Н. А. КРИНИЦКИЙ

АЛГОРИТМЫ ВОКРУГ НАС

Издание второе

**ВВЕДЕНИЕ**

Двадцатый век в области науки и техники принес че­ловечеству много крупных достижений: радио, звуковое кино, телевидение, атомная энергия, космические полеты, электронные вычислительные машины — вот только глав­нейшие вехи, известные каждому. Наверное, не менее известны кибернетика, вирусология, генетика.

Но не всем известно, что крупнейшим достижением на­уки XX в. является теория алгоритмов — новая математи­ческая дисциплина. Теория электронных вычислительных машин, теория и практика программирования *не* могут обойтись без нее. Математическая логика и кибернетика предъявляют на нее свои права. Однако она является самостоятельной наукой, которая готова служить всем наукам, и имеет свое лицо, свой предмет.

Само название — теория алгоритмов — говорит о том, что ее предмет — алгоритмы. Что это такое? Понятие ал­горитма является и очень простым и очень сложным. Его простота — в многочисленности алгоритмов, с которыми мы имеем дело, в их обыденности. Но эти же обстоятель­ства делают его туманным, расплывчатым, трудно поддаю­щимся строгому научному определению.

Слово «алгоритм» происходит от имени узбекского ма­тематика Хорезми (по-арабски ал-Хорезми), который в IX в. н. э. разработал правила четырех арифметических действий над числами в десятичной системе счисления. Совокупность этих правил в Европе стали называть «ал-горизм». Впоследствии это слово переродилось в «алго­ритм» и сделалось собирательным названием отдельных правил определенного вида (и не только правил арифме­тических действий). В течение длительного времени его употребляли только математики, обозначая правила ре­шения различных задач.

В 30-х годах XX в. понятие алгоритма стало объектом математического изучения (прежде им только пользова­лись), а с появлением электронных вычислительных машин получило широкую известность. Развитие электрон­ной вычислительной техники и методов программирования способствовало уяснению того факта, что разработка ал­горитмов является необходимым этапом автоматизации. То, что сегодня записано в виде алгоритма, завтра будет выполняться роботами. В настоящее время слово «алго­ритм» вышло за пределы хматематики. Его стали приме­нять в самых различных областях, понимая под ним точ­но сформулированное правило, назначение которого — быть руководством для достижения необходимого ре­зультата.

Формирование научного понятия алгоритма, ставшее важной проблемой, не закончено и в настоящее время. И хотя теория алгоритмов является математической дисциплиной, она еще не очень похожа на такие широко известные науки, как геометрия или теория чисел. Она еще только зарождается, причем тем исходным материалом, на основании которого должно быть построено широкое на­учное понятие алгоритма, является интуитивное понятие, тоже очень широкое, но недостаточно ясное.

Описывая зарождение теории алгоритмов, мы не пой­дем путем, которым шла история этой науки (хотя о ней и расскажем), а сразу познакомим читателя с современ­ным интуитивным понятием алгоритма. Затем это понятие уточним настолько, чтобы стали возможными изложение традиционных теорий алгоритмов, дальнейшее уточнение понятия алгоритма и, наконец, широкое формальное оп­ределение.

В реальной жизни выполнение всяких действий связано с расходом различных ресурсов: материалов, энергии и времени. Даже производя какие-либо записи, мы расходу­ем ресурсы (например, бумагу, чернила и время). Еще недавно некоторые задачи нельзя было решить из-за слиш­ком большого числа необходимых для этого операций и слишком малой скорости их выполнения. Появление элект­ронных вычислительных машин сделало такие задачи разрешимыми. Это значит, что «математизируя» понятие алгоритма, нужно абстрагироваться, отвлечься от ограни­ченности ресурсов, требуя только их конечности, иначе теория алгоритмов устареет, как только развитие науки и техники позволит переступить через существующие гра­ницы ресурсов.

Алгоритму в интуитивном смысле в книге противопо­ставляется алгоритм в математическом, или формальном смысле. В последнем случае считается, что понятие оп-

ределено методами, принятыми в математике, и основы­вается либо на других понятиях, имеющих математиче­ское определение, либо на первоначальных, описанных настолько четко, что их свойства могут быть приняты за аксиомы новой теории.

Теорию алгоритмов, которой посвящена эта книга, мы называем содержательной в том смысле, что именно ал­горитмы как таковые во всем их разнообразии являются ее предметом. В этом отношении она является противополож: ностью традиционных теорий, которые изучали вопросы существования и несуществования алгоритмов путем све­дения вопросов к исследованию какого-либо одного узкого класса алгоритмов и потому очень многие важнейшие проблемы оставляли вне своего поля зрения. В последнее время традиционные теории алгоритмов нередко объеди­няют названием *логические,* а вышеупомянутую содержа­тельную теорию стали называть *аналитической.*

Для понимания книги не нужна специальная подго­товка, но порою требуется большая внимательность, на­пример, при чтении главы 4, в которой коротко изложены логические теории алгоритмов. Об электронных вычисли­тельных машинах и программировании в этой книге ска­зано очень мало. Лишь столько, сколько нужно для того, чтобы стала ясной связь теории алгоритмов и этой области, которая не только нуждается в результатах теории алго­ритмов, но и порождает многие идеи этой теории.

В заключение автор пользуется случаем выразить глубокую признательность Н. М. Нагорному, оказавшему при подготовке 2-го издания большую помощь.

Глава 1

АЛГОРИТМЫ В ИНТУИТИВНОМ СМЫСЛЕ

§ 1. «Алгоритмические джунгли»

Среди разнообразных правил, с которыми приходится сталкиваться ежедневно и ежечасно, особую роль играют правила, предписывающие последовательность действий, ведущих к достижению некоторого необходимого резуль­тата. Нередко их называют *алгоритмами.* С научной точки зрения к этому названию нужно добавить слова «в инту­итивном смысле».

*Интуицией* называют знание, приобретенное в резуль­тате обширного опыта, но еще не подвергнутое научному анализу и потому недостаточно четкое и строгое. По мере накопления опыта это знание обогащается, и потому наши интуитивные представления о чем-нибудь могут посте­пенно изменяться.

Знания, облеченные в научную, в частности матема­тическую форму, не обладают такой изменчивостью, ха­рактеризуются большой точностью и служат основанием для научных выводов. В тех случаях, когда формализо­ванные знания перестают соответствовать интуитивным представлениям, научные формулировки заменяют новыми.

Разъясним понятие алгоритма в интуитивном смысле на ряде примеров (слова «в интуитивном смысле», когда это не ведет к недоразумениям, будем опускать). К числу алгоритмов не относятся правила, что-либо запрещающие, например: «Вход посторонним запрещен», «Не курить», «Въезд запрещен» (изображается известным каждому во­дителю автомобиля знаком «кирпич»). Не относятся к ним и правила, что-либо разрешающие, такие как «Раз­решена стоянка автотранспорта», «Вход», «Место для курения». А вот — «Уходя, гасите свет», «Идти слева, стоять справа» (на эскалаторе в метрополитене) — это уже алгоритмы, хотя и очень примитивные. Нужно отме­тить одну особенность алгоритма: дискретный характер процесса, определяемого самим алгоритмом. Правило «Во время движения по тротуару придерживайся правой стороны», хотя и является предписанием, но имеет непре­рывный характер и потому не относится к числу алгорит­мов. От него резко отличается текст, который можно встре­тить на некоторых телефонах-автоматах: «Приготовив двухкопеечную монету,

1) опустите ее в приемное отверстие;

2) снимите трубку и ожидайте звуковой сигнал;

3) услышав длинный непрерывный гудок, наберите требуемый номер и ожидайте ответный сигнал;

4) услышав длинные гудки, ждите ответа абонента;

5) "услышав короткие частые гудки, повесьте трубку и получите монету обратно: нужный вам абонент занят».

Подобные правила очень многочисленны и нередко имеют большое значение в нашей жизни. Рождаясь, чело­век сразу попадает в «гущу» алгоритмов.

«Перед кормлением ребенка в бутылочку с кефиром влить пастеризованный охлажденный отвар из риса или другой крупы и сахарный сироп; полученную смесь хо­рошо встряхнуть и подогреть.

Кефир — 5 г, отвар — 45 г, сахарный сироп — 5 г.

Смесь применяется по назначению врача как докорм полутора — двухмесячного ребенка.»[[1]](#footnote-2)

Не думайте, что алгоритмы играют роль только в жизни людей. Вот еще алгоритм.

«Каждого щенка следует кормить отдельно от других, иначе более сильные и активные будут съедать большую порцию.

Подкармливают 3—4 раза в день после того, как щенки пососут мать, равными небольшими порциями, начиная с полстакана молока»[[2]](#footnote-3)

В последнем правиле фраза «...иначе более сильные и активные будут съедать большую порцию» к самому пра­вилу не относится. Такие фразы называют *комментари­ями.* Их отбрасывание на смысл правила не влияет.

Любая женщина (да, и многие мужчины) нередко об­ращаются к поваренной книге и там опять находят алго­ритмы. Приведем и оттуда пример:

«Лимон очистить от кожицы, полученную цедру на­шинковать и ввести в горячий сахарный сироп одновре­менно с желатином. При непрерывном помешивании сироп нагреть до кипения, потом отжать в сироп лимонный сок, добавить лимонную кислоту, профильтровать и охла­дить.

