УДК 004

А. Г. Леонов, Ю. А. Первин

Переход от непосредственного управления исполнителями к составлению программ в пропедевтическом курсе информатики

В статье рассмотрена структура пропедевтической части непрерывного курса школьной информатики и обоснованы его методические этапы: непосредственное управление программными исполнителями, построение алгоритмов управления исполнителями, умозрительное представление алгоритма, элементы программного управления и учебные языковые системы программирования. Выделены специфические особенности учебно-ориентированной системы КуМир.

Ключевые слова: пропедевтический курс, исполнитель, алгоритм, процедура, язык программирования, система программирования, КуМир.

A. G. Leonov, Ju. A. Pervin

Transition from Direct Management by Executors to Making Up of Programmes in the Propaedeutic Course of Informatics

In the article the structure of the propaedeutic part of a non-stop course of school Informatics is regarded and its methodical stages are proved: direct management of programme executors, creation of management algorithms by executors, conceptual representation of the algorithm, elements of the programme control and educational language systems of programming. Specific features of the educational focused system KuMir are allocated.

Keywords: a propaedeutic course, an executor, algorithm, procedure, a programming language, a programming system, KuMir.

Младший школьник (и уж тем более дошкольник) воспринимает окружающий мир как систему конкретных объектов. Генерируемые в его сознании абстракции реальных объектов – это пока еще перспектива его начинающейся школьной жизни. Особенно сложным этапом становления личности является уровень абстракции процессов – действий, выполняемых наблюдаемых и ощущаемых объектов: ребенку трудно построить в своем сознании не сиюминутные действия объектов, а их предстоящие действия, то есть представить умозрительно последовательность воображаемых, планируемых операций.

Стратегическим подходом к формированию у детей младшего возраста умений описывать программируемые действия объектов виделся в использовании учебных роботов-игрушек. Как это ни парадоксально, но во второй половине 80-х годов прошлого века, в самом начале образовательной реформы 1985 года, рядовому учителю информатики достаточно было по дороге в школу зайти в государственный магазин «Детский мир», чтобы на собственную зарплату купить для уроков в своем классе несколько программно-управляемых отечественных игрушек – гусеничных вездеходов, на «приборной панели» которых можно было нажимать сенсорные клавиши с выразительными пиктограммами. Школьник, понимая назначение клавиш и примерно оценивая длину пути и угол поворота управляемой им игрушки (то есть, по существу, – написать программу предстоящего перемещения робота), демонстрировал способность решить конкретную задачу, например, дойти до табуретки, поставленной у дальней стены спортивного зала, и вернуться обратно.

Не менее содержательные задачи возникали перед школьниками в детском летнем компьютерном лагере, куда инженеры-конструкторы привозили таких роботов, в качестве манипуляторов «механическая рука» или модель подъемного крана, загружающего товарные вагоны игрушечной железной дороги.

Экономические и политические реалии сегодняшнего информационного общества, к сожалению, исключают эту перспективу педагогической, инструментально-дидактической «роботизации» общеобразовательной начальной школы, хотя сфера дополнительного образования, представляемая кружками и секциями домов детского и юношеского творчества, и сейчас эффективно использует педаго-

гические возможности учебного роботостроения в компьютерном управлении роботами, которые конструируются детьми [1, 2]. Надо понимать, однако, что концептуальные решения, полученные в системе дополнительного образования, при всей их дидактической значимости могут быть использованы в общей школе самостоятельно, а только после серьезного и глубокого их анализа.

Впрочем, и сегодня интерес педагогов-исследователей к программируемым игрушкам-роботам не иссякает ни в нашей стране, ни за рубежом. В 2012/13 учебном году на Большом Московском семинаре по методике информатизации начального и дошкольного образования был поставлен доклад об успешном опыте группы разработчиков НИИ СИ РАН в детском саду, являющемся апробационной площадкой института [9] по освоению ПиктоМира. Представляет интерес эксперимент греческих исследователей с детьми 4–6 лет, которые не только организовали занятия детей с программируемой игрушкой-роботом «Пчелкой», но и количественно измерили затруднения детей при воображении реальных действий в ходе записи еще не выполненных операций [6].

Первые попытки привлечь учащихся общеобразовательных школ к программированию восходят в нашей стране к 60-м и 70-м годам прошлого века. Они включали как многочисленные методические варианты практики обучения программирования, начиная с московской школы № 444 (школа С. И. Шварцбурда (1918–1996)) [3], так и технологические обобщения чл.-корр. РАО В. М. Монахова [12]. Однако во всех этих попытках, даже в тех случаях, когда ставилась задачи конструирования специализированных языков программирования, ориентированных на обучение школьников [15], проблемы соотношения исполнителей и программирования не обсуждались: объект их исследования сходился в точке пересечения интересов информатики, программирования и психологии.

Острота проблемы была осознана, когда, с одной стороны, начал свое шествие по миру предложенный Сеймуром Пейпертом язык Лого, а с другой стороны, в нашей стране появились системы программных исполнителей, обоснованно претендующие на роль платформы для проектирования системы обучения информатике.

Первые шаги в обучении программированию ребенок делает на экране, разделенном на две неравные части. Он управляет главным исполнителем программы – Черепашкой. Верхняя часть экрана – *рабочее поле*, на котором перемещается и создает свои рисунки Черепашка, а несколько нижних строк экрана образуют *поле ввода команд*. Свои первые действия по управлению Черепашкой ребенок выполняет, задавая ей команды, которые он набирает на компьютерной клавиатуре. Например, команда ВПЕРЕД 60 дает команду Черепашке пройти «вперед» (в ту сторону, куда направлена ее голова), заданное в команде число 60 – это количество черепашьих шагов (в исходном состоянии голова Черепашки направлена в верхнюю часть экрана). По другой команде – НАПРАВО 90 – Черепашка никуда не идет. Стоя на месте, она поворачивается вокруг собственной оси на 90 градусов. Числа, указанные в двух приведенных примерах, являются *параметрами,* которые задаются вместе с командой.

