Евгений Зуев,
технический директор ЗАО "МедиаЛингва:Компиляторы"

Русские "плюсы"

Много лет назад, когда я работал на советском суперкомпьютере "Эльбрус" и писал программы на его базовом языке Эль-76, один из старших коллег-программистов рассказал, как он помогал своей дочери постигать школьный курс информатики. Учитель дал ей задание запрограммировать какой-то несложный алгоритм - бинарный поиск или что-то в этом роде. Домашние объяснения в итоге свелись к программе на Эль-76. Когда дочь показала эту программу учителю, тот буквально остолбенел: во-первых, от того, что, оказывается, есть средства программировать полностью по-русски и, во-вторых, насколько по-русски это наглядно и понятно!

Оставляя в стороне преимущества самого Эль-76 и стоящего "за ним" компьютера, нельзя не отметить: школьный учитель, ничего не зная об "Эльбрусе", тем не менее быстро оценил удобство самой записи на русскоязычном алгоритмическом языке.

Через некоторое время мне пришлось переходить с "Эльбрусов" на персоналки. Тут уж настоящий шок пережил я сам. Было ощущение, что меня забросили в каменный век или, что точнее, в средневековье, где, как известно, истинная роскошь парадных залов аристократических обиталищ соседствовала с примитивнейшими туалетами. Наслаждаясь великолепными цветовыми решениями инструментальных сред, я должен был заново перечитывать книги по системному программированию и вспоминать, что, оттранслировав программу, я, оказывается, не могу сразу отдать ее на выполнение: перед этим необходимо "слинковать" ее с другими модулями. Операционная система (и почему ее так назвали?) не могла сама динамически связать мою программу с необходимыми ей ресурсами (в том числе и с другими моими программами). Вдобавок, считав вроде бы готовый код с диска, ОС должна была еще "настроить" программу на реальные адреса перед тем, как выполнять ее. Тот факт, что в "Эльбрусе" код не содержал адресной информации и поэтому не нуждался ни в редактирования связей, ни в настройке, стал настолько привычным, что понадобилось время, чтобы осознать: лучшие технологические решения не обязательно самые распространенные…

Ну и, конечно, языки программирования: создавая бухгалтерскую программу на Паскале ли, на Си,- пиши `zakaz` вместо `заказ`, либо ищи в словарях, как переводится это слово на английский, вводи соответствующий идентификатор, чтобы через месяц, глядя на экран, долго вспоминать, что имелось в виду под именем `Order`: ордер, заказ или что-то еще. Пусть я даже немного лукавлю и на самом деле вполне могу работать в англоязычной терминологии. Но со своими коллегами-программистами я обсуждаю профессиональные проблемы по-русски! И когда я говорю о реализации шаблонов в нашем компиляторе Си++, рассказывая об алгоритме функции `FunctionInstantiation()`, то *как* мне в беглой русской речи произнести это название? Попробуйте! А теперь представьте, насколько было бы легче, если это имя звучало как `НастройкаФункции()`…

Дело, однако, далеко не только в языке. По-настоящему важный вопрос следует задать так: является ли разработка программ в нашей стране просто родом занятий отдельных людей или коллективов, или это *отрасль промышленности* - пусть находящейся в неразвитом (или, если угодно, в полумертвом) состоянии? И если верно второе, то следующим вопросом, который представляется очень важным, будет такой: как обучать специалистов современным подходам и методам создания программ и, в частности, как обучать их работе на современных промышленных языках программирования?

Необходимо прояснить, по крайней мере, два момента. Во-первых, термин "промышленный". У многих это слово ассоциируется с дымящими заводскими трубами, прокатными станами, километровыми сборочными конвейерами и устрашающих

размеров угольными разрезами. Если это так, то при чем здесь программирование? Однако, большинство этих образов - из уходящей эпохи. Сейчас, говоря о промышленности, имеют в виду, скорее, поезда TGW, летающие по Европе со скоростью 200 км/час и полностью управляемые компьютерами, миллионные тиражи операционной системы Windows и глобальная информационная инфраструктура, контуры которой становятся отчетливее с каждым месяцем.

