

|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА**  **Российский технологический университет»**  **РТУ МИРЭА** |

**Институт информационных технологий (ИИТ)**

**Кафедра инструментального и прикладного программного обеспечения (ИиППО)**

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ

по дисциплине «Разработка серверных частей интернет ресурсов»

Практическое задание № 2

|  |  |
| --- | --- |
| Студента группы | ИКБО-03-20 Голиков Александр |
| (подпись) |
| Старший преподаватель | Волков М. Ю. |
|  | (подпись) |
| Отчет представлен | « » 2022 г. |

Москва 2022 г

**СОДЕРЖАНИЕ**

1. [Цель работы 3](#_bookmark0)
2. [Ход работы 4](#_bookmark1)
3. [Выводы 8](#_bookmark2)
4. [Ответы на вопросы 9](#_bookmark3)

# 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.

# Ход работы

# Упражнение 1

Создам файл drawer.php в котором в зависимости от входного параметра будет суммироваться количество байтов в числе и создаваться фигура (рис. 1).

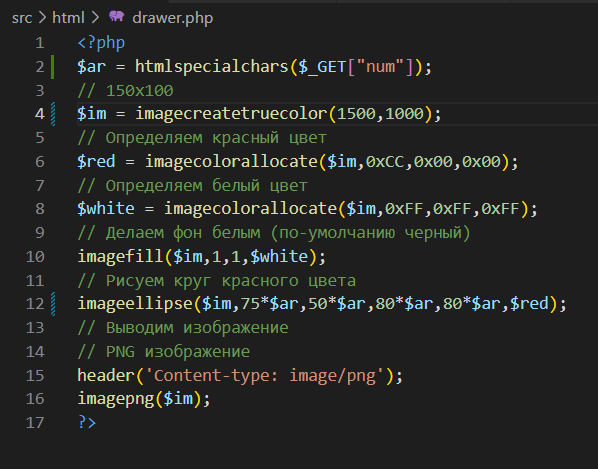


Рисунок 1 – Код drawer.php

От входного параметра зависит тип фигуры, а также ее размеры и размеры толщины границы.

Пример работы (Рисунок 2)

