**Практическая работа №1**

**Обработка элементов формы на языке PHP**

**Цели работы:**

* повторить основные теги языка HTML;
* понять принцип формирования динамических страниц;
* научиться работать с переменными, условным и циклическим операторами языка PHP.

**Что такое PHP**

Основная область применения PHP – создание приложений (скриптов), которые исполняются на стороне сервера. PHP-скрипты размещаются внутри обычного HTML-файла. При этом сервер, прежде чем отправить файл клиенту, передаёт его обработчику PHP. Получается, что вы пишите не программу которая выводит HTML, а HTML-файл который содержит программу, изменяющую HTML-код в зависимости от различных факторов.

Для того чтобы отделить PHP-скрипт от HTML используются специальные теги. Существует 4 способа отделить PHP:

* <?php … ?>
* <? … ?>
* <scriptlanguage=”php”> … </script>
* <% … %>

Самый предпочтительный это первый способ. Второй и четвертый могут быть недоступны, а третий слишком громоздкий. Внутри скриптов можно использовать комментарии # или // для однострочного комментария и \*/ … /\* для многострочного.

**Первая PHP программа**

Сразу же рассмотрим пример программы на PHP:

*<html>*

*<head>*

*<title>Пример</title>*

*</head>*

*<body>*

*<?php*

*echo"<p>Привет, я – скриптPHP!</p>"*

*?>*

*</body>*

*</html>*

Это простой HTML-файл, в который встроен с помощью специального тега код, написанный на PHP. PHP-скрипт в отличие от скрипта JavaScript отличается тем, что второй обрабатывается на клиентской машине, а первый на сервере и клиенту пересылается только измененная HTML страница. Оператор echo вставляет в html-код строку *<p>Привет, я – скрипт PHP!</p>* .

**Переменные и типы данных**

В PHP переменные обозначаются знаком «$» перед именем переменной. Например, $var, причем следует помнить, что имена переменных в PHP чувствительны к регистру. Специально описывать тип переменной ненужно. PHP поддерживает 8 простых типов данных.

4 скалярных:

* boolean (логический) который может принимать значения true и false
* integer (целый) целое знаковое число в восьмеричной, десятичной или шестнадцатеричной системе счисления, для кодирования которого выделяется 32 бита.
* float (с плавающей точкой) вещественное знаковое число с двойной точностью.
* string (строковый) набор символов произвольной длины

2 смешанных:

* Array (массив) массив содержащий элементы любого типа и числовые или строковые индексы.
* Object (объект) экземпляр класса.

2 специальных:

* Resource (ресурс) переменная данного типа содержит ссылку на внешний ресур например базу данных
* Null – переменная не имеет значение. Существует только одно значение типа null – регистронезависимое ключевое слово null.

**Операторы и операции**

Существует множество операций, которые можно производить с выражениями. Рассмотрим основные.

|  |  |
| --- | --- |
| **Арифметические операции** | **Операции присваивания** |
| + - сложение | = - присваивание |
| - - вычитание | += - сложение с присваиванием |
| \* - умножение | .= - конкатенация с присваиванием |
| / - деление | **Операции сравнения** |
| % - остаток от деления | == - равенство |
| **Строковые операции** | === - эквивалентность |
| . – конкатенация (сложение строк) | != (<>) – неравенство |
| **Логические операции** | !== – неэквивалентность |
| and(&&) – конъюнкция (логическое и) | >– больше |
| or (||) – дизъюнкция (логическое или) | <– меньше |
| xor– исключающее или | <= – меньше либо равно |
| ! – инверсия (логическое не) | >= – больше либо равно |

**Условный оператор IF**

Это один из самых важных операторов многих языков, включая PHP. Он позволяет выполнять фрагменты кода в зависимости от условия. Структуру оператора if можно представить следующим образом:

*If (логическое выражение) оператор1; elseif (логическое выражение2)оператор2;else оператор3*

Здесь *логическое выражение* – любое правильное PHP-выражение. В процессе обработки скрипта выражение преобразуется к логическому типу. Если *логическое выражение* истинно, то выполняется *оператор1* иначе, если *логическое выражение2* истинно, то выполняется *оператор2*, то выполняется *оператор3*.

