

The Programmer's Brain

WHAT EVERY PROGRAMMER NEEDS TO KNOW ABOUT COGNITION

FELIENNE HERMANS

FOREWORD BY JON SKEET



MANNING

SHELTER ISLAND

УМ ПРОГРАММИСТА КАК ПОНЯТЬ И ОСМЫСЛИТЬ ЛЮБОЙ КОД

Фелин Херманс

Предисловие Джона Скита

Санкт-Петербург

«БХВ-Петербург»

2023

УДК 004.4
ББК 32.973.26-02
X39

Херманс Ф.

X39 Ум программиста. Как понять и осмыслить любой код: Пер. с англ. — СПб.: БХВ-Петербург, 2023. — 272 с.: ил.

ISBN 978-5-9775-1176-6

Книга освещает практические основы когнитивистики для программистов. Основные темы: осмысление и развитие чужого и собственного кода, изучение новых языков программирования, мнемонические приемы для программистов, поддержка кода в читаемом состоянии. Объяснено, как снижать когнитивную нагрузку при работе программиста, как делать код логичным и понятным для себя и коллег. Рассмотрены приемы именования функций, классов и переменных, подходы к ведению репозитория, совместной разработке и доработке кода.

Для программистов и других IT-специалистов

УДК 004.4
ББК 32.973.26-02

Группа подготовки издания:

Руководитель проекта	<i>Олег Сивченко</i>
Зав. редакцией	<i>Людмила Гауль</i>
Перевод с английского	<i>Кристины Черниковой</i>
Компьютерная верстка	<i>Ольги Сергиенко</i>
Оформление обложки	<i>Зои Канторович</i>

Original English language edition published by Manning Publications.
Copyright (c) 2021 by Manning Publications.
Russian-language edition copyright (c) 2022 by BHV. All rights reserved.

Оригинальное издание на английском языке опубликовано Manning Publications.
© 2021 Manning Publications.
Издание на русском языке © 2022 ООО «БХВ». Все права защищены.

"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20

ISBN 978-1-61729-867-7 (англ.)
ISBN 978-5-9775-1176-6 (рус.)

© Manning Publications, 2021
© Перевод на русский язык, оформление. ООО "БХВ-Петербург",
ООО "БХВ", 2023

Предисловие	13
От автора	15
Благодарности.....	17
О книге	19
Структура книги.....	19
Дискуссионный форум liveBook	20
Об авторе.....	21
Об обложке.....	23
ЧАСТЬ I. ОБ УЛУЧШЕНИИ НАВЫКОВ ЧТЕНИЯ КОДА.....	25
Глава 1. Определение вашего типа замешательства при кодировании	27
1.1. Разные типы замешательства в коде	28
1.1.1. Первый тип замешательства — недостаток знаний.....	29
1.1.2. Второй тип замешательства — недостаток информации.....	29
1.1.3. Третий тип замешательства — недостаток вычислительной мощности	30
1.2. Различные когнитивные процессы, влияющие на процесс кодирования	31
1.2.1. Долговременная память и программирование	31
Программа на APL с точки зрения долговременной памяти.....	32
1.2.2. Кратковременная память и программирование	32
Программа на Java с точки зрения кратковременной памяти	32
1.2.3. Рабочая память и программирование	33
Программа на BASIC с точки зрения рабочей памяти.....	33
1.3. Совместная работа когнитивных процессов	34
1.3.1. Краткое описание того, как когнитивные процессы взаимодействуют друг с другом.....	34
1.3.2. Когнитивные процессы и программирование	35
Выводы	37
Глава 2. Скорочтение кода.....	39
2.1. Быстрое чтение кода.....	40
2.1.1. Что только что происходило в вашем мозге	41
2.1.2. Перепроверка воспроизведенного кода	42
Вторая попытка воспроизведения кода	43

