



Руслан Богатырев

## Оберон как эсперанто программирования

До недавнего времени она проходила по разряду экзотических.  
А между тем эта профессия воплощает в себе едва ли не самые  
ценные свойства человеческого разума...

А.П. Ершов о профессии программиста, 1985

Программирование является, возможно, самой важной  
новой дисциплиной постиндустриальной эры.

Н. Вирт, 2002

### Вавилонское столпотворение

Современное программирование давно уже превысило тот предел сложности, за которым разработчик теряет контроль над пониманием и надежностью создаваемых им систем. Сложные языки, громоздкий инструментарий, большое число правил и еще большее число исключений, запутанная «законодательная база», раздутая до немыслимых размеров «судебная практика», нестабильное и подчас непредсказуемое поведение операционной среды — вот что характеризует нынешние передовые системы программирования.

Если раньше можно было четко выделить язык (канон, стандарт или диалект), его конкретную реализацию (транслятор + исполняющая система) и сопутствующий инструментарий (среда разработки + библиотеки), то сейчас эти понятия перемешались, причем как в восприятии, так и чисто физически. Язык вынужден жить, пораженный метастазами коммерческих систем, развитие которых упорно идет по пути экстенсивного развития. Количество здесь в самом деле способствует переходу в новое качество, только вот до конца ли мы представляем себе глубину той пропасти, которая перед нами теперь разверзлась? И как скоро мы выйдем на понимание того, какие качественные изменения в образовании, академической науке и индустрии потребуются внести, чтобы переломить негативные тенденции?

Как известно, рост производительности процессорной техники все еще подчиняется знаменитому закону Гордона Мура, открытому в 1965 г. Закон гласит, что новые модели микросхем разрабатываются спустя примерно одинаковые периоды (18—24 мес.) после появления своих предшественников, а их емкость (число транзисторов) при этом каждый раз возрастает примерно вдвое. Те языки, которые попали в сферу интересов «большой индустрии», по всей видимости, подчиняются этому же закону (разумеется, речь идет о размере, занимаемом соответствующей системой программирования на диске после ее установки).

Если за точку отсчета взять Turbo Pascal 1.0 (осень 1983 г., 130 Кбайт), а за нынешний ориентир — Visual C++ (Visual Studio 2005 Beta, осень 2005 г., 1,5 Гбайт), то нетрудно видеть, что закон Мура действует и здесь, предсказывая, что к 2006 г. объем в 2 Гбайт станет для языков нормой. В таблице приведены данные по оптимистичной (24 мес.) и пессимистичной (18 мес.) оценкам для удвоения объема.

Табл. 1. Закон Мура в отношении систем программирования

| 24 мес. | 18 мес.      | Кбайт (тыс. байтов) |
|---------|--------------|---------------------|
| 1983    | 1983 (осень) | 130                 |
| 1985    | 1985 (весна) | 260                 |
| 1987    | 1986 (осень) | 520                 |
| 1989    | 1988 (весна) | 1.040               |
| 1991    | 1989 (осень) | 2.080               |
| 1993    | 1991 (весна) | 4.160               |
| 1995    | 1992 (осень) | 8.320               |
| 1997    | 1994 (весна) | 16.640              |
| 1999    | 1995 (осень) | 33.280              |
| 2001    | 1997 (весна) | 66.560              |
| 2003    | 1998 (осень) | 133.120             |
| 2005    | 2000 (весна) | 266.240             |
| 2007    | 2001 (осень) | 532.480             |
| 2009    | 2003 (весна) | 1.064.960           |
| 2011    | 2004 (осень) | 2.129.920           |
| 2013    | 2006 (весна) | 4.259.840           |

Реальные значения выходят за пределы оптимистичной оценки и находятся ближе к пессимистичной. Для сравнения приведем данные некоторых систем программирования.

- 1983. Turbo Pascal 1.0 — 130 Кбайт.
- 1991. Ядро Оберона (включая редактор и компилятор) — около 200 Кбайт.
- 2001. Система XDS-x86 (Modula-2/Oberon-2) — 23 Мбайт.
- 2003. Система BlackBox (Component Pascal) — 34 Мбайт.
- 2005. Visual C++ (из комплекта Visual Studio 2005 Beta) — 1,5 Гбайт.

Важно отметить, что линия Оберона (Oberon, Oberon-2, Component Pascal) не подчиняется этому применению закона Мура к системам программирования. Разумеется, дело отнюдь не только в несопоставимых трудозатратах, финансовых вложениях и игнорировании этого семейства со стороны компаний «большой индустрии». Это следствие многолетней целенаправленной работы группы профессора Вирта (Niklaus Wirth, ETH, Цюрих, р.1934) и его коллег, ставки на предельную простоту базовых элементов и правил их композиции, что автоматически ведет к повышению надежности.

Интересно, если бы классическая математика развивалась экстенсивными рыночными методами современной ИТ-индустрии, на каком уровне развития находилось бы сейчас человечество? А ведь построение языка, включая синтаксис, семантику и прагматику, — это сфера скорее фундаментальной, нежели прикладной науки.

