Общество: социология, психология, педагогика. 2023. № 2. С. 143–151.
Society: Sociology, Psychology, Pedagogics. 2023. No. 2. P. 143–151.

Научная статья
УДК 373:372.800.4
https://doi.org/10.24158/spp.2023.2.19

Выбор инструментальной среды для раннего обучения программированию

Владимир Анатольевич Локалов[1], Игорь Викторович Климов[2], Андрей Сергеевич Миронов[3], Анастасия Геннадьевна Лунёва[4]
[1,2,3,4]Национальный исследовательский университет ИТМО, Санкт-Петербург, Россия
[1]lokalov@itmo.ru, https://orcid.org/0000-0002-4239-6555
[2]igor@itmo.ru, https://orcid.org/0000-0001-9938-5569
[3]asmironov@itmo.ru, https://orcid.org/0000-0001-5336-8800
[4]agluneva@itmo.ru, https://orcid.org/0000-0002-8914-4400

Аннотация. В статье описываются основные подходы к выбору инструментальной среды и языковых средств для раннего обучения детей программированию (с 6–7 лет) в системе дополнительного образования. Анализируется влияние особенностей изучаемых инструментальных сред на риск формирования навыков, оказывающих негативное влияние на дальнейшее профессиональное развитие учащихся в области программирования. Даются рекомендации по выбору инструментальной среды и языка программирования, которые способствуют направленному формированию базовых профессиональных навыков и адекватной профессиональной ориентации учащихся в области программирования. К таким рекомендациям, в частности, относятся разнообразие изучаемых инструментальных сред, освоение учащимися компонентов императивной методологии программирования, фиксация внимания учащихся на необходимости стадии проектирования программ. Описывается практическое применение предлагаемых рекомендаций на вводных курсах Детско-юношеского компьютерного центра Университета ИТМО (ДЮКЦ ИТМО).

Ключевые слова: методика преподавания программирования, раннее обучение программированию, язык программирования, инструментальная среда разработки программ

Для цитирования: Локалов В.А., Климов И.В., Миронов А.С., Лунёва А.Г. Выбор инструментальной среды для раннего обучения программированию // Общество: социология, психология, педагогика. 2023. № 2. С. 143–151. https://doi.org/10.24158/spp.2023.2.19.

Original article

Selecting a Tool Environment for Early Programming Training

Vladimir A. Lokalov[1], Igor V. Klimov[2], Andrei S. Mironov[3], Anastasia G. Luneva[4]
[1,2,3,4]ITMO University, Saint Petersburg, Russia
[1]lokalov@itmo.ru, https://orcid.org/0000-0002-4239-6555
[2]igor@itmo.ru, https://orcid.org/0000-0001-9938-5569
[3]asmironov@itmo.ru, https://orcid.org/0000-0001-5336-8800
[4]agluneva@itmo.ru, https://orcid.org/0000-0002-8914-4400

Abstract. The article outlines the main approaches to the selection of a tool environment and language tools for early programming training of children (from 6–7 years old) in the system of supplementary education. The influence of the features of the studied tool environments on the risk of developing skills that have a negative impact on the further vocational development of students in the field of programming is analyzed. Recommendations on the selection of a tool environment and a programming language that contribute to the directed formation of basic vocational skills and proper professional orientation of students in the field of programming are given. These recommendations, in particular, include: a variety of studied tools environments, assimilation the components of an imperative programming methodology, fixing students' attention on the need for the program design stage. It describes the practical application of the proposed recommendations in the introductory courses of the ITMO University Children and Youth Computer Centre (ITMO CYCC).

Keywords: programming teaching methodology, early programming training, programming language, software development tool environment

For citation: Lokalov, V.A., Klimov, I.V., Mironov, A.S. & Luneva, A.G. (2023) Selecting a Tool Environment for Early Programming Training. *Society: Sociology, Psychology, Pedagogics.* (2), 143–151. Available from: doi:10.24158/spp.2023.2.19 (In Russian).

Введение. Раннее обучение программированию. В последнее время в системе дополнительного образования детей (ДОД) увеличивается число курсов, кружков и факультативов, которые предлагают обучать детей программированию начиная чуть ли не с дошкольного возраста. Родители, которые желают, чтобы их дети как можно раньше освоили программирование, неизбежно сталкиваются с проблемой выбора подходящей для этого образовательной программы ДОД, поскольку варианты инструментальных сред и языков, предлагаемых для изучения, чрезвычайно разнообразны (Scratch, Minecraft, Roblox, Python, LOGO, JavaScript и т. д.). Отметим, что это разнообразие является свидетельством того, что на сегодняшний день и в среде разработчиков дополнительных образовательных программ также еще не сложилось единое представление о том, какую инструментальную среду следует выбрать, чтобы обучать ребенка азам программирования, при условии, что у него еще в должной степени не сформировалось абстрактно-понятийное и системное мышление.

