Содержание

От издательства

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. Издательство «ДМК Пресс» очень серьезно относится 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 авторах

Кати Фислер (Kathi Fisler)

научный сотрудник-исследователь в области информационных технологий в университете Брауна (Провиденс, Род-Айленд, США)

Заместитель директора программы бакалавриата по информационным технологиям (ИТ).

Содиректор программы Bootstrap.

Кати интересуют различные аспекты того, как люди изучают и используют формальные системы. В настоящее время она занимается компьютерным образованием, где рассматриваются модели и представления для объяснения поведения программы (воображаемых машин) и способы использования противоположностей между конкретными примерами для обучения концепциям ИТ. Кати разрабатывает курс (и учебник) по обучению ИТ, ориентированному на данные (наука о данных + структуры данных). Кати также является одним из преподавателей, занимающихся разработкой и оценкой усилий нашей кафедры по интеграции социально ответственного применения ИТ на протяжении всех четырех лет нашей учебной программы по ИТ. Ранее Кати участвовала в разработке схематической логики для проектирования оборудования (конец 1990-х гг.), модульной верификацией функционально ориентированных программ (начало 2000-х гг.) и анализом политики управления доступом и конфиденциальности (середина–конец 2000-х гг.). В этих проектах упор делался на формальные системы, а не на человеческое мышление.

Шрирам Кришнамурти (Shriram Krishnamurthi)

профессор информатики в университете Брауна (Провиденс, Род-Айленд, США)

Основные области исследования (сам Шрирам называет их «скорее исследовательскими взглядами/представлениями»):

- абстракции необходимы для прогресса в информационных технологиях;
- но абстракции могут быть трудными для понимания и изучения;
- тем не менее абстракции еще и прекрасны;
- **как мы можем помочь людям эффективно изучать абстракции?**

Главная цель состоит в том, чтобы добиться прогресса в максимально возможном количестве аспектов этих представлений.

Сначала Шрирам обучался языкам программирования, но в дальнейшем изучал различные аспекты разработки программного обеспечения, формальные методы, взаимодействие человек–компьютер, безопасность и сети. На протяжении многих лет участвовал в нескольких проектах разработки инновационных и полезных программных систем: инструменты JavaScript, Flowlog, Racket (ранее DrScheme), WeScheme, Margrave, Flapjax, FrTime, Continue, FASTLINK, (Per)Mission и др. В настоящее время в основном работает с языком Pyret.

С 2016 г. Шрирам посвятил значительную часть своего времени и энергии самой сложной проблеме – компьютерным исследованиям в области образования, поскольку она требует серьезной работы как с технической точки зрения, так и с учетом человеческого фактора: если вы допустите оплошность, то можете нанести реальный ущерб не только отдельным людям, но и всей отрасли и обществу.

Шрирам занимается информационными технологиями с 1995 г. Возможно, он знаком читателям по таким книгам (написанным в соавторстве), как HtDP, PLAI, PAPL или (в настоящее время) DCIC. Участник социально ориентированной программы Bootstrap, которая используется во всем мире для внедрения информационных технологий в математику, физику, социальные науки и другие дисциплины.

Шрирам являлся заместителем директора программы Brown's Executive Master in Cybersecurity, где отвечал за курс по человеческому фактору. Новая версия программы объединена с информационными технологиями.

Шрирам является лауреатом премии SIGPLAN для молодых исследователей Робина Милнера (Robin Milner Young Researcher Award), премии SIGSOFT для авторитетных преподавателей, премии SIGPLAN в области программного обеспечения (совместно), а также Wriston Fellowship университета Брауна.

Бенджамин С. Лернер (Benjamin S. Lerner)

адъюнкт-профессор (доцент-преподаватель) ИТ Khoury Colledge в Северо-Восточном университете (Northeastern University, Бостон, Массачусетс, США)

Области научных интересов Бенджамина:

- Pyret: язык, разработанный для начального обучения программированию, с акцентом на тестирование, ясность и иногда с ужасными каламбурами из пиратского («пиретского») лексикона;
- семантика веб-программирования: современные веб-программы сочетают в себе богатые структуры данных, хитроумное выполнение на основе событий, данные, получаемые от третьих сторон, и мощные, но мелкомасштабные API. Понимание и анализ этих про-

