Оглавление

Друзья — это семья, которую мы сами себе выбираем.
Я посвящаю книгу моим друзьям Ромуло, Лео,
Мото и Крису, которые постоянно меня торопили,
чтобы я ее, наконец, закончил.

Я знаю, что дважды два — четыре, и был бы рад доказать это, если б мог, — хотя должен заметить, если бы мне удалось дважды два превратить в пять, это доставило бы мне гораздо больше удовольствия.

Лорд Байрон,
из письма будущей жене Аннабелле (1813 год).
Их дочь Ада Лавлейс стала первым программистом

Предисловие

> Каждый в нашей стране должен научиться программировать, потому что это учит думать.
>
> *Стив Джобс*

Когда компьютеры начали менять мир, открывая перед людьми беспрецедентные возможности, расцвела новая наука — *computer science*. Она показала, как использовать компьютеры для решения задач. Это позволило нам использовать весь потенциал вычислительных машин. И мы достигли удивительных, просто сумасшедших результатов.

Computer science повсюду, но эта наука по-прежнему преподается как скучная теория. Многие программисты даже не изучали ее! Однако она крайне важна для эффективного программирования. Некоторые мои друзья не могут найти хорошего программиста, чтобы взять его на работу. Вычислительные мощности сегодня в изобилии, а вот людей, способных ими пользоваться, не хватает.

Рис. 1. Компьютерные задачи[1]

Эта книга — моя скромная попытка помочь миру, а также подтолкнуть *вас* к эффективному использованию компьютеров. В ней понятия computer science представлены в простой форме. Я свел научные подробности к минимуму. Хочется надеяться, что computer science произведет на вас впечатление, и ваш программный код станет лучше.

Эта книга для меня?

Если вы хотите щелкать задачи как орешки, находя эффективные решения, то эта книга для вас. От вас потребуется только чуть-чуть опыта в написании программного кода. Если вам приходилось этим заниматься и вы различаете элементарные операторы вроде `for` и `while`, то все в порядке. В противном случае вы найдете все необходимое (и даже больше) на каких-нибудь онлайновых курсах программирования[2]. Вы можете пройти такой курс всего за неделю, и притом бесплатно. Для тех же, кто уже знаком с информатикой, эта книга станет превосходным повторением пройденного и поможет укрепить знания.

---

[1] Рисунок использован с согласия http://xkcd.com.

[2] См., например, http://code.energy/coding-courses.

Но разве computer science не только для ученых?

Эта книга — для всех. Она о вычислительном мышлении. Вы научитесь превращать задачи в вычислимые системы. Вы также будете использовать вычислительное мышление в повседневных задачах. Упреждающая выборка и кэширование помогут вам упростить процесс упаковки вещей. Освоив параллелизм, вы станете эффективнее управляться на кухне. Ну и, разумеется, ваш программный код будет просто потрясающим.

Да пребудет с вами сила! 

Влад

Глава 1

ОСНОВЫ

> Информатика не более наука о компьютерах, чем астрономия — наука о телескопах. Информатика неразрывно связана с математикой.
>
> *Эдсгер Дейкстра*

Компьютерам нужно, чтобы мы разбивали задачи на посильные для них части. Тут нам понадобится немного математики. Не паникуйте, это не высшая математика — написание хорошего программного кода редко требует знания сложных уравнений. В главе 1 вы найдете набор инструментов для решения разных задач. Вы научитесь:

моделировать *идеи* в блок-схемах и псевдокоде;


отличать правильное от неправильного при помощи *логики*;


выполнять *расчеты*;


уверенно вычислять *вероятности*.

Этого достаточно, чтобы переводить мысли в вычислимые решения.

1.1. Идеи

Оказавшись перед сложной задачей, поднимитесь над ее хитросплетениями и изложите все самое важное на бумаге. Оперативная память человеческого мозга легко переполняется фактами и идеями. Многие подходы к организации работы предполагают изложение мыслей в письменной форме. Есть несколько способов это сделать. Сначала мы посмотрим, как пользоваться блок-схемами для представления процессов. Затем узнаем, как конструировать программируемые процессы на псевдокоде. Мы также попробуем смоделировать простую задачу при помощи математических формул.

