

Оглавление

Вступление	19
Предисловие	20
Благодарности	21
Об авторах.....	22
Введение	23
Кому подойдет язык Rust.....	23
Команды разработчиков.....	23
Студенты.....	24
Компании	24
Разработчики открытого исходного кода	24
Люди, ценящие скорость и стабильность.....	24
Для кого эта книга	25
Как пользоваться этой книгой	25
Ресурсы	27
От издательства	27
Глава 1. Начало работы	28
Установка.....	28
Установка инструмента rustup в Linux или macOS.....	29
Установка инструмента rustup в Windows	30
Обновление и деинсталляция.....	30
Устранение неисправностей	30
Локальная документация	31
Здравствуй, Мир!.....	31
Создание каталога проектов	31
Написание и выполнение программы Rust.....	32
Анатомия программы на языке Rust	33
Компиляция и выполнение являются отдельными шагами.....	34
Здравствуй, Cargo!	35
Создание проекта с помощью Cargo	35
Построение проекта Cargo и его выполнение	37
Сборка для релиза	39

Cargo как общепринятое средство.....	39
Итоги.....	40
Глава 2. Программирование игры-угадайки	41
Настройка нового проекта.....	41
Обработка загаданного числа.....	42
Хранение значений с помощью переменных.....	43
Обработка потенциального сбоя с помощью типа Result	45
Печать значений с помощью заполнителей макрокоманды println!	47
Тестирование первой части.....	47
Генерирование секретного числа	47
Использование упаковки для получения большей функциональности	48
Генерирование случайного числа.....	50
Сравнение загаданного числа с секретным числом.....	52
Вывод нескольких загаданных чисел с помощью цикличности.....	56
Выход из игры после правильно угаданного числа	57
Обработка ввода недопустимых данных	58
Итоги.....	60
Глава 3. Концепции программирования	61
Переменные и изменяемость	61
Различия между переменными и константами	63
Затенение	64
Типы данных	66
Скалярные типы	67
Составные типы	71
Функции	74
Параметры функций.....	75
Инструкции и выражения в телах функций.....	76
Функции с возвращаемыми значениями	78
Комментарии.....	80
Управление потоком.....	81
Выражения if.....	81
Повторение с помощью циклов	85
Итоги.....	89
Глава 4. Концепция владения	90
Что такое владение?	90
Правила владения.....	92

Область видимости переменной	92
Строковый тип	93
Память и выделение пространства	94
Владение и функции	100
Возвращаемые значения и область видимости	101
Ссылки и заимствование.....	102
Изменяемые ссылки	104
Висячие ссылки	107
Правила ссылок	108
Срезовой тип	109
Строковые срезы	111
Другие срезы	115
Итоги.....	115
Глава 5. Использование структур для связанных данных.....	116
Определение и инстанцирование структур	116
Использование краткой инициализации полей: когда у переменных и полей одинаковые имена	118
Создание экземпляров из других экземпляров с помощью синтаксиса обновления структуры.....	118
Использование кортежных структур без именованных полей для создания разных типов	119
Unit-подобные структуры без полей	120
Пример программы с использованием структур	121
Рефакторинг с использованием кортежей	122
Рефакторинг с использованием структур: добавление большего смысла	123
Добавление полезной функциональности с использованием типажей с атрибутом <code>derived</code>	124
Синтаксис методов	126
Определение методов	126
Методы с большим числом параметров	128
Связанные функции	129
Несколько блоков <code>impl</code>	130
Итоги.....	131
Глава 6. Перечисления и сопоставление с паттернами.....	132
Определение перечисления.....	132
Выражение <code>match</code> как оператор управления потоком	139
Паттерны, которые привязываются к значениям	141

Сопоставление с Option<T>	142
Совпадения являются исчерпывающими	143
Заполнитель _	144
Сжатое управление потоком с помощью if let	145
Итоги.....	146
Глава 7. Управление растущими проектами с помощью пакетов, упаковок и модулей	147
Пакеты и упаковки	148
Определение модулей для управления областью видимости и конфиденциальностью	149
Пути для ссылки на элемент в дереве модулей.....	151
Демонстрация путей с помощью ключевого слова pub	154
Начало относительных путей с помощью super	156
Обозначение структур и перечислений как публичных.....	157
Введение путей в область видимости с помощью ключевого слова use	159
Создание идиоматических путей use	160
Предоставление новых имен с помощью ключевого слова as	162
Резкспорт имен с использованием pub	162
Использование внешних пакетов.....	163
Использование вложенных путей для очистки больших списков use	164
Оператор glob	165
Разделение модулей на разные файлы	165
Итоги.....	167
Глава 8. Общие коллекции	168
Хранение списков значений с помощью векторов	168
Создание нового вектора	169
Обновление вектора	169
Отбрасывание вектора отбрасывает его элементы	170
Чтение элементов вектора	170
Перебор значений в векторе	172
Использование перечисления для хранения нескольких типов.....	173
Хранение текста в кодировке UTF-8 с помощью строк	174
Что такое тип String?.....	174
Создание нового экземпляра типа String	175
Обновление строки	176
Индексирование в строках	179
Нарезка строк	181