2.1.3. Перепроверка воспроизведенного.....	44
2.1.4. Почему читать незнакомый код так сложно.....	44
2.2. Преодоление лимитов памяти	45
2.2.1. Сила чанков.....	45
Чанки кода	48
2.2.2. Опытные программисты запоминают код лучше начинающих программистов.....	48
2.3. Вы видите намного больше кода, чем можете прочитать.....	49
2.3.1. Иконическая память	50
Иконическая память и код.....	51
2.3.2. Это не то, что вы помните; это то, как вы запоминаете	51
Как написать код, который можно разделить на чанки	52
Используйте паттерны проектирования	53
Пишите комментарии.....	54
Оставляйте «маячки»	55
2.3.3. Применяйте чанки	58
Выводы	59
Глава 3. Как быстро выучить синтаксис	61
3.1. Советы по запоминанию синтаксиса	62
3.1.1. Отвлечение снижает производительность.....	62
3.2. Как быстро выучить синтаксис с использованием карточек	63
3.2.1. Когда использовать карточки	64
3.2.2. Расширяем набор карточек	64
3.2.3. Убираем ненужные карточки.....	64
3.3. Как не забывать информацию	65
3.3.1. Почему мы забываем.....	65
Иерархия и сеть	66
Кривая забывания.....	66
3.3.2. Интервальное повторение	67
3.4. Как запомнить синтаксис надолго.....	69
3.4.1. Два способа запоминания информации	69
Уровень хранения.....	69
Уровень воспроизведения	69
3.4.2. Просто увидеть недостаточно.....	70
3.4.3. Воспоминания укрепляют память	70
3.4.4. Укрепление памяти путем активного мышления.....	71
Схемы	72
Проработка для запоминания концепций программирования	73
Выводы	74
Глава 4. Как читать сложный код	75
4.1. Почему так тяжело понимать сложный код	76
4.1.1. Чем друг от друга отличаются рабочая память и кратковременная память	77
4.1.2. Типы когнитивной нагрузки и как они связаны с программированием	78
Внутренняя когнитивная нагрузка при чтении кода	78
Внешняя когнитивная нагрузка при чтении кода.....	79
4.2. Способы снижения когнитивной нагрузки.....	80
4.2.1. Рефакторинг	80

4.2.2. Замена незнакомых языковых конструкций.....	82
Лямбда-функции.....	82
Генератор списков.....	83
Тернарные операторы.....	84
4.2.3. Синонимизация — отличное дополнение к дидактическим карточкам.....	85
4.3. Вспомогательные средства при перегрузке рабочей памяти.....	85
4.3.1. Создание графа зависимостей.....	86
4.3.2. Использование таблицы состояний.....	88
4.3.3. Сочетание графов зависимостей и таблиц состояний.....	91
Выводы.....	93

ЧАСТЬ II. ПРОДОЛЖАЕМ ДУМАТЬ О КОДЕ..... 95

Глава 5. Совершенствуем навыки углубленного понимания кода 97

5.1. Роли переменных.....	98
5.1.1. Разные переменные выполняют разные действия.....	98
5.1.2. Одиннадцать ролей, охватывающие почти все переменные.....	99
5.2. Роли и принципы.....	101
5.2.1. Польза ролей.....	102
Практические советы по работе с ролями переменных.....	103
5.2.2. Венгерская нотация.....	104
Системная и прикладная венгерские нотации.....	104
5.3. Углубленное понимание программ.....	106
5.3.1. Понимание текста и понимание плана.....	106
5.3.2. Этапы понимания программы.....	106
Применение этапов углубленного понимания.....	107
5.4. Чтение кода как обычного текста.....	109
5.4.1. Что происходит в мозге при чтении кода.....	110
Поля Бродмана.....	110
Показания фМРТ.....	111
5.4.2. Если вы можете выучить французский, то сможете выучить и Python.....	111
Как люди читают код.....	113
Перед тем как читать код, программисты сканируют его.....	114
Начинающие и опытные программисты читают код по-разному.....	114
5.5. Стратегии понимания текста, которые можно применить к коду.....	115
5.5.1. Активация пассивных знаний.....	116
5.5.2. Наблюдение.....	116
5.5.3. Определение важности разных строк кода.....	117
5.5.4. Предположения о значении имен переменных.....	118
5.5.5. Визуализация.....	119
Таблица операций.....	119
5.5.6. Постановка вопросов.....	120
5.5.7. Резюмирование кода.....	121
Выводы.....	122

Глава 6. Совершенствуем навыки решения задач программирования 123

6.1. Использование моделей для размышлений о коде.....	124
6.1.1. Преимущества использования моделей.....	124
Не все модели одинаково полезны.....	125

6.2. Ментальные модели	126
6.2.1. Подробное исследование ментальных моделей	128
6.2.2. Изучение новых ментальных моделей	129
6.2.3. Как эффективно использовать ментальные модели во время размышлений о коде	130
Ментальные модели в рабочей памяти.....	130
Точные модели работают лучше.....	131
Создание ментальных моделей исходной программы в рабочей памяти.....	131
Ментальные модели в долговременной памяти	132
Создание ментальных моделей исходной программы в долговременной памяти.....	133
Ментальные модели, одновременно хранящиеся в долговременной и рабочей памяти.....	134
6.3. Условные машины.....	135
6.3.1. Что такое условная машина	135
6.3.2. Примеры условных машин	136
6.3.3. Разные уровни условных машин	137
6.4. Условные машины и язык.....	138
6.4.1. Расширяем набор условных машин	139
6.4.2. Разные условные машины могут создать взаимно конфликтующие ментальные модели.....	140
6.5. Условные машины и схемы	141
6.5.1. Почему схема важна	141
6.5.2. Являются ли условные машины семантическими	142
Выводы	142

