Предисловие

Помню, как Дэйв и Энди впервые сообщили в Tweeter о новом издании этой книги. И это было важное известие. Я наблюдал, с каким воодушевлением сообщество программистов встретило это известие, и оживленно откликнулся на него, предвидя, что двадцать лет спустя настоящее издание окажется столь же востребованным, как и прежде.

И то, что книга с подобной историей получила такие отклики, говорит о многом. Мне выпала честь прочитать рукопись этой книги, чтобы написать настоящее предисловие к ней, и тогда я понял, почему она вызвала такой переполох. И хотя это техническая литература, назвав ее таковой, можно сослужить ей плохую службу. Ведь техническая литература нередко отпугивает своим содержимым, изобилующим сложными описаниями, непонятными терминами, замысловатыми примерами, невольно вызывающими у читателя ощущение собственной тупости. Чем опытнее автор книги, тем легче читателю забыть, что при изучении новых концепций он является начинающим.

Несмотря на многолетний опыт программирования, Дэйву и Энди пришлось преодолеть немалые трудности, чтобы писать книгу с воодушевлением людей, только что освоивших уроки данной профессии. Они не обращаются с читателем свысока, не считают его знатоком и даже не предполагают, что он читал первое издание книги. Они воспринимают читателей такими, какими они есть на самом деле, — программистами, которые стремятся совершенствоваться. И чтобы помочь читателям в этом, они посвящают данной цели немало страниц своей книги, делая один практический шаг за другим.

Откровенно говоря, они уже делали это и прежде. Ведь первое издание книги изобиловало реальными примерами, новыми идеями и практическими советами, которые и ныне актуальны для приобретения навыков программирования и выработки умения мыслить как программист. И все же в настоящее обновленное издание были внесены два важных усовершенствования.

Первое усовершенствование вполне очевидно и состоит в исключении устаревших ссылок и примеров, которые были заменены новым, современным содержимым. В настоящем издании вы не найдете примеры инвариантов циклов или сборочных машин. Не обращаясь к прежним примерам, Дэйв и Энди постарались составить материал таким образом, чтобы читатели смогли извлечь из него наибольшую пользу, прорабатывая отдельные уроки. В настоящем издании такие старые принципы, как “не повторяться” (don’t repeat yourself, DRY), очи-

щены он накопившейся пыли и отреставрированы, и сверкают новыми яркими красками.

Но по-настоящему привлекательным настоящее издание делает именно второе усовершенствование. После написания первого издания книги у ее авторов была возможность поразмыслить над тем, что они пытались в ней сказать, что ее читатели должны были вынести из нее и воспринять ее так, как хотелось бы авторам. Они получили отзывы на первое издание, выяснили, что было не так, что требовало уточнения, а что было неверно понято. За двадцать лет книга дошла до рук и сердец программистов во всем мире, а Дэйв и Энди, изучив отзывы о ней, сформулировали новые принципы и понятия.

Они усвоили важность поддержки и осознали, что поддержка, бесспорно, присуща разработчикам программного обеспечения в большей степени, чем представителям большинства других профессий. Поэтому они начали настоящее издание с простого, но мудрого посыла: "это ваша жизнь". Он напоминает нам о нашем потенциале в развитии созданной нами кодовой базы, нашей работе и карьере. Он задает тон всей остальной части книги, которая представляет собой нечто большее, чем очередная техническая литература, изобилующая примерами исходного кода.

Эта книга выгодно отличается от многочисленной технической литературы на данную тему ясным пониманием ее авторов, что в действительности означает быть программистом. Программирование — это попытка облегчить будущее, а следовательно, труд коллег. Это умение быстро оправиться после совершенных оплошностей, воспитание хороших привычек и овладение арсеналом инструментальных средств. Но само программирование является лишь частью мира программиста, который исследуется в этой книге.

Мне пришлось потратить немало времени, размышляя о жизненном пути в программировании. Я не вырос на программировании, не изучал его в колледже и не увлекался в юности техникой. Я приобщился к области программирования к 25 годам, когда мне пришлось самому разбираться, что означает быть программистом. Сообщество программистов заметно отличается от других сообществ, в которые мне довелось входить. В этом сообществе заметна особая преданность обучению и практичности, которая не только оживляет, но и отпугивает.

