1 Введение в паттерны Rust

В этой главе

- ✓ Что такое паттерны проектирования.
- ✓ Чем эта книга отличается от других.
- ✓ Инструменты, необходимые для работы с Rust.

Кем бы вы ни были: начинающим, продвинутым или профессиональным программистом Rust, чтение этой книги станет отличным способом расширить навыки работы с этим языком. Если вы новичок, то изучение паттернов проектирования — прекрасная возможность повысить ваш уровень мастерства. Некоторые части книги могут показаться вам сложными, так что будьте готовы обращаться к сторонним ресурсам за дополнительной информацией. В книге приводятся различные приемы написания качественного кода на Rust, но мы сосредоточимся на паттернах, идиомах и соглашениях, которые широко используются в этом языке и хорошо известны в его сообществе.

Паттерны проектирования — это мощная абстракция, которую любой программист может использовать для написания качественного кода. Люди прекрасно распознают закономерности, поэтому использование понятных и легко узнаваемых паттернов в коде позволяет решить две задачи.

- Во-первых, паттерны дают возможность понять, является ли конкретная структура кода удачной (использование известных паттернов позволяет избежать написания плохого кода).
- Во-вторых, паттерны помогают другим людям понять наш код.

Читать код зачастую сложнее, чем писать. Когда мы читаем чей-то код, в котором используются определенные, известные нам паттерны, то становится проще понимать происходящее в нем. Если вы научите свой мозг узнавать наиболее типичные паттерны, то сможете упростить процесс оценки качества кода, а значит, и ошибок станет меньше.

Что касается программирования, то, зная, какие паттерны подходят для тех или иных ситуаций, вы сможете получить хороший результат за меньшее время. И это знание ничем не отличается от знания того, какие структуры данных или алгоритмы использовать в других обстоятельствах и к каким компромиссам для этого нужно прийти.

В этой книге вы практически не встретите догм. При описании всех паттернов я постараюсь подробно пояснять причины их использования. Будучи программистом, вы вольны экспериментировать, отклоняясь от представленных паттернов, чтобы создавать собственные структуры. Тем не менее я буду приводить продуманные условия, в целом предпочитая их возможности настройки.

Если использовать аналогию, то я предпочитаю пойти в ресторан, где шеф-повар предлагает всего пару блюд, заранее утвержденных как лучшие для этого сезона, а не просматривать десятки или сотни позиций меню, пытаясь понять, какое блюдо вкуснее. Лучшие рестораны обычно предлагают рекомендательный выбор (то есть основанный на доверии к вкусу шеф-повара), и я в этой книге постараюсь сделать так же.

Многие из фрагментов кода в ней являются выдержками из листингов, но вы всегда можете найти рабочие примеры кода на GitHub по адресу https://github.com/brndnmtthws/rust-advanced-techniques-book. Этот код доступен по лицензии Массачусетского технологического института (MIT), которая разрешает его использование, копирование и изменение без каких-либо ограничений. Если у вас есть такая возможность, то я рекомендую в процессе чтения книги работать с полноценными листингами кода (чтобы получить максимальную пользу от прочтения). Все фрагменты есть в репозитории GitHub, где они распределены по каталогам в соответствии с главами. Тем не менее некоторые примеры охватывают несколько разделов или даже глав и названы в зависимости от темы. Отдельные фрагменты кода в книге могут немного отличаться от представленных в репозитории, так как были отредактированы из соображений читабельности, краткости и удобства печати.

1.1. О чем эта книга

В книге мы рассмотрим различные идиомы, паттерны и схемы проектирования. Одни из них будут относиться исключительно к Rust. Другие будут служить иллюстрацией старых концепций, представленных в новом формате в рамках системы уникальных возможностей, синтаксиса и грамматики Rust.

Цель книги — помочь вам понять эти паттерны и научиться с их помощью улучшать программную архитектуру. Освоение и использование паттернов Rust позволит вам писать более эффективный, удобный в сопровождении и масштабируемый код. С каждым новым разделом я буду подробнее объяснять рассматриваемый паттерн и рассказывать, почему он важен и как применять его в реальных сценариях. Вдобавок мы обсудим компромиссы и важные нюансы, связанные с использованием каждого паттерна.