У Черепашки есть инструмент – «перо». Он оставляет след при перемещении, если предварительно была подана команда ПО (ПЕРО ОПУСТИ), или не оставляет его, если перед этим была подана команда ПП (ПОДНИМИ ПЕРО). Теперь уже ясно, что Черепашку можно попросить не просто перемещаться и поворачиваться в рабочем поле, но и, оставляя следы, рисовать любые (в том числе достаточно сложные) рисунки. Последовательность команд, задаваемых ей, является *программой.*

Когда завершены все предусмотренные учителем (или придуманные учеником) перемещения Черепашки, на рабочем столе остается рисунок из нескольких отрезков – черепаший след. Можно сохранить этот завершенный на уроке *проект,* например, для того, чтобы на следующем уроке добавить к нему еще какие-либо детали.

После того, как ребенок с помощью программы сделал первый черепаший рисунок (и, возможно, сохранил проект), происходит обсуждение следующего задания – нарисовать квадрат со стороной длиною в 40 черепашьих шагов.

– Давайте обсудим *план* предстоящей работы, – предлагает учитель.

В ходе коллективных усилий (возможно, отбираемых учителем из детских предложений), учитель, фиксируя шаг за шагом эти предложения, воспроизводит в строке ввода команд следующие команды (поскольку для большинства команд в Лого допускаются двухбуквенные сокращения, все команды вмещаются в одну строку):

ВП 40 ПР 90 ВП 40 ПР 90 ВП 40 ПР 90 ВП 40 ПР 90

Эти восемь команд записаны учителем в одной строке учительского экрана и выведены на проектор школьного класса.

– А теперь смотрите: я нажимаю клавишу ввода (Enter).

– Ах!

Это Черепашка мгновенно нарисовала запланированный рисунок – выполнила программу, составленную из восьми команд.

Важный в этом обсуждении вопрос учителя:

– А как будем рисовать квадрат на следующем уроке?

Лежащая на поверхности идея:

– Выполним такие же действия.

учителем отвергается сразу:

– Но на следующем уроке нам надо будет рисовать такой многоэтажный дом. – И он показывает на плакате рисунок: квадратный фасад трехэтажного дома, на каждом этаже которого – по три квадратных окошка.

Рисунок убеждает детей: рисовать придется не один, а 10 квадратов! Но с точки зрения методики информатики существует еще более сильный аргумент: сохранение проекта оставляет в памяти *результат* программы – выполненный ею рисунок, тогда как здесь речь идет о сохранении *действия.* Так становится актуальным определение *процедуры* как *программы, имеющей имя и хранящейся в памяти, из которой она может быть вызвана по имени для выполнения*. По сути, речь идет о записи формального плана решения задачи – *алгоритма*. Правда, здесь пока еще не предполагается, что дети знают, что такое «алгоритм».

Ясно содержание следующего урока – описание и вызов процедуры. Описание процедуры – это перечень ее команд, окаймленный заголовком – ЭТО <имя процедуры> и окончанием – КОНЕЦ. В Лого можно перевернуть страницу рабочего поля на оборотную сторону, наизнанку для того, чтобы на ней записать и сохранить описание процедуры:

```
ЭТО КВАДРАТ
        ВП 40 ПР 90 ВП 40 ПР 90 ВП 40 ПР 90 ВП 40 ПР 90
КОНЕЦ
```

или (что выразительнее, с точки зрения методики информатики)

```
ЭТО КВАДРАТ
        ВП 40 ПР 90
        ВП 40 ПР 90
        ВП 40 ПР 90
        ВП 40 ПР 90
КОНЕЦ
```

Возвращение с изнанки листа в экран рабочего поля автоматически сохраняет процедуру в памяти, переводя вызов процедуры в статус команды языка. Лого лишь формально различает *вызовы* процедур и *примитивы* – команды, составляющие определение языка.

Здесь важно обратить внимание, что в суждениях о структурах управления процессами в Лого первичным понятием является программа, а вторичным – алгоритм. Такую структуру введения понятий можно называть *индуктивной*. В обучении информатике с помощью языковых систем программирования, имеющих учебную ориентацию, индуктивные структуры используются широко. В частности, это относится и к разным версиям Лого-систем.

Задача педагога (посильная для учителя начальной школы), показав технологию работы с процедурой, объяснив ее описание и вызов, – сделать ее использование непременной частью всех последующих уроков с младшими школьниками. Актуальность такой задачи довелось наблюдать сравнительно недавно (2012/13 учебный год) в эксперименте (априорно не планировавшемся) с группой студентов на факультете информационных технологий (!) одного из университетов. Профессор сделал на лекции представление языка и даже успел после обсуждения примеров работы в поле ввода команд показать механизм описания и вызова процедур, а в начале непосредственно следовавшей лабораторной работы попросил студентов самостоятельно выполнить задание чуть сложнее трехэтажного дома (добавилась только трапеция крыши). Каково же было его удивление, когда он увидел, что студенты работают «вручную», не используя описание процедуры КВАДРАТ, а набирая примитивы в поле ввода команд!

С отношением к фундаментальному понятию процедуры связан один из многочисленных стереотипов, сопровождающих историю школьной информатики, – беспроцедурный язык программирования Бейсик. До сегодняшнего дня он еще почитается некомпетентными методистами как популярное средство обучения в школе. Аргументация этого стереотипа – простота изучения и дешевизна программного обеспечения – сыграли с Бейсиком злую шутку в попытках оправдать его порочную в методическом (и в методологическом!) отношении позицию[14].