Лимонный сок — 8, сахар — 14, желатин — 3, кислота лимонная — 0,1».[[3]](#footnote-4)

Садоводы, и профессионалы и любители, занимающиеся разведением цветов, вероятно, знакомы со следующим алгоритмом:

«Перед посевом на выровненной поверхности маркером или колышком под шнур проводят бороздки глубиной от 0,5 до 1 см на расстоянии 30—35 см друг от друга. В бо­роздки распределяют семена гнездами (по 8—10 зерен в гнездо). Расстояние между гнездами 15—20—25 см в за­висимости от культуры. Заделывают семена перегноем, посыпая его сверху слоем не толще 0,5—1 см».[[4]](#footnote-5)

Интересные примеры алгоритмов представляют ши­роко известные рецепты, по которым в аптеках приготов­ляют и выдают лекарства. Лишь очень опытные врачи со­ставляют каждый раз индивидуальный рецепт, в большин­стве же случаев его выписывают из специального спра­вочника:

«Rp. Arpenali 0,05 D. t. d. N 20 in tabul S. По 1 таблетке З раза в день».[[5]](#footnote-6)

От точности выполнения подобного алгоритма порой зависит жизнь человека. Интересно, что составной частью такого алгоритма является другой алгоритм (для боль­ного), определяющий применение лекарства и служащий не комментарием, а указанием, которое должно быть на­писано на этикетке и сообщено больному при выдаче ему лекарства.

А вот за столом сидит школьник. Чем он занят? По его словам, он готовит уроки. Какое к этому имеют отношение алгоритмы? Оказывается — большое. Он решает примеры по арифметике, складывает десятичные дроби. Спросите его, как он это делает, и он вам ответит:

«Сперва я одну дробь подписываю под другой так, чтобы одноименные разряды стояли друг под другом. Если в одном из чисел не хватает слева или справа цифр, я до­полняю его нулями.

Потом, переходя от разряда к разряду, я складываю стоящие в них цифры и перенос. Число единиц в получив­шемся результате записываю в одноименный разряд сум­мы, а число десятков принимаю за перенос в следующий разряд.

Самый первый перенос (в младший разряд) всегда считается равным нулю. А если в старшем разряде возни­кает перенос, то перед началом обоих чисел нужно припи­сать по нулю. Процесс оканчивается тогда, когда исчерпа­ются все разряды слагаемых».

Это — алгоритм. Может быть, ученик и не сумеет его изложить так, как здесь написано, и ограничится более лаконичным «складываю числа», но он его обычно вы­полняет.

Не только дети, но и взрослые большую часть своего времени расходуют на выполнение алгоритмов. Многие инструкции и приказы, определяющие наши действия на работе,— это алгоритмы. Даже окончив работу и желая отдохнуть, мы постоянно сталкиваемся с ними. Представь­те себе, что, сняв любительский кинофильм, вы собираетесь его проявить. У вас в руках недавно купленный набор «Химикаты для обращаемых кинопленок». Что же вы находите, вскрыв его? Конечно, химикаты, но кроме них инструкцию. Вот один из ее пунктов.

«Приготовление отбеливающего раствора.

Содержимое пакета «Д» растворить в 500 мл воды при температуре 18—20° С, затем осторожно добавить содер­жимое пакета «Е». Объем раствора довести до 1000 мл. Раствор профильтровать»

Это опять алгоритм.

Всюду алгоритмы. Они окружают нас, переплетаются, проникают друг в друга; шага нельзя ступить, не наталки­ваясь на них. Но как разительно отличаются «алгоритми­ческие джунгли» от настоящих, в которых густые спутав­шиеся растения стесняют нас, цепко держат в плену. Уди­вительным образом алгоритмы не связывают нас, а ведут самыми надежными путями к решению сложнейших про­блем.

**§ 2. Исходные данные и результаты. Массовость алгоритма**

Итак, мы в «джунглях». Но чтобы в них ориентировать­ся, нужно понять, что такое алгоритм. Приведенные выше примеры уже позволяют осуществить некоторый анализ; еще ряд примеров содержит глава 2, в которую можно «подглядывать».

Сразу бросается в глаза, что каждый алгоритм предпо­лагает наличие некоторых *исходных данных* и приводит к получению определенного *искомого результата.* Напри­мер, в § 1 для алгоритма приготовления детской пищи исходными данными являются порции кефира (50 г), кру­пяного отвара (45 г) и сахарного песка (5 г), а резуль­татом — соответствующее количество детской пищи (оче­видно, 100 г). Для медицинского рецепта (алгоритма) исходным данным является медикамент арпенал, исполь­зуемый для лечения язвы желудка, а результатом—-коробоч­ка с двадцатью таблетками и надписью «по 1 таблетке 3 раза в день». Для алгоритма сложения исходным дан­ным является пара слагаемых (чисел), а результатом—сум­ма (опять число).

Создается впечатление, что каждый *алгоритм* — это пра­вило, указывающее действия, в результате цепочки кото­рых от исходных данных мы приходим к искомому резуль­тату. Такая цепочка действий называется *алгоритмическим процессом,* а каждое действие — его *шагом.*

Можно подумать, что каждый алгоритм задает вполне определенный процесс. К сожалению, не совсем так. Толь­ко для самых простых алгоритмов можно говорить об опре­деленных алгоритмических процессах. Для более сложных алгоритмов (мы это увидим на стр. 20) указать определен­ный процесс нельзя. Но для тех алгоритмов, о которых мы уже говорили, существование такого процесса не вызы­вает сомнения. Поэтому пока мы говорим о наиболее про­стых алгоритмах; будем считать, что каждый из них задает вполне определенный алгоритмический процесс.

Но вернемся к анализу тех предметов, которые могут быть исходными данными и результатами. Очевидно, для каждого алгоритма можно брать различные варианты исходных данных. Это видно из того, что, например, для алгоритма приготовления детской пищи можно слова «граммы» при описании исходных данных понимать как «весовые части». Качество получаемой пищи при этом не изменится. Может измениться только ее количество. Для рецепта приготовления лимонного желе, очевидно, так и сделано. Многие алгоритмы остаются в силе для различных вариантов исходных данных. Алгоритм сложения можно применить к парам любых положительных чисел. Алго­ритм дополнительного кормления щенят годится не только, например, для Рексика и Бобика, но и для других щенят.

Замеченное нами свойство алгоритмов, перечисленных в § 1 (их применимость к большому числу вариантов ис­ходных данных), называют их *массовостью.* Долгое время считали, что пригодность алгоритмов для многих частных случаев является настолько существенной и важной их чертой, что должна войти в научное определение алгорит­ма. Это исключало[[6]](#footnote-7) многие правила из компетенции науки (из-за их «недостаточной» массовости) и затрудняло[[7]](#footnote-8) как научные исследования, так и применение их результатов на практике (а вдруг перед нами именно ненаучный слу­чай?), что представляет собой серьезные неудобства.

А между тем ценность представляют и правила (алго­ритмы), применимые даже только к одному-единственному варианту исходного данного. К их числу относятся алго­ритмы пользования различными автоматами (например, автоматом, продающим газеты, или телефоном-автоматом, если они рассчитаны на одну определенную монету), алго­ритмы следования по маршруту, начинающемуся в опре­деленном пункте и ведущему в заданное место, и многие другие. Ценность подобных алгоритмов настолько широко известна, что они положены в основу сюжетов многих литературных произведений (вспомним рассказы о кладо­искателях).

Расплывчатость термина «массовость» подтверждается известным *парадоксом Эвбулида,* который иногда называ­ют *парадоксом кучи.* Сущность его можно передать, задавая себе ряд вопросов и тут же отвечая на них. Один камень — это куча? Нет. А два камня? Тоже нет. А три? В конце концов мы либо придем к выводу, что куч не существует, либо вынуждены будем признать, что есть такое число камней, увеличение которого на единицу приводит к по­лучению кучи. И то и другое противоречит фактам и яв­ляется следствием расплывчатости понятия кучи. И все же просто «отмахнуться» от свойства массовости нельзя. Нужно несколько изменить его трактовку, с тем чтобы устранить указанные выше неудобства.

Какой же смысл следует вкладывать в термин «мас­совость»? А вот какой. Нужно считать, что для каждого алгоритма существует некоторый *класс объектов* (пред-

…………………………………………………….

шаги этого процесса бывают достаточно простыми, а их число не очень большим. Практически число совершенных шагов связано с количеством затраченного на их выполне­ние времени, а может быть (да и наверное так!), с рас­ходом ряда других ресурсов.

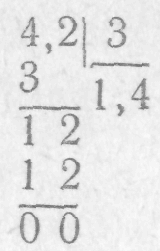
Следует ли при изучении алгоритмов задать для числа шагов какую-нибудь раз и навсегда выбранную границу? Если допустить алгоритмы, выполнение которых требует, например, ста шагов, то почему не допустить и такие, которые потребуют ста одного шага, ста двух шагов и т. д.? Тем более, что развитие науки и техники делает нас богаче ресурсами и позволяет сегодня выполнять раз­личные действия быстрее, чем это было возможно вчера.

Отвлекаясь от реальной ограниченности времени и ре­сурсов, которыми мы располагаем, будем требовать лишь того, чтобы алгоритмический процесс оканчивался после конечного числа шагов и чтобы на каждом шаге не было препятствий для его выполнения. В этом случае и будем считать, что алгоритм применим к исходному данному.

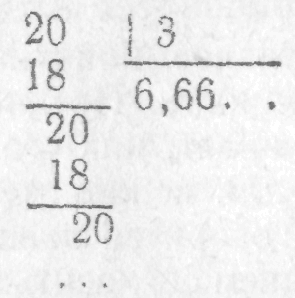
Так как требование завершения алгоритмического про­цесса за конечное число шагов не учитывает реальных воз­можностей, связанных с затратами времени и расходовани­ем ресурсов, то говорят, что при этом алгоритм *потенци­ально* (а не реально) *выполним.*

Неприменимость алгоритма к допустимому исходному данному будет заключаться в том, что алгоритмический процесс либо никогда не оканчивается (при этом говорят, что процесс бесконечен), либо его выполнение во время одного из шагов наталкивается на препятствие (при этом . говорят, что он безрезультатно обрывается).

Проиллюстрируем оба эти случая. Приведем пример бесконечного алгоритмического процесса. Всем известен алгоритм деления десятичных дробей. Числа 4,2 и 3 яв­ляются для него допустимыми исходными данными. При этом получается следующий процесс:



Искомый результат равен 1,4. Но совсем иная картина получится для чисел 20 и 3, которые тоже представляют собой допустимые исходные данные. Для них получится такой процесс:



Сколько бы ни продолжался процесс, он не заканчива­ется и не встречает препятствий. Оказывается, что нельзя получить для исходных данных 20 и 3 искомого результата. Если же оборвать процесс по произволу, то его результат будет только приближением к частному, но не частным. Кстати, алгоритм, предусматривающий обрыв процесса на каком-то шаге, уже не будет тем алгоритмом, который мы рассматриваем.

