Простые массивы

- Введение
- Индексный массив
- Ассоциативный массив

Введение

Для представления набора однотипных данных используются **массивы**, являющиеся наряду с циклами **фундаментальным инструментом программирования**. Массивы позволяют избегать дублирования кода и разрабатывать компактные, понятные и красивые программы.

Массив (Array) — это ещё один **тип данных**, вроде числа или строки.

> ***Важно***. Для наиболее полного использования возможностей массивов они должны применяться в комбинации с **циклами**, поэтому для успешного освоения данной темы необходимо уверенно владеть работой с циклами.

Главное отличие массива от остальных типов данных заключается в его способности хранить в переменной **больше одного значения**.

В предыдущих примерах имя переменной всегда ассоциировалось только с одним значением:

```
$name = "Татьяна";
$age = 31;
```

А если мы хотим хранить в программе не только имя и возраст пользователя, но и, фамилию, email, пол, любимые фильмы.

И вся эта информация относится к одному пользователю. Как хранить и обрабатывать этот набор разнородных данных с помощью простых переменных? Разбираемся.

Очень непросто назвать один самый любимый фильм, а вот вспомнить несколько — намного легче.

Сохранение в **переменную-массив** нескольких значений выглядит так:

```
$fav_movies = ["Собачье сердце", "Полет над гнездом кукушки",
"Тот самый Мюнхгаузен"];
```

В примере выше мы сохранили в переменной $fav_movies сразу три значения. Но сохранить эти данные — это только половина дела. Как с ними потом работать?

Уже знакомый вам способ вывода переменной на экран **не будет работать** с массивами:

```
print("Мои любимые фильмы: " . $fav_movies);

// вывод - Мои любимые фильмы: Array
```

Так увидеть список любимых фильмов не получится. Дело в том, что массив — это не обычная переменная. Массив хранит не простые типы, вроде текста или чисел (их ещё называют **скалярными типами**), а более **сложную структуру данных**.

Сложные структуры данных мы выводим с помощью конструкций:

- **print_r**(), или
- **var_dump**().

example_1. Тестируем вывод массива через print_r

```
<?php

    $fav_movies = ["Собачье сердце", "Полет над гнездом кукушки",
    "Тот самый Мюнхгаузен"];
```

```
    echo "<pre>";
    print_r($fav_movies);
    /*
        Array
        (
            [0] => Собачье сердце
            [1] => Полет над гнездом кукушки
            [2] => Тот самый Мюнхгаузен
        )
    */
?>
```

Я не думаю, что подобный вывод обрадует посетителей вашего сайта. Чтобы решить вопрос работы с массивом необходимо разобраться с его структурой.

В PHP существует три типа массивов:

- **Индексный массив** - массив с числовым индексом.
- **Ассоциативный массив** - массив с именованными ключами.
- **Многомерный массив** - массив, содержащий один или несколько вложенных массивов (вложенный массив может быть как индексным, так и ассоциативным).

Примечание. Последовательно разбираем следующие виды массивов:

- Одномерный индексный массив.
- Одномерный ассоциативный массив.
- Двумерный индексный массив.
- Двумерный ассоциативный массив.
- Трехмерный индексный массив.

- Трехмерный ассоциативный массив.

Индексный массив

Внутри массива у каждого значения есть адрес, по которому к нему можно обратиться. Такой адрес называется **индексом**.

Индекс — это просто **порядковый номер значения внутри массива**. Индексация начинается с нуля, так что первый элемент получает индекс — "0", второй — "1", и так далее.

Чтобы получить определенный элемент массива, **необходимо знать** его индекс.

> ***Важно***. Массив может содержать много значений **под одним именем**, получить доступ к значениям массива можно, ссылаясь на **номер индекса (ключа)**.
>
> **P.S**. Все таки в отношении индексных массивов принято говорить – индекс.

Попробуем вывести названия моих любимых фильмов в виде списка:

example_2. Вывод значений индексного массива

```
<?php
    // определим переменную-массив из трех значений
    $fav_movies = ["Собачье сердце", "Полет над гнездом
    кукушки", "Тот самый Мюнхгаузен"];
    // выведем значения массива в виде списка
    print("Мои любимые фильмы:
        <ul>
```

```
            <li>$fav_movies[0]</li>
            <li>$fav_movies[1]</li>
            <li>$fav_movies[2]</li>
            </ul>
    ");
?>
```

Таким образом массив — это совокупность **множества элементов** вида **индекс : значение**.

Существует **несколько способов** создания индексированных массивов:

- Первый способ представляет использование функции **array()**:

```
$numbers = array();
```

- Второй способ представляет использование квадратных скобок **[]**:

```
$numbers = [];
```

И в первом, и во втором случае определяется **пустой массив** $**numbers**.

