

Обучение программированию: российская перспектива

Текст основан на докладе, представленном на конференцию JMLC'2003 (Joint Modular Languages Conference, Klagenfurt, Austria, August 2003). Английский вариант опубликован в *Modular Programming Languages. Lecture Notes in Computer Science 2789, Springer-Verlag, 2003, ss.69-77. [Английский вариант доклада.](#)*

Ф.В.Ткачев
Институт ядерных исследований РАН
Москва 117312, Россия
<http://www.inr.ac.ru/~info21/>

Уникальное сочетание свойств делает языки семейства Оберон идеальной платформой для разработки алгоритмов для научно-технических расчетов, а также для систематического обучения программированию. Проект Информатика-21 опирается на сильную традицию виртовских языков в России и пропагандирует для широкого использования в российском образовании созданный Н. Виртом и сотр. язык программирования Оберон/Компонентный Паскаль — прямой наследник легендарных Паскаля и Модулы-2. В качестве стратегической цели имеется в виду создание общей системы, в рамках которой учащиеся средней и высшей школы обучались бы фундаментальным основам алгоритмического мышления и программирования столь же основательно, как это имеет место в России с математикой.

1. Образовательный проект Информатика-21 [1] вырос из следующих двух наблюдений: во-первых, языки программирования семейства Оберона ^[1] дают близкое к идеальному решение проблем, с которыми сталкивается программирование в современной физике и др. науках [6]; во-вторых, идеи, заложенные в Оберон, имеют универсальный характер, что дает прочную основу для общего обучения проектированию алгоритмов и программ профессионалов в областях, отличных от собственно информатики, таких как физика, химия, инженерное дело, лингвистика [17] и т.д.

В частности, в физических исследованиях, проводимых в международных лабораториях, таких как CERN [26], наблюдаются следующие тенденции:

— Даже рядовые вычислительные проекты все чаще требуют использования двух и более специализированных независимых систем программирования. Например, давно известна проблема интерфейса между символическим и численными расчетами. Более новый пример — сетевой сервис (web service), основанный на библиотеках программ для численных и символических расчетов, управляемых с помощью базы данных, с внешним интерфейсом, написанным на Java [7].

— Специализированные системы естественным образом эволюционируют в направлении включения основных средств языков программирования общего назначения. В результате происходит дублирование такого ядра основных средств с разнообразными, зачастую ^[2] странными вариациями и пропусками в разных системах. Нарастает практическая необходимость привести разные системы к какому-то общему знаменателю.

— Все чаще исследования проводятся в рамках больших (100-1000 участников), долго живущих (до 20 лет), но в то же время нестабильных коллективов (например, студенты и аспиранты приходят в коллектив лишь на 1-5 лет, но составляют значительную долю «рабочей силы»), с довольно рыхлой структурой управления из-за политических проблем, связанных с международным характером проектов. Кроме того, приходится привлекать к разработке программ рядовых физиков. Внедрение стандартов программирования типа ограничений на используемое подмножество языка оказывается в этих условиях проблематичным.

— Сложность используемого ПО требует, чтобы рядовые программисты-физики понимали основы проектирования программ «в большом».

— Жесткое разделение труда между программистами (даже из числа бывших физиков) и собственно физиками может иметь катастрофические последствия для вычислительного проекта: программист может безуспешно бороться с трудностями, которые можно легко обойти еще на уровне физического формализма.

— И наоборот: хорошее понимание того, как формулы отображаются в алгоритмы и структуры данных, нередко играет весьма существенную эвристическую роль в теоретико-физических исследованиях (ср. изобретение наиболее популярного семейства алгоритмов для одного из важнейших классов задач в теоретической физике элементарных частиц [8]). Понимание основ проектирования алгоритмов — необходимая часть интеллектуального арсенала научно-технического работника.

— Программирование стало рутинным инструментом ученого и инженера, и естественно желание иметь возможность обращаться к нему непосредственно и по мере необходимости — примерно так же непосредственно, как мы дифференцируем и интегрируем, не переучиваясь на профессиональных математиков.

К сожалению, такое непосредственное программирование вряд ли возможно с наиболее популярными системами, несущими груз неоправданной сложности по историческим причинам.

Обозначенные тенденции отчетливо проявляются в физическом сообществе, но их нетрудно обнаружить и в других научных и инженерных дисциплинах.