Теперь приведем пример безрезультатно обрывающе­гося процесса. Представьте себе, что алгоритм состоит из нескольких более простых предписаний, обычно называ­емых *пунктами.*

1. Исходное данное умножить на 2. Перейти к выпол­нению п. 2.

2. К полученному числу прибавить единицу. Опреде­лить остаток *у* от деления этой суммы на 3. Перейти к выполнению п. 3.

3. Разделить исходное данное на *у.* Частное является искомым результатом. Конец.

Пусть целые неотрицательные числа (так называемые *натуральные)* будут допустимыми исходными данными для этого алгоритма.

Для числа 6 алгоритмический процесс будет проте­кать так.

Первый шаг: 6-2=12; переходим к п. 2.

Второй шаг: 12+1 = 13; *у=1;* переходим к п. 3.

Третий шаг: 6 : 1=6. Конец.

Искомый результат равен 6. Иначе будет протекать ал­горитмический процесс для исходного данного 7.

Первый шаг: 7-2=14; переходим к п. 2.

Второй шаг: 14+1 = 15; *у=0;* переходим к п. 3.

Третий шаг: 7:0 — деление невозможно. Процесс на­толкнулся на препятствие и безрезультатно оборвался.

Подчеркнем еще раз, что на практике всегда требуется реальная выполнимость алгоритма, мы же требуем лишь потенциальной выполнимости. Это связано с определен­ной абстракцией.

**§ 4. Понятность алгоритма**

Анализируя интуитивное понятие алгоритма, мы за­мечаем еще одну особенность. Предполагается, что испол­нитель правила всегда знает, как его выполнять. Говорят, что алгоритм *понятен* для исполнителя. В первых книгах по теории алгоритмов говорится даже об их общепонятно­сти. С таким утверждением согласиться нельзя. Даже свой­ство понятности не так просто, как кажется на первый взгляд.

Представим себе, что нами получен некоторый письмен­ный текст. Можно ли утверждать, что он понятен и в ка­ких случаях? Если алфавит, буквы которого использо­ваны в тексте, нам незнаком, то ответ будет один: текст непонятен. Но если все буквы принадлежат знакомому алфавиту, может оказаться, что составляющие его слова нам незнакомы. В этом случае текст тоже непонятен. А если все слова знакомы? Тогда возникает вопрос о способе их со­единения в предложения. Если он противоречит граммати­ческим правилам, опять текст непонятен. А если все грам­матические правила соблюдены? В этом случае неизвестно, понятен текст или нет. Ведь может оказаться, что он явля­ется кодом какого-то другого текста и его скрытый истин­ный смысл не совпадает с его непосредственным смыслом. Если о тексте (кроме него самого) ничего не известно, то назвать его понятным нельзя. Если известно, что текст представляет собой алгоритм, то неопределенность его уменьшается, но незначительно. Полная ясность будет лишь тогда, когда будет известно, что надо делать для того, чтобы этот алгоритм выполнить.

Свойство понятности можно, таким образом, истолко­вать как наличие алгоритма, определяющего процесс вы­полнения алгоритма, заданного в виде текста. Такое объ­яснение «понятности» позволяет предположить, что не только люди, но и животные и некоторые машины могут быть исполнителями алгоритмов.

Итак, каждому исполнителю известен алгоритм, кото­рым нужно руководствоваться для выполнения других алгоритмов, адресованных исполнителям его класса. Ис­полнитель может этот алгоритм знать, например, если он человек, или может быть так дрессирован, чтобы его вы­полнять, если он животное, или может быть так устроен, чтобы его выполнять, если он машина.

В дальнейшем (гл. 9, § 4) читатель узнает, что про некоторые машины (ЭВМ) по отношению к некоторым алгоритмам выполнения алгоритмов (операционным систе­мам) так и напрашивается антропоморфическое выражение «она' его знает». И все же даже у этих машин механизм понимания алгоритмов не тот, что у людей.

Может показаться, что, разъясняя понятие алгоритма, мы апеллировали к этому же понятию и допустили некор­ректное рассуждение, называемое *порочным кругом.* В дан­ном случае это не так (см. § 5).

**§ 5. Рекурсивные определения**

Если хотят ввести новое понятие, то, как говорят мате­матики, ему дают *определение.*

Читатель, безусловно, знаком с так называемыми *пря­мыми определениями.* В них новое понятие выражается через одно или несколько уже известных. Например, если нам уже известно, что такое отрезок прямой, замкнутая ломаная и три, то мы можем определить понятие треу­гольника словами «треугольник — это замкнутая ломаная, состоящая из трех прямых отрезков».

Возможно, читатель слышал и о так называемых по­рочных кругах. *Порочный круг* — это определение, в кото­ром новое понятие выводится либо из самого себя, либо из другого понятия, которое было из него выведено. В ма­тематике порочные круги не употребляются и поэтому по­яснить их определением, взятым из области математики, нельзя. Но неверно было бы утверждать, что порочные круги не могут быть применены с пользой. Так называ­емые толковые словари, в которых разъясняются значения слов, почти сплошь состоят из порочных кругов и тем не менее являются очень ценными научными трудами.

В. И. Даль — составитель первого толкового словаря русского языка. Возьмем для примера из словаря В. И. Да­ля[[8]](#footnote-9) статью «Танцовать». В качестве первого же значения этого слова там приведено: «плясать». Посмотрим теперь статью «Плясать» и в качестве первого значения слова «плясать» прочитаем «танцовать».

Это порочный круг. В последнее время широкое при­менение получили так называемые *рекурсивные определе­ния.* Такое определение в простейшем случае состоит из циклической части, подобной порочному кругу, и прямой части, являющейся входом в циклическую. Это описание рекурсивного определения поясняет лишь его структуру. Нужно еще, чтобы циклическая часть определения не просто выражала новое понятие через него самого, а рас­ширяла его объем. Приведем простейший пример.

Определение. *Словом* называется либо а) пустая строка букв, либо б) слово, к которому приписана буква.

Пункт а) является прямой частью определения. В силу этого пустая строка букв (т. е. то, что стоит между кавыч­ками в записи « ») является словом. Обычно такое слово называют *пустым.* В математике принято допускать су­ществование пустых слов, содержащих 0 букв. В силу второй части определения, приписывая к пустому слову какую-либо букву, мы снова получим слово. Такое слово обычно называют *однобуквенным.* Мы видим, что вторая часть определения не просто говорит, что слово есть слово, а расширяет это понятие. Применяя пункт б) второй раз, мы получим *двухбуквенные* слова и т, д.

Читателю нужно иметь в виду, что этот пример не только служит пояснением рекурсивного определения, но и знакомит нас с принятым в теории алгоритмов понятием слова, которое не совпадает с общепринятым, так как го­ворит только о структуре слова, не требуя от него какого-либо смысла. С общепринятой точки зрения «шмтрс» не является словом, а в смысле нашего определения это самое настоящее слово.

Теперь вернемся к понятию алгоритма. Его связь с понятием *алгоритма выполнения алгоритмов* такова, что допускает возможность рекурсивного определения алго­ритма, что мы И сделаем в дальнейшем (гл. 8, § 7).

**§ 6. Определенность алгоритма**

Обратим внимание еще на одну особенность, присущую каждому алгоритму. Исполнитель алгоритма не только не нуждается в какой-либо фантазии или сообразитель­ности, но, более того, алгоритм не оставляет места для проявления этих качеств, если исполнитель ими обладает. Выполняя алгоритм, действуют механически. Может быть, по мнению читателя это плохо? Может быть, эта черта алгоритма в какой-то мере подавляет творческие способ­ности людей? Ни в коем случае! Люди могут в полной мере проявлять свои творческие способности, разрабатывая алгоритмы. Но после того как они созданы, такое прояв­ление творческих способностей было бы неоправданным расходом психической энергии. Алгоритмы позволяют ее экономить. Таким образом, формулировка алгоритма так точна, что полностью определяет все действия испол­нителя.

Это свойство алгоритмов называют их *определенностью.* Все приведенные примеры алгоритмов демонстрируют его. Если мы внимательно проанализируем алгоритмический процесс, то увидим, что, применяя алгоритм к одному и тому же исходному данному несколько раз, мы всегда будем получать один и тот же результат. Поэтому говорят, что алгоритмы однозначны. Проводя более детальный анализ процесса, путем фиксации промежуточных результатов, получаемых после каждого шага, можно заметить, что при одних и тех же исходных данных возникающие по­следовательности одинаковы. Более того, действия, вы­полняемые на каждом шаге, однозначны независимо от исходных данных алгоритма. Такая особенность алгорит­мического процесса называется *детерминированностью* ал­горитма.

Читателю, настроенному критически, может показаться, что алгоритмы, приведенные в § 1, не очень точны. Например, результат посадки цветов при повторном вы­полнении алгоритма может не полностью совпадать с ре­зультатом первого выполнения (осуществленного, напри­мер, в прошлом году). Однако в пределах требуемой в данной области применения точности оба результата сле­дует признать одинаковыми. Абсолютные точность и одно­значность должны быть присущи математическим алгорит­мам, а «практические» алгоритмы должны быть практи­чески точны.

В приведенных выше примерах определенность алго­ритма заключается в его однозначности и детерминиро­ванности. Важно отметить, что эти особенности процесса сохраняются, если одного исполнителя заменить другим. Нужно лишь, чтобы исполнитель понимал алгоритм.

В § 2 было сказано, что в некоторых случаях нельзя считать, будто каждый алгоритм задает для каждого допу­стимого исходного данного вполне определенный процесс. Сейчас уже можно пролить некоторый свет на это обстоя­тельство. Если алгоритм допускает сразу нескольких исполнителей, то взаимное расположение во времени от­дельных шагов процесса может зависеть от числа испол­нителей и их индивидуальных особенностей.

