1968
Одна из точек зрения на информатику

Р. В. Хэмминг

Белл телефон лабораториз, Инк.
Мюррей Хилл, Нью Джерси

В последнее время многие специалисты склоняются к тому, что информатика должна уделять больше внимания практическим вопросам. Технический аспект имеет важное значение потому, что большинство из возникающих в настоящее время трудностей связаны не с теоретической возможностью сделать что-либо, а скорее с практическим вопросом, каким образом это можно сделать лучше и проще.

Преподавание информатики могло бы стать более эффективным, если внести в него некоторые изменения, например включение в учебные планы практикума по программированию, уделение большего внимания нематематическим дисциплинам, больше практического программирования и меньше абстрактной теории, больше серьезности и меньше игры.

Разрешите мне сначала сказать несколько слов о своих личных впечатлениях. Когда человеку сообщают, что ему присуждена премия Тьюринга, то сначала это его удивляет, особенно если премия присуждена нетеоретику. Через некоторое время удивление сменяется радостью. Несколько позже возникают вопросы. «А почему я? Почему на фоне всего того, что сделано и делается в области вычислений ЭВМ, выделили меня и мою работу?» Мне кажется, это происходит ежегодно с кем-нибудь, а в этом году счастье выпало мне. В любом случае разрешите поблагодарить вас за оказанную мне честь. Это признание относится также и к компании «Белл телефон лабораториз», где я работаю и которая создала все условия для достижения успеха.

Я выбрал для своей лекции тему «Одна из точек зрения на информатику», поскольку вопрос: «Что такое информатика?» постоянно обсуждается специалистами. К тому же во введении к блестящему докладу «Программа-68» [1] отмечается: «Комитет надеется, что продолжающийся диалог по процессу и целям обучения информатике останется первостепенным в ближайшие годы». Наконец, неправильно полагать, что Тьюринг, в честь которого названы наши ежегодные доклады, занимался исключительно машинами Тьюринга; в действительности он внес значительный вклад во многие направления этой науки и наверняка заинтересовался бы данной темой, хотя, возможно, и не совсем тем, о чем я буду говорить.

Вопрос: «Что такое информатика?» в настоящее время звучит в различных формах, среди которых основными являются:

«Что такое информатика сегодня?», «Во что она может развиться?», «Во что она должна развиться?», «Во что она разовьется?».

Ни на один из этих вопросов невозможно дать точный ответ. Много лет назад один знаменитый математик написал книгу «Что такое математика?» и нигде даже не попытался дать определение математике, он просто написал книгу о математике. И даже если вы время от времени будете находить более четкое определение некоторых разделов математики, все равно ее наиболее принятым определением останется: «Математика — это то, чем занимаются математики», за которым следует другое: «Математики — это люди, которые занимаются математикой». То, что является правильным в определении математики, является правильным и в определении других наук: чаще всего не существует ясного, четкого определения отдельной научной области.

Сталкиваясь с подобными трудностями, многие, в том числе иногда и я, чувствовали, что не следует принимать участия в таких дискуссиях, нужно просто продолжать *работу* в этой области. Однако, как хорошо отметил Джордж Форсайт в своей недавней статье [2], «имеет значение, как информатику определяют в Вашингтоне, округ Колумбия». По его словам, там склоняются к мнению, что она является частью прикладной математики, и поэтому обращаются к математикам за советом о распределении финансовых средств. Примерно так же обстоит дело повсюду; и в промышленности, и в высших учебных заведениях можно увидеть, где и как зарождалась вычислительная техника: и в электротехнике, и в физике, и в математике, и даже в области бизнеса. Естественно, что представления людей об этой области науки могут в значительной мере повлиять на ее дальнейшее развитие. Таким образом, хотя мы и не можем надеяться на окончательное разрешение этого вопроса, нам необходимо чаще обновлять и пересматривать наши взгляды на то, чем является предмет нашего обсуждения и чем он должен стать.

