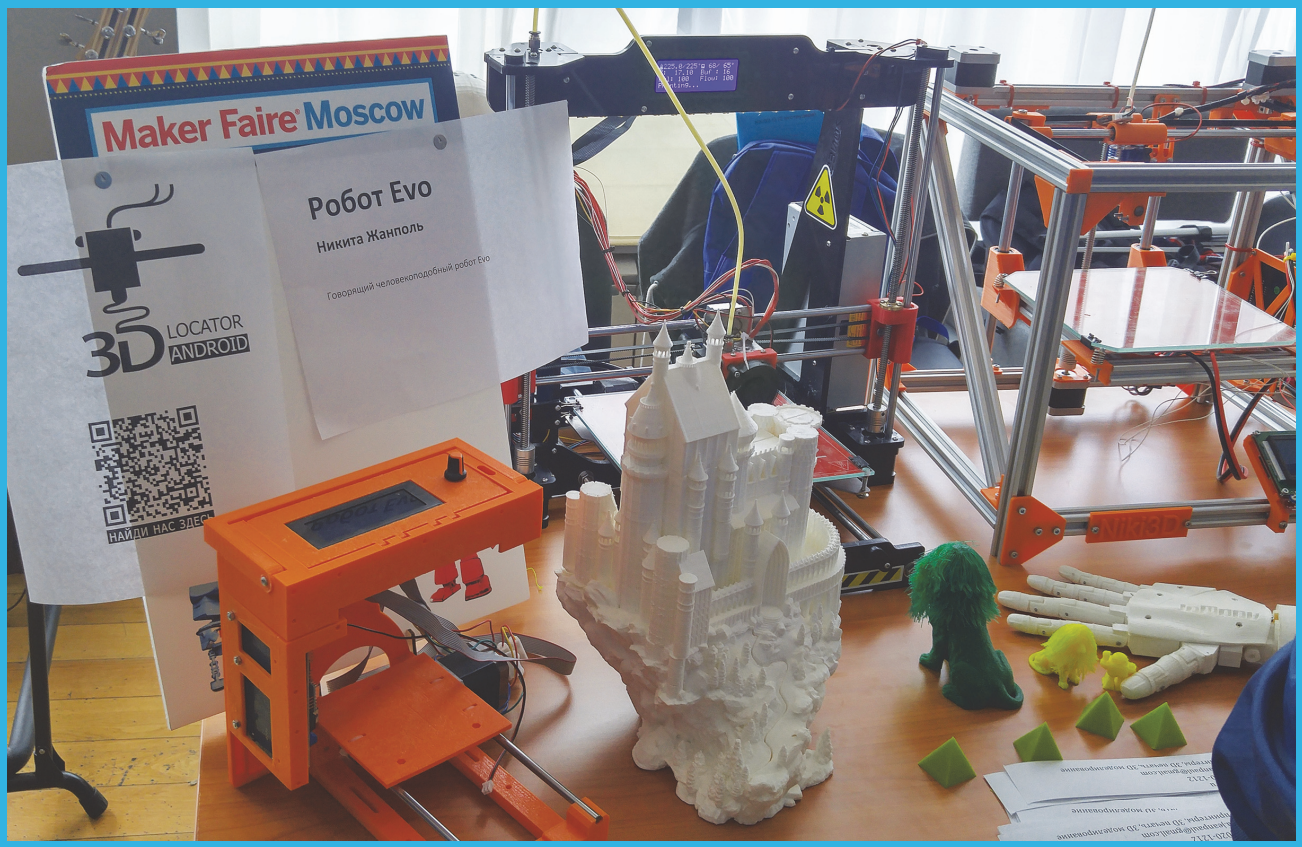
ОБУЧЕНИЕ ДЕТЕЙ ПРОГРАММИРОВАНИЮ: ЗАЛОГ РАЗВИТИЯ ЧЕЛОВЕЧЕСКОГО КАПИТАЛА В XXI ВЕКЕ

Руководство для российских законодателей и практиков в области образования

Сухас Д. Парандекар
Евгений Патаракин
Гульшан Яйла



ГРУППА ВСЕМИРНОГО БАНКА
Образование

ОБУЧЕНИЕ ДЕТЕЙ ПРОГРАММИРОВАНИЮ: ЗАЛОГ РАЗВИТИЯ ЧЕЛОВЕЧЕСКОГО КАПИТАЛА В XXI ВЕКЕ

Руководство для российских законодателей и практиков в области образования

ГРУППА ВСЕМИРНОГО БАНКА
Образование

Обучение детей программированию: залог развития человеческого капитала в XXI веке. Руководство для российских законодателей и практиков в области образования / Сухас Д. Парандекар, Е. Патаракин, Г. Яйла – Москва : Алекс (ИП Поликанин А.А.), 2019. – 164 с.

IНастоящая книга основана на предположении о том, что программирование является важным навыком XXI века, обязательным для всех. Изучение программирования включает не только изучение синтаксиса, грамматики и использования определенного языка программирования. Книга предназначена для высших должностных лиц федерального и регионального уровня, которые осознают, что российская система образования должна учитывать потребности, с которыми столкнутся дети, растущие в сетевом, взаимосвязанном мире развивающихся цифровых технологий и онлайн/офлайн взаимодействия. Книга также предназначена для учителей, родителей и других взрослых, которые хотят узнать больше о подготовке детей к жизни в XXI веке с теоретической и практической точек зрения.

Обложка: стенд на выставке Maker Faire Moscow, сентябрь 2017 года

alexpublishers.ru

ОГЛАВЛЕНИЕ

БЛАГОДАРНОСТИ

Настоящий отчет основан на работе, проделанной группой Всемирного банка (ВБ) при поддержке, одобрении и участии Национального исследовательского университета «Высшая школа экономики», благотворительного фонда Сбербанка «Вклад в будущее», организации «Рыбаков Фонд», НКО «ПроеКТОриЯ», Московского городского педагогического университета, проекта «РОББО», Ассоциации участников рынка артиндустрии. Группа ВБ работала под руководством Директора Всемирного банка Андраша Хорваи, старшего руководителя Глобальной практики в области образования Хайме Сааведры Чандуви, руководителя Глобальной практики в области образования Кейко Мивы, управляющего по практике Гарри Патриноса и руководителя программы и координатора программ в стране Дороты Агаты Новак.

Группу Высшей школы экономики возглавили Исак Фрумин и Мария Добрякова. Группу благотворительного фонда Сбербанка «Вклад в будущее» возглавила Юлия Чечец. Группа «Рыбаков Фонд» работала под руководством Бориса Ярмахова. Группу НКО «ПроеКТОриЯ» возглавили Василий Фокин и Яков Останин. За работу группы Московского городского педагогического университета отвечал Игорь Реморенко, за работу группы Ассоциации участников рынка артиндустрии — Наталья Михайлова.

Наряду с указанными выше экспертами, группа ВБ также хотела бы выразить благодарность другим специалистам за их ценные комментарии и вклад в проведение первых семинаров по этой теме в Москве: Айгуль Ашрафуллиной, Кириллу Баранникову, Ованесу Габриеляну, Сергею Григорьеву, Наринэ Исаханян, Павлу Фролову, Михаилу Молчанову, Оксане Петровой, Татьяне Пирог, Алексею Семенову, Андрею Сельскому, Ивану Симонову, Анне Труниной, Ирине Егоровой, Павлу Заикину.

Рабочей группой ВБ руководили старший экономист Сухас Парандекар и консультант Евгений Патаракин. В состав группы вошли консультант Гулькан Яйла, консультант Полина Завалина, консультант Елена Липилина, младший сотрудник категории специалистов Сидзука Кунимото.

Выражаем признательность экспертам-рецензентам Роберту Хокинсу, Игорю Хейфецу, Джулии Либерман, Хелене Ровнер и Майклу Трукано за их вклад в работу.

Выводы, интерпретации и заключения, сформулированные в этой и других частях данного доклада, выражают мнение только авторов доклада. Они не могут отражать точку зрения Международного банка реконструкции и развития / Всемирного банка, входящих в состав Банка организаций, а также исполнительных директоров Всемирного банка или правительств представляемых ими стран.

ПРЕДИСЛОВИЕ ГЛАВЫ ПРЕДСТАВИТЕЛЬСТВА ВСЕМИРНОГО БАНКА АНДРАША ХОРВАИ

Почти пятая часть XXI века уже стала историей. К кардинальным изменениям в мире ведет появление общедоступных вычислительных устройств, искусственного интеллекта и робототехники. Сфера образования находится в авангарде этих изменений как с точки зрения способа обеспечения обучения, так и с точки зрения его результативности, позволяющей образованным людям каждый день творить новые технологические чудеса. Образование меняется под влиянием технологий, при этом образование является основой новых технологий. Страны, осознающие надвигающиеся перемены в сфере образования, смогут стать конкурентоспособными и обеспечить своим гражданам всеми желаемое процветание. Российская Федерация стремится быть в числе лидеров в быстро формирующейся сфере образования будущего, и Всемирный банк готов поддержать эти преобразования в сотрудничестве с заинтересованными сторонами государственного и частного секторов.

Работа над книгой была начата в рамках взаимодействия Всемирного банка с Высшей школой экономики и благотворительным фондом Сбербанка «Вклад в будущее». В рамках этого взаимодействия Всемирный банк подготовил главу в новой книге «Ключевые компетенции и новая грамотность». Основное внимание в этой главе уделяется программированию как новой грамотности, рассматривается идея о том, что детям следует начинать изучать программирование с самого раннего возраста в детском саду и начальной школе по аналогии с тем, как в недавнем прошлом и, конечно, в XX веке, дети учились читать, писать и изучали числа. В ходе консультаций с заинтересованными сторонами в Российской Федерации и изучения опыта государств всего мира было найдено множество доступных материалов, которые будут полезны высшим должностным лицам и специалистам-практикам России.

Политическим деятелям будут интересны основные вопросы о цели обучения программированию, которая состоит не в том, чтобы обеспечить потребности рынка труда, а в том, чтобы помочь детям освоить такое понятие, как «вычислительное мышление», которое будет подробно описано в первой главе. Специалистов в области образовательных политик и образовательных практик заинтересуют темы, которые могут изменить мировоззрение учеников. Раньше ученикам говорили уделять внимание индивидуальной деятельности, сегодня большое значение приобрело понятие «ремикс» как способ творческого использования работы, выполненной другими людьми, признавая их вклад и предоставляя доступ к своей работе для создания на ее основе новых ремиксов. Оценивание в прошлом проводилось в форме тестирования, в рамках которого определялось знание учащимися правильных ответов на вопросы, сформулированные другими людьми. Однако если речь идет о программировании с точки зрения освоения вычислительного мышления, тему оценивания следует основательно пересмотреть. В книге, которую вы держите в руках или видите на экране перед собой, мы делаем попытку определить некоторые важные вопросы и представить ответы, если они существуют. Что на самом деле означает грамотность? Почему каждый должен освоить вычислительное мышление, и почему этот процесс необходимо начать в раннем возрасте? Может ли так случиться, что на рынке появятся десятки тысяч вакансий для программистов, которые некому будет заполнить, если детей не начать учить программированию сейчас? Или дети настолько загружены учебой, что будет негуманно нагружать их еще больше? Есть мнение, что в будущем программирова-

нием будут заниматься компьютеры, а не люди. А если так, то зачем сейчас заставлять детей учиться программировать? В этой книге мы рассматриваем эти вопросы и делаем попытке найти ответы. Заинтересованные читатели могут более детально изучить обширный список использованных источников и онлайн-приложение с описанием продуктов.

За последние несколько лет федеральное правительство и регионы вложили значительные средства в обучение XXI века. Ярким примером является создание сети «Кванториум» и технопарков для детей и юных духом людей. Частный сектор продолжает наращивать инвестиции и развивать продукты, а государственные инвестиции направлены на достижение целей равенства и справедливости. В этой книге представлена интересная информация и рекомендации для читателей, которые хотят узнать, куда направить эти инвестиции. Всемирный банк готов продолжить сотрудничество с заинтересованными сторонами в целях развития обучения XXI века в Российской Федерации.

АННОТАЦИЯ

Настоящая книга основана на предположении о том, что программирование является важным навыком XXI века, обязательным для всех. Изучение программирования включает не только изучение синтаксиса, грамматики и использования определенного языка программирования, например, Python или Ruby, но и освоение более глубокого понятия вычислительного мышления. Можно изучить аспекты вычислительного мышления и не научиться писать код, но программирование необходимо для получения четкого представления и знаний о вычислительном мышлении.

Книга предназначена для высших должностных лиц федерального и регионального уровня, которые осознают, что российская система образования должна учитывать потребности, с которыми столкнутся дети, растущие в сетевом, взаимосвязанном мире развивающихся цифровых технологий и онлайн/офлайн взаимодействия. Книга также предназначена для учителей, родителей и других взрослых, которые хотят узнать больше о подготовке детей к жизни в XXI веке с теоретической и практической точек зрения. Также целесообразно пояснить, чем эта книга не является — несмотря на отсылки к разнообразным научным работам по этой теме, которых становится все больше, книга не является научной работой, способной привнести новые знания. Большая часть таких научных работ не освещается средствами массовой информации и не попадает в фокус социальных сетей неспециалистов, являясь при этом весьма интересной и актуальной. Эта книга представляет собой попытку перевести некоторые из этих знаний в понятные для должностных лиц и специалистов-практиков термины без излишнего упрощения сложных реалий. Многочисленные ссылки, включая URL-адреса сайтов, позволят заинтересованному читателю глубже вникнуть в любую из обсуждаемых тем.

Книга состоит из пяти глав, описанных ниже.

В первой главе «Введение. Вычислительное мышление» описаны основные элементы, рассматриваемые в книге, начиная с объяснения значения вычислительного мышления. Утверждение о том, что каждый должен обладать определенным навыком или совокупностью знаний, необходимых для повседневной жизни, является образной или упрощенной отсылкой к понятию грамотности. Но что такое грамотность на самом деле? Почему каждый должен овладеть вычислительным мышлением, и почему этот процесс должен быть запущен в раннем возрасте? У концепции вычислительного мышления есть множество сторонников и, возможно, множество критиков. Одним из аргументов, используемым сторонниками и справедливо опровергаемым критиками, является огромный неудовлетворенный спрос на квалифицированных программистов, намечающийся на рынке труда. В этой главе авторы объясняют неправильность этого аргумента — то, что делает программирование необходимым навыком XXI века, выходит за рамки рынка труда. Вероятно, для любой работы в будущем и правда потребуется сочетание человеческих способностей, дополненных вычислительными приумножателями человеческого потенциала. Но настоящая причина, по которой следует изучать программирование и осваивать вычислительное мышление, — это улучшение общего умственного развития (как когнитивного, так и некогнитивного) и увлекательность самого процесса. Читатель, помимо прочтения этой книги, безусловно, откроет для себя радость программирования, окунувшись в жизнь своих детей. Признанный исследователь развития детства Джордан Шапиро, сотрудни-

чающий с Брукингским институтом и некоммерческой организацией Sesame Workshop, говорит о «совместном взаимодействии со средствами массой информации» — идее о том, что родители должны не бояться технологий, а играть в видеоигры вместе с детьми и воспитывать детей с помощью Интернета и социальных сетей. Этот совет можно наиболее успешно применить только в процессе программирования со своими детьми или детьми ваших друзей, племянниками, племянницами или внуками. Чтобы поверить, нужно попробовать!

Во второй главе «Успешный полёт: пять вариантов реализации» обсуждение переходит от теории к практике. Если программирование и правда такой замечательный навык, многие страны, несомненно, уже бы начали реализацию стратегий, направленных на успешное вовлечение детей. Это действительно так. Разработки в этой области проводятся довольно быстро, поэтому любая информация о достижениях, скорее всего, быстро станет неактуальной. Каждый месяц, если не чаще, проходят международные семинары и практикумы по этой теме, и в рамках второй главы мы можем представить только беглый обзор некоторых процессов и вопросов, вызывающих озабоченность. Образ парящей птицы помогает понять пять составляющих: программирование в рамках учебной программы или внеклассного образования — это две головы птицы, учителя — это одно крыло, содержание и контекст обучения — второе крыло, а хвост — это оценивание, определяющее направление его полета. Раздел, посвященный оцениванию, является наиболее академическим разделом книги, поскольку представляет читателям знания с передовых рубежей современных педагогических исследований. На практике потенциал оценивания вычислительного мышления еще не настолько реализован, чтобы радикально изменить подход не только к освоению вычислительного мышления, но и ко всему процессу обучения. Этот раздел будет полезен любому взрослому, который задумывался «зачем вы задаете мне этот вопрос, если вы уже знаете ответ», хотя его будет трудно понять, если бегло ознакомиться с представленными в нем техническими материалами. Программа исследований для разработки принципов оценивания может быть весьма полезной, и в этой главе рассматриваются некоторые возможные варианты.

В третьей главе «Многообразие предложений на рынке» приводится обзор огромного разнообразия доступных на рынке продуктов, созданных для того, чтобы помочь детям научиться программировать и приобрести навыки вычислительного мышления. Необходимо признать, что на этот обзор повлияли успехи языка Scratch. Подобно тому как среди множества поисковых систем есть Google, а среди торговых площадок есть Amazon, есть множество средств обучения программированию и есть среда языка Scratch. Язык Scratch имеет столь богатую предысторию и такую глубокую и обширную программу исследований и применения результатов исследований, что мы могли бы сосредоточиться только на его изучении. Языку Scratch посвящено множество книг, и мы приводим многочисленные цитаты из книги Митча Резника «Спираль обучения. 4 принципа развития детей и взрослых», которая объясняет философию Scratch. Scratch — это программное обеспечение с открытым исходным кодом, которое доступно бесплатно любому ребенку или взрослому при наличии подключения к Интернету или в автономном режиме. Кроме Scratch, существует множество открытых и бесплатных обучающих сред и обучающих продуктов, доступных за определенную плату. Большинство продуктов доступно по всему миру, и довольно много продуктов доступно для русскоязычных пользователей. Российские предприниматели также проявляют креативный подход, и мы включили в нашу книгу раздел, посвященный продуктам, разработанным специально для русскоговорящих пользователей. В этой главе в основном содержится справочная информация, чтобы заинтересованные читатели могли изучить ссылки и информацию о продукте, однако здесь также представлена концепция позитивного развития технологий, первоначально предложенная Марианой Умащи-Берс.

В четвертой контентно-ориентированной главе «Креативное программирование в России» описывается развитие вычислительного мышления и программирования в России. Содержание этой главы будет знакомо ученым и другим специалистам, изучавшим эту тему, однако широкая аудитория также сможет найти здесь интересную информацию. Уже на заре компьютерной эры были дальновидные специалисты, которые осознавали огромный потенциал компьютера как инструмента формирования у детей нового понимания мира и нового способа вовлечения детей в творческую деятельность. Одной из ключевых фигур в этой сфере был работавший в Массачусетском Технологическом Институте (MIT) уроженец Южной Африки Сеймур Пейперт, пытавшийся при помощи языка Logo совершить переворот в образовании. В этой главе мы рассказываем историю Андрея Петровича Ершова — другого выдающегося ученого, продвигавшего идею программирования для детей в Советском Союзе. Здесь также представлены федеральные государственные образовательные стандарты (ФГОС), связанные с программированием, и история Scratch в России. Поскольку целью рабочей группы Всемирного банка было не просто создание теоретического руководства и описание вычислительного мышления, то при содействии партнеров из разных регионов России зимой 2018 года группа провела экспериментальный конкурс Scratch Collab Challenge. В четвертой главе представлены основные выводы этого эксперимента.

Пятая и последняя глава называется «Основные рекомендации». Даже бегло просмотрев содержание этой книги, можно понять, что здесь представлен довольно широкий спектр информации о вычислительном мышлении и программировании. Последняя глава обобщает практические выводы, сделанные на основе всего многообразия представленной информации. Пусть это и заключительные слова книги, но мы надеемся, что это не заключительные слова обсуждения, которое должно продолжаться на разных форумах в Российской Федерации и за ее пределами. Обширная область применения технологий в сфере образования является одной из наиболее динамичных областей деятельности. Отслеживать развитие идей экспертов Всемирного банка вне рамок этой книги можно по ссылке https://www.worldbank.org/en/topic/edutech.

Обобщение основных рекомендаций / выводов для высших должностных лиц и специалистов-практиков

1. Развитие вычислительного мышления через изучение программирования должно стать одной из ключевых образовательных целей для детей всех возрастов.

2. Программирование следует изучать в условиях совместной деятельности и на основании проектов в рамках классного и внеклассного образования. Реализовать программы будет проще в рамках внеклассного образования.

3. Задача учителей состоит в том, чтобы стимулировать новый тип вычислительного мировоззрения, а не обеспечивать подготовку квалифицированных специалистов-компьютерщиков. Такую трансформацию мировоззрения будет проще обеспечить в рамках сообщества ответственных и приверженных своему делу учителей.

4. Программирование лучше всего включить в учебную программу на комплексной основе в качестве сквозной темы. Ученики не должны изучать какой-то один язык программирования, например, Python или JavaScript, а должны сосредоточиться на освоении и применении навыков вычислительного мышления для изучения таких предметов, как физика, история или язык.

5. Оценивание вычислительного мышления следует организовывать не так, как традиционное оценивание. Программирование обеспечивает оперативную обратную связь, поэтому оценивание необходимо в качестве поддержки ученика и учителя в выборе наиболее подходящего процесса обучения.

6. У родителей и других членов семьи есть широкие возможности, чтобы внести свой вклад в развитие у детей навыков вычислительного мышления — необходимо создавать программы в рамках центров внеклассного обучения, таких как кванториумы и технопарки, чтобы родители могли приобрести необходимые знания и навыки.

7. Рынок образовательных продуктов для развития вычислительного мышления переживает бурный рост. Государство может принять меры и поддержать развитие этого рынка посредством институциональной и финансовой поддержки стартапов, а также поддержать ВУЗы в проведении исследований и предоставлении информации потенциальным клиентам.

Во многих современных школах в фразу «обучение с помощью компьютера» вкладывается такой смысл: компьютер обучает ребенка. Можно сказать и так: ***компьютер используется, чтобы программировать ребенка****. А на мой взгляд,* ***ребенок программирует компьютер****, и, делая так, ребенок не только овладевает частичкой самой современной техники, но и приобщается к некоторым из самых глубоких идей естествознания, математики, а также к искусству интеллектуального моделирования.*

Сеймур Пейперт «Переворот в сознании»

1. ВВЕДЕНИЕ. ВЫЧИСЛИТЕЛЬНОЕ МЫШЛЕНИЕ

Вычислительная грамотность, программирование и вычислительное мышление являются одними из терминов, которые стали особенно популярными в школах, семьях и политических дискуссиях. Лидеры со всего мира высказались в поддержку обучения программированию с раннего возраста. Несмотря на активную работу, проводимую для введения обучения программированию в школах, многие учителя, родители и даже должностные лица не понимают, почему программирование считают новой грамотностью XXI века и действительно ли оно идет на пользу обучению детей, а не просто подготавливает их к будущей работе.

В этой главе мы рассмотрим некоторые основополагающие вопросы, затронутые в книге. Для начала мы уточняем, что изучение программирования не является самоцелью. Программирование скорее является способом освоения вычислительного мышления и вычислительной грамотности — популярных терминов, не всегда понятных неспециалистам. Их значение и важность мы объясним в первой части этой главы. По сути, мы живем в мире, в котором способности человека и компьютера переплетаются ускоренными темпами и вычислительное мышление становится необходимостью для всех. Программирование улучшает способность человека логически мыслить, отдаваться когнитивному полету фантазии и решать задачи. Можно сказать, что программирование помогает лучше думать. Во втором разделе этой главы мы описываем, как программирование связано с так называемыми некогнитивными или социально-эмоциональными навыками и взаимодействием для решения проблем, ставших императивом XXI века. Этот раздел выходит за рамки темы навыков и выносит на обсуждение вопросы о цели и мотивации к обучению. В последнем разделе главы рассматривается вопрос о том, почему программирование считают новой грамотностью, и есть ли разумное основание утверждать, что в скором времени каждый должен будет уметь писать код так же, как сейчас каждый должен уметь читать и писать.

