Оглавление

Содержание

Предисловие

Программирование – одна из тех немногих дисциплин, что позволяют творить нечто буквально из ничего. Программист может творить целые миры, которые ведут себя в точности как задумал автор. Для этого нужен лишь компьютер.

Когда я учился в школе, курс программирования был в основном сфокусирован на императивном программировании – сначала это было процедурное программирование на языке C, затем объектно-ориентированное на языках C++ и Java. С поступлением в университет почти ничего не изменилось – основной парадигмой по-прежнему оставалось объектно-ориентированное программирование (ООП).

Поэтому тогда я едва не попался в мыслительную ловушку, убедив себя в том, что все языки программирования, в сущности, одинаковы, различия между ними – чисто синтаксические и программисту довольно изучить такие основополагающие понятия, как ветвление и цикл, в одном языке, чтобы программировать (с небольшими поправками) на всех остальных языках.

С языками функционального программирования я впервые познакомился в университете, когда в рамках одной из дисциплин понадобилось изучить язык LISP. Моей первой реакцией было смоделировать средствами языка LISP условный оператор `if-then-else` и оператор цикла `for`, чтобы сделать язык пригодным для работы. Вместо того чтобы привести свое восприятие в соответствие с языком, я решил доработать язык напильником, чтобы он позволял мне и дальше писать программы таким же образом, каким я привык писать на языке C. Нетрудно догадаться, что в те дни я не увидел никакого смысла в функциональном программировании, ведь все, чего я мог добиться, используя LISP, можно было гораздо проще сделать на языке C.

Прошло немало времени, прежде чем я снова стал поглядывать в сторону функционального программирования. Подтолкнула меня к этому неудовлетворенность слишком медленной эволюцией одного языка, который мне необходимо было использовать для нескольких проектов. В язык наконец добавили оператор цикла `for-each`, и это подавалось как громадное достижение. Программисту оставалось лишь загрузить новый компилятор, и жизнь должна была заиграть новыми красками.

Это заставило меня задуматься. Чтобы получить в свое распоряжение новую языковую конструкцию наподобие цикла `for-each`, мне нужно было дождаться новой версии языка и новой версии компилятора. Но на языке LISP я мог самостоятельно реализовать цикл `for` в виде обыкновенной функции. Никакого обновления компилятора при этом не требовалось.

Именно это склонило меня к функциональному программированию: возможность расширять язык без необходимости менять компилятор. Я по-прежнему оставался в плену объектно-ориентированного мировоззрения, но уже научился использовать конструкции, заимствованные из функционального стиля, чтобы упростить работу по созданию объектно-ориентированного кода.

Тогда я стал посвящать много времени исследованию функциональных языков программирования, таких как Haskell, Scala и Erlang. Меня поразило, что многие проблемы, заставляющие страдать объектно-ориентированных программистов, удается легко решить, посмотрев на них под другим углом – функциональным.

Поскольку большая часть моей работы связана с языком C++, я решил найти способ, как использовать в этом языке приемы функционального программирования. Оказалось, что я в этом не одинок: в мире полно других людей с похожими идеями. Некоторых из них мне посчастливилось встретить на различных конференциях по языку C++. Всякий раз это была превосходная возможность обменяться идеями, научиться чему-то новому и поделиться своим опытом применения идиом функционального программирования на языке C++.

На большей части таких встреч мы с коллегами сходились на том, что было бы замечательно, если бы кто-то написал книгу о функциональном программировании на C++. Вот только каждый из нас хотел, чтобы эту книгу написал кто-то другой, поскольку каждый искал источник готовых идей, пригодных для собственных проектов.

Когда издательство Manning предложило мне стать автором такой книги, это меня поначалу обескуражило: я считал, что мне следовало бы прочесть об этом книгу, а не написать ее. Однако затем я подумал, что если каждый будет рассуждать подобным образом, никто так и не увидит книгу о функциональном программировании на языке C++. Я решил принять предложение и отправиться в это путешествие. То, что из этого получилось, вы сейчас и читаете.

Благодарности

Я хотел бы поблагодарить всех, чье участие сделало эту книгу возможной: профессора Сашу Малкова за то, что привил мне любовь к языку C++; Ако Самарджича за то, что научил меня, насколько важно писать легкочитаемый код; моего друга Николу Йелича, который убедил меня, что функциональное программирование – это здорово; Золтана Порколаба, поддержавшего мою догадку, что функциональное программирование и язык C++ образуют хорошую смесь; Мирьяну Малькович за помощь в обучении наших студентов тонкостям современного языка C++, включая и элементы функционального программирования.