При повторном выполнении алгоритма другим коллек­тивом исполнителей (или тем же коллективом, но в дру­гих условиях) даже для одних и тех же исходных данных процессы могут быть различными. Свойство определенно­сти будет проявляться в том, что алгоритм в целом и пред­писываемые им операции на каждом шаге будут одно­значными.

Свойство определенности тесно связано со свойством понятности. Мы говорили выше (в § 4), что понятность заключается в том, что исполнителю известен алгоритм выполнения адресованных ему алгоритмов. Если такой алгоритм выполнения обладает определенностью, то (как следствие) и выполненные в соответствии с ним алгоритмы окажутся определенными.

Определенность алгоритма, его механический характер снова наводят нас на мысль о том, что исполнителями алгоритмов могут быть не только люди, но и животные, а также особые машины-автоматы. Этим подтверждается аналогичный вывод, сделанный нами в § 4; в гл. 9 это будет доказано.

**§ 7. Выводы**

Теперь мы уже можем точнее сказать, что такое алго­ритм. *Алгоритм* — это правило, сформулированное на некотором языке и определяющее процесс переработки допустимых исходных данных в искомые результаты. Допустимыми исходными данными для этого правила являются предложения языка исходных данных. Правило ха­рактеризуется понятностью для исполнителя, массовостью (т. е. допустимостью для него всех предложений языка исходных данных) и определенностью. В тех случаях, когда оно применимо к допустимому исходному данному, оно потенциально осуществимо.

В приведенных выше примерах алгоритмы порождают четко видимые алгоритмические процессы, каждый шаг **которых** очень прост. Если алгоритм к какому-либо до­пустимому исходному данному неприменим, то алгорит­мический процесс может для этого исходного данного либо продолжаться неограниченно (быть бесконечным), либо безрезультатно обрываться.

Теперь в «алгоритмических джунглях» уже можно в какой-то мере ориентироваться. Мы понимаем, что такое алгоритм и зачем он нужен. И все же, как читатель увидит в дальнейшем, это понимание еще очень неточно и не всег­да достаточно. К понятию алгоритмического процесса нам придется еще в дальнейшем вернуться, а о некоторых его особенностях мы будем говорить уже в следующей главе. И прежде всего о такой особенности, как простота дейст­вий, выполняемых на каждом шаге.

Понятие алгоритма уже обрисовано довольно ясно, и читатель, встретившись с алгоритмом, сразу поймет, с чем имеет дело. И все же пока что сформировано лишь интуи­тивное представление об алгоритме. Если, опираясь на это представление, мы признаем какое-нибудь правило алго­ритмом, то с точки зрения науки это будет «алгоритм в интуитивном смысле». Интуитивному понятию наука ста­вит в соответствие формальное определение, значительно более точное, но, вообще говоря, более узкое.

На первом этапе своего развития теория алгоритмов не стремилась дать достаточно широкого определения ал­горитма, но соотношение между формальным и интуитив­ным понятиями алгоритма всегда было в центре внимания ученых.

В чем же неточность интуитивного понятия алгоритма? В неточности тех терминов, в которых мы его выразили. До сих пор идут споры о том, что такое язык. Неясен смысл таких слов, как «понятность» и «точность». Научный смысл неясен. А интуитивное значение этих слов ясно.

Глава 2

**СОЗДАНИЕ АЛГОРИТМОВ**

**§ 1.** Роль алгоритмов в науке и технике

Как и в повседневной жизни, роль алгоритмов в науке и технике очень велика. Мы знаем, что в каждой научной или технической области почетное место занимают всевоз­можные справочники. Каждый такой справочник — это в значительной его части сборник алгоритмов, накоплен­ных данной научной или технической дисциплиной. Су­ществуют справочники для конструкторов, для инжене­ров-производственников, для техников, для мастеров и квалифицированных рабочих; справочники для врачей, фельдшеров и медицинских сестер; справочники для архи­текторов и строителей; для бухгалтеров и счетоводов и т. д. Алгоритмы — это богатство науки и техники.

Особое значение имеют алгоритмы, накопленные в математике, потому что математика пронизывает другие науки и ее богатство является богатством всех наук. Уже довольно давно ученые и инженеры заметили, что если удалось получить алгоритм решения какой-нибудь задачи, то можно создать машину, которая решала бы эту задачу, т. е. можно автоматизировать ее решение. Практика уп­рямо подтверждала этот факт. Наука его объяснила пол­ностью только недавно. Читатель познакомится с этим в § 3 гл. 6.

Алгоритмы являются: 1) формой изложения научных результатов; 2) руководством к действию при решении уже изученных проблем и, как следствие: 3) средством, позволяющим экономить умственный труд; 4) необходи­мым этапом при автоматизации решения задач; 5) сред-**ством** (инструментом), используемым при исследовании и решении новых проблем (особенно это касается мате­матических алгоритмов); 6) одним из средств обоснования математики (см. гл. 4); 7) одним из средств описания слож­ных процессов (см. гл. 9, 10).

Нужно сразу подчеркнуть, что алгоритмы составляют важную часть каждой науки, но не исчерпывают ее со­держания. Не менее важны, конечно, понятия и опре­деления, входящие в данную науку, установленные ею факты (в математике — это доказанные теоремы), вы­работанный наукой подход к изучаемым объектам и яв­лениям.

Большая ценность алгоритмов обусловливает интерес к ним. Естественно, что специалисты каждой отрасли науки и техники все время ищут алгоритмы решения различных задач. Каждый новый алгоритм немедленно включается в «золотой фонд» науки. При этом интересны как новые алгоритмы, так и алгоритмы для решения вновь постав­ленных проблем.

Несмотря на то, что алгоритмы очень важны для прак­тики, все же утверждение, будто они изучаются и разра­батываются только в связи с требованиями практики, было бы искажением истины. Нередко создают или ищут алго­ритмы для решения задач, которые сами по себе (по край­ней мере в настоящее время) не имеют практического значения. Иногда причиной для изучения той или иной проблемы служит любопытство, иногда — эстетическое чувство (например, теория кажется недостаточно «изящ­ной» без алгоритма решения какой-либо вычурной задачи, возникающей при ее разработке). Иногда большие силы ученых привлекает к себе некоторая проблема потому, что в ее решении ученые видят для себя «дело чести». Многие охотники за алгоритмами не задумываются над тем, нужны ли, и если нет, будут ли когда-либо нужны добываемые ими экземпляры. Жизнь показывает, что многие научные результаты, возникающие даже без учета нужд практики, рано или поздно находят важные прак­тические применения. Охота за алгоритмами — это увле­кательное и важное дело, которому отдают большую часть своего времени многие ученые.

**§ 2. Как возникают алгоритмы**

Одним из источников алгоритмов является практика, которая предоставляет нам две основные возможности: наблюдение и эксперимент (а также любые их комбинации).

Объектом наблюдения может быть какое-либо живое существо (в частности, человек), умеющее решать какую-либо из возникающих перед ним задач. Описывая его действия, анализируя их зависимость от изменяющихся условий, можно получить алгоритм для решения упомя­нутой задачи. Получаемые этим путем алгоритмы обычно называют *имитирующими.* В более сложном случае объ­ектом наблюдения может быть коллектив совместно дей­ствующих живых существ.

В еще более сложных случаях наблюдают какой-либо процесс, протекающий в неживой природе, организме или в обществе, изучают влияние на него различных факторов; в конце концов может быть получен алгоритм управления этим процессом (который будет эффективным, если суще­ствует реальная возможность изменять определяющие процесс факторы). Алгоритмы, полученные таким образом (в том числе и имитирующие), принято называть *эмпири­ческими.* К их числу относятся приведенные в гл. 1 в виде примеров алгоритмы приготовления пищи, докорма щенят, приготовления лекарств.

Алгоритмы, приводящие к решению интересных для нас задач, иногда можно получить экспериментально, под­бирая действия, приводящие к желаемому результату. Их мы не будем выделять в отдельную группу и условно отнесем к эмпирическим.

В качестве второго источника следует указать науч­ную теорию, из основных положений и установленных фактов которой алгоритмы в некоторых случаях могут быть выведены. С такими алгоритмами мы еще встретимся в данной главе. Из числа приведенных в первой главе к этой группе относится алгоритм сложения положительных десятичных дробей.

Третьим источником новых алгоритмов может являться совокупность уже накопленных. Оказывается, с помощью специальных приемов из имеющихся алгоритмов можно получать новые.

Наконец, четвертым источником алгоритмов может быть изобретательность их разработчика. Алгоритмы ко­дирования и декодирования по заданному ключу проис­ходят из этого источника.

Как бы ни был получен алгоритм, он должен быть обос­нован; это означает, что если алгоритм создан для ре­шения определенной задачи, то необходима уверенность в том, что для всех исходных данных, для которых эта задача может быть решена, алгоритм позволяет получить решение и ни для каких исходных данных не дает непра­вильного результата. Это называется *корректностью* алго­ритма.

Корректность эмпирических алгоритмов обычно про­веряют экспериментально. Какую-то уверенность в их кор­ректности можно получить, если их многократное приме­нение всегда приводит к необходимому результату. Однако одно только многократное экспериментальное подтверж­дение еще не вселяет полной уверенности.

Полная уверенность в корректности эмпирического ал­горитма возникает лишь в том случае, когда полученные c его помощью результаты не только подтверждаются экс­периментально, но и согласуются со всеми другими накоп­ленными и объединенными в научную теорию фактами данной области науки или техники.

Если хотя бы один из даваемых алгоритмом результа­тов противоречит хотя бы одному из ранее установлен­ных и получивших признание фактов, эмпирический ал­горитм нельзя признать корректным (хотя после проверки может оказаться некорректным не алгоритм, а тот факт, которому он противоречит). Корректность теоретически обусловленных алгоритмов гарантируется наличием соот­ветствующих доказательств.

