Моей любимой жене Элайзе, которую я очень люблю,
всегда любил и всегда буду любить:

«Когда я беру слово, оно означает то, что я хочу,
не больше и не меньше, – сказал Шалтай-Болтай презрительно».

– Льюис Кэролл (Lewis Carroll),
«Алиса в Зазеркалье» («Through the Looking-Glass»)

JSL
(Джон С. Лакос)

Моим тетям и моему отцу, которые всегда
поддерживали меня в каждом аспекте моей жизни.

VR
(Витторио Ромео)

Елене и моим родителям.

RK
(Ростислав Хлебников)

Покойным Дэвиду и Мэри Мередит, любящим родителям,
которые поддерживали меня во всем, что я делал,
и были бы очень горды, увидев наконец напечатанную книгу своего сына.

AM
(Алисдар Мередит)

Оглавление

Предисловие Шона Эдвардса

Я профессионально пишу программы на C++ уже более 25 лет, даже еще до стандартизации этого языка. Язык C++, по своей сущности предназначенный обеспечивать нулевые издержки и максимальную производительность, неизбежно создает некоторые ограничения – настолько далеко продвинулись синтаксис и безопасность типов. Использовать возможности C++ неразумным образом и получать ошеломляющие критические ошибки всегда было легко. Но поскольку язык был относительно стабильным, хорошие разработчики со временем научились писать надежное программное обеспечение на C++.

Первая стандартизированная версия C++98 формализовала то, что многие уже знали об этом языке. Вторая версия стандарта C++03 включала небольшие исправления и улучшения, но принципиально не меняла способ написания программ. Однако знание того, как программировать на C++, существенно изменилось после публикации стандарта C++11. Впервые за много лет Комитет по стандартам ISO C++ (WG21) добавил совершенно новую функциональность, а также кое-что удалил. Например, были введены `noexcept` и `std::unique_ptr`, а дни использования спецификаций динамических исключений и `std::auto_ptr` были сочтены.

В то же время Комитет по стандартам объявил о своем беспрецедентном намерении выпускать новую версию стандарта C++ каждые три года. Для крупной компании, занимающейся разработкой программного обеспечения, такой как Bloomberg, жизненный цикл программных активов которой измеряется десятилетиями, опора на стандарт языка, который обновляется с такой частотой, особенно проблематична. Компания Bloomberg надежно и точно предоставляет необходимую информацию профессиональному финансовому сообществу уже почти 40 лет, предлагая услуги, которые охватывают такие разнообразные потребности, как финансовая аналитика, маркетинговые решения и рыночные данные в режиме реального времени.

Для поддержки нашего глобального бизнеса компания Bloomberg разработала высокопроизводительные программные системы, работающие с поддержкой масштабирования, и уже более двух десятилетий пишет их в основном на C++. Как вы понимаете, внедрение и проверка новых цепочек инструментальных средств, лежащих в основе всей кодовой базы нашей компании, – непростая задача. Каждое обновление ставит под угрозу стабильность самих программных продуктов, от которых зависят наши клиенты.

Современный C++ предлагает многое – как хорошее, так и плохое. Многие из его новых функций открывают перспективу повышения производительности, выразительности, удобства сопровождения и т. д. С другой стороны, многие из этих функциональных возможностей таят в себе потенциальные ловушки – некоторые очевидны, другие менее очевидны. С каждой новой версией C++ – теперь уже раз в три года – язык постоянно расширяется, соответственно, расширяются и возможности для некорректного использования его функциональности из-за недостатка знаний и опыта.

Использование новых функций и без того сложного языка программирования, такого как C++, с которым многие разработчики могут быть не вполне знакомы, создает особую категорию риска. Менее опытные инженеры могут непреднамеренно ввести новые функции в зрелую кодовую базу, что в этом контексте, возможно, добавит заведомо отрицательную ценность. Как всегда, только время и опыт могут подтвердить, при каких условиях использование новой функции языка C++ будет вполне разумным. Мы как старшие разработчики, руководители групп и технические менеджеры ведущей финансово-технологической компании несем ответственность за защиту нашего основного актива в виде программного обеспечения от неоправданного риска.

Мы не можем оправдывать нестабильность при переписывании всего нашего программного обеспечения каждый раз, когда появляется новая версия языка, но мы не можем оставить наше ПО в навечно законсервированном состоянии и отказаться от важных преимуществ, которые предлагает современный C++. Поэтому мы движемся вперед, но со знанием дела и осторожностью, внедряя новые функциональные возможности только после того, как полностью поймем их смысл. Компания Bloomberg стре-

мится извлечь все возможные преимущества из современного C++, но как промышленное предприятие мы должны делать это безопасно.

Компания Bloomberg спонсировала создание этой книги «Современный C++: безопасное использование», потому что мы чувствовали, что, несмотря на все книги, конференции, блоги и т. д., описывающие функциональные возможности C++11/14, необходимо взглянуть на каждую функцию с точки зрения ее безопасного и эффективного применения в контексте большой, зрелой базы промышленного программного кода. Поэтому в книге представлены подробные объяснения каждой функциональной возможности языка C++11/14, примеры эффективного использования и подводные камни, которых следует избегать. Кроме того, эта книга могла быть написана только сейчас, после долгих лет накопления реального практического опыта. Более того, мы знали, что для написания такой книги у нас есть нужные люди – одни из лучших инженеров и авторов в мире.