грамм требуют сначала создания тестируемой и исполняемой семантики для каждого из данных компонентов, а затем использования соответствующей семантики для проведения анализа программы;

- обеспечение совместимости расширений веб-браузера (проект Conflicts of Interest: Interactions among Browser Extensions): рост популярности Firefox можно в значительной степени объяснить его широко разрекламированными расширениями, которые предлагают универсальность, удобство и относительно низкие кривые обучения как для любителей, так и для опытных программистов. Но вместе с такой возможностью гибкой настройки возникают проблемы: многие расширения не работают должным образом при одновременной установке. Этот проект направлен на предоставление улучшенной модели программирования для расширений, которые могут обнаруживать и, возможно, исправлять подобные конфликты до их возникновения.

Джо Гиббс Политц (Joe Gibbs Politz)

адъюнкт-профессор (доцент-преподаватель) ИТ Калифорнийского университета Сан-Диего (UC San Diego), США

Преподаватель многочисленных курсов по информационным технологиям (ИТ), разработанных с использованием повторно используемых материалов, включая видео, заметки и первоначальный исходный код для контрольных заданий:

- Advanced Compiler Design (UCSD CSE 231);
- Software Engineering (UCSD CSE 110);
- Advanced Data Structures (UCSD CSE 100);
- Basic Data Structures and Object-Oriented Design (UCSD CSE 12);
- Data Structures and Algorithms (Swarthmore CS 35);
- Programming Languages (Brown University CS173; совместно с Шрирамом Кришнамурти)

и многих других.

Часть I

ВВЕДЕНИЕ

Глава 1

Предисловие

1.1. О чем эта книга

Эта книга представляет собой введение в информатику. Она научит вас программировать такими способами, которые имеют практическую ценность и важность. Но этот процесс обучения будет к тому же выходить за рамки программирования в более широкую область информатики, богатой, глубокой, увлекательной и прекрасной многогранной дисциплины. Вы узнаете много полезных вещей, которые сможете сразу же применить на практике, но мы также продемонстрируем вам кое-что из того, что скрывается за ними.

Прежде всего мы хотим предоставить вам способы мышления для решения задач с помощью вычислений. Некоторые из этих способов являются техническими методиками, такими как обработка данных и примеры для разработки решений задач. Другие представляют собой научные методы, такие как способы, позволяющие убедиться в том, что программы надежны и делают именно то, для чего они предназначены. Наконец, некоторые из таких способов имеют социальную составляющую, они заставляют задуматься о влиянии программ на людей.

1.2. Основополагающие принципы, определяющие содержание этой книги

Наша точка зрения основана на многолетнем опыте работы в качестве разработчиков программного обеспечения, исследователей и преподавателей. Эти виды деятельности позволили нам выработать следующие твердые убеждения:

- программное обеспечение пишется не только для того, чтобы его выполнять. Программы следует писать так, чтобы их могли читать и сопровождать другие люди. Часто этим «другим» человеком являетесь вы сами, через шесть месяцев забывшие, что было сделано и почему именно так;

- программисты несут ответственность за то, чтобы созданное ими программное обеспечение соответствовало поставленным целям и являлось надежным. Это требование отражено в различных дисциплинах информатики, таких как тестирование и проверка (верификация) программ;
- программы должны быть предсказуемыми. Мы должны знать, насколько это возможно, как будет вести себя программа, до ее запуска. Это поведение включает в себя не только технические характеристики, такие как время работы, использование пространства памяти, мощность и т. п., но и социальные воздействия и последствия, выгоды и вред. Программисты, как известно, далеко не всегда задумываются о последнем.

1.3. Наш взгляд на данные

Высказанные выше опасения пересекаются с нашими представлениями о развитии информатики как отдельной дисциплины. Все прекрасно понимают, что мы живем в мире, переполненном данными, но к каким последствиям это приводит?

На вычислительном уровне влияние данных стало огромным. Обычно единственным способом сделать программу лучше было непосредственное ее усовершенствование, что часто означало усложнение и воздействие на факторы, которые мы обсуждали выше. Но есть категории программ, для которых существует другой метод: просто передать в ту же программу больше данных или данные лучшего качества, и программа, возможно, улучшится. Эти управляемые данными программы лежат в основе многих инноваций, которые мы наблюдаем в окружающем мире.

