Отзывы читателей книги «Семь языков за семь недель»

Знание множества парадигм оказывает существенное влияние на наши подходы к проектированию, поэтому я всегда нахожусь в поиске хороших книг, которые помогали бы мне в изучении этих парадигм. Данная книга объединяет в себе описание основных парадигм программирования. Брюс имеет богатый опыт изучения и использования множества языков. Теперь и вы сможете воспользоваться его опытом благодаря этой книге. Я настоятельно рекомендую прочитать ее.

– Доктор *Венкат Субраманиам* (Dr. Venkat Subramaniam)
Титулованный автор и основатель Agile Developer, Inc.

Важность изучения новых языков, парадигм и технологий программирования для программиста невозможно преувеличить. Эта книга решает задачу знакомства с семью языками программирования поразительно кратким и необычным способом, показывая сильные их стороны и обосновывая необходимость их изучения. Эта книга послужит отличным введением для всех программистов, желающих расширить свои горизонты или оценить новые для них языки программирования, прежде чем приступать к детальному изучению.

– *Антонио Каньяно* (Antonio Cangiano)
Программист и технический популяризатор, IBM

Пристегните ремни, потому что поездка будет очень быстрой. Эта книга наполнена быстрыми переходами от одного языка программирования к другому. Брюсу удалось выстроить все в один ряд и в результате получить интереснейшую книгу, которая доставит немало удовольствий увлеченным программистам. Если вы любите осваивать новые языки или любите блеснуть своей эрудицией, если вы хотите подняться на новый уровень – эта книга для вас. Она не разочарует вас.

– *Фредерик Дауд* (Frederic Daoud)
Автор книг «Stripes ...and Java Web Development Is Fun Again»
и «Getting Started with Apache Click»

Хотите выбрать свой «язык года» из великолепной семерки? Хотите упорядочить свои представления о программировании в целом? Тогда вы уже нашли то, что вам нужно! Лично я на время чтения этой книги вернулся в свои студенческие годы, когда предпринимал первые попытки проложить курс через свои языки программирования и посто-

янно натыкался на мели. Разница лишь в том, что Брюс не даст вам сбиться с верного курса! Эта книга не предполагает неторопливого чтения – вы вынуждены будете активно работать с ней. Я уверен, что вы найдете ее чрезвычайно увлекательной и весьма практичной.

– *Мэтт Стайн* (Matt Stine)
Руководитель отдела разработки программного обеспечения
в детской больнице святого апостола Иуды Фаддея
(St. Jude Children's Research Hospital), США

На протяжении почти своей учебы в университете по направлению информатики я не хотел быть программистом, но так или иначе стал им. Книга «Семь языков за семь недель» изменила мои взгляды на многие проблемы и напомнила мне, что я все-таки люблю программировать.

– *Трэвис Каспар* (Travis Kaspar)
Программист в Northrop Grumman

Вот уже более 25 лет я занимаюсь созданием программ для разных аппаратных и программных платформ. После прочтения книги «Семь языков за семь недель» я начал понимать принципы оценки достоинств и недостатков языков программирования. Но самое важное, что я почувствовал, по каким критериям выбирать языки для решения разных задач.

– *Крис Капплер* (Chris Kappler)
Старший научный сотрудник, Raytheon BBN Technologies

Содержание

Предисловие

Из еще не написанной книги «How Proust Can Make You a Better Programmer»

Джо Армстронг (Joe Armstrong), создатель языка Erlang

– Редактор Gmail неправильно воспринимает типографские кавычки.

– Печально, – сказала Марджери, – это признак безграмотности программиста и упадка культуры.

– Что будем делать с этим?

– Мы должны настоять, чтобы следующий программист, которого наймем, обязательно прочитал роман «A la recherche du temps perdu»[1].

– Все семь томов?

– Все семь томов.

– Это поможет ему лучше узнать правила пунктуации и правильно интерпретировать кавычки?

