Жизнь в интересные времена

Мое время было временем контрастов. Контраст был между моим гуманитарным образованием и моей последующей научной профессией; между моей работой в промышленности и моей научной карьерой; между изучением естественных языков и занятиями искусственными языками, применяемыми для общения с компьютерами.

Мое образование следовало старым традициям — передо мной были учения и примеры великих мыслителей и писателей классических эпох, заложивших основы интеллектуальной и философской культуры Западной Европы. Я изучал множество разнообразных предметов: латынь, греческий, философию, русский язык, статистику и лингвистику. По контрасту, моя профессиональная карьера относилась к новой ветви науки, к науке о компьютерах. Я ознакомился с нею вначале из промышленного опыта, участвуя в проектировании компьютерных архитектур, языков программирования, компиляторов и операционных систем; я испытал успехи и неудачи в качестве менеджера инженерных проектов программного обеспечения. Затем я перешел к академической жизни и был профессором информатики в двух университетах. В обоих университетах я ввел учебные курсы для студентов и аспирантов по моему предмету, занимался чисто теоретическими исследованиями в области программирования и поддерживал тесные контакты с промышленностью. Теперь я вернулся к работе в программной промышленности, чтобы увидеть, как результаты академических исследований применяются в интересах всех пользователей и производителей компьютерных программ. Сегодня я считаю себя вправе поразмыслить в этих особых

обстоятельствах, каким образом мое образование и опытсформировали меня и подготовили меня к деятельности в области современных технологий. Мне хотелось бы выразить благодарность тем, кто более всего повлиял на мою философию жизни, интеллектуальным лидерам нашего времени и прошлого. Многие из них стояли уже на этой почетной трибуне как лауреаты премии Киото за прошлые годы. Я присоединяю свою благодарность за эту честь к благодарности других лауреатов.

Кентербери: классики.

Я начну мою историю с 1947 года, когда я сдал экзамены и получил стипендию для оплаты обучения и жилья в одной из респектабельных, но не столь знаменитых английских частных школ. Это была Королевская школа в Кентербери, ведущая свое происхождение от монастыря, учрежденного св. Августином в 597 году, в начале его христианской миссии в Англии. Среди выпускников этой школы было два других христианских святых. Около пяти лет я жил в прекрасных окрестностях Кентерберийского собора, имея перед глазами его высокую и изящную башню и слыша звон его колоколов.

В первый год моего учения в Кентербери я занимался десятью предметами, в том числе английским языком, литературой и историей, французским языком, классическими языками — латинским и греческим — и двойной порцией математики, элементарной и высшей. Это не оставляло мне времени для изучения какой бы то ни было другой науки, разве что в виде хобби — по одному уроку в неделю. Меня очень привлекала математика, но на втором году мы должны были выбрать для дальнейшего изучения только два предмета. В традиционных частных школах

предполагалось, что мальчик, хорошо усвоивший классиков, будет продолжать изучение латинского и греческого с небольшой добавкой священного писания или французского.

Наша еженедельная домашняя работа включала перевод отрывка из классического автора на английский, или (что было труднее) перевод куска английской прозы на латинский или греческий. Конечно, я должен был изучить грамматику этих языков. Она значительно сложнее английской, поскольку каждое существительное и каждое прилагательное имеет один из трех родов и пять или шесть падежей, как в единственном, так и во множественном числе. Для определения формы слова было четыре или пять строгих правил, но, кроме того, было множество неправильных форм, которые надо было заучивать отдельно. В каждом сочинении по латыни или по греческому каждое прилагательное должно было согласовываться с существительным, к которому оно относилось, в роде, числе и падеже. Я всегда тщательно проверял свою работу, следуя правилам, но все же продолжал делать ошибки,— их называли вопиющими ошибками, может быть, потому, что мои учителя так легко их находили. В моей

последующей жизни, когда мне пришлось изучать проектирование и реализацию искусственных языков для общения с компьютерами и управления ими, я прилагал все старания, чтобы грамматические правила этих языков были как можно проще и логичнее; я заботился о том, чтобы сам компьютер мог быстро обнаруживать и надежно реагировать на любое нарушение правил во входном тексте.

Несмотря на мои часто повторявшиеся грамматические ошибки, я, в действительности, получал удовольствие от упражнений в переводе и сочинении. Задача состояла в том, чтобы понять назначение, содержание, развитие мысли и стиль некоторого текста на одном языке, а затем реконструировать то же сообщение на другом языке, с другой структурой предложений и другими коннотациями слов. Я усвоил простые правила риторики: равновесие и контраст, повторение и вариации, тезис и антитезис. Это последнее предложение само по себе уравновешенно и контрастно, иллюстрируя тем самым описываемые им правила. Я должен был тщательно переработать его трижды. Как всегда, наилучшие результаты получаются, если испытать много альтернативных формулировок, прежде чем выбрать лучшую. Это в точности то, что я применял в дальнейшем во всей моей научной работе и преподавании, чтобы объяснить мои и чужие идеи как можно более ясным и запоминающимся образом и представить мои аргументы и контраргументы вполне честно, но в самом сильном выражении.

Эти же навыки я пытался передать моим аспирантам, излагавшим результаты своих собственных исследований. Несмотря на официальное изучение классиков, я, к счастью, не вполне забросил мои математические интересы. По некоторым причинам меня особенно заинтересовали вероятности. В одной детской книжке о загадках, фокусах и парадоксах мне попался известный парадокс теории вероятностей: в классе всего лишь из двадцати трех человекесть большой шанс найти двух человек с одинаковым днем рождения. Я выбрал наугад фамилии из списка всех мальчиков школы, выписал их вместе с их днями рождения и убедился в том, что правило соблюдается. Пытаясь выяснить, почему это так, я открыл биномиальные коэффициенты.

Затем мне захотелось обобщить этот результат на большие коллективы, с большим числом людей, имеющих общий день рождения. Это требует вычисления коэффициентов полинома. В конце концов, я показал, что должен быть такой день, когда не менее шести учеников школы имеют общий день рождения. Это можно было проверить по полному списку учеников школы, но я остановился, найдя день рождения одновременно пяти учеников. Наконец, я обнаружил в школьной библиотеке книгу Ланселота Хокбена под названием «Математика для миллионов». Из этой книги я узнал, что коэффициенты полиномов были уже открыты Ньютоном, что меня не слишком разочаровало. Конечно, чтение книг дало мне возможность быстрее продвинуться к пониманию математики вообще и теории вероятностей в частности. Я думаю, что получил большую пользу, занимаясь математикой в качестве хобби, то есть чем-то таким, что делают для решения интересных задач, вместо фиксированной массы знаний, которые просто заучивают, чтобы повторить на экзамене. Я подозреваю, что если бы я выбрал в качестве главного экзаменационного предмета не классиков, а математику, то я мог бы ее разлюбить. Конечно, в то время у меня не было шансов изучить какой-нибудь из разделов математики, которые я теперь считаю интересными и важными для теории программирования и информатики.

Наряду с классической грамматикой и сочинениями наша программа включала также чтение избранных произведений классической литературы, в том числе таких авторов как Цезарь и Цицерон, Демосфен и Платон. Мы должны были приготовить наш урок накануне вечером, изучив очередной отрывок заданного текста, а на следующий день читать его вслух, переводить на английский и обсуждать язык, содержание, контекст и общие вопросы, касающиеся _____этого текста.

Однажды заданным текстом был диалог греческого философа Платона. В этом диалоге фессалийский аристократ Менон спрашивает Сократа: «Можно ли научиться добродетели?» В ответ на это Сократ начинает подробно излагать свои взгляды на обучение и на знания вообще: знание есть прямая форма восприятия действительности в абстрактном смысле, нечто вроде нашей памяти о событиях своего прошлого. Цель обучения состоит лишь в том, чтобы живее напомнить нам то, что мы уже знаем, но, может быть, забыли или смутно вспоминаем. В качестве примера Сократ выбирает геометрическую задачу — построить квадрат вдвое больше заданного квадрата. Он обращается к мальчику — рабу Менона, не имеющему, естественно,

предварительных геометрических знаний. Сократ рисует на песке у своих ног изображение квадрата, чтобы объяснить задачу. Вначале мальчик дает неправильный ответ — квадрат вчетверо большей величины. Ограничиваясь только дополнительными вопросами, Сократ побуждает мальчика найти свою ошибку. Постепенно мальчик сам приходит к правильному решению. Объяснение Сократа состоит в том, что мальчик в действительности знал ответ все время, но забыл его, и, наконец вспомнил, когда вопросы разбудили его память. Его знание имеет начало от времени, предшествующего рождению, когда душа человека непосредственно воспринимает абстрактные понятия математики, не отвлекаясь и не обманываясь чувственными впечатлениями, независимо от конкретного выбора размера и цвета чертежа, от неточности его изображения.

Платон обобщает эту аналогию действия прямой интуиции за пределы математики на более важные философские идеи, такие как добродетель и справедливость, понимание которых требует еще более длительной подготовки. В то время платоново учение об идеях не особенно привлекало мой скептический ум. Однако гораздо позже точка зрения Платона оказалась важной для одного из моих главных открытий в теории программирования, которое связывает абстрактную математическую идею с ее конкретным представлениемв компьютере. Рассмотрим, например, основное представление о целых числах 0, 1, 2, 3,…, с помощью которого мы учимся считать. Ввиду неизбежного ограничения числа микросхем, которые могут быть реализованы на поверхности кремниевого чипа, компьютеры не могут экономно представить неограниченный или бесконечный ряд целых чисел, составляющий предмет арифметики. Вместо этого приходится выбрать фиксированное число разрядов, например, 32. Каждый разряд имеет лишь небольшую область значений: в человеческой арифметике есть десять цифр, а в компьютерной — только две.

