Руслан Богатырев

О программировании и выборе языка для представления алгоритмов

Источник: Мир ПК, #06/2001

«Студия программирования» постепенно трансформировалась из постоянного раздела журнала «Мир ПК» в реальную лабораторию, где ведется экспериментальное программирование, собирается и обобщается наиболее интересная информация со всех концов планеты об истории, новинках и технологиях программирования — одним словом, обо всем, что имеет самое непосредственное отношение к этой особой интеллектуальной деятельности. В этом новом качестве мы представляем вашему вниманию первый выпуск. Основную часть материалов, готовящихся «Студией программирования», составят «нечисленные» алгоритмы и методы программирования, рассказы о разных стилях и языках, будут представлены статьи по занимательному и экспериментальному программированию. Здесь также найдут свое отражение практические приемы и советы. Ждем ваших писем и работ.

О природе компьютерного программирования споры не утихают почти с самого момента осознания его как особого рода человеческой деятельности. Эдсгер Дейкстра как-то высказал мысль о том, что на этот счет есть два распространенных мнения. Первое: программист должен обладать склонностью к разгадыванию головоломок и любить хитроумные уловки. Второе: программирование есть не более чем та или иная оптимизация эффективности вычислительного процесса. По этому вопросу свои взгляды излагали едва ли не все классики программирования. Дональд Кнут исповедует программирование как искусство, понимая под этим то, что можно писать по-настоящему красивые и изящные программы, что «составление программ сродни сочинению стихов или музыки». Тони Хоар и Дэвид Грис отдают дань прежде всего логике и доказательному программированию. Никлаус Вирт считает, что «из ремесла программирование превратилось в академическую дисциплину». Питер Наур высказывает парадоксальную мысль о том, что «интуиция — это основа, на которой должна строиться вся работа по созданию программного обеспечения». Андрей Петрович Ершов был убежден, что «подчинение программирования промышленным методам работы — это неизбежный факт».

По всей видимости, все эти классики правы, но каждый по-своему. Так что же такое программирование на самом деле? В любом случае, несмотря на неразрывную связь между математикой и программированием, последнее по своей природе все же значительно отличается от первого. В программировании в конечном счете многое определяет человеческий фактор: это не просто возможность творить эфемерные абстракции, а реальное создание того, что можно «потрогать» своими руками и применить в качестве инструмента. Другими словами, программирование ближе всего к инженерной деятельности, конструированию, а потому ему сродни понятие изобретательства. Конструируя в XVII в. один из первых механических компьютеров — арифметическую машину в виде скромной шкатулки с колесиками, великий французский математик Блез Паскаль не осознавал, что ему раньше других удалось вплотную подойти к единству математики, инженерии и программирования.

Обычное программирование (говоря по-другому, кустарное, авторское производство программ) на наших глазах все больше уступает место промышленному производству. Сейчас все отчетливее проявляется различие между программированием и его промышленным эквивалентом — программной инженерией (software engineering), которую А.П. Ершов называл технологией программирования. Но от такого разделения ценность программирования не уменьшается. И программирование в малом (programming-in-a-small) со временем еще в большей степени станет творческой лабораторией, мастерской для программирования в большом (programming-in-a-large). Как очень точно подметил Дэвид Грис, «никто не сможет научиться хорошо составлять большие программы, пока он не научится хорошо составлять малые».

К сожалению, революция в области информационных технологий и значительная практическая востребованность «интеллектуальных» рабочих рук сыграли негативную роль в развитии программирования. «В мире разнообразных интерфейсов, постоянно меняющихся языков, систем и утилит, под непрерывным давлением обстоятельств,— не без горечи отмечает Брайан Керниган,— мы зачастую теряем из вида главные принципы, которые должны быть основанием

любой хорошей программы,— простоту, четкость и универсальность». Среди программистов все резче обозначается грань между свободными художниками и ремесленниками. Из-за хронической нехватки времени программисты, занятые в производственном процессе, подчас просто не имеют возможности детально разобраться в сути используемых ими алгоритмов, не говоря уже о том, что все чаще вынуждены заниматься даже не конструированием, а просто сборкой программных элементов из чужих комплектующих. При этом на них психологически давит груз сомнений в корректности созданного. Как говорил Кен Томпсон, «нельзя доверять программе, которую вы не написали полностью сами».