При определении массива мы **сразу можем передать ему начальные данные**. Если применяются квадратные скобки, то элементы массива передаются внутри **скобок**:

```
$numbers = [1, 7, 3, 15];
```

Аналогичное определение массива с помощью функции **array**():

```
$numbers = array(1, 7, 3, 15);
```

Важно. В качестве элемента массива может выступать объект любого типа. Например, другой **массив** (об этом тема Многомерные массивы).

Массивы позволяют перезаписывать существующие значения и добавлять новые. Поскольку нумерация индексов начинается с нуля, то чтобы обратиться к **третьему элементу**, надо использовать индекс **2**.

```
// перезапишем третий элемент массива
// элемент может быть любого типа, пусть это будет строка
$numbers[2] = " Шпрехшталмейстер";
```

При этом нужно учитывать количество элементов массива. Так, мы не можем обратиться к элементу с **несуществующим индексом**:

```
echo $numbers[4];    // элемента с индексом 4 не существует
```

В данном случае в массиве $**numbers** всего 4 элемента, поэтому индексом последнего элемента будет 3. Таким образом, элемента с индексом 4 в массиве не существует, и при попытке получить его значение echo $**numbers**[4] PHP нам покажет предупреждение.

Тем не менее, если мы хотим добавить элемент по еще не существующему индексу, мы можем это сделать:

example_3. Добавление по несуществующему индексу

```
<?php
    // определим переменную-массив из трех значений
    // максимальный индекс - 2
    $fav_movies = ["Собачье сердце", "Полет над гнездом
    кукушки", "Тот самый Мюнхгаузен"];
    // добавим элемент в массив, но на позицию 5
    $fav_movies[5] = "Человек с бульвара Капуцинов";
    echo "<pre>";
    print_r($fav_movies);
?>
```

Общепринятой практикой считается следующий способ добавления нового элемента:

```
// новый элемент сохранится в массиве под индексом 3
$fav_movies[] = "Человек с бульвара Капуцинов";
```

Новый элемент автоматически получит индекс равный **максимальному индексу из существующих + 1**.

> ***Важно***. Необязательно как-то специально инициализировать переменную массива - добавлять новые элементы в массив можно **в процессе написания скрипта**:
>
> ```
> $user[] = "Татьяна";
> // какой-то код..., (или много кода)
> $user[] = 31;
> $user[] = "female";
> $user[] = "pretty-woman@gmail.com";
> ```

example_4. Добавление элементов в массив

```
<?php
    // заполним массив без предварительной его инициализации
    $team[] = "Aerosmith"; // индекс 0
    $team[] = "Pink Floyd"; // индекс 1
    $team[] = "The Beatles"; // индекс 2
    $team[10] = "AC/DC"; // индекс 10
    $team[] = "Metallica"; // индекс 11
    $team[] = "Ленинград"; // индекс 12
    //проверим что получилось
    echo "<pre>";
    print_r($team);
```

```
?>
```

В теме, посвященной строкам, мы говорили, что простые переменные свободно обрабатываются в строках с **двойными** кавычками и в строках с **Heredoc**-синтаксисом. А что с **интерполяцией индексных массивов**?

example_5. Интерполяция индексных массивов

```
<?php

    # инициализируем индексный массив

    $words[] = "мастерская";

    $words[] = "храм";

    $words[] = "работник";

    $str = "Природа не $words[1], а $words[0], и человек в ней
    $words[2] (Евгений Базаров)";

    echo $str;

    # все получилось

?>
```

Как видим, никаких проблем не возникло.

Ассоциативный массив

В предыдущем разделе мы познакомились с так называемыми индексными массивами. Но в PHP существует и чуть более сложный тип массивов — ассоциативные.

Ассоциативные массивы отличаются от простых тем, что вместо индексов у них **ключи**. И если индекс всегда является целым, порядковым числом, то **ключ может быть любой произвольной строкой**.

Ассоциативный массив — это совокупность **множества элементов** вида **ключ : значение**

Мы уже знаем многое о нашем пользователе: его имя, фамилию, возраст, пол, e-mail… Есть только одно неудобство, поймете какое?

```
$user[] = "Татьяна";
$user[] = "pretty-woman@gmail.com";
$user[] = 31;
$user[] = 9;
$user[] = 7;
$user[] = 1;
```

Ну, с индексами 0 и 1 все ясно, user[2] = 31 можно предположить, что это возраст. А дальше беда… 9 - это что? Какие-то 7, 1. Кабинет? Количество выдаваемых предметов?