Как справедливо заметил Никлаус Вирт, собственная сложность языка программирования распространяется на все аспекты его жизни. Именно поэтому после создания в 1970 г. Паскаля профессор Вирт сосредоточился на поиске такого компактного языка, который мог бы стать ядром практически для любой компьютерной системы вне зависимости от ее масштаба и назначения.

В 1977—1981 гг. Вирт со своими коллегами создает язык программирования Modula-2, а также силами аспирантов и студентов ETH строит полностью ориентированный на Modula-2 16-разрядный персональный компьютер Lilith с растровым дисплеем и операционной системой Medos, которая была полностью написана на Modula-2. В 1984 г. за эти работы и за создание Паскаля Вирт был представлен ассоциацией ACM к высшей награде — премии Алана Тьюринга (Alan Turing Award), которая в компьютерном научном мире эквивалентна Нобелевской премии.

Осенью 1985 г. Никлаус Вирт и Юрг Гуткнехт дают старт новому проекту — Oberon. Он был назван так в честь самого дальнего спутника Урана, мимо которого в 1986 г. пролетал американский аппарат Voyager-2. Интересно, что само имя этой «малой луны» было дано в честь короля эльфов из знаменитого произведения Вильяма Шекспира «Сон в летнюю ночь». (Все кратеры на Обероне названы по именам персонажей Шекспира: Гамлет, Антоний, Ромео, Фальстаф, Макбет, Отелло, Лир, так что можно считать, что Оберон посвящен великому английскому драматургу.) Именно язык Оберон и стал вершиной творчества Вирта.

Если сравнить труд художника и скульптора, то проектирование языков профессором Виртом ближе работе великого скульптора, отсекающего все лишнее и превращающего бесформенные образования в настоящие шедевры.

Во-первых, мы сосредоточились на действительно существенных особенностях системы. Мы пренебрегаем всем, что не оказывает существенного влияния на необходимую мощность и гибкость системы. Например, взаимодействие с пользователем в базовой системе ограничивается режимом ввода-вывода текстовой информации — никакой графики, никаких картинок и пиктограмм.

Во-вторых, мы хотели использовать истинно объектно-ориентированный язык программирования — способный обеспечить безопасность типов. В сочетании с нашей верой в то, что первый принцип является даже более обязательным для инструментальных средств, чем для самой создаваемой системы, это вынудило нас спроектировать собственный язык и сконструировать для него компилятор. Так и возник Оберон — язык, в сущности полученный из языка Modula-2 путем удаления из него «ненадежных» особенностей (таких как функции преобразования типа и варианты записи) вместе с не слишком существенными (примеры — тип-диапазон и перечислимый тип).

Наконец, мы полагали, что система должна быть гибко расширяемой — это и позволит ей стать простой, эффективной и полезной. Следовательно, в нее могут быть добавлены новые модули, которые включают в себя новые процедуры на основе вызова уже существующих. Это также означает, что в новых модулях могут быть определены новые типы данных, совместимые с существующими типами. Мы называем их расширенными типами, и они представляют собой единственное фундаментальное понятие, которое было добавлено к Modula-2.

Н. Вирт (1996)

В проекте Lilith центральным звеном триады «язык—компьютер—система» был компьютер, тогда как в проекте Oberon во главу угла была поставлена ОС Oberon. При этом запланированным побочным результатом работ стали блестящие языки Modula-2 и Оберон, наглядно доказавшие свою высокую практическую значимость.

Прелесть Оберона состояла в том, что он оказался проще Паскаля и Modula-2, при этом весьма элегантно решил проблемы мирного сосуществования разных парадигм (подходов) программирования. Отвечая на вызов времени — моду на ООП, Вирт пошел своим путем, добавив в язык не классы и объекты, а понятие проекции (расширения) комбинированного типа (RECORD).

Я бы не сказал, что распространившаяся практика ООП реализовала весь свой потенциал. Наша конечная цель — расширяющее программирование (extensible programming). Под этим я понимаю возможность конструирования таких иерархий модулей, когда каждый модуль добавляет новую функциональность в систему. Расширяющее программирование подразумевает, что добавление модуля возможно без внесения каких-либо изменений в существующие модули — не должно быть необходимости даже их перекомпилировать. Новые модули не только добавляют новые процедуры, но — что более важно — добавляют также новые (расширенные) типы данных. Мы продемонстрировали практичность и экономичность этого подхода при проектировании Oberon System...

Н. Вирт (1997)

По прошествии десятилетия 1995—2005, в течение которого появились и окрепли Delphi (Object Pascal), Java и C#, становится все более очевидно, что Оберон выходит на позиции едва ли не лучшей кандидатуры на роль единого языка, эсперанто программирования.

Казалось бы, зачем это нужно? Ответ прост: чтобы облегчить общение любителей и профессионалов, стимулировать взаимное обогащение идеями программистов из разных языковых миров, предоставить унифицированную, эталонную основу для анализа и синтеза программных решений, ибо думать на многих уровнях гораздо труднее, чем рассуждать на одном.