Естественно, идея целого числа не зависит от выбора представления. Значение седьмого простого числа, конечно, более абстрактно, чем любое из его представлений: 17 в десятичной форме или 010001 — в двоичной. К счастью, поскольку математические абстракции существуют независимо от их представлений на бумаге или в компьютере, мы можем использовать самую математику для изучения отношения между тем и другим. И это позволяет нам при разработке компьютерной программы большую часть времени не думать об особенностях двоичного или десятичного представления, и даже об ограничениях диапазона машинной арифметики. Таково было внезапное прозрение, пришедшее ко мне где-то через двадцать лет, когда я был профессором в Белфасте. Я обобщил его с обыкновенных чисел на другие абстракции и превратил его в содержание часто цитируемой работы о проверке корректности представления данных.Эта тема повторялась во всех моих дальнейших исследованиях.

Теперь я вернусь ко второму году моего обучения в Кентерберийской Королевской школе. Я достаточно быстро продвигался в латыни и греческом, так что уже через год я способен был сносно сдать публичный экзамен, давший мне доступ в старший класс школы. Но прежде чем достигнуть уровня, необходимого для поступления в университет, я должен был снова сдавать тот же экзамен в следующие два года. К счастью, это оставляло мне время для других занятий, главным образом, интеллектуальных, поскольку мои навыки общения, к сожалению, были весьма ущербными, а мои способности к физическим упражнениям и спортивным играм — совсем слабыми. Какое-то время я занимался фехтованием; один год усердно играл в регби — игру, в которой дерзость и превосходство в весе могли произвести сильное впечатление на человека меньшего роста, хотя и более подвижного.Когда мне удавалось освободиться от организованных игр, я проводил дневные часы в школьной библиотеке.

Я обратился к шкафам, содержавшим труды по философии, и прочел много диалогов Платона в переводе Бенджамина Джоуэтта (Benjamin Jowett). Я наткнулся также на объемистый том «Истории Западной философии» Бертрана Рассела, из которой я прочел большие разделы, начиная с древнегреческой философии и почти до нашего времени. Во время школьных каникул я продолжал свое чтение, главным образом — в ночное время, в течение примерно двух часов каждую ночь, когда все остальные в доме спали. Часто я пропускал семейные завтраки по утрам. Иногда я читал романы, от классиков Чарльза Диккенса и Джейн Остин до более современных романов Олдоса Хаксли и П.Г. Вудхауза (P.G. Wodehouse). Я читал книги попсихоанализу, главным образом адлеровской школы социальной психологии, в надежде понять и улучшить мои эмоциональные состояния и мои неуклюжие способы общения. Я читал книги по математике, по философии. Эти последние интересы слились вместе в моем изучении «Введения в математическую философию» Рассела. Подобно Платону и многим дальнейшим философам,

Рассел интересовался природой математической истины. Но со времени Платона открытия в математике неизмеримо выросли. Например, Пифагор и его последователи знали уже целые числа и дроби.

Вероятно, они подозревали о существовании других чисел, вроде корня из двух, но держали эти подозрения в строгой тайне. Но теперь у нас есть отрицательные числа, действительные числа, мнимые числа, матрицы, кватернионы, и кто знает, что еще будет. Каждый вид чисел был открыт — или, может быть, изобретен?— для решения уравнений, которые можно было записать в более знакомой числовой системе, но они не имели в этой системе решения. Например, дроби нужны для решения уравнений вроде $2\mathbf{x} = 1$, означающего, что **x** должен быть дробью с числителем 1 и знаменателем 2. Каждое расширение понятий математики придавало ей большую выразительную силу для описания и решения более трудных задач.

Разумеется, в каждой числовой системе и в каждом разделе математики есть свой набор теорем, доказываемых на основе своего набора аксиом. Кажется, что концепция математической истины неизбежно делится на части. В лучшем случае она оказывается лишь относительной, то есть связанной с аксиомами, выбранными некоторым математиком, заинтересованным в некотором разделе математики или в некотором частном случае. Рассел и его современники поставили себе целью найти общие основания математики, с достаточно мощным концептуальным базисом, позволяющим определить на его основе все другие математические понятия. Необходима была также единая система аксиом, исходя из которой было бы возможно доказать все аксиомы каждой области математики.

Основным понятием, избранным — или открытым?— исследователями того времени, было понятие **множества**, которое определяется как произвольный неупорядоченный набор его отдельных элементов. Каждый элемент может иметь при этом (но не обязательно), некоторое свойство, отличающее его от всех объектов, не входящих в это множество. Например, все простые числа образуют множество чисел, отличных от чисел, разлагающихся на множители. Понятие множества играет роль, подобную платоновской идее, поскольку оно обобщает свойства своих отдельных элементов. Однако множество более конкретно, чем идея, поскольку множество — это произвольное собрание, определяемое лишь его элементами, а не общими свойствами, которые были доступны интуиции Платона.

Очевидно, наша первая задача — определить само понятие целого числа. Это должно быть некоторое множество особого рода. В самом деле, поскольку само число не является индивидуальным объектом, а является свойством некоторого множества, то это должно быть множество, элементы которого сами являются множествами. Например, число двенадцать имеет своими элементами все множества, содержащие в точности двенадцать элементов. Одним из элементов этого числа является множество апостолов Христа — множество, в свою очередь имеющее одним из своих элементов святого Петра. Все английские суды присяжных являются элементами числа двенадцать, поскольку каждый из них имеет двенадцать элементов. Понятие числа определяется как множество всех чисел, т. е. множество множеств множеств. Это повторное применение конструкции множества аналогично применению конструкции функционального пространства, в современных языках функционального программирования высшего порядка.

Конечно, требуется время, чтобы к нему привыкнуть. Определив понятие целого числа как множества, можно определить так же все другие виды чисел. Например, дробь есть пара целых чисел (числитель и знаменатель), не имеющих общего множителя; и эта пара также представляется специальным видом множества. Действительное число определяется как множество дробей, меньших его, а мнимое число есть пара действительных чисел. Эти сложные и мощные конструкции аналогичны более простым, которые используются в компьютерах для представления различных видов чисел.

Через много лет, будучи профессором Оксфордского университета, я смог извлечь реальные преимущества этой избыточной мощи теории множеств для точного определения абстрактных понятий. Моя цель состояла в том, чтобы помочь программистам писать корректные программы. Чтобы определить, что называется корректностью, нам необходимо независимое описание или спецификация того, что программа должна делать, а в некоторых критических случаях еще и того, чего она определенно не должна делать. На каком же языке мы должны писать эту спецификацию?

Первое требование к спецификации состоит в том, что она должна быть на несколько порядков проще и очевиднее той программы, в которую воплотится. Таким образом, нам нужен язык с наибольшей возможной выразительной силой, но при этом вполне точный. Он должен также допускать точно контролируемую степень определенности в отношении проектирования деталей, подлежащих установлению в дальнейшем. Какой же язык лучше подходит для этой цели, чем язык, развитый в течение тысячелетий математиками всех стран,— язык самой математики? И поскольку Рассел свел весь этот язык к основным понятиям теории множеств, с этого очевидно надо было начать. Прямое подтверждение этого составлял тот факт, что многие из основных понятий программирования непосредственно поддавались определению как множества — это были пары, последовательности, функции и отношения. Таковы причины, по которым мой французский коллега Жан-Ремон Абриаль (Jean-Raymond Abrial) избрал в качестве языка спецификации теорию множеств; он назвал этот язык «Z», вероятно, в честь специалиста по теории множеств Цермело (Zermelo).

Применимость теории множеств к программированию была проверена в исследовательской лаборатории IBM в Херсли (Hursley Development Laboratory). Она использовалась там для формальной спецификации некоторых внутренних интерфейсов новой версии одного из их важнейших программных продуктов — Пользовательской Системы Информации и Управления (Customer Information and Control System, CICS). Это привело к заметному улучшению надежности поставляемого продукта. Я уверен, что Цермело, Рассел и другие люди, занимавшиеся основаниями математики, были бы удивлены практическим использованием, которое их работы находят в современной промышленности.

Оксфорд: философия.

В 1952 году я снова держал экзамены и получил стипендию для оплаты моего обучения в Оксфордском университете, а также — квартиры и практических занятий в Мертон-колледже. Причиной этого выбора была семейная традиция. Там учился мой отец, причем учился тем же предметам. Колледж был основан в 1264 году Уолтером де Мертоном (Walter de Merton) значительно позже Королевской школы, но все же это один из старейших колледжей в Оксфорде и в Англии. В первые два года моих университетских занятий я продолжал изучать латынь и греческий, прибавив к моим достижениям сочинение латинских и греческих стихов. Быстрее продвигалось и мое чтение, поскольку я должен был полностью усвоить сочинения Гомера, Геродота и Вергилия, а также избранные сочинения Цицерона, Ювенала, Горация и Еврипида. Свободное время я продолжал посвящать моим философским интересам, в частности, логике и основаниям математики.

