Оглавление

Содержание

Предисловие

Я начал программировать еще подростком, изучая забавные книги с комиксами о волшебниках и черепахах. Я читал журналы, в которых мне показывали, как писать свои собственные простые игры или создавать на экране глупые эффекты. Мне было забавно.

Но когда я поступил в университет, в моих книгах стали обсуждаться банковские счета, балансы, отделы продаж, работники и работодатели. Я стал задумываться, а не будет ли моя жизнь программиста означать надевание серого костюма и написание исходного кода для систем расчета заработной платы. О ужас!

По крайней мере, половина моей группы страстно ненавидела программирование. И я не мог их винить. Почему книги по программированию должны быть такими скучными, функциональными и осмысленными?

Где ощущение приключений и чувство юмора? Юмор и веселье недооценены. Кого волнует, что книга безрассудна и содержит глупые шутки, если она побуждает вас учиться и получать удовольствие от учебы?

Это одна из причин, по которой я написал эту книгу. Я хотел, чтобы читатель получал удовольствие от изучения программирования – не от шуток, а от работы с интересными и забавными примерами программирования.

Обещаю, примеров моделирования отдела продаж не будет. Вместо этого мы будем делать такие вещи, как имитация запуска ракет, притворяться, что Цезарь отправляет секретное сообщение своим полководцам, используя древнеримские методы шифрования, и многое другое.

Вторая важная причина, по которой я захотел написать эту книгу, заключается в том, что люди продолжают меня спрашивать: «Разве язык Julia не предназначен только для науки и ученых?» Язык Julia добился в этой области больших успехов, поэтому сообщество Julia сегодня полно умных людей, которые работают над сложными задачами, такими как разработка новых лекарств и моделирование распространения инфекционных заболеваний, изменение климата и экономика.

Нет и еще раз нет, для того чтобы использовать Julia, вовсе не нужно быть гением или ученым. Julia – замечательный язык общецелевого программирования для всех! Я не ученый, и мне нравится использовать его уже более 9 лет. Вы обнаружите, что с Julia вы сможете решать задачи быстрее и элегантнее, чем раньше. И вишенкой на торте является то, что вычислительно-емкий исходный код будет исполняться невероятно быстро.

Признательности

Эта книга пережила несколько воплощений. В какой-то момент она была в самиздате. Позже меня свел случай с издательством Manning Publications, и мы договорились работать над изданием моей книги. В то время я не осознавал, в какую работу втягиваюсь. Я думал, что внесу лишь незначительные изменения в существующую книгу, но, посмотрев на полученные мною отзывы, понял, что придется вносить много правок.

Иногда хотелось сдаться. Однако, несмотря на трудности, считаю, что обширная система, созданная издательством Manning в помощь нам, авторам, помогла мне сделать книгу значительно лучше. За это я должен поблагодарить Николь Баттерфилд, которая организовала подписание контракта с издательством Manning. У меня было два редактора: Лесли Трайтес на ранней стадии книги и Марина Майклс, которая благодаря своему значительному опыту и твердой руке помогла мне выйти на финишную прямую. Хотел бы выразить благодарность Милану Курчичу, моему редактору-консультанту по техническому развитию, который очень помог своими отзывами в доведении понятности материала до уровня моей целевой аудитории. Мой редактор Кристиан Берк был для меня, не носителя английского языка, неоценим, исправляя любые нечеткие конструкции и грамматику.

Кроме того, хотел бы поблагодарить рецензентов, которые уделили время чтению моей рукописи на разных этапах ее разработки и предоставили бесценные отзывы: Алана Лентона, Аманде Деблер, Энди Робинсона, Криса Бейли, Дэниелу Кенни, Даррина Бишопа, Эли Майост, Эмануэлу Пиччинелли, Ганеша Сваминатана, Герта Ван Лаэтхема, Джеффа Барто, Иво Балберта, Джереми Чена, Джона Зётебир, Джонатана Оуэнса, Йорта Роденбурга, Катю Паткин, Кевина Чунга, Кшиштофа Енджеевски, Луиса Луангкесорна, Марка Томаса, Мора Уайлдера, Майка Барана, Никоса Канакариса, Нинослава Черкеса, Орландо Алехо Мендес Моралеса, Патрика Ригана, Пола Силистяну, Пола Вербеке, Самвида Мистри, Симона Сгуацца, Стива Грей-Уилсона, Тимоти Володзко и Томаса Хеймана.

Особая благодарность Маурицио Томази, техническому корректору, за его последний тщательный просмотр исходного кода незадолго до того, как книга была запущена в производство. Наконец, спасибо создателям языка Julia. Вы создали язык программирования будущего, который, уверен, изменит компьютерную индустрию. Возможно, это покажется преувеличением, но я искренне верю, что Julia станет важной вехой в эволюции языков программирования.

Об этой книге

Книга «Julia в качестве второго языка» представляет собой введение в язык программирования Julia для разработчиков программного обеспечения. Она не только охватывает синтаксис и семантику языка, но и обучает читателя думать и работать как разработчик Julia, уделяя особое внимание интерактивному программированию в среде, основанной на цикле чтения–вычисления–печати (REPL).

Кому стоит прочитать эту книгу?

Книга «Julia в качестве второго языка» написана для разработчиков, интересующихся языком программирования Julia, но не обязательно имеющих научную или математическую подготовку. Книга также является хорошей отправной точкой для всех, кто хочет изучить науку о данных или научные вычисления, поскольку язык Julia очень хорошо подходит для такой работы. Однако это не исключает других применений. Данная книга будет полезна любому разработчику, который хотел бы программировать на современном высокопроизводительном языке, повышающем его продуктивность.

Как эта книга устроена

Книга состоит из пяти частей, включающих 18 глав.

Часть I охватывает основы языка.

