УДК 372.862

Дженжер В.О.
Оренбургский государственный университет
E-mail: vdjenjer@yandex.ru

МЕСТО ПРОГРАММИРОВАНИЯ В КУРСЕ ИНФОРМАТИКИ НАЧАЛЬНОЙ ШКОЛЫ

Рассматриваются сложившиеся подходы к обучению программированию. Предлагается способ обучения программированию в младшей школе.
Ключевые слова: информатика, программирование, начальная школа, Scratch.

Одним из важнейших направлений подготовки любого ИТ-специалиста является обучение программированию. Это один из фундаментальных компонентов информатики, важность которого для современного технологичного общества не подлежит сомнению. Элементы программирования входят даже в образовательные стандарты гуманитарных специальностей, таких, например, как психология и социология. Это говорит о том, что разработчики стандартов высшего профессионального образования не сомневаются в необходимости изучения программирования гуманитариями, хотя бы на начальном уровне. В то же время отношение к информатике вообще, и в частности к программированию, со стороны системы общего образования не столь однозначно.

С одной стороны, известен лозунг академика А.П. Ершова «Программирование – вторая грамотность» (1981 г.). В статье [1] А.П. Ершов излагает свое видение роли компьютера в образовании и показывает, что «грамотность и программирование не только выстраиваются в параллель, соединяясь мостиками аналогий, но и дополняют друг друга, формируя новое представление о гармонии человеческого ума». А.П. Ершов удивительным образом предвосхищает ряд положений современного компетентностного подхода, заявляя, что кроме обычного (книжного) знания человек должен актуализировать накопившееся за тысячелетия операциональное знание. В этом и заключается важнейшее предназначение программирования или, если сказать шире, алгоритмики.

С другой стороны, существует не вполне централизованная оппозиция, противопоставляющая алгоритмизацию – гуманизации. В достаточной степени этот подход выражен в структуре и содержании новых федеральных государственных образовательных стандартов общего образования второго поколения (http://standart.edu.ru), в которых представлен в основном технологический компонент информатики, но сильно обеднено ее фундаментальное ядро. Что еще более неожиданно, информатика в начальной школе разделена пополам: на технологический компонент, который изучается в рамках дисциплины «Технология», и теоретический компонент, перенесенный на уроки математики. Стоит ли говорить, что программированию при таком подходе совершенно не остается места.

Удержаться на золотой середине можно за счет вариативной части стандартов второго поколения и при поддержке администрации школы. Мы придерживаемся мнения, что информатика не должна ни сводиться к технологии, ни преподаваться как один из разделов математики. Это подразумевает, в частности, что кроме информационных технологий, элементов информационного моделирования и математической основы теории информации должно изучаться и программирование. При таком подходе возникает ряд вопросов: когда начинать, чему учить и что мы вообще хотим получить в результате.

Непростой проблемой является определение оптимального возраста ученика, начиная с которого он может воспринимать достаточно абстрактные концепции программирования. Не вдаваясь глубоко в эту проблему, отметим, что нам близки конструктивистские идеи Ж. Пиаже, С. Пейперта, А. Кея и других исследователей, показавших, что любая деятельность может приобрести смысл и интерес, если заменить изучение объекта его конструированием. Такие действия требуют от ученика не только наличия определенных знаний об объекте, но и неких метапредметных, операциональных умений, связанных с конструированием. Ж. Пиаже замечает, что любое знание связано с действием.

Как мы знаем, подход С. Пейперта оказался не умозрительным, а вполне конкретным: он

создал язык программирования, которым может овладеть даже ребенок детсадовского возраста. И все потому, что действия, составляющие алгоритм, ребенок, в принципе, может выполнить сам, безо всякого компьютера: пройди вперед пять шагов, поверни направо, подними перо и т. д. Все легко представимо и вообразимо, все имеет смысл. Очень важно, что и результат работы программы также имеет вполне ясный и наглядный смысл, без излишних абстракций: в итоге получается графическое изображение объекта.