Очень интересен вопрос об установлении корректности алгоритмов, полученных на основе других, ранее разра­ботанных и заведомо корректных алгоритмов. Решение этого вопроса зависит от приема, который был применен для получения нового алгоритма.

Перечислим наиболее часто применяемые приемы.

1) Конструирование алгоритмов. Этот прием заключается в том, что новый алгоритм получают комбинированием уже известных алгоритмов как составных частей.

2) Эквивалентные преобразования ал­горитмов. Два алгоритма называют *эквивалентными,* если: а) всякий вариант исходного данного, допустимый для одного из них, допустим и для другого; б) применимость одного алгоритма к какому-либо исходному данному гарантирует, что и другой алгоритм применим к этому исходному данному; в) результаты, даваемые этими алго­ритмами для одного и того же исходного данного, между собой одинаковы. Замечу, что совершенно неправильно эквивалентные алгоритмы называть различными формами одного и того же алгоритма.

Всякое изменение алгоритма, в результате которого снова получается алгоритм (а не что-нибудь иное) и при этом эквивалентный исходному алгоритму, называется *эквивалентным преобразованием* алгоритма. Примером эк­вивалентного преобразования алгоритма является его перевод с одного языка на другой (если, конечно, он то­чен). Но известны и другие эквивалентные преобразования. С помощью эквивалентных преобразований можно су­щественно изменять алгоритмы. Конечно, разработчика алгоритмов интересуют такие эквивалентные преобразо­вания, в результате которых можно получить алгоритм, который чем-то лучше исходного.

3) Сужающие преобразования. Они при­водят к алгоритмам решения задач, являющихся частными случаями тех задач, для решения которых были предна­значены исходные алгоритмы. Обычно вслед за таким сужением производят эквивалентное преобразование, при котором используют возникающие при сужении возмож­ности улучшения алгоритма.

4) Применение формального метода к нематематической проблеме. Этот прием заключается в том, что нематематическую проблему (при­надлежащую какой-либо технической, естественнонаучной или социально-экономической дисциплине) формулируют математически. При этом может оказаться, что известен алгоритм решения получившейся математической задачи. Этот алгоритм и принимается за искомый. Если готового алгоритма не окажется, то делают попытку его разработки, тем самым обращаясь ко второму из указанных выше ис­точников для получения алгоритмов.

Корректность алгоритмов, полученных путем конст­руирования, не вызывает сомнений, если алгоритмы, ис­пользованные в качестве «строительного материала», дают точные результаты. Если же их результаты являются приближенными, как это часто бывает на практике, то обоснование корректности может требовать сложных ис­следований.

Доказательством корректности алгоритмов, получен­ных с помощью эквивалентных преобразований, является правильность преобразований. Мы не можем пока более определенно говорить об эквивалентных преобразованиях, потому что накопленные нами, читатель, знания об алго­ритмах еще недостаточны.

Корректность алгоритмов, полученных путем сужаю­щих преобразований, обеспечивается проверкой (доказа­тельством) того, что каждый результат, получаемый суженным алгоритмом, тождествен с результатом, который для того же варианта исходного данного дает исходный алгоритм.

Наконец, корректность алгоритма, полученного в ре­зультате применения формального метода, выясняется либо так же, как для эмпирических алгоритмов, либо а) оценкой так называемой адекватности полученной мате­матической задачи (т. е. возможности получения при ее решении результата, достаточно близкого к искомому результату) и б) доказательством корректности алгоритма решения математической задачи.

Алгоритмы, полученные в результате изобретательно­сти разработчика, также требуют обоснования. Обычно с ними поступают либо как с эмпирическими, либо (уже после их получения) проделывают все действия, предус­матриваемые формальным методом.

**§ 8. Задачи на построение алгоритмов**

Можно было бы продолжить изложение и обоснование различных алгоритмов, накопленных в математике. Вза­мен этого мы лишь перечислим некоторые из них, для того чтобы создать у читателя правильное впечатление об их разнообразии и многочисленности. Алгоритмами являются правила для решения систем алгебраических линейных уравнений (существует большое число таких правил), правила дифференцирования функций и правила интегри­рования, изучаемые в курсе математического анализа, правило Штурма — Лиувилля нахождения приближенного значения корня произвольного уравнения f(х)=0. Алго­ритмами являются также и многие другие правила для решения различных задач с помощью циркуля и линейки, известные читателю из школьного курса. Если бы мы вздумали приводить здесь все эти алгоритмы и обосно­вывать их корректность, то, безусловно, не довели бы эту работу до конца из-за ее большого объема.

Читатель уже представляет себе, как появляются ал­горитмы. Обычно алгоритм разрабатывают, имея в виду какую-нибудь задачу. Для ее решения и создают алгоритм. При этом перед математиком возникает задача, коренным образом отличающаяся от той, для решения которой дол­жен быть создан алгоритм. Эту задачу можно сформули­ровать так: «Задан такой-то класс исходных данных и такая-то задача (проблема), для которой эти исходные данные допустимы. Требуется найти алгоритм, решающий указанную проблему», т. е. перед математиком возникает задача нахождения алгоритма.

Очень большое число таких задач на нахождение ал­горитма математики успешно решили. Но целый ряд за­дач на построение алгоритмов упорно не поддавался ре­шению. Приведем одну из них.

В 1742 г. математик X. Гольдбах, петербургский ака­демик, в письме к Л. Эйлеру выдвинул проблему: доказать, что каждое целое число, которое больше или равно шести, может быть представлено в виде суммы трех простых чисел.

Этой проблеме можно придать следующий вид. Найти алгоритм, который позволил бы для любого целого числа, большего, чем 6, найти хотя бы одно разложение на три простых слагаемых. В ответном письме Л. Эйлер заметил, что для четных чисел эта проблема эквивалентна проблеме разложения числа на два простых слагаемых. В 1937 г. И. М. Виноградов доказал, что всякое достаточно большое нечетное число представляется суммой трех простых чи­сел; впоследствии была указана и нижняя граница, пред­полагаемая в доказательстве И. М. Виноградова, так что для нечетных чисел проблема Гольдбаха уже решена, но для четных чисел она не решена и до настоящего времени.

Заметим, что алгоритм ее решения был бы очень не­сложен: если задано четное *N,* нужно было бы (с помощью алгоритма Эратосфена) найти все не превосходящие его простые числа, далее последовательно отнимать каждое из них от заданного N и смотреть, не содержится ли раз­ность среди уже полученных простых чисел. Беда в том, что до сих пор не удалось доказать корректность этого алгоритма, и потому нельзя его считать алгоритмом раз­ложения любого четного числа на два простых слагаемых.

Интересно отметить, что некоторые задачи на нахож­дение алгоритма, долго не поддававшиеся решению, ока­зались неразрешимыми. К их числу, например, относятся той очень древние геометрические проблемы: задача о квадратуре круга, задача о трисекции угла и задача об удвоении куба.

*Задача о квадратуре круга* заключается в следующем: требуется найти метод (алгоритм) построения с помощью циркуля и линейки квадрата, равновеликого данному кругу. Эта задача является массовой, потому что исходным дан­ным для нее может быть любой круг.

*Задача трисекции угла* гласит: требуется найти общий метод (алгоритм) деления произвольного угла с помощью циркуля и линейки на три равные части. Проблема тоже является массовой, потому что исходным данным может быть любой угол.

Наконец, *задача удвоения куба* гласит: требуется найти алгоритм, позволяющий по стороне любого куба с помощью циркуля и линейки построить сторону куба, объем которого вдвое больше объема заданного.

Невозможность решения этих задач была строго дока­зана. Неудача попыток решения некоторых проблем после того, как была доказана невозможность решения некото­рых других проблем, породила определенную «подозри­тельность». Ученые стали опасаться, что затрачивая без успеха свои усилия на решение некоторых проблем, они стараются достигнуть невозможного. Возникло мнение о необходимости выявления неразрешимых проблем. Для задач на отыскание алгоритма это должно сводиться к разработке методов доказательства несуществования ал­горитма.

Нужно подчеркнуть, что алгоритмы для квадратуры круга, трисекции угла и удвоения куба невозможны, если допускать только операции, выполнимые с помощью циркуля и линейки, причем линейка используется только для проведения прямых через пары точек и никак иначе. Уже Архимед предложил метод трисекции угла, в кото­ром допускалась операция, состоящая из двух действий: 1) нанесения на линейке двух точек, копирующих две данные точки чертежа, 2) такого перемещения линейки, чтобы одна из отмеченных на ней точек скользила по пря­мой, а другая по окружности.

Это значит, что за счет расширения набора допусти­мых операций иногда можно часть проблем, для которых нет алгоритма, сделать разрешимыми. Конечно, если ничем не ограничиваться при определении допустимых операций, то многие проблемы станут разрешимыми (но все ли? об этом см. в § 1 гл. 5).

В конце второй главы мы снова пришли к вопросу о том, сколь сложны или, лучше, сколь просты должны быть операции, выполнение которых может быть рассмат­риваемо как отдельный шаг алгоритмического процесса. Во всяком случае, нельзя считать допустимой такую опе­рацию, способ получения результата которой нам неиз­вестен. Иначе многие нерешенные проблемы мы стали бы ошибочно считать решенными.

**§ 4. Антиномии**

В то время, когда А. Пуанкаре провозгласил, что математика обрела, наконец, надежный фундамент, сама арифметика пошатнулась из-за того, что в теории множеств были обнаружены *противоречия (парадоксы),* вошедшие в историю математики под названием *антиномий.*

Парадокс Рассела (открыт в 1902 г.). Если парадокс Кантора возникает для множества, которое со­держит себя в качестве своего элемента, то парадокс Рас­села связан с множествами, не содержащими себя в каче­стве своих элементов. Для удобства будем множество, не содержащее себя в качестве элемента, называть *обычным,* а множество, содержащее себя в качестве элемента,— *не­обычным.*

Парадоксальным является множество всех обычных и только обычных множеств. Чтобы в этом убедиться, проверим, является ли оно само обычным или необычным. Сперва предположим, что оно обычное. Но тогда, будучи множеством всех обычных множеств, оно содержит и себя. Стало быть, оно необычное. Предположив, что оно обыч­ное, мы получили противоречие.

