



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»

Студент группы ИКБО24-20

Постнов Никита
Сергеевич

(подпись студента)

ПРИНЯЛ: старший преподаватель Волков М.Ю.

Работа представлена

«____» _____ 2022 г.

(подпись руководителя)

Допущен к работе

«____» _____ 2

СОДЕРЖАНИЕ

| | |
|---|----|
| 1. Цель работы | 3 |
| 1.1. Упражнение 1 | 3 |
| 1.2. Упражнение 2 | 3 |
| 1.1. Упражнение 3 | 3 |
| 2. Ход работы | 4 |
| 2.1. Упражнение 1 | 5 |
| 2.2. Упражнение 2 | 7 |
| 2.3. Упражнение 3 | 8 |
| 3. Ответы на вопросы к практической работе..... | 11 |
| 3.1. Конфигурационный файл <code>php.ini</code> | 11 |
| 3.2. Как написать простой скрипт на <code>php</code> | 11 |
| 3.3. Основные правила, связанные с переменными в <code>php</code> | 11 |
| 3.4. Основные типы данных в <code>php</code> | 11 |
| 3.5. Какие существуют функции для работы с переменными в <code>php</code> вне зависимости от типа данных | 11 |
| 3.6. Предопределенные переменные в <code>php</code> | 11 |
| 3.7. Переменные переменных в <code>php</code> | 11 |
| 3.8. Выражения в <code>php</code> | 11 |
| 3.9. Арифметические операторы в <code>php</code> | 11 |
| 3.10. Битовые операции в <code>php</code> | 11 |
| 3.11. Оператор присваивания в <code>php</code> | 11 |
| 3.12. Операторы сравнения в <code>php</code> | 11 |
| 3.13. Логические операторы в <code>php</code> | 11 |
| 3.14. Условная конструкция в <code>php</code> | 11 |
| 3.15. Циклы в <code>php</code> | 11 |
| 3.16. Конструкции <code>switch</code> и <code>match</code> в <code>php</code> | 11 |
| 3.17. <code>Include</code> и <code>require</code> в <code>php</code> | 11 |
| 3.18. Функции в <code>php</code> | 11 |
| 4. Ссылка на удаленный репозиторий проекта | 12 |
| ВЫВОД..... | 12 |

1. 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. Предполагается создание 1 общего проекта с разделенными сервисами, разделением проекта на файлы для разделения функционала и переиспользования файлов. Каждый сервис должен состоять как минимум из 2 файлов.

1.1. Упражнение 1

Предлагается создать веб-сервис Drawer для рисования svg объектов. Кодирование фигуры состоит из нескольких параметров: форма, цвет, размеры ограничивающего прямоугольника примитива. Должен передаваться всего 1 целочисленный параметр, который будет содержать всю информацию о фигуре, которую требуется нарисовать.

1.2. Упражнение 2

Реализовать сортировку вставками на языке программирования РНР. Массив передается скрипту как параметр строка: состоящий из значений элементов массива, разделенных запятыми. Итогом является веб-страница, содержащая отсортированный массив.

1.3 Упражнение 3

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

2. Ход работы

Для облегчения работы с рекомендуемыми инструментами используются ранее написанные скрипт инициализации БД для СУБД MySQL и скрипт генерации тестовой страницы вместе с оформлением на языке PHP.

Данные файлы были помещены в папку “pr2” и созданы файлы “drawer.php”, “sort.php”, “unix.php”, файл “index.php” был отредактирован, как домашняя страница с навигационными элементами по другим файлам php.