В лаборатории CERN [26], начиная с 1994 была сделана попытка создать стандартную платформу для программирования на основе C++ в рамках мега-проекта LHC (Large Hadron Collider — Большой Адронный Коллайдер; 2006-2020).

Нет нужды объяснять, что C++ не может обеспечить решение перечисленных выше проблем из-за критических дефектов дизайна и экстравагантной сложности. В коллективах физиков даже имели место конфликты по поводу C++, и ряд специалистов не спешит переключаться на чрезмерно сложный C++, предпочитая пока оставаться с фортраном. Фактический провал C++ иллюстрируется разработанной в CERN библиотекой ROOT для анализа и визуализации данных [18]; библиотека, написанная на C++, печально известна своей способностью генерировать фатальные ошибки.

Очевидно, продуманный минимализм средств является ключевым требованием, если мы хотим, чтобы язык программирования мирно «уживался» в мозгу исследователей со всей необходимой физикой, математикой и проч. знаниями.

2. Автор доклада занимается разработкой вычислительных алгоритмов для теоретической физики элементарных частиц с 1978 как на уровне формализма (квантовая теория поля), так и на уровне программной реализации. В плане программирования, последний мой значительный законченный проект касался экспериментальной обработки данных [10], другой был посвящен численным расчетам [9], но в основном речь идет о крупномасштабных аналитических вычислениях для нужд теории ([8] и ссылки там). Большой, «промышленный» масштаб таких вычислений (потребности в вычислительных ресурсах здесь практически неограничены, т.к. имеется бесконечный ряд задач нарастающей сложности) ставит во главу угла эффективность программ. Возможность использовать структуры данных, специально адаптированных для конкретного класса задач, а также компилировать существенные фрагменты кода может повысить эффективность на порядки величины по сравнению даже со специализированными системами компьютерной алгебры обычного типа (пример в [11] показывает повышение эффективности в пятьдесят миллионов раз). Легкая модифицируемость программ — важность которой хорошо известна в коммерческом программировании — приобретает здесь ключевое значение, т.к. иначе невозможно проделать алгоритмическое экспериментирование, чтобы построить удовлетворительные (тем более оптимальные) программы. Здесь имеется в виду как уровень кода (статическая защита по типам и т.п.), так и хороший дизайн программы в целом (тщательная стратификация, модуляризация и т.д.).

Все эти соображения привели меня в конце концов к принятию на вооружение, начиная с 1995, системы программирования Оберон для все моей деятельности по проектированию алгоритмов (точнее, варианта Оберона-2, ныне известного как Компонентный Паскаль и

реализованного в системе BlackBox Component Builder компании Oberon microsystems, Inc. [4]). Оберон/Компонентный Паскаль зарекомендовал себя с самой лучшей стороны во всем диапазоне задач, с которыми мне приходится иметь дело: от разного рода символических вычислений (алгебра большого объема, графы, комбинаторика) до классических численных расчетов (многомерная оптимизация и адаптивные методы интегрирования Монте Карло). Безопасность по типам (strict static type safety), настоящая модульность, автоматическое управление памятью (сбор мусора), а также очень быстрый компилятор в «прозрачной» среде разработки (где, в частности, отсутствует шаг линкования, а также нет ни специального отладочного режима, ни соответствующих опций компилятора и т.п.), к тому же весьма нетребовательной к машинным ресурсам, приводят к удивительно продуктивному циклу разработки программ и алгоритмического экспериментирования, давая ощутимое преимущество даже в таких проектах, где в конце концов требуется код на другом языке программирования (С или фортран).^[4]

В целом для творческой деятельности по проектированию алгоритмов, — особенно в тех случаях, когда важна эффективность программы во время выполнения, — Оберон оказывается больше чем просто внешним инструментом, устраняя обычный для других систем технический барьер, отделяющий мыслительный процесс от реализации идей в коде.

3. Несмотря на несколько лет поисков, найти соучастников для проектов, основанных на Обероне, среди коллег не удалось (несмотря на заявляемый совершенно искренний интерес: мои коллеги-теоретики, очевидно, слишком заняты отладкой своих программ на фортране и С++ — и в этом меньшая доля шутики, чем может показаться). Это убедило меня в необходимости начать экспериментальный курс современных методов программирования на физическом факультете МГУ в весеннем семестре 2001 г. Опыт быстро выявил следующее:

— Разные курсы в обычной программе обучения программированию (которая охватывает первые два года и включает обычные вводные курсы, практикум, а также учебный мини-проект по моделированию инженерно-физических задач) используют разные платформы программирования (от Матлаба до С++) в соответствии с личными предпочтениями или предрассудками конкретного преподавателя (практически никто из которых не владеет систематическими методами программирования как «в малом», так и «в большом»).