Конечно же, не стоит ставить утопическую задачу вытеснения всех существующих языков и замены их на один самый лучший. Эсперанто программирования — носитель ключевых идей и алгоритмов с последующей конкретизацией (привязкой) к промышленным языкам. При таком подходе одним из главных критериев для выбора эсперанто является нейтральность языка. Об этом и других критериях поговорим чуть позже, а пока обратимся к истокам классического эсперанто, единого международного языка, предложенного в XIX веке выходцем из России — доктором Людвигом Заменгофом (1859—1917), который в 1887 г. (за 100 лет до Оберона) опубликовал книгу "Lingvo internacia" под таинственным псевдонимом Doktoro Esperanto.

## Эсперанто

«Можно констатировать, — писал в 1980 г. известный французский специалист по эсперанто Клод Пирон, — что в настоящее время общество следующим образом относится к проблеме международного языка: огромное большинство не имеет никаких контактов с чудом и поэтому относится к нему скептически. Небольшое число людей "испытало на себе" это чудо, не может отрицать ощущений, даваемых им их глазами и ушами и чувствует раздражение от того, что внешний мир этого чуда не приемлет. Другая малая группа людей познакомилась с этим чудом (частично или полностью), и на основе этого у них развилось более или менее фанатичное чувство необходимости миссионерской деятельности...»

Эсперанто — искусственный язык, созданный конкретным человеком, сумевшим построить органичную основу для языкового общения. Специалисты признали его, но должного распространения он все еще не получил. Пожалуй, основной причиной крайне сдержанного отношения к классическому эсперанто в современном мире служит психологический аспект — боязнь потери национальных и культурных традиций, хранителем которых выступает тот или иной язык. При этом сам факт, что выхолащенный американский диалект английского де-факто все активнее берет на себя функции единого языка, *вытесняя* другие, никак не смущает критиков эсперанто, утверждающих, что *этом* нейтральный международный язык (т.е. эсперанто) похоронит остальные.

«Если встать на такую точку зрения, — пишет Клод Пирон, — то эсперанто будет выглядеть как агрессор, как грабитель, который хочет отнять у нас нашу языковую принадлежность и культуру, которой мы столь многим обязаны. Это какой-то бандит, который хочет разрушить результат многовекового творчества и даже каким-то образом разрушить нашу душу, которую он хочет превратить в нечто мертвое, унылое, безликое, некультурное, словом, в нечто безжизненное и тупое. Эсперанто даже более ужасен, чем какой-то грабитель; тот является по крайней мере живым существом, в то время как плановый язык лишен всех признаков живого: его сердце не бьется, его кровь не пульсирует. Это может быть только какой-то робот, какой-то чудовищный автомат, который, возможно, даже полон ревности к другим, настоящим языкам (живым или мертвым) и поэтому хочет их безжалостно разрушить».

Идею распространения эсперанто в той или иной мере разделяли многие известные ученые и писатели. Упомяну лишь три имени — Лев Толстой, Константин Циолковский, Умберто Эко.

В качестве преддверия к изучению других языков язык столь простой, столь правильный, столь богатый гласными, как эсперанто, имел бы большую ценность, в особенности для англичан. Поэтому следовало бы обучать детей сначала эсперанто, чтобы потом перейти к французскому, латинскому, немецкому и греческому.

Б. Майер, проф. латинского языка  
Кембриджского университета

«Легкость обучения его такова, — отмечал в одном из своих писем Л. Н. Толстой, — что, получив лет шесть тому назад эсперантскую грамматику, словарь и статьи, написанные на этом языке, я после не более чем двух часов занятий был в состоянии если не писать, то свободно читать на этом языке. Во всяком случае жертвы, которые принесет каждый человек нашего европейского мира, посвятив несколько времени на изучение этого языка, так незначительны, а последствия, которые могут произойти от усвоения всеми, — хотя бы только европейцами и американцами — всеми христианами, — этого языка, так огромны, что нельзя не сделать этой попытки».

Для того чтобы люди понимали друг друга, нужно или то, чтобы все языки сами собой слились в один (что если и случится когда-либо, то только через большое время), или то, чтобы знание всех языков так распространилось, чтобы не только все сочинения были переведены на все языки, но и все бы люди знали так много языков, чтобы все имели возможность на том или другом языке сообщаться друг с другом, или то, чтобы был избран всеми один язык, которому обязательно обучились бы все народы, или, наконец-то (как это предполагается волапюкистами и эсперантистами), чтобы все люди разных народностей составили бы себе один международный облегченный язык и все обучились ему. В этом состоит мысль эсперантистов. Мне кажется, что это последнее предположение самое разумное и, главное, скорее всего осуществимое.

Л.Н. Толстой

К.Э. Циолковский проблеме единого языка посвятил отдельную главу в брошюре «Образование Земли и солнечных систем» (1915). Он писал: «Как важно людям понимать друг друга! По преданию вначале люди имели один язык, но в наказание потеряли общий язык и заговорили на разных. Прекратились общее согласие и деятельность, направленная к одной цели...» В 1927 г.

Циолковский развивает свои взгляды на единый язык в работе «Общечеловеческая азбука, правописание и язык». В середине 1930-х гг. он приходит к мысли, что лучше эсперанто ничто не сможет решить поставленной задачи: «Разумеется, эсперанто самый лучший из всех искусственных языков. Несомненная простота алфавита, изумительная легкость грамматики, распространенность словаря делают его изобретателя бессмертным».