Почет и уважение Сергею Платонову и Йенсу Веллеру за организацию превосходных конференций по языку C++ для тех из нас, кто все еще живет старыми традициями. Можно смело сказать, что без всех перечисленных людей эта книга бы не состоялась.

Я хотел бы поблагодарить своих родителей, сестру Соню и свою вторую половинку Милицу за то, что всегда поддерживали меня в смелых начинаниях наподобие этого. Кроме того, я благодарен своим давним товарищам по проекту KDE, которые помогли мне вырасти как разработчику за прошедшее десятилетие, – прежде всего Марко Мартину, Аарону Сеиго и Себастиану Кюглеру.

Огромная благодарность команде редакторов, которую для меня организовало издательство Manning: Майклу (или просто Майку) Стивенсу за самое непринужденное из всех моих собеседований; замечательным ответственным редакторам Марине Майклс и Лесли Трайтс, которые научили меня писать книгу (благодаря им я научился гораздо большему, чем мог ожидать); техническому редактору Марку Эльстону за то, что заставлял меня держаться ближе к практике; блестящему Юнвэю Ву, который не только проделал работу по вычитке книги, но и помог улучшить рукопись множеством различных способов. Надеюсь, что доставил всем им не слишком много хлопот.

Также выражаю свою признательность всем, кто предоставил свой отзыв на рукопись: Андреасу Шабусу, Биннуру Курту, Дэвиду Кернсу, Димитрису Пападопулосу, Дрору Хелперу, Фредерику Флейолю, Георгу Эрхардту, Джанлуиджи Спануоло, Глену Сиракавиту, Жану Франсуа Морену, Керти Шетти, Марко Массенцио, Нику Гидео, Никосу Атанасиу, Нитину Годе, Ольве Маудалю, Патрику Регану, Шону Липпи, а в особенности Тимоти Театро, Ади Шавиту, Суманту Тамбе, Джану Лоренцо Меоччи и Николе Гиганте.

Об этой книге

Эта книга предназначена не для того, чтобы научить читателя языку программирования C++. Она повествует о функциональном программировании и о том, как воплотить его в языке C++. Функциональное программирование предлагает непривычный взгляд на разработку программ и иной подход к программированию, по сравнению с императивным и объектно-ориентированным стилем, который обычно используется совместно с языком C++.

Многие, увидев заглавие этой книги, могут счесть его странным, поскольку C++ часто по ошибке принимают за объектно-ориентированный язык. Однако хотя язык C++ и впрямь хорошо поддерживает объектно-ориентированную парадигму, он на самом деле выходит далеко за ее границы. Так, он поддерживает еще и процедурную парадигму, а в поддержке обобщенного программирования большинству других языков с ним не сравниться. Кроме того, язык C++ весьма хорошо поддерживает большинство, если не все, из идиом функционального программирования, в чем читатель сможет вскоре убедиться. С каждой новой версией язык C++ пополнялся новыми средствами, делающими функциональное программирование на нем все более удобным.

Кому стоит читать эту книгу

Эта книга предназначена в первую очередь для профессиональных разработчиков на языке C++. Предполагается, что читатель умеет самостоятельно настраивать среду для сборки программ, устанавливать и использовать сторонние библиотеки. Кроме того, от читателя потребуется хотя бы начальное знакомство со стандартной библиотекой шаблонов, автоматическим выводом типов – параметров шаблона и примитивами параллельного программирования – например, двоичными семафорами.

Впрочем, книга не окажется полностью непонятной для читателя, не искушенного в разработке на языке C++. В конце каждой главы приведен список статей, разъясняющих языковые конструкции, которые могли бы показаться читателю малознакомыми.

Последовательность чтения

Данную книгу лучше всего читать последовательно, так как каждая последующая глава опирается на понятия, разобранные в предыдущих. Если что-либо остается непонятным после первого прочтения, лучше перечитать сложный раздел еще раз, прежде чем двигаться дальше, по-

скольку сложность материала возрастает с каждой главой. Единственное исключение здесь составляет глава 8, которую читатель может пропустить, если не интересуется методами долговременного хранения структур данных.

Книга разделена на две части. Первая часть охватывает идиомы функционального программирования и способы их воплощения в языке C++.

