Леонов Александр Георгиевич
Первин Юрий Абрамович

РОЛЬ И МЕСТО ТЕМЫ «ЭЛЕМЕНТЫ ПРОГРАММИРОВАНИЯ» В ОБЩЕМ ШКОЛЬНОМ ИНФОРМАТИЧЕСКОМ ОБРАЗОВАНИИ

Программист обязан обладать способностью первоклассного математика к абстракции и логическому мышлению в сочетании с эдисоновским талантом сооружать все, что угодно, из нуля и единицы, он должен соединять в себе аккуратность бухгалтера с проницательностью разведчика, фантазию автора детективных романов с трезвой практичностью бизнесмена, а, кроме того, иметь вкус к коллективному труду, быть лояльным к организатору работ и так далее... Программист – солдат второй промышленной революции и, как таковой, должен обладать революционным мышлением и мужеством.

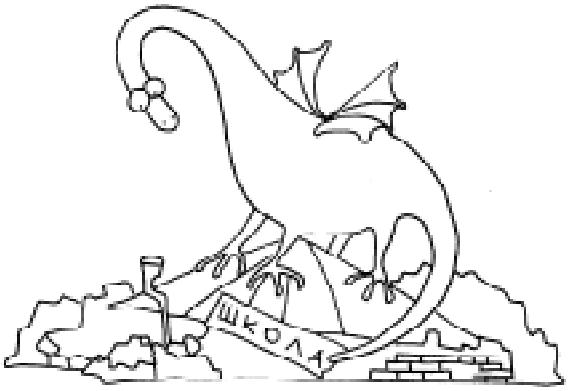
А.П. Ершов
«Программирование – вторая грамотность»

Мы хотим пригласить читателя к обсуждению общих концепций темы «Элементы программирования» и высказать по этому поводу свое мнение, которое, конечно, имеет право отличаться от того, что предложено в этой статье.

1. Исторические предпосылки школьного курса программирования.

В теме, обсуждающей место элементов программирования в школьном курсе информатики, трудно обойтись без ретроспективного взгляда в историю, хотя пятнадцатилетний период, истекший с первых шагов школьной информатики в нашей стране, вряд ли можно называть «историей»: ведь многие из тех, кто стоял у истоков, продолжают трудиться на ниве образования до сих пор. Актуальность такого исторического ракурса состоит в прослеживании тенденций развития методических и программных средств, использовавшихся в педагогической предметной области. Эти тенденции поучительны: они могут помочь как разработчикам программно-методического обеспечения, так и сегодняшним школьным учителям информатики. Впрочем, как это ни парадоксально, целый ряд практических выводов, которые можно было бы извлечь из истории и учесть в сегодняшней школьной информатике, оказались проигнорированными. Таким образом, повторение некоторых важных тезисов, которые представлялись осознанными и доказанными десять лет тому назад, остается и сегодня актуальным. Достаточно назвать, например, порочные методические характеристики Бейсика как учебного языка программирования. Как много об этом говорилось! И, тем не менее, сегодня, в конце 1999 года, на научно-практических конференциях выступают учителя с опытом работы по преподаванию Бейсика в начальной школе (!) и даже готовят педагогические диссертации о «методических находках» в обучении Бейсику первоклассников. Следует сказать, что критические стрелы в Бейсик сегодня надо направлять очень нацеленно: за последние годы появилось много новых средств, в которых фигурирует название старого языка (QuickBasic, VisualBasic...), совпадающие с Бейсиком только по созвучию и не имеющие с ним ни методической, ни структурной общности. Здесь речь идет именно о «классическом» старом Бейсике, который успел

нанести большой вред системе народного образования и который, к сожалению, еще не покинул наши школы.