Блок-схемы

Когда разработчики «Википедии» обсуждали организацию коллективной работы, они создали блок-схему дискуссии. Договариваться проще, если все инициативы перед глазами и объединены в общую картину (рис. 1.1).

Компьютерный код, как и изображенный выше процесс редактирования вики-страницы, по существу является процессом. Программисты часто пользуются блок-схемами для изображения вычислительных процессов на бумаге. Чтобы другие могли понимать ваши блок-схемы, вы должны соблюдать следующие рекомендации[1]:

- записывайте состояния и инструкции внутри прямоугольников;
- записывайте принятие решений, когда процесс может пойти различными путями, внутри ромбов;
- никогда не объединяйте инструкции с принятием решений;

[1] Адаптация схемы с сайта http://wikipedia.org.

- соединяйте стрелкой каждый последующий шаг с предыдущим;
- отмечайте начало и конец процесса.

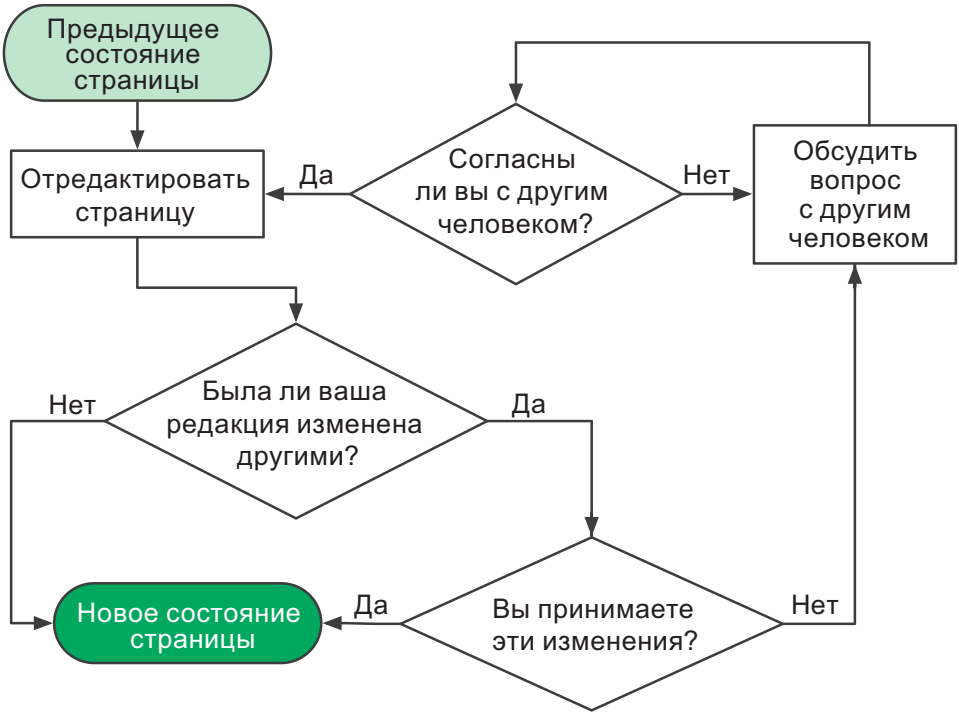


Рис. 1.1. Редакционный процесс в «Википедии»[1]

[1] См., например, http://code.energy/coding-courses.

Рассмотрим составление блок-схемы на примере задачи поиска наибольшего из трех чисел (рис. 1.2).

