Программирование как искусство

Дональд Е. Кнут

Отрывок из постановления комитета по премиям Тьюринга 1974 г., процитированный его председателем Бернардом А. Галлером, на презентации этой лекции 11 ноября 1974 г. на ежегодной конференции ACM в Сан-Диего.

Премия Тьюринга ежегодно присуждается ассоциацией ACM человеку, внесшему вклад технического характера в деятельность программистского сообщества, причем такой вклад, который оказал существенное влияние на достаточно важную область информатики.

«Премия Тьюринга 1974 г. присуждается профессору Стэнфордского университета Дональду Е. Кнуту за ряд выдающихся работ в области анализа алгоритмов и разработки языков программирования и в особенности за его вклад в развитие «искусства программирования», осуществленный серией известных книг с этим общим названием. Методы программирования, алгоритмы, теоретические построения, изложенные в указанных книгах, послужили основой преподавания информатики и организующим началом в развитии этой дисциплины».

Однако эта официальная формулировка, обосновывающая присуждение премии Тьюринга за 1974 г., не может в полной мере отразить роль Дональда Кнута в современной информатике и компьютерной индустрии. Я знаком с профессором Аланом Дж. Перлисом, первым лауреатом премии Тьюринга, человеком, который обладает способностью в любой научной дискуссии настолько глубоко проникать в суть рассматриваемых проблем, что его высказывания неизменно оказываются в центре всего последующего обсуждения. Подобным образом алгоритмы, терминология, глубокие идеи, составившие содержание ряда блестящих книг и статей Дональда Кнута, также становятся предметом обсуждения почти во всех областях нашей науки. Такое дается нелегко. Всякому автору известно, какого тяжкого труда и величайшей организованности требует написание даже одного тома. Тем более не могут не вызывать восхищения ясность подхода, терпение и энергия, которыми должен был обладать Дональд Кнут, чтобы поставить перед собой задачу создать семь томов и с такой настойчивостью и скрупулезностью приступить к ее осуществлению.

Важно отметить, что эта премия и другие, которых он также был удостоен, присуждены ему после выхода трех томов его труда. Мы искренне рады признать заслуги и оценить преданность Дональда Кнута нашей науке. Мне чрезвычайно приятно быть председателем комитета, избравшего Дональда Кнута лауреатом премии Тьюринга Ассоциации АСМ 1974 г.

* * *

Когда в 1959 г. начал публиковаться журнал Communications of the ACM, то члены редакционной коллегии АСМ, обсуждая задачи периодических изданий ассоциации, сделали следующее замечание [2]: «Если мы хотим, чтобы программирование стало важной частью информатики и компьютерных исследований, то следует всемерно способствовать превращению программирования из искусства в строгую науку». Эта же тема неоднократно поднималась и в последующие годы; так, в 1970 г. мы читаем о «первых шагах на пути превращения программирования из искусства в науку» [26]. Тем временем мы и в самом деле преуспели в преобразовании программирования в науку, причем замечательно простым способом — решив называть ее «компьютерной наукой»[1].

Во всех этих высказываниях неявно присутствует мысль о чем-то нежелательном для нас в той области человеческой деятельности, которую принято называть искусством; только став строгой Наукой, она может приобрести реальную значимость. Между тем вот уже 12 лет я работаю над серией книг под названием «Искусство программирования». Меня часто спрашивают, почему я выбрал такое название; причем некоторые, по-видимому, не верят, что я действительно назвал их именно так, поскольку мне пришлось видеть по крайней мере одну библиографическую ссылку на книги с названием «Дело программирования»[2].

В своей лекции я попытаюсь объяснить, почему я считаю подходящим именно слово «искусство»; я рассмотрю, что значит «быть искусством» и что значит в противоположность этому «быть наукой»; я попытаюсь выяснить, хороши ли искусства или плохи; и я попытаюсь показать, что правильный взгляд на предмет может помочь нам лучше делать свое дело.

Один из первых случаев, когда возник разговор о названии моих книг, произошел в 1966 г. во время последнего национального съезда АСМ в Южной Калифорнии. Это было еще до вы-

---

[1] «computer science» — информатика.

[2] В названии книг «The Art of Computer Programming» слово «art» («искусство») заменено на «act» («дело», акт»).

хода в свет первого тома. Я припоминаю, что мы с моим другом ужинали в отеле, и он, зная, насколько высоким было мое самомнение уже в те годы, спросил, правда ли, что я собираюсь назвать свои книги «Введением в Дона Кнута». Я ответил, что, напротив, собираюсь назвать их как раз в *его* честь. Звали его Арт Эванс[1]. («Искусство программирования собственной персоной».)

Из этой истории мы можем заключить, что слово «art» («искусство») многозначно. Пожалуй, самое замечательное в этом слове — то, что оно употребляется в нескольких различных смыслах, и каждый из них вполне применим к программированию. Готовясь к лекции, я отправился в библиотеку, чтобы выяснить, что писали люди о слове «искусство» в разные времена. Проведя несколько дней за этим увлекательнейшим занятием, я пришел к выводу, что слово «art», вероятно, одно из самых интересных слов в английском языке.

