Система ИТ-образования с точки зрения российских национальных интересов и научно-образовательных традиций

Доклад представлен на Второе Всероссийское совещание "Актуальные проблемы информатики в современном российском образовании", 27-29 июня 2005, МГУ

В.А.Матвеев[(1)], А.А.Колташев[(2)], Н.В.Чистяков[(3)], А.И.Попков[(4)], Н.П.Кучер[(5)], Ф.В.Ткачев[(6)]

[(1)] Директор Института ядерных исследований РАН, председатель Совета Россия-CERN. [(2)] Начальник сектора системного программирования космических аппаратов, НПО прикладной механики им. М.Ф.Решетнева. [(3)] Главный конструктор комплексов дистанционно пилотируемых летательных аппаратов, Научно-производственный центр НОВИК-XXI. [(4)] Руководитель школы программирования «Лидер», г. Стрежевой, Томская обл. [(5)] Директор Лицея Научного центра РАН в г. Троицке, Московская обл. [(6)] Ведущий научный сотрудник Отдела теоретической физики Института ядерных исследований РАН, координатор проекта Информатика-21.

Цель доклада — зафиксировать некоторые тезисы по проблеме современного ИТ-образования в России, а также сформулировать конкретные предложения по обучению программированию как одной из важнейших профессий ближайшего постиндустриального будущего.

1. В текущих дискуссиях по этой проблеме отражаются, в сущности, лишь краткосрочные интересы российской гражданской ИТ-индустрии, которая развивается сегодня в кильватере зарубежной. При этом не обсуждаются интересы обширной категории специалистов научно-технических специальностей, и практически полностью игнорируются интересы стратегически важных секторов (аэрокосмос, ВПК, ядерноэнергетическая сфера, фундаментальная наука). Также отсутствует понимание ценности имеющейся традиции использования Паскаля в преподавании программирования как странового конкурентного преимущества, нуждающегося в защите. На основе «традиции Паскаля» и по образцу уникальной и успешной системы математического образования может быть создана настоящая *система* ИТ-образования. Для этого в качестве первого шага предлагается внедрение единой технологической среды для систематического преподавания основ программирования в старших классах средней школы и на младших курсах университетов. Среда, практически идеально удовлетворяющая всем возникающим здесь многообразным требованиям (бесплатность, образцовое качество и т.д. — см. обсуждение далее в докладе), основана на самой современной, отшлифованной и усовершенствованной версии Паскаля, адаптирована под нужды российского преподавания (вплоть до русификации сообщений компилятора) и успешно применяется в педагогической практике с 2001 г. как на школьном, так и на университетском уровнях (физический факультет МГУ) в рамках проекта Информатика-21 [1]. Внедрение этой единой технологической среды создаст беспрецедентное общее «пространство» с уникальными условиями для эффективного обучения и интенсификации процессов накопления и обмена методическим опытом. Благодаря легкости программирования, в т.ч. нетривиальной интерактивной графики, такое «единое пространство» будет прогнозируемо взаимодействовать с курсами по другим предметам, внося ценнейший элемент активности в процесс обучения (через упражнения, содержание которых заимствовано из других предметов, и т.п.). Наконец, высокое качество и современность используемых технологий при полной открытости учащимся и преподавателям всех исходников (включая компилятор и проч.) предоставляет уникальные возможности для повышения качества подготовки будущих ИТ-профессионалов. Все это создаст практические предпосылки для реализации известных рекомендаций ЮНЕСКО по ИТ-образованию [2]. Безотлагательное, приоритетное создание такого «единого пространства» посредством внедрения единой среды, основанной на современной, методически безупречной версии Паскаля, является давно назревшим шагом, который будет иметь долгосрочные последствия для превращения России в ИТ-державу первого ряда.

Наши заключения основаны на совокупном опыте преподавания, охватывающем учащихся 12-22 лет, а также на опыте профессиональной работы в областях и ситуациях, где требования к качеству программ выше среднего по ИТ-индустрии.

2. *«Программирование — вторая грамотность».* Этот лозунг А.П.Ершова подзабыт, но нуждается в том, чтобы его освежить, т.к. его актуальность только выросла за прошедшие тридцать лет.

Чтобы построить адекватную информационную модель/управляющую систему нужны знания не только ИТ, но и конкретной предметной области. Выясняются следующие факты:

1) Специалистов, не являющихся профессиональными программистами, но занятых серьезным программированием существенную долю своего времени, *очень* много [3]. (Оценка Microsoft: в три раза больше, чем программистов-профессионалов [4].) Фактически любой профессионал в любой области интеллектуально-нагруженной деятельности, начав использовать компьютер, постепенно «втягивается» в программирование, просто следуя внутренней логике развития своих профессиональных проектов.

