Сергей Свердлов

Маленький большой язык Оберон

Источник: PC Week/RE, 1997, №35.

Несколько лет назад мне в руки попал номер журнала "Юный натуралист", на обложке которого красовалась усатая и бородатая собачья физиономия. Статья об этой симпатяге называлась "Маленькая большая собака". Ключевой тезис той статьи, посвященной цвергшнауцеру (таково название породы), состоял в том, что при своих миниатюрных размерах — чуть больше кошки — такой пес обладает всеми положительными качествами "настоящих" больших собак: умен, смел, уравновешен. "Цверг" совершенно лишен каких-либо признаков вырождения, присущих большинству карликовых пород. В общем, минимальная настоящая собака. При этом он весьма неприхотлив, отличный сторож. Я был совершенно покорен и решил, что если заведу собаку, то непременно такую.

Подобные чувства я испытал и при первом знакомстве с языком Оберон. Он поразительно прост, но тем не менее содержит все необходимые средства и структурного, и объектно-ориентированного, и модульно-компонентного программирования. И никаких, заметьте, признаков вырождения! Оберон можно по праву считать минимальным универсальным языком программирования высокого уровня.

Большая малая планета Ceres

Оберон — результат тридцатилетней эволюции линии языков, берущей начало от Алгола-60 . Создан язык профессором Швейцарского федерального технологического университета (ETH), знаменитым Никлаусом Виртом (Niklaus Wirth), автором Паскаля и Модулы-2.

В 1985 г. в ETH началась работа над проектом операционной системы для однопользовательской рабочей станции Ceres — компьютера, базирующегося на RISC- процессоре семейства NS32000. Ceres (по-русски — Церера) — имя древнеримской богини земледелия и плодородия и одновременно название первого по времени открытия (1801 г.) и самого большого по размерам астероида. Выбрав для рабочей станции название малой планеты, авторы проекта, несомненно, хотели подчеркнуть необходимость ограничить сложность системы, сохранить ее обозримой. Вирт всегда был сторонником простых и компактных решений, надежных и эффективных.

Система была задумана как набор раздельно компилируемых модулей-компонентов с тщательно проработанными программными интерфейсами. А разработка приложений в такой системе должна сводиться к расширению этого основного набора.

Первоначально для создания системы планировалось использовать язык Модула-2, который имеет эффективную поддержку модульной разработки. Но по мере продвижения проекта становилось все более очевидно, что кроме расширяемости системы в процедурном смысле необходима и поддержка расширяемости типов данных. В Модуле же, как известно, не предусмотрено определение одного типа данных как расширения уже существующего. Эти потребности стимулировали дополнить язык средствами расширения типов.

Кроме того, в новой системе было решено использовать для управления памятью механизм сборки мусора. В этом случае программист освобождается от нетривиальной и чреватой ошибками работы по правильному освобождению блоков динамической памяти. За него ее выполняет специальная подсистема — сборщик мусора, который начинает работать, когда при очередном запросе блока динамической памяти ее не хватает. "Мусорщик" пытается обнаружить в памяти "брошенные" блоки, т. е. такие, на которые нет ссылок из загруженных в данный момент модулей, и возвращает эти блоки в свободную для распределения "кучу". Все, что нужно сделать программисту, чтобы созданная им динамическая структура (сколь угодно сложно организованный список, дерево и т. п.) была утилизирована, так это присвоить указателю на эту структуру значение NIL. При необходимости сборщик мусора может вызываться из программы и явно.

При реализации механизма сборка мусора необходимо обеспечить мусорщика во время выполнения программы необходимой информацией как о блоках распределенной памяти, так и о размещении в памяти указателей, которые на эти блоки могут ссылаться. Наличие в языке Модула-2 записей с вариантами затрудняло решение этой задачи. Средства расширения типов, которые должны были дополнить язык, делали записи с вариантами лишними, и от них решено было избавиться. Но как только вы ограничиваете язык, что-либо удаляя из него, теряется совместимость с прежними версиями. Старые программы не будут работать в новой системе.