Но имеет ли смысл вся эта активность с точки зрения развития личности ребенка? По мнению С. Пейперта, – несомненно, поскольку в данном случае язык программирования – это «объект, с помощью которого думают». Алан Кей, один из создателей LOGO, добавляет, что средства, которые существенно меняют способы мышления, должны быть доступны ребенку как можно раньше. А.П. Ершов, который с интересом наблюдал за экспериментами Пейперта, заметил: «Хотелось бы подчеркнуть, что речь идет не о том, чтобы навязать детям новые, несвойственные им навыки и знания, а о том, чтобы проявить и сформулировать те стороны мышления и поведения, которые реально существуют, но формируются стихийно, неосознанно» [1]. Таким образом, становится ясным, что программирование – это не самоцель. В процессе изучения программирования (и алгоритмики вообще) формируются специфические «функциональные мозговые органы» (А.Н. Леонтьев). И очень важно, что эти «органы» формируются в общении и предметной деятельности ребенка. Программирование – достаточно своеобразная деятельность, поэтому и формируемые ею «мозговые органы» специфичны.

С. Пейперт считал, что разработанный им LOGO является моделью языка, реализующего конструктивистский подход, и со временем появится множество более современных и более продуманных языковых концепций. Сегодня существует огромное количество интересных программных продуктов, в той или иной степени реализующих конструктивистские идеи. Отметим среди них Scratch [2]; одну из современных реализаций Smalltalk – Squeak [3]; современные клоны LOGO – Star Logo [4] и NetLogo [5]; среду 3D-программирования Alice [6]; фреймворк, позволяющий создавать программы на Java в стиле, напоминающем Scratch, – Greenfoot [7]. Отдельно можно выделить продукты, ориентированные на создание игр: основанный на Python – Pygame [8], Game Maker [9], Little Wizard [10] и другие. Все эти проекты так или иначе предполагают изучение программирования, но базируются при этом на нетрадиционном подходе. Поясним сказанное и рассмотрим несколько известных способов организации материала по программированию.

Существует два различных жанра: описание языка и описание алгоритмов. К первому, например, относится [11], а ко второму – известная монография [12]. Разумеется, невозможно говорить о языке и не сказать ни слова об алгоритмах, иначе не привести сколь-нибудь содержательных примеров. Да и для объяснения алгоритмики нельзя обойтись без языка программирования, хотя порой выбранный язык может быть очень экзотичным, как, например, в классическом труде Д. Кнута «Искусство программирования». Рассмотрим типичную структуру книг по языкам программирования.

Начиная с самых первых трудов по программированию были выработаны неизменные до сего дня каноны преподавания. Если мы говорим об императивном программировании, то вначале рассматриваются алфавит, простые типы, основные встроенные функции и синтаксические конструкции (операторы), затем структурные типы и средства работы с ними. Параллельно могут изучаться и основные алгоритмы. Например, в издании 1980 года [13] две главы (вторая и третья) посвящены изучению Алгола-60. В первой из них идет описание синтаксиса языка, а в следующей рассматриваются структуры данных. Похожий способ изложения можно обнаружить в [14] и в [15]. Все эти книги следуют с интервалом примерно в 10 лет между собой, но имеют вышеописанную каноническую структуру. Не сильно меняется картина и в современных учебниках, выходящих через 30 лет после [13], разве что дополнительно происходит ориентация учащегося на работу с тем или иным (обычно объектно ориентированным) фреймворком.

Такой подход оправдывает себя, если поставлена цель обучить специалиста, да и то лишь в случае изначально хорошо подготовленного и профессионально ориентированного студента. В общеобразовательной школе изложение материала в названном ключе, по нашему мнению, за очень редким исключением вообще не имеет смыс-

ла. Причем буквально: не имеет смысла для ученика, поскольку ему заранее даются ответы на еще не заданные вопросы, а набор предлагаемых заданий не является личностно значимым. Конечно, подобная ситуация наблюдается и с другими предметами, и в этом, на наш взгляд, недостаток современного подхода к содержанию образования и частным методикам. В начальной школе подобный способ организации изучения программирования вовсе недопустим.