Общее теоретическое и практическое решение этой проблемы было предложено еще в XX в. Сеймуром Пейпертом – изобретателем языка LOGO. Он полагал, что одним из важнейших направлений развития современной педагогики является «создание условий, при которых могут возникать интеллектуальные модели», позволяющие учащимся «ассимилировать» (усваивать) новые знания и умения (Пейперт, 1989: 10).

Описывая язык LOGO как инструмент, благодаря которому у учащихся в голове должны сформироваться указанные модели, С. Пейперт подчеркивал, что аффективная сторона процесса обучения не менее важна, чем его когнитивная сторона. Это означает, что программирование на LOGO должно приносить ребенку удовольствие и являться увлечением, благодаря которому плодотворные идеи, возникшие при написании программ, он в дальнейшем захочет перенести на новые объекты. В дальнейшем это будет неизбежно приводить как к совершенствованию самих интеллектуальных моделей, так и к поиску более совершенных языковых средств.

Разработка LOGO как специального языка программирования для детей базировалась на теории Ж. Пиаже, согласно которой умственное развитие детей 7–11 лет находится на стадии конкретных операций (2004: 136). На этом этапе дети еще не в состоянии использовать профессиональные языки программирования, требующие отвлеченного мышления. Написание программы на языке LOGO имитирует процесс разговора с конкретным образом исполнителя. По замыслу автора, программируя, ребенок, обращаясь к этому образу (изначально – черепашке) и заставляя его выполнять команды, учится разговаривать с компьютером.

Нужно подчеркнуть, что любая реализация языка LOGO обязательно должна включать в себя визуальную инструментальную среду, позволяющую в реальном времени отображать операции, выполняемые исполнителем. Наглядность образа исполнителя, процесса и результата его работы, а также понятность команд позволяют использовать данный язык для изучения программирования начиная с младшего школьного и даже дошкольного возраста (при условии, конечно, что дошкольник уже научился читать и писать).

С течением времени стали появляться языки программирования, близкие по концепции языку LOGO, однако в значительной степени отличающиеся от LOGO как по способу взаимодействия ребенка и компьютера, так и по функционалу, предоставляемому ребенку инструментальной средой программирования. К таким языкам, безусловно, относится Scratch.

Среда Scratch, как и среда LOGO, также ориентирована на раннее изучение программирования. В ней создаются программы, управляющие действиями исполнителей – спрайтов. Спрайты могут иметь разнообразные графические представления («костюмы»). Важным сходством Scratch и LOGO является нацеленность на аффективную сторону процесса создания программы. Ребенку предоставляется ряд графических инструментов, позволяющих реализовать свои идеи (прежде всего идеи по созданию первых игр), а также средства коммуникации, с помощью которых он может делиться достижениями в сетевом сообществе разработчиков программ на Scratch (сайт Scratch.mit.edu), что, по идее, должно служить средством дополнительной мотивации совершенствования в области программирования.

Заметим, что концепция Scratch в отличие от концепции LOGO ушла от имитации разговора с компьютером. Процесс программирования фактически стал напоминать сборку управляющих пазлов, что существенно упростило разработку программ. Ребенку теперь не нужно набирать текст и отслеживать правильность синтаксических структур. Соблюдение правил построения программы на Scratch легко проверить с помощью визуальной оценки того, можно ли присоединить один модуль к другому.

Появление средств, подобных Scratch, связано не столько с возрастающей потребностью в программистских кадрах, сколько со становлением глобальной парадигмы компьютерного образования (Computing Curricula…, 2020), согласно которой умение программировать является важнейшей составляющей компьютерной подготовки. Поэтому в настоящее время в Интернете

можно встретить образовательные ресурсы, целью которых выступает обеспечение доступа к изучению программирования и информатики широкому кругу лиц, практически без возрастных и социальных ограничений. К таким ресурсам, в частности, относится сайт Code.org, где даже четырехлетние дети могут начать составлять свои первые программы. Каждый обучающий раздел этого ресурса разделен на последовательность шагов, и на каждом шаге от ребенка требуется найти решение задачи-головоломки в виде алгоритма. Эти задачи формулируются в понятных и близких детям контекстах любимых мультфильмов и компьютерных игр. Например, дети могут помогать Angry Birds выбираться из лабиринта или рисовать для Анны и Эльзы из мультфильма «Холодное сердце» заданные узоры на льду.

