ТУБ №2:
Алгоритм, свойства правильного алгоритма, способы записи алгоритмов

Понятие алгоритма

«Ой, да зачем мне эти ваши алгоритмы! Разобрался с постановкой задачи — и садись сразу программу писать, чего время зря тратить!»

Такой подход можно часто заметить у начинающих. А мы возьмём да и согласимся!

В самом деле, чего тянуть? Чем быстрее готова программа, тем быстрее будут получены деньги за её разработку и тем быстрее можно будет взяться за следующий проект. Заказчики тоже любят, чтобы их пожелания быстро выполнялись. Ну вот себя вспомните, сдаёте Вы в ремонт телефон, например: «Когда можно будет забрать? — Да Вы погуляйте часик и приходите». Разве не с этой фразы рай на земле начинается?

Правда, законы природы неумолимы: чем сильнее сокращается тот или иной этап разработки программы, тем больше риск ошибок, потому что в спешке легко упустить какие-то нюансы. Но с Вами-то этого не случится! Уже ведь тысячу таких программ написали. А если где-то всё-таки накосячим, то исправим, делов-то! И когда заказчик стоит в дверном проёме с топором наперевес — это он просто поблагодарить пришёл, подарок принёс.

Критично настроенный читатель скажет, что с топором в общественный транспорт могут не пустить. Ага, пусть попробуют! Но если серьёзно, то программисты по сути как раз тем и занимаются, что придумывают алгоритмы и записывают их в виде программ. А ещё целыми днями пьют кофе и играют в настольный теннис, но это если нет на месте начальства и совести.

Чтобы понять, почему разработка алгоритмов неизбежна, хорошо бы для начала дать пару определений. Возьмём их из умных книжек.

> Алгоритмизация — сведение задачи к последовательности этапов, выполняемых друг за другом так, что результаты предыдущих этапов используются при выполнении следующих.
>
> Алгоритм — система правил, чётко описывающая последовательность действий, которые необходимо выполнить для решения задачи.

Хороший пример алгоритма в повседневной жизни — это всевозможные пошаговые инструкции. Те же кулинарные рецепты, например: всё чётко, по полочкам, этот продукт возьмите в таком количестве, потом вон тот добавьте, перемешайте, подуйте, поплюйте, вспомните, что молоко забыли купить.

Но на самом деле этим не ограничивается. Алгоритм — это в первую очередь идея решения. Как при приготовлении изысканного блюда каперсы можно заменить маринованными огурцами, а маскарпоне — смесью творога и сливок, так и при написании программы отдельные правила алгоритма могут быть слегка изменены с учётом возможностей используемого языка программирования или ограничений аппаратного обеспечения, на котором программа будет выполняться.

Нужно понимать, что от выбора алгоритма во многом зависит качество конечного продукта, т.е. программы. Неправильно подобранный алгоритм может, например, сделать программу невыносимо медленной в работе. Проведём аналогию с классической задачей на сообразительность.

> На сковороде помещается 2 котлеты. На поджаривание котлеты с одной стороны уходит одна минута. Как быстрее всего поджарить 3 котлеты?

Решение, которым обычно пользуются незнакомые с этой задачей люди, требует 4 минуты: сначала обжарить с двух сторон две котлеты — это 2 минуты, затем разобраться с оставшейся котлетой — ещё 2 минуты. Этот подход безусловно хорош своей очевидностью и, возможно, отсутствием необходимости напрягать мозг в процессе обжаривания (обжаривания котлет, конечно). На этом, впрочем, его преимущества и заканчиваются.

Между тем, можно уложиться и в три минуты. Первая минута: обжариваем две котлеты с одной стороны. Вторая минута: одну из котлет заменяем той, которая ещё не была обжарена, вторую переворачиваем. Третья минута: одна котлета уже готова, две другие нужно обжарить с одной стороны, это и делаем. Кажется, только что мы справились с задачей на 25% быстрее!

Свойства правильного алгоритма

Ситуации, когда простое и очевидное решение оказывается неэффективным, встречаются повсеместно. Иногда это остаётся незаметным из-за того, что котлеты нужно доставить на другой конец города и выигранная минута легко потеряется на светофорах, но намного чаще котлет оказывается несколько тысяч штук да ещё служба доставки выбирает самый короткий путь вместо того, чтобы избегать дорог с пробками, — и тогда неэффективные решения накапливаются и все вместе ощутимо замедляют работу программы.

Но ещё хуже, когда алгоритм изначально неправильный. Мы, конечно, не будем рассматривать случаи, когда алгоритм не соответствует поставленной задаче. Нас интересует, почему алгоритм для конкретной задачи всё же может оказаться плохим.

Сформулируем 4 свойства правильного алгоритма:

1) дискретность;
2) определённость (детерминированность);
3) результативность (конечность);
4) массовость.

Дискретность алгоритма

Понятие «дискретный» может быть знакомо читателю из математики: там ему противопоставляется понятие «непрерывный». Дискретность алгоритма означает, что в алгоритме могут быть выделены отдельные шаги, причём выполняться они должны последовательно, один за другим, и каждый следующий шаг должен использовать результаты, полученные на предыдущих шагах.