Другое направление в методике информатики – *дедуктивные* структуры введения понятий программы и алгоритма, когда алгоритм первичен, а программа, являющаяся компьютерной реализацией алгоритма, вторична. Такой методический подход оправдан в системе программных исполнителей. Он обеспечивает математизацию общего пропедевтического курса информатики, столь необходимую современной школе. Каждый исполнитель при этом ориентируется на формирование конкретных умений и навыков операционного (алгоритмического) стиля мышления, а из исполнителей извлекаются методические инварианты, которые используются при проектировании языка управления роботами, образующие цепочку учебных программных систем непрерывного школьного курса информатики:

программные исполнители →
язык управления исполнителями →
учебный язык программирования →
учебно-ориентированный язык →
профессионально-ориентированные языки →

Эти программные средства многократно описывались в отечественной методической литературе от [5] до [4] и [13]. В работах Г. А. Звенигородского построен язык управления роботами Робик и учебно-ориентированный Рапира.

Наиболее близкой к образовательной реформе 1985 года как по хронологии, так и по концепциям стала известная программно-методическая система КуМир (Комплект учебных Миров или Миры Кушниренко) – язык и система программирования, предназначенная для поддержки начальных курсов информатики и программирования в средней и высшей школе. Эта система стала первой среди тех, которые поддержали качественными программными разработками методику, предложенную во второй половине 1980-х годов академиком А. П. Ершовым и его командой. Эта методика широко использовалась в средних школах СССР, и в частности России. Система КуМир использует созданный А. П. Ершовым школьный алгоритмический язык – простой алголоподобный язык с русской лексикой и встроенными командами управления программными исполнителями (Робот, Чертежник).

КуМир – практикум по основам алгоритмизации, реализованный в СССР и в России с 90-х годах прошлого столетия, – до сих пор не забыт системой школьного образования России, в первую очередь благодаря тому, что целое семейство КуМиров было реализовано и эксплуатировалось практически на всех компьютерах, попавших в школы СССР, и в различных операционных системах. В 1995 году КуМир был рекомендован Министерством образования в качестве основного учебного материала по курсу «Основы информатики и вычислительной техники» на базе ставших самыми массовыми учебниками школьной информатики А. Г. Кушниренко, Г. В. Лебедева (1957–2004) и Р. А. Свореня [7] и А. Г. Кушниренко, Г. В. Лебедева, Я. Н. Зайдельмана [8]. Более того, в настоящее время ведется разработка и педагогическое внедрение новой версии КуМира в Linux и Windows. Эта разработка выполнена А. Г. Кушниренко и А. Г. Леоновым [10].

Распространение учебной системы на столь мало похожие друг на друга компьютерные системы потребовало создания не только определенной разработки дисциплины, но и единого интерфейса с обучаемыми. После детального изучения эргономических характеристик различных компьютерных систем был придуман и стандартизован единый интерфейс. Клавиши и функционал КуМира описывались для некоторого виртуального компьютера. Эти виртуальные клавиши на виртуальной клавиатуре использовались при описании функций КуМира в документации, но имели разное расположение на различных реальных клавиатурах.

Такой подход позволил не только иметь единый комплект документации и методических материалов, но и упрощал для ученика, учителя и методиста переход с одного школьного компьютера на другой. Другие практикумы, разрабатываемые тем же коллективом, в том числе и текстовый редактор МикроМир, имели аналогичный интерфейс. Простые практикумы («Резчик металла» – практикум по составлению линейных программ, клавиатурный тренажер, «Стековый калькулятор») пред-

варяли работу учащихся с КуМиром, попутно снимая непроизводительные расходы, связанные с изучением клавиатуры и команд: команды во всех практикумах были идентичны.

Другой причиной популярности КуМира была востребованность методики информатики. К моменту появления на свет Е-практикума (предшественника Кумира) во всех старших классах СССР преподавался так называемый безмашинный курс информатики, и параллельно в школы и педвузы стала поступать вычислительная техника. Так что, как это ни парадоксально, основной задачей КуМира была программная поддержка на школьных машинах безмашинного курса информатики.

Третьей причиной успешности КуМира было использование уже проверенной к тому времени на мехмате МГУ идеи редактирования, компиляции и метафоры «полей программы», которые дают огромное повышение производительности труда обучаемого во вводном курсе программирования.

Будучи системой с дедуктивной структурой перехода от непосредственного управления к программному, КуМир позволяет легко (одним щелчком мыши) переходить из режима *пошагового* выполнения команд в *программный* режим и обратно, демпфируя тем самым потенциально возможные когнитивные трудности для учащихся. Следует отметить, что из многочисленных версий Лого только ПервоЛого (система с самым юным контингентом пользователей-учеников) допускает пошаговый режим.

К 2005–2006 годам в НИИСИ РАН по заказу Российской академии наук был начат проект по созданию новой многоплатформенной системы КуМир. К концу 2008 года такая система была разработана. Она распространяется свободно на условиях лицензии GNU 2.0. Эта лицензия разрешает бессрочно использовать КуМир на любом количестве компьютеров в любых целях без оформления каких-либо дополнительных документов. В обновленной системе КуМир используется школьный алгоритмический язык с русской лексикой и встроенными исполнителями Робот и Чертежник.

При вводе программы КуМир осуществляет постоянный полный контроль ее правильности, сообщая на полях программы об обнаруженных ошибках.

При выполнении программы в пошаговом режиме КуМир выводит на поля результаты операций присваивания и значения логических выражений. Это позволяет ускорить процесс освоения азов программирования. Новый КуМир полностью поддерживает изданные учебники: все алгоритмы из этих учебников и решения всех задач выполнимы в КуМире.

Он предлагает новые возможности по работе с исполнителями.

1. Три исполнителя – Робот, Чертежник, Файловая система – жестко встроены в систему КуМир. Для того чтобы активизировать соответствующий встроенный исполнитель, нужно в начале КуМир-программы добавить команду *использовать* и имя нужного исполнителя.