В таких ситуациях помогают ассоциативные массивы. Запись всей информации о пользователе с помощью ассоциативного массива выглядит следующим образом:

```
$user["name"] = "Татьяна";
$user["email"] = "pretty-woman@gmail.com";
$user["age"] = 31;
$user["general_experience"] = 9;
$user["edu_experience"] = 7;
$user["children"] = 1;
```

А! Вот как, оказывается:

```
$user["general_experience"] = 9; // общий стаж
$user["edu_experience"] = 7; // стаж в учебном заведении
$user["children"] = 1; // количество детей
```

Принципы работы с ассоциативным и индексным массивами аналогичны.

example_6. Ассоциативный массив

```php
<?php
    // можно инициализировать массив
    // $user = array();
    // $user = [];
    $user["name"] = "Татьяна";
    $user["surname"] = "М.";
    $user["age"] = 31;
    $user["email"] = "pretty-woman@gmail.com";
    $user["general_experience"] = 9;
    $user["edu_experience"] = 7;
    $user["children"] = 1;
    // так будет работать,
    // только непонятно, зачем смешивать индексный и
    ассоциативный массивы
    $user[1] = "г. Печора";
    $user[] = "Драники";
    // проверим что получилось
    echo "<h3>Hi, Miss " . $user['name'] . " " .
    $user['surname'] . ", вот твои личные данные:</h3>";
    echo "<pre>";
    print_r($user);
?>
```

> ***К сведению***. Массив может содержать **другой массив** в качестве одного из значений.

Отсюда напрашивается вывод:

~~Ассоциативный массив лучше индексного массива~~! Нет? Тогда может - ~~ассоциативный массив хуже индексного массива~~? Тоже нет…

Все просто, он не лучше и не хуже. Это разные массивы, и они замечательно дополняют друг друга. Об этом тема - **Многомерные массивы**.

Интерполяции ассоциативных массивов имеет некоторые особенности:

```
// так работать не будет
echo "<h3>Hi, Miss $user['name'] $user['surname'], ...";
```

> ***Важно***. Ключи, заключённые в кавычки, работают только с синтаксисом **фигурных скобок**.

```
// а так будет
echo "<h3>Hi, Miss {$user['name']}  {$user['surname']}, ...";
// так тоже будет
echo "<h3>Hi, Miss $user[name] $user[surname], ...";
```

example_7. Интерполяция ассоциативных массивов

```
<?php
     // можно инициализировать массив
     // $user = array();
     // $user = [];
     $user["name"] = "Татьяна";
     $user["surname"] = "М.";
```

```php
    $user["age"] = 31;
    $user["email"] = "pretty-woman@gmail.com";
    $user["general_experience"] = 9;
    $user["edu_experience"] = 7;
    $user["children"] = 1;
    // так будет работать,
    // только непонятно, зачем смешивать индексный и ассоциативный массивы
    $user[1] = "г. Печора";
    $user[] = "Дранки";
    // так работать не будет
    echo "<h3>Hi, Miss $user['name'] $user['surname'], вот твои личные данные:</h3>";
    echo "<pre>";
    // print_r($user);
    // а так будет
    //echo "<h3>Hi, Miss {$user['name']}  {$user['surname']}, вот твои личные данные:</h3>";
    //echo "<pre>";
    //print_r($user);
    // так тоже будет
    // echo "<h3>Hi, Miss $user[name] $user[surname], вот твои личные данные:</h3>";
    // echo "<pre>";
    // print_r($user);
?>
```

Задание № 18

==========

Перед вами информация по преподавателю. Преобразуйте представленные данные в **индексный массив**.

Используя конструкцию **var_dump**, выведите массив в браузер.

Фамилия: Лаврецкая
Имя: Елизавета
Отчество: Викторовна
Дата рождения: 25.06.1980
Должность: Преподаватель
Основная должность: Заместитель директора по УР
Категория: Высшая
Уровень образования: Высшее профессиональное
Учебное заведение: Московский государственный институт электронной техники (технический университет)
Квалификация: Менеджер
Специализация Менеджмент организации
Стаж в учебном заведении: 19
Полный стаж: 22
E-mail: lovel@mail.ru

Задание № 19

==========

Перед вами информация по преподавателю. Преобразуйте представленные данные в **ассоциативный массив**. Ключи массива создайте либо транслитом, либо с использованием переводчика.

Например:

- ключ Фамилия - **familiya** | **surname**.
- ключ Имя - **imya** | **name**.

Используя конструкцию **print_r**, выведите массив в браузер.

