

УДК 004.4
ББК 32.973.26-018.2
Д86

А

Д86 **Душкин, Роман Викторович.** Функциональное программирование на языке Haskell / Р. В. Душкин. — 2-е изд., эл. — 1 файл pdf : 609 с. — Москва : ДМК Пресс, 2023. — Систем. требования: Adobe Reader XI либо Adobe Digital Editions 4.5 ; экран 10". — Текст : электронный.
ISBN 978-5-89818-623-4

Данная книга является первым в России изданием, рассматривающая функциональное программирование в полном объеме, достаточном для понимания новичку и для использования книги в качестве справочного пособия теми, кто уже использует парадигму функционального программирования в своей практике. Изучение прикладных основ показано на примере языка Haskell, на сегодняшний день являющегося самым мощным и развитым инструментом функционального программирования.

Издание можно использовать и в качестве учебника по функциональному программированию, и в качестве самостоятельного учебного пособия по смежным дисциплинам, в первую очередь по комбинаторной логике и λ -исчислению.

Также книга будет интересна тем, кто всерьез занимается изучением новых компьютерных технологий, искусственного интеллекта и экспертных систем.

На сайте издательства dmk.pp.ru размещены материалы с транслятором Haskell, а также различными библиотеками к нему, дополнительными утилитами и рабочими примерами программ, рассмотренных в книге.

УДК 004.4
ББК 32.973.26-018.2

Электронное издание на основе печатного издания: Функциональное программирование на языке Haskell / Р. В. Душкин. — Москва : ДМК Пресс, 2016. — 608 с. — ISBN 978-5-97060-362-8. — Текст : непосредственный.

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.

В соответствии со ст. 1299 и 1301 ГК РФ при устранении ограничений, установленных техническими средствами защиты авторских прав, правообладатель вправе требовать от нарушителя возмещения убытков или выплаты компенсации.

ISBN 978-5-89818-623-4

© Душкин Р. В., 2007
© Издание, ДМК Пресс, 2016

А

Оглавление

Содержание	5
Введение	19
1 Основы функционального программирования	27
1.1 История функционального программирования	28
1.2 Основные свойства функциональных языков	44
1.3 Типовые задачи, решаемые методами функционального программирования	58
1.4 Конструирование функций	69
1.5 Доказательство свойств функций	86
2 Базовые принципы языка Haskell	99
2.1 Списки — основа функциональных языков	100
2.2 Функции как описания процессов вычисления	118
2.3 Типизация данных и функций	131
2.4 Элементы программирования	142
2.5 Модули и абстрактные типы данных	153
3 Классы и их экземпляры	164
3.1 Параметрический полиморфизм данных	165
3.2 Классы в языке Haskell как способ абстракции действительности .	170
3.3 Наследование и реализация	180
3.4 Стандартные классы языка Haskell	192
3.5 Сравнение с другими языками программирования	207

4	Монады — последовательное выполнение действий в функциональной парадигме	213
4.1	Монада как тип-контейнер	214
4.2	Последовательное выполнение действий	222
4.3	Операции ввода/вывода в языке Haskell	239
4.4	Стандартные монады языка Haskell	252
4.5	Разработка собственных монад	262
5	Комбинаторная логика и λ-исчисление	273
5.1	Основы комбинаторной логики	274
5.2	Абстракция функций как вычислительных процессов	288
5.3	λ -исчисление как теоретическая основа функционального программирования	298
5.4	Кодирование данных в λ -исчислении	307
5.5	Редукция и вычисления в функциональных языках	315
6	Трансляторы программ	331
6.1	Математическая лингвистика	331
6.2	Краткое введение в теорию построения трансляторов	346
6.3	Реализация трансляторов на языке Haskell	360
6.4	Библиотеки для создания трансляторов	372
6.5	Частичные вычисления, трансформация программ и суперкомпиляция	381
7	Функциональное программирование и искусственный интеллект	395
7.1	Основные задачи искусственного интеллекта	396
7.2	Нечеткая математика и функциональное программирование	407
7.3	Логический вывод на знаниях	429
7.4	Общение с компьютером на естественном языке	443
7.5	Перспективы функционального программирования	454
	Заключение	463
	Ответы на задачи для самостоятельного решения	465
	Решения задач из главы 1	465