Второе замечание относится непосредственно к обучению. Дело в том, что традиционно изучение программирования начинается с какого-либо простого, но "правильно" и регулярно построенного языка, например Pascal. Именно такие языки дают возможность постичь базовые понятия информатики и попросту научиться программировать. Само по себе это совершенно правильно. Однако, во всем мире наблюдается (и осознается как существенная проблема) заметный разрыв между сферой образования и промышленностью. Проще говоря, языки, ориентированные на обучение, практически не используются при создании реальных систем, и наоборот, средства разработки индустриальных систем явно непригодны для целей обучения. Не будем сейчас говорить о причинах такого разрыва и способах его преодоления; отнесемся к нему как к данности.

Таким образом, вопрос, заданный выше, можно сформулировать так: как обеспечить для начинающих программистов, которые уже освоили азы программирования, естественный и эффективный переход с *учебного* языка на *промышленный*? Попытка ответить на этот вопрос и составляет цель статьи.

Если коротко, то решение проблемы наша компания видит в создании **взаимосвязанного комплекса программных средств, учебных материалов и методик, направленных на поддержку процесса освоения языка Си++**.

Рассмотрим наиболее существенные аспекты этого направления. Первое - выбор языка. Можно по-разному относиться к языку Си++; мы сами, изучив его за несколько лет почти "до донышка" и реализовав его компилятор, вовсе не являемся его ярыми поклонниками. Громадный объем языка, несистематическое проектирование, обилие правил и исключений из них, переусложненная семантика в сочетании с тяжелым грузом архаичных подходов и решений не позволяют назвать этот язык выдающимся произведением науки программирования. Однако бессмысленно ломать по этому поводу копья, имея непреложный факт: в мировом масштабе это практически единственный инструмент (рядом с ним можно поставить только Ада 95), адекватный современным задачам проектирования, разработки и сопровождения больших и сложных программных комплексов. Системы, реализованные на Си++, составят основной объем ПО, который будет работать по крайней мере до середины будущего столетия. Именно поэтому, говоря о промышленном языке, мы имеем в виду прежде всего Си++.

Далее, необходимо уточнить, на какую категорию программистов следует ориентироваться при создании такого обучающего комплекса. Самое первое, что здесь следует сказать,- нет смысла посягать на сферу начального обучения. Си++ - очень сложный для изучения объект, и приступая к его освоению, крайне желательно уже иметь некоторый багаж знаний и практического опыта. С другой стороны, ни по одной своей характеристике Си++ явно не предназначен для обучения. Поэтому бессмысленно делать "Си++ для детского сада" или даже "Си++ для средней школы". С другой стороны, начинающие программисты, которые (в школе ли, в университете) уже освоили азы программирования, или специалисты, которые программировали на языках предыдущего поколения много лет, сейчас стоят перед непростой задачей перехода на современный индустриальный инструмент. Все они наверняка оценят наличие специальной системы, ориентированной на их потребности.

Каковы же основные черты такого комплекса? В целом он, очевидно, должен представлять собой то, что традиционно называется "системой программирования на языке Си++". Он будет включать компоненты, которые всегда присутствуют в такого рода системах,- компилятор, библиотеки, пользовательскую среду с редактором, отладчиком, менеджером проектов и т.д. Однако, ориентация на цели обучения предопределяет ряд особенностей, которые несколько отличаются от привычных.

Первое - оперативная помощь. В системе, ориентированной на обучение, удельный вес этого компонента должен быть, очевидно, существенно большим, нежели в обычных системах. Это относится как к объему помощи, так и к ее направленности. Например,

каждое диагностическое сообщение компилятора должно подробно объясняться; для всех случаев появления сообщения должны быть не формальные (как у многих известных систем), а содержательные примеры. Конструкции языка, библиотечные функции, приемы и "правильный" стиль программирования - все должно подробно объясняться, на все должны быть примеры, и все это должно быть легко доступно. Такая же обширная помощь должна быть, разумеется, и по самой системе (меню, команды и т.д.).