Появление нового языка стало неизбежным. Новый язык именно тем и отличается от модернизированной версии старого, что наряду с добавлениями здесь сделаны изъятия. Вы не задумывались, почему новые языковые предложения Н. Вирта не называются, скажем, Паскаль-5, Паскаль++ или, к примеру, Супер Паскаль ? Да потому, что классик находит в себе смелость отказаться от не оправдавших себя или избыточных элементов, наращивая мощь языка без увеличения, а то и со снижением его сложности.

Оберон — это Паскаль сегодня

Описание Оберона было опубликовано Н. Виртом в 1988 г. Больше всего язык похож на своего непосредственного предшественника — Модулу-2. Отличия же от Паскаля таковы, что освоение Оберона не составит труда для программистов, знающих Паскаль. Н. Вирт подчеркивает, что Оберон получен путем изъятия из Модулы многого и добавлением лишь некоторых усовершенствований. Про Паскаль говорили, что это расширенный вариант суженного Алгола. Теперь этот подход применен к Модуле-2. Вот список потерь и приобретений.

Из Модулы-2 удалены:

1. Записи с вариантами.
2. Непрозрачный (скрытый) экспорт типов.
3. Перечислимые типы.
4. Ограниченные типы (диапазоны).
5. Тип CARDINAL.
6. Указатели не на записи и массивы.
7. Массивы с нецелочисленными индексами и отличной от нуля нижней границей.
8. Локальные модули.
9. Неквалифицированный именем модуля импорт идентификаторов.
10. Модули определений, главный модуль и понятие главной программы.
11. Прежняя форма оператора WITH.
12. Оператор FOR.
13. Типы ADDRESS и WORD (заменены типом BYTE) и адресная арифметика; преобразование типов, обозначаемое идентификатором типа.
14. Средства параллельного программирования.

Новые возможности, появившиеся в Обероне.

1. Расширение типов. Проверка и охрана типа.
2. Поглощение типов.
3. Многомерные открытые массивы.

Как вам нравится список изъятий? Ничего не жаль? Я, например, был просто ошарашен, впервые узнав, что из языка убрали цикл FOR. Но, поразмыслив, увидел в этом пользу для себя. Дело в том, что при обучении программированию (тем более при переучивании) одним из трудных моментов является приобретение навыков применения циклов с пред- и постусловием. Юных самоучек, мышление которых бывает почти необратимо испорчено употреблением Бейсика , с огромным трудом удается убедить в том, что не все циклы — это циклы с заранее известным числом повторений. А тут — красота, ни цикла FOR, ни тем более GOTO совсем нет, волей-неволей научишься применять WHILE, REPEAT и LOOP!

Некоторые другие сокращения в языке также могут вызывать вопросы. Так, недавно в телеконференции Usenet comp.lang.oberon прошла небольшая дискуссия о перечислимых типах, которых нет в Обероне.

Мне представляется, что аскетизм языка Оберон является его исключительно полезной чертой, особенно если язык используется для изучения программирования. Сочетание простоты, строгости и неизбыточности предоставляет начинающему программисту великолепную возможность, не заблудившись в дебрях, выработать хороший стиль, освоив при этом и структурное, и объектно-ориентированное, и модульно-компонентное программирование.

Средства модульно-компонентного и объектно-ориентированного программирования — суть принципиальных отличий Оберона от языков-предшественников.

Усовершенствована по сравнению с Модулой-2 структура программы. Вернее, понятия программы, как таковой, в Обероне вообще нет. Все, с чем вы имеете дело, — это совокупность модулей-компонентов, которые загружаются в память динамически. Инициировать выполнение можно вызовом команды, в качестве которой рассматривается любая экспортированная процедура без параметров. Кстати, экспорт в Обероне оформляется исключительно изящно. Достаточно после имени экспортируемого объекта (процедуры, константы, переменной, типа) поставить звездочку (*). При этом программисту не нужно вручную выписывать спецификацию модуля, рискуя внести несоответствие между спецификацией и реализацией. Перечень экспортированного — интерфейс модуля создается автоматически.

