



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт Информационных технологий

Кафедра Инструментального и прикладного программного обеспечения

ПРАКТИЧЕСКАЯ РАБОТА №2

по дисциплине «Разработка серверных частей интернет-ресурсов»

Тема практической работы:

Студент группы ИКБО-16-19

Мурадов Н.Н.

(подпись студента)

Руководитель практической работы

преподаватель Волков М.Ю.

(подпись руководителя)

Работа представлена

« ____ » _____ 2021 г.

Допущен к работе

« ____ » _____ 2021 г.

Москва 2021

Цель работы

Используя серверную конфигурацию, разработанную в прошлой практической работе выполнить следующие упражнения. Предполагается создать 3 независимых сервиса, устойчивых к минимальному набору самых простых ошибок.

Задание 1:

Предлагается создать веб-сервис Drawer для рисования svg объектов. Ему передается один параметр - целое число, представляющее закодированная фигура для рисования.

Код программы:

docker-compose.yml – общий для всех трех заданий

```
version: '3'
services:
  drawer:
    build: ./drawer
    volumes:
      - ./drawer:/var/www/html
    ports:
      - 4543:80
  sortirovki:
    build: ./sortirovki
    volumes:
      - ./sortirovki:/var/www/html
    ports:
      - 4544:80
  uncom:
    build: ./uncom
    volumes:
      - ./uncom:/var/www/html
    ports:
      - 4545:80
```

Dockerfile

```
FROM php:7.2-apache
RUN docker-php-ext-install mysqli pdo pdo_mysql && docker-php-ext-enable pdo_mysql
RUN echo "ServerName localhost" >> /etc/apache2/apache2.conf
```

Index.php

```
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Drawer</title>
  </head>
  <body>
    <form action="script.php">
      <input name="param_name" />
      <button type="submit">Отправить</button>
    </form>
  </body>
</html>
```

script.php

```
<?php
$stri = $_GET['param_name'];
$dvu = $stri;
$form = $dvu & 0b1111000000000000;
$cvet = $dvu & 0b0000111100000000;
$razmer_w = $dvu & 0b0000000011110000;
$razmer_h = $dvu & 0b0000000000001111;
$form = $form >> 12;
$cvet = $cvet >> 8;
$razmer_w = $razmer_w >> 4;
$razmer_w = $razmer_w * 100;
$razmer_r = $razmer_h * 10;
$razmer_h = $razmer_h * 100;
$razmer_x = $razmer_w / 2;
$razmer_y = $razmer_h / 2;
$r_cvet = 0;
switch($cvet):
  case 1:
    $r_cvet = '#ff0000';#красный
    break;
  case 2:
    $r_cvet = '#00ff00';#зелёный
    break;
  case 3:
    $r_cvet = '#0000ff';#синий
    break;
  case 4:
    $r_cvet = '#000000';#чёрный
    break;
  case 5:
    $r_cvet = '#ffffff';#белый
    break;
endswitch;

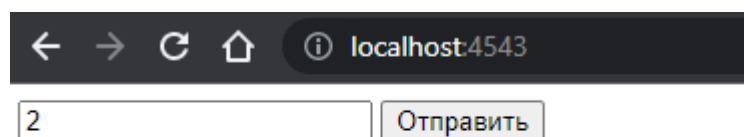
switch($form):
```

```

    case 1:
        echo "<svg width='$razmer_w' height='$razmer_h' r='$razmer_r'
xmlns='http://www.w3.org/1999/svg'>
        <circle cx='$razmer_x' cy='$razmer_y' r='$razmer_r' fill='$r_cvet' />
        </svg>";
    break;
    case 2:
        echo "<svg width='$razmer_w' height='$razmer_h'
xmlns='http://www.w3.org/2000/svg'>
        <rect x='$razmer_x' y='$razmer_y' width='$razmer_w' height='$razmer_h'
fill='$r_cvet' />
        </svg>";
    break;
    case 3:
        echo "<svg width='$razmer_w' height='$razmer_h' r='$razmer_r'
xmlns='http://www.w3.org/1999/svg'>
        <rect x='$razmer_x' y='$razmer_y' width='$razmer_w' height='$razmer_h'
fill='$r_cvet' />
        </svg>";
    break;
endswitch;
#0b0010000100110011(8499) - красный квадрат 300x300
?>