Номенклатура этих исполнителей определена заранее. Расширение списка встроенных исполнителей возможно только путем изменения исходного кода.

2. Для подключения исполнителей «сторонних» производителей, написанных, например, на языке С, создан дополнительный механизм подключения таких исполнителей. За счет этого КуМир получает возможность управлять не только программными исполнителями, но и реальными, например, «настоящим» роботом LEGO Mindstorm, используя канал bluetooth.

3. В КуМире пользователь может создавать собственные исполнители.

Для этого в языке предусмотрена конструкция *исп – кон исп*.

КуМир-исполнители могут быть расположены либо в файле вместе с основной программой, либо в отдельном файле. Они доступны редактированию и выполнению в пошаговом режиме, как и обычная КуМир-программа. Таким образом, КуМир предоставляет базу для перехода от изучения основ объектно-ориентированного программирования в учебной среде к изучению производственных языков ООП. Той же цели служит и препроцессор КуМир → C++: написав и отладив программу на КуМире, обучаемый может нажатием одной кнопки создать из КуМир-программы без ошибок работающую программу на C++ с текстом, похожим на исходную программу на алгоритмическом языке.

Кроме того, аппарат программных исполнителей можно использовать для создания обстановки, задания данных, а также для проверки правильности программы ученика. Например, в КуМир можно включить исполнитель с фиксированным набором предписаний-программ, одна из которых исполняется перед выполнением программы ученика, а другая – после ее окончания. Таким образом, педагог может задать различные данные для групп учащихся или индивидуально для каждого ученика, при этом обучающийся не только не сможет изменить заданные данные, но и даже увидеть их.

КуМировский исполнитель Робот живет в популярной для пропедевтического курса информатики клетчатой среде (программы Гости и Дед Мазай в программах Ассоциации «Компьютер и детство», Конюх и Кукарача в Роботландии…). Клетки среды могут отделяться друг от друга стенами. Рабочее поле Робота можно редактировать.

При таком редактировании щелчок мышкой в клетке закрашивает ее, а щелчок между клетками устанавливает стену. В начальном положении стен нет, а Робот стоит «дома» в левом верхнем углу рабочего поля.

Для того, чтобы в описываемом на КуМире алгоритме можно было воспользоваться исполнителем Робот, следует в начале описания поместить команду

использовать Робот

и только после этого начинать описание строкой, которая именует алгоритм. Например: алг Закрашенный квадрат.

Обратите внимание, что пользователь строит не описание программы, а описание алгоритма. Это важное понятие известно ребенку с третьего урока начального курса: в Роботе использована дедуктивная структура отношений алгоритма и программы: алгоритм – первичен, программа – вторична.

СКИ Робота включает 17 команд. На первых уроках вводятся пять команд-предписаний:

ВПРАВО
ВЛЕВО
ВВЕРХ
ВНИЗ
ЗАКРАСИТЬ.

Позднее, при изучении условий в командах циклов и ветвлений появляются еще 12 команд-вопросов:

лог сверху стена	лог сверху свободно
лог снизу стена	лог снизу свободно
лог справа стена	лог справа свободно
лог слева стена	лог слева свободно
лог клетка закрашена	лог клетка не закрашена

вещ температура
вещ радиация

Первые десять из них – это логические значения *истина* или *ложь*, а последние два – это датчики, которые вырабатывают вещественные значения.

С самых первых шагов освоения программного исполнителя школьники учатся оперировать алгоритмами: алгоритмы используются в циклах и на ветвях условных команд; в числе первых понятий вводятся *вспомогательные алгоритмы*, играющие ту же роль, что и процедуры в Лого, минуя искусственности сложного методического этапа индуктивных структур.

На рис. 1 показан результат линейный алгоритм передвижения Робота из левого нижнего угла (А) в правый верхний квадрата 4×4 (Б) с закрашиванием клеток, которые помечены звездочками (в примерах команды языка и ключевые слова КуМира подчеркнуты):

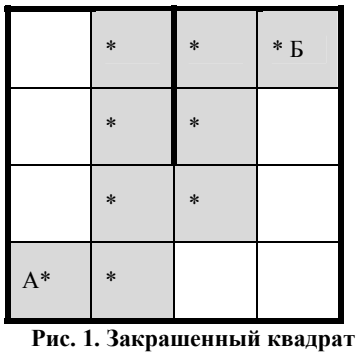


Рис. 1. Закрашенный квадрат

Вот описание этого алгоритма в КуМире:

использовать Робот
алг Закраска
- дано | Робот в точке А
- надо | Робот в точке Б, отмеченные клетки закрашены
нач
- закрасить; вверх; вправо
- закрасить; вверх;
- закрасить; вниз; вправо
- закрасить; вверх;
- закрасить; вверх;
- закрасить
кон

Школьники учатся мыслить алгоритмическими структурами, выделяя блоки-алгоритмы по поставленному заданию и синтезируя из них сложный образ. Так, к рассказу о циклах учитель просит детей нарисовать орнамент (рис. 2):