Глава 7. Заблуждения

7.1. Почему второй язык программирования выучить намного проще, чем первый	144
7.1.1. Как увеличить шанс воспользоваться знаниями по программированию.....	146
7.1.2. Разные виды трансференции.....	147
Осознанная и неосознанная трансференция	147
Близкая и дальняя трансференция	147
7.1.3. Знания: добро или зло?.....	148
7.1.4. Сложности трансференции	149
7.2. Заблуждения. Ошибки в мышлении.....	150
7.2.1. Исправление заблуждений путем концептуальных замен.....	152
7.2.2. Подавление заблуждений.....	152
7.2.3. Заблуждения о языках программирования	153
7.2.4. Предотвращение заблуждений при изучении нового языка программирования.....	155
7.2.5. Выявление заблуждений в новой базе кода.....	156
Выводы	157

ЧАСТЬ III. О ХОРОШЕМ КОДЕ

Глава 8. Совершенствуем навыки присваивания имен.....

8.1. Почему присваивание имен так важно	162
8.1.1. Почему присваивание имени так важно	162
Имена составляют существенную часть кодовой базы.....	163

Имена играют роль в обзорах кода	163
Имена — это самая удобная форма документации	163
Имена могут служить маячками	163
8.1.2. Разные точки зрения на присваивание имен	163
Хорошее имя можно определить синтаксически	164
Имена во всей базе кода должны быть единообразны	165
8.1.3. Важно грамотно подбирать имена	166
Заключения о практике присваивания имен	167
8.2. Когнитивные аспекты присваивания имен	167
8.2.1. Форматирование имен поддерживает кратковременную память	168
8.2.2. Понятные имена лучше закрепляются в долговременной памяти	169
8.2.3. Полезная информация в именах переменных	170
8.2.4. Когда стоит оценивать качество имен	171
8.3. Какие типы имен проще всего понимать	172
8.3.1. Использовать аббревиатуры или нет?	172
Однбуквенные имена переменных	173
8.3.2. Змеиный или верблюжий регистры?	176
8.4. Влияние имен на ошибки кода	177
8.4.1. В коде с некачественными именами больше ошибок	177
8.5. Как выбирать хорошие имена	178
8.5.1. Шаблоны имен	178
8.5.2. Трехступенчатая модель Фейтельсона для хороших имен переменных	180
Трехступенчатая модель во всех деталях	181
Успех трехступенчатой модели Фейтельсона	181
Выводы	182

Глава 9. Боремся с плохим кодом и когнитивной нагрузкой.

Две концепции	183
9.1. Почему код с запахами кода создает большую когнитивную нагрузку	184
9.1.1. Краткая информации о запахах кода	184
Запахи кода на уровне метода	186
Запахи кода на уровне класса	186
Запахи кода на уровне базы кода	187
Влияние запахов кода	187
9.1.2. Как запахи кода вредят мышлению	188
«Длинный список параметров», сложные «Операторы переключения» — перегрузка рабочей памяти	188
«Всемогущий класс», «Длинный метод» — невозможно эффективно разбить код на чанки	189
«Клоны кода» — невозможно правильно разбить код на чанки	189
9.2. Зависимость когнитивной нагрузки от плохих имен	190
9.2.1. Лингвистические антипаттерны проектирования	190
9.2.2. Измерение когнитивной нагрузки	191
Шкала Пааса для когнитивной нагрузки	192
Измерение нагрузки по глазам	193
Измерение нагрузки по коже	193
Измерение нагрузки по мозгу	194
Запись биотоков мозга	194
Функциональная fNIRS-томография и программирование	195