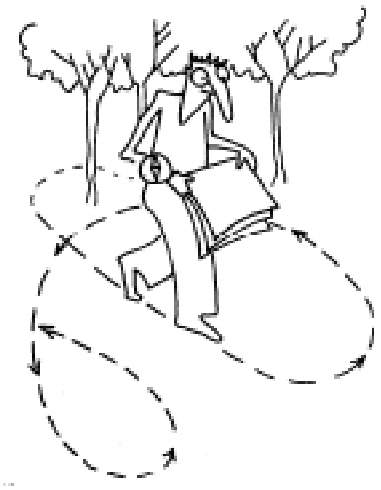


Педагогические эксперименты по преподаванию программирования в школе начались задолго до появления дисциплины с названием «школьная информатика» и даже до появления термина «информатика» в сегодняшнем его толковании. Все педагогические проблемы информатики были вынужденно сосредоточены в то время внутри одной из ветвей информатики – программирования. Это не мешало постановке задач в самом общем виде хотя бы уже потому, что программирование можно рассматривать как концентрированный сгусток основных понятий информатики. Первую скрипку в практике школьного обучения программированию сыграли профессионалы-программисты (Г.А. Звенигородский, А.Г. Кушниренко, А.Л. Семенов и многие другие), которые по разным причинам осознанно пришли в школу: одни – как разработчики учебных программных средств, другие – как организаторы новых форм учебного процесса, третьи – как практики-экспериментаторы. В результате энтузиазма таких профессионалов-программистов и родились те эмпирические приемы и наблюдения, затем постепенно выросшие в обобщенную педагогическую дисциплину – частную методику преподавания программирования, а затем и информатики. В этот «доинформатический» период школьной информатики многообразие языков программирования порождало многочисленные споры педагогов-экспериментаторов: одни утверждали, что надо учить детей программированию на Алголе, потому что на этом языке решалось большинство задач того времени во всем мире (такие идеи высказывались в начале 80-х годов); другие говорили, что компактность концептуальной базы Фортрана делает его подходящим педагогическим инструментом; третьи для тех же обоснований находили аргументацию в гибкости управляющих структур Алгола. С появлением Бейсика многие стали считать этот язык наиболее удобным для обучения программированию в силу его простоты и диалоговых возможностей.

Многообразие языков – объективное явление в мире информатики. Тот или иной язык отражает прежде всего специфику соответствующей предметной области. Ведь языкотворчество оказалось необходимым для того, чтобы построить наиболее эффективный инструмент для предметной области. Поэтому всегда считались нелепыми попытки писать программы для расчета траекторий элементарных частиц на Коболе, так же, как для составления экономических ведомостей и текстовых преобразований мало пригоден Фортран. Таким образом, для целей педагогической предметной области, столь не похожей ни на сферу экономических задач, ни на вычислительные проблемы ядерной физики, был нужен язык, учитывающий особенности пользовательского контингента (школьные учителя и учащиеся) и принятые в школе требования к дидактическим инструментам. Можно отметить, что язык программирования Бейсик, создававшийся как учебный язык, получил

широкое распространение, однако его ценность оспаривается многими исследователями и практиками-педагогами. Освоить этот язык действительно легко, но концептуально язык оказался крайне бедным, так что перейти после него к какому-либо другому языку или просто научиться программировать в хорошем стиле крайне тяжело.

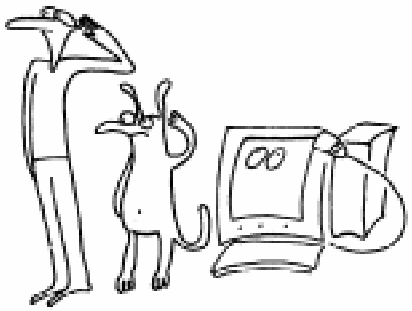
С такими механизмами Бейсика, как метки и безусловные передачи управления, непосредственно связано и антипедагогические применения блок-схем, воспеваемых их сторонниками как проявление наглядности схем программ. А между тем блок-схемы наглядны только до тех пор, пока схема программы размещается на одном листочке. Недаром в таком известном центре работы с детьми, как районная школа юных программистов Новосибирского Академгородка, еще в конце 70-х – начале 80-х за четырехлетний цикл обучения учащиеся не рисовали ни единой блок-схемы, овладевая, тем не менее, многими языковыми системами и становясь впоследствии высокообразованными профессионалами-программистами. При этом речь идет не о педагогическом фокусе или отдельном исключительном эксперименте, а о научно обоснованной методике преподавания современного школьного курса информатики, которая благодаря усилиям А.П. Ершова стала основой национальной программы информатизации отечественной школы.