Во многих отношениях мне было бы удобнее поговорить об узкой методической стороне информатики — во всяком случае, это было бы проще. Однако мне хотелось бы особо подчеркнуть опасность потеряться в деталях предмета, особенно в ближайшее время, когда ожидается настоящий потоп научных статей. Мы должны уделять достаточное внимание разносторонности подготовки специалистов, причем в условиях, когда возрастает необходимость специализации для получения тем диссертационных работ, публикации множества статей и др. Мы обязаны готовить студентов к 2000 году — году, когда многие из них достигнут вершины в своей профессиональной деятельности. Мне

кажется, что выражение «специализация — путь к тривиальности» более всего подходит именно к информатике.

Я уверен, вы знаете, что объем наших научных знаний удваивается каждые 15—17 лет. Но я более чем уверен, что в информатике темпы его роста сейчас намного выше, во всяком случае, они были более высокими в течение последних 15 лет. Мы должны учитывать такой прирост информации во всех наших планах и признать, что мы стоим на пороге появления «почти бесконечного» объема знаний. Во многих отношениях классическое представление о том, что ученый знает около 90% всей информации, связанной с предметом его исследований, сегодня уже неверно. Все более узкая специализация не может стать решением этой проблемы, так как часть трудностей состоит в быстро растущем взаимодействии отраслей знаний. По моему личному мнению, нам необходимо обращать больше внимания на качественную сторону материала, чем на количественную, и учитывать, что внимательный, критический, хорошо продуманный научный обзор может гораздо больше повлиять на развитие нашей области, чем новый второстепенный материал.

Мы живем в мире серых полутонов, но для обсуждения проблемы (или, вернее, даже для ее обдумывания) часто необходимо четкое разделение на «черное» и «белое». Правда, поступая так, мы грешим против истины, но иначе мы, по всей видимости, не сможем двигаться вперед. Я полагаю, что большую часть противопоставлений в моем докладе вы будете рассматривать именно в этом свете. Честно говоря, я сам в каком-то смысле не верю в их абсолютную истинность, но мне кажется, нет другого такого же простого способа обсуждения этой темы.

В качестве примера разрешите мне привести условное различие между фундаментальными и прикладными исследованиями: фундаментальная наука изучает, *что* является возможным, а что — нет, в то время как прикладная наука занимается *выбором* из множества возможных путей *такого,* который бы соответствовал многим, часто плохо сформулированным экономическим и практическим целям. Мы называем наш предмет «информатикой», но мне кажется, что точнее было бы назвать его «компьютерной инженерией» (computer engineering), если бы не существовало вероятности неправильного толкования такого названия. Большей частью мы не подвергаем сомнению возможность существования монитора, алгоритма, планировщика или компилятора, скорее мы занимаемся поиском практически работоспособного технического решения с разумными затратами времени и усилий. Хотя я и не стану менять название «computer science» на computer engineering», я все-таки

хотел бы, чтобы в преподавание этой дисциплины вносилось больше практического, прикладного, чем мы обычно находим в учебных программах.

Существует и другая причина выделения практической стороны. Предугадывая, насколько это возможно, будущее нашего предмета, я предвижу, что скоро мы будем нуждаться в больших средствах. Но, как известно, общество обычно, хотя и не всегда, охотнее выделяет средства на те цели, которые дают практическую отдачу, чем на цели, которые оно рассматривает как непрактичную деятельность, занятные игры и т. д. Если мы хотим получить крупные суммы, в которых несомненно будем нуждаться, нам необходимо найти практическое применение нашему предмету. Многие из вас заметили, что мы уже заработали себе плохую репутацию во многих областях. Конечно, есть некоторые исключения, однако все вы знаете, как плохо до сих пор мы удовлетворяли потребности в программном обеспечении.

В основе информатики лежит техническое устройство — вычислительная машина. Без нее почти все, чем мы занимаемся, превратится в пустое словословие, едва ли отличающееся от знаменитой схоластики средневековья. Основатели ACM ясно высказались, что все то, чем мы занимались или чем мы должны были заниматься, основывалось на этом техническом устройстве, и они намеренно ввели в название своей ассоциации термин «машина» (machinery). Есть специалисты, которым бы хотелось исключить этот термин с целью символического освобождения нашей сферы деятельности от реальности, однако до сих пор их усилия не увенчались успехом. Я не сожалею о первоначальном выборе и сейчас еще полагаю, что для нас очень важно признать тот факт, что компьютер — машина, обрабатывающая информацию — лежит в основе нашей сферы деятельности.