9.2.3. Лингвистические антипаттерны и когнитивная нагрузка.....	195
9.2.4. Почему лингвистические антипаттерны вызывают замешательство	196
Выводы	197
Глава 10. Совершенствуем навыки решения сложных задач.....	199
10.1. Что такое решение задач.....	200
10.1.1. Элементы решения задач	200
10.1.2. Пространство состояний.....	200
10.2. Какую роль при решении задач программирования играет долговременная память	201
10.2.1. Решение задачи — это отдельный когнитивный процесс?.....	201
При решении задач вы используете долговременную память	202
Вашему мозгу проще решить знакомые задачи	202
10.2.2. Как научить долговременную память решать задачи.....	203
10.2.3. Два вида памяти, наиболее существенные при решении задачи.....	203
Какие виды памяти играют роль при решении задач	204
Потеря знаний или навыков	205
10.3. Автоматизация: создание имплицитной памяти	206
10.3.1. Имплицитная память с течением времени	207
Когнитивный этап	208
Ассоциативный этап	208
Автономный этап	209
10.3.2. Почему автоматизация помогает программировать быстрее	210
10.3.3. Улучшение имплицитной памяти	211
10.4. Обучение на основе кода и его объяснения	212
10.4.1. Новый вид когнитивной нагрузки: соответствующая нагрузка	213
10.4.2. Примеры с решением на практике	215
Работайте вместе с коллегой.....	215
Используйте GitHub.....	215
Читайте книги или блоги об исходном коде.....	216
Выводы	216

ЧАСТЬ IV. О СОВМЕСТНОЙ РАБОТЕ НАД КОДОМ..... 217

Глава 11. Процесс написания кода

11.1. Различные активности, выполняемые во время программирования.....	220
11.1.1. Поиск	220
11.1.2. Осмысление	221
11.1.3. Переписывание	221
11.1.4. Нарращивание.....	222
11.1.5. Исследование	222
11.1.6. А как же отладка?.....	223
11.2. Программист отвлекся	223
11.2.1. Задачи программирования нуждаются в «разогреве»	224
11.2.2. Что происходит после отвлечения	225
11.2.3. Как подготовиться к отвлечению.....	225
Сохраняйте воображаемую модель	225
Помогите своей проспективной памяти.....	226
Определитесь с промежуточными целями.....	228

11.2.4. Когда отвлекать программиста	228
11.2.5. Пара слов о многозадачности.....	230
Многозадачность и автоматизация.....	230
Исследования многозадачности.....	231
Выводы	231
Глава 12. Проектирование и усовершенствование больших систем	233
12.1. Проверка свойств базы кода	234
12.1.1. Когнитивные измерения	234
Подверженность ошибкам	235
Согласованность	236
Размытость	236
Скрытые зависимости.....	237
Преждевременная фиксация решения.....	238
Вязкость	238
Поэтапное оценивание.....	239
Выразительность ролей	239
Близость соответствия	240
Трудность мыслительных операций.....	241
Вторичные обозначения	242
Градиент абстракции	242
Наглядность	243
12.1.2. Использование когнитивных измерений базы кода для улучшения базы кода.....	243
12.1.3. Проектные маневры и их плюсы и минусы.....	244
Подверженность ошибкам и вязкость.....	244
Преждевременная фиксация решения и поэтапное оценивание против подверженности ошибкам	245
Выразительность ролей и размытость	245
12.2. Измерения и активности	245
12.2.1. Влияние измерений на разные активности.....	245
Поиск.....	245
Осмысление	246
Переписывание.....	246
Нарращивание	246
Исследование.....	247
12.2.2. Изменение базы кода под ожидаемые активности	247
Выводы	247
Глава 13. Как ввести новых программистов в курс дела	249
13.1. Проблемы процесса адаптации.....	249
13.2. Различия между профессионалами и новичками.....	251
13.2.1. Поведение новичка более подробно	251
Оригинальная концепция Пиаже	251
Концепция неопиажизма для программирования	252
При изучении новой информации вы можете временно забывать некоторые вещи.....	255
13.2.2. Разница между вещественным и абстрактным видением концепций.....	255

13.3. Активности для улучшения процесса адаптации	258
13.3.1. Ограничение заданий до одной активности	258
13.3.2. Поддержка памяти новичка	259
Поддержка долговременной памяти: объяснение релевантной информации	259
Поддержка кратковременной памяти: ставьте небольшие конкретные задачи	260
Поддержка рабочей памяти: используйте диаграммы	261
13.3.3. Совместное чтение кода	261
Активация	262
Определение важности	262
Постановка предположений	262
Наблюдение	262
Визуализация	263
Постановка вопросов	263
Резюмирование	263
Выводы	263
Эпилог. Пара слов перед прощанием	265
Предметный указатель	267

Предисловие

Большую часть своей жизни я размышлял о программировании. Если вы сейчас читаете эту книгу, то готов поспорить, что и вы тоже. Однако я никогда не тратил много времени на размышления о том, как я думаю. Для меня всегда было важно понятие о наших мыслительных процессах и то, как мы взаимодействуем с кодом как люди, однако я не опирался ни на какие научные исследования. Давайте я приведу три примера.

