

*Посвящается моим родителям, Хильдегард и Мартину.
Люблю вас всегда!*

Оглавление

1 ■ Вступление	22
2 ■ Античное хранение секретов на современных машинах	36
3 ■ Каждая строка считается	70
4 ■ Утилита нумерации строк	89
5 ■ Слова и графы	120
6 ■ Решение игры в цепочки слов	151
7 ■ Работа с CSV-файлами	169
8 ■ Программа для работы с CSV-файлами	194
9 ■ QuickCheck и случайные тесты	219
10 ■ Цифровая музыкальная шкатулка	261
11 ■ Программирование музыкальных композиций	295
12 ■ Разбор пиксельных данных	315
13 ■ Параллельная обработка изображений	364
14 ■ Файлы и исключения	401
15 ■ Применение трансформеров для синхронизации	419
16 ■ JSON и SQL	447
17 ■ Построение API с применением Servant	466

Содержание

	<i>Предисловие</i>	14
	<i>Благодарности</i>	15
	<i>Об этой книге</i>	17
	<i>Об авторе</i>	20
	<i>Об иллюстрации на обложке</i>	21
1	<i>Вступление</i>	22
	1.1 Что такое Haskell?	23
	1.1.1 Абстракции и теория	23
	1.1.2 Островок безопасности	24
	1.2 Путь чистого функционального программирования	26
	1.2.1 Декларативный рецепт	26
	1.2.2 От торта к программам	28
	1.2.3 Это все для простоты	29
	1.3 Использование абстракции	31
	1.3.1 Преимущества	31
	1.3.2 Недостатки	33
	1.4 Чему мы научились	33
	Резюме	34
2	<i>Античное хранение секретов на современных машинах</i>	36
	2.1 Введение в Haskell	37
	2.1.1 Шифр Цезаря	38
	2.1.2 Новый проект	39
	2.1.3 Первый модуль	42
	2.2 Типичные типы и фантастические функции	44
	2.2.1 Типы на атомарном уровне	45
	2.2.2 Списки и кортежи	46

2.2.3	Типы функций.....	48
2.2.4	Добавление типов к математическим операциям	49
2.3	На помощь приходят алфавиты	50
2.3.1	Синонимы типов делают код понятнее.....	51
2.3.2	Виды букв.....	54
2.3.3	Логические комбинации.....	55
2.4	Вращение колеса.....	57
2.4.1	Нахождение индекса элемента	57
2.4.2	Нахождение элемента по индексу.....	60
2.4.3	Охрана потока управления.....	62
2.5	Преобразование строки.....	64
2.5.1	Отображение высшего порядка.....	64
2.5.2	Параметризация типов	65
2.5.3	Завершенный шифр.....	67
	Резюме	69

3	Каждая строка считается.....	70
3.1	Общение с внешним миром	72
3.1.1	Простые действия ввода-вывода	72
3.1.2	Моделирование цикла	74
3.1.3	Выход из рекурсивного действия.....	75
3.2	Чистые функции внутри действий.....	77
3.2.1	Чтение и модификация пользовательских данных	78
3.2.2	Поток данных между чистым и нечистым кодами	79
3.3	Чтение из окружения	80
3.3.1	Разбор аргументов командной строки	81
3.3.2	Кодирование ошибок с помощью Maybe.....	82
3.4	Пример: чтение и печать аргументов командной строки	84
3.4.1	Ключевое слово let	86
3.4.2	Запуск программ с помощью stack.....	87
	Резюме	87

4	Утилита нумерации строк.....	89
4.1	Чтение файлов и преобразование их содержимого.....	90
4.1.1	Написание чистой библиотеки	91
4.1.2	Соккрытие вспомогательных аргументов	94
4.2	Параметризация поведения функций высшего порядка.....	96
4.2.1	Частичное применение функции.....	98
4.3	Алгебраические структуры данных как кодирование возможностей.....	101
4.3.1	Типы-суммы или помеченные объединения.....	102
4.3.2	Не повторяйся.....	104
4.3.3	Функция zip.....	105
4.3.4	Работа с отсутствующими значениями	108
4.3.5	Печать списка значений с помощью тарМ	109
4.4	От библиотеки к исполняемому файлу.....	110
4.4.1	Задание и разбор параметров командной строки.....	111
4.4.2	Итоги проекта	114
	Резюме	119