Но, может быть, оно необычное, и дело с концом? Про­верим. Если оно необычное, то, будучи множеством только обычных, оно себя в качестве элемента не содержит, а значит, является обычным. Опять противоречие.

Интересно, что парадокс Рассела может возникнуть и для каталогов, которыми мы уже пользовались для по­строения парадокса Кантора.

Парадоксальным оказывается каталог всех несамоназывающихся и только несамоназывающихся каталогов. Он не может быть самоназывающимся (содержать сведе­ния о самом себе), так как является каталогом только не­самоназывающихся каталогов. Точно так же он не может быть и несамоназывающимся, так как при этом не содер­жал бы сведений о себе (несамоназывающемся), но дол­жен был бы содержать.

Парадокс брадобрея. Парадокс Рассела мож­но сформулировать без привлечения понятия множества. Представим себе, что один из солдат оказался по профес­сии парикмахером. Узнав об этом, командир полка при­казал ему брить всех тех и только тех, кто сам себя не бреет. Все было хорошо, пока не пришло время побрить самого себя. Оказалось, что побрить себя нельзя, так как приказано брить только тех, кто себя не бреет; не брить себя тоже нельзя, потому что приказано брить всех, кто себя не бреет.

Не кажется ли читателю, что положение брадобрея по­добно положению юноши, решившего купить костюм, цена которого меньше 50 рублей (потому что это дешевый ко­стюм) и больше 150 рублей (потому что это хороший ко­стюм)? Разница лишь в том, что условия для покупки костюма всегда противоречивы (не зависят от объекта по­купки), а условия, при которых следует брить, не всегда: их противоречивость или непротиворечивость зависит от объекта бритья.

**§ 5. Выводы из антиномий**

Открытие антиномий потрясло математику и математи­ков как землетрясение. Их наличие, и при этом в такой области, как теория множеств (и логика, потому что па­радокс брадобрея имеет логический характер), заставило опасаться того, что многие результаты, полученные в ма­тематике, тоже противоречивы. Можно было ожидать, что в дальнейшем будут появляться еще новые парадоксы. Все в математике стало казаться неустойчивым, потому что самый ее фундамент дал трещину.

Нужно сказать, что математики реагировали на земле­трясение по-разному. Одни стали во всем сомневаться. Из­вестный математик Ю. Дедекинд после опубликования анти­номии Рассела на некоторое время прекратил публика­цию своих работ. Математик Г. Фреге кончал в это время издание своего большого труда, подготовке которого он по­святил десять лет жизни. В первой же фразе послесловия он говорит, что фундамент построенного им здания поко­леблен парадоксом Рассела. А. Пуанкаре, о котором мы уже говорили, изменил свое отношение к теории множеств. Были, конечно, и такие математики, которые на открытие антиномий никак не реагировали и бездумно продолжали применять теорию множеств, правда, в той ее части, в которой не обнаружено антиномий. Этих математиков обычно называют *последователями классицизма.* Но мно­гие математики стали искать пути устранения проти­воречий.

Некоторые полагали, что противоречия возникают бла­годаря дефектам самой логики и стали пересматривать именно ее.

Другие предполагали, что парадоксы возникают из-за некорректности понятия множества, и занялись поиском такого определения множества, которое было бы свободно от внутренних противоречий.

Третьи — получившие название *формалистов* — при­шли к выводу, что математика должна быть аксиоматизи­рована, а затем все теоремы совершенно формально дока­заны. При этом возник вопрос о формальном языке для математики, таком, чтобы с аксиомами, изложенными на нем, можно было поступать как с определенными комби­нациями символов. Доказательство теорем при этом при­нимало вид переработки исходных аксиом с помощью правил вывода в новые комбинации символов — теоремы. Эти математики разработали новую дисциплину — тео­рию математического вывода, называемую *метаматема­тикой .*

Четвертые усмотрели причину кризиса математики в том, что ряд математических объектов и методов являются неконструктивными. Разъясним последнюю точку зрения.

В теории множеств допускаются «готовые» бесконечные множества, уже существующие, уже завершенные. Завер­шенное бесконечное множество называют актуально бес­конечным. Расходуя ограниченное количество ресурсов на каждом шаге, имеющем фиксированную длительность, построить такое множество ни реально, ни потенциально нельзя. Проверить, обладают ли все элементы такого множества каким-либо свойством, тоже нельзя, так как никакая ограниченная скорость проверки не дает воз­можности охватить их все. Другое дело, потенциально бесконечное, или потенциально осуществимое множество. Такое множество в каждый момент конечно, но есть прием, позволяющий добавить к нему всегда еще несколько (а потом еще несколько, и еще несколько, и так далее и, зна­чит, сколько угодно) элементов. Анализ элементов такого множества можно провести исследованием правила, которое позволяет получать все новые и новые элементы этого «кон­структивного» множества.

Актуально бесконечное множество, будучи недоступ­но ни построению, ни проверке, обязывает нас слепо до­верять тем правилам логики, с помощью которых мы опре­делили свойства его элементов. Указанные правила логики не основываются ни на каких фактах, доступных проверке, во всяком случае в смысле их правильности для акту­ально бесконечных множеств. Значит, говорят математики, придерживающиеся этой четвертой точки зрения, всей части теории множеств, имеющей деле с актуальной бес­конечностью, доверять нельзя.

Будучи едины в своем отношении к актуальной беско­нечности и в своем требовании конструктивности, сторон­ники четвертой точки зрения неодинаково решают вопрос о том, что допустимо в качестве исходного материала для конструкций. Таким образом, эта точка зрения делится на две группы.

Математики первой группы считают, что главным основанием для выбора какого-либо математического объ­екта в качестве исходного для дальнейших построений является его интуитивная очевидность. Читатель, веро­ятно, согласится, что выбор, опирающийся на интуицию, не может не быть субъективным. То, что интуитивно ясно одному, совершенно неясно другому. Это течение в мате­матике получило название *интуиционизма.*

Математики второй группы считают, что исходным мате­риалом для построений могут быть лишь наиболее про­стые математические объекты, применение которых оправ­дано всей практикой человечества, причем количество их типов должно быть ограничено. В качестве основного сред­ства получения новых математических объектов должны служить алгоритмы. Это направление получило название *конструктивного.*

Нужно признать, что ученые всех перечисленных на­правлений внесли большой вклад в сокровищницу матема­тики, получили большое число очень интересных, глу­боких и важных научных результатов. Нужно также подчеркнуть, что наше описание различных точек зрения, возникающих в результате обнаружения антиномий, яв­ляется упрощенным и неточным. Но более подробно осветить этот вопрос автор не имеет возможности.

Из всех указанных направлений мы выделим послед­нее — конструктивное, поскольку оно для обоснования математики приняло на вооружение алгоритмы, и его сто­ронники стали разрабатывать теорию алгоритмов.

Итак, мы ознакомились со второй причиной для раз­работки теории алгоритмов: необходимостью обоснования математики.

На этом автор хотел закончить главу, но вдруг понял, что любознательный читатель, узнав о том, что произошло в математическом мире в начале XX в., несомненно за­хочет узнать — а как же дело обстоит сейчас? Рассыпа­лась ли математика, как карточный домик, или она устоя­ла, преодолела свой кризис?

Конечно, появление антиномий потрясло математику. Верно и то, что кризис математики еще и до сих пор полностью не преодолен. Три четверти столетия — слишком малый для этого срок.

Но в общем-то оснований для отчаяния нет. Хотя тео­рия множеств и не в полном объеме проанализирована и соответствующим образом перестроена, но из нее выде­лена и переработана определенная часть, пока достаточная для обоснования всех остальных математических дисцип­лин. Математики могут пользоваться этой частью теории множеств, избегая, пока что, еще не «разминированных» областей. Сущность антиномий глубоко исследована. Со­временная математика впитала в себя положительные результаты, полученные сторонниками всех перечисленных направлений.

В дальнейшем изложении мы иногда будем пользо­ваться терминологией и понятиями теории множеств. Но, занимаясь алгоритмами, будем пребывать в мире конструк­тивных построений и термины теории множеств применять только для того, чтобы сделать язык более кратким и выразительным.

**ЗАКЛЮЧЕНИЕ**

**§ 1. Может ли машина мыслить?**

**Может ли человек решить алгоритмически**

**неразрешимую проблему?**

Заключение должно быть коротким. Поэтому очень сжа­то скажем о некоторых интересных вопросах, на первый взгляд имеющих отношение к теории алгоритмов.

Некоторые горячие головы утверждают, что машина, как и человек, может мыслить, хотя бы в потенции. При этом высказывают опасение, что со временем машины станут умнее людей и даже, чего доброго, поработят их. Читатель, наверное, хотел бы составить свое мнение по этому вопросу.

Известно, что мышление — это высшая форма движения материи, протекающая в мозге человека. Некоторые «фило­софы» делают отсюда вывод, что машины не могут мыслить. Ход их рассуждений можно проиллюстрировать следующей схемой: селедка — рыба; акула — не селедка; значит, она не рыба.

Действительно, они считают, что человек может мыс­лить, машина — не человек; значит, она не может мыслить. Это неверное рассуждение. Оно не опровергает того, что машины могут мыслить. Прошу читателя не делать из этих слов вывода о том, будто автор считает, что машины могут мыслить.

Чтобы ответить на такой сложный вопрос, нужно преж­де всего решить, по каким признакам можно распознать способность к мышлению. Даже среди людей есть индиви­дуумы, которые не могут мыслить.

Но всегда ли можно решить, является ли человек мыс­лящим или нет? Для решения таких вопросов назначают экспертные комиссии и не всегда получают ясный ответ. Еще сложнее дело в случае, когда вопрос ставится не о конкретной имеющейся машине, а о машинах вообще, о будущих машинах. Автор может только сказать: ни одна из известных машин не мыслит, а может ли вообще машина мыслить? Неизвестно. Но почему бы и нет?

