#### МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

ГЛАВНОЕ УПРАВЛЕНИЕ ПО ОБРАЗОВАНИЮ

МОГИЛЕВСКОГО ОБЛАСТНОГО ИСПОЛНИТЕЛЬНОГО КОМИТЕТА

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«МОГИЛЕВСКИЙ ГОСУДАРСТВЕННЫЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ»

**WEB-ПРОГРАММИРОВАНИЕ**

ДОМАШНЯЯ КОНТРОЛЬНАЯ РАБОТА №1

Выполнил Учащийся группы ПО-455

И. А. Дудкин

Шифр 16

2023**27 Опишите структуру PHP-сценария, основные элементы синтаксиса, способы внедрения PHP-сценария в веб-документ, назначение протокола HTTP и его методы, особенности передачи данных формы PHP-сценарию.**

В ходе своей работы модуль *PHP* читает и исполняет сценарий (*скрипт*) – текстовый файл, содержащий набор команд. Результат выполнения сценария – это *HTML*-документ, который *PHP* передает серверу *APACHE*, а тот уже – запрашивающему клиентскому браузеру.

*PHP*-сценарий следует сохранять в файлах с расширением имени*.php* в каталоге *htdocs* сервера *APACHE*. Запускать сценарий на выполнение можно только из браузера (*недвойным щелчком!),*набрав в адресной строке:

HTTP://localhost/имя.php

В этом случае мы создаем запрос по протоколу *HTTP*. В этом случае *APACHE* передает *PHP*-сценарий на обработку модулю *PHP*. Модуль *PHP* будет обрабатывать и выполнять только те команды скрипта, которые заключены в специальные теги-*дескрипторы PHP*:

*<?php*

*код на PHP*

*?>*

Действия, которые должен выполнить *PHP*-модуль, указываются *PHP*-операторами, помещающимися между открывающими и закрывающими скобками дескриптора.

*PHP*-сценарий можно встраивать в *HTML*-документ. В этом случае модуль *PHP* передает веб-серверу теги *HTML* без изменений и начнет обработку только команд, помещенных между дескрипторами *PHP*:

<HTML>

<HEAD>

<TITLE>

Сайт с PHP-кодом

</title>

</head>

<BODY BGCOLOR=silver TEXT=blue>

<?php

Echo ‘Это php-скрипт’;

/>

</body>

</html>

HTTP (англ. HyperText Transfer Protocol — «протокол передачи гипертекста») — протокол прикладного уровня передачи данных, изначально — в виде гипертекстовых документов в формате HTML, в настоящее время используется для передачи произвольных данных.

Методы

Метод HTTP (англ. HTTP Method) — последовательность из любых символов, кроме управляющих и разделителей, указывающая на основную операцию над ресурсом. Обычно метод представляет собой короткое английское слово, записанное заглавными буквами. Обратите внимание, что название метода чувствительно к регистру.

Сервер может использовать любые методы, не существует обязательных методов для сервера или клиента. Если сервер не распознал указанный клиентом метод, то он должен вернуть статус 501 (Not Implemented). Если серверу метод известен, но он неприменим к конкретному ресурсу, то возвращается сообщение с кодом 405 (Method Not Allowed). В обоих случаях серверу следует включить в сообщение ответа заголовок Allow со списком поддерживаемых методов.

Кроме методов GET и HEAD, часто применяется метод POST.

OPTIONS

Используется для определения возможностей веб-сервера или параметров соединения для конкретного ресурса. В ответ серверу следует включить заголовок Allow со списком поддерживаемых методов. Также в заголовке ответа может включаться информация о поддерживаемых расширениях.

Предполагается, что запрос клиента может содержать тело сообщения для указания интересующих его сведений. Формат тела и порядок работы с ним в настоящий момент не определён; сервер пока должен его игнорировать. Аналогичная ситуация и с телом в ответе сервера.

Для того, чтобы узнать возможности всего сервера, клиент должен указать в URI звёздочку — «\*». Запросы «OPTIONS \* HTTP/1.1» могут также применяться для проверки работоспособности сервера (аналогично «пингованию») и тестирования на предмет поддержки сервером протокола HTTP версии 1.1.

Результат выполнения этого метода не кэшируется.

GET

Используется для запроса содержимого указанного ресурса. С помощью метода GET можно также начать какой-либо процесс. В этом случае в тело ответного сообщения следует включить информацию о ходе выполнения процесса.

Клиент может передавать параметры выполнения запроса в URI целевого ресурса после символа «?»:

GET /path/resource?param1=value1&param2=value2 HTTP/1.1

Согласно стандарту HTTP, запросы типа GET считаются идемпотентными

Кроме обычного метода GET, различают ещё

* Условный GET — содержит заголовки If-Modified-Since, If-Match, If-Range и подобные;
* Частичный GET — содержит в запросе Range.