```

Результат:



The image shows a web browser interface. At the top, there is a dark address bar with navigation icons (back, forward, refresh, home) and an information icon followed by the text 'localhost:4543'. Below the address bar, there is a light-colored text input field containing the number '2'. To the right of the input field is a button with the text 'Отправить' (Send).

Рис. 1 – Отправка запроса

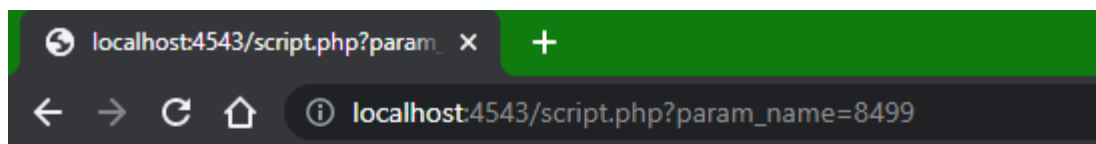


Рис. 2 – Результат отрисовки фигуры

Задание 2:

Реализовать одну из сортировок на языке программирования PHP по варианту:

Вариант 4 – Сортировка выбором.

Код программы:

docker-compose.yml – общий для всех трех заданий

```
version: '3'
services:
  drawer:
    build: ./drawer
    volumes:
      - ./drawer:/var/www/html
    ports:
      - 4543:80
  sortirovki:
    build: ./sortirovki
    volumes:
      - ./sortirovki:/var/www/html
    ports:
      - 4544:80
  uncom:
    build: ./uncom
    volumes:
      - ./uncom:/var/www/html
    ports:
      - 4545:80
```

Dockerfile

```
FROM php:7.2-apache
RUN docker-php-ext-install mysqli pdo pdo_mysql && docker-php-ext-enable pdo_mysql
RUN echo "ServerName localhost" >> /etc/apache2/apache2.conf
```

Index.html

```
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Sortirovka</title>
  </head>
  <body>
    <form method="get" action="script.php">
      <input type="text" name="my_sort">
      <input type="submit" value="Сортировать!">
    </body>
</html>
```

script.php

```
<?php
$array = explode(" ", $_GET["my_sort"]);
sel_sort($array);
echo implode(" ", $array);
function sel_sort (&$array) {
    $size = count($array);
    for ($i = 0; $i < $size; $i++)
    {
        $min = $i;
        for ($j = $i + 1; $j < $size; $j++)
        {
            if ($array[$j] < $array[$min])
            {
                $min = $j;
            }
        }
        $temp = $array[$i];
        $array[$i] = $array[$min];
        $array[$min] = $temp;
    }
}
?>
```

Результат:

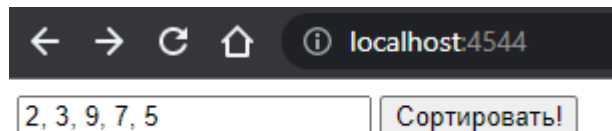


Рис.3 – Ввод неотсортированного массива

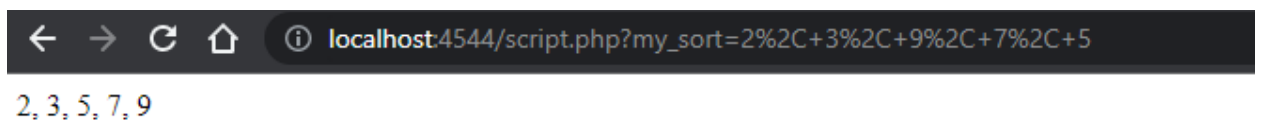


Рис.4 – Отсортированный массив

Задание 3:

Реализовать информационно-административную веб-страницу о сервере с помощью таких команд Unix как: ls, ps, whoami, id и так далее.