2. Программное управление исполнителем как методический прием обучения основам программирования.

Когда встал вопрос о подходах к проектированию отечественных курсов информатики в школе, то все ведущие группы российских разработчиков – информатика для старшеклассников в группе Московского университета, курс информатики для центральных классов средней школы в екатеринбургской группе, курс раннего обучения информатике в Роботландии (Переславль-Залесский) – с вполне объяснимым единодушием выбрали в качестве стратегической основы не высокоуровневые языковые системы учебных языков программирования, а системы программных исполнителей.

В учебнике для школьника третьего класса исполнитель определяется так: «Исполнителем называется человек, коллектив, животное или техническое устройство, которые понимают и умеют очень точно исполнять задаваемые им команды.»



В приведенном определении умышленно опущено соотношение между исполнителем и компьютерной программой. В этом скрыт определенный умысел: позднее, познакомившись с разнообразным миром исполнителей (главным образом, благодаря их многочисленным программным реализациям), школьники отчетливо осознают, что программный исполнитель (один их частных, но постоянно используемых на уроках информатики типов исполнителей) – это представленная на компьютерном экране модель реального или воображаемого устройства, живого существа,

процесса, мыслительной схемы. Требования к учебному программному обеспечению, вытекают из главной цели школьного информатического образования – формирования стиля мышления у молодого поколения. Логическое следствие этих требований – ориентация на систему программных исполнителей, каждый из которых предназначается для формирования конкретного навыка. Каждая из систем учебного программного обеспечения в зависимости от предметной области (в первую очередь, от ориентации на возрастной контингент учащихся) имеет разное количество исполнителей. Естественно, что наибольшим числом учебных программных исполнителей обладает курс раннего обучения информатике. С позиций дидактики предпочтение системам программных исполнителей перед языковыми системами было отдано не случайно:

– современный ребенок, приходя в школу, уже имеет интуитивное представление о роботах, компьютерах и их возможностях; однако между этим представлением и систематическим изучением языковых средств информатики существует ощутимая ниша, которую целесообразно заполнить средствами, доступными младшим школьникам – программными и аппаратными исполнителями-роботами;

– операционная система, созданная языковой системой программирования, даже такой богатой, как Лого, будучи единственной, приходит в противоречие с дидактическим принципом многообразия форм обучения; многообразие же операционных средств, порождаемых системой исполнителей, практически неисчерпаемо;

– так как языковая учебная среда создает у учащегося противоестественное ощущение первичности программы и вторичности алгоритма, полезно построить систему программных исполнителей, которые помогают школьнику понять сущность и свойства алгоритмов раньше, чем он освоит языки программирования;

– разнообразный мир исполнителей конструктивно демонстрирует межпредметные связи информатики и, следовательно, готовит школьника к активной жизни в новом информационном обществе;

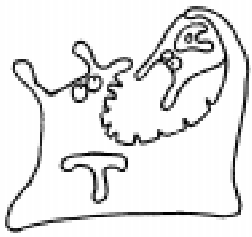
– специализированные исполнители, ориентированные на задачи художественного, эстетического и музыкального воспитания, могут служить целям гуманизации образования с большей эффективностью, нежели универсальные языковые средства.