Порядок выполнения подобных запросов определён стандартами отдельно.

HEAD

Аналогичен методу GET, за исключением того, что в ответе сервера отсутствует тело. Запрос HEAD обычно применяется для извлечения метаданных, проверки наличия ресурса (валидация URL) и чтобы узнать, не изменился ли он с момента последнего обращения.

Заголовки ответа могут кэшироваться. При несовпадении метаданных ресурса с соответствующей информацией в кэше — копия ресурса помечается как устаревшая.

POST

Применяется для передачи пользовательских данных заданному ресурсу. Например, в блогах посетители обычно могут вводить свои комментарии к записям в HTML-форму, после чего они передаются серверу методом POST, и он помещает их на страницу. При этом передаваемые данные (в примере с блогами — текст комментария) включаются в тело запроса. Аналогично с помощью метода POST обычно загружаются файлы на сервер.

В отличие от метода GET, метод POST не считается идемпотентным, то есть многократное повторение одних и тех же запросов POST может возвращать разные результаты.

При результате выполнения 200 (Ok) в тело ответа следует включить сообщение об итоге выполнения запроса. Если был создан ресурс, то серверу следует вернуть ответ 201 (Created) с указанием URI нового ресурса в заголовке Location.

Сообщение ответа сервера на выполнение метода POST не кэшируется. POST http://soglasiepre.virtusystems.ru/Authentication\_JSON\_AppService.axd/Login HTTP/1.1 Accept: \*/\* User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; MathPlayer 2.10d; MRA 5.4 (build 02647); .NET CLR 2.0.50727; Content-Type: application/json; charset=utf-8 Host: vfos.virtusystems.ru Content-Length: 76 Expect: 100-continue Connection: Keep-Alive

PUT

Применяется для загрузки содержимого запроса на указанный в запросе URI. Если по заданному URI не существует ресурса, то сервер создаёт его и возвращает статус 201 (Created). Если же ресурс был изменён, то сервер возвращает 200 (Ok) или 204 (No Content). Сервер не должен игнорировать некорректные заголовки Content-\*, передаваемые клиентом вместе с сообщением. Если какой-то из этих заголовков не может быть распознан или недопустим при текущих условиях, то необходимо вернуть код ошибки 501 (Not Implemented).

Фундаментальное различие методов POST и PUT заключается в понимании предназначений URI ресурсов. Метод POST предполагает, что по указанному URI будет производиться обработка передаваемого клиентом содержимого. Используя PUT, клиент предполагает, что загружаемое содержимое соответствует находящемуся по данному URI ресурсу.

Сообщения ответов сервера на метод PUT не кэшируются.

PATCH

Аналогично PUT, но применяется только к фрагменту ресурса.

DELETE

Удаляет указанный ресурс.

TRACE

Возвращает полученный запрос так, что клиент может увидеть, какую информацию промежуточные серверы добавляют или изменяют в запросе.

CONNECT

Преобразует соединение запроса в прозрачный TCP/IP-туннель, обычно чтобы содействовать установлению защищённого SSL-соединения через нешифрованный прокси.

HTTP-методы GET и POST — самые распространённые способы отправить или получить данные с сервера. Но в разных случаях оба метода могут быть небезопасными или неудобными в использовании. В этой заметке рассмотрим, какой метод когда использовать.

GET — метод для чтения данных с сайта. Например, для доступа к указанной странице. Он говорит серверу, что клиент хочет прочитать указанный документ. На практике этот метод используется чаще всего, например, в интернет-магазинах на странице каталога. Фильтры, которые выбирает пользователь, передаются через метод GET.

:authority: htmlacademy.ru

:method: GET

:path: /tutorial/php/http-header

POST — метод для отправки данных на сайт. Чаще всего с помощью метода POST передаются формы.

URL-адрес запроса: https://htmlacademy.ru/consulting

Метод запроса: POST

Код состояния: 200

## Формат запроса

Протокол HTTP очень прост и состоит, по сути, из двух частей — заголовков и тела запроса или ответа.

**Тело запроса** — это информация, которую передал браузер при запросе страницы. Но тело запроса присутствует только если браузер запросил страницу методом POST. Например, если отправлена форма, то телом запроса будет содержание формы.

Пример GET-запроса. Информация передаётся прямо в заголовке.

