*Решения и примеры для программистов на PHP*

# PHP
# Сборник рецептов

СИМВОЛ®
O'REILLY®

*Дэвид Скляр и Адам Трахтенберг*

# PHP Cookbook

*David Sklar and Adam Trachtenberg*

O'REILLY®

# PHP

# Сборник рецептов

*Дэвид Скляр и Адам Трахтенберг*

*Санкт-Петербург — Москва*
*2005*

Дэвид Скляр и Адам Трахтенберг

# PHP. Сборник рецептов

Перевод А. Петухова


| | |
|---|---|
| Главный редактор | *А. Галунов* |
| Зав. редакцией | *Н. Макарова* |
| Научные редакторы | *Д. Горяинов, Р. Шевченко* |
| Редактор | *В. Овчинников* |
| Корректор | *С. Доничкина* |
| Верстка | *Н. Гриценко* |

*Скляр Д., Трахтенберг А.*
PHP. Сборник рецептов. – Пер. с англ. – СПб: Символ-Плюс, 2005. – 672 с., ил.
ISBN 5-93286-059-6



«PHP. Сборник рецептов» Дэвида Скляра и Адама Трахтенберга содержит практические примеры и решения разнообразных задач, ежедневно возникающих перед программистами. Каждая задача снабжена проработанным решением – «рецептом», содержащим небольшой фрагмент кода, который можно вставлять прямо в приложение. Представлено более 250 рецептов – от самых простых, таких как посылка запроса в базу данных и получение доступа к URL, до полноценных программ, демонстрирующих более трудные задачи, например вывод HTML-таблиц и создание диаграмм. Рассмотрена работа со строками, числами, датами и временем, а также с массивами, файлами и каталогами. Обсуждаются переменные, функции, классы и объекты, регулярные выражения, шифрование и безопасность, интернет-службы, графика, интернационализация и локализация, PEAR, PHP в командной строке и PHP-GTK, формы, XML и доступ к базам данных.

Книга будет полезна всем, кто программирует на PHP, независимо от уровня их подготовки – от новичков до опытных профессионалов.



**ISBN 5-93286-059-6**
**ISBN 1-56592-681-1 (англ)**





Издательство «Символ-Плюс». 199034, Санкт-Петербург, 16 линия, 7,
тел. (812) 324-5353, edit@symbol.ru. Лицензия ЛП N 000054 от 25.12.98.
Налоговая льгота – общероссийский классификатор продукции
ОК 005-93, том 2; 953000 – книги и брошюры.

Подписано в печать 27.12.2004. Формат 70×100¹/16. Печать офсетная.
Объем 42 печ. л. Тираж 2000 экз. Заказ №
Отпечатано с готовых диапозитивов в ГУП «Типография «Наука»
199034, Санкт-Петербург, 9 линия, 12.

# Оглавление

# Предисловие

PHP лежит в основе миллионов динамических веб-приложений. Широкий набор возможностей, понятный синтаксис и поддержка различных операционных систем и веб-серверов сделали его идеальным языком как для быстрой разработки несложных веб-сайтов, так и для кропотливой и методичной работы над сложными веб-системами.

Одной из главных причин успеха PHP как языка веб-сценариев стало то, что он изначально задумывался как инструмент обработки HTML-форм и создания веб-страниц. Это сразу сделало PHP весьма дружественной средой разработки для Всемирной паутины. Кроме того, PHP – полиглот. PHP может общаться с большим количеством баз данных и знает многочисленные протоколы Интернета. PHP упрощает анализ данных броузера и может выполнять HTTP-запросы. Эта веб-специфическая направленность распространяется на все рецепты и примеры книги «PHP. Сборник рецептов».

Эта книга представляет собой сборник решений наиболее распространенных задач из практики PHP. Мы постарались включить материал, который будет привлекателен для всех – от новичков до мастеров. И если мы достигли цели, то вы узнаете что-нибудь новое (а возможно, многому научитесь). Здесь найдутся подсказки для тех, кто ежедневно программирует на PHP, и тех, кто пришел в PHP с опытом работы на других языках.

PHP – свободно распространяемая среда программирования. Его можно взять в виде исходного или двоичного кода с сайта *http://www.php.net/*. Веб-сайт PHP также содержит инструкции по установке, полную документацию и ссылки на источники в Интернете, рабочие группы пользователей, списки почтовой рассылки и другие ресурсы PHP.

## Для кого эта книга

Это книга для программистов, решающих практические задачи с помощью PHP. Если же вы совсем не знаете PHP, то пусть это будет ваша вторая книга по PHP. Первой должна быть «Programming PHP», также от издательства O'Reilly & Associates.[1]

Тем, кто уже знаком с PHP, эта книга поможет преодолеть определенные трудности и будет верным спутником на всю жизнь (по крайней

[1] Lerdorf R., Tatroe K. «Programming PHP», O'Reilly, 2002.

мере, пока вы занимаетесь программированием). Кроме того, здесь показано, как решать конкретные задачи, такие как отправка почты или создание SOAP-сервера, способ решения которых на других языках, возможно, вам уже известен. Программисты, переводящие приложения с других языков на PHP, найдут в этой книге надежного помощника.

## Что в этой книге

Мы не ожидаем, что вы сядете и прочитаете эту книгу от корки до корки (хотя будем счастливы, если вы это сделаете!). Программисты на PHP постоянно сталкиваются лицом к лицу с множеством сложных задач по широкому кругу вопросов. Обратитесь к этому сборнику, чтобы найти решение встретившейся задачи. Каждый рецепт представляет собой разъяснение, а это хорошая отправная точка на пути к успешному решению. Если в рецепте что-то не рассматривается подробно, то приводятся ссылки на соответствующие рецепты книги и другие онлайновые и автономные источники.

Если вы соберетесь прочитать всю главу сразу, это будет правильное решение. В основном сложность рецептов возрастает от начала к концу главы. В конце многих глав есть примеры программ, объединяющих рассмотренные примеры. Введение в начале каждой главы дает вводный обзор и основополагающие сведения о материале, рассматриваемом в этой главе, а также акцентирует внимание на особо важных рецептах.

Книга начинается с четырех глав, повествующих об основных типах данных. В главе 1 подробно рассматриваются такие вопросы, как обработка подстрок, управление регистром, разделение строки на более мелкие части и анализ данных, разделенных запятыми. Глава 2 посвящена операциям над числами с плавающей точкой, случайным числам, преобразованию чисел из одной системы счисления в другую и форматированию чисел. В главе 3 показано, как работать с датами и временем, как их форматировать, как работать с часовыми поясами и летним временем и как определять время с точностью до микросекунды. Глава 4 посвящена операциям с массивами, таким как выполнение циклов, объединение, обращение, сортировка и извлечение определенных элементов.

В следующих трех главах обсуждаются блоки, образующие программу. В главе 5 рассматриваются примечательные возможности PHP в области работы с переменными, такие как значения по умолчанию, статические переменные и создание строкового представления сложных типов данных. Рецепты в главе 6 имеют дело с применением функций в PHP: обработка аргументов, передача и возвращение переменных по ссылке, создание функций во время выполнения и использование области видимости переменных. Рецепты главы 7 посвящены объектно-ориентированным возможностям PHP, таким как использование пе-

резагрузки и полиморфизма, определение конструкторов и клонирование объектов.

Ядро этой книги составляют пять глав, посвященных центральным темам веб-программирования. Глава 8 рассматривает работу с cookies, заголовки, механизм аутентификации, переменные конфигурации и другие традиционные аспекты веб-приложений. В главе 9 идет речь об обработке и проверке подлинности ввода в формах, отображении форм с сообщениями об ошибках и о преобразовании в escape-последовательности специальных символов в пользовательских данных. В главе 10 объяснены различия между текстовым файлом, DBM и базами данных SQL, и на основе уровня абстракции базы данных PEAR DB показано, как присваивать уникальные значения идентификаторов, извлекать строки, изменять данные, преобразовывать кавычки в escape-последовательности и регистрировать отладочную информацию. В главе 11 основное внимание уделено извлечению URL и обработке HTML, но рассказано и о применении шаблонов и об анализе журнала доступа к серверу. Глава 12 посвящена XML и связанным с ним форматам, в том числе DOM, SAX, XSLT, XML-RPC и SOAP.