Существует, впрочем, и иной поход к обоснованию системы программных исполнителей в качестве дидактического инструмента информатического образования. Этот подход базируется на концепциях сложившегося в последнее десятилетие объектно-ориентированного программирования, продолжающего отслеживать параллели между структурами программирования и компонентами алгоритмизуемой умственной деятельности. С позиций объектно-ориентированного программирования его базовое понятие – объект – представляет собою выкристаллизованное представление исполнителя.

Кроме своей педагогической направленности на формирование конкретных умений и навыков учащихся, программный исполнитель как дидактический инструмент должен обладать свойством наглядности. Записывая программу на любом языке программирования, мы записываем ее для некоторого исполнителя или строим свой исполнитель. Но для обучения младших школьников чисто информационные исполнители не наглядны, поэтому в начальной школе столь важны исполнители с наглядной средой, адекватной детским представлениям и детскому уровню сформированности абстракций.

В последнее время достаточно часто звучат слова о необходимости превалирующей роли «технологического» компонента в современном информатическом образовании. При этом приводятся аргументы: возросшая роль информационных технологий и таких инструментов информационного общества, как базы данных, электронные таблицы, редакторы, издательские системы, средства электронных коммуникаций, делают необходимым изу-

чение этих средств на школьном уровне, а при общем дефиците учебного времени лучше пожертвовать теоретическими разделами информатики. Алгоритмика приносится в жертву технологии. Однако важнейшей целью школьного курса информатики (или любой другой реализации информатического образования молодого поколения) является социальная задача формирования стиля мышления учащихся, адекватного требованиям современного информационного общества. С этой точки зрения, в работах основоположников отечественной школьной информатики было показано, что только информатика (и именно информатика!) способна решить эту социальную задачу, поскольку информатика – это единственная из научных дисциплин, представленных совокупностью школьных предметов, которая располагает необходимым для этой цели концептуальным запасом – набором понятий, представлений, механизмов и идей, используемых при формировании стиля мышления. Среди этих понятий выделяются своей значимостью управляющие структуры и структуры данных: с помощью управляющих структур формируются такие важнейшие компоненты стиля мышления, как умение планировать структуру своей деятельности, умение искать информацию, необходимую для решения поставленной задачи, навыки инструментирования деятельности.



В информатике показано, что всякая алгоритмизуемая мыслительная деятельность может быть описана совокупностью базовых управляющих структур – последовательности, ветвления и цикла, а также таких алгоритмических структур, как процедура и функция. Каждая из таких структур в информатике представляется как фундаментальная, теоретическая структура. Практические же, реализационные аспекты этих важнейших (и достаточно сложных!) понятий в наиболее чистом виде «материализуются» в конструкциях языков программирования.

Хотя пропедевтика всех этих понятий предусмотрена при изучении отдельных тем в курсе раннего обучения информатике и реализуется в ходе знакомства с программными исполнителями (линейные программы Машиниста и Раскрашки, ветвления Конструктора Сказок и Угадайки, повторяющиеся операции в адаптированных редакторах), тем не менее, на определенном этапе обучения становится актуальным переход от конкретных реализаций исполнителей к обобщенным категориям языков программирования. Поэтому, если курс информатического образования претендует на то, чтобы сохранить представление об информатике как о мировоззренческой научной дисциплине, он должен содержать в своем составе раздел, рассказывающий об элементах программирования средствами языковых систем высокого уровня. Появление учебного языка программирования в курсе раннего обучения информатике, использующего понятие программного исполнителя в качестве методического фундамента, имеет совершенно естественную мотивацию. Роль компьютера в роли посредника между человеком, задумавшим алгоритм, и исполнителем, выполняющим этот алгоритм в виде последовательности команд из системы команд исполнителя (СКИ), может быть продемонстрирована на примере, вообще говоря, любого исполнителя. Из дидактических соображений – наглядное представление на экране, способность к перемещениям, дискретность движений, простота и выразительность системы команд – в разных учебных системах в качестве такого исполнителя используются программы, моделирующие перемещение некоторого робота по клетчатому полю (Чертежник и Робот в КуМире, Кенгуренок у екатеринбуржцев, Муравей в Робике, Кукарача – в Роботландии).