Для меня вхождение в такое сообщество сродни открытию совершенно нового мира или, по крайней мере, нового города, где мне нужно познакомиться с соседями, выбрать свой продуктовый магазин, найти лучшую из кофеен. Потребуется немало времени, чтобы сориентироваться на местности, найти наиболее эффективные маршруты, избегать улиц с интенсивным движением, знать, когда наиболее вероятны транспортные пробки. Погодные условия также могут быть иными, чем в старом месте жительства, а следовательно, придется сменить свой гардероб.

Первые несколько недель, а то и месяцев пребывания в новом городе могут оказаться ужасными. И в этот период было бы замечательно иметь дружелюбного знающего соседа, давно живущего в городе. Кто может провести для вас экскурсию по кофейням? Только тот, кто хорошо знает местную культуру, чувствует пульс жизни города и поможет вам не только почувствовать себя как дома, но и стать полезным членом сообщества. И такими соседями для вас, читатели, окажутся Дэйв и Энди.

У относительно начинающего может легко вызвать потрясение не столько сам акт программирования, сколько процесс его становления как программиста. Ведь для этого потребуется определенный сдвиг в образе мышления, изменение привычек, видов поведения и ожиданий. Процесс вашего становления лучшим программистом не произойдет лишь потому, что вы знаете, как программировать. Ему должно удовлетворять твердое намерение и тщательно продуманная практика. И эта книга служит руководством по эффективному становлению лучшим программистом.

Но имейте в виду, что в книге ничего не говорится о том, как следует программировать. Никаких философских рассуждений и обоснований по этому поводу вы в ней не найдете. Авторы просто и ясно поясняют, как стать программистом-прагматиком и выбрать свой подход к программированию, оставляя за читателями право самим решать, желают ли они стать именно такими программистами. Если вы считаете, что быть программистом-прагматиком — не для вас, то авторы ничего против этого не будут иметь. Но если вы все же решите им стать, они станут вашими дружелюбными соседями, показывающими путь к желанной цели.

Сарон Йитбарек (Saron Yitbarek),
учредитель и генеральный директор компании
CodeNewbie Host of Command Line Heroes.

Вступление ко второму изданию

В далекие 1990-е годы мы работали с компаниями, испытывающими затруднения в своих проектах. И оказалось, что мы каждый раз спрашивали у них одно и то же: “Может, вам лучше сначала тестировать свои программные продукты, а затем поставлять их? Почему код создается только на машинах разработчиков и никто не спрашивает о нем мнения пользователей?”

Чтобы сэкономить время новых клиентов, мы стали делать заметки, которые в конечном счете превратились в эту книгу. И к нашему удивлению, книга, по-видимому, нашла живой отклик у читателей, не потеряв свою популярность в течение прошедших двадцати лет.

Но двадцать лет — немалый срок для разработки программного обеспечения. Если попытаться переместить разработчика во времени из 1999 года и ввести его в современную команду, он будет бороться с чуждым ему новым миром. Но ведь и мир 1990-х годов в равной степени чужд современному разработчику. Ссылки в настоящем издании на литературу по CORBA, CASE, индексированным циклам были бы в лучшем случае забавны и, вероятнее всего, сбивали бы с толку.

В то же время прошедшие двадцать лет не оказали никакого влияния на то, что называется общими принципами. Ведь могла измениться технология, но не люди. Некогда пригодные нормы практики и методы остаются таковыми и поныне. Эти вопросы, рассматривавшиеся ранее в книге, теперь вполне созрели.

И когда настало время выпустить настоящее 20-летнее юбилейное издание, нам пришлось принять нелегкое решение. С одной стороны, мы могли бы просмотреть и обновить описание тех технологий, на которые мы ссылались в предыдущем издании, и на этом поставить точку. А с другой стороны, мы могли бы пересмотреть свои рекомендации по поводу тех норм практики, которые мы рекомендовали раньше, исходя из опыта, накопившегося за прошедшие двадцать лет. Но в конечном счете мы сделали и то и другое.

В итоге книга стала чем-то подобным *кораблю Тесея*[1]. Около одной трети тем в ней оказались совершенно новыми, а остальные были частично или полностью переписаны. Мы намеревались сделать изложение материала более понятным, уместным и, как можно надеяться, непреходящим.

[1] Известный философский парадокс: если с годами заменить каждую вышедшую из строя часть корабля, то останется ли он в итоге тем же самым кораблем?

