

Оглавление



Предисловие	20
Благодарности	21
Об этой книге	22
Кому адресована книга	22
Структура издания	22
О примерах программного кода	23
Об авторе	23
От издательства	24
Часть I. Функциональный инструментарий	
1 Изучение функционального программирования	26
Возможно, вы купили эту книгу потому, что	27
Что нужно знать перед тем, как начать	28
Как выглядят функции	29
Встречайте: функция	30
Когда код лжет	31
Императивный и декларативный стили	32
Кофе-брейк: императивный и декларативный стили	33
Объяснение для кофе-брейка: императивный и декларативный стили	34
Насколько полезно изучать функциональное программирование	35

Прыжок в Scala	36
Практика функций в Scala	37
Подготовка инструментов	38
Знакомство с REPL	39
Пишем свои первые функции!	40
Как использовать эту книгу	41
Резюме	42

2 Чистые функции 43

Зачем нужны чистые функции	44
Императивное решение	45
Ошибка в коде	46
Передача копий данных	47
Ошибка в коде... снова	48
Повторные вычисления вместо сохранения	49
Сосредоточение внимания на логике путем передачи состояния	50
Куда пропало состояние	51
Разница между чистыми и нечистыми функциями	52
Кофе-брейк: преобразование в чистую функцию	53
Объяснение для кофе-брейка: преобразование в чистую функцию	54
Мы доверяем чистым функциям	56
Чистые функции в языках программирования	57
Трудно оставаться чистым	58
Чистые функции и чистый код	59
Кофе-брейк: чистая или нечистая	60
Объяснение для кофе-брейка: чистая или нечистая	61
Использование Scala для написания чистых функций	62
Практика чистых функций в Scala	63
Тестирование чистых функций	64
Кофе-брейк: тестирование чистых функций	65
Объяснение для кофе-брейка: тестирование чистых функций	66
Резюме	67

3 Неизменяемые значения 68

Топливо для двигателя	69
Еще один пример неизменяемости	70
Можно ли доверять этой функции	71
Изменяемость опасна	72

.....	Функции, которые лгут... снова.....	73
.....	Борьба с изменяемостью за счет использования копий.....	74
.....	Кофе-брейк: обжигаемся на изменяемости.....	75
.....	Объяснение для кофе-брейка: обжигаемся на изменяемости.....	76
.....	Знакомьтесь: общее изменяемое состояние.....	80
.....	Влияние состояния на возможность программирования.....	82
.....	Работа с движущимися частями.....	84
.....	Работа с движущимися частями в ФП.....	85
.....	Неизменяемые значения в Scala.....	86
.....	Вырабатываем интуитивное понимание неизменности.....	87
.....	Кофе-брейк: неизменяемый тип String.....	88
.....	Объяснение для кофе-брейка: неизменяемый тип String.....	89
.....	Постойте... Разве это не плохо?.....	90
.....	Чисто функциональный подход к общему изменяемому состоянию.....	91
.....	Практика работы с неизменяемыми списками.....	93
.....	Резюме.....	94
4	Функции как значения	95
.....	Реализация требований в виде функций.....	96
.....	Нечистые функции и изменяемые значения наносят ответный удар.....	97
.....	Использование Java Streams для сортировки списка.....	98
.....	Сигнатуры функций должны рассказывать всю правду.....	99
.....	Изменение требований.....	100
.....	Мы можем передавать код в аргументах!.....	102
.....	Использование значений Function в Java.....	103
.....	Использование синтаксиса Function для устранения повторяющегося кода.....	104
.....	Передача пользовательских функций в виде аргументов.....	105
.....	Кофе-брейк: функции как параметры.....	106
.....	Объяснение для кофе-брейка: функции как параметры.....	107
.....	Проблемы с чтением функционального кода на Java.....	108
.....	Передача функций в Scala.....	109
.....	Глубокое погружение в sortBy.....	110
.....	Сигнатуры с параметрами-функциями в Scala.....	111
.....	Передача функций в виде аргументов в Scala.....	112
.....	Практика передачи функций.....	113
.....	Использование декларативного программирования.....	114
.....	Передача функций пользовательским функциям.....	115
.....	Маленькие функции и их обязанности.....	116