Фамилия: Лаврецкая
Имя: Елизавета
Отчество: Викторовна
Дата рождения: 25.06.1980
Должность: Преподаватель
Основная должность: Заместитель директора по УР
Категория: Высшая
Уровень образования: Высшее профессиональное
Учебное заведение: Московский государственный институт электронной техники (технический университет)
Квалификация: Менеджер
Специализация Менеджмент организации
Стаж в учебном заведении: 19
Полный стаж: 22
E-mail: lovel@mail.ru

Задание № 20

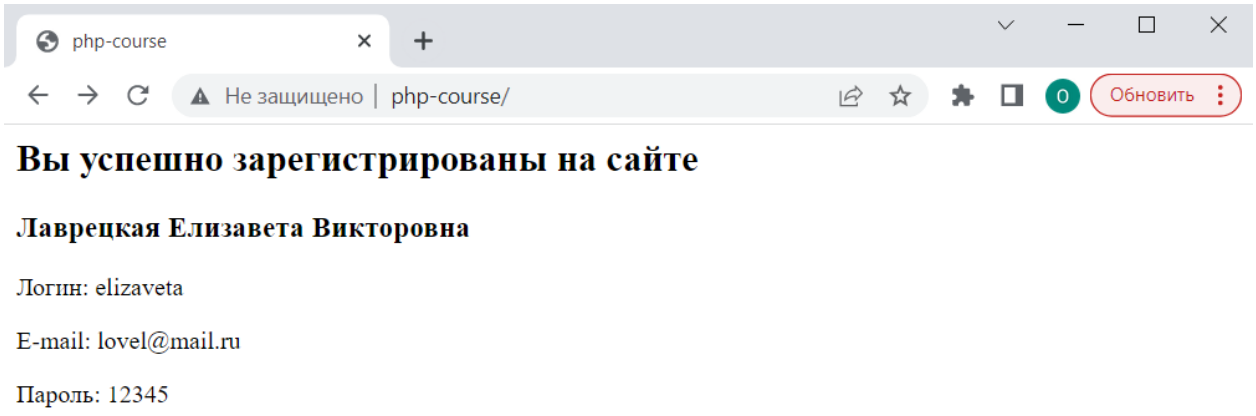
==========

Перед вами информация по преподавателю:

- Фамилия: Лаврецкая
- Имя: Елизавета
- Отчество: Викторовна
- Логин: elizaveta
- Пароль: 12345
- Email: lovel@mail.ru

Преобразуйте представленные данные в **один из массивов по вашему усмотрению**.

Выполните вывод данных массива в браузер. **Примерный** формат вывода продемонстрирован на рисунке.



Многомерные массивы

Введение

В предыдущих примерах рассматривались только **одномерные массивы**, где значения элементов представляли числа или строки. Но в PHP массивы могут также быть **многомерными**, то есть такими, где элемент массива сам является массивом.

> ***Важно***. **Многомерный массив** - массив, содержащий один или несколько массивов.

PHP распознает многомерные массивы, два, три, четыре, пять или больше уровней. Измерение массива указывает количество индексов, необходимых для выбора элемента.

Для **двумерного массива**, необходимо **два** индекса для выбора элемента (двумерный массив - массив массивов).

Для **трехмерного массива**, необходимо **три** индекса для выбора элемента (трехмерный массив - массив массивов массивов).

Для **обработки значений многомерных массивов** (интерполяции) в строках всегда придерживайтесь золотого правила использования **фигурных скобок**.

> ***Важно***. **При использовании многомерных массивов внутри строк всегда используйте фигурные скобки**.

Несмотря на то, что PHP поддерживает многомерные массивы, управление массивами **более трех уровней** вложения - сложная задача для разработчика. А так как в подавляющем количестве приложений информация берется из **таблиц баз данных**, и таблица представляет собой **двумерный массив**, то и мы остановимся на рассмотрении двумерных массивов.

Индексный двумерный массив

Двумерный массив представляет частный случай многомерного массива. Взгляните на таблицу.

Страна	Столица	Население (млрд)
Китай	Пекин	1,40
Индия	Нью-Дели	1,37
США	Вашингтон	0,32
Индонезия	Джакарта	0,24

Данные из таблицы выше можно хранить в **индексном двумерном массиве**:

```
$country = array(
      array("Китай", "Пекин", 1.40),
      array("Индия", "Нью-Дели", 1.37),
      array("США", "Вашингтон", 0.32),
      array("Индонезия", "Джакарта", 0.24)
);
```

Сокращенное определение:

```
$country = [
      ["Китай", "Пекин", 1.40],
      ["Индия", "Нью-Дели", 1.37],
```

```
    ["США", "Вашингтон", 0.32],
    ["Индонезия", "Джакарта", 0.24]
];
```