1.1. Зачем преподавать детям программирование? Часть 1. Индивидуальный компонент

Из идеи о всеобщем изучении программирования следует вопрос — когда начинать изучение? Все сходятся на том, что начинать следует в детском возрасте, предпочтительно в раннем детстве, когда только начинают формироваться навыки мышления. В этом разделе главы мы выйдем за рамки понятия вычислительной грамотности и рассмотрим программирование в более широком контексте с точки зрения преимуществ познания и метапознания (мышления о мышлении). Чтобы понять, зачем это нужно, мы сначала должны рассмотреть важное понятие вычислительного мышления.

1.1.1. Определение вычислительного мышления

«Вычислительное мышление является фундаментальным навыком для всех, не только для ученых, работающих в области компьютерных наук. К аналитическим способностям каждого ребенка, наряду с чтением, письмом и арифметикой, мы должны добавить вычислительное мышление».

- Жаннет М. Уинг [1]

Мы считаем термины «кодирование» и «программирование» взаимозаменяемыми, однако, оба термина обозначают конкретные применения вычислительного мышления (ВМ), которое можно определить как использование комплекса навыков, которые помогают сформулировать задачу таким образом, чтобы компьютер помог нам решить ее. Вычислительное мышление состоит из четырех основных компонентов, как показано на рисунке 1.1 [2]

Рисунок 1.1. Четыре основных компонента вычислительного мышления

Программирование можно определить как «процесс разработки и выполнения различных наборов инструкций, позволяющих компьютеру выполнить определенное задание, решить задачу и обеспечить интерактивность человека» [3]. Таким образом, программирование можно считать реализацией ВМ или средством развития ВМ. Составные элементы ВМ, такие как декомпозиция и выделение паттернов, универсальны и могут применяться не только для программирования компьютера. В действительности они являются структурными элементами таких способностей, как креативность, решение задач и — что даже более важно, чем решение задач, — постановка задач.

1.1.2. Взаимосвязь между программированием и вычислительным мышлением

На практике в разных странах и внутри стран существуют разные определения таких терминов, как ВМ, программирование, компьютерные науки и цифровая грамотность, и взаимосвязи между ними. Однако существует единое мнение о том, что программирование является инструментом для развития ВМ, при этом ВМ включает более широкий спектр способностей, например, анализ задач и алгоритмическое мышление. В этом отношении программирование является составной частью ВМ, конкретизирует понятия ВМ и может стать инструментом получения знаний. Например, программированию предшествуют процессы анализа и декомпозиции задачи. Программирование — это только один из способов освоения ВМ, однако его можно считать самым эффективным [4].

Рисунок 1.2. Определение и взаимосвязь между кодированием и вычислительным мышлением

В этой книге мы используем краткий термин «кодирование» для обозначения «программирования для развития или углубления вычислительного мышления». Обобщение используется из соображений удобства. В контексте, раскрываемом в этой книге, термин «кодирование» используется чаще, чем «программирование». Программирование в основном используется в профессиональном контексте, например, код, написанный профессиональными программистами для различных применений: от авиационной техники до визуализации зигонемы.

1.1.3. Обучение «умному мышлению» независимо от того, считают ли вас «умным»

Программирование связано с вычислительным мышлением, и, обучая детей программированию, мы действительно хотим, чтобы они развивали свои способности к вычислительному мышлению. Именно эта способность позволит детям свободно владеть языком кода в мире, где программирование будет считаться грамотностью. Но не только страх считаться безграмотными мотивирует детей к изучению программирования. Более важным является вопрос развития детей, озвученный много десятилетий назад Сеймуром Пейпертом, который выдвинул идею конструкционизма. Пейперт считал, что дети не выступают пассивными получателями знаний, а наоборот, сами их активно конструируют. Пейперт продолжил развивать подход известного идеолога конструктивизма в педагогике Жана Пиаже — идею о том, что дети учатся быстрее всего, когда они сами формируют свои представления об увиденном и делают собственные заключения, а не когда им говорят, как они должны воспринимать мир.

Конструкционизм разделяет позицию конструктивизма в том, что обучение — это «построение структур знаний» независимо от условий обучения, однако добавляет идею о том, что обучение происходит особенно эффективно, когда ученик сознательно вовлечен в создание объекта, будь то песочный замок или теория вселенной... [5]

Конструкционизм объединяет два типа конструирования: дети конструируют вещи в реальном мире и конструируют новые идеи в своей голове. Эти два типа конструирования создают непрерывную спираль обучения, поскольку, когда у ребенка появляются новые идеи, он конструирует новые вещи в реальном мире.

Программирование помогает детям воплотить эти идеи в жизнь. Все началось с языка программирования Logo, разработанного Пейпертом и его коллегами в 1967 году. Logo стал первым языком программирования для детей, в котором дети использовали программирование главным образом для того, чтобы управлять движениями робота-черепашки. После Logo было разработано множество новых сред программирования, призванных помочь детям создавать вещи в реальном мире и в своей голове. Сегодня одной из самых известных детских сред разработки кода является Scratch — потомок Logo. Энтузиазм 12-летней участницы Scratch-сообщества иллюстрирует то, что Сеймур Пейперт называл конструкционизмом:

«Scratch позволяет мне самой придумывать идеи для проектов — и заниматься тем, что мне по-настоящему интересно. Многим нравится решать проблемы, особенно трудные, но вот отыскать ключик к задаче, которую сам себе поставил, — это гораздо интереснее и мотивирует куда сильнее».

- Цитата из (Resnick 2017, p. 58)

Программирование помогает детям учиться мыслить дисциплинированно, размышлять о таком мышлении и учиться вместе с другими. Аспект совместного обучения или социальный конструктивизм согласно учению Льва Выготского будет рассмотрен в следующем разделе. В этом разделе мы рассмотрим основы мышления. Некоторые дети склонны мыслить абстрактно, другие испытывают трудности с абстрактным мышлением, при этом абстрактное мышление необходимо, чтобы думать о более сложных вещах, учитывая биологические ограничения объема рабочей памяти. Строка кода (или позднее цветной блок или фигура, как в Scratch) на экране компьютера является физическим представлением или проявлением мысли «продвинуть на 10 шагов». Возможность увидеть мысль на экране помогает ею управлять, например, составить последовательность с другой мыслью — «повернуть направо». Дети начинают понимать, что алгоритмы можно увидеть везде: в том, как мы печем печенье или чистим зубы по утрам. Программирование обеспечивает легкий доступ к метапознанию или мышлению о мышлении. Это помогает ребенку понять, получится ли у Scratch-персонажа (именуемого спрайт) преодолеть препятствия и перейти из одной части лабиринта в другую «по клику левой стрелкой» в результате определенной последовательности событий.

Программирование использует для понимания мира основные мотивации, что делает его по-настоящему эффективным инструментом обучения. Мышление требует энергии и силы. Мы делаем что-то добровольно, только если занятие нам интересно и приносит радость. Благодаря мгновенной обратной связи, которую мы получаем при программировании, мы испытываем возбуждение, которое стимулирует обучение и помогает развитию субъектности и самодетерминации. Прежде чем мы перейдем к видам мышления, которые развивает программирование, следует отметить возможность изучения программирования вне теории конструкционизма. Хотя базовые подходы к освоению вычислительного мышления через программирование были сформированы в рамках теории конструкционизма, это не значит, что конструкционистский подход является единственной основой изучения программирования. Например, взрослых, впервые изучающих программирование, или молодых людей или взрослых, которые изучают программирование в профессиональных целях, не обязательно увлекать проектами с играми, анимацией и музыкой. Такие ученики могут самостоятельно разобраться с правилами синтаксиса и изучить такие языки, как Python или C++, в среде их рабочего применения.

Навыки постановки и решения задач

В своей увлекательной книге «Спираль обучения. 4 принципа развития детей и взрослых» Митчел Резник из Массачусетского технологического института описывает посещение школы Brightworks в районе Мишен Дистрикт в Сан-Франциско (Resnick 2017, pp. 51–54). Проектное обучение, практикуемое в Brightworks, резко контрастирует с традиционной организацией работы в классе. Резник приводит примеры того, как ученики в большинстве школ сталкиваются с потоком задач, которые их учат решать. В зависимости от возрастной группы ученикам преподают сложение дробей или изменение временной формы глагола. Они также могут изучать или запоминать факты, но не решать реальные задачи.

«Но когда учащиеся решают задачи в отрыве друг от друга и от реальной жизни, то и знания у них нередко получаются разрозненными, дети не понимают, для чего изучали те или иные темы или где они могут пригодиться» (Resnick 2017, p. 53).

Когда дети изучают программирование в рамках проекта, который они разработали сами, меняется весь сценарий их подхода к решению задач. В отличие от фрагментированного обучения в большинстве традиционных учреждений, описанного Резником, дети, пишущие код в рамках проектов, конструируют знания для себя, и это оказывает долговременное влияние на то, как они думают. Невозможно создать проект, не имея цели. Постановка собственной цели или общей цели команды важна для мотивации и энтузиазма. Программирование помогает ученику продолжать решать задачу, имея такую мотивацию. Когда ученик понимает, что у него нет навыка, необходимого для решения задачи, и самостоятельно приобретает этот навык, он учится учиться. Связи между элементами знаний образуются сами собой, и ученик знает, как вписать единицу знаний в общую картину. Учитель, конечно, содействует и направляет процесс познания ученика, помогает ученику научиться определять подходящие ему стратегии решения задач, но руководит процессом сам ученик.

Креативное мышление

«Творческая природа программирования не требует особых доказательств. Хочется высказать, быть может, более спорную мысль, что в своей творческой природе программирование идет намного дальше большинства других профессий, приближаясь к математике и писательскому делу. В большинстве других профессий мы лишь «приручаем» силы природы — просто «сажаем тигра в бензобак», — используя те или иные физические или биологические явления, не обязательно постигая их сущность. В программировании же мы в некотором смысле идем до конца. Один из тезисов современной теории познания гласит «мы знаем это, если можем это запрограммировать».

- А. П. Ершов [7]

Возможно, нам не удастся научить учеников креативности в форме установки «будь креативным», но мы можем создать условия и стимулы для экспериментов, развития и углубления их креативности. Пабло Пикассо говорил, что каждый ребенок — художник, трудность в том, чтобы остаться художником, выйдя из детского возраста. Программирование помогает проявиться художнику в каждом ребенке. Это обусловлено сильным сходством между программированием и игрой. Программирование вообще можно считать прототипом игры. Программист и дизайнер игр Ян Богост объяснял, что «сила игровых программ заключается не в получаемом вознаграждении или удовольствии, а в структурированном ограничении дизайна, которое открывает множество возможных пространств для игры» [8]. Программирование создает трудности или препятствия, которые программист должен преодолеть. Преодоление этих трудностей и есть та самая магия, о которой говорит ученый-мечтатель Андрей Ершов в приведенной выше цитате.

В разных странах молодые пользователи проводят много времени, потребляя информацию в сети Интернет, взаимодействуя с компьютерами и играя в игры. Большинство детей не проявляют интереса к созданию собственных артефактов, игр или художественных работ. Благодаря распространению таких инструментов программирования для детей, как Scratch, Alice или Kodu, и осознанию родителями и учителями идей, изложенных в этой книге, начала меняться «потребительская» модель взаимодействия с компьютером. При помощи этих инструментов дети создают разнообразные игры, анимации, истории и художественные работы [6].

Исследования показывают, что креативная вычислительная деятельность может быть особенно полезной для молодежи из группы риска или людей с низкой самооценкой, обусловленной такими причинами, как этническое происхождение, низкий социально-экономический статус или плохая успеваемость.

В рамках интересной экспериментальной оценки, проведенной в Израиле, исследователи проверили влияние двух разных типов видео на творческие способности начинающих Scratch-программистов в возрасте 14-17 лет [9]. Экспериментальной группе показали видеолекцию о пластичности мозга и податливости интеллекта. Контрольной группе показали нейтральное видео о работе мозга. Затем обе группы посмотрели обучающие видеоролики, посвященные Scratch, и создали свои Scratch-проекты, которые были оценены приглашенными экспертами. Scratch-проекты контрольной и экспериментальной группы оценивались по четырем параметрам, включая ясность идеи, качество программирования, креативность и оригинальность, а также эстетические характеристики дизайна. Исследование показало, что Scratch-проекты учащихся, входивших в состав экспериментальной группы, получили более высокие оценки по параметрам креативности и эстетических характеристик.

Осознание важности и полезности неудач

«Благодаря этому она набралась храбрости и больше не боится пробовать новые вещи. Даже если первая попытка оканчивается неудачей, дочка воспринимает ее вовсе не как «конец игры», а как напоминание о том, что стоит попробовать другой путь, что в нужное место можно идти разными маршрутами и что не бывает маршрутов «правильных» и «неправильных».

- Мнение матери об опыте работы ее дочери со Scratch, цитата из [6, p. 148]

В процессе обучения программированию дети учатся пересматривать свое мышление, когда программы не работают так, как задумано. Навыки программирования, навыки понимания новых идей и навыки обучения связаны друг с другом. По сравнению с другими видами учебной деятельности, программирование — более щадящий процесс, где дети могут делать ошибки и учиться на них. При написании кода ребенок может легко отменить свои действия и исправить ошибки, внести изменения и попробовать что-то новое. Ошибки в программировании обычно называют «багами», а не «неудачами».

Важным процессом в программировании является «отладка», то есть выявление проблемы и применение нескольких методов для ее решения [6]. Сеймур Пейперт и Шерри Теркл определили это понятие как конструирование обучения учащимися путем организации и реорганизации набора материалов, а не просто получения контента от учителя [10]. Обучение через «отладку» даже начали использовать в некоторых заданиях «Исправь баг» в Scratch. В этих заданиях предусмотрена ошибка, и пользователи должны найти решение [4].

Традиционные системы образования не приветствуют неудачи — плохо не ответить на вопросы учителя об атомном номере селена. После такой неудачи не предусмотрена возможность обучения или восстановления. Однако когда код импровизированной двери, ведущей к миске для кота, не срабатывает, и дверь не открывается, когда сенсор обнаруживает кота, вы пытаетесь понять, что пошло не так, — вы понимаете, что в коде, который вы написали, есть ошибка в калибровке сенсора — и, скорее всего, вы уже не совершите эту ошибку в следующем проекте. Это можно назвать «антихрупким обучением», когда с каждой неудачей ваше понимание и навыки становятся только лучше, и вы можете решать более сложные задачи [11].

Поиск информации и умение найти помощь и ресурсы важны для рынка труда и повседневной жизни в XXI веке. Процесс отладки, который помогает детям преодолеть путь от идеи до совместного использования проекта, дает возможность сформировать полезные привычки обучения в течение всей жизни. В будущем дети смогут применять навыки, приобретенные в процессе отладки, во всех сферах своей жизни [12].

Альтернативы программированию для развития когнитивных и некогнитивных навыков

В ходе недавнего мета-анализа были изучены более 100 экспериментальных исследований, посвященных влиянию обучения программированию [13] на креативность, логику и математические навыки, а также влияние на навыки программирования. Для сравнения воздействия различных вмешательств и исследования разных объемов выборки в литературе обычно используется такое понятие, как величина эффекта. Величина эффекта — это расхождение в результатах интервенционной или экспериментальной группы и контрольной группы, выраженное в единицах стандартного отклонения. Важно описывать расхождение в единицах стандартного отклонения, потому что большое расхождение в контексте большой базовой вариабельности может не иметь какой-либо экономической или политической значимости. Отрицательная величина эффекта может возникнуть, когда вмешательство приводит к результатам ниже, чем у контрольной группы. Нулевая величина эффекта не означает ни положительное, ни отрицательное воздействие. В учебной литературе величина эффекта +0,10 и более уже считается статистически значимой, величина эффекта 0,40 и более может рассматриваться как обладающая потенциалом для повышения успеваемости учащихся.[1]

Исследование Шерера и соавторов показало, что изучение программирования имеет стандартизованную величину эффекта +0,75 для навыков программирования, +0,73 для креативности, +0,57 для математических навыков и +0,37 для логики. Другое недавнее исследование качественно дополнило этот анализ в рамках изучения влияния, отмеченного в 10 недавних исследованиях [14]. В этом исследовании отмечалось, что изучение программирования сильнее всего повлияло на математические навыки, но положительные результаты также были отмечены в области социальных навыков, самоорганизации и критического мышления. Совершенно очевидно, что если бы развитие когнитивных и некогнитивных навыков было целью обучения вне зависимости от вычислительного мышления, за последние годы было бы разработано множество программ, которые бы предвосхитили появление компьютеров в классах. В рамках исследования по оценке воздействия международной признанной программы Escuela Nueva или «Новая школа», основанной на совместном и взаимном обучении, была отмечена величина эффекта менее 0,20 для когнитивных навыков и 0,40 для некогнитивных навыков [15]. В упомянутом выше сборнике Джона Хэтти указана величина эффекта 0,34 для «взаимного обучения» и величина эффекта 0,28 для «программ развития всей школы».

[1] https://www.visiblelearningplus.com/content/research-john-hattie

1.2. Зачем преподавать детям программирование? Часть 2. Совместная деятельность

В этом разделе мы представим аргументы в пользу программирования как способа научиться взаимодействовать и понять, как устроен наш мир, а также как использовать мотивацию для обучения.

«До тех пор, пока программирование будут считать просто формой мышления и задействования разума человека, а не формой участия и выражения, мы не сможем воспользоваться преимуществом изучения и преподавания кода в современных сетевых сообществах».
- Ясмин Б. Кафай и Куинн Берк [4]

Улучшение навыков вычислительного мышления у детей является одним из ключевых результатов программирования, однако сужение потенциала программирования только до развития навыков решения задач и других навыков не позволит нам увидеть полную картину. На самом деле, программирование позволяет учащимся раскрыть свой творческий потенциал, выразить себя и поучаствовать в жизни сообщества, внося свой вклад и извлекая свои уроки, — это своего рода становление. До сегодняшнего дня системы образования, порожденные промышленной революцией, несут бремя своего происхождения, — ведь всеобщее образование обеспечило необходимое количество послушных, дисциплинированных и пунктуальных работников. Возможно, именно поэтому большинство человеческих обществ недооценивают игру как инструмент обучения.

В своей книге «Связанный код: почему дети должны учиться программировать» Ясмин Кафай и Куинн Берк рассказывают о четырех основных тенденциях в сфере креативного программирования для детей. Эти четыре тенденции, кратко описанные ниже, составляют дорожную карту потенциальной местности для обучения программированию. Важным аспектом всех этих тенденций является общность двух аспектов: первый аспект — это идея самовыражения или личного самосознания с помощью программ, написанных детьми, например, запись своего голоса или музыки, проигрываемой в Scratch-истории; второй аспект — это высокий уровень внутренней мотивации, который часто отмечают у детей, занимающихся программными проектами. В этом случае взрослым не приходится беспокоиться о том, как заставить детей сделать что-то полезное, дети просто развлекаются, и им не нужна внешняя мотивация.

Программирование — не единственный современный способ для сегодняшних подключенных детей получить доступ к творческим возможностям совместной работы и подготовиться к жизни в постиндустриальном мире. Количество поклонников таких видеоигр, как Fortnite и Minecraft, составляет сотни миллионов молодых людей по всему миру. В своей книге «Новое детство: воспитание детей для жизни в подключенном мире» Джордан Шапиро рассказывает, каким образом даже игра в видеоигры (в отличие от программирования или создания новых игр) может быть полезна детям [16]. Многим детям понравится создавать собственные видеоигры, однако, по мнению Шапиро, даже играя в Mortal Kombat и Frogger, дети развивают интеллектуальные способности, которые будут им необходимы для жизни в подключенном мире. Он приводит описание «процедурной риторики» Яна Богоста [8] — идеи о том, что ограничения или правила видеоигр генерируют способности «создавать аргументы посредством процессов».[2]

[2] В этой книге мы обсуждаем программирование, а не чрезвычайно популярные видеоигры. Для объединения этих двух тем было разработано много идей относительно игры Minecraft [17].

1.2.1. От кода к приложению

«Вычислительное участие — это решение задач с другими, разработка интуитивно понятных систем с другими и для других, а также изучение культурной и социальной природы поведения человека с применением понятий, практик и подходов компьютерных наук».

(Кафай и Берк, 2014)

Кафай и Берк ввели термин «вычислительное участие» для обозначения вовлечения учеников в деятельность, выходящую за рамки вычислительного мышления. Работа детей в группах и создание ремиксов кода способствует вычислительному участию. Авторы рассказывают об увлекательном научном исследовании, которое было проведено тридцать лет назад и направлено на то, чтобы побудить учеников четвертого класса разработать инструктивное программное обеспечение для того, чтобы рассказать о дробях ученикам третьего класса. Исследователи посчитали творческий аспект этого задания наиболее важным. Реалистичность опыта работы в классе играет важную роль, это означает, что обучение подразумевает решение реальных задач. Поэтому важно, чтобы задания по написанию кода были направлены не просто на решение головоломки, а чтобы у них была цель или «применение». Это не игра в обучающие игры, а создание собственных обучающих игр, которые, благодаря появлению широкой сетевой аудитории, оценивает не только учитель.

1.2.2. От инструментов к сообществам

Превратив программирование в работу сообщества, преподаватели могут трансформировать школы в среду для совместной работы. Когда дети пишут код для игр, историй или художественных работ, эти объекты не стоит считать просто способом потренировать интеллектуальные способности. Это объекты, которыми можно делиться, которые связывают детей друг с другом. Онлайн-сообщества, созданные на основе таких инструментов программирования, как Scratch или Kodu, можно создать и в школе, и в клубе, где молодежь сможет делиться своими разработками и учиться друг у друга.

Переход от инструментов к сообществам — ключевое изменение, которое влияет на роль учителя. Когда изучение программирования означало изучение языка как инструмента, учителя, естественно, сначала изучали и осваивали инструменты сами, после чего они могли начать обучение. Здесь происходил серьезный отсев тех, кто мог бы преподавать программирование: как правило, это были только преподаватели ИКТ или компьютерных наук, которых было мало, и расширить их штат было большой проблемой. Зависимость от учителей-специалистов мешала должным образом развивать и популяризировать программирование. С переходом к сообществам и возникшим изобилием учебных ресурсов ученику и учителю достаточно просто подключиться к сети или сообществу. Учитель уже не должен быть самым умным в классе и не должен иметь ответы на все вопросы.

1.2.3. От чистого листа к ремиксу

Кафай и Берк считают, что изучение программирования путем освоения азов синтаксиса и грамматики, начав с базового минимума, — занятие довольно скучное. Это все равно, что учиться писать прозу на уроках по изучению языка, когда сначала изучают существительные, глаголы и обстоятельства. Открывая Scratch-игру или анимацию, которую скретчер выложил на платформе, вы видите очевидное приглашение — L⊙⊙k Inside! Модель сформирована следующим образом — Используй -> Меняй -> Создавай: учащийся сначала использует игру, анимацию или приложение, чтобы понять их функционал и назначение, затем экспериментирует с открытым исходным кодом, чтобы скорректировать или изменить поведение. Этот второй этап имеет принципиальное значение, так как он помогает быстро понять, каким образом алго-

ритм или набор процедур работает на практике. В итоге учащийся может создать собственный проект в виде ремикса других проектов. При этом ученик может не создавать новых элементов программы или добавлять в программу многочисленные изменения.