Каким образом мы сумеем привнести практический аспект, о котором я говорю, а также восстановить нашу репутацию в глазах общества и дать ему то, что необходимо в данный момент? Может быть, наиболее значительным направлением является то, которое мы избрали в нашей практической деятельности и преподавании, хотя и проводимые нами исследования также будут иметь большое значение. Крайне важно избежать пустословия и игры, которым часто предаются «чистые» математики. Прав или неправ «чистый» математик, утверждая, что то, что сегодня кажется совершенно бесполезным, завтра может пригодиться? Учитывая сегодняшнюю ситуацию, я очень сомневаюсь в его правоте. Это не тот путь, который помог бы получить крупные средства, так необходимые для дальнейшего развития нашей науки. Нельзя считать информатику похо-

жей на математику: основным критерием приемлемости должен быть опыт реального мира, а не эстетические соображения.

Если бы мне пришлось составлять программу преподавания нашего предмета, я бы сделал больший, чем в «Программе-68», упор на лабораторную работу и, в частности, обязал бы любого работающего в этой области специалиста, аспиранта или студента посещать лабораторные занятия, где они должны разработать, написать и отладить программу достаточного размера, а также составить документацию. Эта программа может быть моделирующим устройством или упрощенным компилятором для определенного компьютера. О результатах можно было бы судить по стилю программирования, практической эффективности, количеству технических дефектов и по документации. Если хотя бы один из перечисленных этапов работы выполнен плохо, я бы не счел такого кандидата успешно сдавшим курс. При оценке подобной работы необходимо делать четкое разграничение между сообразительностью и настоящим пониманием предмета. Сообразительность была существенна в прошлом, но в настоящее время ее совершенно недостаточно.

Я потребовал бы также введения дополнительного курса еще в одной серьезной области помимо информатики и математики. Без опыта работы на компьютере над реальной задачей наш выпускник может знать все о замечательном инструменте, за исключением того, как им пользоваться. Такой специалист является всего-навсего техником, обладающим большими навыками умелого манипулирования инструментом, но не знающим, как и когда использовать компьютер по его истинному назначению. Я считаю, что настала пора прекратить подготовку «ученых дураков» — у нас более чем достаточно «компьютерщиков». Сейчас нам нужны профессионалы!

Необходимость программирования реальных задач признана и в «Программе-68» в следующей форме: «Эта цель может быть достигнута путем организации летней практики, совмещения работы и обучения, введения неполного рабочего дня в компьютерных центрах, организации специальных курсов или каким-либо другим подходящим способом». Я считаю, что таким способом могут стать лабораторные курсы с жесткой программой под вашим собственным контролем, а все вышеприведенные предложения Комитета навряд ли будут эффективными или достаточными.

Возможно, наиболее неоднозначным вопросом в составлении учебных программ по подготовке специалистов по информатике является включение в них математических курсов. Многие из нас пришли в эту область, имея солидную математическую подготовку, и поэтому считаем, что такая подготовка необходи-

ма всем специалистам. Слишком часто учитель пытается сделать из ученика свою собственную копию. Однако нетрудно заметить, что в прошлом многие настоящие специалисты в области программного обеспечения не имели соответствующей математической подготовки, хотя большинство из них были прирожденными математиками (в смысле того, чем математика является в действительности, а не того, как ее зачастую преподают).

В прошлом я считал, что требование глубоких математических знаний при подготовке исключит из информатики большую часть лучших специалистов. Но с возрастанием значимости планирования и распределения ресурсов компьютеров я вынужден был пересмотреть свою точку зрения. Хотя и очевидно, что частично это будет решаться аппаратными средствами, трудно предположить, что в ближайшее время, как минимум, в течение пяти лет, программисту не потребуется много заниматься планированием и распределением ресурсов. Если это действительно станет характерным, то мы будем вынуждены серьезно рассмотреть вопрос о подготовке специалистов. Если мы не будем давать такой подготовки, то наш специалист по информатике придет к пониманию того, что он является просто «техником», который всего лишь программирует то, что ему заказывают другие. Более того, программистское мастерство, которое раньше мы считали большим достижением, часто зависело от сообразительности и ловкости программиста и почти не требовало математических знаний. Кажется, этот период уже проходит, и если мы хотим, чтобы наши выпускники были способны внести серьезный вклад в науку, мы обязаны дать им хорошую математическую подготовку.