- Глава 1 вкратце знакомит читателя с функциональным программированием и преимуществами, которые оно может привнести в мир C++.
- Глава 2 посвящена функциям высшего порядка – функциям, которые принимают другие функции в качестве аргументов или возвращают функции в качестве значений. Данное понятие иллюстрируется некоторыми из множества полезных функций высшего порядка, включенных в стандартную библиотеку языка программирования C++.
- Глава 3 повествует обо всех разнообразных сущностях, которые в языке C++ рассматриваются как функции, – от обычных функций в смысле языка C до функциональных объектов и лямбда-функций.
- Глава 4 содержит изложение различных способов создания новых функций из имеющихся. В этой главе рассказано о частичном применении функций с использованием операции `std::bind` и лямбда-выражений, а также представлен необычный взгляд на функции, известный как *карринг*.
- В главе 5 говорится о важности неизменяемых данных, то есть объектов данных, которые невозможно модифицировать после создания. Здесь освещены проблемы, возникающие при наличии изменяемого состояния, и методы создания программ без присваивания переменным новых значений.
- Глава 6 посвящена подробному разбору понятия ленивых вычислений. В ней показано, как ленивый порядок вычислений можно использовать для оптимизации, начиная с простых задач наподобие конкатенации строк и до алгоритмов оптимизации, основанных на методе динамического программирования.
- В главе 7 рассказано о диапазонах – современном дополнении к алгоритмам стандартной библиотеки, призванном повысить удобство восприятия кода и его производительность.
- Глава 8 содержит изложение неизменяемых структур данных, т. е. структур данных, которые хранят историю своих предыдущих состояний, пополняя ее при каждой модификации.

Во второй части книги речь пойдет о более сложных понятиях, по большей части относящихся к разработке программ в функциональном стиле.

- В главе 9 речь идет о том, как избавить программу от недопустимых состояний с помощью суммы типов. Показано, как реализовать сумму типов на основе наследования и шаблона `std::variant` и как

для их обработки использовать перегруженные функциональные объекты.

- Глава 10 посвящена функторам и монадам – абстракциям, способным существенно помочь при работе с обобщенными типами и, в частности, позволяющим создавать функции для работы с векторами, значениями типа `optional` и фьючерсами.
- В главе 11 содержится объяснение техник метапрограммирования на шаблонах, способствующих функциональному программированию на языке C++. В частности, разобраны статическая интроспекция, вызываемые объекты и техники метапрограммирования на шаблонах, предназначенные для создания встроенных предметно-ориентированных языков.
- В главе 12 весь предшествующий материал книги собран воедино, чтобы продемонстрировать функциональный подход к разработке параллельных программных систем. В этой главе рассказано, как монаду продолжения можно применить для создания реактивных программ.
- Глава 13 знакомит читателя с функциональным подходом к тестированию и отладке программ.

Автор советует читателю по мере чтения книги реализовывать все изложенные в ней понятия и запускать все встречающиеся примеры кода. Большую часть подходов, описанных в книге, можно использовать и в старых версиях языка C++, однако это потребовало бы написания большого объема дублирующегося кода; поэтому в книге упор сделан главным образом на языковые стандарты C++14 и C++17.

Примеры кода созданы в предположении, что у читателя есть работающий компилятор с поддержкой стандарта C++17. Можно, например, пользоваться компилятором GCC, как автор этой книги, или Clang. Последние выпущенные версии обоих этих компиляторов одинаково хорошо поддерживают все элементы стандарта C++17, использованные в данной книге. Все примеры кода протестированы с версиями[1] GCC 7.2 и Clang 5.0.

Для работы с примерами можно пользоваться обычным текстовым редактором и запускать компилятор вручную из командной строки через посредство утилиты GNU `make` (к каждому примеру приложен несложный сценарий Makefile); также можно воспользоваться полновесной интегрированной средой разработки наподобие QtCreator (www.qt.io), Eclipse (www.eclipse.org), Kdevelop (www.kdevelop.org) или CLion (www.jetbrains.com/clion) и импортировать в нее примеры. Тем, кто намерен пользоваться средой Microsoft Visual Studio, рекомендуется установить наиболее свежую версию, какую возможно загрузить, и настроить ее на применение компилятора Clang вместо используемого по умолчанию компилятора Microsoft Visual C++ (MSVC), которому на момент написания этого введения не хватало некоторых важных мелочей.