Объектно-ориентированным языком делают Оберон средства расширения типов. При конструировании объектной модели языка Вирт уделил первоочередное внимание не внесению в язык модной терминологии, а поддержке реальных программистских потребностей. Слова object в Обероне вообще нет; объект — это просто расширяемая запись. Реальной же является потребность в организации гетерогенных (полиморфных, состоящих из разнотипных элементов-объектов с индивидуальным поведением) динамических структур данных. При этом важна не только сама возможность построения таких структур, но и наличие адекватных механизмов, позволяющих оперировать их элементами, используя легальные и безопасные средства. В Обероне такими средствами являются проверка и охрана типа. Указатели и параметры-переменные типа "запись" могут иметь как статический (определенный при описании), так и динамический тип (соответствующий типу того объекта, на который в данный момент фактически ссылается указатель). Проверка типа позволяет узнать динамический тип объекта, а охрана — обратиться к фактически имеющимся полям, имея твердую гарантию их существования. В свое время, не обнаружив таких возможностей в Турбо-Паскале версии 6.0 (они появились лишь в 8-й версии компилятора фирмы Borland), я был сильно удивлен и долго не мог в это поверить. Как же так? Имея указатель на объект, нельзя полноценно оперировать этим объектом? Тогда в моей программе появились странные конструкции, когда объект копирует сам себя и встраивает свою же копию в список рядом с собою. Кому-то такие выверты могут казаться изящными возможностями объектной технологии, мне же необходимость подобных ухищрений представляется противоестественной.

Одна из основных черт Оберона — строгость. Строгий контроль соответствия типов, унаследованный от Паскаля и Модулы-2, строгие правила экспорта и импорта, строгий синтаксис обращения к полям записей. Соображения безопасности и надежности всегда были на одном из первых мест в языковом творчестве Никлауса Вирта. После знакомства с Обероном я не раз ловил себя на том, что, используя этот язык, я не сделал бы в своих программах тех ошибок, которые делал раньше. В то же время Оберон допускает гибкость при обращении с данными. Это относится к уже обсуждавшемуся расширению типов, а также к простой и красивой концепции поглощения типов. Повсеместный переход к тридцатидвухразрядным архитектурам определяет большое разнообразие типов числовых данных, встраиваемых в языки программирования, что делает неудобным практически полный запрет Модулы-2 на присваивание неидентичных типов.

Числовые типы Оберона образуют иерархию:

```
SHORTINT <= INTEGER <= LONGINT <= REAL <= LONGREAL.
```

В этой цепочке значения "меньшего" типа могут быть присвоены переменным "большего" типа.

"Никто не обнимет необъятного"

Одним из главных впечатлений, полученных мною при первом знакомстве с языком Оберон , было ощущение его обозримости ("объятности", как сказал бы Козьма Прутков). После двух прочтений официального описания (21 страница) на неродном английском у меня сформировалось твердое убеждение, что я уже знаю этот язык (Оберон, не английский). Некоторые нюансы расширения типов требовалось еще понять, но в целом все было достаточно ясно. Такое свойство языка невозможно переоценить. Ничего подобного я не испытываю даже в

отношении языка Паскаль в версиях фирмы Borland, которые использую для разработки программ и в преподавании уже почти десять лет. Совсем недавно, например, обнаружил, что слово export является зарезервированным и его нельзя использовать как идентификатор. Ситуация с Паскалем становится все больше похожа на ситуацию с языком ПЛ/1, про который было известно, что его невозможно знать целиком, а можно лишь обустроиться в своем уголке, взяв туда то, что нужно для решения конкретной задачи.

