Визитка

АЛЕКСЕЙ ВТОРНИКОВ, программист. Разработчик ПО для банков и страховых компаний (хотя не отказывается от интересных задач в других областях). Основной «недостаток» – предпочитает командную строку любым IDE

Машина Тьюринга

Гений, которого вынудили умереть

Как бы ни отличались современные компьютеры, и в особенности их программное обеспечение, от того, что когда-то придумал на прогулках по парку английский математик Тьюринг, все они фактически являются потомками его великого изобретения

> Чтобы изобретать, нужно думать около.
>
> **Поль Сурьё (из книги Жака Адамара «Исследование психологии изобретения в области математики»)**

Яблоко с цианидом

Алан Мэтисон Тьюринг, родившийся недалеко от Лондона в 1912 году, прожил короткую и внешне, быть может, не очень яркую, но чрезвычайно насыщенную интеллектуальными и практическими достижениями жизнь. Его ранняя трагическая смерть (Тюринг умер, не дожив нескольких недель до 42 лет) несмываемым пятном лежит на совести демагогов – ревнителей моральной «чистоты», публично осудивших великого английского математика и криптоаналитика за его нетрадиционную сексуальную ориентацию.

Во время Второй мировой войны, когда знания и талант молодого, но уже известного математика были необходимы британскому правительству, армии и флоту для работ по взлому немецких криптографических машин Enigma, это обстоятельство отнюдь не послужило препятствием для привлечения Тьюринга к сверхсекретным работам в Блечли Парке – штаб-квартире правительственного криптографического подразделения. Когда это было нужно – никто, в том числе спецслужбы, почти наверняка прекрасно осведомленные о сексуальной ориентации Тьюринга, и не вспомнил о существовании закона, преследующего таких, как он. Но спустя несколько лет по окончании войны (после заурядной уголовной истории с ограблением его квартиры) это обнаружилось, и британская Фемида «спохватилась».

Заслуги Тьюринга по расшифровке немецких военных кодов, спасшие жизни сотен тысяч людей, были проигнорированы; никто из тех, кто мог его защитить, не шевельнул пальцем. Публично опозоренный и обесчещенный, вынужденный пройти принудительный курс гормонального лечения, психологически сломленный, лишенный возможности работать и преподавать, однажды Тьюринг не выдержал и надкусил яблоко, пропитанное раствором цианида, который он сам же и приготовил. Только спустя 55 лет после его смерти были принесены официальные публичные извинения.

Математика с его смертью лишилась одного из самых ярких и оригинальных мыслителей. Алан М. Тьюринг не только решил одну из труднейших математических проблем; фактически он предложил большее – новый метод решения математических задач (машину Тьюринга), который в конечном счете привел к возникновению computer science и современных компьютеров. Алан М. Тьюринг стоял у самых истоков программирования и искусственного интеллекта. У него не было учеников в общепринятом смысле, но можно утверждать, что любой программист, даже не осознавая того, является если и не учеником, но уж точно его последователем. Именем ученого названа своего рода Нобелевская премия в области computer science – премия Тьюринга, которой удостаиваются те, кто внес выдающийся вклад в эту область исследований. Однако пора обратиться к главной теме статьи – как и для чего Тьюрингом была придумана машина, названная его именем.

Начало

В начале XX века выдающийся немецкий математик Давид Гильберт (1862-1943), много сделавший для развития математической логики и оснований математики, предложил свести процесс доказательства любых математических утверждений к абстрактному манипулированию символами. Для этого прежде всего необходим язык, на котором можно записывать понятия, определения, аксиомы и теоремы математики. Такие языки были известны и использовались математиками уже давно (скажем, язык математической логики, язык арифметики, язык алгебры, язык теории групп, язык теории множеств и т.д.).

Всякий язык включает в себя символы, формирующие его алфавит и правила для образования грамматически верных выражений (примером такого языка может служить любой язык программирования). Подобные выражения представляются в виде цепочек символов конечной длины. Кроме того, необходимы правила вывода, описывающие

то, как можно преобразовывать одни цепочки символов в другие.

Разумеется, должны оставаться и неопределяемые понятия, в истинности которых мы не должны сомневаться (примерами таких понятий могут служить, скажем, понятия точки в геометрии или элемента в теории множеств), в противном случае исчезает отправной пункт для последующих рассуждений.