Нам пришлось принять ряд трудных решений. В частности, мы опустили приложение "Первоисточники" как потому, что было бы невозможно сохранить его актуальность, так и потому, что нужные первоисточники читателю проще найти самостоятельно. Кроме того, мы реорганизовали и переписали темы, касающиеся параллелизма, принимая во внимание современное изобилие аппаратного обеспечения для параллельных вычислений и недостаток способов их организации. Наконец, мы добавили материал, отражающий перемены в отношениях и средах: от движения за гибкую разработку, началу которого мы способствовали, до растущего принятия идиом функционального программирования и потребности принимать во внимание конфиденциальность и безопасность.

Любопытно, однако, что споров по поводу содержимого настоящего издания у нас было меньше, чем при написании первого издания этой книги. Мы оба считали, что выявить важный материал на этот раз было проще.

Так или иначе, настоящее издание увидело свет, так что пользуйтесь им во благо, возможно, приняв новые нормы практики, а может быть, решив, что некоторая часть предлагаемого нами материала вам не подходит. Но в любом случае мы надеемся, что вы глубже вникнете в свое ремесло и откликнетесь на нашу книгу. А самое главное, не забудьте позабавиться.

Структура книги

Эта книга написана как собрание кратких тем, каждая из которых самодостаточна и посвящена конкретному предмету. В тексте вы найдете многочисленные перекрестные ссылки, помогающие ввести каждую тему в соответствующий контекст. Отдельные темы можно читать в произвольном порядке, поскольку эта книга составлена таким образом, чтобы не читать ее от корки до корки.

Периодически в книге вам будут встречаться врезки под заголовком "Совет *nn*" (например, "Совет 1. Заботьтесь о своем ремесле"). Мы рассматриваем подобные советы как обращающие на себя внимание места в тексте книги. Они имеют самостоятельное значение и служат для того, чтобы пользоваться ими ежедневно в практической деятельности.

В настоящем издании были включены упражнения и задачи там, где это было уместно. Как правило, упражнения имеют относительно простые решения, тогда как задачи допускают многие решения. Чтобы дать представление о ходе своих рассуждений, мы выделили решения упражнений в отдельное приложение к книге, хотя лишь немногие из них имеют единственное *правильное* решение. А предлагаемые задачи могут послужить основанием для групповых обсуждений или самостоятельных работ на специализированных курсах по программированию. Кроме того, в конце книги приведен перечень литературы (книг и статей), на которую делаются явные ссылки в тексте самой книги.

Что означает имя

“— Когда я беру слово, оно означает то, что я хочу, не больше и не меньше, — сказал Шалтай-Болтай презрительно”[2].

Льюис Кэрролл, ***Алиса в Зазеркалье***

Текст книги испещрен различными профессиональными терминами: как обычными словами, несколько искаженными, чтобы выразить какой-то особый технический смысл, так и жуткими словами, намеренно выбранными для обозначения отдельных понятий из вычислительной техники теми специалистами в данной области, которые пренебрежительно относятся к языку. При первом употреблении термина делается попытка определить понятие, которое он обозначает, или хотя бы намекнуть на его смысловое значение. Но, на наш взгляд, одни термины так и остались невостребованными, тогда как другие (например, *объект* и *реляционная база данных*) — настолько распространены, что не требуют дополнительного определения. Если же вам встретится незнакомый прежде термин, не пропускайте его, а найдите время, чтобы выяснить его обозначение в Интернете или справочнике по вычислительной технике. И если ваши поиски завершатся удачно, то сообщите нам по электронной почте, чтобы мы добавили найденное вами определение термина в последующее издание книги.

Несмотря на все сказанное выше, мы все же решили отомстить специалистам по вычислительной технике. В частности, мы решили пренебречь некоторыми вполне пригодными терминами для обозначения отдельных понятий. Почему? А потому, что существующая ныне терминология, как правило, ограничивается конкретной предметной областью или стадией разработки. Но ведь один из основных принципов данной книги состоит в том, что бо́льшая часть рекомендуемых нами методик оказывается универсальной. Так, модульность применима к исходному коду, проектным решениям, документации и организации команд разработчиков. Когда же требовалось употребить обычный термин в более широком контексте, то возникало недоразумение, поскольку нам не удавалось преодолеть границы той смысловой нагрузки, которую нес первоначальный термин. И когда происходило нечто подобное, мы решали пренебречь языковыми нормами, изобретая свои термины.

[2] Перевод Н.М. Демуровой.