Как и было обещано, Комитет по стандартам C++ придерживался своего графика, иногда с учетом крупных мировых событий, и были опубликованы две дополнительные версии стандарта, C++17 и C++20. По мере накопления сообществом опыта практического использования новых функциональных возможностей, представленных в этих стандартах, я полагаю, что будущие издания этой книги будут содержать соответствующие рекомендации и критические замечания.

Если вы писали программы на этом языке более десяти лет, то, несомненно, заметили, что стать опытным программистом на C++ стало сложнее, чем раньше. Эта книга поможет вам ориентироваться в современном мире C++, чтобы вы тоже почувствовали себя уверенно, применяя C++11/14 способами, которые действительно повышают ценность без неоправданного риска для дорогостоящих инвестиций вашей организации в программное обеспечение.

– *Шон Эдвардс* (Shawn Edwards),
технический директор (CTO) компании Bloomberg LP,
август 2021 г.

Предисловие Андрея Александреску

Вам нравятся системы управления версиями – Git, Perforce, Mercurial и им подобные? Мне очень нравятся. Я понятия не имею, как любая из современных сложных программных систем могла быть создана без использования системы управления версиями.

Одним из самых полезных свойств программного обеспечения для управления версиями является представление различий (diff view), наиболее существенное параллельное представление изменений в большой системе по сравнению с предыдущей, хорошо знакомой версией той же системы. Такое представление различий часто является наилучшим способом обзора кода, оценки сложности функциональных возможностей, поиска ошибок и, что наиболее важно, знакомства с новой системой. Почти каждый рабочий день я внимательно изучаю представления различий, исследуя одно или несколько достигнутых преимуществ. Такое представление различий является доказательством того, что мы действительно можем иметь в своем распоряжении часто упоминаемые в наших желаниях полезные средства.

В этой книге представлена новая концепция – сравнение между классическим C++, т. е. C++03, базовой версией языка, и современным C++, т. е. C++ после 2011 года, с его дополнительными функциональными возможностями. Различное представление функциональных возможностей языка программирования – потрясающая идея с интересными последствиями.

Книга «Современный C++: безопасное использование» предназначена для крупной категории программистов: тех, кто ежедневно работает со сложными системами C++ с длительным жизненным циклом, и кто знаком с C++03, поскольку указанные системы были написаны с использованием этой технологии. Классы. Наследование. Полиморфизм. Шаблоны. Стандартная библиотека STL. Да, они хорошо знают эти понятия и работают с ними каждый день в сложных предметных областях. Переопределять эти классические функциональные возможности нет необходимости. Но некоторые программисты, возможно, менее довольны изобилием новых функциональных возможностей, стандартизируемых каждые три года, начиная с C++11. У них нет времени на отслеживание деятельности Комитета по стандартам C++. Каждый час, потраченный на изучение новых функциональных возможностей C++, – это время, отнимаемое у основной функциональности системы, поэтому новая «крутая» функция должна оправдывать эти потери. Книга «Современный C++: безопасное использование» разумно оптимизирована для сохранения наилучшего соотношения полезности в производственной среде и времени, затрачиваемого на обучение.

С педагогической точки зрения эта книга решает почти невыполнимую задачу: описание «частной разности» (уж позвольте мне, как истинному ботанику, использовать математически обоснованную метафору) для каждой отдельной новой функциональной возможности, добавленной в C++ после 2003 года. Что я под этим подразумеваю? Когда книга учит особенностям языка, неизбежны споры: при обсуждении какой-либо одной отдельно взятой особенности обязательно привлекается большинство других характеристик. Как однажды сказал мне Скотт Мейерс (Scott Meyers): «Когда вы изучаете язык, то одновременно постигаете сущность всех его функциональных возможностей». Авторы тщательно разделили процессы обучения каждой новой функциональной возможности, поэтому если вы хотите узнать, скажем, об общих лямбда-выражениях, вы можете прочитать о них с минимальным воздействием со стороны любого другого нового функционального свойства языка. При необходимости взаимодействие между обсуждаемой функциональной возможностью и другими подробно оговаривается, документируется, и дается соответствующая перекрестная ссылка. В результате получается детально согласованная сама по себе книга, которую можно читать от корки до корки или разделить на части по глобальным темам, взаимосвязанным функциональным возможностям или по отдельным частным вопросам.

Главы 1, 2 и 3 напоминают своего рода противоположно направленную «Божественную комедию», в которой, как вы помните, поэта Данте надежные проводники ведут через ад (Inferno), чистилище (Purgatorio) и рай (Paradiso). Соответствующие главы по-

могут вам перейти от безопасных (Safe) к условно безопасным (Conditionally Safe) и небезопасным (Unsafe) функциональным возможностям современного C++.