2) Все чаще возникают ситуации, когда даже если специалист сам не занимается разработкой программ, ему нужно взаимодействовать с программистом. Проблема взаимопонимания существенно упростится, если специалист будет понимать хотя бы базовый набор терминов (ввод данных, процедура, диалог, база данных…)

3) Сложность программ, которые создают эти программисты-непрофессионалы, может быть очень высокой (факт, хорошо известный, например, в физике), и имеет тенденцию к росту. Даже просто настройка достаточно мощного каркаса (framework — современный вариант того, что раньше называли библиотекой процедур) под конкретную задачу обычно требует полноценного программирования.

4) Программисты-непрофессионалы используют даже большее разнообразие систем программирования, чем профессионалы. Это связано с большим разнообразием специализированных систем, предназначенных для узких классов задач (только в физике элементарных частиц используется несколько таких систем). Однако по мере усложнения задач наблюдается четкая тенденция к тому, что все эти специализированные системы эволюционируют в сторону универсальности, т.е. в итоге содержат всё более полный базовый комплект средств «общего» программирования, хотя и со специфическими вариациями. В результате специалисты все чаще предпочитают брать в качестве основы для своих проектов профессиональные системы программирования общего назначения.

5) В голове программиста-непрофессионала знание ИТ должно уживаться со знанием конкретной области (например, у физика это может быть одновременно квантовая теория поля, феноменология элементарных частиц, а также ряд разделов чистой и прикладной математики). Ясно, что здесь возникает сильнейший дополнительный (по сравнению с ИТ-профессионалами) стимул стремиться к предельной простоте инструментальных средств (в частности, языков программирования). Это входит в известное противоречие с другим фундаментальным фактом, а именно, что для мира ИТ как корпорации, противостоящей остальному миру, избыточная сложность инструментария ИТ есть источник дополнительной «ренты сложности», получаемой за счет остального мира. Речь идет не столько о заговоре, сколько о хорошо известных науке «коллективных эффектах»: ведь даже молекулы H_2O «умеют» находить наиболее выгодные конфигурации (кристаллы льда, минимизирующие коллективную потенциальную энергию молекул). Примеров сознательной и бессознательной эксплуатации избыточной сложности в мире ИТ множество (нередко через сопутствующую сложности ненадежность, как, например, в случае рынка антивирусных и т.п. программ). Сложность можно искусственно создать, размножая «квази-стандарты» в целях защиты своего положения монополиста (чем систематически злоупотребляют доминирующие на рынке компании). Другой пример: физик, выучивший бессмысленно сложный C++, мгновенно становится ценным специалистом среди коллег, если каким-либо образом возникнет общее мнение, что отныне все программы в лаборатории должны писаться на C++. В свою очередь, такое мнение легче будет принято коллегами, если предмет мнения (в данном случае C++) столь сложен, что

разобраться в нем и в аргументах, выдвигаемых в его защиту, невозможно, не бросив все текущие эксперименты на достаточно долгое время.

6) В ряде серьезных областей (аэрокосмос, «умное оружие», ядерная энергетика, дорогостоящая экспериментальная физика, автоматизированные управляющие системы на транспоре …) ПО функционирует в условиях, где программный сбой *невозможно помыслить*. По сравнению с традиционными бизнес-приложениями, это накладывает гораздо более жесткие требования на качество программ, базовую подготовку программиста и используемые технологии, начиная с языка программирования. Здесь «не проходит» многое из мифологии, исторически возникшей в среде самоучек, каковыми в сущности является значительная доля (если не большинство) программистов-профессионалов, обслуживающих обычный бизнес.

3. *Об общем базовом курсе программирования.* Из сказанного можно сделать выводы:

Во-первых, проблематика преподавания программирования касается гораздо более широкого круга учащихся, чем будущие ИТ-профессионалы. На самом деле основы программирования («алгоритмика» и т.п.) — это дисциплина, сравнимая по общему значению с математикой и близкая последней по духу. Соответствующие знания относятся к категории общезначимых (фундаментальных) и должны преподаваться как таковые. Объем базового комплекта знаний должен включать, наряду с формальной логикой, например, достаточно уверенное умение работать с простейшими динамическими структурами данных (списками). С другой стороны, при обычной практике учащиеся слишком рано «грузятся» слишком большим объемом совершенно несущественных и неуниверсальных деталей (что особенно справедливо для курсов, базирующихся на языках семейства Си). Наблюдения показывают, что в результате цель овладения действительно важными, базовыми и общими знаниями просто не достигается. Необходимо решительно отделить здесь «мух от котлет» — винить студентов в нерадивости бессмысленно и безответственно: например, объем базовых знаний, преподаваемых на младших курсах физического факультете МГУ, очень велик и без программирования.