GET /blog/?name1=value1&name2=value2 HTTP/1.1

Host: htmlacademy.ru

Пример POST-запроса. Информация передаётся в теле запроса:

POST /blog/ HTTP/1.1

Host: htmlacademy.ru

name1=value1&name2=value2

## GET для безопасных действий, POST для опасных

Говоря совсем просто, GET-запросы лучше не использовать с приватной информацией. Вот почему:

* **Они кэшируются.** Это значит, что логин и пароль, переданные через GET-запрос, могут остаться в интернете навсегда, например, в веб-архиве или кэше Гугла.
* **Остаются в истории браузера.** Чтобы узнать, какие данные отправлялись, достаточно нажать Ctrl+H.
* **Сохраняются в закладках и пересылаются.** Можно не обратить внимания и опубликовать в соцсетях или отправить ссылку с приватной информацией в GET-запросе.
* **Сохраняются в логах сервера.** Например, нельзя отправлять данные банковских карт через GET-запрос, так как это создаёт риски для пользователей.

Таким образом, любые важные данные — логины, пароли, данные карты, персональные данные — лучше передавать с помощью метода POST. Также метод POST поддерживает тип кодирования данных multipart/form-data, что позволяет передавать файлы.

## Ещё раз коротко

**GET**

* Фильтры в интернет-магазинах
* Передача параметров через ссылку
* Другие безопасные запросы

**POST**

* Любые формы с паролями или банковскими картами
* Формы заявок с персональными данными
* Отправка файлов

**57 Дан массив a = [15, 97, 2, 158, 26, 268, 88, 105, 43]. При помощи цикла while найти сумму нечетных элементов данного массива.**

**Выполнение задания.**

<!-- index.html -->

<!DOCTYPE html> - указываем тип документа

<html lang='ru'> - язык страницы для браузера

<head> - заголовочная часть

<meta charset='utf-8'> - кодировка

<title>Дудкин задание 57</title> - Заголовок страницы

</head>

<body> - начинаем тело страницы

<script> - и начинаем описание скрипта

function summodd(){ - определяем функцию нажатия на кнопку

let array = [15, 97, 2, 158, 26, 268, 88, 105, 43];// массив данных условия

let result = 0; // сюда будем суммировать

let i = 0; // переменная цикла для доступа к элементам массива

while ( i < array.length) { перебираем элементы массива

result += array[i]; // суммируем текущий элемент

i = i + 2; // выбираем элемент с шагом 2

}

console.log(result); // выводим результат в консоль

}

</script>

<input type=button value='Вычислить' onclick = summodd()><br> - кнопка для выполнения поиска суммы

<br>

<br> - кратко опишем задание на странице

Функция вычисляет сумму элементов массива [15, 97, 2, 158, 26, 268, 88, 105, 43] <br>

находящиеся на нечетных местах.

</body>

</html> - завершаем тело страницы и саму страницу

Первоначальный вид страницы представлен на рисунке 1.

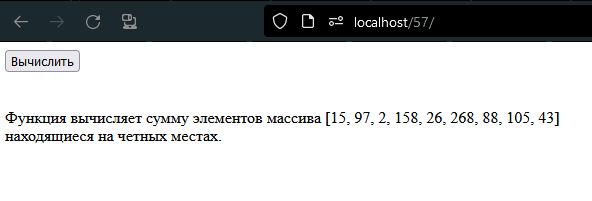


Рисунок 1. Первоначальный вид страницы.

После нажатия на кнопку браузер примет вид представленный на рисунке 2.

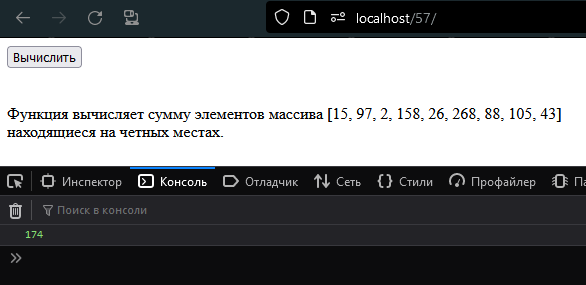


Рисунок 2. Вид браузера после выполнения задания.

**Исходный текст страницы:**

<!-- index.html -->

<!DOCTYPE html>

<html lang='ru'>

<head>

<meta charset='utf-8'>

<title>Дудкин задание 57</title>

</head>

<body>

<script>

function summodd(){

let array = [15, 97, 2, 158, 26, 268, 88, 105, 43];

let result = 0;

let i = 0;

while ( i < array.length) {

result += array[i];

i = i + 2;

}

console.log(result);

}

</script>

<input type=button value='Вычислить' onclick = summodd()><br>

<br>

<br>

Функция вычисляет сумму элементов массива [15, 97, 2, 158, 26, 268, 88, 105, 43] <br>

находящиеся на нечетных местах.