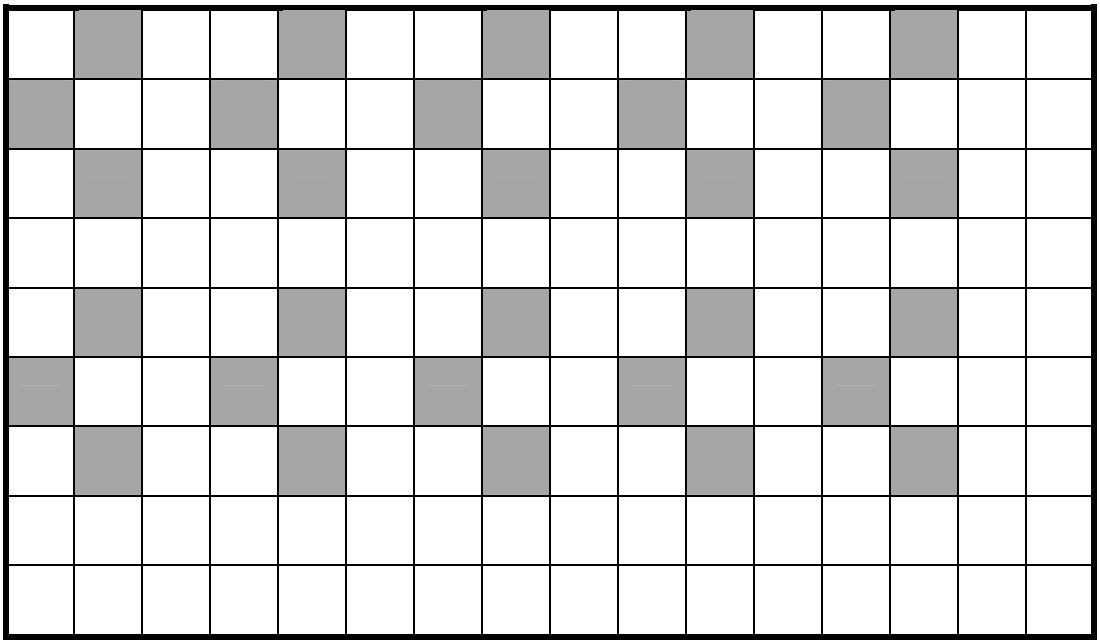


Рис. 2. Построение орнамента

Вот как записывается алгоритм построения требуемого орнамента средствами исполнителя Робот:

использовать Робот
алг Узоры
- дано | Робот в левом верхнем углу, в верхней части поля – орнамент
- надо | Закрасить клетки нижней строки орнамента
нач
- нц 2 раз
- - нц 5 раз
- - - Рисунок
- - - вправо; вправо
- - кц
- - нц 15 раз
- - - влево
- - кц
- - нц 4 раз

```
- - - вниз
- - кц
- кц
кон
алг Рисунок
- дано | Робот в левой верхнем углу клетки 3×2
- надо | Робот стоит на две клетки левее старта
нач
- вправо; закрасить
- вниз; влево; закрасить
- вниз; влево; закрасить
- вниз; вправо; закрасить
- вверх; вверх
кон
```

Команды ветвления в методическом отношении сложнее циклов и появляются в поле зрения школьника вслед за обсуждением циклов с фиксированным (задаваемым) числом шагов и циклов *пока*. Синтаксическая схема полного условного оператора для Робота имеет вид:

```
ЕСЛИ условие
        ТО серия 1
        ИНАЧЕ серия 2
ВСЕ
```

а сокращенное ветвление:

```
ЕСЛИ условие
        ТО серия 1
ВСЕ
```

Пример: алгоритм (рис. 3), который при движении слева направо закрашивает не все клетки горизонтального коридора, а только те из них, которые имеют выход вверх. В исходном состоянии Робот стоит в левом конце коридора.

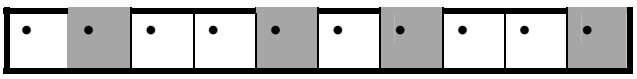


Рис. 3. Алгоритм с ветвлением в сокращенной форе

Переход к «старшему брату» – еще одному кумировскому исполнителю Чертежнику – для школьников, познакомившихся с Роботом, несложен. Правда, для более или менее сложных задач, размещение достаточно большого числа клеточек среды может показаться проблемой. Но ведь в КуМире есть возможности редактирования среды, в том числе увеличения количества строк и столбцов этой клетчатой среды.

СКИ Чертежника даже менее объемна, чем у Робота. Специфических команд всего четыре. Две из них совпадают по своим возможностям с управлением пером в Лого:

```
ПОДНЯТЬ ПЕРО
ОПУСТИТЬ ПЕРО.
```

Две другие показывают два разных способа перемещения рисующего инструмента – абсолютного и относительного:

```
СМЕСТИТЬСЯ В ТОЧКУ (АРГ ВЕЩ X, Y)
СМЕСТИТЬСЯ НА ВЕКТОР (АРГ ВЕЩ A, B)
```

Этими средствами (в совокупности с тем, которые уже известны по упражнениям Робота и операторами присваивания x:= a; x:=x+d;…) можно запрограммировать построение параболы [7]:

```
использовать Чертежник
алг парабола (арг вещ a, b, арг цел n)
    дано n > 0 | перо Чертежника поднято
    надо | нарисован график функции y = x**2 на участке a, b
```

```
                | в виде ломаной из n звеньев, перо в точке (b, b**2)
                | и поднято
нач вещ x, d
- x := a; d := (b – a)/n
- сместиться в точку (a, a**2)
- опустить перо
- нц n раз
-- x := x + l
-- сместиться в точку (x, x**2)
- кц
- поднять перо
кон
```

Внимательного взгляда на два разных кумировских исполнителя и их сравнения достаточно, чтобы провести некоторые на поверхности лежащие обобщения, полезные для последующих выводов. Рассматривая алгоритмы управления двумя представленными здесь исполнителями КуМира – Роботом и Чертежником, – можно выделить их общие, инвариантные элементы:

- описание алгоритмов;
- описание установок программы «Дано» и «Надо»;
- а также команды цикла n раз, пока и ветвлений с их принципами структурирования записи алгоритмов.

Очевидно, что эти инварианты могут быть использованы для описания алгоритмов управления и другими подобными исполнителями.