Теперь, двумерный массив $**country** содержит четыре элемента-массива и имеет два показателя: **строку** и **столбец**. Соответственно, чтобы получить доступ к элементам массива нужно указать **два индекса** (строки и столбца):

example_1. Индексный двумерный массив

```
<?php
    // двумерный индексный массив
    $country = [
        ["Китай", "Пекин", 1.40],
        ["Индия", "Нью-Дели", 1.37],
        ["США", "Вашингтон", 0.32],
        ["Индонезия", "Джакарта", 0.24]
    ];
    // вывод данных двумерного массива, конкатенация
    echo "Страна: " . $country[0][0] . ", Столица: " .
    $country[0][1] . ", Население: " . $country[0][2] . "
    (млрд.чел.)<br>";
    echo "Страна: " . $country[1][0] . ", Столица: " .
    $country[1][1] . ", Население: " . $country[1][2] . "
    (млрд.чел.)<br>";
    echo "Страна: " . $country[2][0] . ", Столица: " .
    $country[2][1] . ", Население: " . $country[2][2] . "
    (млрд.чел.)<br>";
    echo "Страна: " . $country[3][0] . ", Столица: " .
    $country[3][1] . ", Население: " . $country[3][2] . "
    (млрд.чел.)<br>";
```

```
    echo "<p><hr />";
    // или так
    // вывод данных двумерного массива, интерполяция
    echo "Страна: {$country[0][0]}, Столица: {$country[0][1]},
    Население: {$country[0][2]} (млрд.чел.)<br>";
    echo "Страна: {$country[1][0]}, Столица: {$country[1][1]},
    Население: {$country[1][2]} (млрд.чел.)<br>";
    echo "Страна: {$country[2][0]}, Столица: {$country[2][1]},
    Население: {$country[2][2]} (млрд.чел.)<br>";
    echo "Страна: {$country[3][0]}, Столица: {$country[3][1]},
    Население: {$country[3][2]} (млрд.чел.)<br>";
?>
```

Ассоциативный двумерный массив

Также можно определять **двумерные ассоциативные массивы**:

```
$country = [
    ["name" => "Китай", "capital" => "Пекин", "population" =>
    1.40],
    ["name" => "Индия", "capital" => "Нью-Дели", "population" =>
    1.37],
    ["name" => "США", "capital" => "Вашингтон", "population" =>
    0.32],
    ["name" => "Индонезия", "capital" => "Джакарта",
    "population" => 0.24]
];
```

example_2. Ассоциативный двумерный массив

```
<?php
```

```php
// двумерный ассоциативный массив
$country = [
    ["name" => "Китай", "capital" => "Пекин", "population"
    => 1.40],
    ["name" => "Индия", "capital" => "Нью-Дели",
    "population" => 1.37],
    ["name" => "США", "capital" => "Вашингтон",
    "population" => 0.32],
    ["name" => "Индонезия", "capital" => "Джакарта",
    "population" => 0.24]
];
// разными способами организуем вывод
// простая конкатенация
echo "
Страна: " . $country[0]["name"] . ",
Столица: " . $country[0]["capital"] . ",
Население: " . $country[0]["population"] . "
(млрд.чел.)<p>";
// интерполяция сложных переменных
echo "Страна: {$country[1]['name']},
Столица: {$country[1]['capital']},
Население: {$country[1]['population']} (млрд.чел.)<p>";
// синтаксис heredoc плюс интерполяция,
// ключ в одинарных кавычках
echo <<<HERE
Страна: {$country[2]['name']},
Столица: {$country[2]['capital']},
Население: {$country[2]['population']} (млрд.чел.)<p>
```

```
    HERE;
    // синтаксис heredoc плюс интерполяция,
    // ключ в двойных кавычках
    echo <<<HERE
    Страна: {$country[3]["name"]} ,
    Столица: {$country[3]["capital"]} ,
    Население: {$country[3]["population"]} (млрд.чел.)<p>
    HERE;
?>
```

Итоги

Тема **массивов**, она не просто важная, она основополагающая при программировании баз данных. Поэтому подведу небольшие итоги.

Википедия:	**Таблица** — это совокупность связанных данных, хранящихся в структурированном виде в базе данных. Она состоит из столбцов и строк. В реляционных базах данных таблица — это набор элементов данных (значений), использующий модель вертикальных столбцов (имеющих уникальное имя) и горизонтальных строк. **Ячейка** — место, где строка и столбец пересекаются. Таблица содержит определенное число столбцов, но может иметь любое количество строк.

При работе с базами данных таблицы будут рассматриваться нами как **двумерные ассоциативные массивы**.

Важно. Если попытаться уложить теорию баз данных в несколько

ключевых строк, то вот они:

- **База данных** – одна или несколько взаимосвязанных **таблиц**.
- **Таблица** – индексный неупорядоченный массив строк.
- **Строка** – ассоциативный (именованный) упорядоченный массив ячеек.

В приведенном ниже примере представлена таблица, состоящая из 4 строк. Индекс первой строки – 0, индекс последней строки 3.

Таблица

	name	capital	population
0	Китай	Пекин	1,40
1	Индия	Нью-Дели	1,37
2	США	Вашингтон	0,32
3	Индонезия	Джакарта	0,24

Каждая строка состоит из именованных ячеек. Обратиться к ячейке можно по ее имени.