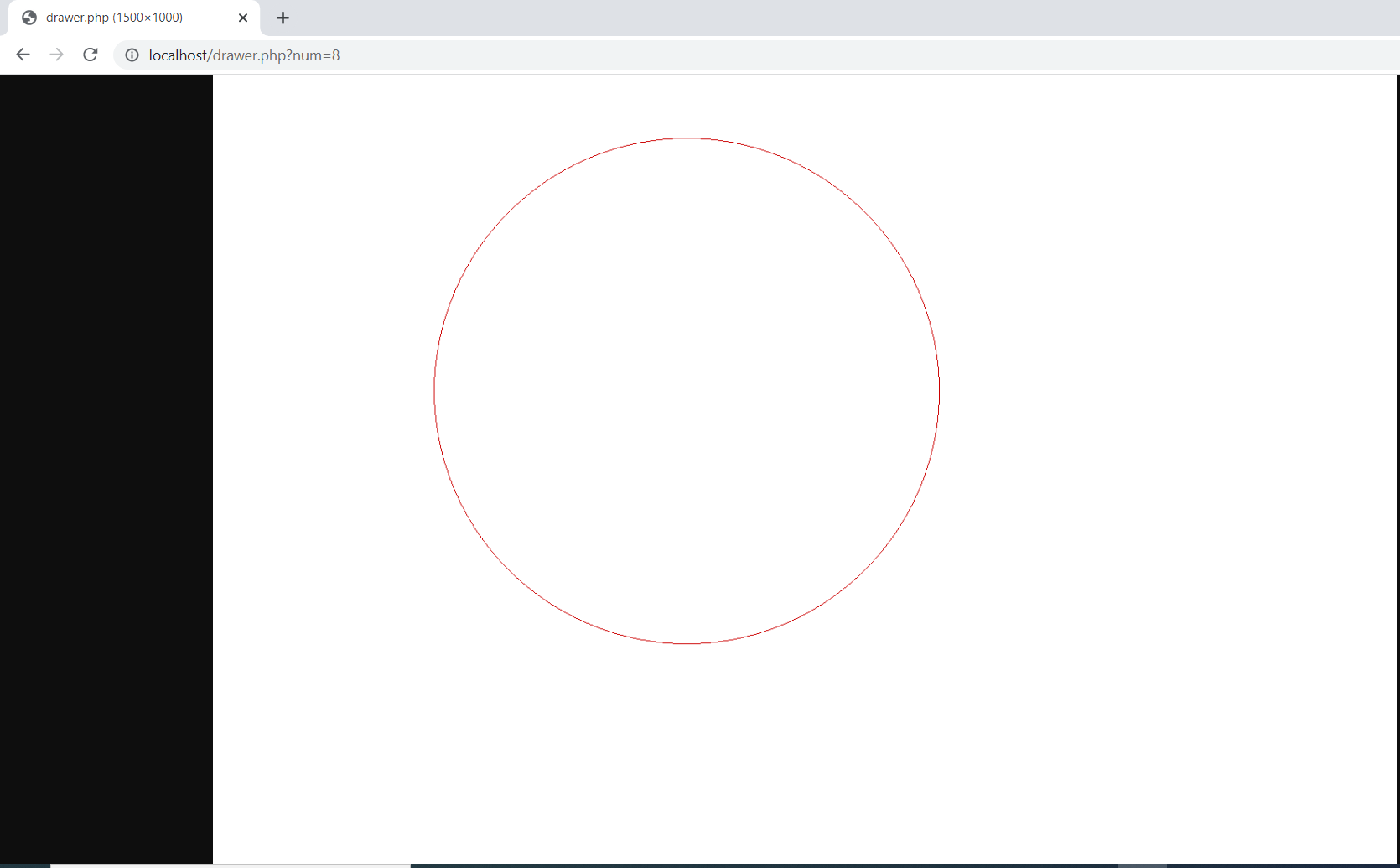


Рисунок 2 – Пример работы drawer.php

## Упражнение 2

В файле sorted.php будет реализована сортировка слиянием входного массива (Рисунок 3.1, Рисунок 3.2)



Рисунок 3.1 – Код sorted.php

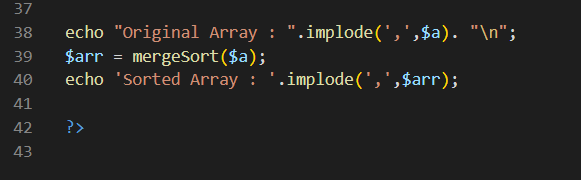


Рисунок 3.2 – Код sorted.php

Пример работы (Рисунок 4)

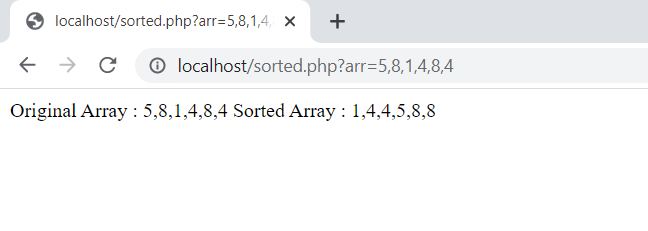


Рисунок 4 – Пример работы sorted.php

## Упражнение 3

В файле shell.php (Рисунок 5) была реализована логика командной строчки, которая в параметрах запроса получает вводимую команду. В целях безопасности некоторые команды были заблокированы.

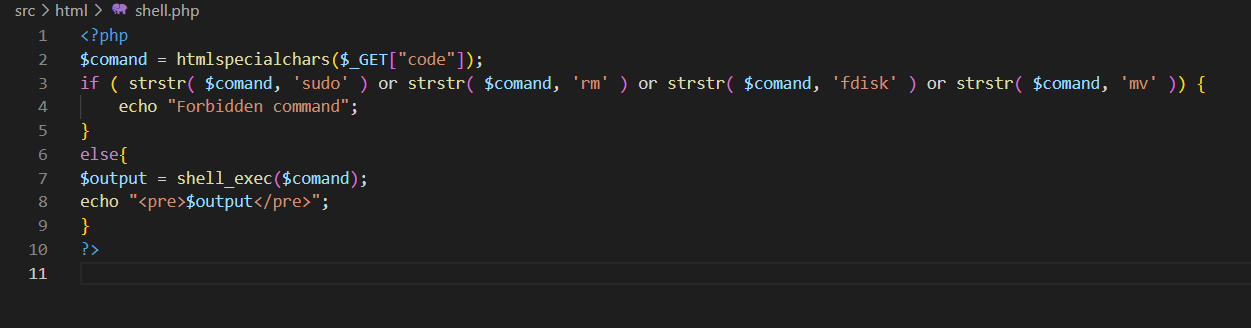


Рисунок 5 – Файл shell.php

Пример работы представленного кода (Рисунок 6-7)

Рисунок 6-7 – Пример работы shell.php

Файл Dockerfile содержит установку требуемых для выполнения упражнений библиотек (Рисунок 8).