- Глава 1 объясняет особенности языка Julia, причину его создания и преимущества использования данного языка программирования.
- Глава 2 посвящена работе с числами на языке Julia. В ней показываются способы применения его интерактивной среды REPL в качестве очень сложного калькулятора.
- Глава 3 объясняет инструкции потока управления, такие как инструкции `if`, циклы `while` и `for`, путем реализации тригонометрической функции и вычисления чисел Фибоначчи.
- Глава 4 объясняет способы работы с коллекциями чисел, используя тип `Array`. Читатели рассмотрят пример с данными о продажах пиццы.
- Глава 5 посвящена работе с текстом. В этой главе вы познакомитесь с созданием демонстраций красиво отформатированных данных о продажах пиццы с использованием цвета, а также с чтением данных о пицце из файлов и записью их в файлы.

- Глава 6 посвящена реализации программы конвертации римских цифр в десятичный формат, используя словарный тип.

Часть II посвящена более подробному описанию системы типов языка Julia.

- Глава 7 излагает иерархию типов Julia и способы определения своих собственных составных типов. Это одна из самых важных глав, потому что в ней также объясняется одна из самых важных и уникальных функциональностей языка Julia – множественная диспетчеризация.
- Глава 8 демонстрирует пример исходного кода имитации ракеты, который мы будем использовать в нескольких последующих главах. Эта глава посвящена определению типов для различных частей ракеты.
- Глава 9 посвящена более подробному рассмотрению числовой конвертации и приведению типов в Julia путем создания примера исходного кода, работающего с разными единицами измерения градусов. Эта глава помогает закрепить понимание системы множественной диспетчеризации Julia.
- Глава 10 посвящена способам представления несуществующих, отсутствующих или неопределенных объектов на языке Julia.

В части III вновь рассматриваются типы коллекций, такие как массивы, словари и строковые литералы, которые были рассмотрены в части I, но на этот раз во всех подробностях.

- Глава 11 посвящена более подробному рассмотрению строковых литералов, в том числе использованию Юникода и UTF-8 в Julia, а также их влиянию на применение вами строковых литералов.
- Глава 12 посвящена объяснению общих для всех коллекций Julia черт, таких как прокручивание элементов в цикле, а также разработка своих собственных коллекций.
- Глава 13 демонстрирует несколько примеров исходного кода с использованием множеств и операций на множествах, чтобы показать организацию и поиск данных во многих типах приложений.
- Глава 14 иллюстрирует приемы работы с массивами разных размерностей, такими как векторы и матрицы, и их комбинирования.

Часть IV посвящена методам организации исходного кода на разных уровнях, включая модуляризацию на уровне функций вплоть до пакетов, файлов и каталогов.

- Глава 15 посвящена более подробному рассмотрению применений функций в Julia с акцентом на отличия функционального программирования от объектно ориентированного.
- Глава 16 посвящена организации исходного кода в модули, использованию сторонних пакетов и созданию своих собствен-

ных пакетов для распространения исходного кода среди других разработчиков.

Часть V посвящена углубленному изложению деталей, которые трудно объяснить без предыдущих глав, которые заложили основы.

- Глава 17 основана на главе 5. Вы подробно узнаете о системе ввода-вывода Julia, читая и записывая ракетные двигатели в файлы, сокеты и конвейеры в формате CSV.
- Глава 18 объясняет приемы определения параметрического типа данных и причину, по которой параметрические типы выгодны для производительности, потребления памяти и правильности исходного кода.

Об исходном коде

Эта книга содержит много примеров исходного кода, как в пронумерованных листингах, так и в виде обычного текста. В обоих случаях исходный код отформатирован `вот таким шрифтом фиксированной ширины`.

Во многих случаях изначальный исходный код был переформатирован; мы добавили разрывы строк и переработали отступы, чтобы они соответствовали доступному пространству книжной страницы. В редких случаях даже этого было недостаточно, и листинги содержали маркеры продолжения строки (➥). Кроме того, комментарии в исходном коде часто из листингов удалялись, когда исходный код описывался в тексте. Многие листинги сопровождаются аннотациями к исходному коду, выделяя важные концепции.

Бóльшая часть исходного кода пишется в интерактивной среде REPL[1] (работащей в цикле чтения–вычисления–печати) языка Julia либо в оболочке Unix. В этих случаях вы видите приглашение `julia>`, `shell>`, `help?>` или `$`. При наборе текста исходного кода их вставлять не нужно. Однако при вставке примеров исходного кода в окно терминала приглашение обычно отфильтровывается самим языком Julia.

Исходный код, предназначенный для записи в файл, обычно не отображается с приглашением. Однако если хотите, можете вставлять этот исходный код в интерактивную среду REPL.

Исполняемые фрагменты исходного кода можно получить из liveBook-версии (онлайновой версии) этой книги по адресу https://livebook.manning.com/book/julia-as-a-second-language. Полный исходный код примеров книги доступен для скачивания с веб-сайта издательства Manning по адресу https://www.manning.com/books/julia-as-a-second-language и с GitHub по адресу https://github.com/ordovician/code-samples-julia-second-language.

Для выполнения примеров исходного кода этой книги рекомендуется версия Julia 1.7 или выше.

---

[1] Англ. read-evaluate-print-loop. – *Прим. перев.*

Дискуссионный форум liveBook

Покупка книги «Julia в качестве второго языка» включает в себя бесплатный доступ к онлайновой платформе чтения книг издательства Manning под названием liveBook. Используя эксклюзивные возможности обсуждения на liveBook, можно прикреплять комментарии к книге глобально либо к определенным разделам или абзацам. Там можно легко делать заметки для себя, задавать технические вопросы и отвечать на них, а также получать помощь от автора и других пользователей. В целях получения доступа к форуму перейдите по ссылке https://livebook.manning.com/book/julia-as-a-second-language/discussion. Подробнее о форумах издательства Manning и правилах поведения также можете узнать по адресу https://livebook.manning.com/discussion.

