

# Введение

Проблемно-ориентированное программирование является подходом в технологиях программирования, при использовании которого для решения определённой задачи (проблемы) разрабатывается собственный язык программирования, «заточенный» специально под изучаемую проблему. Естественно, что такая проблема должна быть достаточно сложна, чтобы создание специализированного формального языка для её решения было целесообразным. В настоящее время проблемно-ориентированное программирование всё больше и больше внедряется в область информационных технологий, когда для общих или специализированных систем (офисные приложения, САПР и расчётные системы — CAD/CAM/CAE, и т. д.) создаются специальные языки исполнения сценариев, встроенные в эти системы. Используя такие языки исполнения сценариев продвинутые пользователи могут расширять функциональность систем, имея доступ к их внутренней функциональности.

Теория синтаксического анализа позволяет создавать специальные инструменты (так называемые «синтаксические анализаторы»), которые используются для анализа исходных текстов на специализированных языках программирования и генерации либо исполняемого кода (компиляция), либо каких-либо непосредственно ответных действий системы (интерпретация). Синтаксический анализ является подразделом формальной математической лингвистики и используется в ряде областей науки и техники, начиная от анализа простейших формальных языков до понимания текстов на естественных языках, автоматического перевода, генерации текстов на естественном языке и т. д.

Таким образом, синтаксический анализ и проблемно-ориентированное программирование являются актуальными инструментальными навыками, применимыми во многих областях науки и техники. Изучение этих подходов позволит разрабатывать современное программное обеспечение, отвеча-

ющее запросам конкурентной среды и требованиям пользователей. Кроме того, развитие технической базы вычислительной техники, повышение её быстродействия и увеличение памяти позволит на практике использовать все методы математической лингвистики без потери производительности создаваемых систем.

Предлагаемая книга «Проблемно-ориентированное программирование и синтаксический анализ на языке Haskell» является первым изданием на русском языке, рассматривающим проблемно-ориентированное программирование и прикладные аспекты математической лингвистики. В книге описываются теоретические основы и практические приёмы использования методов синтаксического анализа формальных языков, дано введение в теорию синтаксического анализа, кратко описываются существующие методы и подходы к разбору формальных и естественных языков. Особое внимание уделено парадигме проблемно-ориентированного программирования в части реализации собственных проблемно-ориентированных языков для решения конкретных задач. Все теоретические положения, описываемые в книге, снабжены примерами на функциональном языке программирования Haskell. Также приводится описание библиотек и утилит, предназначенных для создания трансляторов.

Книга рассчитана на всех, кто интересуется современными тенденциями в области развития компьютерной науки и технологии, а также смежными областями — математической лингвистикой (теорией формальных языков и грамматик) и искусственным интеллектом. Она может служить дополнительным учебным пособием для студентов и аспирантов, обучающихся по специальности «прикладная математика», «информатика» и схожих специальностей. Также книга будет полезна в качестве справочного пособия по синтаксическому анализу при помощи формальных функций для использования в научных и инженерных областях деятельности.

Для чтения книги необходимо обладать начальными познаниями в дискретной математике, достаточными для понимания принципов теории вычислимости,  $\lambda$ -исчисления и комбинаторной логики, а также иметь представление о методах разработки алгоритмов и алгоритмическом решении задач (теория алгоритмов). Кроме того, желательно знание математической лингвистики на уровне понимания принципов морфологического, синтаксического и семантического анализа. Знание теории формальных грамматик и методов их разбора, а также знание теории построения трансляторов желательно, но не необходимо, поскольку в книге даются все необходимые определения и формализмы. В книге приводятся краткие отступления

---

в математическую теорию, поэтому она позволит обновить в памяти знания по перечисленным темам. Само собой, желательно знание синтаксиса языка Haskell и понимание основных идиом функционального программирования (это необходимо для полноценного понимания примеров, данных в книге).

Необходимость написания книги вызвана тем, что на текущий момент нет актуальных справочно-методических изданий на русском языке, которые в полной мере описывали бы современные подходы и методы синтаксического анализа в частности и математической лингвистики в целом. Кроме того, не существует книг, посвящённых проблемно-ориентированному программированию, поскольку данная тема является новой и пока ещё не рассматривается в широком аспекте. Наконец, существующие книги по математической лингвистике достаточно устарели (хотя до сих пор и являются классическими трудами), поэтому актуальность настоящего издания определяется новыми знаниями, изложенными в нём.

Отличительная особенность книги заключается в том, что все перечисленные вопросы рассматриваются в ней с точки зрения функционального программирования и языка Haskell. Парадигма функционального программирования самым естественным образом предназначена для решения задач синтаксического анализа и теории построения трансляторов. Более того, для языка Haskell имеются многочисленные специальные библиотеки и утилиты, которые позволяют автоматизировать процесс создания транслятора для языка, для которого построена формальная грамматика. Этот вопрос самым детальным образом раскрывается в настоящем издании.

