30 главных
правил чистого,
безопасного
и быстрого
кода

КРАСИВЫЙ C++

**Создатель языка C++
Бьерн Страуструп
рекомендует**

ДЖ. ГАЙ ДЭВИДСОН / КЕЙТ ГРЕГОРИ

Beautiful C++

30 Core Guidelines for Writing Clean, Safe, and Fast Code

J. Guy Davidson
Kate Gregory

Addison-Wesley

Boston • Columbus • New York • San Francisco • Amsterdam • Cape Town
Dubai • London • Madrid • Milan • Munich • Paris • Montreal • Toronto • Delhi • Mexico City
São Paulo • Sydney • Hong Kong • Seoul • Singapore • Taipei • Tokyo

Красивый C++

30 главных правил чистого, безопасного и быстрого кода

Дж. Гай Дэвидсон
Кейт Грегори

ПИТЕР®
Санкт-Петербург • Москва • Минск
2023

ББК 32.973.2-018.1
УДК 004.43
Д94

Дэвидсон Дж. Гай, Грегори Кейт

Д94 Красивый C++: 30 главных правил чистого, безопасного и быстрого кода. — СПб.: Питер, 2023. — 368 с.: ил. — (Серия «Для профессионалов»).

ISBN 978-5-4461-2272-1

Написание качественного кода на C++ не должно быть трудной задачей. Если разработчик будет следовать рекомендациям, приведенным в C++ Core Guidelines, то он будет писать исключительно надежные, эффективные и прекрасно работающие программы на C++. Но руководство настолько переполнено советами, что порой трудно понять, с чего начать. Начните с «Красивого C++»!

Опытные программисты Гай Дэвидсон и Кейт Грегори выбрали 30 основных рекомендаций, которые посчитали особенно ценными, и дают подробные практические советы, которые помогут улучшить ваш стиль разработки на C++. Для удобства книга структурирована в точном соответствии с официальным веб-сайтом C++ Core Guidelines.

16+ (В соответствии с Федеральным законом от 29 декабря 2010 г. № 436-ФЗ.)

ББК 32.973.2-018.1
УДК 004.43

ISBN 978-0137647842 англ.
ISBN 978-5-4461-2272-1

Оглавление

ЧАСТЬ II
НЕ НАВРЕДИТЕ СЕБЕ

ЧАСТЬ III
ПРЕКРАТИТЕ ЭТО ИСПОЛЬЗОВАТЬ

ЧАСТЬ IV
ИСПОЛЬЗУЙТЕ НОВУЮ ОСОБЕННОСТЬ ПРАВИЛЬНО

Брину,

Шинейд,

Рори и Лоис.

С.47: с любовью, Гай Дэвидсон

Джиму Эллисону, хотя он едва ли прочитает эти строки.
Весь в исследовательской работе. Хлое и Аише, которые
прежде не упоминались на первых страницах книг.

Кейт Грегори

Избранные рекомендации по C++

P.2. Придерживайтесь стандарта ISO C++ (глава 1.1).

https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#Rp-Cplusplus

P.4. В идеале программа должна быть статически типобезопасной (глава 5.1).

https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#Rp-typesafe

P.10. Неизменяемые данные предпочтительнее изменяемых (глава 5.2).

https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#Rp-mutable

P.11. Инкапсулируйте беспорядочные конструкции, а не разбрасывайте их по всему коду (глава 2.1).

https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#Rp-library

I.3. Избегайте синглтонов (глава 3.2).

https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#Ri-singleton

I.11. Никогда не передавайте владение через простой указатель (T*) или ссылку (T&) (глава 3.1).

https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#Ri-raw

I.23. Минимизируйте число параметров в функциях (глава 2.2).

https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#Ri-nargs

I.26. Если нужен кросс-компилируемый ABI, используйте подмножество в стиле C (глава 2.3).

https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#Ri-abi

I.30. Инкапсулируйте нарушения правил (глава 5.3).