В моем первом семестре я включился, с несколькими друзьями, в маленькую группу с классическими и математическими интересами, собиравшуюся за кофе поздно вечером, непосредственно перед сном. Все мы купили себе по экземпляру «Математической логики» Уилларда Куайна (Willard Van Orman Quine), лауреата премии Киото по философии 1996 года. Я все еще храню мой экземпляр. Это было второе издание книги и, как утверждалось на суперобложке, в нем была устранена некоторая логическая неточность, вместе с другими недостатками первого издания. Эту неточность обнаружил Беркли Россер (Berkley Rosser) лишь после опубликования первого издания. Конечно, логическая неточность — это гораздо больше, чем простой логический недостаток; она делает несостоятельной всю логику, и обесценивает все использующие ее доказательства. Но вся цель логики как раз и состоит в том, чтобы обеспечить правильность доказательств. Именно обнаружение ошибок вроде этой в интуитивных математических рассуждениях о множествах заставило математиков девятнадцатого века усомниться в надежности интуиции, и заставило философов усомниться в платоновской доктрине прямого интуитивноговосприятия математического знания. Они обратились после этого к изучению и формализации практики математических доказательств как единственной надежной гарантии математической истины. Теперь главным результатом профессиональной деятельности математика является не просто теорема, но ее доказательство.

В современной логике доказательство рассматривается всего лишь как текст, как последовательность строчек, наподобие записи шахматных ходов. Правильность доказательства может быть проверена в точности так же, как правильность игры или грамматическая правильность латинского предложения — исследованием каждой строчки, хода или слова в

отдельности. Это лучше всего делать без всякого понимания содержания доказательства, стратегии игры или смысла предложения. В случае доказательства каждая строчка должна быть либо копией одной из аксиом, либо частным случаем предыдущей строчки, либо выводом из двух предыдущих строчек по правилу под названием **modus ponens**. Эта формальная точка зрения на структуру логического доказательства восходит к Аристотелю; она была широко применена Евклидом к геометрии и к теории чисел. Теперь она лежит в основе всех попыток применения компьютеров в математике, или к построению и проверке математических доказательств. Конечно, математики весьма свободны в выборе своих определений и теорем, которые они хотят доказать. Подобно шахматистам, они могут оценить тонкость, экономность и элегантность необычного доказательства. Они вполне свободны также в выборе основных аксиом, то есть строчек, которые могут появиться в любом месте доказательства без дальнейшего обоснования.

Именно аксиомы определяют, какой областью математики они занимаются — так сказать, в какую игру они хотят в данном случае играть. Они могут предпочесть игру с простыми правилами и легко запоминающимися аксиомами, или какую- нибудь широко известную и распространенную игру вроде шахмат или футбола. Однако невозможно даже обсуждать вопрос об истинности аксиом, поскольку они никогда не могут быть обоснованы в пределах самой математики. Для этого потребовалось бы другое доказательство, которое должно было бы содержать по меньшей мере одну строку, принимаемую за аксиому. Но как обосновать эту строку? Эта аргументация, свидетельствующая о невозможности, обычна в философии. Она называется бесконечной регрессией, поскольку вопрос не может повторяться без конца.

Простейший способ избежать этого состоит в том, чтобы остановиться до первого шага. Надо просто принять, что истинность аксиом не может быть логически доказана. Роль ученого состоит в том, чтобы подвергнуть математически выраженную гипотезу строгому ряду экспериментальных проверок. Если она выдерживает все попытки опровержения, то математические доказательства, принимающие эту гипотезу за аксиому, приведут к теоремам, применимым для предсказания и управления некоторыми классами явлений реального мира.

Через двадцать пять лет я вернулся к этому взгляду на доказательство в первой моей опубликованной статье, посвященной аксиоматическому базису компьютерного программирования. К тому времени разработка языков программирования стала весьма популярным видом научной работы. К сожалению, каждый из соперничающих языков содержал слишком много произвольных решений относительно деталей обозначений, и, казалось, не было научного способа оценить или даже аккуратно изложить эти решения и, тем более, не было надежды достигнуть соглашения между соперничавшими учеными, проектировавшими эти языки. При выборе языка программирования для практического использования разумнее всего было бы взять самый старый, предпочтительно — поддерживаемый самой крупной компьютерной компанией, поскольку такой язык вероятно дольше всего сохранился бы в будущем.

Это был хороший путь, но весьма разочаровывавший исследователей, разрабатывавших новые языки. Языки программирования казались чем-то вроде игр, зависевших от моды, соглашения и вкуса. В таком случае лучше всего было определять игру, задавая правила ее ходов. Следуя идее РобертаФлойда (Robert Floyd), я предложил правила, подобные правилам математики, в виде аксиом и шагов вывода. Их следовало использовать в доказательствах точно так же, как в математике, и с той же целью: уменьшить частоту ошибок программирования или, лучше сказать, устранить риск ошибки еще до того, как программа выполняется. Конечно, я не претендовал на истинность аксиом. Их справедливость для некоторой частной реализации языка была предоставлена ответственности инженера, проектирующего эту реализацию. Вследствие такого разделения ответственности между разработчиком языка, проектировщиком и программистом, теоремы, относящиеся к каждой конкретной программе, должны быть применимы при каждом прогоне программы, обеспечивая правильность результатов.

Я предположил, что желательной целью в проектировании языка программирования должна быть простота аксиом и легкость применения правил. Может быть, это дало нам, наконец, нечто вроде объективного и научного критерия качества разнообразных решений, принимаемых в проектируемом языке. Это предположение вскоре было проверено при испытании проекта языка **паскаль** Никлауса Вирта, а затем — при проектировании языка **эвклид** Батлера Лэмпсона и его группы в Научно-исследовательском центре вычислительной техники в Пало-Альто. Некоторые из этих идей в дальнейшем получили применение в более модных языках вроде Visual Basic, Java,

и C++, главным образом — для обнаружения и устранения ошибок в проверяемой программе, а не для доказательства их отсутствия в программе.

Я возвращаюсь к Оксфорду 1952 года. Группа, встречавшаяся поздно вечером для изучения математической логики, так и не достигла конца книги Куайна. Мы продолжали встречаться регулярно (хотя и не слишком часто) после окончания вечерней работы или выпивки; но мы предпочитали играть в карты, особенно в покер, игру, зависящую от случая, искусства и блефа. Я придерживался научной стратегии, используя мои знания теории вероятностей, так что я никогда не проигрывал слишком много. Но главное — мы получали удовольствие от самой игры. В колледже и в университете у меня было несколько более респектабельных занятий в свободное время. Я играл в шахматы за мой колледж во второй команде, а также сыграл роли в двух пьесах, поставленных Драматическим обществом колледжа,— в «Скупом» Мольера и в «Птицах» Аристофана. Это была превосходная тренировка для будущего университетского преподавателя; мне приходилось приспосабливать мой голос, чтобы он был слышен в конце средневекового зала с высокой деревянной крышей, или при шелесте ветра в деревьях на открытом месте. Я научился обращать себе на пользу нервные спазмы, возникающие перед выходом на сцену. (Боюсь, что в наши дни их останавливают лекарством, регулирующим сердцебиение).

Я продолжил свою театральную деятельность, вступив в Экспериментальный Театральный Клуб университета, где я перешел от исполнения ролей к постановке пьес. Мне доставляли удовольствие выбор и подготовка актеров, а при случае — организация публичных постановок. В мои постановки входила музыкальная версия моего собственного перевода комедии «Пленники» латинского драматурга Плавта и чтение диалога Платона «Пир», с Невиллом Когиллом (Nevill Coghill) в роли Сократа. Я научился распределять ответственность, когда это было нужно, и не делать этого в противном случае; я научился выслушивать соображения других и, в случае необходимости, реагировать на них. Опять-таки, это была превосходная подготовка к решению административных задач, обременяющих руководителей университета.

Мой курс обучения в Оксфорде назывался Literae Humaniories и следовал старинной традиции. Через два года мы приступили к двум новым предметам, древней истории и философии, в которых мы изучали по оригинальным работам Фукидида, Тацита, Платона и Аристотеля. Мы изучали также европейских философов Нового времени — Декарта, Юма и Канта. Мы познакомились с Карлом Поппером (лауреатом премии Киото 1992 года), автором доктрины, согласно которой смысл научной гипотезы состоит в экспериментах, тщательно задуманных для ее опровержения. Именно отсутствие опровержения оправдывает веру в научные теории и действия, основанные на этой вере.

Но увлекательнее всего была для меня современная философия, которую мы изучали в оксфордской школе лингвистической философии, возглавляемой Остином (Austin), Хейром (Hare) и Райлом (Ryle). Именно в этом курсе философии я впервые столкнулся с работой компьютера в виде знаменитой **машины Тьюринга**, изобретенной в 1937 году Аланом Тьюрингом в качестве «бумажного проекта», без перспективы сооружения работающей модели. Тьюринг был знаком с работами логиков по определению поддающихся проверке правил, обеспечивающих справедливость доказательств, и хотел дать точное определение того, что мы называем чисто механической процедурой над символами — определение, которое может быть в точности выполнено физическим устройством, не понимающим ни смысла символов, ни цели их употребления.

Теперь это устройство мы попросту называем компьютером. Однако в то время, для того чтобы уточнить свою идею, Тьюрингу в самом деле пришлось изобрести компьютер. Существенными особенностями вычислительного устройства являются следующие: во-первых, программа, записанная в виде конечной последовательности символов, то есть в виде текста, который может храниться и подвергаться обработке в памяти компьютера; во-вторых, легко можно написать программу, содержащую замкнутый цикл, что заставит компьютер вечно продолжать вычисления, никогда не приходя к ответу. Главная цель машины Тьюринга состояла в доказательстве, что мощность таких механизмов оказывается существенно ограниченной.