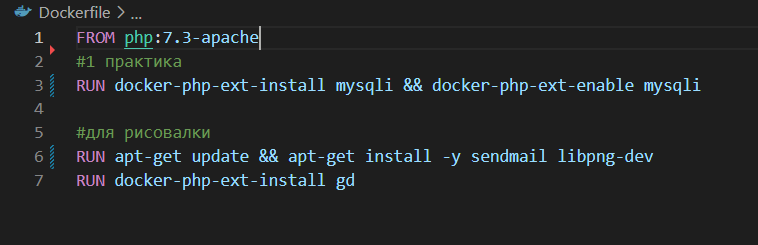


Рисунок 8 – Файл Dockerfile

# 

# Выводы

В ходе работы была создана конфигурация серверного программного обеспечения, состоящего из:

1. Веб-сервера Apache;
2. СУБД MySQL;
3. Языка программирования PHP.

А также были приобретены знания в области создания картинок, эмуляторов командной строки, а также вывода отсортированных последовательностей

# Ответы на вопросы к практической работе

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. Для CGI и CLI версий это происходит при каждом вызове.

1. ***Как написать простой скрипт на php.***

Необходимо создать файл с расширением *.php* в корневом каталоге веб-сервера (*DOCUMENT\_ROOT*) и запишите в него классическую стркутуру HTML. Для того чтобы использовать PHP необходимо заключить код внутри тега <?PHP?>.

1. ***Основные правила, связанные с переменными в php.***

Правила для переменных PHP:

* Переменная начинается с знака $, за которым следует имя переменной
* Имя переменной должно начинаться с буквы или символа подчеркивания
* Имя переменной не может начинаться с числа
* Имя переменной может содержать только буквенно-цифровые символы и знаки подчеркивания (a-z, 0-9 и \_)

***4. Основные типы данных в php***

В PHP есть десять базовых типов данных:

* bool (логический тип)
* int (целые числа)
* float (дробные числа)
* string (строки)
* array (массивы)
* object (объекты)
* callable (функции)
* mixed (любой тип)
* resource (ресурсы)
* null (отсутствие значения)