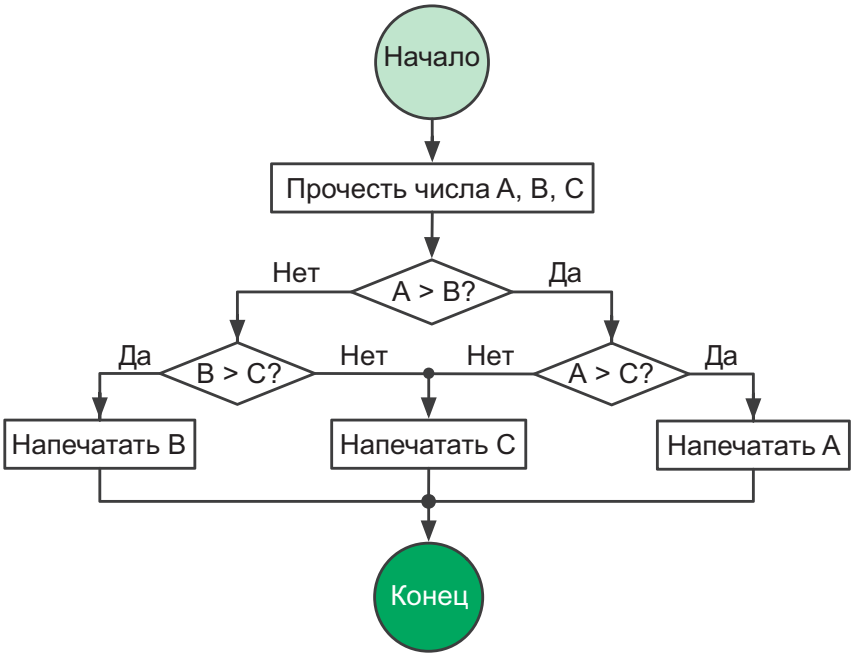


Рис. 1.2. Поиск наибольшего из трех чисел

Псевдокод

Так же, как блок-схемы, *псевдокод* выражает вычислительные процессы. Псевдокод — это код, удобный для нашего восприятия, но непонятный для машины. Следующий пример передает тот же процесс, что был изображен на рис. 1.2. Задержитесь на минуту и проверьте, как он работает с разными значениями A, B и C[1].

[1] Здесь ← означает оператор присваивания: $x \leftarrow 1$ следует читать как *«Присвоить x значение 1»*.

```
function maximum(A, B, C)
    if A > B
         if A > C
              max ← A
         else
              max ← C
    else
         if B > C
              max ← B
         else
              max ← C
    print max
```

Заметили, что этот пример полностью игнорирует синтаксические правила языков программирования? В псевдокод можно вставлять даже разговорные фразы! Когда вы пишете псевдокод, дайте своей творческой мысли течь свободно — как при составлении блок-схем (рис. 1.3 😄).

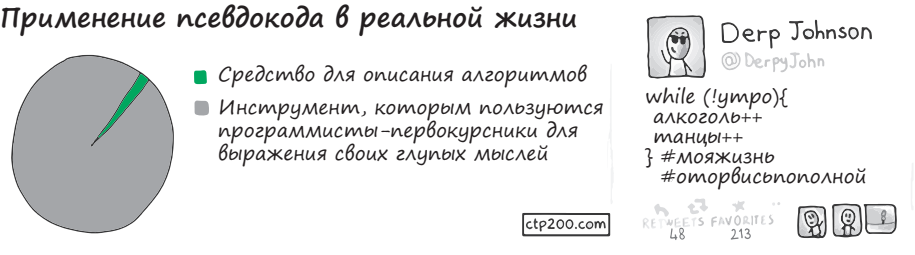


Рис. 1.3. Псевдокод в реальной жизни[1]

Математические модели

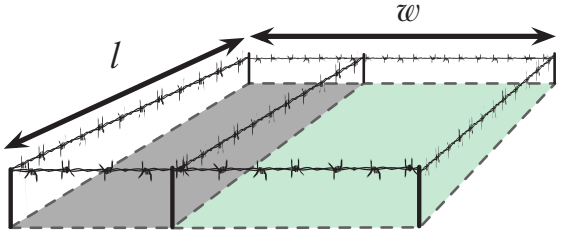
Модель — это набор идей, которые описывают задачу и ее свойства. Модель помогает рассуждать и принимать решения относительно задачи. Создание моделей настолько важно, что их преподают в школе — ведь в математике нужно иметь представление, как последовательно решать уравнения и совершать другие операции с числами и переменными.

[1] Любезно предоставлено http://ctp200.com.

Математические модели имеют большое преимущество: их можно приспособить для компьютеров при помощи четко сформулированных математических методов. Если ваша модель основана на графах, используйте теорию графов. Если она задействует уравнения, используйте алгебру. *Встаньте на плечи гигантов*, которые создали эти инструменты, и вы достигнете цели. Давайте посмотрим, как они работают, на примере типичной задачи из средней школы.