— Студенты впустую растрачивают силы, разбираясь со случайными особенностями устаревших систем, вместо того, чтобы овладеть основами систематического построения программ (пошаговое уточнение, инвариант цикла, интерфейсы как контракты и т.п.).

— В обычно предлагаемых курсах полностью отсутствуют ключевые понятия проектирования программных комплексов — программирования «в большом» (модифицируемость программных систем [software evolvability], интерфейсы, паттерны, ...).

Короче говоря, сложность и разноречивость платформ программирования, используемых в разных курсах, имеет резко негативное влияние на то, что студенты реально узнают о программировании.

С другой стороны, прозрачный минимализм Оберона/Компонентного Паскаля позволил «упаковать» в односеместровый лекционный курс (формально подобный стандартному вводному курсу) большой диапазон современных методов программирования, включая элементы дисциплины программирования Дейкстры, базовые динамические структуры данных, основы объектно- и компонентно-ориентированного программирования вместе с основными паттернами (factory, carrier/rider, разделение интерфейса и реализации), основы интерактивной графики (MVC). Курс в определенной степени стабилизировался в третьем «издании», но его можно обогатить по мере накопления опыта и библиотеки примеров программ. Длительность курса в один семестр (16 лекций) диктовалась внешними обстоятельствами и, разумеется, является недостаточной: в идеале следовало бы посвятить один полный семестр программированию «в малом» и еще один — программированию «в большом». И все же сам факт того, что подобный односеместровый курс оказался вообще возможен, демонстрирует превосходство Оберона/Компонентного Паскаля как базового языка.

Этот пункт следует подчеркнуть особо: реальный объем сведений универсального характера, которые должны составить содержание подобных общих курсов, фактически невелик, и

вполне возможно равномерно распределить его на несколько лет (скажем, 8-11 классы средних школ и первые два года университетского обучения) без заметного увеличения общей нагрузки — тем более, что примеры программ могут иллюстрировать содержание других предметов, помогая их усвоению.

Лекции нашего курса основываются на примерах программ нарастающей сложности; их содержательная тематика коррелирует с основной программой обучения (математические курсы, физический практикум). Эффективность лекций существенно возрастает благодаря раздаче студентам перед каждой лекцией соответствующих распечатанных материалов (полные примеры программ, изображение диалогов, отрывки из документации, комментарии и т.п.). Учебник — который, конечно, был бы важным дополнением и заведомо необходим для широкого распространения курса, а также для самостоятельного обучения — налагает ограничения на характер и объем предоставляемых материалов, а компьютерная презентация — иногда незаменимая — ограничивает активность студентов в добавлении их собственных заметок.

Любопытно, что курс привлекает студентов двух противоположных категорий: во-первых, это те, для кого технические детали, с которыми неизбежно приходится иметь дело в курсах, основанных на архаичных языках, оказываются препятствием для усвоения сущности программирования;

во-вторых, это студенты, уже имеющие опыт программирования, для которых стандартные курсы оказываются фактически бессодержательными. Первая категория получает возможность практически немедленно решать интересные задачи. Вторая — подняться над уровнем технических деталей кодирования и взглянуть на предмет с систематической точки зрения, а также быстро войти в важнейшую проблематику проектирования больших программ. Отмечено посещение курса и преподавателями, ищущими эффективные возможности повысить собственную научную продуктивность. Все это — следствие простоты Оберона/Компонентного Паскаля, которую не следует путать с примитивностью, т.к. достигнута она за счет в высшей степени точного дизайна (хороший аналог — знаменитый «калашников»).

4. Правильное позиционирование курса очень важно ввиду «религиозного» характера дебатов по поводу языков программирования. В настоящее время курс позиционируется сл. образом:

— Оберон/Компонентный Паскаль обеспечивает максимально безболезненное введение в современное программирование. После изучения основ проектирования программ на Обероне, переход на другие популярные языки сводится к изучению несущественных деталей (синтаксических и т.п.) в существующих специальных курсах, традиционно посвященных почти исключительно таким деталям (фортран, С, С++).