Передача встроенных функций.....	117
Кофе-брейк: передача функций в Scala.....	118
Объяснение для кофе-брейка: передача функций в Scala.....	119
Чего еще можно добиться, просто передавая функции.....	120
Применение функции к каждому элементу списка.....	121
Применение функции к каждому элементу списка с помощью map.....	122
Знакомство с map.....	123
Практика map.....	124
Изучите однажды, используйте постоянно.....	125
Возврат части списка, соответствующей условию.....	126
Возврат части списка с помощью filter.....	127
Знакомство с filter.....	128
Практика filter.....	129
Насколько далеко мы зашли в нашем путешествии.....	130
Не повторяйся?.....	131
Легко ли использовать мой API.....	132
Добавления нового параметра недостаточно.....	133
Функции могут возвращать функции.....	134
Использование функций, возвращающих функции.....	135
Функции — это значения.....	136
Кофе-брейк: возврат функций.....	137
Объяснение для кофе-брейка: возврат функций.....	138
Проектирование функциональных API.....	139
Итеративный дизайн функциональных API.....	140
Возврат функций из возвращаемых функций.....	141
Как вернуть функцию из возвращаемой функции.....	142
Использование гибкого API, построенного с использованием возвращаемых функций.....	143
Использование нескольких списков параметров в функциях.....	144
У нас есть карринг!.....	145
Практика каррирования.....	146
Программирование с передачей функций в виде значений.....	147
Свертка множества значений в одно.....	148
Свертка множества значений в одно с помощью foldLeft.....	149
Знакомство с foldLeft.....	150
Каждый должен знать и уметь пользоваться foldLeft.....	151
Практика foldLeft.....	152
Моделирование неизменяемых данных.....	153
Использование типов-произведений с функциями высшего порядка.....	154
Более лаконичный синтаксис встроенных функций.....	155
Резюме.....	156

Часть II. Функциональные программы

5 Последовательные программы 158

Написание конвейерных алгоритмов.....	159
Составление больших программ из мелких деталей.....	160
Императивный подход.....	161
flatten и flatMap.....	162
Практические примеры использования flatMap.....	163
flatMap и изменение размера списка.....	164
Кофе-брейк: работа со списками списков.....	165
Объяснение для кофе-брейка: работа со списками списков.....	166
Объединение в цепочку вызовов flatMap и map.....	167
Вложенные вызовы flatMap.....	168
Значения, зависящие от других значений.....	169
Практика использования вложенных вызовов flatMap.....	170
Улучшенный синтаксис вложенных вызовов flatMap.....	171
for-выражения во спасение!.....	172
Кофе-брейк: flatMap и for-выражение.....	173
Объяснение кофе-брейка: flatMap и for-выражение.....	174
Знакомство с for-выражениями.....	175
Это не тот for, который вы знаете!.....	176
Внутреннее устройство for-выражения.....	177
Более сложные for-выражения.....	178
Проверка всех комбинаций с помощью for-выражения.....	179
Приемы фильтрации.....	180
Кофе-брейк: методы фильтрации.....	181
Объяснение для кофе-брейка: методы фильтрации.....	182
В поисках большей абстракции.....	183
Сравнение map, foldLeft и flatMap.....	184
Использование for-выражений с множествами Set.....	185
Использование for-выражений с данными нескольких типов.....	186
Практика for-выражений.....	187
Определение for-выражения... снова.....	188
Использование for-выражений с типами, не являющимися коллекциями.....	189
Избегайте значений null: тип Option.....	190
Парсинг в виде конвейера.....	191
Кофе-брейк: парсинг с Option.....	192
Объяснение кофе-брейка: парсинг с Option.....	193
Резюме.....	194

6	Обработка ошибок	195
	Изящная обработка множества различных ошибок.....	196
	Возможно ли вообще справиться со всеми ними.....	197
	Сортировка списка телесериалов по продолжительности их выхода.....	198
	Реализация требования сортировки.....	199
	Обработка данных, поступающих из внешнего мира.....	200
	Функциональный дизайн: конструирование из небольших блоков.....	201
	Парсинг строк в неизменяемые объекты.....	202
	Парсинг списка — это парсинг одного элемента.....	203
	Парсинг String в TvShow.....	204
	А как насчет возможных ошибок?.....	205
	Является ли возврат null хорошей идеей?.....	206
	Как наиболее изящно обрабатывать потенциальные ошибки.....	207
	Реализация функции, возвращающей Option.....	208
	Option вынуждает обрабатывать возможные ошибки.....	209
	Конструирование из небольших блоков.....	210
	Функциональный дизайн состоит из маленьких блоков.....	211
	Написание небольшой безопасной функции, возвращающей Option.....	212
	Функции, значения и выражения.....	215
	Практика безопасных функций, возвращающих Option.....	216
	Как распространяются ошибки.....	217
	Значения представляют ошибки.....	218
	Option, for-выражения и контролируемые исключения.....	219
	Не лучше ли использовать контролируемые исключения?.....	220
	Условное восстановление.....	221
	Условное восстановление в императивном стиле.....	222
	Условное восстановление в функциональном стиле.....	223
	Контролируемые исключения не комбинируются друг с другом, в отличие от значений Option!.....	224
	Как работает orElse.....	225
	Практика функциональной обработки ошибок.....	226
	Функции комбинируются даже при наличии ошибок.....	227
	Компилятор напоминает, что ошибки должны быть обработаны.....	228
	Ошибки компиляции нам на пользу!.....	229
	Преобразование списка значений Option в простой список.....	230
	Пусть компилятор будет нашим проводником.....	231
	...но не будем слишком доверять компилятору!.....	232
	Кофе-брейк: стратегии обработки ошибок.....	233
	Объяснение для кофе-брейка: стратегии обработки ошибок.....	234
	Две разные стратегии обработки ошибок.....	235