Имея эти компоненты, можно попытаться выразить все (или почти все) математические утверждения. Гильберт предложил свести все доказательства математики к манипуляциям с символами и выражениями (правильно построенными цепочками символов), отвлекаясь от их математического смысла или содержания. Таким образом, семантическое содержание должно было уступить место чисто синтаксическому. Вся математика должна была описываться этим языком (метаязыком или языком о языке).

В учебнике английского языка, написанном на русском, последний и является метаязыком – языком для объяснения английского.

Так было положено начало теории доказательств (другой распространенный термин – метаматематика) – математической дисциплине, изучающей саму математику.

Понятно, что никто не запрещает наполнять такие доказательства каким либо смыслом, геометрической или физической интерпретацией (например, как это часто делается при изучении дифференциального и интегрального исчислений), но, по мнению Гильберта, все это лишнее – вполне достаточно синтаксической правильности последовательности цепочек строк, представляющих собой доказательство.

Такой подход называется финитизмом, а доказательства, полученные таким способом, – финитными.

В 1928 году Гильберт со своим учеником Вильгельмом Аккерманом написали первый современный учебник по математической логике [1]. Эта небольшая книга, выросшая из лекций, прочитанных Гильбертом в течение трех семестров студентам математикам и которую по первым буквам фамилий их авторов часто обозначают как H&A, сыграла выдающуюся роль в становлении многих разделов оснований математики. В ней впервые связно изложен финитный подход к построению математической логики.

Из следующей системы аксиом логики высказываний H&A (здесь «∨» – это символ логической операции «или» и «→» – символ логического следования; «→» является сокращением для «¬X ∨ Y», где «¬» – символ логического отрицания):

```
X ∨ X → X                                   (a)
X → X ∨ Y                                   (b)
X ∨ Y → Y ∨ X                               (c)
(X → Y) → (Z ∨ X → Z ∨ Y)                   (d)
```

к которым присоединяются две аксиомы с кванторами всеобщности и существования (соответственно ∀ и ∃):

```
∀x F(x) → F (y)                             (e)
F(y) → ∃x (Fx)                              (f)
```

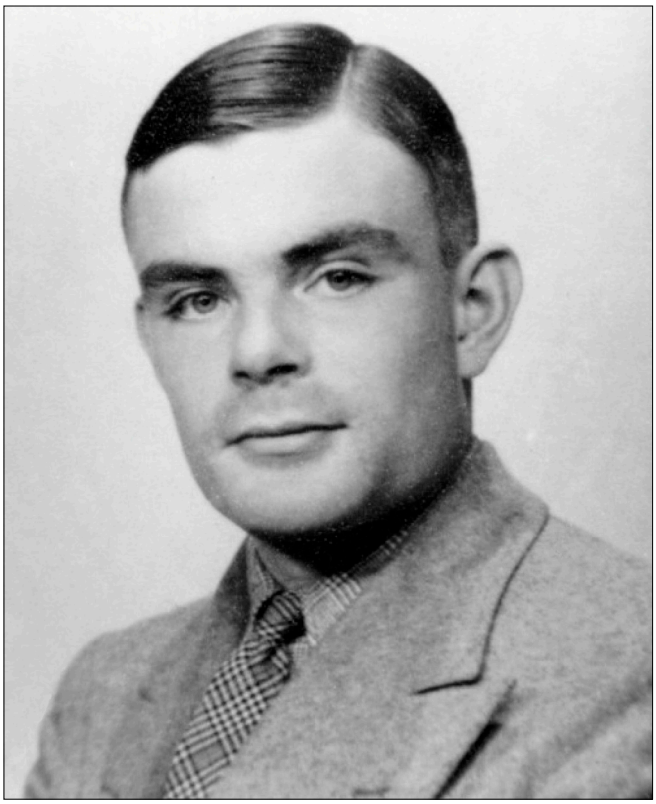
и два правила вывода – подстановки и правило заключения (из X и X → Y следует Y), строится законченный фрагмент математической логики – функциональное исчисление первого порядка (сейчас принят иной термин – логика первого порядка).

Обращаю внимание на то, что появление кванторов вводит в математическую теорию понятие бесконечности. Обойтись без бесконечности невозможно: такие фундаментальные понятия математики, как прямая, натуральный ряд, предел последовательности, множество, – все они опираются на это представление. Понятие бесконечности послужило (и до сих пор служит) источником как математических, так и философских споров и явилось причиной возникновения нескольких направлений в вопросах изучения оснований математики.