Лучший пример реализации недискретного алгоритма — водитель маршрутки, пытающийся одновременно вести транспортное средство, принимать оплату за проезд, разговаривать по телефону, курить и мысленно планировать выходные, причём используя в планах на выходные деньги, которые заработает только в следующем рейсе. (Справедливости ради, в данном случае правильнее говорить о нескольких параллельно обрабатываемых алгоритмах.)

Определённость (детерминированность) алгоритма

Определённость (или детерминированность) означает для алгоритма, что по завершении выполнения очередного шага можно однозначно сказать, какой шаг алгоритма должен быть следующим, а сами шаги (они же правила алгоритма) должны достаточно конкретно описывать необходимые для решения задачи действия.

«В понедельник и среду врач работает в первую смену, во вторник и четверг — во вторую». В повседневной жизни на случай пятницы, субботы и воскресенья обычно есть какое-то подразумевающееся решение: например, в эти дни у врача выходной. При разработке алгоритмов для программ создать ситуацию неопределённости намного проще: «Если $x > 0$, то оставить его без изменений, если $x < 0$, изменить знак на противоположный» — что делать, если $x = 0$?

Результативность алгоритма

Результативностью (или конечностью) характеризуется алгоритм, который для всех возможных исходных данных позволяет гарантированно получить результат за конечное число шагов.

Пример алгоритма, не являющегося результативным:

> 1) набрать воду в электрический чайник;
> 2) включить чайник;
> 3) посидеть на YouTube 2–3 часа за просмотром видео с котятками;
> 4) подойти к чайнику;
> 5) если вода в чайнике холодная, перейти к пункту 2;
> 6) заварить чай и выпить его.

Массовость алгоритма

Массовость алгоритма проявляется в том, что алгоритм позволяет решить поставленную задачу при всех возможных исходных данных, предусмотренных задачей. Например, если написанный по Вашему алгоритму калькулятор складывает только чётные числа, то о массовости речи идти не будет.

Способы записи алгоритмов

Обратите внимание, что ни для одного из рассмотренных нами примеров не потребовалось знание языков программирования. Алгоритмы не зависят от языка программирования, от того, на каком компьютере будет выполняться программа, — и это очень даже хорошо.

Во-первых, программистам иногда приходится обмениваться опытом решения тех или иных задач. Можно, конечно, было бы сказать, что любой программист просто обязан знать какой-то определённый язык программирования. Тот же Pascal, например. Или C. Или Python. Или Java. Или C#. Или PHP. Или Ruby. Ну, Вы поняли. Древние египтяне тоже были уверены, что все должны знать древнеегипетский, — а теперь мы даже не можем точно сказать, как звали их фараонов.

Во-вторых, есть том самый заказчик, который, как правило, вообще не разбирается в программировании, зато следует какому-то алгоритму, который как раз и нужно запрограммировать, чтобы облегчить заказчику жизнь. Ну не обучать же заказчика программированию! Он ведь тогда сам себе эту программу напишет, а нам-то тоже кушать хочется.

В общем, алгоритм — штука полезная. А значит, пришло время обсудить и способы описания алгоритмов. И здесь не всё так просто. Казалось бы, ну опишите суть алгоритма словами — и никаких проблем. Но со словами как раз и возникают проблемы:

> Жена отправляет мужа-программиста в магазин:
> — Купи батон хлеба, если будут яйца — возьми десяток.
> Муж возвращается из магазина с десятью батонами.
> — Ты зачем столько хлеба купил?
> — Так ведь яйца были…

Естественные языки позволяют формулировать мысль очень неоднозначно. И хотя приведённый пример — это, конечно, сильное преувеличение, использовать естественный язык для описания алгоритмов очень неудобно.

Назовём три самых распространённых способа описания алгоритмов:

1) словесное описание;
2) графическое представление алгоритма;
3) запись алгоритма на алгоритмическом языке (языке программирования).

Каждый из этих способов описания алгоритмов достоин отдельного обсуждения. Пока же кратко охарактеризуем каждый из них.

Словесное описание. Может показаться, что это как раз и есть использование естественного языка. Да, но не совсем. Чтобы избежать двусмысленности, обычно при использовании этого способа очень сильно ограничивают себя в том, какие слова и фразы можно использовать, и заранее договариваются, как их понимать.

Графическое представление алгоритма. По сравнению с текстовой информацией картинки воспринимаются людьми намного проще. Особенно людьми неподготовленными. Иной человек не то что на языке программирования мысль не выразит — в магазине словами объяснить не сможет, что ему надо! А в графической форме изобразит. Тем более, что способов графического представления алгоритмов много, будет из чего выбрать.

Запись на алгоритмическом языке. Способ, который использовали ещё древние греки для доказательства теорем из геометрии: брали палку, чертили на песке эти свои треугольники и говорили «Смотри!» Прошли сотни лет, а ничего не изменилось: пишем программу и показываем её другим, и пусть сами разбираются, что там получилось.

Отличительная черта хорошего программиста — способность по ситуации использовать разные способы описания алгоритмов. Чем больше таких способов в арсенале специалиста, тем выше его квалификация и шансы занять руководящую позицию в команде. А какой программист не мечтает стать руководителем проекта?