Стратегия обработки ошибок «все или ничего»	236
Свертка списка значений Option в значение Option со списком	238
Теперь мы знаем, как обработать множество ошибок одновременно!	239
Как узнать, в чем причина неудачи	240
Мы должны передать информацию об ошибке в возвращаемом значении	241
Передача сведений об ошибке с использованием Either	242
Переход на использование Either	243
Возврат Either вместо Option	244
Практика безопасных функций, возвращающих Either	248
Навыки работы с Option пригодились и для работы с Either	249
Кофе-брейк: обработка ошибок с использованием Either	250
Объяснение для кофе-брейка: обработка ошибок с использованием Either	251
Работа с Option/Either	252
Резюме	253
7 Требования как типы	254
Моделирование данных для минимизации ошибок программистов	255
Хорошо смоделированные данные не лгут	256
Проектирование с использованием уже известного нам (простых типов)	257
Использование данных, смоделированных как простые типы	258
Кофе-брейк: недостатки простых типов	259
Объяснение для кофе-брейка: недостатки простых типов	260
Проблемы использования простых типов в моделях	261
Использование простых типов усложняет нашу работу!	262
Новые типы защищают от передачи параметров не на своих местах	263
Использование новых типов в моделях данных	264
Практика использования новых типов	265
Гарантии возможности только допустимых комбинаций данных	266
Моделирование возможности отсутствия данных	267
Изменения в модели вызывают изменения в логике	268
Использование данных, смоделированных как значения Option	269
Функции высшего порядка решают!	270
Вероятно, для решения этой проблемы существует функция высшего порядка!	271
Кофе-брейк: forall/exists/contains	272
Объяснение для кофе-брейка: forall/exists/contains	273
Объединение понятий внутри одного типа-произведения	274
Моделирование конечных диапазонов значений	275

Использование типа-суммы	276
Улучшение модели с помощью типов-сумм	277
Использование комбинации «тип-сумма + тип-произведение»	278
Типы-произведения + типы-суммы = алгебраические типы данных (ADT)	279
Использование моделей на основе ADT в реализациях поведения (функциях)	280
Деструктуризация ADT с помощью сопоставления с образцом	281
Дублирование кода и правило DRY	282
Практика сопоставления с образцом	283
Новые типы, ADT и сопоставление с образцом в дикой природе	284
Что можно сказать о наследовании	285
Кофе-брейк: проектирование функциональных данных	286
Объяснение для кофе-брейка: дизайн функциональных данных	288
Моделирование поведения	289
Моделирование поведения как данных	290
Реализация функций с параметрами на основе ADT	291
Кофе-брейк: проектирование и удобство сопровождения	292
Объяснение для кофе-брейка: проектирование и удобство сопровождения	293
Резюме	294
8 Ввод-вывод как значения	296
Общение с внешним миром	297
Интеграция с внешним API	298
Свойства операции ввода-вывода с побочным эффектом	299
Императивное решение для кода ввода-вывода с побочными эффектами	300
Проблемы императивного подхода к вводу-выводу	301
Позволит ли ФП добиться большего успеха	302
Ввод-вывод и использование его результата	303
Императивный ввод-вывод	304
Вычисления как значения IO	305
Значения IO	306
Значения IO в реальной жизни	307
Удаляем загрязнения	308
Использование значений, полученных из двух операций ввода-вывода	309
Объединение двух значений IO в одно	310
Практика создания и объединения значений IO	311
Разделение задач при работе только со значениями	312
Тип IO — вирусный	313