Общий вид описания алгоритма

```
алг имя (аргументы и результаты)
      дано условия применимости алгоритма
      надо цель выполнения алгоритма
нач описание промежуточных величин
|     тело алгоритма (последовательность команд)
|
кон
```

Команды алгоритмического языка

```
нц число повторений раз
|       тело цикла (серия команд)
|
кц

нц пока условие
|       тело цикла (серия команд)
|
кц

нц для i от i1 до i2
|         тело цикла (серия команд)
|
кц

нц пока условие
|       тело цикла (серия команд)
|
кц
```

```
если условие
|    то серия1
|    иначе серия2
|
все
```

```
если условие
|    то серия1
|
все
```

Если Чертежник назван здесь «старшим братом» Робота, то его «младшим братом» сами разработчики называют ПиктоМир – свободно распространяемую программную систему для изучения азов программирования дошкольниками и младшими школьниками. ПиктоМир позволяет ребенку «собрать» из пиктограмм на экране компьютера несложную программу, управляющую виртуальным исполнителем-роботом. ПиктоМир в первую очередь ориентирован на дошкольников, еще не умеющих писать, или на младшеклассников, не очень любящих писать. Именно эта ориентация на отсутствие навыков чтения текста и письма (клавиатурного набора) привели к жесткого условию на проект – использования выбор пиктограмм вместо текстового набора заданий с клавиатуры.

При желании ПиктоМир-программу можно сохранить в КуМире и продолжить работ над ней в КуМире.

В центре внимания ПиктоМира находится исполнитель Вертун.

В описываемой здесь обновленной версии программно-методической системы КуМир [10] предлагается знакомство с программой-роботом Вертуном (программисты и авторы идеи – А. Г. Кушниренко, А. Г. Леонов). Его среда – это узкие коридоры (шириной с Вертуна), по которым он может перемещаться. Впрочем, перемещаться, в прямом смысле этого слова, он может только ВПЕРЕД, в том направлении, куда направлена его камера. Но он может, кроме того, еще и поворачиваться НАЛЕВО или НАПРАВО точно на 90 градусов. Значит, если Вертуну потребуется сменить направление на противоположное, то для этого ему придется дать две команды подряд:

ВПРАВО ВПРАВО

(или, с тем же результатом

ВЛЕВО ВЛЕВО).

Направление, по которому надо двигаться ВПЕРЕД, распознается по направлению камеры, водруженной на голову Вертуна. Впрочем, задавая эти (и другие) команды, не надо набирать их на компьютерной клавиатуре: достаточно выбрать нужную команду из пиктограмм над рабочим полем Вертуна[1].

Вот пример программы, одной из самых простых среди тех, которые доступны Вертуну (рис. 4): робот установлен в левом краю коридора; его камера направлена не вдоль коридора, а направо, лицом к наблюдателю-пользователю. За время своего путешествия в правый конец коридора, Вертун должен закрасить клеточку, в которой он в текущий момент находится (каждая клеточка закрашивается командой, изображаемой ведерком, из которой выливается краска). Из полиграфических соображений свойственное экранному представлению среды аксонометрическое изображение исполнителя и его среды заменено схемой-планом (рис. 4):

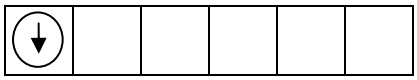


Рис. 4. Исходное положение Вертуна в простой программе

а используемые команды – пиктограммами:

ВПЕРЕД –

ВПРАВО – 

ВЛЕВО – 

ЗАКРАСЬ 

В исполнителе Вертун предусмотрена деятельность в одном из двух режимов – *пошаговом* (когда каждая команда программы выполняется отдельно) и *программном* (когда выполняются последовательно одна за другой *все* предварительно записанные команды программы, как на рис. 5). Первое упражнение с программой ребенок должен выполнять, конечно, в пошаговом режиме.

Первые две команды этой программы могут выполняться в любом порядке. Здесь выбран такой порядок: сначала закрашивается первая клетка, а затем, оставаясь в той же клетке, Вертун поворачивается камерой вдоль коридора – налево. Далее выполняются пять повторяющихся пар команд ВПЕРЕД ЗАКРАСКА:

Наличие двух режимов не только разрешает, но и методически требует сразу же после правильного выполнения программы в пошаговом режиме запустить ее еще раз в программном режиме (запуск в программном режиме – это щелчок по большой кнопке с треугольником, а в пошаговом – по маленькой кнопке, размещенной непосредственно под большой).

Правильное выполнение программы завершается сообщением Вертуна о выполнении задания. Впрочем, могут возникать и другие сообщения. Например, если самая последняя команда приведенной выше программы не будет записана, то вместе с сообщением о выполнении программы Вертун скажет еще и о том, что

Не закрашена 1 клетка.

Следующее задание немного сложнее (рис. 5): все правила управления Вертуном сохраняются, однако среда (расположение коридоров) становится другой. Заметим, что порядок усложнения заданий может нарастать не фиксированным путем, а по воле учителя, дифференцированно выбирающего для учеников индивидуальные траектории изучения материала.

В этой задаче вслед за серией последовательных команд появляется следующая управляющая структура – цикл n раз. Здесь необходимо увидеть повторяющиеся действия и число этих действий – четыре (обратите внимание на то, что в этом задании часть клеток – в середине каждого ребра квадрата – не закрашивается). Тогда в начале программы следует установить число в виде игральной косточки с четырьмя точками (начало цикла, задающее число его повторений), а вслед за ним записывается повторяемая последовательность действий (тело цикла):

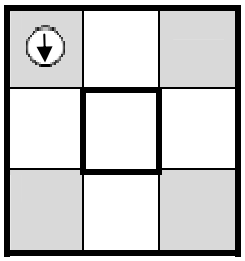


Рис. 5. Цикл n раз в программе

Уже в этой простой программе надо обратить внимание детей на требование реинтерабельности программ, принципиальное при создании многократно используемых конструкций. Для этой программы каждый из фрагментов должен начинаться с закрашивания клеточки и движения ВПЕРЕД. Тогда перед началом нового фрагмента Вертун должен быть в состоянии, соответствующем началу предшествовавшего.