> **Загон для скота**  На ферме содержат два вида домашних животных. У вас есть 100 мотков проволоки для сооружения прямоугольного загона и перегородки внутри него, отделяющей одних животных от других. Как поставить забор, чтобы площадь пастбища была максимальной?

Начнем с того, что именно требуется определить; w и l — это размеры пастбища; $w \times l$ — его площадь. Сделать площадь максимальной означает использовать всю проволоку, потому мы устанавливаем связь между w и l, с одной стороны, и 100 мотками, с другой:



$$A = w \times l$$

$$100 = 2w + 3l$$

Подберем w и l, при которых площадь A будет максимальной.

Подставив l из второго уравнения $\left(l = \frac{100 - 20w}{3} \right)$ в первое, получаем:

$$A = \frac{100}{3} w - \frac{2}{3} w^2.$$

Да это же квадратное уравнение! Его максимум легко найти при помощи *формулы корней квадратного уравнения*, которую проходят в средней школе. Квадратные уравнения так же важны для программиста, как мультиварка — для повара. Они экономят время. Квадратные уравнения помогают быстрее решать множество задач, а это для вас самое главное. Повар знает свои инструменты, вы должны знать свои. Математическое моделирование вам просто необходимо. А еще вам потребуется логика.

1.2. Логика

Программистам приходится иметь дело с логическими задачами так часто, что у них от этого ум за разум заходит. Однако на самом деле многие из них логику не изучали и пользуются ею бессознательно. Освоив формальную логику, мы сможем осознанно использовать ее для решения задач.

Рис. 1.4. Логика программиста[1]

[1] Любезно предоставлено http://programmers.life.

Для начала мы поэкспериментируем с логическими высказываниями и операторами. Затем научимся решать задачи с таблицами истинности и увидим, как компьютеры опираются на логику.

Операторы

В математике переменные и операторы (+, ×, −, ...) используются для моделирования числовых задач. В математической логике переменные и операторы указывают на достоверность. Они выражают не числа, а истинность (true) или ложность (false). Например, достоверность выражения «Если вода в бассейне теплая, то я буду плавать» основывается на достоверности двух вещей, которые можно преобразовать в логические переменные A и B:

A : Вода в бассейне теплая.

B : Я плаваю.

Они либо истинны (true), либо ложны (false)[1]. A = `True` обозначает теплую воду в бассейне, B = `False` обозначает «Я не плаваю». Переменная B не может быть *наполовину истинной*, потому что я не способен плавать лишь отчасти. Зависимость между переменными обозначается символом →, *условным оператором*. $A \rightarrow B$ выражает идею, что A = `True` влечет за собой B = `True`:

$A \rightarrow B$: если вода в бассейне теплая, то я буду плавать.

При помощи других операторов можно выражать другие идеи. Для отрицания идеи используется знак !, *оператор отрицания*. $!A$ противоположно A:

$!A$: Вода в бассейне холодная.

$!B$: Я не плаваю.

[1] В нечеткой логике значения могут быть промежуточными, но в этой книге она рассматриваться не будет.

Противопоставление. Если дано $A \rightarrow B$, и я при этом не плаваю, что можно сказать о воде в бассейне? Теплая вода *влечет за собой* плавание, потому, если его нет, вода в бассейне не может быть теплой. Каждое условное выражение имеет *противопоставленный* ему эквивалент:

Для любых двух переменных A и B

$$A \rightarrow B \text{ тождественно } !B \rightarrow !A.$$

Еще пример: если вы не умеете писать хороший код, значит, вы не прочли эту книгу. *Противопоставлением данному суждению является такое:* если вы прочли эту книгу, значит, вы умеете писать хороший код. *Оба предложения сообщают одно и то же, но по-разному*[1].

Двусторонняя условная зависимость. Обратите внимание, что высказывание «*Если вода в бассейне теплая, то я буду плавать*» не означает: «Я буду плавать только в теплой воде». Данное высказывание ничего не говорит насчет холодных бассейнов. Другими словами, $A \rightarrow B$ не означает $B \rightarrow A$. Чтобы выразить оба условных суждения, используйте *двустороннюю условную зависимость*:

$A \leftrightarrow B$: Я буду плавать, если и только если вода в бассейне теплая.