В H&A была озвучена и проблема разрешимости (от нем. entscheidungsproblem – этот немецкий термин далее будет встречаться неоднократно). Хотя речь в книге шла прежде всего о математической логике, Гильберт надеялся, что вопрос о разрешимости может быть применен ко всей математике. Позже, в фундаментальном двухтомнике [2], проблема разрешимости была сформулирована как «...проблема нахождения общих методов распознавания общезначимости или же выполнимости логических формул». Проблема разрешимости успешно была решена для частных, но важных разделов математической логики – исчисления высказываний и некоторых случаев логики первого порядка, и это вселяло надежды, что нечто подобное может быть осуществлено и для всей математики.

Положительное решение проблемы разрешимости сулило немалые дивиденды: в случае успеха можно было надеяться с помощью не вызывающих сомнения методов получать доказательства путем выписывания последова-

Алан Мэтисон Тьюринг



тельностей строк, из которых каждая последующая следовала из аксиом и ранее выведенных строк, считающихся уже доказанными.

Рассмотрим, например, систему аксиом Пеано для натурального ряда [3]:

- 0 есть число (1).
- Последующий элемент каждого числа есть число (2).
- Никакие два числа не имеют одного и того же последующего элемента (3).
- 0 не является последующим элементом ни для какого числа (4).
- Аксиома индукции (любое свойство, которое принадлежит 0, а также последующему элементу каждого числа, имеющему это свойство, принадлежит всем числам) (5).

Теперь, имея три «примитивных» (следуя терминологии Б. Рассела) идеи – «0», «число» и «последующий элемент» – можно последовательно построить арифметику натуральных, рациональных, действительных, ... чисел (см., например, [4]).

Используя современную терминологию, проблему разрешения можно сформулировать так: найти обладающие широкой общностью методы (сегодня мы называем это алгоритмами), которые позволят строить цепочки доказательств от посылок к заключениям. То есть требовалось свести проблему доказательства теорем, требующую огромных затрат времени, сил и интеллекта, к процедуре, которая из имеющихся аксиом породила бы все теоремы.

Памятник Алану Тьюрингу в Сэквиль Парке

Или чуть иначе: существует ли метод, который позволяет решить, является ли заданное утверждение теоремой или не является?

Казалось, что теоретически это возможно. Соблазнительно было бы иметь волшебное устройство, вложив в которое необходимые «ингредиенты», можно было получить все возможные следствия. Правда, при этом оставался вопрос этического характера: если бы такое устройство было построено, то что сталось бы с профессией математика, но математика не оперирует такого рода понятиями.

Предвозвестие беды

Первый удар по программе последовал в 1931 году – всего три года спустя после выхода в свет H&A и формулировки Гильбертом проблемы разрешимости. Удар нанес австрийский логик Курт Гедель (который, кстати, годом ранее доказал, что логика первого порядка полна, т.е. всякая истинная формула логики предикатов является теоремой в исчислении предикатов). Удар получился сокрушительным и жестоким: оказалось, что в самой старой и, как считалось, наиболее изученной математической дисциплине – арифметике – могут существовать истинные, но недоказуемые средствами самой арифметики утверждения. Более точно Гедель показал, что можно построить утверждение, которое само о себе утверждает, что оно недоказуемо.

Результаты Геделя невозможно было опровергнуть: его доказательство было логически безупречным. Тогда, может быть, дело в том, что сама арифметика – основа основ всей математики – противоречива?

Проблема непротиворечивости – центральная проблема оснований математики. Математическая система противоречива, если из ее аксиом можно вывести две противоречащие друг другу теоремы.

Такая перспектива была куда страшнее, чем геделевские результаты, ведь это грозило уничтожить все здание, создававшееся титаническим трудом поколений математиков на протяжении тысяч лет.

К счастью, все «обошлось». Несколькими годами позже немецкий математик Герхард Генцен показал, что арифметика непротиворечива (правда, для этого Генцену пришлось привлечь более сильные и уже не финитные методы доказательства).