5	Слова и графы	120
5.1	Построение графа	121
5.1.1	Полиморфные типы	122
5.1.2	Знакомство с новым модулем	125
5.1.3	Класс типов <code>Eq</code> и ограничения типа	127
5.1.4	Функция <code>flip</code>	129
5.2	Инкапсуляция реализации	130
5.2.1	Добавление и удаление записей	132
5.2.2	Использование списков экспорта для сокрытия конструкторов	133
5.2.3	Класс типов <code>Show</code>	137
5.3	Использование и повторное использование кода	139
5.3.1	Квалифицированный импорт	139
5.3.2	Построение карт перестановок	141
5.3.3	Создание карты перестановок по словарю	144
5.4	Параметризация преобразований	145
5.4.1	Списковые включения	147
	Резюме	149
6	Решение игры в цепочки слов	151
6.1	Поиск в ширину	152
6.1.1	Общее описание алгоритма	152
6.1.2	Отслеживание состояния поиска	154
6.1.3	Нахождение решения с помощью обратного прохода	156
6.2	Область видимости переменной типа	157
6.2.1	Навешивание квантора всеобщности	157
6.2.2	Расширения языка	159
6.2.3	Использование переменных типа с лексической областью видимости	160
6.3	Повышение производительности с помощью хеш-таблиц	162
6.3.1	Анализ производительности с помощью профилирования	163
6.3.2	Добавление зависимостей в проект	165
6.3.3	Ленивое вычисление	167
	Резюме	168
7	Работа с CSV-файлами	169
7.1	Моделирование CSV-данных	170
7.1.1	Синтаксис записей	171
7.1.2	Кодирование ошибок типом <code>Either</code>	175
7.2	Умные конструкторы	176
7.2.1	Проверка свойства на этапе конструирования	176
7.2.2	Предоставление небезопасной альтернативы	178
7.2.3	Оператор доллар	180
7.3	Использование классов типов	181
7.3.1	Полугруппа и моноид	183
7.3.2	Класс типов <code>IsString</code>	186
7.4	Создание нового класса типов	188
7.4.1	Класс типов для вырезания структур данных	188
7.4.2	Резэкспорт модулей	192
	Резюме	193

8	Программа для работы с CSV-файлами	194
8.1	Разбор данных	195
8.1.1	Разбор числовых значений	197
8.2	Свертка структур данных	198
8.2.1	Концепция свертки	198
8.2.2	Структура данных для разбора	202
8.2.3	Класс типов Functor	203
8.2.4	Использование свертки при разборе	205
8.3	Печать CSV-таблицы	207
8.3.1	Операции с CSV-таблицами	210
8.4	Простой парсер командной строки	212
8.4.1	Поддержка флагов и более сложных аргументов	213
	Резюме	218

9	QuickCheck и случайные тесты	219
9.1	Как тестировать	220
9.1.1	Тестирование свойств	221
9.1.2	Генерирование случайных значений	222
9.1.3	Random и Uniform	224
9.1.4	Использование глобального генератора случайных значений	226
9.1.5	Простой тест свойств	228
9.1.6	Определение постулов для случайных значений	230
9.2	Рандомизированное тестирование	235
9.2.1	Преимущество ссылочной прозрачности	238
9.3	Каркас тестирования QuickCheck	239
9.3.1	Fun Property в QuickCheck	240
9.4	Генерирование случайных значений для тестирования	242
9.4.1	Генератор случайных значений Gen	243
9.4.2	Пример: AssocMap	246
9.4.3	Сокращение тестов	248
9.5	Практическое использование тестирования свойств	250
9.5.1	Уровень детализации и отчеты о покрытии	251
9.5.2	Модификация параметров теста	254
9.5.3	Построение комплектов тестов	254
9.5.4	Эффективность тестирования	258
	Резюме	260