Методы перебора строк.....	182
Строки не так просты.....	182
Хранение ключей со связанными значениями в хеш-отображениях.....	183
Создание нового хеш-отображения.....	183
Хеш-отображения и владение.....	184
Доступ к значениям в хеш-отображении.....	185
Обновление хеш-отображения.....	186
Хеширующие функции.....	188
Итоги.....	188
Глава 9. Обработка ошибок.....	190
Неустранимые ошибки и макрокоманда panic!.....	190
Использование обратной трассировки при вызове panic!.....	192
Устраняемые ошибки с помощью Result.....	194
Применение выражения match с разными ошибками.....	197
Краткие формы для паники в случае ошибки: unwrap и expect.....	198
Распространение ошибок.....	200
Паниковать! Или не паниковать!.....	205
Примеры, прототипный код и тесты.....	205
Случаи, когда у вас больше информации, чем у компилятора.....	206
Принципы обработки ошибок.....	206
Создание настраиваемых типов для проверки допустимости.....	208
Итоги.....	210
Глава 10. Обобщенные типы, типаж и жизненный цикл.....	211
Удаление повторов путем извлечения функции.....	212
Обобщенные типы данных.....	214
В определениях функций.....	214
В определениях структуры.....	217
В определениях перечислений.....	219
В определениях методов.....	220
Производительность кода с использованием обобщений.....	222
Типаж: определение совместного поведения.....	223
Определение типажа.....	223
Реализация типажа в типе.....	224
Реализации по умолчанию.....	226
Типаж в качестве параметров.....	228
Возвращение типов, реализующих типаж.....	230

Исправление функции <code>largest</code> с помощью границ типажа	231
Использование границ типажа для условной реализации методов	233
Проверка ссылок с помощью жизненных циклов	235
Предотвращение висячих ссылок с помощью жизненного цикла	235
Контролер заимствования	236
Обобщенные жизненные циклы в функциях	237
Синтаксис аннотаций жизненных циклов	239
Аннотации жизненных циклов в сигнатурах функций	240
Мышление в терминах жизненных циклов	242
Аннотации жизненных циклов в определениях структур	244
Пропуск жизненного цикла	244
Аннотации жизненных циклов в определениях методов	247
Статический жизненный цикл	248
Параметры обобщенного типа, границы типажа и жизненный цикл вместе	249
Итоги	249
Глава 11. Автоматизированные тесты	250
Как писать тесты	251
Анатомия функции тестирования	251
Проверка результатов с помощью макрокоманды <code>assert!</code>	255
Проверка равенства с помощью макрокоманд <code>assert_eq!</code> и <code>assert_ne!</code>	257
Добавление сообщений об ошибках для пользователя	260
Проверка на панику с помощью атрибута <code>should_panic</code>	261
Использование типа <code>Result<T, E></code> в тестах	265
Контроль выполнения тестов	265
Параллельное и последовательное выполнение тестов	266
Показ результатов функции	267
Выполнение подмножества тестов по имени	268
Игнорирование нескольких тестов, только если не запрошено иное	270
Организация тестов	271
Модульные тесты	271
Интеграционные тесты	273
Интеграционные тесты для двоичных упаковок	277
Итоги	277
Глава 12. Проект ввода-вывода: сборка программы командной строки	278
Принятие аргументов командной строки	279
Чтение значений аргументов	279
Сохранение значений аргументов в переменных	281

Чтение файла	282
Рефакторинг с целью улучшения модульности и обработки ошибок	283
Разделение обязанностей в двоичных проектах	284
Исправление обработки ошибок	289
Извлечение алгоритма из функции main	292
Разбивка кода в библиотечную упаковку	295
Развитие функциональности библиотеки с помощью методики разработки на основе тестов	296
Написание провального теста	297
Написание кода для успешного завершения теста	299
Работа с переменными среды	302
Написание провального теста для функции search, нечувствительной к регистру	303
Реализация функции search_case_insensitive	304
Запись сообщений об ошибках в стандартный вывод ошибок вместо стандартного вывода данных	308
Проверка места, куда записываются ошибки	308
Запись сообщения об ошибках в стандартный вывод ошибок	309
Итоги	310