В качестве альтернативы стандартной схеме изучения программирования мы предлагаем воспользоваться конструктивистскими идеями и отказаться от изучения инструмента прежде возникновения необходимости в нем. Это возможно, если не «изучать язык программирования», а просто программировать. Именно так может строиться курс программирования на языке Scratch [2].

Следует отметить, что хотя весь курс и концентрируется вокруг языка программирования, основная наша задача – развитие личности ребенка. Вот почему можно увидеть в проводимых занятиях и творческие мастерские, и тематические презентации, и создание динамических комиксов, и элементы исследовательской деятельности, и лишь на последнем плане – непосредственно программирование. Выбор языка программирования Scratch обусловлен следующими его особенностями.

Среда программирования. Scratch – это в первую очередь система программирования, обладающая всеми необходимыми атрибутами. Scratch имеет собственный редактор текста программы, построенный на интересной идее конструкторов Lego: все операторы языка и другие его элементы представлены блоками, которые могут соединяться один с другим, образуя скрипт (фрагмент кода). Важной особенностью блоков является их «специализация»: имеется несколько видов блоков и они могут составляться не произвольным образом, а лишь сообразно своему назначению. Так и в конструкторе Lego не каждую деталь можно соединить с любой другой. Это ограничивает количество возможных вариантов соединения и, соответственно, исключает возможность появления синтаксических ошибок. Кроме этого имеется транслятор, унаследованный от Squeak, и отладчик, позволяющий выполнять программы в пошаговом режиме.

Как язык программирования Scratch представляет собой разновидность объектно ориентированного языка, наследника первого объектно ориентированного языка Smalltalk. Такая родословная Scratch позволяет программисту свободно использовать его в качестве инструмента для моделирования объектов и процессов реального мира. Встроенная и интуитивно понятная графическая подсистема языка позволяет легко проводить визуализацию динамики модели, а также включать в нее элемент интерактивности.

Одной из важнейших особенностей Scratch как языка программирования является его событийно ориентированный характер. Это означает, что все объекты взаимодействуют при помощи обмена сообщениями. Такая схема обмена информацией делает Scratch близким к современным объектно ориентированным языкам и позволяет впоследствии более просто организовать переход к изучению Java, Delphi, C# и др.

Scratch является языком, в котором последовательно реализована идея многопоточности. Каждый скрипт любого объекта запускается в отдельном потоке. В отличие от процедурных языков, в которых принято последовательное выполнение кода, в Scratch скрипты могут выполняться и параллельно. Причем никаких дополнительных действий выполнять не требуется: все это особенности языка.

Наконец, мы не знаем другого языка программирования, который бы имел средства для написания программ, умеющих получать информацию извне (из реального мира) без помощи клавиатуры и мыши и без программирования портов. При наличии специального устройства, подключаемого к порту USB, программа на Scratch может «слышать» звуки, «чувствовать» освещенность в комнате, «ощущать» движения пользователя и выполнять в соответствии с полученной информацией различные действия.

Таким образом, можно выделить следующие имманентные свойства Scratch, имеющие значительный педагогический потенциал.

Простота и дружественность интерфейса позволяют начинать изучение программирования, как только дети научатся читать.

Редактор текстов в виде конструктора дает возможность на подсознательном уровне превратить «учебу» в «не учебу», а кроме того – сократить количество ошибок в программе.

Ориентированность на графику, так как доказана эффективность обучения с опорой на наглядно-образное мышление.

Объектная ориентированность позволяет изучить основные способы создания программ с объектами. Заметим, что Scratch – своеобразный объектно ориентированный язык. В нем отсутствуют концепции, без которых многие вообще не представляют себе объектно ориентированную технологию. Например, при изучении Scratch ученик не описывает классов, не использует полиморфизм, не строит деревьев наследования, не сталкивается с инкапсуляцией. Есть только объект, его поля (переменные) и методы (скрипты). Но отсутствие названных концепций создает существенные *трудности* программирования, особенно заметные при разработке сложных проектов. Многочисленные повторения кода ведут в конце концов к идее наследования, многочисленные однотипные сообщения для разных объектов – к идеям полиморфизма и абстрактных классов. Легкость изменения полей (и совершения ошибок – при таких изменениях) приводит к мысли о сокрытии данных, то есть инкапсуляции. При этом важно, что эти идеи не сообщаются педагогом неподготовленному ученику (даже для студентов 1–2 курса математического факультета названные концепции бывают слишком абстрактны), а самозарождаются и впоследствии падают на подготовленную почву.