Нет никаких сомнений в том, что безопасные функциональные возможности (см. главу 1) существенно улучшат ваш код, где бы вы их ни использовали. Изучение и применение рекомендаций, изложенных в главе 1, – это самый быстрый способ для группы разработчиков начать практическое использование современного C++ в производственной среде. `override`? Пользуйтесь на здоровье. Разделители цифр? Налетайте. Явные операторы преобразования? Потрясающе. Такие функциональные возможности рекомендуются в полной мере и безоговорочно. В главе 2 обсуждаются условно безопасные функциональные возможности, которые вам подходят, но с некоторыми оговорками. Списки инициализаторов? Давайте обсудим. Диапазон цикла `for`? Пара вещей, о которых следует помнить. Ссылки на rvalue? Обсуждение будет долгим – запасайтесь кофе. И последнее по списку, но не по важности: небезопасные функциональные возможности, которые могут оказаться весьма сложными и требуют практических навыков и предельного внимания при использовании. Их применение должно быть максимально ограничено и завернуто в интерфейсы. Типы стандартной компоновки? Гораздо сложнее, чем может показаться на первый взгляд. Спецификатор `noexcept`? Применять с осторожностью, на собственный страх и риск. Встроенные пространства имен? Лучше отказаться. Подробная информация, примеры и обсуждения доступны для каждой отдельной функциональной возможности, добавленной после стандарта C++03.

Здесь авторы не без иронии используют слово «небезопасно». Ничто из того, о чем рассказывается в этой книге, не является небезопасным в традиционном понимании информатики. Вместо этого подумайте о несколько другом значении слова «безопасность», когда оно используется, скажем, в хозяйственном магазине. Какова безопасность различных инструментов для тех, кто только начинает ими пользоваться? Отвертка безопасна, электродрель условно безопасна, а сварочный аппарат небезопасен.

Возможно, вы задумаетесь над этими словами: «Звучит внушительно. Но что лежит в основе такой классификации?» На самом деле книга «Современный C++: безопасное использование» вовсе не является непререкаемым сводом догм и правил, она объективна и основана на обширном опыте сообщества, который собрали и обработали авторы. Они намеренно, иногда с болью в душе, скрывают свои мнения и точки зрения. Разделы «Варианты использования» и «Потенциальные ловушки», взятые из промышленного кода, представляют собой не только поучительные примеры, но и практические доказательства.

Только с течением времени можно выделить практический опыт сообщества программистов в отношении каждой функциональной возможности и ее применимости, поэтому в книге обсуждаются функциональные возможности, добавленные в C++14, несмотря на то что стандарт C++20 уже вышел. Использование функциональных возможностей в течение многих лет может заменить горячие споры об идеях дизайна языка беспристрастным, трудно заработанным опытом, который определяет строго объективный подход, примененный в этой книге. По словам Джона Лакоса (John Lakos), «мы описываем степень ожога, который вы можете получить, если положите руку на горячую плиту, но не запрещаем вам этого делать». В результате получается совершенно не оторванный от реальной жизни материал для чтения, не более предвзятый, чем книга по экспериментальной физике. Последовательное исключение внесения собственного эго и мнения в анализ требует, как ни странно, огромного объема работы. Латинская пословица ars est celare artem (искусство – в умении скрыть искусство) звучит типично (для латыни) кратко, загадочно и немного запутанно. (Является ли латынь своего рода языком APL по сравнению с современными естественными языками?) Это выражение дословно переводится как «искусство должно скрывать искусство», но глубокий смысл ближе к фразе «истинное искусство не является подчеркнуто вычурным». Настоящие художники никогда не оставляют отпечатков пальцев на своих работах. В самом конкретном смысле это и было целью написания книги «Современный C++: безопасное использование», поскольку вы не найдете в ней никаких особых мнений, проповедей или даже неоправданно витиеватых словесных конструкций. (Ожесточенные споры возникали по поводу идеального, максимально спартанского выбора слов в том или ином абзаце.) Я уверен, что такой отточенный язык книги оценит любой читатель.

Особая дополнительная привлекательность

«Единственный вид письма – это переписывание», – гласит известная цитата. Это вдвойне верно для технических книг. Качество любого (технического) учебника заключается в готовности его авторов переделывать свою работу, а также в глубине и широте группы его рецензентов, поддерживающей процесс пересмотра. Но переписать книгу не просто! Вы когда-нибудь писали код, а затем отказывались от его пересмотра, потому что он вам чрезвычайно нравился? Умножьте это чувство на 1024, и вы узнаете, как авторы книг относятся к переписыванию тех фрагментов, в которые они уже вложили свою душу. Вы действительно должны быть приверженцем повышения качества, чтобы сохранить спокойствие во время такого испытания.

То, что авторы настаивают на качестве, напоминает о том, что мне больше всего нравится в этой книге и что в то же время труднее всего объяснить. Я называю это дополнительной привлекательностью.

Я заметил кое-что в действительно великих трудах – будь то инженерное дело, искусство, спорт или любая другая сложная человеческая деятельность. Почти всегда отличная работа – это результат усилий талантливых людей, выходящих за рамки того, что можно было бы считать разумным. Высоко оценивая такую работу, мы неявно признаем огромные возможности в сочетании с соответствующими невероятными усилиями по их реализации. Хорошую работу можно сделать быстро, но великий труд – невозможно.