Таблица[2] – обращение к третьей строке таблицы.

name	capital	population
США	Вашингтон	0,32

Таблица[2] ["capital"] – обращение к третьей строке, ячейке с именем **capital**.

capital
Вашингтон

Таблица[2] ["capital"] = "Вашингтон"

Задание № 21

==========

Вам дан файл с дискографией группы **Pink Floyd**. Преобразуйте представленную таблицу в **двумерный индексный массив**.

Используя конструкцию **print_r,** выведите массив в браузер.

Задание № 22

==========

Преобразуйте двумерный индексный массив предыдущего **Задания** в **двумерный ассоциативный**.

Используя конструкцию **var_damp,** выведите массив в браузер.

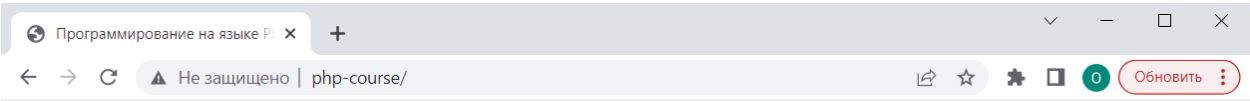
Задание № 23

==========

В директории раздаточного материала вам предложен файл с **двумерным массивом** альбомов группы Pink Floyd.

Используя инструкцию **echo** выведите данные массива в браузер. Технологию оформления строки (одинарные, двойные кавычки, HEREDOC) выберите **самостоятельно** по своему усмотрению.

Примерный формат вывод представлен на скриншоте.



Многомерные массивы

ID	Название альбома	Дата выпуска	Лейбл	Формат	Статус
1	Atom Heart Mother	10 октября 1970	EMI, Harvest, Capitol	LP, CD	Золотой (USA)
2	Meddle	30 октября 1971	EMI, Harvest, Capitol	Vinyl, Кассета, CD	Платиновый (USA)
3	Obscured by Clouds	3 июня 1972	EMI, Harvest, Capitol	LP, Кассета, CD	Золотой (USA)
4	The Dark Side of the Moon	17 марта 1973	Harvest, Capitol, EMI	LP, Кассета, CD, SACD	Платиновый (USA), Платиновый (GBR), Бриллиантовый (CAN)
5	Wish You Were Here	15 сентября 1975	Harvest, EMI, Columbia, Capitol	LP, 8-track, Кассета, CD, SACD	Платиновый (USA), Золотой (GBR), Платиновый (CAN)
6	Animals	23 января 1977	Harvest, EMI Columbia, Capitol	LP, 8-track, Кассета, CD	Платиновый (USA), Золотой (GBR), Платиновый (CAN)
7	The Wall	30 ноября 1979	Harvest, EMI Columbia, Capitol	LP, 8-track, Кассета, CD	Платиновый (USA), Платиновый (GBR), Бриллиантовый (CAN), Платиновый (NLD)
8	The Final Cut	21 марта 1983	Harvest, EMI Columbia, Capitol	LP, 8-track, Кассета, CD	Платиновый (USA), Золотой (GBR), Золотой(NLD)
9	A Momentary Lapse of Reason	8 сентября 1987	EMI, Columbia	LP, Кассета, CD	Платиновый (USA), Золотой (GBR), Платиновый (CAN), Золотой(NLD)
10	The Division Bell	30 марта 1994	EMI, Columbia	LP, Кассета, CD	Платиновый (USA), Платиновый (GBR), Платиновый (CAN), Платиновый(NLD)

Задание № 24

==========

Если быть внимательным, то не сложно увидеть, что в полях **Лейбл**, **Формат**, **Статус** таблицы Дискография находятся не атомарные значения.

> ***Важно***. В теории баз данных ***атомарность*** означает **неделимость данных**. Это значит, что в базе не должно быть поля **ФИО**, должны быть три поля – **Фамилия**, **Имя**, **Отчество**.

В самом деле, перечисленные поля представляют хранимые значения в виде **списка**.

Это приводит к тому, что получить **Статус** альбома **Atom Heart Mother** в США - затруднительно. При обращении к пятой ячейке первой строки таблицы мы получаем **не значение, а список**, который надо как-то преобразовать в **атомарное** (неделимое, простое) значение для вывода.

Задача.

Преобразуйте **ассоциативный массив** Дискография из **двумерного** в **многомерный**.

Используя конструкцию **print_r**, выведите массив в браузер.

Задание № 25

==========

Поиграем в ролевую игру.

Вы являетесь разработчиком государственных стандартов по Литературе. Вам выпала обязанность составить анкету по литературе для старшеклассников.