Решения задач из главы 2	467
Решения задач из главы 3	474
Решения задач из главы 4	477
Решения задач из главы 5	483
Решения задач из главы 6	488
А Функциональные языки программирования и Интернет-ресурсы по функциональному программированию	496
Функциональные языки программирования	496
Русские Интернет-ресурсы	502
Иностранные Интернет-ресурсы	503
В Опции различных сред разработки на языке Haskell	506
Интегрированная среда разработки HUGS 98	506
Компилятор GHC	510
Компилятор NHC	523
Компилятор компиляторов Happy	528
С Описание стандартного модуля Prelude	531
Функции	531
Описание некоторых операторов языка Haskell	586
Д Краткий словарь терминов из области функционального программирования	589
Литература	599
Общая литература по функциональному программированию	599
Книги, руководства и статьи по языку Haskell	601
Комбинаторная логика и λ -исчисление	602
Математическая лингвистика и теория построения трансляторов	603
Искусственный интеллект	604

Содержание

Введение	19
Краткая биография автора	21
О пользовании книгой	22
Состав и структура представления информации	24
Благодарности	26
Контактная информация	26
 1 Основы функционального программирования	27
<i>В этой главе рассматриваются основополагающие принципы функционального программирования как отдельного направления в математической науке и технологии создания программного обеспечения. Приводится история развития функционального программирования, описываются предпосылки его развития. В главе рассматривается больше теоретического материала, нежели практического, поэтому пока изложение ведется без рассмотрения синтаксиса языка Haskell. Однако для приведения примеров используется именно этот язык наряду с математическими формулами. Изложение знаний о языке Haskell начинается с главы 2.</i>	
 1.1 История функционального программирования	28
<i>Краткая история развития теории функционального программирования в мире и в России. Разработка функциональных языков для подтверждения теоретических выкладок. Проблемы, с которыми столкнулись исследователи при разработке функциональных языков программирования. Современное состояние теории функционального программирования. Стандарт Haskell 98 как результат унификации и стандартизации процессов развития функционального программирования.</i>	
<i>Предпосылки создания функционального программирования</i>	34
<i>История языка Haskell</i>	38

Заключительные слова	42
1.2 Основные свойства функциональных языков	44
<i>Описание основных свойств функциональных языков программирования. Краткость и простота, строгая типизация, модульность, функциональные значения и объекты, чистота и отложенные вычисления. Понимание свойств функциональных языков на примере языка Haskell.</i>	
Краткость и простота	44
Строгая типизация	48
Модульность	50
Функции — это значения и объекты вычисления	51
Чистота (отсутствие побочных эффектов и детерминированность)	53
Отложенные (ленивые) вычисления	55
1.3 Типовые задачи, решаемые методами ФП	58
<i>Краткое описание семи типовых задач, которые решаются методами функционального программирования. Получение остаточной процедуры. Построение математического описания функций. Определение формальной семантики языка программирования. Описание динамических структур данных. Автоматическое построение «значительной» части программы по описанию структур данных, которые обрабатываются создаваемой программой. Доказательство наличия некоторого свойства программы. Эквивалентная трансформация программ.</i>	
Получение остаточной процедуры	60
Построение математического описания функций	62
Определение формальной семантики языка программирования	64
Описание динамических структур данных	64
Автоматическое построение функций по описанию структур данных	66
Доказательство наличия некоторого свойства программы	67
Эквивалентная трансформация программ	68
1.4 Конструирование функций	69
<i>Описание метода конструирования функций, предложенного Ч. Хоаром (синтаксически ориентированное конструирование). Метаязык для конструирования функций. Примеры определения типов и функций для обработки этих типов.</i>	
Декартово произведение	70
Размеченное объединение	71

Примеры определения типов данных	72
--	----

1.5 Доказательство свойств функций	86
--	----

Задача доказательства свойств функций. Описание процесса доказательства свойств функций в зависимости от типа области определения функций. Примеры доказательства свойств функций.

Область определения D — линейно-упорядоченное множество	88
---	----

Множество D определяется как индуктивный класс	89
--	----

Рассмотрение некоторых примеров доказательства свойств функций	91
--	----

Вопросы для самоконтроля	95
--------------------------------	----

Задачи для самостоятельного решения	97
---	----

2 Базовые принципы языка Haskell	99
---	-----------

Глава посвящена введению в основные положения языка Haskell, приводится описание синтаксиса для решения основных задач по созданию отдельных функций и законченных модулей. Рассматриваются базовые объекты «список» и «функция» для изучения в рамках функционального программирования, а также их реализация на языке Haskell.