Я — ведущий участник .NET-проекта Noda Time, предоставляющего набор типов даты и времени, альтернативный встроенному в .NET набору. Это была отличная возможность погрузиться в проектирование программных интерфейсов и особенно в придумывание имен! Я увидел множество проблем, связанных с именами: они звучат так, словно меняют уже существующее значение, хотя на самом деле возвращают новое значение! Поэтому я постарался использовать такие имена, чтобы код с ошибками при чтении выглядел неправильно. Например, тип `LocalDate` с методом `PlusDays`, а не `AddDays`. Надеюсь, что большинству C#-разработчиков данный код покажется неправильным:

```
date.PlusDays(1);
```

В то время как следующий код будет более понятным:

```
tomorrow = today.PlusDays(1);
```

Сравните с методом `AddDays` .NET-типа `DateTime`:

```
date.AddDays(1);
```

Кажется, что это всего лишь изменение даты и ошибки тут нет, хотя этот вариант, как и первый, неправильный.

Второй пример, более общего характера, также взят из проекта Noda Time. В то время как многие библиотеки стараются (из лучших побуждений) выполнять всю тяжелую работу за разработчика, мы хотим, чтобы пользователи Noda Time заранее продумали код обработки даты и времени. Мы пытаемся заставить пользователей однозначно сформулировать, чего именно они хотят достичь, а затем помогаем им выразить это в коде.

И наконец, концептуальный пример: какие значения хранятся в переменных Java и C# и что происходит, когда вы передаете аргумент методу. Мне кажется, что большую часть своей жизни я пытаюсь опровергнуть концепцию того, что на Java объекты передаются по ссылке. Похоже, что так и есть: я уже четверть века помогаю другим разработчикам настраивать их мысленные модели.

Получается, мне всегда было важно, как думают другие программисты, но у меня не было никаких знаний в этой области — я довольствовался догадками на своем выстраданном опыте. Эта книга помогает мне это изменить, хотя и не является отправной точкой для меня.

С Фелиной Херманс я познакомился в 2017 году в Осло на конференции NDC, где она выступала с презентацией «Programming Is Writing Is Programming». И моя реакция в Твиттере говорит сама за себя: «Мне понадобится время, чтобы осознать все это! Но я потрясен. Просто потрясен». Я минимум трижды (конечно, в разное время) посещал эту презентацию Фелины и каждый раз узнавал что-то новое. Наконец-то я получил научное объяснение тому, что пытался делать, а также узнал и такие вещи, которые заставили меня изменить свой подход к работе.

Во время чтения этой книги у меня постоянно возникали реакции вроде «Я до этого не додумался!» и «О, теперь я понимаю!». Подозреваю, что помимо очевидной пользы от таких практических советов, как, например, применение дидактических карточек, книга даст толчок, окажет более глубокое влияние. Возможно, вы будете тщательнее обдумывать, куда именно в код нужно вставлять пустую строку. Возможно, измените задачи, предлагаемые новеньким в команде, либо скорректируете сроки выполнения этих задач. Возможно, мы станем по-другому объяснять концепции на платформе Stack Overflow.

Как бы там ни было, Фелина предоставила нам сокровищницу идей, которые мы обдумаем в рабочей памяти, а затем сохраним в долговременной — ведь мысли о мышлении вызывают привыкание!

*Джон Скит,
менеджер по персоналу, Google*

Примерно десять лет назад я начала обучать детей программированию. В тот момент я осознала, что практически ничего не знаю о том, как люди используют свой мозг для выполнения различных задач, особенно когда дело касается программирования. И хотя я изучала программирование в университете, ничто из университетского курса не готовило меня к тому, что я буду рассуждать о том, как думают программисты.

Если вы изучали информатику в вузе или постигали программирование самостоятельно, то вы, скорее всего, ничего не знаете о когнитивных функциях мозга. А значит, вы не в курсе, как можно тренировать свой мозг для упрощения процесса чтения и написания кода. Я тоже ничего об этом не ведала, но занятия с детьми погрузили меня в эту тему. Я узнала очень много о том, как мы думаем и как мы учимся. Эта книга — результат многолетних исследований, в ходе которых я прочитала множество книг, общалась с людьми и посещала выступления и конференции, посвященные обучению и мышлению.

Понимание, как работает мозг, представляет собой очень интересную тему, однако это понимание также важно и для программирования. Программирование считается одним из самых сложных когнитивных видов деятельности, ведь программист не только решает проблему абстрактным способом, но и обращается с программой. Все это требует колоссального уровня внимания, которого у большинства людей просто нет. Пропустили пробел? Ошибка. Неверно индексировали массив? Ошибка. Не разобрались в нюансах работы исходного кода? Ошибка.