Всегда найдется множество понятий и функций, определимых математически, и даже арифметически, которые никогда не могут быть вычислены никакой машиной вообще. Первой (и притом наиболее полезной) функцией, которая оказывается невычислимой, является функция, принимающая в качестве аргумента некоторый текст и спрашивающая, окончит ли когда-нибудь

работу машина Тьюринга, использующая этот текст в качестве программы. Рассуждения Тьюринга показывают, что такую программу написать невозможно.

Доказательство Тьюринга показывает также, что мощность любого языка программирования крайне ограничена. Например, никакой язык программирования не может надежным образом решить, равны ли две функции, записанные в этом языке. Было бы совершенно непрактично пытаться проводить математические рассуждения без понятия равенства. Поэтому при составлении компьютерных программ вообще и при рассуждениях о правильности программ надо пользоваться языком, гораздо более мощным, чем может быть любой осуществимый язык программирования. Я применил этот аргумент много лет спустя, рекомендуя язык спецификации Абриаля **Z**. В то время был широко распространен взгляд, что языки спецификации должны быть осуществимы на компьютере, но язык **Z**, очевидно, не таков. И это вовсе не критическое замечание, потому что в этом состоит его главное достоинство. В рассуждениях о спецификациях и программах существенно используется вся мощь математики, в том числе простые логические понятия: конъюнкции «и», отрицания «не» и равенства «=».

Результат Тьюринга показывает, что эти важнейшие понятия никогда не могут быть включены в выполнимый язык программирования. Результат Тьюринга объясняет также одну из причин, по которой программы должны быть столь сложными. Каждая программа является по существу конструктивным доказательством вычислимости некоторой функции. Еще более очевидная причина — это требование эффективности. Единственное решение состоит в доказательстве, что сложная программа удовлетворяет гораздо более простой спецификации. Спецификация же должна быть, в самом деле, столь простой, чтобы очевидным образом удовлетворять потребностям пользователя. Нет надежды когда-либо доказать правильность данной спецификации. Для этого понадобилась бы более простая спецификация, которой она удовлетворяет. Это опять приводит к бесконечной регрессии. Решение состоит, конечно, в том, чтобы с самого начала использовать простейшую возможную спецификацию.

Лондон: русский язык.

В 1956 году, сдав экзамен по древней истории и философии, я подал заявку на стипендию для подготовки докторской диссертации по философии. Пожалуй, мне повезло, что я ее не получил. Иначе я мог бы до сих пор остаться второсортным философом, не имея оправдания в том, что я, в то же время, специалист по вычислительным наукам. Так или иначе, у меня нет настоящей докторской степени, а только почетные докторские звания, что доставило мне гораздо больше чести. В то время молодые люди в Англии должны были проходить двухлетнюю принудительную военную службу (отмененную вскоре после этого). Ввиду моей квалификации в латыни и греческом я легко получил направление на курсы изучения русского языка Королевского Военного Флота (мне помогло в этом то обстоятельство, что мой дядя был капитаном флота). Мы прошли стандартную военную подготовку, нас научили стрелять из винтовок, и мы совершили в течение нескольких дней поездки на эсминце и минном тральщике, с тем единственным результатом, что почувствовали себя совсем больными; но мы не провели на море ни одной ночи, и никогда не приближались к чему- нибудь вроде военных действий.

Главное обучение русскому языку производилось на суше, в филиале Школы Славянских исследований Лондонского университета. Мы получили весьма основательные знания в формальной русской грамматике, которая была не менее сложной, чем латинская или греческая, со склонением существительных и прилагательных по шести падежам, хотя, к счастью, лишь с двумя родами. Нас предупредили, что грамматические ошибки быстро приведут нас к исключению из школы, потере офицерского звания и возвращению к трудностям и учениям обычной военной службы. Поэтому я применил то же усердие, что и в моих латинских и греческих сочинениях, проверяя каждый род и каждый падеж по всем правилам; но я по-прежнему делал немало ошибок, как и в моих классических латинских и греческих переводах.

По сравнению с классическими языками, главная разница была в том, что русский язык мы изучали как разговорный язык, а в разговоре нет времени применять какие-либо грамматические правила. Поэтому я весьма удивился, когда через некоторое время стал правильно говорить по-русски, по большей части, не думая о грамматике, точно так же, как это делают маленькие дети при обучении языкам. Вскоре я вполне освоился даже с невероятно сложными правилами для русских числительных. Не знаю, как объяснить это явление. По-видимому, в естественных языках,

как и в математической философии, есть разница между формализованным подходом Аристотеля, весьма пригодным для компьютеров, и интуитивным подходом Платона, более близким к человеческому пониманию и опыту.

Даже во время самого интенсивного обучения русскому языку я уделял внимание моим интересам к философии и логике. Имея в 140 виду применения к нечетким знаниям, я занимался логическими системами с более чем двумя значениями истинности, где кроме значений «истинно» и «ложно» могут быть также такие значения как «вероятно», «возможно» и «маловероятно». Я определил единственный логический оператор, позволяющий построить из него все другие операторы любой многозначной логики. Это оживило мои надежды продолжить эту работу в аспирантуре, и я даже прошел интервью с Карлом Поппером из Лондонского университета. Я объяснил ему мои идеи весьма неудачно, и завершил это объяснение извинением, что у меня слишком много идей. На это Поппер сказал: «Да, и задача исследования состоит в том, чтобы отличать хорошие идеи от плохих». Это был полезный совет, и я больше не добивался приема в аспирантуру. Вместо этого я начал уже думать о работе с компьютерами в промышленности. Мне попалось на глаза объявление, предлагавшее работу в английской сети ресторанов под названием «Чайные дома Лайонса». У этой компании было отделение, проектировавшее, строившее и программировавшее свои собственные компьютеры, вначале — лишь для управления зарплатой и снабжением. Впоследствии эта компания стала продавать компьютеры для тех же целей другим компаниям. Я пошел на собеседование к ним вместес коллегой по русским курсам, чтобы получить дальнейшую информацию, но без немедленной перспективы приступить к работе, поскольку наши русские курсы должны были продолжаться еще год. Но здесь у меня возникла идея, какую карьеру я должен в конце концов избрать.

Оксфорд: статистика.

Окончив мой курс русского языка, я решил работать в промышленности. Но мне не хотелось заниматься бухгалтерией или стать администратором, как это случалось обычно с выпускниками гуманитарных факультетов. Я хотел получить научную или техническую работу. Чтобы получить и сохранить такую работу, я считал необходимой соответствующую техническую квалификацию. Оказалось, что я могу ее получить посредством годичной аспирантуры (снова в Оксфорде). Выбранный мною курс должен был доставить мне диплом по статистике, что позволило бы заниматься (на этот раз лишь в качестве хобби) моими философскими интересами в области теории вероятностей, рассматриваемой как основа нечетких знаний.

Я предполагал, что буду платить за обучение и покрывать прожиточные расходы за счет скромных сбережений, накопленных для меня родителями. Но после начала занятий я обнаружил, что английское Министерство образования предлагает стипендию всем студентам, изучающим такие практически важные курсы. Поэтому я смог сохранить свои сбережения, пока они не понадобились мне для покупки моего первого дома. Я должен извиниться перед нынешними студентами, которым приходится, одолжив деньги, платить за обучение гораздо больше, чем платил я.

Я встретился с некоторыми трудностями, убеждая статистиков в Оксфорде, что я достаточно компетентен в математике для того, чтобы слушать их курс. Они рекомендовали мне использовать некоторое время до начала семестра, чтобы выучить кое-что из дифференциального и интегрального исчисления, а также приобрести навыки в быстрых арифметических вычислениях — сложении, умножении и извлечении квадратных корней. Я изучил также матричную алгебру, которую использовал впоследствии, чтобы уяснить себе понятие корреляции в совместном распределении вероятностей. Я был весьма благодарен за доверие, оказанное мне при поступлении на этот курс. Позднее, когда мне пришлось принимать студентов в аспирантуру по вычислительным наукам, я всегда проявлял большую симпатию ктем, кто получил гуманитарную подготовку, особенно в области языков.

В Оксфорде есть традиция, позволяющая студентам, изучающим любой предмет, посещать лекции по любому другому предмету, и я не преминул воспользоваться этим, выбрав из имевшихся аспирантских курсов мой излюбленный предмет — философию. Самый интересный курс по философии читал Хао Ван (Hao Wang), который только что составил программу для одного из первых компьютеров IBM 704, проверившую все доказательства в первых девяти

главах фундаментального труда Рассела и Уайтхеда «Principia Mathematica». Это было замечательное достижение в области программирования, которое осуществило на практике идеи Тьюринга, цели первых логиков и мечты немецкого философа семнадцатого века Лейбница.

Хао Ван упомянул также о возможности логики с ограничением разрядности чисел в заданной системе счисления. Эту идею я использовал в моей первой публикации о проверке программ в качестве защитной меры от ошибок переполнения, связанных с конечным представлением чисел в компьютере. На аспирантских лекциях Хао Вана в Оксфорде я познакомился с японской студенткой Хиде Исигуро (Hide Ishiguro), сделавшей впоследствии выдающуюся карьеру в качестве философа в Соединенных Штатах и Японии. Я отмечаю с удовольствием, что она присутствует здесь сейчас.