Исходный код и другие ресурсы

Бо́льшая часть исходного кода, приведенного в этой книге, взята из компилируемых исходных файлов и свободно доступна для загрузки с нашего веб-сайта[3]. Там приведены также ссылки на другие полезные ресурсы наряду с обновлениями книги и новостями о других разработках программистов-прагматиков.

Присылайте нам свои отзывы

Мы будем рады вашим откликам на книгу. Пишите нам по адресу электронной почты `ppbook@pragprog.com`.

Благодарности за второе издание книги

Мы выгодно воспользовались буквально тысячами интересных бесед о программировании, проведенных за последние двадцать лет, встречая разных людей на конференциях, курсах, а иногда и просто в самолете. Каждая из таких бесед расширила наше представление о процессе разработки программного обеспечения и способствовала обновлению настоящего издания. Благодарим всех участников этих бесед и просим известить нас, если мы в чем-то не правы.

Благодарим также участников процесса создания бета-версии этой книги. Ваши вопросы и комментарии помогли нам лучше разъяснить отдельные вопросы, затронутые в книге.

Прежде чем перейти к бета-версии этой книги, мы поделились ее рукописью с несколькими людьми, чтобы они оставили свои комментарии к ней. Благодарим ВМ (Вики) Брассера (VM (Vicky) Brasseur), Джеффа Лангра (Jeff Langr) и Кима Шриера (Kim Shrier) за подробные комментарии, а Хозе Валима (Jose Valim) и Ника Катберта (Nick Cuthbert) — за техническое рецензирование.

Благодарим Рона Джеффриса (Ron Jeffries) за предоставленную нам возможность воспользоваться примером из судоку. Выражаем большую признательность сотрудникам издательства Pearson, согласившихся на то, чтобы мы составили эту книгу по-своему. Особая благодарность выражается незаменимой Джанет Ферлоу (Janet Furlow), которая мастерски справляется со всем, за что бы она ни бралась. В частности, она следила за тем, чтобы мы поэтапно выполняли свою работу над книгой в срок.

Наконец, выражаем во всеуслышание слова искренней признательности всем программистам-прагматикам, сделавшим программирование лучше для всех за последние двадцать лет. Теперь остается еще двадцать лет.

[3] См. по адресу `https://pragprog.com/titles/tpp20/source_code`.

Из вступления к первому изданию

Эта книга поможет вам стать более совершенным программистом. Вы можете быть самостоятельным разработчиком, членом команды в крупном проекте или консультантом, одновременно работающим со многими компаниями. Так или иначе, эта книга поможет вам в индивидуальном плане лучше выполнять свою работу. Она не носит теоретический характер, а сосредоточена на практических вопросах и применении накопленного опыта для принятия более обоснованных решений. Термин *прагматик* происходит от латинского слова *pragmaticus*, обозначающего "опытный в деле" и производного, в свою очередь, от греческого слова πραγματικός, обозначающего "пригодный для использования". Это книга о делании.

Программирование — это ремесло. В простейшем случае оно сводится к тому, чтобы заставить компьютер сделать то, что требуется программисту или пользователю его программы. Программист выступает отчасти в роли слушателя, советчика, переводчика и диктатора. Он пытается уяснить туманные исходные требования и найти способ выразить их таким образом, чтобы вычислительная машина смогла воздать им должное. Он старается задокументировать и организовать результаты своих трудов таким образом, чтобы другие могли их понять и опереться на них. Более того, программист старается сделать все это, несмотря на неумолимые сроки выполнения проекта. Каждый день программист творит небольшие чудеса, а это трудное дело.

Многие люди готовы оказать помощь программисту. Так, поставщики инструментальных средств на все лады расхваливают чудеса, которые способны творить их программные продукты. Знатоки методологии обещают, что их методики гарантированно принесут ожидаемые результаты. И каждый заявляет, что его язык программирования самый лучший, а операционная система дает ответы на все мыслимые вопросы.

Все это, конечно, не соответствует действительности, ведь простых ответов не бывает. Не существует *самого лучшего* решения, будь то инструментальное средство, язык программирования или операционная система. А могут быть лишь такие системы, которые более пригодны в конкретных обстоятельствах.