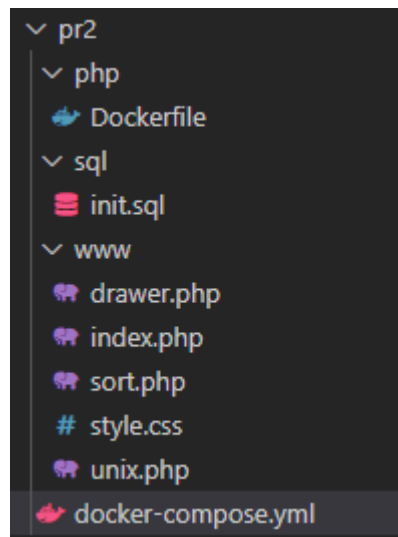


Рисунок 2.1 – Папка “pr2”

Dockerfile был настроен для создания образа запуска базового php проекта с поддержкой MySQL (рис 2.2).

```
FROM php:7.2-apache
RUN apt-get update && docker-php-ext-install mysqli
```

Листинг 2.2 – Файл Dockerfile

Для совместной работы контейнера сервера и контейнера базы данных был создан файл “docker-compose.yml” (рис 2.3). Для контейнера базы данных используется готовый образ MariaDB.

```
version: '3' # версия Докера
services: # контейнеры которые мы хотим запустить.
```

```

  php-apache: # имя сервиса, может быть любым, но желательно, чтобы оно
описывало суть этого контейнера.
  build: # шаги, описывающие процесс билдинга.
    ./php # где находится Dockerfile, из которого будем билдить образ
для контейнера.
  ports: # маппинг портов основной ОС к контейнеру.
    - 9000:80
  volumes:
    - ./www:/var/www/html # линкуем папку с index.php и style.css с
папкой в виртуальной машине.
  depends_on:
    - database # запустится после базы данных.

database:
  image: mariadb:latest # образ mysql.
  restart: always # пуолитика перезагрузок
  volumes:
    - "./sql:/docker-entrypoint-initdb.d" # линкуем файл инициализации
бд к файлу запуска.
  environment:
    MARIADB_ROOT_PASSWORD: password # пароль рут пользователя

```

Листинг 2.3 – Файл docker-compose.yml

С помощью команды “docker-compose up --build” можно создать и запустить контейнеры. Затем можно проверить работоспособность сервера и базы данных перейдя по ссылке “localhost:9000” (рис 2.4 - 2.5).

2 Упражнение 1

Для выполнения этого упражнения был создан “driwer.php”, результат работы и код которого представлен ниже:

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="author" content="Сироткин Денис">
  <meta name="description" content="Практика 2">
  <meta name="keywords" content="PCЧИР, php, HTML, CSS">
  <link rel="stylesheet" href="style.css">
  <title>Drawer</title>
</head>
<body>
  <div>
    <?php
    $num = $_GET['num'];
    if (isset($num) && is numeric($num)) {

```

```

echo 'Число: ' . $num . ' Двоичный вид: '
    . sprintf("%07d", decbin(strval($num))) . '<br>';
// 00    0 0 0  00
// Shape R G B  Size
$shape = ($num >> 5) & 3; // 00-круг 01-прямоуг 10-квадр
11-треуг

$r = ($num >> 4) & 1;
$g = ($num >> 3) & 1; // RGB
$b = ($num >> 2) & 1;
$size = (($num >> 0) & 3) + 1; // 00-мал 01-сред 10-бол 11-
очбол

// HEX цвет
$color = '#'
    . ($r == 1 ? 'ff' : "00")
    . ($g == 1 ? 'ff' : "00")
    . ($b == 1 ? 'ff' : "00") . '';

// Увеличение размера
$size = $size * 100;

$shape_tag = '';
switch ($shape) {
    case 0: // Круг
        $radius = ($size / 2);
        $shape_tag = "circle "
            // Размер
            . " cx=" . ($radius + 10) . " cy=" . ($radius +
10)

            // Радиус
            . " r=" . $radius . " ";
        break;
    case 1: // Прямоугольник
        $shape_tag = "rect "
            // Размер
            . "width=" . ($size * 2) . " height=" . ($size);
        break;
    case 2: // Квадрат
        $shape_tag = "rect "
            // Размер
            . "width=" . ($size) . " height=" . ($size);
        break;
    case 3: // Треугольник
        $side = $size;
        $shape_tag = "polygon points="
            // Точки
            . ($side / 2 + 5) . ",10"
            . " 10," . ($side) . " "
            . ($side) . "," . ($side) . " ";
        break;
}
echo '<svg>';

```