Вирт взял эпиграфом к описанию Оберона высказывание Альберта Эйнштейна: "Make it as simple as possible, but not simpler" (Делай как можно проще, но не проще, чем нужно). Однако молодым соратникам Вирта аскетизм Учителя, наверное, казался чрезмерным. Почти сразу после появления языка возникли предложения по его "улучшению". И, конечно же, все они сводились к расширениям языка. В 1989 г. вышла статья Ханспетера Мессенбека (Hanspeter Mossenbock) и Йозефа Темпла (Josef Templ) "Object Oberon — a Modest Object-Oriented Language" ("Объектный Оберон — скромный объектно- ориентированный язык"). Почитая скромность добродетелью, авторы не смогли избежать соблазна вставить модное словечко "объект" и в название языка и дважды в название статьи. Двумя годами позже Мессенбек опубликовал в ETH сообщение о языке Оберон-2. Основным поводом к "усовершенствованиям" Оберона служило отсутствие в языке того, что принято называть виртуальными методами. В языке Вирта индивидуальное поведение объектов можно обеспечить, используя в записях поля процедурного типа. Такое поле-процедура выполняет функции обработчика сообщений. Сами сообщения, являясь расширяемыми записями, индивидуальны для каждого типа объекта. Несомненное преимущество этого подхода — отказ от включения в язык еще одного механизма, в значительной степени дублирующего уже имеющиеся возможности, но усложняющего и язык, и компилятор. Но все же молодой напор взял верх. Вирт поставил свое имя на описании расширенного Оберона, за которым закрепилось название Оберон-2. Эпиграфа Эйнштейна на этом документе уже нет. Авторами описания языка Оберон-2 являются Х. Мёссенбёк и Н. Вирт. Сейчас Оберон-2 считается фактическим стандартом языка.

Основные нововведения Оберона-2 — это связанные с типом процедуры. Знатокам Object Pascal, C++ и Java они известны под названием виртуальных методов. Авторы Оберона-2 не стали вводить новых терминов, а обошлись уже имеющимися. Довольно остроумно выстроен синтаксис описаний связанных процедур. Не потребовалось никаких новых служебных слов, а формальный параметр-приемник, обозначающий экземпляр объекта внутри такой процедуры, описывается явно. Описания связанных процедур располагаются отдельно от описания типа запись, с которым они связаны. Связь же устанавливается по типу параметра-приемника.

Кроме связанных процедур в язык внесены еще некоторые усовершенствования. Предусмотрен экспорт только для чтения. Если после имени в описании переменной или поля записи поставить знак "—" вместо "*", то это имя экспортируется, но соответствующая переменная или поле не могут быть изменены вне экспортирующего их модуля. Согласитесь, что это гораздо выразительнее и проще, чем употребление public, private и protected.

Другие изменения — расширение применения открытых массивов, которые теперь могут использоваться не только как формальные параметры, но и в качестве базового типа указателей, расширение оператора WITH и, наконец... как возвращение в язык оператора FOR!

Принятые расширения хорошо продуманы, но, честное слово, без них можно было бы обойтись. Кроме того, при тщательном рассмотрении в этих нововведениях можно обнаружить ряд коллизий, от которых оригинальный Оберон был избавлен. Вот маленькая деталь, иллюстрирующая отношение самого Н. Вирта к этим расширениям. На ftp-сервере ETH можно найти исходный текст Оберон — системы для компьютера Ceres. Компилятор и ряд других модулей системы написаны лично Н. Виртом. Последние изменения в текст компилятора внесены Виртом 14 декабря 1993 г., намного позже появления Оберона-2, датой рождения которого считается 1992 г. В этом компиляторе нет никаких следов связанных процедур, в то время как цикл FOR реализован.

Ну а в завершение этого раздела позволю себе замечание личного свойства. Помните цвергшнауцера? Маленькая такая собачка. Так вот, когда дочь все же уговорила меня завести собаку, я приобрел... миттельшнауцера. Это то же самое, только покрупнее. Не смог ограничиться малым, теперь вот гуляю с ним. Но повторю вслед за К. Прутковым: "Плюнь тому в глаза, кто скажет, что можно обнять необъятное".

"Дубовые требования"

Летом 1993 г. в английском городке Кройдоне в отеле "Дубовый" собралось около 30 разработчиков компиляторов и прикладных программистов, чтобы согласовать единые

требования к реализациям Оберона-2. Среди участников встречи были и двое наших соотечественников. Результатом совещания стал документ, получивший название "Дубовые требования" (Oakwood Guidelines). Интересное совпадение: самый модный ныне язык Java назывался вначале Oak (Дуб).

Авторы "Дубовых требований" понимали, что стремление разработчиков трансляторов "улучшить" Оберон необходимо ввести в цивилизованное русло, чтобы обеспечить насколько возможно совместимость различных реализаций языка. При этом они ссылались на негативный опыт стандартизации Модулы-2, когда процесс развивался спонтанно.