В вечернее время я участвовал в работе аспирантского философского общества имени Джоуэтта (Jowett). Здесь я представил мою первую исследовательскую работу по философии статистики. Меня занимал вопрос о том, что утверждение, приписывающее вероятность некоторому событию, никогда не может быть опровергнуто наблюдением, и тем самым не может иметь научной значимости в смысле Поппера. Как бы часто ни происходило событие, упрямый теоретик мог бы без конца утверждать, что оно имеет очень низкую вероятность. Решение парадокса состоит в том, что сторонник вероятностной гипотезы должен иметь добавочные обязательства. Прежде всего, он должен быть согласен заключать пари или идти на риск (в реальных или условных деньгах), зависящий от вероятности, которую гипотеза приписывает данному событию. Таким образом, если гипотеза неверна, то он, по крайней мере, теряет все увеличивающуюся сумму. Во-вторых, он должен заранее установить предельную сумму, которую он готов потерять (если не в реальных деньгах, то в условных), после чего он обещает отказаться от своей гипотезы. Именно это обещание восстанавливает, в несколько измененной форме, критерий Поппера для опровержения научной значимости статистической гипотезы.

Как обнаружилось, мои навыки в быстрых арифметических вычислениях оказались ненужными, потому что отдел биометрии, где мы изучали статистику, установил множество настольных вычислительных машин. Я научился хорошо пользоваться ими, но гораздо больше заинтересовал меня курс программирования на новом компьютере компании Ferranti. Этот компьютер, названный по имени греческого бога Меркурия, университет купил и установил незадолго до того. Преподавателем был специалист по численному анализу Лесли Фокс (Leslie Fox), который оказался у меня заведующим кафедрой двадцатью годами позже, когда я вернулся в Оксфорд в качестве профессора. Он научил нас единственно доступному в то время, достаточно примитивному языку программирования высокого уровня Mercury Autocod.

Я закодировал в качестве моей первой программы приближенный метод решения задачи о вероятностях в игре двух участников с нулевой суммой, наподобие «Ножа для разрезания бумаги». (Результат чтения «Теории игр» фон Неймана и Моргенштерна). Было очень интересно видеть, как программа вводится и компилируется. Через несколько секунд работы она выдала шесть требуемых чисел. Но я никак не мог определить, правильны ли эти числа, потому что забыл включить проверочную программу, которая легко могла бы мне это сообщить. Теперь такие проверки называются *assertions*. Впоследствии я стал рекомендовать их как основу проверки корректности программ. Мне хотелось бы провести аналогию между двумя научными дисциплинами — статистикой и информатикой. Обе они представляют собой развившиеся в двадцатом веке отрасли математики, еще не вполне принятыеболее традиционными математиками в основное русло этой науки. Такие ученые называют эти новые дисциплины «применяемой математикой» (Applicable Mathematics), что на одну ступень ниже по статусу, чем «прикладная математика» (Applied Mathematics). Но эти дисциплины, безусловно, могут носить такое название с гордостью. Подобно информатике, статистика обязана своим статусом ценным применениям к другим наукам. Она все больше применяется к гуманитарным наукам — сначала в сельскохозяйственной науке и генетике, а теперь также в медицине, физике элементарных частиц, лингвистике и литературоведении, географии и т. д. Как и в информатике, статистические исследования охватывают широкий диапазон — от проектирования и анализа конкретных практических экспериментов до использования в своих основаниях глубоких математических теорий — пространств с мерой и топологии в статистике, доменов и категорий в информатике.

Иногда теория, встречаясь с практикой, приводит к приятным и полезным результатам; но чаще между ними возникает некоторое напряжение. Надо скорее приветствовать, чем сожалеть об этом, поскольку это характерно для всех научных дисциплин, сохраняющих свою

жизнеспособность. И в статистике, и в информатике наш худший враг — невежество. Слишком много людей, которые могли бы извлечь пользу из известной методики, но никогда о ней не слышали, или просто не верят тому, что слышали. Каждый специалист по статистике может рассказать вам одну и ту же историю: его коллега- ученый предъявляет массу данных, собранных им в своей области, и просит помочь проанализировать их. Первый вопрос статистика — «Какую гипотезу вы пытаетесь проверить?» Слишком часто на это отвечают еще более фундаментальным вопросом: «А что такое гипотеза?» Бедному статистику остается только начать лекцию об основных принципах статистических выводов. Важным предварительным условием является простая, ясная и строго формулированная гипотеза, с не слишком большим числом свободных параметров, подлежащих оценке. Ее должен сопровождать строгий проект эксперимента для проверки именно этой гипотезы, вместе с тщательной рандомизацией выборок, чтобы избежать всех известных видов пристрастного отбора. Лишь в этом случае мощные математические методы статистики могут быть привлечены для получения правильного вывода. Любые данные, собранные до формулировки гипотезы, в действительности, следовало бы отбросить. Однако в случае крайней необходимости, либо из любопытства или любезности, статистик делает всё возможное, чтобы подогнать некоторую правдоподобную гипотезу и, таким образом, извлечь хоть какую-нибудь пользу из уже собранных данных.

Специалист по информатике может рассказать вам очень похожую историю. Компьютерные программы все более используются в критических ситуациях: для ядерных электростанций, для легковых и грузовых автомобилей, для кредитных карточек. Перед поставкой такой программы инженер или банкир обращаются к специалисту по информатике, требуя от него заверения в корректности программы. Прежде всего, мы, конечно, задаем заказчику вопрос: «Какую спецификацию вы пытаетесь получить?» И слишком часто мы слышим в ответ: «А что такое спецификация?» Здесь опять-таки приходится начать лекцию об основных принципах программирования. Важным предварительным условием надежного программирования является простая, но строго сформулированная абстрактная спецификация требуемого поведения программы, ясно устанавливающая, что дозволяется делать с помощью этой программы и чего нужно избегать.

Это должно сопровождаться тщательной разработкой структуры программы, с указанием интерфейсов и тщательным анализом, выполненным с помощью математических вычислений и доказательств. Лишь после этого компоненты программы подлежат кодированию, при разумной математической гарантии, что после сборки окончательного продукта этот продукт будет, в самом деле, работать сразу же после поставки и будет работать всегда. Любой код, написанный до формулировки спецификации, должен быть выброшен. В случае крайней необходимости и любопытства или из любезности специалист по информатике часто сделает все возможное, чтобы подогнать спецификацию к уже написанному коду: иногда произведенный таким образом анализ обнаруживает ошибки, которые вряд ли удалось бы обнаружить другим способом.

Москва: лингвистика.

Во время последнего семестра моего курса статистики в Оксфорде я увидел на доске объявлений моего колледжа рекламу Британского Совета, официального органа, способствующего связям Англии с другими странами мира в области культуры и образования. Эта организация только что заключила соглашение с Советским Союзом о ежегодном обмене двадцатью аспирантами. Время подачи заявлений истекало на следующий день. Я тут же решил испытать этот шанс. Если мне удастся соорудить заявку в течение двадцати четырех часов, я ее подам, и если мне повезет, воспользуюсь этим предложением.

И мне повезло! Это было интересное приключение. В 1960 году Советский Союз был по-прежнему недоступен для иностранцев, и я вошел в первую группу иностранных учащихся, которые могли посетить эту страну и познакомиться с драматически противостоящей нам политической системой. Русская культура развивалась вполне самостоятельно, в изоляции от греческих и латинских влияний, воздействовавших на Европу в эпоху Возрождения. Я хотел также использовать этот случай для упражнения и совершенствования в русском языке. Наконец, я хотел глубже изучить теоретические основы теории вероятностей, пионером которых был, как известно, великий русский математик Андрей Николаевич Колмогоров. Именно на его кафедру я

был зачислен в Московском государственном университете. Вначале было трудно. Чтобы слушать лекцию на чужом языке в течение целого часа, требуется немалая сосредоточенность, а в Москве лекции продолжались целых два часа. Я затратил весь день, пытаясь купить в Москве подушку для сидения, но безуспешно; ее прислали мне по почте родители. Разрешив обе эти проблемы, я приступил к настоящей проблеме — труднее всего была сама математика, и особенно трудно было для новичка изучить абстрактную теорию меры на аспирантском уровне.

К счастью, я нашел в университетской библиотеке превосходную книгу Халмоша (Halmos), которую я изучил и усвоил с большим удовольствием. В ней вводился ряд технических понятий из теории множеств, теории решеток и теории приближений, с которыми я встретился впоследствии в моей работе над теорией доменов, применяемой к изучению компьютерных языков программирования.

Будучи в Москве, я получил письмо, определившее все будущее направление всей моей жизни. Письмо пришло из Национальной физической лаборатории, главной исследовательской лаборатории английского правительства, расположенной в Тэддингтоне близ Лондона. Основная задача этой лаборатории состояла в поддержании и совершенствовании стандартов научных измерений, например, для определения времени и расстояния. В этом письме меня приглашали посетить Тэддингтон, чтобы обсудить мое назначение на весьма престижный пост старшего научного сотрудника, для разработки нового проекта компьютерной программы автоматического перевода с русского языка на английский. Это послужило достаточной причиной, чтобы уменьшить внимание к моим математическим занятиям и ознакомиться с текущим состоянием русских работ по машинному переводу — разумеется, с английского на русский. Я прочел первые номера русского журнала «Машинный перевод», встретился с некоторыми авторами и поговорил с ними. Но я никогда не видел ни одного советского компьютера — в то время они были слишком секретными, чтобы их можно было показывать иностранцам.

Задача машинного перевода начинается с отделения корня каждого слова от его грамматических приставок, указывающих число, род и падеж. Затем корень используется для сверки с основным словарем языка. Далее следует грамматический анализ предложения на языке подлинника. После обнаружения и устранения двусмысленностей должно быть синтезировано переведенное предложение с применением грамматики и порядка слов языка перевода. Я немало размышлял обо всех этих трех этапах. Синтез предложений на языке перевода был предметом первой научной статьи, когда-либо написанной мною. Она была написана по-русски, отпечатана на русской машинке, одолженной у одного из друзей, и опубликована в журнале «Машинный перевод» Мои занятия синтаксическим анализом привели меня в московскую библиотеку имени Ленина, единственную, где имелись относящиеся сюда работы Ноама Хомского, награжденного в 1988 году премией Киото за его работы по Cognitive Science. Его теории синтаксиса являются точным выражением правил, которые должны быть использованы компьютером для проверки грамматики предложения. Упрощение этих правил для искусственного языка, несомненно, является наилучшим способом уменьшить риск грамматической ошибки, допущенной человеком.