Интересна и критическая точка зрения наших великих современников. Итальянский писатель Умберто Эко, автор знаменитых романов «Имя Розы» (1980) и «Маятник Фуко» (1988), доктор философских наук, профессор семиотики старейшего в Европе Болонского Университета, почетный профессор 32 университетов мира, включая Сорбонну и Оксфорд, считает, что универсальный язык, *заменяющий другие*, — это скорее утопия.

Эта тема существовала на протяжении всей европейской истории. Утопическая, как поиск Грааля, и, следовательно, обреченная на неудачу. Но и эта неудача тоже интересует меня; хотя каждая попытка найти универсальный язык терпит неудачу, тем не менее возникает так называемый побочный эффект: язык Луллия не стал языком религиозной гармонии, но создал язык на основе комбинаторики, где уже могло существовать слово «компьютер». Язык Вилкинса тоже потерпел неудачу как универсальный язык, но в то же время он создал все современные категории естественных наук. Язык, предложенный Лейбницем, также потерпел неудачу, но стал основой для создания современной формальной логики. Так, каждая попытка создания универсального языка терпела фиаско, но, тем не менее, оставляла свой след в истории.

Умберто Эко

Гораздо более реальна и продуктивна модель со ставкой на сосуществование и развитие национальных языков с выделением общего языка. «Европа не переплавляет, подобно США, и не находит политическое единство, превосходящее всевозможные языковые различия, так, как это происходит в Новом Свете, — пишет Умберто Эко. — Цель новой Европы — движение к мультилингвизму; мы должны связывать наши надежды с многоязычной Европой... Даже если бы было решено говорить на эсперанто в европейском парламенте и в аэропортах, тем не менее многоязычие должно стать истинным единством Европы. Европе необходимо брать в качестве модели Швейцарию, а не Италию».

Как известно, в Швейцарии четыре государственных языка: немецкий, французский, итальянский, ретороманский. Германо-швейцарцы, немцы и австрийцы составляют 64% населения, франко-швейцарцы и французы — 18%, итало-швейцарцы и итальянцы — 10%, ретороманцы — 0,8%. И даже языку столь малой общности дан статус государственного! То, что Оберон родился и вырос в Швейцарии, только добавляет ему веса в деле единения интеллектуальных ресурсов человечества.

Если продолжить аналогию между миром людей и миром машин, то можно заметить, что среди естественных языков существуют два крупных семейства: индоевропейские и неиндоевропейские языки. В мире языков программирования также выделяются два больших направления: традиционные и сценарные языки. При этом среди традиционных наиболее заметны языки Си-семейства и языки Паскаль-семейства. Первым можно поставить в соответствие группу германских языков (включающих английский, немецкий, норвежский и др.), а вторым — романскую группу, появившуюся на основе латыни (испанский, итальянский, французский и др.). Тогда картина будет выглядеть так:

Табл. 2. Естественные языки и языки программирования

| Группа     | Естественный язык             | Язык программирования |
|------------|-------------------------------|-----------------------|
| германская | британский английский         | C++                   |
|            | североамериканский английский | Си                    |
|            | шотландский английский        | C#                    |
|            | немецкий                      | Java                  |
| италийская | латинский                     | Паскаль               |
| романская  | испанский                     | Delphi                |
|            | итальянский                   | Modula-2              |

Подобная аналогия формирует нешаблонное представление о противоборстве языков программирования и влиянии их друг на друга, хотя, как и любая аналогия, весьма субъективна.

Изящный Оберон в качестве эсперанто программирования имеет блестящие перспективы: нейтралитет (*neutrala lingvo*), стабильность, безупречная репутация, преемственность традиций, выразительная мощь, отображение на ведущие языки, простота изучения и восприятия, наличие проверенного годами инструментария. Что важно, это не мертвый язык. Семейство Оберон, включающее языки Оберон, Oberon-2, Component Pascal, Active Oberon, Zonnon и созданное в стенах знаменитой Высшей Политехнической школы ETH в Цюрихе, где учились Альберт Эйнштейн и Джон фон Нейман, продолжает развиваться на всем спектре ведущих платформ (Win32, Linux, .NET, Java/Eclipse) и в самых разных сферах: от мобильных телефонов и промышленных роботов до систем управления гигантскими гидросооружениями мира. Отсутствие коммерческих реализаций в лице ведущих компаний мира дает Оберону право претендовать на нейтралитет, на центральную роль, подобную той, которую долгое время играет Швейцария в банковском деле.

Практика показывает, что языку программирования в современных условиях пробиться в люди можно лишь одним из двух способов. Первый состоит в том, что его начнет активно продвигать и использовать в качестве стенобитного орудия для крушения фортификационных сооружений своих конкурентов один из лидеров рынка. Но одного этого недостаточно — IBM столько лет продвигала Smalltalk и — почти безрезультатно. Второй путь — собрать языку под свои знамена просвещенные народные массы, которые объединятся вокруг реальных открытых проектов по его развитию и совершенствованию. Сценарные языки наглядно доказали всему миру, что второй путь — не утопия.