```

        echo '<' . $shape_tag . ' fill=' . $color . ' />';
        echo '</svg>';
    } else {
        echo "<p>Команда должна быть введена в формате: ?num=(Ваша
команда)</p>";
        echo "<p>Например: ?num=1</p>";
        echo '<a href="http://localhost:9000/drawer.php?
num=1">Пример</a>';
    }
    ?>
</div>
</body>
</html>

```

Листинг 2.1.1 – Файл driver.php

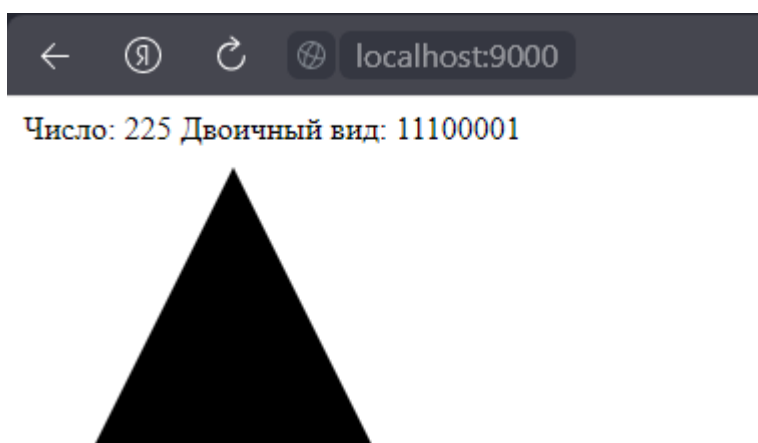


Рисунок 2.1.2 – Вывод файла driver.php

2.2 Упражнение 2

Для выполнения этого упражнения был создан “sort.php”, результат работы и код которого представлен ниже:

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="author" content="Сироткин Денис">
    <meta name="description" content="Практика 2">
    <meta name="keywords" content="PCЧП, php, HTML, CSS">
    <link rel="stylesheet" href="style.css">
    <title>Сортировка выбором</title>
</head>
<body>
    <div>

```

```

<?php
    //Сортировка выбором
    function selectSort($arr) {
        for($i = 0; $i < count($arr)-1; $i++){
            $minIndex = $i;
            for($j = $i + 1; $j < count($arr); $j++) {
                if ($arr[$j] < $arr[$minIndex]) {
                    $minIndex = $j;
                }
            }
            $temp = $arr[$i];
            $arr[$i] = $arr[$minIndex];
            $arr[$minIndex] = $temp;
        }
        return $arr;
    }

    if (isset($_GET['array'])) {
        // Массив
        echo "<p>Массив: [";
        echo implode(", ", explode(",", $_GET["array"]));

        // Отсортированный массив
        echo "]\<p>\n<p>Отсортированный массив: [";
        echo implode(", ", selectSort(explode(",",
$_GET["array"])));
        echo "]\<p>";
    } else {
        echo "<p>Отсутствуют данные в параметре строки</p>";
        echo '<a href="http://localhost:9000/sort.php?
array=5,4,3,2,1">Пример</a>';
    }

    ?>
</div>
</body>
</html>