История показывает, что лишь относительно небольшая часть людей в возрасте после 30 лет, не говоря уже о более старшем возрасте, может действительно освоить новое в математике; таким образом, если в будущем математика будет играть значительную роль, мы должны дать учащимся соответствующую математическую подготовку еще в школе. Мы, конечно, можем временно обойти это решение, предложив учащимся два параллельных курса: один с изучением математики, другой без ее изучения, предупредив, однако, что второй путь ведет в тупик, по крайней мере в отношении последующего университетского обучения (предполагая, что мы убеждены в первостепенном значении математики для обучения на факультете информатики).

Признав необходимость основательной математической подготовки, мы тут же сталкиваемся с еще более сложной задачей — понять, какие именно курсы нам нужны. Несмотря на многочисленные утверждения теоретиков о фундаментальной

важности математики, необходимо признать, что мы используем сравнительно малую ее часть на практике. Однако, по моему мнению, необходимо, чтобы каждый специалист в области информатики прослушал хотя бы один математический курс. Но, видимо, все дело в том, что организация курсов формальной математики не соответствует сегодняшим требованиям нашего предмета. Нам нужны абстрактная алгебра, теория массового обслуживания, статистика, включая планирование экспериментов, основы теории вероятности, возможно с некоторыми элементами цепей Маркова, отдельные вопросы теории информации и кодирования, кое-что о скорости передачи сигналов, некоторые разделы теории графов и т. п. Однако, как хорошо известно, наша область быстро изменяется, и, возможно, завтра нам могут понадобиться теория функции комплексного переменного, топология и другие дисциплины.

Как я уже говорил, планирование математических курсов является наиболее сложной частью программы. После долгих размышлений на эту тему я пришел к выводу, что если мы хотим, чтобы наши выпускники внесли значительный вклад в науку и не опускались до уровня техников, умеющих, по мнению других специалистов, только управлять инструментом, лучше дать им слишком много математики, чем слишком мало. Я очень хорошо понимаю, что такой подход может исключить из нашей области многих из тех, кто в прошлом содействовал ее развитию, и этот вывод меня совсем не радует, однако он именно таков. Дальнейшее развитие информатики требует владения математикой.

Наиболее часто учебные программы по информатике упрекают в том, что они почти полностью игнорируют практическую сторону и язык Кобол. Я полагаю, что вопрос о том, нужно или нет преподавать на факультете информатики практическое применение компьютеров или язык Кобол, связан отнюдь не с их прикладным значением. Скорее, как мне кажется, вопрос заключается в том, может ли факультет административного управления выполнить эту работу намного лучше нас и является ли то, что присуще приложению к бизнесу, принципиальным для других аспектов информатики. И то, что говорилось о применении компьютеров к бизнесу, надо полагать, относится и к другим областям их применения, которые можно преподавать на других факультетах. Я глубоко уверен, что с теми ограниченными ресурсами, которыми мы располагаем в данный момент и которыми будем располагать еще в течение длительного времени, мы не должны пытаться преподавать практические применения компьютеров на нашем факультете, скорее их нужно вводить на тех факультетах, где и предполагается их использование.

Проблема роли аналоговых вычислений в учебной программе факультета информатики отличается от подобной проблемы, стоящей перед применением компьютеров в специальных областях, так как нет другого места, где она могла бы решаться. Нет ни малейшего сомнения в том, что аналоговые вычислительные машины экономически важны и будут оставаться такими еще некоторое время. Но нет также ни малейшего сомнения и в том, что класс таких машин, включая даже гибридные компьютеры, не обладает в настоящее время теми интеллектуальными возможностями, которые присущи цифровому вычислению. Кроме того, сущность хорошего аналогового вычисления заключается в понимании физических ограничений оборудования и в специфическом искусстве масштабирования, особенно по временной переменной, что достаточно чуждо остальной области информатики. Таким образом, налицо тенденция скорее к игнорированию аналогового вычисления, чем к отказу от него. Его или не преподают, или делают факультативным, и это, вероятно, лучшее, что можно сделать в настоящее время, когда в центре внимания находится универсальный цифровой компьютер.