Ориентация на обработку событий способствует формированию метаумения устанавливать причинно-следственные связи, развитию логического мышления, закладывает основы системного мировосприятия через демонстрацию систем как объектов со связями. Наблюдая за поведением таких объектов, ученик приобретает умение отделять существенные признаки предмета от несущественных. Такая аналитико-синтетическая работа характерна для начальных этапов поисково-исследовательской деятельности (Н.А. Менчинская).

Многопоточность позволяет не просто строить модели объектов, но создавать модели действительно комплексных систем, причем без излишних технических сложностей. Кроме того, параллельное программирование сегодня является чрезвычайно востребованной технологией, что делает раннее знакомство детей с ней полезным в плане будущей профессиональной ориентации.

Среда проектирования. Scratch не только язык программирования, но и удачная среда для проектной деятельности, поскольку все необходимое для такой деятельности включено в его состав. Кроме названных выше редактора, компилятора и отладчика имеются:

– графический редактор для создания и модификации визуальных объектов;

– библиотека готовых графических объектов (некоторые из них содержат наборы скриптов);

– библиотека звуков и музыкальных фрагментов;

– большое количество примеров.

Непосредственно в среду Scratch не входит, но также играет огромную роль динамичное и дружелюбное *сообщество любителей Scratch*, к которому можно подключиться при помощи Интернета. В сообществе, организованном при Массачусетском технологическом институте (США), можно разместить свои работы, послушать похвалы и критику, найти единомышленников, почерпнуть новые идеи. Одна из ключевых особенностей сообщества Scratch состоит в том, что проекты распространяются с исходными кодами, так что в принципе отсутствует (или почти отсутствует) идея копирайта. Поскольку речь не идет о коммерческих разработках, это не вызывает недовольства. Напротив, именно благодаря такой свободе возможен быстрый обмен идеями, что не может не сказаться на количестве работ, выложенных в открытый доступ.

Несмотря на то, что каждый может взять любую работу и позаимствовать оттуда код, повторяющихся работ (попросту говоря, плагиата) очень мало. И это несмотря на то, что заимствования кода приветствуются. По всей видимости, здесь мы наблюдаем процесс самоорганизации, в результате которого банальный плагиат трансформируется в заимствование и *дальнейшее развитие идей*.

Уточним возможность реализации *междисциплинарных проектов*. Именно междисциплинарность позволит школьнику создать единую картину мира, наводя мостики между различными, иногда, на первый взгляд, довольно далекими друг от друга науками. В этом случае возможности Scratch неоценимы, так как он доступен не только представителю точных наук. Учитель-гуманитарий, например, может использовать Scratch для создания *динамичных и интерактивных презентаций*.

Отметим, что работа в среде Scratch может быть организована как индивидуально, так и коллективно. Разделение функций и ролей среди участников проекта может быть основано на следующих принципах:

– по функции или роду деятельности (сценарист, художник, программист и т. п.);

– по частям проекта (каждый участник выполняет одновременно несколько ролей, разрабатывая свою часть общего проекта).

Первый способ деления может быть применен в разновозрастных группах, в группах с разным опытом проектной работы, в группах, где учащиеся имеют различные способности или типы мышления, т. е. в разнородных группах. Второй способ подходит для однородных групп. Однако эти рекомендации не носят жесткого характера и могут варьироваться в зависимости от ситуации.

Среда моделирования. Перечисленные особенности Scratch делают его очень удобной, практически идеальной средой для обучения моделированию.

Моделирование представляет собой один из наиболее универсальных методов познания действительности. Для нас в настоящем контексте важен его педагогический потенциал, а также возможность реализовать этот потенциал в Scratch.