Ремикс как метод прекрасно вписывается в современную профессиональную среду программирования, построенную на источниках с открытым кодом и нормах поведения, поощряющих совместное использование. Репутацию профессиональных программистов часто создает код, который они выкладывают в виде пакетов с открытым исходным кодом на открытых платформах. На одном из самых популярных сайтов для программистов StackExchange.com применяется сложный подход, когда качество и количество ваших совместных вкладов оценивают те, кто обратился за помощью. Практика создания ремиксов является инновационной для традиционной модели, потому что в доцифровом мире копирование считалось синонимом мошенничества. Ученики не должны были шуметь и разговаривать друг с другом в классе.

1.2.4. От экранов к осязаемым объектам

Четвертая тенденция, определенная Кафай и Берком, касается растущей популярности творческой деятельности, в которой программирование связано с созданием материальных продуктов. В этом смысле идея низких полов, широких стен и высоких потолков реализуется в полном объеме. С такими платформами, как Makeymakey, рассчитанными на самых маленьких разработчиков, очень просто начать работать. Робототехника всего лишь одно из применений. Широкие стены символизируют различные типы проектов, которые становятся все более популярными. Кафай и Берк обращают внимание на электронные ткани и креативные проекты с использованием умных тканей и носимых технологий. Lilypad Arduino — один из простых в использовании комплектов размером с бейдж, который можно сшивать и запрограммировать. Совмещение шитья с программирования помогает генерировать конструктивные идеи о гендерных ролях и вычислительном мышлении.

1.3. Является ли программирование новой грамотностью?

«Мы начинаем учить детей читать и писать в раннем детстве. При этом мы не считаем, что каждый ребенок станет профессиональным писателем. Я считаю, что текстовая грамотность является важным навыком и интеллектуальным инструментом для всех. Так же и с программированием. Я не призываю к тому, чтобы все дети стали разработчиками ПО и программистами, я хочу, чтобы они овладели вычислительной грамотностью и смогли стать не только потребителями, но и создателями цифровых артефактов».

- Аннет Ви [12]

В исследовании по изучению грамотности, подробно описанном Аннет Ви, рассматриваются исторические предпосылки обязательности навыков чтения и письма для каждого человека. Далее Аннет Ви описывает, как можно определить аналогичную тенденцию в отношении программирования, — идею о том, что в XXI веке каждый должен будет уметь писать код[3]. Ограниченность нашей публикации исключает возможность подробно описать новые исследования в сфере грамотности и другие работы Ви, однако мы смогли включить в эту книгу некоторые основные идеи.

[3] Этот раздел основан главным образом на идеях, представленных в книге Аннет Ви «Кодовая грамотность: как компьютерное программирование меняет письменность» (*Coding Literacy: How Computer Programming is Changing Writing, Vee 2017*). В своей книге Ви подробно описывает, как можно использовать теоретические инструменты исследований грамотности для понимания программирования в его историческом, социальном и концептуальном контексте. Краткая версия ее продуманных аргументов и эмпирических данных доступна по ссылке http://www.annettevee.com/blog/2013/12/11/is-coding-the-new-literacy-everyone-should-learn-moving-beyond-yes-or-no/

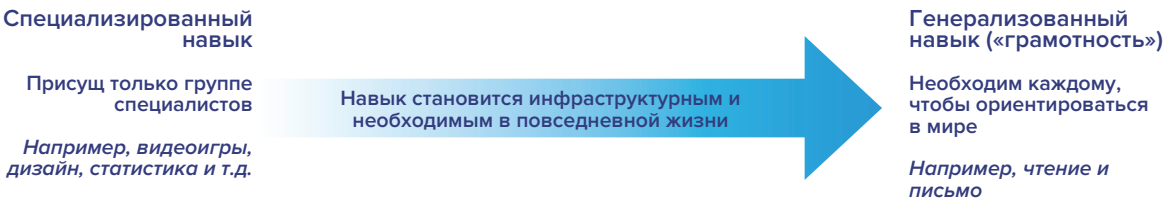
1.3.1. Что такое «грамотность»?

Аннет Ви определяет грамотность как «умение человека, необходимое ему в повседневной жизни для творческих, коммуникативных и риторических целей». В настоящее время людям необходимы навыки чтения и письма, чтобы понимать окружающий мир и ориентироваться в нем. Это основной функционал коммуникационной практики практически любого общества, если только общество не изолировано от остального мира. Хотя грамотность можно использовать в разговорной речи для определения различных наборов навыков, например «дизайнерская грамотность», «арифметическая грамотность» или «грамотность в сфере видеоигр», на самом деле грамотность — многофункциональный термин, обозначающий необходимость или востребованность навыка в повседневной жизни. Пока навык воспринимается как исключительно специализированное умение, которым пользуется отдельная группа людей, он не может считаться грамотностью. Поэтому «грамотность» означает генерализованный навык «чтения и письма», а не специализированный навык, доступный лишь некоторым.

1.3.2. Грамотность как генерализованный навык

Та степень, в которой общество нуждается в навыках чтения и письма для взаимодействия со своими членами, определяет, является ли грамотность генерализованным навыком. Например, в некоторых изолированных обществах охотников-собирателей способность интерпретировать тексты еще не свидетельствует о грамотности, потому что это не является необходимостью для представителей этого общества. Следовательно, отнесение какого-либо навыка к понятию грамотности зависит от общественного контекста[4].

Рисунок 1.3. Специализированный и генерализованный навык



Понимание истории грамотности может помочь нам осознать, как другие наборы навыков, такие как программирование ЭВМ, могут функционировать в обществе, и можно ли их считать новой грамотностью в настоящее время или в будущем. По мнению Ви, текстовая грамотность возникла в рамках инициатив центральных органов управления, которые распространились на другие крупные учреждения и коммерческие структуры и, наконец, были перенесены в повседневную жизнь граждан. Она выделяет два этапа в истории грамотности:

Этап 1. Текст становится частью инфраструктуры

Тексты заняли центральное место в жизни людей, поскольку они способствовали становлению развивающихся институтов (государственного аппарата, договорного права и издательского дела) и обеспечивали возможность роста населения и увеличения объемов информации. Первое крупномасштабное документально подтвержденное принятие текста произошло в Англии в XI веке в силу потребности нормандских завоевателей, которые стали новыми правителями

[4] Ви представляет идеи, основанные на обширной и разнообразной литературе, посвященной вопросам грамотности. Подробная информация об оригинальных источниках исследований грамотности и соответствующей литературе в этой области представлена в ее книге.

Англии, в информационном обеспечении. По приказу короля Вильгельма I была проведена всеобщая перепись, известная как Книга страшного суда. Книга страшного суда — древнейший публичный акт и учредительный документ национальных архивов Англии. После этой переписи, инициированной центральным органом управления, английские провинции начали внедрять письменность в XIII веке. Текст использовался как инструмент создания законов и разработки стратегий. Особенно это касается земельных законов, которые стали оформлять в виде письменных текстов вместо персональных свидетельств, для чего люди должны были вписать свои имена.

Этап 2. Текст становится неотъемлемой частью повседневной жизни

Начиная с XIX века, благодаря массовым кампаниям по ликвидации неграмотности, большинство людей освоили навыки письма и чтения. Грамотность была необходима, чтобы получить доступ к информации, представленной в газетах, которые описывали местные и мировые события, в альманахах, которые публиковали рекомендации для фермеров, и счетах для отслеживания задолженностей. Такие концепции, как почтовое сообщение, письменные налоговые счета, публичные вывески и всеобщее образование, были основаны на предположении о том, что большинство граждан умеют читать и писать. Таким образом, чем больше людей осваивали навыки чтения и письма, тем более массовой должна была становиться грамотность. Эта потребность сегодня настолько актуальна, что наша повседневная жизнь немыслима без текстовой грамотности.

1.3.3. Вычислительная грамотность

«Программирование — современная грамотность, которая помогает совершенствовать такие навыки XXI века, как решение проблем, работа в команде и аналитическое мышление».

- Европейская комиссия[5]

«Программирование — новая грамотность. Чтобы добиться успеха в обществе будущего, молодые люди должны научиться проектировать, создавать и выражать себя при помощи цифровых технологий».

- Митчел Резник [6]

«Грамотность является навыком инфраструктурным и абсолютно необходимым для образования, для формирования грамотных и компетентных людей. Инфраструктурность значит, что грамотность является не просто результатом образовательного процесса, а его движущей силой. Учебник разработан для каждого предмета, а не только для изучения английского языка или других предметов, связанных с освоением грамотности. Если вы плохо читаете или не обладаете базовыми математическими знаниями, учебники по истории, естествознанию или математике будут бесполезны для вас. Компьютеры могут стать технической основой новой и гораздо лучшей грамотности, которая во многом будет схожа с существующей грамотностью, степень проникновения и глубина влияния которой будет сравнима с тем, что мы уже испытали в процессе достижения всеобщей текстовой грамотности».

- Андреа ДиСесса [18]

Сегодня люди во всем мире могут мгновенно связаться друг с другом независимо от расстояния. Компьютерный код — это инфраструктура, обеспечивающая современную коммуникацию посредством обработки текстов, электронной почты, социальных сетей и многих других средств. Наряду с обеспечением коммуникации компьютерный код стал частью всех аспектов нашей жизни — от управления состоянием здоровья и поиска работы до возможности получения нового гражданства. Например, Эстония стала первой страной в мире, разработавшей

[5] https://ec.europa.eu/digital-single-market/en/coding-21st-century-skill

программу электронного резидентства (e-residency), которая позволяет людям в любой стране мира подать заявку на получение цифрового эстонского удостоверения личности государственного образца. Таким образом, Эстония создала «цифровую нацию из граждан мира».[6]

Поскольку компьютерное программирование приобретает все большее значение для повседневной жизни, и многим людям становится проще его изучить и использовать, мы надеемся, что свободное владение языком компьютеров и взаимодействие с ними получит широкое распространение. Этот аргумент подводит нас к термину «вычислительная грамотность», определенному Аннет Ви как «сочетание способностей разбить сложный процесс на небольшие операции и затем выразить — или «записать» — эти процедуры при помощи кода, который может быть «прочитан» неодушевленной единицей, такой как компьютер».

История вычислительной грамотности

Историческая динамика освоения компьютеров в повседневной жизни напоминает историю развития текстовой грамотности. Для вычислительной грамотности мы можем отметить последовательность этапов, аналогичную описанной выше последовательности для текстовой грамотности. Одно существенное отличие состоит в том, что процесс, который в одном случае растянулся на века, теперь происходит в течение десятилетий или даже меньших периодов.

Этап 1. Компьютеры становятся частью инфраструктуры

Первый этап истории вычислительной грамотности повторяет развитие текстовой грамотности: в обоих случаях были приняты меры на государственном уровне для регистрации данных переписи. Пример, используемый Ви, касается переписи населения США. Ввиду необходимости зафиксировать данные переписи 1890 года Бюро переписи населения США начало использовать аналоговую ЭВМ под названием «табулятор Холлерита».

В течение следующих нескольких десятилетий компьютеры освоили центральные органы управления, затем промышленные отрасли, учебные и научно-исследовательские учреждения. Университеты, авиакомпании и банковский сектор начали использовать компьютеры в 1950-х годах и продолжили в 1970-х годах.

Примерно в 1980-х годах компьютеры стали доступными для большинства людей, и знания о компьютерах начали проникать из областей знаний профильных специалистов в жизнь простых людей. Люди стали все чаще взаимодействовать с компьютерами в рамках систем здравоохранения, местных правительственных систем управления данными и систем образования, и это лишь некоторые примеры.

Мало кто знает, что в 1980-х годах в Советском Союзе было начато массовое движение за распространение компьютерной грамотности под руководством члена Академии наук Андрея Ершова [19]. В 1984 году Министерство образования Советского Союза предписало ввести курс вычислительных наук во всех средних школах страны. Реализация этой программы провалилась из-за нехватки компьютеров в большинстве школ в то время, хотя были предприняты попытки обучения алгоритмическому мышлению даже в отсутствие компьютеров. Мы вернемся к этой теме в следующей главе, когда будем обсуждать обучающие среды для воспитанников детских садов и учеников младших классов, предназначенные для изучения вычислительных концепций без использования компьютеров. Интересно, что даже в условиях централизованного аппарата советской системы образования была сделана попытка реализовать потенциал внешкольных компьютерных клубов и других внеклассных возможностей для детей и взрослых. История обучения детей программированию в России более подробно описана в главе 4.

[6] https://e-resident.gov.ee/

Этап 2. Компьютеры становятся неотъемлемой частью повседневной жизни

После 1990-х годов, особенно с появлением общего доступа к Интернету по всему миру, способность писать код стала выходить за рамки исключительной компетенции профессиональных программистов. Вначале, чтобы использовать электронные таблицы или текстовые редакторы, необходимо было лишь элементарное понимание вычислительных концепций. Тем не менее, широко применяемые приложения, такие как электронные таблицы, можно использовать с большей эффективностью и результативностью, если пользователь развивает вычислительные навыки.

Современным примером является использование макросов Microsoft-Excel для выполнения повторяющихся действий в нескольких электронных таблицах или файлах. Еще одним примером может служить использование сводных таблиц для проведения анализа. Непросто перейти от метода проб и ошибок к подходу с использованием программирования, так как необходимо изучить компьютерный язык Visual Basic, поэтому большинство пользователей MS-Excel ограничиваются использованием без программирования. Исследователи работали над решением этой проблемы, в результате чего был разработан Sprego[7].

Аннет Ви отмечает работу журналистов в качестве примера передачи программирования от экспертов к новичкам. Многие журналисты пока имеют возможность игнорировать возможности программирования, однако, в силу того, что электронные СМИ получают все большее распространение, сами журналисты должны понимать, как объединять визуальные и звуковые программные элементы с текстом для создания интерактивной графики и наглядного представления информации. Журналистам, юристам, врачам, учителям и другим специалистам будет проще выполнять свою работу, если они научатся писать код, а не будут ждать и зависеть от профессиональных программистов.

1.3.4. Будущее вычислительной грамотности

«Большинство людей не станут профессиональными журналистами или писателями, однако важно, чтобы все научились писать. Так же и с программированием — по тем же причинам. Большинство людей не станут профессиональными программистами или учеными в области компьютерных наук, но научиться бегло кодировать полезно каждому. Свободное владение навыком письма или программирования помогает развивать мышление, развивать мнение, развивать личность».

- Митчел Резник [6]

«Артур Кларк сказал: «В будущем всякий человек, полностью несведущий в естественных науках, окажется, честно говоря, необразованным. И если он будет, как делают это сейчас некоторые, кичиться своей неосведомленностью, он окажется точно в таком положении, как неграмотные средневековые бароны, гордо заявлявшие, что счетом и письмом у них занимаются секретари». Средневековые бароны и их потомки ушли в небытие, считать и писать научился каждый, а у секретарей появились новые хозяева и новые обязанности.
То же должно произойти и с программированием: руководители, не имеющие представления об ЭВМ и программировании, уйдут в небытие, профессиональные программисты станут системными аналитиками и системными программистами, а программировать сумеет каждый, что я и называю второй грамотностью».

- Андрей Ершов [7]

Вычислительные возможности стали более мощными и развитыми, при этом стало проще изучать компьютерные языки. Синтаксис компьютерного кода стал напоминать человеческий язык (особенно английский) благодаря замене цифр словами, автоматическому управлению

[7] http://sprego.hu/; http://www.ppig.org/library/paper/sprego-end-user-programming-spreadsheets

памятью, возможности добавлять комментарии и примечания для читателей кода и распространению ориентированных на пользователя сред программирования. С развитием языков визуального программирования, в которых используются формы и цвета, а не текстовый код, можно надеяться, что в будущем программирование станет еще доступнее для изучения и освоения.

Аннет Ви утверждает, что мы уже можем использовать термин «вычислительная безграмотность» аналогично термину «безграмотность» для обозначения некоего недостатка, который влечет за собой серьезные последствия для благосостояния и гражданских прав человека, не владеющего грамотностью. Поскольку сегодня безграмотность является барьером, мешающим неграмотному человеку ориентироваться в мире, можно ожидать, что в ближайшем будущем люди, не владеющие вычислительной грамотностью, будут зависеть от других в плане управления своей профессиональной, общественной и личной жизнью.

Дуглас Рашкофф заявил: «Все просто: программируй или будь программируемым» [20]. Это выражение может показаться опрометчивым, но в действительности Рашкофф представляет продуманное видение мира вокруг нас. Рашкофф отмечает, что цифровые технологии — это не просто инструменты, как ручка или отбойный молоток. Дело в том, что цифровые технологии запрограммированы, то есть инструкции по их использованию являются изолированными, что делает их мощными и позволяет тем, кто ими управляет, быть сверхмощными. Рашкофф объясняет, часто красочным и провокационным языком, что потенциал предыдущих технологий не был реализован из-за ограниченного контроля над их использованием — *«после появления текста люди могли собираться на городской площади и слушать, как раввин читает им слово Божье. Но только раввин мог прочесть свиток... После появления печатного станка многие научились читать, но только избранные, имеющие доступ к станку, могли писать... Люди на шаг отставали от избранных».* Для решения этой проблемы в будущем Рашкофф предлагает всем изучать программирование для жизни в условиях формирующейся гиперподключенной экономики, основанной на участии.

Выводы главы 1. Введение. Вычислительное мышление

- **Значение и значимость вычислительного мышления**. Кодирование или программирование — это способы приобрести умение, навык и способность к вычислительному мышлению, которое становится полезным и в конечном итоге необходимым для всех профессий. Четыре основных столпа вычислительного мышления: декомпозиция, выделение паттернов, абстрагирование и автоматизация. Программирование помогает детям научиться мыслить дисциплинированно, размышлять о своем мышлении и учиться вместе с другими.

- **Совместность — важнейший элемент обучения XXI века.** Вычислительное мышление предшествует вычислительному участию, а изучение программирования дает возможность перейти к вычислительному участию, которое выходит за рамки решения отдельных задач и креативности и имеет своей целью совместное решение реальных задач и обучение самовыражению. Программирование полезно не только для одного человека, оно помогает людям учиться жить и работать в тесном контакте.

- **Вычислительная грамотность.** Историческое исследование грамотности, того, как чтение и письмо были вначале прерогативой избранных, а затем стали необходимостью для всех, позволяет получить полезные сведения о развитии программирования как грамотности. Безграмотность традиционно считают серьезным недостатком, который необходимо устранить, поскольку неграмотному человеку трудно управлять повседневной жизнью и осуществлять свои права и обязанности. Исследователи и мыслители, такие как Аннет Ви, представляют доказательства относительно того, почему «вычислительная безграмотность» может стать таким же недостатком в так называемом информационном веке.

2. УСПЕШНЫЙ ПОЛЕТ: ПЯТЬ ВАРИАНТОВ РЕАЛИЗАЦИИ

Символ или образ двуглавой птицы очень уместен, так как он помогает быстро и легко понять основные варианты, доступные должностным лицам и специалистам-практикам, которые стремятся внедрить политику всеобщего обучения программированию.[8] Две головы символизируют два основных сосуществующих подхода к программированию: одна голова символизирует программирование в рамках учебной программы, другая голова — программирование в рамках внеклассного образования. Крылья отвечают за двигательную активность, полет вперед и вверх: одно крыло символизирует учителей, второе — другие аспекты среды обучения, включая платформы или продукты, подключение к Интернету и т.д. Хвост, который отвечает за направленное движение при полете, представляет собой аспект оценивания, принципиально важный для понимания иного значения оценивания в контексте программирования. Образ двуглавой птицы является основой этой главы — он помогает ориентироваться и упрощает варианты и задачи, которые стоят перед должностными лицами и специалистами-практиками.

Система образования каждой страны, а также региона и района внутри страны имеет свои особенности, которые отличают ее от других систем образования. Даже школы одного района, как правило, отличаются друг от друга, и учителя также привносят разнообразие в силу своего профессионального опыта и карьерного пути. Системы образования представляют собой прототипы сложных адаптивных систем с чрезвычайно высоким уровнем многообразия [22]. Это многообразие влияет также и на внедрение программирования, поскольку из-за чрезвычайной разрозненности опыта ни один пример нельзя считать «передовой практикой». Как писал выдающийся специалист по управлению и основатель онлайн-программы HBX Гарвардской школы бизнеса Бхарат Ананд в своей книге «Ловушка контента: руководство стратега по цифровым изменениям», выбор предполагает компромиссы, бессмысленно пытаться определить общую передовую практику для конкретной цели [23]. Тем не менее, стоит подробнее рассмотреть выбор различных систем образования в отношении пяти характеристик двуглавой птицы.

2.1. Первая голова птицы: программирование в рамках учебной программы

Мы начнем с рассмотрения такого варианта, как внедрение программирования в учебную программу, хотя он не является приоритетным по отношению к изучению программирования в рамках внеклассного образования, так же, как и одна голова орла не приоритетнее другой.

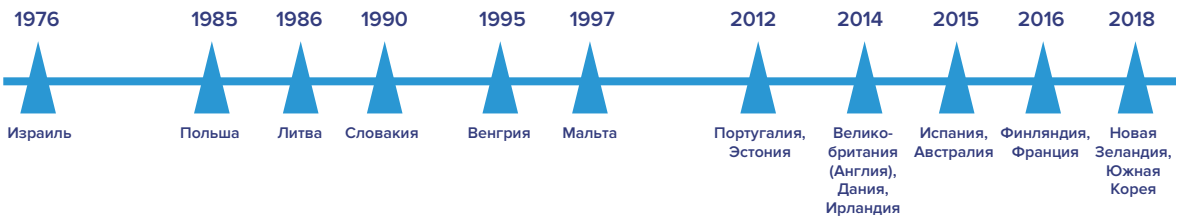
2.1.1. Страны, включившие программирование в школьную программу

С появлением персонального компьютера в 1980-х годах многие страны начали предпринимать шаги по включению обучения программированию в школьную программу. Учитывая растущую потребность в навыках вычислительного мышления в экономике и общественной жизни и ощутимый дефицит квалифицированных кадров в сфере информационных и коммуникационных технологий, руководства многих стран начали разрабатывать стратегии по внедрению

[8] Образ двуглавого орла в качестве метафоры обучения программированию символизирует современность и преемственность легендарной традиции, которая должна быть хорошо известна русскому читателю. Однако образ орла, упомянутый здесь, не впервые помогает понять и запомнить что-то важное в контексте образования. Бывший министр образования Нидерландов и бывший вице-президент Всемирного банка сделали популярным образ орла в качестве метафоры [21].

программирования в систему образования. Многие страны находятся в процессе включения программирования в учебную программу в основном на уровне среднего образования. При этом программирование и вычислительное мышление также стали частью программы начального образования [3]. В этом разделе мы кратко рассмотрим опыт разных стран.