Уже при демонстрации простого перемещения исполнителя на несколько шагов в одном направлении достаточно

наглядно стимулируется введение языка, необходимого для управления таким роботом. Действительно, записывать последовательность команд

ВПРАВО ВПРАВО ВПРАВО
ВПРАВО ВПРАВО ВПРАВО (1)

достаточно трудоемко. Однако школьник, знакомый к этому времени с возможностями компьютера и, в частности, текстового редактора, умеющего анализировать тексты, легко соглашается с тем, что компьютеру можно дать команду

ПОВТОРИ 6 ВПРАВО (2)

которую тот легко преобразует в запись (1) перед тем, как передавать ее исполнителю.

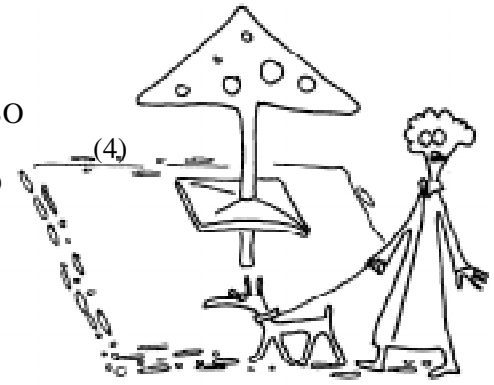
Тогда становится возможным командовать исполнителем, которому потребуется выполнить и более сложные перемещения:

ПОВТОРИ 6 ВПРАВО
ПОВТОРИ 6 ВНИЗ
ПОВТОРИ 6 ВЛЕВО (3)
ПОВТОРИ 6 ВВЕРХ

Идею поручить компьютеру «развертывание» последовательностей команд (2) или (3) можно завершить только после того, как будет согласована еще одна условность: каким образом отмечаются начало и конец таких последовательностей. Предлагается (например) в начале последовательности записывать слова ЭТО <имя процедуры>, а в конце – слово КОНЕЦ (ср. с Лого):

ЭТО Прогулка
ПОВТОРИ 6 ВПРАВО
ПОВТОРИ 6 ВНИЗ
ПОВТОРИ 6 ВЛЕВО
ПОВТОРИ 6 ВВЕРХ
КОНЕЦ

(4)



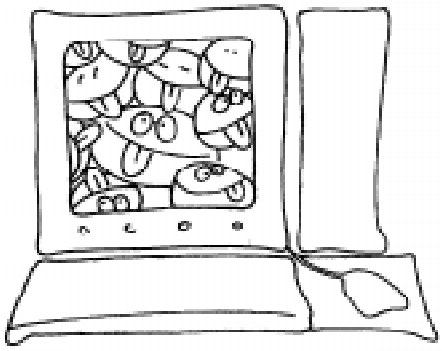
Тем самым определяется понятие процедуры вспомогательного алгоритма и ее имени. Имя процедуры необходимо для обращения к описанию (4), то есть для выполнения описанных в процедуре команд. Понятие процедуры и роль компьютера-посредника приводят к тому, что исполнитель становится «обучаемым»: пользователь может с помощью компьютера и языка, описывающего выражения типа (2) и (4), задавать исполнителю команды, исходно отсутствовавшие в СКИ, и управлять исполнителем в режиме, который называют режимом программного (в отличие от непосредственного) выполнения.

Далее, столь же просто и естественно вводятся понятия вложенных процедур и рекурсий, ветвлений. Так появляется язык управления исполнителем. Понятно, что возможности такого языка в значительной степени определяются средой и СКИ исполнителя, тем не менее главной его особенностью является несомненно тот факт, что построенный таким образом язык есть язык программирования, с помощью которого могут быть изучены основные понятия программирования как научной дисциплины.