Во-вторых, еще раз вспомним, в какой огромной степени наша жизнь стала зависеть от всякого рода автоматизированных управляющих систем разного масштаба, и насколько важна корректность всего огромного и *растущего* массива ПО, пронизывающего окружающий нас мир. Страшно подумать, что этот массив ПО создают программисты, подавляющее большинство которых не прошло школу систематического программирования и используют соответственно неадекватные средства программирования (результат мы все видим при работе с офисным ПО). Причем со временем важность этой проблемы только растет. Вывод здесь однозначный: в преподавании основ программирования с самого начала (т.е. примерно с 12 лет, когда только начинает формироваться способность к абстрактному мышлению, чтобы обеспечить самый глубокий импринтинг этих идей в умах учащихся) должен быть сделан упор на формирования ментальности, направленной на построение корректных и надежных программ, т.е. на систематический, доказательный характер программирования (в отличие от спонтанно приобретаемой привычки составлять программы «методом проб и ошибок»).

В-третьих, систематическое программирование — слишком важное и нетривиальное дело (самостоятельно додуматься в процессе «профессионального программирования» до понятия инвариант цикла нереально), чтобы осложнять его освоение параллельным изучением темных деталей громоздких коммерческих систем. Следует четко разделить цели и, соответственно, содержание курсов: специальную подготовку (изучение специфики конкретных систем программирования, по случайным историческим причинам популярных в данный момент у конкретного класса специалистов) следует выделить в специальные курсы, читаемые после прочного усвоения базового материала. Базовые курсы должны быть полностью сфокусированы на освоении базовых понятий и формировании адекватной ментальности.

В-четвертых, курс программирования все-таки должен базироваться на каком-то языке и системе программирования. Утверждается, что задача базового общего курса, сформулированная выше, слишком важна сама по себе и требует, чтобы именно ее решению был подчинен в первую очередь выбор. Естественно хотя бы первоначальные ориентиры искать в реальной практике. Но где? Ясно, что лишь в последнюю очередь — в обычном коммерческом программировании: там слишком велики субъективные факторы, связанные с эффектами монополизма, «отвязного» маркетинга, «стадными» эффектами, конкурентной борьбой, а главное, в

конечном счете, с тем, что и «спрос», и «предложение» на мировом рынке инструментальных средств в основном сформированы массой программистов, не только не прошедших надлежащую школу систематического программирования, но слишком часто не имеющих даже приличного базового математического образования, и для которых понятие доказательного рассуждения, судя по всему, — пустой звук.

Мы уже указали на сферы, где технологии программирования подвергаются весьма жесткому отбору именно в отношении надежности получающихся программ: аэрокосмос и т.д. А это однозначно указывает и на соответствующие языки программирования: это созданные в школе тьюринговского лауреата (1984) Никлауса Вирта язык Паскаль и его производные [5].

4. ***Паскаль, Модула-2, Оберон, Компонентный Паскаль.*** Удивительно, насколько плохо широкие массы ИТ-профессионалов (включая ИТ-профессуру — и это при постоянных жалобах на упадок дисциплины computer science) знают об уникальных разработках, выполненных классиком информатики Н.Виртом и его коллегами после изобретения Паскаля (1970). Не вдаваясь в детали (см., например, [1] и ссылки там), напомним основные факты.

Научно-инженерная школа языков и систем программирования, созданная Н.Виртом начиная с конца 60-х гг. в университете ETH (Цюрих, Швейцария), является ведущей в мире. Сам Н.Вирт является автором серии языков программирования, реализующих философию, намеченную в Алголе-60. Это Паскаль (1970), Модула-2 (1980), Оберон (1988) — единая линия все более совершенных ЯП, сохраняющих близкую преемственность. Отсутствие 100%-ной совместимости между ними побудило академически корректного Н.Вирта дать им разные имена, что, конечно, помешало распространению Модулы и Оберона. Все три языка очень близки, отличаются ясным, четким синтаксисом и все более полно реализуют концепцию строгой статической типизации данных, что на корню исключает обширные классы программистских ошибок. Все три языка в нарастающей степени поддерживают систематические методы программирования для эффективного создания эффективного и надежного ПО от небольших программ до полных операционных систем.