По странному стечению обстоятельств я в конце концов увлекся этой книгой – во-первых, из-за одной рецензии. Потом читал снова и снова – всего четыре полных прохода по всей книге. Стремление к совершенству так же заразительно, как склонность к неряшливости, но несравненно интереснее. («Уничтожить!» Джон Лакос (John Lakos) терпеливо присылал мне по электронной почте каждую новую редакцию. Мои рецензии, часто весьма едкие, мотивировали его как ничто другое.) Другие рецензенты – представители Комитета по стандартам C++, отраслевые эксперты по C++, эксперты по C++03, ранее незнакомые с версиями стандарта C++1x, эксперты по программной архитектуре, эксперты по многопоточности, эксперты по процессам, даже эксперты по LaTeX – сделали то же самое, в результате чего каждое предложение, которое вы прочтете, подвергалось критическому анализу десятки раз и, вероятно, столько же раз переписывалось. Со своей стороны я настолько увлекся проектом и бескомпромиссным подходом авторов к качеству, что в итоге написал для книги полноценный материал. (Любая ошибка в разделе 2.1 «Шаблоны функций с переменным числом аргументов» – моя вина.) Создание этой книги потребовало огромного объема работы, большего, чем я мог ожидать, – это все, на что я мог надеяться. Я считал, что стал слишком стар, чтобы продолжать работать всю ночь, но, очевидно, ошибся.

При таком активном участии я могу утверждать: в этой книге действительно заложена дополнительная привлекательность. Обсуждение идет, как и должно, никакой лишней чепухи, примеры кода точны и красноречивы. Я считаю, что «Современный C++: безопасное использование» – великолепная работа. Надеюсь, что помимо извлечения практических уроков из этой книги вы черпаете из нее вдохновение, для того чтобы добавить больше привлекательности в собственную работу. Я добавил – это я точно знаю.

– *Андрей Александреску* (Andrei Alexandrescu),
май 2021 г.

Благодарности

Книга «Современный C++: безопасное использование» – это работа всего сообщества программистов на C++ в целом, а не только нескольких авторов. Эта книга содержит знания, полученные из самых глубин дизайна языка до границ разработки надежного программного обеспечения. Эксперты, находящиеся на одной границе этого диапазона, простирающегося от проектирования языка до разработки приложений, возможно, не слишком хорошо знакомы с особенностями другой границы. Хотя мы, четыре автора, названных в начале этой книги, являемся профессиональными старшими инженерами-программистами, наши совокупные знания изначально охватывали не все, что здесь представлено, но мы полагались на многих наших коллег – от наших сотрудников-разработчиков в компании Bloomberg до ведущей рабочей группы Комитета по стандартам C++, а также на самого Бьярна Страуструпа (Bjarne Stroustrup), – чтобы заполнить пробелы в наших знаниях и исправить имеющиеся у нас заблуждения.

Каждый из команды Bloomberg BDE, созданной в декабре 2001 г., так или иначе внес непосредственный вклад в публикацию этой книги: Парса Амини (Parsa Amini), Джошуа Берн (Joshua Berne), Харри Ботт (Harry Bott), Стивен Брайтштайн (Steven Breitstein), Натан Берджерс (Nathan Burgers), Билл Чепмен (Bill Chapman), Атилла Фехер (Attila Feher), Мунго Джилл (Mungo Gill), Ростислав Хлебников (Rostislav Khlebnikov), Джеффри Мендельсон (Jeffrey Mendelsohn), Алисдар Мередит (Alisdair Meredith), Хайман Розен (Hyman Rosen), а также второй менеджер команды BDE (с апреля 2019 г.) Майк Вершелл (Mike Verschell).

Секретарь Комитета по стандартам ISO C++ и директор ISO C++ Foundation Нина Раннс (Nina Ranns) была нашим главным исследователем и предоставила возможность доступа к глубинам стандарта ядра языка C++. Мы полагались на нее, чтобы добраться до истины: версии стандарта выходят каждые три года, а отчеты о дефектах имеют обратную силу по отношению к предыдущим стандартам, поэтому истина – это эфемерное и неуловимое существо, зависящее от контекста. Тем не менее Нина обеспечила нам полное понимание того, что существовало в действительности в то время, и тщательно рассмотрела каждую функциональную возможность ядра языка, описываемую в этой книге: см. разделы 2.5, 2.11, 2.18 и 3.5, а также несколько примеров.

Джошуа Берн (Joshua Berne), старший инженер-программист в группе BDE Bloomberg и активный член основной рабочей группы (CWG) и исследовательской группы по контрактам (SG21) Комитета по стандартам C++, выполнял несколько ролей: Джош был нашим связующим звеном между ядром языка и разработкой программного обеспечения, выполняя структурное переформулирование основных функциональных возможностей, включая упомянутые функциональные возможности и многие другие. Все сравнительные исследования, проведенные для этой книги, были разработаны, выполнены и/или проанализированы Джошем. Он предоставил технические знания, необходимые для полноценной работы LaTeX, разработал и реализовал словарь терминов (глоссарий), включающий автоматизированные обратные ссылки в конкретные разделы, в которых используются эти термины. Важно отметить, что Джош был голосом разума на протяжении всего проекта.