</body>

</html>

**61 Дан двумерный массив (матрица), состоящий из N строк и N элементов, где N вводится с клавиатуры при помощи prompt. Заполните каждую строку случайными числами в диапазоне от -10 до 30. Определите:**

**- количество строк, не содержащих ни одного нулевого элемента;**

**- максимальное из чисел в заданном массиве.**

**Выполнение задания.**

<!-- index.html -->

<!DOCTYPE html> - тип документа

<html lang='ru'> - язык документа

<head> - заголовочная часть

<meta charset='utf-8'> - кодировка

<title>Дудкин задание 61</title> - заголовок страницы

</head>

<body>

<script> - описание скрипта

function mass(){

// Ввести n // запросим размер массива

var n = prompt("Введите размер массива N");

// Объявление массива

var A = new Array()

// Указать количество строк

A.length = n;

// Указать количество столбцов

for (var i=0; i<n; i++)

{

// Создать подмассив в массиве

A[i] = new Array()

// Установить длину массива

A[i].length = n

}

// Заполнить массив A значениями -10 -- 30

var t = 0;

for (var i=0; i<A.length; i++){

for (var j=0; j<A[i].length; j++){

// найдем случайное число в нужном диапазоне

t = Math.floor(Math.random() \* 40)-10;

A[i][j] = t

}

}

// зададим и обнулим переменные

var zer = 0;

var tzer = 0;

var maxn = A[0][0]; // для начала примем первый элемент как MAX

// обойдем все строки массива

for (var i=0; i<A.length; i++){

tzer = 0;

for (var j=0; j<A[i].length; j++){

// ищем нулевые элементы в строке

if (A[i][j]==0) tzer +=1;

// и сразу ищем максимальный элемент

if (A[i][j]>maxn) maxn = A[i][j];

}

// если нулевых элементов не обнаружено

//увеличиваем счетчик ненулевых строк

if (tzer==0) zer +=1;

}

// вывести массив

console.log(A);

// вывести счетчик ненулевых строк

console.log("Строк не содержащих ни одного нулевого элемента: ",zer);

// и максимальный элемент

console.log("Максимальный элемент: ",maxn);

}

</script>

<input type=button value='Выполнить' onclick = mass()><br>

<br>

<br> - выведем на экран содержимое самого задания

Дан двумерный массив (матрица), состоящий из N строк и N элементов, где N вводится с клавиатуры при помощи prompt.<br>

Заполните каждую строку случайными числами в диапазоне от -10 до 30. Определите:<br>

- количество строк, не содержащих ни одного нулевого элемента;<br>

- максимальное из чисел в заданном массиве.

</body> - завершение тела

</html> - завершение документа.

Вид браузера после открытия страницы представлен на рисунке 3.

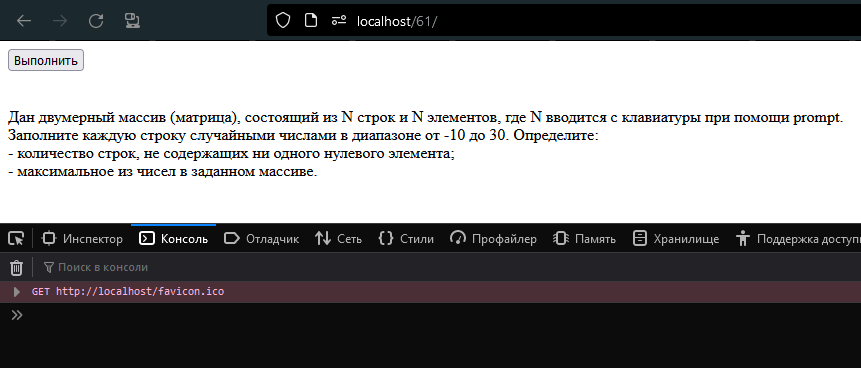


Рисунок 3. Вид браузера после открытия страницы.

Вид браузер с запросом размерности представлен на рисунке 4.

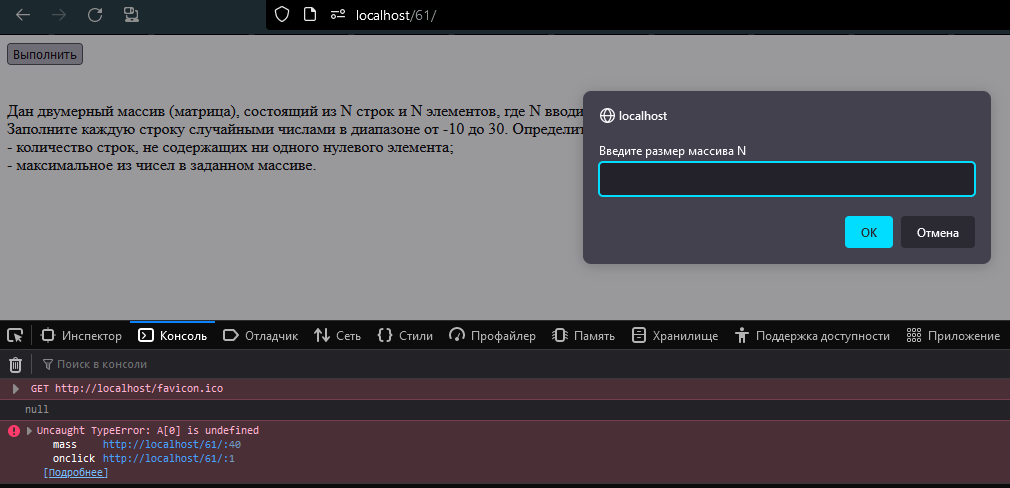


Рисунок 4. Вид браузер ас запросом размера.