ИСКУССТВА В ДРЕВНОСТИ

Если обратиться к латинским истокам, то мы обнаружим слова «ars», «artis», означающим «умение». Примечательно также, что соответствующее греческое слово τεχνη является общим корнем слов «техника» и «технология»».

В наше время, когда говорят об «искусстве», то это слово, вероятно, ассоциируется в первую очередь с «изящными искусствами», такими, как живопись или скульптура, но до XX в. в это слово вкладывался совсем другой смысл. Поскольку прежнее значение слова «искусство» все еще проявляется во многих идиомах, особенно когда искусство противопоставляется науке, то я хотел бы потратить несколько минут на рассуждения о слове «искусство» в его классическом смысле.

Первые университеты, основанные в средние века, предназначались для обучения семи так называемым «свободным искусствам»[2], а именно грамматике, риторике, логике, арифметике, геометрии, музыке и астрономии. Заметим, что набор дисциплин весьма далек от программы современных гуманитарных колледжей и что по крайней мере три из семи свободных искусств являются важными компонентами информатики. В те времена под искусством понималось нечто, порождаемое силой человеческого интеллекта, в отличие от видов деятельности, вытекающих из природных потребностей или инстинктов. «Свободные искусства» — это освобожденная, духовная деятель-

---

[1] Art Evans, art (англ.) — искусство.

[2] «Liberal Arts» — в современном английском языке это словосочетание означает «гуманитарные науки».

ность в противоположность искусствам, связанным с физическим трудом, вроде пахоты (ср. с [6]). В средние века слово «искусство» само по себе обычно обозначало логику [4], под которой в свою очередь понималось изучение силлогизмов.

НАУКА В ПРОТИВОПОЛОЖНОСТЬ ИСКУССТВУ

Слово «наука», по-видимому, долгое время употреблялось примерно в том же смысле, что и «искусство», говорили, например, также о семи свободных науках, тех же самых, что и семь свободных искусств [1]. Дунс Скотус в XIII в. называл логику «Наукой Наук, Искусством Искусств» (см. 12, с. 34f). С развитием цивилизации и познания слова эти приобретали все более самостоятельные значения: под наукой понималось знание, под искусством — приложение знания. Так, наука астрономия составляла фундамент искусства навигации. То есть различие между этими словами было примерно такое же, как в наше время между словами «наука» и «техника».

В XIX в. о соотношении между наукой и искусством писали многие авторы, и, на мой взгляд, наилучшее рассуждение по этому поводу принадлежит Джону Стюарту Миллю. В его работе [28], 1843 г., говорится, в частности, следующее:

> Для того чтобы составить фундамент одного искусства, чаще всего необходимо бывает несколько наук. Человеческая деятельность слишком сложна, поэтому для того, чтобы что-то сделать, зачастую требуется знать природу и свойства многих вещей... Искусство, вообще говоря, состоит из ряда научных истин, организованных наиболее удобным образом не с точки зрения их логического осмысления, а с точки зрения их практического применения. Наука организует и упорядочивает свои истины таким образом, чтобы как можно шире охватить с единых позиций закономерности Вселенной. Искусство... соединяет в себе многие сведения из области науки, иногда далекие друг от друга, относящиеся к созданию разнообразных и разнородных условий, необходимых для произведения всевозможных действий, вызванных потребностями практической жизни.

Просматривая многие и многие рассуждения о смысле слова «искусство», я обнаружил, что уже по крайней мере в течение двух столетий различные авторы призывают к превращению искусства в науки. Например, в предисловии к учебнику минералогии, написанному в 1784 г., читаем: «До 1780 г. минералогия, хотя и воспринималась многими с некоторой натяжкой как Искусство, едва ли могла считаться Наукой».

В большинстве словарей слово «наука» определяется как логически организованное и систематизированное знание в форме общих «законов». Преимущество науки заключается в том, что она избавляет нас от необходимости обдумывать множество частных случаев, позволяя мыслить с помощью понятий более высокого уровня абстракции. Как писал в 1853 г. Джон

Раскин [32]: «Задача науки состоит в том, чтобы видимость заменять фактами, впечатления — доказательствами».

Я думаю, что если бы авторы, работы которых я изучал, жили в наши дни, то они согласились бы с такой формулировкой: «Наука — это знание, настолько понятное нам, что мы могли бы обучить ему вычислительную машину; там, где мы еще не все до конца понимаем, начинается область искусства». Понятие алгоритма или программы — чрезвычайно полезный тест глубины наших познаний о каком-то предмете, поэтому превращение искусства в науку означает, что мы способны автоматизировать данную область деятельности.

Несмотря на то что искусственный интеллект достиг значительного прогресса, существует тем не менее огромная пропасть между тем, что способны делать обычные люди, и тем, что смогут в обозримом будущем делать вычислительные машины. Таинственные вспышки озарения, проявляющиеся у людей, когда они говорят, слушают, занимаются творчеством и даже пишут программы, все еще находится вне досягаемости науки; поэтому почти все, что мы делаем, до сих пор является искусством.