Второе - специальные прикладные библиотеки, ориентированные именно на освоение языка. Такие библиотеки (например, двумерная графика, простые средства создания графического пользовательского интерфейса, математические вычисления и т.п.) должны помочь начинающему программисту как освоить объектно-ориентированный подход, так и приобрести первые навыки работы с библиотечными окружениями. Такие библиотеки могли бы, помимо прочего, играть роль своеобразного "переходника" между стандартными для некоторого операционного окружения библиотеками (например, Windows API) и программой начинающего программиста. В противном случае ему неизбежно придется одновременно осваивать две очень непростые вещи - собственно язык Си++ и системные библиотеки, которые свяжут его программу с конкретной средой.

Третий аспект заключается в особенностях выполнения программ на языке Си++. Хорошо известно, что язык Си++ (как и его предшественник - Си) - незащищенный. Иными словами, семантически некорректные ситуации в них "заложены" изначально, и они не контролируются ни при компиляции, ни при исполнении. Примерами могут служить выход индекса за границы массива, работа с неинициализированными данными, обращение к несуществующему динамическому объекту и т.д. Наличие в системе развитого отладчика исправляет ситуацию лишь частично. Если же говорить о начинающем программисте, то выявить подобные ситуации и, что важнее, научиться в дальнейшем их не допускать, будет для него очень непросто. В то же время можно предположить, что быстродействие результирующей программы не является в такой системе критическим фактором, хотя бы потому, что в большинстве случаев программы будут сравнительно небольшими.

По этим причинам в системе, ориентированной на обучение, была бы оправданной не вполне традиционная схема выполнения программ на Си++. Компилятор, производя полный комплекс семантических проверок (согласно Стандарту Си++), генерировал бы не выполнимый объектный код, а некоторое промежуточное представление (ПП), которое содержит всю информацию об исходной программе. А выполнение программы могло бы быть реализовано как *интерпретация* этого промежуточного представления, например с помощью специального компонента, который можно назвать *виртуальной машиной* Си++.

Описанная схема давно и хорошо известна. Недавние технологические решения, связанные с языком Java и его виртуальной машиной, показывают жизнеспособность и достоинства этого подхода, хотя и недостатки тоже вполне очевидны. Однако сейчас не хотелось бы останавливаться на обсуждении этих вопросов, а отметить преимущества такой схемы в контексте заявленных целей. Разумеется, интерпретационная схема исполнения хороша не потому, что ее проще реализовать (сложность семантики языка Си++ такова, что сделать для него генератор кода не труднее, чем хорошую виртуальную машину). Принципиальный выигрыш заключается в том, что именно такой, полностью контролируемый режим исполнения может обеспечить *диагностический сервис*, необходимый для начинающего программиста. С помощью виртуальной машины можно выполнять такие динамические проверки, которые крайне затруднительны или вообще невозможны при выполнении сгенерированного объектного кода.

Четвертое: в такой системе необходим традиционный учебник по языку Си++. При этом было бы крайне желательно, чтобы, во-первых, такой учебник был написан людьми, хорошо понимающими особенности целевой (русскоязычной) аудитории и, во-вторых, был внутренне связан с самой программной системой. Учебник должен быть компактным (устрашающие объемы современных книг по программированию сами по себе способны отпугнуть начинающих) и в то же время объяснять все основные свойства языка. Далее, такая книга не может быть просто добросовестным описанием языка; она обязательно должна нести определенный методический заряд, помогающий читателю увидеть за непривычными конструкциями обоснованные технологические подходы и методы,

дисциплинирующий и не позволяющий ему "свалиться" в расхлябанный "хакерский" стиль программирования, столь характерный, к сожалению, для многих… Вывод из сказанного один: было бы правильно, если такой учебник был написан либо авторами самой программной системы, либо в тесном сотрудничестве с ними.