Алгоритмы безусловно являются краеугольным камнем программирования. Однако их нельзя воспринимать как аксиомы. «Многие программисты настолько слепо верят в алгоритмы, что даже не пытаются задуматься о том, как они работают», — ужасался Кнут. По мнению Вирта, программы — это органичное соединение алгоритмов со структурами данных. С этим спорить трудно, однако алгоритмы и структуры данных — все же еще не программы. Это лишь строительные блоки, из которых можно соорудить нечто, хаотически их нагромождая, а можно возвести красивый, уютный и долговечный дом. Чтобы здание не развалилось, нужен связующий материал. В программировании это модули, предложенные в начале 1970-х годов Дэвидом Парнасом и ставшие в наши дни программными компонентами. Наконец, требуется последнее звено. Интуиция, как неотъемлемая часть любого из видов умственной деятельности, как непосредственное постижение истины без логического обоснования, — это именно то, что способно придать программированию законченный вид. Но это и то, чему научиться неимоверно сложно. Именно ей и вопросам эвристического программирования мы будем уделять наибольшее внимание в рубрике «Занимательное программирование».

Что касается рубрики «В мире алгоритмов», то наш подход будет в большей степени нацелен на прагматический обзор ключевых алгоритмов. К сожалению, эволюция программирования привела к тому, что с течением времени многие идеи были искажены или утрачены. В частности, понятие алгоритмических языков заменилось понятием языков программирования, которые затем стали растворяться в среде реализации (включающей в себя среду трансляции, исполняющую систему и набор «стандартных» библиотек). Тем не менее разделять эти три понятия, на наш взгляд, очень важно. Рассуждения на уровне языка программирования (а уже тем более среды реализации) при изложении алгоритмов способны настолько затуманить идеи, что за второстепенными техническими деталями будет просто потеряна связующая нить. Ну а без понимания идей алгоритма применять его и переносить на другой язык (в другую среду) станет довольно проблематично. Вот почему в книгах, посвященных алгоритмам, нередко можно встретить изложение как на псевдоязыке, придуманном автором (вроде языка ассемблера машины MIX Д. Кнута), так и на несуществующих диалектах Паскаля. Язык Паскаль давно уже превратился в эсперанто алгоритмов, однако проблема в том, что, когда говорят «Паскаль», непонятно, о какой его разновидности идет речь. Программирование — это все же не математика. Идеи и решения здесь, как нигде, требуется проверять опытным путем. Отсюда вывод: алгоритмический язык должен быть не абстрактным, а реально существующим языком, но при этом оставаться нейтральным, стоящим особняком по отношению к другим языкам.

В настоящее время можно выделить, пожалуй, шесть главных универсальных языков программирования, которые оказывают наибольшее влияние на программную индустрию: Visual Basic, Си, Си++, Object Pascal, Java, Ада. С учетом планов реализации к осени 2001 г. ряда языков на новой платформе Microsoft под названием .NET можно упомянуть еще три: Eiffel, Оберон, C#. Каждый из них в той или иной мере можно использовать в качестве алгоритмического языка, имея в виду то, что алгоритмический язык концептуально так же отличается от языка программирования, как язык проектирования/макетирования от языка реализации. Он должен быть простым (с небольшим числом базовых понятий) и в то же время достаточно выразительным (чтобы с его помощью относительно легко можно было представить наиболее распространенные алгоритмические решения). Он должен быть консервативным (зафиксировавшим свои структуры и механизмы) и развивающимся (дающим основу для новых диалектов). Он должен также иметь лаконичное описание и быть доступным на большинстве наиболее распространенных и перспективных компьютерных платформ. Исходя из таких требований для «Студии программирования», свой выбор мы остановили на несущественно отличающихся друг от друга языках Оберон (1988) и Оберон-2 (1991), созданных автором Паскаля Никлаусом Виртом в содружестве соответственно с Юргом Гуткнехтом и Ханспетером Мёссенбёком. Важно отметить, что, несмотря на свою относительную малоизвестность, Оберон и Оберон-2 являются эффективными языками реализации и во многих отношениях не только не уступают, но и превосходят своих старших и даже более юных «коллег» — Object Pascal, Java, C#.