В дополнение к этому техническому влиянию данные могут иметь еще и глубокое педагогическое воздействие. В большинстве случаев процессу начального обучения программированию наносят ущерб придуманные данные, которые не имеют реального смысла, не вызывают никакого интереса и не дают никаких практических результатов (а также часто сопровождаются искусственно созданными проблемами). Имея в своем распоряжении реальные данные, обучающиеся могут самостоятельно регулировать процесс обучения, сосредоточившись на задачах, которые они считают практически целесообразными, более информативными или просто интересными, задавая вопросы и отвечая на те из них, которые они считают наиболее полезными. Разумеется, с этой точки зрения программы запрашивают данные: т. е. программы являются инструментами для ответов на вопросы. В свою очередь, особое внимание, уделяемое реальным данным и ответам на практические вопросы, позволяет нам обсуждать социальное воздействие информатики и вычислительной техники.

Эти явления породили совершенно новые области исследований, обычно называемые наукой о данных. Но типовые учебные программы по науке о данных также имеют много ограничений. Они уделяют слишком мало

внимания тому, что нам известно о трудностях обучения программированию, а также надежности программного обеспечения. И такие учебные программы абсолютно не учитывают того, что их данные часто весьма ограничены по своей структуре. На этих ограничениях обычно заканчивается наука о данных и начинается информатика. В частности, структура данных служит отправным пунктом для размышлений и реализации некоторых из вышеперечисленных основополагающих принципов – производительности, надежности и предсказуемости – с использованием многочисленных инструментальных средств информатики.

1.4. Что делает эту книгу особенной

Во-первых, мы предлагаем новую точку зрения на структурирование учебных программ по информатике, которую мы называем ориентацией на данные. Мы рассматриваем учебную программу, ориентированную на данные, как

Более подробно об этом подходе читайте в нашем эссе (https://cs.brown.edu/~sk/Publications/Papers/Published/kf-data-centric/).

ориентация на данные = наука о данных + структуры данных

именно в этом порядке: начинаем с основных принципов науки о данных, затем переходим к классическим принципам для структур данных и прочих разделов информатики. Эта книга излагает такую точку зрения конкретно и подробно.

Во-вторых, в компьютерном образовании много говорится о гипотетических машинах – абстракциях поведения программ, предназначенных для того, чтобы помочь учащимся понять, как эти программы работают, – но в действительности они используются лишь в немногих учебных программах. Мы серьезно относимся к гипотетическим машинам, разрабатывая последовательный ряд таких машин и вводя их в учебную программу. Это связано с нашим убеждением о том, что программы – это не только объекты, которые выполняются, но и объекты, которые мы подробно рассматриваем.

В-третьих, мы включаем в текст книги контент о социальной ответственности информатики. В отличие от других усилий, направленных на ознакомление обучающихся с этикой или со скрытыми потенциальными опасностями технологии в целом, мы стремимся показать обучающимся, как конструкции и концепции, которые они прямо сейчас превращают в код, могут привести к неблагоприятным последствиям, если использовать их, пренебрегая осторожностью. Сохраняя сосредоточенность на тестировании и конкретных примерах, мы вводим несколько тем, заставляющих обучающихся принимать во внимание предположения на уровне конкретных данных. Этот материал приводится в явной форме на протяжении всей книги.

Наконец, эта книга основана на результатах недавних, постоянно продолжающихся научных исследований. Выбор учебного материала, порядок его представления, методы программирования и многое другое основаны на том, что мы знаем из научно-исследовательской литературы. Во многих слу-

чаях мы сами проводим исследования, поэтому учебная программа и научные исследования образуют своеобразный симбиоз. Вы можете найти наши материалы (некоторые написаны совместно, некоторые – в сотрудничестве с другими авторами) на соответствующих страницах (https://www.ccs.neu.edu/home/blerner/papers.html).

1.5. Для кого предназначена эта книга

Эта книга написана в основном для студентов начальных курсов, изучающих информатику в высшем учебном заведении (колледж или университет). Однако многие – особенно начальные – части этой книги также подходят для среднего образования (например, в США приблизительно для 6–12 классов, или для возраста 12–18 лет). В действительности мы видим естественную преемственность между средним и высшим образованием и считаем, что эта книга может послужить полезным связующим звеном между ними.