Лори Хьюз (Lori Hughes), наш руководитель проекта, главный технический редактор, а также дизайнер, наборщица и верстальщица в LaTeX, вероятно, сказала бы вам, что уход за кошками – детская игра по сравнению с тем, что ей пришлось пережить во время этого проекта. Упорство, напористость и трудолюбие, которые она неустанно демонстрировала, возможно, являются единственной причиной, по которой эта книга была опубликована в 2021 г. (а не в течение этого десятилетия). Короче говоря, Лори – наша прочная основа, она ветеран lakos20, и мы с нетерпением ждем возможности поработать с ней над всеми нашими запланированными будущими проектами, например над аллокаторами (lakos22), контрактами (lakos23), томами II и III, следующими за lakos20 (lakos2a и lakos2b), и ожидаемыми будущими изданиями этой книги, учитывающими стандарты C++17, C++20 и т. д.

Пабло Халперн (Pablo Halpern), бывший член группы Bloomberg BDE, активный член комитета по стандартам C++, создатель модели аллокатора `std::pmr`, а теперь постоянный сотрудник BDE, работающий над поддержкой на уровне языка для распределителей локальной памяти, стал «автором-невидимкой» описаний нескольких функциональных возможностей в этой книге (например, см. раздел 2.20) и обеспечил крупномасштабную реструктуризацию многих других (например, см. раздел 2.11). Примечательно, что из всех тех, кто не являлся авторами, предоставившими черновики в окончательной форме, только Пабло смог написать в стиле, близком к авторскому. Он также выполнил исследование для статьи, заказанной авторами этой книги, продемонстрировав, что операции перемещения `move` изначально выполняются быстрее, но во время выполнения могут иметь негативные общие последствия из-за фрагментации памяти; см. halpern21c.

Доктор Андрей Александреску (Andrei Alexandrescu), автор выдающейся книги «Modern C++ Design» (Addison-Wesley, 2001), соавтор книги «C++ Coding Standards» (Addison-Wesley, 2005) и главный участник разработки языка D, неоднократно призывался для оказания помощи в этом дерзком проекте: (1) как автор-эксперт для предоставления доступного руководства по использованию шаблонов с переменным числом аргументов для тех, кто привык к их аналогам в C++03 (см. раздел 2.21), (2) как технический рецензент, основной задачей которого было устранить неудобства стиля изложения Джона Лакоса с его многочисленными вводными фразами и сносками, и (3) как талисман и энтузиаст, поддерживающий наши усилия по внедрению в широкие массы пользователей C++03 дополняющих функциональных возможностей из стандарта C++11/14. Андрей также великодушно согласился написать предисловие к этой книге, подтверждая ее полезность для ведущих разработчиков, знакомых с классическим C++.

Гарольд Ботт (Harold Bott), технический ассистент Джона на его курсе Advanced C++ в 1990-х гг. в Колумбийском университете, снова вышел на связь с Джоном в 2019 г. С тех пор Гарри стал движущей силой в стремлении к завершению этой книги. После месяца исследований совместно с Ниной Раннс Джон поручил Гарри, бывшему программисту Goldman Sachs и исполнительному директору JP Morgan, подготовить для ознакомления важнейшую функциональную особенность современного C++ (см. раздел 2.18) – это действительно весьма сложная задача. После того как обзоры были готовы и потребовались исправления, Гарри работал с Джоном практически круглосуточно в течение почти трех недель подряд, чтобы учесть отзывы рецензентов и довести описание этой важной функциональной возможности до состояния, в котором оно представлено здесь.

Мунго Джилл (Mungo Gill) является одним из самых новых штатных сотрудников группы BDE и обладает более чем 30-летним опытом работы в таких известных организациях, как Salomon Brothers, Citigroup, Lehman Brothers, Google и Citadel Securities. Мунго просмотрел каждую строку этой книги и предоставил ценные отзывы с точки зрения старшего специалиста-практика. Он также координировал процесс составления определений словаря терминов и обеспечивал согласование мнений множества экспертов в различных областях.

Клэй Уилсон (Clay Wilson), член команды BDE с 2003 г., является еще одним ветераном lakos20. Клэй в течение последних 18 лет был нашим «финишером», когда дело доходит до завершающей проверки компонентов программного обеспечения. Его внимание к деталям и точность, как мы убедились на собственном опыте, не имеют себе равных. Клэй просмотрел большую часть этой книги, и мы с нетерпением ждем возможности продолжения совместной работы над будущими проектами.

Стивен Брайтштайн (Steven Breitstein), член BDE с 2004 г. и выпускник lakos20, просмотрел каждую строку и каждый фрагмент кода в этой книге и внес бесчисленное множество предложений по явному улучшению подачи материала. Он также улучшил и собственноручно применил все правки к словарю терминов для этой книги.

Хайман Розен (Hyman Rosen) ушел из группы Bloomberg BDE в апреле 2021 г. Он был экспертом по практическим вариантам использования в реальном мире некоторых формально небезопасных функций современного C++, таких как применение расширенной дружбы (см. раздел 3.3) с шаблоном curiously recurring template pattern (CRTP). Вы найдете много других подобных шаблонов, разбросанных по всей книге.