Одна из основных черт Оберона — строгость. Строгий контроль соответствия типов, унаследованный от Паскаля и Модулы-2 , строгие правила экспорта и импорта, строгий синтаксис обращения к полям записей положений описания языка существовала. Кроме того, средства ввода-вывода не являются частью собственно языка и соглашение о составе и спецификации стандартных библиотек позволило бы значительно улучшить совместимость разных реализаций. Попытка решить оба эти вопроса сделана в "Дубовых требованиях". Но что вызывает удивление, так это внимание, которое авторы "Требований" уделили расширениям языка. Признавая, что единственным официальным документом, определяющим Оберон-2, по-прежнему является описание, публикуемое ETH, создатели "Требований" оправдывают свое большое внимание к расширениям тем, что эти расширения не предлагаются, а определены на случай. Так что если кто-то захочет язык расширить (комплексными числами, к примеру), то уж пусть расширяет как сказано. Противоречивая позиция. Гораздо симпатичней выглядит подход, принятый в отношении языка Ада, когда никакие расширения в принципе не допускаются. Вообще, документ производит неоднозначное впечатление. Он довольно многословен. Трактовки некоторых вопросов не привносят большей ясности по сравнению с описанием языка, а вот расширениям уделяется непропорционально большое внимание. Создается впечатление, что те "улучшения", которые молодым энтузиастам не удалось включить в язык, попали в "Дубовые требования". А подписи Н. Вирта под этим документом нет.

Оберон — спутник Урана. Или Марса?

Язык Оберон носит имя одного из спутников планеты Уран. К выбору такого названия Вирта подтолкнуло событие, произошедшее в то время, когда он работал над проектом языка и операционной системы. В январе 1986 г. американский космический аппарат "Вояджер-2", пролетая мимо Урана, сделал снимки планеты и ее спутников. Был сфотографирован и Оберон. На Вирта сильное впечатление произвела филигранная точность эксперимента, в ходе которого немалую роль сыграл бортовой компьютер "Вояджера". И это при том, что до сближения с Ураном аппарат находился в полете почти 10 лет. К тому же спутник Оберон был открыт англичанином Уильямом Гершелем в 1787 г., т. е. ровно за двести лет до описываемых событий. Возможно, на выбор названия повлияло и созвучие со словом "объект". А еще в немецком языке (родном языке Н. Вирта) есть слова "oberste" — высший, верховный и "Oberst" — полковник. Последнее вызывает некоторые ассоциации.

Военные, особенно американские, внесли немалый вклад в развитие языков программирования. Разработчик первого компилятора Грейс Хоппер была морским офицером. Пентагоновским комитетом разработан язык Кобол. И, конечно, крупнейшим и самым известным военным проектом в области языков программирования является создание языка Ада и внедрение его в качестве стандарта для разработки систем военного назначения в США.

Я не располагаю достоверными сведениями на этот счет, но, кажется, в восьмидесятых годах и у нас в стране были попытки использовать язык Ада для аналогичных целей. Однако Ада — большой и сложный язык. Создание компилятора Ады — нелегкая задача. Разработка компилятора, а вернее, компиляторов и кросс-компиляторов, их тестирование и сопровождение требует немалых затрат. Вероятно, отсутствие необходимых ресурсов, да в известной мере и деградация в те годы отечественной программной индустрии, поглощенной адаптацией программ для ЕС и СМ ЭВМ, не позволили решить столь масштабную задачу. Как свидетельствуют специалисты, участвовавшие в процессе создания ПО для встроенных систем военного назначения, их разработка велась часто не то что не на языке высокого уровня, но даже не на ассемблере, а в двоичных кодах. При этом шутили: "Языки высокого уровня нужны плохим программистам, а программисты хорошие пишут в машинных кодах".

Наверное, не осилить нашему ВПК внедрение языка, подобного Аде, и сейчас. Так, может, попробовать в этой роли Оберон ? Помните, какие страсти кипели в свое время вокруг СОИ — стратегической оборонной инициативы — программы создания в США космической системы противоракетной обороны. Тогда из уст академика В. Гольданского, который был советником

Горбачева в этих делах, мы услышали слова "асимметричный ответ". Наши придумали что-то более дешевое, но, по их словам, очень эффективное. Не сможет ли Оберон стать "асимметричным ответом" Аде? Предпосылок для этого масса.