В процессе программирования вы можете совершить множество ошибок. Из этой книги вы узнаете, что причиной многих ошибок являются когнитивные проблемы. Например, пропуск пробела может означать, что вы недостаточно хорошо овладели синтаксисом языка программирования. Ошибка индексации массива может указывать на то, что вы имеете неправильное представление о коде. Непонимание исходного кода говорит об отсутствии навыков чтения кода.

Цель этой книги проста — помочь вам понять, как мозг обрабатывает программный код. Понимание того, что делает ваш мозг с новой информацией, поможет повысить ваше мастерство, так как профессиональные программисты постоянно сталкиваются с чем-то новым. После того как мы узнаем о том, как код воздействует на мозг, мы рассмотрим методы улучшения навыков обработки кода.

Благодарности

Я очень хорошо понимаю, как мне повезло, что я смогла закончить книгу по любимой теме. Если бы не определенная череда событий, произошедших в определенные моменты, моя жизнь была бы совсем другой и я не написала бы эту книгу. Десятки самых различных встреч с замечательными людьми внесли неоценимый вклад в эту книгу и в мою работу. Имена наиболее значимых я хочу назвать.

Марлиз Альдеверелд (Marlies Aldewereld) впервые познакомила меня с программированием и изучением языков. Мэрилин Смит (Marileen Smit) подтянула меня в психологии, благодаря чему я написала эту книгу. Грег Уилсон (Greg Wilson) вернул в тренды тему обучения программированию. Питер Наббе (Peter Nabbe) и Роб Хогерворд (Rob Hoogerwoord) были для меня примером для подражания в сфере обучения. Штефан Ханенберг (Stefan Hanenberg) дал мне совет, определивший направленность моих исследований. Катя Мордаунт (Katja Mordaunt) открыла первый в мире клуб чтения кода. Размышления Йевеллина Фалко (Llewellyn Falco) о задачах привели в порядок мои мысли об обучении. Рико Хейберс (Rico Huijbers) был моей поддержкой в моменты, когда я совершенно не знала, что делать.

Я также хочу поблагодарить людей из издательства Manning: Марьям Басэ (Marjan Bace), Майка Стивенса (Mike Stephens), Тришу Лаувар (Tricia Louvar), Берта Бейтса (Bert Bates), Михаела Батинича (Mihaela Batinic), Бекки Рейнхарт (Becky Reinhart), Мелиссу Айс (Melissa Ice), Дженнифер Хаул (Jennifer Houle), Пола Уэллса (Paul Wells), Джерри Кюха (Jerry Kuch), Рейчел Хед (Rachel Head), Себастьяна Портебуа (Sébastien Portebois), Кэндис Гилхули (Candace Gillhoolley), Криса Кауфмана (Chris Kaufmann), Матко Хрватина (Matko Hrvatin), Ивана Мартиновича (Ivan Martinovic), Бранко Латинчика (Branko Latincic) и Андрея Хофшустера (Andrej Hofšuster) — за то, что они взялись за работу, когда у меня были только наброски, и превратили их во что-то понятное, интересное и легко читаемое.

А также всех рецензентов: Адама Качмарека (Adam Kaczmarek), Адриана Бэйертца (Adriaan Beiertz), Алекса Риоса (Alex Rios), Ариэль Гаминьо (Ariel Gamiño), Бена МакНамара (Ben McNamara), Билла Митчелла (Bill Mitchell), Билли О'Каллагана (Billy O'Callaghan), Бруно Соннино (Bruno Sonnino), Чарльза Лэма (Charles Lam), Клаудию Мадертанер (Claudia Maderthaner), Клиффорда Тербера (Clifford Thurber), Даниэлу Запату Риско (Daniela Zapata Riesco), Эмануэля Орижи (Emanuele Origi), Джорджа Онофрея (George Onofrei), Джорджа Томаса (George Thomas), Гилберто Такари (Gilberto Taccari), Хэйма Рамана (Haim Raman), Жауме Лопеса (Jaume