Вид браузера после нажатия на кнопку представлен на рисунке 5.

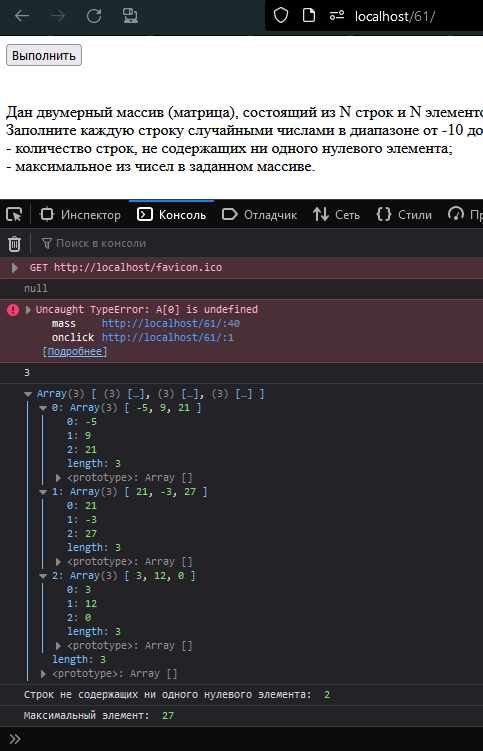


Рисунок 5. Вид браузера после нажатия на кнопку.

**Исходный код страницы.**

<!-- index.html -->

<!DOCTYPE html>

<html lang='ru'>

<head>

<meta charset='utf-8'>

<title>Дудкин задание 61</title>

</head>

<body>

<script>

function mass(){

// Ввести n

var n = prompt("Введите размер массива N");

// Объявление массива

var A = new Array()

// Указать количество строк

A.length = n;

// Указать количество столбцов

for (var i=0; i<n; i++)

{

// Создать подмассив в массиве

A[i] = new Array()

// Установить длину массива

A[i].length = n

}

// Заполнить массив A значениями -10 -- 30

var t = 0;

for (var i=0; i<A.length; i++){

for (var j=0; j<A[i].length; j++){

// найдем случайное число в нужном диапазоне

t = Math.floor(Math.random() \* 40)-10;

// рандом возвращает число от нуля до 1, поэтому домножим его на ширину необходимого нам диапазона (40=30+10) и сдвинем влево по числовой прямой.

A[i][j] = t

}

}

// зададим и обнулим переменные

var zer = 0;

var tzer = 0;

var maxn = A[0][0];

// обойдем все строки массива

for (var i=0; i<A.length; i++){

tzer = 0;

for (var j=0; j<A[i].length; j++){

// ищем нулевые элементы в строке

if (A[i][j]==0) tzer +=1;

// и сразу ищем максимальный элемент

if (A[i][j]>maxn) maxn = A[i][j];

}

// если нулевых элементов не обнаружено

//увеличиваем счетчик ненулевых строк

if (tzer==0) zer +=1;

}

// вывести массив

console.log(A);

// вывести счетчик ненулевых строк

console.log("Строк не содержащих ни одного нулевого элемента: ",zer);

// и максимальный элемент

console.log("Максимальный элемент: ",maxn);

}

</script>

<input type=button value='Выполнить' onclick = mass()><br>

<br>

<br>

Дан двумерный массив (матрица), состоящий из N строк и N элементов, где N вводится с клавиатуры при помощи prompt.<br>

Заполните каждую строку случайными числами в диапазоне от -10 до 30. Определите:<br>

- количество строк, не содержащих ни одного нулевого элемента;<br>

- максимальное из чисел в заданном массиве.

</body>

</html>

**97 Спроектировать структуру базы данных о квартирах, предназначенных для продажи: район, этаж, площадь, количество комнат, сведения о владельце, цена.**

**Запросы:**

**а) вывести информацию о квартирах, расположенных на заданном этаже;**

**б) вывести информацию о квартирах, имеющих площадь из указанного интервала.**

**Выполнение задания.**

Создадим файл index.php. В нем сразу создадим базу данных, подключим её для использования, создадим таблицу и заполним её тестовыми данными и выведем кнопку для выполнения этих действий и подготовки к работе.

Вид страницы после открытия представлен на рисунке 6.