С этой точки зрения, разумеется, весьма желательно превращение программирования в науку, и за 15 лет, прошедшие со дня публикации тех замечаний, которые я процитировал в начале своей лекции, мы несомненно проделали немалый путь в этом направлении. Пятнадцать лет назад мы настолько плохо понимали, что такое программирование, что даже *помыслить* не могли о доказательствах правильности программ; мы простонапросто гоняли программу до тех пор, пока не «убеждались», что она работает. В те времена мы не представляли даже, как достаточно строго сформулировать саму *идею* правильности программы. Лишь в последние годы мы начали разбираться в процессах абстракции, лежащих в основе конструирования и понимания программ. Наше новое знание о программировании дает громадную практическую отдачу, несмотря на то что лишь для небольшого числа программ реально получено строгое доказательство правильности; однако благодаря ему мы стали лучше понимать принципы построения программ. Суть в том, что теперь, когда мы пишем программу, мы уже знаем, что в принципе при желании могли бы построить строгое доказательство ее правильности, так как нам уже известно, как формулируются подобные доказательства. Этот новый научный фундамент позволяет писать программы, значительно более надежные, чем те, что мы писали раньше, когда в основе суждений об их правильности лежала только интуиция.

Область «автоматизации программирования» — одно из основных направлений исследований по искусственному интеллек-

ту. Приверженцы этого направления были бы счастливы, если бы могли прочесть лекцию под названием «Программирование как артефакт[1])» (имея в виду, что программирование превратилось всего-навсего в реликвию минувших времен), поскольку их цель — создать машину, способную по одной только спецификации задачи написать программу лучше любого из нас. Лично я не думаю, что эта цель когда-либо может быть вполне достигнута, тем не менее я убежден, что эти исследования имеют чрезвычайно важное значение, поскольку любое новое знание о программировании помогает нам усовершенствоваться в своем искусстве. В таком смысле следует постоянно стремиться превращать в науку всякое искусство, поскольку этот процесс способствует развитию самого искусства.

Я не могу устоять перед искушением рассказать здесь еще один случай, имеющий отношение к науке и искусству. Несколько лет назад, когда я был в Чикагском университете, при входе в одно из зданий я обратил внимание на две таблички. На одной из них было написано «Информатика» и изображена стрелка направо. На другой было написано «Информация» и нарисована стрелка налево. Иначе говоря, наука об информации идет одним путем, искусство информации — другим.

НАУКА И ИСКУССТВО

Итак, мы пришли к выводу, что современное программирование является одновременно и искусством, и наукой и что эти два аспекта замечательнейшим образом друг друга дополняют. По-видимому, и большинство других авторов, задавшихся аналогичным вопросом, приходили точно к такому же выводу — что их предмет также является и наукой, и искусством независимо от того, о каком именно предмете идет речь (см. [25]). Я обнаружил самоучитель по фотографии, написанный в 1893 г., в котором утверждается, что «занятие фотографированием — это одновременно и искусство, и наука» [13]. На самом деле, даже взяв в руки словарь, для того чтобы справиться о значениях слов «искусство» и «наука», и случайно заглянув в предисловие редактора, я обнаружил, что оно начинается словами: «Составление словаря есть одновременно наука и искусство». Редактор словаря Funk & Wagnall [27] замечает, что кропотливый процесс накопления и классификации сведений о словах носит научный характер, в то время как тщательное формулирование их определений требует навыков точного и

---

[1]) Артефакт — в археологии так называют любой обнаруженный при раскопках предмет, сделанный человеческими руками, в отличие от находок, имеющих естественное происхождение. — *Прим. ред.*

лаконичного изложения. «Наука без искусства вряд ли будет полезной, искусство без науки заведомо было бы неточным».

Готовясь к этой лекции, я изучил картотеку Станфордской библиотеки, для того чтобы посмотреть, как другие авторы употребляют слова «искусство» и «наука» в названиях своих книг. Это оказалось небезынтересным занятием.

Я обнаружил, например, две книги с названием «Искусство игры на фортепиано» [5, 15], а также книги с названиями «Научный подход к технике игры на фортепиано» [10] и «Наука практической игры на фортепиано» [30]. Нашлась также книга под названием «Искусство игры на фортепиано: научный подход» [22].

Затем мне попалась небольшая приятная книжица под названием «Нежное искусство математики» [31]. Тут я слегка взгрустнул оттого, что, пожалуй, не смог бы вполне искренне назвать программирование «нежным искусством».

Несколько лет назад мне довелось познакомиться с книгой под названием «Искусство вычислений», выпущенной в Сан-Франциско в 1879 г. автором по имени С. Фрушер Говард [14]. Это была книга по практической деловой арифметике, распроданная в количестве свыше 400 000 экземпляров в нескольких изданиях, вышедших до 1890 г. Меня весьма развлекло чтение предисловия к этой книге, так как из него видно, что взгляды Фрушера и смысл, вкладываемый им в заголовок книги, в корне отличаются от того, что имел в виду я; он пишет: «Знание Науки о числах совершенно не имеет значения, навыки в Искусстве вычислений абсолютно необходимы».