При всем этом важно понимать, что не нужно слепо использовать предложенные паттерны. Это лишь инструменты, которые вы можете адаптировать и изменять под свои конкретные нужды. Как я уже сказал выше, будучи программистом, вы вольны экспериментировать и отклоняться от описанных паттернов, создавая собственные уникальные структуры. Дочитав книгу до конца, вы будете твердо понимать различные идиомы и паттерны проектирования в Rust, а также получите знания и навыки для их эффективного применения в собственных проектах.

Многие из паттернов, рассматриваемых в классической работе «Банды четырех» «Паттерны объектно-ориентированного проектирования», относятся строго к объектно-ориентированному программированию (ООП) на C++. И разработчики Rust проделали прекрасную работу, сделав эти паттерны неактуальными за счет предоставления более эффективных альтернатив или добавления их в стандартную библиотеку (например, итераторов). И хотя предвестие грядущей смерти ООП сильно преувеличено, используемые в Rust абстракции после их освоения оказываются более понятными интуитивно.

Объектно-ориентированное программирование зачастую ведет к написанию лишнего шаблонного кода и использованию сложных паттернов. Иногда эту сложность ООП мы оправдываем некой необходимостью, однако, по сути, просто увеличиваем когнитивную нагрузку. Но сложные системы склонны чаще давать сбои, причем более критические, в сравнении с простыми. Кроме того, их сложнее понимать.

Я нахожу подход Rust к проектированию программного обеспечения и архитектуре более актуальным и надеюсь, что вы тоже. Разработчики этого языка отказались от множества устаревших элементов ООП, сосредоточившись на том, что необходимо для создания качественного ПО. Rust не страдает культом сложности, которым славятся языки наподобие C++ и Java.

1.2. Что такое паттерны проектирования

Дать определение понятию *«паттерны проектирования»* непросто — зачастую это понимаешь, только когда сталкиваешься с ними на практике. Чем больше паттернов вы изучаете, тем проще становится узнавать их в чужом коде или реализовывать самостоятельно. Освоение наиболее распространенных видов

паттернов позволит вам быстро узнавать их и повторять. *Паттернами* они называются потому, что часто повторяются в различных контекстах, а *паттернами проектирования* — потому, что представляют собой высокоуровневые абстракции, которые помогают разработчикам проектировать и создавать программное обеспечение.

Некоторые свойства паттернов проектирования типичны для всех паттернов и не ограничены конкретным языком программирования. Вот эти свойства (хотя их список может быть неполным).

- Их можно *использовать повторно*.
- Спектр их применения очень широк.
- Они решают задачи таким образом, который позволяет легко прослеживать механизм работы чужого кода.
- Они легко узнаваемы и понятны для опытных разработчиков.
- Код, в котором не используются устоявшиеся паттерны, может попадать в категорию *антипаттернов*.

В отношении последнего пункта вы можете подумать: «Но я же вот только что придумал такой классный паттерн!» Возможно, так и есть, но пока он не станет широко используемым и известным, *не стоит* ожидать, что другие будут понимать или применять его. Хорошие паттерны проектирования со временем становятся широко распространенными и являются легкими для понимания.

При этом, как я уже упоминал выше, паттерны лишь предоставляют знакомый шаблон для нового дизайна программного обеспечения, оставляя за вами выбор деталей реализации. Хороший паттерн применим к широкому спектру приложений и накладывает минимум ограничений на их авторов. Такие паттерны развиваются по мере появления в языке новых функций и парадигм, а суть многих фундаментальных паттернов за последние десятилетия изменилась несильно.

В книге я буду использовать широкое определение паттернов и паттернов проектирования. *Паттернами* я буду называть приемы, идиомы и соглашения, которые широко используются и известны в сообществе Rust. Эти паттерны могут быть как большими и сложными, имеющими в своем составе несколько структур и компонентов, так и весьма скромными, состоящими из одной функции или метода. Что же касается *паттернов проектирования*, то под ними я понимаю широко применимые паттерны, которые служат в качестве шаблонов для проектирования кода и решают распространенные задачи программирования. Я буду использовать термины «паттерны» и «паттерны проектирования» взаимозаменяемо, но обычно под первыми имеется в виду подмножество вторых.

