Душкин Р. В.

Практика работы на языке Haskell + CD



Москва, 2010

УДК 004.4
ББК 32.973.26-018.2
Д86

Душкин Р. В.
Д86 Практика работы на языке Haskell. – М.: ДМК Пресс, 2010. – 288 с., ил.
ISBN 978-5-94074-588-4

В книге рассматриваются прикладные аспекты работы на языке функционального программирования Haskell. Приводятся описания инструментальных средств пяти классов – трансляторов, интегрированных сред разработки, вспомогательных утилит, специализированных библиотек и справочно-архивных систем. Для каждого программного средства приводится краткое описание, его функциональность и примеры использования.

Книга станет хорошим подспорьем как для начинающих программистов, так и для профессионалов, использующих в своей практике функциональную парадигму программирования.

К книге прилагается компакт-диск, на котором имеются все описанные в книге инструменты для полноценной работы на языке Haskell.

УДК 004.4
ББК 32.973.26-018.2

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав.

Материал, изложенный в данной книге, многократно проверен. Но, поскольку вероятность технических ошибок все равно существует, издательство не может гарантировать абсолютную точность и правильность приводимых сведений. В связи с этим издательство не несет ответственности за возможные ошибки, связанные с использованием книги.

ISBN 978-5-94074-588-4

© Душкин Р. В., 2010
© Оформление ДМК Пресс, 2010

Оглавление

Новая парадигма **6**

Краткое содержание книги **8**

1 Трансляторы **11**

1.1 Интерпретатор HUGS 11

1.1.1 Паспорт программного средства 12

1.1.2 Общее описание 12

1.1.3 Функциональность и использование 13

1.1.4 Другие способы запуска интерпретатора HUGS 23

1.1.5 Часто задаваемые вопросы 24

1.1.6 Окончательные замечания 26

1.2 Компилятор GHC 26

1.2.1 Паспорт программного средства 27

1.2.2 Общее описание 27

1.2.3 Функциональность и использование 28

1.2.4 Советы о различных способах компиляции 43

1.2.5 Директивы компилятора 47

1.2.6 Кратко о расширениях языка Haskell 51

2 Интегрированная среда разработки **53**

2.1 Универсальная среда разработки Eclipse 54

2.1.1 Паспорт программного средства 54

2.1.2 Общее описание 55

2.1.3 Функциональность и использование 57

2.2 Надстройка EclipseFP 58
2.2.1 Паспорт программного средства 59
2.2.2 Общее описание 59
2.2.3 Функциональность и использование 61

3 Утилиты **70**
3.1 Препроцессор DrIFT 70
3.1.1 Паспорт программного средства 72
3.1.2 Общее описание 72
3.1.3 Функциональность и использование 75
3.1.4 Разработка собственных правил 78
3.2 Отладчик Buddha 81
3.2.1 Паспорт программного средства 83
3.2.2 Общее описание 83
3.2.3 Функциональность и использование 84
3.2.4 Команды отладчика 88
3.3 Оптимизатор HLint 91
3.3.1 Паспорт программного средства 92
3.3.2 Установка, запуск и некоторые особенности 92
3.3.3 Добавление новых правил 96
3.4 Система сборки документации Haddock 97
3.4.1 Паспорт программного средства 99
3.4.2 Общее описание 99
3.4.3 Функциональность и использование 105
3.5 Система контроля версий Darcs 111
3.5.1 Паспорт программного средства 112
3.5.2 Использование программного средства 113
3.5.3 Набор команд для управления репозиторием 115
3.6 Инсталляционная система Cabal 140
3.6.1 Паспорт программного средства 142
3.6.2 Общее описание 142
3.6.3 Функциональность и использование 143
3.6.4 Часто задаваемые вопросы 156

4 Библиотеки **160**
4.1 Библиотека комбинаторов синтаксического анализа Parsec 161
4.1.1 Паспорт программного средства 163
4.1.2 Примеры вариантов использования 163
4.1.3 Экспортируемые программные сущности 169
4.2 Библиотека комбинаторов для вывода информации PPrint 195
4.2.1 Паспорт программного средства 196
4.2.2 Общее описание 197
4.2.3 Экспортируемые программные сущности 199
4.3 Работа с базами данных HaskellDB 212
4.3.1 Паспорт программного средства 212
4.3.2 Общее описание 213
4.3.3 Экспортируемые программные сущности 217
4.4 Разработка графических интерфейсов пользователя wxHaskell . . . 233
4.4.1 Паспорт программного средства 234
4.4.2 Общее описание 234
4.4.3 Примеры использования 235
4.5 Организация сетевого взаимодействия HaskellNet 242
4.5.1 Паспорт программного средства 243
4.5.2 Экспортируемые программные сущности 243