https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#Ri-encapsulate

F.21. Для возврата нескольких выходных значений используйте структуры или кортежи (глава 4.1).

https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#Rf-out-multi

F.51. Если есть выбор, используйте аргументы по умолчанию вместо перегрузки (глава 1.2).

https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#Rf-default-args

C.45. Не определяйте конструктор по умолчанию, который просто инициализирует переменные-члены; для этой цели лучше использовать внутриклассовые инициализаторы членов (глава 1.3).

https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#Rc-default

C.47. Определяйте и инициализируйте переменные-члены в порядке их объявления (глава 2.4).

https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#Rc-order

C.90. Полагайтесь на конструкторы и операторы присваивания вместо memset и memcpy (глава 3.3).

https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#Rc-memset

C.131. Избегайте тривиальных геттеров и сеттеров (глава 1.4).

https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#c131-avoid-trivial-getters-and-setters

Enum.3. Старайтесь использовать классы-перечисления вместо простых перечислений (глава 4.2).

https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#Renum-class

ES.5. Минимизируйте области видимости (глава 4.3).

https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#Res-scope

ES.10. Объявляйте имена по одному в каждом объявлении (глава 1.5).

https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#Res-name-one

ES.22. Не объявляйте переменные, пока не получите значения для их инициализации (глава 5.4).

https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#Res-init

ES.50. Не приводите переменные с квалификатором const к неконстантному типу (глава 3.4).

https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#Res-casts-const

Per.7. При проектировании учитывайте возможность последующей оптимизации (глава 5.5).

https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#Rper-efficiency

CP.3. Сведите к минимуму явное совместное использование записываемых данных (глава 2.5).

https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#Rconc-data

E.6. Используйте идиому RAII для предотвращения утечек памяти (глава 5.6).

https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#Re-raii

E.28. При обработке ошибок избегайте глобальных состояний (например, errno) (глава 3.5).

https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#Re-no-throw

Con.5. Используйте constexpr для определения значений, которые можно вычислить на этапе компиляции (глава 4.4).

https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#Rconst-constexpr

T.1. Используйте шаблоны для повышения уровня абстрактности кода (глава 4.5).

https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#Rt-raise

T.10. Задавайте концепции для всех аргументов шаблона (глава 4.6).

https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#Rt-concepts

T.120. Используйте метапрограммирование шаблонов, только когда это действительно необходимо (глава 2.6).

https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#Rt-metameta

SF.7. Не используйте using namespace в глобальной области видимости в заголовочном файле (глава 3.6).

https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#Rs-using-directive

NR.2. Функции не обязательно должны иметь только один оператор возврата (глава 1.6).

https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines#Rnr-single-return

Предисловие

Я получил истинное удовольствие, прочитав книгу «Красивый C++». Особенно мне понравилось, что она представляет основные рекомендации C++ совсем не так, как *C++ Core Guidelines*[1]. В Руководстве правила описываются довольно кратко, густо сдобрены техническими терминами и предполагают широкое использование средств статического анализа. Эта же книга рассказывает истории, в большинстве своем взятые из игровой индустрии и основанные на эволюции кода и методов на протяжении десятилетий. В ней правила представлены с точки зрения разработчика с акцентом на преимуществах, которые можно получить, следуя этим правилам, и на неприятностях, которые могут возникнуть в результате их игнорирования. Каждое правило сопровождается более обширным обсуждением, чем может предложить C++ Core Guidelines.

Руководство стремится максимально полно охватить все правила. Естественно, подбор правил написания хорошего кода в этой книге по умолчанию не может быть полным руководством по языку. Но если ограничиться степенью полноты, необходимой для понимания обсуждаемой проблемы, то станет очевидно: Руководство C++ Core не предназначено для систематического чтения. Я рекомендую прочитать в нем разделы In: Introduction и P: Philosophy, чтобы получить представление о цели этого руководства и его концептуальной основе. А для выборочного знакомства с основными правилами создания хорошего кода, исходя из вкуса, своего видения и опыта, я советую прочитать книгу, которую вы держите в руках. Для истинных профессионалов это будет легким и увлекательным чтением. А большинство остальных разработчиков смогут узнать из нее что-то новое и полезное.