```

Листинг 2.2.1 – Файл sort.php

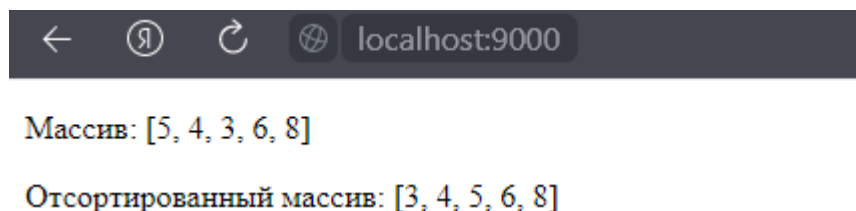


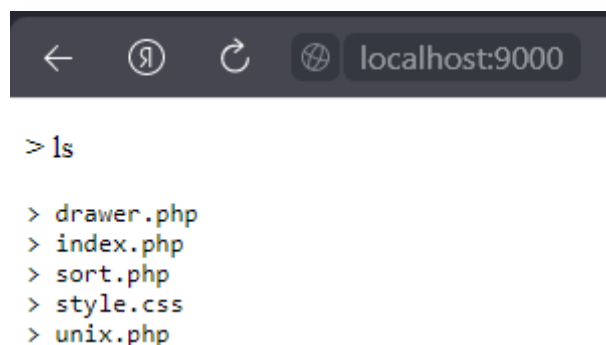
Рисунок 2.2.2 – Вывод файла sort.php

2 Упражнение 3

Для выполнения этого упражнения был создан “unix.php”, результат работы и код которого представлен ниже:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="author" content="Сироткин Денис">
  <meta name="description" content="Практика 2">
  <meta name="keywords" content="PCЧИР, php, HTML, CSS">
  <link rel="stylesheet" href="style.css">
  <title>Drawer</title>
</head>
<body>
  <div>
    <?php
      function print_cmd($cmd) {
        $lines = array();
        exec($cmd, $lines);
        echo "<p>> $cmd </p>";
        echo "<pre>> " . implode("\n> ", $lines) . "</pre>";
      }
      // Список команд
      $command = $_GET["cmd"];
      $command_list = array('ps', 'ls', 'whoami', 'id', 'echo');
      if (in_array(explode(" ", $_GET["cmd"])[0], $command_list)){
        print_cmd($command);
      } else {
        echo "<p>Команда должна быть введена в формате: ?cmd= (Ваша команда)</p>";
        echo "<p>Например: ?cmd=ls</p>";
        echo "<p>Список доступных команд: 'ps', 'ls', 'whoami', 'id', 'echo (ваш текст)'</p>";
        echo '<a href="http://localhost:9000/unix.php?cmd=ls">Пример</a>';
      }
    <?>
  </div>
</body>
</html>
```

Листинг 2.2.1 – Файл unix.php