Еще один шаг вперед в коллекции сред Вертуна приводит к использованию важного понятия процедуры – именованной программы (рис. 6):

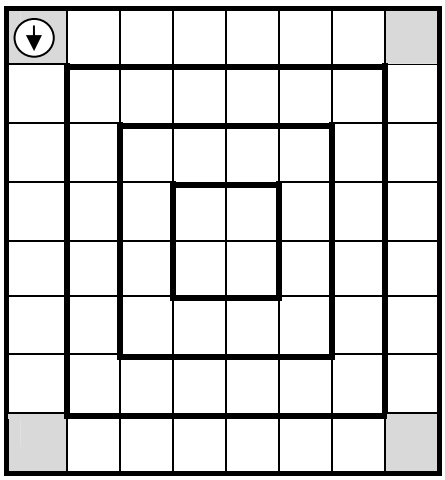
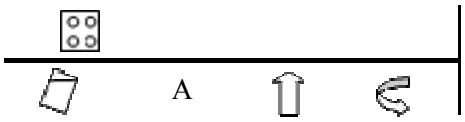


Рис. 6. Цикл n раз в программе

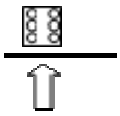
Взгляд на среду Вертуна в этой задаче выделяет два участвующих в ней алгоритма. Один из них – это перемещение по стороне большого квадрата, с закрашиванием при этом только угловых клеток. Другой – четырехкратное обращение к первой.

Чтобы сделать такое выделение фрагментов в поставленной задаче – пройти по указанному маршруту-коридору, выполнив закраску угловых клеток, – удобно назвать прогулку по стороне большого квадрата каким-нибудь именем. Здесь таким именем выбрана буква А. Тогда алгоритм решения (*главный алгоритм задачи*) – это 4-кратное реинтерабельное повторению алгоритма, названного именем А. Выделение повторяющихся фрагментов-процедур – *вспомогательных алгоритмов*, главное из умений программировать сложный процесс.

Главный алгоритм:

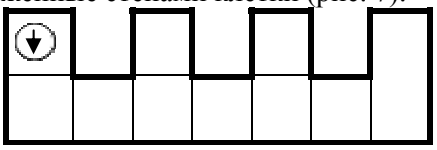


Команда А (вспомогательный алгоритм)



Процесс усложнения сред Вертуна нескончаем.

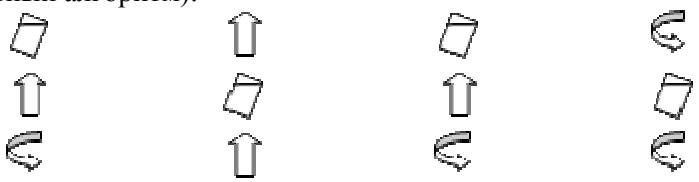
Завершим его здесь несколько более сложной задачей, записав один из вариантов решения. Исполнитель должен закрасить окруженные стенами клетки (рис. 7):



Главный алгоритм:

A A A

Команда А (вспомогательный алгоритм):



Приведенные примеры демонстрируют фрагменты коллекции сред Вертуна: не меняя свои функциональные возможности, программа исполнителя Вертуна представляет широкое множество алгоритмических задач.

Библиографический список

1. Андреев, Д. В., Изосимова, Л. М. Аппаратное и программное обеспечение курса робототехники на основе технологий LEGO [Текст] / Д. В. Андреев, Л. М. Изосимова. // Ярославский педагогический вестник. – 2011. – Том III (Естественные науки), № 1. – С. 51–60.

2. Белиовская, Л. Г., Белиовский, А. Е. Программируем микроконтроллер NXT в LabView [Текст] / Л. Г. Белиовская, А. Е. Белиовский. – М. : ДМК Пресс, 2010.

3. Давыдова, В. Е. Опыт 444 школы [Текст] / В. Е. Давыдова. // Информатика. –2007. – № 2. – С. 33–38.

4. Дуванов, А. А., Шумилина, Н. Д. Азбука Роботландии [Текст] / А. А. Дуванов, Н. Д. Шумилина. // Вестник Ярославского регионального отделения РАЕН. – 2012. – № 1. – С. 35–40.

5. Звенигородский, Г. А. Первые уроки программирования [Текст] / Г. А. Звенигородский. – М. : Наука, библиотечка «Квант». – Вып. 41. – 1985. – 208 с.

6. Комис, В., Мисирли, А. Педагогическая робототехника и предварительные концепции программирования в детском саду: исследование на примере программируемой игрушки Пчелка [Текст] / В. Комис, А. Мисирли // Труды Большого Московского семинара по методике раннего обучения информатике : сб. под ред. Ю. А. Первина, И. В. Соколовой. – Т. 3. – М. : Изд. РГСУ, 2012. – С. 106–118.

7. Кушниренко, А. Г. Основы информатики и вычислительной техники : учебное пособие для 10–11-х кл. общеобразовательных учреждений [Текст] / А. Г. Кушниренко, Г. В. Лебедев, Р. А. Своренгъ. – М. : Просвещение, 1990.

8. Кушниренко, А. Г. Информатика 7–9 классы [Текст] / А. Г. Кушниренко, Г. В. Лебедев, Я. Н. Зайдельман. – М. : Дрофа, 2000 – С. 336.

9. Кушниренко, А. Г. Пиктомир: пропедевтика алгоритмического языка (опыт обучения программированию старших дошкольников) [Текст] / А. Г. Кушниренко, И. Б. Рогожкина, А. Г. Леонов // Труды Большого Московского семинара по методике раннего обучения информатике : сб. под ред. Ю. А. Первина, И. В. Соколовой. – Т. 3. – М. : Изд. РГСУ, 2012. – С. 119–129.