Бьерн Страуструп (Bjarne Stroustrup),
июнь 2021 года

---

[1] Руководство на английском языке свободно доступно по адресу https://github.com/isocpp/CppCoreGuidelines. — *Примеч. пер.*

Вступление

Сложность разработки программ на C++ уменьшается с выходом каждого нового стандарта и каждой новой книги. Конференций, блогов и книг более чем достаточно, и это хорошо. Но в мире не хватает инженеров с достаточно высоким уровнем подготовки для решения вполне реальных задач.

Несмотря на постоянное упрощение языка, еще многое предстоит узнать о том, как писать хороший код на C++. Бьерн Страуструп, изобретатель языка C++, и Герб Саттер (Herb Sutter), руководитель органа по стандартизации C++, посвятили много сил и времени созданию учебных материалов как для изучения C++, так и для повышения квалификации разработчиков на C++. Среди них можно назвать *The C++ Programming Language*[1] и *A Tour of C++*[2], а также *Exceptional C++*[3] и *C++ Coding Standards*[4].

Проблема книг даже такого скромного объема заключается в том, что они отражают состояние дел на момент их издания, тогда как C++ — это постоянно развивающийся язык. То, что было хорошим советом в 1998 году, может потерять актуальность. Развивающемуся языку нужен развивающийся путеводитель.

В руководстве содержатся простые и практичные советы по улучшению стиля кода на C++, чтобы вы могли писать правильный, производительный и эффективный код с первой попытки.

---

[1] *Stroustrup B.* The C++ Programming Language, Fourth Edition. — Boston: Addison-Wesley, 2013 (*Страуструп Б.* Язык программирования C++. *4*-е изд.).

[2] *Stroustrup B.* A Tour of C++, Second Edition. — Boston: Addison-Wesley, 2018 (*Страуструп Б.* Язык программирования C++. Краткий курс. 2-е изд.).

[3] *Sutter H.* Exceptional C++. — Reading, MA: Addison-Wesley, 1999 (*Саттер Г.* Новые сложные задачи на C++).

[4] *Sutter H., Alexandrescu A.* C++ Coding Standards. — Boston: Addison-Wesley, 2004 (*Саттер Г., Александреску А.* Стандарты программирования на C++).

На конференции CppCon в 2015 году Бьерном Страуструпом и Гербом Саттером в ходе их двух[1] основных докладов[2] был запущен онлайн-ресурс C++ Core Guidelines[3]. В нем содержатся простые и практичные советы по улучшению стиля кода на C++, чтобы вы могли писать правильный, производительный и эффективный код с первой попытки. Это постоянно развивающийся документ, в котором нуждаются специалисты, пишущие на C++, и авторы будут рады, если вы пришлете им свои предложения с исправлениями и улучшениями. Желательно, чтобы все, от новичков до ветеранов, следовали советам из этого Руководства.

В конце февраля 2020 года на #include discord[4] Кейт Грегори (Kate Gregory) заявила, что хотела бы издать книгу о Core Guidelines, и я, Гай Дэвидсон, сразу ухватился за эту идею. Кейт выступила на CppCon 2017[5], где рассмотрела только десять основных правил. Я разделяю ее энтузиазм по продвижению передовых приемов программирования.

Я возглавляю отдел инженерно-технических методов в *Creative Assembly*, старейшей и крупнейшей британской студии, занимающейся разработкой игр. В нем я проработал большую часть из последних 20 с лишним лет, помогая превращать наших замечательных инженеров в великих специалистов. По нашему наблюдению, несмотря на доступность и простоту Core Guidelines, разработчики мало знакомы с этим Руководством. Мы в меру своих сил и возможностей пропагандируем его и поэтому решили написать книгу, потому что литературы о рекомендациях и правилах, перечисленных в Руководстве, пока недостаточно.