5 Справочные системы и прочие инструменты **273**
5.1 Общий архив библиотек Hackage 273
5.1.1 Паспорт программного средства 275
5.2 Система поиска Hoogle 275
5.2.1 Паспорт программного средства 276
5.2.2 Функциональность и использование 276
5.3 Утилита HsColour 278
5.3.1 Паспорт программного средства 279
5.3.2 Использование утилиты 279
5.3.3 Конфигурирование цветовых палитр 280

Заключение **282**

Литература **284**

Новая парадигма

Внимательное изучение опыта написания и последующего распространения первых книг на русском языке о функциональном языке программирования Haskell [4, 5] подсказывает, что при работе над этими и другими подобными книгами упускается один очень важный момент. Более того, некоторые заинтересованные читатели в своих отзывах явно указывали на эту проблему, предлагая сменить сухой теоретический стиль изложения далёких от практического применения фундаментальных знаний на более приземлённое описание того, что и как можно сделать при помощи функционального программирования и функциональных языков. В книге [4] была произведена попытка сделать это, но, тем не менее, книга вышла несколько оторванной от реальности. Впрочем, такова, по всей видимости, судьба любого справочника.

Вместе с тем ещё в 1998 году один из апологетов функционального программирования Ф. Уодлер написал статью [22] о том, почему функциональные языки программирования не получают широкого распространения. В качестве одной из причин было указано крайне малое количество инструментальных средств, утилит, библиотек и прочих инструментов, позволяющих потенциальному разработчику сесть и начать работать на понравившемся ему функциональном языке программирования. Но вот на дворе XXI век, и к настоящему времени для языка Haskell выпущено огромное количество разнообразнейших инструментов, позволяющих создавать полноценные приложения для любой предметной области. Но почему же интерес к этому языку и его популярность всё ещё не могут быть сравнимы с популярностью таких языков, как Java или C++?

Один из возможных вариантов ответа заключается в том, что кроме замечательных инструментов необходимо создавать методическое обеспечение в должном объёме, которое позволит людям, не имеющим достаточного энтузиазма, на-

чать хотя бы интересоваться проблемой. Но уже упомянутые книги, предоставляющие фундаментальные знания, отпугивают новичков кажущейся на первый взгляд «суровой» математикой. А новичкам интересно найти в одном месте все инструменты, быстро установить их на свой персональный компьютер и написать свою первую программу «`Hello, world!`». Но в книгах и статьях новичков продолжают пугать быстрыми сортировками, оторванными от жизни факториалами и числами Фибоначчи.

Кому это нужно? Нет, несомненно, базовое образование и понимание глубинных основ науки о вычислениях крайне желательно для профессионального программиста. Но всё это может быть навёрстано позже, когда сформируется потребность понимания процессов, происходящих в недрах трансляторов функциональных языков. «Что такое ленивые вычисления?», «Почему ядро языка Haskell представляет собой типизированное λ-исчисление?», «Зачем программисту комбинаторная логика?», «Какие реальные применения в области программирования могут иметь монады?» и т. д. — все эти вопросы заинтересованный функциональным программированием специалист сможет задать себе и своим коллегам намного позже, когда проникнется сутью и духом функционального программирования. И тогда он найдёт всю необходимую теоретическую информацию, которой создано более чем достаточно.

Итак, в деле популяризации функционального программирования объявляется *новая парадигма*. Она заключается в предоставлении читателю конкретной прикладной информации о том, как применять функциональное программирование на практике для разработки программных систем любой сложности. Более того, в её рамках будут отсутствовать многомудрые повествования о морфизмах, теории категорий, β-редукции и прочих подобных вещах. И настоящая книга следует этой парадигме — она описывает существующий инструментарий, который уже сегодня можно взять и использовать на практике. Более того, на прилагаемом к книге CD записаны все программные средства, описываемые в книге, поэтому читатель сможет воспользоваться всеми новыми знаниями непосредственно в процессе чтения.

В добрый путь!

Краткое содержание книги

Данная книга разбита на пять глав, каждая из которых посвящена отдельному классу программных средств, предназначенных для работы с языком Haskell — разработки приложений на нём. Главы посвящены следующим классам инструментария:

1) трансляторам;

2) интегрированным средам разработки;

3) дополнительным утилитам;

4) важнейшим библиотекам языка;

5) справочным системам.

Выбор перечисленных классов программных средств обусловлен кибернетическим пониманием процесса преобразования идеи в разуме разработчика в создаваемое им программное средство, которое создаётся при помощи системы взаимосвязанных инструментов. Этот процесс схематично показан на рис. 1.