Издательство Manning видит свою обязанность перед читателями в том, чтобы предоставлять место, где может происходить содержательный диалог между отдельными читателями и между читателями и автором. Это обязательство не требует от автора какого-то конкретного объема участия, чей вклад в форум остается добровольным (и неоплачиваемым). Мы предлагаем вам попробовать задать автору несколько сложных вопросов, чтобы поддержать его интерес! Форум и архивы предыдущих обсуждений будут доступны на вебсайте издателя на протяжении всего времени, пока книга находится в печати.

Другие онлайновые ресурсы

Нужна дополнительная помощь? Язык Julia имеет активное рабочее пространство/сообщество Slack с более чем 10 000 участников, со многими из которых вы можете общаться в реальном времени. Информация о регистрации находится на странице https://julialang.org/slack.

- Julia Discourse (https://discourse.julialang.org) – это форумная платформа, где можно задать вопросы, связанные с Julia.
- На странице сообщества Julia по адресу https://julialang.org/community есть информация о каналах YouTube, предстоящих мероприятиях Julia, репозиториях GitHub и каналах Twitter.
- Официальная документация по языку и стандартной библиотеке Julia находится по адресу https://docs.julialang.org/en/v1/.

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

Эрик Энгхейм – писатель, докладчик на конференциях, автор видеокурсов и разработчик программного обеспечения. Бóльшую часть своей карьеры он посвятил разработке программного обеспечения 3D-моделирования и симулирования резервуаров в норвежской газовой и нефтяной промышленности. Эрик также несколько лет работал разработчиком iOS и Android. Эрик программирует на Julia, а также пишет и снимает видео о данном языке с 2013 года.

Часть I

Основы

Эти главы охватывают весь язык Julia на базовом уровне. В последующих главах затронутые в этой части темы будут расширены. Вы научитесь работать с числами, массивами, инструкциями `if`, циклами `for`, строковыми литералами, базовым вводом-выводом, а также узнаете о хранении и извлечении данных из словарей. В последующих частях книги эти темы будут представлены подробнее.

1 Почему именно Julia?

Эта глава охватывает следующие ниже темы:

- тип задач, решаемых с помощью Julia;
- преимущества быстрого динамически типизированного языка;
- как Julia повышает продуктивность программиста.

У вас есть возможность выбирать из сотен языков программирования, многие из которых намного популярнее языка Julia. Так почему же именно Julia?

Хотите писать исходный код быстрее, чем раньше? А что скажете насчет создания систем с меньшим числом строк исходного кода, которое требуется в обычной ситуации? Наверняка такая продуктивность будет достигнута за счет плачевной производительности и большого потребления памяти. Отнюдь. На самом деле язык Julia предпочтителен для климатических моделей следующего поколения, которые имеют экстремальные требования по производительности и памяти.

Знаю, что такие похвалы, несомненно, создадут впечатление неудачной рекламной уловки продавца подержанных автомобилей, но нельзя отрицать, что Julia во многих отношениях является революционным языком программирования. Возможно, вы спросите: «Раз уж Julia – такой замечательный язык, то почему им не пользуются

все? Почему так много людей до сих пор используют язык программирования C?» Тут важны знакомство, пакеты, библиотеки и сообщество разработчиков. Перенос специфического программного обеспечения, разработанного в крупных организациях, не делается под влиянием момента.

Многих из вас, кто читает эту книгу, возможно, не интересует язык более эффективного и продуктивного программирования. Вместо этого вас прежде всего интересует, а что, собственно говоря, на нем можно разрабатывать. На этот вопрос есть простой ответ: все, что угодно. Julia – это язык общецелевого программирования.

По-видимому, такой ответ не будет удовлетворительным. В принципе, на JavaScript тоже можно разрабатывать все, что угодно. Тем не менее вы знаете, что JavaScript доминирует в разработке клиентских частей веб-приложений. На Lua тоже можно писать что угодно, но он в основном используется как скриптовый язык для компьютерных игр. При чтении этой книги, возможно, вас в первую очередь будет интересовать, какую именно работу Julia может для вас выполнять.

В настоящее время сообщество Julia наиболее активно занимается научными вычислениями, наукой о данных и машинным обучением. Но изучение языка Julia – это еще и ставка на будущее. Язык с такими мощными возможностями не останется в узкой нише. Если вы будете читать дальше, то вам станут понятнее особенности языка Julia и причины, по которым он обладает таким потенциалом. Я также затрону области, в которых Julia не идеален.

1.1 Что из себя представляет язык Julia?

Julia – это язык общецелевого и многоплатформенного программирования, который характерен следующим:

- пригодностью для численного анализа и вычислительной науки;
- динамической типизацией;
- высокой производительностью и компиляцией во время выполнения программы;
- автоматическим управлением памятью (сборкой мусора);
- компонуемостью.

Это немало, и некоторые из приведенных выше свойств выглядят противоречиво. Так как же языку Julia удается быть общецелевым и при этом приспособленным для численного программирования? Он предназначен для общих задач, потому что, как и Python, язык Julia можно использовать практически для чего угодно. Он ориентирован на числовую обработку, потому что, как и MATLAB, хорошо подходит для численного программирования. Но он не ограничен численным программированием; он неплох и для других целей. Под *компонуемостью* подразумевается легкость, с которой Julia позволяет выражать многие архитектурные шаблоны объектно ориенти-

рованного и функционального программирования, способствуя переиспользованию исходного кода.

1.1.1 Плюсы и минусы статически и динамически типизированных языков

Давайте сосредоточимся на одном аспекте языка Julia: на его динамической типизируемости. Обычно языки программирования делятся на две большие категории:

- динамически типизированные;
- статически типизированные.

> *В статических языках типы есть у выражений; в динамических языках типы есть у значений.*
>
> – Стефан Карпински (Stefan Karpinski),
> создатель языка Julia

Примерами статически типизированных языков являются C/C++, C#, Java, Swift, Go, Rust, Pascal и Fortran. В статически типизированном языке проверки типов выполняются по всему исходному коду перед выполнением программы.

Примерами динамически типизированных языков являются Python, Perl, Ruby, JavaScript, MATLAB и LISP. Динамически типизированные языки выполняют проверку типов во время работы программы. К сожалению, динамически типизированные языки работают очень медленно.