Собственно Core Guidelines можно найти по адресу https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines. Оно наполнено замечательными советами, из-за чего порой трудно понять, с чего начать. Можно, конечно, читать

---

[1] Youtube.com. 2021. CppCon 2015: *Stroustrup B.* Writing Good C++14. Доступно по адресу https://www.youtube.com/watch?v=1OEu9C51K2A.

[2] Youtube.com. 2021. CppCon 2015: *Sutter H.* Writing Good C++14... By Default. Доступно по адресу https://www.youtube.com/watch?v=hEx5DNLWGgA.

[3] Isocpp.github.io. 2021. C++ Core Guidelines. Copyright © Standard C++ Foundation and its contributors. Доступно по адресу https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines.

[4] #include <C++>. 2021. #include <C++>. Доступно по адресу https://www.includecpp.org/.

[5] Youtube.com. 2021. CppCon 2017: *Gregory K.* 10 Core Guidelines You Need to Start Using Now. Доступно по адресу https://www.youtube.com/watch?v=XkDEzfpdcSg.

все подряд, сверху вниз, но понять и усвоить весь набор рекомендаций без повторного чтения практически невозможно. Рекомендации организованы в 22 основных раздела с такими названиями, как Interfaces («Интерфейсы»), Functions («Функции»), Concurrency («Конкуренция») и т. д. Каждый раздел включает отдельный набор правил и рекомендаций, иногда исчисляемых единицами, иногда десятками. Рекомендации идентифицируются по первой букве из названия раздела и номеру рекомендации в разделе. Например, «F.3. Функции должны быть короткими и простыми» — третья рекомендация в разделе F, Functions.

Все рекомендации построены по одному формату. Они начинаются с названия, которое представлено как посыл к действию (делайте это, не делайте этого, избегайте этого, предпочитайте это), затем следует причина и несколько примеров, а также исключения из правила (если они есть). В конце дается примечание, описывающее, как обеспечить соблюдение данного правила. Примечания по применению включают и советы авторам инструментов статического анализа, и советы разработчикам, как проводить проверку кода. Чтобы читать эти рекомендации, нужен определенный навык; решение о том, какие из них взять на вооружение, является вопросом личных предпочтений. Позвольте нам продемонстрировать, как начать пользоваться мудростью этого Руководства.

В C++ есть свои острые углы, есть и свои пыльные тайники, которые не так часто посещаются в современном C++. Мы хотим увести вас от них и показать, что C++ совсем не трудный и не сложный язык и его использование вполне можно доверить большинству разработчиков.

О КНИГЕ

В этой книге мы предлагаем 30 избранных рекомендаций по программированию на C++, которые сами считаем лучшими. Мы подробно объясняем эти рекомендации в надежде, что вы будете придерживаться хотя бы их, если решите не исследовать остальные советы в Core Guidelines. Рекомендации, которые мы отобрали, не обязательно являются самыми важными, но они, вне всяких сомнений, изменят ваш код к лучшему. Конечно, полезно ознакомиться также со множеством других рекомендаций и следовать им. Мы надеемся, что вы прочтете остальные советы и попробуете их

применить к своему коду. Руководство предназначено для всех разработчиков на C++, с любым уровнем опыта, и эта книга адресована тому же кругу людей. Материал не усложняется по ходу книги, и главы можно читать в любом порядке. Они не зависят друг от друга, хотя кое-где присутствуют ссылки на другие главы. Объем каждой главы примерно 3000 слов, так что вы можете считать эту книгу скорее томиком для чтения на диване по вечерам, чем учебником. Цель книги не в том, чтобы научить вас программировать на C++, а чтобы познакомить с советами, как улучшить свой стиль.

Мы разделили рекомендации на пять частей по шесть глав в соответствии с первоначальной презентацией Кейт Грегори на CppCon в 2017 году.