В настоящее время чувствуется «игровой» налет в программах многих наших курсов информатики. Мне постоянно приходится выслушивать от друзей, которые хотели бы взять на работу хороших знающих специалистов в области программного обеспечения, что те, кто к ним обращается, не представляют для них интереса как специалисты. По их мнению, наши выпускники в основном стремятся играть в игры, писать хитроумные программы, которые на самом деле не работают, трюковые программы и так далее, но неспособны сконцентрировать свои усилия таким образом, чтобы то, что они обещают, было сделано вовремя и в практически пригодной форме. Если бы я услышал эту жалобу лишь однажды от друга, воображающего себя заправским инженером-практиком, я бы ею пренебрег; к сожалению, я слышал это неоднократно от многих компетентных, разумных и понимающих людей. Как я уже говорил, поскольку мы так отчаянно нуждаемся в финансовой поддержке для текущей работы и будущего расширения наших возможностей, нам следовало бы подумать, как избежать таких замечаний о наших выпускниках в ближайшие годы. Собираемся ли мы продолжать выпускать «продукцию», отвергаемую столь многими, или же мы собираемся выпускать ответственных, работоспособных специалистов, в которых действительно нуждается наше общество? Я полагаю, что необходимость выбора последней альтернативы становится все более ясной, и отсюда мое внимание к практическим аспектам информатики.

Одна из причин, по которым наши выпускники больше заинтересованы в «красивом» программировании, чем в практических результатах, состоит в том, что многие наши учебные курсы преподаются людьми с привычками и навыками «чистых» математиков. Чистый математик начинает с поставленной задачи или с некоего выбранного им варианта постановки этой задачи и получает то, что он называет ее решением. В прикладной математике необходимо добавить два критически важных шага: во-первых, исследование связи между математической моделью и реальной ситуацией и, во-вторых, связи между результатами, выдаваемыми этой моделью, и реальной ситуацией (или, если вам угодно, интерпретация этих результатов в терминах реальной ситуации). Вот в этом-то и заключается резкое различие. Человек, занимающийся прикладной математикой, должен быть готов принять на себя ответственность, заявив: «Если вы сделаете то-то и то-то, вы с высокой точностью увидите то-то и то-то, и поэтому оправданно продвигаться вперед и тратить время и средства указанным способом», тогда как чистый математик только пожмет плечами и заметит: «Я за это не отвечаю». Но кто-то же должен взять на себя ответственность за выбор того или иного направления, и мне кажется, что тот, кто берет на себя такую ответственность, должен получить больше доверия, чем это обычно у нас бывает. Таким образом, в процессе преподавания мы обязаны особенно подчеркивать необходимость принятия на себя полной ответственности за решение *всей* проблемы, а не только чисто математической ее части. Это и есть еще одна причина, почему я обратил внимание на техническую сторону различных предметов и постарался приуменьшить чисто математические аспекты.

Трудность, конечно, заключается в том, что большинство преподавателей в информатике являются чистыми математиками, а также в том, что преподавать чистую математику гораздо легче, чем прикладную. Относительно немного преподавателей способны вести данный предмет именно так, как мне представляется необходимым. Это означает, что мы должны делать все возможное с помощью тех средств, которыми располагаем. С осторожностью выбирать необходимое нам направление, прививать, где это возможно, свойственные практику чувство ответственности и прагматизма, а не только умение доказывать существование решения.