В заголовках некоторых книг употребляются оба слова, «искусство» и «наука», например: «Наука бытия и искусство жития» Махариши Махеш Йоги [24]. Существует также книга под названием «Искусство научного открытия» [11], в которой анализируется, как были сделаны некоторые великие научные открытия.

Однако довольно о классическом понимании слова «искусство». На самом деле, выбирая название для своих книг, я думал вовсе не об искусстве в таком значении, а прежде всего о теперешнем содержании этого слова. Пожалуй, одна из интереснейших книг, с которыми мне довелось познакомиться в ходе моих исследований, — это недавняя работа Роберта М. Мюллера «Наука искусства» [29]. Из всех уже упоминавшихся изданий книга Мюллера ближе всего подходит к выражению той идеи, которую я хотел бы сделать основной темой своей сегодняшней лекции, рассматривая истинное содержание искусства, как мы его понимаем сегодня. Он замечает: «Когда-то считалось, что образное мышление, присущее людям искусства, смертельно для науки. И наоборот, сухая логика науки, полагали

тогда, звучит как смертный приговор для любых взлетов фантазии». Далее он исследует преимущества синтеза науки и искусства.

Научный подход обычно характеризуют такими словами, как логический, систематический, холодный, беспристрастный, рациональный, в то время как художественный подход характеризуют словами эстетический, творческий, гуманитарный, эмоциональный, иррациональный. Мне представляется, что применительно к программированию оба этих на первый взгляд несовместимых подхода чрезвычайно ценны.

Эмма Лехмер в 1956 г. писала, что она считает программирование «в одинаковой мере и точной наукой, и захватывающим искусством» [23]. Х. С. М. Коксетер в 1957 г. заметил, что иногда чувствует себя «не столько ученым, сколько художником» [7]. Именно в это время Ч. П. Сноу высказывает свою тревогу по поводу углубляющейся пропасти между «двумя культурами» среди образованных людей [34, 35]. Он указывает, что если мы действительно стремимся достичь прогресса, то должны научиться сочетать ценности науки и искусства.

ПРОИЗВЕДЕНИЯ ИСКУССТВА

Когда я сижу в аудитории, слушая длинную лекцию, то примерно к этому времени внимание мое обычно начинает рассеиваться. Поэтому я хотел бы знать: не слишком ли вас утомили мои разглагольствования об «искусстве» и «науке»? Я все же надеюсь, что вы в состоянии внимательно дослушать остаток лекции, поскольку как раз сейчас мы подошли к той ее части, которая мне особенно дорога.

Когда я говорю о программировании как искусстве, то я думаю о нем в первую очередь как о некоторой художественной *форме* в эстетическом смысле. Главную цель своей деятельности как преподавателя и автора я вижу в том, чтобы помочь людям научиться писать *красивые программы*. Именно поэтому мне было особенно приятно узнать [33], что мои книги даже появились в библиотеке Изящных искусств Корнуэльского университета. (Однако мои три тома, по-видимому, прочно осели на полках без всякого употребления; боюсь, что служащие библиотеки ошиблись, истолковав название моих книг слишком буквально.)

Я чувствую, что составление программ сродни сочинению стихов или музыки. Как сказал Андрей Ершов [9], программирование способно дать нам одновременно и интеллектуальность, и эмоциональное удовлетворение, поскольку овладеть сложностью и установить систему согласованных правил — это настоящий подвиг.

Более того, читая программы, написанные другими людьми, мы можем воспринимать некоторые из них как настоящие произведения искусства. Я до сих пор не могу забыть волнения, охватившего меня, когда я в 1958 г. читал листинг ассемблерной программы SOAP II Стэна Полея. Возможно, вы сочтете меня сумасшедшим, да и вкусы с тех пор переменились, но в то время для меня было чрезвычайно важно увидеть, насколько изящной может быть системная программа, в особенности по сравнению с теми тяжеловесными кодами, которые я обнаружил, изучая листинги других программ того времени. Возможность писать красивые программы даже на ассемблере — именно это в первую очередь сделало меня приверженцем программирования.

Бывают программы изящные, щегольские, бывают блестящие программы. Я убежден, что можно создавать *грандиозные* программы, *благородные* программы и даже поистине *великие* программы!

Недавно я обсуждал эти идеи с Микаэлем Фишером, и он высказал мысль о том, что программисты могли бы продавать свои авторские программы коллекционерам как произведения искусства. ACM могла бы основать специальную экспертную комиссию для подтверждения подлинности каждого вновь поступившего программистского изделия, а квалифицированные торговые агенты и люди особой новой профессии — критики программ — назначили бы цену. Осуществив эту идею, мы получили бы неплохой способ повысить свои доходы.