В части I «Bikeshedding — это плохо» мы представляем рекомендации, которые помогут выбрать верное решение из нескольких вариантов и двигаться дальше с минимумом суеты и споров. Понятие bikeshedding[1] (bike — «велосипед», shed — «навес», то есть bikeshedding — «ставить велосипед под навес». — *Примеч. ред.*) заимствовано из «закона тривиальности» К. Норткота Паркинсона (C. Northcote Parkinson), согласно которому члены организации нередко придают несоразмерное значение тривиальным вопросам, таким как выбор цвета навеса для велосипедов, вместо критериев испытаний атомной электростанции, к которой он, навес, прилагается. Какова причина тривиального подхода? Да просто велосипед — это единственная вещь, о которой все хоть что-то знают.

В части II «Не навредите себе» мы предоставляем рекомендации по предотвращению травм при написании кода. Одна из проблем, связанных с остаточной сложностью C++, заключается в возможности в некоторых ситуациях выстрелить себе в ногу. Например, несмотря на допустимость заполнять список инициализации конструктора в любом порядке, никогда не следует этого делать.

Часть III называется «Прекратите это использовать» и касается элементов языка, сохраненных ради обратной совместимости, а также советов, которые раньше считались ценными, но утеряли свою значимость благодаря нововведениям в языке. По мере развития C++ то, что раньше казалось хорошей

[1] 2021. Доступно по адресу https://exceptionnotfound.net/bikeshedding-the-daily-software-anti-pattern/.

идеей, сейчас может оказаться не таким ценным, как предполагалось изначально. Процесс стандартизации исправляет эти моменты, но вы должны знать о них, потому что можете столкнуться с ними, сопровождая старые проекты. Язык C++ гарантирует обратную совместимость: код, написанный 50 лет назад на C, будет компилироваться и сегодня.

Цель книги не в том, чтобы научить вас программировать на C++, а чтобы познакомить с советами, как улучшить свой стиль.

Далее следует часть IV под заголовком «Используйте новую особенность правильно». Такие особенности, как концепции, `constexpr`, структурное связывание и т. д., требуют осторожности при развертывании. Опять же C++ — это развивающийся стандарт, и с каждым выпуском появляется что-то новое, требующее освоения для поддержки. Хотя эта книга не ставит своей целью научить вас новым возможностям, которые определяются стандартом C++20, эти рекомендации помогут вам понять, как воспринимать их.

Часть V, последняя, называется «Пишите хороший код по умолчанию». Здесь описываются простые рекомендации, которые, если следовать им, помогут вам писать хороший код, не сильно задумываясь о происходящем. Придерживаясь их, вы научитесь писать красивый идиоматический код на C++, и он будет понят и оценен вашими коллегами.

На протяжении всей книги, как и в любом хорошем повествовании, возникает и развивается обсуждение разных тем. Мы, авторы, испытывали особое удовольствие от возможности увидеть мотивы, стоящие за рекомендациями, и проанализировать широкое их применение. Надеемся, что такое же удовольствие испытаете и вы при чтении. Многие рекомендации, если внимательно присмотреться, просто иначе формулируют некоторые из фундаментальных истин разработки программного обеспечения. Знание этих истин значительно улучшит ваши навыки программирования.

Мы искренне надеемся, что эта книга вам понравится и принесет пользу[1].

[1] Как уже понятно из введения и вступительного слова авторов, эта книга не просто о программировании, а о программировании, поданном через призму другой книги, название которой — C++ Core Guideline. Чтобы отличать ее от прочих руководств по языку C, упоминаемых далее по тексту, она фигурирует как Руководство — с прописной буквы. Все остальные пособия и труды называются в книге просто руководствами. — *Примеч. ред.*

КОД ПРИМЕРОВ

Все примеры кода доступны на сайте Compiler Explorer. Мэтт Годболт (Matt Godbolt) любезно зарезервировал постоянные ссылки для каждой главы, которые формируются путем присоединения номера главы к базовому адресу *https://godbolt.org/z/cg30-ch*. Например, ссылка https://godbolt.org/z/cg30-ch1.3 приведет вас к примерам кода для главы 1.3. Мы рекомендуем начать с https://godbolt.org/z/cg30-ch0.0, где приводятся инструкции по использованию веб-сайта и взаимодействию с кодом.