Следующий раздел книги представляет серию глав о других возможностях и расширениях PHP, которые составляют значительную часть полезной функциональности. Это рецепты, помогающие создавать приложения более устойчивые, защищенные, дружественные к пользователю и более эффективные. В главе 13 рассматриваются регулярные выражения, включая определение допустимых адресов электронной почты, выделение текста внутри тегов HTML и использование поглощающего и непоглощающего сравнения. В главе 14 обсуждается шифрование, включая генерирование и сохранение паролей, совместное использование закодированной информации, запись шифрованных данных в файл или базу данных и применение SSL. Глава 15 показывает, как динамически создавать графику с помощью рецептов по рисованию текста, линий, многоугольников и кривых. В главе 16 рассказано, как сделать приложения дружественными в мировом масштабе, и приведены рецепты по использованию локализации и локализованного текста, местных дат и времени, значений валюты и изображений. В главе 17 обсуждаются связанные с сетью Интернет задачи, такие как чтение и отправка электронной почты и сообщений новостных групп, использование FTP и LDAP, поиск с применением сервисов DNS и Whois.

Главы 18 и 19 посвящены файловой системе. Глава 18 фокусируется на файлах: их открытие и закрытие с использованием временных файлов, блокировка файла, посылка сжатых файлов и обработка содержимого файлов. Глава 19 имеет дело с каталогами и метаданными файлов. Она содержит рецепты по изменению прав доступа к файлу и прав владения, перемещению и удалению файла, обработке всех файлов каталога.

И наконец, две главы, расширяющие сферу деятельности PHP. Глава 20 посвящена использованию PHP вне веб-программирования. В ее рецептах рассматриваются темы, связанные с работой в командной строке, анализ аргументов программы и чтение паролей, темы, относящиеся к построению клиентских GUI-приложений с помощью PHP-GTK, например для отображения элементов управления окном, реагирования на действия пользователя и показа меню. В главе 21 рассказано о PEAR, расширении PHP и хранилище приложений. PEAR – это набор программ PHP, предоставляющий различные функции и расширения PHP. Мы используем модули PEAR на всем протяжении книги и показываем, как инсталлировать и обновлять их.

# Другие источники

## Веб-сайты

Существует огромное количество онлайновых справочных материалов по PHP. Быстрое интернет-соединение, предоставляющее все от аннотированных руководств по PHP до сайтов с периодическими статьями и учебными пособиями, достойно конкурирует с большой книжной полкой, заставленной книгами по PHP. Вот некоторые ключевые сайты:

*Аннотированные сведения по PHP: http://www.php.net/manual/*

Сайт переведен на семнадцать языков и включает как официальную документацию, так и возможности языка и комментарии пользователей.

*Списки почтовых рассылок PHP: http://www.php.net/mailing-lists.php*

Здесь много списков почтовых рассылок, посвященных инсталляции, программированию, расширению PHP и различным другим темам. Веб-интерфейс исключительно для чтения списков почтовых рассылок можно найти на *http://news.php.net/*.

*Архивы презентаций PHP: http://conf.php.net/*

Собрание презентаций PHP, показанных на различных конференциях.

*PEAR: http://pear.php.net/*

Здесь PEAR определяется как «структура и система распространения повторно используемых компонентов PHP». На этом сайте можно найти массу полезных PHP-классов и простых программ.

*PHP.net: Путеводитель туриста: http://www.php.net/sites.php*

Это руководство по различным веб-сайтам под эгидой *php.net*.

*База знаний по PHP: http://php.faqts.com/*

Множество вопросов и ответов от сообщества PHP, а также ссылки на другие источники.

*PHP DevCenter: http://www.onlamp.com/php/*

Коллекция статей и учебных пособий по PHP с хорошим сочетанием начальных и более сложных тем.

## Книги

В этом разделе перечислены книги – полезные справочники и учебные пособия по созданию приложений с помощью PHP. Большинство из них посвящены веб-программированию; кроме того, понадобятся книги по MySQL, HTML, XML и HTTP.

В конце раздела мы перечислили несколько книг, полезных для любого программиста, независимо от выбранного языка. Эти работы могут улучшить ваши навыки программирования, научив думать о программировании как о части более масштабного процесса решения задачи.

- «Programming PHP» (Программирование на PHP) Кевина Тэтро (Kevin Tatroe) и Расмуса Лердорфа (Rasmus Lerdorf), O'Reilly.
- «HTML and XHTML: The Definitive Guide» Чака Муссиано (Chuck Musciano) и Билла Кеннеди (Bill Kennedy), O'Reilly.[1]
- «Dynamic HTML: The Definitive Guide» Дэнни Гудмена (Danny Goodman), O'Reilly.
- «Mastering Regular Expressions» Джеффри Фридла (Jeffrey E. F. Friedl), O'Reilly.[2]
- «XML in a Nutshell» Эллиотта Расти Харольда (Elliotte Rusty Harold) и У. Скотта Минса (W. Scott Means), O'Reilly.[3]
- «MySQL Reference Manual» Майкла «Монти» Видениуса (Michael «Monty» Widenius) и Дэвида Эксмарка (David Axmark), O'Reilly и MySQL AB (см. *http://www.mysql.com/documentation/*).
- «MySQL» Поля Дюбуа (Paul DuBois), New Riders.[4]
- «Web Security, Privacy, and Commerce» Симсона Гарфинкеля (Simson Garfinkel) и Жэне Спаффорда (Gene Spafford), O'Reilly.
- «Web Services Essentials» Этана Керами (Ethan Cerami), O'Reilly.
- «HTTP Pocket Reference» Клинтона Вонга (Clinton Wong), O'Reilly.
- «The Practice of Programming», Брайана У. Кернигана (Brian W. Kernighan) и Роба Пайка (Rob Pike), Addison-Wesley.

---

1 Чак Муссиано и Билл Кеннеди «HTML и XHTML. Подробное руководство», 4-е издание. – Пер. с англ. – СПб: Символ-Плюс, 2002.

2 Джеффри Фридл «Регулярные выражения», 2-е издание. – Пер. с англ. – СПб: Питер, 2003.

3 Эллиот Расти Харольд и Скотт Минс «XML. Справочник». – Пер. с англ. – СПб: Символ-Плюс, 2002.

4 Дюбуа П. «MySQL», 2-е издание. – Пер. с англ. – М : Вильямс, 2004.

- «Programming Pearls» Джона Луиса Бентли (Jon Louis Bentley), Addison-Wesley.[1]
- «The Mythical Man-Month» Фредерика П. Брукса (Frederick P. Brooks), Addison-Wesley.[2]

# Соглашения, принятые в этой книге

## Соглашения по программированию

В большинстве случаев мы опускали в примерах этой книги комбинации символов `<?php` и `?>` – открывающий и закрывающий теги, которые начинают и заканчивают программу, написанную на PHP, за исключением тех примеров, где тело программы включает открывающий или закрывающий тег. Для того чтобы свести к минимуму конфликты имен, названия функций и классов в данном сборнике начинаются с префикса `pc_`.

Примеры в этой книге были написаны для выполнения в версии PHP 4.2.2. Простые программы должны работать и в UNIX, и в Windows, за исключением случаев, особо упомянутых в тексте. Некоторые функции, особенно связанные с XML, были написаны для выполнения в версии PHP 4.3.0. Зависимость от возможностей, не представленных в версии PHP 4.2.2, отмечалась в тексте особо.