– Необязательно, но этот роман сделает его лучшим программистом, потому что в нем он найдет многие секреты мастерства...

Обучение программированию сродни обучению плаванию. Никакая теория не заменит практических занятий, когда обучаемый молотит руками и ногами по воде, судорожно хватая ртом воздух. Впервые погрузившись в воду с головой, вы паникуете, но когда вы качаетесь на поверхности, неторопливо вдыхая воздух, вы чувствуете легкий восторг. Вы думаете: «Я могу плавать!» По крайней мере, именно эта мысль пришла мне в голову, когда я научился плавать.

То же самое происходит, когда вы обучаетесь программированию. Первые шаги даются с большим трудом, и вам нужен хороший тренер, который поможет вам прыгнуть в воду.

Брюс Тейт (Bruce Tate) – именно такой тренер. Эта книга поможет вам сделать самый первый и трудный шаг в обучении программированию.

Предположим, что вы миновали первый сложный этап загрузки и установки интерпретатора или компилятора языка, интересного вам. Что делать дальше? Какую программу написать первой?

---

[1] Пруст М. В поисках утраченного времени. – М.: АСТ, 2011. ISBN: 978-5-17-067438-1. – *Прим. перев.*

Брюс четко и обстоятельно отвечает на этот вопрос. Просто вводите программы и их фрагменты, предлагаемые в книге, и наблюдайте получаемые результаты. Не думайте пока о создании собственной программы – просто пробуйте воспроизводить примеры из книги. Набравшись опыта, вы сможете приступить к созданию собственного проекта.

Первый шаг в обретении новых знаний заключается не в попытке создать что-то свое, а в повторении того, что было уже создано другими. Это самый быстрый способ овладеть навыками программирования.

Первые шаги в изучении нового языка программирования обычно заключаются не в глубоком проникновении в его философию, а в знакомстве с особенностями расстановки точек с запятой и запятых и исследовании сообщений, которые выдает система, когда вы допускаете ошибки. И только когда вы преодолеете первый этап, научившись вводить программы без ошибок и компилировать их, можно начинать задумываться о назначении различных языковых конструкций.

Пройдя первый этап механистического ввода программы и ее запуска, можно откинуться на спинку кресла и расслабиться. Все остальное за вас сделает ваше подсознание. В то время как сознание будет запоминать правила расстановки точек с запятой, подсознание будет стараться проникнуть в суть программных конструкций. И однажды вы обнаружите, что понимаете логику программы и роль данных конструкций в данном языке.

Поверхностное знакомство со многими языками весьма полезно. Я часто обнаруживаю, что знакомство с Python или Ruby помогает мне решить конкретную задачу. Программы, которые я загружаю из Интернета, часто написаны на разных языках, и мне нередко требуется внести в них небольшие изменения перед использованием.

Каждый язык имеет свой набор идиом, свои достоинства и недостатки. Изучив несколько разных языков программирования, вы сможете выбирать тот язык, который лучше подходит для решения текущей задачи.

Мне приятно видеть, что Брюс обладает разнообразием вкусов в отношении языков программирования. Он охватывает не только широко известные языки, такие как Ruby, но также менее распространенные и недооцененные, такие как Io. В конечном счете программирование – это понимание, а понимание – это познание идей. Соответственно, открытость новым идеям дает более глубокое понимание программирования вообще.

Гуру может сказать, что для более глубокого познания математики необходимо изучить латынь. Так же и с программированием. Чтобы полнее понимать объектно-ориентированное программирование, следует изучить логическое или функциональное программирование. Чтобы полнее понимать функциональное программирование, следует изучить программирование на языке Ассемблера.

Когда я еще только учился программированию, большой популярностью пользовались книги, проводящие сравнительный анализ разных языков программирования, но большинство из них были чересчур академичны и почти не несли практических рекомендаций по использованию сравниваемых языков. Такой подход был характерен для того времени. Вы могли многое узнать об идеях, положенных в основу языка, но не имели возможности опробовать их на практике.