Все это означало одно – средствами арифметики невозможно доказать все ее теоремы. Это выглядит парадоксально, но Гедель был прав: система аксиом арифметики неполна. Попытка расширить список аксиом, присоединив к ним такую недоказуемую внутри арифметики теорему, была бы обречена на провал: новая система все равно оставалась неполной. Иными словами, система аксиом арифметики существенно непополнима. Результат Геделя получил название «теоремы о неполноте арифметики» (книг, посвященных результатам, полученным Геделем, не счесть; из популярных см., например, [5, 6]).

Результаты Геделя фактически «подрыли корни» под проблемой разрешимости; положительное решение последней стало весьма призрачным. Для отдельных разделов математики решение было возможным, но не для всей математики. Земля стала уходить из-под ног программы Гильберта; надежды на то, что удастся найти метод, позволяющий выводить истинные теоремы чисто синтаксически

(т.е. путем манипулирования символами на бумаге), таяли: если уж в арифметике есть истинные теоремы, истинность которых сама арифметика не в состоянии доказать, не выходя за пределы своих методов, то что тут говорить о других, более сложных областях.

Окончательно надежду Гильберта построить разрешающий метод «похоронили» практически одновременно (в 1936 году) американский логик Алонзо Черч и Алан Тьюринг. При том, что их подходы к проблеме разрешения были совершенно различными, оба – и Черч, и Тьюринг – пришли к одному и тому же выводу: entscheidungsproblem неразрешима. Проще говоря, не стоит тратить силы и время на поиски универсальных алгоритмов – их не существует, и точка. В этой статье будет рассмотрен подход, использованный Тьюрингом.

Плодотворная «дебютная» идея, или О пользе прогулок на свежем воздухе

Можно только предполагать, как 24-летнему математику, который отнюдь не был вундеркиндом (Тьюринг смог поступить в университет не с первой попытки), пришла в голову мысль использовать для исследования entscheidungsproblem машину. Дело, видимо, в том, что в отличие от подавляющего большинства математиков – людей, совершенно неспособных что-либо сделать своими руками, Тьюринг был человеком мастеровым и не чуждым техническому творчеству (что в полной мере проявилось позже, при создании машин для взламывания немецких военных шифров).

Разрушительные результаты Геделя однозначно свидетельствовали – нет никакой гарантии, что имеющихся аксиом хватит для вывода всех теорем, и, возможно, мы никогда не сможем доказать некоторые теоремы (например, знаменитую проблему Гольдбаха из теории чисел) именно потому, что арифметика неполна и в ней чего-то принципиально «не хватает».

Во время послеобеденных прогулок в голове Тьюринга созрел иной план поиска выхода – может, вместо того, чтобы пытаться решить проблему разрешимости аналитически (т.е. примерно тем же путем, каким Гедель получил свои теоремы о неполноте), стоит попытаться построить (разумеется, чисто умозрительно) своего рода математическую машину и исследовать проблему разрешимости с ее помощью. Свои идеи Тьюринг изложил в небольшой статье, опубликованной в трудах Лондонского математического общества в 1936 году [7] и несколько позже в небольшом трехстраничном добавлении к ней.

Замечу, что, как это ни парадоксально, одна из наиболее важных – если не самая важная – в истории computer science статья так никогда и не была переведа на русский язык, и желающему ознакомиться с ней придется читать ее на языке оригинала. Аналогичная ситуация, кстати, сложилась и со статьей Курта Геделя, в которой изложены его фундаментальные результаты о неполноте арифметики.

Идея Тьюринга была великолепной (одним из первых, кстати, это оценил уже упоминавшийся Алонзо Черч, благодаря которому конструкция и получила название «машина Тьюринга»). Тьюринг совершил настоящий научный подвиг – он нашел в себе мужество взглянуть на проблему с иной точки зрения (обращаю внимание читателя на эпиграф к статье). И это привело его к успеху.

Машина, которую построил Тьюринг, была очень простой, и в этом заключался большой смысл – ему нужно было быть уверенным, что ее использование не привнесет в проблему разрешения новых логических сложностей.

Существует несколько настоящих из «крови и плоти» моделей машины Тьюринга. Наиболее интересная из них, на мой взгляд, представлена на сайте http://aturingmachine.com. Там же заинтересованный читатель может найти описание механической и электронной частей модели.

Но простота конструкции отнюдь не значила, что машина Тьюринга мало на что пригодна – вот уж нет. Это полноценный компьютер, нисколько не слабее любых самых современных и мощных машин, разве что очень примитивно устроенный. И в этой самой простоте – залог успеха: простота не закрывала саму проблему, а помогала ее решить.