10	Цифровая музыкальная шкатулка	261
10.1	Моделирование звука с помощью чисел	262
10.1.1	Зоопарк классов числовых типов	263
10.1.2	Создание периодических функций	267
10.2	Работа с бесконечными списками	273
10.2.1	Подъем, спад, поддержание и затухание	273
10.2.2	Построение бесконечных списков и работа с ними	275
10.3	Управление синтезом	278
10.3.1	Селекторы частичных полей	278
10.3.2	Функция как тип	281
10.4	Нотные модели	283
10.4.1	Класс типов для высоты звука	283

10.4.2	<i>Tun Ratio</i>	288
10.4.3	<i>Различные способы возведения в степень</i>	291
Резюме		293

11	Программирование музыкальных композиций	295
11.1	Полиморфные структуры данных с несколькими типами	296
11.1.1	Навешивание квантора существования	297
11.1.2	Использование типов с квантором существования	299
11.2	Интерпретация структур	300
11.2.1	Микширование сигналов	303
11.2.2	Полифонические группы	306
11.3	Реализация предметно-ориентированного языка	309
11.3.1	Упрощение синтаксиса	310
11.3.2	Пользовательские операторы для спископодобных структур данных	310
11.3.3	Объявления приоритета операторов	312
Резюме		314

12	Разбор пиксельных данных	315
12.1	Написание парсера	316
12.1.1	Переносимые изображения из прошлого	316
12.1.2	Как разбирать файл	319
12.1.3	Композиция эффектов	324
12.1.4	Выбор варианта	330
12.1.5	Введение в монады	334
12.1.6	Обсуждение монад	341
12.1.7	Как правильно ошибаться	343
12.2	Разбор по-крупному	346
12.2.1	Введение в библиотеку <i>Attoparsec</i>	348
12.2.2	Разбор изображений	350
12.2.3	Выбор формата	354
12.2.4	Собираем все парсеры вместе	359
Резюме		362

13	Параллельная обработка изображений	364
13.1	Предоставление информации о типе вызывающей стороне	365
13.1.1	Проблемы полиморфизма по возвращаемому значению	365
13.1.2	Обобщенные алгебраические типы данных	369
13.1.3	<i>Typ Vector</i>	371
13.1.4	Динамические типы с навешенными кванторами существования	374
13.2	Проверка разобранных данных	375
13.3	Обобщенный алгоритм преобразования изображения	382
13.3.1	Применение матриц преобразования к отображениям	384
13.3.2	Экспорт изображений в формате <i>PNG</i>	387
13.4	Использование распараллеливания для преобразования данных	389
13.4.1	Измерение времени	392

13.4.2	Как устроен параллелизм?	393
13.4.3	Искры и контексты выполнения <i>Haskell</i>	395
	Резюме	400

14	Файлы и исключения	401
14.1	Открытие и чтение файлов	402
14.1.1	<i>System.IO</i> и <i>Handle</i>	402
14.1.2	Буферизация и поиск	404
14.2	Чтение байтов из файла	406
14.2.1	Захват ресурсов и функция <i>bracket</i>	408
14.3	Работа с файловой системой и исключения	409
14.3.1	Модули <i>System.Directory</i> и <i>System.FilePath</i>	409
14.3.2	Перечисление файлов и каталогов	410
14.3.3	Основы исключений	411
14.4	Возбуждение и перехват исключений	413
14.4.1	Обработка ошибок	415
	Резюме	418