В Обероне система контроля типов распространена на динамические записи и в известном смысле завершена, что является выдающимся достижением, до сих пор не повторенным в других эффективно компилируемых процедурных языках (собственно, пытаться «повторять» Оберон также трудно и бессмысленно, как и АК-47). При этом в язык включен базовый набор средств объектно-ориентированного программирования вместе (и это ключевой момент) с механизмом автоматического управления памятью. Последнее обстоятельство в известном смысле придает Оберонам синтетический «процедурно-функциональный» характер: программист получает возможность эффективной работы со сколь угодно сложными динамическими структурами данных («умные» алгоритмы), но изначальная процедурная основа Оберона делает возможной простоту языка и эффективность кода без сложных оптимизирующих компиляторов.

Следует подчеркнуть, что Н.Вирт — ведущий специалист не только по дизайну языков, но и по построению компиляторов [6]. Например, Оберон проектировался одновременно с написанием компилятора, так что в языке нет «подводных камней» для разработчика компилятора — в резком контрасте, например, с популярным Java. В результате компиляторы Оберона исключительно быстрые, а по эффективности порождаемого кода конкурируют с лучшими оптимизирующими компиляторами C++ [7].

Еще одна особенность школы Вирта: системный характер языковых проектов. И Модула-2, и Оберон были частью законченных проектов рабочих станций (Lilith и Ceres), для которых с нуля были написаны как операционные системы, отличавшиеся высокой надежностью, так и полный базовый набор ПО. То есть это полноценные языки индустриального качества.

Школа Вирта пользуется огромным авторитетом у ведущих специалистов. Сам Н.Вирт и его ближайший сподвижник и соавтор по проекту Оберон Ю.Гуткнехт — постоянные гости исследовательских лабораторий ведущих корпораций. Инженеры исследовательской лаборатории SUN ознакомились с компилятором Оберона в исходных текстах еще в 1991 г., в самом начале проекта, позднее названного Java. Сильнейшее влияние Оберона и школы Вирта в целом на Java, как и на конкурирующий проект .NET/C#, вполне очевидно, если взглянуть на отличия последних от C++, реакцией на сложность которого они являются — это (относительный)

минимализм, строгая типизация, отказ от множественного наследования. Разумеется, как во всех подражаниях, выполненных без должной квалификации, по сравнению с Обероном в этих новых языках гораздо хуже проработаны детали (что особенно остро чувствуют разработчики компиляторов) и в целом сложность языка остается неоправданно высокой (наследие C++), а качество — низким [8].

Существует несколько вариантов системы Оберон, в том числе как минимум три операционных системы (ETH Oberon, XO/2, Bluebottle). К сожалению, Оберон достиг зрелости одновременно с началом дорогостоящей маркетинговой кампании Java, и, будучи академической разработкой, оказался в тени. Однако, как это обычно бывает, Java оказалась вовсе не такой панацей, как было обещано, и в настоящее время наблюдается вторая волна популярности Оберонов (см. материалы рабочего совещания в лаборатории CERN [7]). Разные реализации Оберонов используются в проектах, где требуется предельная надежность без потери эффективности. Например, по всей Швейцарии устанавливается автоматизированная система контроля дорожного движения, полностью написанная на варианте Оберона XO/2. Трудно представить себе, что ПО такого уровня надежности можно было бы столь же эффективно создать на любом из существующих Си-образных языков. Обероны используются в американском NASA, европейском ВПК (BAE Systems, DuPont Ballistic Lab), в сложнейших расчетах субъядерной физики, в крупнейших банках (Royal Bank of Canada, Swiss Re и т.п.), в системе управления крупнейшим в мире каскадом ГЭС на Амазонке (корпорация Alstom Power) и др.

Очевидно, что по совокупности технологических и практических качеств Обероны в настоящий момент — наиболее совершенные универсальные языки программирования индустриального качества, причем их полный потенциал еще не раскрыт, т.к. доминирующий стиль программирования не ориентирован на полное использование возможностей, открываемых автоматическим управлением памятью.

Нас будет интересовать вариант Оберона, специально созданный в 1993 г. учениками Н.Вирта для интегрированной работы в популярных операционных системах.[1] Соответствующий вариант Оберона получил название ***Компонентный Паскаль***, а система программирования — ***Блэкбокс*** (BlackBox Component Builder). Заметим, что со-дизайнер Блэкбокса, ученик Н.Вирта Клеменс Шиперски — ведущий мировой специалист по компонентно-ориентированному программированию [9], и был нанят в Microsoft Research Lab. для работы над проектом .NET. Одно это — достаточная рекомендация, чтобы серьезно отнестись к Компонентному Паскалю и Блэкбоксу.

Блэкбокс бесплатен и полностью открыт. Как и все Обероны, это простая, прозрачная, эффективная и расширяемая среда разработки. Блэкбокс содержит все необходимые средства интеграции с операционной системой (подробнее см. [1] и ссылки там).