Популярные персонажи, понятные простые задачи, а также мультимедийные подкрепления каждого успешно выполненного алгоритма способствуют тому, чтобы процесс программирования на Code.org приобрел положительно эмоциональную окраску для ребенка. Как и в Scratch, в качестве инструментального средства для «сборки» программ здесь нужно использовать набор визуальных блоков, содержащих команды. Однако сами команды являются не стандартными командами какого-либо языка программирования, а специфическими для каждой из предлагаемых ребенку задач. Детям, не умеющим читать, предлагаются условные графические обозначения команд (например, стрелкой обозначается перемещение на один шаг в соответствующем направлении).

Идея привлечения детей к написанию программ через мотивирующий контекст в настоящее время распространена. Ряд ее реализаций связан с построением инструментальных сред или специализированных библиотек, позволяющих детям писать разнообразные расширения для любимых игр, используя профессиональный язык программирования. В частности, желание научиться разрабатывать собственные коллекции модов для «Майнкрафт» может мотивировать ребенка на изучение такого языка программирования, как Java.

Отдельная версия игры Майнкрафт – Minecraft Education Edition (сайт Education.minecraft.net) – специально создана для того, чтобы на ее основе была возможность реализовать обучающие курсы, в том числе по программированию. В качестве готовых методических решений предлагаются серии уроков по программированию, в которых осуществлен постепенный переход от блочного программирования (типа Scratch) к написанию текстов на языках JavaScript и Python.

Стоит упомянуть еще одну игровую среду, в составе которой есть средства, используемые в настоящее время для начального обучения программированию: Roblox. В Roblox интегрированы:

– инструментальные средства для создания игр;
– трехмерный редактор;
– средства разработки и отладки скриптов на языке Lua;
– разработанные игры;
– средства для коммуникаций членов сообщества Roblox;
– средства коммерциализации разработок членов сообщества.

Несмотря на то что программирование скриптов в Roblox сложнее, чем в описанных ранее средах (не используется блочное программирование, не всегда очевидная система программируемых игровых объектов, событий и т. п.), эту среду часто используют для начального обучения программированию, однако ее обычно предлагают для изучения школьникам начиная примерно с 10 лет.

Итак, чтобы научить детей, не обладающих в должной степени развитым отвлеченным мышлением писать программы, предлагается та или иная визуально-инструментальная среда, благодаря которой дети получают возможность решать задачи (составлять программы) на уровне конкретных операций. Эти задачи должны формулироваться в понятном и эмоционально значимом для детей контексте. Поэтому в настоящее время для начального обучения программированию активно используются игровые инструментальные среды.

Риски раннего обучения программированию. Несмотря на ряд положительных моментов использования той или иной инструментальной среды для раннего обучения программированию, такое обучение может быть сопряжено с рисками возникновения проблем на пути дальнейшего профессионального развития учащихся в области программирования.

Прежде всего необходимо отметить, что ребенок, который начал изучать программирование в игровых средах, может неадекватно понимать сам термин «умение программировать». Это искаженное восприятие во многом провоцируют и авторы образовательных программ, которые включают слово «программирование» в название курсов («Программирование в “Майнкрафт”», например, на сайте Itgen.io). В результате обучения на таких курсах учащиеся, которые сумели написать лишь несколько строк программного кода, причем, как правило, с помощью готовых библиотек, или даже просто смогли соединить ряд управляющих блоков, думают, что они уже

умеют программировать, поскольку их программы «заработали». Причина такой оценки во многом объективна. Она состоит в том, что на ранних этапах обучение может вестись только с помощью упрощенных инструментальных средств. Подобное упрощение не влекло бы за собой никаких неприятных последствий и считалось бы просто первой фазой на пути реализации дидактического принципа от простого к сложному, если бы все фазы перехода к сложному были бы методически разработаны и реализованы. В реальности ввиду отсутствия полноценных методических и практических разработок, которые позволяли бы гарантированно осуществить пошаговый переход от «детского» программирования к «взрослому», мы имеем дело с противоположной ситуацией, которая теоретически может привести к риску формирования автоматических неразвитых форм действий (навыков «детского стиля программирования»), в последствии способных препятствовать дальнейшему развитию полноценных профессиональных действий.