Противники машин в области интеллекта говорят, что признаком мышления является способность решать алго­ритмически неразрешимые проблемы. Алгоритмы, которые соответствовали бы таким проблемам, не существуют. Значит, нет и программ. Отсюда вытекает, что машина (из-за отсутствия программы) не может решать алгоритми­чески неразрешимые проблемы. А вот человек — другое дело. Человек — творец. Он даже алгоритмически нераз­решимые проблемы может решать! Этой точки зрения при­держиваются не только простые смертные, но даже некото­рые специалисты в области кибернетики. Правы ли они? Конечно, нет! Мы помним, что некоторые неразрешимые проблемы заключаются в том, что предлагается построить несуществующий объект. Например, каталог всех несамо-называющихся и только несамоназывающихся каталогов, или каталог всех каталогов, цена каждого из которых на единицу больше максимальной из цен указанных в них книг. Хотелось бы посмотреть, как вышеназванные про­тивники машин решили бы хоть одну из этих проблем. Правда, не все неразрешимые проблемы столь безна­дежны, как названные. Некоторые неразрешимые проблемы содержат в себе разрешимые подпроблемы. Их может решать человек, но для их решения возможен и алгоритм. Другими словами, обращаясь к алгоритмически неразрешимым про­блемам, мы не установим различия между «интеллектуаль­ными» возможностями людей и машин.

**§ 2. Детерминированность машин. Самообучение**

Отмечают еще, что машины, являясь физическими моде­лями алгоритмов, действуют детерминирование Человек же в некоторых случаях может действовать, не ограничивая себя столь узкими рамками. Считают, что деятельность че­ловеческого мозга подчинена законам теории вероятностей и является стохастической.

И здесь автор вынужден занять осторожную выжида­тельную позицию. Связано это с тем, что механизм перера­ботки человеческим мозгом поступающей в него информа­ции еще не изучен. Что же можно о нем сказать, если он нам еще не известен?

Наряду с этим известно, что в состав некоторых машин включают физические приборы, называемые датчиками случайных чисел. Такие машины могут получать в процессе выполнения программ некоторые случайные результаты. Истолкование случайного числа как команды и посылка его в регистр команд ничего хорошего не дает. Но разумное применение случайных чисел позволяет программно моде­лировать реальные процессы, протекающие в условиях помех, и получать близкие к реальному результаты.

Во всяком случае, стохастичность деятельности челове­ческого мозга и обязательная детерминированность ма­шины — это еще не доказанные утверждения. Верны они или нет — покажет будущее.

Для человека характерна способность накапливать опыт и менять в соответствии с ним свое поведение. Говорят, что человек способен к самообучению. Оказывается, что при соответствующей операционной системе и машина ста­новится самообучающейся. Уже составлено немало про­грамм самообучения машины при решении ею той или

иной задачи.

Возможности самообучения пока что малы, так как современная ЭВМ очень похожа на слепого и глухонемого человека. Она может ощупью читать информацию, нане­сенную на перфоносители, и вслепую выдавать инфор­мацию. Безусловно, оснащение машины разнообразными и многочисленными устройствами ввода и устройствами выдачи информации повысит возможности самообучения машин. Но ведь каждый человек сперва обучается и начи­нает это делать с момента рождения. Лишь потом, имея уже огромные запасы информации, он начинает самообу­чаться.

В области самообучения машинам до людей еще далеко. Но машины будущего скорее всего будут обучающимися и самообучающимися. Варварский метод «начинки» машин огромным числом заранее составленных программ, безус­ловно, будет изжит, так как он слишком трудоемок и не позволит эффективно использовать машины будущего.

**§ 3. Сознание машин. Алгоритмическое моделирование**

Если Вы, уважаемый читатель, дошли до этих строк, то автор может считать, что не зря трудился. Автор надеет­ся, что ему удалось показать, какую роль в нашей жизни и науке играют алгоритмы. Мы их встречаем везде и всегда, даже в музыке (здесь ноты — это алгоритмы).

Интерес к науке об алгоритмах вполне естествен. Их повсеместное распространение, их большое значение во всех областях нашей деятельности заставляют интересо­ваться этой наукой.

При первом знакомстве с алгоритмами мы обратили внимание на определенную связь между ними и протекаю­щими вокруг нас процессами. Автор сразу предупредил, что связь не является абсолютной. Когда мы глубже вникли в существо понятия алгоритма, то обнаружили, что один и тот же алгоритм может вызывать ра?личные процессы, ведущие к одинаковым результатам при одинаковых ис­ходных данных.

Впоследствии мы применили алгоритмы для описания процессов. Это может быть методом алгоритмизации, если процесс для нас безразличен. Но это становится способом фиксации процессов в тех случаях, когда для нас важны процессы как таковые.

Нежесткая связь между процессами и алгоритмами дает возможность улучшения хода или содержания некоторых шагов процессов, автоматизации их, повышения эффектив­ности.

Являясь математической дисциплиной, теория алгорит­мов в отличие от некоторых дедуктивных, абстрактных разделов математики непосредственно изучает определен­ные явления реального мира. Ее следует отнести к так на­зываемой прикладной математике.

На страницах этой книги неоднократно упоминался основной тезис теории алгоритмов. Обычно он связывался с традиционными теориями избранных алгоритмов. Как звучит этот тезис, если иметь в виду широкое формальное определение алгоритма, мы уже сказали, но все же повто­рим.

Основной тезис. Для любого алгоритма (в ин­туитивном смысле) над формальным языком L, если его запись можно рассматривать как конструкцию, существует эквивалентный ему алгоритм в широком формальном смысле, имеющий ту же запись.

Это означает, что современная теория алгоритмов охва­тывает все практически важные случаи. Ее дальнейшие обобщения связаны с обобщением понятия конструкции. Сегодняшние потребности теории ЭВМ и программирования

она может обеспечить.

Кончая книгу, хочется заглянуть немного вперед, хотя бы в ближайшее будущее. Если наука эффективна, она должна позволять делать прогнозы.

Из широкого формального определения алгоритма вы­текает, что алгоритм не только может, решая задачу, перерабатывать свою запись, но может перерабатывать и запись своего алгоритма выполнения. Для этого нужно более отчетливо вспомнить его «папу» (алгоритм выпол­нения), который тоже алгоритм и имеет своего «папу», приходящегося нашему алгоритму «дедушкой». Это значит, что, работая, алгоритм выполнения может перерабатывать и себя. Обозначая исходное данное через *sit j,* алгоритм— через *tt,* а результат—через рг\_*},* можем составить формулу из которой указанная возможность и вытекает.

Применяя это соображение к ЭВМ, в 1977 г. (в 1-м из­дании данной книги) мы пришли к выводу о возможности машины, которая обладала бы «способностью» перестраи­ваться, если этого требует программа. Появление таких машин теперь — свершившийся факт.

В § 1 данной главы мы почти допустили возможность мышления машин. Но наша книга посвящена не машинам, а алгоритмам. Сформулируем же упомянутую проблему в терминах теории алгоритмов. Очевидно, вопрос ставится не о том, чтобы любая машина мыслила, а о возможности создания такой машины, для которой можно составить программу мышления. Если отвлечься от ограниченности ресурсов времени и ЗУ, вопрос упрощается: достаточно, чтобы машина была универсальной (типа ЭВМ).

Мы знаем, что ЭВМ являются физическими моделями коллективов алгоритмов. Теперь представим себе условия, в которых обычно находится мыслящий человек. Он рас­полагает некоторыми сведениями, хранящимися у него в мозге; оперируя ими, он по мере надобности обращается к внешним источникам — книгам, справочникам, задает воп­росы другим людям; кроме того, он, может быть, получает сведения путем экспериментов. В конце концов он создает некоторый результат своего мышления. Для упрощения можно считать, что мышление производится без привлече­ния экспериментов. Правда, с самого начала мы приняли еще одно упрощение: ограничились случаем, когда мышле­ние протекает «в тиши кабинета», тогда как нередко чело­век мыслит, находясь во взаимодействии с изменяющимся реальным миром. Между мыслящим человеком и коллекти­вом алгоритмов напрашивается чисто внешняя аналогия. То, что хранится у человека в мозге (его знания),— подоб­но основному операнду; сведения, получаемые извне по мере надобности, подобны потокам частных операндов(см. конец § 10 гл. 8); действия, совершаемые мозгом, напоминают нам процесс выполнения открытого коллектива алго­ритмов. Очень упрощая картину, можно предположить, что дополнительные сведения собраны заранее в некоторой информационной системе (см. § 2 гл. 10) без обновления, присоединяя которую к открытому коллективу алгорит­мов, мы превращаем его в закрытый. В аналитической тео­рии алгоритмов доказано, что каждый закрытый коллектив алгоритмов эквивалентен некоторому одиночному алго­ритму.

Таким образом, в самом простом случае проблема воз­можности мышления машин сводится к вопросу возмож­ности разработки алгоритма, эквивалентно моделирующего работу мозга. Мозг (и даже мозг всех людей) хранит конеч­ный объем сведений и существует конечное время. Не бу­дет грубой ошибкой, если мы посчитаем, что сведения хра­нятся в мозге в форме символьной конструкции (реализо­ванной физически или, если хотите,— биологически). Су­ществует только конечное число символьных конструкций, которые могут быть размещены в ЗУ конечного размера (т. е. в мозге и в информационной системе). Таким образом, перед нами задача о построении алгоритма, входной язык операндов которого конечен. Такая задача алгоритмически разрешима, но только потенциально. Реально ее разрешить методом алгоритмизации нельзя из-за ее большой трудоем­кости.