В настоящее время мы можем не только знакомиться с идеями, но и пытаться применять их на практике. Это большая разница, как между стоянием на краю бассейна с мыслью «а хорошо ли мне будет в воде» и получением удовольствия от плавания.

Я настоятельно рекомендую прочитать эту книгу и надеюсь, что вам она понравится так же, как и мне.

Джо Армстронг (Joe Armstrong),
создатель языка Erlang
2 марта 2010
Стокгольм

Глава 1

Введение

Люди изучают иностранные языки по самым разным причинам. Первый язык изучается, чтобы жить. Знание родного языка помогает общаться с окружающими. Побудительные мотивы к изучению второго языка у разных людей могут быть разными. Кому-то знание иностранного языка может пригодиться для построения карьеры или для адаптации в новом окружении при смене места жительства. Но иногда решение изучить новый язык принимается не потому, что он необходим, а просто ради желания учиться. Знакомство со вторым языком может расширить кругозор. Кому-то знание каждого нового языка помогает сформировать новый способ мышления.

То же относится и к языкам программирования. В этой книге я представлю вам семь разных языков. Моя цель вовсе не в том, чтобы заставить вас, как это делают многие мамы, заставляя своих чад выпить утром ложку рыбьего жира. Я хочу пригласить вас в увлекательное путешествие, которое изменит ваши взгляды на программирование. Я не буду делать из вас экспертов, но я расскажу вам чуть больше, чем требуется, чтобы написать программу «Hello, World».

1.1. Логика описания

Часто, приступая к изучению нового языка или фреймворка, я стараюсь найти в Интернете интерактивное руководство, чтобы опробовать возможности языка в управляемом окружении. Я могу написать свой сценарий, чтобы заняться дальнейшими исследованиями, но обычно я ищу информацию, которая быстро разбудит во мне интерес, пример синтаксического сахара и описание базовых концепций.

Однако обычно для меня этого недостаточно. Если я встретил новый язык, являющийся более чем тонкой оберткой вокруг уже известного мне языка, мне потребуется более глубокое его исследование. Эта книга даст вам такую возможность целых семь раз. Здесь вы найдете ответы на следующие вопросы:

- *Поддерживаемая модель типов данных.* Строгая (как в Java) или слабая (как в C), статическая (как в Java) или динамическая (как в Ruby). Во всех языках, описываемых в этой книге, используются модели со строгим контролем типов, но среди них вы встретите и статические, и динамические разновидности. Вы увидите, какое влияние на разработчика оказывает модель типов и как она сказывается на способах решения задач. Все языки, представленные в этой книге, имеют модели типов со своими неповторимыми особенностями.
- *Модель программирования.* Объектно-ориентированная (OO), функциональная, процедурная или смешанная? Языки программирования, описываемые в этой книге, поддерживают четыре разные модели программирования, а некоторые – даже комбинации сразу нескольких из них. Здесь вы познакомитесь с языком логического программирования (Prolog), двумя языками, полностью поддерживающими объектно-ориентированные концепции (Ruby, Scala), четырьмя языками, имеющими функциональную природу (Scala, Erlang, Clojure, Haskell), и одним языком, основанном на использовании прототипов (Io). Некоторые языки, такие как Scala, поддерживают сразу несколько парадигм. Мультиметоды в Clojure позволят вам даже реализовать собственную парадигму. Знакомство с новыми парадигмами программирования является одной из важнейших особенностей этой книги.
- *Тип языка.* Компилирующий или интерпретирующий, наличие виртуальной машины. При исследовании языков в данной книге я буду использовать интерактивные оболочки, если таковые имеются. Программный код больших проектов я всегда сохраняю в файлах. Но мы не будем заниматься созданием проектов, достаточно больших, чтобы полностью раскрыть возможности модели пакетов.
- *Конструкции принятия решений и базовые структуры данных.* Возможно, вас удивит, что существует множество языков программирования, в которых поддерживается возможность принятия решений без операторов `if` и `while` и их разновидностей. Вы познакомитесь с приемом сопоставления с образцом в языке Erlang и унификации в языке Prolog. Коллекции играют важную роль практически в любом языке. В одних языках, таких как Smalltalk и Lisp, коллекции являются определяющими характеристиками языка. В других, таких как C++ и Java,

напротив, коллекции организованы нелогично, непоследовательно. Но в любом случае мы обязательно будем знакомиться с поддержкой коллекций.