## Типографские соглашения

В этой книге приняты следующие типографские соглашения:

*Курсив*

Для имен файлов и каталогов, адресов электронной почты и URL, а также для новых терминов в том месте, где они определяются.

`Моноширинный`

Для текстов программ и ключевых слов, имен переменных, функций, командных опций, параметров, классов и HTML-тегов.

**`Моноширинный полужирный`**

Для выделения строк вывода в листинге программы и командных строк, которые должен ввести пользователь.

*`Моноширинный курсив`*

Употребляется для обозначения элементов, которые надо заменить действительными значениями из вашей собственной программы.

---

[1] Бентли Дж. Л. «Жемчужины программирования», 2-е издание – Пер. с англ. – СПб: Питер, 2002.

[2] Фредерик Брукс «Мифический человеко-месяц». – Пер. с англ. – СПб: Символ-Плюс, 2000.

## Комментарии и вопросы

Пожалуйста, направляйте комментарии и вопросы, касающиеся этой книги, издателю:

O'Reilly & Associates, Inc.

1005 Gravenstein Highway North

Sebastopol, CA 95472

(800) 998-9938 (в Соединенных Штатах или Канаде)

(707) 829-0515 (международный/местный)

(707) 829-0104 (факс)

Мы поддерживаем веб-страницу для этой книги, где приводим ошибки, примеры и другую дополнительную информацию. Вот ее адрес:

*http://www.oreilly.com/catalog/phpckbk*

Можно сделать замечание или задать вопрос, послав электронное письмо по адресу:

*bookquestions@oreilly.com*

За дополнительной информацией по этой книге, о конференциях, Resource Centers и O'Reilly Network обращайтесь к веб-сайту O'Reilly:

*http://www.oreilly.com*

## Благодарности

Особая благодарность всем, кто пожертвовал своим временем, не пожалел творческих способностей и знаний, чтобы сделать PHP тем, что он в настоящее время собой представляет. Результатом этих поразительных усилий добровольцев стало создание не только сотни тысяч строк исходного кода, но и всеобъемлющей документации, инфраструктуры тестирования, множества дополнительных приложений и библиотек и бурно растущего сообщества пользователей во всем мире. Мы взволнованы представившейся нам почетной возможностью познакомить сообщество PHP с книгой «PHP. Сборник рецептов».

Благодарим также наших рецензентов: Стига Баккена (Stig Bakken), Шейна Каравео (Shane Caraveo), Айка де Лоренцо (Ike DeLorenzo), Расмуса Лэрдофа (Rasmus Lerdorf), Адама Мортона (Adam Morton), Офира Прусака (Ophir Prusak), Кевина Тэтроу (Kevin Tatroe) и Натана Торкингтона (Nathan Torkington). Они выловили значительное количество ошибок и внесли массу полезных предложений для улучшения этой книги. Мы бы хотели особенно отметить Ната Торкингтона (Nat Torkington), в изобилии снабдившего нас полезными изменениями и добавлениями.

Student.Net Publishing, Student.Com и TVGrid.Com обеспечили богатое окружение для экспериментов с PHP. Появление этой книги стало

возможным во многом благодаря нашему опыту работы в этих сообществах. Брет Мартин (Bret Martin) и Miranda Productions предоставили хостинг и другие средства, обеспечившие возможность нашей удаленной совместной работы во время написания книги. Мы находимся на расстоянии четырех миль друг от друга, но для Манхэттена это удаленно.

Последние по порядку, но не по значению, благодарности нашему редактору, Пауле Фергюсон (Paula Ferguson). Она твердой рукой провела «PHP. Сборник рецептов» через весь издательский процесс в O'Reilly – от того момента, когда она потрясающе быстро (с точки зрения наших друзей) приняла наше скромное предложение издать эту книгу, до окончательной обработки изменений, предложенных нами в последнюю минуту. Без нее эта книга никогда бы не превратилась из идеи в реальность.

## Дэвид Скляр

Благодарю Адама за совместное написание этой книги (и вылавливание всех мест, где я поставил слишком много круглых скобок).

Спасибо моим родителям, которые не догадывались о том, к чему это приведет, когда 20 лет назад купили мне тот 4-килобайтный компьютер Radio Shack Color.

Благодарю Сюзанну (Susannah) за стойкую любовь и поддержку и за то, что в трудные минуты напоминала мне, что жизнь – это не параграф.

## Адам Трахтенберг

Трудно выразить, как я обязан Дэвиду за совместную работу над книгой «PHP. Сборник рецептов». Его замечания радикально улучшили мое умение писать, а его непоколебимая пунктуальность помогала мне придерживаться графика работы.

Благодарю Coleco и их компьютер Adam за то, что они позволили мне вообразить себя первым ребенком, который владеет компьютером, названным в его честь.

Спасибо всем моим друзьям и одноклассникам по бизнес-школе, которым надоело слушать, как я говорю: «Извините, я сегодня вечером должен идти работать над книгой», и кто продолжал разговаривать со мной, после того как я отвечал на их звонки с двухнедельным опозданием.

Особые благодарности Элизабет Хондл (Elizabeth Hondl). Ее по-детски искренняя заинтересованность веб-технологиями доказывает, что если задавать достаточно много вопросов, то из них вполне может получиться книга.

Спасибо моему брату, родителям и всей семье, от которых я получил так много. Их одобрение и любовь поддерживают меня.

# 1

# Строки

## 1.0. Введение

Строки в PHP – это последовательность символов, такая как «We hold these truths to be self evident», или «Жил да был», или даже «111211211». При чтении из файла или выводе в броузер данные представляются в виде строк.

Отдельные символы можно считать элементами индексированного массива, как в C. Первый символ в строке имеет нулевой индекс. Например:

```
$neighbor = 'Hilda';
print $neighbor[3];
d
```

Однако строки в PHP отличаются от строк в C тем, что они могут быть бинарными (то есть могут содержать символ NULL) и могут произвольно менять свой размер, который ограничен лишь объемом доступной памяти.

Строки можно задавать тремя способами: в одинарных кавычках, в двойных кавычках и в формате встроенного документа (heredoc). При использовании одинарных кавычек следует экранировать только два специальных символа: обратную косую черту и саму одинарную кавычку:

```
print 'I have gone to the store.';
print 'I\'ve gone to the store.';
print 'Would you pay $1.75 for 8 ounces of tap water?';
print 'In double-quoted strings, newline is represented by \n';
I have gone to the store.
I've gone to the store.
Would you pay $1.75 for 8 ounces of tap water?
In double-quoted strings, newline is represented by \n
(В строках с двойными кавычками символ новой строки представлен \n)
```

Задавать строки в одинарных кавычках проще и быстрее, поскольку в этом случае PHP не проверяет наличие переменных или почти всех escape-последовательностей. Строки в двойных кавычках не распознают escape-код одинарной кавычки, но распознают вставленные в строку переменные и escape-последовательности, представленные в табл. 1.1.

*Таблица 1.1. Escape-последовательности строки в двойных кавычках*

| Escape-последовательность | Символ |
|---|---|
| \n | Новая строка (ASCII 10) |
| \r | Возврат каретки (ASCII 13) |
| \t | Табуляция (ASCII 9) |
| \\ | Обратная косая черта |
| \$ | Знак доллара |
| \" | Двойные кавычки |
| \{ | Левая фигурная скобка |
| \} | Правая фигурная скобка |
| \[ | Левая скобка |
| \] | Правая скобка |
| от \0 до \777 | Восьмеричное значение |
| от \x0 до \xFF | Шестнадцатеричное значение |

Например:

```
print "I've gone to the store.";
print "The sauce cost \$10.25.";
$cost = '$10.25';
print "The sauce cost $cost.";
print "The sauce cost \$\061\060.\x32\x35.";
I've gone to the store.
The sauce cost $10.25.
The sauce cost $10.25.
The sauce cost $10.25.
```

Последняя строчка кода печатает цену соуса правильно, поскольку символ 1 имеет десятичный ASCII-код 49 и восьмеричный код 061. Символ 0 имеет десятичный ASCII-код 48 и восьмеричный код 060; 2 имеет десятичный ASCII-код 50 и шестнадцатеричный код 32; а 5 имеет десятичный ASCII-код 53 и шестнадцатеричный код 35.