Взгляните на лучших представителей сценарных языков (PHP, Python). Их опыт убедительно показывает, что не нужно оглядываться на сильных мира сего, а необходимо настойчиво отвоевывать место под солнцем, понимая, в чем прелесть избранного инструмента и в чем его преимущества перед другими. В этом отношении движение открытых исходных текстов (OpenSource) становится едва ли не единственной возможностью разорвать круговую поруку лидеров ИТ-индустрии.

Что же мешает Оберону стать эсперанто программирования?

Первая причина — недостаточная известность за пределами относительного узкого круга специалистов. В среде ИТ-профессионалов от языка Оберон, как правило, отмахиваются, обвиняя его в излишнем академизме. Мол, в теории все, конечно, замечательно, а вот в реальной жизни... Что же, комплекс неполноценности для языков Оберон-семейства надо потихоньку изживать. Американское космическое агентство NASA, европейский военно-промышленный комплекс (BAE Systems, DuPont Ballistic Lab), ядерная физика (проекты CERN), крупнейшие банки (Royal Bank of Canada, Swiss Re), специальный инструментарий для конструкторских бюро автомобильного гиганта BMW, средства лазерной топографии (Optech), система управления крупнейшим в мире каскадом ГЭС на Амазонке (Alstom Power), федеральная система управления дорожным движением в Швейцарии — вот лишь некоторые примеры использования Оберонов в сегодняшнем индустриальном мире.

Для оценки степени известности языка и его привлекательности посмотрим следующую таблицу. Из нее видно, что Оберон не так уж плохо известен, как это априори можно было предполагать.

Табл. 3. Привлекательность языка для изучения

| Показатели                                                  | Оберон<br>(1988) | Си<br>(1971) | C++<br>(1986) | Delphi<br>(1995) | Java<br>(1995) | C#<br>(2000) |
|-------------------------------------------------------------|------------------|--------------|---------------|------------------|----------------|--------------|
| Время жизни языка [лет]                                     | 17               | 34           | 19            | 10               | 10             | 5            |
| Известность языка [web-стр.] <sup>1)</sup>                  | 148.000          | 19.600.000   | 6.910.000     | 757.000          | 13.400.000     | 1.550.000    |
| Известность среды [web-стр.] <sup>2)</sup>                  | 648.000          | 198.000.000  | 23.600.000    | 4.820.000        | 84.600.000     | 5.680.000    |
| Показатель развития <sup>3)</sup>                           | 4,38             | 10,10        | 3,42          | 6,37             | 6,31           | 3,66         |
| Известность в сфере образования<br>[web-стр.] <sup>4)</sup> | 95.500           | 47.900.000   | 2.130.000     | 668.000          | 6.180.000      | 782.000      |
| Кол-во книг <sup>5)</sup>                                   | 739              | 65775        | 7908          | 4388             | 13699          | 2122         |
| Время обучения/освоения [лет]                               | 0,5              | 1,0          | 3,0           | 1,5              | 2,0            | 2,0          |

**Примечания.**

<sup>1)</sup> Выполнялся запрос вида <Oberon programming language>. Данные на середину августа 2005 г. Измерения проводились с помощью поисковой системы Google.

<sup>2)</sup> Выполнялся запрос вида <Oberon ~software>. Тильда используется в Google для поиска по синонимам. Данные на середину августа 2005 г. Измерения проводились с помощью поисковой системы Google.

<sup>3)</sup> Является отношением известности среды к известности языка (см. выше).

<sup>4)</sup> Выполнялся запрос вида <Oberon language ~education>. Данные на середину августа 2005 г. Измерения проводились с помощью поисковой системы Google.

<sup>5)</sup> Выполнялся запрос <Oberon programming language> в Amazon.com (раздел Books). Показывает кол-во книг в крупнейшем в мире интернет-магазине, где упоминается данный язык. Информация по состоянию на середину августа 2005 г.

В скобках после названия языка приведен год его создания.

Вторая причина критически сдержанного отношения к Оберону — страх риска, неуверенность в том, что имеет смысл вкладывать свой интеллектуальный потенциал в развитие малоизвестного языка.

Среди эмоциональных элементов, блокирующих благоприятное отношение к эсперанто и в то же время вызывающих страх перед ним, следует, по-видимому, упомянуть и страх риска. Он проявляется несколькими путями. Например, при изучении языковой проблемы в целом может возникнуть риск демонстрации того, как кто-либо с самого начала допускает ошибку в отношении языка. А поскольку узнать о своей собственной ошибке бывает очень унижительно, то поэтому человек предпочитает не подвергать себя такому риску. Далее существует опасность, что другие станут издеваться над нами. Даже если мы в частном порядке осознаем, что эсперанто не является бессмысленной фантазией, сможем ли мы набраться храбрости и публично заявить об этом в сегодняшней интеллектуальной атмосфере? Существует также опасность, что в случае, если мы поддерживаем и ободряем эсперанто, мы, согласно логике вещей, должны будем и учить его. И если кто-то уже сильно занят в своей личной жизни и профессиональной деятельности, идея снова начать учиться выглядит не очень привлекательной. Существует также еще одна опасность: если, например, мы решим начать учить эсперанто, а он не получит официального признания, то тогда наши усилия будут потрачены напрасно.