Во-первых, Оберон много проще Ады. Разработка компилятора для него — задача на порядок более простая, чем для Ады. Уже упоминавшийся мной компилятор Оберона, написанный Виртом, — это программа (на Обероне, конечно) размером 3962 строки. Простота компилятора важна не только на этапе его первоначальной реализации, но и при последующей адаптации к различным, в том числе нестандартным, архитектурам. Малые затраты на модификацию компилятора (или кросс-компилятора) для его настройки на генерацию другого машинного кода позволят использовать язык высокого уровня даже тогда, когда речь идет о сугубо специализированных системах, основанных на специфической аппаратной базе, когда создание системы программирования для более сложного языка было бы не оправданно.

Во-вторых, Оберон — язык надежного программирования. Надежность и строгий контроль всегда были одним из основных критериев в языковом творчестве Н. Вирта. Встроенные в Оберон механизмы контроля отшлифованы тридцатилетней эволюцией и рукой Мастера: строгий контроль типов, полный отказ от неявного употребления имен, охраняемый доступ к полям полиморфных объектов, строго регламентированный экспорт. Нужно ли говорить, что требование надежности является ключевым для обсуждаемой сферы применения.

Большая экономия может быть получена за счет минимизации затрат на обучение программистов. Никакого переобучения, по сути, и не требуется. Позиции Паскаля в нашей стране традиционно сильны, а Оберон — его прямой потомок. К тому же этот язык настолько прост и обозрим, используемая в нем нотация так естественна, традиционна и прозрачна, а положенные в его основу концепции так ясны, что любой имеющий представление о программировании без труда освоит Оберон.

В пользу Оберона говорит и стабильность его спецификации. Документ, определяющий современное состояние языка Оберон, датируется 1990 годом. В спецификацию Оберона-2 с 1993 по март 1995 г. внесено всего 10 мелких поправок. Все они умещаются на одной странице. Это свидетельство зрелости языка.

Потребность в едином языке для военных, более простом, чем Ада, подтверждается и тем, что в США наряду с Адой все шире используется Джовиал (см. PC Week/RE, N6 /97, с. 47) — старый, но достаточно простой язык, ровесник и родственник Алгола-60. Связке Джовиал —Ада предвещают большое будущее в военной сфере. Но зачем какие-то связки, если есть язык такой же систематичный, мощный и надежный, как Ада, такой же простой и эффективный, как Алгол или Джовиал, и такой же современный как... Оберон!

Оберон в Интернет и в мире

Нужно прямо сказать, что сегодня Оберон не входит в число самых популярных языков. В те годы, когда он родился, а тем более сейчас, основным козырем в конкуренции программных систем и языков программирования стали деньги. Очень большие деньги. Основные игроки этого рынка тогда делали ставки на Си++, Бейсик и отчасти на Паскаль. Сейчас затеяна игра вокруг языка Java. Речь при этом, заметьте, идет не о стремлении сэкономить, а о желании выудить деньги у потребителя. В такой ситуации нестабильность спецификаций, например, с позиции поставщиков, перестает быть недостатком, а превращается в достоинство. Примеры того — Borland Pascal и Visual Basic. Оберон же — стабильный и "дешевый" язык. В азартной игре на деньги с частой сменой версий ему участвовать трудно. Со времени появления Оберона прошло меньше десяти лет. Это достаточно малый срок для языка, распространяющегося "естественным" путем. Вспомните, что вы слышали о Паскале в 80-м году, когда ему было десять.

Первые реализации Оберона появились в ETH в конце восьмидесятых. После оригинальной Оберон-системы для компьютера Ceres были созданы ее модификации. Сейчас они доступны на ftp-сервере ETH под общим названием ETH-Oberon system V4 для следующих платформ: Amiga, DECStation, HP700, Linux, MacII, PowerMac, RS6000, SPARC, SiliconGraphics, Windows. Впечатляющий список, не правда ли?