ВКУС И СТИЛЬ

Если говорить серьезно, то я рад, что идея стиля в программировании наконец-то обрела признание, и надеюсь, что большинство из вас уже видели великолепную маленькую книжку Кернинга и Плоджера «Начала стилистики программирования» [16]. В этой связи важно помнить о том, что не существует единственного «наилучшего» стиля; каждый из нас имеет свои предпочтения, и было бы ошибкой пытаться принуждать людей к каким-то неестественным для них формам. Часто приходится слышать высказывания: «Я не разбираюсь в искусстве, но я знаю, что мне нравится». Важно, чтобы вам в самом деле нравился стиль, которого вы придерживаетесь; он должен служить для вас наилучшим способом самовыражения.

Эдсгер Дейкстра подчеркивал этот момент в предисловии к своей книге «Краткое введение в искусство программирования» [8].

> Моя цель состоит лишь в том, чтобы объяснить важность хорошего стиля в программировании, однако представленные в книге конкретные сти-

листические элементы служат лишь для иллюстрации тех преимуществ, которые вообще можно получить, используя понятие «стиль». В этом отношении я чувствую себя сродни преподавателю композиции в консерватории. Он не учит своих подопечных тому, как следует писать конкретную симфонию, но он должен помочь им обрести собственный стиль и объяснить им, что под этим подразумевается. (Эта аналогия и привела меня к мысли говорить именно об «Искусстве программирования».)

Зададимся теперь вопросом: что же такое хороший стиль? что такое плохой стиль? В этом отношении не следует быть слишком суровым в суждениях о работах других людей. Философ Джереми Бентам в начале девятнадцатого века сформулировал эту мысль так [3, книга 3, гл. 1]:

> Те, кто судят об изяществе и хорошем вкусе, мнят себя благодетелями рода человеческого, в то время как на самом деле они лишь мешают людям получать наслаждение... Нет такого вкуса, который безусловно заслуживал бы эпитета *хороший,* если только это не вкус к занятиям, которые помимо производимой ими приятности заключают в себе и некоторую долю полезности; не существует и такого вкуса, который можно было бы безусловно охарактеризовать как плохой, если только это не вкус к занятиям, имеющим вредную направленность.

Настаивая на своих предубеждениях и пытаясь «реформировать» вкус другого человека, мы, возможно, сами того не осознавая, лишаем его вполне законного удовольствия. Поэтому я не берусь осуждать многое из того, что делают другие программисты, даже если ни за что не стал бы делать так, как они. Важно лишь, что *сами они* воспринимают создаваемое ими как нечто прекрасное.

В отрывке, который я только что процитировал, Бентам дает нам одно указание относительно того, какие из принципов эстетики предпочтительнее прочих, а именно «полезность» результата. Мы можем располагать определенной свободой в выборе собственных критериев прекрасного, однако замечательно, если изделие, которое кажется нам красивым, воспринимается другими как полезное. Я должен признаться, что обожаю писать программы, и больше всего я люблю писать программы, наилучшие в каком-либо отношении.

Существует, разумеется, множество точек зрения, согласно которым программа может оцениваться как «хорошая». Во-первых, очень хорошо, если программа правильно работает. Во-вторых, часто бывает желательно, чтобы программу можно было легко изменять, когда возникает такая необходимость. Обе эти цели достигаются, если программа легкочитаема и понятна для человека, знакомого с соответствующим языком.

Другое важное свойство, которым должна обладать хорошая программа, — корректное взаимодействие с пользователем, в особенности это касается обработки ошибок во входных данных. Придумывание осмысленных сообщений об ошибках, раз-

работка гибких форматов входных данных, не провоцирующих пользователя на ошибки, есть настоящее искусство.

Еще один важный аспект качества программ — эффективность использования ресурсов вычислительной машины. К сожалению, многие сейчас пренебрегают эффективностью, говоря, что это дурной вкус. Причина заключается в том, что мы переживаем реакцию на те времена, когда эффективность считалась единственным достойным внимания критерием качества программ и программисты были настолько озабочены эффективностью, что ради ее достижения излишне усложняли свои программы. В результате чрезмерных ухищрений эффективность фактически сводилась на нет из-за сложности отладки и сопровождения программ. Суть в том, что программисты тратили слишком много усилий, заботясь об оптимизации не там и не тогда, когда нужно. Поспешная, непродуманная оптимизация является корнем всех (или почти всех) зол в программировании.

Грош была бы нам цена, если бы наши представления об оптимизации сводились лишь к процентам потерянного или выигранного пространства или времени счета. Когда мы покупаем автомобиль, большинству из нас безразлична разница в цене в 50 или 100 долларов. В то же время мы готовы предпринять специальную поездку в определенный магазин, чтобы приобрести за 25 центов вещь, которая стоит 50 центов. Я думаю, что всему свое время и свое место — и заботам об эффективности в том числе. Свои взгляды по поводу истинной роли эффективности я изложил в статье о структурном программировании, которая появится в очередном номере журнала Computer Surveys [21].