- *Основные особенности языка, придающие ему уникальность.* Некоторые из представленных языков поддерживают дополнительные возможности, упрощающие программирование параллельных вычислений. Другие предоставляют уникальные высокоуровневые конструкции, такие как макросы в Clojure или средства интерпретации сообщений в Io. Третьи основаны на суперзаряженной виртуальной машине, такой как BEAM в Erlang, благодаря которой Erlang позволяет создавать распределенные отказоустойчивые системы намного быстрее, чем любые другие языки. Некоторые языки поддерживают модели программирования, сфокусированные на решении узкого круга задач, например использование логики для поиска решения в рамках имеющихся ограничений.

Прочитав эту книгу, вы не станете экспертом ни в одном из рассматриваемых здесь языков, но вы *будете* знать об их уникальных особенностях и возможностях. А теперь пройдемся по списку языков.

1.2. Языки

Выбор языков для этой книги был сделан намного проще, чем вам могло бы показаться. Я просто прислушался к пожеланиям потенциальных читателей. К концу голосования имелось восемь кандидатов. Я выбросил из списка JavaScript, потому что он и так пользуется большой популярностью, и заменил его следующим по популярности языком на основе прототипов – Io. Я также убрал Python, потому что я хотел включить в книгу не более одного объектно-ориентированного языка, а выше в списке уже стоял язык Ruby. Благодаря этому в списке освободилось место для неожиданного кандидата, языка Prolog, который вошел в первую десятку. Ниже перечислены языки, которые я выбрал, и перечислены причины, почему я это сделал:

- *Ruby.* Этот объектно-ориентированный язык высоко ценится за простоту использования и удобочитаемость синтаксиса. Первоначально я вообще не хотел включать в книгу объектно-ориентированные языки, но мне хотелось сравнить другие парадигмы программирования с объектно-ориентированной, поэтому было решено включить хотя бы один объектно-ориен-

тированный язык. Кроме того, я хотел бы погрузиться в Ruby чуть глубже, чем многие программисты, и познакомить читателей с основными решениями, повлиявшими на дизайн языка. Также я посчитал нужным окунуться в метапрограммирование на Ruby, позволяющее расширять синтаксис языка. В целом я остался доволен результатом.

- *Io*. Как и Prolog, язык Io – один из наиболее противоречивых языков, включенных в книгу. Он не добился коммерческого успеха, но его конструкции параллельного программирования и однородный синтаксис заслуживают большего внимания. Минимальный синтаксис обладает широкими возможностями, а сходства с языком Lisp иногда прямо-таки бросаются в глаза. Io не имеет обширной родословной, основан на прототипах, подобно JavaScript, и поддерживает уникальный механизм управления сообщениями, познакомиться с которым вам будет очень интересно.
- *Prolog*. Да, я знаю, это очень старый язык, но я также знаю, что он чрезвычайно мощный. Реализация игры судоку на Prolog открыла мне глаза на этот язык. То, что на Prolog получалось почти играючи, на Java или C требовало приложения серьезных усилий. Джо Армстронг (Joe Armstrong), создатель языка Erlang, помог мне более взвешенно оценить этот язык, оказавший немалое влияние на Erlang. Если прежде вам не приходилось сталкиваться с языком Prolog, я думаю, что вы будете приятно удивлены.
- *Scala*. Один из языков нового поколения, основанных на виртуальной машине Java. Язык Scala привнес мощные функциональные концепции в экосистему Java. Он также поддерживает парадигму объектно-ориентированного программирования. Оглядываясь назад, я вижу поразительное сходство с C++, который способствовал сближению процедурного и объектно-ориентированного программирования. По мере знакомства с сообществом Scala вы поймете, почему программистами, исповедующими исключительно функциональный стиль, Scala воспринимается как ересь, а разработчиками на Java – как благословение.
- *Erlang*. Один из старейших языков в этом списке, Erlang позиционируется как функциональный язык, обладающий средствами поддержки параллельных вычислений, а также создания распределенных и отказоустойчивых систем. Создатели