Очень жаль, что на ранних этапах развития информатики для достижения успеха решающее значение имели талант и способность справляться с громадным объемом мелочей. Но если мы хотим, чтобы студент вырос в специалиста, способного справляться с более обширными разделами информатики, то

он должен иметь и развивать другие способности, которые не используются и не тренируются на начальных этапах обучения. Многие наши выпускники никогда не делают этого второго шага. Примерно так же обстоят дела и в математике: в первые годы обучения требуется тривиальное владение арифметическими операциями и формальными символьными операциями школьной алгебры, но в высшей математике для успеха необходимы совсем другие способности. Как я уже говорил, многие программисты, сделавшие успешную карьеру в области, в которой самую важную роль играют мелочи, не смогли развить более широких способностей, и они продолжают преподавать и распространять свою науку о мелочах. На более высоких уровнях информатики требуется не «черно-белое» мышление, характерное для большей части математики, но способность к взвешенным субъективным суждениям, учитывающим противоречия между поставленными целями, что характерно для прикладных технических дисциплин.

Пока что я обрисовал программирование как дисциплину, для которой, выражаясь словами одного из моих друзей, свойственно изобретение специальных трюков для каждого конкретного случая при отсутствии общих принципов. В такой оценке программирования как трюкачества так много правды, что затруднительно обсуждать вопрос о том, чему следует учить на курсах программирования. Столь многое из того, что мы сделали, сделано специально для конкретных случаев, и мы находились под таким давлением требований получить хоть что-то работающее как можно быстрее, то у нас теперь есть лишь немного ценного, способного выдержать придирчивый взгляд ученого или инженера, спрашивающего: «В чем состоит содержание науки о программном обеспечении?» Сколь немногочисленны трудные для понимания идеи в этой области! Сколь многое представляет собой просто нагромождение одной мелочи за другой без всякого тщательного анализа! И если компиляторы, содержащие 50 тысяч слов, могут быть позже заменены новыми, всего лишь в пять тысяч слов, то как далеки от разумного были первоначальные решения!

Я далеко не эксперт в области программного обеспечения, и мне сложно сделать серьезные предположения по поводу того, что необходимо предпринять в этой области, и тем не менее я чувствую, что очень часто мы довольствовались таким низким уровнем качества, что сами себе нанесли серьезный урон. Кажется, что мы неспособны использовать машину, которая, по нашему общему мнению, является мощным инструментом управления и преобразования информации, для решения наших собственных задач в области программного обеспечения. У нас есть компиляторы, ассемблеры, мониторы и т. п., но

только для других, и в то же время, анализируя повсеместную работу программиста, я часто удивляюсь тому, насколько мало он использует машину в своей собственной работе. У меня накопилось достаточно опыта в дискуссиях со специалистами в области программного обеспечения, чтобы настаивать на обучении использованию машин практически на всех стадиях нашей работы. Очень немногие из системных программистов вообще пытаются использовать компьютер в своей работе. Существует множество ситуаций, когда использование компьютера может оказать программисту огромную помощь. Я припоминаю один случай, когда неспециалист, имевший большую программу на языке Фортран, составленную на стороне, хотел приспособить ее для нашего (местного) использования и составил простую программу на Фортране для локализации всех операторов ввода-вывода и всех библиотечных ссылок. Мой личный опыт показывает, что большинство программистов предпочитают просматривать данные распечатки текстов программ, отыскивая нужные им строки, и, конечно, сначала пропускают некоторые из них. Мне кажется, необходимо убедить программистов, что компьютер является их самым мощным подручным средством и что они должны как можно чаще использовать его в своей работе, особенно при необходимости просмотра длинных списков символов, вместо того чтобы выполнять эту операцию вручную, как это практикуется у нас повсеместно. Если все то, о чем я говорю сейчас, действительно верно, значит, мы упустили этот момент в нашей прежней работе со студентами. Конечно, наиболее подготовленные из них используют компьютер именно так, как я рекомендую, однако, по моим наблюдениям, большинство выпускников-программистов совершенно не пользуются им.

Образно говоря, наши сегодняшние методы преподавания программирования сводятся к тому, что мы даем первокурсникам учебник грамматики и словарь и говорим им, что отныне они великие писатели. Мы практически не знакомим их со стилем программирования. И действительно, до сих пор я не встречал ни учебника стилистики, ни «Антологии программ». Программирование — такое же сложное искусство, как и литературное творчество. И там и здесь предпочтение отдается краткости. Анализируя метод обучения хорошему литературному языку — написание сочинений, составление докладов и выступлений, по которым оцениваются успехи студентов, — мы сразу заметим, что в преподавании программирования подобный подход отсутствует. К сожалению, лишь немногие программисты действительно обладают тем, что я называю «стилем», и стремятся создавать настоящие произведения искусства в программировании. В результате лишь немногие из них

владеют изящным, «поэтическим» стилем, большинство же пишут в грубой, тяжелой «прозе».