Блэкбокс представляет собой исключительно удачный — близкий к идеальному — вариант системы Оберон для использования в систематических курсах программирования. При этом Блэкбокс вовсе не педагогическая игрушка, это полноценный профессиональный инструмент, по эффективности (как программирования, так и скомпилированного кода), надежности, адаптируемости и расширяемости *качественно* превосходящий, например, среду Delphi. Блэкбокс реально используется в очень ответственных проектах размером до миллиона строк (см. [7]).

5. *Проект Информатика-21. Практические мотивировки.* Адаптация и продвижение Блэкбокса и др. Оберонов в систему образования — цель научно-образовательного проекта Информатика-21, стартовавшего в 2001 г., и «вышедшего в Сеть» осенью 2002 г. Проект с самого начала нацелен на решение вполне конкретных, долгосрочных задач. А именно, Компонентный Паскаль/Оберон дает возможность заняться эффективным решением задач, с которыми «борется» современная физика элементарных частиц. С одной стороны — это задачи, в которых важны как вычислительная эффективность, так и существенно динамические структуры данных [10]. С другой стороны, легко настраиваемая мощная графическая система

---

[1] Первоначальный ETH Oberon еще раньше был перенесен на множество платформ. Однако на момент создания системы в середине 1980-х гг. никаких стандартов для графических сред еще не было, поэтому экспериментальный интерфейс ETH Oberon препятствует его использованию в преподавании.

Блэкбокса делает его очень удобным инструментом для задач, требующих гибкой визуализации экспериментальных данных. Наконец, простота Компонентного Паскаля (в отличие от других «мощных» языков) позволяет быстро осваивать его даже физикам, не имеющих опыта серьезного программирования. Хорошая интеграция Блэкбокса с операционной системой позволяет использовать уже имеющиеся библиотеки и программы (впрочем, имеется и обширная математическая библиотека для Блэкбокса [11]). Открытость исходников вместе с общим дизайном, изначально предусматривавшим легкую переносимость на другие платформы, позволяет ожидать скорый перенос Блэкбокса на излюбленную физиками ОС Linux. В описанной нише синтетических научных задач у Компонентного Паскаля и Блэкбокса нет конкурентов по совокупности технологических причин и объективных обстоятельств, и со временем его позиции в этой нише будут только укрепляться. Поэтому Блэкбокс принят в Институте ядерных исследований РАН в качестве основы для долгосрочных исследований, связанных с участием в международной европейской лаборатории CERN (проект ускорителя LHC, 2007-2022 гг.).

Курсы программирования, которые составили основу проекта Информатика-21, с самого начала имели конкретную цель: готовить студентов для участия в описанных проектах фундаментальной физики. Расширение целей — служить площадкой для образовательного эксперимента с более общим охватом — продиктовано внутренней логикой проекта: более широкие цели, важность которых очевидна, облегчают привлечение участников и способствуют устойчивости всего предприятия в перспективе, что важно ввиду долгосрочного характера указанных физических проектов. Кроме того, опыт быстро показал, что проект действительно очень сильно «попадает в резонанс» с нуждами системы образования, предлагая реальные и эффективные решения целого комплекса давно назревших проблем, хорошо осознаваемых опытными преподавателями (см. доклады С.З.Свердлова и А.И.Попкова на данном совещании).

6. *Проект Информатика-21. Структура проекта.* Публичными «воротами» в проект является сайт [1], где содержится вся информация в проекте. Сформировалась панель высококвалифицированных консультантов, среди которых и Никлаус Вирт. На сайте содержатся справочные материалы, объясняющие, как начать работать с Блэкбоксом, и ориентированные прежде всего на школьных учителей и начинающих программистов.

В рамках проекта разработаны две конфигурации системы Блэкбокс, ориентированные на школьные и университетские курсы, соответственно. Комплекты доступны свободно и бесплатно с сайта проекта. В школьном комплекте русифицированы как меню, так и сообщения компилятора (причем преподаватель может легко менять их). Кроме того, в нем содержится модуль эмуляции графики Турбо Паскаля для облегчения перехода учителей на Блэкбокс, комплект вспомогательных модулей для использования на олимпиадах по программированию, а также ряд примеров. Университетский комплект является суперсетом школьного, но оставлены английские меню и сообщения компилятора (поддержка русского алфавита в идентификаторах и документации сохраняется). Пакет содержит библиотеку для численной математики, поддержку OpenGL и т.п.

Основными экспериментальными площадками проекта являются следующие курсы, читаемые с 2001 гг. по наст. время:

— Курс «Введение в современное программирование», физический факультет МГУ.
— Курс программирования в обычном школьном формате, лицей г. Троицка, Московская обл.
— Школа программирования «Лидер» в формате внешкольных занятий, г. Стрежевой, Томская обл. (см. отдельный доклад руководителя школы А.И.Попкова)

В проекте участвует журнал Мир ПК (научный редактор Р.П.Богатырев), регулярно предоставляющий место для материалов проекта на CD-приложении «Школа программирования».