И, наконец, **последнее**: входной язык и компилятор. Есть искушение в простой логике: если система создается для обучения, то нет смысла реализовывать в компиляторе все возможности языка, тем более, что многие из них (например, шаблоны) как будто специально спроектированы так, чтобы сделать их реализацию буквально программистским подвигом. Однако если следовать такой логике, то встает вопрос: что выбросить? И тут оказывается, что если задумываться о правильном стиле программирования на Си++, о том, что называется современной парадигмой программирования, и с этой точки зрения рассматривать те или иные свойства языка, то, строго говоря, выбросить *ничего нельзя*. Например, шаблоны являются важнейшим инструментом обобщенного программирования - очень интересного и многообещающего подхода, одного из наиболее заметных достижений науки программирования последних лет; к тому же все стандартные библиотеки основаны на них. Выброси из компилятора реализацию шаблонов - и учебник по языку тут же превратится в ординарный пересказ первых изданий Страуструпа, Шилдта и многих других авторов. Откажись от поддержки исключительных ситуаций - и станет непонятно, как донести до пользователя принципы безопасного программирования.

Таким образом, единственно возможной альтернативой представляется точное следование Стандарту языка Си++ в полном объеме, сколь бы тяжелой эта задача ни казалась.

На этом пути есть и еще варианты: зачем реализовывать систему целиком? Почему не воспользоваться тем, что уже есть? Ведь технически можно, например, встроить в свою систему компилятор любого известного производителя, организовав передачу ему исходных текстов и перехват сообщений? Или, наоборот, включить свой компилятор в "фирменную" среду разработки - многие известные системы допускают подобные вещи? Возможен и такой относительно простой путь, как создание не полноценного компилятора, а препроцессора вроде известного cfront…

Однако, внимательный анализ показывает, что ни один из подобных путей не подходит. Во-первых, такая "полуфабрикатная" система будет неизбежно "привязана" к включаемой (или включающей) системе, и в случае смены версии последней ни одна фирма не даст гарантию ее совместимости с нашими компонентами. Но более существенен другой аспект: такой путь не позволит обеспечить приемлемый для целей обучения уровень локализации компилятора.

Тут мы подходим, видимо, к самому "чувствительному" вопросу из затронутых в статье. Что должна означать "локализация" или, говоря проще, русификация системы? Прежде всего, вся ее "текстовая инфраструктура" - среда разработки, оперативная помощь, учебник,- должна быть русскоязычной. Все это кажется вполне естественным и даже необходимым для системы, призванной помочь в начальном освоении Си++. (Боюсь, однако, что далеко не все согласятся даже с этим, вроде бы, очевидным, соображением…)

Далее, русскоязычным хотелось бы сделать и сам компилятор: он должен выдавать диагностические сообщения по-русски, допускать русскоязычные комментарии, идентификаторы и… служебные слова. Вот об этом - подробнее.

Из всего сказанного выше, надеюсь, ясно, что "русификация Си++" - не единственное и далеко не самое важное в описываемом проекте. На самом деле почти по всем перечисленным аспектам отсутствует сам предмет обсуждения. Язык диагностических сообщений компилятора и всей системы - вопрос реализации, и, видимо, мало кто сомневается в том, что русскоязычному программисту, как бы хорошо он ни знал английский, удобнее работать с русскими текстами.

Что касается языка комментариев и идентификаторов, то обратимся к официальному документу. Если исходить из Международного Стандарта языка Си++ (ISO/IEC 14882), который был принят в сентябре 1998 года, то в нем разрешаются произвольные символы в комментариях и *допускаются* кириллические символы в идентификаторах:

расширенное множество входных символов языка включает, кроме базового набора, также несколько национальных алфавитов, в том числе и символы кириллицы. Соответствующие подмножества непосредственно указаны в Приложении Е, которое имеет статус нормативного, то есть обязательного для любой реализации языка. Таким образом, русскоязычные идентификаторы - требование Стандарта, и если уж ему следовать, то по-другому делать просто нельзя.