**Циклический оператор for**

Структура:

*for (выражение1; выражение2; выражение3) {блок\_выполнения}*

Здесь *выражение1* вычисляется один раз, *выражение2* – в начале каждой итерации и если оно истинно, то цикл продолжается, после выполнения *блока\_выполнения* вычисляется *выражение3*. Любое выражение может быть пустым. Если *выражение2* пустое, то оно считается истинным, то есть цикл можно будет остановить только с помощью оператора *break*.

**Обработка элементов формы**

Зачастую бывают такие ситуации, когда от пользователя должны быть получены некоторые данные, необходимые для решения поставленной задачи. Кроме того, существует такое свойство алгоритма, как массовость – если с одними исходными данными был получен верный результат, то и с другими исходными данными, приемлемыми в данной задаче, также должен быть получен верный результат. Для реализации этих целей возможно использование форм HTML-страниц. Для создания форм имеется специальный тег <form>. У этого тега имеются такие атрибуты как action и method. Первый задаёт имя программы, которая будет обрабатывать данные полученные с формы, второй – определяет метод запроса. Отправка данных формы происходит по нажатию кнопки типа submit.

**Метод GET**

При отправке данных с помощью метода GET содержимое формы добавляется к URL после знака вопроса в виде пар имя=значения, объединенных с помощью знака &:

action?name1=value1&name2=value2.

Здесь action – это URL-адрес программы, которая должна обрабатывать форму. Имена name1, name2 имена элементов формы, а value1, value2 – значения этих элементов. В принципе создавать HTML-форму необязательно, достаточно добавить к адресу нужные переменные и их значения, поэтому данный метод имеет существенный недостаток – любой может поделать значения параметров. С другой стороны этот недостаток позволяет достаточно удобно отлаживать скрипты.

**Пример:**

*<HTML>*

*<BODY>*

*<FORM METHOD=get action=1.HTML>*

*name1 <input type=text name="name1"><br><br>*

*name2 <input type=text name="name2"><br><br>*

*name3 <input type=text name="name3"><br><br>*

*<INPUT type=submit value="отправить"><br>*

*</form>*

*</BODY>*

*</HTML>*

**Метод POST**

При использовании этого метода содержимое формы кодируется точно так же, как и для метода GET, но вместо добавления строки к URL содержимое запроса посылается блоком данных как часть операции POST. Этот метод рекомендуется для передачи больших по объему блоков данных.

Информация, введенная пользователем и отправленная серверу с помощью метода POST, подается на стандартный ввод программе. Передать данные методом POST можно только с помощью HTML-формы, поскольку данные передаются в теле запроса, а не в заголовке. При использовании POST пользователь не видит передаваемые серверу данные.

При отправке данных на сервер любым методом передаются не только сами данные, веденные пользователем, но и ряд переменных, называемых переменными окружения, характеризующих клиента, историю его работы, пути к файлам и т.п. вот некоторые из них:

REMOTE\_ADDR – IP-адрес хоста, отправляющего запрос;

REMOTE\_HOST – имя компьютера, с которого отправлен запрос;

HTTP\_REFERER – адрес страницы, ссылающейся на текущий скрипт;

REQUEST\_METHOD – метод, который был использован при отправке запроса;

QUERY\_STRING – строка, находящаяся в URL после знака вопроса;

SCRIPT\_NAME – виртуальный путь к программе, которая должна выполняться;

HTTP\_USER\_AGENT – информация о браузере, который использует клиент.

Обработка запросов

После того как данные с формы были отправлены программе, их следует обработать соответствующим образом. Если данные отправлялись методом GET, то в программе указанной в атрибуте action создаются соответствующие переменные. В нашем случае это $name1, $name2, $name3. с помощью них можно спокойно получать данные с формы. Если отправлялись данные методом POST, то для получения данных можно воспользоваться массив $\_REQUEST используя ячейки с индексом name1, name2, name3. также этот массив можно использовать при передаче данных методом GET. Чтобы точно определить с помощью какого метода были отправлены данные, возможно воспользоваться массивом $\_POST  и $\_GET или $HTTP\_POST\_VARS и $HTTP\_GET\_VARS.

Для доступа к переменным окружения необходимо воспользоваться функцией getenv(). Например echo getenv(“REQUEST\_METHOD”) выведет на экран использованный метод передачи.