Идеи Хомского были впервые подхвачены Джоном Бэкусом и использованы Питером Науром при проектировании международного алгоритмического языка АЛГОЛ 60. С тех пор они оказали огромное влияние на разработку других языков программирования, на компьютерную лингвистику и на информатику в целом. В последнее время они получили еще более широкое и непосредственно практическое применение в маркировочном языке XML, стандартизованном в качестве средства для обмена высоко структурированными данными между компьютерами. Для меня наиболее важным результатом предложенной мне работы над машинным переводом в тэддингтонской группе было изобретение сортирующего алгоритма, который я впоследствии назвал быстрой сортировкой (quicksort).

В то время компьютеру было очень трудно справляться в словаре, чтобы найти перевод каждого слова предложения на языке подлинника. Словарь хранился в алфавитном порядке на длинной магнитной ленте, чтение которой занимало несколько минут, даже если оно сводилось к простому сканированию от начала до конца. Было бы весьма неэффективно выполнять сканирование отдельно для каждого слова предложения. Более эффективная идея состояла в том, чтобы сначала расположить слова предложения в алфавитном порядке, так, чтобы все они могли быть прочитаны в течение одного прохода ленты. При этом каждая словарная статья может просматриваться, когда она проходит под читающей головкой.

Я задался вопросом, каким образом компьютер сможет выполнить этот предварительный процесс упорядочения. Моя первая идея, вероятно, пришла бы в голову каждому: выбирать слова по одному за раз, помещая их в надлежащее место уже построенной последовательности слов. Но я знал, что для компьютера такая операция была бы весьма неэффективна: для предложения всего лишь из двадцати слов могло бы потребоваться до двухсот операций сравнения и перестановки. Дальше возникла другая идея: выбрать случайным образом одно из слов самого предложения; затем просмотреть предложение, отобрав все слова, стоящие в алфавитном порядке ниже выбранного слова, и расположить их вместе (например, слева), а все слова выше его — справа; и, наконец, поставить выбранное слово в середину. Я разработал весьма эффективный способ хранения обеих групп, используя оба конца одного и того же массива, где хранится исходное предложение. Это было важно, потому что в то время оперативная память была очень маленькой — всего несколько килобайт, и не было свободного места, чтобы хранить слова во время сортировки.

Конечно, задача сортировки этим не исчерпывается. Ее можно было бы завершить, если бы удалось сортировать левую и правую группы отдельно, тщательно их разделив. Почему бы этого не сделать точно таким же образом, как мы уже это сделали, выбирая слова этих групп? Тогда каждая из групп разобьется на две, так что всего получится четыре группы. Продолжая этот процесс столько, сколько нужно, мы придем к набору групп, каждая из которых содержит лишь одно слово. Конечно, их невозможно делить дальше, но, к счастью, это и не нужно. Поскольку все группы в течение всего процесса располагались в правильном порядке, этим завершается упорядочение всего предложения.

Я думаю, что эта идея пришла мне в голову внезапно, но требовалась еще тщательная проверка, будет ли она, в самом деле, работать и как быстро. Поэтому я принялся писать программу для выполнения этой процедуры. Я пользовался при этом единственным языком программирования, который я знал — языком Автокод Меркурий, выученным мною в Оксфорде в предыдущем году. Но это была для меня слишком сложная работа. Я не мог выполнить необходимые вспомогательные операции, чтобы разделять мои группы и помнить, что еще осталось сделать. Поэтому я отложил свою идею и вспомнил о ней снова лишь в следующем году, когда я располагал уже интеллектуальными средствами и надлежащим языком программирования, чтобы справиться со всеми трудностями. В конце моего пребывания в Москве я получил другое, совершенно неожиданное письмо, на этот раз от моего дяди, который вышел в отставку из Королевского флота и стал главным менеджером Ассоциации производителей научных приборов Великобритании. Он организовывал выставку в центре Москвы, где все производители могли бы выставлять и продавать свои изделия. Он предложил мне щедрый гонорар — двадцать фунтов стерлингов за рабочую неделю. Я должен был действовать как переводчик со стороны экспонентов и приезжающих из Англии лекторов, обеспечивая их связь с русской публикой — учеными и потенциальными покупателями, посещающими выставку. Поэтому я прервал мой приятный отпуск с экскурсией в Грузию и Армению, устроенной моими университетскими хозяевами для нашей группы в конце годового пребывания. Я вылетел обратно в Москву, чтобы помочь моему дяде устроить выставку.

Одним из экспонентов была компания Elliott Brothers, Ltd., производившая мини-компьютеры для научных вычислений в своем Отделении вычислительной техники в Борхемвуде (Borhamwood). В Москве они демонстрировали работу своего новейшего компьютера Elliott 803. Он имел около шестнадцати килобайт оперативной памяти и работал с удивительной скоростью — свыше тысячи простых операций в секунду. У него не было ни принтера, ни магнитных лент, ни арифметики с плавающей запятой, и он стоил всего лишь двадцать девять тысяч фунтов стерлингов, что соответствует, вероятно, полумиллиону нынешних фунтов. Но он был достаточно надежен, чтобы его можно было перевезти в автомобиле за тысячу миль, через всю Европу. После этого он работал в московском выставочном зале в течение всего московского лета, при московских источниках питания. Экземпляры этого компьютера еще сохранились в научных музеях Англии. Иногда они приводятся в рабочее состояние энтузиастами, которые до сих пор используют некоторые программы, разработанные и написанные мною в начале шестидесятых годов.

Компьютер привлекал мое внимание больше всех экспонатов, и я проводил большую часть времени возле него в выставочном зале. По окончании выставки управляющий Отделения вычислительной техники предложил мне бесплатное возвращение в Англию в пустом

автомобиле, доставившем в Москву компьютер: мое знание русского языка, местного населения и бюрократии должно было быть весьма полезным шоферу в этой поездке. Меня пригласили также посетить при возвращении завод фирмы Elliott Brothers, Ltd. в Борхемвуде и обсудить вопрос о постоянной работе. Таким образом, мои дела шли отлично. Мне предложили в перспективе две должности. Двадцать фунтов от моего дяди превратились в сорок, поскольку экспоненты его Ассоциации полагали, что я проделал для них хорошую работу. Кроме того, Британский Совет возместил мне за счет государства стоимость моей поездки по железной дороге и пароходом из Москвы в Лондон. Таким образом, я смог приобрести себе новую одежду, которая не нужна была мне, когда я служил на флоте и которую, конечно, не хотел покупать в Москве.

Борхемвуд: работа с компьютерами

Мое первое рабочее интервью было в Национальной физической лаборатории в Тэддингтоне. Там я увидел знаменитый прототип компьютера ACE, первоначально спроектированный при консультации Алана Тьюринга. Функциональное устройство ACE было выполнено на электронных лампах, несколько десятков его регистров прямого доступа — на акустических линиях задержки, а оперативная память — на магнитных барабанах. Стойки компьютера занимали большой зал, но архитектура была значительно более примитивной, чем у компьютера Ferranti Mercury, приобретенного Оксфордским университетом двумя годами ранее, и еще более отсталой по сравнению с транзисторами и запоминающим устройством на магнитных сердечниках компьютера Elliott 803. При дальнейшем обсуждении предложенных мне условий работы оказалось, что должность старшего научного сотрудника была указана ошибочно. В действительности, я вовсе не должен был стать научным сотрудником. Я должен был получить техническую должность в качестве экспериментатора. Когда же мои работодатели обнаружили, что у меня нет ученой степени, они сказали мне, что могут нанять меня лишь на временную работу, и что я вовсе не могу рассчитывать на постоянную должность в Гражданской научной службе. Мне рассказывали, что в Лаборатории удивились, когда я отклонил перспективу такой нисколько не соблазнительной карьеры.

Однако главная причина, по которой я отклонил это предложение, состояла в том, что я потерял веру в осуществимость машинного перевода с естественного языка. Я осознал, что недостаточно хранить в памяти словарь всех отдельных слов языка; необходимо хранить еще большее число сочетаний слов и выражений. Предполагалось, что задача упростится, если ограничиться научной литературой, но в действительности это делало ее еще труднее, поскольку каждое важное открытие в науке приводит к появлению целого ряда совершенно новых слов и выражений, для которых надо ввести соответствующие, тщательно подобранные, выражения в другом языке. Неосуществимость проекта была признана, когда лет через пять лет от него отказались.