В динамических языках к значениям, таким как числа, буквенные символы и строковые литералы, прикреплены теги, указывающие тип, к которому они относятся. Эти теги позволяют программам, написанным на динамически типизированном языке, проверять правильность типов во время их исполнения.

Язык Julia необычен тем, что он является динамически типизированным и высокопроизводительным языком. Для многих наличие этих двух свойств выглядит как противоречие. Подобная уникальная черта языка Julia стала возможной благодаря тому, что данный язык был специально разработан под JIT-компиляцию[1] и в нем для всех вызовов функций используется особая функциональность, именуемая *множественной диспетчеризацией*. В таких языках, как C/C++ и Fortran, используется AOT-компиляция[2]. Компилятор транслирует всю программу в машинный код перед ее исполнением. В других языках, таких как Python, Ruby и Basic, используется интерпретатор. В интерпретируемых языках каждая строка исходного кода чита-

---

[1] Англ. just-in-time (JIT) compilation; син. компиляция во время выполнения программы. – *Прим. перев.*

[2] Англ. ahead-of-time (AOT) compilation; син. компиляция перед выполнением программы. – *Прим. перев.*

ется и интерпретируется во время выполения программы и затем инструкции исполняются. Теперь, когда у вас есть представление об особенностях языка Julia, можно приступить к обсуждению его привлекательности.

> **Конструктивное исполнение языка и JIT-компиляция**
>
> В принципе, язык программирования отделен от метода, используемого для работы с ним. И тем не менее вы обнаружите, что я говорю о Julia как о JIT-компилируемом языке, а о Fortran как о AOT-компилируемом языке. Строго говоря, это не совсем точно. Например, Julia может работать и посредством интерпретатора. Однако большинство языков были разработаны под конкретную форму исполнения. Julia был разработан под JIT-компиляцию.

1.2 Julia сочетает в себе элегантность, продуктивность и производительность

Хотя производительность является одним из ключевых преимуществ Julia, в 2013 году мое внимание привлекли его продуманность, мощность и простота в использовании. У меня была программа, которую я переписал на нескольких языках, чтобы сравнить выразительность, простоту в использовании и продуктивность программировния на каждом языке. На языке Julia мне удалось написать самый элегантный, компактный и легко читаемый вариант этого исходного кода. С тех пор я испробовал много других языков программирования, но так и не приблизился к тому, чего добился на языке Julia. Вот несколько однострочников, иллюстрирующих выразительность данного языка.

Листинг 1.1 Однострочники на языке Julia

```
filter(!isempty, readlines(filename)) # удалить пустые строки
filter(endswith(".png"), readdir())   # получить PNG-файлы
findall(==(4), [4, 8, 4, 2, 5, 1])    # найти все индексы числа 4
```

Занимаясь программированием с 1990-х годов, у меня были периоды, когда я чувствовал, что ну все, с меня программирования достаточно; язык Julia помог мне вернуть радость от программирования. Частично причина заключалась в том, что, освоив Julia, чувствуешь, что в твоем инструментарии появился язык, который работает как член твоего коллектива, а не против тебя. Думаю, что многие из нас имели опыт работы над задачей в ситуации, когда у вас было хорошее представление о методе ее решения, но используемый язык портил все дело. Ограничения языка вынуждают добавлять одну программную уловку за другой. На языке Julia я могу разрабатывать программное обеспечение так, как хочу, без языковых препон.

Еще один аспект, повышающий продуктивность и удовольствие, – поставляемая в комплекте обширная стандартная библиотека. Вы беретесь за дело с места в карьер и можете делать многое, не рыская по всему интернету в поисках какой-нибудь библиотеки, которая делает то, что вы хотите. Julia поможет вам, независимо от ваших потребностей, будь то линейная алгебра, статистика, HTTP, манипулирование строковыми литералами или же работа с разными форматами даты. А если нужной возможности в стандартной библиотеке нет, то в Julia есть тесно интегрированный менеджер пакетов, который упрощает добавление сторонних библиотек. Программирование на языке Julia почти заставляет вас чувствовать себя виноватым либо избалованным, потому что есть возможность разрабатывать богатые и элегантные абстракции без ущерба для целей по производительности.

Еще одним существенным преимуществом Julia является то, что данному языку легко обучаться. Эта простота помогает Julia с течением времени расширять сообщество разработчиков. В целях понимания причины легкой усваиваемости языка Julia давайте возьмем всем известную программу *Привет, мир*, написанную на Julia:

```
print("Hello world")
```

При ее выполнении приведенный выше исходный код выводит на экран текст *Hello world*. Несмотря на ее тривиальность, для того чтобы сделать нечто такое же простое, во многих языках требуется немало сложных строительных лесов. Ниже приведена программа на Java, которая делает то же самое:

```
public class Main {
    public static void main(String[] args) {
        System.out.print("Hello world");
    }
}
```

Приведенный выше фрагмент исходного кода одновременно выливает на новичка гораздо большее, нередко ошеломляющее, число концепций. Язык Julia легче в освоении, потому что он позволяет сосредотачиваться на одной концепции за раз. Например, можно научиться писать функцию, даже не видя определения типа. Благодаря огромному числу функциональностей, доступных прямо «из коробки», для написания полезного исходного кода даже не нужно уметь импортировать внешние библиотеки.

1.3 Для чего был создан язык Julia

В целях более глубокого ознакомления с тем, что Julia дает вам как разработчику, прежде всего необходимо лучше понять причину соз-

дания языка Julia. Создатели языка программирования Julia хотели решить то, что они назвали *проблемой двух языков*.

Эта проблема связана с тем, что многие программы пишутся с использованием двух разных языков программирования, каждый из которых имеет разные характеристики. В научной сфере часто предпочтение отдается динамическим языкам машинного обучения и анализа данных. Однако эти языки обычно не дают достаточно хорошей производительности. Поэтому программные решения нередко приходится переписывать на высокопроизводительных статически типизированных языках. Но в чем причина существования этого предпочтения? Почему бы не писать весь исходный код на традиционном высокопроизводительном статически типизированном языке?