Клод Пирон (1980)

Подобно классическому эсперанто Оберон пока еще не сумел побороть стереотипы общественного сознания. Да у него, в общем, и не было такой возможности — за пределами швейцарского ЕТН язык развивали лишь единицы.

Наконец, третья причина — фундаментальная: отсутствие в программировании (большом и малом) четкого осознания того, что для развития компьютерных наук (да и программной инженерии) необходима единая языково-символьная основа, подобная символике математики и физики.

Пожалуй, лучше всех эту идею сформулировал 20 лет назад академик А.П. Ершов.

В качестве альтернативы единому языку программирования выдвигаем понятие о языковой среде, которую мы называем лексиконом программирования. Лексикон программирования — это лингвистическая система с фразовой структурой, содержащая в себе формальную нотацию для выражения всех общезначимых конструкций, употребляемых при формулировании условий задач, при синтезе и преобразовании программ.

В качестве основы лексикон содержит стандартную математическую символику алгебры теории множеств, математической логики. В дополнение к ней лексикон содержит нотацию для основных конструкций и примитивов программирования на самых разных уровнях. Смысл этих конструкций сам выражается средствами лексикона. Лексикон содержит развитую систему именования и разнообразные правила подстановки.

Чем лексикон отличается от языка программирования? Он выражает не только и не столько программы, сколько их свойства и наши суждения о них.

А.П. Ершов. Предварительные соображения  
о лексиконе программирования (1985)

Итак, лексикон программирования Ершова — это более глубокое понятие, нежели эсперанто программирования. Тем не менее на данном этапе развития этой отрасли знаний Оберон как форпост лексикона, пожалуй, лучшее приближение к идеалу. Если принять данную позицию, то интересно посмотреть, какие она открывает возможности: в чем прелесть единого языка?

### Общий язык программирования

В своей работе о лексиконе программирования (1985) Ершов писал: «Мечта об общем идеальном, или, лучше сказать, априорном, языке до сих пор не оставляет программистов. Главные вехи развития этой мечты: Алгол-60, Кобол, ПЛ/1 и, наконец, Ада. Каждый из этих языков, а также многие другие становились заметными явлениями в эволюции программирования, но в целом они лишь повысили разнообразие языковой практики программирования. Пожалуй, только Фортран и Бейсик реально сдерживают напор “вавилонского столпотворения”, но это ни у кого не вызывает удовлетворения».

С учетом реалий дня сегодняшнего список академика Ершова можно дополнить языками C++ и Java. За развитием каждого из них стоит крупная корпорация (Microsoft и IBM соответственно).

Практически все упомянутые языки (за исключением Алгола-60) достаточно сложны и, по образному выражению В.Ш. Кауфмана, исповедуют принцип сундука.

Как показывает опыт, безудержное применение принципа сундука ведет к громоздким, сложным, дорогим в реализации, обучении и использовании языкам-монстрам с тяжеловесным базисом и несбалансированными средствами развития.

В.Ш. Кауфман «Языки программирования.  
Концепции и принципы». — М.: Радио и связь, 1993

Принципу сундука (хранить множество вещей на всякий случай про запас) Вирт противопоставил принцип чемоданчика (держат только самое необходимое). При этом он наглядно показал, что такие языки-чемоданчики самодостаточны настолько, что для них можно построить с нуля силами малой группы исследователей за небольшой срок (3—5 лет) новый компьютер и новую операционную систему: компьютеры Lilith и Ceres, системы Medos и Oberon (соответственно для языков Modula-2 и Оберон).

Не исключено, что со временем в лексиконе сложится небольшое число конструкций программ, которым в силу экономии мышления будет приписана стандартная семантика. Этой семантикой может владеть каждый человек со средним образованием. Взятые вместе, эти конструкции и образуют общий язык программирования...

А.П. Ершов. Предварительные соображения  
о лексиконе программирования (1985)

Еще одним следствием работ Вирта и его коллег стало торжество принципа асимметричного ответа на проблемы компьютерной отрасли: маленький коллектив при грамотном подходе и глубоком понимании фундаментальных принципов своей науки в состоянии не только бросить вызов гигантам рынка с их огромным потенциалом и миллиардным бюджетом на исследования и разработки, но и на годы опередить их!

Интересно, как же соотносится Оберон с доминирующими языками, насколько он прост и конкурентоспособен?

Язык программирования не должен быть одной лишь математической теорией. Он должен быть практическим инструментом. Это подразумевает определенные ограничения, накладываемые на краткость формализма. Несколько языковых средств Оберона с чисто теоретической точки



зрения являются излишними. И, тем не менее, они присутствуют в языке из сугубо практических соображений, то ли для удобства самого программиста, то ли для достижения эффективной кодогенерации без использования в компиляторах сложных «оптимизирующих» алгоритмов сопоставления шаблонов... Все эти аргументы нужно иметь в виду, когда производится сравнение Оберона с иными языками. Ни язык, ни описывающий его документ не достигают идеала; но Оберон приближается к этой цели гораздо лучше своих предшественников.

Никлаус Вирт (1988)

Традиционные сравнения языков, где рассматриваются их конструкции и выявляется наличие/отсутствие тех или иных языковых механизмов, грешат излишне прямолинейным, фрагментарным подходом.

