно начинаться со знака доллара $;

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

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

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

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

4.1.5.1 Функции определения и задания типа переменных

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

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

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

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

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

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

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

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

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

Пример 4.4

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

4.1.5.2 Присвоение значений. Оператор присваивания

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

Пример 4.5

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

Особенности оператора присваивания:

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

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

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

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

4.1.5.3 Проверка существования переменной

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

Пример 4.6

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

4.1.5.4 Удаление переменных

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

Пример 4.7

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

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

4.1.5.5 Логические переменные и их особенности в PHP

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

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

Пример 4.8

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

```
?>
```

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

Пример 4.9

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

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

4.1.6 Выражения и операции

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

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

В PHP выделяют следующие виды операций:

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

4.1.6.1 Арифметические операции

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

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

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

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

4.1.6.2 Строковые операции

Строковых операций в PHP всего две:

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

Пример 4.10

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

Все остальные действия над строками выполняются стандартными функциями.

4.1.6.3 Операции присваивания

В операциях присваивания правое выражение присваивается переменной слева.

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

Пример 4.11

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

4.1.6.4 Операции сравнения

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

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

Пример 4.12

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

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

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

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

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

4.1.6.5 Логические операции

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

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

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

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

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

4.1.6.6 Приоритет операций

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

Таблица 4.2 – Приоритет арифметических и логических операций

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

4.1.7 Конструкции PHP (операторы выбора, операторы цикла)

4.1.7.1 Условный оператор (if … else)

Формат условного оператора:

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

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

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

Существует альтернативная форма конструкции if … else:

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

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

Пример 4.13

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

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

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

4.1.7.2 Переключатель (switch)

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

Синтаксис переключателя:

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

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

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

Пример 4.14

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

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

Существует альтернативная форма конструкции *switch*:

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

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

4.1.7.3 Циклы

В PHP определены 4 разных оператора цикла:

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

Цикл с предусловием while

Синтаксис цикла с предусловием:

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

Принцип работы цикла с предусловием:

– вычисляется значение логического выражения;

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

Альтернативный синтаксис цикла с предусловием:

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

Цикл с предусловием do ... while

Синтаксис цикла с постусловием:

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

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

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

Итерационный цикл for

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

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

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

Альтернативный синтаксис итерационного цикла:

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

Итерационный цикл foreach

Синтаксис итерационного цикла *foreach*:

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

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

Пример 4.15

Вывод всех переменных окружения

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

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

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

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

4.2 ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

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

а) номер счета;

б) баланс на начало месяца;

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

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

д) допустимый предельный размер кредита.

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

Пример расчета:

Введите номер счета: 100

Введите начальный баланс: 5394.78

Введите общую сумму расходов: 1000.00

Введите общую сумму кредита: 500.00

Введите предельный размер кредита: 5500.00

Счет: 100

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

2 Простые проценты по ссуде рассчитываются по формуле

interest = principal * rate * days / 365

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

Пример расчета:

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

4.3 КОНТРОЛЬНЫЕ ВОПРОСЫ

1 Выберите правильные варианты имен переменных в PHP?

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

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

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

3 Какие типы данных поддерживаются в PHP?

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

4 Какая функция позволяет определить тип переменной?

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

5 Какая функция позволяет изменить тип переменной?

– isset();

– gettype();

– settype();

– define();

– defined().

6 Какая функция позволяет проверить определена ли переменная?

– isset();

– gettype();

– settype();

– define();

– defined().

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

– My favorite food is;

– My favorite food is $f;

– My favorite food is meatloaf;

– meatloaf;

– нет верных ответов.

8 Скрипты PHP:

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

– сервер передает PHP-скрипт, интерпретирующийся браузером клиента;

– браузер передает PHP-скрипт, интерпретирующийся сервером;

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

– нет правильных вариантов.

9 Выберите правильные варианты перехода в PHP:

– <% … %>;

– <? … ?>;

– <?br … ?>;

– <?php … ?>;

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

10 Выберите правильные варианты написания комментариев в PHP:

– #это комментарии;

– &это комментарии;

– /* а это многострочные комментарии */;

– //это комментарии;

–
 это комментарии</br>.

11 Какие утверждения справедливы для переменных?

– имя переменной должно начинаться со знака доллара $;

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

– имя переменной не должно содержать никаких других символов, кроме

символов латинского алфавита и цифр;

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

– имена переменных не чувствительны к регистру;

– имена переменных чувствительны к регистру.

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

– переменная равна true;

– переменная равна 1;

– переменная равна 1. Переменная равна true;

– переменная равна true. Переменная равна 1;

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

13 Какая функция позволяет удалить переменную?

– isset();

– gettype();

– settype();

– unset();

– defined().

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

– 300;

– 100200;

– 100200 300;

– ничего на экран выведено не будет;

– нет верных ответов.

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

– Hello World!;

– Hello;

– World!;

– ничего на экран выведено не будет;

– нет верных ответов.

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

– переменные равны;

– переменные эквивалентны;

– переменные равны. Переменные эквивалентны;

– ничего на экран выведено не будет;

– нет верных ответов.

17 Какие циклы существуют в PHP?

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