**Пример передачи данных методом POST**

Сначала проверяется, существует ли переменная. Если переменная существует, её значение увеличивается на единицу. Если страница открыта впервые, переменной присваивается значение единицы. В форме необходимо прописать имя страницы с ответом и передачу переменной. Фрагмент кода страницы с заданием:

*<?php*

*if (isset($\_POST['NB']))*

*{$NB=$\_POST['NB']; $NB++;}*

*else*

*{$NB=1;}*

*?>*

*<BR>Билет<?php echo $NB?><BR>*

*<FORM action="a.phtml" method="POST">*

*<INPUT type="Submit" value="Ответы">*

*<INPUT type="hidden" name="NB" value="<?php echo $NB?>">*

*</FORM>*

Фрагмент кода страницы с ответами:

*Билет<?php $NB=$\_POST['NB'];*

*echo $NB?><BR>*

*<FORM action="q.phtml" method="POST">*

*<INPUT type="Submit" value="Следующий">*

*<INPUT type="hidden" name="NB" value="<?php echo $NB?>">*

*</FORM>*

**Установка запрета кэширования страницы**

Чтобы пользователь не смог вернуться на страницу с выданным заданием и ввести ответы, которые он получил на странице с ответами, необходимо добавить http-заголовок Cache-Control. Фрагмент кода страницы с заданием:

*<?php*

*header("Cache-Control: no-store, no-cache, must-revalidate");*

*header("Expires: " . date("r"));*

*?>*

**Работа с функциями time() и date()**

В начале скрипта, работающего с заданием, задаём переменную $TO, которая отвечает за время выполнения билета в секундах. Фрагмент кода страницы, отвечающего за вывод времени выполнения билета на экран, будет выглядеть так:

*Задание 1. Определение вероятности.*

*<?php*

*echo " ("; echo (int)($TO/60); echo " мин";*

*if (($tmp=$TO%60)!=0) echo " ".$tmp." сек";*

*echo ")";*

*?>*

Для фиксации случайных значений задаём переменную $IS, а для фиксации времени начала ответа пользователя на билет – переменную $T:

*$IS=mt\_rand(1000,1000000)+(time()%1000);*

*mt\_srand($IS);*

*$T=time();*

При отправлении данных через форму необходимо добавить эти переменные, используя тип hidden (скрытое поле, не отображаемое на странице):

*<INPUT type="hidden" name="IS" value="<?php echo $IS ?>">*

*<INPUT type="hidden" name="T" value="<?php echo $T ?>">*

*<INPUT type="hidden" name="TO" value="<?php echo $TO ?>">*

На странице с ответами необходимо задать переменные $IS, $T и переменную $time, вычислив разницу между текущим временем и значением переменной $T:

*$IS=$\_POST['IS'];*

*mt\_srand($IS);*

*$T=$\_POST['T'];*

*$time=time()-$T;*

Затем определяем, превышено установленное время ($TO) или нет:

*$TO=$\_POST['TO'];*

*if($time>$TO)echo "<FONT size=4 color=red><b>Превышено контрольное время.<BR></b></FONT>";*

Для вывода текущей даты и текущего времени воспользуемся функцией date():

*<?php print(date( "l dS of F Y h:i:s A" )); ?>*

**Использование файла .htaccess**

Чтобы предотвратить возникновения проблем с кодировкой web-страниц, рекомендуется в папке, в которой содержатся страницы, создать файл .htaccess, в который записать следующие строки, если используете кодировку ANSI (Win-1251):

*DefaultLanguage ru*

*AddDefaultCharset windows-1251*

*php\_valuedefault\_charset "cp1251"*

Соответственно, если используете кодировку UTF-8, необходимо прописать:

*AddDefaultCharset UTF-8*

# Задание

1. Создать учебник по одной из задач ЕГЭ, содержащий варианты заданий, решения и ответы на задания (2 веб-станицы). Пример страницы с ответами и решениями представлен на рисунке 1. Для оформления страниц использовать максимальное количество тегов HTML5, при этом их использование должно быть уместным. В отчёте укажите используемые теги с атрибутами.

2. Создать программу, решающую поставленную задачу. На экран выводятся однотипные задания (не менее 4-х вариантов), содержащие случайные значения (таким образом, чтобы задания не повторялись), формулы, демонстрирующие решение этих заданий (с подставленными значениями), и ответы к этим заданиям.