Кофе-брейк: работа со значениями.....	314
Объяснение для кофе-брейка: работа со значениями.....	315
На пути к функциональному вводу-выводу.....	316
Как быть со сбоями ввода-вывода.....	317
Программа, описываемая значением IO, может завершиться неудачей!.....	318
Помните orElse?	319
Отложенные и немедленные вычисления.....	320
Реализация стратегий восстановления с использованием IO.orElse.....	321
Реализация запасных вариантов с использованием orElse и pure.....	322
Практика восстановления после сбоев в значениях IO.....	323
Где должны обрабатываться потенциальные сбои.....	324
На пути к функциональному вводу-выводу с обработкой сбоев.....	325
Чистые функции не лгут даже в небезопасном мире!.....	326
Функциональная архитектура.....	327
Использование IO для сохранения данных.....	328
Кофе-брейк: использование IO для сохранения данных.....	331
Объяснение для кофе-брейка: использование IO для сохранения данных.....	332
Интерпретация всего как значений.....	333
Интерпретация повторных попыток как значений.....	334
Интерпретация неизвестного количества вызовов API как значения.....	336
Практика восприятия функциональных сигнатур.....	338
Резюме.....	340
9 Потоки данных как значения.....	342
Бесконечность не предел.....	343
Работа с неизвестным количеством значений.....	344
Работа с внешними нечистыми вызовами API (снова).....	345
Функциональный подход к проектированию.....	346
Неизменяемые ассоциативные массивы.....	347
Практика неизменяемых ассоциативных массивов.....	348
Сколько вызовов IO следует сделать.....	349
Проектирование снизу вверх.....	350
Расширенные операции со списком.....	351
Знакомство с кортежами.....	352
Упаковка и отбрасывание.....	353
Сопоставление с образцом для кортежей.....	354
Кофе-брейк: ассоциативные массивы и кортежи.....	355
Объяснение для кофе-брейка: ассоциативные массивы и кортежи.....	356
Функциональные пазлы.....	357

Следование за типами в восходящем проектировании.....	358
Прототипирование и тупики	359
Рекурсивные функции.....	360
Бесконечность и «ленивые» вычисления	361
Структура рекурсивной функции	362
Обработка отсутствия значения в будущем (с использованием рекурсии)	363
Полезность бесконечных рекурсивных вызовов	364
Кофе-брейк: рекурсия и бесконечность	365
Объяснение для кофе-брейка: рекурсия и бесконечность.....	366
Создание различных программ IO с использованием рекурсии	367
Использование рекурсии для выполнения произвольного количества вызовов	368
Проблемы рекурсивной версии	369
Потоки данных.....	370
Потоки данных в императивных языках.....	371
Вычисление значений по требованию	372
Потоковая обработка, производители и потребители	373
Типы Stream и IO.....	374
Функциональный тип Stream	375
Потоки в ФП — это значения	376
Потоки — это рекурсивные значения.....	377
Примитивные операции и комбинаторы.....	378
Потоки значений IO	379
Бесконечные потоки значений IO.....	380
Запуск программы ради побочных эффектов	381
Практика работы с потоками.....	382
Использование преимуществ потоков	383
Бесконечный поток вызовов API.....	384
Обработка ошибок ввода-вывода в потоках.....	385
Разделение ответственности.....	386
Скользящие окна	387
Ожидание между вызовами ввода-вывода	390
Упаковка потоков	392
Преимущества потоковой обработки	393
Резюме.....	394
10 Параллельное программирование	396
Потоки выполнения повсюду	397
Декларативный параллелизм.....	398
Последовательные и параллельные вычисления.....	399

Кофе-брейк: последовательное мышление.....	400
Объяснение для кофе-брейка: последовательное мышление.....	401
Необходимость пакетной обработки.....	402
Пакетная реализация.....	403
Параллельный мир.....	404
Параллельное состояние.....	405
Императивный параллелизм.....	406
Атомарные ссылки.....	408
Знакомство с Ref.....	409
Обновление значений Ref.....	410
Использование значений Ref.....	411
Делаем все параллельно.....	412
parSequence в действии.....	413
Практика одновременно выполняющихся значений IO.....	416
Моделирование параллелизма.....	417
Программирование с использованием ссылок Ref и волокон.....	418
Значения IO, работающие бесконечно.....	420
Кофе-брейк: параллельное мышление.....	421
Объяснение для кофе-брейка: параллельное мышление.....	422
Необходимость асинхронности.....	423
Подготовка к асинхронному доступу.....	424
Проектирование функциональных асинхронных программ.....	425
Управление виртуальными потоками вручную.....	426
Программирование функциональных асинхронных программ.....	427
Резюме.....	428