Код программы:

docker-compose.yml – общий для всех трех заданий

```
version: '3'
services:
  drawer:
    build: ./drawer
    volumes:
      - ./drawer:/var/www/html
    ports:
      - 4543:80
  sortirovki:
    build: ./sortirovki
    volumes:
      - ./sortirovki:/var/www/html
    ports:
      - 4544:80
  uncom:
    build: ./uncom
    volumes:
      - ./uncom:/var/www/html
    ports:
      - 4545:80
```

Dockerfile

```
FROM php:7.2-apache
RUN docker-php-ext-install mysqli pdo pdo_mysql && docker-php-ext-enable pdo_mysql
RUN echo "ServerName localhost" >> /etc/apache2/apache2.conf
```

Index.php

```
<html lang="en">
  <head>
    <title>Uncom</title>
  </head>
  <body>
    <h1>Unix commands</h1>
    <?php
      $output = shell_exec('ls');
      echo "<pre>$output</pre>";
      $output = shell_exec('ps');
      echo "<pre>$output</pre>";
      $output = shell_exec('w');
      echo "<pre>$output</pre>";
      $output = shell_exec('id');
      echo "<pre>$output</pre>";
```



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

Результат:

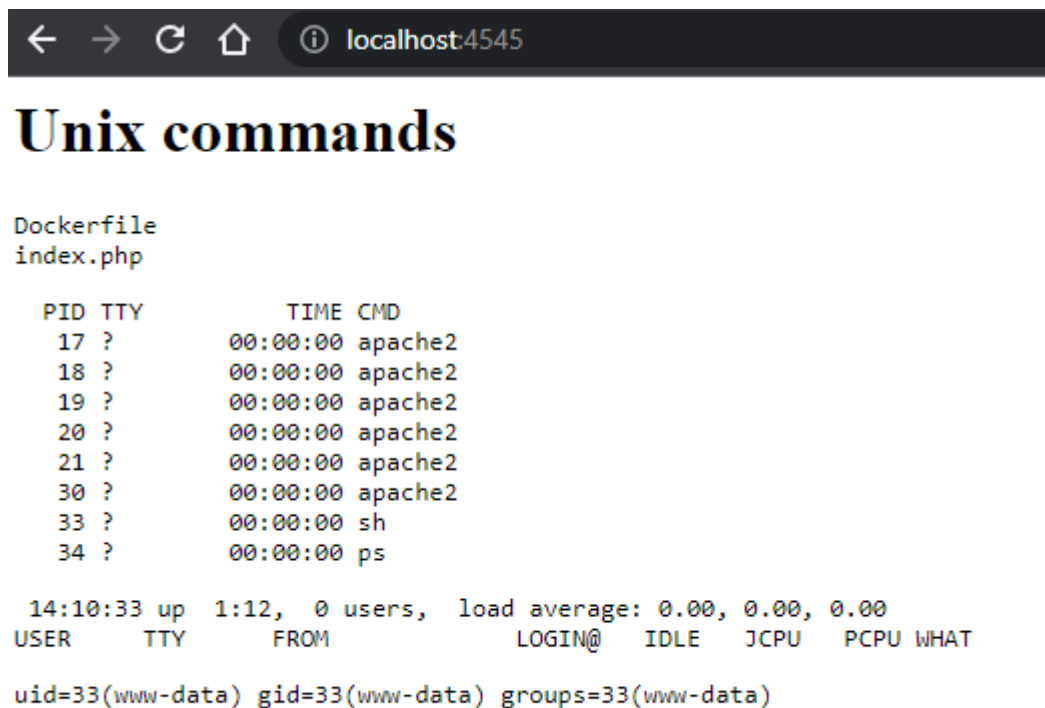


Рис. 5 – Результат

Вопросы:

1. Конфигурационный файл `php.ini`.