3. Дополнить задание формой для осуществления ввода ответа на задание. Предусмотреть ввод с использованием метода POST, проверку верности вводимого ответа и вывод IP-адреса компьютера, отправляющего запрос.

Необходимо вывести название вуза, номер задания в соответствии с классификатором ЕГЭ, тему задания, время выполнения, номер билета, предусмотреть вывод справки в отдельном окне, предусмотреть вывод текущего времени. Хотя бы одно число в каждом пункте задания должно генерироваться случайным образом (всего не менее 5 пунктов таким образом, чтобы все задания были различны). После нажатия кнопки ответы предусмотреть переход к следующему билету. Предусмотреть вывод ответа пользователя (синим цветом) и верного ответа в случае, если ответ пользователя неверен (красным цветом), вывод фразы «Ошибочный ответ!» в случае ошибки, вывод фразы «Превышено контрольное время!» в случае превышения заданного лимита времени на решение задачи, вывод текущего времени, вывод решения задачи вместе с формулами и подставленными в формулы значениями из условия задачи. Предусмотреть невозможность кэширования страницы, то есть перехода «назад» к странице с теми же условиями задачи.

Осуществлять проверку верности заполнения полей на уровне , используя HTML5.

Правильность ответов проверять на стороне сервера с помощью PHP, сразу на странице ответов.