Файл анкеты прилагается. Проанализируйте содержимое, подумайте, как можно было бы представить анкету в виде **многомерного массива**. Массив должен хранить:

- **вопросы анкеты**,
- **варианты ответов** по каждому вопросу,
- **правильный ответ** или **ответы** (если несколько) по каждому вопросу.

P.S. С помощью многомерного массива можно представить практически любую информацию. Подумайте, смогли бы вы сохранить в массив предложенную анкету. Посмотрите один из вариантов решения поставленной задачи. Придумайте свой вариант хранения какой-либо информации в виде многомерного (минимум двумерного) массива.

ВНИМАНИЕ: Обработка массивов - отдельная тема.

Связанные массивы

- Введение
- Разделяем массивы согласно сущностям
- Практика применения

Введение

Вся информация в компьютере хранится исключительно в виде ноликов и единиц, с помощью специально созданных для этих целей форматов.

Хранить **текстовую информацию** — это один формат. Переводим набор символов, из которых состоит текст, в последовательный двоичный код (**0** или **1)** и с помощью формата говорим компьютеру - эти данные надо обрабатывать как **текстовый файл**. В таком случае последовательность бит это **буква**.

Например:

10100000 - **буква** а.

Хранить **графическую информацию** — это другой формат. Переводим набор точек, из которых состоит изображение, в последовательный двоичный код (**0** или **1**) и с помощью другого формата говорим компьютеру - эти данные надо обрабатывать как **графический файл**. В таком случае последовательность бит это **точка** определенного цвета.

Например:

00000000 - **точка** черного цвета.

Поэтому, когда мы слышим - большие объемы информации в памяти компьютера хранятся в таблицах баз данных, это не совсем правда. Скорее так - табличная форма отображения данных используется исключительно для наилучшего **восприятия информации** человеком.

Когда программист получает данные из базы для дальнейшей их **обработки** и **отображения**, то и хранить эти данные в скрипте лучше всего таким образом, чтобы данные повторяли табличное представление.

Никто делать именно так не заставляет, но это удобно и практикуется повсеместно.

> ***Важно***. При **обработке данных** удобно разрабатывать **структуру массивов** так, чтобы они повторяли табличную логику организации данных.

Разделяем массивы согласно сущностям

До сих пор мы рассматривали одномерные и многомерные массивы в их несколько оторванной от предметной области действительности. Дело в том, что хранящаяся в компьютере информация должна отражать объекты реального мира. Людям свойственно воспринимать окружающий мир как множество **взаимодействующих между собой объектов**, поддающихся определенной классификации.

В таком представлении информацию легче **воспринимать** и уж тем более **хранить** в компьютере.

Сейчас приведу примеры, о чем это я.

В файле **example_1.php** представлен уже знакомый **массив** дискографии группы Pink Floyd. Довольно большой массив, но он представляет только малую часть информации о группе - **ее альбомы** (и то не все). А если возникнет необходимость хранить полный набор данных о творчестве британского коллектива? Его **историю**, **контракты**, **иллюстрации** альбомов, **туры**, **тексты** песен, **участников**, наконец.

То, что массив разрастется в глубину (увеличится количество информации) не беда, компьютер справится. Но что критично - массив обязательно разрастется в ширину. Существенно усложнится сама **структура массива**. А вот структуру создает человек, и более того, при создании алгоритма обработки, человеку придется каким-то образом помнить - где какой ключ (индекс) массива и за что отвечает.

Пропуская всю лирику реляционной теории хранения и обработки множеств, сразу скажу - выход давно и успешно найден. При программировании приложений принято разделять информацию на **сущности**, описывающие предметную область.

> ***Из теории отношений***. **Сущность** (таблица) – **реальный** либо **воображаемый объект**, имеющий существенное значение для рассматриваемой предметной области, **информация о котором подлежит хранению**.

А это значит, что в приведенном выше примере должен быть не один массив, а **несколько**, ну например таких:

- Информация о группе.
- Дискография.
- Тексты песен.
- Участники.
- и т.д.

Разработчики с некоторым стажем уже поняли, я, безусловно, клоню в сторону **реляционных** систем управления **базами данных**. При разработке реляционных баз данных создается несколько **взаимосвязанных** таблиц, в каждой из которых хранится информация

об объекте реального мира (сущности). А раз так, то и в скрипте мы будем получать несколько **связанных между собой массивов**.

> ***К сведению***. Пользователь, знающий современные способы хранения больших объемов данных, может возразить, почему речь пошла о реляционных отношениях - есть **Денормализация**, **NoSQL базы**. Отвечу.
>
> Прежде чем говорить о денормализации, необходимо изучить понятие **нормализации**.
>
> Прежде чем изучать NoSQL, хорошо бы понять теорию **SQL управления** реляционными базами данных.

Рассмотрим всем известный пример взаимодействия двух объектов реального мира - **писателей** и их **произведений**.