Традиционным и эффективным способом быстрой реализации нового языка является создание конверторов с него на другие языки. Несколько таких систем разработаны и для Оберона . Они, как правило, предусматривают преобразование программы на Обероне в программу на Си. Одна из таких систем — Ofront — транслирует в Си и реализована для AIX, HPUX, IRIX 5, Linux, SunOS 4, SunOS 5. Недавно в телеконференции comp.lang.oberon появились сведения о переносе таким способом Оберона на компьютер Be с его операционной системой Be OS. С создания транслятора Оберона (и Модулы-2) в ANSI Си начинала свою работу с Обероном и новосибирская фирма

xTech. Сейчас в числе ее Оберон-продуктов под общим названием XDS трансляторы в Си и Си++, компиляторы Оберона в родной код для нескольких платформ. Интересен список покупателей систем XDS. Среди них NASA, Оксфордский университет, Siemens, Агентство оборонных исследований (Великобритания). Как видите, Оберон используют те, кто, несомненно, умеет ценить надежность и экономичность. Оптимизирующие компиляторы XDS генерируют очень эффективный код. Так, на платформе Windows 95 тест Dhrystone дает для XDS на 40% лучший результат по сравнению с Microsoft Visual C и на 14% лучше, чем Watcom C.

Пожалуй, самая известная Оберон-система — Oberon/F, недавно переименованная в BlackBox Component Builder. Разработка Oberon/F начата в 1992 г. цюрихской компанией Oberon microsystems, Inc., тесно связанной с ETH. В отличие от систем ETH, которые обладают оригинальным и не очень привычным пользовательским интерфейсом и являются почти самостоятельными операционными системами, Oberon/F поддерживает традиционный для Windows и Macintosh интерфейс.

Систем программирования для Оберона создано уже много, и перечислять их здесь было бы слишком долго. На Web-сайте студента Тель-Авивского университета (по совместительству сетевого администратора) Гая Ладена (Guy Laden), который собрал массу сведений, относящихся к Оберону, насчитывается 23 ссылки на различные Оберон-компиляторы. Туда я и направляю любознательного читателя. Там, на Oberon Reference Site, вы сможете найти и русский перевод официального описания языка Оберон-2 .

Анализ информации из Интернет позволяет судить о распространенности языка Оберон в современном мире. Всего активней его используют в европейских университетах. Это университеты Швейцарии (Цюрих, Базель), Австрии (Линц, Инсбрук), Германии (Гейдельберг, Мюнхен, Оснабрюк, Ульм, Карлсруэ, Бонн, Штутгарт, Кобленц, Берлин, Ганновер, Дармштадт, Аахен), Голландии (Антверпен, Эйндховен, Дельфт), Великобритании (Лондон, университет графства Кент). Причислим сюда российские университеты (Новосибирск, Вологда). Понять масштабы использования непросто, но и на других континентах Оберон известен. Например, в Массачусетском технологическом институте, Стэнфордском университете, университетах Калифорнии, Техаса, Алабамы, Джорджии, Нью-Йорка, Вашингтона и Чикаго (США), университете Оттавы (Канада), в техническом университете Квинсленда (Австралия), университетах Сан-Паулу (Бразилия) и Гонконга (простите, особого района Сянган, КНР). Интерес к языку Оберон проявляют в компаниях Boeing, Bosch, Siemens, Alcatel, Motorola, DEC, Apple, Sun, в Университете Федеральных вооруженных сил (Мюнхен).

Питер, ты прав!

Оберон молод, но еще моложе язык Ява, которому уделяется сейчас беспрецедентно большое внимание. Можно найти общие черты у двух этих языков. Существенны и различия. Сравнивать детально Оберон и Яву здесь, наверное, неуместно. Замечу только, что Ява считается простым языком, в то время как его спецификация (James Gosling, Bill Joy, Guy Steele. "The Java Language Specification") — это книга почти в 800 (восемьсот!) страниц. Справедливости ради надо сказать, что сам стиль этого описания очень детальный. Семантика арифметических операций, например, определена очень подробно со скрупулезным обсуждением всех мыслимых особых случаев. Кроме того, на упомянутых 800 страницах помещено описание не только самого языка, но и стандартных библиотек.

