

Содержание

От издательства	13
Об авторах	14
Часть I. ВВЕДЕНИЕ	17
Глава 1. Предисловие	18
1.1. О чем эта книга	18
1.2. основополагающие принципы, определяющие содержание этой книги	18
1.3. Наш взгляд на данные	19
1.4. Что делает эту книгу особенной	20
1.5. Для кого предназначена эта книга	21
1.6. Структура книги	21
1.7. Организация материала книги	22
1.8. Наш выбор языка программирования	24
1.9. Обратная связь, сообщения об ошибках и комментарии	25
Глава 2. Благодарности	26
Часть II. ОСНОВЫ	28
Глава 3. Начинаем работу	29
3.1. Поясняющий пример: флаги	29
3.2. Числа	30
3.3. Выражения	32
3.4. Терминология	33
3.5. Строки	33
3.6. Изображения	34
3.6.1. Объединение изображений	35
3.6.2. Создание флага	36
3.7. Небольшое отступление: типы, ошибки и документация	37
3.7.1. Типы и контракты	37
3.7.2. Ошибки формата и нотации	39
3.7.3. Поиск других функций: документация	40
Глава 4. Именованые значения	42
4.1. Панель определений	42
4.2. Именованые значения	42
4.2.1. Сравнение имен и строк	43
4.2.2. Сравнение выражений с инструкциями	44
4.3. Каталог программы	45
4.3.1. Объяснение смысла кнопки Run	46
4.4. Использование имен для оптимизации создания изображений	48

Глава 5. От повторяющихся выражений к функциям	50
5.1. Учебный пример: похожие флаги	50
5.2. Определение функций.....	51
5.2.1. Как вычисляются функции.....	52
5.2.2. Аннотации типов	53
5.2.3. Документация	55
5.3. Практическая методика разработки функций: расчет веса на Луне.....	56
5.4. Документирование функций с примерами	57
5.5. Практическая методика разработки функций: стоимость авторучек	58
5.6. Резюме: определение функций.....	61
Глава 6. Условные и логические выражения	63
6.1. Учебный пример: вычисление стоимости доставки	63
6.2. Условные выражения: вычисления с принятием решений	64
6.3. Логические выражения.....	65
6.3.1. Другие логические операции	66
6.3.2. Объединение логических выражений	68
6.4. Как задать сразу несколько вопросов.....	68
6.5. Вычисление методом упрощения выражений.....	72
6.6. Совместное использование функций	73
6.6.1. Как вычисляются совместно используемые функции	74
6.6.2. Совместное использование функций и внутренний каталог.....	75
6.7. Вложенные условные выражения	76
6.8. Резюме: логические и условные выражения	80
Глава 7. Введение в табличные данные	82
7.1. Создание табличных данных.....	84
7.2. Извлечение значений строк и ячеек.....	86
7.3. Функции для работы со строками	88
7.4. Обработка строк таблицы	89
7.4.1. Поиск строк.....	90
7.4.2. Упорядочение строк.....	91
7.4.3. Добавление новых столбцов.....	93
7.4.4. Вычисление новых значений столбца	95
7.5. Примеры функций для создания таблиц	96
Глава 8. Обработка таблиц	98
8.1. Очистка таблиц данных.....	99
8.1.1. Загрузка таблиц данных	99
8.1.2. Обработка отсутствующих элементов.....	100
8.1.3. Нормализация данных	102
8.1.4. Нормализация, систематическое применение.....	106
8.1.4.1. Использование программ для обнаружения ошибок в данных	107
8.2. Планирование задач	108
8.3. Подготовка таблиц данных	111
8.3.1. Создание групп по категориям.....	112
8.3.2. Разделение столбцов	112
8.4. Управление и именование таблиц данных	114
8.5. Визуальные представления и графики.....	115
8.6. Резюме: управление анализом данных.....	117