1.3.1 Ученые нуждаются в интерактивном программировании, которое предлагается динамически типизированными языками

Ученые начали писать программное обеспечение, в том числе крупномасштабные метеорологические симуляции, на Fortran[1] и с использованием нейронных сетей[2] на C или C++[3]. Эти языки обеспечивают производительность, необходимую для решения указанных крупномасштабных задач. Однако эти языки имеют свою цену. Они тяготеют к жесткости, многословности и малой выразительности – все это снижает продуктивность программиста.

Однако фундаментальная проблема заключается в том, что эти языки не подходят для интерактивного программирования. Что я имею в виду? Интерактивное программирование – это возможность писать исходный код и получать немедленный отклик.

Интерактивное программирование имеет большое значение в науке о данных и машинном обучении. В типичном процессе анализа данных разработчик загружает большие объемы данных в интерактивную среду программирования и затем их обследует. После этого он выполняет различные виды анализа данных. Эти виды анализа могут предусматривать поиск среднего и максимального значений или построение гистограмм. Результаты первичного анализа сообщают программисту о характере последующих шагов.

На рис. 1.1 этот процесс показан на динамически типизированном языке. Все начинается с выполнения исходного кода загруз-

---

[1] Fortran (Formula Translation) – это старый язык научных вычислений.

[2] Нейронные сети – это своего рода алгоритм, симулирующий работу человеческого мозга.

[3] C и C++ – это родственные и широко используемые статически типизированные языки системного программирования.

ки данных, которые затем можно обследовать. Однако при этом не обязательно проходить весь этот процесс после изменения исходного кода. Есть возможность изменять исходный код и сразу же видеть изменения. Перезагрузка огромных объемов данных не требуется.

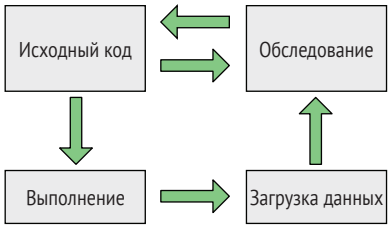

Рис. 1.1 В динамически типизированных языках можно переключаться между программированием и обследованием. Большие наборы данных не нужно перезагружать в память

Давайте сравним этот опыт с использованием статически типизированного языка, к примеру такого, как Fortran, C/C++ или Java[1]. Разработчик должен писать исходный код загрузки данных и обследовать их, ничего не зная о том, как выглядят данные. Затем ему приходится ждать до тех пор, пока программа не сделает следующее:

1. скомпилируется;
2. запустится, а затем загрузит большой объем данных.

В этот момент разработчик видит график данных и статистические величины, которые дают ему информацию, необходимую для выбора следующего вида анализа. Но выбор следующего вида анализа потребовал бы повторения всего цикла заново. На каждой итерации должен перезагружаться большой блок данных. За счет этого чрезвычайно замедляется каждая итерация, замедляя весь процесс анализа в целом. Это статический, неинтерактивный способ программирования (рис. 1.2).

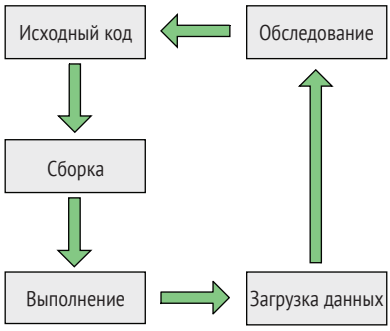

Рис. 1.2 Статически типизированные языки требуют повторения всего цикла

[1] Java используется во многих веб-серверных программах и в телефонах на базе Android.

1.3.2 Разработчикам в других областях тоже нужна интерактивность, предлагаемая динамически типизированным языком

Эта проблема не уникальна для ученых; разработчики игр уже давно столкнулись с той же проблемой. Игровые движки обычно пишутся на таких языках, как C или C++, которые компилируются в быстрый машинный код. Эта часть программного обеспечения нередко выполняет понятные и четко определенные действия, такие как рисование объектов на экране и проверка на столкновение объектов друг с другом.

Как и у аналитика данных, у разработчика игр есть много исходного кода, для удовлетворительной работы которого требуются многочисленные итерации. В частности, при разработке хорошего игрового процесса требуется много экспериментов и итераций. Нужно настраивать и изменять исходный код поведения персонажей в игре. Нужно многократно экспериментировать с композицией карты или уровнем, чтобы все работало, как надо. По этой причине почти во всех игровых движках используется второй язык, который позволяет изменять исходный код на лету. Часто это такие языки, как Lua[1], JavaScript и Python[2].

На этих языках исходный код игровых персонажей и карт можно изменять без перекомпиляции и перезагрузки карт, уровней и персонажей. Благодаря этому появляется возможность экспериментировать с игрой, делать паузы, вносить изменения в исходный код и сразу же продолжать с новыми изменениями.

Профессионалы машинного обучения сталкиваются с аналогичными проблемами. Они разрабатывают предсказательные модели, такие как нейронные сети, в которые они подают большие объемы данных, чтобы тренировать модели. Нередко здесь требуется столько же искусства, сколько и науки. Для обеспечения результативной работы моделей необходимо экспериментировать. Если приходится перезагружать тренировочные данные при каждом изменении модели, то процесс разработки замедляется. По этой причине в научном сообществе стали очень популярными динамически типизированные языки, такие как Python, R и MATLAB.

Однако, поскольку эти языки не очень быстрые, их используют в паре с такими языками, как Fortran и C/C++, чтобы выжимать нужную производительность. Нейронная сеть, созданная с помощью TensorFlow[3] или PyTorch[4], состоит из компонентов, написанных на

---

[1] Изначально Lua создавался как конфигурационный язык, но сегодня он в основном используется для написания игр.

[2] Сегодня Python – это один из самых популярных языков науки о данных и машинного обучения.