Строки встроенного документа распознают все те вхождения и escape-коды, что и строки в двойных кавычках, но при этом они допускают использование двойных кавычек. Встроенный документ начинается с <<< и метки. Эта метка (без ограничивающих ее пробельных символов) с точкой с запятой в конце оператора (если это необходимо) заканчивает встроенный документ. Например:

```
print <<< END
It's funny when signs say things like:
   Original "Root" Beer
   "Free" Gift
   Shoes cleaned while "you" wait
or have other misquoted words.
END;
It's funny when signs say things like:
   Original "Root" Beer
   "Free" Gift
   Shoes cleaned while "you" wait
or have other misquoted words.
```

При использовании встроенного документа сохраняются новые строки, интервалы и кавычки. Метка конца строки чувствительна к регистру, и согласно принятому соглашению она записывается в верхнем регистре. Таким образом, следующее выражение правильное:

```
print <<< PARSLEY
It's easy to grow fresh:
Parsley
Chives
on your windowsill
PARSLEY;
```

Как и следующее:

```
print <<< DOGS
If you like pets, yell out:
DOGS AND CATS ARE GREAT!
DOGS;
```

Встроенный документ удобен при выводе на печать документа HTML с включенными в него переменными:

```
if ($remaining_cards > 0) {
    $url = '/deal.php';
    $text = 'Deal More Cards';
} else {
    $url = '/new-game.php';
    $text = 'Start a New Game';
}
print <<< HTML
There are <b>$remaining_cards</b> left.
<p>
<a href="$url">$text</a>
HTML;
```

Здесь точка с запятой после ограничителя конца строки необходима, поскольку она сообщает PHP о конце оператора. Однако в некоторых случаях точку с запятой ставить не надо:

```
$a = <<< END
Once upon a time, there was a
```

```
END
. ' boy!';
print $a;
Once upon a time, there was a boy!
```

Точки с запятой нет, поскольку данное выражение необходимо продолжить на следующей строке. Заметим также, что оператор конкатенации строк и ограничитель конца строки следует располагать в разных строках для того, чтобы PHP смог распознать ограничитель.

# 1.1. Доступ к подстрокам

## Задача

Предположим, что требуется выделить часть строки, начиная с определенной позиции. Например, необходимы первые восемь символов имени пользователя, введенного в форму.

## Решение

Для выделения подстроки применяется функция `substr()`:

```
$substring = substr($string,$start,$length);
$username = substr($_REQUEST['username'],0,8);
```

## Обсуждение

Если `$start` и `$length` больше нуля, то функция `substr()` возвращает `$length` символов строки, начиная с позиции `$start`. Первый символ находится в нулевой позиции:

```
print substr('watch out for that tree',6,5);
out f
```

Если пропустить `$length`, то функция `substr()` возвратит строку, начиная с позиции `$start` до конца первоначальной строки:

```
print substr('watch out for that tree',17);
t tree
```

Если `$start` плюс `$length` указывает на позицию за концом, то функция `substr()` возвращает всю строку до конца, начиная с позиции `$start`:

```
print substr('watch out for that tree',20,5);
ree
```

Если число `$start` отрицательное, то функция `substr()` определяет начальную позицию подстроки, считая от конца строки к началу:

```
print substr('watch out for that tree',-6);
print substr('watch out for that tree',-17,5);
t tree
out f
```

Если число $length отрицательное, то substr() определяет конечную позицию подстроки, считая от конца строки к началу:

```
print substr('watch out for that tree',15,-2);
print substr('watch out for that tree',-4,-1);
hat tr
tre
```

## См. также

Документацию по функции substr() на *http://www.php.net/substr*.

# 1.2. Замещение подстрок

## Задача

Требуется заменить подстроку другой строкой. Например, перед тем как напечатать номер кредитной карты, вы хотите скрыть все цифры ее номера, за исключением последних четырех.

## Решение

Используйте функцию substr_replace():

```
// Все, начиная с позиции $start до конца строки $old_string,
// заносится в $new_substring
$new_string = substr_replace($old_string,$new_substring,$start);

// $length символов, начиная с позиции $start, заменяются на $new_substring
$new_string = substr_replace($old_string,$new_substring,$start,$length);
```

## Обсуждение

Без аргумента $length, функция substr_replace() заменяет все, начиная с позиции $start до конца строки. Если значение $length определено, то замещается только это количество:

```
print substr_replace('My pet is a blue dog.','fish.',12);
print substr_replace('My pet is a blue dog.','green',12,4);
$credit_card = '4111 1111 1111 1111';
print substr_replace($credit_card,'xxxx ',0,strlen($credit_card)-4);
My pet is a fish.
My pet is a green dog.
xxxx 1111
```

Если число $start отрицательное, то новая подстрока помещается в позицию $start, считая от конца строки $old_string, а не с начала:

```
print substr_replace('My pet is a blue dog.','fish.',-9);
print substr_replace('My pet is a blue dog.','green',-9,4);
My pet is a fish.
My pet is a green dog.
```

Если $start и $length равны 0, то новая подстрока вставляется в начало $old_string:

```
print substr_replace('My pet is a blue dog.','Title: ',0,0);
Title: My pet is a blue dog.
```

Функция substr_replace() удобна, когда текст невозможно отобразить за один раз, и вы хотите показать его часть со ссылкой на остальное содержание. Например, следующее выражение отображает 25 строк текста с многоточием в качестве ссылки на страницу с продолжением:

```
$r = mysql_query("SELECT id,message FROM messages WHERE id = $id") or die();
$ob = mysql_fetch_object($r);
printf('<a href="more-text.php?id=%d">%s</a>',
       $ob->id, substr_replace($ob->message,' ...',25));
```

На странице *more-text.php* для извлечения полного текста сообщения и его отображения может использоваться идентификатор сообщения, переданный в строке запроса.

## См. также

Документацию по функции substr_replace() на *http://www.php.net/substr-replace*.

# 1.3. Посимвольная обработка строк

## Задача

Нужно обработать каждый символ строки по отдельности.

## Решение

Цикл по символам строки с помощью оператора for. В этом примере подсчитываются гласные в строке:

```
$string = "This weekend, I'm going shopping for a pet chicken.";
$vowels = 0;
for ($i = 0, $j = strlen($string); $i < $j; $i++) {
    if (strstr('aeiouAEIOU',$string[$i])) {
        $vowels++;
    }
}
```

## Обсуждение

Посимвольная обработка – это самый простой способ подсчета последовательности «Смотри и говори»:

```
function lookandsay($s) {
    // инициализируем возвращаемое значение пустой строкой
    $r = '';
    // переменная $m, которая содержит подсчитываемые символы,
```

```
    // инициализируется первым символом * в строке
    $m = $s[0];
    // $n, количество обнаруженных символов $m, инициализируется значением 1
    $n = 1;
    for ($i = 1, $j = strlen($s); $i < $j; $i++) {
        // если символ совпадает с последним символом
        if ($s[$i] == $m) {
            // увеличиваем на единицу значение счетчика этих символов
            $n++;
        } else {
            // иначе добавляем значение счетчика и символа
               к возвращаемому значению //
            $r .= $n.$m;
            // устанавливаем искомый символ в значение текущего символа //
            $m = $s[$i];
            // и сбрасываем счетчик в 1 //
            $n = 1;
        }
    }
    // возвращаем построенную строку, а также последнее значение
       счетчика и символ //
    return $r.$n.$m;
}

for ($i = 0, $s = 1; $i < 10; $i++) {
    $s = lookandsay($s);
    print "$s\n";
}
1
11
21
1211
111221
312211
13112221
1113213211
31131211131221
13211311123113112211
```

Это называется последовательностью «Смотри и говори», поскольку каждый элемент мы получаем, глядя на предыдущие элементы и говоря, сколько их. Например, глядя на первый элемент, 1, мы говорим «один». Следовательно, второй элемент – «11», то есть две единицы, поэтому третий элемент – «21». Он представляет собой одну двойку и одну единицу, поэтому четвертый элемент – «1211» и т. д.