15	Применение трансформеров для синхронизации	419
15.1	Монадные трансформеры	420
15.1.1	Чтение окружения с помощью <i>ReaderT</i>	420
15.1.2	<i>StateT</i> и <i>WriterT</i>	423
15.1.3	Организация стека трансформеров с помощью <i>RWST</i>	429
15.2	Реализация приложения	430
15.3	Организация интерфейса с командной строкой	436
15.3.1	Разбор аргументов с помощью <i>optparse-applicative</i>	436
15.3.2	Перечисление пользовательских типов	440
15.3.3	Класс <i>Enum</i>	441
15.3.4	Интерфейс с командной строкой для <i>App</i>	443
	Резюме	446

16	JSON и SQL	447
16.1	Кодирование значений в формате JSON	448
16.1.1	<i>Aeson</i> и разбор JSON	448
16.1.2	Сериализация в формат JSON	452
16.2	Вывод классов типов с помощью <i>Generic</i>	454
16.3	Использование базы данных <i>SQLite</i>	456
16.3.1	Основы <i>sqlite-simple</i>	456
16.3.2	Функции <i>ToRow</i> и <i>FromRow</i>	457
16.3.3	Определение действий для доступа к базе данных	461
	Резюме	465

17	Построение API с применением <i>Servant</i>	466
17.1	Определение типобезопасного API	466
17.1.1	<i>Servant</i> и типизированные API	467
17.1.2	Фантомные типы	470
17.1.3	Реализация API	472
17.2	Запуск приложения	477

17.2.1	WAI-приложение	478
17.2.2	Средний уровень приложения.....	482
17.2.3	Порождение HTML-кода по данным.....	484
17.3	Вывод клиента.....	490
17.3.1	Использование <i>servant-client</i>	490
17.3.2	Определение команд.....	491
17.3.3	Реализация клиента	494
	Резюме	497
	<i>Приложение А. Комплект инструментов Haskell</i>	498
	<i>Приложение В. Ленивое вычисление</i>	502
	<i>Предметный указатель</i>	508

Предисловие

Haskell – язык, который при взгляде извне может показаться окутанным тайной, изобилующим научной терминологией и непонятными концепциями. Кое-что из этого правда, но я абсолютно уверен, что решить задачу на этом языке можно, не обладая степенью PhD по математике. Все дело в правильной подводке. На Haskell можно очень быстро начать писать реальные программы, и книга именно об этом.

Открою вам маленький секрет: моя первая попытка изучить Haskell окончилась весьма плачевно. Никакой другой язык не казался мне таким странным и сбивающим с толку (за исключением разве что Prolog и Uiua). Прошло много лет, прежде чем я снова вернулся к Haskell, и счастье мое не знало пределов!

А изменилась только подводка. В тот момент у меня за плечами уже было несколько лет опыта работы с языками функционального программирования, и я был хорошо знаком с академическим жаргоном, неотъемлемым от рассказа о Haskell. Прежде всего я понимал, как его применять, а ведь язык обретает смысл, только если осмысленно применять его к решаемой задаче, – поэтому я считаю этот аспект критически важным для изучения языка. И в этой книге я хочу акцентировать на нем внимание.

Но зачем вообще изучать Haskell? Что мы от этого получим? Помимо права гордиться проделанной работой, мы получаем совершенно новый способ структурирования программ и обдумывания задач. Haskell – чистый функциональный язык, он учит нас минимизировать состояние, в результате чего программу становится принципиально проще понять. Кроме того, умение применять Haskell к реальным задачам – еще один инструмент в вашем багаже, благодаря которому вы совершенствуетесь как программист. Да и другие функциональные языки после Haskell обычно изучать гораздо проще. Неважно, хотите ли вы изучить Haskell, чтобы более корректно писать программы, из чистого любопытства или чтобы расширить горизонты, эта книга послужит вам путеводителем по глубокому и полному чудес миру Haskell.

Благодарности

Никакая книга не возникает из ничего и не пишется одиноким автором за один вечер (о, как бы я этого желал!) – и эта книга не исключение. Она никогда не появилась бы на свет без неустанной работы многих людей и без неоценимой помощи еще гораздо большего их числа.