МЕНЬШЕ СРЕДСТВ — БОЛЬШЕ УДОВОЛЬСТВИЯ

Я обратил внимание на одно любопытное обстоятельство, связанное с эстетическим удовлетворением, — мы получаем значительно больше удовольствия, если нам удается добиться чего-либо ограниченными средствами. Например, программа, которой я более всего доволен и горд, — это компилятор, который я написал когда-то для примитивной мини-машины с памятью всего лишь в 4096 слов по 16 бит в слове. Человек, сумевший достичь результата при столь суровых ограничениях, может чувствовать себя поистине виртуозом.

Аналогичные явления наблюдаются и в других сферах жизни. Например, люди частенько влюбляются в свои фольксвагены, но мало кто способен влюбиться в линкольн континенталь (хотя он ездит гораздо лучше). В те времена, когда я изучал

программирование, популярным развлечением было составление программ, которые бы помещались на одной перфокарте и выполняли бы как можно больше действий. Думаю, что примерно такое же удовольствие испытывают приверженцы АПЛ, сочиняя свои «однострочники». Любопытно, что и теперь, когда мы преподаем программирование, редко удается по-настоящему увлечь студента, до тех пор пока он не пройдет курс, включающий непосредственную практику на мини-машинах. Работа на мощных машинах с изощренными операционными системами и языками, видимо, не способствует зарождению теплых чувств к программированию, по крайней мере вначале.

Не совсем ясно, как применить этот принцип, для того чтобы сделать работу программистов более привлекательной. Ясно, что программисты не обрадовались бы, если бы начальство вдруг объявило, что будут установлены новые машины с вдвое меньшей памятью, чем прежние. Думаю, вряд ли можно надеяться, что даже самые преданные «асы» программирования будут рады такому новшеству, потому что никто не любит без необходимости расставаться с имеющимися удобствами. Приведу другой пример, который поможет прояснить дело. В 20-х годах режиссеры отчаянно сопротивлялись введению звукового кино, поскольку они законно гордились своим умением передавать человеческую речь без звука. Аналогично, истинные мастера программирования вполне могли бы воспротивиться введению более мощных аппаратных средств; современные устройства массовой памяти, похоже, губят всю прелесть старых добрых ленточных методов сортировки. Однако современные режиссеры вовсе не собираются возвращаться назад к немому кино, и не потому что ленивы, а потому что знают, что и с более совершенной техникой можно делать великолепные картины. Форма искусства изменилась, но остался неограниченный простор для мастерства.

Каким же образом оттачивают они свое умение? Похоже, лучшие режиссеры разных времен осваивали свое искусство в относительно примитивных условиях, часто в других странах, не имеющих развитой киноиндустрии. И наиболее интересные результаты в программировании также, похоже, были получены людьми, не имеющими доступа к большим машинам. Мораль этих наблюдений, по-моему, заключается в том, что мы должны сознательно использовать в образовании идею ограниченных ресурсов. Каждый из нас лишь выиграет, создавая время от времени «игрушечные» программы с заданными искусственными ограничениями, заставляющими нас до предела напрягать свои способности. Нельзя все время купаться в роскоши — это делает нас апатичными. Искусство решения мини-задач на пределе своих возможностей оттачивает наше умение для ре-

альных задач; к тому же, имея такой опыт, мы получим больше удовольствия, работая в менее ограниченных условиях.

Точно так же не нужно шарахаться от «искусства для искусства» или испытывать неловкость по поводу программ, написанных просто для развлечения. Когда-то я получил массу удовольствия, написав программу на Алголе, которая состояла из одной инструкции обращения к процедуре вычисления скалярного произведения, но таким необычным способом, что вместо вычисления этой величины она вычисляла m-е простое число [19]. Несколько лет назад студенты Станфорда были увлечены задачей нахождения кратчайшей программы на Фортране, которая бы печатала сама себя, в том смысле, что текст ее выдачи был идентичен ее собственному исходному тексту. Такую же задачу можно рассмотреть и для других языков. Не думаю, что это было для них пустой тратой времени; да и Джереми Бентам, которого я цитировал выше, вряд ли стал бы отрицать полезность подобного времяпрепровождения [3, кн. 3, гл. 1]. «Напротив, — писал он, — не существует занятия, полезность которого была бы более неоспоримой. В чем же заключается тогда смысл полезности, как не в том, чтобы служить для нас источником удовольствия?»

О ПОЛЬЗЕ ХОРОШИХ ИНСТРУМЕНТОВ

Одна из особенностей современного искусства — подчеркнуто творческий характер. Многие художники в наше время, похоже, менее всего озабочены созданием прекрасных произведений; важна лишь новизна идеи. Я бы не советовал уподоблять программирование современному искусству в таком смысле, но это наводит на одну мысль, которая представляется мне важной. Бывает, что приходится выполнять почти безнадежно скучное программистское задание, которое не оставляет практически никаких возможностей для проявления творчества. В такой ситуации некто мог бы прийти ко мне и сказать: «И таково-то ваше распрекрасное программирование? Хорошо вам рассуждать о том, что я, мол, должен получать удовлетворение, создавая очаровательные изящные программы, но как, скажите на милость, я могу превратить в искусство эту нудную работенку?»