## См. также

Документацию по оператору `for` на *http://www.php.net/for*; дополнительную информацию о последовательности «Смотри и говори» на *http://mathworld.wolfram.com/LookandSaySequence.html*.

# 1.4. Пословный или посимвольный переворот строки

## Задача

Требуется перевернуть слова или символы в строке.

## Решение

Для посимвольного переворота строки применяется функция `strrev()`:

```
print strrev('This is not a palindrome.');
.emordnilap a ton si sihT
```

Чтобы перевернуть строку пословно, надо разобрать строку на слова, перевернуть слова, а затем собрать их заново в строку:

```
$s = "Once upon a time there was a turtle.";
// разбиваем строку на слова
$words = explode(' ',$s);
// обращаем массив слов
$words = array_reverse($words);
// $s = join(' ',$words);
print $s;
turtle. a was there time a upon Once
```

## Обсуждение

Пословное обращение строки может быть также выполнено в одной строке:

```
$reversed_s = join(' ',array_reverse(explode(' ',$s)));
```

## См. также

Рецепт 18.7, в котором рассматриваются последствия применения непробельных символов в качестве границы слов; документацию по функции `strrev()` на *http://www.php.net/strrev* и по функции `array_reverse()` на *http://www.php.net/array-reverse*.

# 1.5. Расширение и сжатие табуляций

## Задача

Нужно заменить пробелы на табуляцию (или табуляцию на пробелы) и в то же время сохранить выравнивание теста по позициям табуляции. Например, вы хотите отобразить для пользователя текст стандартным образом.

## Решение

Для замены пробелов на табуляцию или табуляции на пробелы следует применять функцию `str_replace()`:

```
$r = mysql_query("SELECT message FROM messages WHERE id = 1") or die();
$ob = mysql_fetch_object($r);
$tabbed = str_replace('  ',"\t",$ob->message);
$spaced = str_replace("\t",'  ',$ob->message);

print "With Tabs: <pre>$tabbed</pre>";
print "With Spaces: <pre>$spaced</pre>";
```

Однако если для преобразования применяется функция `str_replace()`, то позиции табуляции нарушаются. Если вы хотите ставить табуляцию через каждые восемь символов, то в строке, начинающейся с пятибуквенного слова и табуляции, необходимо заменить табуляцию на три пробела, а не на один. Для замены табуляции на пробелы с учетом позиций табуляции следует применять функцию `pc_tab_expand()`, показанную в примере 1.1.

*Пример 1.1. pc_tab_expand()*

```
function pc_tab_expand($a) {
  $tab_stop = 8;
  while (strstr($a,"\t")) {
    $a = preg_replace('/^([^\t]*)(\t+)/e',
                      "'\\1'.str_repeat(' ',strlen('\\2') *
                       $tab_stop - strlen('\\1') % $tab_stop)",$a);
  }
  return $a;
}

$spaced = pc_tab_expand($ob->message);
```

Для обратной замены пробелов на табуляцию можно воспользоваться функцией `pc_tab_unexpand()`, показанной в примере 1.2.

*Пример 1.2. pc_tab_unexpand()*

```
function pc_tab_unexpand($x) {
  $tab_stop = 8;

  $lines = explode("\n",$x);
  for ($i = 0, $j = count($lines); $i < $j; $i++) {
    $lines[$i] = pc_tab_expand($lines[$i]);
    $e = preg_split("/(.\{$tab_stop})/",$lines[$i],-
1,PREG_SPLIT_DELIM_CAPTURE);
    $lastbit = array_pop($e);
    if (!isset($lastbit)) { $lastbit = ''; }
    if ($lastbit == str_repeat(' ',$tab_stop)) { $lastbit = "\t"; }
    for ($m = 0, $n = count($e); $m < $n; $m++) {
      $e[$m] = preg_replace('/  +$',"\t",$e[$m]);
    }
```

```
        $lines[$i] = join(``,$e).$lastbit;
      }
      $x = join("\n", $lines);
      return $x;
    }

    $tabbed = pc_tab_unexpand($ob->message);
```

Обе функции принимают в качестве аргумента строку и возвращают ее, модифицировав соответствующим образом.

## Обсуждение

Каждая функция предполагает наличие позиций табуляции через каждые восемь пробелов, но это можно изменить, задав переменную `$tab_stop`.

Регулярное выражение в `pc_tab_expand()` соответствует и группе табуляций, и всему тексту в строке перед группой табуляций. Оно должно соответствовать тексту перед табуляциями, поскольку от длины этого текста зависит количество пробелов, замещающих табуляции, а последующий текст должен быть выровнен по позиции следующей табуляции. Эта функция не просто заменяет каждую табуляцию на восемь пробелов; она выравнивает текст, стоящий после табуляции, по позициям табуляций.

Точно так же функция `pc_tab_unexpand()` не только ищет восемь последовательных пробелов, а затем заменяет их одним символом табуляции. Она делит каждую строку на участки по восемь символов, а затем замещает пробелы в конце этих участков (по крайней мере два пробела) на табуляции. Это не только сохраняет выравнивание текста по позициям табуляций, но и сохраняет пробелы в строке.

## См. также

Документацию по функции `str_replace()` на *http://www.php.net/str-replace*.

# 1.6. Управление регистром

## Задача

Необходимо переключиться на прописные буквы, или в нижний регистр или иным образом изменить регистр символов в строке. Например, требуется сделать прописными начальные буквы имен и сохранить нижний регистр остальных букв.

## Решение

Первые буквы одного или более слов можно сделать прописными с помощью функции `ucfirst()` или функции `ucwords()`:

```
print ucfirst("how do you do today?");
print ucwords("the prince of wales");
How do you do today?
The Prince Of Wales
```

Регистр всей строки изменяется функцией `strtolower()` или функцией `strtoupper()`:

```
print strtoupper("i'm not yelling!");
// Стандарт XHTML требует, чтобы символы в тегах были в нижнем регистре
print strtolower('<A HREF="one.php">one</A>');
I'M NOT YELLING!
<a href="one.php">one</a>
```

## Обсуждение

Первый символ строки можно сделать прописным посредством функции `ucfirst()`:

```
print ucfirst('monkey face');
print ucfirst('1 monkey face');
Monkey face
1 monkey face
```

Обратите внимание, что во второй строке вывода слово «monkey» начинается со строчной буквы.

Функция `ucwords()` позволяет сделать прописным первый символ каждого слова в строке:

```
print ucwords('1 monkey face');
print ucwords("don't play zone defense against the philadelphia 76-ers");
1 Monkey Face
Don't Play Zone Defense Against The Philadelphia 76-ers
```

Как и следовало ожидать, функция `ucwords()` не делает прописной букву «t» в слове «don't». Но она также не делает прописной букву «e» в «76-e». Для функции `ucwords()` слово – это любая последовательность непробельных символов, за которой расположен один или несколько пробельных. Символы «'» и «-» не являются пробельными, поэтому функция `ucwords()` не считает «t» в «don't» или «e» в «76-e» начальными символами слов.