Прежде всего я хочу поблагодарить тех, кто всего ближе ко мне в жизни: родителей за их непрерывную поддержку на протяжении всей моей жизни. Без вас я не стал бы тем, кто я есть, и за это я вам вечно благодарен. Кроме того, хочу сказать спасибо Али, всегда готового слушать мои тирады на темы программирования.

Далее я хочу выразить благодарность всему коллективу издательства Manning, а особенно моим редакторам, Патрику Барбу и Кэти Спосато, за слова ободрения и терпение, с которым они переносили меня и эту книгу. Отдельное спасибо Александру Вершилову, который настойчиво побуждал меня уделять внимание техническим аспектам Haskell. Александр 15 лет работал программистом, имеет степень PhD по физике, а сейчас возглавляет команду инженеров, работающих над образовательным проектом на Haskell. Большое спасибо Ивану Мартиновичу за помощь в технических вопросах процесса написания книги. Отдельная благодарность техническому редактору, Тимо фон Хольцу. Без твоих подсказок эта книга никогда не была бы доведена до конца.

Спасибо также бесчисленным рецензентам: Алексу Мэйсону, Алексею Выскубову, Энди Дженнингс, Анкиту Митталу, Антону Ричу, Кэмерону Пресли, Клиффорду Тэрберу, Давиду Паку, Дамиану Эстебану, Дэну Шейху, Эрнесто Босси, Эрнесто Гарсиа, Франсу Ойлинки, Гаутаме Садависаму, Грегорио Пиччоли, Гуннару Альбергу, Джеку Келли, Джоэлю Холмсу, Джуральдиньо Шикину, Каю Гэллиену, Кате Паткин, Кристофу Семьену, Киту Л. Манноку, Кэвину Чунгу, Магнусу Тэрнингу, Мансуру Мухитдинову, Марко Пероне, Марку Эльсону, Майклу Д. Роучу, Полу Силистеану, Тэду Мейеру и Винисиосу Венцу. Ваши предложения помогли сделать книгу лучше.

Наконец, было бы непростительным упущением не поблагодарить чудесное сообщество Haskell – и не только за то, что оно вдохнуло жизнь в этот язык, но и за постоянные нововведения, добавляемые с неустанным рвением. Пишущие на Haskell – одни из самых умных, талантливых и доброжелательных людей, с которыми мне приходилось общаться, и надеюсь, что эта книга привлечет новых адептов в наше прекрасное сообщество.

Об этой книге

Эта книга ставит задачей научить языку Haskell с нуля до создания практически полезных приложений, а не нудных упражнений. Слишком часто к Haskell относятся как к чисто академическому языку. Долой это предвзятое мнение! Haskell практичен настолько, насколько это нужно.

Сместив акцент на применение языка, мы поможем читателю лучше понять фундаментально присущие ему сильные стороны, а также то, как его красивые абстракции становятся не просто мысленным экспериментом, а реальным инструментом для реальной работы.

На кого рассчитана эта книга

Это книга для программистов, которые уже знакомы хотя бы с одним языком программирования и хотят распробовать на вкус функциональное программирование на Haskell. Предварительное знакомство с функциональным программированием не предполагается. Книга не является полным руководством по языку, но может послужить точкой входа, которая позволит читателю приступить к написанию собственного кода.

Структура книги

Эта книга состоит из 15 глав, в которых описываются девять программных проектов. Понятия и конструкции языка объясняются по мере необходимости.

- Глава 1 содержит введение в Haskell и книгу в целом. В ней приводятся исторические сведения и рассматриваются основы функционального программирования, не вдаваясь в специфику кода на Haskell.
- В главе 2 начинается вступительный проект: шифр Цезаря, древний алгоритм шифрования.