```
<  ↻  ↺  🌐  localhost:9000

> ls

> drawer.php
> index.php
> sort.php
> style.css
> unix.php
```

Рисунок 2.2.2 – Вывод файла unix.php

3. Ответы на вопросы к практической работе

3.1 Конфигурационный файл php.ini.

В конфигурационном файле сосредоточены настройки интерпретатора PHP и его многочисленных расширений. Содержимое данного файла представляет собой секции и директивы. Секции заключаются в квадратные скобки, после них идут директивы формата “directive = value”. Комментариями являются строки, начинающиеся с точки запятой. Возможности настройки данного файла обширны: языковые опции, ограничение ресурсов, настройка производительности, настройка обработки данных, настройка путей и директорий загрузки файлов и тому подобное.

3.2 Как написать простой скрипт на php.

Весь код обернут в тег `<?php`, а сам код представляет собой оператор вывода `echo` и строку “Hello World”, которую данный оператор собственно выводит на экран. Важно отметить, что как самостоятельный код или как вставка в html-код скрипт PHP всегда оборачивается в тег `<?php`. В первом случае конец тега можно опустить. Данный синтаксис работает только внутри тега `<?php`. Вне его работают правила HTML.

Также следует упомянуть еще раз, что, если файл содержит только код PHP, предпочтительно опустить закрывающий тег в конце файла. Это помогает избежать добавления случайных символов пробела или перевода строки после закрывающего тега PHP, которые могут

послужить причиной нежелательных эффектов, так как PHP начинает выводить данные в буфер при отсутствии намерения у программиста выводить какие-либо данные в этой точке скрипта.

3.3 Основные правила, связанные с переменными в php.

Основные правила, связанные с переменными в языке PHP:

1. Объявление переменной начинается со знака \$. Данная особенность языка облегчает интерпретатору выделение переменных в тексте.
2. Имена переменных должны состоять из латинских букв, цифр и нижнего подчеркивания. Хотя данное правило не является обязательным, рекомендуется его соблюдать.
3. Имена переменных чувствительны к регистру.

3.4 Основные типы данных в php.

PHP поддерживает десять основных типов переменных:

- 1) integer: целое число со знаком, размер зависит от разрядности системы;
- 2) double: вещественное число, число с плавающей точкой, отводится 8 байт. Для обозначения бесконечности используется INF, а для обозначения несуществующего числа NAN;
- 3) boolean: логическая переменная с двумя возможными состояниями true и false;
- 4) string: набор символов;
- 5) array: ассоциативный массив. В PHP нет обычного классического массива, как в других языках программирования, здесь массив представляет набор элементов, представляющих пары ключ => значение. Ключами могут быть как целые числа, так и строки;
- 6) object: ссылка на объект, который реализует несколько принципов ООП;
- 7) resource: некоторый ресурс, обрабатываемый языком особым образом;

8) `null`: специальное значение;

9) `callable`: нововведение версии 5.4. Этим типом является функция обратного вызова. Данный подход используется, когда в функцию нужно передать другую функцию, которая в этом случае и называется функцией обратного вызова;

10) `iterable`: псевдотип, введенный в PHP 7.1. Он принимает любой массив или объект, реализующий интерфейс `Traversable`. Используется как тип параметра для указания, что функция принимает набор значений, но ей не важна форма этого набора, пока он будет использоваться с `foreach`.

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

Первой важной функцией является `gettype`. Эта функция возвращает тип, переданной ей переменной. Так как в PHP нет жесткого задания типа переменной, а также применяется динамическая типизация, обозначающая связывание типа переменной с 30 ней при присвоении ей значения, она важна.

Второй необходимой функцией является `is_type`. Она возвращает ИСТИНУ или ЛОЖЬ в зависимости от того, принадлежит переменная типу `type` или нет.

Третьей важной функцией является `isset`. Она возвращает ИСТИНУ в случае, если переменная задана каким-либо значением. Чаще всего данная функция используется для проверки существования ключа в массиве.

Следующая необходимой функцией является функция `settype`, которая отвечает за преобразование типов. На вход ей подаются ссылка на переменную и тип данных к которой эту переменную нужно преобразовать.

Еще одним важным инструментом при написании PHP-скриптов является функция `unset`. Данная функция удаляет переменную.

Последняя это `var_dump`. Результатом данной функции является удобно воспринимаемое человеком представление того, что ей передано с указанием типа и значения. Данная функция полезна в процессе отладки PHP-скриптов.

3.6Предопределенные переменные в php.