Файл конфигурации (`php.ini`) считывается при запуске PHP. Для версий серверных модулей PHP это происходит только один раз при запуске веб-сервера.

2. Как написать простой скрипт на `php`.
Создать `php` файл и внутри `<?php ?>` писать скрипт.
3. Основные правила, связанные с переменными в `php`.
 - a. Имя переменной чувствительно к регистру.
 - b. Правильное имя переменной должно начинаться с буквы или символа подчёркивания и состоять из букв, цифр и символов подчёркивания в любом количестве.
 - c. По умолчанию переменные всегда присваиваются по значению.

4. Основные типы данных в php.

- a. bool (логический тип)
- b. int (целые числа)
- c. float (дробные числа)
- d. string (строки)
- e. array (массивы)
- f. object (объекты)
- g. callable (функции)
- h. mixed (любой тип)
- i. resource (ресурсы)
- j. null (отсутствие значения)

5. Какие существуют функции для работы с переменными в php вне зависимости от типа данных.

- a. boolval — Возвращает логическое значение переменной
- b. debug_zval_dump — Выводит строковое представление внутренней структуры zval
- c. doubleval — Псевдоним floatval
- d. empty — Проверяет, пуста ли переменная
- e. floatval — Возвращает значение переменной в виде числа с плавающей точкой
- f. get_debug_type — Возвращает имя типа переменной в виде, подходящем для отладки
- g. get_defined_vars — Возвращает массив всех определённых переменных
- h. get_resource_id — Возвращает целочисленный идентификатор для данного ресурса
- i. get_resource_type — Возвращает тип ресурса
- j. gettype — Возвращает тип переменной
- k. intval — Возвращает целое значение переменной
- l. is_array — Определяет, является ли переменная массивом
- m. is_bool — Проверяет, является ли переменная булевой
- n. is_callable — Проверяет, что значение может быть вызвано как функция в текущей области видимости
- o. is_countable — Проверить, что содержимое переменной является счётным значением
- p. is_double — Псевдоним is_float
- q. is_float — Проверяет, является ли переменная числом с плавающей точкой
- r. is_int — Проверяет, является ли переменная целым числом
- s. is_integer — Псевдоним is_int

- t. `is_iterable` — Проверяет, является ли переменная итерируемой
- u. `is_long` — Псевдоним `is_int`
- v. `is_null` — Проверяет, является ли значение переменной равным `null`
- w. `is_numeric` — Проверяет, является ли переменная числом или строкой, содержащей число
- x. `is_object` — Проверяет, является ли переменная объектом
- y. `is_real` — Псевдоним `is_float`
- z. `is_resource` — Проверяет, является ли переменная ресурсом
- aa. `is_scalar` — Проверяет, является ли переменная скалярным значением
- bb. `is_string` — Проверяет, является ли переменная строкой
- cc. `isset` — Определяет, была ли установлена переменная значением, отличным от `null`
- dd. `print_r` — Выводит удобочитаемую информацию о переменной
- ee. `serialize` — Генерирует пригодное для хранения представление переменной
- ff. `settype` — Задаёт тип переменной
- gg. `strval` — Возвращает строковое значение переменной
- hh. `unserialize` — Создаёт PHP-значение из хранимого представления
- ii. `unset` — Удаляет переменную
- jj. `var_dump` — Выводит информацию о переменной
- kk. `var_export` — Выводит или возвращает интерпретируемое строковое представление переменной

6. Предопределенные переменные в php.

PHP предоставляет всем скриптам большое количество предопределённых переменных. Эти переменные содержат всё, от внешних данных до переменных среды окружения, от текста сообщений об ошибках до последних полученных заголовков.

Например - `$_GET` — Переменные HTTP GET

`$_POST` — Переменные HTTP POST

7. Переменные переменных в php.