Lopez), Джозефа Перения (Joseph Perenia), Кента Спиллнера (Kent Spillner), Кимберли Уинстон-Джексона (Kimberly Winston-Jackson), Мануэля Гонзалеса (Manuel Gonzalez), Марцина Сэка (Marcin Sęk), Марка Харриса (Mark Harris), Мартина Кнудсена (Martin Knudsen), Майка Хьюитсона (Mike Hewitson), Майка Тэйлора (Mike Taylor), Орландо Мендеса Моралеса (Orlando Méndez Morales), Педро Серомено (Pedro Seromenho), Питера Моргана (Peter Morgan), Саманту Берк (Samantha Berk), Себастьяна Феллинга (Sebastian Felling), Себастьяна Портебуа (Sébastien Portebois), Саймона Чоке (Simon Tschöke), Стефано Онгарейо (Stefano Ongarello), Томаса Хансена (Thomas Overby Hansen), Тима ван Дерзена (Tim van Deurzen), Туомо Каллиокоски (Tuomo Kalliokoski), Унникришнана Кумара (Unnikrishnan Kumar), Василе Бориса (Vasile Boris), Виктора Бека (Viktor Bek), Закари Бейела (Zachery Beuel) и Чжицзюнь Лю (Zhijun Liu). Ваши предложения помогли мне сделать эту книгу лучше!

Мозг программиста — это книга для программистов всех уровней, которые хотят разобраться, как работает их мозг и как можно улучшить свои навыки и стиль программирования. В данной книге будут показаны примеры кода на разных языках, включая JavaScript, Python и Java. Если вам удобно читать исходный код на языке, который вы никогда не встречали, то вам не нужны серьезные знания какого-либо из этих языков программирования.

Чтобы получить максимальную пользу от этой книги, желателен опыт работы в группе разработчиков или опыт работы над крупными программными системами, и привлечением людей в команду. Мы будем часто обращаться к подобным ситуациям, так что читатели, которые уже имеют опыт в данной области, получают намного больше ценной информации, чем другие. Человек, который может связать новую информацию с уже имеющимися у него знаниями и опытом, обучается продуктивнее и быстрее.

Несмотря на то что здесь представлены многие темы, связанные с когнитивистикой, эта книга предназначена в основном для программистов. В приведенных примерах мы всегда будем рассматривать работу мозга в контексте результатов исследований программирования и языков программирования.

Структура книги

Книга состоит из 13 глав, разделенных на четыре части. Главы следует читать по порядку, так как все они связаны. Каждая глава содержит прикладные примеры и упражнения, способствующие лучшему усвоению и закреплению полученного материала. В некоторых случаях для выполнения упражнения вам потребуется подыскать кодовую базу — так вы сможете работать с подходящим вам контекстом.

Вы также можете использовать полученные знания в вашей повседневной работе. Я думаю, что эту книгу можно изучать долгое время вот с каким планом: сначала вы изучаете главу, применяете упражнения из главы в своей практике программирования, а затем переходите к изучению следующих глав:

- В *главе 1* рассматриваются три когнитивных процесса, которые играют роль при программировании, а также рассматривается их связь с типами замешательства.
- В *главе 2* вы найдете советы о том, как быстро читать код и понимать его.
- В *главе 3* вы узнаете, как наиболее эффективно изучать синтаксис и концепции программирования.

- В *главе 4* вы узнаете, как читать сложный код.
- В *главе 5* показаны методы, которые помогут глубже понять незнакомый код.
- В *главе 6* рассматриваются методы, которые «прокачают» способность решать задачи программирования.
- В *главе 7* вы найдете советы о том, как избежать ошибок в коде и его понимании.
- В *главе 8* вы узнаете, как выбрать подходящее имя для переменной.
- *Глава 9* посвящена признакам кода «с запахом» и лежащим в их основе когнитивным причинам.
- В *главе 10* рассматриваются более сложные способы решения трудных задач.
- В *главе 11* описывается процесс кодирования и исследуется разнообразие задач программирования.
- В *главе 12* вы узнаете, как улучшить большие кодовые базы.
- В *главе 13* вы узнаете, как сделать процесс адаптации новых сотрудников менее болезненным.

В данной книге вы найдете множество примеров исходного кода: и в пронумерованных листингах, и просто в тексте. В обоих случаях при написании исходного кода используется моноширинный шрифт — так он бросается в глаза в обычном тексте. Иногда код может быть выделен **полужирным шрифтом** — так показаны изменения в коде, использовавшемся в предыдущих шагах в главе (например, при добавлении новой функции к уже существующей строке кода).

Во многих случаях первоначальный исходный код подвергается некоторым изменениям: например, мы добавили переносы или изменили отступы для того, чтобы код уместился на странице книги. Иногда этого было недостаточно, поэтому в листингах есть метка продолжения строки (➔). Кроме того, если исходный код описывается в тексте, то комментарии в исходном коде удаляются. Большинство листингов даются с аннотациями, в которых описаны основные идеи и концепции.