Стивен Дьюхерст (Stephen Dewhurst), признанный во всем мире эксперт по программированию на C++ и широко известный автор многих книг по C++, докладчик на конференциях и профессиональный инструктор по C++ (в том числе более десяти лет в Bloomberg), проанализировал каждую функцию в этой книге и предоставил подробнейшие, ценные с практической точки зрения отзывы, а также вариант использования – см. подраздел 2.14.2.8 в разделе 2.14.

Джеффри Олкин (Jeffrey Olkin), присоединившийся к Bloomberg в 2011 г., является одним из ведущих архитекторов программного обеспечения. Он был структурным редактором lakos20 и с самого начала был активным участником создания этой книги, просматривая многие функции, помогая систематизировать предварительный материал и постоянно поддерживая информативную и всегда полезную обратную связь.

Стив Дауни (Steve Downey), ведущий разработчик в Bloomberg с 2003 г., член Комитета по стандартам C++ и эксперт во многих областях, предоставил большую часть развернутого материала, найденного в несколько специализированной, условно безопасной функциональной возможности C++11 – см. раздел 1.17. Майк Жиру (Mike Giroux) и Олег Субботин (Oleg Subbotin) собрали и предоставили эталонный материал еще для одной условно безопасной функции C++11 – см. раздел 2.9.

Шон Парент (Sean Parent) представил подраздел, в котором оценивается строгость текущих требований к перемещаемым объектам для стандартных контейнеров, – см. подраздел 2.18.4.5 в разделе 2.18. Найл Дуглас (Niall Douglas) написал подраздел, подробно описывающий его огромный опыт работы с одной из небезопасных функциональных возможностей C++, – см. подраздел 3.4.7.1 в разделе 3.4. Нильс Деккер (Niels Dekker) рассмотрел еще одну небезопасную функциональную возможность C++11 (см. раздел 3.5) и предоставил ценную дополнительную информацию, а также ссылки на свои собственные исследования производительности. Кевин Кляйн (Kevin Klein) помог организовать и подготовить материал еще для одной небезопасной функциональной возможности C++11; см. раздел 3.2.

Многие ведущие инженеры-программисты на C++, инструкторы и профессиональные разработчики рецензировали эту работу и предоставляли подробные отзывы: Адиль Аль-Ясири (Adil Al-Yasiri), Андрей Александреску (Andrei Alexandrescu), Парса Амини (Parsa Amini), Брайан Би (Brian Bi), Франк Бирбахер (Frank Birbacher), Гарри Ботт (Harry Bott), Стив Брайтштайн (Steve Breitstein), Томас Канабрава (Tomaz Canabrava), Билл Чепмен (Bill Chapman), Маршалл Клау (Marshall Clow), Стивен Дьюхерст (Stephen Dewhurst), Акшайе Даван (Akshaye Dhawan), Найл Дуглас (Niall Douglas), Стив Дауни (Steve Downey), Том Экклз (Tom Eccles), Аттила Фехер (Attila Feher), Кевин Флеминг (Kevin Fleming), Х. Даниэль Гарсиа (J. Daniel Garcia), Мунго Джилл (Mungo Gill), Майк Жиру (Mike Giroux), Кевин Кляйн (Kevin Klein), Джефф Мендельсон (Jeff Mendelsohn), Джеффри Олкин (Jeffrey Olkin), Нина Раннс (Nina Ranns), Хайман Розен (Hyman Rosen), Даниэль Руозо (Daniel Ruoso), Бен Сакс (Ben Saks), Ричард Смит (Richard Smith), Олег Субботин (Oleg Subbotin), Джулиан Темплман (Julian Templeman), Майк Вершелл (Mike Verschell), Клэй Уилсон (Clay Wilson) и Джей Си ван Винкел (JC van Winkel).

В дополнение к анализу особенностей этой книги по мере ее написания группа BDE создала предварительную версию словаря терминов, после чего мы снова полагались на Джоша Берна (Josh Berne), Брайана Би (Brian Bi), Гарри Ботта (Harry Bott), Мунго Джилла (Mungo Gill), Пабло Халперна (Pablo Halpern) и Нину Раннс (Nina Ranns) для его уточнения и включения в окончательную версию, прежде чем рассмотреть конечный вариант книги в целом. Джефф Мендельсон (Jef Mendelsohn) и Натан Бергерс (Nathan Burgers) помогли выделить основную идею этой книги для размещения на задней обложке. Мы хотим поблагодарить всех членов Комитета по стандартам, которые предоставили ценную информацию при изучении подробностей, истории и т. д., связанных с различными функциональными возможностями языка, представленными в этой книге. Мы хотели бы отдельно поблагодарить Бьярна Страуструпа (Bjarne Stroustrup) за дружелюбные ответы на наши острые вопросы, касающиеся всего, что связано с C++. Говард Хиннант (Howard Hinnant) подтвердил, среди прочего, подробности того, почему и как первоначально были придуманы `xvalues` и как они с тех пор трансформировались (и стали известными также под именем «delta») от своей первоначальной концепции до их со-

временного определения. Майкл Вонг (Michael Wong), Пол МакКенни (Paul McKenney) и Мажед Майкл (Maged Michae) проверили и утвердили наше представление атрибута [[carries_dependency]] – см. раздел 3.1. Невозможно не поблагодарить Ричарда Смита (Richard Smith) за его детальный обзор и многочисленные предложения о том, как исправить и улучшить нашу интерпретацию основной функциональной возможности современного C++ (см. раздел 2.18). Мы надеемся, что Ричард будет рецензировать каждую функциональную возможность в последующих изданиях этой книги.