Но сделанные нами упрощения слишком велики. Алго­ритм мышления, который мы получим, будет слишком при­митивен. Он будет эквивалентен поиску ответа в некото­ром справочнике (хотя и в очень большом). Можно также сказать, что мы решим задачу алгоритмизации только уже осуществленного мышления, что не представляет интереса. Проблема станет тем интереснее, чем привлекаемая ин­формационная система будет допускать обновление инфор­мации. Но даже и в этом случае она остается сильно «уре­занной» из-за того, что набор дополнительных операндов фиксирован. Впрочем, алгоритм мышления, справляющий­ся со своей задачей, в последнем усложненном случае уже «умнее» каждого из породивших его людей, имевших в сво­ем мозге те сведения, которые являются допустимым ос­новным операндом. Остается открытым только один воп­рос: возможен ли такой алгоритм?

Человеческий мозг может мыслить, несмотря на изме­нения, происходящие в мире (и, следовательно, в исход­ных данных для мышления), потому что он сам изменя­ется не только от поколения к поколению, но и в течение жизни отдельного индивидуума. Соответствующая алгоритмическая проблема состоит в вопросе о возможности открытого коллектива алгоритмов, который бы не только перерабатывал операнды, но и видоизменял сам себя.

С проблемой мышления связан вопрос о сознании. Имеется в виду не философское значение этого слова, позволяющее ставить такие кардинальные проблемы, как проблема значение – способность человека отделять себя от остального мира. По отношению к человеку мышление является частью сознания. О мышлении машин мы говорили вне связи с сознанием. Мы как бы предполагали, что мышление машины - это выполнение алгоритма, перерабатывающего информацию эквивалентно тому, как это делает мозг. При таком понимании мышления машин сознание оказывается формой мышления.

Действительно, можно себе представить, что основной операнд открытого коллектива алгоритмов состоит из двух частей, одна из которых является символьной моделью самой ЭВМ (или коллектива алгоритмов), а другая - символьной моделью той части реального мира, которая "известна" машине. Такая пара моделей чем-то похожа на сознание (в том узком смысле, в котором мы условились понимать сознание). Здесь в своих обсуждениях проблемы алгоритмического моделирования сознания мы остановимся, потому что эта проблема выходит за рамки нашей книги. Она очень интересна, но упомянута только для того, чтобы обратить внимание читателя на так называемое алгоритмическое (или программное) моделирование.

Поясним сущность понятия модели. Предположим, что имеется некоторый объект, подлежащий исследованию. Назовем его *прототипом. Моделью прототипа* называется любой объект, так соответствующий прототипу, что описание некоторых его свойств можно перевести в описание тех свойств прототипа, которые нас интересуют. Моделями пользуются тогда, когда исследование самого прототипа слишком сложно или почему-то невозможно. Как видно из этого описания, модель не является копией прототипа (хотя и это возможно), а в такой мере ему соответствует, что исследование некоторых свойств прототипа можно заменить исследование некоторых (вообще говоря, других) свойств модели. Первоначально модели применялись только в технике. Например, перед тем как строить какое-либо сооружение, сперва изготовляли его уменьшенную модель из удобного для этого материала. Модель испытывали и по результатам судили о свойствах будущего сооружения. Немного позже было замечено, что прототип и его модель допускают одно и то же математическое описание или их математические описания могут быть с помощью некото­рого преобразования сведены одно к другому (требуется, чтобы описание модели можно было свести к описанию прототипа).

Наконец поняли, что самое математическое описание прототипа можно считать его моделью (математической). Эта простая мысль явилась результатом длительных ис­следований и размышлений. Сейчас ее справедливость не вызывает сомнений! Математическую теорию моделей мож­но найти в книгах по абстрактной алгебре и в книгах по математической логике. Абстрактная алгебра утверждает, что модель — это множество, на котором задана некоторая совокупность отношений. Логика рассматривает модели систем аксиом — множества, элементы которых таковы, что свойства этих элементов удовлетворяют указанным аксиомам и следствиям из них. Мы видим уже три различ­ных определения математической модели. Два последних являются частными случаями первого. Повторим его: ма­тематической моделью прототипа называется некоторое его математическое описание. Это математическое описание может быть произведено различными математическими средствами. В частности, описание прототипа может быть алгоритмическим, а если оно рассчитано на применение ЭВМ, то — программным.

При алгоритмическом моделировании модель имеет две компоненты: символьную конструкцию из класса некото­рых символьных конструкций и алгоритм или коллектив алгоритмов. Программная модель обычно имеет еще неко­торые компоненты: программу внесения изменений в сим­вольную конструкцию (хранящуюся в запоминающих уст­ройствах) и программу обработки результатов моделиро­вания. При программном моделировании могут встретиться два вида моделей — функциональные и имитационные.

Функциональные модели строят для моделирования од­них алгоритмов в виде других. Например, можно одну ЭВМ, являющуюся, как уже известно, физической моделью алгоритма, моделировать на другой. При этом функцио­нальной моделью первой ЭВМ будет программа второй. В более простом случае алгоритмическая модель эквива­лентна прототипу. В более сложных случаях функциональ­ная модель не эквивалентна, а, как говорят,— равносиль­на прототипу. В первом случае исходные данные и результаты прототипа и модели соответственно одинаковы, во вто­ром случае известен простой прием преобразования исход­ных данных прототипа в исходные данные моделирующего алгоритма и результатов моделирующего алгоритма в ре­зультате прототипа. Программное моделирование такого вида является одним из средств, используемых при разра­ботке ЭВМ и сложных программных комплексов.

Имитационное моделирование применяется для иссле­дования различных процессов путем их условного воспроиз­ведения, при котором каждый шаг процесса-прототипа за­меняется одним или несколькими шагами моделирующего алгоритма. Алгоритмическое описание процессов (см. § 3 гл. 10) является частным случаем имитационного моделиро­вания. Большой интерес представляет так называемое статистическое моделирование, при котором имитация процесса сводится к вычислению какого-либо параметра или ряда параметров, изменяющихся при функционирова­нии прототипа и записи получаемых значений. После того как произведено большое число имитаций при соответствую­щих изменениях имитации, полученные результаты подвер­гаются статистической обработке точно так же, как обычно обрабатывают результаты эксперимента. Статистическое имитационное моделирование иногда позволяет обойтись без дорогостоящих опытов. § 4. Последние замечания

В заключение данной книги скажем несколько слов о вопросах, относящихся к области мировоззрения и являю­щихся по отношению к аналитической теории алгоритмов предварительными. Эти вопросы уже были очень коротко затронуты в § 5 гл. 3 и в § 3 гл. 5. Там упоминалось, что теория алгоритмов является основой конструктивного направления в обосновании математики. Сразу отметим, что конструктивные идеи в математике не сводятся к теории алгоритмов, которая лишь наиболее последовательна в этом отношении. Ограничимся только теорией алгоритмов.. В теории алгоритмов основой для получения всевозмож­ных конструктивных объектов являются некоторые перво­начальные объекты. Математического обоснования их кон­структивности нет. С таким же правом их можно считать и неконструктивными. Доматематическое их обоснование заключается в том, что их практическое существование или (если это операции) практическая возможность их выпол­нения ни у кого не вызывают сомнения. В логических теориях алгоритмов первоначальными являются буквы, сло­ва и некоторые операции (например, в теории нормальных алгоритмов — марковские подстановки); первоначальной является также способность преобразовывать любую сим­вольную конструкцию в слово. Вопрос о том, откуда берут­ся слова-операнды ни явно, ни неявно не затрагивается. В аналитической теории алгоритмов первоначальными являются буквы, связи, и натуральные операции. Символь­ные конструкции из букв и связей получаются уже сред­ствами теории (поэтому в состав теории алгоритмов вхо­дит раздел о формальных языках). Понятие актуальной бесконечности ни явно, ни неявно не привлекается. Разно­чтения символьных конструкций тоже нет. Доматематическим их обоснованием является признание существо­вания реального мира, абстрактными образами объектов и процессов которого являются символьные конструкции и алгоритмы. Применяемые языки поэтому наделены смыслом. Способность преобразовывать любую символьную конст­рукцию в слово не считается первоначальной и обеспечивает­ся возможностью ограниченного применения произвольного выбора из конечного числа элементов (в процессе произ­вольной нумерации с последующим получением результа­тов всех возможных нумераций и выбора из них одного, лексикографически не старше всех других).

Конечно, не запрещается математику изучать аналити­ческую теорию алгоритмов без того, чтобы задумываться о реальном мире, хотя это и нельзя одобрить. Точно так же не запрещается, изучая логическую теорию алгоритмов, считать, что слова являются описаниями реальных объек­тов. Но логическая теория алгоритмов даже не намекает на какой-нибудь способ описания объектов в виде слов, тогда как аналитическая теория такие намеки делает. На­пример, для описания объектов напрашивается применение приема, называемого *структуризацией*, который предусматривает расчленение объекта на составные части — более простые объекты, к которым опять применяется тот же прием, и так до тех пор, пока не будет получена конструк­ция, построенная с помощью связей из таких простых объектов, какие мы уже умеем представлять в виде сим­вольных конструкций. Точно так же сложные операции конструируются из натуральных путем построения алго­ритмов.

1. Детское питание.— М.: Госторгиздат, 1963, с. 107. [↑](#footnote-ref-2)
2. Заводчиков П. А. и др. Справочная **книга по собаковод­ству.—** М.: Сельхозиздат, 1960, с. 152. [↑](#footnote-ref-3)
3. Кулинария,— М.: Госторгиздат, 1955, с. 626. [↑](#footnote-ref-4)
4. Мерло А. С. Советы цветоводам.— Минск, 1965, с. 20. [↑](#footnote-ref-5)
5. Машковский М. Д. Лекарственные средства, т. 1,— М.: Медицина, 1972, с. 200. [↑](#footnote-ref-6)
6. Например, алгоритмы, имеющие один-единственный вариант исходных данных. [↑](#footnote-ref-7)
7. При теоретических исследованиях приходилось бы каждый Газ, встречаясь с алгоритмом, убеждаться, что свойство массовости налицо. [↑](#footnote-ref-8)
8. Д а л ь В. И. Толковый словарь живого великорусского язы­ка.— М., 1955. [↑](#footnote-ref-9)