Здесь теплая вода в бассейне равнозначна тому, что я буду плавать: знание о воде в бассейне означает знание о том, что я буду плавать, *и наоборот*. Опять же, остерегайтесь *обратной ошибки*: никогда не предполагайте, что $B \rightarrow A$ следует из $A \rightarrow B$.

AND, OR и XOR. Эти логические операторы — самые известные, поскольку они часто записываются в исходном коде в явном виде — `AND` (И), `OR` (ИЛИ) и `XOR` (исключающее ИЛИ). `AND` возвращает `True`, если все идеи истинны; `OR` возвращает `True`, если любая идея истинна; `XOR` возвращает `True`, если идеи взаимоисключающие. Представим вечеринку, где подают водку и вино:

[1] И, между прочим  , они оба *фактически* истинные.

A : Вы пили вино.

B : Вы пили водку.

A OR B : Вы пили.

A AND B : Вы пили и то и другое.

A XOR B : Вы пили, не смешивая.

Проверьте, правильно ли вы понимаете, как работают эти операторы. В табл. 1.1 перечислены все возможные комбинации двух переменных. Обратите внимание, что $A \rightarrow B$ тождественно $!A$ OR B, а A XOR B тождественно $!(A \leftrightarrow B)$.

A	B	! A	A → B	A ↔ B	A AND B	A OR B	A XOR B
✓	✓	✗	✓	✓	✓	✓	✗
✓	✗	✗	✗	✗	✗	✓	✓
✗	✓	✓	✓	✗	✗	✓	✓
✗	✗	✓	✓	✓	✗	✗	✗

Таблица 1.1. Логические операции для четырех возможных комбинаций A и B

Булева алгебра

Булева алгебра[1] позволяет упрощать логические выражения точно так же, как элементарная алгебра упрощает числовые.

Ассоциативность. Для последовательностей, состоящих только из операций AND или OR, круглые скобки не имеют значения. Так же, как последовательности только из операций сложения или умножения в элементарной алгебре, эти операции могут вычисляться в любом порядке.

[1] Названа так в честь Джорджа Буля (1815–1864). Его публикации положили начало математической логике.

$$A \text{ AND } (B \text{ AND } C) = (A \text{ AND } B) \text{ AND } C;$$

$$A \text{ OR } (B \text{ OR } C) = (A \text{ OR } B) \text{ OR } C.$$

Дистрибутивность. В элементарной алгебре мы раскрываем скобки: a × (b + c) = (a × b) + (a × c). Точно так же и в логике выполнение операции AND после OR эквивалентно выполнению операции OR над результатами операций AND и наоборот:

$$A \text{ AND } (B \text{ OR } C) = (A \text{ AND } B) \text{ OR } (A \text{ AND } C);$$

$$A \text{ OR } (B \text{ AND } C) = (A \text{ OR } B) \text{ AND } (A \text{ OR } C).$$

Правило де Моргана[1]**.** Одновременно лета *и* зимы не бывает, поэтому у нас *либо не* лето, *либо не* зима. С другой стороны, оба выражения «не лето» и «не зима» истинны, *если (и только)* у нас не тот случай, когда *либо* лето, *либо* зима. Согласно этой логике, выполнение операций AND может быть сведено к операциям OR и наоборот:

$$!(A \text{ AND } B) = !A \text{ OR } !\, B;$$

$$!A \text{ AND } !B = !(A \text{ OR } B).$$

Эти правила позволяют преобразовывать логические модели, раскрывать их свойства и упрощать выражения. Давайте решим задачу.

> **Перегрев сервера**  Сервер выходит из строя из-за перегрева, когда кондиционирование воздуха выключено. Он также выходит из строя из-за перегрева, если барахлит кулер. При каких условиях сервер работает?

Моделируя эту задачу в логических переменных, можно в одном выражении сформулировать условия, когда сервер выходит из строя:

---

[1] Огастес де Морган дружил с Джорджем Булем. Кроме того, он обучал молодую Аду Лавлейс, ставшую первым программистом за век до того, как был создан первый компьютер.