2.1 Списки — основа функциональных языков	100
---	-----

Понятие списка в функциональном программировании. Списки как основная структура для работы с функциональными языками. Базисные операции для работы со списками. Списки и списочные структуры. Программная реализация списков в функциональных языках. Списки в языке Haskell. Генераторы списков и математические последовательности. Бесконечные списки и другие структуры данных. Кортёжи.

Проекция списков в язык Haskell	101
---------------------------------------	-----

Несколько слов о программной реализации	104
---	-----

Примеры	106
---------------	-----

Определители списков и математические последовательности	110
--	-----

Кортёжи	117
---------------	-----

2.2 Списки — основа функциональных языков	118
---	-----

Функция — основной объект изучения функционального программирования. Соглашения по именованию объектов в языке Haskell. Описание и определение функций на языке Haskell. Образцы и клозы. Передача параметров и возвращение значений функциями. Инфиксный способ записи функций. Функция как объект

для передачи в другие функции. Программа на языке *Haskell* — функция, описывающая процесс вычисления.

Соглашения по именованию	118
Общий вид определения функции	119
Образцы и клозы	119
Вызовы функций	125
Использование λ -исчисления	126
Инфиксный способ записи функций	127
Несколько слов о функциях высшего порядка	131

2.3 Списки — основа функциональных языков 131

Структуры и типы данных. Типы функций. Каррированные и некаррированные функции. Язык Haskell и его механизмы для организации каррированных и некаррированных функций. Описание типов функций на языке Haskell. Частичное применение. Ленивые (отложенные) вычисления на языке Haskell.

Структуры данных и их типы	132
Синонимы типов	136
Типы функций в функциональных языках	137
Полиморфные типы	140

2.4 Списки — основа функциональных языков 142

Отражающие выражения и конструкции. Локальные переменные для оптимизации кода на функциональном языке и на языке Haskell. Использование накапливающего параметра (аккумулятора) для оптимизации процесса вычислений. Принципы построения определений функций с накапливающим параметром. Головная и хвостовая рекурсии.

Охрана	143
Ветвление алгоритма	145
Локальные переменные	146
Двумерный синтаксис	149
Накапливающий параметр — аккумулятор	150
Принципы построения определений с накапливающим параметром	152

2.5 Списки — основа функциональных языков 153

Модули как способы структуризации и организации программ на языке Haskell. Импорт и экспорт данных при помощи модулей. Соккрытие данных. Абстрактные типы данных и интерфейсы. Иные аспекты использования модулей.

Абстрактные типы данных	156
Другие аспекты использования модулей	157
Литературный код	158

Вопросы для самоконтроля	160
Задачи для самостоятельного решения	161

3 Классы и их экземпляры 164

Эта глава посвящена рассмотрению симбиоза парадигм функционального и объектно-ориентированного программирования. Большинство современных функциональных языков поддерживают механизмы и методы, разработанные в рамках объектно-ориентированного программирования, в том числе и такие базовые концепты, как «наследование», «инкапсуляция» и «полиморфизм». Не обошел своим вниманием этот аспект и язык Haskell, в котором имеются достаточные средства для программирования в объектно-ориентированном стиле.

3.1 Параметрический полиморфизм данных 165

Понятие класса и его реализации в языке Haskell. Чистый (параметрический) полиморфизм на языке Haskell. Примеры параметрического полиморфизма в императивных и функциональных языках, а также в языке Haskell.

3.2 Классы в языке Haskell как способ абстракции действительности 170

Расширенное описание понятия класса в языке Haskell. Класс как высшая абстракция данных и методов для их обработки. Методы класса — шаблоны функций для реализации обработки данных. Минимальное описание методов класса и связь методов.

Модель типизации Хиндли-Милнера 171

Определение классов 176

3.3 Наследование и реализация 180

Наследование классов и наследование методов. Экземпляры классов — реализация интерфейсов, предоставляемых реализуемым классом. Реализация методов для обработки данных. Класс — шаблон типа, реализация класса — тип данных.

Наследование 180

Реализация 182

Реализация для существующих типов 186

Сорта типов 187

Дополнительные возможности при определении типов данных 189

3.4 Стандартные классы языка Haskell192

Краткое описание всех стандартных классов, разработанных для облегчения программирования на языке Haskell. Дерево наследования стандартных классов. Типичные способы использования стандартных классов языка Haskell. Реализация стандартных классов — типы в языке Haskell.