3. Учебные языковые системы программирования и их место в общеобразовательном курсе информатики. Методическая цепочка учебных языков программирования.

Исходя из требований педагогической предметной области, языковая система программирования, предназначаемая для изучения элементов программирования в рамках школьного образования, должна быть специализированно учебной. Каждый из языков программирования в известной степени универсален, поэтому объясним тот факт, что многим «моноязычным» программистам свойствен консер-

ватизм мышления: умея запрограммировать задачу на одном из языков, они не имеют стимула к переходу на другой язык. Такой консерватизм не вписывается в главную задачу школьной информатики, состоящую в формировании операционного стиля мышления учащихся. Кроме того, эффективность решения задачи (и квалификация программиста тоже) достигается не только (и не столько) программированием алгоритма на некотором языке, но также выбором (а иногда и обоснованием такого выбора) языка программирования, адекватного требованиям соответствующей предметной области. Поэтому в методическом плане исключительно важно, чтобы в начале освоения программирования (в том числе, и в общеобразовательном учреждении) учащимся было показано многоязычие в мире программирования и аргументированно обоснована актуальность такого многоязычия.



Впервые идея многоязычия в обучении программированию была декларирована, обоснована и реализована в группе академика А.П. Ершова в виде методической цепочки «Робик-Рапира». Робик – язык начального обучения, использовавшийся для управления простыми программными исполнителями (Дежурик, Маляр, Муравей...), а Рапира – учебно-ориентированный язык с развитыми управляющими структурами и структурами данных. Эта цепочка была реализована в учебном процессе Районной школы юных программистов в новосибирском Академгородке, через которую прошли десятки младших школьников. Она явилась конструктивным доказательством правильности идеи о многоязычии в обучении программированию. Завершалась цепочка веером профессиональных языков, связываемых с задачами из разных предметных областей. Характерные особенности «учебной» цепочки языков:

– максимальная простота начального языка обучения, первого в цепочке;

– близость (концептуальная совместимость) завершающего языка цепочки к профессиональным языкам (возможно, к одному из классов типовых профессиональных языков);

– наличие ситуации, методически стимулирующей переход от одного (предшествующего) языка к другому (последующему); например, потребность в понятии переменной, к которой учащиеся подведены учителем в предшествующем языке цепочки, но не могут быть в нем освоены.

Поэтому к перечисленным выше особенностям языковой цепочки общеобразовательного информатического образования должны быть добавлены еще несколько принципиальных требований:

– язык, начинающий цепочку, должен непосредственно опираться на основные понятия предшествующих разделов, в частности, на понятие программного исполнителя и круг связанных с ним идей;

– язык, завершающий цепочку, должен иметь средства для описания объектов из изучаемых позднее тем: баз данных, электронных таблиц, электронных моделей, издательских систем, коммуникационных средств.

В связи с этими условиями, появляется возможность говорить о методической цепочке учебных языков программирования в непрерывном курсе информатического образования более конкретно. На роль начального языка в обсуждаемых конкретных условиях может претендовать только язык, подобный языку управления исполнителем Кукарача, который описывается в последнем, завершающем разделе программно-методической системы Роботландия. Наглядность среды, единство интерфейса Кукарачи и множества дру-

гих изученных ранее исполнителей Роботландии, непосредственное использование инструментальных средств Роботландии при редактировании программ – все это делает язык управления Кукарачей реальным претендентом в начало обсуждаемой цепочки. Например, программы Роботландии практически полностью включаются в предшествующие «Элементам программирования» разделы такого школьного курса информатического образования, как «Информационная культура».

С другой стороны, конец цепочки столь же определен. В качестве средства описания информационных систем и информационных моделей в одной из систем программного обеспечения курса информатики в старших классах использован учебный язык КуМир (выросший из Е-языка, названного так его разработчиками в память об А.П. Ершове, который был автором концепций этого учебного языка). Опыт использования КуМира в сквозном непрерывном курсе «Информационная культура» позволяет совершенно естественно включить его в завершающее звено раздела «Элементы программирования».