В многочисленных книгах по математической логике, теории рекурсивных функций и информатике машина Тьюринга описана описана не раз и не два. Все эти описания в равной степени эквиваленты, но порой довольно значительно отличаются в технических деталях. Выбрать из них лучшее – задача сложная. Возможно, это был вариант, предложенный Марвином Минским [8], кстати, одним из лауреатов премии Тьюринга. Я тем не менее не буду отклоняться от варианта, предложенного самим Тьюрингом в его статье.

Тьюринг воспользовался блестящей находкой Геделя – кодированием вычислимых функций (т.е. функций, вычислимых посредством алгоритмов). Класс таких функций – они называются рекурсивными – очень широк: практически все сколько-нибудь значимые математические функции являются вычислимыми. Гедель придумал способ, как сопоставить каждой из таких функций уникальное натуральное число. Теперь можно было рассуждать о вычислимых функциях, оперируя натуральными числами.

В основе кодирования Геделя лежит так называемая основная теорема арифметики: всякое натуральное число может быть единственным (до порядка множителей) образом разложено на произведение простых чисел. Скажем, число 10 единственным образом можно разложить (современный термин – «факторизовать») в виде 2x5; число 20 можно единственным образом разложить как произведение 2 x 2 x 5 или 2^2 x 5 и т.д.

Посредством этого метода Гедель смог закодировать все арифметические утверждения и вывести путем тонких рассуждений свои теоремы о неполноте.

Я здесь не буду воспроизводить доказательство, данное Тьюрингом, о неразрешимости entscheidungsproblem – мне не хочется создавать иллюзию понимания; само по себе доказательство просто, но «головоломно» (эта характеристика принадлежит М. Минскому).

В действительности доказательство опирается на возможность задания машине Тьюринга описания самой себя. Машина Тьюринга, способная выполнять программы, описывающие ее работу (т.е. программа, получающая саму себя на входе), называется универсальной машиной Тьюринга. Ничего необычного в таком «самоприменении» (от англ. self-referencing) нет – это пример рекурсивной функции, подобной классическим функциям факториала или последовательности Фибоначчи, правда, несколько необычного вида.

Будет куда интереснее и полезнее либо найти и попытаться прочитать статью самого Тьюринга, либо воспользоваться хорошим университетским учебником по теории рекурсивных функций (я рекомендую, помимо упомянутой выше книги М. Минского, следующие: [9-11] и особенно вышедшую совсем недавно научно-популярную книгу [12].

Немногие математики того времени сумели понять значение статьи Тьюринга. Из крупных математиков, кроме Алонзо Черча, следует упомянуть, пожалуй, лишь Джона фон Неймана, который не только очень высоко оценил результаты, полученные Тьюрингом, но и активно рекомендовал его статью своим коллегам. Увы, такова судьба многих великих открытий. Справедливости ради надо отметить, что работа Тьюринга недолго была в тени, и менее чем через 20 лет математики отдали ему должное.

Важное открытие

Машина Тьюринга (далее я буду использовать сокращение «МТ») – это воображаемое устройство, состоящее из бесконечной ленты, разделенной на ячейки и головки чтения/записи. В оригинальной МТ ячейка может содержать символы трех типов: 0, 1 и быть пустой (в последнем случае ячейка не содержит ни 0, ни 1, и ее удобно обозначить через «-»).

Замечу, что МТ не налагает ограничений на используемый набор символов: можно применять любые символы, которые программист сочтет нужным, – это существенно упрощает составление программ для МТ (вскоре будет приведен соответствующий пример).

В каждый отдельный момент времени головка находится над одной из ячеек и считывает ее содержимое. В зависимости от содержимого ячейки МТ может выполнить одну из следующих операций: стереть текущее содержимое ячейки (операция «E»), напечатать что-либо в ячейке («P»), сдвинуть головку вдоль ленты на одну ячейку влево («L») или вправо («R»). Полезна и еще одна операция – останов («H»).

Всякий раз МТ находится в одном из помеченных состояний (такая пометка может быть чем угодно – буквой, словом или числом – главное, чтобы МТ была в состоянии отличить одно состояние от других). Работа МТ состоит в считывании символа из ячейки, над которой находится головка, выполнении некоторых операций и переходе к следующему состоянию.