Гай Дэвидсон (Guy Davidson),
@hatcat01 hatcat.com.

Кейт Грегори (Kate Gregory),
@gregcons gregcons.com.

Октябрь 2021 года

Благодарности

Годы 2020-й и 2021-й были для нас довольно неспокойными, и мы хотели бы поблагодарить многих людей, так или иначе оказавших нам поддержку, когда мы работали над этой книгой.

Конечно же, мы хотим поблагодарить Бьерна Страуструпа и Герба Саттера за создание рекомендаций Core Guidelines и за то, что побудили нас написать о них. Мы также хотим поблагодарить участников CppCon за их помощь в обсуждении некоторых из этих рекомендаций.

Наши семьи оказали жизненно необходимую поддержку во время процесса написания, требующего уединения и сосредоточенности. Без поддержки со стороны самых близких нам было бы значительно тяжелее работать.

Легион друзей в дискорд-группе #include со штаб-квартирой на includecpp.org продолжает поддерживать нас в нашей повседневной практике использования C++ с июля 2017 года[1]. Мы пожертвуем вам одну десятую нашего дохода от этой книги. Низкий вам поклон.

Свою помощь нам оказали также несколько членов комитета ISO WG21 C++, поддерживающего стандарт. Мы хотели бы поблагодарить Майкла Вонга (Michael Wong) и Тони ван Эрда (Tony van Eerd) за их участие.

Все примеры кода доступны на Compiler Explorer[2] по постоянным и понятным ссылкам, благодаря любезности Мэтта Годболта, создателя этого прекрасного сервиса. Мы выражаем ему нашу благодарность и уверяем, что его усилия оказались, несомненно, очень важными для сообщества C++.

Cppreference.com[3] послужил нам отличным исследовательским инструментом при первоначальной подготовке каждой главы, поэтому мы весьма признательны создателю и владельцу сайта Нейту Колю (Nate Kohl), администраторам Повиласу Канапицкасу (Povilas Kanapickas) и Сергею

[1] https://twitter.com/hatcat01/status/885973064600760320

[2] https://godbolt.org/z/cg30-ch0.0

[3] https://ru.cppreference.com/w

Зубкову (Sergey Zubkov), а также Тиму Сонгу (Tim Song) и всем другим участникам проекта и благодарим их за неустанную поддержку этого прекрасного ресурса. Они — герои сообщества.

После написания главы 3.6 нам стало ясно, что своим вдохновением мы во многом обязаны статье Артура О'Дуайера (Arthur O'Dwyer). Большое ему спасибо за его неизменную преданную службу обществу. В его блоге также можно найти рассказы о предпринимаемых им усилиях по раскрытию некоторых самых ранних приключенческих текстовых игровых программ 1970-х и 1980-х годов[1].

Для такой книги, как эта, нужна армия редакторов, поэтому мы выражаем благодарность Бьерну Страуструпу, Роджеру Орру (Roger Orr), Клэр Макрей (Clare Macrae), Артуру О'Дуайеру, Ивану Чукичу (Ivan Čukić), Райнеру Гримму (Rainer Grimm) и Мэтту Годболту.

Неоценимую помощь нам оказала и команда Addison-Wesley, поэтому мы выражаем огромную благодарность Грегори Доенчу (Gregory Doench), Одри Дойл (Audrey Doyle), Асвини Кумару (Aswini Kumar), Менке Мехте (Menka Mehta), Джули Нахил (Julie Nahil) и Марку Таберу (Mark Taber).