Что такое антипаттерны

Антипаттерны — это «горе-родственники» паттернов проектирования. Обычно мы рассматриваем паттерны проектирования как правильный способ решения определенного класса задач. Исходя из этого, антипаттерны можно назвать ошибочным способом решения. В книге мы не будем разбирать их подробно, поскольку Rust разработан так, чтобы реализовать антипаттерны в нем было сложно в принципе.

Антипаттерны в большинстве случаев являются неправильным инструментом для выполнения задачи. Вы же не станете закручивать шуруп молотком, как и забивать гвоздь отверткой.

Речь об антипаттернах пойдет в главе 10. Но по ходу повествования я буду показывать, где не стоит использовать конкретные паттерны.

Кроме того, следует сразу проговорить отличие *паттернов* от *идиом* в контексте этой книги. Существует несколько определений различий этих терминов, но я сосредоточусь на двух основных моментах: идиомы обычно относятся к самому коду, а паттерны — к дизайну и архитектуре вашего ПО. Иными словами, паттерны состоят из идиом. Некоторые из них также могут быть идиомами (например, в них отдается предпочтение итераторам вместо циклов `for`), но идиома — это не паттерн, поскольку, например, использование змеиного регистра для имен переменных паттерном не является. Идиомы обычно имеют отношение к синтаксису и форматированию кода, например к соглашениям по именованию, стилю кода и прочим низкоуровневым деталям.

В иерархическом смысле можно рассматривать идиомы как нижний уровень абстракции, паттерны проектирования — как средний, а общую архитектуру — как верхний (рис. 1.1). Архитектура любой системы состоит из множества небольших единиц паттернов проектирования, которые сами состоят из множества идиом.

Рис. 1.1. Иерархия идиом, паттернов и архитектуры

Паттерны проектирования и языки программирования можно рассматривать аналогично тому, как мы рассматриваем письменные и разговорные языки. Любой язык развивается, в нем появляются новые слова, а старые выходят из моды.

Но если вы попробуете изобрести собственные слова или фразы, то они могут показаться другим людям бессмыслицей. Весь смысл любого языка в том, чтобы отчетливо доносить некий смысл, быть понятным для других и создавать чувство связи с остальными людьми. Если говорить в контексте программирования, то вы можете решить

отказаться от устоявшихся норм и двигаться в собственном ритме. И это может быть нормально, но тогда велик шанс, что другие не смогут понять ваш код и не захотят с ним работать. В некоторых случаях такой компромисс приемлем, но программное обеспечение часто используется в социальных контекстах, подразумевающих участие клиентов, пользователей, менеджеров, коллег и т. д. Один в поле не воин.

Об изучении паттернов проектирования сложно говорить, не упоминая книгу «Банды четырех» «Паттерны объектно-ориентированного проектирования», широко известную среди программистов как канонический учебник по данной теме. Эта книга была написана Эрихом Гаммой, Ричардом Хельмом, Ральфом Джонсоном и Джоном Влиссидесом и выпущена издательством Addison-Wesley Professional в 1994 году. В ней приводятся примеры, написанные на C++ и Smalltalk.

Некоторые представленные в ней паттерны с тех пор были привнесены во многие языки программирования в качестве основных составляющих. Лучшими из примеров, пожалуй, будут итераторы, которые есть практически в любом языке программирования и ключевой библиотеке. Объясняется это полезностью паттерна, качеством решения с его помощью задачи перебора элементов в структуре данных и понятностью. Реализовывать итераторы с нуля, чтобы понять принцип их работы, до сих пор весьма интересно, но в большинстве языков можно использовать их встроенные эквиваленты.

Паттерны проектирования являются одним из трех столпов создания хорошего программного обеспечения; два других — алгоритмы и структуры данных (рис. 1.2). Как разработчику ПО, вам нужно хорошо разбираться в каждой из этих областей. Освоения одних только паттернов проектирования будет недостаточно.

Рис. 1.2. Три столпа разработки хорошего программного обеспечения

Таким образом, паттерны проектирования — это высокоуровневые абстракции над основной грамматикой и синтаксисом языка программирования, которые позволяют эффективно передавать идеи и создавать качественный код. Правильная передача смысла является ответственностью отправителя сообщения, а не его получателя, хотя желательно, чтобы и тот и другой говорили на одном языке.