Что же, это правда, не каждая программистская работа сама по себе может доставить удовольствие. Возьмите домохозяйку, которой каждый день приходится вытирать один и тот же стол; не всякая ситуация дает простор для проявления творчества и мастерства. Но и в этом случае дело можно поправить: даже рутинная работа может доставлять удовольствие, если предметы, с которыми вы имеете дело, красивы. На-

пример, вытирание изо дня в день одного и того же стола может и в самом деле нравиться, если это великолепно отделанный стол, изготовленный из дерева ценной породы.

Бывают ситуации, когда мы призваны не сочинять, а *исполнять* симфонию; но исполнение прекрасного музыкального произведения — истинное наслаждение, несмотря на то что наша свобода ограничена предписаниями композитора. Так и программист иногда выступает в роли не столько художника, сколько ремесленника, но и работа ремесленника может доставлять радость, если он имеет дело с хорошими инструментами и материалами.

Поэтому в заключение я хотел бы обратиться к системным программистам и конструкторам вычислительных машин, создающим системы, которыми пользуются все остальные: «Пожалуйста, дайте нам такие инструменты, которыми было бы приятно пользоваться, а не такие, с которыми приходится вести постоянную борьбу. Пожалуйста, дайте нам инструменты, которые улучшали бы наше настроение и тем вдохновляли бы на создание более качественных программ».

Мне бывает очень трудно убедить своих молодых коллег в том, что программирование прекрасно, поскольку первое, что я должен им объяснять, — это как пробивается перфокарта «слэш слэш JOB равно то-то и то-то»[1]. Даже язык управления заданиями можно сконструировать таким образом, чтобы с ним было приятно иметь дело, и он вовсе не обязан быть чисто функциональным.

Конструкторы вычислительной техники могли бы сделать свои машины более приятными в употреблении, например предоставив плавающую арифметику, удовлетворяющую простым математическим законам. То, что мы имеем сейчас, делает чрезвычайно трудным строгий численный анализ; должным образом организованные операции могли бы стимулировать специалистов численного анализа на создание более качественных подпрограмм с гарантированной точностью (см. [20, с. 204]).

Посмотрим теперь, что могут сделать разработчики программного обеспечения. Лучшее средство для поддержания хорошего расположения духа у пользователя — это предоставить ему программы, с которыми можно взаимодействовать. Не нужно делать системы слишком уж автоматизированными, так что вся их работа проходит где-то за сценой, — следует оставлять и программисту-пользователю какие-то возможности для проявления творческих возможностей. Общая черта, свойственная всем программистам, — любовь к непосредственной работе за машиной; так давайте же не будем лишать их этого удоволь-

---

[1] //JOB= ... — одна из команд языка управления заданиями.

ствия. С какими-то заданиями лучше справляется машина, другие лучше предоставить человеческому разумению; правильно построенная программа должна обеспечивать верное соотношение между тем и другим. (В течение многих лет я стремился избегать излишней автоматизации, см. [18].)

Хорошим примером здесь могут служить системы для измерения программ. Многие годы программисты не имели понятия о реальном распределении вычислительных затрат по отдельным участкам программ. Опыт показывает, что почти все программисты неправильно представляют себе, где на самом деле находятся узкие места в их программах; неудивительно, что попытки оптимизации так часто оказывались бесплодными, если программисты не имели информации о реальном распределении времени счета по строкам программы. Подобные усилия напоминают попытки молодой пары составить рациональный семейный бюджет, не имея понятия о ценах на продукты, одежду и домашнюю утварь. Единственное, что предоставлялось программисту, — это оптимизирующий компилятор, который проделывает нечто таинственное над транслируемой программой, но никогда не объясняет, что же он делает. К счастью, наконец-то забрезжила надежда на появление систем, которые все же предполагают наличие у пользователя кое-какого интеллекта; они автоматически предоставляют инструментовку программы и соответствующие указания о реальном распределении затрат. Эти экспериментальные системы имели громадный успех потому, что они дают измеримое улучшение качества программ, и еще потому, что пользоваться ими — одно удовольствие. Поэтому я убежден, что такие системы обязательно станут стандартным элементом программного обеспечения — это лишь вопрос времени. В моей статье в журнале Computer Surveys [21] можно найти дальнейшее обсуждение этого вопроса, а также некоторые соображения по поводу того, как интерактивные программы могут способствовать более полному удовлетворению программистов-пользователей.

Долг разработчиков языков — создать такие языки программирования, которые способствовали бы хорошему стилю, поскольку известно, что стиль существенно зависит от языка, на котором формулируется программа. Всплеск интереса к структурному программированию, который мы сейчас переживаем, показал, что ни один из существующих языков не является идеальным инструментом для манипулирования с программами и структурами данных; неясно также, каким должен быть идеальный язык. Поэтому я предполагаю в ближайшие годы провести ряд тщательных экспериментов в области конструирования языков.