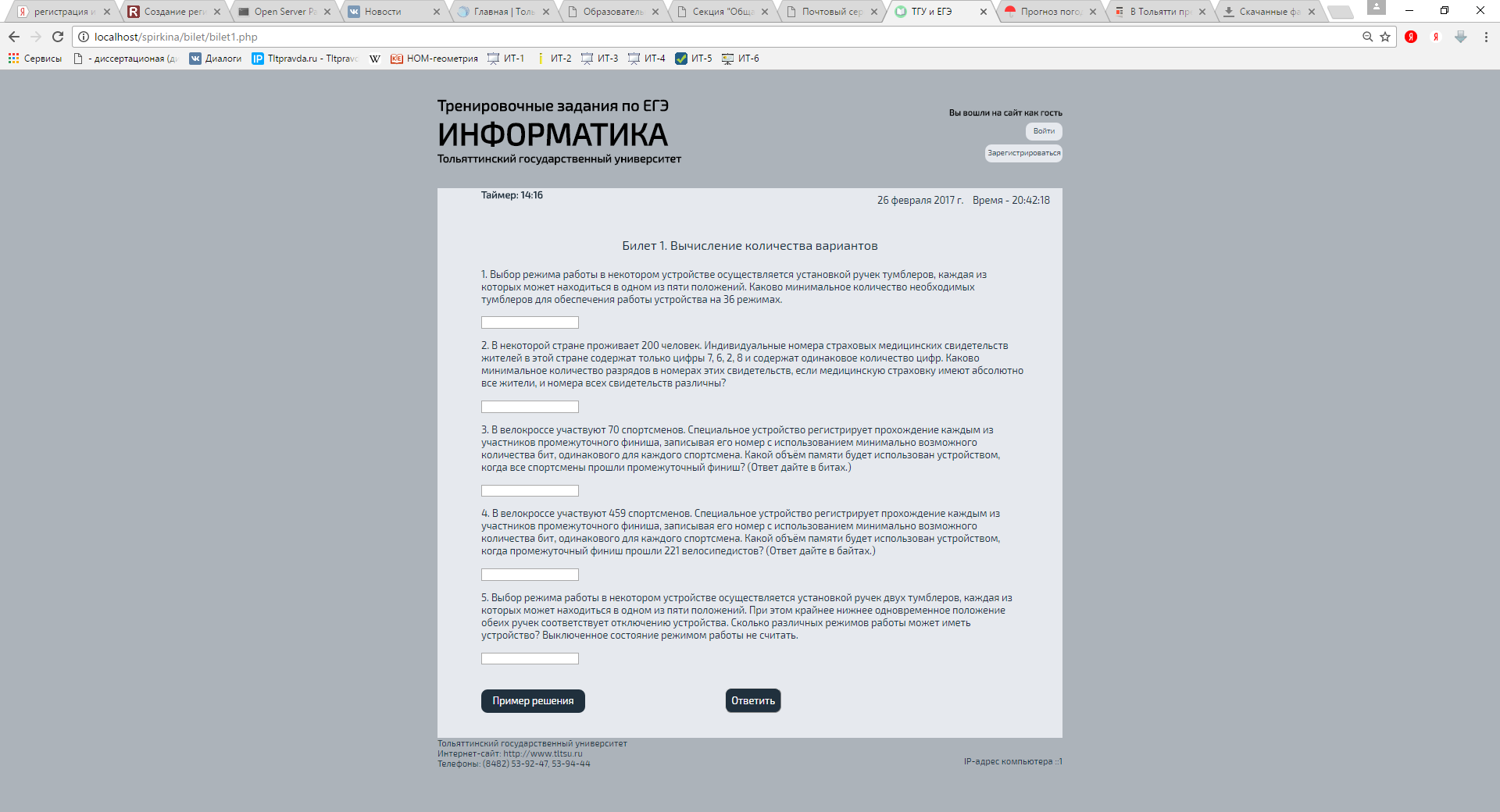


Рисунок 1