— Оберон/Компонентный Паскаль оказывается весьма производительным инструментом, радикально упрощающим работу студентов по обработке данных в физическом лабораторном практикуме, в практикуме по численным методам, а также в последующих индивидуальных исследованиях.

— Даже если в дальнейшем приходится иметь дело с другими языками, следование систематическому стилю, усвоенному при работе в Обероне, позволяет существенно нейтрализовать влияние дефектов дизайна этих языков (достаточно для каждой из конструкций Оберона выбрать способ эмулировать ее на целевом языке, что автоматически порождает разумный стандарт кодирования). Например, студенты рассказывали о заметном облегчении процесса решения задач в практикуме по численным методам, если задания сначала разрабатываются и отлаживаются на Обероне, и только уже готовые программы транслируются в С в соответствии с требованием преподавателя.

— Критически важно в российском контексте, что использование Оберона существенно ослабляет ограничения, связанные с «железом»: Обероны вполне функциональны на ПК класса 386 и прекрасно работают на 486, но этого оказывается достаточно, чтобы серьезно осваивать качественное современное (объектно- и компонентно-ориентированное) программирование.

— И последнее. Системы Java и С# корпораций Sun и Microsoft — два языка

программирования, борющиеся за доминирование в бизнес-ориентированной индустрии ПО — в плане идеологии имеют больше общего с Обероном (минимализм, автоматическое управление памятью, отказ от множественного наследования), чем со своими синтаксическими предшественниками С и С++. ^[5] Этот факт можно интерпретировать как возникновение консенсуса среди лидеров рынка ПО в отношении ядра существенных (в т.ч. объектно-ориентированных) средств, которыми должен обладать современный язык программирования общего назначения — этот консенсус можно назвать *стандартной парадигмой современного программирования*. Из трех языков, Оберон остается лучше остальных приспособленным для вычислительных приложений, для встроенных систем, а также — благодаря безупречно чистому дизайну — наиболее легким введением в круг методов современного программирования.

5. В процессе чтения курса стала очевидной и серьезная проблема. А именно, изрядное число студентов приходит в университет уже с опытом программирования. К сожалению, их опыт базируется либо на каком-либо варианте старого паскаля, либо (реже) на С/С++ — причем обучали их обычно не слишком компетентные школьные учителя, причем как учителя, так и учащиеся обычно путают сложность языка программирования с его мощностью (корреляция на самом деле, скорее, обратная). Такие студенты избегают «вводного» курса, предпочитая заниматься в свободное время коммерческим программированием. А плохие привычки в программировании, сформированные неадекватным обучением, устраняются с трудом. (Здесь имеется довольно близкая аналогия со спортом, где правильная постановка техники — «школа» — совершенно необходима для высших достижений. Эта аналогия помогает мотивировать таких студентов предпринять усилия по переучиванию.)

Как только выявилась проблема с преподаванием на уровне средней школы, осенью 2001 г. был начат экспериментальный курс в лицее Научного центра РАН в г. Троицке (Моск. обл.). Курс, читаемый в формате обычного школьного курса информатики, был мотивирован прежде всего практической необходимостью обеспечить приток хорошо обученных студентов, но также и идеалистическим взглядом на проблему как имеющую всеобщий и стратегически важный характер.

6. Необходимо сделать несколько замечаний по поводу образовательной системы в России в свете дискуссий о том, как извлечь выгоду из сильной российской системы образования и фундаментальной науки в процессе пост-индустриальной трансформации общества. Отметим малоизвестный по идеологическим причинам факт, что российская традиция солидного математического образования возникла как прямой результат образовательной реформы 1871 года, проведенной в жизнь императорским правительством после покушения на жизнь Александра II в 1866 г. При этом максимальный упор в гимназической программе был сделан на классические языки и математику, выбранных на роль противоядия для распространяющегося свободомыслия ^[12]. После 1917 г. латынь и древнегреческий были заменены на естественные науки, а математика — снова благодаря своей идеологической нейтральности — осталась нетронутой в образовательном проекте большевиков, которым теперь было охвачено практически все население страны. Получившаяся образовательная система с солидным всеобщим математическим обучением оказалась ^[17] отличным фундаментом для развития выдающихся школ математики и теоретической физики, а также, среди прочего, аэрокосмической промышленности.