Проект имеет активный форум, вокруг которого формируется сообщество преподавателей, использующих Блэкбокс в своих курсах, а также программистов, занимающихся поддержкой Блэкбокса в формате open source, а также переводами документации.

Никлаус Вирт в качестве своего вклада в проект переделал на Оберон свои бестселлеры: «Алгоритмы и структуры данных» и «Программирование на языке Оберон». Книги переводятся на русский язык.

Активизация и прогресс проекта по всем направлениям за последний год впечатляет.

7. *Университетский опыт.* Курс на физфаке МГУ читается в осеннем семестре в формате одна лекция в неделю (всего 16 лекций), плюс практические занятия в присутствии преподавателя в компьютерном зале в объеме двух сессий в неделю по 4 часа. Первоначально курс позиционировался для студентов старших курсов теоретических кафедр, но уже во втором «издании» он был перенацелен на всех желающих, но прежде всего на студентов младших курсов. Это связано с тем, что подготовка студентов, прошедших через обычные курсы введения в ИТ (первые два года обучения) оказалась совершенно неудовлетворительной. Получившийся курс естественно вписывается в качестве первого курса двухгодичного цикла общей подготовки студентов-физиков по ИТ. Благодаря минимализму Компонентного Паскаля в односеместровый курс удается вместить довольно большой и необычный для вводного курса материал по программированию как «в малом», так и «в большом», включая систематическую работу со списками и основные «паттерны» объектно-ориентированного программирования. К сожалению, пока нет возможности читать курс два семестра, что позволило бы более систематически и полно изложить материал.

Отзывы студентов и наблюдения над ними подтверждают, что освоение всех структур языка, основных идей и методов проходит без каких-либо проблем. Более того, первокурсники высказывают сожаление, что у них не было возможности еще в 8-10 классах (когда школьники еще не загружены подготовкой к вступительным экзаменам) позаниматься подобным образом, чтобы освоить работу со списками и т.п. — ничего сложного в этом нет при наличии в языке автоматического управления памятью. Студенты очень быстро научаются делать простейшие полезные программы, так что Блэкбокс сразу становится эффективным средством поддержки работы в лабораторном физическом практикуме (обработка данных и т.п.) — за счет чего получаются дополнительные упражнения в программировании. Из-за того, что язык прост, а среда разработки очень «диалогична», возникает в высшей степени полезная синергия с основными курсами, прежде всего с математикой (которая на младших курса физфака МГУ достаточно сложна и объемна): нетривиальные программные конструкции иллюстрируют и иллюстрируются на актуальном для слушателей материале. Студентами 4-го курса (отделение ядерной физики) было подтверждено, что работа в практикуме по численным методам значительно облегчается при использовании Блэкбокса, даже когда задания требуется сдавать на других языках (фортран или Си): отладить алгоритм на Компонентном Паскале и квази-механически перенести его на другой язык оказывается проще всего. Подобный сценарий разработки програмного обеспечения (при полной автоматизации промежуточного шага перевода на другой язык программирования) давно используется в индустриальном контексте для предшественника Оберона Модулы-2 [5]. Серьезный опыт такого рода получен и при создании нетривиальных новых алгоритмов для реальных приложений в физике [7] (где, кстати, замечено, что использование автоматического управления памятью на промежуточных шагах разработки алгоритма по результативному эффекту подобно выходу в комплексную плоскость при вычислении вещественных интегралов). Это неудивительно, т.к. молниеносная компиляция и динамическая загрузка модулей в Оберонах делают их отличным средством моделирования и макетирования.

Опыт полностью подтверждает мысль, что простота Компонентного Паскаля и Блэкбокса позволяет заметно повысить эффективность всего цикла освоения основ программирования, разведя три разных заботы: усвоение базовых принципов программирования; численные методы; изучение особенностей морально устаревших, но по инерции еще используемых ЯП (фортран, Си, Си++). Это позволяет полностью исключить последние из курса основ программирования для студентов младших курсов университетов, выделив соответствующий материал в спецкурсы, где можно сосредоточиться на подводных камнях, скажем, языка Си, не озабачиваясь смыслом программирования.

Такая реорганизация представляется чрезвычайно желательной: человек, прочувствовавший образцы качественного программирования, какое поощряет Оберон/Компонентный Паскаль, без труда воспроизведет такие образцы на любом процедурном языке, с которым ему придется столкнуться. Но обратное неверно: язык, который самой своей структурой поощряет «хакерский» стиль, наносит трудноисправимый вред, если используется в качестве первого.