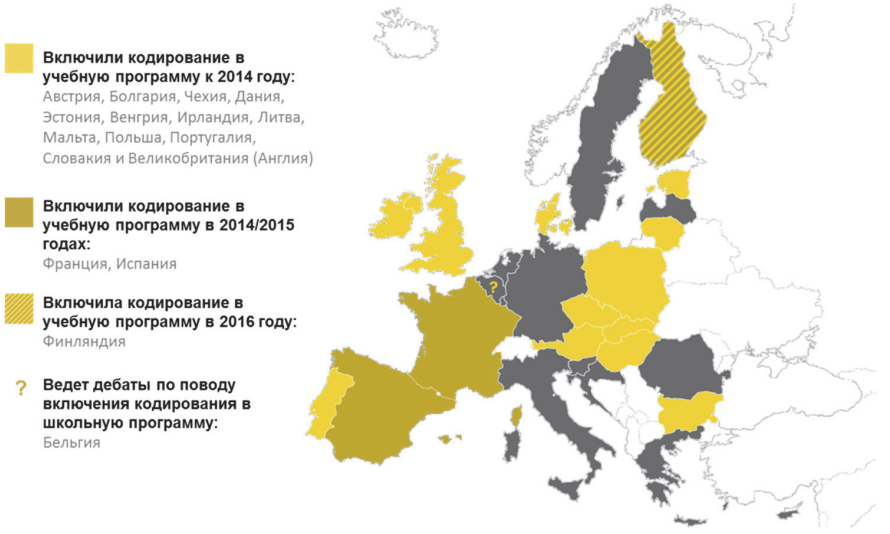
Рисунок 2.1. Хроника включения программирования в школьные программы разных стран



Израиль. Израиль одним из первых включил компьютерные науки в учебную программу, обновив несколько лет назад учебную программу для старших классов [24]. Курс компьютерных наук в старших классах направлен не на подготовку будущих программистов, а на ознакомление учащихся с логическим и алгоритмическим мышлением и погружение их в различные среды разработки на начальном этапе. Израиль предлагает ряд обязательных и факультативных курсов, позволяющих изучить основы компьютерных наук. Для тех, кто проявляет к этим наукам повышенный интерес, предусмотрено больше времени и материала [25].

В 2016 году Европейская комиссия в рамках своей программы развития новых навыков New Skills Agenda призвала все европейские государства-члены «больше инвестировать в формирование навыков в цифровой среде, включая программирование / компьютерные науки, в рамках всего спектра форм образования и профессиональной подготовки» [26]. По состоянию на 2016 год 16 европейских стран включили программирование в учебные программы на нацио-

Рисунок 2.2. 16 европейских стран, которые к 2016 году включили программирование в учебную программу



Заимствовано с сайта ErActiv, на котором кратко изложены результаты исследования Европейской школьной сети Schoolnet

нальном, региональном или местном уровне. 8 из них уже внедрили или разработали планы по внедрению программирования в учебную программу начальной школы, а 12 европейских стран уже внедрили или разработали планы по внедрению программирования в учебную программу средней школы [3] Опыт отдельных европейских стран изложен ниже на основе работы [25].

Польша. Польша долгие годы (начиная с 1985 года) внедряла программирование в учебную программу. Информатику как отдельную дисциплину преподавали в школах, начиная с первого класса, а недавно Польша разработала новую учебную программу, в рамках которой ученики могут применять навыки вычислительного мышления при изучении других школьных предметов.

Литва. Предмет «Информационные технологии» в Литве знакомит учеников не только с цифровыми технологиями, но и с этикой и правовыми принципами программирования. Предмет преподают как в младших, так и в старших классах средней школы. В старших классах средней школы ученики также проходят углубленный курс изучения информационных технологий.

Словакия. Словакия стала одним из первопроходцев в Европе: программирование — неотъемлемая часть учебной программы от начальной школы до старших классов средней школы.

Венгрия. Венгрия долгое время внедряла программирование в школьную программу. Однако в последние годы Венгрия разрабатывает стратегии включения программирования в программы всех уровней школьного образования, начиная с начальной школы.

Мальта. Мальта включила программирование в учебную программу старших классов средней школы. Однако стратегия Учебной программы Департамента электронного обучения явно поддерживает идею включения навыков вычислительного мышления в обучение, начиная с детского сада и до 11 класса.

Португалия. Вычислительное мышление является одной из задач обучения учеников 7 и 8 классов. В 2015-2016 гг. Министерство образования запустило пилотный проект для начальных школ под названием «Введение в программирование», в котором приняли участие 27 000 учеников третьих и четвертых классов и порядка 670 преподавателей. Проект охватывал две основные темы: вычислительное мышление и языки программирования. Инициатива была продлена на 2016-2017 учебный год. Ожидается, что в ней примут участие порядка 56 000 учеников и 1 600 преподавателей.

Эстония. Эстония является одним из мировых лидеров в области технологий, на долю наукоемких отраслей промышленности приходится 15% ВВП страны [27]. Однако Эстония испытывает нехватку программистов для этой процветающей отрасли. С 2012 года в эстонских школах начали преподавать программирование ученикам в возрасте от шести лет [28]. Эстония является одной из двух европейских стран, которые включили программирование в систему образования на всех уровнях, — от начальной школы до старших классов средней школы.

Великобритания (Англия). Королевское общество, одна из старейших научных организаций в мире, провело исследование, опубликованное в 2012 году, чтобы понять причину снижения интереса к ИКТ в школах, когда ученики стали жаловаться на скучные и повторяющиеся уроки. В 2012 году была начата крупная реформа, в рамках которой старое содержание уроков по ИКТ было заменено на новое, включающее три элемента: цифровую грамотность, информационные технологии и компьютерные науки. Обучение программированию с использованием Scratch было одним из основных элементов реформы, которая также включала тщательный анализ стратегии. В 2017 году был опубликован обзор результатов деятельности за пять лет с указанием недостаточной подготовленности и готовности учителей [29].

Дания. Считается, что учебная программа Дании разработана с учетом потребностей технологических компаний страны. Большое внимание уделяется таким темам, как дизайн пользовательского интерфейса, что интересно крупным компаниям, и влияние цифровых технологий на общество [24].

Финляндия. Финляндия — первая страна, применившая междисциплинарный подход к внедрению программирования, в рамках которого все учебные предметы должны иметь целью укрепление потенциала учеников в сфере ИКТ, начиная с 1 класса. Кроме того, программирование преподают в рамках двух предметов: математика с 1 класса и уроки труда с 3 класса — такой подход стал частью новой межкомпетентностной учебной программы, внедренной в Финляндии с 2016 года.

Новая Зеландия. Министр образования Новой Зеландии объявила, что в 2018 году цифровые технологии будут полностью включены в учебную программу для 1-13 классов [25]. Министерство образования поставило себе цель добиться всеобщей «технологической грамотности» к 10 классу с тем, чтобы все ученики могли использовать и создавать цифровые технологии для решения задач и использования доступных возможностей. В 13 классе ученики будут выбирать специализацию. Правительство объявило о намерении предоставить ресурсы для подготовки учителей к работе в системе образования, ориентированной на цифровые технологии [30,31].

Австралия. В 2015 году в Австралии была принята новая учебная программа под названием «Цифровые технологии», в рамках которой программирование стало обязательным с первого года обучения в школе до 10 класса. Эта новая учебная программа ориентирована на креативное использование технологий с применением трех элементов обучения: цифровых систем, данных и информации и создания цифровых решений [32].

Южная Корея. В 2018 году Южная Корея объявила курс изучения программного обеспечения обязательным для начальной и средней школы. Цель состоит в том, чтобы развить у детей младшего школьного возраста навыки вычислительного мышления с использованием языка визуального программирования, например, Scratch. Упор в учебной программе делается на способности решать реальные задачи: «Учащиеся должны применять навыки вычислительного мышления в реальных ситуациях, взаимодействуя со своими сверстниками» [33].

США. В США нет общенациональной учебной программы, разработанной для внедрения программирования или другой цели. Различные штаты внедрили обучение программированию на разных уровнях. Однако на данный момент преподавание программирования в начальной школе ограничено. Правительство оказывает поддержку нескольким организациям, продвигающим изучение программирования в школах. Одной из самых известных организаций является Code.org, получившая широкую поддержку от индустрии высоких технологий в США. Code.org стремится сделать возможным изучение компьютерных наук в каждой государственной школе. Благодаря усилиям Code.org и крупнейшего разработчика ПО Microsoft 24 штата разрешили засчитывать баллы по компьютерным наукам в счет зачетных баллов по математике или естественным наукам, необходимых для окончания средней школы [34]. Некоторые штаты, включая Арканзас, Вирджинию и Индиану, стали включать вычислительное мышление и цифровую грамотность в учебную программу начальной школы.

Анализ примера школьного округа г. Питсбург

В рамках этого примера мы проанализируем переход от традиционной учебной программы к программе, включающей обучение навыкам вычислительного мышления (ВМ), в 1-12 классах в пригородном школьном округе г. Питсбург. По состоянию на 2016 год школьный округ Сауз Файет насчитывал более 3 000 учеников четырех школ, расположенных в одном кампусе. По-

рядка 12% учеников в округе имеют право на бесплатные или льготные обеды, то есть этот округ можно считать относительно богатым по сравнению с другими округами в штате Пенсильвания, где бесплатные или льготные обеды получают в среднем 40% учеников.

Модель перехода называется «Модель STEAM-студии для инноваций» (STEAM — научно-технические дисциплины и гуманитарные науки) и предполагает организацию экспериментальной учебной деятельности для учеников, которая способствует популяризации ВМ в школе и за ее пределами.

Преобразование учебной программы началось в 2010 году с создания условий для активизации и поддержки инноваций. Прежде всего, округ инвестировал в технологическую инфраструктуру: все школьные здания были подключены к высокоскоростной широкополосной связи. Все настольные компьютеры были подключены к сети 1 Гб. Округ предоставляет девайс каждому ученику: iPad для учеников 1 и 2 классов, ноутбук, планшет для учеников старших классов. Наряду с оборудованием округ также расширил штат сотрудников, нанял нового директора по информационным технологиям и инновациям и помощника по информационным технологиям.

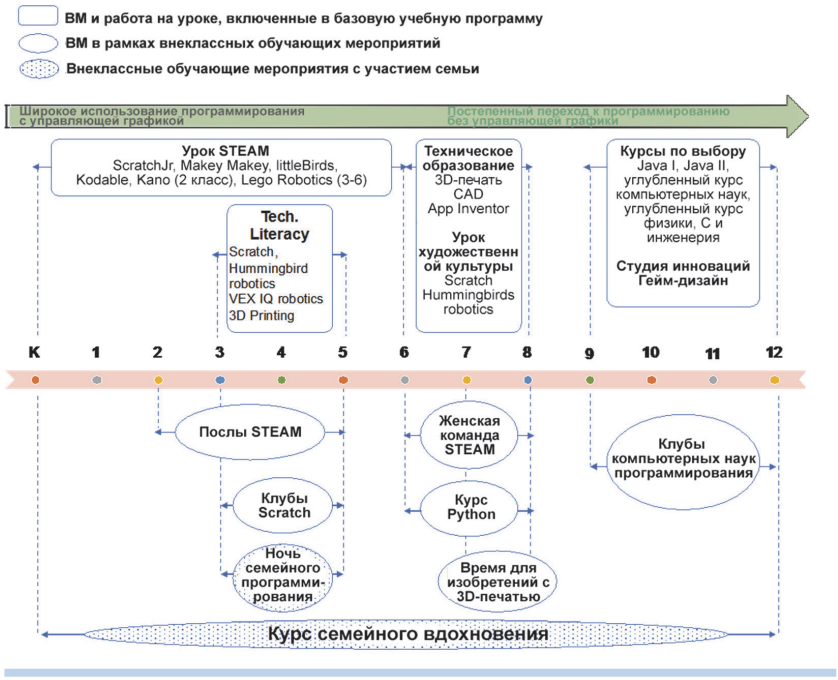
На начальном этапе инициативные «ведущие учителя» начали использовать Scratch в учебной программе начальной и средней школы. Директор по информационным технологиям помогал учителям разрабатывать стратегии обучения и управления работой в классе. Со временем Scratch стали использовать не только в средней, но и в начальной школе на уроках художественной культуры, английского языка и математики, где его применяют все шире. Администрация округа поняла, что традиционные практики повышения квалификации учителей в сфере ВМ не работают. Поэтому округ нанял преподавателей STEAM, заменив или создав новые рабочие места с детского сада до 8 класса. Преподаватели STEAM разрабатывают учебные программы вместе с другими учителями, чтобы поддержать и улучшить традиционное обучение, используя ВМ.

Руководство округа приобрело коммерческий онлайн-инструмент для схематичного отображения учебной программы. С помощью этого инструмента они отобразили каждый элемент учебной программы в соответствии с понятиями ВМ и необходимыми навыками. Схема помогла определить учебные мероприятия, требующиеся для выявления пробелов и повторений в обучении. Для проведения этих мероприятий округ построил лаборатории STEAM-студии в каждой школе, где реальные задачи решают в педагогическом контексте конструкционизма. Кроме того, для учеников, интересующихся программированием, предусмотрены внешкольные учебные мероприятия, где они могут заниматься своим увлечением после школы. Округ также использует внешкольные мероприятия в качестве инновационной программы-инкубатора, где можно опробовать новые методы до их включения в учебную программу.

На рисунке 2.3 показано, как ВМ было включено в школьную учебную программу и программу внеклассных мероприятий.

Партнерства имеют решающее значение для успеха модели STEAM-студии в Питсбурге. Округ извлек пользу из активного участия в Питсбургской сети Remake Learning — консорциуме новаторов и более 200 организаций, работающих вместе для создания возможностей для обучения детей и молодежи посредством использования технологий, средств массовой информации и искусства. Модель также поддержали ключевые специалисты инициативы по внедрению ВМ в учебную программу, основанной Международным обществом по внедрению технологий в образование (ISTE) и Ассоциацией преподавателей компьютерных наук (CSTA) при поддержке Национального научного фонда (NSF).

Рисунок 2.3. Схема учебной программы и технологических инструментов для разных классов



Источник: Образование с ориентацией на будущее: формирование новаторов завтрашнего дня [35]

Способности учеников в сфере инноваций и лидерства, развившиеся в результате применения новой модели, оцениваются по уровню их участия в Программе новых инновационных лидеров. Стандартизированные тесты, основанные на академических стандартах Пенсильвании, показали, что оценки по ежегодным тестам штата по чтению и математике в целом по округу были самыми высокими на юго-западе Пенсильвании с 2015 года.

2.1.2. Страны, рассматривающие возможность включить программирование в школьную программу

Несмотря на отсутствие у некоторых стран конкретных планов по включению программирования в учебные программы, они в настоящее время изучают возможности включения программирования в школьную программу.

В **Китае** программирование еще не включено в программу доуниверситетского образования. Интересным примером в Китае является курс CS0 «Компьютеры для высшего образования», который должны пройти все студенты университетов, независимо от специальности. Этот обязательный курс посещают порядка шести миллионов студентов в год с момента его введения в 1997 году. В последние годы акцент в этом курсе смещается в сторону вычислительного мышления. Обязательный характер курса CS0 дает неожиданные положительные результаты, например, был разработан курс по вычислительному мышлению для глухих студентов [36].

В **Германии** включение программирования в учебные программы происходит по-разному в 16 землях, составляющих Федеративную Республику. В пяти землях компьютерные науки (КН) являются обязательными для всех учащихся средней школы, в пяти других землях КН не пре-

подают вообще. В земле Шлезвиг-Гольштейн школы могут сами решать, преподавать ли КН в рамках обязательного или факультативного курса. Баден-Вюртемберг ввел новую государственную учебную программу, в которой КН является частью междисциплинарного подхода, и изучение программирования включено в задачу освоения медийной грамотности [37].

В **Нидерландах** компьютерная наука является факультативным предметом в системе среднего образования. Национальный институт по разработке учебных программ работает над составлением учебной программы КН [3]. Мы также знаем о следующих планах на 2016 год из работы [25]. В **Норвегии** программирование включено в учебные планы для младших классов 143 средних школ в рамках пилотной программы, однако пока нет окончательного плана по обязательному изучению программирования. В Греции в рамках доклада 2006 года, подготовленного Комитетом по вопросам непрерывного образования Парламента Греции, были внесены предложения по включению вычислительного мышления в учебную программу в качестве краткосрочного приоритета. В **Швеции** Шведское национальное агентство по образованию разрабатывает механизмы развития цифровой компетенции и навыков программирования у детей в рамках национальных учебных программ. В **Японии** Министерство образования, культуры, спорта, науки и технологий объявило, что в 2020 году компьютерное программирование станет обязательным предметом в начальных школах, в 2021 году — в младших классах средней школы и в 2022 году — в старших классах средней школы.

2.1.3. Степень включения в учебные программы

Большинство стран начинают включать программирование в учебные программы на уровне среднего образования, хотя в настоящее время все больше стран вводят уроки программирования на уровне начального образования. Поскольку программирование стало популярным в школах только в последние годы, пока нет достаточных обоснованных научных данных о том, какой возраст следует считать оптимальным для начала изучения программирования. Восемь европейских стран включили программирование в учебную программу начальной школы: Эстония, Франция, Испания, Словакия, Великобритания, Финляндия, Польша, Португалия. Эстония и Словакия — две европейские страны, включившие программирование в учебные программы на всех уровнях школьного образования. В Словакии занятия являются обязательными, в Эстонии — факультативными. В Великобритании изучение программирования является обязательным в государственных школах и факультативным в частных школах, хотя многие частные школы предпочитают включить этот предмет в свои программы.

Израиль также включил программирование в учебные программы на всех уровнях образования. На протяжении многих лет компьютерные науки преподавали в старших классах. Постепенно этот предмет был введен в младших классах средней школы, а затем в начальной школе. Помимо уровня включения серьезные последствия также будет иметь выбор обязательных или факультативных уроков программирования. Например, в США во многих штатах курсы компьютерных наук являются факультативными. Когда школьным округам задают вопрос, почему в их школах не проводятся занятия по компьютерным наукам, они обычно указывают две причины: одна из них заключается в отсутствии интереса со стороны учеников. Но проблема отсутствия интереса на самом деле обусловлена плотным учебным графиком. Второй типичный ответ — нехватка квалифицированных учителей для проведения этих занятий. Однако если компьютерные науки не будут включены в перечень обязательных курсов, школьные округи и государственные департаменты, естественно, не захотят выделять ограниченные бюджетные средства на подготовку учителей по необязательному предмету. Поэтому некоторые специалисты считают, что факультативность занятий по программированию может создать порочный круг, который ограничит возможности доступа к образованию в сфере программирования для многих детей.

Существует три основных подхода, которые страны использовали для включения программирования в учебную программу. Эти подходы описаны ниже.

Подход А. Введение нового специально разработанного предмета

В 12 европейских странах программирование / информационные технологии сами по себе являются отдельными предметами. Преподавание программирования в рамках отдельного предмета гарантирует, что для этих целей будет выделено заранее определенное количество часов. Учебные программы большинства стран уже перегружены, что предполагает определенные компромиссы: некоторые предметы придется сократить / исключить для включения в программу программирования [3,38].

Названия курсов программирования и стран, в которых эти курсы преподают:

- Разработка программного обеспечения (в Австрии)
- Информатика (в Венгрии, Польше, Словакии)
- Программирование (в Словакии, Чешской Республике)
- Программирование (в Ирландии)
- Компьютерные науки (в Израиле)
- Компьютерная техника (на Мальте)
- Компьютерные системы (в Великобритании)
- Информационные технологии (Литва)
- Программное обеспечение (Южная Корея)

Подход Б. Включение в уже изучаемые предметы

Программирование все чаще включают в другие предметы, особенно в программу изучения математики и иногда естественных наук. Преимуществом изучения программирования в рамках других предметов является отсутствие необходимости менять распределение учебных часов. Для внедрения этого подхода учителя должны обладать высокой квалификацией, но при этом нет необходимости «урезать» материал, поскольку то, что раньше преподавали без программирования, теперь преподают с программированием. Сам подход применим только в том случае, если учителя могут научить основным понятиям термодинамики, таким как тепло, давление, объем и температура, с теми же задачами, что и раньше, только теперь ученики должны будут решить эти задачи с помощью кода. Таким образом, они будут одновременно изучать физику и программирование.

Программирование было включено в следующие уже изучаемые предметы в указанных странах:

- математика (в Дании, Эстонии, Франции, Испании);
- физика (в Дании);
- химия (в Дании);
- технология (в Эстонии, Франции);
- домоводство (в Южной Корее, курс «sil-gwa»).

Подход В. Только межпредметная стратегия

Финляндия — единственная страна в Европе, которая выбрала межпредметную стратегию. В рамках этой стратегии «компетенция в сфере ИКТ» является одной из семи междисциплинарных компетенций, определенных Базовой национальной учебной программой, и все изучаемые предметы (в частности, уроки математики и труда) должны способствовать развитию компетенции в сфере ИКТ, начиная с 1 класса.

Анализ примера Великобритании

До 2012 года: «Отключение или перезапуск?»

В докладе Королевского общества 2012 года «Отключение или перезапуск?» было отмечено крайне неудовлетворительное качество компьютерного образования в школах Великобритании в то время. К 2012 году в Англии была разработана учебная программа в сфере ИКТ с акцентом на двух областях: цифровая грамотность и информационные технологии. Эти две области не включали компьютерные науки и вычислительное мышление. Результатом стал приоритет в обучении использованию основных инструментов, которые ученики уже освоили за пределами школы. Скучная учебная программа сформировала негативное отношение к ИКТ, которое подорвало интерес большинства учащихся к занятиям по ИКТ [39].

2012-2017 гг.: реформа стандарта компьютерного образования

В ответ на результаты, представленные в докладе «Отключение или перезапуск?», в 2012 году Англия, Шотландия, Уэльс и Северная Ирландия начали реформу компьютерного образования. Основным аспектом реформы стал новый национальный стандарт компьютерного образования: Англия сделала информатику обязательным предметом для детей в возрасте от 5 до 16 лет. В ранее разработанную учебную программу по ИКТ этот предмет добавил третий аспект — компьютерные науки. Шотландия также обновила свою «Учебную программу для отличной учебы — технологии» исходя из этих трех аспектов. С 2018 года Уэльс заменил свою учебную программу по ИКТ на «Учебную программу для жизни» с элементами компьютерных наук. Северная Ирландия начала внедрять предмет «Использование ИКТ» в учебную программу. Все четыре учебные программы объединяет особое внимание, уделяемое компьютерным наукам с целью овладения навыками вычислительного мышления.

Министерство образования вкладывает средства в технические ресурсы для содействия внедрению новой учебной программы в школах. Например, в рамках программы Barefoot Computing Project были выделены бесплатные учебные материалы по компьютерным наукам и проведены семинары по всей Великобритании для 40 000 учителей начальных классов, не имеющих соответствующего опыта. Что касается аппаратного обеспечения, у школ был доступ к недорогому оборудованию, например, Raspberry Pi и micro:bit от BBC.

Через пять лет был проведен обзор новой политики 2012 года, отчет о котором был представлен Королевскому обществу в 2017 году. Этой практики стоит придерживаться в будущем, поскольку правительства, как правило, больше склонны переключаться на новые инициативы, а не проводить научные исследования прошлых начинаний с целью понять, какие меры сработали и какие проблемы остались. Обзор Королевского общества показал, что основные проблемы связаны со способностью учителей принять новую учебную программу: хотя по данным правительства Великобритании 70 процентов учащихся в Англии имеют возможность пройти курс обучения компьютерным наукам для получения Общего свидетельства о среднем образовании[a], только 11 процентов сделали это в 2017 году. В этой связи правительства недавно начали внедрять новые инициативы, описанные ниже, для решения этих проблем.