1.3. Чем эта книга отличается от других

После выхода книги «Банды четырех», ставшей впоследствии канонической, появилось много других книг по данной теме, и в этом смысле наша ничем от них не отличается. Тем не менее в ней представлены идеи, которые касаются конкретно Rust. Этот язык становится все более популярным, поэтому очень важно каталогизировать, документировать и описывать используемые в нем паттерны.

В отличие от работы «Банды четырех» моя книга не является каталогом, а содержит описание, примеры и реализацию конкретных паттернов. Есть две причины, по которым я не хочу собирать паттерны в каталог и классифицировать: они не являются просто шаблонами или шаблонным кодом, и, просто их копируя, вы приблизитесь к полноценному завершенному коду не более чем на 10 %. Эта книга для читателей, которые стремятся к знаниям и личному росту.

Приведу аналогию с едой. Конкретное блюдо (например, лазанья) можно рассмотреть в роли паттерна проектирования. Оно является частью большого обеда, состоящего из нескольких смен блюд, напитков и безупречного сервиса. Реальная же сложность для повара в том, чтобы решить, как изготовить свою версию блюда, где взять ингредиенты и как все объединить, а затем преподнести в аппетитном виде. (Как известно любому, кто работал в ресторане, презентация — самый важный элемент.) Программирование же одновременно и наука, и искусство. Это очень креативный процесс, который выходит за рамки простого написания кода. И подражанием здесь мало чего добьешься.

Уникальные особенности Rust требуют уделять больше внимания проектированию API и созданию качественного кода. В частности, нужно продумывать управление памятью и временем жизни объектов, передавать значения между контекстами, избегать состояний гонки и следить, чтобы API были эргономичными. Кроме того, Rust полон возможностей для создания или открытия новых паттернов, которые определенно продолжат развиваться и после выхода этой книги. Прежде чем мы сможем отправиться на Марс, нам нужно создать ракету, которая нас туда доставит, а также решить множество задач, которые возникнут на протяжении семи месяцев полета.

Rust — удивительный язык, уникальность которого отчасти связана с тем, как он развивался, — это полностью заслуга участников сообщества. Его абстракции одновременно открывают новые паттерны, делая старые неактуальными. Изучить синтаксис языка — это одна задача, а вот писать качественный код, используя корректные паттерны в нужных местах, причем делая это правильно, — совсем другая.

1.4. Инструменты, необходимые для работы с Rust

В книге приводятся примеры кода, которые свободно доступны по лицензии MIT. Для получения копии кода вам потребуется подключенный к интернету компьютер с поддерживаемой операционной системой (https://mng.bz/JZpa),

а также инструменты, перечисленные в табл. 1.1. Подробнее об установке этих инструментов читайте в приложении.

Таблица 1.1. Необходимые инструменты

Название	Описание
`git`	Исходный код книги хранится в открытом репозитории на GitHub по адресу https://github.com/brndnmtthws/idiomatic-rust-book
`rustup`	Инструмент Rust для управления компонентами языка. Будет регулировать установку `rustc` и прочих компонентов
`gcc` или `clang`	Сборка некоторых примеров кода потребует наличия копии GCC или Clang. Для большинства Clang наверняка станет более подходящим выбором, поэтому по умолчанию мы будем иметь в виду его. Но при желании, встречая команду `clang`, можете свободно заменять ее на `gcc`

Резюме

- Хорошие паттерны проектирования можно использовать повторно, они применимы во множестве ситуаций и могут решать типичные задачи программирования.
- Отличительные особенности хорошего паттерна проектирования — его понятность и то, что со временем он становится широко распространенным.
- Антипаттернами называются сложные, вредные или нечетко определенные паттерны.
- В этой книге приводятся характерные для Rust паттерны проектирования, которые используют уникальные особенности языка и его инструментов.
- Вам потребуется свежая версия Rust, Git и современный компилятор наподобие GNU GCC или LLVM Clang.
- Если вы хотите получить максимум пользы от прочтения книги, то рекомендую параллельно изучать примеры кода, доступные по адресу https://github.com/brndnmtthws/idiomatic-rust-book.