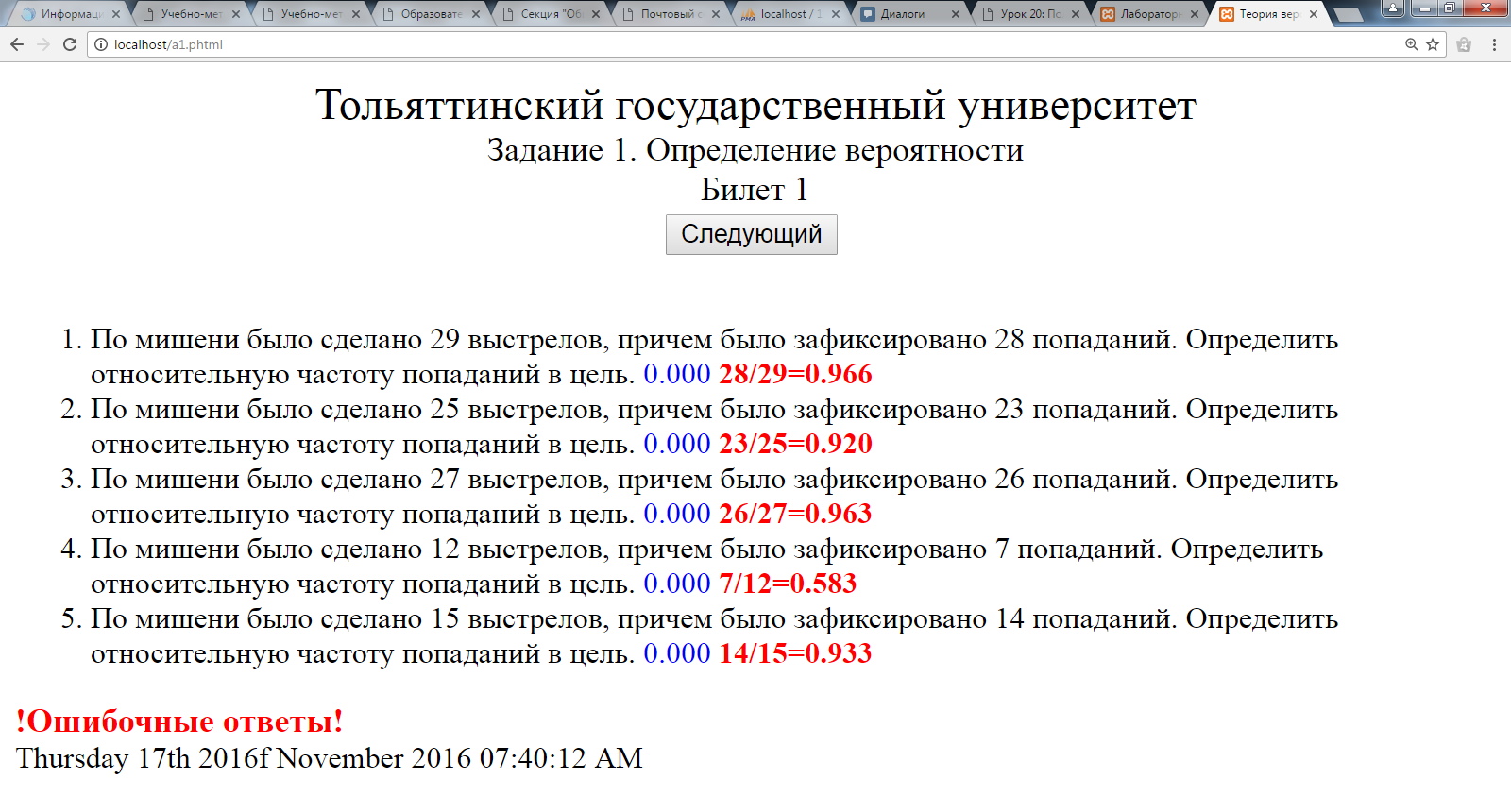


Рисунок 2

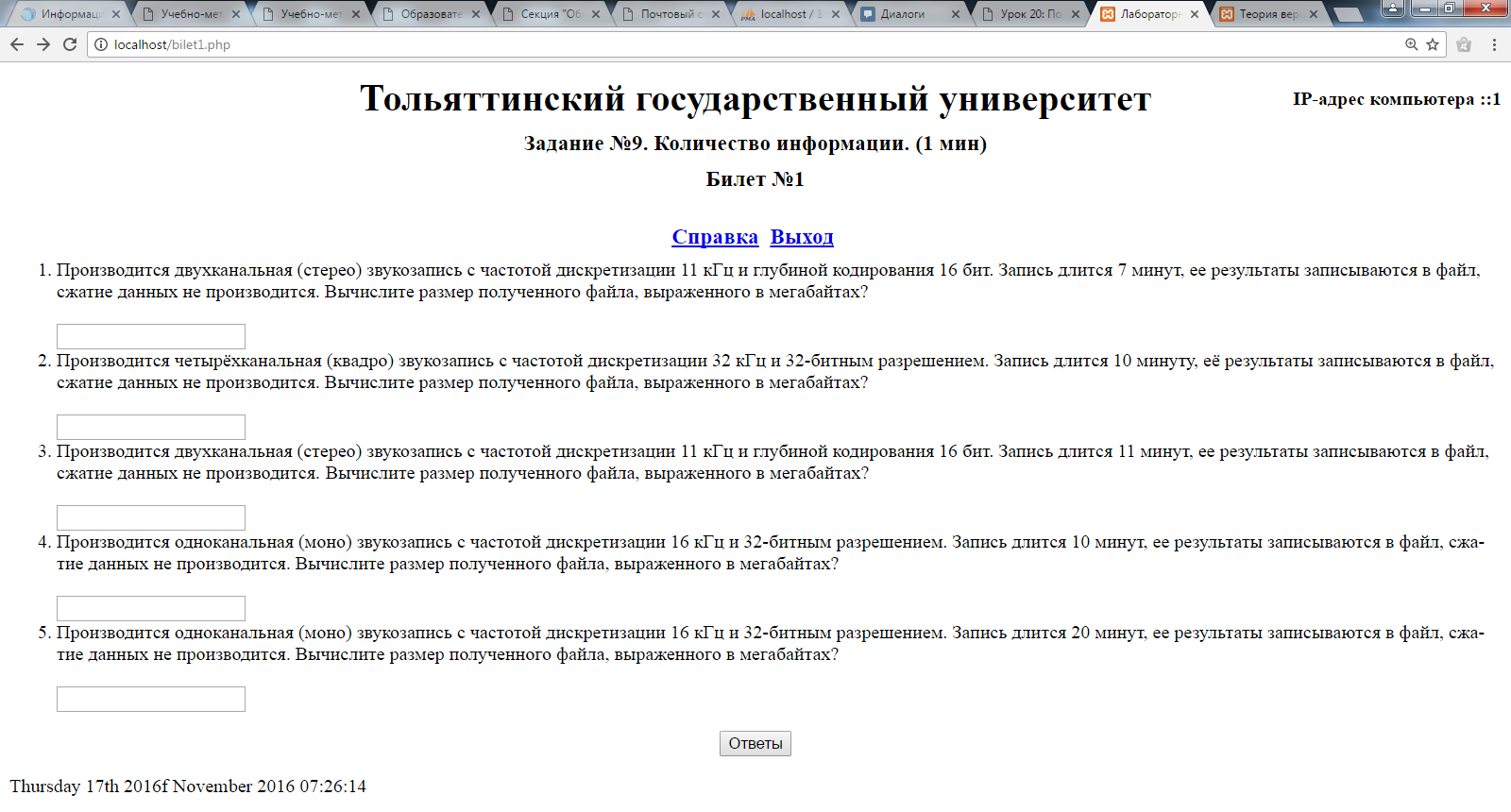