Команда издательства Pearson – Грег Донч (Greg Doench), наш редактор и бесстрашный лидер, Джули Нэхил (Julie Nahil), исполнительный менеджер, и Ким Уимпсетт (Kim Wimpsett), выпускающий редактор, – оказали мощную поддержку в стремлении написать эту книгу быстро и аккуратно, несмотря на нестандартный рабочий процесс. Изначально мы планировали, что книга будет состоять из 300–400 страниц и будет готова к концу 2020 года. Но этого не произошло. Каким-то образом Грег и Джули нашли способ правильно организовать рабочий процесс и напечатать эту книгу к зимним праздникам в конце 2021 г. – спасибо им за это!

Онлайн-компиляторы, такие как Godbolt (Compiler Explorer) и Wandbox, оказались бесценными при создании этой книги, позволив авторам быстро оценивать и передавать друг другу примеры кода, протестированные в различных версиях нескольких компиляторов, воспринимающих разные диалекты языка.

Мы хотим выразить признательность сотрудникам компании Bloomberg, привлеченным для защиты интеллектуальной собственности Bloomberg, – благодаря им интеллектуальная собственность и данные клиентов компании никоим образом не были скомпрометированы какой-либо частью этой книги, на что были сделаны соответствующие ссылки: Том Арчидиаконо (Tom Arcidiacono), Кевин П. Флеминг (Kevin P. Fleming) и Хаим Хаас (Chaim Haas).

Кроме того, мы хотим выразить признательность и поблагодарить руководство компании Bloomberg не только за поддержку, но и за требование выполнить эту важную работу для компании, а затем предоставить ее в свободном доступе. В 2012 г. Владимир Клячко (Vladimir Kliatchko), в то время и по сей день являющийся руководителем отделения инженерных разработок в Bloomberg, поручил Джону Лакосу, который сотрудничал с Ростиславом Хлебниковым, написать статью khlebnikov18, в которой кратко описаны преимущества C++11 и способы его наилучшего использования. Эта короткая статья по C++11 на 11 страницах, набранных шрифтом в 11 пт, была действительно хорошо принята, широко распространена и полностью одобрена 85 процентами членов Комитета по стандартам, которые ее рецензировали. После этого Андрей Басов (Andrei Basov), менеджер отдела инженерных разработок Middleware и Core Services, Акшай Даван (Akshaye Dhawan), технический руководитель отдела обучения, документации и управления производством, и Адам Вольф (Adam Wolf), руководитель отдела разработки программной инфраструктуры, поощряли и поддерживали нас в стремлении к более всеобъемлющему, ориентированному на практическую деятельность изложению особенностей использования современного C++, включая стандарт C++14, в форме книги.

Наконец, создание этой книги было бы невозможным без щедрого покровительства нашего директора по технологиям Шона Эдвардса (Shawn Edwards). Без его поддержки и особенно его спонсорства огромные технические ресурсы, необходимые для создания этой книги, никогда бы не были пущены в ход. Шон, за плечами которого блестящая карьера разработчика, руководителя группы и технического менеджера, а теперь и крупного руководителя, любезно предоставил предисловие к этой книге.

Об авторах

Джон Лакос (John Lakos), автор книг «Large-Scale C++ Software Design» (Addison-Wesley, 1996) и «Large-Scale C++ Volume I: Process and Architecture» (Addison-Wesley, 2020), работает в компании Bloomberg в Нью-Йорке в качестве старшего архитектора и инструктора по разработке программного обеспечения на C++ по всему миру. Он также является активным членом с правом голоса рабочей группы по развитию языка Комитета по стандартам C++. С 1997 по 2001 г. доктор Лакос руководил проектированием и разработкой инфраструктурных библиотек для собственных аналитических финансовых приложений в Bear Stearns. С 1983 по 1997 г. доктор Лакос трудился в компании Mentor Graphics, где разрабатывал крупные программные платформы и продвинутые приложения ICCAD, при создании которых он получил несколько патентов на программное обеспечение. Джон Лакос получил докторскую степень в области информатики в 1997 г. и степень доктора наук по электротехнике (1989 г.) в Колумбийском университете. Доктор Лакос получил степень бакалавра в Массачусетском технологическом институте по математике (1982 г.) и информатике (1981 г.).

Витторио Ромео (Vittorio Romeo) (бакалавр информатики, 2016 г.) – старший инженер-программист в компании Bloomberg в Лондоне, где он создает особо важное ПО среднего звена на C++ и обучает сотни сотрудников программированию на современном C++. Он начал программировать в возрасте 8 лет и сразу увлекся C++. Витторио создал несколько библиотек и игр на C++ с открытым исходным кодом, опубликовал множество видеокурсов и учебных пособий и активно участвует в процессе стандартизации ISO C++. Он является активным членом сообщества C++ с горячим желанием делиться своими знаниями и учиться у других: более 20 раз выступал на международных конференциях C++ (включая CppCon, C++Now, ++it, ACCU, C++ On Sea, C++ Russia и Meeting C++) по разнообразным темам: от разработки игр до метапрограммирования шаблонов. У Витторио есть собственный веб-сайт (**https://vittorioromeo.info/**) с продвинутыми статьями и материалами по C++ и канал на YouTube (**https://www.youtube.com/channel/UC1XihgHdkNOQd5IBHnlZWbA**), где размещены широко известные современные руководства по C++11/14. Витторио является активным участником StackOverfow, уделяя большое внимание ответам на интересные вопросы по C++ (его репутация 75k+). В свободное от написания кода время Витторио увлекается тяжелой атлетикой и фитнесом, а также компьютерными играми и научно-фантастическими фильмами.