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

* [boolval](https://www.php.net/manual/ru/function.boolval.php) — Возвращает логическое значение переменной
* [debug\_zval\_dump](https://www.php.net/manual/ru/function.debug-zval-dump.php) — Выводит строковое представление внутренней структуры zval
* [doubleval](https://www.php.net/manual/ru/function.doubleval.php) — Псевдоним floatval
* [empty](https://www.php.net/manual/ru/function.empty.php) — Проверяет, пуста ли переменная
* [floatval](https://www.php.net/manual/ru/function.floatval.php) — Возвращает значение переменной в виде числа с плавающей точкой
* [get\_debug\_type](https://www.php.net/manual/ru/function.get-debug-type.php) — Возвращает имя типа переменной в виде, подходящем для отладки
* [get\_defined\_vars](https://www.php.net/manual/ru/function.get-defined-vars.php) — Возвращает массив всех определённых переменных
* [get\_resource\_id](https://www.php.net/manual/ru/function.get-resource-id.php) — Возвращает целочисленный идентификатор для данного ресурса
* [get\_resource\_type](https://www.php.net/manual/ru/function.get-resource-type.php) — Возвращает тип ресурса
* [gettype](https://www.php.net/manual/ru/function.gettype.php) — Возвращает тип переменной
* [intval](https://www.php.net/manual/ru/function.intval.php) — Возвращает целое значение переменной
* [is\_array](https://www.php.net/manual/ru/function.is-array.php) — Определяет, является ли переменная массивом
* [is\_bool](https://www.php.net/manual/ru/function.is-bool.php) — Проверяет, является ли переменная булевой
* [is\_callable](https://www.php.net/manual/ru/function.is-callable.php) — Проверяет, что значение может быть вызвано как функция в текущей области видимости
* [is\_countable](https://www.php.net/manual/ru/function.is-countable.php) — Проверить, что содержимое переменной является счётным значением
* [is\_double](https://www.php.net/manual/ru/function.is-double.php) — Псевдоним is\_float
* [is\_float](https://www.php.net/manual/ru/function.is-float.php) — Проверяет, является ли переменная числом с плавающей точкой
* [is\_int](https://www.php.net/manual/ru/function.is-int.php) — Проверяет, является ли переменная целым числом
* [is\_integer](https://www.php.net/manual/ru/function.is-integer.php) — Псевдоним is\_int
* [is\_iterable](https://www.php.net/manual/ru/function.is-iterable.php) — Проверяет, является ли переменная итерируемой
* [is\_long](https://www.php.net/manual/ru/function.is-long.php) — Псевдоним is\_int
* [is\_null](https://www.php.net/manual/ru/function.is-null.php) — Проверяет, является ли значение переменной равным null
* [is\_numeric](https://www.php.net/manual/ru/function.is-numeric.php) — Проверяет, является ли переменная числом или строкой, содержащей число
* [is\_object](https://www.php.net/manual/ru/function.is-object.php) — Проверяет, является ли переменная объектом
* [is\_real](https://www.php.net/manual/ru/function.is-real.php) — Псевдоним is\_float
* [is\_resource](https://www.php.net/manual/ru/function.is-resource.php) — Проверяет, является ли переменная ресурсом
* [is\_scalar](https://www.php.net/manual/ru/function.is-scalar.php) — Проверяет, является ли переменная скалярным значением
* [is\_string](https://www.php.net/manual/ru/function.is-string.php) — Проверяет, является ли переменная строкой
* [isset](https://www.php.net/manual/ru/function.isset.php) — Определяет, была ли установлена переменная значением, отличным от null
* [print\_r](https://www.php.net/manual/ru/function.print-r.php) — Выводит удобочитаемую информацию о переменной
* [serialize](https://www.php.net/manual/ru/function.serialize.php) — Генерирует пригодное для хранения представление переменной
* [settype](https://www.php.net/manual/ru/function.settype.php) — Задаёт тип переменной
* [strval](https://www.php.net/manual/ru/function.strval.php) — Возвращает строковое значение переменной
* [unserialize](https://www.php.net/manual/ru/function.unserialize.php) — Создаёт PHP-значение из хранимого представления
* [unset](https://www.php.net/manual/ru/function.unset.php) — Удаляет переменную
* [var\_dump](https://www.php.net/manual/ru/function.var-dump.php) — Выводит информацию о переменной
* [var\_export](https://www.php.net/manual/ru/function.var-export.php) — Выводит или возвращает интерпретируемое строковое представление переменной

6***. Предопределенные переменные в php.***

* [Суперглобальные переменные](https://www.php.net/manual/ru/language.variables.superglobals.php) — Встроенные переменные, которые всегда доступны во всех областях
* [$GLOBALS](https://www.php.net/manual/ru/reserved.variables.globals.php) — Ссылки на все переменные глобальной области видимости
* [$\_SERVER](https://www.php.net/manual/ru/reserved.variables.server.php) — Информация о сервере и среде исполнения
* [$\_GET](https://www.php.net/manual/ru/reserved.variables.get.php) — Переменные HTTP GET
* [$\_POST](https://www.php.net/manual/ru/reserved.variables.post.php) — Переменные HTTP POST
* [$\_FILES](https://www.php.net/manual/ru/reserved.variables.files.php) — Переменные файлов, загруженных по HTTP
* [$\_REQUEST](https://www.php.net/manual/ru/reserved.variables.request.php) — Переменные HTTP-запроса
* [$\_SESSION](https://www.php.net/manual/ru/reserved.variables.session.php) — Переменные сессии
* [$\_ENV](https://www.php.net/manual/ru/reserved.variables.environment.php) — Переменные окружения
* [$\_COOKIE](https://www.php.net/manual/ru/reserved.variables.cookies.php) — HTTP Cookies
* [$php\_errormsg](https://www.php.net/manual/ru/reserved.variables.phperrormsg.php) — Предыдущее сообщение об ошибке
* [$http\_response\_header](https://www.php.net/manual/ru/reserved.variables.httpresponseheader.php) — Заголовки ответов HTTP
* [$argc](https://www.php.net/manual/ru/reserved.variables.argc.php) — Количество аргументов, переданных скрипту
* [$argv](https://www.php.net/manual/ru/reserved.variables.argv.php) — Массив переданных скрипту аргументов

***7. Переменные переменных в php.***

Иногда бывает удобно иметь переменными имена переменных. То есть, имя переменной, которое может быть определено и изменено динамически. Переменная переменной берет значение переменной и рассматривает его как имя переменной.