Следующее состояние – вовсе не обязательно состояние, текстуально следующее за текущим состоянием; оно может располагаться на значительном «удалении» от текущего.

Операций может быть не одна, а несколько (например, стереть символ, напечатать 1, сдвинуть головку влево, что можно записать как «E, P1, L»). Разумеется, проще всего работу МТ продемонстрировать на нескольких примерах, чем я сейчас и займусь.

Пример 1

На пустой ленте напечатать пять единиц и остановиться.

Состояние	Текущий символ	Операция	Следующее состояние
a	-	P1, R	b
b	-	P1, R	c
c	-	P1, R	d
d	-	P1, R	e
e	-	P1, H	

Легко проверить, что эта программа МТ действительно печатает на пустой ленте последовательность «11111», после чего останавливается.

Пример 2

На пустой ленте напечатать последовательность из 0 и 1.

Состояние	Текущий символ	Операция	Следующее состояние
a	-	P1, R	b
b	-	P0, R	a

Эта программа МТ, никогда не останавливаясь, будет печатать последовательность «101010...». Это не должно нас беспокоить, т.к. лента МТ также бесконечна. Главное в этом примере то, что из него видно, как в программе для МТ можно организовать цикл.

Пример 3

На пустой ленте напечатать последовательность из 0 и 1, разделенных пустым символом (это, кстати, самый первый пример из статьи Тьюринга, в котором я только изменил обозначения состояний).

Состояние	Текущий символ	Операция	Следующее состояние
a	-	P0, R	b
b	-	R	c
c	-	P1, R	d
d	-	R	a

На ленте будет напечатано «0-1-0-1-0-1-0...», и так до бесконечности (я использовал символ «-» для обозначения пробела).

Пример 4

Сложение чисел. На ленте имеются два числа, представленные в единичной форме (т.е. число «3» представляется как последовательность «111»), разделенных одной пустой ячейкой. Записать на ленте последовательность, представляющую собой сумму двух чисел.

Пусть информация на ленте МТ представлена в виде «-111-11111-». По окончании работы программы на ленте должна остаться последовательность «-11111111-». Головка МТ в начале работы считывает самую левую единицу.

Состояние	Текущий символ	Операция	Следующее состояние
a	1	R	a
a	-	P1, R	b
b	1	R	b
b	-	L	c
c	1	E, H	

Обращаю внимание, что начальных (текстуально в программе – самых левых) состояний может быть сколько угодно. МТ может находиться в состоянии b и считывать символ 1, а может находиться в том же состоянии, но считывать пустой символ. Таким образом, программы МТ носят условный характер: при заданном состоянии и текущем символе требуется выбрать ту или иную операцию.

Программа работает так: первая последовательность из единиц сканируется слева направо до тех пор, пока не будет обнаружена ячейка, содержащая пустой символ. В эту ячейку записывается единица. Теперь общее число единиц на ленте ровно на одну больше, чем надо. Чтобы «скорректировать» сумму к правильному значению, необхо-

димо найти самую правую единицу, стереть ее и остановиться. Упрощенно ход работы МТ выглядит так: «-111-11111-» «-111**1**1111**1**-» «-11111111-», где выделены единица, записанная вместо пустого символа, и «лишняя» единица в конце промежуточной последовательности.

В заключение я хочу предложить читателям несколько упражнений (в порядке возрастания сложности) для развития навыков программирования для МТ.

Упражнение 1: написать программу, перемножающую два числа, записанных в формате предыдущего примера.

Упражнение 2: написать программу, вычитающую одно число из другого.

В этом упражнении есть маленький подвох: чему равна разность двух чисел, обозначенных как «a» и «b», если a < b? Ведь для натуральных чисел (т.е. для чисел из ряда 0, 1, 2...) операция вычитания не всюду определена – отрицательных чисел «не предусмотрено». Обычно математики определяют операцию вычитания на натуральных числах так: если a ≥ b, то разность вычисляется как обычно; если a < b, то разность a и b равна 0 (т.н. «усеченная» разность).

Упражнение 3: проверка правильности вложения скобок. Имеется строка, состоящая из произвольного количества символов «(» и «)». В строке «(())» скобки сбалансированы, равно как и в последовательности «((()))». А вот для строк «()(» и «)()(» скобки не сбалансированы. Написать программу для МТ, которая проверяет баланс скобок.