Следует также отметить, что в последние годы языки функционального программирования являются, по сути, «тестовой лабораторией», в которой проходят обкатку многие прогрессивные идеи, реализуемые затем в самых популярных императивных языках (таких, как C++, C#, Java). В качестве примеров подобных успешных заимствований можно привести механизм generics в языке Java, инструментальное средство LINQ в языках семейства .Net, автоматический вывод типов выражений и методов в языке C#, анонимные методы и классы в языке Java. На сегодняшний день Haskell находится на переднем крае этой волны инноваций, и знакомство с этим языком будет полезным для всех, кто интересуется программированием или занимается им профессионально.

Наконец, необходимо сказать, что в теории проектирования языков для определённой проблемной области существует некоторая омонимия терминов. Термин «*проблемно-ориентированный язык*» используется для обозначения двух сходных, но разных понятий. В последнее время компа-

ния Microsoft продвигает парадигму проблемно-ориентированных языков проектирования взамен универсального языка UML. Настоящая книга не рассматривает этот аспект, ограничиваясь только изучением проблемно-ориентированных языков программирования.

## Краткая историческая справка

Интерес человека к собственному языку зародился, скорее всего, с возникновением разумной речи. Издревле люди пытались изучать свой язык, и языки соседних народов. С возникновением научного подхода изучение языка встало на методологическую основу, и ещё в V веке до н. э. древнегреческий философ Платон написал диалог «Кратил», в котором содержались некоторые представления о преобразовании идеи в текст. Это была первая европейская работа по науке, которую много позже назвали «языкознанием» (или «лингвистикой»). Однако древние учёные рассматривали языкознание лишь как вспомогательную науку для логики, при помощи которой последняя изучала способы формирования и выражения мысли.

Аристотель весьма серьёзно относился к языкознанию, посвятив ему множество своих сочинений. В его работах «Категории», «Об истолковании», «Топика» изложена цельная логико-языковая концепция. Аристотель первый из античных мыслителей подошёл к проблеме грамматической формы, развивал учение о частях речи как грамматически различающихся классах слов. Его концепция развивалась в европейской логике и грамматике средневековья, а логическая грамматика не потеряла значения до XX века, особенно в школьной практике. Например, некоторые грамматические термины в русской грамматике — кальки введённых Аристотелем терминов.

В другом направлении двигались древнеиндийские учёные. Так, к примеру, древнеиндийский грамматик Панини (V — IV века до н. э.), в отличие от Платона и Аристотеля, построивших общефилософские системы взглядов, рассматривает язык в самом себе и для себя, притом главным образом в формальном отношении, без системы семантики. Его нормативная грамматика «Аштадхьяи» исчерпывающе описывает фонетику, морфологию, словообразование и элементы синтаксиса древнеиндийского языка. Панини впервые и лингвистически вполне точно ввёл понятия корня, аффикса, основы слова и понятие порождения словоформ.

В дальнейшем наука о языке развивалась в совершенно различных направлениях, от генеалогической классификации языков, до структурной лингвистики, основанной на положениях теории систем. И, наконец, в 50-

ых годах XX века возникло новое направление в языкознании, получившее название «математическая лингвистика». Эта дисциплина разрабатывает формальный аппарат для описания строения естественных и искусственных языков. В математической лингвистике используются по преимуществу идеи и методы алгебры, теории алгоритмов и теории автоматов.

Математическое описание языка основано на восходящем к Ф. де Соссюру представлении о языке как механизме, функционирование которого проявляется в речевой деятельности его носителей. Её результатом являются «правильные тексты» — последовательности речевых единиц, подчиняющиеся определённым закономерностям, многие из которых допускают математическое описание. Изучение способов математического описания правильных текстов (в первую очередь предложений) составляет содержание одного из разделов математической лингвистики — теории способов описания синтаксической структуры.

Другой раздел математической лингвистики, занимающий в ней центральное место, — теория формальных грамматик, возникшая главным образом благодаря работам Н. Хомского. Она изучает способы описания закономерностей, которые характеризуют уже не отдельный текст, а всю совокупность правильных текстов того или иного языка. Эти закономерности описываются путём построения «формальной грамматики» — абстрактного «механизма», позволяющего с помощью единообразной процедуры получать правильные тексты данного языка вместе с описаниями их структуры. Наиболее широко используемый тип формальной грамматики — так называемая порождающая грамматика, или грамматика Хомского.