CouchDB, одной из новейших облачных баз данных, выбрали язык Erlang и никогда не жалели о своем выборе. Потратив совсем немного времени на этот язык, вы поймете – почему. Erlang значительно упрощает разработку параллельных, распределенных и отказоустойчивых приложений.

- *Clojure*. Еще один язык на основе JVM. Это – диалект Lisp, реализующий механизмы параллельных вычислений, в корне отличающиеся от тех, что мы привыкли использовать в JVM. Это единственный язык в книге, использующий ту же стратегию версионирования, что применяется в системах управления базами данных для поддержки многозадачного доступа. Будучи диалектом Lisp, язык Clojure обладает огромной мощностью и поддерживает, возможно, самую гибкую модель программирования из всех языков, рассматриваемых в этой книге. Но, в отличие от других диалектов Lisp, он существенно уменьшает количество круглых скобок в исходном коде и опирается на огромную экосистему, включая гигантскую библиотеку Java и повсеместно доступные платформы развертывания.
- *Haskell*. Этот язык является единственным исключительно функциональным языком программирования из числа рассматриваемых в книге. Это означает, что в нем полностью отсутствуют изменяемые переменные. Любая функция на этом языке, вызванная с одними и теми же параметрами, всегда будет возвращать одно и то же значение. Из всех строго типизированных языков Haskell поддерживает самую обширную модель типов. Как и в случае с языком Prolog, потребуется некоторое время, чтобы научиться писать на этом языке, но результат стоит того.

Прошу прощения, если ваш любимый язык не попал в этот список. Я уже получил массу гневных писем по этому поводу. Мы выдвинули на голосование, упоминавшееся выше, несколько десятков языков. Я выбрал для освещения в книге не самые популярные языки, но каждый из них обладает уникальными особенностями, знакомство с которыми пойдет вам на пользу.

1.3. Купите эту книгу

...если вы опытный программист и желаете расширить свой кругозор. Это утверждение может показаться туманным, но не судите меня строго.

Учитесь учиться

Дейв Томас (Dave Thomas) – один из основателей этого издательства – каждый год обучал новым языкам программирования тысячи студентов. В самом худшем случае, изучая новый язык программирования, вы научитесь вкладывать новые концепции в свой код на своем языке.

Работа над этой книга оказала существенное влияние на код, который я пишу на Ruby. Он стал более функциональным, более удобочитаемым, и в нем стало меньше повторяющихся фрагментов. Я реже использую изменяемые переменные и успешнее справляюсь с задачами, применяя блоки кода и функции высшего порядка. Я также использую некоторые приемы, необычные для Ruby, но они делают мой код более кратким и выразительным.

В лучшем же случае вы начнете новую карьеру. Каждые десять лет происходит смена парадигмы программирования. Когда язык Java стал слишком тесным для меня, я начал экспериментировать с Ruby, чтобы лучше понять используемые в нем подходы к разработке вебприложений. После нескольких успешных побочных проектов я продолжил свою карьеру в этом направлении и никогда не жалел об этом. Моя карьера программиста на Ruby начиналась с простых экспериментов и выросла в то, что выросла.