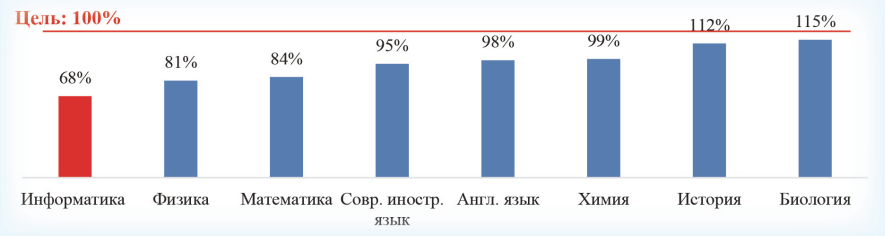
2018 и последующие годы: проблемы и планы на будущее

1. Нехватка квалифицированных учителей

Первая актуальная проблема для проведения реформ в Англии — это нехватка квалифицированных учителей информатики. Как показано на рисунке, только 68 процентов учителей, необходимых для преподавания этого предмета, были доступны для приема на работу с 2012 по 2017 год, по другим предметам этот показатель был намного выше. Для нанятых учителей нет четко определенных требований к повышению квалификации. Например, учителя в Шотландии должны ежегодно проходить 35 часов непрерывного повышения квалификации, но не существует условий для определенного контента. Опрос Королевского общества, в котором приняли участие порядка 900 учителей, отвечающих за компьютерное образование, показал, что почти половина учителей оценила свою уверенность в преподавании информатики на уровне 7 и ниже баллов из 10 (Королевское общество, 2017).

Нехватка учителей информатики в Англии

Процент найма учителей по сравнению с целевыми показателями в Англии (2012-2017 гг.)



Источник: [39].

Для решения проблемы найма квалифицированных учителей в ноябре 2018 года Министерство образования создало Национальный центр компьютерного образования, вложив 84 млн. фунтов стерлингов (techUK, 2018) с целью предоставить ресурсы для непрерывного профессионального развития учителей информатики по всей Англии, что должно повысить уверенность учителей и результативность новой учебной программы. Центром руководят организация STEM Learning, фонд Raspberry Pi и Британское компьютерное общество, объединяющие опыт частного сектора, правительства и гражданского общества (National Centre for Computing Education, 2018).

2. Гендерный дисбаланс

В образовании в области компьютерных наук в Великобритании преобладают в основном мужчины. По данным Объединенного совета по квалификациям [29], среди 16-летних учащихся в 2017 году только 20 процентов претендентов на получение общего свидетельства о среднем образовании по информатике были девочки.

В июне 2018 года Министерство образования пригласило соответствующие организации принять участие в «Пилотной программе по обеспечению гендерного баланса в сфере вычислительных технологий» в рамках тендера на сумму 2,4 млн. фунтов стерлингов. Цель программы состояла в разработке «разнообразных инновационных мер» для увеличения

числа девушек, изучающих компьютерные науки, с помощью исследования для оценки воздействия, которое позволит расширить программу в будущем.

Отсутствие исследований в области компьютерного образования в рамках полного среднего образования

Сегодня большинство исследований в области компьютерного образования сосредоточены на высших учебных заведениях, лишь небольшое количество исследований затрагивают уровень полного среднего образования в Великобритании. Это ограничивает потенциал принятия обоснованных решений в сфере компьютерного образования. По данным Королевского общества, «если мы собираемся потратить 1 миллиард фунтов стерлингов на зарплаты учителей информатики, мы можем себе позволить потратить 200 миллионов фунтов стерлингов на использование фактической информации для повышения эффективности их работы на 20 процентов» (Foxall and Mist, 2018).

Две инициативы, направленные на решение этой проблемы, заключались в создании нового исследовательского центра в области компьютерного образования на базе Королевского колледжа в Лондоне и составлении совместного доклада «Использование исследований в сфере образования» Королевским обществом и Британской академией [40].

[a] Общее свидетельство о среднем образовании — это сведения об образовании с разбивкой по предметам, изученным к 16 годам на уровне средней школы за два года.

2.2. Вторая голова птицы: программирование в рамках внешкольного образования

«Преобразование системы образования — самый серьезный вызов и самая большая возможность для движения мейкеров. Ученики стремятся сами управлять своим образованием, стремятся участвовать в творческих и стимулирующих процессах. Многие понимают разницу между болью образования и удовольствием от реального обучения. К сожалению, они вынуждены искать возможности проявить себя и показать, на что они способны, вне стен школы».

- Дейл Доэрти [41]

Процесс включения программирования в формальное школьное образование будет, вероятно, проходить медленно, но неуклонно на протяжении ряда лет, так как очевидными станут серьезные проблемы, связанные с необходимостью менять мировоззрение и навыки учителей. Число неформальных внеклассных мероприятий по изучению программирования, менее ограниченных правилами и бюрократической инерцией, растет во всем мире. В этом разделе нашей книги мы рассмотрим проблемы и формирующиеся тенденции в сфере изучения программирования в рамках внеклассных мероприятий.

2.2.1. Порядок и субъектность при изучении программирования

«Формальная обстановка, как правило, может обеспечить механизм для развития систематического мышления и подходов, а неформальная обстановка может помочь детям развить мотивацию и определить сферу интересов. Идеальная обстановка должна совмещать и то, и другое».

- Митчел Резник

«Последние сто лет мы считали школу основным местом учебы. Но школа все чаще перестает быть единственным местом учебы».

- Карен Бреннан [42]

История внеклассного образования насчитывает несколько веков — собственно говоря, внеклассное образование существовала до того, как кто-то придумал предметы, уроки, учебники и «учебную программу». С течением времени приоритеты изменились, и внеклассное образование стало основным, а художественная культура, спорт, игры, активный отдых, музыка и танцы стали «дополнительными». Не все образование можно было описать старой пословицей «Делу время, потехе час». Долгое время разные теории образования, в том числе придавали большое значение внеклассному образованию [43,44]. За последние десятилетия маятник качнулся в сторону признания значимости внеклассного образования, однако происходит этот процесс очень медленно. Учитывая современное восприятие ценности образования как инструмента для развития человеческого капитала, важным событием стало решение компании Google из Alphabet Corporation предоставлять всем своим сотрудникам свободное время — практика, связанная с разработкой революционных продуктов [45].

Освоение навыков вычислительного мышления через программирование имеет особые отношения с внеклассным образованием. Программирование — это образовательная деятельность, которая возникла во внешкольных клубах и сейчас переходит в школы. Внешкольные занятия или занятия в выходные дни, как правило, не упорядочены или менее упорядочены, чем урок в обычном классе. Участие во внеклассных мероприятиях, как правило, добровольно, а уроки в школе обязательны, нравятся они вам или нет. Дети, занимающиеся программированием после школы, чем-то напоминают сотрудников компании Google, которые по пятницам наслаждаются своим свободным временем, — при желании можно валять дурака, но есть вероятность, что вы предпочтете заняться делом по собственному желанию. Большинство детей предпочли бы остаться без присмотра, если бы у них появилась возможность заняться чем-то интересным с друзьями, однако некоторым детям, похоже, нужно хотя бы иногда говорить, что делать.

Карен Бреннан считает, что для понимания этого феномена мы должны понять взаимосвязь между порядком и субъектностью [46]. Порядок в этом контексте означает правила, роли и ресурсы в учебных средах. Бреннан приводит пример плана урока как ресурса, представляющего собой явную форму порядка. Учитель, исполняющий роль эксперта в классе, считается примером предполагаемого или подразумеваемого порядка. Под субъектностью ребенка понимается способность ребенка определять и достигать целей обучения. Бреннан подробно описывает, почему порядок и свобода воли должны сосуществовать. За несколько лет до работы Бреннан Дэниэл Шварц представил веские аргументы, ставящие под сомнение утверждение о том, что свобода воли имеет преимущественное значение для обучения. Дело не только в том, что некоторым детям нравится, когда им говорят, что делать, а другим — нет. Шварц ввел понятие «продуктивной субъектности» — обучение происходит, когда мы участвуем в построении смысла в сотрудничестве с другими [47]. Одной из причин эффективности сотрудничества является то, что в группе мы можем что-то создать, например, представить идею, а затем мы наблюдаем за влиянием, которое оказывает то, что мы создали. Экспериментальное исследование значимости обучения показало, что люди, обучавшие других, учатся лучше, когда наблюдают за тем, как их ученики отвечают на вопросы, а не когда отвечают на вопросы сами. Смежное исследование показало, что люди, наблюдавшие за тем, как оценивали их «ученика», усвоили больше, чем те, кто получал оценку своей работы, — во втором исследовании в роли учеников выступали придуманные компьютеризированные агенты [48]. Одна из причин эффективности программирования заключается в том, что программирование может в максимальной степени использовать этот рекурсивный эффект обратной связи.

Нам важно понять, почему программирование не следует преподавать в рамках традиционной модели обучения, когда учитель дает указания, а учащиеся работают индивидуально и следуют этим указаниям. Чтобы изучить разницу между упорядоченным обучением и внешкольной средой с высоким уровнем проявления свободы воли, Кафай и Берк создали две среды обучения, в которых десять недель подряд необходимо было создавать цифровые истории в Scratch с одним и тем же преподавателем [4]. Эти две среды и результаты работы представлены в таблице 2.1.

Таблица 2.1. Сравнение учебной среды в классе и клубе

	Упорядоченное классное занятие	Клуб по интересам
Характеристики среды	- Стандартная учебная программа - Ученики должны были представить раскадровку в Scratch - Проекты получили оценки - Посещение обязательно	- Позволяет ученикам работать на своих местах - Клуб не ставит целей перед учениками - Никаких оценок проектам - Учеников поощряют участвовать в заключительном представлении проектов
Результаты	- 90% учеников завершили свою историю -Средняя продолжительность историй составила 2-3 минуты - Истории были основаны на реальных событиях или картинках из фильмов, книг и т.д. - 10% проектов были ремиксами - Ученики дважды обвинили своих одноклассников в мошенничестве в виде ремиксов чужих работ	- 71% учеников завершили свою цифровую историю - Ученики сделали в два раза больше проектов, чем в классе - Истории содержали больше концепций программирования: интерактивные художественные работы и видеоигры - 26% проектов были ремиксами

Источник: Кафай и Берк, 2014.

Эти результаты доказывают, что клубы по интересам занимают детей больше, чем упорядоченные занятия в классе. Хотя процент учеников, завершивших свои проекты, был меньше в клубе, ученики в клубе создали намного больше проектов, чем их сверстники в классе, использовали больше креативных навыков программирования и вдвое больше сотрудничали со своими сверстниками. Чтобы найти золотую середину между занятиями в классе и клубами, Митчел Резник предлагает «хорошим педагогам и хорошим наставникам плавно переходить от роли катализатора к роли консультанта, посредника и помощника» [6]. Резник описывает четыре роли:

- *Катализатор*: учителя «высекают искру», ускоряющую процесс обучения, например, учитель может показывать образцы проектов, когда ученик зашел в тупик на ранней стадии проекта. Учитель или наставник может задавать вопросы, чтобы подстегнуть творческое мышление: «Что натолкнуло тебя на эту идею?», «Что произойдет, если ты изменишь этот код?».

- *Консультант*: учителя могут подсказывать со стороны, а не вещать с кафедры. Их роль заключается не в том, чтобы давать указания, а в том, чтобы оказывать поддержку в том

виде, как бизнес-консультант поддерживает руководителя. Эта роль очень сложная, так как учитель должен обладать достаточной квалификацией, чтобы давать советы и вносить предложения, а не просто решать проблемы ученика.

- *Посредник*: идея заключается в том, чтобы наставник помог создать «сообщество совместного обучения», в котором ученики помогают друг другу. Для этого учитель должен знать сильные и слабые стороны каждого ученика, а также управлять эмоциональной составляющей работы и следить за тем, чтобы ученики вносили свой вклад, а не просто пользовались подвернувшимися возможностями.

- *Помощник*: это, пожалуй, самая инновационная роль по сравнению с традиционными ролями учителя. В этом случае у учителя есть собственный проект, для которого ему необходима помощь и вклад учеников, выступающих в качестве помощников в его проекте.

2.2.2. От внеклассных занятий к обычной школе

«Это не школы, какими мы их знаем, это общественные клубы, включающие от нескольких сотен до многих тысяч членов. Каждый клуб имеет здание, в котором члены клуба занимаются танцами и вместе проводят время. Члены школы самбы собираются вместе вечерами в субботу и воскресенье, чтобы потанцевать, выпить и встретиться с друзьями. В течение года каждая школа самбы, выбрав тему своего представления на следующем карнавале, отбирает ведущих исполнителей, пишет и переписывает стихотворные тексты, ставит и разучивает танцы. Возрастной диапазон членов школы колеблется от детского возраста до возраста бабушек и дедушек, в нее входят как новички, так и профессионалы. Но они танцуют вместе, и каждый учится и обучает, танцуя. Даже ведущие исполнители должны разучивать свои сложные роли».

- Сеймур Пейперт [49]

Почти 50 лет назад Сеймур Пейперт представил себе «вычислительную школу самбы» по аналогии со школами самбы в Бразилии, где ученики всех возрастов могли бы собираться и делать что-то вместе. Хотя идея Пейперта еще не полностью реализована, во всем мире для вычислительных клубов, пространств мейкеров и внеклассных занятий по программированию наметился обнадеживающий прогресс. Особенности таких внеклассных занятий [50]:

Соединение интересов учеников.

Объединение учеников с общими интересами сплачивает их на личностном уровне. Такое сплочение становится мощным стимулом для учеников заниматься еще усерднее.

Объединение учеников в сообщество.

Сообщества позволяют ученикам взаимодействовать с другими учениками со схожими интересами и учиться друг у друга, поощряя при этом совместную работу.

Ученики совершенствуются путем «мейкинга».

Уникальной особенностью большинства внеклассных мероприятий является то, что они объединяют учеников для «мейкинга» физических или цифровых проектов. Движение мейкеров в последнее время становится все более популярным во всем мире, поскольку создание чего-то реального дает ученикам стимул учиться, возможность использовать свои знания на практике и понимание, чему еще они должны научиться.

Несмотря на то, что движение мейкеров для детей в целом основано на внешкольных меро-

приятиях, на базе этой философии можно создать и школы. Наглядным примером в дискуссии о будущем образования является основанная в 2011 году школа Brightworks для детей от 5 до 15 лет в Сан-Франциско, где для обучения детей используется полностью проектно-ориентированный подход. Вместо обычной линейной учебной программы с последовательностью уроков, разбитых по предметам, школа продвигает собственную педагогическую инновацию — учебную дугу. Учащиеся изучают темы в каждой дуге, которая включает этапы исследования, выражения и экспозиции. Митч Резник вспоминает о посещении школы Brightworks с людьми, которые скептически относятся к проектно-ориентированному подходу школы к обучению [6]. Резник выступает против распространенной идеи о том, что дети должны сначала усвоить основы, а затем применять эти основы в проекте, и дает понять, что традиционная форма обучения в корне ошибочна:

«На учащихся непрерывным потоком изливаются учебные материалы и приемы решения задач, заостряющие внимание на конкретных понятиях. Тут их учат, как перемножать дроби. Там — как вычислить, какой выигрыш в механической силе дает зубчатая передача. На первый взгляд такой подход вроде бы имеет смысл. Но когда учащиеся решают задачи в отрыве друг от друга и от реальной жизни, то и знания у них нередко получаются разрозненными, дети не понимают, для чего изучали те или иные темы и где они могут пригодиться».

- Митч Резник [6]

Реализовать концепцию связанных знаний, которую Резник рекомендует в качестве основного подхода к образованию, сложно в отличие от занятий по программированию в рамках внеклассных мероприятий. Философия школы Brightworks на данном этапе может считаться экспериментальной, но это тот эксперимент, за ходом которого педагоги всего мира захотят следить очень пристально, поскольку он дает реальное понимание концепции проектно-ориентированного обучения.

2.2.3. Типы внеклассных занятий по программированию во всем мире

Тип 1. Послеурочные программы

В мире существует множество коммерческих и некоммерческих организаций, предлагающих послеурочные программы. Начиная с Coddy в Российской Федерации и заканчивая Caustics Digital Academy в Лос-Анджелесе, Computhink в Сингапуре, Kodluyoruz и MucitKids в Турции, послеурочные программы становятся все более доступными.
Также повсеместно проводятся различные волонтерские мероприятия на местном уровне. Наиболее распространенными из них являются CoderDojo, Code Club и Clubhouse Network.

CoderDojo — бесплатный внешкольный клуб для детей в возрасте от 7 до 17 лет под руководством волонтеров. Работает в 65 странах. В настоящее время насчитывает более 1 100 активных волонтеров и 45 000 детей.

Code Club — всемирная сеть бесплатных клубов по программированию для детей в возрасте от 9 до 13 лет под руководством волонтеров. Работает в 140 странах. Аудитория более 10 000 клубов Code Club насчитывает 150 000 детей по всему миру.

ClubHouse Network — международное сообщество бесплатных внеклассных сред обучения для малообеспеченной молодежи. 100 клубов ClubHouse находятся в 19 странах и работают с 25 000 молодых людей в год. Целью работы ClubHouse Network является переход от принципа «думай сам» к принципу «сделаем вместе».

Тип 2. Хакатоны

Хакатон — это собрание, на котором программисты должны написать совместный код за короткое время в экстремальных условиях. Хакатоны длятся минимум несколько дней — или выходные — но, как правило, не больше недели [51,52]. Одним из успешных примеров хакатонов, привлекающих молодых участников, является Local Hack Day. Этот хакатон был впервые организован в 2013 году компанией Major League Hacking, расположенной в Нью-Йорке и организующей хакерские соревнования весь год. В декабре 2017 года около 275 сообществ по всему миру приняли участие в 12-часовом хакатоне. Один из хакатонов был организован 14-летним учеником средней школы в Калифорнии.

Тип 3. Мероприятия, посвященные программированию

В рамках таких мероприятий участники получают материалы для проведения определенных занятий, посвященных программированию, в своих сообществах. Учитель, родитель или ребенок может легко провести занятие по программированию, собрав других детей в сообществе и скачав общие указания на сайте. Три самых популярных и успешных международных мероприятия, посвященных программированию, проводятся некоммерческой организацией Code.org — Час кода (Hour of Code), Неделя кода в Европе (Codeweek EU) и конкурс по информатике «Бобер» (Bebras Challenge).

Час кода — часовая учебная программа, разработанная для всех возрастов на более чем 45 языках. Час кода проводится в 180 странах. 263 миллиона человек во всем мире поучаствовали в Часе кода.

Неделя кода — двухнедельные мероприятия для детей, организованные волонтерами по всему миру. Инициатором в 2013 году стала Комиссар Европейской комиссии Нели Крус. Сегодня Неделя кода проводится в 50 странах. В 2016 году в Неделе кода в Европе приняли участие почти миллион человек.

Бобер. Ежегодный конкурс «Бобер» проходит обычно в ноябре. Участникам конкурса от 8 до 19 лет. У каждого участника есть 45 минут, чтобы решить 18 задач из разных областей информатики, не требующих навыков программирования. Задачи делятся по пяти возрастным группам от 8 до 19 лет. Мероприятия можно проводить в школах под наблюдением учителей. Победители получают сертификаты и награды. Некоторые страны проводят второй этап конкурса, как правило, в январе-феврале, для лучших участников первого этапа [25].

День Scratch — это мероприятия, посвященные празднованию выпуска Scratch по всему миру. Участники делятся своими достижениями в Scratch, а энтузиасты используют это мероприятие для увеличения постоянно растущего числа скретчеров.

Тип 4. Детские технопарки

Технопарки — это образовательные площадки для школьников, оснащенные высокотехнологичным оборудованием, чтобы мотивировать следующее поколение высококвалифицированных специалистов в сфере ИКТ и проектирования.

Кванториум — растущая сеть детских технопарков в Российской Федерации. В настоящее время сеть насчитывает 24 технопарка в 19 регионах страны. Примером деятельности технопарка в Москве является организация 94-часового курса самолетостроения на протяжении 9 месяцев с использованием специализированного инженерного ПО Autodesk Inventor.

2.3. Левое крыло птицы: выбор способа повышения профессиональной квалификации преподавателей

Для внедрения новых учебных программ, неотъемлемой частью которых является программирование, необходимо оказать поддержку учителям в плане их навыков и знаний в сфере обучения программированию. Сегодня многие страны сталкиваются с такой серьезной проблемой как нехватка учителей, способных преподавать программирование. Проблема связана с профессиональными предпочтениями выпускников математических факультетов и факультетов компьютерных технологий, которые выбирают более прибыльные профессии, а не преподавание. С другой стороны, дипломированным специалистам в области гуманитарных и общественных наук, составляющим большинство учителей, необходима серьезная подготовка для обучения детей программированию [34].

Когда речь заходит о подготовке учителей к преподаванию программирования или, в целом, к обучению навыкам вычислительного мышления / компьютерным наукам, возникает ряд вопросов или вариантов: a) специально обученные учителя компьютерных наук или учителя других предметов / общих дисциплин?; b) предварительная подготовка учителей или обучение уже работающих учителей на месте работы?; c) очное или онлайн обучение? Мы рассмотрим эти вопросы в данном разделе.

2.3.1. Преподаватель специализированного предмета или преподаватель общего профиля

В Израиле учителя должны иметь диплом по специальности «Преподавание компьютерных наук», чтобы преподавать программирование. Наличие диплома бакалавра по специальности «Компьютерные науки» и официальная аттестация преподавателя, прошедшего подготовку, являются обязательными требованиями для лиц, желающих преподавать компьютерные науки в израильских средних школах. В 2000 году Министерство образования основало Израильский национальный центр для преподавателей компьютерных наук («Махшава»), который стал профессиональным хабом для всех преподавателей компьютерных наук в Израиле. Этот центр контролирует организацию и финансирование штатных учебных программ, семинаров и методических руководств для повышения квалификации учителей. В рамках своей деятельности Махшава проводит летние семинары под названием «Новые области компьютерных наук» для «ведущих учителей», чтобы поддержать их профессиональное лидерство в сообществе преподавателей компьютерных наук. Эта программа направлена на признание ценности опытных учителей-энтузиастов, выступающих в роли тренеров среди коллег в сообществе преподавателей [3].

Ответ на вопрос о том, должны ли все страны перенять опыт Израиля, зависит от контекста и актуальности. В некоторых странах, ярким примером которых является Россия, преподаватели-предметники в средней школе должны иметь высшее образование как в области педагогики, так и по своему предмету. В дошкольных учреждениях и в начальной школе обучение проводят преподаватели общего профиля, хотя даже на этом уровне может возникнуть необходимость пригласить несколько преподавателей компьютерных наук, чтобы организовать предварительное обучение или обучение преподавателей общего профиля на месте работы. Такие преподаватели-предметники могут быть не в каждой школе, но они могут стать ресурсом, используемым при необходимости, например, для обучения и поддержки учителя или группы учителей, столкнувшихся с трудностями. Если речь идет о всеобщем изучении программирования, даже в средней школе оно должно стать частью других предметов, например, физики или биологии. Некоторые специалисты считают, что использование программирования поможет детям стремительно повысить качество изучения таких предметов, как естествознание.[9]

[9] Эта идея лежит в основе концепции адаптивного гейм-дизайна (Scalable Game Design, SGD) — см. (Басс и Гамбоа, 2017).

Таким образом, нужны как преподаватели-предметники, так и преподаватели общего профиля, способные преподавать программирование. В начальной школе часто присутствует только один классный руководитель, который просто должен иметь определенный уровень владения материалом, как правило, не такой высокий, как все думают. Тем не менее, несколько преподавателей компьютерных наук могут принести пользу начальной школе, так как они помогут решить проблемы, связанные с повышением квалификации или обучением учителей. В средней школе, как правило, есть преподаватель компьютерных наук, но преподаватели других предметов также должны развивать навыки программирования, чтобы использовать междисциплинарные преимущества изучения программирования, и делать это последовательно, поскольку учителя, скорее всего, оценят высокую эффективность этой педагогической методики.