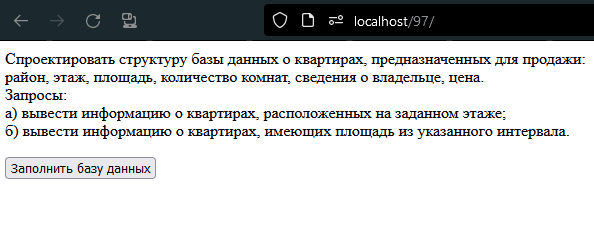


Рисунок 6. Вид страницы после открытия.

После нажатия на кнопку, база данных создана и заполнена тестовыми данными. Выведем таблицу на экран со списком возможных действий. Вид страницы на втором шаге представлен на рисунке 7.



Рисунок 7. Вид страницы на втором шаге.

Для выполнения первого задания нажмем соответствующую кнопку и заполним необходимые параметры. Вид страницы с заполненными параметрами представлен на рисунке 8.

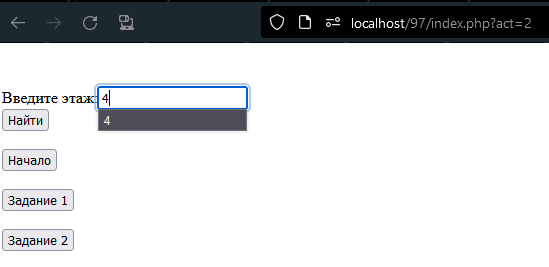


Рисунок 8. Вид страницы с заполненными параметрами.

Результат работы приложения представлен на рисунке 9.

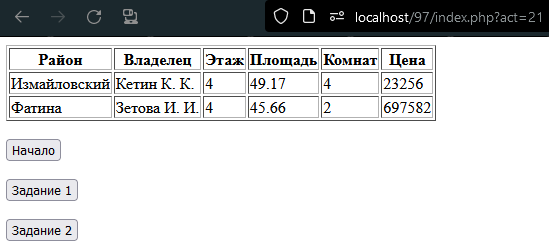


Рисунок 9. Результат работы приложения.

Нажмем кнопку Задание 2 для выполнения второго задания и заполним требующиеся данные. Вид страницы с заполненными данными представлен на рисунке 10.

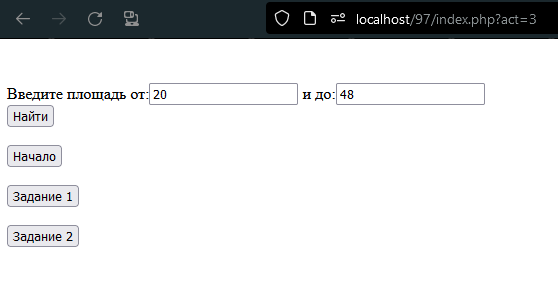


Рисунок 10. Вид страницы с заполненными данными.

Результат поиска представлен на рисунке 11.

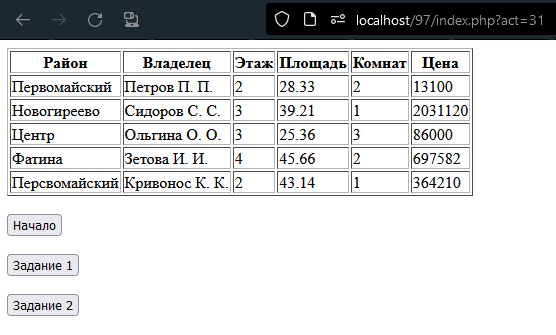


Рисунок 11. Результат поиска.

**Исходный код скрипта с пояснениями.**

<!-- index.php -->

<!-- заголовки файла для корректного отображения в браузере-->

<!DOCTYPE html>

<html lang='ru'>

<head>

<meta charset='utf-8'>

<title>Добро пожаловать!</title>

</head>

<body>

<!-- начинается код-->

<?php

// тут мы переносим переменные параметры из глобального массива

// в локальный, сделано для безопасности на стороне сервера

// ограничение по передаче переменных

isset($\_GET['act'])?$act = $\_GET['act']:$act=0;

isset($\_POST['et'])?$et = $\_POST['et']:$et="";

isset($\_POST['sq1'])?$sq1 = $\_POST['sq1']:$sq1="0";

isset($\_POST['sq2'])?$sq2 = $\_POST['sq2']:$sq2="0";

// соединяемся с базой данных

$link = mysqli\_connect("localhost", "root", "");

// если скрипт запускается впервые а базы не существует то создадим её

$sql = "CREATE DATABASE IF NOT EXISTS `KVART` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci;";

$result = mysqli\_query($link, $sql);

// и выберем для работы

$sql = "USE `KVART`;";

if (!isset($act)) $act = 0;

$result = mysqli\_query($link, $sql);

// организуем выбор режимов и этапов работы

switch ($act)