***8. Выражения в php.***

Выражения - это самые важные строительные элементы PHP. Почти всё, что вы пишете в PHP, является выражением. Самое простое и точное определение выражения - "все что угодно, имеющее значение".

Основными формами выражений являются константы и переменные. Если вы записываете $a = 5, вы присваиваете 5 переменной *$a*. 5, очевидно, имеет значение 5 или, другими словами, 5 - это выражение со значением 5 (в данном случае 5 - это целочисленная константа).

***9. Арифметические операторы в php.***



***10. Битовые операции в php.***

| **Побитовые операторы** | | |
| --- | --- | --- |
| **Пример** | **Название** | **Результат** |
| **$a & $b** | И | Устанавливаются только те биты, которые установлены и в $a, и в $b. |
| **$a | $b** | Или | Устанавливаются те биты, которые установлены в $a или в $b. |
| **$a ^ $b** | Исключающее или | Устанавливаются только те биты, которые установлены либо только в $a, либо только в $b, но не в обоих одновременно. |
| **~ $a** | Отрицание | Устанавливаются те биты, которые не установлены в $a, и наоборот. |
| **$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 "=". На первый взгляд может показаться, что это оператор "равно". На самом деле это не так. В действительности 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.

***12. Операторы сравнения в php.***

| **Пример** | **Название** | **Результат** |
| --- | --- | --- |
| $a == $b | Равно | **true** если $a равно $b после преобразования типов. |
| $a === $b | Тождественно 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 | Тождественно 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.***

| **Пример** | **Название** | **Результат** |
| --- | --- | --- |
| $a and $b | И | **true**, если и $a, и $b **true**. |
| $a or $b | Или | **true**, если или $a, или $b **true**. |
| $a xor $b | Исключающее или | **true**, если $a, или $b **true**, но не оба. |
| ! $a | Отрицание | **true**, если $a не **true**. |
| $a && $b | И | **true**, если и $a, и $b **true**. |
| $a || $b | Или | **true**, если или $a, или $b **true**. |

***14. Условная конструкция в php.***

Конструкция if является одной из наиболее важных во многих языках программирования, в том числе и PHP. Она предоставляет возможность условного выполнения фрагментов кода. Структура if реализована в PHP по аналогии с языком C:

if (выражение)

инструкция

***15. Циклы в php.***

PHP поддерживает три вида циклов:

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

***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 **сase**. И если совпадение будет найдено, то будет выполняться определенный блок сase.

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. Итак, match в скобках также принимает некоторое сравниваемое выражение (в данном случае это переменная $a). Сам блок кода match также обертывается в фигурные скобки, однако в конце после закрывающей фигурной скобки необходимо поставить точку с запятой. А вместо операторов case просто указываются значения, с которыми сравнивается выражение.

***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.***

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]\*$.

Функции не обязаны быть определены до их использования, *исключая* тот случай, когда функции определяются условно, как это показано в двух последующих примерах.

В случае, когда функция определяется в зависимости от какого-либо условия, например, как это показано в двух приведённых ниже примерах, обработка описания функции должна *предшествовать* её вызову.

# Ссылка на удалённый репозиторий проекта

https://github.com/Rpqshka/University-5-Sem/tree/main/РСЧИР/Практика%202

1. **Список использованных источников**
2. Видео “Введение в Докер” на английском языке от создателя: Introduction to Docker (https://[www.youtube.com/watch?v=Q5POuMHxW-0)](http://www.youtube.com/watch?v=Q5POuMHxW-0))
3. Статья о назначении докера простыми словами: https://habr.com/ru/post/309556/
4. Более сложная и подробная статья про докер: https://habr.com/ru/post/277699/