Именно здесь и может пригодиться прагматизм, который означает, что не следует привязываться к какой-нибудь конкретной технологии, но нужно иметь достаточно обширные знания и опыт, позволяющие принимать правильные решения в конкретных обстоятельствах. Знания проистекают из понимания основных принципов вычислительной техники, а опыт — из обширного ряда практических проектов. Теория в сочетании с практикой составляют прочный фундамент для программиста.

Свой подход программист приспосабливает к текущим обстоятельствам и окружению. Он оценивает относительную важность всех факторов, оказывающих влияние на проект и пользуется своим опытом для выработки подходящих решений. И делает он это постоянно по мере продвижения своей работы. Программисты-прагматики доводят свою работу до конца и делают ее хорошо.

Кому адресована эта книга

Она адресована тем, кто стремится стать более эффективным и продуктивным программистом. Вы, вероятно, испытываете разочарование от того, что не раскрываете, как вам кажется, свой истинный потенциал. А может быть, вы обращаете внимание на своих коллег, которые пользуются инструментальными средствами с целью повысить производительность своего труда. Возможно, в своей нынешней работе вы пользуетесь старыми технологиями и хотите узнать, как приложить новые идеи к тому, что вы делаете.

Мы не претендуем ни на то, чтобы ответить на все (или хотя бы большинство) вопросы, возникающие у программистов, ни на применимость наших идей во всех возможных случаях. Мы можем лишь сказать, что если вы выберете наш подход, то быстро приобретете необходимый опыт, производительность вашего труда возрастет и вы станете лучше понимать весь процесс разработки программного обеспечения. И в конечном счете вы сможете писать более качественные программы.

Что означает быть программистом-прагматиком

Каждый разработчик неповторим, отличаясь своими сильными и слабыми сторонами, симпатиями и антипатиями. Со временем каждый разработчик создает свою собственную среду, отражающую его индивидуальность в той же неумолимой степени, как и его увлечения, одежда или прическа. Но если вы являетесь программистом-прагматиком, то вам присущи многие из перечисленных ниже общих характеристик.

- **Своевременно принимаете новшества на вооружение и быстро приспосабливаетесь к ним.** Инстинктивно чувствуете преимущества отдельных технологий и методик, и вам нравится опробовать их. Если вам дают что-то новое, вы быстро осваиваете его, присоединяя приобретенное к остальным своим знаниям. Ваша уверенность рождается с опытом.
- **Любознательны.** Стремитесь задавать вопросы. Например: как вы это сделали? Возникли ли у вас трудности с данной библиотекой? Что такое квантовые вычисления, о которых я слышал? Каким образом реализуются символические ссылки? Задавая подобные вопросы, вы как барахольщик собираете мелкие факты, каждый из которых может оказать влияние на какое-нибудь решение многие годы спустя.

- **Мыслите критически.** Редко принимаете что-то на веру, не получив сначала подтверждающие факты. Так, если коллеги говорят: “Потому что надо делать так!” или если поставщик обещает решить все ваши проблемы, вы нюхом чуете предстоящие трудности.
- **Реалистичны.** Стараетесь понять потаенный характер каждого возникающего у вас затруднения. Подобный реализм дает вам ясное понимание, в чем именно состоят трудности и как долго их придется преодолевать. Глубоко понимая, что процесс *должен* быть трудным или его завершение *отнимет* время, вы приобретаете необходимую выдержку, чтобы справиться с ним.
- **Мастер на все руки.** Прилежно стараетесь освоить обширный ряд технологий и сред, работая над тем, чтобы быть в курсе новых разработок. И хотя ваша текущая работа может потребовать специальной квалификации, вы всегда готовы перейти в новые сферы деятельности и принять новые вызовы.

Мы оставили рассмотрение большинства основных характеристик напоследок. Они присущи всем программистам-прагматикам и настолько просты, что их можно сформулировать в виде отдельных советов, как показано ниже.

Совет 1 Заботьтесь о своем ремесле

Мы считаем, что разрабатывать новое программное обеспечение нет смысла, если не позаботиться о том, чтобы делать это как следует.

Совет 2 Думайте! О своей работе

Чтобы стать программистом-прагматиком, мы призываем вас думать о том, что вы делаете в тот момент, когда вы это делаете. Это не единовременная ревизия текущих норм практики, а непрерывная критическая оценка каждого принимаемого решения, производимая каждый день и в каждом проекте. Никогда не двигайтесь на автопилоте. Постоянно думайте, критически оценивая свою работу в реальном времени. Старый девиз корпорации IBM “ДУМАЙ!” (THINK!) является мантрой для программиста-прагматика.