{

case 0:// часть которая выпоняется если нет параметров, основная

echo"Спроектировать структуру базы данных о квартирах, предназначенных для продажи:<br>

район, этаж, площадь, количество комнат, сведения о владельце, цена.<br>

Запросы:<br>

а) вывести информацию о квартирах, расположенных на заданном этаже;<br>

б) вывести информацию о квартирах, имеющих площадь из указанного интервала.<br>

";

echo"<br><form action='index.php?act=1' method=POST><input type=submit value='Заполнить базу данных'></form><br>";

break; // выводим кнопку для создания и заполнения тоаблицы

case 1: // в этой части мы создаем таблицу, если её не существует

echo"<br> База данных квартир создана и заполнена. Содержимое: <br><br>";

$sql = "CREATE TABLE IF NOT EXISTS `dat` (`raon` VARCHAR(100) NOT NULL , `mast` VARCHAR(500) NULL , `etag` INT NOT NULL , `sq` FLOAT NOT NULL , `rooms` INT NOT NULL , `cena` FLOAT NOT NULL ) ENGINE = InnoDB;";

$result = mysqli\_query($link, $sql);

// а если существует то очищаем

$sql = "TRUNCATE TABLE `dat`";

$result = mysqli\_query($link, $sql);

// и вставляем 10 записей с журналами

$sql = "INSERT INTO `dat` (`raon`, `mast`, `etag`, `sq`, `rooms`, `cena`) VALUES ('Ленинский', 'Иванов И. И.', '1', '18.22', '3', '23120');";

$result = mysqli\_query($link, $sql);

$sql = "INSERT INTO `dat` (`raon`, `mast`, `etag`, `sq`, `rooms`, `cena`) VALUES ('Первомайский', 'Петров П. П.', '2', '28.33', '2', '13100');";

$result = mysqli\_query($link, $sql);

$sql = "INSERT INTO `dat` (`raon`, `mast`, `etag`, `sq`, `rooms`, `cena`) VALUES ('Новогиреево', 'Сидоров С. С.', '3', '39.21', '1', '2031125');";

$result = mysqli\_query($link, $sql);

$sql = "INSERT INTO `dat` (`raon`, `mast`, `etag`, `sq`, `rooms`, `cena`) VALUES ('Измайловский', 'Кетин К. К.', '4', '49.17', '4', '23256');";

$result = mysqli\_query($link, $sql);

$sql = "INSERT INTO `dat` (`raon`, `mast`, `etag`, `sq`, `rooms`, `cena`) VALUES ('Шабаны', 'Шаблин Ш. Ш.', '1', '56.77', '4', '150758');";

$result = mysqli\_query($link, $sql);

$sql = "INSERT INTO `dat` (`raon`, `mast`, `etag`, `sq`, `rooms`, `cena`) VALUES ('Серебрянка', 'Рамин Р. Р.', '2', '65.23', '3', '772544');";

$result = mysqli\_query($link, $sql);

$sql = "INSERT INTO `dat` (`raon`, `mast`, `etag`, `sq`, `rooms`, `cena`) VALUES ('Центр', 'Ольгина О. О.', '3', '25.36', '3', '86000');";

$result = mysqli\_query($link, $sql);

$sql = "INSERT INTO `dat` (`raon`, `mast`, `etag`, `sq`, `rooms`, `cena`) VALUES ('Фатина', 'Зетова И. И.', '4', '45.66', '2', '697582');";

$result = mysqli\_query($link, $sql);

$sql = "INSERT INTO `dat` (`raon`, `mast`, `etag`, `sq`, `rooms`, `cena`) VALUES ('Колос', 'Почубеев П. Ч.', '1', '78.22', '1', '545263');";

$result = mysqli\_query($link, $sql);

$sql = "INSERT INTO `dat` (`raon`, `mast`, `etag`, `sq`, `rooms`, `cena`) VALUES ('Персвомайский', 'Кривонос К. К.', '2', '43.14', '1', '364210');";

$result = mysqli\_query($link, $sql);

// выводим данные в таблицу

echo"<table border=1>";// начало таблицы

$sql = 'SELECT \* FROM dat';// выбираем все записи

$result = mysqli\_query($link, $sql);

// выводим заголовки таблицы

echo"<tr><th>Район</th><th>Владелец</th><th>Этаж</th><th>Площадь</th><th>Комнат</th><th>Цена</th></tr>";

while ($row = mysqli\_fetch\_array($result)) {// и по каждой записи из выборки

echo"<tr><td>" . $row['raon'] . "</td><td>" . $row['mast'] . "</td><td>". $row['etag'] . "</td><td>". $row['sq'] . "</td>";

echo "<td>" . $row['rooms'] . "</td><td>". $row['cena'] . "</td></tr>";