Проанализируем основные типы упрощений, которые используются в описанных ранее инструментальных средах программирования в целях выявления их возможного влияния на профессиональное развитие учащихся в области программирования.

В наибольшей степени теоретически обоснованными (с точки зрения учета особенностей детского мышления) выглядят способы упрощения, используемые в среде LOGO в ее классическом варианте, т. е. в среде, которая включает в себя интерпретатор, редактор кода и средства отображения действий черепашки. Этот язык, как уже было сказано, создан для того, чтобы ребенок учился разговаривать с компьютером и составлял программы, опираясь на свое развитое конкретное мышление. Объявление новой процедуры в LOGO имитирует процесс объяснения того, как нужно выполнять новую команду. Например, в некоторых русских реализациях LOGO чтобы исполнитель (чаще всего черепашка) нарисовал квадрат, ему нужно сначала сказать «это квадрат» или «выучи квадрат» и далее после списка параметров в скобках необходимо перечислить все команды, выполнение которых «научит» компьютер (черепашку) рисовать квадраты (английский вариант – to square {...}).

Подобные языковые структуры позволяют рано познакомить учащихся с принципом модульной организации программ, а также приучить их использовать метод пошаговой детализации для разработки алгоритма. Одним из самых существенных недостатков языка LOGO является отсутствие типизации данных, что в общем естественно для «разговорного» типа общения. Если ребенок «говорит» компьютеру: «Нарисуй квадрат со стороной 40», то он вряд ли в таком разговоре сочтет целесообразным уточнять, является ли 40 вещественным или целым числом, тем более что указанные понятия могут быть вообще ему не знакомы. Как показывает опыт работы, сложности при изучении LOGO начинают возникать, когда ребенку пытаются объяснить необходимость применения переменных и параметров при описании процедур. Визуально-конкретная задача, например нарисовать квадрат со стороной 40, препятствует возникновению отвлеченного представления о том, что с помощью одних и тех же операций можно рисовать любые равносторонние фигуры со стороной любого размера.

Известно, что со временем определенный способ решения задач перерастает в привычку, от которой впоследствии трудно избавится. Отсутствие размышлений на этапе анализа задачи по поводу структур данных, используемых для ее решения, приводит к риску автоматизации этого процесса и проблемам при решении сложных задач с применением типизированных языков программирования, таких, например, как Java или C++. Совершенно очевидно, что ребенку, привыкшему к LOGO, потребуются неимоверные усилия, чтобы понять, что может означать такое понятие, как структура данных или объект.

Язык Scratch, без сомнения, можно рассматривать как некое упрощение языка LOGO, и поэтому его использование для начального обучения приводит к тем же рискам, что и применение LOGO. Дополнительные упрощения языка Scratch обусловливают также дополнительные проблемы, которые все больше отдаляют учащегося от профессионального стиля написания программ и понимания им основных концепций программирования.

В частности, то, что учащийся собирает пазл-программу, а не набирает ее текст вручную, приводит к тому, что его внимание направлено на контроль в первую очередь геометрического соответствия программных блоков, а не правильности синтаксических структур. При сборке пазла программы синтаксическая ошибка воспринимается как отсутствие возможности стыковки двух визуальных элементов, а не как нарушение синтаксических правил. Для ликвидации подобной ошибки детям совсем не обязательно знать эти правила. Более того, они даже могут визуально не оценивать возможность стыковки программных блоков. Для этого подходит и простой подбор нужного элемента на основе метода проб и ошибок.

Не лучше дело обстоит и с пониманием учащимися того, что такое структуры данных. Упомянутые минусы, связанные с отсутствием типизации переменных, усугубляются еще и тем, что, программируя в Scratch, обучаемые даже не могут познакомиться со всеми функциями и свойствами

переменных. Поскольку Scratch не предоставляет возможности определять программные модули (процедуры и/или функции), учащиеся принципиально не могут применять переменные в качестве параметров этих модулей. В программах на Scratch детей обычно учат использованию переменных для организации циклов, условных переходов, хранения текущих состояний управляемого объекта, а также запоминания различных состояний игрового окружения (например, счета в игре).