2.3.2. Предварительное обучение или обучение на месте работы

Двенадцать европейских стран (Австрия, Болгария, Франция, Эстония, Венгрия, Ирландия, Мальта, Польша, Португалия, Словакия, Испания, Великобритания), внедривших программирование в учебную программу, организуют как обучение на месте работы, так и предварительное обучение, чтобы поддержать учителей в процессе преподавания программирования в классе. Основным организатором обучения обычно выступает ВУЗ, однако коммерческие компании и некоммерческие организации также привлекают к обучению [3].

Южная Корея сталкивается с серьезными проблемами при подготовке учителей, поскольку новая обязательная учебная программа начальной школы, включающая программирование, подразумевает, что учителя, которые раньше никогда не преподавали программирование, должны быстро войти в курс дела. Поэтому Южная Корея начала организовывать обучение на месте работы, чтобы учителя, особенно в начальной школе, могли освоить навыки программирования и работать в рамках новой учебной программы, принятой в 2018 году. К концу 2018 года Южная Корея намерена организовать специализированное обучение в области программного обеспечения для 60 000 учителей начальных школ (30% от общего числа), 6 000 из них пройдут углубленную подготовку. Кроме того, 1 800 учителей младших классов средней школы с дипломами преподавателей информационных технологий / информатики, пройдут дополнительное обучение в области программного обеспечения (Боккони и соавт., 2016). В Новой Зеландии Новозеландская ассоциация преподавателей компьютерных, цифровых и информационных технологий оказывает поддержку учителям, которые должны преподавать программирование, но не соответствуют новым требованиям [3].

При организации предварительного обучения и обучения на месте работе на должном уровне возникают серьезные проблемы. Было проведено интересное исследование, чтобы понять, как представляют себе вычислительное мышление учащиеся, проходившие предварительное обучение в крупном американском университете (Ядав, Стивенсон и Хонг, 2017). Авторы исследования обнаружили, что учащиеся обладали элементарным или упрощенным представлением о вычислительном мышлении, например, они ошибочно полагали, что вычислительное мышление схоже с математикой и решением задач. Лишь немногие учащиеся смогли определить основные компоненты вычислительного мышления: декомпозицию, выделение паттернов, абстрагирование и создание алгоритмов, которые помогают формулировать и решать задачи с помощью компьютеров. Вряд ли многие учащиеся читали канонический текст о компьютерном программировании, содержащий объяснение основного отличия вычислительной науки от математики, который мы приводим ниже:

Компьютерная революция — это революция в том, как мы мыслим и как мы выражаем наши мысли. Сущность этих изменений состоит в появлении дисциплины, которую можно назвать компьютерной эпистемологией, — исследования структуры знания с императивной точки зрения, в противоположность более декларативной точке зрения классических математических

дисциплин. Математика дает нам структуру, в которой мы можем точно описывать понятия типа «что это такое». Вычислительная наука дает нам структуру, в которой мы можем точно описывать понятия типа «как сделать». [53]

Возможно, будущие учителя не использовали такой сложный термин, как «компьютерная эпистемология», для обозначения императивного аспекта компьютерного программирования (программа «указывает» компьютеру, что делать, и в этом состоит императивный аспект) и обозначения его отличия от декларативного аспекта, присутствующего в математике, — «отношение длины окружности к ее диаметру обозначается иррациональным числом π». Полезно знать и задуматься об этом отличии, потому что оно лежит в основе того, почему программирование считают грамотностью, как чтение и умение считать. Тем не менее, у будущих учителей было некоторое интуитивное понимание, хотя, возможно, они смогли сформулировать его научными или философскими терминами.

«Мне кажется, ВМ хорошо вписывается в математику, поскольку они тесно связаны между собой. Я бы попытался объяснить учащимся, почему решение задачи определенным образом логично или что нужно сделать, чтобы решить задачу (так они смогут применить свою логику, чтобы решить задачу последовательно); вычислительное мышление можно применить в рамках совместной работы учащихся над решением математической задачи. Это позволит им решить задачу системно и логично»

Ответы, полученные в ходе опроса учителей, проходящих предварительную подготовку, из работы [54].

Авторы исследования пришли к выводу, что одним из способов повышения осведомленности и освоения навыков ВМ являются дополнения к обязательным курсам по педагогической психологии или образовательным технологиям, которые преподают будущим учителям. Министерство образования США выделило финансирование на программу для ВУЗов, которые учат преподавателей разрабатывать новые программы в сфере образовательных технологий и повышения профессиональной квалификации. Основной способ оценивания подхода студентов, проходящих предварительное обучение, под названием «Изучение отношения к работе с применением компьютера», описан в Приложении 1. Этот способ использовался в исследованиях, целью которых было выявить влияние вводных программ предварительного обучения освоение учителями вычислительного мышления [55,56].

2.3.3. Мировоззрение учителей как препятствие и возможность

Безусловно, мировоззрение учителей и будущих учителей, проходящих предварительное обучение, — это самая большая проблема, требующая решения. Хотя учителя с опытом работы в сфере точных наук или декоративно-прикладного искусства иногда положительно относятся к изучению программирования и компьютерного мышления, мнение учителей об их способности преподавать программирование зачастую трудно назвать положительным. В ходе исследования, проведенного в 2013 году в Южной Африке, были получены интересные результаты [57]. Правительство Южной Африки обратило внимание на явный спад интереса к компьютерным наукам среди учеников старших классов. Была введена программа, в рамках которой ученики осваивали программирование с использованием Scratch в 10 классе, затем изучали языки программирования Java или Delphi в 11 и 12 классах. Среди учителей 10 классов был проведен опрос спустя несколько месяцев после внедрения программы. Были выявлены как положительные, так и отрицательные мнения учителей о Scratch, хотя учителя, похоже, отмечали энтузиазм учеников в отношении Scratch и признавали тот факт, что Scratch помог ученикам проявить свою креативность. Преподаватели сообщили о разных проблемах, в том числе о том, что ученики веселились вместо того, чтобы серьезно заниматься, но весьма показательным был тот факт, что «ученики изучают Scratch такими темпами, что учителя не успевают за ними».

Идея о том, что в рамках определенной деятельности учитель может быть не самым умным в классе, является революционной для большинства учителей. Несмотря на десятилетия, потраченные на проведение реформ «нового образования» и внедрение программ личностно-ориентированного обучения почти во всех странах, само понятие учителя и ученика укоренилось в концепции обучения как передача знаний или навыков от знающего к незнающему. Идея учителя, выступающего в роли консультанта, который может подсказать со стороны, а не вещать с кафедры, оказалась труднореализуемой на практике.

Чтобы не обидеть ни учителей, ни учеников, необходимо найти золотую середину. В то время как некоторые ученики радуются, получив свободу изучать материал без определенного порядка, большинству учеников нужен порядок или поддержка. Очевидно, что к преподаванию программирования не применим традиционный подход, когда учитель демонстрирует навык, а ученик стремится подражать учителю в процессе тренировки или практики — в программировании слишком много настроек или индивидуальных изменений. Также верно и то, что обучение программированию является типичным примером приобретения навыков обучения с поддержкой, доступной через Интернет, через используемую учениками программу или платформу. Совместное обучение само по себе является целью, которую ученики достигают, помогая друг другу. Тем не менее, одно дело, когда учитель не говорит ученику ответ, потому что хочет, чтобы ученик научился выявлять проблемы или находить помощь онлайн. Другое дело, когда учитель не говорит ответ, потому что не имеет о нем ни малейшего понятия. Личностно-ориентированное преподавание на самом деле требует больше усилий, чем передача знаний. Даже при том, что могут быть случаи, когда ученик, а не учитель знает ответ лучше и находит его быстрее, учитель должен быть опытным и уверенным в своих силах в сфере понятий ВМ.

2.3.4. Онлайн-ресурсы, включая системы коллегиальной поддержки

Онлайн-ресурсы полезны для учителей, преподающих разные предметы, но особенно важны для преподавания и изучения программирования. Такие онлайн-ресурсы могут быть представлены в виде онлайн-порталов (например, в Эстонии), специальных сайтов, посвященных программированию, и платформ сообществ (например, в Болгарии, Франции, Финляндии) [3]. Правительства некоторых стран применили другие инновационные методы для поддержки учителей. Например, в рамках «системы наставников» в Финляндии каждой школе предоставляют наставника, который оказывает содействие другим учителям в реализации новой учебной программы в сфере программирования. После разработки MOOC для обучения учителей использованию Scratch и других инструментов, учителя сами стали создавать учебные материалы, включая видео. Учителя научились пользоваться Eliademy, системой управления обучением на базе открытой платформы Moodle, и стали использовать другие платформы, такие как формы Google для обратной связи и Padlet для обмена учебными материалами [58].

Коллегиальное обучение среди учителей — важный компонент обучения преподаванию программирования, в рамках которого онлайн-сообщества дополняются групповыми встречами. Европейская комиссия призывает должностных лиц в ЕС поддерживать сети преподавателей и коллегиальное обучение преподаванию программирования. В некоторых странах учителя проходят подготовку в рамках инициатив «снизу». Например, в Дании Министерство образования не занимается подготовкой учителей к преподаванию программирования. Вместо этого подготовкой учителей занимается движение преподавателей, предпринимателей и программистов под названием Coding Pirates. Еще одним примером являются семинары компании Code.org для преподавателей в США. С 2014 по 2017 годы 30 000 учителей в США прошли подготовку в рамках профессиональных семинаров и конференций. В Англии Министерство образования сотрудничает с такими профессиональными организациями, как Британское компьютерное общество и Computing at School (CAS), чтобы обеспечить подготовку учителей в сфере программирования и вычислительных наук. В рамках этой программы педагоги-наставники, выбранные из числа самых опытных и мотивированных учителей, проходят 5-10-дневную подготовку

в течение шести и более месяцев, затем каждый педагог-наставник помогает 40 учителям в своем сообществе. По состоянию на 2017 год подготовку прошли 350 педагогов-наставников, планируется увеличить их число до 600 в 2018 году. Педагоги-наставники участвуют в работе Сети передовых научно-исследовательских центров в области преподавания компьютерных наук.

Таблица 2.2. Способы, используемые европейскими странами для обучения учителей преподаванию программирования

	Предварительное обучение или обучение на рабочем месте		Инициативы «снизу»	Онлайн-ресурсы
	Министерство образования	ВУЗы, компании, НКО		
Австрия		✓		✓ Digicomp
Болгария		✓	✓ Сообщество Scratch в Болгарии	
Чешская Республика	✓	✓	✓ Союз специалистов по информатике в сфере образования	✓
Дания	✓	✓	✓ Coding Pirates	✓
Эстония	✓ Фонд HITSA: совместный проект правительства, ВУЗов и частного сектора	✓ Фонд HITSA		✓ Вебинары и онлайн-ресурсы фонда HITSA
Финляндия	✓	✓	✓	✓ Code ABC и MOOC Хельсинского университета
Франция	✓ Министерство образования координирует программы подготовки, местные власти внедряют их			✓ Институты Inria и Tangara
Венгрия	✓	✓	✓ Ассоциация преподавателей ИКТ	✓
Ирландия	✓	✓	✓	✓
Литва	✓	✓	✓	✓
Мальта	✓			
Польша	✓	✓	✓	✓
Португалия	✓	✓		
Словакия	✓ Центр методологии и педагогики при Министерстве образования			
Испания	✓			
Великобритания (Англия)	✓	✓	✓	✓ Программа Barefoot

2.4. Правое крыло птицы: контент для обучения программированию

Для успешного обучения программированию необходимы учебные материалы, соответствующие возрасту. В раннем возрасте, когда ученики еще не умеют читать, они могут использовать простые экранные объекты, имеющие цвет и форму, как в Scratch Jr. Так называемые неподключенные среды могут быть особенно интересны младшей возрастной группе. Выбор платформы, без сомнения, важен, но также важно уделять внимание действиям при использовании платформы. Одна из возможностей современных технологий позволяет нивелировать проблемы, с которыми сталкиваются инвалиды, и преодолеть гендерные стереотипы. Поскольку многие платформы, такие как Scratch или Agent Sheets, предоставляются бесплатно с открытым исходным кодом, формируется ошибочное мнение, что объем ресурсов не важен. Мы рассмотрим эти вопросы в данном разделе.

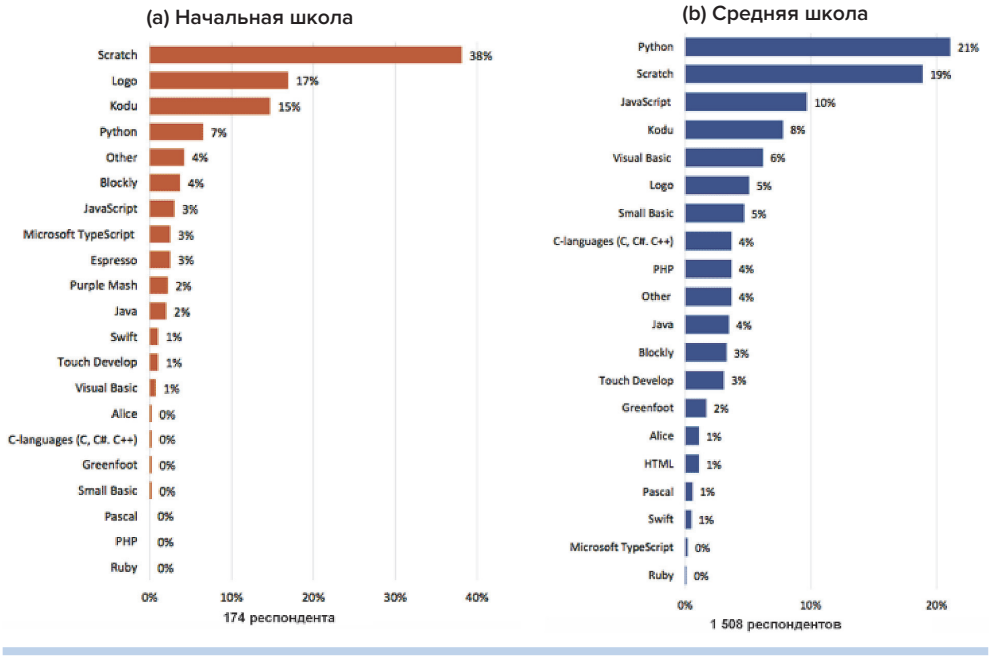
2.4.1. Платформа и вид деятельности

Не утихает спор между так называемыми технофилами и технофобами — людьми, которые считают технологии панацеей, которая исправит все ошибки в образовании, и людьми, которые верят, что технологии только приближают тот момент, когда люди превратятся в автоматы [6,59]. Кэти Дэвидсон беспокоит идея совместной и общественной деятельности, лежащая в основе «нового образования». Она говорит о гибкой разработке ПО как о примере трудовой этики XXI века, где первостепенное значение имеет командная работа и сотрудничество, включая осмысление ошибок друг друга. Прежде чем переходить к выбору высокотехнологичной, низкотехнологичной деятельности или деятельности без гаджетов, Митчел Резник рекомендует учителям поискать занятия и инструменты, которые помогут детям мыслить и самовыражаться творчески [6]. С точки зрения стратегии и цели, которую преследует учитель на занятиях в школе или после школы, не стоит упускать из вида конечную цель. Изучение компьютерного языка или развитие навыков использования такой платформы, как Scratch, — это всего лишь средство для достижения цели; выбор платформы и вида деятельности должен служить цели развития навыков вычислительного мышления через творческое сотрудничество.

В случае с платформами оказалось, что платформы визуального программирования, такие как Scratch, подходят детям в разных контекстах, что подтверждают миллионы пользователей Scratch или скретчеров. Профессиональные языки, такие как Java, могут быть слишком сложными для детей в качестве вводного языка. Например, Мальта ввела Java в свои учебные программы, однако результаты оценки, проведенной в 2014 году, показали, что Java не оправдал себя как средство освоения навыков программирования и творческого решения задач, потому что у него сложный синтаксис, и детям приходится отвлекаться на команды и правила, а не изучать работу кода [3].

Южная Корея решает эту проблему, используя в начальной школе учебные материалы (например, настольные игры), не требующие гаджетов, и языки визуального программирования. Когда ученики переходят в среднюю школу, они начинают изучать более продвинутые языки, такие как Java. Одним из важных моментов, который следует учитывать при разработке контента для обучения программированию, является определение этапа, на котором ученики смогут перейти с визуального на текстовое программирование. На рисунке 2.4 представлены языки, используемые в Великобритании, согласно опросу, проведенному Королевским обществом в 2017 году.

Рисунок 2.4. Языки программирования в школах Великобритании



Источник: Королевское общество, 2017.

Проектная деятельность является значимой и мотивирующей для учащихся всех возрастных групп, может потому, что она дает простую возможность для самовыражения. Большинство детей считают невероятно интересными создание игр для других детей и роботов. Игры и игровые среды разработки кода позволяют детям самим регулировать свою деятельность посредством указаний и обратной связи и дают стимул продолжать работу в виде уровней игры. В ходе исследования, в котором приняли участие учителя начальных классов в Финляндии, было установлено, что многие учителя используют в классе несколько игр или игровых сред разработки кода. Как правило, они начинают с упражнений по координации с детьми, когда один ребенок становится «роботом», а другой дает ему команды. Затем они переходят к играм с обучающими наборами для занятий робототехникой и работе в Scratch Jr. По мере развития навыков детей в классе начинают использовать более продвинутые игровые среды, такие как Scratch, Kodu и Tynker classrooms [38]. Однако при использовании игровых сред не следует заострять внимание на факторах внешней мотивации, таких как баллы и значки, лучше поощрять факторы внутренней мотивации детей, чтобы обучение не зависело только от внешних поощрений.

2.4.2. Инклюзивный подход для учета гендерных аспектов и потребностей учащихся с ограниченными возможностями

Равный доступ к изучению программирования является важной темой, особенно когда речь идет о внеклассных мероприятиях, за которые родителям, возможно, придется платить. В этом разделе мы рассмотрим вопрос равного доступа для устранения гендерного дисбаланса и учета потребностей учащихся с ограниченными возможностями. Недостаточная представленность женщин в отрасли ИКТ широко обсуждается во всем мире и является частью проблемы недостаточной представленности женщин в сфере точных наук. Однако проводимая информационно-разъяснительная работа, похоже, меняет ситуацию. В гостевом блоге основатель code.org Хади Партови пишет о растущем числе девочек, изучающих программирование http://blogs.worldbank.org/education/schools-are-teaching-10-million-girls-code-gender-equality-real-possibility.

Удовольствие, которое приносит программирование благодаря мгновенной обратной связи и бесконечным вариациям, может стать орудием для преодоления гендерного дисбаланса в сфере точных наук. Девочкам, которым не хватает уверенности в своих способностях в изучении наук, возможно, из-за того, что им было не интересно заниматься наукой в отрыве от реальности, может понравиться наука благодаря успехам в программировании.

Всегда будут девочки, мечтающие создавать боевых роботов-чемпионов, однако исследования показывают, что существуют аспекты креативного программирования, которые больше нравятся девочкам, чем мальчикам: сочинение историй, использование новых материалов и носимых датчиков или управляющих устройств, таких как Lilypad Arduino [4,60]. Келлехер в рамках своего исследования проверила гипотезу о том, что сочинение историй и творческое воображение более интересно девочкам с использованием Storytelling Alice — модифицированной версии 3D среды программирования Alice. Исследование показало, что ученицы средней школы проводят на 42% больше времени за программированием со Storytelling Alice, чем девочки, использующие обычную среду Alice, и в три раза чаще уделяют программированию дополнительное время. В продолжение своей работы Келлехер разработала Looking Glass — среду программирования для «создания и совместного использования анимационных историй, простых игр и даже виртуальных питомцев».[10]

Ранее наблюдался существенный разрыв между развитием чтения и письма, особенно в доисторические времена, и развитием языка Брайля для слабовидящих и незрячих. Азбука Брайля была разработана в начале XIX века слепым подростком Луи Брайлем. Не все знают, что Брайль создал ремикс оригинального языка, разработанного Шарлем Барбье де ла Сером, зрячим французским капитаном артиллерии Наполеона.[11] По приказу Наполеона Барбье попытался создать тактильный код, который могли бы использовать солдаты по ночам в артиллерийских лагерях, чтобы без света пальцами читать напечатанный текст и таким образом оставаться невидимыми для врага. Изобретение Барбье было слишком сложным и никогда не использовалось, но модификации, которые внес Брайль с помощью сокурсников из Королевского института для слепых, оказались весьма эффективными. Еще одним замечательным примером эффективности совместной работы в творчестве служит развитие языка жестов для глухих детей в Никарагуа после того, как в 1970-х годах впервые была открыта школа для глухих и созданы условия для продолжительных контактов — этот случай описан в журнал Science [61].

В XXI веке благодаря более глубокому пониманию и восприятию формирующаяся грамотность в сфере программирования с большой вероятностью будет доступна людям с ограниченными возможностями с самого начала без длительного периода ожидания, как это происходило раньше. На рисунке 2.5 представлены некоторые из доступных технологий — серийно производимая технология, в которой движения и последовательности постукиваний пальцами заменяют клавиатуру.

Интересно, что продукт позиционируют как доступное всем устройство для ввода текста одной рукой, причем слепые или слабовидящие составляют только одну группу пользователей. Для таких сред с управляющей графикой, как Scratch, которая очень популярна среди детей, предприниматели пытались создать возможности доступа с помощью звуков и сенсорных экранов [62]. Еще один интересный способ для слепых или слабовидящих изучить программирование — это 3D-печать, когда код, написанный учеником, создает осязаемые объекты. Это обеспечивает оперативную оценку результатов кода, которую зрячий пользователь воспринимает

[10] Доступно для бесплатного скачивания, создано интернет-сообщество, есть простые возможности создания ремиксов и совместного использования http://lookingglass.wustl.edu/.
[11] http://www.afb.org/info/living-with-vision-loss/braille/what-is-braille/123; https://www.avh.asso.fr/fr

Рисунок 2.5. Технологии для людей с ограниченными возможностями, позволяющие им получить доступ к программированию и работе на компьютере

Источник: https://www.tapwithus.com/

Доступные технологии для людей с ограниченными возможностями, помогающие им работать на компьютере

- Сигналы мышц / электродные датчики
- Слежение за движением глаз / монитор, следящий за движением глаз
- Слежение за положением головы / движением головы
- Световое устройство ввода
- Система с использованием давления воздуха
- Портативный компьютер Брайля
- Преобразование текст-речь / речь-текст
- Носимая клавиатура, активируемая постукиванием (на фото справа)

Источник: еженедельник Computerworld, февраль 2010 года

как данность [63]. Одна из интересных возможностей для совместной работы, зависящая от доступности программирования, возникает, когда дети или молодые люди занимаются программированием и созданием, ставя перед собой определенную цель.

Кэти Дэвидсон сделала важный вывод об образовании XXI века. Она утверждает, что пришло время заменить то, что она считает изжившей себя моделью образования — передачу знаний от учителя-профессионала к неопытному ученику. Вместо этого она предлагает поддерживать целевые проекты, в которых учитель помогает ученику определять и решать задачи из реальной жизни [59]. Она приводит в пример Сару Хендрен, преподавателя Инженерно-технического колледжа им. Олина, которая работает со студентами над разработкой вспомогательных технических средств. Поскольку все больше детей изучают программирование, и в колледж поступают все более опытные школьники, курс программотехники, который преподает Сара Хендрен, вполне может стать обычным повсеместно распространенным курсом. А пока начать может даже ученик средней школы.