В результате остается единственное спорное решение - перевод служебных слов, или точнее, допущение в языке русскоязычных вариантов служебных слов Си++. Прежде всего, нуждается в обосновании сама идея: действительно ли русскоязычные эквиваленты так уж необходимы? Объем статьи не позволяет привести все аргументы в пользу этого решения; заметим только, что уж если можно использовать кириллические идентификаторы, то текст, представляющий мешанину из английских (к тому же часто сокращенных) служебных слов и русских имен трудно назвать иначе, как смесью "французского с нижегородским"… Сказав "а", надо говорить и "б".

Разумеется, локализованная система должна, вообще говоря, обеспечивать определенную степень совместимости. Иными словами, необходимо сделать так, чтобы программа, разработанная в такой системе согласно требованиям Стандарта, корректно обрабатывалась в любой другой системе. Кроме того, нелишним будет некоторый конвертор, преобразующий альтернативные служебные слова в их прототипы.

Имеется сильное возражение формального характера. В отличие от идентификаторов, служебные слова считаются иероглифами языка, то есть неразрывными комбинациями определенных (латинских) символов с фиксированной семантикой. Изменение этих иероглифов, либо введение альтернативных, строго говоря,- нарушение Стандарта.

Все это правильно. Тем не менее, нужно высказать несколько замечаний. Во-первых, наличие русскоязычных эквивалентов служебных слов *не отменяет* оригинальных. При желании русскими аналогами можно не пользоваться и в скором времени просто забыть о такой возможности данного компилятора! Во-вторых, согласитесь, такое нарушение Стандарта носит непринципиальный характер: мы не изменяем семантику языка, не отменяем и не изменяем какое-либо его свойство, не добавляем в язык что-то свое (как, между прочим, напропалую делают и Microsoft, и Borland!) - мы просто вводим альтернативную систему обзначений, дополнительные иероглифы.

На нашей стороне будет и историческая практика. Хорошо известно (или хорошо забыто?), что первые отечественные компиляторы с Алгола-60 и Фортрана поддерживали полностью локализованные версии этих языков (автор лично программировал на них еще учась в университете). Известно, что все отечественные реализации Алгола-68 (а их было, по крайней мере, две) также включали русскоязычные версии служебных слов, что, между прочим, допускал и стандарт этого языка. В свое время осуществлялся широкомасштабный проект (к сожалению, незавершенный) создания семейства языков, основанных на Аде; это семейство, опять-таки, было полностью локализовано. Наконец, в Новосибирске был реализован (доведен до официальной приемки) компилятор PL/I для "Эльбрусов" с возможностью программировать по-русски. Не убеждает? Есть и другие примеры из совсем недавней истории.

Неужели все специалисты, участвующие в этих проектах, были неправы? Неужели сейчас ситуация с языками программирования настолько изменилась, что идея русификации выглядит нелепой? Или русский язык каким-то образом доказал свою непригодность как язык технического общения?

Наша компания в скором времени собирается предложить некоторый вариант системы программирования на Си++, созданной согласно идеям и требованиям, рассмотренным в этой статье. В частности, на web-странице www.medialingua.ru/products/tecnolo/cfront/keywords.htm выложен предлагаемый нами вариант перевода служебных слов Си++. Насколько он удачен - покажет время и ваши мнения, которые, будьте уверены, будут внимательнейшим образом рассматриваться.

В некоторых электронных форумах уже проходят дискуссии, вызванные появлением наших предложений. Эти дискуссии, а также письма, приходящие к нам, показали, что проблема, связанная с такой важной областью, как освоение современных языков

программирования, зачастую неоправданно сужается, в результате остается без внимания и выхолащивается сама идея. Так что этой статьей мы просто хотели внести ясность.