Я очень сомневаюсь в том, что стиль программирования тесно связан с каким-либо определенным машинным языком, как и в том, что литературный стиль какого-то определенного естественного языка может действительно значительно отличаться от литературного стиля в другом языке. Конечно, в разных языках существуют какие-то определенные способы выражения мыслей, однако основные нормы хорошего стиля одинаковы для большинства западноевропейских языков, во всяком случае, для тех из них, которые я знаю. И я думаю, что все это верно и для сегодняшних цифровых машин.

Конечно, когда я утверждаю, что в образование необходимо внести больше от инженерного, чем от научного творчества, меня могут неправильно понять. Поэтому я должен пояснить, что пришел в информатику, защитив докторскую диссертацию по чистой математике. Несмотря на то что я постоянно призываю обращать больше внимания на практические занятия и развитие инженерного дара, я тем не менее отдаю себе отчет в том, что в силу отсутствия необходимых нам теорий мы не до конца понимаем то, чем в действительности занимаемся.

Действительно, одним из наших слабых мест, по моему мнению, является то, что если Ньютон мог сказать: «Я видел немного дальше, чем другие, потому что стоял на плечах гигантов», то сегодня мы вынуждены признать, что стоим на ногах друг у друга. Быть может, основной проблемой, стоящей перед нами, является необходимость перехода к такой ситуации, когда мы могли бы продолжить работу предыдущего поколения, а не делать ее заново. Ведь все-таки наука должна накапливать знания, а не бесконечно повторять одно и то же.

Это рассуждение подводит меня к еще одному различию,— различию между ненаправляемым поиском и фундаментальными исследованиями. Все хотели бы заниматься ненаправляемым поиском, и большинству хотелось бы верить, что именно такой тип исследований и является фундаментальным. Мне кажется, что фундаментальными следует считать такие исследования, на результатах которых будет строиться работа других исследователей. В конце концов, какой другой смысл можно вложить в понятие «фундаментальные?» Я полагаю, и опыт это подтверждает, что лишь весьма немногие ученые *способны* проводить фундаментальные исследования. И хотя невозможно с полной уверенностью сказать заранее, станет ли данная работа фундаментальной, можно довольно точно предсказать ее дальнейшую судьбу. Занимаясь этим вопросом, я пришел к заключению, что возможности той или иной работы превратиться в фундаментальное исследование зависят не столько от

проблемы, которую она решает, сколько от подхода к ее решению.

Численные методы являются единственной частью нашей учебной программы, которую принимают практически все, так как они имеют хоть какое-то содержание. Тем не менее нас часто и, надо сказать, справедливо упрекают в том, что бо́льшая часть нашего учебного материала написана для математиков, и действительно, он содержит больше теоретического, чем практического материала. Причина, конечно, заключается в том, что многие специалисты, работающие в этой области, пришли в нее из математики, отчасти так и остались математиками и до сих пор подсознательно подходят ко всему с точки зрения математики. Я уверен, что многие из вас знакомы с моими возражениями по этому поводу, и мне, видимо, не нужно повторять их [3].

Многие коллеги говорили мне, и я сам отмечал, что большинство курсов, предложенных в программе преподавания нашего предмета, чаще всего перегружены деталями, за которыми не видно основных идей. Не стоит давать студентам все методы нахождения вещественных нулей функции, лучше показать наиболее типичные и эффективные из них, которые иллюстрировали бы основные концепции численных методов. И то, что я сейчас говорил о численных методах, еще в большей степени относится к курсам программного обеспечения. Мне, как и другим специалистам, кажется, что эти курсы не содержат достаточного количества фундаментальных идей в области программного обеспечения, которые могли бы оправдать то учебное время, которое для них отводится. Из этих курсов необходимо исключить массу ненужных мелочей и оставить только тот материал, который действительно содержит важные идеи.