Допустим, у вас имеется таблица, информацию из которой необходимо хранить и обрабатывать в приложении.

№	Автор	Произведение
1	Чехов А.П.	Дядя Ваня Три сестры Вишнёвый сад
2	Довлатов С.Д.	Заповедник Компромисс Марш одиноких Соло на ундервуде
3	Булгаков М.А.	Белая гвардия Ханский огонь
4	Севела Э.Е.	Легенды Инвалидной улицы

Мы уже знаем, что подобные таблицы легко представить в виде **многомерного ассоциативного массива**. Например, так:

```
$biblio = [
     [
          "id" => 1,
          "author" => "Чехов А.П.",
          "books" => ["Дядя Ваня", "Три сестры", "Вишнёвый сад"]

      ],
     [
          ...
     ]
]
```

Исходя из необходимости разделять информацию о разных сущностях, разобьем массив на два – **авторы** и **книги**.

```
$authors = [
     [
          "id" => 1,
          "name" => "Чехов А.П."
     ],
     [
          "id" => 2,
          "name" => "Довлатов С.Д."
     ],
     [
          "id" => 3,
          "name" => "Булгаков М.А."
     ],
     [
          ...
     ]
]
$books = [
     [
          "id" => 1,
          "names" => ["Белая гвардия", "Ханский огонь"]
     ],
     [
```

```
            "id" => 2,
            "names" => ["Дядя Ваня", "Три сестры", "Вишнёвый сад"]
      ],
      [
            ...
      ]
]
```

То, как мы разделили массив на два - очень хорошо. Но теперь проблемы поменялись местами, что хорошо человеку - плохо компьютеру.

Ключевая проблема, - каким образом компьютер определит, какая книга из массива $**books** принадлежит какому автору из массива $**authors**.

Для этих целей существует простой способ. Достаточно добавить **дополнительный ключ** в массив $**books**, определяющий принадлежность книги автору.

Вот так:

```
$books = [
      [
            "id" => 1,
            "names" => ["Белая гвардия", "Ханский огонь"],
            "id_author" => 3
      ],
      [
            "id" => 2,
            "names" => ["Дядя Ваня", "Три сестры", "Вишнёвый сад"],
            "id_author" => 1
      ],
      [
            ...
      ]
]
```

Вот теперь гораздо лучше. И даже не знающий предметной области легко сопоставит отношение **книга - автор**. А уж компьютеру с его быстродействием сделать это - совсем не проблема.

Исходный массив – example_2.php .

Разделенный массив – example_3.php.

> ***Важно***. Идентификаторы (**id**) массивов называются **первичными ключами**. Идентификатор **id_author** таблицы $books называется **внешним ключом** и служит для связи между таблицами.

> ***К сведению***. Совсем не обязательно пытаться вникнуть в основы реляционных баз данных. Достаточно согласиться, что все написанное о разделении таблиц - естественно и логично. А еще свыкнуться с мыслью, что понятие **первичного** и **внешнего** ключа будет преследовать вас все время разработки веб-приложений.

Вот так, незаметно, мы прошли краткий курс по теории разработки реляционных баз данных.

Это все хорошо, но зачем нам это все надо? Какой в этом практический смысл? Скоро разберемся.

Практика применения

Тему "Простые массивы" я начал с утверждения:

> ***Вспомним***. Для наиболее полного использования **возможностей массивов** они должны применяться в комбинации с **циклами**, поэтому для успешного освоения данной темы необходимо уверенно владеть работой с циклами.

Сейчас это утверждение еще более актуально. И никакой практики применения не может быть без знания **циклов**.

Ну, а в случае со связанными массивами – **вложенных циклов**!

Задание № 26

==========

Вам дан файл PHP-дампа **db_music.php** учебной базы данных db_music. Используя данные **связанных массивов**, определите информацию о хранящихся в массивах **альбомах** следующих **групп**:

1) AC/DC
2) Scorpions
3) Aerosmith

Запишите полученную информацию.

Альбомы группы **AC/DC**:

-
-
-

Альбомы группы **Scorpions**:

-
-
-

Альбомы группы **Aerosmith**:

-
-
-

Задание № 27

==========

Вам дан файл PHP-дампа **db_music.php** учебной базы данных db_music. Соберите **информацию о группе**, выпустившей альбом '**The Razors Edge**' (id_album = 8).

Запишите полученную **информацию**:

- Идентификатор (id_team): _____
- Название (name): _____
- Алиас (alias): _____
- Страна (country): _____
- Контент (content): _____
- Год основания (date): _____
- Стиль (style): _____
- Примечание (note): _____

Задание № 28

==========

Вам дан файл PHP-дампа **db_music.php** учебной базы данных db_music. Используя данные **связанных массивов**, найдите и выпишите названия **групп**, у которых нет **треков**.