Совместному описанию языков управления Кукарачей и Кумира в одном разделе-модуле курса способствует и близость интерфейсов обеих языковых систем: среда Кукарачи – клетчатое поле; в таком же клетчатом поле работают первые исполнители Кумира – Робот и Чертежник; наборы используемых в языках команд достаточно близки не только содержательно, но и по форме. Более того, это обстоятельство позволяет сделать пару «Кукарача-Кумир» не только минимальной, но и методически оптимальной цепочкой учебных языков. Объединение таких двух языков в одном модуле превращает совокупность учебных программных средств, ориентированных на разные возрасты обучаемых, в методически единую систему непрерывного информатического образования.

4. Относительная позиция темы «Элементы программирования» в курсе непрерывного информатического образования.

В курсе раннего обучения информатике («Роботландия») и в позднее появившемся непрерывном курсе школьного информатического образования «Информационная культура» декларируется, что программирование не должно занимать значительную (по времени) часть курса. Действительно, многие педагоги-информатики обоснованно считают, что программирование в общеобразовательной школе не должно сводиться к трудоемкому ремеслу написания простых или сложных программ на том или ином языке, а, скорее, демонстрировать конкретное воплощение тех фундаментальных понятий информатики, которые необходимы в процессе формирования у школьников умений и навыков операционного стиля мышления. Не случайно разработчики Роботландии, сами непосредственно участвовавшие в педагогической работе по внедрению этой системы, неоднократно отмечали, что первое осознанное употребление слова «программа» дети делают в последней четверти курса, существенно ближе к его завершению, чем к началу. Такая «демонстрационная» роль элементов программирования в общеобразовательном курсе информатики неизбежно требует относительно небольшого (по времени) объема с учетом того, что школьники за пределами этого раздела смогли:

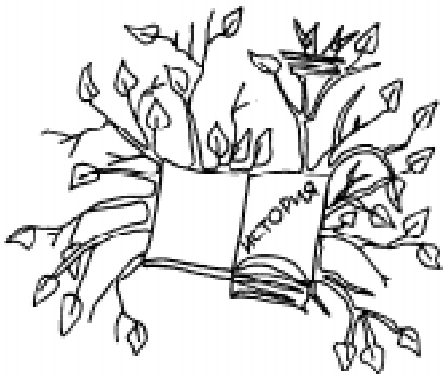
– во-первых, глубоко изучить фундаментальные понятия информатики, не лежащие непосредственно в разделе «Элементы программирования» – информация, виды ее представления, информационные процессы, алгоритмы, исполнители;

– во-вторых, познакомиться с многообразием технологических приложений информатики, требующих развитых приклад-

ных навыков операционного стиля мышления, которые формируются позднее, после изучения элементов программирования.

Это обстоятельство не представляется ограничением, если изучению раздела «Элементы программирования» будет предшествовать достаточно серьезное освоение фундаментальных понятий информатики. С этой точки зрения, выделение разделу «Элементы программирования» одного модуля (то есть, по существу, одного учебного года из десяти) представляется методически оправданным.

Еще на этапе проектирования Роботландии место элементов программирования на базе языка управления Кукарачей было достаточно четко определено: этот раздел был последним в курсе раннего обучения информатике, занимая в нем кульминационное, последнее (по времени) место. Этот язык следовал после понятий алгоритма, обширной темы исполнителей, непосредственно вслед за еще более объемной темой информационных редакторов. Во-первых, для эффективного освоения элементов программирования оказывались необходимыми знания, умения и навыки, сформированные в предшествовавших разделах; некоторые из этих более ранних разделов имели среди своих педагогических целей пропедевтику понятий, необходимых при изучении программирования. Например, ветвящиеся истории, созданные в Конструкторе Сказок, и «программы» управления исполнителем Угадайка могли быть использованы в качестве пропедевтики сложной управляющей структуры ветвления, а работа на оборотной, «программной» странице графического редактора-конструктора Раскрашка служила пропедевтикой понятия линейной программы. Во-вторых, непосредственное следование Кукарачи за другими программами Роботландии могло быть использовано для обобщения многих важных теоретических и практических понятий и механизмов информатики – алгоритм, система команд, среда, файл, редактирование и т.д.