На этом фоне крайне интересным выглядит анализ, проведенный С. Свердловым, который брал за основу базовые синтаксические показатели языка. В качестве эталона взят язык с лучшими показателями (Оберон).

Общее число лексем в описании синтаксиса языка может служить обобщенной характеристикой размера этого описания. Число лексем использовать в качестве меры объема гораздо лучше, чем, скажем, число знаков в описании. В этом случае значение нашего критерия не будет зависеть от того, на каком языке (русском, английском) или какими конкретно словами названы нетерминалы — понятия языка. Число различных нетерминалов — следующая характеристика, которую мы будем вычислять. Количество используемых для описания языка понятий — несомненно важнейшее свойство, от которого зависит легкость освоения этого языка. Можно заметить, что число нетерминалов должно быть равно числу правил в описании синтаксиса, поскольку для каждого понятия обязано существовать ровно одно правило.

С. Свердлов. Арифметика синтаксиса

Табл. 4. Синтаксические метрики для ведущих языков

|                     | Оберон  | Си        | C++       | Delphi                 | Java      |
|---------------------|---------|-----------|-----------|------------------------|-----------|
| Лексем/нетерминалов | 765/62  | 917/53    | 1662/126  | 1825/180 <sup>1)</sup> | 1771/174  |
| Отношение к эталону | 1,0/1,0 | 1,20/0,85 | 2,17/2,03 | 2,39/2,90              | 2,32/2,81 |

<sup>1)</sup> Данные приведены для Object Pascal (Delphi 1.0)

Анализ международного ECMA-стандарта языка C# (ECMA 334. C# Language Specification. June 2005) показывает, что число нетерминалов в его описании равно 407. — Р.Б.

По сравнению со своими предшественниками Оберон по этим количественным показателям также выглядит весьма убедительно.

Табл. 4. Синтаксические метрики для языков Вирта

|                       | Паскаль   | Modula-2  | Оберон  |
|-----------------------|-----------|-----------|---------|
| Лексем/нетерминалов   | 1012/110  | 887/70    | 765/62  |
| Отношение к эталону   | 1,32/1,77 | 1,16/1,13 | 1,0/1,0 |
| Описание языка (стр.) | 50        | 40        | 16      |

Рассмотрим теперь качественные показатели, которые характеризуют возможности языка по представлению наиболее важных парадигм программирования. В своей работе «Научные основы доказательного программирования» А.П. Ершов выделял три вида программирования:

- 1) синтезирующее (автоматическое, автоматизированное или ручное манипулирование знанием о задаче с синтезом соответствующей программы);
- 2) сборочное (построение программы из уже существующих и корректных фрагментов);
- 3) конкретизирующее (построение специализированных программ из универсальных заготовок).

Составим таблицу парадигм, дабы отразить их роль в каждом из трех упомянутых видов, предложенных Ершовым.

Табл. 5. Парадигмы и виды программирования

| Парадигмы                          | Виды программирования |           |                  |
|------------------------------------|-----------------------|-----------|------------------|
|                                    | синтезирующее         | сборочное | конкретизирующее |
| 1. Процедурное программирование    | ✓                     | ✓         | –                |
| 2. Модульное программирование      | –                     | ✓         | –                |
| 3. АТД (абстрактные типы данных)   | ✓                     | ✓         | –                |
| 4. Расширяющее программирование    | ✓                     | ✓         | ✓                |
| 5. ООП                             | ✓                     | ✓         | ✓                |
| 6. Компонентное программирование   | ✓                     | ✓         | ✓                |
| 7. Композиционное программирование | –                     | ✓         | ✓                |
| 8. Обобщенное программирование     | –                     | –         | ✓                |
| 9. Параллельное программирование   | ✓                     | ✓         | –                |
| 10. Рефлексивное программирование  | ✓                     | –         | –                |

**Примечания.** Расширяющее программирование (возможность расширения/проекции типа, а не класса) и композиционное программирование (composability, исповедуемое языком Zonnon) — еще не устоявшиеся термины, тем не менее, здесь они учитываются из-за своей важности. Рефлексивное программирование (reflective programming) — вполне зрелый термин, используемый в качестве синонима для метапрограммирования (возможности программы оперировать во время выполнения кодом как данными, т.н. RTTI).

Табл. 6. Поддержка парадигм в ведущих языках

| Парадигмы                          | Оберон  | Си    | C++     | Delphi  | Java    | C#      |
|------------------------------------|---------|-------|---------|---------|---------|---------|
| 1. Процедурное программирование    | +       | +     | +       | +       | +       | +       |
| 2. Модульное программирование      | +       | -/+   | -/+     | +/-     | +/-     | +/-     |
| 3. АТД (абстрактные типы данных)   | +       | –     | +       | +       | +       | +       |
| 4. Расширяющее программирование    | +       | –     | +/-     | +/-     | +/-     | +/-     |
| 5. ООП                             | +/-     | –     | +       | +       | +       | +       |
| 6. Компонентное программирование   | +/-     | –     | +/-     | +/-     | +/-     | +/-     |
| 7. Композиционное программирование | -/+     | –     | –       | –       | –       | –       |
| 8. Обобщенное программирование     | –       | –     | +/-     | –       | –       | +/-     |
| 9. Параллельное программирование   | –       | –     | –       | –       | +       | +       |
| 10. Рефлексивное программирование  | +/-     | –     | +/-     | +/-     | +/-     | +       |
| РЕЙТИНГ                            | 15/16/8 | 3/4/0 | 15/14/9 | 15/14/7 | 18/18/7 | 19/18/9 |

**Примечание.** Рейтинг вычисляется с разбивкой на тройку «синтезирующее / сборочное / конкретизирующее» путем суммирования баллов поддержки соответствующей парадигмы.