Напрашивается идея развить основательную математическую традицию в образовании, дополнив ее столь же основательной системой обучения программированию (здесь, конечно, речь идет об алгоритмическом мышлении и о «бессмертных принципах», а не о «практической информатике» как изучении особенностей коммерческих систем). Идея согласуется с объективной тенденцией: аналитики отмечают, что на международном рынке оффшорного программирования российские программисты тяготеют к специализации на сложном, математически насыщенном софте (ср. проекты Нижегородского центра разработки ПО корпорации Intel ^[19]). Хотя в финансовом выражении обороты пока невелики по сравнению, скажем, с индийской индустрией ПО, но темпы роста впечатляющие (до 50% в год ^[27]), к тому же производство сложного специализированного ПО гораздо труднее автоматизировать, а его роль и ценность растет, постепенно приобретая стратегическое значение.

7. Реализация подобного плана должна учесть ряд обстоятельств, специфичных для России и др. государств СНГ:

— Устаревшее «железо» в большинстве школ (где оно вообще есть) будет еще в течение какого-то времени препятствием для использования «больших» систем программирования.

— Плохое знание иностранных языков создает определенные информационные трудности, но, с другой стороны, создает и определенный иммунитет к маркетинговым влияниям из-за рубежа.

— Сильная традиция использования паскаля: паскаль (различные версии) с большим отрывом лидирует в обучении программированию в России. Частично это объясняется популярностью алгола-60 в университетском образовании во времена СССР в 70-е гг., что, в свою очередь, имело место благодаря влиянию математического сообщества, которое с самого начала приветствовало систематический подход к программированию и осознало универсальную важность ремесла программирования для будущего (одним из влиятельных специалистов в этом отношении был А.П. Ершов [20]; его наследие живо, особенно в Сибири, где, кстати, размещается значительная часть российской аэрокосмической промышленности).

— Армия хорошо обученных учителей математики, из чьих рядов нередко рекрутируются учителя программирования, и для которых дейкстровская дисциплина программирования и регулярность Оберона будут определенно привлекательны. Профессиональные математики нередко вовлечены в обучение в физико-математических школах; ср. учебник с примерами систематического построения алгоритмов на паскале, написанный математиком [13]. Отметим, что Оберон упоминается во введении к учебнику как «потенциально более элегантный выбор» языка программирования для записи примеров.

— Отсутствие четких ориентиров, на которое жалуются учителя. Для тех, кто изначально получил математическое образование, очевидной моделью является евклидова геометрия, и они будут приветствовать столь же систематическое изложение оснований программирования.

— Традиция паскаля в образовании коррелирует с промышленным использованием виртуальных языков. Один пример — значительное и активное сообщество программистов, работающих на Дельфи [21]. Другой пример: встроенное ПО для российских спутников связи пишется целиком на Модуле-2 [22].

Ввиду сказанного закономерно, что Оберон/Компонентный Паскаль должен приобретать популярность на территории бывшего СССР. И в самом деле, распространение Оберона происходит сразу несколькими путями. Уже упоминались научные и образовательные проекты в МГУ и институтах РАН. В аэрокосмической промышленности Оберон изучается как естественный наследник Модулы-2. С 1998 г. существует созданный С.З.Свердловым (Вологодский ун-т) сайт [14], пропагандирующий Оберон для студентов университетов, специализирующихся в информатике; сайт предлагает богатую коллекцию переводов на русский язык публикаций, относящихся к Оберону. Подтверждением эффективности использования Оберона в обучении студентов является тот факт, что вологодская студенческая команда пробилась в финал 27-го международного чемпионата мира по программированию (Бeverли Хиллс, США, 25 марта 2003 г.). Стоит отметить внимание, уделявшееся Оберону в ИТ-периодике на русском языке (в частности, статьи Р.Богатырева, например, [25]). С 1999 г. Оберон/Компонентный Паскаль пропагандируется как лучшая платформа для обучения программированию литовским Институтом математики и информатики [24]. Начиная с осени 2003 г., в университете г. Ош (Киргизия) реализуется система подготовки профессиональных программистов на основе Оберона [23].

8. Проект Информатика-21 [1] пытается скоординировать эти «низовые» тенденции. Проект возник как сайт для нужд упоминавшегося курса современного программирования на физфаке МГУ, а затем для распространения информации об Обероне вместе с ориентированными на русскоязычные школы материалами (прежде всего, пакета русификации для системы Блэкбокс, включая русскоязычные сообщения компилятора, начальные инструкции для работы в Блэкбоксе, примеры разработки программ и т.п.). Упор был сделан на средние школы, т.к. именно здесь больше всего нужна помощь, а эффект перехода на Компонентный Паскаль/Оберон был бы максимальным в долгосрочной перспективе.