Наконец, с повсеместным внедрением компьютеров и вычислительных систем возникла потребность в новой науке, которая была названа «компьютерной лингвистикой». Это — уже направление в прикладной лингвистике, ориентированное на использование компьютерных инструментов — программ, компьютерных технологий организации и обработки данных — для моделирования функционирования языка в тех или иных условиях, ситуациях, проблемных сферах и т. д., а также вся сфера применения компьютерных моделей языка в лингвистике и смежных дисциплинах. Одним из направлений деятельности этого раздела науки о языке является теория о создании искусственных языков, предназначенных для «общения» с компьютерами (под «общением» здесь понимается широкий ряд процессов, начиная от простого программирования до организации сложных систем типа «человек — ЭВМ»).

Развитие компьютерной лингвистики и информатики привело к тому, что после создания огромного множества языков программирования общего плана исследователи пришли к выводу о том, что зачастую для решения определённых задач или классов задач более эффективно будет разрабатывать свой конкретный язык, «заточенный» непосредственно под решаемую задачу. Так родилась идея проблемно-ориентированного языка, DSL — от английского *Domain Specific Language*. В представленной книге проблемам изучения и построения DSL (а в дальнейшем это понятие будет обозначаться именно так для краткости и удобства) будет посвящён основной объём текста.

## Структура книги

Книга состоит из девяти глав, общий принцип написания которых заключается в проведении читателя от простых тем к сложным при помощи исследования проблем и решения задач. При написании книги авторы исходили из предположения, что наибольший методический эффект будет иметь текст «от задач», когда сам текст книги пишется на основе решения определённых практических задач. Авторы не ставили перед собой цели сделать каждую главу самостоятельным произведением, поэтому читателю рекомендуется знакомиться с главами последовательно, ничего не пропуская.

В главе ?? для рассмотрения читателем предлагается простой проблемно-ориентированный язык, который предназначается для использования в несложной практической задаче — воспроизведении протоколов шахматных партий. Рассматриваются возможные подходы к решению этой задачи, показываются преимущества использования DSL. Приводится неформальное описание предлагаемого DSL. При помощи рассуждения о свойствах, которыми должна обладать реализация DSL, читатель подводится к тому, что необходимо — представление состояния доски, операции по манипуляции им и средства ввода-вывода, причём определение состояния органично проистекает из дискуссии о свойствах DSL. Приводится пример описания состояния и операций по манипуляции им на языке Haskell, проводятся параллели с возможными реализациями на других языках. Обсуждаются два возможных подхода к реализации DSL — встраивание в язык и расширение языка.

В главе ?? производится расширение возможностей созданного в главе ?? языка. Этот язык дополняется средствами ввода-вывода, демонстрируется его работоспособность. На умоглядном примере взаимодействия

с другими реализациями этой же задачи (на других языках) показывается необходимость формального описания языков и грамматик. Кратко описываются: определение формальной грамматики, представляется нотация для описания формальных грамматик (расширенная нотация Бэкуса-Наура, контекстные диаграммы, математическая нотация). Приводится расширенная форма Бэкуса-Наура предложенного DSL. Демонстрируется пример на рассматриваемом DSL, который не удовлетворяет грамматике, но обрабатывается созданным решением.

В главе ?? читатель знакомится с понятием генератора лексических и синтаксических анализаторов. Использование таких генераторов демонстрируется на примере DSL из главы ??: существующее решение переписывается с использованием инструментальных средств `Alex` и `Happy`. Читатель знакомится с классификацией грамматик по Н. Хомскому, объясняются практические последствия использования грамматики того или иного типа (сложность парсера, возможность организации потокового парсера, потребление памяти в процессе синтаксического анализа и т. п.).

Глава ?? книги знакомит читателя с понятием комбинатора синтаксического анализа. Внимание читателя обращается на то, что при реализации инструмента анализа простых языков при помощи генераторов лексических и синтаксических анализаторов существенная часть программного кода состоит из сопряжения синтаксического анализатора и ядра языка. Читатель знакомится с комбинаторным подходом к построению синтаксических анализаторов. На примере DSL из главы ?? демонстрируется, как использование комбинаторных парсеров (библиотека `Parsec`) позволяет упростить программный код DSL. Проводятся параллели с аналогичными подходами (или их отсутствием) в других языках программирования.

В главе ?? детально рассматривается тезис о том, что в любой сравнительно сложной информационной системе неявно реализован один или несколько DSL (так называемое «десятое правило Гринспуна»). Совместно с читателем на примерах прослеживается развитие простой программы и производится наблюдение за тем, как по мере усложнения программного кода в нём появляются участки, которые разумно выделить в отдельный DSL.

Глава ?? посвящена введению в принципы проектирования DSL. Обсуждаются характеристики, которым должен обладать хороший и удобный в использовании DSL, и способы построения DSL, удовлетворяющих заданным характеристикам. Обсуждение сопровождается многочисленными примерами широко известных и повсеместно используемых DSL.