Серьезные и, казалось бы, качественные отличия можно было бы ожидать, когда для раннего обучения программированию пытаются использовать такие языки, как Python, JavaScript или даже Java. Однако знакомство с возможностями указанных языков чаще всего предлагается осуществлять исключительно в игровом контексте. Это означает, что разрабатываемые программы нацелены преимущественно на решение очевидных и понятных ребенку игровых задач и не предполагают какого-либо этапа проектирования, в котором возможен альтернативный выбор инструментального решения.

Так, в среде Minecraft Education Edition, которая, в частности, предназначена для знакомства с такими языками программирования, как JavaScript и Python, написание кода – лишь средство, которое учащиеся учатся использовать для достижения тех или иных игровых целей. На каждый новый урок ставится новая игровая цель, для достижения которой изучается новое языковое средство (например, структура управления или данных). Поэтому нельзя удивляться следующему фрагменту тематического планирования : «день четвертый – циклы while:

– циклы while со счетчиком;

– циклы while с использованием булевой логики;

– проклятье, цветочный след, состязание ныряльщиков, танцпол, сообщения в чате».

В результате такого подхода появляется риск того, что в уме учащихся инструментальные средства соответствуют не какому-то классу типовых задач, а конкретным игровым целям.

На сайте Codekingdoms.com предлагаются курсы программирования, на которых детей также учат создавать программы для игр Roblox и Minecraft, но уже на языке Java. Уточним, что речь снова идет не о программировании в его классическом понимании, а о некоторой среде, позволяющей детям на базе множества ориентированных на игру шаблонов, включающих основные структурные элементы языка Java, легко составлять программу, работоспособность которой сразу можно проверить на соответствующем игровом сервере. Главными компонентами такой среды являются следующие:

– видеоуроки, в которых объясняется игровой сюжет, связанный с дальнейшим кодированием;

– «умная» консоль, на которую выводятся подсказки и сообщения об ошибках, а также окно прогресса в изучении учебных разделов;

– библиотека структурных элементов языка, из которой можно выбирать различные языковые конструкции;

– инструментальная панель для отладки разработанной программы на игровом сервере (здесь же есть кнопка переключения в чисто текстовый вариант);

– рабочее пространство для написания кода.

На рабочем пространстве в начале каждого занятия помещается языковая заготовка, которую детям нужно доработать, следуя инструкциям. Там же видны строки с подключаемыми библиотеками, а также начало определения класса, которое учащимся необходимо завершить (добавить свои переменные и методы), чтобы решить поставленную игровую задачу. Строки с подключением библиотек сделаны малозаметными, чтобы учащиеся, которым смысл этих строк не объясняется, не обращали на них особого внимания.

Несмотря на то что Java является типизированным языком, необходимость типизации, а также ее смысл в контексте описанной методики не могут быть осмыслены ребенком. Она скорее будет представляться ему помехой, ненужным компонентом для решения игровых задач, чем свойством языка программирования, которое требуется для четкого описания и корректной обработки данных. Возможно, конечно, что присутствие непонятных строк в исходном шаблоне программы учащийся начнет фиксировать в памяти и в дальнейшем захочет узнать, зачем они нужны. Однако никто не может гарантировать, что такой интерес обязательно возникнет, особенно если он не будет вызван какой-то необходимостью.

Заметим также, что преимущественная направленность на игровой результат не только не создает условия для того, чтобы учащиеся задумались о семантике всех предлагаемых им языковых структур, но и отвлекает внимание учащихся от следующего:

– именно специально разработанные библиотеки, а не язык в его «чистом» виде, предоставляют возможность решать поставленные задачи;

– разработанный код, даже если он правильно выполняется, может быть разного уровня качества, в том числе различаться по таким параметрам, как надежность и эффективность.

Таким образом, можно утверждать, что использование типизированных языков для начальной стадии обучения программирования само по себе не является достаточным условием даже для более или менее приблизительного знакомства учащихся с тем, что такое профессиональный подход к разработке программ. Методики раннего обучения программированию не ориентированы и не предполагают получение результата, который бы в дальнейшем гарантировал как сдвиг мотива, связанного с игрой, на цель – освоение профессионального стиля программирования, так и формирование соответствующей понятийной системы.

Итак, основные риски, связанные с ранним изучением программирования состоят в следующем:

– отсутствие критичности как к уровню своей подготовки в области программирования, так и к качеству полученного результата (лишь бы программа работала);

– формирование привычки сразу начинать кодирование без предварительного обдумывания алгоритмического решения задачи, т. е. минуя этап проектирования;

– формирование привычки решать задачи «в лоб», без каких-либо попыток их обобщения и формализации;

– ослабление синтаксического и семантического контроля в процессе разработки программ.