Ни `ucfirst()`, ни `ucwords()` не изменяют регистр не первых символов:

```
print ucfirst('macWorld says I should get a iBook');
print ucwords('eTunaFish.com might buy itunaFish.Com!');
MacWorld says I should get a iBook
ETunaFish.com Might Buy ItunaFish.Com!
```

Функции `strtolower()` и `strtoupper()` работают с целыми строками, а не только с отдельными символами. Функция `strtolower()` переводит все алфавитные символы в нижний регистр, а функция `strtoupper()` – в верхний:

```
print strtolower("I programmed the WOPR and the TRS-80.");
print strtoupper('"since feeling is first" is a poem by e. e. cummings.');
i programmed the wopr and the trs-80.
"SINCE FEELING IS FIRST" IS A POEM BY E. E. CUMMINGS.
```

При определении верхнего и нижнего регистров приоритетными для этих функций являются их локальные настройки.[1]

## См. также

Дополнительную информацию о локальных настройках в главе 16; документацию по функции `ucfirst()` на *http://www.php.net/ucfirst*, по функции `ucwords()` на *http://www.php.net/ucwords*, по функции `strtolower()` на *http://www.php.net/strtolower* и по функции `strtoupper()` на *http://www.php.net/strtoupper*.

# 1.7. Включение функций и выражений в строки

## Задача

Вставить результаты выполнения функции или выражения в строку.

## Решение

Когда значение, которое необходимо вставить в строку, не может быть в нее включено, следует применять оператор конкатенации строк (.):

```
print 'You have '.($_REQUEST['boys'] + $_REQUEST['girls']).' children.';
print "The word '$word' is ".strlen($word).' characters long.';
print 'You owe '.$amounts['payment'].' immediately';
print "My circle's diameter is ".$circle->getDiameter().' inches.';
```

## Обсуждение

Можно поместить переменные, свойства объекта и элементы массива (если индекс не в кавычках) непосредственно в строку в двойных кавычках:

```
print "I have $children children.";
print "You owe $amounts[payment] immediately.";
print "My circle's diameter is $circle->diameter inches.";
```

Точно так же непосредственная вставка или конкатенация строк работает во встроенном документе. Вставка с помощью конкатенации строк во встроенном документе может выглядеть немного странно, поскольку ограничитель встроенного документа и оператор конкатенации должны располагаться в разных строках:

---

[1] Для того чтобы все эти функции работали с русскими буквами, необходимо установить в окружении сценария нужную локализацию с помощью функции `setlocale()`. – *Примеч. науч. ред.*

```
print <<< END
Right now, the time is
END
. strftime('%c') . <<< END
 but tomorrow it will be
END
. strftime('%c',time() + 86400);
```

Кроме того, если вы производите вставку во встроенный документ, не забудьте добавить пробелы так, чтобы вся строка выглядела правильно. В предыдущем примере строка «Right now the time» должна включать замыкающий пробел, а строка «but tomorrow it will be» должна включать пробелы в начале и в конце.

## См. также

Описание синтаксиса размещения так называемых «переменных переменных» (таких как `${"amount_$i"}`) в рецепте 5.4; документацию по оператору конкатенации строк на *http://www.php.net/language.operators.string*.

# 1.8. Удаление пробельных символов из строки

## Задача

Надо удалить пробельные символы в начале или в конце строки. Например, привести в порядок данные, введенные пользователем, прежде чем счесть их действительными.

## Решение

Следует обратиться к функциям `ltrim()`, `rtrim()` или `trim()`. Функция `ltrim()` удаляет пробельные символы в начале строки, `rtrim()` — в конце строки, а функция `trim()` — и в начале, и в конце строки:

```
$zipcode = trim($_REQUEST['zipcode']);
$no_linefeed = rtrim($_REQUEST['text']);
$name = ltrim($_REQUEST['name']);
```

## Обсуждение

Эти функции считают пробельными следующие символы: символ новой строки, возврат каретки, пробел, горизонтальную и вертикальную табуляции и символ NULL.

Удаление пробельных символов из строки позволяет сэкономить память и может сделать более корректным отображение форматированных данных или текста, например, содержащегося в тегах `<pre>`. При проверке пользовательского ввода сначала нужно обрезать пробелы, так чтобы не заставлять того, кто ввел «98052 » вместо своего почтово-

го индекса, исправлять ошибку, которая, собственно, таковой не является. Отбрасывание начальных и конечных пробелов в тексте перед его проверкой означает, например, что «salami\n» будет равно «salami». Неплохо также нормализовать данные путем обрезания пробелов перед их занесением в базу данных.

Кроме того, функция `trim()` способна удалять из строки символы, определенные пользователем. Удаляемые символы передаются в качестве второго аргумента. Можно указать интервал символов с помощью двоеточия между первым и последним символом интервала.

```
// Удаление цифр и пробела в начале строки
print ltrim('10 PRINT A$',' 0..9');
// Удаление точки с запятой в конце строки
print rtrim('SELECT * FROM turtles;',';');
PRINT A$
SELECT * FROM turtles
```

PHP рассматривает `chop()` как синоним `rtrim()`. Однако лучше использовать `rtrim()`, поскольку поведение функции `chop()` в PHP отличается от поведения `chop()` в Perl (вместо которой в любом случае лучше применять функцию `chomp()`), а ее применение может затруднить другим чтение вашего кода.

## См. также

Документацию по функции `trim()` на *http://www.php.net/trim*, по функции `ltrim()` на *http://www.php.net/ltrim* и по функции `rtrim()` на *http://www.php.net/rtrim*.

# 1.9. Анализ данных, разделенных запятой

## Задача

Есть данные, разделенные запятыми (формат CSV), например, файл, экспортированный из Excel или из базы данных, и необходимо извлечь записи и поля в формате, с которым можно работать в PHP.

## Решение

Если CSV-данные представляют собой файл (или они доступны через URL), то откройте файл с помощью функции `fopen()` и прочитайте данные с помощью функции `fgetcsv()`. Данные будут представлены в виде HTML-таблицы:

```
$fp = fopen('sample2.csv','r') or die("can't open file");
print "<table>\n";
while($csv_line = fgetcsv($fp,1024)) {
    print '<tr>';
    for ($i = 0, $j = count($csv_line); $i < $j; $i++) {
        print '<td>'.$csv_line[$i].'</td>';
```

```
    }
    print "</tr>\n";
}
print '</table>\n';
fclose($fp) or die("can't close file");
```

## Обсуждение

Второй аргумент в `fgetcsv()` должен превышать максимальную длину строки в вашем CSV-файле. (Не забудьте посчитать пробельные символы, ограничивающие строку.) Если длина читаемой строки превышает 1 Кбайт, то число 1024, использованное в данном рецепте, надо заменить на действительную длину строки.

Функции `fgetcsv()` можно передать необязательный третий параметр, ограничитель, используемый вместо запятой. Однако применение другого ограничителя до некоторой степени лишает смысла CSV как наиболее простого способа обмена табличными данными.

Не старайтесь избегать функции `fgetcsv()`, а просто читайте строку и вызывайте функцию `explode()` в случае запятых. CSV является более сложным, когда имеешь дело с внедренными запятыми и двойными кавычками. Использование функции `fgetcsv()` защитит ваш код от трудноуловимых ошибок.

## См. также

Документацию по функции `fgetcsv()` на *http://www.php.net/fgetcsv*.

# 1.10. Анализ данных, состоящих из полей фиксированной ширины

## Задача

Необходимо разбить на части записи фиксированной ширины в строке.