1.6. Структура книги

В отличие от некоторых других учебных пособий, эта книга не придерживается строго иерархического порядка изложения материала. Скорее, это непрерывно продолжающееся обсуждение с возвратами к предыдущим темам. Чаще всего мы будем создавать программы постепенно, как это могла бы делать пара сотрудничающих программистов. Мы будем совершать ошибки, но не потому, что не знаем, как сделать все правильно, а потому, что для вас это самый лучший способ обучения. Наличие ошибок делает невозможным пассивное чтение: вы непременно должны глубоко осваивать материал, потому что нет никакой уверенности в достоверности предлагаемого для чтения текста.

В итоге вы всегда будете получать правильный ответ. Однако такой нелинейный путь в краткосрочной перспективе вызывает больше разочарований (у вас часто будет возникать соблазн сказать: «Просто сразу скажите мне правильный ответ!») и делает книгу плохим справочным пособием (невозможно открыть случайную страницу с уверенностью в том, что все написанное на ней правильно). Но такое чувство разочарования является правильным восприятием обучения. Другого пути мы не знаем.

Мы используем визуальное форматирование, чтобы выделить некоторые из этих моментов. Таким образом, время от времени вы будете встречать следующие выделенные фрагменты:

Упражнение

Это упражнение. Попробуйте выполнить его.

Это обычное упражнение в учебной литературе. Его нужно выполнить самостоятельно. Если вы используете эту книгу как часть учебного курса, то такое упражнение вполне можно предложить в качестве домашнего задания. Но кроме обычных упражнений вам также будут встречаться похожие на них вопросы, которые выглядят следующим образом:

> ***Выполните прямо сейчас***
> Здесь выполняется некоторое действие. Вы его видите?

Когда встречается один из таких выделенных фрагментов, остановитесь. Прочтите выделенный текст, подумайте и сформулируйте ответ, прежде чем продолжить чтение. Вы обязательно должны сделать именно так, потому что в действительности это упражнение, но ответ уже есть в книге – чаще всего в тексте, непосредственно следующем за ним (т. е. в той части, которую вы читаете прямо сейчас), – или вы можете найти ответ самостоятельно, выполнив программу. Если вы просто продолжите читать дальше, то получите ответ, не думая самостоятельно (или не увидите его вообще, если инструкции предназначены для запуска программы), но в результате (а) не сможете проверить свои знания и (б) не улучшите свою интуицию. Другими словами, в выделенных фрагментах представлены очевидные дополнительные попытки поощрения активного обучения. Но в конечном счете мы можем только поощрять стремление к активному обучению, однако его практическое применение зависит только от вас.

Специальные стратегии проектирования и разработки программ приводятся в блоках текста, выделенных следующим образом:

> **Стратегия: как это сделать...**
>
> Здесь располагается краткое описание способа сделать что-либо.

Наконец, подобным образом выделяется материал по социально ответственному применению информатики – в виде показанного ниже блока текста:

> **Ответственное применение информатики: вы учитывали, что...**
>
> Здесь размещается описание социальных опасностей, возникающих из-за необдуманного использования материала.

1.7. Организация материала книги

Книга содержит четыре части:

1) часть II – введение в программирование для начинающих, которые обучаются программированию и элементарному анализу данных.

В ней представлены основные концепции программирования посредством создания изображений и обработки таблиц, затем рассматриваются списки, деревья и написание программ, реагирующих на действия пользователя, и все это с учетом ориентации на данные. Условная гипотетическая машина в этом разделе основана на подстановке;

2) часть III – здесь рассматривается асимптотическая сложность, рекуррентные выражения и основные графовые алгоритмы;
3) часть V – рассматривается работа с изменяемыми переменными и изменяемыми структурированными данными, формируется понимание изменяемых списков и хеш-таблиц (и их обработки). В этом разделе мы переходим на язык Python. Он позволяет расширить возможности тестирования, чтобы подробно рассмотреть нюансы программ с динамическими изменениями. Условная гипотетическая машина в этой части отделяет среду именования (называемую здесь каталогом (directory)) от динамически распределяемой памяти (кучи) значений структурированных данных;
4) часть VI – здесь мы возвращаемся к темам алгоритмов, основанных на оценке состояния и на структурах данных с отслеживанием их состояния.