Вокруг проекта сформировалась панель консультантов, представляющих аэрокосмическую индустрию, Российскую академию наук и МГУ. Проекту помогают добровольные координаторы

в Сибири (А.И.Попков, Стрежевский филиал компании Сибинтек и школа программирования «Лидер»), в Средней Азии (проф. Кубанычбек Тажмамат уулу, Ошский ун-т), где проблемы образования подобны российским, и на Украине (В.В. Лось, Харьковский политехнический ун-т). К настоящему моменту с сайта проекта скачано порядка сотни копий пакета, специально предназначенного для школ, что, учитывая наличие на сайте аналогичного пакета для профессиональных разработчиков, дает оценку числа преподавателей, заинтересовавшихся системой Блэкбокс. Поддержка публикации материалов заявлена Учебно-научным центром доуниверситетского образования при МГУ.

Курсы программирования для школ (8-11 классы) должны подводить посредством простых, тщательно построенных примеров к систематическим методам построения программ, которые в полном объеме должны преподаваться в общих курсах программирования в университетах. В этом отношении главной целью проекта является накопление комментированных образцовых примеров программ, по-возможности увязанных с тематикой других предметов. Кроме того, еще только предстоит осознать далеко идущие последствия для преподавания (как, впрочем, и для дизайна алгоритмов) такой особенности Оберона, как автоматическое управление памятью.^[8]

Как показывает наш опыт лицейского преподавания, вполне возможно и интересно вводить динамические структуры данных (простые списки) гораздо раньше, чем это обычно делается (в обычном преподавании нередко отчетливо виден крен в сторону задач, мотивированных тематикой старого, «фортранного» научно-технического программирования — в ущерб, например, записям и спискам).

Еще очень ограниченные познания в математике 12-летних школьников являются препятствием для изучения ими программ вычислительного типа, тогда как усвоение манипуляций со списками оказывается несложным (постольку, поскольку отпадает необходимость управлять памятью вручную) и интересным: с помощью мела на доске очень легко отслеживать меняющуюся картину ссылок при, скажем, добавлении элемента в список. При этом интерес школьников возбуждают программы, манипулирующие информацией о них самих (простешая база данных класса с оценками и тому подобной информацией). Совершенно очевидно, что Компонентный Паскаль/Оберон открывает нетривиальные новые возможности для более эффективной (в ряде отношений) организации курса.

Для подобных образовательных проектов большую помощь принес бы образцовый учебник программирования, задачу создания которого недавно поставил Никлаус Вирт [15]. К счастью, сам по себе Оберон/Компонентный Паскаль великолепным по точности образом суммирует ключевые понятия программирования и дает отличную техническую основу, на которой можно уже сейчас строить современную общую систему преподавания программирования и информатики.

Автор благодарит профессора Никлауса Вирта за многочисленные замечания о преподавании программирования, а также профессора Юрга Гуткнехта, предложившего название для данного доклада.

Литература

1. Проект Информатика-21, <http://www.inr.ac.ru/~info21/>
2. Wirth, N.: The Programming Language Oberon, Software — Practice and Experience, 18 (1988) 671–690;
Wirth, N., Gutknecht, J.: Project Oberon: the Design of an Operating System and Compiler, ACM Press (1992)
3. Wirth, N., Mossenbock, H.: Oberon-2 Language Report (1992)
4. Oberon microsystems, Inc.: Component Pascal Language Report (2001);
<http://www.oberon.ch>
5. Gutknecht, J.: in Proc. of JMLC'1997. Lecture Notes in Computer Sciences 1024, Springer Verlag;
Real, P.: Active Oberon Language Report, <http://bluebottle.ethz.ch/language-report/>