Дискуссионный форум liveBook

Приобретая книгу «Мозг программиста», вы получаете бесплатный доступ к частному веб-форуму Manning Publications, где вы можете оставить свои отзывы и комментарии о книге, а также задать технические вопросы и получить помощь от автора книги и других пользователей. Чтобы получить доступ к форуму, перейдите по ссылке <https://livebook.manning.com/book/the-programmers-brain/discussion>. Больше о форумах Manning и правилах поведения на форуме можно узнать по адресу <https://livebook.manning.com/#!/discussion>.

Нашей обязанностью было создание уголка, где читатели могли бы вести содержательный диалог и друг с другом, и с автором. Вам необязательно ответят, так как участие автора на форуме является добровольным (и не оплачивается). Тем не менее можно задавать сложные вопросы автору!

Об авторе

Доктор Фелина Херманс — доцент Лейденского университета в Нидерландах, где она проводит научные исследования языков программирования и методов обучения программированию. Она читает лекции в академии учителей Амстердамского свободного университета, специализируясь на дидактике компьютерных наук, а также преподает в средней школе Кралингена в Роттердаме.

Фелина является создателем языка программирования Hedy, предназначенного для начинающих программистов, а также ведет подкаст Software Engineering Radio, один из крупнейших интернет-подкастов о программном обеспечении.

Рисунок на обложке данной книги называется «Femme Sauvage du Canada», или «Коренная жительница Канады». Изображение взято из коллекции костюмов разных стран Жака Грассе де Сен-Совера (1757–1810) под названием «Costumes civils actuels de tous les peuples connus», изданной во Франции в 1788 году. Каждая иллюстрация была нарисована и раскрашена вручную. Показанное Грассе де Сен-Совером разнообразие напоминает нам о том, насколько города и регионы мира были культурно обособлены всего лишь 200 лет назад. Изолированные друг от друга, люди разговаривали на разных языках и диалектах. Встретив человека в городе или в деревне, по одной лишь одежде можно было определить, чем он занимается по жизни и где живет.

С тех пор манера одеваться сильно изменилась, а многие характерные признаки регионов просто исчезли. Теперь очень сложно отличить жителей разных континентов, не говоря уже о городах или странах. Может быть, мы променяли культурное разнообразие на разнообразную персональную жизнь — и уж точно на разнообразную и быстро развивающуюся технологическую жизнь.

И когда одну книгу о компьютерах с трудом получается отличить от другой, издательство Manning демонстрирует изобретательность компьютерного производства с помощью обложек книг, которые основаны на богатом разнообразии региональной жизни 200 лет назад, описанном в книге Грассе де Сен-Совера.

Часть I

Об улучшении навыков чтения кода

Чтение кода — основная часть программирования, однако не всегда даже опытные разработчики знают, как правильно это делать. Чтению кода не учат, и это редко практикуется, а незнакомый код сбивает с толку. Первые главы этой книги помогут вам понять, почему чтение кода — это сложно, и что можно сделать, чтобы читать код было проще.

1

Определение вашего типа замешательства при кодировании

В этой главе:

- вы узнаете о том, как можно запутаться при кодировании;
- сравните три когнитивных процесса, играющие роль при кодировании;
- поймете, как разные когнитивные процессы дополняют друг друга.

Замешательство — неотъемлемая часть программирования. Когда вы изучаете новый язык программирования, фреймворк или какую-либо концепцию, новые идеи могут напугать вас. Читая незнакомый код или собственный, но написанный вами много лет назад, вы можете не понять, для чего он нужен или почему он написан именно так. Когда вы приступаете к работе в новой сфере деятельности, то от новых терминов и сленга у вас может получиться мешанина в голове.

Конечно, нет ничего странного в том, что первое время вы будете испытывать замешательство — однако не стоит пребывать в этом состоянии больше необходимого. Эта глава поможет вам распознать и определить ваше замешательство. Возможно, вы никогда не задумывались об этом, но существует несколько способов впасть в замешательство. Замешательство от незнания значения какого-то понятия прикладной области отличается от замешательства при разборе шаг за шагом сложного алгоритма.

Разные типы замешательства относятся к разным когнитивным процессам. На нескольких примерах программного кода мы подробно рассмотрим три типа замешательства и объясним, что происходит в вашей голове.

К концу этой главы вы научитесь распознавать различные способы, которыми код может вызвать замешательство, и определять когнитивный процесс, происходящий в каждом случае в вашем мозге. В следующих главах вы узнаете, как улучшить эти когнитивные процессы.