Переменная переменной - имя переменной, которое может быть определено и изменено динамически. Обычная переменная в таком выражении:

```
<?php
```

```
$a = 'hello';
```

```
?>
```

Переменная переменной берет значение переменной и рассматривает его как имя переменной. В вышеприведённом примере `hello` может быть использовано как имя переменной при помощи двух знаков доллара. То есть:

```
<?php
$a = 'world';
?>
```

еперь в дереве символов PHP определены и содержатся две переменные: \$a, содержащая "hello" и \$hello, содержащая "world".

8. Выражения в php.

Выражение – это "все что угодно, имеющее значение". Основными формами выражений являются константы и переменные. Немного более сложными примерами выражений являются функции.

9. Арифметические операторы в php.

- +\$a Идентичность
- -\$a Отрицание
- \$a + \$b Сложение
- \$a - \$b Вычитание
- \$a * \$b Умножение
- \$a / \$b Деление
- \$a % \$b Деление по модулю
- \$a ** \$b Возведение в степень

10. Битовые операции в php.

- \$a & \$b И
- \$a | \$b Или
- \$a ^ \$b Исключающее или
- ~ \$a Отрицание
- \$a << \$b Сдвиг влево - Все биты переменной \$a сдвигаются на \$b позиций влево (каждая позиция подразумевает "умножение на 2")
- \$a >> \$b Сдвиг вправо - Все биты переменной \$a сдвигаются на \$b позиций вправо (каждая позиция подразумевает "деление на 2")

11. Оператор присваивания в php.

Базовый оператор присваивания обозначается как "=".

```
<?php
$a = ($b = 4) + 5; // $a теперь равно 9, а $b было присвоено 4.
?>
```

В дополнение к базовому оператору присваивания имеются "комбинированные операторы" для всех бинарных арифметических операций, операций объединения массивов и строковых операций, которые

позволяют использовать некоторое значение в выражении, а затем установить его как результат данного выражения.

12. Операторы сравнения в php.

- $\$a == \b - Равно true если $\$a$ равно $\$b$ после преобразования типов.
- $\$a === \b - Тожественно равно true если $\$a$ равно $\$b$ и имеет тот же тип.
- $\$a != \b - Не равно true если $\$a$ не равно $\$b$ после преобразования типов.
- $\$a <> \b - Не равно true если $\$a$ не равно $\$b$ после преобразования типов.
- $\$a !== \b - Тожественно не равно true если $\$a$ не равно $\$b$, или они разных типов.
- $\$a < \b - Меньше true если $\$a$ строго меньше $\$b$.
- $\$a > \b - Больше true если $\$a$ строго больше $\$b$.
- $\$a <= \b - Меньше или равно true если $\$a$ меньше или равно $\$b$.
- $\$a >= \b - Больше или равно true если $\$a$ больше или равно $\$b$.
- $\$a <=> \b - Космический корабль (spaceship) Число типа int меньше, больше или равное нулю, когда $\$a$ соответственно меньше, больше или равно $\$b$.

13. Логические операторы в php.

- $\$a \text{ and } \b - И
- $\$a \text{ or } \b - Или
- $\$a \text{ xor } \b - Исключающее или
- $! \$a$ - Отрицание.
- $\$a \&\& \b - И
- $\$a || \b - Или

14. Условная конструкция в php.

Конструкция if (условие) проверяет истинность некоторого условия, и если оно окажется истинным, то выполняется блок выражений, стоящих после if. Если же условие ложно, то есть равно false, тогда блок if не выполняется. Например:

```
<?php
$a = 4;
if($a>0){
    echo "Переменная а больше нуля";
}
echo "<br>конец выполнения программы";
```

?>

Блок выражений ограничивается фигурными скобками. И так как в данном случае условие истинно (то есть равно true): значение переменной \$a больше 0, то блок инструкций в фигурных скобках также будет выполняться. Если бы значение \$a было бы меньше 0, то блок if не выполнялся.