10. Кушниренко, А. Г., Леонов, А. Г. КуМир вернулся! [Текст] / А. Г. Кушниренко, А. Г. Леонов. // Информатика, – 2009. – № 6 (583) – С. 8–10.

11. Леонов, А. Г., Первин, Ю. А. Элементы программирования в непрерывном курсе школьной информатики [Текст] / А. Г. Леонов, Ю. А. Первин. // Ярославский педагогический вестник. – 2013. – Т. 3 (Естественные науки). – № 1. – С. 45–50.

12. Монахов, В. М. Педагогическая технология профессора В. М. Монахова [Текст] / В. М. Монахов. // Спец. выпуск «Педагогического вестника». – Успешное обучение, 1997.

13. Первин, Ю. А. Динамика вузовского курса «Теории и методики обучения информатике» (концепции, опыт, рекомендации) [Текст] / Ю. А. Первин. – LAMBERT Academic Publisher, Berlin, 2012. – 332 с.

14. Первин, Ю. А. Методика раннего обучения информатике [Текст] / Ю. А. Первин. – 2-е издание. –М. : БИНОМ, Лаборатория знаний, 2008. – 288 с.

15. Первин, Ю. А. Об эксперименте по преподаванию программирования в младших классах средней школы [Текст] / Ю. А. Первин. // Кибернетика. – № 2. – 1974.

Bibliograficheskij spisok

1. Andreev, D. V., Izosimova, L. M. Apparatnoe i programmnoe obespechenie kursa robototehniki na osnove tehnologij LEGO [Tekst] / D. V. Andreev, L. M. Izosimova. // Jaroslavskij pedagogicheskij vestnik. – 2011. – Tom III (Estestvennye nauki), № 1. – S. 51–60.

2. Beliovskaja, L. G., Beliovskij, A. E. Programmiruem mikrokontroller NXT v LabView [Tekst] / L. G. Beliovskaja, A. E. Beliovskij. – M. : DMK Press, 2010.

3. Davydova, V. E. Opyt 444 shkoly [Tekst] / V. E. Davydova // Informatika. –2007. – № 2. – S. 33–38.

4. Duvanov, A. A., Shumilina, N. D. Azbuka Robotlandii [Tekst] / A. A. Duvanov, N. D. Shumilina. // Vestnik Jaroslavskogo regional'nogo otdelenija RAEN. – 2012. – № 1. – S. 35–40.

5. Zvenigorodskij, G. A. Pervye uroki programmirovanija [Tekst] / G. A. Zvenigorodskij. – M. : Nauka, bibliotechka «Kvanta». – Vyp. 41. – 1985. – 208 s.

6. Komis, V., Misirli, A. Pedagogicheskaja robototehnika i predvaritel'nye koncepcii programmirovanija v detskom sadu: issledovanie na primere programmiruemoj igrushki Pchelka [Tekst] / V. Komis, A. Misirli // Trudy Bol'shogo Moskovskogo seminara po metodike rannego obuchenija informatike : sb. pod red. Ju. A. Pervina, I. V. Sokolovoj. – T. 3. – M. : Izd. RGSU, 2012. – S. 106–118.

7. Kushnirenko, A. G. Osnovy informatiki i vychislitel'noj tehniki : uchebnoe posobie dlja 10–11-h kl. obshheobrazovatel'nyh uchrezhdenij [Tekst] / A. G. Kushnirenko, G. V. Lebedev, R. A. Svoreng'. – M. : Prosveshhenie, 1990.

8. Kushnirenko, A. G. Informatika 7–9 klassy [Tekst] / A. G. Kushnirenko, G. V. Lebedev, Ja. N. Zajdel'man. – M. : Drofa, 2000 – S. 336.

9. Kushnirenko, A. G. Piktomir: propedevtika algoritmicheskogo jazyka (opyt obuchenija programmirovaniju starshih doshkol'nikov) [Tekst] / A. G. Kushnirenko, I. B. Rogozhkina, A. G. Leonov // Trudy Bol'shogo Moskov-skogo

seminara po metodike rannego obuchenija informatike : sb. pod red. Ju. A. Pervina, I. V. Sokolovoj. – T. 3. – M. : Izd. RGSU, 2012. – S. 119–129.

10. Kushnirenko, A. G., Leonov, A. G. KuMir vernulsja! [Tekst] / A. G. Kushnirenko, A. G. Leonov. // Informatika, – 2009. – № 6 (583) – S. 8–10.

11. Leonov, A. G., Pervin, Ju. A. Jelementy programmirovanija v nepreryvnom kurse shkol'noj informatiki [Tekst] / A. G. Leonov, Ju. A. Pervin // Jaroslavskij pedagogicheskij vestnik. – 2013. – T. 3 (Estestvennye nauki). – № 1. – S. 45–50.

12. Monahov, V. M. Pedagogicheskaja tehnologija professora V. M. Monahova [Tekst] / V. M. Monahov. // Spec. vypusk «Pedagogicheskogo vestnika». – Uspeshnoe obuchenie, 1997.

13. Pervin, Ju. A. Dinamika vuzovskogo kursa «Teorii i metodiki obuchenija informatike» (koncepcii, opyt, rekomendacii) [Tekst] / Ju. A. Pervin. – LAMBERT Academic Publisher, Berlin, 2012. – 332 s.

14. Pervin, Ju. A. Metodika rannego obuchenija informatike [Tekst] / Ju. A. Pervin. – 2-e izdanie. –M. : BINOM, Laboratorija znanij, 2008. – 288 s.

15. Pervin, Ju. A. Ob jeksperimente po prepodavaniju programmirovanija v mladshih klassah srednej shko-ly [Tekst] / Ju. A. Pervin // Kibernetika. – № 2. – 1974.

[1] В программах, работающих на IPad'ах с сенсорными экранами, выбор задаваемой Вертуну команды делается не щелчком мыши, а легким касанием пальца.