Глава 9. От таблиц к спискам	119
9.1. Основные статистические вопросы.....	119
9.2. Извлечение столбца из таблицы.....	120
9.3. Объяснение смысла списков.....	121
9.3.1. Списки как анонимные данные.....	121
9.3.2. Создание литеральных списков.....	122
9.4. Операции со списками.....	123
9.4.1. Встроенные операции со списками чисел.....	123
9.4.2. Встроенные операции для любых списков.....	123
9.4.3. Небольшое отступление о соглашениях об именовании.....	124
9.4.4. Получение элементов по позиции.....	125
9.4.5. Преобразование списков.....	126
9.4.6. Резюме: краткий обзор операций для работы со списками.....	127
9.5. Лямбда: анонимные функции.....	129
9.6. Совместное использование списков и таблиц.....	130
Глава 10. Обработка списков	133
10.1. Создание списков и разделение их на части.....	133
10.2. Несколько упражнений с примерами.....	136
10.3. Структурированные задачи со скалярными ответами.....	136
10.3.1. my-len: примеры.....	136
10.3.2. my-sum: примеры.....	138
10.3.3. От примеров к исходному коду.....	139
10.4. Структурированные задачи, в которых выполняется преобразование списков.....	141
10.4.1. my-doubles: примеры и код.....	142
10.4.2. my-str-len: примеры и код.....	144
10.5. Структурированные задачи, которые выбирают элементы из списков.....	145
10.5.1. my-pos-nums: примеры и код.....	145
10.5.2. my-alternating: примеры и код.....	147
10.6. Структурированные задачи на нестрогих областях определения.....	150
10.6.1. my-max: примеры.....	150
10.6.2. my-max: от примеров к коду.....	152
10.7. Более структурированные задачи со скалярными ответами.....	154
10.7.1. my-avg: примеры.....	154
10.8. Структурированные задачи с аккумуляторами.....	156
10.8.1. my-running-sum: первая попытка.....	156
10.8.2. my-running-sum: примеры и код.....	156
10.8.3. my-alternating: примеры и код.....	158
10.9. Работа с несколькими ответами.....	159
10.9.1. uniq: постановка задачи.....	159
10.9.2. uniq: примеры.....	159
10.9.3. uniq: код.....	160
10.9.4. uniq: сокращенное вычисление.....	162
10.9.5. uniq: пример и варианты кода.....	163
10.9.6. uniq: почему создается список?.....	164
10.10. Мономорфные списки и полиморфные типы.....	164
Глава 11. Введение в структурированные данные	166
11.1. Объяснение типов сложных составных данных.....	166

11.1.1. Первый взгляд на структурированные данные	166
11.1.2. Первый взгляд на условные данные	167
11.2. Определение и создание структурированных и условных данных.....	168
11.2.1. Определение и создание структурированных данных	168
11.2.2. Аннотации для структурированных данных	169
11.2.3. Определение и создание условных данных	170
11.3. Программирование со структурированными и условными данными.....	171
11.3.1. Извлечение полей из структурированных данных	171
11.3.2. Различение вариантов условных данных	172
11.3.3. Обработка полей вариантов	173
Глава 12. Наборы структурированных данных	175
12.1. Списки как наборы данных	176
12.2. Множества как наборы данных.....	178
12.2.1. Выбор элементов из множеств	179
12.2.2. Вычисления с использованием множеств.....	180
12.3. Сочетание структурированных и объединенных в набор данных	181
12.4. Задача проектирования данных: представление опросов.....	182
Глава 13. Рекурсивные данные	185
13.1. Функции для обработки рекурсивных данных	187
13.2. Шаблон для обработки рекурсивных данных	192
Глава 14. Деревья	195
14.1. Задача проектирования данных – данные родословной	195
14.1.1. Вычисление генетических родителей по таблице родословной	196
14.1.2. Вычисление прародителей по таблице родословной.....	198
14.1.3. Создание типа данных для деревьев родословной	199
14.2. Программы для обработки деревьев родословной.....	201
14.3. Резюме: методика решения задач о деревьях	203
14.4. Учебные вопросы	203
Глава 15. Функции как данные	205
15.1. Немного математического анализа	205
15.2. Удобная сокращенная форма записи для анонимных функций	208
15.3. Потоки из функций.....	208
15.4. Объединение сил: потоки производных	214
Глава 16. Интерактивные игры как системы с обратной связью	216
16.1. Немного об анимации с обратной связью	217
16.2. Предварительные условия.....	218
16.3. Версия: самолет пересекает экран.....	218
16.3.1. Обновление состояния окружающей среды	219
16.3.2. Вывод представления состояния окружающей среды	220
16.3.3. Наблюдение за временем (и совмещение всех элементов).....	221
16.4. Версия: непрерывное циклическое движение.....	222
16.5. Версия: снижение.....	223
16.5.1. Движение самолета.....	224
16.5.2. Визуализация сцены.....	225
16.5.3. Завершающие штрихи.....	226