Если блок if содержит всего одну инструкцию, то можно опустить фигурные скобки:

```
<?php
$a = 4;
if($a>0)
echo "Переменная a больше нуля";
echo "<br>конец выполнения программы";
?>
```

Можно в одной строке поместить всю конструкцию:

```
if($a>0) echo "Переменная a больше нуля";
```

15. Циклы в php.

Циклы позволяют повторять определенное (и даже неопределенное - когда работа цикла зависит от условия) количество раз различные операторы. Данные операторы называются телом цикла. Проход цикла называется итерацией.

PHP поддерживает 4 вида циклов:

- Цикл с предусловием (while);
- Цикл с постусловием (do-while);
- Цикл со счетчиком (for);
- Специальный цикл перебора массивов (foreach).

При использовании циклов есть возможность использования операторов break и continue. Первый из них прерывает работу всего цикла, а второй - только текущей итерации.

16. Конструкции switch и match в php.

Конструкция switch..case является альтернативой использованию конструкции if..elseif..else. Оператор switch получает некоторое выражение и сравнивает его с набором значений.

После ключевого слова `switch` в скобках идет сравниваемое выражение. Значение этого выражения последовательно сравнивается со значениями, помещенными после операторов `case`. И если совпадение будет найдено, то будет выполняться определенный блок `case`.

Начиная с версии 8.0 в PHP была добавлена поддержка другой, похожей конструкции - `match`. Она позволяет оптимизировать конструкцию `switch`. Конструкция `match` также принимает некоторое выражение и сравнивает его с набором значений.

17. Include и require в php.

В PHP есть две функции, которые используются для помещения содержимого файла, содержащего исходный код PHP, в другой файл PHP. Это функции `Include()` и `Require()`. Обе функции одинаковы, но они имеют одно различие. Разница в том, что функция `include()` выдает предупреждение, но скрипт продолжит выполнение, а функция `require()` выдает предупреждение и фатальную ошибку, т.е. скрипт не будет продолжать выполнение. Эти две функции используются для помещения данных файла в другой файл PHP перед его выполнением сервером.

18. Функции в php.

Внутри функции можно использовать любой корректный PHP-код, в том числе другие функции и даже объявления классов.

Имена функций следуют тем же правилам, что и другие метки в PHP. Корректное имя функции начинается с буквы или знака подчёркивания, за которым следует любое количество букв, цифр или знаков подчёркивания. В качестве регулярного выражения оно может быть выражено так: `^[a-zA-Z_\x80-\xff][a-zA-Z0-9_\x80-\xff]*$`. Все функции и классы PHP имеют глобальную область видимости - они могут быть вызваны вне функции, даже если были определены внутри и наоборот.

Вывод

При выполнении текущей практической работы получили навыки работы с функциями, алгоритмами сортировки и использованием различных методов. Сделали конфигурацию из PHP и Apache.

Список использованных источников

1. Моуэт, Э. Использование Docker / Э. Моуэт ; научный редактор А. А. Маркелов ; перевод с английского А. В. Снастина. — Москва : ДМК Пресс, 2017. — 354 с. — ISBN 978-5-97060-426-7. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/93576> (дата обращения: 16.09.2021). — Режим доступа: для авториз. пользователей.
2. Алибеков, Б. И. Лабораторный практикум по Web-программированию на PHP : учебное пособие / Б. И. Алибеков. — Махачкала : ДГУ, 2018. — 273 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/158357> (дата обращения: 16.09.2021). — Режим доступа: для авториз. пользователей.
3. Джош, Л. Современный PHP. Новые возможности и передовой опыт / Л. Джош ; перевод с английского Р. Н. Рагимов. — Москва : ДМК Пресс, 2016. — 304 с. — ISBN 978-5-97060-184-6. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/93269> (дата обращения: 16.09.2021). — Режим доступа: для авториз. пользователей.
4. Одиночкина, С. В. Web-программирование PHP : учебно-методическое пособие / С. В. Одиночкина. — Санкт-Петербург : НИУ ИТМО, 2012. — 79 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/43562> (дата обращения: 29.09.2021). — Режим доступа: для авториз. пользователей.