## Решение

Это делается при помощи функции `substr()`:

```
$fp = fopen('fixed-width-records.txt','r') or die ("can't open file");
while ($s = fgets($fp,1024)) {
    $fields[1] = substr($s,0,10);       // первое поле:
                                           первые 10 символов строки
    $fields[2] = substr($s,10,5);       // второе поле:
                                           следующие 5 символов строки
    $fields[3] = substr($s,15,12);      // третье поле:
                                           следующие 12 символов строки
    // функция обработки полей
    process_fields($fields);
}
fclose($fp) or die("can't close file");
```

Или функции `unpack()`:

```
$fp = fopen('fixed-width-records.txt','r') or die ("can't open file");
while ($s = fgets($fp,1024)) {
    // ассоциативный массив с ключами "title", "author" и "publication_year"
    $fields = unpack('A25title/A14author/A4publication_year',$s);
    // функция обработки полей
    process_fields($fields);
}
fclose($fp) or die("can't close file");
```

## Обсуждение

Данные, в которых каждому полю выделено фиксированное число символов в строке, могут выглядеть, как этот список книг, названий и дат опубликования:

```
$booklist=<<<END
Elmer Gantry             Sinclair Lewis1927
The Scarlatti InheritanceRobert Ludlum 1971
The Parsifal Mosaic      Robert Ludlum 1982
Sophie's Choice          William Styron1979
END;
```

В каждой строке название занимает 25 символов, имя автора – следующие 14 символов, а год публикации – следующие 4 символа. Зная ширину полей, очень просто с помощью функции `substr()` перенести поля в массив.

```
$books = explode("\n",$booklist);

for($i = 0, $j = count($books); $i < $j; $i++) {
  $book_array[$i]['title'] = substr($books[$i],0,25);
  $book_array[$i]['author'] = substr($books[$i],25,14);
  $book_array[$i]['publication_year'] = substr($books[$i],39,4);
}
```

Разбиение переменной `$booklist` на массив строк позволяет применить один код разбора одной строки ко всем строкам, прочитанным из файла.

Цикл можно сделать более гибким, определив отдельные массивы для имен полей и их ширины, которые могут быть переданы в анализирующую функцию, как показано в функции `pc_fixed_width_substr()` примера 1.3.

*Пример 1.3. pc_fixed_width_substr( )*

```
function pc_fixed_width_substr($fields,$data) {
  $r = array();
  for ($i = 0, $j = count($data); $i < $j; $i++) {
    $line_pos = 0;
    foreach($fields as $field_name => $field_length) {
      $r[$i][$field_name] = rtrim(substr($data[$i],$line_pos,$field_length));
      $line_pos += $field_length;
    }
```

```
  }
  Return $r;
}

$book_fields = array('title' => 25,
                     'author' => 14,
                     'publication_year' => 4);

$book_array = pc_fixed_width_substr($book_fields,$books);
```

Переменная `$line_pos` отслеживает начало каждого поля, и она увеличивается на ширину предыдущего поля по мере того, как код обрабатывает каждую строку. Для удаления пробельных символов в конце каждого поля предназначена функция `rtrim()`.

Как альтернатива функции `substr()` для извлечения полей может применяться функция `unpack()`. Вместо того чтобы задавать имена полей и их ширину в виде ассоциативных массивов, создайте строку форматирования для функции. Код для извлечения полей фиксированной ширины аналогичен функции `pc_fixed_width_unpack()`, показанной в примере 1.4.

*Пример 1.4. pc_fixed_width_unpack( )*

```
function pc_fixed_width_unpack($format_string,$data) {
  $r = array();
  for ($i = 0, $j = count($data); $i < $j; $i++) {
    $r[$i] = unpack($format_string,$data[$i]);
  }
  return $r;
}

$book_array = pc_fixed_width_unpack('A25title/A14author/A4publication_year',
                                    $books);
```

Формат `A` означает «строку в обрамлении пробелов», поэтому нет необходимости удалять завершающие пробелы с помощью функции `rtrim()`.

Поля, перенесенные с помощью какой-либо функции в переменную `$book_array`, могут быть отображены в виде HTML-таблицы, например:

```
$book_array = pc_fixed_width_unpack('A25title/A14author/A4publication_year',
                                    $books);
print "<table>\n";
// печатаем строку заголовка
print '<tr><td>';
print join('</td><td>',array_keys($book_array[0]));
print "</td></tr>\n";
// печатаем каждую строку данных
foreach ($book_array as $row) {
    print '<tr><td>';
    print join('</td><td>',array_values($row));
    print "</td></tr>\n";
}
print '</table>\n';
```

Объединение данных с помощью тегов </td><td> формирует строку таблицы, не включая в нее начальный <td> и заключительный </td> теги. Печатая <tr><td> перед выводом объединенных данных и </td></tr> вслед за выводом объединенных данных, мы формируем полную строку таблицы.

И функция substr(), и функция unpack() имеют одинаковые возможности, если поля фиксированной ширины содержат строки, но функция unpack() является наилучшим решением при наличии полей других типов данных.

## См. также

Более подробную информацию о функции unpack() в рецепте 1.13 и по адресу *http://www.php.net/unpack*; рецепт 4.8, который посвящен функции join().

# 1.11. Разбиение строк

## Задача

Необходимо разделить строку на части. Например, нужно получить доступ к каждой из строк, которые пользователь вводит в поле <textarea> формы.

## Решение

Если в качестве разделителя частей строк выступает строковая константа, то следует применять функцию explode():

```
$words = explode(' ','My sentence is not very complicated');
```

Функция split() или функция preg_split() применяются, если при описании разделителя требуется регулярное выражение POSIX или Perl:

```
$words = split(' +','This sentence  has  some extra whitespace  in it.');
$words = preg_split('/\d\. /','my day: 1. get up 2. get dressed 3. eat toast');
$lines = preg_split('/[\n\r]+/',$_REQUEST['textarea']);
```

В случае чувствительного к регистру разделителя применяется функция spliti() или флаг /i в функции preg_split():

```
$words = spliti(' x ','31 inches x 22 inches X 9 inches');
$words = preg_split('/ x /i','31 inches x 22 inches X 9 inches');
```

## Обсуждение

Простейшим решением из всех приведенных выше является использование explode(). Передайте ей разделитель строки, саму строку, которую необходимо разделить, и в качестве необязательного параметра предельное количество возвращаемых элементов:

```
$dwarves = 'dopey,sleepy,happy,grumpy,sneezy,bashful,doc';
$dwarf_array = explode(',',$dwarves);
```

Теперь переменная `$dwarf_array` — это массив из семи элементов:

```
print_r($dwarf_array);
Array
(
    [0] => dopey
    [1] => sleepy
    [2] => happy
    [3] => grumpy
    [4] => sneezy
    [5] => bashful
    [6] => doc
)
```

Если заданный предел меньше количества возможных частей, то последняя часть содержит все остальное:

```
$dwarf_array = explode(',',$dwarves,5);
print_r($dwarf_array);
Array
(
    [0] => dopey
    [1] => sleepy
    [2] => happy
    [3] => grumpy
    [4] => sneezy,bashful,doc
)
```

Функция `explode()` трактует разделитель строки буквально. Если разделитель строки определяется как запятая с пробелом, то данная функция делит строку по пробелу, следующему за запятой, а не по запятой или пробелу.