16.6. Версия: ответная реакция на нажатия клавиш	226
16.7. Версия: посадка	228
16.8. Версия: закрепленный воздушный шар	230
16.9. Версия: следите за топливным баком	232
16.10. Версия: воздушный шар тоже двигается	234
16.11. Версия: один, два, ... девяносто девять летающих воздушных шаров	235
Глава 17. Примеры, тестирование и проверка программ	236
17.1. От примеров к тестам	236
17.2. Улучшенные сравнения	238
17.3. Когда тесты не проходят	241
17.4. Прогнозирование тестирования	242
Часть III. АЛГОРИТМЫ	244
Глава 18. Прогнозирование роста	245
18.1. Маленькая (правдивая) история	245
18.2. Основной аналитический принцип	249
18.3. Модель стоимости для времени выполнения Puret	250
18.4. Размер входных данных	251
18.5. Табличный метод для отдельных структурированных рекурсивных функций	252
18.6. Создание рекуррентных последовательностей	254
18.7. Форма записи для функций	256
18.8. Сравнение функций	256
18.9. Объединение O-больших без проблем	258
18.10. Решение рекуррентных последовательностей	259
Глава 19. Обратимся к множествам	263
19.1. Представление множеств с помощью списков	264
19.1.1. Варианты выбора представления	264
19.1.2. Временная сложность	265
19.1.3. Выбор одного из представлений	266
19.1.4. Другие операции	268
19.2. Как заставить множества расти на деревьях	269
19.2.1. Преобразование значений в упорядоченные значения	270
19.2.2. Использование двоичных деревьев	272
19.2.3. Точный баланс: обрезка деревьев	276
19.2.3.1. Вариант левое–левое	279
19.2.3.2. Вариант левое–правое	280
19.2.3.3. Существуют ли какие-либо другие варианты?	281
Глава 20. Хэллоун-анализ	283
20.1. Первый пример	283
20.2. Новая форма анализа	283
20.3. Пример: очереди из списков	284
20.3.1. Представления в виде списка	284
20.3.2. Первоначальный анализ	285
20.3.3. Более разнообразные последовательности операций	285
20.3.4. Второй этап анализа	287

20.3.5. Сравнение амортизации с отдельными операциями	287
20.4. Материал для дополнительного чтения	287
Глава 21. Совместное использование значений и равенство	288
21.1. Новый взгляд на равенство	288
21.2. Стоимость вычисления ссылок	292
21.3. Формы записи равенства	294
21.4. В интернете никто не знает, что вы НАГ	294
21.5. НАГ был всегда	296
21.6. От ацикличности к циклам	297
Глава 22. Графы	299
22.1. Объяснение сущности графов	299
22.2. Представления	303
22.2.1. Связи по имени	303
22.2.2. Связи по индексам	305
22.2.3. Список ребер	307
22.2.4. Абстрагирующие представления	308
22.3. Измерение сложности для графов	308
22.4. Достижимость	309
22.4.1. Простая рекурсия	309
22.4.2. Приведение в порядок цикла	310
22.4.3. Проход с использованием памяти	311
22.4.4. Улучшенный интерфейс	312
22.5. Обход в глубину и в ширину	313
22.6. Графы со взвешенными ребрами	314
22.7. Наикратчайшие (или наилегчайшие) пути	315
22.8. Моравские остовные деревья	317
22.8.1. Глобальная задача	318
22.8.2. Жадное решение	318
22.8.3. Другое жадное решение	319
22.8.4. Третье решение	320
22.8.5. Проверка связности компонентов	321
Часть IV. ОТ PYRET К PYTHON	326
Глава 23. От Pyret к Python	327
23.1. Выражения, функции и типы	327
23.2. Возврат значений из функций	329
23.3. Примеры и варианты тестов	330
23.4. Небольшое отступление по поводу чисел	331
23.5. Условные выражения	333
23.6. Создание и обработка списков	333
23.6.1. Фильтры, отображения и друзья	334
23.7. Данные с компонентами	335
23.7.1. Доступ к полям внутри классов данных	336
23.8. Обход списков	336
23.8.1. Представляем циклы <code>for</code>	336
23.8.1.1. Небольшое отступление о порядке обработки элементов списка	338