<i>Класс Bounded</i>	192
<i>Класс Enum</i>	193
<i>Класс Eq</i>	195
<i>Класс Floating</i>	195
<i>Класс Fractional</i>	196
<i>Класс Functor</i>	197
<i>Класс Integral</i>	198
<i>Класс Ix</i>	199
<i>Класс Monad</i>	199
<i>Класс Num</i>	200
<i>Класс Ord</i>	201
<i>Класс Read</i>	202
<i>Класс Real</i>	203
<i>Класс RealFloat</i>	203
<i>Класс RealFrac</i>	204
<i>Класс Show</i>	206

3.5 Сравнение с другими языками программирования207

Более или менее полное сравнение понятий «класс» и «реализация класса» в языке Haskell с объектно-ориентированными языками программирования (на примере языков C++ и Java, а также некоторых других языков). Глобальные отличия понятия «класс» в функциональных и объектно-ориентированных языках.

<i>Окончательные замечания</i>	209
--------------------------------------	-----

<i>Вопросы для самоконтроля</i>	210
---------------------------------------	-----

<i>Задачи для самостоятельного решения</i>	211
--	-----

4 Монады — последовательное выполнение действий в функциональной парадигме 213

Глава описывает такое незаурядное понятие, введенное в функциональной парадигме программирования, как «монада». Монады, основанные на математической теории категорий, позволяют внедрить в функциональный подход опре-

деленные структуры для выполнения императивных действий, как, например, операции ввода/вывода, обработка исключений, хранение состояний в процессе вычислений, и многие другие действия, связанные с побочными эффектами. Вместе с тем монады позволяют обернуть императивные действия в функциональную оболочку, спрятав все от «императивного мира» внутри монады.

4.1 Монада как тип-контейнер 214

Описание монады как типа-контейнера. Использование монад в функциональных языках. Свойства монадических типов. Операции связывания с передачей и без передачи результата выполнения операции на предыдущем шаге. Правила построения монад.

Определение понятия «монада» 215

Нотация `do` 218

Правила построения монад 220

4.2 Последовательное выполнение действий 222

Действие — элемент функциональной парадигмы. Императивный код внутри функционального. Выполнение действий и возвращение результата. Сокращенный способ записи последовательности действий. Списки действий. Программирование при помощи действий.

Класс `Computations` 224

Монада `State` 227

4.3 Операции ввода/вывода в языке Haskell 239

Более или менее полное описание базовых операций ввода/вывода в языке Haskell. Монада `IO`. Обработка исключений. Использование файлов, каналов и обработчиков. Нарушение теоретических принципов функционального программирования в монаде `IO`.

Действия ввода/вывода 240

Программирование при помощи действий 244

Обработка исключений 245

Файлы и потоки 247

Окончательные замечания 251

4.4 Стандартные монады языка Haskell 252

Подробное описание монадических типов в стандартной библиотеке языка Haskell. Назначение и применимость монадических типов. Примеры использования стандартных монадических типов (кроме списков и монады `IO`). Модуль `M Monad`. Монады `Glasgow Haskell Compiler`.

Модуль <i>Monad</i>	253
Стандартные монады	257
4.5 Разработка собственных монад	262
<i>Критерии возможности и необходимости разработки собственного монадического типа. Комбинирование монадических вычислений. Преобразователи монад. Примеры преобразования.</i>	
Комбинирование монадических вычислений	263
Преобразователи монад	264
Пример с преобразователем <i>StateT</i>	267
Окончательные замечания	268
Вопросы для самоконтроля	269
Задачи для самостоятельного решения	270
5 Комбинаторная логика и λ -исчисление	273
<i>В главе рассматриваются основополагающие теоретические формализмы, которые стояли у истоков функционального программирования, а именно комбинаторная логика и λ-исчисление, разработанные в качестве расширений формальной логики и теории множеств в начале XX в. Приводятся самые основы этих направлений дискретной математики, достаточные для понимания сути того, что в свое время стояло за парадигмой функционального программирования.</i>	
5.1 Основы комбинаторной логики	274
<i>Введение в комбинаторную логику. Поверхностное описание принципов комбинаторной логики. Комбинаторы и вычисления при помощи комбинаторов. Базисы в комбинаторной логике. Использование базисных комбинаторов для выражения любых вычислительных процессов. Числа и иные математические объекты в виде комбинаторов.</i>	
Базовые комбинаторы	274
Комбинатор неподвижной точки	281
Нумералы и арифметические операции	283
Заключительные слова	286
5.2 Абстракция функций как вычислительных процессов	288
<i>Функция — объект математического исследования. Вычислительный процесс — функция. Описание функций как λ-выражений. Свободные и связанные идентификаторы.</i>	

фикаторы. Применение (апликация) значений к λ -выражениям. Непрерывная точка функций и теорема о непрерывной точке.