Перечисленные риски показывают, что ни одна из описанных инструментальных сред в отдельности не может обеспечить плавный переход к профессиональному уровню программирования. Эти среды можно рассматривать только как вспомогательные инструменты, которые (при исключении возможных рисков) можно использовать на подготовительном этапе к изучению основ программирования. Они могут дать сильную внешнюю мотивацию к дальнейшему развитию учащихся в области программирования, а также познакомить их с написанием программного кода.

Выбор инструментальной среды. Предыдущий анализ показывает, что раннее обучение программированию может способствовать (если, конечно, исключить риски) формированию некоторых компонентов профессиональной подготовки программиста. Теперь необходимо определить, можно ли дать рекомендации по выбору инструментальной среды для ранних этапов обучения программированию, которые, с одной стороны, позволили бы обеспечить непрерывную мотивацию учащихся к изучению программирования на этих этапах, а с другой – способствовали бы направленному освоению ими методологии, с овладения которой обычно начинается подготовка профессионального программиста, а именно – методологии императивного программирования (Одинцов, 2004: 43).

Для решения задач в рамках императивной методологии программист должен овладеть следующими навыками:

– структурное кодирование;

– модульное проектирование;

– проектирование структуры данных.

Легко заметить, что любое блочное программирование, например на Scratch, способствует развитию понимания, что такое структурное кодирование на наглядном содержании. Визуальные блоки, которые реализуют три основные структуры управления (последовательность, ветвление и повтор), легко вкладываются друг в друга. Это позволяет постепенно переходить от простейших задач на последовательность действий к задачам, требующим более сложного алгоритмического решения, например ко вложенным циклам.

Однако блочное программирование не сможет помочь, когда нужно научить учащихся проектировать задачу «сверху вниз», поскольку в нем, как уже говорилось, отсутствуют более или менее приемлемые средства определения программных модулей типа процедур и функций с параметрами. Поэтому для ознакомления учащихся с тем, что такое модульное проектирование, на определенном этапе обучения следует отказаться от блочного программирования и перейти к инструментальным средам, позволяющим решать задачи с помощью основного метода в императивном программировании, а именно – метода пошаговой детализации. Для этой цели можно использовать языки, подобные LOGO. Например, чтобы объяснить черепашке, как рисовать домик, нужно сначала объяснить, как рисовать стены, затем крышу. Если нужно нарисовать город, то черепашке сначала следует объяснить, что город – это определенное количество разных домов, затем – что такое дом и т. д.

Процесс освоения методологии императивного программирования в средах, принципиально основанных на обработке событий, будет сталкиваться с определенными сложностями. Так, в Roblox императивную методологию программирования можно осваивать только на примерах написания отдельных скриптов, привязывая их к тому или иному локальному объекту в иерархической структуре объектов Roblox, отображающихся в специальном окне. Сдвиг фокуса внимания на разработку локального обработчика событий явно не способствует формированию

системного понимания взаимодействия игровых объектов, а также осознанию необходимости обдумывания структур данных, которые нужно определить для решения задачи хотя бы на уровне простого определения набора переменных (учащиеся не понимают, зачем использовать переменные, когда просто можно обойтись конкретными числовыми значениями). Это обстоятельство лишний раз доказывает, что обучение программированию в игровых средах, равно как и в других инструментальных средах, ориентированных на раннее изучение программирования, не позволяет направленно формировать представление учащихся о структурах данных.

Во избежание развития устойчивых неправильных навыков программирования, связанных с особенностями той или иной инструментальной среды для раннего изучения программирования, имеет смысл надолго не задерживаться на изучении каждой конкретной среды. Следует отдать предпочтение многообразию изучаемых сред, а не углубленности их изучения. Это, в свою очередь, будет способствовать развитию широкой ориентировочной основы умственных действий будущих программистов, которая особенно важна на этапах анализа и проектирования решения задачи еще до выбора программных средств ее реализации. Нужно отметить необходимость фиксации внимания учащихся на этом этапе, начиная с самых ранних стадий обучения программированию, при изучении любых инструментальных сред.

Формы организации этапа проектирования программ должны меняться при переходе от менее сложных алгоритмов к более сложным и целиком и полностью зависеть от предметной области, для которой разрабатываются программы и которая должна обеспечивать мотивацию учащихся. Для очевидных задач, чье решение сводится к перетаскиванию двух-трех командных блоков, вполне можно обойтись устным обсуждением. Сложные задачи требуют уже письменной фиксации алгоритма на естественном языке или в виде графической схемы, отражающей структуру решаемой задачи.