Только сейчас машинный перевод становится возможным. Но он не опирается на изощренные синтаксические и аналитические методы, которые мы разрабатывали в шестидесятые годы. Вместо этого, как я и предсказывал, машина хранит большой массив предварительно переведенных текстов, и использует сравнение структуры (pattern) переводимого предложения с полным источником предложений языка. Как я подозреваю, этот новый метод представляет полезную аналогию с интуитивным подходом, который позволяет человеку овладеть структурой и грамматикой естественного языка, используя вместо аналитических правил одну только подсознательную память. Гораздо более успешным было мое второе рабочее интервью в фирме Elliott Brothers, Ltd.. Мне предложили стандартное стартовое жалование в восемьсот фунтов в год, с прибавлением ста фунтов за мое знание русского языка, которое могло быть полезно компании. Каждый год меня посылали летом с компьютером на московскую выставку, в автофургоне, вместе с шофером. Я должен был отвечать на вопросы множества русских посетителей и беседовать с потенциальными покупателями машины. На выставочном стенде самыми частыми посетителями были покупатели машин прежних образцов, у которых возникали проблемы с нашим оборудованием или программным обеспечением. Они работали в столь секретных институтах, что им не разрешали даже писать нам, чтобы получить простые ответы на самые обычные вопросы. Поэтому им приходилось ждать нашего следующего приезда в Москву, чтобы найти решение своих проблем.

Моя главная работа в фирме Elliott Brothers состояла в программировании — я должен был писать библиотечные программы в десятичном машинном коде для компьютера 803, который я впервые увидел в Москве. После предварительных испытаний, внушивших моему боссу уверенность в моей квалификации, мне доверили ввод в эксплуатацию остроумного нового метода сортировки слов или чисел, вводимых в компьютер в возрастающем порядке, незадолго перед этим разработанного и опубликованного Шеллом. Большое преимущество этого метода состояло в том, что сортировка производилась в той же области памяти, которую занимали подлежавшие сортировке слова, и не требовалось почти никакой дополнительной памяти. Я выполнял в точности данные мне инструкции. Мне доставляла подлинное удовольствие оптимизация внутренних циклов программы, позволявшая извлекать выгоду из самых остроумных команд машинного кода. С большим удовольствием я занимался также оформлением программ согласно стандартам, установленным для документации, которая поставлялась потребителям в составе нашей библиотеки. Интересно было даже испытание программы — обнаружение ошибок напоминало решение математических головоломок. Замечательно было, что мне за это еще и платили! К счастью для меня, это чудо продолжалось в течение большей части моей трудовой жизни.

Моя единственная проблема в это время состояла в том, что мой босс был лишь четырьмя годами старше меня, и его босс тоже был еще молодым человеком. Ясно было, что я пропустил первую большую волну распространения компьютеров, и мне предстояло долго ждать какого-либо повышения в должности. К счастью, эта моя оценка оказалась совершенно ошибочной. Мог ли я знать, что в течение моей трудовой жизни быстродействие компьютеров вырастет примерно в миллион раз? Что они станут в тысячу раз дешевле? В миллион раз меньше? И что при этом емкость их внутренней памяти вырастет в тысячу раз. В то же время, принципы программирования, с которыми я впервые встретился в промышленности сорок лет назад, и которые были предметом всего моего преподавания и исследований в университетах, остались в наши дни столь же незыблемыми, какими они были всегда.Пожалуй, они стали еще важнее, поскольку теперь они применяются на практике миллионами программистов во всем мире.

Сдав мою законченную программу сортировки Шелла, я очень скромно дал понять моему боссу, что у меня есть гораздо более быстрый метод сортировки. Он в это не поверил и предложил мне пари на шесть пенсов, что я ошибаюсь. Тогда я объяснил ему метод, открытый мною при размышлениях о машинном переводе в Московском университете. Он признал, что я выиграл шесть пенсов. Но это не было достаточной причиной, чтобы переписать заново только что завершенную программу сортировки. Меня ожидали более срочные задачи. Первая задача состояла в том, чтобы написать программу для лентопротяжного устройства, недавно присоединенного к компьютеру 803. Я вспомнил, что в основе первоначальной машины Тьюринга было хранение данных на ленте, и изучил ее снова, в надежде найти полезные понятия и абстракции, чтобы применить их в моем программировании. Однако вскоре мне пришлось расстаться с этой идеей по серьезным практическим причинам. Так я впервые пережил знаменитый конфликт между теорией и практикой вычислений.

К счастью, в свободное время я продолжал думать о моем алгоритме сортировки, чтобы выяснить, какова его скорость. Я написал систему совместных разностных уравнений, определяющих среднее число необходимых сравнений и обменов. Однажды в воскресенье я лежал на своем диване, лениво перебирая формулы. Вдруг я остановился, обнаружив, что я нашел новое доказательство того очевидного факта, что нуль равен нулю. Это нередко случается с математиками-любителями, и лучшее, что можно в таких случаях сделать, это вернуться к предыдущим выкладкам и тщательно их проверить. Проделав это, я обнаружил, к своему удивлению, что я не ошибся: я действительно нашел именно ту формулу, которая решала систему уравнений и давала среднюю скорость сортирующей программы. Она работала именно с той скоростью, на которую я рассчитывал. Это воодушевило меня настолько, что я написал отчет о моем методе в научной статье под названием “Quicksort”, которая была опубликована в British Computer Journal в 1962 году.

Десятью годами позже я посетил Стенфордский университет по приглашению Дональда Кнута, лауреата премии Киото по информатике за 1996 год. Он сказал мне, что его вдохновил мой анализ сложности алгоритма quicksort, и что он и его ученики применили впоследствии свои гораздо более мощные аналитические методы и получили значительно больше сведений о

статистическом распределении времени сортировки для этого алгоритма. Когда прошло шесть месяцев моей работы в компании Elliott Brothers, мне поручили еще более важную задачу — разработать новый язык программирования высокого уровня для новых, более быстродействующих моделей компьютеров этой компании. По особому везению в мои руки попал экземпляр отчета Питера Наура о Международном алгоритмическом языке АЛГОЛ 60, недавно разработанном комитетом специалистов. Мы решили использовать подмножество этого языка. По еще большему везению, моей помощницей оказалась программистка Джил Пим, которая перешла на этот проект после применения автокода к предыдущему компьютеру фирмы Elliott. Вскоре после этого мы поженились и до сих пор живем в счастливом браке.

Одной из многих прогрессивных особенностей языка АЛГОЛ 60 были рекурсивные процедуры. Процедурой называется повторно применяемая часть программы, которая служит некоторой полезной цели, вроде сортировки, применяемой в различных приложениях. Рекурсивная процедура — это процедура, нарушающая запрет Аристотеля применять в определениях уравнения, содержащие определяемый член с обеих сторон. В самом деле, если вы попытаетесь использовать такое определение, заменив определяемое слово экземпляром другой стороны уравнения, это приведет к бесконечной регрессии. Но при некоторых разумных условиях процедура, рекурсивно использующая самое себя, будет работать правильно. Это кажется почти чудом, поскольку при записи определения процедуры вы можете уже успешно пользоваться той самой процедурой, которую вы еще окончательно не записали.

Я относился к рекурсии с особым энтузиазмом, поскольку она позволила мне опубликовать на языке АГОЛ 60 очень ясно сформулированную программу, вместо моей прежней сложной программы сортировки. Напомню, что она начинается с разделения слов предложения на две группы. Затем надо использовать в точности ту же процедуру для сортировки этих групп. При этом группы сохраняют правильный порядок, так что после сортировки обеих групп оказывается отсортированным все предложение. Единственное требование состоит в том, чтобы ни одна из групп не оказалась пустой, и чтобы группы, состоящие в точности из одного слова, опускались. Только с помощью языка АЛГОЛ 60 я смог формулировать эту идею с полной ясностью. С этого времени я всегда считал высшей целью языка программирования возможность просто и изящно выражать на нем хорошие идеи.

В то далекое время, в 1961 году, написание компилятора для языка программирования было весьма устрашающей задачей, и рекурсии языка АГОЛ 60 делали ее еще сложнее. К счастью, у моей компании была библиотека с хорошей подборкой журналов, которую я просматривал почти каждую неделю. Я нашел там выпуск журнала Communications of the ACM за апрель 1960 года, посвященный применению языков программирования, содержащих рекурсию. На меня произвела наибольшее впечатление статья о рекурсивных функциях от символических выражений, написанная Джоном МакКарти, лауреатом премии Киото по информатике и искусственному интеллекту за 1988 год. В ней было приведено удивительно ясное описание первой версии чисто функционального языка программирования LISP. Этот язык был непосредственно подсказан теориями знаменитого логика Стивена Клини, и включал его концепцию обобщенной арифметики и его общее определение частично рекурсивных функций.

Меня восхитила очень простая LISP программа, написанная МакКарти для определения посредством интерпретации любой другой LISP программы, предъявленной ей в качестве данных. Она была намного проще сложных конструкций, придуманных Аланом Тьюрингом для выполнения той же задачи в его машине. И теперь это не было уже теоретическое упражнение, эта программа, в самом деле, работала на реальных компьютерах. Меня столь же восхитил первый пример в руководстве по программированию для версии 1.5 языка LISP. Это была замечательно простая запись того самого алгоритма, который Хао Ван применил раньше для проверки теорем первых девяти глав «Principia Mathematica» Рассела.

Через десять лет я проработал лето в Стенфорде, в Лаборатории искусственного интеллекта Джона МакКарти. После моего опыта с машинным переводом я весьма скептически смотрел на практические перспективы искусственного интеллекта вообще. Но идеи МакКарти, относившиеся к функциональному программированию, к структурам данных, к аксиомам и к недетерминизму, сыграли, по крайней мере, для меня, историческую роль стимулятора во всем дальнейшем развитии этого предмета. После внимательного изучения статей МакКарти и других

авторов я нашел ключевую идею, позволившую мне разработать и написать АЛГОЛ-компилятор для машины Elliott.