Недавние исследования подробно описали проблемы, связанные с играми для мотивации пациентов, перенесших инсульт [64]. Игры могут помочь разнообразить повторяющиеся действия, необходимые для реабилитации таких пациентов. Однако в существующие игры, по-видимому, не так интересно играть. Педагог из средней школы штата Джорджия в США уже начала решать эту проблему, разработав для семиклассников занятия в Scratch, а также ресурсы для других учителей, которые захотят создать ремикс этого материала.[12] Эти ресурсы охватывают такие понятия программирования, как динамическое программирование игр. Уровень сложности автоматически повышается по мере того, как игрок овладевает навыками, что поддерживает его мотивацию.

[12] http://scratched.gse.harvard.edu/resources/lets-play-dynamic-physical-therapy-games-using-scratch-and-makey-makey-plus-turning-code-c

2.4.3. Значимость кадровых ресурсов, условий деятельности и оборудования

Кадровые ресурсы

Когда в школьном округе Сауз Файет под Питсбургом была разработана учебная программа по человеко-ориентированному проектированию и всеобщая грамотность в сфере программирования была положена в основу студийной модели обучения, одним из первых решений было создать новую должность директора по технологиям и инновациям. Округ внедрил программу интегрированного программирования, в рамках которой учителя общего профиля должны были включить программирование в свои уроки, — учителя должны были быстро учиться, некоторые справились с этой задачей, но многие не смогли. Затем округ нанял преподавателей научно-технических дисциплин и гуманитарных наук, которые работали непосредственно с учениками и помогали другим учителям. Программа была успешно реализована с точки зрения улучшения показателей когнитивных достижений и способностей учащихся в области вычислительного мышления [35]. Это не означает, что администрации школ должны кинуться нанимать людей и тем самым повысить стоимость образования. Но это говорит о том, что серьезные реформы неосуществимы без инвестиций в трудовые ресурсы.

Стоит отметить один интересный феномен, который делает задачу обновления кадровых ресурсов несколько менее обременительной, — энтузиасты-программисты и мейкеры из сообщества преподавателей и общества в целом, которые готовы оказать содействие и выступить в роли волонтеров. Оказывается, не только дети могут заразиться энтузиазмом или получать удовольствие от программирования. Инициативы под руководством энтузиастов — это не только индивидуальный или личный вклад, но и крупные филантропические организации, такие как code.org, #Yeswecode, Girls Who Code и другие организации с международным участием. Многонациональные технологические компании также поддерживают преподавание программирования в государственных школах в большинстве стран из соображений альтруизма или личного интереса.

Условия деятельности

Пример STEAM-студий в округе Сауз Файет также является показательным с точки зрения попытки представить вычислительное мышление, интегрировав его в учебную программу. Другие округи также инвестировали в лаборатории, студии или пространства для мейкеров. Округ Сауз Файет особенно интересен тем, что на его территории расположен один учебный кампус, и его можно считать отдельным миром. Под среднюю школу отведены три этажа, по одному для 3, 4 и 5 классов. В центре каждого этажа находится STEAM-студия с классными комнатами по обе стороны — «схема расположения, которая отражает философию трансформации всего округа» [35]. Обстановка, стимулирующая общение за столиком и способствующая совместной работе, не менее важна, чем используемый в классе контент [12]. Во многих компьютерных классах стоят настольные компьютеры, которые нельзя произвольно перемещать с места на место, однако компьютерные классы должны представлять собой максимально гибкие пространства. Важными факторами при изучении программирования являются взаимодействие учеников друг с другом, парное программирование и взаимодействие с учителем. Поэтому ноутбуки / планшеты в классе больше подходят для создания трансформируемой среды обучения. С ними учитель сможет более мобильно подходить к занятиям с гаджетами и без и легче настраивать класс на запланированную работу [29].

Как и в случае с кадровыми ресурсами, важно не просто инвестировать средства в здания, а делать это с учетом конкретных аспектов использования, например, для улучшения групповой работы. Когда единственными местами общего пользования в школах были спортивные залы для занятия физкультурой, библиотеки для чтения книг и периодических изданий и кафе для организации школьного питания, каждой школе вполне хватало одного такого помещения. Од-

нако для того, чтобы идея программирования как нового вида грамотности и компьютерного мышления, интегрированного в учебную программу, стала реальностью, ученикам и учителям потребуется более частый и непосредственный доступ. Ресурсная комната на каждом этаже может оказаться очень полезной для этих целей. Предусмотрено множество реформ для коллективного и совместного обучения, но, когда речь идет о программировании и вычислительном мышлении, полноценное традиционное преподавание, при котором учитель стоит перед сидящими за партами детьми, становится практически невозможным. Поскольку редко какая школа может себе позволить полностью изменить материально-техническую базу, власти могут обратиться к развивающейся концепции смены целевого назначения объектов путем их реконструкции для обеспечения устойчивого развития [65].

Оборудование

Чтобы увлечь детей программированием и создать условия, в которых пробудится их энтузиазм для работы с кодом, нужны качественные обучающие материалы. Учебные роботы — один из наиболее интересных инструментов, поскольку они мгновенно реагируют на действия детей. Несмотря на то, что учебные роботы стремительно дешевеют, большинство моделей все еще слишком дороги для школ. На момент завершения работы над этой книгой во второй половине 2018 года на рынке еще не появились недорогие роботы с богатым функционалом, но стоит отметить некоторые интересные инновации.

Одной из таких инноваций является аппаратное обеспечение с открытой спецификацией BBC micro:bit, которое претендует на звание самого дешевого компьютера в мире (некоторые версии доступны по цене от 15 долларов США). Micro Bit — это компактная плата размером примерно с половину кредитной карты с 25 светодиодами, которые можно запрограммировать на отображение букв, цифр и других форм, и чипом Bluetooth для беспроводного подключения. Для своего размера и стоимости он обладает огромным диапазоном возможностей. micro:bit разработан для того, чтобы вдохновить молодых людей на написание кода и креативное использование цифровых технологий. В 2016 году в рамках инициативы Make it Digital Британской вещательной корпорации каждый ученик 7 класса по всей Великобритании получил micro:bit бесплатно. Благодаря поддержке Британской вещательной корпорации была создана некоммерческая организация micro:bit Educational Foundation, чтобы BBC micro:bit и вспомогательные ресурсы можно было купить во всем мире.

2.5. Хвост птицы: оценивание знаний учащихся в сфере вычислительного мышления

«В какой-то момент учащиеся должны начать создавать собственные стимулы и использовать свои таланты и знания, чтобы влиять на решения и задачи, которые составляют их жизнь. Нельзя просто сидеть и ждать, что учитель (политик, продавец или религиозный лидер) скажет, «правильный» их ответ или нет. Эффективная оценка дает учащимся возможность задавать вопросы для размышления и рассматривать разные стратегии обучения и действия. Со временем учащиеся настолько продвигаются в своем обучении, что могут использовать свои знания для конструирования смысла, приобретают навыки самомониторинга, благодаря которым они осознают, что чего-то не понимают, и находят способы решить, что делать дальше».

- Лорна Эрл «Оценивание как обучение» [66]

Уникальный метод постоянного обучения на практике порождает два своеобразных метода оценивания вычислительного мышления: оценочное мнение, генерируемое машиной, и понятие индивидуальной траектории обучения. Поскольку традиционные дисциплины переходят в цифровую среду и начинают использовать программирование, возможно со временем эти

методы станут частью любой оценки, но на данный момент этот процесс особенно активно проходит в сфере изучения программирования. Решение реальных задач, как правило, отличается от традиционных методов преподавания и обучения — в реальной жизни задачи не появляются в известном формате или по заданной схеме, и самое главное, у них может быть несколько решений [67]. В этом разделе мы поделимся особыми соображениями в отношении оценивания вычислительного мышления, представим два примера используемых инструментов и опишем тенденции их развития. Мы также кратко рассмотрим, как вычислительное мышление способствовало улучшению когнитивных и некогнитивных навыков, таких как работа в команде и взаимодействие.

2.5.1. Ключевые соображения относительно оценивания вычислительного мышления

Подтверждая инертность, характерную для систем образования, многие страны, внедрившие программирование в учебные программы, оценивают навыки программирования учащихся с помощью тестов с вопросами с несколькими вариантами ответов. Однако между вычислительным мышлением и другими предметами существуют основные или принципиальные различия. Лорна Эрл разработала простую и очень полезную классификацию для оценивания — она описывает традиционную оценку в форме экзаменов как «оценивание обучения» — ученик должен что-то выучить, а экзамен представляет собой форму проверки освоения материала. Сейчас популярной становится идея формирующего оценивания, когда оценивают не прошлые успехи ученика, а высказывают мнение о работе ученика, включая устные или письменные комментарии, чтобы помочь ему исправить ошибки и усовершенствовать навыки, — Эрл называет это «оцениванием для обучения». Существует также такой подход (и он идеально подходит для вычислительного мышления), как «оценивание как обучение», когда ученик сам ставит перед собой цели и имеет средства понять, насколько он продвинулся в их достижении и что ему делать дальше.

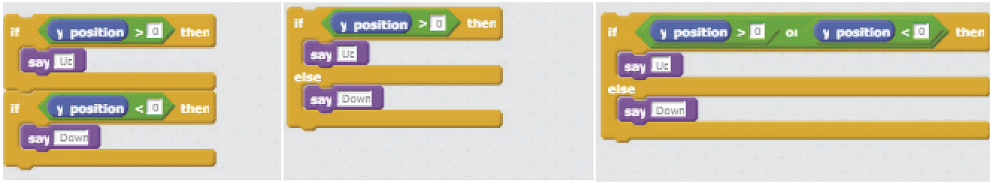
Когда речь заходит об оценивании компьютерного мышления, кардинально меняется роль учителя и принцип использование вопросов. Программа либо работает, либо нет — обратная связь приходит мгновенно, и ученик знает, что нужно попробовать другой способ, затем еще один, пока цель не будет достигнута. Не исключено, что с учетом установленных ограничений задачу вообще невозможно решить, но ученик сам должен это понять. В этом случае ученик сам задает вопросы, сам осознает, какого навыка ему не хватает и какой навык нужно освоить для решения задачи. Не нужно тратить время на то, чтобы задавать ученику вопрос, на который учитель знает ответ, но хочет понять, знает ли ответ ученик. Наоборот, именно ученик почти всегда задает вопрос! Роль учителя заключается не в том, чтобы дать ответ, а в том, чтобы помочь ученику определить оптимальный процесс мышления и работы. Иногда учитель может не знать ответ, что совершенно нормально, если он знает, как помочь ученику найти ответ. В изучении программирования важен процесс отладки программы, важен систематический и последовательный подход к отсеиванию возможных ответов — это один из способов, при котором оценивание становится формой обучения.

2.5.2. Инструмент автоматического оценивания компьютерного мышления: Dr. Scratch

Dr. Scratch — это бесплатный аналитический онлайн-инструмент оценивания вычислительного мышления на основе контента Scratch-программы. Пользователю всего лишь нужно дать URL-адрес Scratch-программы на сайте http://www.drscratch.org/. Scratch доступен на нескольких языках, включая русский, который можно вставить в инструмент, а интерфейс инструмента, разработанный испанскими исследователями, доступен на испанском, португальском, итальянском и русском языке. Dr. Scratch анализирует Scratch-проекты по семи параметрам вычислительного мышления: контроль потока, представление данных, абстракция, интерактивное вза-

имодействие с пользователями, синхронизация, параллелизм и логика. Для каждого из этих семи параметров авторы определили критерий оценивания, который соответствует сложности программы вплоть до уровня профессионального владения, как показано в примере на рисунке 2.6 для параметра «логическое мышление».

Рисунок 2.6. Пример трех уровней развития параметра «логическое мышление» в Dr. Scratch



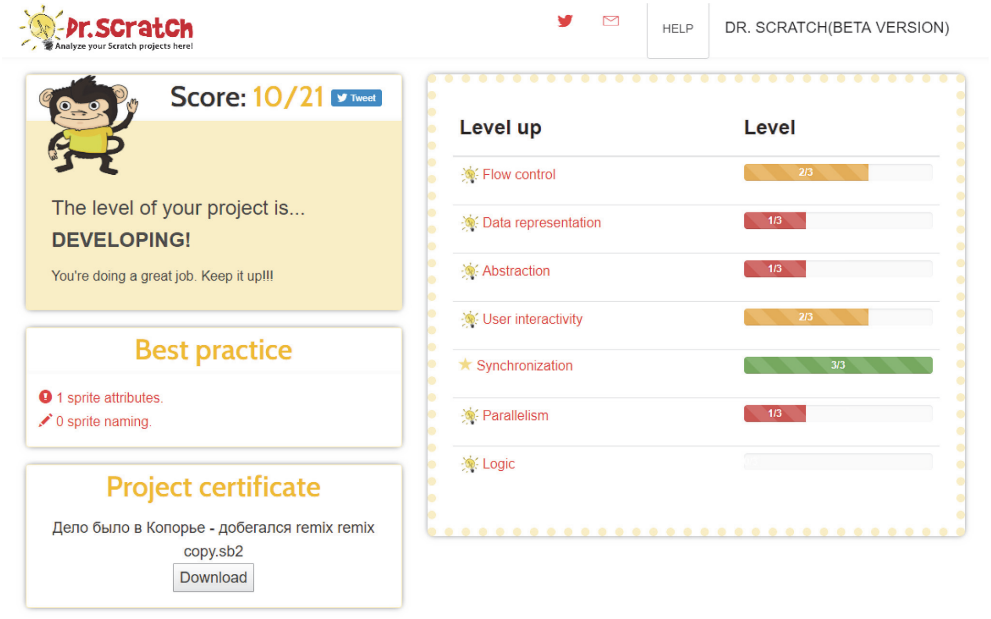
Базовый уровень: 1 балл
Использует блок if

Промежуточный уровень: 2 балла
Использует блок if else

Профессиональный уровень: 3 балла
Использует логические операции

Пользователь видит оценки в Dr. Scratch на панели, как показано на рисунке 2.7. Dr. Scratch был предметом ряда исследовательских проектов, авторы которых пытались установить его достоверность, надежность и полезность. Авторы одного из этих исследовательских проектов сравнивали оценки, полученные в Dr. Scratch, с показателями, используемыми для определения сложности программной системы. В ходе исследования было установлено положительное соотношение между Dr. Scratch и другими показателями, которое авторы интерпретировали как подтверждение достоверности Dr. Scratch (Moreno-León, Robles, and Román-González 2016). Некоторые исследователи предупреждают, что дети могут использовать конструкции, не понимая их с вычислительной точки зрения, и что такие инструменты, как Dr. Scratch, следует дополнять другими методами. Одним из таких методов является опрос с использованием артефактов, когда ученик отвечает на вопросы об использовании определенных артефактов в своем коде.

Рисунок 2.7. Скриншот результатов анализа Dr. Scratch



Важно определить цель оценивания. Простое составление табеля успеваемости по проекту не сильно отличается от проведения теста с вопросами и вариантами ответов для проверки знаний ученика. С точки зрения ученика, оценка может помочь понять, какие еще навыки нужно освоить, чтобы программа заработала. Исследователи проанализировали выборку Scratch-проектов и провели кластерный анализ оценочных данных. Они определили три кластера возрастающей сложности: кластер 1 — анимационные, художественные и музыкальные проекты; кластер 2 — истории; кластер 3 — игры [68]. Исследователи предлагают разработать траекторию обучения на основе анализа предыдущих проектов. На панели будут представлены рекомендации по другим полезным проектам в зависимости от достигнутого уровня и типа проекта. Общий принцип изучения такого ПО, как Scratch, называется «используй-меняй-создавай» — имея доступ к общему исходному коду, пользователь может поэкспериментировать с проектом, чтобы понять, как он работает, а затем поменять часть проекта для создания различных эффектов и создать новый проект, используя навыки, полученные в рамках предыдущих двух этапов. В классе учитель на своей панели может видеть сводные данные всей группы.

Автоматическое оценивание можно дополнить устной оценкой или опросом с использованием артефактов. Исследование, в котором приняли участие учителя из Великобритании, также показало, что учителя поддерживают идею устной оценки, потому что устное объяснение принципа решения задачи учеником не менее ценно, чем письменное. Объяснение логики, лежащей в основе программных проектов, также лучше подходит ученикам с хорошими способностями в сфере решения задач, которым трудно излагать свои рассуждения на бумаге [29].

Примеры вопросов для устной оценки:

- Объясните, что представляет собой выбранный проект.
- Опишите, как можно усовершенствовать проект.
- Исправьте ошибку в проекте.
- Создайте ремикс проекта, добавив новую функцию.

2.5.3. Перспективное направление оценивания: зона потока для AgentSheets

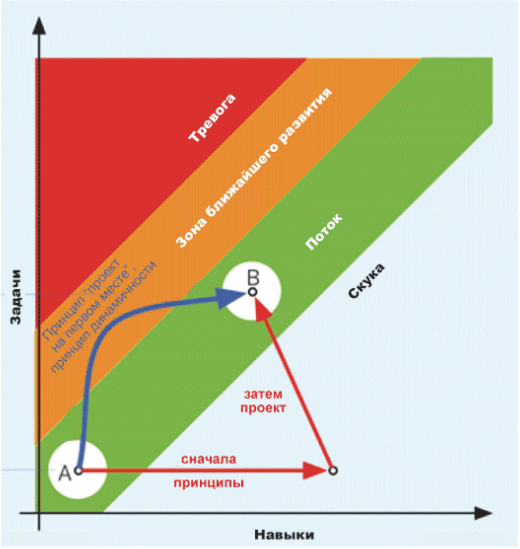
AgentSheets, разработанный в Университете Колорадо в Боулдере, — это еще один инструмент для детей, изучающих программирование с помощью развивающих игр, направленный на внедрение такого изучения в основные научные концепции: концепции, которые дети изучают для создания игр, можно затем применить к моделированию научных принципов. AgentSheets доступен в Интернете на сайте http://www.agentsheets.com/. Он разработан по принципу низкого пола и высокого потолка, то есть с ним легко начать создавать свои первые игры (обучение займет всего несколько часов), можно переходить к более сложным проектам и применять вычислительное мышление.

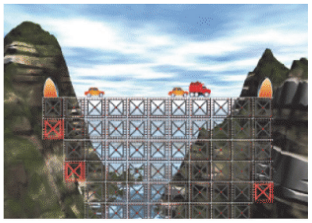
Критерии оценивания и методологию, разработанную для AgentSheets, можно использовать в качестве способа или перспективного направления оценки другого программного обеспечения, которое дети используют для освоения вычислительного мышления [69,70]. Этот подход объединяет три понятия: (i) зона ближайшего потока; (ii) схожесть поведения программ (СПП); и (iii) анализ паттернов вычислительного мышления (АПВМ). Все три понятия описаны ниже.

Зона ближайшего потока

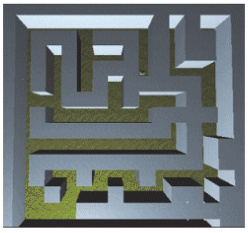
Понятие *зоны ближайшего развития*, введенное Львом Выготским и означающее развитие способностей ребенка через поддержку и руководство в процессе решения сложных задач, которые не являются невозможно трудными, было объединено с идеей Чиксентмихайи о *потоковом состоянии*, в котором пребывает человек, осуществляя деятельность с сильной моти-

Рисунок 2.8. Зона ближайшего потока: принцип «проект на первом месте» обеспечивает мотивацию для приобретения навыков





Проект “Мостостроитель”



Проект “Зомби-лабиринт”

Источник: Басавапатна и соавторы, 2013.

Источник: Сайт AgentsSheets.

вацией и надлежащим контролем. Эту концепцию проще понять с помощью диаграммы и описания особенностей изучения программирования посредством изучения принципов создания компьютерной игры.

Ребенок должен освоить навык, например, люди останавливаются возле стен, а зомби может пробить стену. Возникает вопрос: должен ли ребенок сначала усвоить ряд принципов, например, разные правила для разных персонажей, или ему будет проще решать проблемы по мере их возникновения при создании проекта и осваивать определенный навык, когда это необходимо. Исследователи утверждают, что правило «проект на первом месте» помогает сохранять мотивацию детей и развивать навыки, но в конечном итоге это вопрос опыта, и ответы на него могут зависеть от самого ребенка и проектов, которые он создает. Если задача слишком сложна для уровня навыков, которым уже обладает ребенок, он может выйти за пределы своей зоны ближайшего развития, оказаться в зоне тревоги и потерять интерес к работе. Если задания слишком просты, например, следования указаниям в руководстве для освоения навыка, ребенку будет скучно. Возможно ли, чтобы метод оценивания помогал ребенку оставаться в зеленой или оранжевой зоне, как показано на рисунке? Чтобы ответить на этот вопрос, нам нужно рассмотреть интересную концепцию измерения.

Схожесть поведения программ (СПП)

Измерение СПП — ответ на критику синтаксической оценки, используемой в Dr. Scratch, которая является самостоятельным или внутренним показателем сложности. Синтаксическая оценка не отслеживает поэтапные изменения качества программ, поскольку этот показатель не учитывает контекст или цель. Альтернатива была заимствована из другого направления под названием «Латентно-семантический анализ» (ЛСА), который используют для поиска семантической информации при сравнении естественноязыкового текста. Чаще всего ЛСА используют для определения плагиата — ни общие слова, ни общие выражения в двух текстах не являются явным признаком плагиата. Однако наличие больших фрагментов с одинаковым

значением может быть признаком копирования в двух рассматриваемых отрывках текста. Математически идентичные принципы, которые сравнивают два фрагмента естественноязыкового текста, можно развить до сравнения двух программ, что приведет к измерению схожести поведения программ.

Персонажи игры в AgentSheets используют комбинацию разных условий и разных действий (одна из таких комбинаций: если нажата клавиша стрелки «влево», персонаж перемещается влево на один шаг). Таким образом, каждую игру можно представить в виде вектора, причем каждый элемент вектора обозначает отдельную комбинацию условий и действий, используемую для реализации игры. Две игры u и v можно сравнить, изучив коэффициент Охаи их векторных представлений, как показано ниже. Если в играх присутствуют одинаковые элементы, коэффициент Охаи будет равен 1; если векторы ортогональны, например, у них нет общих правил, коэффициент Охаи будет равен нулю.

$$\text{СПП}\ (u, v) = \frac{\sum_{i=1}^{n} u_i v_i}{\sqrt{\sum_{i=1}^{n} {u_i}^2} \cdot \sqrt{\sum_{i=1}^{n} {v_i}^2}} \qquad (1)$$

Анализ паттернов вычислительного мышления (АПВМ)

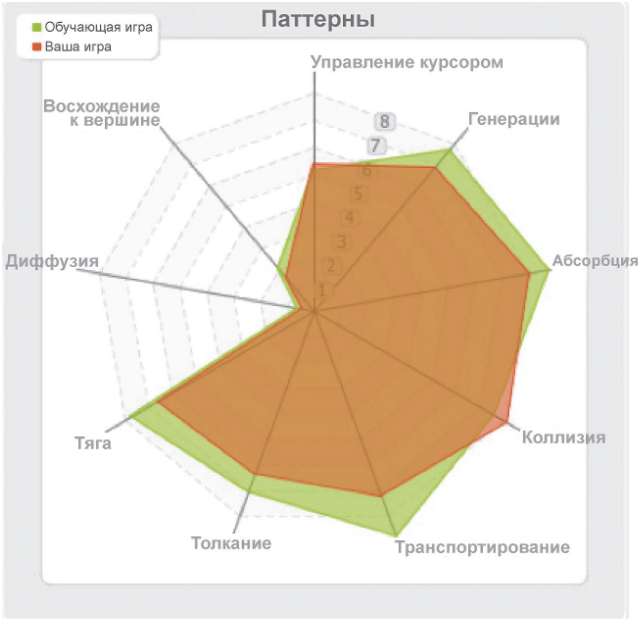
За многие годы разработки и использования AgentSheets исследователи смогли выделить девять канонических паттернов, относящихся к понятиям вычислительного мышления. Их называют переносимыми понятиями, имея в виду перенесение между играми и научными понятиями. В качестве примера можно привести «поиск восхождением к вершине» — общий алгоритм, используемый для оптимизации в различных контекстах. Агент, использующий алгоритм поиска восхождением к вершине, просматривает граничащие искомые значения и осуществляет переход к наибольшему значению. Такими значениями могут быть, например, «след» другого агента, который может быть полезен в зомби-игре, но этот принцип можно перенести в другие контексты оптимизации [71].