Мысленный образ модели основан на анализе изучаемого объекта и может быть представлен в примитивах некоторого языка своими компонентами и связями. В нашем случае идеальный объект моделирования разделяется на компоненты, а затем воссоединяется (синтезируется) в новой среде, отличной от мозга, а именно в среде Scratch: происходит объективация идеальной структуры. Такой подход характерен для построения любой модели, поэтому простейшие аналитико-синтетические действия будут вынужденно производиться учеником.

Следующим шагом после создания модели является проверка ее на адекватность и, если нужно, – коррекция. Тестирование производится в режиме игры с моделью. Это не наблюдение за столбцами цифр, не слежение за колебаниями абстрактных графиков, а именно игра. Во время игры автор замечает свои недоработки, неточности и ошибки. Затем соотносит их с кодом и вносит необходимые исправления. Заметим, что во многих курсах моделирования этап анализа проводится очень скомканно. Происходит это в силу того, что результаты моделирования слишком абстрактны и не вызывают сопереживания у учеников. С моделями на Scratch интересно поиграть, а если в «игре» что-то не так, то можно легко исправить ошибку.

Среди моделей на Scratch можно выделить: простую или интерактивную анимацию; феноменологическую модель объекта, процесса или явления; математическую модель.

Отметим, что может быть очень простая программа для реализации математической модели и очень сложная для интерактивной анимации.

Таким образом, от других языков программирования, используемых в школе, Scratch отличается изначальной направленностью на создание моделей, работу с моделями. Это делает его незаменимым инструментом для организации проектной научно-познавательной деятельности.

Среда для творчества. Было бы неверным представлять Scratch только как средство организации учебной деятельности. Он может использоваться и как инструмент творчества. В Интернете огромное количество проектов исключительно эстетической направленности. Богатство встроенных визуальных эффектов делает Scratch очень привлекательным в качестве средства самовыражения. Несмотря на отсутствие «научности», такие проекты лишь первый шаг к более серьезной проектной научно-познавательной деятельности учеников, так как по мере роста мастерства растут и здоровые амбиции создателей. Хочется новых идей, новых инструментов, что и выводит автора «творческих» проектов к «исследовательским» проектам (слова здесь взяты в кавычки потому, что зачастую очень сложно бывает провести границу между проектами разного вида).

Отметим, что Scratch может использоваться как единый инструмент для самых различных возрастов и типов мышления. Практически с того момента, как ребенок научился читать (и даже раньше: просто в этом случае блоки языка рассматриваются как своеобразные иероглифы) и до 14–16 лет.

По итогам трехлетней работы со школьниками в МОУ «Гимназия №3» г. Оренбурга и МОУ «Лицей №1» г. Тюльгана у нас сложился свой взгляд на преподавание программирования, главной особенностью которого является подход «от задач». Мы не «изучаем операторы языка», а используем их, если они понадобились нам для решения данной конкретной задачи. Например, команда ветвления (в обычном ее понимании) вводится только в двенадцатом проекте, причем каждый проект рассчитан в среднем на 3–5 занятий. В стандартном курсе программи-

рования эта тема рассматривается в самом начале, иначе просто невозможно придумать более или менее содержательных и интересных задач.

Важными особенностями организации наших занятий являются их необязательность, отсутствие отметок, свобода в выборе темы проекта, свобода объединения в группы. Приветствуются междисциплинарные проекты. Готовые работы оцениваются на итоговых фестивалях, демонстрируются на общих (с учениками) родительских собраниях. В ближайшем будущем мы собираемся организовать разновозрастные группы учеников. Мы считаем, что «отзадачный» подход вполне оправдан в преподавании основ программирования в общеобразовательной школе.