6. See e.g. F.V.Tkachov: From Novel Mathematics To Efficient Algorithms, <http://arXiv.org/abs/hep-ph/0202033>
7. Bardin, D., et al.: Project SANC (2003); e.g. <http://arxiv.org/abs/hep-ph/0212209>
8. Tkachov, F.V.: Algebraic Algorithms for Loop Calculations, <http://arXiv.org/abs/hep-ph/9609429>
9. Tkachov, F.V., Manakova, G.I., Tatarchenko, A.F: How Much Better Than Vegas Can We Integrate in Many Dimensions? FERMILAB-CONF-95-213-T (1995)
10. Tkachov, F.V.: Verification of the Optimal Jet Finder, <http://arXiv.org/abs/hep-ph/0111035>
11. <http://www.inr.ac.ru/~ftkachov/projects/bear/dirac.htm>
12. Корнилов, А.А.: Курс истории России в XIX веке, Высшая школа, Москва (1993)
13. Шеть, А.: Теоремы и задачи по программированию, Москва (1995); Birkhauser (2000)
14. <http://www.uni-vologda.ac.ru/oberon/>
15. Wirth, N.: Computing Science Education: The Road not Taken, Opening Address at the ITiCSE Conference, Aarhus (2002)
16. Veltman, M. and Williams, D.N.: Schoonschip '91, <http://arxiv.org/abs/hep-ph/9306228>
17. Ср. программный пакет CHRISTINE для лингвистических исследований, созданный лингвистом Дж.Сэмпсоном (G.Sampson): <http://www.grsampson.net/RChristine.html>
18. <http://root.cern.ch/>
19. <http://www.intel.com/jobs/russia/sites/nizhny.htm>
20. Архив А.Ершова: <http://www.iis.nsk.su:81/ershov/english/>
21. <http://www.delphikingdom.ru/>
22. Колташев, А.А.: доклад на данной конференции.
23. <http://www.kubanych.ktnet.kg/>
24. <http://www.mii.lt/>; <http://aldona.mii.lt/pms/jpm/>
25. Р.Богатырев, Гадание на кофейной гуще, Мир ПК, №2, 1998, стр. 120–133.
26. <http://www.cern.ch>
27. Ведомости, 5 июня 2003 г.

[1] Для целей настоящей дискуссии разница между первоначальным Обероном [2] и языками Оберон-2 [3], Компонентный Паскаль [4], Active Oberon [5], и т.д., в основном несущественна.

[2] Например, рудиментарные наборы управляющих структур в системах для символических

вычислений таких как SCHOONSCHIP [16] и его производные, которые весьма популярны в некоторых разделах физики.

[3] В марте 2003 достаточно несложная прикладная программа, использующая ROOT (под управлением ОС Linux), для обработки и довольно простой визуализации данных по космическим лучам сгенерировала три ошибки типа segment violation за менее чем пять минут демонстрации. Другой пример: многоопытный и весьма уважаемый программист сменила специализацию, чтобы не иметь дела с постоянно «грохающимся» программным обеспечением, написанным на C++, работа с которым напоминает работу на больших ЭВМ советского производства в 80-х гг., когда ЕС-1060 «зависала» каждые 15-30 минут.

[4] Дать точную оценку повышения производительности, достигаемому благодаря простоте и надежности Оберона, нелегко из-за отсутствия однозначных метрик. Можно указать пример: поиск эффективного алгоритма для одной специальной задачи оптимизации в $O(2000)$ -мерной области [10] потребовал меньше времени на Компонентном Паскале (4 недели), чем последующий перенос на фортран. При этом для устранения возникшей проблемы с ошибками округления пришлось вернуться на КП, т.к. сделать это непосредственно на фортране оказалось тяжело.

[5] Известно, что Оберон оказал непосредственное формирующее влияние на разработку языка Java [25]. Добавим, что исходные тексты компилятора Оберона были изучены в корпорации Sun уже около 1991 г., т.е. задолго до того, как был объявлен язык Java.

[6] Реформа была «продавлена» через все комиссии (коих, как и сейчас, было достаточно) министром народного просвещения графом Д.А.Толстым. Похоже, что уникальная российская математическая традиция обязана этому реакционеру в большей степени, чем гениальным одиночкам вроде Л. Эйлера и П.Л. Чебышева.

[7] Кроме физико-теоретической школы Л.Д.Ландау (который, как известно, брал интегралы «спинным мозгом»), можно указать соперничающую с ней школу Н.Н.Боголюбова, пришедшего в физику из математики, а также современную школу математической физики академика Л.Д.Фаддеева, также выросшую непосредственно из российской математической традиции.

[8] В теоретическом плане можно показать, что Оберон-2 содержит в оболочке процедурного языка все важнейшие механизмы функционального программирования, включая т.наз. currying.