«Наивное» определение λ -исчисления	289
Связь с комбинаторной логикой	292
Редукция	293
Тезис Черча-Тьюринга	294

5.3 λ -исчисление как теоретическая основа функционального программирования

298

Предположение о том, что любая функция представима в виде λ -выражения. Интенционал и экстенционал функций. Формальная система. Построение формальной системы для обоснования теории функционального программирования. Правила вывода. Соответствия между вычислениями функциональных программ и редукцией λ -выражений.

Построение формальной системы	299
Функциональное программирование как формальная система	303
Теорема Черча-Россера	305

5.4 Кодирование данных в λ -исчислении

307

Механизм кодирования данных в λ -исчислении. λ -исчисление — достаточный формализм для представления значений истинности, упорядоченных пар, натуральных чисел, списков, а также базовых операций над этими объектами.

Булевские значения	308
Упорядоченные пары	309
Натуральные числа	311
Списки	314

5.5 Редукция и вычисления в функциональных языках

315

Понятие редукции. Частичные вычисления с точки зрения редукции λ -выражений. Различные редукционные стратегии и их свойства.

Стратегия редукции и стратегия вычислений	315
Ленивая редукция	321

Вопросы для самоконтроля

327

Задачи для самостоятельного решения

329

6 Трансляторы программ

331

В главе 6 представлено краткое описание теории построения трансляторов для интерпретации и компиляции языков программирования. Теория рассматривается на примерах, написанных на языке Haskell. Кроме того, рассматриваются способы построения парсеров, а также готовые библиотеки для синтаксического анализа. Суперкомпиляция.

6.1 Математическая лингвистика 331

Краткое введение в математическую лингвистику. Обзор методов и принципов математической лингвистики. Классификация языков и грамматик. Конечные автоматы и контекстно-свободные языки. Грамматики типа $LL(k)$. Контекстно-зависимые грамматики.

Базовые понятия 332

Расширенная нотация Бэкуса — Наура 335

Классификация грамматик 337

Конечные лингвистические автоматы 338

Синтаксический анализ контекстно-свободных языков 343

6.2 Краткое введение в теорию построения трансляторов 346

Трансляторы: определения, типы и классификация, применимость для тех или иных типов языков и грамматик. Интерпретаторы и компиляторы. Компиляторы компиляторов. Методы построения трансляторов. Автоматическое построение транслятора по грамматике языка (для ограниченного множества языков). Понятие трансформационной грамматики.

Классификация трансляторов и их типовые структуры 347

Трансформационные грамматики 354

Автоматическое построение анализатора для отдельных типов языков 358

6.3 Реализация трансляторов на языке Haskell 360

Функциональные языки, как естественный инструмент реализации трансляторов. Синтаксические анализаторы. Методы написания трансляторов на языке Haskell. Примеры синтаксических анализаторов для различных форматов данных. Пример вычисления арифметических выражений, записанных в различной нотации.

Простейшие парсеры 361

Комбинаторы синтаксического анализа 363

Дополнительные комбинаторы синтаксического анализа 366

Анализ нотации Бэкуса — Наура 368

6.4	Библиотеки для создания трансляторов	372
	<i>Описание имеющихся библиотек для языка Haskell, предназначенных для создания трансляторов. Монадическая библиотека Parsec для самостоятельного создания трансляторов. Компилятор компиляторов Happy.</i>	
	Монадическая библиотека парсеров Parsec	372
	Компилятор компиляторов Happy	377
6.5	Частичные вычисления, трансформация программ и суперкомпиляция	381
	<i>Задача трансформации программ. Частичные вычисления, как инструмент для трансформации программ. Частичный вычислитель — инструмент для получения остаточного кода заданной функциональной программы. Интерпретатор, компилятор и компилятор компиляторов, их связь. Проекция Футамуры-Турчина. Суперкомпиляция.</i>	
	Частичные вычисления и трансляция программ	382
	Проекция Футамуры — Турчина	385
	Трансформация программ	387
	Вопросы для самоконтроля	392
	Задачи для самостоятельного решения	394
7	ФП и искусственный интеллект	395
	<i>Заключительная глава книги рассматривает такую область человеческого знания, как искусственный интеллект, то есть методы решения слабоформализованных задач, для которых не существует алгоритмического решения либо такое решение слишком сложно. Такой интерес связан с тем, что именно парадигма функционального программирования нашла свое непосредственное применение в рамках искусственного интеллекта.</i>	
7.1	Основные задачи искусственного интеллекта	396
	<i>Историческая справка о развитии искусственного интеллекта, как области научного исследования. Введение в базовые понятия искусственного интеллекта. Место функционального программирования в искусственном интеллекте. Функциональное и логическое программирования. Задачи искусственного интеллекта, которые могут быть решены при помощи методов и средств функционального программирования.</i>	
	История развития искусственного интеллекта	398
	Различные подходы к построению систем искусственного интеллекта	402
	Место функционального программирования в искусственном интеллекте	405