---

[1] На момент перевода этого введения актуальны версии GCC 9.1 и Clang 8.0.0. Качество поддержки стандарта C++17 более не должно составлять предмета для беспокойства. – *Прим. перев.*

Хотя для компиляции большей части примеров не нужны никакие внешние зависимости, некоторым примерам все же нужны сторонние библиотеки, такие как range-v3, catch и JSON, клоны которых доступны вместе с кодом примеров в директории common/3rd-party, а также коллекция библиотек Boost, которую можно загрузить с сайта www.boost.org.

Оформление кода и загрузка примеров

Исходный код примеров к этой книге можно загрузить на сайте издательства по адресу www.manning.com/books/functional-programming-in-c-plus-plus и с системы GitLab по адресу https://gitlab.com/manning-fp-cpp-book.

В книге содержится много примеров исходного кода – как в виде нумерованных листингов, так и короткими вставками в обычный текст. В обоих случаях код набран `моноширинным шрифтом наподобие этого`, чтобы его легко было отличить от текста на естественном языке.

Во многих случаях первоначальный текст примеров пришлось видоизменить, добавив разрывы строк и отступы, чтобы улучшить размещение исходного кода на книжной странице. В редких случаях даже этого оказалось недостаточно, и тогда в листинг пришлось ввести знаки продолжения строки ➥. Кроме того, комментарии из исходного кода часто удалялись, если этот код сопровождается подробными пояснениями в основном тексте. Пояснения прилагаются ко многим листингам, помогая понять важные детали.

Спорить о стилях оформления исходного кода – превосходная возможность впустую потратить уйму времени. Это особенно характерно для языка C++, где едва ли не каждый проект претендует на собственный стиль.

Автор старался следовать соглашениям, используемым в ряде других книг по языку C++. Два стилистических правила стоит оговорить здесь особо:

- классы, моделирующие сущности из реального мира, например людей или домашних животных, получают имена с суффиксом `_t`. Благодаря этому становится проще понять в каждом конкретном случае, идет речь о реальном объекте (например, человеке) или о типе – имя `person_t` читается проще, чем «тип “человек”»;
- имена закрытых переменных-членов имеют приставку `m_`. Это отличает их от статических переменных-членов, чьи имена начинаются с префикса `s_`.

Форум книги

Покупка этой книги дает бесплатный доступ к закрытому форуму, функционирующему под управлением издательства Manning Publications, где читатели могут оставлять комментарии о книге, задавать технические вопросы и получать помощь от автора и других пользователей. Чтобы

получить доступ к форуму, нужно зайти на страницу https://forums.manning.com/forums/functional-programming-in-c-plus-plus. Больше о форумах издательства Manning и о принятых там правилах поведения можно узнать на странице https://forums.manning.com/forums/about.

Издательство Manning берет на себя обязательство перед своими читателями обеспечить им удобную площадку для общения читателей как между собой, так и с автором книги. Это, однако, не предполагает какого-либо определенного объема участия со стороны автора, чье общение на форуме остается исключительно добровольным и неоплачиваемым. Задавая автору интересные вопросы, читатели могут освежать и подогревать его интерес к теме книги. Форум и архивы предыдущих обсуждений будут доступны на сайте издательства все время, пока книга остается в продаже.

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв прямо на нашем сайте www.dmkpress.com, зайдя на страницу книги, и оставить комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com, при этом напишите название книги в теме письма.

Если есть тема, в которой вы квалифицированы, и вы заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу http://dmkpress.com/authors/publish_book/ или напишите в издательство по адресу dmkpress@gmail.com.

Скачивание исходного кода примеров

Скачать файлы с дополнительной информацией для книг издательства «ДМК Пресс» можно на сайте www.dmkpress.com или www.дмк.рф на странице с описанием соответствующей книги.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы удостовериться в качестве наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг – возможно, ошибку в тексте или в коде, – мы будем очень благодарны, если вы сообщите нам о ней. Сделав это, вы избавите других читателей от расстройств и поможете нам улучшить последующие версии этой книги.

Если вы найдете какие-либо ошибки в коде, пожалуйста, сообщите о них главному редактору по адресу dmkpress@gmail.com, и мы исправим это в следующих тиражах.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательства «ДМК Пресс» и Manning очень серьезно относятся к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконно выполненной копией любой нашей книги, пожалуйста, сообщите нам адрес копии или веб-сайта, чтобы мы могли применить санкции.