[3] TensorFlow – это популярная библиотека машинного обучения и платформа на Python.

[4] PyTorch – это популярная платформа машинного обучения на Python.

C/C++. Python используется для организации и соединения этих компонентов. Таким образом, во время выполнения эти компоненты можно реорганизовывать с помощью Python, не перезагружая всю программу.

Климатические и макроэкономические модели в процессе разработки, как правило, сначала разрабатываются на динамическом языке и тестируются на малом наборе данных. По завершении модели многие организации нанимают разработчиков на C/C++ или Fortran, чтобы те переписывали вычислительное решение на высокопроизводительном языке. Как следствие появляется лишний шаг, усложняющий процессы разработки и увеличивающий затраты.

1.4 Более высокая производительность языка Julia решает проблему двух языков

Язык Julia был создан решить проблему необходимости использования двух языков. Он позволяет сочетать гибкость динамически типизированного языка с производительностью статически типизированного языка. Вот почему приобрела популярность следующая ниже поговорка:

> *Julia ходит как Python и бегает как C.*
>
> – Популярная поговорка в сообществе Julia

Используя язык Julia, разработчики во многих областях могут писать исходный код с той же продуктивностью, что и на таких языках, как Python, Ruby, R и MATLAB. По этой причине язык Julia оказал глубокое влияние на отрасль. В июльском выпуске журнала Nature за 2019 год было взято несколько интервью с различными учеными об использовании ими языка Julia.

Например, благодаря переносу вычислительных моделей с R на Julia Мельбурнский университет добился 800-кратного улучшения. Джейн Херриман (Jane Herriman) из Калифорнийского технологического института материаловедения сообщает, что с тех пор, как она переписала свой Python'овский исходный код на Julia, его работа ускорились в десять раз.

> *За час можно сделать то, на что в противном случае ушли бы недели или даже месяцы.*
>
> – Майкл Штумпф (Michael Stumpf)

В 2019 году на Международной конференции по суперкомпьютерам (SC19) один из создателей языка Julia Алан Эдельман (Alan Edelman) рассказал, как группа из Массачусетского технологического института (MIT) переписала часть своей климатической Fortran'овской

модели на языке Julia. Они заранее определили, что для них будет терпимо трехкратное замедление работы исходного кода. По их мнению, это было бы приемлемым компромиссом при получении доступа к языку высокого уровня с более высокой продуктивностью программирования. Вместо этого за счет переключения на Julia они получили 3-кратный прирост скорости.

Это всего лишь несколько историй из большого числа прецедентов, когда язык Julia революционизировал научные вычисления и высокопроизводительные вычисления в целом. Избегая проблемы двух языков, ученые получают возможность работать намного быстрее, чем раньше.

1.5 Julia для всех

Возможно, эти истории создают ложное впечатление о том, что Julia – это язык для умников в белых лабораторных халатах. Ничего подобного. Оказывается, многие черты, которые делают язык Julia отличным языком для ученых, также делают его отличным языком для всех остальных. Julia предлагает:

- мощные средства модуляризации и реиспользования исходного кода;
- строгую систему типов, помогающую выявлять дефекты в исходном коде во время выполнения;
- изощренную систему сокращения повторяющегося стереотипного исходного кода (метапрограммирование[1]);
- богатую и гибкую систему типов, позволяющую моделировать широкий спектр задач;
- хорошо оснащенную стандартную библиотеку и различные сторонние библиотеки для решения различных задач;
- отличные средства обработки строковых литералов. Эта способность обычно является ключевым преимуществом любого многофункционального языка программирования, и это то, что изначально сделало популярными такие языки, как Perl, Python и Ruby;
- простое взаимодействие с большим числом других языков программирования и инструментов.

Хотя важное преимущество языка Julia относится к устранению им проблемы двух языков, это вовсе не означает, что отпадает потребность в интерфейсе с существующим исходным кодом, написанным на Fortran, C или C++. Смысл решения проблемы двух языков состоит в предотвращении необходимости писать исходный код на Fortran или C всякий раз, когда программист сталкивается с проблемой про-

[1] Метапрограммирование – это исходный код для написания исходного кода. Данная продвинутая концепция в книге не рассматривается.

изводительности. На всем протяжении можно оставаться в рамках языка Julia.

Однако если кто-то уже решил вашу задачу на другом языке, то, пожалуй, переписывание этого решения с чистого листа на языке Julia не будет иметь смысла. Python, R, C, C++ и Fortran имеют большие пакеты, которые создавались в течение многих лет, и сообщество Julia не может заменить их за одну ночь. В целях обеспечения продуктивной работы разработчикам Julia нужно уметь пользоваться существующими программными решениями.

В долгосрочной перспективе перенос унаследованного программного обеспечения на язык Julia даст очевидное преимущество. Поддержка старых библиотек Fortran часто требует от разработчиков гораздо больше усилий, чем поддержка библиотеки Julia.

Наибольшее преимущество, вероятно, заключается в предоставляемой языком Julia комбинаторной мощности. Определенные типы задач требуют создания больших монолитных библиотек. Julia, напротив, исключительно хорошо подходит для разработки малых библиотек, которые можно легко комбинировать, чтобы соответствовать функциональности, предлагаемой большими монолитными библиотеками на других языках. Приведу пример.

Машинное обучение, будучи животрепещущей темой, приводит в действие технологии беспилотного вождения автомобилей, распознавания лиц, распознавания голоса и многие другие инновационные технологии. Самыми известными пакетами машинного обучения являются PyTorch и TensorFlow. Эти пакеты представляют собой огромные монолиты, поддерживаемые большими коллективами разработчиков и сопроводителей. Между ними нет никакого обмена исходным кодом. В Julia есть огромное число библиотек машинного обучения, таких как Knet, Flux (см. https://fluxml.ai) и Mocha (см. http://mng.bz/epxG). По сравнению с ними эти библиотеки – крошечные. Почему? Потому что возможностей PyTorch и TensorFlow можно достичь путем комбинирования нескольких малых библиотек Julia. Объяснение механизма и причин, по которым это работает, является сложной темой, требующей гораздо более глубоких знаний о языке Julia и о том, как работают нейросетевые библиотеки.

