ПРОФЕССИОНАЛЬНО

Пит Гудлиф, Робърт Мартин,
Дэвидэлис Стинеллис, Кевлин Хенни и др.

97

ЭТЮДОВ ДЛЯ ПРОГРАММИСТОВ

O'REILLY®

СИМВОЛ®

16 лет вместе
с профессионалами

97 Things Every Programmer Should Know

Collective Wisdom from the Experts

Pete Goodliffe, Robert Martin,
Diomidis Spinellis, Kevlin Henney and others

Edited by Kevlin Henney

O'REILLY®

ПРОФЕССИОНАЛЬНО

97 этюдов для программистов

Опыт ведущих экспертов

Пит Гудлиф, Роберт Мартин,
Диомидис Спинеллис, Кевлин Хенни и др.

под редакцией Кевлина Хенни

СИМВОЛ®

Санкт-Петербург — Москва
2012

Серия «Профессионально»

Пит Гудлиф, Роберт Мартин,
Диомидис Спинеллис, Кевлин Хенни и др.

97 этюдов для программистов
Опыт ведущих экспертов

Перевод С. Маккавеева

Главный редактор	*А. Галунов*
Зав. редакцией	*Н. Макарова*
Науч. редактор	*А. Долгушин*
Редактор	*М. Зислис*
Корректор	*О. Макарова*
Верстка	*Д. Орлова*

Гудлиф П., Мартин Р., Спинеллис Д., Хенни К. и др.

97 этюдов для программистов. Опыт ведущих экспертов. – Пер. с англ. – СПб.: Символ-Плюс, 2012. – 256 с., ил.

ISBN 978-5-93286-198-1

Приобщитесь к мудрости экспертов и запомните то, что должен знать каждый программист, с каким бы языком и на какой платформе он ни работал. 97 кратких и очень полезных советов повысят ваш профессионализм посредством новых подходов к старым проблемам, лучших практик и разумных подсказок, предназначенных для оттачивания мастерства.

Авторы этой книги, очень опытные и признанные в отрасли специалисты, передадут вам практические знания и принципы, полезные для проектов любого типа. Статьи касаются разных тем: от рекомендаций по написанию кода до культуры, от выбора алгоритмов до гибкого программирования, от приемов реализации до профессионализма, от стиля до сущности. Новички смогут познакомиться с фундаментальными положениями, а для профессионалов сборник сможет стать отправной точкой для обсуждений.

ISBN 978-5-93286-198-1
ISBN 978-0-596-80948-5 (англ)

Издательство «Символ-Плюс». 199034, Санкт-Петербург, 16 линия, 7,
тел. (812) 380-5007, www.symbol.ru. Лицензия ЛП N 000054 от 25.12.98.
Подписано в печать 29.02.2012. Формат 70×90 $^{1}/_{16}$.
Печать офсетная. Объем 16 печ. л.

Отсутствующим друзьям посвящается

Оглавление

Статьи по категориям

Ошибки и исправления

Сборка и развертывание

Правила написания и форматирования кода

Принципы проектирования и техника написания кода

Предметно-ориентированное мышление

Ошибки, их обработка и исключения

Обучение, мастерство и опыт

Ночное и волшебное

Производительность, оптимизация и представление

Профессионализм, мировоззрение и позиция

Языки и парадигмы программирования

Рефакторинг и забота о коде

Повторное использование и повторение кода

Графики, сроки и оценки

Простота

Командная работа и сотрудничество

Тесты, тестирование и тестировщики

Инструменты, автоматизация, среды разработки

Пользователи и заказчики

Предисловие

Новейший компьютер способен лишь с большей скоростью усложнить древнейшую проблему отношений между людьми, и в конечном итоге участнику общения по-прежнему придется решать, что и как говорить.

Эдвард Р. Мэроу (Edward R. Murrow)

Программистам есть, над чем думать. Языки программирования, приемы программирования, среды разработки, стили написания кода, инструменты, процессы разработки, планы работ, совещания, архитектуры программ, шаблоны проектирования, динамика командного взаимодействия, код, технические требования, дефекты, качество кода. И другое. Много чего еще.

Здесь мы находим искусство, ремесло и науку, которые простираются далеко за рамки программы. Деятельность программиста объединяет дискретный мир компьютеров и текучий мир человеческих занятий. Программисты служат связующим звеном между бизнесом с его расплывчатыми договорными истинами и выверенной, бескомпромиссной областью, где царят биты, байты и построенные на их основе пользовательские типы.