Тут основная сложность в том, что количество скобок и количество уровней их вложенности заранее не известны (указание: удобно расширить набор символов МТ, включив в него дополнительные символы «(» и «)» и тем самым избегнув проблемы кодирования скобок нулями и единицами).

Не скрою, это упражнение весьма сложное: надо будет предусмотреть все возможные варианты сочетания символов, так что любителям ребусов будет над чем поломать голову. Обязательно проверьте решение на нескольких вариантах последовательностей скобок.

Несложно написать программу на любом высокоуровневом языке программирования, которая эмулирует работу МТ. Такой программе можно «поручить» проверку программ для МТ. Много таких программ можно найти в Интернете (правда, не исключено, что для них придется немного поправить исходные тексты программ, приведенных в статье).

Упражнение 4: подпрограммы. В этом упражнении я предлагаю поразмышлять над тем, как смоделировать на МТ концепцию подпрограмм. Где и как должен сохраняться адрес возврата? Это интересная задача для любителей нестандартных решений.

***

Когда-то выдающийся американский физик-теоретик Ричард Фейнман в своих знаменитых «Фейнмановских лекциях по физике» (правда, по иному поводу) сказал: «...мы сможем продемонстрировать одно из прекрасных свойств... – как много в ней удается вывести из столь малого».

Работа Алана М. Тьюринга по праву может считаться классической иллюстрацией слов Фейнмана. Действительно, кто бы мог предположить, что математическая статья в специализированном журнале по малопонятной и далекой от повседневных проблем тематике, к тому же написанная практически не известным математиком, окажет такое воздействие на нашу цивилизацию. Но не прошло и 20 лет, как компьютеры стали производиться в промышленных масштабах. Поначалу это были огромные многотонные чудовища, пожирающие сотни киловатт электроэнергии и способные обогреть выделявшимся теплом целый квартал. Их программирование было немногим легче, чем программирование МТ. Но очень скоро компьютеры, причем куда более мощные, чем, вероятно, даже Тьюринг мог себе вообразить, уменьшились в размерах до крохотных пластинок и капсул, встраиваемых в автомобили, телевизоры, стиральные машины и космические спутники.

Как бы ни отличались современные компьютеры и в особенности их программное обеспечение от того, что когда-то на прогулках по парку придумал Тьюринг, все они фактически являются потомками великого изобретения – машины Тьюринга.

Что дальше?

Эта статья ни в малейшей степени не претендует на сколько-нибудь полное изложение вопросов, касающихся entscheidungsproblem. Заинтересовавшегося читателя я отсылаю к любому приличному учебнику по математической логике, скажем, А.Черча, С.Клини, Э.Мендельсона или Дж. Шенфилда (имеются их переводы на русский язык). Его ждет нелегкая работа, но результаты этой области математики столь красивы и элегантны, что такая работа того стоит, уж поверьте! EOF

1. Гильберт Д., Аккерман В. Основы теоретической логики. – 2-е изд. – М.: URRS, 2010.
2. Гильберт Д., Бернайс П. Основания математики. – М.: «Наука», 1979, 1982.
3. Рассел Б. Введение в математическую философию. – Новосибирск: Сибирское университетское издательство, 2007.
4. Ландау Э. Основы анализа. Действия над целыми, рациональными, иррациональными, комплексными чисолами. – 3-е изд. – М.: URRS, 2010.
5. Нагель Э., Ньюмен Дж. Теорема Геделя. – 2-е изд. – М.: URRS, 2010.
6. Успенский В.А. Теорема Геделя о неполноте. – М.: «Наука», 1982.
7. Alan M.Turing. On Computable Numbers with an Application to the Entscheidungsproblem. – Proceedings of the London Mathematical Society. Second Series 42: p. 230-265, 1936.
8. Минский М. Вычисления и автоматы. – М.: «Мир», 1971.
9. Булос Дж. и Джеффри Р. Вычислимость и логика. – М.: «Мир», 1994.
10. Мальцев А.И. Алгоритмы и рекурсивные функции. – М.: «Наука», 1986,
11. Картленд Н. Вычислимость. Введение в теорию рекурсивных функций. – М.: «Мир», 1983.
12. Charles Petzold.The Annotated Turing: A Guided Tour through Alan Turing's Historic Paper on Computability and the Turing Machine. – Wiley Publishing, Inc., 2008.