Глава 13. Функциональные средства языка:

итераторы и замыкания	311
Замыкание: анонимные функции, которые могут захватывать среду	311
Создание абстракции поведения с помощью замыканий	312
Логический вывод типа и аннотация замыкания	317
Ограничения в реализации структуры Cacher	322
Захватывание среды с помощью замыканий	323
Обработка серии элементов с помощью итераторов	326
Типаж Iterator и метод next	327
Методы, которые потребляют итератор	328
Методы, которые производят другие итераторы	329
Использование замыканий, которые захватывают свою среду	330
Создание собственных итераторов с помощью типажа Iterator	331
Улучшение проекта ввода-вывода	334
Удаление метода clone с помощью Iterator	334
Написание более ясного кода с помощью итераторных адаптеров	337
Сравнение производительности: циклы против итераторов	338
Итоги	340

Глава 14. Подробнее о Cargo и Crates.io	341
Собственная настройка сборок с помощью релизных профилей	341
Публикация упаковки для Crates.io	343
Внесение полезных документационных комментариев	343
Экспорт удобного публичного API с использованием pub	347
Настройка учетной записи Crates.io	351
Добавление метаданных в новую упаковку	351
Публикация в Crates.io	353
Публикация новой версии существующей упаковки.....	353
Удаление версий из Crates.io с помощью команды cargo yank.....	353
Рабочие пространства Cargo.....	354
Создание рабочего пространства	354
Создание второй упаковки в рабочем пространстве	355
Установка двоичных файлов из Crates.io с помощью команды cargo install.....	360
Расширение Cargo с помощью индивидуальных команд.....	361
Итоги.....	361
Глава 15. Умные указатели.....	362
Использование <code>Box<T></code> для указания на данные в куче	363
Использование <code>Box<T></code> для хранения данных в куче	364
Применение рекурсивных типов с помощью умных указателей <code>Box</code>	365
Трактовка умных указателей как обыкновенных ссылок с помощью типажа <code>Deref</code>	369
Следование по указателю к значению с помощью оператора разыменования	370
Использование <code>Box<T></code> в качестве ссылки	371
Определение собственного умного указателя.....	371
Трактовка типа как ссылки путем реализации типажа <code>Deref</code>	372
Скрытые принудительные приведения типов посредством <code>deref</code> с функциями и методами.....	374
Как принудительное приведение типа посредством <code>deref</code> взаимодействует с изменяемостью	375
Выполнение кода при очистке с помощью типажа <code>Drop</code>	376
Досрочное отбрасывание значения с помощью <code>std::mem::drop</code>	378
<code>Rc<T></code> — умный указатель подсчета ссылок.....	380
Применение <code>Rc<T></code> для совместного использования данных.....	380
Клонирование <code>Rc<T></code> увеличивает число ссылок	383
<code>RefCell<T></code> и паттерн внутренней изменяемости	384

Соблюдение правил заимствования во время выполнения с помощью RefCell<T>	384
Внутренняя изменяемость: изменяемое заимствование неизменяемого значения.....	386
Наличие нескольких владельцев изменяемых данных путем сочетания Rc<T> и RefCell<T>	392
Циклы в переходах по ссылкам приводят к утечке памяти.....	394
Создание цикла в переходах по ссылкам.....	394
Предотвращение циклов в переходах по ссылкам: превращение Rc<T> в Weak<T>.....	397
Итоги.....	403
Глава 16. Конкурентность без страха	404
Использование потоков исполнения для одновременного выполнения кода.....	405
Создание нового потока с помощью spawn.....	407
Ожидание завершения работы всех потоков с использованием дескрипторов join.....	408
Использование замыкания move с потоками.....	410
Использование передачи сообщений для пересылки данных между потоками.....	413
Каналы и передача владения	416
Отправка нескольких значений и ожидание приемника	417
Создание нескольких производителей путем клонирования передатчика.....	418
Конкурентность совместного состояния.....	420
Использование мьютексов для обеспечения доступа к данным из одного потока за один раз.....	420
Сходства между RefCell<T>/Rc<T> и Mutex<T>/Arc<T>	428
Расширяемая конкурентность с типажми Send и Sync	428
Разрешение передавать владение между потоками с помощью Send.....	429
Разрешение доступа из нескольких потоков исполнения с помощью Sync	429
Реализовывать Send и Sync вручную небезопасно.....	430
Итоги.....	430
Глава 17. Средства объектно-ориентированного программирования	431
Характеристики объектно-ориентированных языков.....	431
Объекты содержат данные и поведение	432
Инкапсуляция, которая скрывает детали реализации	432
Наследование как система типов и как совместное использование кода	434
Использование типажных объектов, допускающих значения разных типов	435
Определение типажа для часто встречающегося поведения	436