Учитывая объемы знаний, работы и разнообразие способов ее выполнения, никакой человек или источник не может претендовать на знание «истинного пути». Поэтому, опираясь на народную мудрость и накопленный опыт, книга «97 этюдов для программистов» предлагает не столько упорядоченную общую картину, сколько пеструю мозаику мнений о том, что должно быть известно каждому программисту. Она касается разных тем: от рекомендаций по написанию кода до культуры, от выбора алгоритмов до гибкого программирования, от приемов реализации до профессионализма, от стиля до сущности.

Отдельные статьи не стыкуются между собой, да и цель ставилась скорее противоположная. Ценность отдельной статьи здесь как раз в том, что она не похожа на другие. А ценность сборника в целом состоит в том, что статьи дополняют, подтверждают одна другую и даже противоречат друг другу. Они не связаны общим сюжетом: читатель сам может оценить материал, поразмышлять над ним и увязать прочитанное, сравнив новое с собственными контекстом, знаниями и опытом.

Лицензионные права

Контакты

На веб-странице книги перечислены найденные ошибки и приводятся дополнительные сведения:

http://www.oreilly.com/catalog/9780596809485/

Сопроводительный сайт, где опубликованы все статьи, биографии авторов и другие данные, находится по адресу:

http://programmer.97things.oreilly.com

Вы также можете следить за новостями и исправлениями книги в Twitter:

http://twitter.com/97TEPSK

Комментарии и технические вопросы, касающиеся этой книги, можно отправить электронной почтой:

bookquestions@oreilly.com

Дополнительная информация о наших книгах, Центрах ресурсов и сети O'Reilly Network приведена на нашем веб-сайте:

http://www.oreilly.com/

Safari® Books Online

Safari Books Online – цифровая библиотека, которая дает возможность быстро находить ответы на ваши вопросы в 7500 технических книг, справочников и видеозаписей.

Подписка Safari дает право читать любую страницу и смотреть любое видео в режиме онлайн. Читайте книги на сотовых телефонах и мобильных устройствах. Получайте доступ к новым изданиям до выхода их из печати. Получайте эксклюзивный доступ к рукописям в процессе работы и отправляйте замечания авторам. Копируйте текст примеров кода, загружайте главы, создавайте закладки и заметки, печатайте страницы – вот лишь некоторые из множества функций, экономящих ваше время.

O'Reilly Media опубликовала эту книгу в Safari Books Online. Чтобы получить полный цифровой доступ к этой книге и книгам схожей тематики, выпущенным

O'Reilly и другими издательствами, оформите бесплатную подписку на *http://my.safaribooksonline.com*.

Благодарности

Проекту «97 этюдов для программистов» прямо или косвенно отдали свое время и знания многие люди. Все они заслуживают благодарности.

Ричард Монсон-Хейфел (Richard Monson-Haefel) – редактор серии «97 Things» и редактор первой книги из этой серии, «97 Things Every Software Architect Should Know»[1], в написании которой я принимал участие. Спасибо Ричарду за идею серии и за ее открытость для потенциальных участников, а также за то, что он так энергично поддерживал мои предложения по данной книге.

Хочу поблагодарить всех тех, кто отдал свое время и силы, участвуя в создании текстов для этого проекта: как тех, чьи статьи опубликованы в этой книге, так и тех, чьи тексты не попали в нее, но опубликованы на веб-сайте. Большое количество и высокое качество материала весьма затруднили процесс окончательного отбора – жестко фиксированное в названии книги число не оставило места для дополнительных статей.

Я также благодарен за дополнительные отзывы, комментарии и предложения, авторами которых были Джованни Аспрони (Giovanni Asproni), Пол Колин Глостер (Paul Colin Gloster) и Микаэль Хунгер (Michael Hunger).

Спасибо O'Reilly за предоставленную этому проекту поддержку – от вики-хостинга, сделавшего книгу реальностью, до обеспечения всех стадий процесса публикации в бумажном виде. Сотрудники O'Reilly, которых я хотел бы особо поблагодарить: Майк Лукидес (Mike Loukides), Лорел Акерман (Laurel Ackerman), Эди Фридман (Edie Freedman), Эд Стивенсон (Ed Stephenson) и Рейчел Монахан (Rachel Monaghan).

Дело не только в том, что текст книги рождался в среде Веб: через Веб проект также приобрел известность и популярность. Спасибо всем, кто распространял сведения о нем через социальные сети, блоги и прочими путями.

Хочу также поблагодарить свою жену Кэролин за то, что привносит порядок в мой хаос, и двух моих сыновей, Стефана и Янника, за то, что часть этого порядка они вновь превращают в хаос.

Надеюсь, что эта книга станет для вас источником информации, открытий и вдохновения.

Приятного чтения!

Кевлин Хенни (Kevlin Henney)

---

[1] Нил Форд, Майкл Найгард, Билл де Ора и др. «97 этюдов для архитекторов программных систем». – Пер. с англ. – СПб.: Символ-Плюс, 2010.