В главе ?? рассматриваются сложные DSL и демонстрируется построение DSL способом «встраивания в язык». Описывается реализация функционального языка, обрабатывающего правила. Показывается важность технологий программирования, дополняющих и расширяющих стандартные — монады, строгая статическая типизация и др. Приводятся многочисленные примеры программ на таком языке, кроме того приводится описание практического использования созданного DSL для вычислений: цепи Маркова, машина Тьюринга, простое  $\lambda$ -исчисление.

Поскольку в главе ?? реализован вполне законченный программный модуль, позволяющий обрабатывать правила, в главе ?? создаётся искусственная сущность, способная вести беседу на естественном языке. Приводится краткое описание теории анализа естественного языка, современных подходов к решению этой задачи. Разъясняется то, как при помощи несложных теоретических механизмов (цепи Маркова) можно построить достаточно мощного электронного собеседника. В конечном итоге производится расширение функциональности до бота, участвующего в чатах или интернет-конференциях.

Наконец, глава ?? повествует о том, как от реализации DSL можно перейти к более мощным задачам. Рассказывается, как можно создать частичный вычислитель для функционального языка, реализованного в главе ?. Проводится изучение вопросов перехода от частичного вычислителя к интерпретатору, компилятору и компилятору компиляторов (проекция Футамоуры — Турчина).

## Соглашения об оформлении

Наконец, настало время традиционного описания соглашений об оформлении текста, исходных кодов и прочего в настоящей книге. В целях единообразия представления здесь используются определённое форматирование текста, которое используется для выделения специальные структурные элементы. В отличие от предыдущих изданий при вёрстке 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: `map`, `init`, `Monad`, `snd` и т. д. Ключевые слова в свою очередь записываются полужирным начертанием обычного шрифта: **do**, **where**, **instance**. Знаки операций и специальные символы при записи внутри тек-

ста ограничиваются круглыми скобками: `(//)`, `(\$)` и т. д., а сами скобки при необходимости выделения в тексте записываются в кавычках: «`[`», «`]`».

При этом отдельные определения программных сущностей оформляются программными блоками с «подсветкой» ключевых слов, причём сам текст исходных кодов записывается моноширинным шрифтом с небольшой разрядкой:

```
sequence :: Monad m => [m a] -> m ()
sequence []      = return ()
sequence (x:xs) = do x
                  sequence xs
```

Исходные коды на других языках, в том числе и проектируемых в настоящей книге, оформляются абсолютно по таким же соглашениям.

## Благодарности

Авторы выражают искреннюю благодарность всем своим читателям, которые только одним лишь фактом того, что держат в руках эту книгу и читают эти строки, уже полностью оправдали весь труд, который был затрачен на выпуск настоящего издания. Вместе с тем авторы хотели бы особенно подчеркнуть заслугу тех, без мудрых советов которых книга была бы менее живой и интересной. Сие суть: \*\*\*.

## Об авторах

Данная книга является экспериментальным совместным трудом двух авторов, по отдельности известных в качестве популяризаторов идеологии и методов функционального программирования. При этом один автор больше известен как многосторонний практик, а второй — теоретик. Авторы надеются, что подобный дуумвират самым позитивным образом скажется как на качестве книги, так и на заинтересованности читателей.

*Дмитрий Евгеньевич Астапов* является известным в своём окружении популяризатором функционального программирования и языка Haskell. Является автором публикаций по нечёткой логике, математической лингвистике, а также учебных пособий по языку Haskell (в том числе и на английском языке). Участвовал в проведении инициативных программ Google Summer of Code в 2006 и 2007 годах в качестве одного из наставников сообщества

языка Haskell. Работает в области автоматизации предприятий телекоммуникационной индустрии, где использует проблемно-ориентированный подход для решения промышленных задач.

*Роман Викторович Душкин* является автором нескольких книг и методических пособий на русском языке о функциональном программировании на языке Haskell (см. [?, ?, ?]), а также множества научных публикаций по темам нечёткой математики, искусственного интеллекта и функционального программирования в российских и зарубежных научных изданиях. Состоит в Российской Ассоциации Искусственного Интеллекта (РАИИ) и в своё время участвовал во множестве национальных и международных научных конференциях, проводимых под её эгидой. С 2001 года читал лекции по функциональному программированию в Московском инженерно-физическом институте (МИФИ). В настоящее время работает в области автоматизации промышленности и государственного управления.

И как обычно авторы будут крайне признательны каждому читателю, кто не только прочитает предлагаемую книгу (хотя и этого уже достаточно для благодарностей), но и пришлёт свои отзывы по любому из следующих адресов электронной почты: `dastapov@gmail.com` или `dushkin.roman@gmail.com`.