[1] https://quuxplusone.github.io/blog

Об авторах

Дж. Гай Дэвидсон впервые познакомился с компьютерами благодаря Acorn Atom в 1980 году. Еще будучи подростком, он писал игры для различных домашних компьютеров: Sinclair Research ZX81 и ZX Spectrum, а также Atari ST. После получения степени по математике в Университете Сассекса он увлекся театром и играл на клавишных инструментах в соул-группе. В начале 1990-х стал заниматься разработкой приложений для презентаций, а в 1997-м перешел в игровую индустрию, начав работать в Codemasters в их лондонском офисе.

В 1999 году перешел в Creative Assembly, где сейчас возглавляет отдел инженерно-технических методов. Работает над франшизой *Total War*, курируя дискографию, а также формулируя и развивая стандарты программирования в команде инженеров. Входит в состав консультативных советов IGGI, группы BSI C++ и комитета ISO C++. Занимает пост ответственного за стандарты в комитете ACCU и входит в программный комитет конференции ACCU. Является модератором на дискорд-сервере #include <C++>. Отвечает за внутреннюю политику и нормы в нескольких организациях. Его можно увидеть на конференциях и встречах по C++, особенно на посвященных добавлению методов линейной алгебры в стандартную библиотеку.

В свободное время он оказывает наставническую поддержку по вопросам программирования на C++ через Prospela и BAME in Games; помогает школам, колледжам и университетам через UKIE, STEMNet и в качестве Video Game Ambassador; практикует и преподает тай-чи в стиле У; изучает игру на фортепиано; поет первый бас в Брайтонском фестивальном хоре; управляет местным киноклубом; является членом BAFTA с правом голоса; дважды баллотировался (безуспешно) на выборах в местный совет от имени партии зеленых Англии и Уэльса; пытается выучить испанский. Иногда его можно встретить за карточным столом играющим в бридж по пенни за очко. Вероятно, у него есть и другие увлечения: он большой непоседа.

Кейт Грегори познакомилась с программированием в Университете Ватерлоо в 1977 году и никогда не оглядывалась назад с сомнением или

сожалением. Имеет степень в области химического машиностроения, что лишний раз подтверждает, что диплом не всегда говорит о наклонностях человека. На цокольном этаже ее сельского дома в Онтарио есть небольшая комната со старыми компьютерами PET, C64, домашней системой 6502 и т. д., служащими напоминаниями о более простых временах. С 1986 года вместе с мужем руководит компанией Gregory Consulting, помогая клиентам по всему миру.

Кейт выступала с докладами на пяти континентах, любит искать заковыристые головоломки и затем делиться их решением, а также проводит много времени, добровольно участвуя в различных мероприятиях, посвященных языку C++. Самым уважаемым из них является группа #include <C++>, которая оказывает огромное влияние на эту отрасль, делает программирование на C++ более гостеприимным и дружелюбным. Их дискорд-сервер — теплое, уютное место для изучения C++ новичками и одновременно кают-компания для совместной работы над статьями для WG21, позволяющими взглянуть по-иному на язык, который мы все используем, или же... что-то среднее между ними двумя.

Ее отрывают от клавиатуры внуки, озера и кемпинги Онтарио, весла для каноэ и дым костра, а также соблазны аэропортов по всему миру. Гурманка, игрок в настольные игры, безотказная палочка-выручалочка, не способная ответить отказом на просьбу о помощи, она так же активна в реальной жизни, как и в интернете, но менее ярка и заметна. После того как в 2016 году пережила меланому IV стадии, она стала меньше беспокоиться о том, что думают другие и чего от нее ожидают, и больше о том, чего она хочет для своего будущего. Это дает свои результаты[1].

[1] Вся книга написана от лица обоих авторов, но иногда слово в повествовании предоставляется одному Гаю Дэвидсону. Отступления в тексте, в которых он щедро делится с читателями личными знаниями, опытом и житейской мудростью, оформлены как исторические экскурсы. — *Примеч. ред.*

От издательства

Ваши замечания, предложения, вопросы отправляйте по адресу comp@piter.com (издательство «Питер», компьютерная редакция).

Мы будем рады узнать ваше мнение!

На веб-сайте издательства www.piter.com вы найдете подробную информацию о наших книгах.