7.2	Нечеткая математика и функциональное программирование	407
	<i>Небольшой экскурс в нечеткую математику. Функции принадлежности и лингвистические переменные. Базовые операции над функциями принадлежности. Кусочно-линейные функции принадлежности. Операции сравнения, арифметические и логические операции над кусочно-линейными функциями принадлежности. Использование языка Haskell для реализации методов обработки кусочно-линейных функций принадлежности.</i>	
	Базовые концепты нечеткой логики	407
	От нечеткой логики к нечеткой математике	409
	Функции принадлежности как способ описания нечетких значений	411
	Нечеткие и лингвистические переменные	417
	Операции над функциями принадлежности	418
	Пример модуля для обработки кусочно-линейных функций принадлежности	423
7.3	Логический вывод на знаниях	429
	<i>Знания и данные. Модели представления знаний. Понятие логического вывода на знаниях. Стратегии вывода на знаниях. Машины вывода. Эволюция машинного вывода на знаниях. Интерпретаторы функциональных языков как естественные машины вывода. Язык Haskell и его возможности в логическом выводе на знаниях. Универсальный вывод на продукционной модели знания.</i>	
	Знания и данные	430
	Вывод на знаниях	434
	Прямой нечеткий вывод	437
	Обратный нечеткий вывод	439
	Некоторые окончательные замечания о машинном выводе	442
7.4	Общение с компьютером на естественном языке	443
	<i>Принципы общения с компьютерными системами на естественном языке. Ограниченный естественный язык, деловая проза. Трансляция фраз на естественном языке во внутренний язык представления смысла. Понимание текстов на естественном языке. Использование методов функционального программирования.</i>	
	Обобщенная схема интеллектуальных диалоговых систем	444
	Схема анализа входного текста	447
	Некоторые окончательные замечания	453
7.5	Перспективы функционального программирования	454

Описание видения будущего функционального программирования, функциональных языков и языка Haskell. Значение функциональной парадигмы для технологии программирования вообще.

Вопросы для самоконтроля 460

Задачи для самостоятельного решения 461

Заключение 463

Ответы на задачи для самостоятельного решения 465

Решения задач из главы 1 465

Решения задач из главы 2 467

Решения задач из главы 3 474

Решения задач из главы 4 477

Решения задач из главы 5 483

Решения задач из главы 6 488

А Функциональные языки программирования и Интернет-ресурсы по функциональному программированию 496

Список наиболее известных и широко используемых функциональных языков программирования с краткой аннотацией. Список Интернет-ресурсов, посвященных функциональному программированию как в русском сегменте Интернета, так и во всем остальном мире.

Функциональные языки программирования 496

Русские Интернет-ресурсы 502

Иностранные Интернет-ресурсы 503

В Опции различных сред разработки на языке Haskell 506

Описание типичных настроек интегрированных сред разработки и компиляторов на примере продуктов HUGS 98 и GHC для полноценной работы и выполнения различных задач на языке Haskell. Список команд, ключей командной строки и внутренних директив интерпретатора HUGS 98. Список параметров командной строки для компилятора GHC. Другие функциональные средства разработки.

Интегрированная среда разработки HUGS 98 506

Компилятор GHC 510

Компилятор GHC 523

Компилятор компиляторов HARRY 528

С	Описание стандартного модуля Prelude	531
	<i>Список функций из стандартного модуля языка Haskell Prelude с более или менее подробным описанием.</i>	
Функции		531
Описание некоторых операторов языка Haskell		586
Д	Краткий словарь терминов из области функционального программирования	589
	<i>Краткий словарь терминов, имеющих отношение к функциональному программированию. для каждого термина даются ссылки на страницы, где он описан, а также переводы на некоторые иностранные языки.</i>	
Литература		599
Общая литература по функциональному программированию		599
Книги, руководства и статьи по языку Haskell		601
Комбинаторная логика и λ -исчисление		602
Математическая лингвистика и теория построения трансляторов		603
Искусственный интеллект		604