ЗАКЛЮЧЕНИЕ

Итак, мы видели, что программирование — это искусство, поскольку оно является приложением накопленных знаний для практических целей, поскольку оно требует умения и мастерства, и в особенности потому, что продукты программирования могут представлять эстетическую ценность. Программист, который бессознательно ощущает себя художником, получает удовольствие от своей работы и справляется с ней лучше. Поэтому мы можем только радоваться тому, что люди, выступающие на конференциях по программированию, говорят о «State of the Art».

ЛИТЕРАТУРА

1. Bailey, Nathan. The Universal Etymological English Dictionary. T. Cos. London, 1787. См. «Art», «Liberal» и «Science».
2. Bauer, Walter F., Juncosa, Mario L., and Perlis, Alan J. ACM publication policies and plans. J. ACM 6 (Apr. 1959), 121—122.
3. Bentham, Jeremy. The Rationale of Reward. Trans. from Theorie des peines et des recompenses, 1811, by Richard Smith, J. & H. L. Hunt, London, 1825.
4. The Century Dictionary and Cyclopedia 1. The Century Co., New York, 1889.
5. Clementi, Muzio. The Art of Playing the Piano. Trans. from L'art de jouer le pianoforte by Max Vogrich. Schirmer, New York, 1898.
6. Colvin Sidney. «Art» Encyclopaedia Britannica, eds. 9, 11, 12, 13, 1875—1926.
7. Coxeter, H. S. M. Convocation adress, Proc. 4th Canadian Math. Congress, 1957, pp. 8—10.
8. Dijkstra, Edsger W. EWD316: A Short Introduction to the Art of Programming. T. H. Eindhoven, The Netherlands, Aug. 1971.
9. Ershov, A. P. Aethetics and human factors in programming. Comm. ACM 15 [July 1972], 501—505.
10. Fielden, Thomas. The Science of Pianoforte Technique. Macmillan, London, 1927.
11. Gore, George. The Art of Scientific Discovery. Longmans, Green, London, 1878.
12. Hamilton, William. Lectures on Logic 1. Wm. Backwood, Edinburgh, 1874.
13. Hodges, John A. Elementary Photography: «The Amateur Photographer» Library 7. London, 1893. Sixth ed., rewised and enlarged, 1907, p. 58.
14. Hovard C. Frusher. Hovard's Art of Computation and golden rule for equation of payments for schools, business colleges and selfculture... C. F. Hovard, San Francisco, 1879.
15. Hummel, J. N. The Art of Playing the Piano Forte. Boosey, London, 1827.
16. Kernighan B. W., and Plauger, P. J. The Elements of Programming Style. McGrow-Hill, New York, 1974.
17. Kirwan, Richard. Elements of Mineralogy. Elsmy, London, 1784.
18. Knuth, Donald E. Minimizing drum latency time. J. ACM 8 (Apr. 1961), 119—150.
19. Knuth, Donald E. and Merner, J. N. ALGOL 60 confidential. Comm. ACM 4 (June 1961), 268—272.
20. Knuth, Donald E. Seminumerical Algorithms: The Art об Computer Programming 2. Addison — Wesley, Reading, Mass., 1969.
21. Knuth, Donald E. Structured programming with go to statements. Computing Surveys 6 (Dec. 1974), 261—301.

22. Kochevitsky, George. The Art of Piano Playing: A Scientific Approach. Summy — Birchard, Evanston , Ill., 1967.
23. Lehmer, Emma. Number theory on SWAC. Prom. Symp. Applied Math. Soc. (1956), 103—108.
24. Mahesh Yogi, Maharishi. The Science of Being and Art of Living. Allen & Unwin, London, 1963.
25. Malevinsky, Moses L. The Science of Playwriting. Brentano's, New York, 1925.
26. Manna, Zohar, and Pnueli, Amir. Formalization of properties of functional programs. J. ACM 17 (July 1970), 555—569.
27. Markwardt, Albert H. Preface to Funk and Wagnall's Standard College Dictionary. Harcourt, Brace & World, New York, 1963, vii.
28. Mill, John Stuart. A System of Logic, Rationcinative and Inductive. London, 1843. Цитаты взяты из предисловия, § 2, и из Книги 6, гл. 11 (12 в более поздних изданиях), § 5.
29. Mueller, Robert E. The Scienee of Art. Jhon Day, New York, 1967.
30. Parsons, Albert Ross. The Science of Pianoforte Practice. Schirmer, New York, 1886.
31. Pedoe, Daniel. The Gentle Art of Mathematics. English U. Press, London, 1953.
32. Ruskin, Jhon. The Stones of Venice 3. London, 1853.
33. Salton, G. A. Частная переписка. Июнь 21, 1974.
34. Show, C. P. The two cultures. The New Statesman and Nation 52 (Oct. 6, 1956), 413—414.
35. Snow, C. P. The Two Cultures: and a Second Look. Cambridge University Press, 1964.