Разрешите мне сейчас затронуть деликатную тему, — тему профессиональной этики. Много раз отмечалось, что поведение программистов по сравнению с бухгалтерским персоналом во многом оставляет желать лучшего [4]. Кажется, мы не прививаем нашим студентам чувства «неприкосновенности» информации, касающейся людей или частных компаний. Мои наблюдения говорят о том, что программисты лишь поверхностно знакомы с нормами этического поведения. Например, многие из них совершенно уверены в том, что имеют полное право забрать с собой при смене места работы любую программу. Мы должны иметь это в виду и изучить, каким образом основы этического поведения преподаются на курсах бухгалтерского учета, выпускники которых обладают большими навыками в области этики. Мы много говорим об опасности создания крупных банков личных данных на людей, но практически не занимаемся этическим воспитанием наших выпускников.

Далее я хотел бы коснуться такой темы, как нормы профессионального поведения. О них говорилось в недавних публикациях [5], и они показались мне действительно необходимыми, однако я считаю себя вправе вновь задать вопрос, каким образом они включены в программу обучения наших студентов и действительно ли мы способны привить студентам правила такого поведения. Конечно, недостаточно только зачитывать их в аудитории каждый день. Такой путь обучения этическим и профессиональным нормам поведения неэффективен. Совершенно очевидно, что преподаватели других учебных заведений находят способы доводить до студентов нормы профессионального поведения, которые (хотя впоследствии и не все их придерживаются) преподаются лучше, чем у нас. Таким образом, нам необходимо разобраться в их методах и приспособить их к нашим нуждам.

И наконец, разрешите мне перейти еще к одной актуальной теме, которую мы часто затрагиваем, — теме социальной ответственности. Мы часто говорим об этом на встречах, обсуждаем ее в кулуарах, за чашкой кофе и даже в барах, однако я снова задаю вопрос: «Каким образом тема социальной ответственности раскрывается в наших учебных программах?» Тот факт, что мы не имеем разработанных правил для этого, не может служить оправданием ее отсутствия в наших программах.

Я полагаю, что все три темы — этического и профессионального поведения и социальной ответственности — должны быть в обязательном порядке включены в программу подготовки программистов. Мне лично кажется, что введение отдельного курса по этим темам не будет эффективным. Исходя из моего собственного представления о методах привития этих норм студентам, лучше всего они передаются через поведение самих преподавателей. Поучительны неожиданные вещи: форма, в которой даются замечания преподавателя, его манеры и умение держать себя. Таким образом, сам преподаватель должен понять, что значительная часть его деятельности как преподавателя заключается в выборе способов и методов доведения этих деликатных, ускользающих моментов до своих студентов. Преподаватель не имеет права сказать: «Это меня не касается». Об этом должны постоянно говорить все преподаватели или не говорить вообще. Если эти навыки не прививать студентам каким-либо образом, то наша область сохранит свою сегодняшнюю репутацию, которая может удивить вас, если коллеги с других факультетов честно выскажут свое мнение.

В заключение разрешите мне обрисовать истинную перспективу информатики. Это совершенно новая область, находящаяся в постоянном развитии. У нас было очень мало времени для того, чтобы подготовиться и осуществить на практике многое

из того, что мы уже знали раньше. Но по крайней мере в университетах мы многого добились: нам удалось учредить самостоятельные факультеты, на которых действуют курсы с неплохими программами, необходимым оборудованием и соответствующим профессорско-преподавательским составом. Теперь мы хорошо подготовлены и считаем, что именно сейчас необходимо углубить, усилить и улучшить нашу деятельность таким образом, чтобы мы могли гордиться тем, *что* мы преподаем, *как* мы это преподаем, и студентами, которых мы выпускаем. Мы готовим не просто техников, ученых дураков или «компьютерщиков»; мы отдаем себе отчет в том, что в наше сложное время мы должны готовить специалистов, способных принять на себя ответственность за решение проблем в нашем быстро меняющемся обществе. В противном случае мы должны будем признать, что не состоялись как преподаватели и лидеры в такой интересной и важной области, как информатика.