Если такая работа покажется вам трудной, значит, вы выказываете *реалистичную* характеристику. На это потребуется немного вашего драгоценного времени, которого, вероятно, и так не хватает. Но в награду за усердие вы более активно вовлекаетесь в свое любимое дело, чувствуя, что вполне владеете неуклонно расширяющимся рядом предметов, а также получаете удовольствие от

постоянного совершенствования. А в долгосрочной перспективе потраченное вами время окупится сторицей, когда вы и ваша команда станете работать более эффективно, чтобы писать легко сопровождаемый код и проводить меньше времени на совещаниях.

Индивидуальные прагматики и крупные команды

Некоторые считают, что индивидуальности нет места в крупных командах или сложных проектах. "Программное обеспечение — техническая дисциплина, — говорят они, — которая нарушается, если отдельные члены команды принимают решения самостоятельно". Мы категорически не согласны с таким утверждением.

В построении программного обеспечения *должна* присутствовать техническая составляющая, но это совсем не мешает проявлять индивидуальное мастерство. Рассмотрим в качестве примера крупные кафедральные соборы, построенные в Европе в период Средневековья. Воздвижение каждого из них потребовало немалых затрат, измеряемых тысячами человеко-часов в течение многих десятилетий. Усвоенные уроки передавались от одного поколения строителей к другому поколению, которое совершенствовало строительную технику своими индивидуальными достижениями. Но ведь плотники, каменотесы, резчики и стеклодувы были ремесленниками, воплощавшими технические требования в нечто целое, выходившее за пределы исключительно механической стороны строительства. *Даже те, кто лишь тешут камни, должны всегда представлять себе целые соборы*.

Во всей структуре проекта всегда найдется место индивидуальности и мастерству. И это особенно справедливо для текущего состояния разработки программного обеспечения. Это состояние может показаться через сто лет таким же архаичным, как и методы, применявшиеся средневековыми строителями кафедральных соборов, современным инженерам-строителям, хотя мастерство в данной отрасли по-прежнему в почете.

Это непрерывный процесс

> *Турист, однажды посетивший Итонский колледж в Англии, спросил садовника, как ему удается поддерживать лужайку в идеальном состоянии. "Очень просто, — ответил тот. — Нужно лишь смахивать росу каждое утро, стричь траву каждый день и сволакивать ее раз в неделю".*
>
> *"И это все?" — спросил турист.*
>
> *"Абсолютно все! — ответил садовник. — Делайте это в течение 500 лет, и у вас тоже получится прекрасная лужайка".*

Прекрасные лужайки требуют небольшого ухода, как, впрочем, и отличные программисты. Консультанты по управленческим вопросам любят вставлять словечко *кайзен* в свои беседы. Термин *кайзен* по-японски обозначает понятие непрерывного процесса внесения многих мелких усовершенствований, что послужило одной из главных причин для коренных изменений в производительности и качестве изготовления японских товаров и широко размножившееся по всему миру. Кайзен применяется и на индивидуальном уровне, где отдельные личности ежедневно работают над совершенствованием собственных навыков, пополняя свой арсенал новыми инструментальными средствами. Но, в отличие от итонских лужаек, они начинают замечать результаты буквально через считанные дни. А с годами они изумляются, насколько расцвел их личный опыт, и развились их индивидуальные навыки.

От издательства

Вы, читатель этой книги, и есть главный ее критик и комментатор. Мы ценим ваше мнение и хотим знать, что было сделано нами правильно, что можно было сделать лучше и что еще вы хотели бы увидеть изданным нами. Нам интересно услышать и любые другие замечания, которые вам хотелось бы высказать в наш адрес.

Мы ждем ваших комментариев и надеемся на них. Вы можете прислать нам электронное сообщение, либо просто посетить наш веб-сайт и оставить свои замечания там. Одним словом, любым удобным для вас способом дайте нам знать, нравится или нет вам эта книга, а также выскажите свое мнение о том, как сделать наши книги более интересными для вас.

Посылая письмо или сообщение, не забудьте указать название книги и ее авторов, а также ваш обратный адрес. Мы внимательно ознакомимся с вашим мнением и обязательно учтем его при отборе и подготовке к изданию последующих книг.

Наши электронные адреса:

E-mail: `info@dialektika.com`

WWW: `http://www.dialektika.com`