Пожалуйста, свяжитесь с нами по адресу dmkpress@gmail.com со ссылкой на подозрительные материалы.

Мы высоко ценим любую помощь по защите наших авторов, помогающую нам предоставлять вам качественные материалы.

Об авторе

Иван Чукич преподает современные методы программирования на языке C++ и функциональное программирование на факультете математики в Белграде. Он использует язык C++ с 1998 г. Он исследовал функциональное программирование на языке C++ перед и во время подготовки своей диссертации, а также применяет методы функционального программирования в реальных проектах, которыми пользуются сотни миллионов человек по всему миру. Иван – один из ключевых разработчиков среды KDE, крупнейшего в мире проекта с открытым кодом на языке C++.

1 Введение в функциональное программирование

О чем говорится в этой главе:

- понятие о функциональном программировании;
- рассуждение в терминах предназначения вместо шагов алгоритма;
- понятие о чистых функциях;
- преимущества функционального программирования;
- эволюция C++ в язык функционального программирования.

Каждому программисту приходится изучить за свою жизнь целый ряд языков программирования. Как правило, программист останавливается на двух или трех языках, на которых лично ему удобнее всего работать. Часто можно услышать, что изучить очередной язык программирования просто – что различия между языками в основном касаются лишь синтаксиса и что все языки предоставляют примерно одинаковые возможности. Тот, кто знает язык C++, наверняка легко выучит языки Java и C#, и наоборот.

В этом утверждении есть доля истины. Однако, берясь за изучение нового языка, мы невольно пытаемся имитировать на нем тот стиль программирования, который выработался в связи с предыдущим языком. Когда я впервые начал применять функциональный язык во время учебы в университете, то сразу начал с попыток определить на нем привычные операторы цикла `for` и `while` и оператор ветвления `if-then-else`. Это именно тот подход, которым пользуется большинство из нас, чтобы просто сдать экзамен и никогда больше не возвращаться к изученному.

Широко известен афоризм, что тот, у кого из инструментов есть лишь молоток, будет стремиться любую задачу считать гвоздем. Этот прин-

цип работает и в обратную сторону: если работать только с гвоздями, то любой попавший в руки инструмент будет использоваться как молоток. Многие программисты, попробовав язык функционального программирования, решают, что он не стоит затраченных усилий; они не видят в новом языке преимуществ, поскольку пытаются использовать этот новый инструмент таким же способом, как использовали бы старый.

Цель этой книги состоит не в том, чтобы научить читателя новому языку программирования; вместо этого книга призвана научить иному способу использования старого языка (а именно языка C++) – способу, настолько отличному от привычного, что у программиста впрямь может возникнуть *ощущение*, что он использует новый язык. Этот новый стиль программирования помогает создавать более продуманные программы и писать более безопасный, понятный и читаемый код и даже – не побоюсь сказать – более изящный, чем код, который обычно пишут на языке C++.

1.1 Что такое функциональное программирование

Функциональное программирование – это довольно старая парадигма, зародившаяся в академической среде в конце 1950-х годов и долгое время остававшаяся в этой нише. Хотя эта парадигма всегда была излюбленной темой для научных исследований, она никогда не пользовалась популярностью в «реальном мире». Вместо этого повсеместное распространение получили императивные языки: сперва процедурные, затем объектно-ориентированные.

Часто звучали предсказания, что однажды функциональные языки будут править миром, но этого до сих пор не произошло. Наиболее известные языки функционального программирования, такие как Haskell или LISP, все еще не входят в десятку наиболее популярных языков. Первые места по традиции прочно занимают императивные языки, к которым относятся языки C, Java и C++. Это предсказание, подобно большинству других, чтобы считаться сбывшимся, нуждается в известной свободе интерпретации. Вместо того чтобы популярность завоевали языки функционального программирования, произошло нечто иное: в популярные языки программирования стали добавлять все более элементов, заимствованных из языков функционального программирования.

Что *собой представляет* функциональное программирование (ФП)? На этот вопрос непросто ответить, так как не существует единого общепринятого определения. Согласно известному афоризму, если двух программистов, работающих в функциональном стиле, спросить, что такое ФП, они дадут по меньшей мере три различных ответа. Каждый специалист обычно старается определить ФП через другие связанные с ним понятия: чистые функции, ленивые вычисления, сопоставление с образцом и другие – и обычно при этом перечисляет характеристики своего любимого языка.