Наличие большого числа малых библиотек является преимуществом для общих приложений. Любой, кто разрабатывает программное обеспечение, выиграет от возможности реиспользовать существующие части программного обеспечения множеством новых способов, без необходимости изобретать велосипед. При использовании традиционных языков программирования часто приходится реализовывать одну и ту же функциональность. Например, библиотеки TensorFlow и PyTorch имеют много повторяющейся функциональности. Julia избегает дублирования, добавляя в библиотеки гораздо больше функциональных возможностей, общих между многими библиотеками машинного обучения. По мере прочтения глав этой книги вам будет становиться все яснее, как Julia удается

это делать и почему во многих других языках эта возможность труднореализуема.

1.6 Что можно разрабатывать на языке Julia?

В принципе, на языке Julia можно разрабатывать все, что угодно. Однако у каждого языка есть экосистема пакетов и сообщество, которое подталкивает к одним типам разработки, а не к другим. Julia ничем не отличается.

1.6.1 Julia в науке

Язык Julia имеет сильное присутствие в науках. Данный язык используется, например, в:

- вычислительной биологии;
- статистике;
- машинном обучении;
- обработке изображений;
- вычислительном исчислении;
- физике.

Но язык Julia охватывает гораздо больше областей. Например, он используется в торговле энергией. Федеральная резервная система США применяет его для разработки сложных макроэкономических моделей. Лауреат Нобелевской премии Томас Дж. Сарджент (Thomas J. Sargent) основал платформу QuantEcon, которая продвигает педагогику в области квантитативной экономики с использованием как Julia, так и Python. Он горячий сторонник языка Julia, поскольку большие задачи в макроэкономике трудно решаемы с помощью других языков программирования. В интервью Лукасу Бивальду (Lukas Biewald) Питер Норвиг (Peter Norvig), работающий в Google известный исследователь искусственного интеллекта (ИИ), высказался, что, по его мнению, мир машинного обучения только выиграет от перехода на Julia.

> *Я был бы счастливее, если бы язык Julia стал основным языком ИИ.*
>
> – Питер Норвиг,
> автор книги «Искусственный интеллект: современный подход»[1]

Медико-биологические науки – еще одна очевидная область применения языка Julia. К 2025 году ежегодно будет собираться от 2 до 40 экзабайт данных о геноме человека. Большинство мейнстримных программ не смогут обрабатывать данные в таком масштабе. Для этого понадобится высокопроизводительный язык, такой как Julia,

[1] Artificial Intelligence, A Modern Approach. – *Прим. перев.*

который будет способен работать с различными форматами на различном оборудовании с максимально возможной производительностью.

На момент написания этой книги COVID-19 по-прежнему остается серьезной проблемой в мире. Пакет Julia `Pathogen` используется для моделирования инфекционных заболеваний и применяется исследователями COVID-19.

1.6.2 Использование Julia в ненаучных целях

А как насчет его ненаучного использования? В языке Julia также есть целый ряд пакетов для других интересных областей:

- `Genie` – полнофункциональный веб-фреймворк на основе архитектурного шаблона MVC;
- `Blink` – создание приложений с графическим пользовательским интерфейсом Electron на языке Julia;
- `GTK` – создание приложений с графическим пользовательским интерфейсом Julia с использованием популярного инструментария Linux GUI GTK;
- `QML` – создание кросс-платформенных графических интерфейсов с использованием языка разметки QML, используемого в инструментарии Qt GUI;
- `GameZero` – разработка игр для начинающих;
- `Luxor` – рисование векторных изображений;
- `Miletus` – составление финансовых контрактов;
- `TerminalMenus` – внедрение интерактивных меню в терминале;
- `Gumbo` – синтаксический разбор HTML-страниц;
- `Cascadia` – библиотека селекторов CSS для просмотра веб-страниц и извлечения полезной информации из веб-страниц;
- `QRCode` – создание изображений QR-кодов, широко применяемых в рекламе для отображения машиночитаемых URL-адресов.

Как видите, в Julia есть пакеты для общецелевого программирования.

1.7 Где язык Julia менее идеален

В принципе, Julia можно использовать практически для чего угодно, но поскольку данный язык молод, выбор библиотек не во всех областях одинаково обширен. Например, выбор пакетов для веб-разработки ограничен. Создать мобильное приложение на Julia пока что не получится. Он также не подходит для малых краткосрочных скриптов – таких, которые часто пишутся на Bash, Python или Ruby. Эти ограничения связаны с тем, что Julia компилируется на основе технологии JIT.

Это означает, что программы на Julia запускаются медленнее, чем, например, программы на Python или Bash, но начинают работать намного быстрее после того, как JIT-компилятор конвертирует критически важные части исходного кода в машинный код. Сообщество Julia постоянно прилагает усилия, чтобы минимизировать эту проблему, и существует громадное число способов ее решения. Решения предусматривают более оптимальное кеширование предыдущих JIT-компиляций и более избирательное отношение к тому, что следует JIT-компилировать.

Julia также неидеален для реально-временных систем. В реально-временной системе программное обеспечение должно реагировать на происходящее в реальном мире. Например, это можно сравнить с метеорологическим симулятором. В метеорологическом симуляторе не имеют значения события, происходящие в мире за пределами компьютера, на котором выполняется симуляция.

Однако если ваша программа должна обрабатывать данные, поступающие от измерительного прибора каждую миллисекунду, то внезапные сбои или задержки просто недопустимы. В противном случае есть риск потери важных данных измерений. Язык Julia укомплектован сборщиком мусора. Это означает, что данные, которые в вашей программе больше не используются, автоматически рециркулируются для других целей. Процесс определения планируемой к рециркуляции памяти обычно приводит к краткострочным случайным задержкам и заминкам в выполнении программы.