Ростислав Хлебников (Rostislav Khlebnikov) является руководителем группы BDE Solutions, которая работает над различными библиотеками BDE, такими как библиотека для связи по протоколу HTTP/2, и участвует в других проектах, включая улучшение взаимодействия библиотек BDE с типами словарей стандартной библиотеки. Он является активным членом Комитета по стандартам C++ и участвовал в конференции CppCon 2019. До работы в Bloomberg доктор Хлебников получил степень бакалавра в области прикладной математики и информатики в Санкт-Петербургском государственном политехническом университете (Россия) и степень доктора информатики в Технологическом университете Граца (Австрия). Ростислав профессионально работал инженером-программистом на C++ более 15 лет.

Алисдар Мередит (Alisdair Meredith) был членом Комитета по стандартам C++ с момента создания C++11 на заседании комитета в Оксфорде в 2003 г., уделяя особое внимание интеграции функциональных возможностей и активному поиску и устранению несоответствий в языке. Алисдар был председателем LWG, когда были опубликованы стандарты C++11 и C++14, и он отдает должное напряженной работе предыдущего председателя, Ховарда Хиннанта (Howard Hinnant). Алисдар был постоянным докладчиком на конференциях в течение почти 15 лет, представляя новые работы комитета по стандартам C++. Алисдар присоединился к группе Bloomberg BDE в 2009 г. В течение десяти лет до этого Алисдар работал профессиональным программистом приложений на C++ в автогонках Формула-1 с командами Benetton и Renault, выиграв вместе с ними два чемпионата мира. Между чемпионатами Алисдар около года проработал менеджером по выпуску ПО в Borland, занимаясь маркетингом продуктов компании на C++. Алисдар любит путешествия, вкусные обеды и подводное плавание с дыхательной трубкой.

Глава 0

Экскурс в компьютерные системы

Приветствуем всех читателей! Книга «Современный C++: безопасное использование» – справочное руководство, предназначенное для профессионалов, разрабатывающих и сопровождающих крупномасштабные сложные программные системы, написанные на языке C++, и желающих в полной мере овладеть современными функциональными возможностями C++.

В этой книге основное внимание уделяется продуктивной ценности каждой новой функциональной возможности языка, начиная с C++11, особенно когда рассматриваемые системы и организации разбираются с точки зрения масштабирования. Мы намеренно оставили в стороне основные принципы и идиомы языка, – какими бы остроумными и интеллектуально привлекательными они ни выглядели, – которые могут повредить конечному результату применительно к крупномасштабным системам. Вместо этого мы сосредоточились на принятии разумных экономических и проектных решений с учетом неизбежных компромиссов, возникающих в любой инженерной дисциплине. При этом мы стараемся избегать субъективных мнений и рекомендаций.

Как известно, Ричард Фейнман (Richard Feynman) сказал: «Если что-либо не соответствует эксперименту, это неправильно. В этом простом утверждении – ключ к науке»[1]. Нет лучшего способа поэкспериментировать с функциональной возможностью языка, чем позволить времени сделать свою работу. Мы серьезно обдумали этот принцип и решили рассматривать только те средства современного C++, которые были частью стандарта не менее пяти лет, что, по нашему мнению, дает достаточную перспективу для правильной оценки практического влияния новых функциональных возможностей. Таким образом, мы можем опираться на практический опыт, чтобы обеспечить тщательное и всестороннее изложение материала, учитывая ограниченное время для профессионального развития большинства читателей. Если вы ищете способы повышения производительности с применением надежных и действительно современных средств C++, то мы надеемся, что эта книга станет необходимым источником информации, к которому вы будете обращаться снова и снова.

То, чего нет в книге, так же важно, как и то, что в ней есть. Книга «Современный C++: безопасное использование» («Embracing Modern C++ Safely», известная также в виде аббревиатуры EMC++S) не является учебным пособием по программированию на C++ или даже справочником по новым функциональным возможностям C++. Мы предполагаем, что вы опытный разработчик, руководитель группы или менеджер, а также что вы уже в полной мере освоили классический C++98/03 и теперь ищете четко определенные, целенаправленные способы интеграции современных функциональных возможностей C++ в свой набор инструментов.

0.1. Чем эта книга отличается от других

Цель книги, которую вы сейчас читаете, – всегда оставаться объективной, эмпирической и практической. Мы просто представляем функциональные возможности, их при-

[1] Ричард Фейнман, лекция в Корнеллском университете (Cornell University), 1964 г. Видеозапись и комментарии доступны здесь: **https://fs.blog/2009/12/mental-model-scientific-method**.