Итак, при выборе инструментальной среды для ранних этапов изучения программирования можно руководствоваться следующими рекомендациями:

– на разных этапах начального обучения программированию учащиеся должны познакомиться с несколькими инструментальными средами (от максимально упрощенных к минимально упрощенным);

– основной целью обучения в указанных средах должно быть освоение учащимися структурного кодирования и модульного проектирования;

– особое внимание следует уделять изучению той предметной области, для которой пишутся программы, поскольку она является фактором мотивации, а также этапу проектирования программ.

Практика начального обучения программированию в системе ДОД. В заключение на примере Детско-юношеского центра Университета ИТМО (ДЮКЦ ИТМО) покажем, как на основе изложенных рекомендаций можно организовать раннее обучение программированию в контексте не отдельного курса, а системы курсов, главной задачей которой является профессиональная ориентация учащихся в различных направлениях компьютерных технологий, а также развитие у них соответствующих базовых профессиональных навыков, в том числе в области программирования (Lokalov, 2014: 28).

В соответствии с концепцией обучения ДЮКЦ ИТМО система компьютерных курсов представляет собой трехуровневую структуру (Локалов, 2015: 199). На первом находятся вводные курсы, основное назначение которых – формирование широкого кругозора в области компьютерных технологий, позволяющего учащимся в дальнейшем осознанно выбирать более специализированные курсы, связанные с различными направлениями компьютерных технологий, в том числе с программированием (курс P – «Основы программирования»). Как уже отмечалось, изучение инструментальных сред, ориентированных на раннее обучение программированию, следует рассматривать только как подготовительный этап к изучению основ программирования, и поэтому эти среды изучаются на вводных курсах. Несмотря на то что знакомство с программированием является важным компонентом каждого из этих курсов, слово «программирование» отсутствует в их названиях, поскольку внимание учащихся должно быть сосредоточено прежде всего на предметной области, для которой они будут разрабатывать свои программы.

Так, на курсе BV («Введение в компьютерную графику») учащиеся разрабатывают на Scratch несложные игры, придумывают и рисуют персонажей, фоны для этих игр. На курсе BS («Введение в интернет-технологии и презентационную графику») этот же язык используется для создания динамических презентаций и интерактивных приложений. Курс BM («Введение в программное и аппаратное обеспечение компьютера») дает возможность познакомиться с примерами применения языков программирования и научиться разрабатывать небольшие программы

разного функционального назначения. С помощью LOGO генерируются узоры («черепашья» графика), на Python решаются несложные вычислительные задачи, а C# используется для программирования внешних устройств, которые учащиеся создают на платформе Arduino.

Методики обучения программированию на любом из вводных курсов должны быть нацелены на усвоение учащимися тех компонентов подготовки, которые будут востребованы на курсе «Основы программирования». К этим компонентам относятся следующие:

– анализ исходных данных (условий) решаемой задачи и разработка ее алгоритмического решения до набора программного кода;

– знание и соблюдение синтаксических правил при написании программы и поиске синтаксических ошибок;

– понимание ограниченной функциональности готовых программных инструментальных средств.

Кроме этого, курс BV позволяет сформировать умение работать с координатами компьютерного изображения и управлять движением графического объекта, а на курсе BM учащиеся получают представления о принципах работы аппаратного и программного обеспечения компьютера, в том числе представления о функциях процессора, оперативной памяти, структуре файлов и файловой системы.

В процессе проведения занятий по основам программирования (курс P) наблюдалась устойчивая связь между успешностью прохождения учащимися вводных курсов и успеваемостью на курсе P. Кроме того, был обнаружен ряд проблем в обучении (связанных с описанными рисками) у тех учащихся, которые до поступления на курс P проходили программирование в игровых средах или «программировали» преимущественно на основе готовых библиотек. Наиболее характерными проблемами для таких учащихся были следующие:

– неадекватная оценка своего умения программировать;

– трудности при формализации задачи и этапа проектирования;

– отсутствие привычки синтаксического и семантического контроля текста программы.