В этом смысле использование трудночитаемых языков семейства Си во вводных курсах явно противопоказано. Здесь уместна спортивная аналогия: из любых правил бывают исключения, и мастер может эффектно применить «нетехничный» прием, но в любом виде спорта при обучении будущих мастеров *абсолютна обязательна постановка правильной техники* с самых первых тренировок. Совершенно непонятно, почему этот закон — очевидный для любого тренера фигурного катания или тенниса — не должен применяться при обучении такому важному навыку как программирование.

8. *Лицейский опыт.* Экспериментальный курс программирования в Лицее Научного центра РАН в Красной Пахре, г. Троицк, Московской обл., проводится в формате стандартной школьной программы (что сделано намеренно). Эта часть эксперимента была затеяна, т.к. при чтении университетского курса сразу выяснилось, что многие студенты приходят на первый курс с уже испорченной техникой программирования. Причем это зачастую те студенты, которые больше всего интересуются программированием и которых жалко терять. Переучивать их достаточно трудно, в том числе и потому, что им уже неинтересно посещать вводный курс.

Главный вывод: никаких проблем с переходом на Компонентный Паскаль у мало-мальски компетентного преподавателя не возникает: отличия Компонентного Паскаля от старого минимальны, причем новый Паскаль проще старого на общем подмножестве.

Интересное методическое наблюдение таково. В начале курса программирования (8-9-й классы) школьники знают еще очень мало, и нелегко подбирать содержательные задачи. В значительной мере поэтому учителя программирования любят занятие под названием «рисование блок-схем», позволяющее «красиво» организовать классную работу с работой учащихся у доски и т.п. Однако уже лет двадцать известно, что блок-схемы — негодный прием, который когда-то был придуман, чтобы бороться с проблемами, сейчас уже давно решенными структурным программированием. Серьезные программисты давно используют форматирование программ с помощью отступов вместо блок-схем. Но чем заменить такую удобную для классной работы тему? Оказывается, есть решение, которое одновременно убивает обоих зайцев: это тема «простые списки». Обычно тема работы со списками относится к «продвинутым», т.к. в старых ЯП (операторы new/dispose) программисту может быть очень трудно отследить момент, когда нужно применять dispose. Но в Обероне/Компонентном Паскале встроенный механизм автоматического управления решает эту проблему: о dispose можно просто забыть. Это означает две вещи. Во-первых, появляется замечательная тема, требующая рисования картинок у доски (на самом деле и в профессиональной работе доска с мелом/фломастером — лучший способ отслеживать связи в списках) и способная в этом отношении заменить бессмысленное рисование блок-схем. Во-вторых, можно придумать вполне достаточно задач со списками, доступных по содержанию 8-классникам: например, работа со списками фамилий детей с оценками — что резко активизирует интерес детей, т.к. тема отметок в школе остро актуальна. Вполне реально обсуждать сортировку списка вставками: никакой особой математики для этого не требуется, хотя это не очень простая для школьников тема (впрочем, школьники в этом возрасте еще не перегружены «тяжелыми» предметами и подготовкой к поступлению в вузы).

Общая проблема, которая выявляется в школьном преподавании, это неразделенность двух в сущности разных предметов: один, сугубо прагматически ориентированный, это «курс вождения персонального компьютера», это фактически современный урок труда; другой предмет — «основы алгоритмики», т.е. собственно основы программирования в том смысле, о котором мы говорим. В первом усваиваются вещи, которые через 5-10 лет устареют. Второй предмет по сути является разделом математики, и его содержание не устареет никогда, но отношение к нему в школе такое же, как и к уроку труда. Здесь возникает различие между «фундаментальным» образованием и «практически-ориентированным». Не входя в детали, просто отметим, что второе всегда может опереться на обоюдный интерес работодателя и будущего работника (или его родителей), в то время как «фундаментальное» образование, долгосрочные последствия которого крайне важны, но не очевидны с «практической» точки зрения, — ответственность прежде всего государства и в первую очередь представляет собой предмет госстандартов.

Напрашивается вывод, что эти две половины курса ИТ должны быть разведены в школьной программе. Более того, «фундаментальной» половине должен быть придан совершенно другой статус. Правильнее всего передать эту половину полностью в ведение школьных преподавателей математики: именно их расположенность к доказательным рассуждениям должна сыграть решающую роль в выработке отношения к программированию как к целенаправленной доказательной деятельности. (Разумеется, необходимо ввести соответствующие курсы и в педагогических университетах; о весьма положительном опыте использования Оберонов в этом контексте см. доклад С.З.Свердлова на данной конференции.)