Эту проблему невозможно переоценить. Робототехнические комплексы, требующие реально-временного поведения, уже создаются на Julia. Исследователи из Массачусетского технологического института просимулировали реально-временное управление роботом-гуманоидом Boston Dynamics Atlas, удерживающим равновесие на плоской поверхности. Цель симуляции состояла в том, чтобы доказать, что язык Julia можно использовать для онлайнового управления роботами путем регулировки выделения и высвождения памяти.

Julia плохо подходит для встраиваемых систем с ограниченной памятью. Причина в том, что Julia добивается высокой производительности за счет создания узкоспециализированных версий одного и того же исходного кода. Следовательно, потребление памяти для самого исходного кода Julia будет выше, чем, скажем, для исходного кода C, C++ или Python.

Наконец, точно так же, как Python, Ruby и другие динамические языки, Julia не подходит для типичного системного программирования, такого как создание систем баз данных или ядер операционных систем. Эти задачи, как правило, требуют детального контроля за использованием ресурсов, чего Julia не предлагает. Julia – это язык высокого уровня, ориентированный на простоту использования, а значит, многие детали использования ресурсов абстрагированы.

1.8 Что вы узнаете из этой книги

Если вы уже программируете на другом языке, эта книга для вас. Каждый язык программирования имеет уникальный набор функциональностей, инструментов и сообществ. В этой книге я сосредоточусь на уникальных характеристиках языка Julia, а также на инструментах и сообществе, созданном вокруг Julia, включая следующие неотъемлемые аспекты:

1. интерактивное программирование с использованием программы командной строки, работающей в цикле чтения–вычисления–печати (REPL)[1];
2. научно и математикоориентированные примеры исходного кода;
3. уникальная функциональность множественной диспетчеризации и система типов Julia;
4. функциональное программирование и его сравнение с объектно ориентированным программированием;
5. пакетно ориентированная разработка вместо разработки, ориентированной на приложения.

Разработка на основе интерактивной среды REPL означает возможность запускать инструмент командной строки Julia и начинать набирать выражения Julia, которые вычисляются при нажатии клавиши **Enter**:

```
julia> reverse("abc")
"cba"

julia> 3+5
8
```

На протяжении бóльшей части книги я придерживаюсь именно такого подхода; возможно, указанный подход будет малознаком читателям, пришедшим из таких языков, как C/C++, Java и C#, но в сообществе Julia этот стиль разработки нередко является предпочтительным. Интерактивная среда REPL используется для экспериментирования, тестирования и отладки.

Поскольку Julia широко применяется в науке о данных, машинном обучении, математике и естественных науках, в этой книге используется много примеров, ориентированных на науку и математику, таких как вычисление значений синуса или симуляция запуска ракеты, а не на разработку веб-сайта или системы инвентаризации. В целом математика в этой книге излагается на уровне средней школы.

В этой книге вы найдете подробное описание системы множественной диспетчеризации и системы типов Julia. Указанные системы важны тем, что именно благодаря им Julia достигает высокой произ-

[1] REPL (англ. read-evaluate-print loop) – это интерактивная программа командной строки языка программирования.

водительности. Так как многие новички программирования на языке Julia в этих системах путаются, я остановлюсь на них несколько подробнее.

Поскольку в индустрии программного обеспечения по-прежнему доминируют языки объектно ориентированного программирования, переход к более функциональному стилю программирования Julia, возможно, будет дезориентировать. Поэтому я посвятил место демонстрации того, как одни и те же задачи можно решать в функциональном и объектно ориентированном стиле. На протяжении всей книги используются многие предпочтительные практики функционального программирования.

Работая с этой книгой, вы не увидите большого числа готовых приложений, то есть таких, которые запускаются при нажатии по значку приложения. Вы также не увидите созданных на языке Julia инструментов командной строки, которые можно выполнять из консоли. Этот подход будет новым, например, для разработчиков на Ruby и Python, которые очень привыкли создавать программное обеспечение в виде инструментов командной строки.

В отличие от этого подхода, сообщество разработчиков на языке Julia очень ориентировано на пакеты. Они рекомендуют создавать пакеты поверх автономных приложений, так как ими легче делиться с другими и реиспользовать. Это предпочтение отражено в цепочке инструментов Julia и менеджере пакетов. Julia вовсе не мешает создавать приложения, но эта книга поможет вам принять образ мыслей, ориентированный в первую очередь на пакеты. Сначала создавать пакет, а затем превращать его в приложение.

Пакетно ориентированное мышление заметно в том, как доставляются инструменты Julia. Работа с менеджером пакетов и отладчиком сводится к загрузке определенных пакетов в интерактивную среду Julia и исполнению команд в ней, а не в оболочке. Этот подход к работе, возможно, будет знаком пользователям MATLAB и R. В этих двух языках основное внимание обычно уделяется пакетам, а не приложениям.

Пользующийся языком Julia типичный статистик, ученый или аналитик данных может загружать предпочитаемые пакеты в свою среду Julia и выполнять команды Julia, не кликая по какому-либо приложению, созданному на языке Julia. Интерактивная среда REPL обычно является неотъемлемой частью большинства рабочих процессов языка Julia.

Резюме

- Статическая типизация упрощает создание высокопроизводительных языков программирования и отлавливание ошибок типов до выполнения программы.

- Динамическая типизация позволяет создавать языки интерактивного программирования. В этом преимущество программирования, требующего быстрой итеративной обработки.
- Разработка научного исходного кода нередко требует возможности легко экспериментировать с большими наборами данных. А это требует возможностей интерактивного программирования, предлагаемого динамически типизированными языками.
- Научный исходный код часто требует высокой производительности, обеспечить которую динамически типизированные языки обычно не способны.
- Julia решает проблему двух языков, предлагая высокопроизводительный динамически типизированый язык. Эта способность помогает внедрять Julia в областях, требующих высокой производительности, таких как климатическое моделирование, астрономия и макроэкономические симуляции.
- Julia не ограничивается наукой: он также является отличным языком общецелевого программирования.