Указанные факты позволяют сделать предположение, что выполнение разработанных нами рекомендаций по организации раннего обучения программированию дают определенный результат. Тем не менее даже при мониторинге процесса развития базовых способностей, связанных с профессиональным программированием, не удается полностью избежать трудностей, возникающих при переходе из инструментальной игровой среды к профессиональным средам для разработки программ. Это обусловлено прежде всего иными мотивационными механизмами, которые должны включаться при данном переходе. В какой-то момент должен произойти сдвиг с мотива поиграть и получить удовольствие от игры на цель написать программу на хорошем профессиональном уровне и испытать много положительных эмоций от процесса программирования как такового и от осознания собственных возможностей как программиста. К сожалению, это происходит далеко не всегда.

Список источников:

Локалов В.А. Принципы организации профессионально ориентированной системы курсов для дополнительного образования школьников в области ИТ // Преподавание информационных технология в Российской Федерации : материалы 13-й Открытой всерос. конф. / отв. ред.: С.В. Русаков, Ю.А. Аляев. Пермь, 2015. С. 198–199.

Одинцов И.О. Профессиональное программирование. Системный подход. 2-е изд., перераб. и доп. СПб., 2004. 610 с.

Пейперт С. Переворот в сознании: дети, компьютеры и плодотворные идеи / пер. с англ. под ред. А.В. Беляевой, В.В. Леонаса. М., 1989. 224 с.

Пиаже Ж. Психология интеллекта. СПб., 2003. 191 с.

Computing Curricula 2020: Paradigms for global computing education. N. Y., 2020. 205 p. https://doi.org/10.1145/3467967.

Lokalov V.A. Children's computer club as an example of non-formal educational system in the field of informatics // Journal of International Scientific Publication: Educational Alternatives. 2014. Vol. 12. P. 27–37.

References:

Computing Curricula 2020: Paradigms for global computing education. (2020) New York, Association for Computing Machinery. Available from: doi:10.1145/3467967.

Lokalov, V.A. (2015) Principles of Organizing a Vocationally Oriented System of Courses for Supplementary IT Education of Schoolchildren. In: Rusakov, S.V. & Alyaev, Yu.A. (eds.) Teaching Information Technology in the Russian Federation : Proceedings of the 13th All-Russian Open Conference. Perm, Permskii Gosudarstvennyi Natsional'nyi Issledovatel'skii Universitet, 198–199. (In Russian)

Lokalov, V.A. (2014) Children's computer club as an example of non-formal educational system in the field of informatics. Journal of International Scientific Publication: Educational Alternatives. 12, 27–37.

Odintsov, I.O. (2004) Professional Programming. System Approach. 2nd ed. Revised and Extended. Saint Petersburg, BHV-Peterburg. (In Russian)

Papert, S. (1989) Revolution in Consciousness: Children, Computers and Fruitful Ideas. Moscow, Pedagogika. (In Russian)

Piaget, J. (2003) Psychology of intelligence. Saint Petersburg, Piter. (In Russian)

Информация об авторах

В.А. Локалов – кандидат педагогических наук, доцент, старший преподаватель факультета программной инженерии и компьютерной техники, Национальный исследовательский университет ИТМО, Санкт-Петербург, Россия.

https://elibrary.ru/author_items.asp?authorid=808204

И.В. Климов – старший преподаватель факультета программной инженерии и компьютерной техники, Национальный исследовательский университет ИТМО, Санкт-Петербург, Россия.

https://elibrary.ru/author_items.asp?authorid=983155

А.С. Миронов – преподаватель факультета программной инженерии и компьютерной техники, Национальный исследовательский университет ИТМО, Санкт-Петербург, Россия.

А.Г. Лунёва – ассистент факультета программной инженерии и компьютерной техники, Национальный исследовательский университет ИТМО, Санкт-Петербург, Россия.

Information about the authors

V.A. Lokalov – PhD in Pedagogy, Associate Professor, Senior Lecturer, Software and Computer Engineering Department, National Research University ITMO, St. Petersburg, Russia.

https://elibrary.ru/author_items.asp?authorid=808204

I.V. Klimov – Senior Lecturer, Software and Computer Engineering Department, National Research University ITMO, St. Petersburg, Russia.

https://elibrary.ru/author_items.asp?authorid=983155

A.S. Mironov – Lecturer, Software and Computer Engineering Department, National Research University ITMO, St. Petersburg, Russia.

A.G. Luneva – Assistant Professor, Software and Computer Engineering Department, National Research University ITMO, St. Petersburg, Russia.

Статья поступила в редакцию / The article was submitted 28.12.2022;
Одобрена после рецензирования / Approved after reviewing 18.01.2023;
Принята к публикации / Accepted for publication 21.02.2023.