Где получить помощь в трудный момент

Многие читатели этой книги слишком молоды, чтобы помнить времена, когда в последний раз произошла смена парадигм. Переход на объектно-ориентированную парадигму потерпел несколько неудачных попыток, но беда в том, что старая парадигма структурного программирования просто не справлялась со все возрастающей сложностью требований, предъявляемых к современному программному обеспечению. Успех языка Java послужил серьезным толчком в этом направлении, и новая парадигма заработала. Многие разработчики вынуждены были полностью менять свои навыки, образ мышления, используемые инструменты и подходы к проектированию приложений.

Возможно, мы находимся уже на полпути к следующей трансформации. Но на этот раз толчком к ней будет служить изменение архитектуры компьютеров. Пять языков из семи в этой книге реализуют превосходные модели параллельных вычислений. (Исключениями являются Ruby и Prolog.) Не важно, собираетесь вы поменять свой

язык программирования в ближайшее время или нет, я все же рискну предположить, что некоторые языки из этой книги покажутся вам весьма привлекательными. Попробуйте отложенные операции в Io, акторы в Scala или философию Erlang «позвольте приложению потерпеть неудачу». Разберитесь с тем, как программисты на Haskell обходятся без изменяемых переменных или как Clojure использует механизм версионирования для решения проблем, связанных с многозадачностью.

Интересные решения на языке Erlang можно также найти в некоторых облачных базах данных. Доктор Джо Армстронг (Joe Armstrong) создал этот язык на основе языка Prolog.

1.4. Не покупайте эту книгу

...пока не прочтете этот раздел и не согласитесь со мной. Я собираюсь заключить соглашение с вами. Вы должны согласиться с тем, что основное внимание будет уделяться самим языкам программирования, а не тонкостям их установки. Со своей стороны я постараюсь рассказать вам как можно больше за короткое время. В вашем распоряжении имеется поисковая система Google, поэтому вы не должны полагаться на мою помощь в установке, но когда вы прочтете книгу, объем ваших знаний увеличится, потому что я собираюсь копать достаточно глубоко.

Имейте также в виду, что семь языков – это достаточно честолюбивая цель для нас обоих. Как читателю вам придется настроить свой мозг на знакомство с семью разными видами синтаксиса, четырьмя парадигмами программирования, четырьмя десятилетиями разработки языков и многим другим. Как автор я должен буду охватить широкий спектр тем. Я изучил некоторые из этих языков в процессе работы над данной книгой, и, чтобы охватить наиболее важные особенности каждого языка, мне придется пойти на некоторые упрощения.

Здесь рассказывается не только о синтаксисе, но и о многом другом

Чтобы по-настоящему понять замыслы разработчиков языков, у вас должно быть желание пойти дальше знакомства с основами их синтаксиса. То есть вам придется писать программы посложнее, чем «Hello, World» или даже вычисление чисел Фибоначчи. При знакомстве с языком Ruby вы займетесь метапрограммированием. При

знакомстве с Prolog напишете программу для игры в судоку. А при знакомстве с Erlang – программу мониторинга, которая будет определять момент завершения другого процесса и запускать третий или сообщать пользователю об ошибке.

Своим решением пойти дальше освещения основ я беру на себя обязательство и предлагаю соглашение. Обязательство: я не буду ограничиваться поверхностным освещением. И соглашение: я хочу, чтобы вы признали, что я не в состоянии охватить все основы, которые можно найти в специализированных книгах. Я редко использую механизм обработки исключений, кроме случаев, когда это является основополагающей особенностью языка. Я не буду углубляться в тонкости моделей пакетов, потому что мы будем иметь дело с очень маленькими проектами, и эти знания нам не потребуются. Я не буду заниматься описанием примитивов, не нужных для решения простых задач, которые я буду ставить перед вами.

Здесь не описывается порядок установки

Платформа – вот одна из самых больших моих проблем. У меня был опыт непосредственного общения с читателями различных книг, использующими три разные версии Windows, Mac OS X и как минимум пять разных версий Unix. Еще я видел множество комментариев в различных форумах, где упоминается еще большее разнообразие платформ. Освещение семи языков на семи платформах – это практически неодолимая тема для одного автора и, может быть, даже для нескольких авторов. Я не в состоянии описать установку семи языков, поэтому я не буду даже пытаться сделать это.