9. ***Главный вывод из опыта проекта Информатика-21***. В настоящее время плохо осознается, насколько сильно тормозит развитие *системы* ИТ-образования отсутствие единой современной технологической платформы на уровне вводных курсов. Однако существует совершенно реальная, опробованная в почти пятилетней практике преподавания в школе и университете, возможность создать «единое пространство» для преподавания фундаментальных основ программирования, на основе безупречно современного Компонентного Паскаля и соответствующей образцовой, бесплатной, открытой, коммерчески нейтральной, компактной и эффективной системы программирования Блэкбокс из семейства Оберонов, занимающей прочную нишу в фундаментальных научных исследованиях и приложениях с особыми требованиями на надежность программ. Оберон/Компонентный Паскаль является «общим делителем» целого ряда современных ЯП и может играть роль настоящего lingua franca в образовании. Переход с него на любые другие языки в случае узкопрофессиональной специализации сводится лишь к изучению специфических особенностей последних (часто дефектов дизайна и соответствующих «подводных камней»). Других реальных кандидатов на роль объединяющей платформы, столь же полно удовлетворяющих всем многообразным требованиям, которые здесь возникают, сейчас не видно.

Такое пространство должно охватывать старшие классы средней школы (8-11) и первые два года университетского обучения, и — насколько это позволяет специфика — в максимальной степени строиться по образцу существующей системы математического образования, вплоть до передачи школьных уроков программирования в ведение учителей математики.

Первый шаг — перевод курсов со старого на Компонентный Паскаль может быть сделан с минимальными усилиями без изменения содержания курсов. Для такого перевода нужен, грубо говоря, один приказ министра образования и науки Российской Федерации.[2] Никакого риска в таком шаге нет, так как терять просто нечего, а качество и современность Компонентного Паскаля и Блэкбокса заведомо намного выше, чем популярного сейчас в школе, но давно устаревшего Турбо Паскаля.

Второй этап — постепенная модификация программ обучения в направлении максимального использования новых возможностей, открываемых наличием в новом Паскале механизма автоматического управления памятью, а также «наведение мостов» с другими предметами (последний процесс приобрел бы широкомасштабный характер, как только в школу начнут приходить выпускники педвузов, поголовно прошедшие через соответствующий курс основ программирования).

Сопротивление такому плану можно ожидать со стороны ИТ-профессуры, включенной в финансовые контуры ИТ-индустрии и фактически участвующей в маркетинговых кампаниях соответствующих фирм. Однако все-таки нужно правильно расставить приоритеты: превращать российские университеты (особенно ведущие) в профтехучилища для отдельных, пусть и крупных зарубежных компаний не кажется правильным. Оптимальный баланс интересов был бы достигнут, если бы квази-маркетинговая деятельность была локализована в курсах специализации и исключена из общих вводных курсов, посвященных фундаментальным основам предмета.

---

[2] Очевидно, какое-то время на переход нужен. Но в любом случае Компонентный Паскаль должен быть разрешен к использованию на олимпиадах по программированию, чтобы школьные учителя имели ясный ориентир.

Невозможно вообразить все далеко идущие последствия внедрения единой современной платформы во вводные курсы программирования и придания таким курсам систематического характера по образцу системы математического образования. Все указывает на то, что такой шаг был бы крайне полезен — тем более, что он не несет в себе никаких рисков и вполне реален прямо сейчас.

Ссылки

[1] Сайт проекта Информатика-21: http://www.inr.ac.ru/~info21/
[2] UNESCO Computing Curricula 2001: http://se.math.spbu.ru/cc2001/
[3] F.V. Tkachov, in: Lecture Notes in Computer Science 2789. Springer-Verlag, 2003.
[4] Microsoft Corp., press release 04 July 2004.
[5] См. A.A. Koltashev, in: Lecture Notes in Computer Science 2789. Springer-Verlag, 2003, об использовании Модулы-2 в российском спутникостроении.
[6] N. Wirth, Theory and Techniques of Compiler Construction. Addison-Wesley, 1996.
[7] Workshop Oberon Day @ CERN, 10 March 2004: http://cern.ch/oberon.day/
[8] H. Thimbleby, A critique of Java, http://www.cs.mdx.ac.uk/harold/papers/javaspae.html
[9] C. Szyperski, Component Software - Beyond Object-Oriented Programming. Addison-Wesley, 1998.
[10] F.V. Tkachov, talk at Workshop on Computer Particle Physics: Automatic Calculation for Future Colliders, Tokyo, Japan, 28-30 Nov 2001. e-Print Archive: hep-ph/0202033
[11] R.D. Campbell, http://www.zinnamturm.de/#Lib

04 июня 2005