Таким образом, относительное (по отношению к другим разделам курса) расположение элементов программирования следует считать не специфической особенностью конкретного курса раннего обучения информатике (Роботландии), а инвариантом раннего информатического образования. Следовательно, такое же относительное расположение должно сохраняться и в других реализациях раннего (или непрерывного) информатического образования.

5. Место темы «Элементы программирования» в курсе непрерывного информатического образования.

В силу вышеуказанных посылок нетрудно сделать вывод об абсолютном положении обсуждаемого раздела в сквозном школьном (1-11) курсе непрерывного информатического образования. Поскольку такие курсы существуют (пример – «Информационная культура»), то приводимые здесь положения могут рассматриваться не только в качестве некоторых теоретических суждений, но и (обсуждаемых) практических рекомендаций. С одной стороны, разделы, непосредственно предшествующие элементам программирования (информационные редакторы, точнее первый виток дидактической спирали в школьном знакомстве с информационными технологиями), завершаются Модулем 6 (шестой класс средней школы). С другой стороны, нет возможности (и необходимости) опускать последующие модули курса (информационные хранили-

ща, кодирование информации, информационные модели, информационные технологии) ниже по возрастной шкале. Потребность в языке достаточно высокого уровня, необходимом для их изложения, заставляет начинать последовательность этих разделов с восьмого класса.

Таким образом, абсолютное место для раздела «Элементы программирования» в непрерывном курсе информатического образования определяется совершенно точно – седьмой класс.

Предложенным здесь методическим и логическим аргументам может противопоставляться один контр-тезис. В исторически первой (и, по существу, пока единственной) программно-методической системе начального информатического образования – Роботландии, представлявшей двухлетний курс раннего обучения информатике, Кукарача размещался во втором учебном году (при двух часах в неделю). Таким образом, когда Роботландия начиналась в третьем классе (типовая рекомендуемая схема), то исполнитель Кукарача изучался в четвертом (или пятом) классе (в зависимости от системы начальной школы 1-3 или 1-4).

Не отдавая такого рода «практическим» аргументам приоритетного значения, полезно, тем не менее, обсудить выводы из упомянутого практического опыта.

Трудность раздела для учащихся четвертого класса приводила к тому, что Кукарача занимал в учебном плане несравнимый с другими разделами объем – практически целый учебный год. Это обстоятельство оказывалось в противоречии с одним из главных дидактических принципов, положенных в основу выбора системы программных исполнителей в качестве определяющего типа учебного программного обеспечения – принципом многообразия учебных сред. Сделать Кукарачу более «динамичным» было трудно, в первую очередь, в силу возрастных особенностей учащихся. Во многих темах Кукарачи (например, в цикле ПОКА или в рекурсиях) требовалось умение подняться на такой уровень абстракции, который оказывался выше возможностей большинства четвероклассников. Обстоятельства, связанные с рекомендацией включить Кукарачу в качестве дидактического инструмента в Модуль 7 класса непрерывного информатического образования в качестве одного из компонентов раздела «Элементы программирования», являются предпосылкой к преодолению отмеченных выше трудностей, которые были связаны (как теперь видно) с неоправданно ранним началом изучения программирования.

Леонов Александр Георгиевич,
кандидат физ.-мат. наук,
доцент МГУ им. М.В. Ломоносова,
директор предприятия «Инфомир».

Первин Юрий Абрамович,
доктор педагогических наук,
директор фирмы «Роботландия».