23.8.2. Использование циклов <code>for</code> в функциях, создающих списки	339
23.8.3. Резюме: шаблон обработки списков для Python	340
Часть V. ПРОГРАММИРОВАНИЕ С СОХРАНЕНИЕМ СОСТОЯНИЯ	341
Глава 24. Изменение структурированных данных	342
24.1. Изменение полей структурированных данных	343
24.2. Изменение совместно используемых данных	344
24.3. Объяснение функционирования памяти	346
24.4. Переменные и равенство	347
24.5. Хранение простых данных в памяти	348
Глава 25. Изменение переменных	350
25.1. Изменение переменных в памяти	350
25.2. Изменение переменных, связанных со списками	354
25.3. Создание функций, изменяющих переменные	355
25.3.1. Аннотация <code>global</code>	356
25.4. Тестирование функций, изменяющих глобальные переменные	357
25.4.1. Внутренняя структура функции тестирования	361
25.4.2. Общие выводы о тестировании изменений	361
Глава 26. Возврат к спискам и переменным	363
26.1. Обновление совместно используемого списка	363
26.1.1. Операции, изменяющие списки	364
26.2. Списки в памяти	365
26.3. Практический пример: данные для совместно используемых банковских счетов	367
26.4. Циклические ссылки	371
26.4.1. Тестирование циклических данных	373
26.4.2. Снова переменные: функция для создания счетов для новых клиентов	373
26.5. Многочисленные роли переменных	374
26.6. Управление всеми счетами	375
Глава 27. Хеш-таблицы и словари	377
27.1. Поиск по условиям, отличающимся от ключей	378
27.2. Словари с более сложными значениями	379
27.3. Использование структурированных данных в качестве ключей	380
Часть VI. ДОПОЛНИТЕЛЬНЫЕ ТЕМЫ	382
Глава 28. Алгоритмы, использующие состояние	383
28.1. И снова о непересекающихся множествах	383
28.1.1. Оптимизации	384
28.1.2. Анализ	385
28.2. Установка членства методом обратного хеширования	386
28.2.1. Улучшение времени доступа	387
28.2.2. Улучшенное хеширование	389
28.2.3. Фильтры Блума	389

28.3. Устранение повторных вычислений с помощью запоминания ответов	391
28.3.1. Интересная числовая последовательность	391
28.3.1.1. Использование состояния для запоминания предыдущих ответов	393
28.3.1.2. От дерева вычислений к НАГ	394
28.3.1.3. Сложность чисел	395
28.3.1.4. Абстрагирование мемоизации	395
28.3.2. Редакторское расстояние для исправления орфографических ошибок	397
28.3.3. Природа как машинистка с неловкими пальцами	402
28.3.4. Динамическое программирование	403
28.3.4.1. Числа Каталана и динамическое программирование.....	404
28.3.4.2. Расстояние Левенштейна и динамическое программирование	405
28.3.5. Сравнение мемоизации и динамического программирования.....	408
Часть VII. ПРИЛОЖЕНИЯ	411
Глава 29. Pyret для пользователей Racket и Scheme	412
29.1. Числа, строки и логические значения	412
29.2. Инфиксные выражения	413
29.3. Определение и применение функций	413
29.4. Тесты	414
29.5. Имена переменных	415
29.6. Определения данных	415
29.7. Условные выражения.....	417
29.8. Списки.....	419
29.9. Функции первого класса.....	420
29.10. Аннотации	420
29.11. Что еще?	420
Глава 30. Сравнение Pyret с Python	421
Глава 31. Сравнение этой книги с книгой «Как проектировать программы» (HtDP)	424
Глава 32. Примечания к текущей редакции книги	427
Глава 33. Словарь терминов	428
Предметный указатель	431

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 Рурет.

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 *HiDP*, *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:

- Рурет: 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 Калифорнского 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 нелнейный 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 получают 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.

Конец ознакомительного фрагмента.

Приобрести книгу можно

в интернет-магазине

«Электронный универс»

e-Univers.ru