Рисунок 3

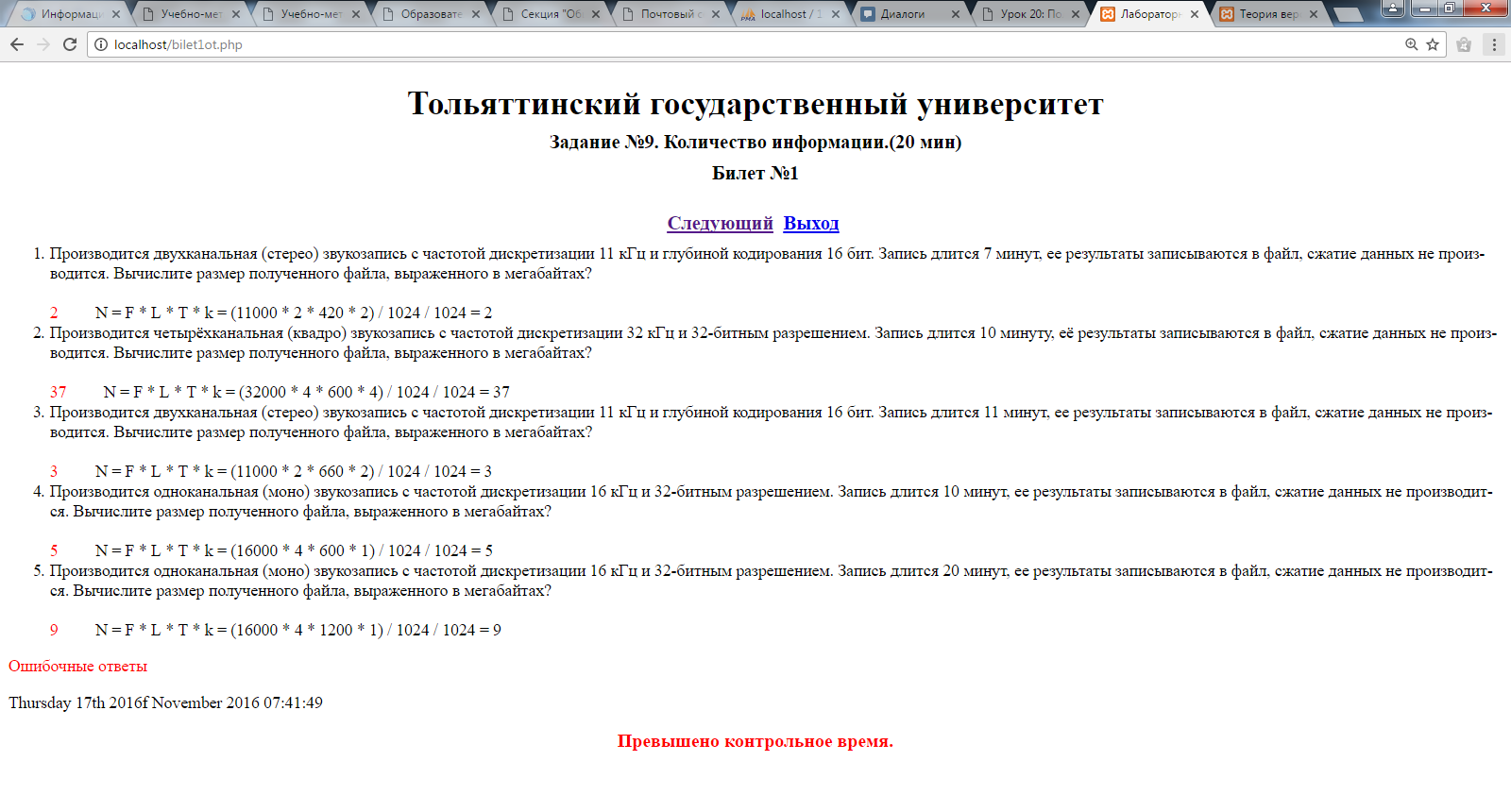


Рисунок 4