В стандартизации библиотек и среды выполнения (виртуальной машины), возможно, и состоит главное преимущество технологии Java . Что касается самого языка, то здесь существуют разные мнения. В ситуации с Обероном, мне кажется, все немного наоборот. Сам язык очень привлекателен, а вот концепция Оберон-среды, которой ее разработчики придают большое значение, достаточно специфична. Беден и рекомендованный "Дубовыми требованиями" набор стандартных библиотек.

Но достоинства двух технологий могут быть объединены. Около года назад обозреватель PC Week Питер Коффи высказал мысль о том, что Java, а точнее, виртуальная машина Java (JVM), может и будет служить платформой для создания компиляторов с различных языков в байт-код JVM. Питер оказался прав. Уже в скором времени он смог опубликовать в PC Week сообщения о компиляторах языка Ада, генерирующих байт-код.

А теперь вы можете посмотреть и на результат компиляции программы на Обероне в код виртуальной машины Ява. В качестве примера возьмем используемый Тестовым центром PC Week Labs Tak-тест. Вот исходный текст соответствующего модуля на Обероне.

```
MODULE TakTest;
  TYPE tNumber* = INTEGER;

  PROCEDURE Tak*( x, y, z : tNumber ) : tNumber;
  BEGIN
    IF y>=x THEN RETURN z END;
    RETURN Tak( Tak(x-1,y,z), Tak(y-1,z,x), Tak(z-1,x,y) )
  END Tak;

END TakTest.
```

Выполнение теста состоит в выяснении того, сколько раз за одну секунду тестируемая система может вычислить значение Tak (18, 12, 6) (оно, для справки, равно 17). В нашем случае Tak — экспортируемая модулем TakTest процедура-функция.

Ну а это — результат дизассемблирования байт-кода, сгенерированного компилятором Оберона при трансляции модуля TakTest.

```
 0 ( 140) iload_1
 1 ( 141) iload_0
 2 ( 142) if_icmplt 5
 5 ( 145) iload_2
 6 ( 146) ireturn
 7 ( 147) iload_0
 8 ( 148) iconst_1
 9 ( 149) isub
10 ( 150) iload_1
11 ( 151) iload_2
12 ( 152) invokestatic 3
15 ( 155) iload_1
16 ( 156) iconst_1
17 ( 157) isub
18 ( 158) iload_2
19 ( 159) iload_0
20 ( 160) invokestatic 3
23 ( 163) iload_2
24 ( 164) iconst_1
25 ( 165) isub
26 ( 166) iload_0
27 ( 167) iload_1
28 ( 168) invokestatic 3
31 ( 171) invokestatic 3
34 ( 174) ireturn
```

Код JVM — это, по сути, обратная польская запись программы, интерпретация которой основана на использовании стека. Числа слева в каждой строке означают адрес инструкции относительно начала кода метода (в данном случае мы видим код метода Tak, его размер 35 байт). Числа в скобках — смещение от начала файла класса. Каждая строка содержит мнемонику одной команды JVM. За некоторыми командами следуют числа — это либо относительные адреса (if_icmplt 5 — переход "если меньше" на 5 байт вперед), либо ссылки на константный пул файла класса (invokestatic 3 — вызов процедурой Tak самой себя).

И это все о нем (вместо заключения)

- Оберон — самый простой язык высокого уровня, обладающий средствами структурного, модульно-компонентного и объектно-ориентированного программирования.
- Простая модульная структура со строго регламентированными экспортом-импортом, исчерпывающим межмодульным контролем и автоматическим формированием интерфейса.
- Простая, полная и ясная парадигма объектно-ориентированного программирования.
- Совершенный механизм управления динамической памятью на основе сборщика мусора.
- Оберон — язык надежного программирования, содержащий совершенные механизмы контроля.
- Оберон — зрелый и стабильный язык.
- Оберон — идеальный язык для обучения программированию.
- Переход на Оберон опытных программистов практически не требует переподготовки.
- Использование Оберона в качестве стандартного языка в отраслях, требующих повышенной надежности, унификации и мобильности ПО, позволит достичь технических преимуществ и экономии средств.