Функция `split()` предоставляет большую гибкость. Вместо строкового литерала в качестве разделителя она использует регулярное выражение POSIX:

```
$more_dwarves = 'cheeky,fatso, wonder boy, chunky,growly, groggy, winky';
$more_dwarf_array = split(', ?',$more_dwarves);
```

Это регулярное выражение разделяет строку по запятой, за которой следует необязательный пробел, что позволяет правильно определить всех новых гномов. Таким образом, пробелы в их именах не разделяют их на части, но каждое имя выделяется независимо от того, отделяется ли оно с помощью запятой «,» или с помощью запятой с пробелом «, »:

```
print_r($more_dwarf_array);
Array
(
    [0] => cheeky
    [1] => fatso
    [2] => wonder boy
```

```
    [3] => chunky
    [4] => growly
    [5] => groggy
    [6] => winky
)
```

Существует функция `preg_split()`, которая подобно функции `split()` использует Perl-совместимые регулярные выражения вместо регулярных выражений POSIX. Функция `preg_split()` предоставляет преимущества различных расширений регулярных выражений в Perl, а также хитрые приемы, такие как включение текста-разделителя в возвращаемый массив строк:

```
$math = "3 + 2 / 7 - 9";
$stack = preg_split('/ *([+\-\/*]) */',$math,-1,PREG_SPLIT_DELIM_CAPTURE);
print_r($stack);
Array
(
    [0] => 3
    [1] => +
    [2] => 2
    [3] => /
    [4] => 7
    [5] => -
    [6] => 9
)
```

Разделитель-регулярное выражение ищет математические операторы (`+`, `-`, `/`, `*`), окруженные необязательными начальными или завершающими пробелами. Флаг `PREG_SPLIT_DELIM_CAPTURE` приказывает функции `preg_split()` включить совпадения как часть разделителя-регулярного выражения, заключенного в кавычки, в возвращаемый строковый массив. В кавычках только символы математических операций, поэтому возвращенный массив не содержит пробелов.

## См. также

Документацию по функции `explode()` на *http://www.php.net/explode*, по функции `split()` на *http://www.php.net/split* и по функции `preg_split()` на *http://www.php.net/preg-split*. Регулярные выражения более подробно рассматриваются далее.

# 1.12. Упаковка текста в строки определенной длины

## Задача

Необходимо упаковать линии текста в строку. Например, нужно отобразить текст, содержащийся в тегах `<pre>`/`</ pre>`, в пределах окна броузера обычного размера.

## Решение

Это делается при помощи функции `wordwrap()`:

```
$s = "Four score and seven years ago our fathers brought forth on this
continent a new nation, conceived in liberty and dedicated to the proposition
that all men are created equal.";

print "<pre>\n".wordwrap($s)."\n</pre>";
<pre>
Four score and seven years ago our fathers brought forth on this continent
a new nation, conceived in liberty and dedicated to the proposition that
all men are created equal.
</pre>
```

## Обсуждение

По умолчанию функция `wordwrap()` упаковывает текст в строки по 75 символов. Необязательный второй аргумент позволяет изменять длину строки:

```
print wordwrap($s,50);
Four score and seven years ago our fathers brought
forth on this continent a new nation, conceived in
liberty and dedicated to the proposition that all
men are created equal.
```

Для указания конца строки можно использовать не только символы «\n». Для получения двойного интервала между строками используйте «\n\ n»:

```
print wordwrap($s,50,"\n\n");
Four score and seven years ago our fathers brought

forth on this continent a new nation, conceived in

liberty and dedicated to the proposition that all

men are created equal.
```

В функции есть необязательный четвертый аргумент, управляющий обработкой слов, длина которых превышает указанную длину строки. Если этот аргумент равен 1, то слова разбиваются на части. В противном случае они простираются за указанный предел длины строки:

```
print wordwrap('jabberwocky',5);
print wordwrap('jabberwocky',5,"\n",1);
jabberwocky

jabbe
rwock
y
```

## См. также

Документацию по функции `wordwrap()` на *http://www.php.net/wordwrap*.

# 1.13. Хранение двоичных данных в строках

## Задача

Необходимо проанализировать строку, которая содержит значения, закодированные с помощью двоичной структуры, или закодировать значения в строку. Например, нужно сохранить числа в их двоичном представлении, а не как последовательность ASCII-символов.

## Решение

Для сохранения двоичных данных в строке применяется функция `pack()`:

```
$packed = pack('S4', 1974, 106, 28225, 32725);
```

Функция `unpack()` позволяет извлекать двоичные данные из строки:

```
$nums = unpack('S4',$packed);
```

## Обсуждение

Первым аргументом функции `pack()` является строка формата, который описывает способ кодировки данных, передаваемых остальными аргументами. Строка формата `S4` указывает, что функция `pack()` должна сформировать из входных данных четыре беззнаковых коротких целых (short) 16-битных числа в соответствии с машинным порядком байтов. В качестве входа даны числа 1974, 106, 28225 и 32725, а возвращаются восемь байт: 182, 7, 106, 0, 65, 110, 213 и 127. Каждая двухбайтная пара соответствует входному числу: $7 \times 256 + 182 = 1974$; $0 \times 256 + 106 = 106$; $110 \times 256 + 65 = 28225$; $127 \times 256 + 213 = 32725$.

Первый аргумент функции `unpack()` – это тоже строка формата, а второй аргумент представляет декодируемые данные. В результате передачи строки формата, равной `S4`, восьмибайтная последовательность, произведенная функцией `pack()`, приводит к получению четырехэлементного массива исходных чисел:

```
print_r($nums);
Array
(
    [1] => 1974
    [2] => 106
    [3] => 28225
    [4] => 32725
)
```

В функции `unpack()` за символом форматирования и его множителем может следовать строка, которая выступает в качестве индекса массива. Например:

```
$nums = unpack('S4num',$packed);
print_r($nums);
```

```
Array
(
    [num1] => 1974
    [num2] => 106
    [num3] => 28225
    [num4] => 32725
)
```

В функции `unpack()` несколько символов форматирования должны разделяться символом /:

```
$nums = unpack('S1a/S1b/S1c/S1d',$packed);
print_r($nums);
Array
(
    [a] => 1974
    [b] => 106
    [c] => 28225
    [d] => 32725
)
```

В табл. 1.2 приведены символы форматирования, которые можно использовать в функциях `pack()` и `unpack()`.

*Таблица 1.2. Символы форматирования функций pack( ) и unpack( )*

| Символ форматирования | Тип данных |
|---|---|
| a | Строка, дополненная символами NUL |
| A | Строка, дополненная пробелами |
| h | Шестнадцатеричная строка, первый полубайт младший |
| H | Шестнадцатеричная строка, первый полубайт старший |
| c | signed char |
| C | unsigned char |
| s | signed short (16 бит, машинный порядок байтов) |
| S | unsigned short (16 бит, машинный порядок байтов) |
| n | unsigned short (16 бит, обратный порядок байтов) |
| v | unsigned short (16 бит, прямой порядок байтов) |
| i | signed int (машинно-зависимый размер и порядок байтов) |
| I | unsigned int (машинно-зависимый размер и порядок байтов) |
| l | signed long (32 бита, машинный порядок байтов) |
| L | unsigned long (32 бита, машинный порядок байтов) |
| N | unsigned long (32 бита, обратный порядок байтов) |
| V | unsigned long (32 бита, прямой порядок байтов) |
| f | float (машинно-зависимый размер и представление) |

*Таблица 1.2 (продолжение)*

| Символ форматирования | Тип данных |
|---|---|
| d | double (машинно-зависимый размер и представление) |
| x | NUL-байт |
| X | Возврат на один байт |
| @ | Заполнение нулями по абсолютному адресу |

Для a, A, h и H число после символа форматирования означает длину строки. Например, A25 означает строку из 25 символов, дополненную пробелами. В случае других символов форматирования это число означает количество данных указанного типа, последовательно появляющихся в строке. Остальные возможные данные можно указать при помощи символа «*».

С помощью функции unpack() можно преобразовывать различные типы данных. Этот пример заполняет массив ASCII-кодами каждого символа, находящегося в $s:

```
$s = 'platypus';
$ascii = unpack('c*',$s);
print_r($ascii);
Array
(
    [1] => 112
    [2] => 108
    [3] => 97
    [4] => 116
    [5] => 121
    [6] => 112
    [7] => 117
    [8] => 115
)
```

## См. также

Документацию по функции pack() на *http://www.php.net/pack* и по функции unpack() на *http://www.php.net/unpack*.