} // формируем строки таблицы

echo"</table>";// закрываем таблицу

// далее выводим кнопки с выбором варианта действий

echo"<br><form action='index.php?act=0' method=POST><input type=submit value='Начало'></form>";

echo"<br><form action='index.php?act=2' method=POST><input type=submit value='Задание 1'></form>";

echo"<br><form action='index.php?act=3' method=POST><input type=submit value='Задание 2'></form>";

break;

case 2: // Выбрано действие поиска квартиры по этажу, создаем форму

echo"<br><form action='index.php?act=21' method=POST>

<br>Введите этаж:<input type=text name='et'><br>

<input type=submit value='Найти'></form>"; // в ней запрашиваем название

// далее выводим кнопки с выбором варианта действий

echo"<br><form action='index.php?act=0' method=POST><input type=submit value='Начало'></form>";

echo"<br><form action='index.php?act=2' method=POST><input type=submit value='Задание 1'></form>";

echo"<br><form action='index.php?act=3' method=POST><input type=submit value='Задание 2'></form>";

break;

case 21: // выбираем квартиру по этажу, et - параметр с этажом, формируем запрос

$sql = "SELECT \* FROM dat where etag='".$et."'";

$result = mysqli\_query($link, $sql);

// выводим данные в таблицу

echo"<table border=1>";

echo"<tr><th>Район</th><th>Владелец</th><th>Этаж</th><th>Площадь</th><th>Комнат</th><th>Цена</th></tr>";

while ($row = mysqli\_fetch\_array($result)) {

echo"<tr><td>" . $row['raon'] . "</td><td>" . $row['mast'] . "</td><td>". $row['etag'] . "</td><td>". $row['sq'] . "</td>";

echo "<td>" . $row['rooms'] . "</td><td>". $row['cena'] . "</td></tr>";

}

echo"</table>";

// далее выводим кнопки с выбором варианта действий

echo"<br><form action='index.php?act=0' method=POST><input type=submit value='Начало'></form>";

echo"<br><form action='index.php?act=2' method=POST><input type=submit value='Задание 1'></form>";

echo"<br><form action='index.php?act=3' method=POST><input type=submit value='Задание 2'></form>";

break;

case 3: // выводим форму для выбора диапазона площадей

echo"<br><form action='index.php?act=31' method=POST>

<br>Введите площадь от:<input type=text name='sq1'> и до:<input type=text name='sq2'><br>

<input type=submit value='Найти'></form>";

// далее выводим кнопки с выбором варианта действий

echo"<br><form action='index.php?act=0' method=POST><input type=submit value='Начало'></form>";

echo"<br><form action='index.php?act=2' method=POST><input type=submit value='Задание 1'></form>";

echo"<br><form action='index.php?act=3' method=POST><input type=submit value='Задание 2'></form>";

break;

case 31: // формируем запрос с параметрами s1 и s2

$sql = "SELECT \* FROM dat where sq between ".$sq1." and ".$sq2;

$result = mysqli\_query($link, $sql);

// выводим результат таблицей

echo"<table border=1>";

echo"<tr><th>Район</th><th>Владелец</th><th>Этаж</th><th>Площадь</th><th>Комнат</th><th>Цена</th></tr>";

while ($row = mysqli\_fetch\_array($result)) {

echo"<tr><td>" . $row['raon'] . "</td><td>" . $row['mast'] . "</td><td>". $row['etag'] . "</td><td>". $row['sq'] . "</td>";

echo "<td>" . $row['rooms'] . "</td><td>". $row['cena'] . "</td></tr>";

}

echo"</table>";

// далее выводим кнопки с выбором варианта действий

echo"<br><form action='index.php?act=0' method=POST><input type=submit value='Начало'></form>";

echo"<br><form action='index.php?act=2' method=POST><input type=submit value='Задание 1'></form>";

echo"<br><form action='index.php?act=3' method=POST><input type=submit value='Задание 2'></form>";

break;

}

?>

</body></html>

**Список использованных источников**

1 Минник, К. JavaScript / К. Минник, Е. Холланд. - М. : Москва, 2017. - 320 с.

2 Маккоу, А Веб-приложения на JavaScript / А. Маккоу. - СПб. : Питер, 2014. - 285 с.

3 Макфарланд, Д. JavaScript и jQuery. Исчерпывающее руководство / Д. Макфарланд. - СПб.: Питер, 2016. - 880 с.

4 Жадаев, А.Г. PHP для начинающих / А.Г. Жадаев – СПб.: Питер, 2014. – 288 с.

5 Котеров Д.В. PHP 7 в подлиннике / Д.В. Котеров – СПб.: Питер, 2016. – 1073 с.

6 Никольский А.П. JavaScript на примерах / А.П. Никольский – СПб.: Питер, 2018. – 272 c.