Это была та же идея, которая была использована в моей программе сортировки на языке АЛГОЛ 60 — то есть концепция рекурсии. Весь компилятор представлял собой процедуру, которая транслировала один оператор программы. Используя синтаксический метод Ноама Хомского, любую программу на языке АЛГОЛ 60 можно описать одним оператором, который содержит в себе много других, меньших операторов. При этом, когда компилятор встречается с одним из этих меньших операторов, он просто принимается рекурсивно компилировать этот внутренний оператор, прежде чем продолжать компиляцию остальной программы. Таким образом, магия рекурсии, как и все другие хорошие идеи в программировании, применима к большим программам, так же как и к малым.

Наш проект реализации языка АЛГОЛ 60 успешно развивался. Примерно через год я начал надеяться, что его вскоре он будет готов для поставки, и сообщил это моим менеджерам. Вскоре после этого старший менеджер Отделения вычислительной техники должен был лететь в Нью-Йорк, чтобы попытаться возобновить договор о продаже компьютера Elliott заказчику, в последнюю минуту отменившему свой заказ в пользу компьютера IBM. Перспектива получить компилятор АЛГОЛ 60 для компьютера Elliott 803 произвела на этого пользователя такое впечатление, что он все-таки изменил свои намерения в нашу пользу. Это доставило нашему скромному проекту немалую репутацию, а также — чего никогда раньше не было — жесткий срок поставки! Могу с удовольствием сообщить, что мы в этот срок уложились; хотя, как это обычно бывает, потребовалось немало дополнительного труда, чтобы сделать компилятор более удобным в эффективной операционной среде.

После неожиданного успеха нашего АЛГОЛ-компилятора, мы обратились к более честолюбивому проекту: разработать математическое обеспечение ряда операционных систем для более сложных конфигураций компьютера, с устройствами для чтения перфокарт, устройствами построчной печати, магнитными лентами и даже с внешней памятью на магнитных сердечниках. Это последнее устройство было вдвое дешевле и вдвое больше оперативной памяти, но в пятнадцать раз медленнее. Наше программное обеспечение стало известно под названием Elliott 503 Mark II.

Я составил документацию, описывающую соответствующие понятия и устройства, и мы разослали ее имеющимся и перспективным заказчикам. Мы начали работу с командой из пятнадцати программистов, с намеченным сроком поставки примерно через восемнадцать месяцев, в марте 1965 года. После того, как я инициировал разработку программного обеспечения Mark II, я был внезапно назначен на головокружительный пост помощника главного инженера, ответственного за развитие и проектирование продуктов компании,— как аппаратуры, так и программного обеспечения.

Хотя я по-прежнему отвечал в качестве руководителя за программное обеспечение системы Elliott 503 Mark II, я стал уделять ему меньше внимания, чем новым продуктам компании, и едва не упустил из виду, что срок поставки миновал без всякого результата. Программисты пересмотрели свои планы, и была установлена новая дата поставки — тремя месяцами позже, в июне 1965 года. Незачем говорить, что и этот срок прошел без результата. К тому времени наши заказчики начали раздражаться, и мои менеджеры распорядились, чтобы я принял на себя личную ответственность за проект.

Стало ясно, что первоначальные спецификации программного обеспечения не могут быть выполнены, и должны быть резко сокращены. Были отозваны занятые на других проектах опытные программисты и даже менеджеры. Мы решили сосредоточиться сначала на поставке нового компилятора для АЛГОЛ'а 60, что должно было занять, согласно тщательной оценке, еще четыре месяца. Программисты работали день и ночь, чтобы завершить все блоки компилятора АЛГОЛ. К нашему удовольствию, они справились с этой работой в срок. Это была первая крупная продукция действующего программного обеспечения, выпущенная компанией за два года.

Наша радость продолжалась недолго: поставка компилятора оказалась невозможной. Вследствие несогласования оперативной памяти с внешней памятью, скорость компиляции составила всего два символа в секунду, что выглядело весьма неблагоприятно, поскольку существующая версия компилятора обрабатывала около тысячи символов в секунду. Создалось безвыходное положение: приходилось отказаться от всего проекта программного обеспечения

Elliott 503 Mark II, и вместе с этим пропадало свыше тридцати человеко-лет программной работы, что было эквивалентно приблизительно всей активной трудовой жизни одного человека. И за эту расточительность отвечал я, как разработчик и как руководитель. Как же мы справились с этой катастрофой?

Мы снова начали с нашего первоначального работающего компилятора для языка АЛГОЛ 60, и стали прибавлять к нему минимальные добавочные средства, которых от нас требовали заказчики. Этот план удался. Требуемое программное обеспечение начало поставляться в назначенные сроки. По мере того как возрастала наша уверенность и уверенность наших заказчиков, мы смогли приняться за выполнение несколько более трудных требований. В течение года мы справились с бедствием. В конце второго года у нас уже появились достаточно удовлетворенные потребители. Таким образом, с помощью здравого смысла и компромисса мы добились некоторого успеха. Но я был недоволен.

Я не мог понять, почему проектирование и реализация операционной системы должны быть настолько труднее, чем в случае компилятора. По этой причине я посвятил мои дальнейшие исследования проблемам параллельного программирования и языковым конструкциям, которые могли бы обеспечить строгое структурирование операционных систем — таким конструкциям как мониторы и связанные последовательные процессы. Когда компания осознала, что она оправилась от прошлых неудач, наступило время обратить внимание на будущее, на проектирование еще более мощных быстродействующих машин, необходимых для того, чтобы идти в ногу с успехами в области аппаратной технологии и машинной архитектуры. Для этой цели я был освобожден от должности технического руководителя Отделения развития и назначен главным научным специалистом Отделения компьютерных исследований.

С небольшой группой коллег мы занялись исследованиями таких архитектурных нововведений как сверхоперативная память и системы замещения страниц (подкачки), именуемых теперь виртуальной памятью. Мы осознали, что решающим фактором для продажи любого нового компьютера становится его программное обеспечение, и в особенности, его операционная система. Компания все еще не поставляла такие системы, а я даже не знал, как следует, что это такое. Поэтому я посетил Математическую лабораторию Кембриджского университета, чтобы изучитькомпьютер Titan и операционную систему Titan. Эта система была разработана сотрудниками и преподавателями университета под руководством Мориса Уилкса, лауреата премии Киото по информационным технологиям за 1992 год. Я воспользовался его работами в области сверхоперативной памяти, а также его пониманием структуры операционной системы в целом.

В 1967 году компания Elliott Brothers была поглощена другой конкурирующей компанией, и наше намерение спроектировать новый компьютер, было отменено. В следующем году мы слились с еще большей компанией. Я просматривал тогда предложения о трудоустройстве, и нашел вакантное место заведующего кафедрой информатики Королевского университета в Белфасте. Как это ни странно, я послал туда заявление и, к моему большому удивлению, меня приняли на эту должность. Так началась моя университетская карьера. Во всех исследованиях, предпринятых мною с тех пор, я вдохновлялся проблемами, с которыми встретился в моей предыдущей промышленной карьере и в моем последующем сотрудничестве с промышленностью. Меня вдохновляло также значение для компьютерных наук некоторых глубоких философских идей и размышлений, занимавших человеческий ум в течение ряда столетий.

Я решил разобраться в этих вопросах на самом серьезном уровне. Меня не беспокоило, что это занятие могут назвать исследованием оснований, чистой наукой или наукой будущего. В самом деле, как я и предсказывал, мои ранние исследования по теории программирования стали применяться в промышленности лишь после того, как я завершил мою университетскую карьеру. Это было для меня подлинной поддержкой, поскольку оказалось, что избранные мною предметы исследования могут доставить мне ряд интересных задач в течение всей моей научной жизни. Так оно и было. В прошлом году я достиг предельного возраста ухода на пенсию, установленного в британских университетах. Но на этом не закончилась моя исследовательская карьера. Я занял должность старшего научного сотрудника в кембриджском исследовательском отделении ведущей всемирной компании программного обеспечения Майкрософт. Здесь я получил необычайную возможность непосредственно наблюдать, как некоторые из моих ранних исследовательских результатов, вместе с результатами некоторых других ученых, постепенно

пробивают себе дорогу в промышленную практику. В настоящее время они используются преимущественно при испытаниях и отладке программ. Они начинают находить применение в проектировании современных языков программирования.

Можно надеяться, что развитие новых средств программирования будет еще более тесно связано с основополагающими теориями программирования. Мне доставило большое удовольствие поделиться с вами этими рассказами о прожитой мной интересной жизни. Надеюсь, они показывают, что несмотря на частые перемены направления, иногда приводившие к драматическим противоречиям, я предпринимал серьезные усилия, чтобы объединить открытия великих философов и логиков прошлого с опытом успехов и неудач в промышленности, чтобы использовать их для построения фундаментальных теорий в новой важной области техники. Для младших слушателей этой аудитории мои странные приключения могут содержать некий урок: не следуйте ни моему примеру, ни примеру кого-нибудь другого. Выберите свой собственный путь в том направлении, которое вас больше всего интересует в данное время. Если вы хорошо чувствуете себя в устойчивом течении профессиональной карьеры, я желаю вам всяческого успеха. Но если ваши интересы ведут вас в необычном направлении, не бойтесь отклоняться от принятых шаблонов. Не бойтесь перемен, как бы драматичны ни были возникающие при этом контрасты. Если вы упорно размышляете, обобщая ваш опыт, вы сможете применить его в самых неожиданных случаях. В этом бесконечном мире у каждого человека свой собственный опыт, собственные интересы и собственная личность. И мы должны поощрять и развивать это безграничное разнообразие, чтобы обеспечить непрерывный прогресс наших знаний, непрерывное обновление наших всевозможных культур, и, в конечном счете, благополучие и счастье человечества.