+ (3 балла) поддерживается,  
 +/- (2 балла) имитируется,  
 -/+ (1 балл) частично имитируется,  
 – (0 баллов) отсутствует.

Как видно из таблицы, Оберон находится в лидирующей группе, немного уступая только C# и Java. А вот преемники Оберона (см. следующую таблицу) в богатстве своих возможностей оставляют промышленные языки далеко позади.

Табл. 7. Поддержка парадигм в языках Оберон-семейства

| Парадигмы                          | Оберон  | Oberon-2 | Component Pascal | Active Oberon | Zonnon   |
|------------------------------------|---------|----------|------------------|---------------|----------|
| 1. Процедурное программирование    | +       | +        | +                | +             | +        |
| 2. Модульное программирование      | +       | +        | +                | +             | +        |
| 3. АТД (абстрактные типы данных)   | +       | +        | +                | +             | +        |
| 4. Расширяющее программирование    | +       | +        | +                | +             | +        |
| 5. ООП                             | +/-     | +        | +                | +             | +        |
| 6. Компонентное программирование   | +/-     | +/-      | +                | +             | +        |
| 7. Композиционное программирование | -/+     | -/+      | -/+              | -/+           | +        |
| 8. Обобщенное программирование     | –       | –        | –                | –             | +/-      |
| 9. Параллельное программирование   | –       | –        | –                | +             | +        |
| 10. Рефлексивное программирование  | +/-     | +/-      | +/-              | +/-           | +        |
| РЕЙТИНГ                            | 15/16/8 | 16/17/9  | 17/18/10         | 20/22/10      | 21/24/14 |

Вернемся к ключевой статье Ершова о лексиконе программирования.

Давайте вообразим, как выглядит программирование в лексиконе. Синтезирующее программирование проходит полностью в лексиконе вплоть до получения алгоритма нужной степени детализации. Если это заготовка (модуль или универсальный алгоритм), то она в таком виде и остается. Если программа должна быть передана машине, то она подвергается особой процедуре лингвизации (фортранизации, паскализации, алголизации и т.п.). Лингвизация — это перековка программы, которая, сохраняя ее правильность, придает ей стилистические особенности рабочего алгоритмического языка, после чего прямая транслитерация превращает программу в текст на этом языке, который выполняется или пополняет рабочую библиотеку. Лингвизация может быть творческим процессом, выполняемым специалистом по данному рабочему языку...

Программа, составленная сразу на рабочем языке, получает «права гражданства» только после полного аннотирования на лексиконе. Эта аннотация является свидетельством ее правильности, ее постоянной тенью и позволяет с сохранением функции и структуры транслировать ее в случае необходимости назад в лексикон...

А.П. Ершов. Предварительные соображения  
о лексиконе программирования (1985)

Итак, лингвизация — естественный путь отображения программных текстов на Обероне (эсперанто) на конкретный целевой язык. Для представления алгоритмов и настройки их на определенный язык роль такого эсперанто переоценить трудно.

Лексикону учат в вузе раньше, чем какому бы то ни было рабочему языку (игрушечный практикум не в счет). Доказательное программирование является основой курса программирования, подкрепленного каторжным тренингом в духе лучших задачников по математическому анализу. Программиста бьют по рукам, если он посмеет написать оператор цикла, не найдя перед этим его инварианта.

На лексиконе должно быть написано объемистое руководство программиста, содержащее теории наиболее ходовых предметных областей, семантику фундаментальных конструкций программирования, логические и алгебраические законы, базовые трансформации, связывающие основные модели вычислений, перечень стилей основных рабочих алгоритмических языков и правила транслитерации в них.

На лексиконе должно быть переписано «Искусство программирования» Д. Кнута в виде свода фундаментальных алгоритмов с полным сертификатом их правильности.

А.П. Ершов. Предварительные соображения  
о лексиконе программирования (1985)

Благодаря своей простоте и продуманности Оберон легко осваивается пользователями самого разного уровня профессиональной подготовки в области программирования — школьниками, студентами, учеными, инженерами. Именно эта его отличительная черта положена в основу научно-образовательного проекта «Информатика-21» (<http://www.inr.ac.ru/~info21>) — создания единой языковой платформы систематического преподавания основ программирования в XXI в.

С точки зрения восприятия (вспомним слова Л.Н. Толстого об эсперанто!) Оберон практически вне конкуренции, читать исходные тексты на нем может человек, знакомый с любым диалектом Паскаля даже поверхностно. При этом Оберон и его преемники остаются одними из лучших языков для исследовательского программирования, для профессионалов высшей категории.