Эти части были тщательно разработаны, чтобы обеспечить отсутствие зависимостей между частями III и V. Это позволяет гибко планировать предложения нескольких различных типов учебных курсов. Например, реорганизовав материал этих частей, мы уже сейчас предлагаем два совершенно различных курса, которые могли бы использовать другие читатели:

- вводный курс может использовать части II и V (без части III) для представления ориентированной на данные точки зрения информатики и обучения студентов основным навыкам программирования на языке Python;
- более продвинутый курс, предполагающий, что обучающиеся уже знакомы с основами функционального программирования (например, по первым частям книги «How to Design Programs» (https://htdp.org/) («Как проектировать программы»), можно было бы начать непосредственно с части III или, возможно, с отдельных разделов части II для включения недостающего материала (например, работы с таблицами). Этот курс может быть продолжен в части V, за которой следует часть VI.

Описанные выше курсы аналогичны соответственно учебным курсам CSCI 0111 (https://cs.brown.edu/courses/csci0111/) и CSCI 0190 (https://cs.brown.edu/courses/csci0190/) в университете Брауна (Провиденс, Род-Айленд, США). На указанных здесь страницах хранятся все предыдущие экземпляры курсов, включая все задания и сопутствующие материалы. Читатели могут использовать их в своих учебных курсах.

Многие из этих курсов будут изучать студенты, которые ранее занимались программированием с сохранением состояния (на Python, Java, Scratch или других языках). По нашему опыту, большинство таких студентов получали либо весьма неполные, либо откровенно вводящие в заблуждение

объяснения и метафоры состояния (например, «переменная – это (черный) ящик»). Из-за этого они плохо понимают излагаемый здесь материал, за исключением самых элементарных основ, особенно когда они начинают изучать такие важные темы, как алиасинг (альтернативные имена переменных). В результате многие из этих студентов сочли одновременно новым и весьма познавательным надлежащее объяснение того, как на самом деле работать с состоянием, с помощью нашей искусственной гипотетической машины. По этой причине мы рекомендуем проходить этот материал медленно и внимательно.

Разумеется, мы предлагаем читателям создавать собственные комбинации из глав, входящих в вышеперечисленные части книги. Мы очень хотели бы узнать о других проектах.

1.8. Наш выбор языка программирования

Если бы мы хотели разбогатеть, то написали бы эту книгу полностью на Python. На момент написания книги Python переживает свой расцвет как язык для учебных целей (точно так же, как Java до него, C++ до этого, ранее C, а еще раньше Pascal и так далее). Вне всякого сомнения, у Python есть много привлекательных свойств, и не в последнюю очередь его присутствие в верхних позициях списков вакансий. Но Python неоднократно разочаровывал нас как отправной пункт для изучения программирования (см. главу 30).

В итоге в нашей книге используются два языка программирования. Сначала применяется язык под названием Pyret (https://www.pyret.org/), который мы специально спроектировали и создали для собственных потребностей, как следствие наших разочарований. Он был специально разработан для стиля программирования, описанного в этой книге, поэтому оба языка могут гармонично сосуществовать. Pyret основан на Python, а также на многих других превосходных языках программирования. Таким образом, начинающие программисты получат именно те знания, которые им необходимы, в то время как программистам, уже хорошо знакомым с языковым зверинцем, от змей до верблюдов, Pyret должен показаться знакомым и удобным.

Далее, учитывая значимость Python как стандартного языка обмена информацией и наличие расширяющих его возможности библиотек, в части V книги Python рассматривается весьма подробно. Вместо того чтобы начинать с нуля в Python, мы предлагаем систематический и постепенный переход к этому языку на основе ранее изученного материала. Мы считаем, что это поможет вам изучить программирование в целом лучше, чем если бы вы были знакомы только с одним языком программирования. Но в то же время мы считаем, что такой подход поможет вам лучше понять Python: подобно тому, как вы начинаете больше ценить свой язык, страну и культуру, когда покидаете ее пределы и знакомитесь с другими странами и культурами.