Часть III. Прикладное функциональное ПРОГРАММИРОВАНИЕ

11 Разработка функциональных программ	430
Заставьте это работать, заставьте работать правильно, заставьте работать быстро.....	431
Моделирование с использованием неизменяемых значений.....	433
Моделирование предметной области и ФП.....	434
Моделирование доступа к данным.....	435
Мешок функций.....	436
Бизнес-логика как чистая функция.....	437
Отделение задачи доступа к данным.....	438

Интеграция с API с применением императивных библиотек и IO.....	439
Следуя проекту	442
Реализация действий ввода в виде значений IO	443
Отделение библиотеки ввода-вывода от других задач	445
Каррирование и инверсия управления.....	446
Функции как значения	447
Связываем все вместе.....	448
Мы заставили решение работать	449
Заставляем работать правильно.....	450
Утечки ресурсов.....	451
Управление ресурсами.....	452
Использование значения Resource	453
Мы заставили работать правильно.....	454
Кофе-брейк: заставьте работать быстро	455
Объяснение для кофе-брейка: заставьте работать быстро	456
Резюме.....	457
12 Тестирование функциональных программ	458
У вас есть тесты?.....	459
Тесты — это просто функции.....	460
Выбор функций для тестирования.....	461
Тестирование на примерах.....	462
Практика тестирования на примерах.....	463
Создание хороших примеров.....	464
Генерирование свойств.....	465
Тестирование на основе свойств.....	466
Тестирование путем предоставления свойств.....	467
Делегирование работы путем передачи функций	468
Интерпретация сбоя тестов на основе свойств	469
Ошибка в тесте или в программе?.....	470
Нестандартные генераторы	471
Тестирование более сложных случаев в удобочитаемой форме	473
Поиск и исправление ошибок в реализации.....	474
Кофе-брейк: тесты на основе свойств	475
Объяснение для кофе-брейка: тесты на основе свойств	476
Свойства и примеры	477
Охват требований	478
Тестирование требований с побочными эффектами.....	479
Определение правильного теста для работы	480
Тесты для проверки использования данных.....	481

Практика имитации внешних сервисов с использованием IO	483
Тестирование и дизайн	484
Тесты для проверки интеграции с сервисами	485
Локальные серверы как значения Resource в интеграционных тестах	486
Написание изолированных интеграционных тестов	487
Интеграция с сервисом — единая ответственность	488
Кофе-брейк: написание интеграционных тестов	489
Объяснение для кофе-брейка: написание интеграционных тестов	490
Интеграционные тесты выполняются дольше	491
Интеграционные тесты на основе свойств	492
Выбор правильного подхода к тестированию	493
Разработка через тестирование	494
Написание теста для несуществующей функции	495
Красный, зеленый, рефакторинг	496
Делаем тесты зелеными	497
Добавление красных тестов	498
Последняя итерация TDD	499
Резюме	500
Последний танец	501

Приложение А. Памятка по Scala

Определение значения	502
Определение функции	502
Вызов функции	502
Создание неизменяемых коллекций	502
Передача функции по имени	502
Передача анонимной функции	502
Передача анонимной функции с двумя параметрами	502
Определение функций с несколькими списками параметров (каррирование)	503
Объект Math	503
Определение case-класса (типа-произведения) и создание его значения	503
Точечный синтаксис для доступа к значениям в case-классе	503
Синтаксис определения анонимных функций с символом подчеркивания	503
Отсутствующая реализация: ???	503
Интерполяция строк	503
Передача многострочной функции	503
Автоматическое определение типов и пустые списки	503
Автоматическое определение типа и форсирование типа	504
Определение for-выражения	504
Объекты как модули и объект как мешки типов и функций	504

484	Определение непрозрачного типа (newtype).....	504
484	Импорт всего из объекта с использованием синтаксиса подчеркивания.....	504
484	Создание и использование значения непрозрачного типа.....	504
484	Определение перечислений (типов-сумм).....	505
484	Сопоставление с образцом.....	505
484	Именованые параметры в конструкторах и функциях классов.....	505
484	Использование интерфейсов trait для определения пакетов функций.....	505
484	Создание экземпляров интерфейсов trait (пакетов функций).....	505
484	Значение Unit в Scala.....	505
484	Создание неизменяемого типа Map.....	506
484	Передача функций, соответствующих образцу.....	506
484	Игнорирование значения с помощью символа подчеркивания.....	506
484	Протяженность интервалов времени и большие числа.....	506

Приложение Б. Жемчужины функционального программирования..... 507