Возвращаясь к теме переменных в PHP, следует упомянуть существование предопределенных переменных. Любому запускаемому скрипту PHP предоставляет большое количество предопределенных переменных. Однако многие из этих переменных не могут быть полностью задокументированы, поскольку они зависят от запускающего скрипт сервера, его версии и настроек, а также других факторов. Некоторые из этих переменных недоступны, когда PHP запущен из командной строки. PHP предоставляет дополнительный набор предопределенных массивов, содержащих переменные сервера, если они доступны, окружения и пользовательского ввода. Эти массивы являются особыми, поскольку они становятся глобальными автоматически, то есть автоматически доступны в любой области видимости. По этой причине они также известны как “автоглобальные” или “суперглобальные” переменные.

3.7Переменные переменных в php.

То есть объявляется переменная с динамическим именем, хранимым в переменной \$a. Переменная переменной берет значение переменной и рассматривает его как имя переменной.

3.8Выражения в php.

Выражения – это самые важные строительные элементы PHP. Почти всё, что пишется на PHP, является выражением. Самое простое и точное определение выражения – "все, что угодно, имеющее значение". Самым простым примером выражения является переменная или константа в операторе присваивания.

3.9Арифметические операторы в php.

Всего существует 8 основных арифметических операторов, записываемых с помощью 5 символов.

Первый оператор – это оператор идентичности, он отвечает за конвертацию в целочисленное или в значение с плавающей точкой в зависимости от того, что больше подходит.

Вторым оператором является оператор смены знака. Следует заметить, что возможны арифметические операции с числами в строковом формате, если возможно их преобразование.

Операторы сложения, вычитания и умножения работают по правилам математики, отсюда же идут правила их приоритета.

Операция деления работает в зависимости от результата этого деления. Если делится нацело, то возвращается целое число, иначе число с плавающей точкой.

Про оператор вычисления остатка от деления нужно помнить, то что он преобразует операнды к целым числам путем отбрасывания дробной части.

Также наряду с классическими арифметическими операторами в PHP присутствует оператор возведения в степень, который работает, как с целыми, так и с числами с плавающей точкой.

3.10 Битовые операции в php.

Вторыми базовыми операциями являются битовые операции, представленные широким спектром в PHP. Побитовые операторы позволяют считывать и устанавливать конкретные биты целых чисел. Классические битовые операторы И, ИЛИ, ИСКЛЮЧАЮЩЕЕ ИЛИ, ОТРИЦАНИЕ и конечно же БИТОВЫЙ СДВИГ.

3.11 Оператор присваивания в php.

Отдельным разделом рассматриваются возможные операции с оператором присваивания. Основным является оператор простого присваивания “=”. Оператор присваивания означает, что левый операнд получает значение правого выражения, то есть устанавливается значением. Результатом выполнения оператора присваивания является само присвоенное значение. Таким образом, результат выполнения “\$a = 3” будет равен 3. 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.

3.12 Операторы сравнения в php.

| Название | Обозначение | Описание |
|------------------------------------|---------------|--|
| Равно | $\$a == \b | true если \$a равно \$b после преобразования типов. |
| Тождественно равно | $\$a === \b | true если \$a равно \$b и имеет тот же тип. |
| Не равно | $\$a != \b | true если \$a не равно \$b после преобразования типов. |
| Не равно | $\$a <> \b | true если \$a не равно \$b после преобразования типов. |
| Тождественно не равно | $\$a !== \b | true если \$a не равно \$b, или они разных типов. |
| Меньше | $\$a < \b | true если \$a строго меньше \$b. |
| Больше | $\$a > \b | true если \$a строго больше \$b. |
| Меньше или равно | $\$a <= \b | true если \$a меньше или равно \$b. |
| Больше или равно | $\$a >= \b | true если \$a больше или равно \$b. |
| Космический корабль (spaceship) | | |

$\$a \leq \b

Число типа `int` меньше, больше или равно

нулю, когда \$a\$
соответственно
меньше, больше или
равно \$b\$.

3.13 Логические операторы в php.

| Название | Обозначение | Описание |
|-----------------|-------------|--|
| И | \$a and \$b | true, если и \$a, и \$b true. |
| И | \$a && \$b | true, если и \$a, и \$b true. |
| ИЛИ | \$a or \$b | true, если или \$a, или \$b true. |
| ИЛИ | \$a \$b | true, если или \$a, или \$b true. |
| Отрицание | ! \$a | true, если \$a не true. |
| Исключающее ИЛИ | \$a xor \$b | true, если \$a, или \$b true, но не оба. |

3.14 Условная конструкция в php.

Данная конструкция предоставляет возможность условного выполнения фрагментов кода. Структура if реализована в PHP по аналогии с языком C. Нужно понимать, что выражение else выполняется только, если выражение if вычисляется как false, и если нет других любых выражений elseif, или если они все равны false. Конструкция elseif, есть сочетание конструкций if и else. Аналогично else, она расширяет оператор if для выполнения различных выражений в случае, когда условие начального оператора if эквивалентно false. Однако, в отличие от else, выполнение альтернативного выражения произойдет только тогда, когда условие оператора elseif будет являться равным true.