21.05.2010

Список использованной литературы:
1. Ершов А. П. Программирование – вторая грамотность // Архив А.П. Ершова. URL: http://www.ershov.ras.ru/russian/second_literacy/article.html (дата обращения: 12.04.2010)
2. Сайт сообщества Scratch. URL: http://scratch.mit.edu/ (дата обращения: 12.04.2010)
3. Сайт сообщества Squeak. URL: http://www.squeak.org/ (дата обращения: 12.04.2010)
4. Сайт сообщества Star Logo. URL: http://education.mit.edu/starlogo/ (дата обращения: 12.04.2010)
5. Сайт сообщества Netlogo. URL: http://ccl.northwestern.edu/netlogo/ (дата обращения: 12.04.2010)
6. Сайт сообщества Alice. URL: http://www.alice.org/ (дата обращения: 12.04.2010)
7. Сайт сообщества Greenfoot. URL: http://www.greenfoot.org/ (дата обращения: 12.04.2010)
8. Сайт сообщества Pygame. URL: http://www.pygame.org/ (дата обращения: 12.04.2010)
9. Сайт сообщества Game Maker. URL: http://www.slw-soft.com/ (дата обращения: 12.04.2010)
10. Сайт сообщества Little Wizard. URL: http://littlewizard.sourceforge.net/ (дата обращения: 12.04.2010)
11. Иенсен К., Вирт Н. Паскаль: Руководство для пользователя. - М.: Компьютер, 1993. – 256 с.
12. Вирт Н. Алгоритмы + структуры данных = программы. – М.: Мир, 1985. – 189 с.
13. Любимский Э. З., Мартынюк В. В., Трифонов Н. П. Программирование. – М.: Наука. Главная редакция физико-математической литературы, 1980. – 608 с.
14. Уинер Р. Язык Турбо Си. – М.: Мир, 1991. – 384 с.
15. Семакин И. Г., Шестаков А. П. Основы программирования: Учебник. – М.: Мастерство; НМЦ СПО; Высшая школа, 2001. – 432 с.

Исследования выполнены при поддержке Федерального агентства по образованию в рамках реализации аналитической ведомственной целевой программы «Развитие научного потенциала высшей школы» (2009–2010 годы) (№3.1.2/4125).

Сведения об авторе: Дженжер Вадим Олегович, доцент кафедры администрирования информационных систем Оренбургского государственного университета, кандидат физико-математических наук, доцент
460018, г. Оренбург, пр-т Победы, 13, ауд. 1502, тел. (3532)372539
e-mail: vdjenjer@yandex.ru

Dzhenzher V.O.
The place of programming in the course of the information theory of elementary school
The author examined the prevailing approaches to the instruction of programming and proposed the method of instruction in programming in the elementary school.
The key words: information theory, programming, elementary school, Scratch.

Bibliography:
1. Ershov A. P. Programming is the second literacy. // A. P. Ershov Archive. URL: http://www.ershov.ras.ru/russian/second_literacy/article.html (last accessed: 12.04.2010) (in Russian)
2. The Scratch Comunity Site. URL: http://scratch.mit.edu/ (last accessed: 12.04.2010)
3. The Squeak Comunity Site. URL: http://www.squeak.org/ (last accessed: 12.04.2010)
4. The Star Logo Comunity Site. URL: http://education.mit.edu/starlogo/ (last accessed: 12.04.2010)
5. The Netlogo Comunity Site. URL: http://ccl.northwestern.edu/netlogo/ (last accessed: 12.04.2010)
6. The Alice Comunity Site. URL: http://www.alice.org/ (last accessed: 12.04.2010)
7. The Greenfoot Comunity Site. URL: http://www.greenfoot.org/ (last accessed: 12.04.2010)
8. The Pygame Comunity Site. URL: http://www.pygame.org/ (last accessed: 12.04.2010)
9. The Game Maker Comunity Site. URL: http://www.slw-soft.com/ (last accessed: 12.04.2010)
10. The Little Wizard Comunity Site. URL: http://littlewizard.sourceforge.net/ (last accessed: 12.04.2010)
11. Jensen K., Wirth N. Pascal: The user guide. – M.: Computer, 1993. – 256 p. (in Russian)
12. Wirth N. Algorithms + Data Structures = Programms. – M.: Mir, 1985. – 189 p. (in Russian)
13. Lubimskiy E. Z., Martynuk V. V., Trifonov N. P. The Programming. – M.: Nauka, 1980. – 608 p. (in Russian)
14. Wiener R. The Turbo C Language. – M.: Mr, 1991 – 384 p. (in Russian)
15. Semakin I. G., Shestakov A. P. The Base of Programming. – M.: Masterstvo; NMC SPO; Vyshaya shkola, 2001. – 432 p. (in Russian)