Я надеюсь, что вас не интересует устаревшее руководство по установке. Языки и платформы постоянно развиваются. Я скажу вам, куда обратиться, чтобы узнать, как установить язык, и назову номер версии, которую я использую. Используя эту информацию, вы сможете найти свежие инструкции по установке из тех же источников, что и все остальные. Я не смогу дать здесь исчерпывающие инструкции по установке.

Это не справочник по программированию

Я старался создать достаточно полный обзор языков программирования. В некоторых случаях мне даже удалось получить интервью у их создателей. Я уверен, что эта книга очень хорошо передает дух каждого рассматриваемого в ней языка. Тем не менее постарайтесь понять, что я не в состоянии написать исчерпывающие руководства по

использованию всех семи языков. В этом смысле я хотел бы провести аналогию с языками человеческого общения.

Знание языка на уровне, достаточном для туриста, значительно отличается от знания языка на уровне его носителя. Я бегло говорю на английском и с запинками на испанском. Я знаю несколько фраз еще на трех языках. Я смогу заказать рыбу в Японии. Я смогу спросить, как пройти в туалетную комнату в Италии. Но я знаю свои границы. С точки зрения программирования, я бегло говорю на Basic, C, C++, Java, C#, JavaScript, Ruby и нескольких других языках. Я говорю с запинками на десятках других языков, включая и языки, охватываемые этой книгой. Я не являюсь квалифицированным специалистом по шести языкам из этого списка. Последние пять лет я пишу почти исключительно на Ruby. Но я не смогу рассказать вам, как написать вебсервер на Io или базу данных на Erlang.

Я потерпел бы неудачу, если попытался бы написать исчерпывающий справочник по каждому из этих языков. Я мог бы написать отдельное руководство по программированию на любом из языков, описываемых здесь, но оно получилось бы по объему ничуть не меньше, чем эта книга. Я представлю вашему вниманию достаточно начальной информации. Я покажу вам примеры программ на каждом из языков. Я тщательно проверю все мои примеры, чтобы они компилировались и выполнялись без сучка и задоринки. Но я не смогу помочь вам в ваших попытках приступить к самостоятельному программированию, даже если захотел бы.

Все языки в этом списке имеют исключительно доброжелательные сообщества. И это был один из критериев отбора. Каждый пример я постараюсь сопроводить разделом, в котором буду просить вас найти дополнительные ресурсы. Сделано это преднамеренно – попробовав несколько раз выполнить поиск самостоятельно, вы будете чувствовать себя гораздо увереннее.

Я буду постоянно подталкивать вас

В этой книге я собираюсь пойти чуть дальше, чем в двадцатиминутном руководстве. Вы не хуже меня знакомы с Google и сможете без труда найти простые примеры для каждого из обсуждаемых языков. Я проведу для вас экскурс в каждый из языков. Буду давать вам решать небольшие задачи и по одному программному проекту каждую неделю. Это будет непросто, зато интересно.

Если просто читать эту книгу, вы лишь познакомитесь с несколькими разновидностями синтаксиса, и не более. Если вы будете искать

в Интернете ответы на вопросы, прежде чем опробовать примеры у себя, то наверняка потерпите неудачу. Вы должны сначала попробовать выполнить примеры, понимая, что в некоторых случаях будете ошибаться. Синтаксис всегда дается проще, чем внутренняя философия языка.

Если после прочтения этого описания вы почувствовали неуверенность, я предлагаю выбрать другую книгу, а эту положить обратно на полку – она едва ли вам понравится. Возможно, вам лучше подойдут семь отдельных книг по программированию. Но если вас захватила перспектива быстро окунуться в программный код, тогда давайте двигаться дальше.

1.5. Заключительное замечание

В этой точке мне хотелось написать что-нибудь ободряющее, но все сводится к единственному слову.

Развлекайтесь!