3.15 Циклы в php.

Для общего понимания введем следующее определение: Цикл – разновидность управляющей конструкции в высокоуровневых языках программирования, предназначенная для организации многократного исполнения набора инструкций.

While

Циклы `while` являются простейшим видом циклов в PHP. Они ведут себя так же, как и в языке C. Данная конструкция предполагает циклическое выполнение какого-то участка кода с предварительной проверкой условия на необходимость выполнения.

Do-while

Цикл `do-while` очень похож на цикл `while`, с тем отличием, что истинность выражения проверяется в конце итерации, а не в начале. Главное отличие от обычного цикла `while` в том, что первая итерация цикла `do-while` гарантированно выполнится (истинность выражения проверяется в конце итерации), тогда как она может не выполниться в обычном цикле `while` (истинность выражения которого проверяется в начале выполнения каждой итерации, и если изначально имеет значение `false`, то выполнение цикла будет прервано сразу).

Универсальный цикл for

Данный тип цикла чаще всего используется для перебора значений счетчика и каких-либо действий с использованием данного счетчика. Данный же цикл назван универсальным, т.к. предполагает создавать не такие тривиальные действия.

Инструкции break и continue

Инструкции `break` и `continue` знакомы по практически любому другому языку программирования, используемые для более тонкого управления циклами.

Foreach

В PHP существует и особый тип цикла для перебора элементов массива и итерируемых объектов. Следует напомнить, что массив есть встроенный тип для PHP и представляет во всех случаях набор пар ключ-значение.

3.16 Конструкции switch и match в php.

Switch

В ситуации сложных и многоуровневых условных конструкций на помощь приходит конструкция `switch-case`. Данная конструкция используется, когда требуется сравнивать одну и ту же переменную (или выражение) с множеством различных значений и выполнять различные участки кода в зависимости от того, какое значение принимает эта переменная (или выражение).

Match

В версии PHP 8.0.0 появилась новая конструкция для ветвления потока исполнения на основании проверки совпадения значения с заданным условием. Конструкция аналогична `switch`, но имеет некоторые существенные различия, которые способствуют применению данной конструкции. Главным отличием от `switch` является возвращение результата, а также привязка сравнения также к типам.

3.17 Include и require в php.

Идея разделения программы на несколько файлов очень важна для переиспользования кода, создания шаблонов и т.д. Первое, о чем нужно сказать в данном разделе это задание путей поиска файлов для PHP. Это задается директивой `include_path` в файле `php.ini`, указывается список директорий, в которых функции `require`, `include`, `fopen()`, `file()`, `readfile()` и `file_get_contents()` ищут файлы. Формат соответствует формату системной переменной окружения `PATH`: список директорий, разделенных двоеточием в Unix или точкой с запятой в Windows. Возможно добавление в ходе выполнения сценария еще какой-либо директории с помощью функции `set_include_path()`. Для того, чтобы не было ошибок повторения существуют выражения `require_once` и `include_once`. Эти инструкции второй раз не срабатывают в одном файле или при более сложной иерархии.

3.18 Функции в php.

Для понимания синтаксиса описания функций и их использования нужно понимать, что:

1) у функции может быть переменное количество параметров и параметры по умолчанию;

2) функция имеет собственную область видимости. Это нужно учитывать при работе с глобальными переменными программы и с локальными переменными функции, которые уничтожаются при окончании ее работы;

3) в PHP тип возвращаемого значения может быть любым и тип возвращаемого значения не регламентируется. То есть функция может возвращать и число, и строку одновременно;

4) допускается создание и использование при необходимости анонимных функций;

5) в пределах одного сценария не должно быть определений двух функций с одинаковыми именами.

4. Ссылка на удаленный репозиторий проекта

Полный код проекта можно найти по ссылке:

<https://github.com/StormCrownSC/RSCIR>

ВЫВОД

Были созданы php файлы, выполняющие поставленные задачи. Реализован функционал рисования svg фигур, сортировки и выполнения стандартных Unix команд. Также составлена главная страница, отвечающая за навигацию между ними.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Разработка серверных частей интернет-ресурсов [Электронный ресурс].

URL: https://online-edu.mirea.ru/pluginfile.php?file=%2F980342%2Fmod_resource%2Fcontent%2F0%2Fpdf24_merged.pdf

Дата обращения 22.09.2022

2. Руководство по PHP

URL: <https://www.php.net/manual/ru/index.php> Дата обращения 22.09.2022

3. “Документация Docker” [Электронный ресурс]. URL: <https://docs.docker.com/> (дата обращения 20.09.2022)