Реализация типажа	438
Типажные объекты выполняют динамическую диспетчеризацию	441
Объектная безопасность необходима для типажных объектов.....	442
Реализация объектно-ориентированного паттерна проектирования	443
Определение поста и создание нового экземпляра в состоянии черновика ...	445
Хранение текста поста	446
Делаем пустой черновик	446
Запрос на проверку статьи изменяет ее состояние	447
Добавление метода approve, который изменяет поведение метода content	449
Компромиссы паттерна переходов между состояниями	452
Итоги.....	457
Глава 18. Паттерны и сопоставление.....	458
Где могут использоваться паттерны	459
Ветви выражения match	459
Условные выражения if let.....	459
Условные циклы while let.....	461
Циклы for	461
Инструкции let.....	462
Параметры функций.....	463
Опровержимость: возможность несовпадения паттерна	464
Синтаксис паттернов	466
Сопоставление литералов	466
Сопоставление именованных переменных	466
Несколько паттернов.....	468
Сопоставление интервалов значений с помощью синтаксиса ...	468
Деструктурирование для выделения значений	469
Игнорирование значений в паттерне.....	474
Дополнительные условия с ограничителями совпадений.....	479
Привязки @.....	481
Итоги.....	482
Глава 19. Продвинутое средства	483
Небезопасный Rust	483
Небезопасные сверхспособности	484
Применение оператора разыменования к сырому указателю.....	485
Вызов небезопасной функции или метода	487

Обращение к изменяемой статической переменной или ее модифицирование	492
Реализация небезопасного типажа	493
Когда использовать небезопасный код	494
Продвинутые типажи	494
Детализация заполнительных типов в определениях типажей с помощью связанных типов	494
Параметры обобщенного типа по умолчанию и перегрузка операторов	496
Полный синтаксис для устранения неоднозначности: вызов методов с одинаковым именем	498
Использование супертипажей, требующих функциональности одного типажа внутри другого типажа	502
Использование паттерна <code>newtype</code> для реализации внешних типажей во внешних типах	504
Продвинутые типы	505
Использование паттерна <code>newtype</code> для безопасности типов и абстракции	506
Создание синонимов типов с помощью псевдонимов типов	506
Тип <code>never</code> , который никогда не возвращается	508
Динамически изменяемые типы и типаж <code>Sized</code>	510
Продвинутые функции и замыкания	512
Указатели функций	512
Возвращающие замыкания	514
Макрокоманды	515
Разница между макрокомандами и функциями	516
Декларативные макрокоманды с помощью <code>macro_rules!</code> для общего метапрограммирования	516
Процедурные макрокоманды для генерирования кода из атрибутов	519
Как написать настраиваемую макрокоманду <code>derive</code>	520
Макрокоманды, подобные атрибутам	525
Макрокоманды, подобные функциям	526
Итоги	527
Глава 20. Финальный проект: сборка многопоточного сервера	528
Сборка однопоточного сервера	529
Прослушивание TCP-соединения	529
Чтение запроса	531
HTTP-запрос	533
Написание ответа	534
Возвращение реального HTML	535

Проверка запроса и выборочный ответ	537
Небольшой рефакторинг	538
Превращение однопоточного сервера в многопоточный.....	540
Моделирование медленного запроса в текущей реализации сервера	540
Повышение пропускной способности с помощью пула потоков исполнения	541
Корректное отключение и очистка	561
Реализация типажа Drop для ThreadPool	562
Подача потокам сигнала об остановке прослушивания заданий	564
Итоги.....	569
Приложение А. Ключевые слова	570
Ключевые слова, употребляемые в настоящее время	570
Ключевые слова, зарезервированные для использования в будущем	572
Сырые идентификаторы	572
Приложение Б. Операторы и символы	574
Операторы	574
Неоператорные символы	576
Приложение В. Генерируемые типы	581
Debug для вывода рабочей информации	582
PartialEq и Eq для сравнений равенств.....	582
PartialOrd и Ord для сравнений порядка.....	583
Clone и Copy для дублирования значений.....	583
Хеш для отображения значения в значение фиксированного размера.....	584
Default для значений по умолчанию	585
Приложение Г. Полезные инструменты разработки	586
Автоматическое форматирование с помощью rustfmt	586
Исправляйте код с помощью rustfix	586
Статический анализ кода с помощью Clippy.....	588
Интеграция с IDE с помощью языкового сервера Rust Language Server.....	589
Приложение Д. Редакции.....	590