Будьте благоразумны

Себ Роуз

В любом деле будь благоразумен и думай о последствиях.

Неизвестный

Как бы успокаивающе ни выглядел график работы в начале итерации, в какой-то момент неизбежно возникает нехватка времени. Если приходится разрываться между «сделать правильно» и «сделать быстро», часто возникает соблазн «сделать быстро» с оговоркой, что вы исправите решение позже, когда появится время. Вы совершенно искренне даете обещание именно так и поступить – даете самому себе, команде или заказчику. Но очень часто на следующей итерации возникают уже другие проблемы, которым и приходится посвящать свое внимание. Такую отложенную работу называют *техническим долгом*, и хорошего от него не жди. В своей классификации технических долгов Мартин Фаулер называет такой вид долга *умышленным техническим долгом*, и его не следует путать с *непреднамеренным техническим долгом*.[1]

Технический долг подобен кредиту: в краткосрочной перспективе он выгоден, но по нему приходится выплачивать проценты до полного погашения займа. Срезая углы при написании кода, вы затрудняете как разработку новой функциональности, так и рефакторинг. Это создает благоприятную почву для появления ошибок и нестабильных *тестовых сценариев* (*test cases*). Чем дольше долг существует, тем тяжелее последствия. К тому времени, как дойдут руки внести запланированные исправления, может оказаться, что на основе изначального сомнительного кода уже выстроена целая гора не вполне верных с точки зрения проектирования решений, а это значительно осложнит рефакторинг и исправление этого кода. На самом деле, к решению изначальной проблемы часто возвращаются лишь тогда, когда *уже нет выбора*, кроме как вернуться и все исправить. И зачастую к этому моменту исправление оказывается уже настолько

[1] *http://martinfowler.com/bliki/TechnicalDebtQuadrant.html*

сложным, что вы просто не можете себе позволить потерять так много времени или пойти на подобный риск.

Бывают ситуации, когда приходится идти на создание технического долга, если необходимо уложиться в срок или частично реализовать некую функциональность. Старайтесь не оказываться в таких ситуациях, однако если положение совершенно безвыходное, действуйте. Но (и это увесистое *но*) вы должны вести учет своего технического долга и гасить его как можно скорее, иначе проблемы растут, как снежный ком. И если уж вы пошли на такой компромисс, составьте карточку с заданием (task card) или создайте запись в системе учета дефектов, чтобы не забыть о проблеме.

Если вы планируете погасить свой долг на следующей итерации, потери будут минимальными. На непогашенный долг капают проценты, за которыми нужно постоянно следить, чтобы видеть реальную конечную цену. Это подчеркивает влияние технического долга проекта на его бизнес-стоимость и позволяет разумно расставлять акценты в вопросах погашения такого долга. Способ начисления и отслеживания процентов зависит от конкретного проекта, но отслеживать их должны вы.

Гасите технические долги как можно скорее. Поступать иначе – неблагоразумно.

Применяйте принципы функционального программирования

Эдвард Гарсон

Функциональное программирование недавно снова обратило на себя внимание большинства в сообществе программистов. Отчасти благодаря тому, что *эмерджентные свойства* функциональной парадигмы созвучны решению задач, возникающих в нашей отрасли в связи с ростом значимости многоядерных архитектур. И хотя данное применение, несомненно, важно, однако не оно является главным основанием для моего наставления *познать функциональное программирование*.

Овладев парадигмой функционального программирования, программист может значительно повысить качество кода, создаваемого в других контекстах. Глубокое понимание парадигмы функционального программирования и ее применение на практике помогут вам проектировать системы, обладающие гораздо большей степенью *ссылочной прозрачности* (*referential transparency*).

Ссылочная прозрачность является качеством очень желательным: она предполагает, что функции неизменно дают одинаковые результаты на одинаковых входных данных независимо от места и времени обращения к этим функциям. Вычисление функции, таким образом, слабо зависит от побочных эффектов изменяющегося (mutable) состояния – в идеале не зависит от них вообще.

Один из главных источников дефектов в коде на императивном языке программирования – изменяемые (mutable) переменные. Каждому читателю наверняка приходилось разбираться, почему в каком-либо конкретном случае некоторое значение не соответствовало ожидаемому. Семантика областей видимости может препятствовать появлению таких коварных ошибок или, по крайней мере, значительно сужать возможную область их появления. Но истинной причиной их возникновения может быть сама концепция проектирования такого кода, который беспорядочно полагается на изменяемость (mutability).

И в этом отношении нам определенно не стоит ждать особой помощи от собственной отрасли. Вводные тексты по объектно-ориентированному программированию скрыто пропагандируют подобные конструкции. В них часто приводятся