- В главах 3 и 4 мы познакомимся с клоном программы `n1` для подсчета строк текстового файла; это первая в книге исполняемая программа.
- Предметом глав 5 и 6 служит обработка данных и проектирование алгоритмов на примере игры в цепочку слов.
- В главах 7 и 8 мы познакомимся с практическим инструментом для разбора, сериализации, поиска и модификации CSV-файлов.
- Глава 9 посвящена проверке свойств и содержит практические советы по написанию комплектов тестов для вашей программы.
- В главах 10 и 11 рассматривается программный синтезатор со встроенным предметно-зависимым языком для сочинения музыки.
- Главы 12 и 13 содержат введение в библиотеку для разбора данных изображений и предоставляют распараллеленный конвейер преобразования изображений.
- Главы 14 и 15 посвящены работе с исключениями в приложении для синхронизации файлов.
- В главах 16 и 17 читатель научится писать сервер REST API с использованием JSON, базы данных и автоматически генерируемого клиентского приложения.

Вообще говоря, каждая глава опирается на знания, полученные в предыдущих главах, поэтому читать их лучше подряд, ничего не пропуская. Даже если сам проект вас не интересует, необходимо освоить излагаемые в главе ключевые концепции, прежде чем переходить к следующей.

О примерах кода

В этой книге много примеров кода как в пронумерованных листингах, так и в виде небольших фрагментов прямо в тексте. В обоих случаях исходный код набран моноширинным шрифтом, чтобы отделить его от обычного текста. Иногда код набран **полужирным шрифтом**, чтобы показать, что он отличается от предыдущих шагов в этой же главе, например когда в старую строку кода добавляется новая возможность.

Во многих случаях оригинальный исходный код переформатирован; мы разбивали строки и изменяли отступы, сообразуясь с размером печатной страницы. Иногда этого оказывалось недостаточно, и тогда мы добавляли маркеры продолжения строк (**➞**). Кроме того, комментарии, присутствующие в исходном коде, часто удалялись из листингов, если код описан в тексте. Многие листинги сопровождаются аннотациями, подчеркивающими важные концепции.

Исполняемые фрагменты кода можно найти в онлайн-версии книги на сайте <https://livebook.manning.com/book/learn-haskell-by-example>. Полный код примеров можно скачать бесплатно с сайта издательства Manning по адресу <https://www.manning.com/books/learn-haskell-by-example>, также он доступен на Github по адресу <https://github.com/phagenlocher/learn-haskell-by-example>.

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 на нашем сайте www.dmkpress.com, зайдя на страницу книги и оставив комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com; при этом укажите название книги в теме письма.

Если вы являетесь экспертом в какой-либо области и 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.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы обеспечить высокое качество наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг, мы будем очень благодарны, если вы сообщите о ней главному редактору по адресу dmkpress@gmail.com. Сделав это, вы избавите других читателей от недопонимания и поможете нам улучшить последующие издания этой книги.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательства «ДМК Пресс» и Manning Publications очень серьезно относятся к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконной публикацией какой-либо из наших книг, пожалуйста, пришлите нам ссылку на интернет-ресурс, чтобы мы могли применить санкции.

Ссылку на подозрительные материалы можно прислать по адресу электронной почты dmkpress@gmail.com.

Мы высоко ценим любую помощь по защите наших авторов, благодаря которой мы можем предоставлять вам качественные материалы.

Об авторе



Филипп Хагенлохер (Philipp Hagenlocher) получил степень магистра информатики в Мюнхенском техническом университете, где приобрел солидные знания в области формальных методов и функционального программирования. Он работает программистом на Haskell на полной ставке и занимается распределенными системами, для которых абсолютная корректность программы – насущная необходимость. Филипп уже давно питает страсть к обучению других концепциям функционального программирования и на своем опыте знает, какие трудности могут возникнуть у читателя. Его серия роликов «Haskell for Imperative Programmers» на YouTube собрала миллион просмотров.

Конец ознакомительного фрагмента.

Приобрести книгу можно

в интернет-магазине

«Электронный универс»

e-Univers.ru