Каждая глава делится на разделы, каждый раздел отведён одному продукту одного из подклассов того класса программных средств, которому посвящена глава. Так, к примеру, в первой главе, посвящённой трансляторам, описывается по одному компилятору и интерпретатору. Так сделано сознательно — в книге представлен оптимальный (но не полный, поскольку охватить всё сразу невозможно), на взгляд автора, набор инструментов. В любом случае, в описании каждого конкретного программного средства в обязательном порядке указываются некоторые альтернативные решения, которые каждый читатель сможет найти в сети Интернет.

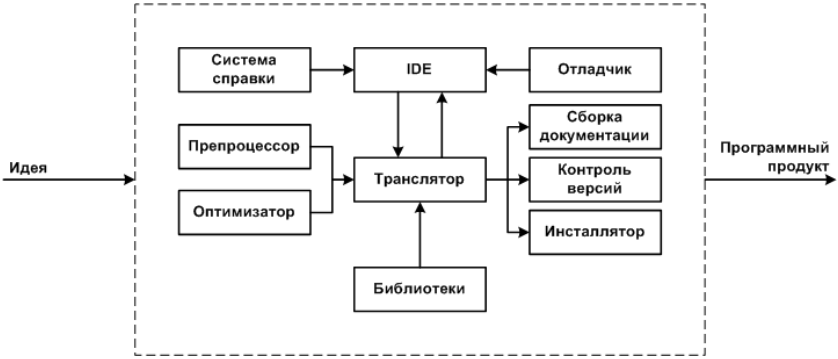


Рис. 1. Кибернетическая схема преобразования идеи в законченное программное средство

Каждое описываемое программное средство рассматривается в следующем ключе. В самом начале идёт так называемый «паспорт программного средства» — перечень значений ключевых характеристик. Этот перечень можно использовать для быстрого ознакомления с основным назначением программного средства, а также для сравнения их друг с другом. На стр. 12 показан пример такого паспорта для интерпретатора HUGS. Все таблицы паспортов унифицированы, другое дело, что в некоторых случаях сравнение программных средств не имеет смысла (пример — компилятор и библиотека для него).

Смысл граф паспорта в большинстве случаев понятен из их названия. В графе «Последняя версия» приводится номер выпуска соответствующего инструментального средства на весну — лето 2009 года. Графа «Язык» показывает язык графического интерфейса пользователя и (или) документации. Графа «Размер» содержит указание на размер инсталляционного пакета под операционную систему Windows. Наконец, в графе «Аналоги» представлены в большинстве случаев лишь некоторые аналоги, которые показались автору вполне достойными конкурентами рассматриваемому средству.

В целях единообразия представления исходных кодов здесь используется определённое форматирование текста, которое специальным образом выделяет структурные элементы функций и других программных сущностей. В отличие от предыдущих публикаций автора при вёрстке этой книги использовался пакет

LaTeX «listings» со специально включённой поддержкой языка Haskell, а потому структурные элементы оформляются в стиле этого пакета.

Таким образом, обычные идентификаторы из программ на языке Haskell при упоминании в тексте книги записываются при помощи моноширинного шрифта: `foldr`, `last`, `Functor`, `fst` и т. д. Ключевые слова, в свою очередь, выделяются полужирным начертанием обычного шрифта: **let**, **class**, **module**. Знаки операций и специальные символы при записи внутри текста ограничиваются круглыми скобками: (//), ($) и т. д., а сами скобки при необходимости выделения в тексте записываются в кавычках: «(», «)».

Отдельные определения программных сущностей оформляются программными блоками с акцентированием ключевых слов, причём сам текст исходных кодов записывается моноширинным шрифтом с небольшой разрядкой:

```
class Pretty a where
  pretty     :: a -> Doc
  prettyList :: [a] -> Doc

  prettyList = list . map pretty
```

К книге прикладывается компакт-диск, на который записаны все описываемые в книге инструменты. Само собой, те из них, которые находятся в активной стадии разработки, могут быть выпущены в новой версии, поэтому читателю рекомендуется внимательно смотреть в паспорте программного средства в графу «частота обновления», на основании записи в которой станет ясно, имеет ли смысл проверять наличие новой версии соответствующего инструмента на его официальном сайте. К имеющимся на компакт-диске инструментам по возможности прикладывается и документация на них.

Для описания представленных в книге программных продуктов, библиотек и инструментов использовались материалы и документация, прилагаемые к соответствующим программам, а также дополнительные статьи и книги, перечисленные в списке литературы.