АПВМ — это применение уравнения СПП (1), приведенного выше, попарно к каждому из канонических паттернов вычислительного мышления, в результате которого можно обнаружить похожие фрагменты кода в оцениваемой программе. В результате получается вектор длиной m, в котором каждый m элемент представляет оценку канонического паттерна. В контексте обучения используется обучающая игра, которая дает представление об оптимальной комбинации канонических паттернов, и игра, рассматриваемая в сравнении с обучающей игрой. Каждая ось полярного графика, представленного ниже, символизирует канонические паттерны в порядке возрастания степени по часовой стрелке, начиная с управления курсором. Оценка по каждому параметру представляет собой 10-кратное значение СПП.

Представленный график легко перевести в числовые значения для определения расхождений между тестируемой игрой и обучающей в виде нормализованной суммы расхождений — корня отклонений в квадрате, разделенного на корень числа используемых параметров. Преимущество этого подхода состоит в том, что с его помощью можно обнаружить избыточный код или ненужную сложность. Усредненные значения по различным проектам будут составлять оценку навыков ученика.

В теории тестовых заданий — самой распространенной теории тестирования — сложность вопроса соотносят со способностями человека путем сравнения работы людей с разными способностями над вопросами теста разной сложности. Аналогичным образом можно рассчитать сложность игры по шкале, используемой для определения навыков людей в определенный момент времени. Басавапатна и соавторы предлагают использовать эти оценки для определения предполагаемой траектории обучения в зависимости от уровня освоенных навыков. Если

Рисунок 2.9. Диаграмма АПВМ из работы Ко и соавт., 2010



речь идет о курсе, состоящем из ряда задач, сложность каждой следующей задачи, которую должен решить ученик, может быть повышена ровно настолько, чтобы ученик оставался в зоне ближайшего развития или зонах потока, представленных на рисунке 2.8 [69].

На момент написания этой книги использование АПВМ для разработки и внедрения курсов по вычислительному мышлению находилось на стадии предложения, раскрывающего потенциал компьютеризированной оценки текущей работы учеников. Отпадает необходимость для ученика и учителя тратить время на отдельное тестирование, как это обычно происходит при изучении традиционных предметов и использовании систем оценки. Оценивание на основе АПВМ можно сочетать с соответствующим подходом в педагогике, когда учитель привлекает учащихся к анализу мгновенно полученных результатов их работы по программированию и помогает учащимся определить их траекторию обучения. Таким образом, обучение вычислительному мышлению через программирование помогает ученикам приобретать навыки обучения.

С самых первых дней обучения детей вычислительному мышлению одной из предполагаемых причин было утверждение, что освоение программирования помогает улучшить общие когнитивные навыки. Вполне логично, что занятие, которое помогает мыслить организованнее и системнее, сделает вас в целом умнее. Некоторым детям одного возраста легко дается абстрагирование; другим мыслить абстрактно может помочь программирование. Идею можно визуализировать на экране в виде цветного блока или написанных слов кода — манипуляции с блоком на экране могут помочь развить способность к мысленным манипуляциям, по крайней мере, в теории. Обратная связь и анализ, как в АПВМ, дают возможность глубинно задуматься о мышлении, что поможет развить когнитивные способности в процессе распределения способностей. Также можно предположить, что различные подходы к обучению программированию могут по-разному влиять на развитие когнитивных способностей — с помощью научно-исследовательских данных можно определить оптимальные элементы, которые усиливают благоприятное воздействие.

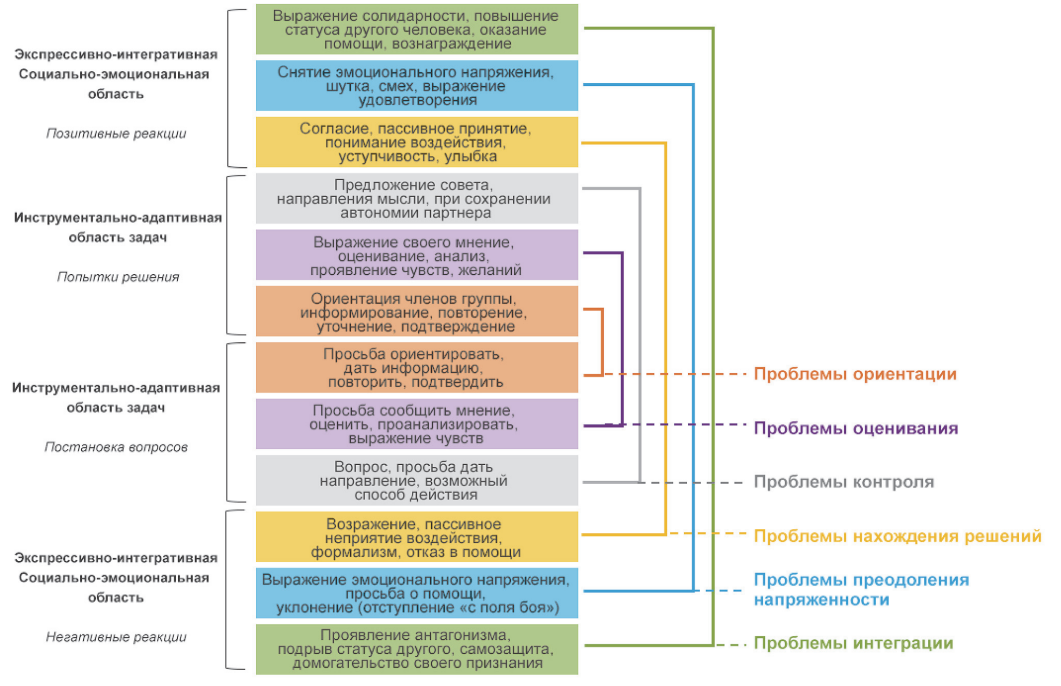
2.5.4. Оценка формирования взаимодействия и работы в команде

Интересное приложение для оценки вычислительного мышления представлено в исследовании трехлетней инициативы (2014–2016 гг.) в сфере обучения детей вычислительному мышлению под названием uCode@UWG в Университете Западной Джорджии в США [72]. Программа основана на модели CoderDojo — всемирном движении, которое поощряет сообщества создавать лагеря программирования свободного доступа или додзё. Целью uCode@UWG было «создание среды, в которой дети могли бы получить доступ к компьютерному программированию — в концептуальном и культурном плане — в веселой, неформальной обстановке при поддержке специалистов и внимательных волонтеров из местного сообщества в г. Карролтон, Джорджия». Программа организована при поддержке волонтеров. В данном случае в качестве наставников выступали специалисты в сфере компьютерных наук из университета и частного сектора. Новинкой стало рекомендуемое участие родителей в работе детей в возрасте от 7 до 17 лет.

Исследователи, которые придумали и реализовали uCode@UWG, решили использовать методологию под названием «анализ внутригруппового взаимодействия» — удивительно актуальный протокол наблюдения, разработанный для изучения небольших групп задолго до того, как эти группы начнут изучать программирование [73]. Категории, предложенные Бейлсом, учитывают двусторонний или парный характер группового взаимодействия, например, вопросы и ответы на них, согласие или несогласие с предложенным планом действий и т.д. Бейлс сделал главное — он разделил эти пары на разные значимые группы.

Как показано на рисунке 2.10, 12 категорий взаимодействия разделены на шесть пар, читать которые следует по направлению от центра к краю. Первая пара охватывает проблемы ориентации — в группе, особенно в начале занятия, будут индивиды, которых необходимо ориентировать — для обеспечения эффективного общения предусмотрены повторения и подтверждения.

Рисунок 2.10. Анализ внутригруппового взаимодействия, методика Роберта Бейлса



Соответственно, найдется человек или люди, которые обеспечат необходимую ориентацию, которые будут повторять, уточнять и подтверждать. Классическим примером такого поведения является кабина пилота в самолете, в которой пилот и второй пилот проходятся по списку, чтобы убедиться, что все процедуры, необходимые для безопасного полета, выполнены [74]. Высокоэффективная групповая работа, вероятно, будет соответствовать одному из двух паттернов — будет присутствовать симметричная коммуникация, то есть запросы на ориентацию одних членов группы будут удовлетворены другими членами; или группа проработала долгое время, и ориентация не требуется. Признаками нефункциональной группы являются неудовлетворенные запросы на ориентацию. Если ориентацию обеспечивают в отсутствие запроса на нее, это уже не столько группа программирования, сколько традиционный класс, в котором учитель обеспечивает необходимую ориентацию.

На рисунке указаны еще пять пар — *проблемы оценивания, проблемы контроля* (которые зачастую включают сложные аспекты представления предложений и запроса предложений), *проблемы нахождения решений* (если группа должна принимать совместные решения, есть ли индивид, который всегда со всем согласен или всегда со всем не согласен, или налажен ли процесс слушания и выражения), *проблемы преодоления напряженности* (например, снятие напряженности с помощью шуток или смеха) и *проблемы интеграции*, относящиеся к групповой солидарности или антагонизму в противоположность солидарности. Бейлс также разделил 12 видов поведения, которые могут быть вербальными или невербальными, на четыре группы. Средние три пары в совокупности называются инструментально-адаптивной областью задач, они относятся к вопросам и ответам, которые способствуют выполнению работы группы. Три пары по краям называются экспрессивно-интегративной социально-эмоциональной областью, включающей позитивные и негативные реакции, связанные с чувствами членов группы и с тем, как им разрешено или запрещено выражать свои чувства.

В процессе реализации исследователи ucode@UWG случайным образом выбрали группу участников и провели незаметные наблюдения за каждым индивидом в течение 5-10 минут, а также внесли данные о проявлении видов поведения в таблицу, представленную на рисунке 2.10b. В таблице используется код для указания пола каждого человека (F – женщина или M – мужчина) и возраст в годах, потому что исследователи хотели обратить внимание на эти две переменные. Итоговые данные по строкам указывают на проявление каждого вида поведения у 17 наблюдаемых учащихся. В исследовании не приводятся экспериментальные данные, свидетельствующие о том, что участие в программировании способно ускорить работу в команде, совместную работу и приобретение лидерских навыков. Метод, основанный на категориях Бейлса, вселяет надежду — данные, представленные на рисунке 2.10b, можно подвергнуть анализу на основе модели Раша и получить баллы по каждому индивиду [75] Развитие технологий цифрового видео и носимой электроники может помочь автоматизировать процесс сбора данных. Оценивать можно будет не только программирование, но и обратную связь в отношении навыков работы в команде и взаимодействия.

Рисунок 2.11. Данные анализа внутригруппового взаимодействия Бейлса от ucode@UWG [76]

Код участника	Сентябрь 2015					Октябрь 2015					Количество проявлений
	F11	M9	M12	M7	F12	M15	F11	F9	F12	M13	
Выражение солидарности			1	2	2		1	2	1	3	12
Снятие эмоционального напряжения	4	2	2	2	1		2	3	1	1	18
Согласие	6	3	3	1	1		1	1		2	18
Предложение совета	1			2			1			2	6
Выражение мнения		2	1	3	1		1				8
Ориентация членов	2							2			4
Просьба ориентировать	1	2				1					4
Просьба сообщить мнение			1			1			2		4
Вопрос											0
Возражение						1					1
Выражение эмоционального напряжения											0
Проявление антагонизма											0

Выводы главы 2. Пять вариантов реализации

- **Первая голова: кодирование в рамках учебной программы**. Такой подход можно считать традиционным. Страны пробовали разные подходы: преподавать кодирование в рамках отдельного предмета, включить кодирование в традиционные предметы и включить кодирование в рамках междисциплинарной стратегии. Тенденция такова, что кодирование начинают преподавать в начальной школе и переходят к более углубленному преподаванию в средней школе. Некоторые страны, например, Великобритания, ввели радикальные изменения — ранее существовавшие системы, ориентированные на приобретение общих навыков работы с компьютером, теперь делают упор на кодирование, вычислительное мышление и вычислительные науки.

- **Вторая голова: кодирование в рамках внешкольного образования**. Для преподавания кодирования нужны механизмы подачи материала, кардинально отличающиеся от традиционных предметов. Кодирование нельзя преподавать в виде передачи знаний на уроках или передачи навыков через демонстрацию и копирование. Внеклассные мероприятия уже основаны на объединении интересов учащихся, включении учащихся в сообщество и создании артефактов — на элементах, необходимых для изучения кодирования.

- **Левое крыло: повышение профессиональной квалификации преподавателей**. Рассуждая о преподавании кодирования, важно понимать, что линейная модель — разработка учебной программы, разработка соответствующего плана для освоения этой программы, затем подача этой программы учителем — уже не применима. Теперь задача состоит в том, чтобы изменить мировоззрение учителя в сторону модели обучения, при которой учитель не всегда должен быть самым умным в классе, а повышать квалификацию можно через онлайн-сети или очно.

- **Правое крыло: контент для обучения программированию**. Спрос на контент огромен и динамичен, и у любого регионального министра образования или национального управления, как правило, есть ряд специалистов, продвигающих новейшие продукты. Стоит обратить внимание на деятельность в рамках платформ. Рынок то-

варов будет рассмотрен более подробно в следующей главе. Здесь мы отмечаем, что платформы должны уделять особое внимание инклюзивному подходу для учета гендерных аспектов и потребностей учащихся с ограниченными возможностями, и описываем интересные разработки в этой области.

- **Хвост: оценивание знаний учащихся в сфере вычислительного мышления**. Поскольку программирование подразумевает мгновенную обратную связь, цель оценивания для определения знаний и навыков учащихся меняется коренным образом. Тенденцией в этой области является автоматическая оценка в процессе, пока учащийся разрабатывает свой проект. Эта тенденция еще не укоренилась, но очень интересные возможности открываются в рамках анализа паттернов вычислительного мышления (АПВМ) и создания систем, которые будут стимулировать учащихся двигаться по оптимальной траектории развития.

3. МНОГООБРАЗИЕ ПРЕДЛОЖЕНИЙ НА РЫНКЕ

В последние годы значительно выросло число инструментов программирования для детей, и на рынке постоянно появляются новые продукты. Ввиду такого многообразия инструментов с разными функциями учителям, родителям и самим детям было бы полезно выбрать инструменты, подходящие для их конкретных целей. Чтобы представить читателям информацию о доступных инструментах, в этой главе мы составили список из примерно девяноста инструментов, предназначенных для совершенствования навыков программирования / вычислительного мышления у детей. Двадцать из этих инструментов были изначально разработаны для русскоязычной аудитории, в основном (но не только), из Российской Федерации.

3.1. Влияние программирования на позитивное развитие молодежи

Профессор Марина Умащи Берс из Университета Тафтса разработала теоретическую концепцию взаимосвязи между технологиями и развитием навыков и ценностей [77]. Согласно этой концепции под названием «Позитивное развитие технологий» (ПРТ), глубинное влияние такой деятельности, как программирование, выходит за рамки развития вычислительного мышления и других навыков. В ходе подробного исследования этой темы было получено эмпирическое обоснование концепции. Был разработан инструментарий, позволяющий применять эту концепции в различных технологических контекстах. Инструментарий представлен в Приложении 2a (вопросник для наблюдения за действиями детей) и Приложении 2b (вопросник для наблюдения за техническими средствами и учителем или помощником). Эта концепция подтверждает убеждение о том, что любой продукт или инструмент невозможно осмысленно оценить без соответствующего контекста — от способа использования инструмента зависит его эффективность. Для инструментов должны быть предусмотрены особые возможности, например, упрощение создания ремиксов тесно связано с возможностями развития совместной работы, разработка игр привлекает к игре [78].

Берс полагает, что предыдущие обсуждения вопросов молодежи и технологий были построены непродуктивно как реакция на негативные тенденции — молодежь в группе риска и защита молодежи от негативных аспектов технологий, таких как ненадлежащее использование соцсетей и случаи патологической зависимости от компьютерных игр. Вместо этого она предлагает сосредоточиться на положительных моментах и исходить из того, что молодежь уже часто использует компьютеры разными способами, как в школе, так и за ее пределами. Первоначально концепция ПРТ разрабатывалась с прицелом на молодежь или детей старшего возраста, однако позднее в нее стали включать и детей.

3.1.1. Концепция позитивного развития технологий

«Конечная цель образовательных программ, разработанных в рамках концепции ПРТ, состоит не только в том, чтобы научить детей писать код или освоить вычислительное мышление, но и привить им позитивные модели поведения. Концепция ПРТ позволяет реализовать видение программирования как грамотности, которая расширяет возможности человека».

- Марина Умащи Берс [12]

Берс предложила модель шести компонентов, в которой технологии обладают особым функционалом, формирующим у детей модели поведения, способствующие развитию личностных ресурсов. Личностные ресурсы также можно разделить на шесть компонентов, которые отражают основные характеристики здорового и позитивного развития молодежи, являющего

залогом процветания и укрепления сообществ. Шесть ресурсов и соответствующие модели поведения, формируемые технологическими решениями, представлены в таблице 3.1. Есть два набора ресурсов / моделей поведения: первый набор относится к внутриличностному поведению и внутриличностным ресурсам, второй — к межличностным. Три пары моделей поведения / ресурсов внутриличностной сферы — это создание контента / компетенция; креативность / уверенность и поведение / моральная устойчивость. Соответствующие пары в межличностной сфере — это коммуникация / связь; сотрудничество / забота и создание сообщества / вклад. Пояснения для каждой из этих пар приводятся по очереди.

3.1.2. Создание контента и компетенция

«Иерархия и абстрагирование относятся к эстетике «планировщика» и высоко ценятся структурными программистами, при этом программисты-бриколеры, как ученые-бриколеры Леви-Стросса, предпочитают согласование и перераспределение материалов. Бриколер чем-то напоминает художника, который, сделав мазок, отходит от холста, смотрит на него со стороны и, только поразмыслив, решает, что делать дальше. Бриколеры используют искусство ассоциаций и взаимодействий. Для планировщиков ошибки — это промахи. Бриколеры используют навигацию промежуточных корректировок. Для планировщиков программа — это инструмент преднамеренного контроля. Бриколеры ставят перед собой цели, но стремятся достичь их в духе совместного начинания с машиной. Для планировщиков наладить работу программы значит «сказать свое слово». Бриколеры воспринимают этот процесс скорее как беседу, а не монолог».

Ш. Теркл, С. Пейперт- [10]

Инструменты, направленные на развитие навыков программирования и вычислительного мышления у детей, подразумевают создание контента разными способами. Участие детей в создании контента и освоении процесса разработки и программирования повышает уровень их компетенции в сфере компьютерной и технологической грамотности [81]. Кроме того, эффективные инструменты позволяют «самоделкиным» работать с материалами и придумывать идеи для создания, разработки и программирования, что также способствует повышению уровня компетенции детей.

Такую «самодеятельность» определяют как эксперименты пользователей с фрагментами кода и командами, напоминающие возню с механическими или электронными деталями [82]. В этом отношении возня не обязательно должна быть связана с созданием физических объектов. Это скорее подход к созданию вещей, неважно, физически существующих или виртуальных. Ребенок может экспериментировать, когда сочиняет историю или пишет код. Ключевой идеей является дух экспериментов и взаимодействия [6].

Доступные инструменты для детей доказывают, что есть несколько способов создания контента, от сочинения историй до создания художественных работ и мобильных приложений. Один инструмент сам по себе может и не создаст простор для творческих возможностей, однако важно, чтобы в ходе создания контента дети получили то, что Пейперт называл «объекты, помогающие думать».

3.1.3. Креативность и уверенность

Для работы с инструментами программирования и развития вычислительного мышления детям необходимы навыки решения задач. В процессе решения креативно сформулированных задач дети развивают уверенность в своих способностях к обучению.

Таблица 3.1. Позитивное развитие технологий

	Формируемые ресурсы	Определение	Поддерживаемые модели поведения	Определение
Внутрилич-ностная сфера (вклад в себя)	Компетенция	Способность использовать технологии, создавать или разрабатывать проекты с использованием компьютера для достижения цели, а также для отладки проектов и решения задач.	Создание контента	Создание личностно значимых проектов; молодежь становится производителем, а не потребителем; молодежь учится участвовать в онлайн-культуре.
	Уверенность	Ощущение себя как человека, способного действовать и учиться добиваться успеха в техноемкой среде, находить помощь в случае необходимости и упорно преодолевать технические трудности.	Креативность	Повышение уверенности в себе через опыт, способствующий творческому выражению, обмену и размышлению.
	Моральная устойчивость	Осознание и уважение неприкосновенности личности, моральных и социальных ценностей при ответственном использовании технологий и умение выразить себя через технологии.	Выбор линии поведения	Ограниченная площадка, где молодежь мо-жет делать выбор, рисковать, сталкиваться с последствиями и размышлять; экспериментировать с морально-этическими аспектами и нормами сообщества.
Межличност-ная сфера (вклад в других)	Связь	Технологии помогают установить и поддерживать позитивные связи и отношения.	Коммуникация	Синхронная и асинхронная коммуникация через мультимедиа (текст, голос, звук, видео и т.д.); развитие языка и грамотности; развитие связей между молодежью и взрослыми.
	Забота	Чувство сострадания и готовность удовлетворять потребности и решать проблемы других людей, помогать людям, столкнувшимся с техническими трудностями и использовать технологии в качестве средства оказания помощи другим.	Сотрудничество	Общие задачи требуют, чтобы молодые люди зависели и реагировали друг на друга; опыт включает техническую и социальную поддержку со стороны сверстников и взрослых; молодежь использует технологии для оказания помощи другим.
	Вклад	Ориентированность на вклад в общество, используя и предлагая технологии для реше-ния общественных / социальных проблем	Создание сообщества	Общее чувство ответ-ственности сообщества; механизмы вклада в общее благо; демократи-ческоеучастие.

Источник: [79,80]

Однако дети, особенно те, кто только начал осваивать технологические инструменты, могут столкнуться с трудностями в попытке добиться интеллектуальных или креативных результатов. Ученики, толком не знакомые с программированием, не имеют четкого представления о том, что они хотят создать, что можно создать или насколько сложными могут быть те или иные проекты. Эта ситуация может стать разочарованием для ребенка. Один из создателей Scratch Митчел Резник тоже признает эту проблему: «Мы знаем, что некоторые новички теряются, увидев множество разнообразных инструментов и проектов на сайте Scratch. А значит, нужно несколько упорядочить первые шаги в работе и дать начинающим кое-какие исходные ориентиры. Мы стараемся, чтобы такой наш подход ни в коей мере не ущемлял свободу творчества и чтобы любой новичок, придя к нам на сайт, мог заниматься проектами, которые его больше всего увлекают и соответствуют его индивидуальности» [6, p. 82].

Кроме проблем, связанных с работой над первым проектом, есть еще один момент: если ребенок не может связать задачу в области программирования с реальной задачей из своей жизни, он делает вывод, что программирование — это путь социальной изоляции [83]. Поэтому Лай и Ко предлагают давать детям понятные задачи из жизни, которые они будут решать средствами программирования. Такие задачи могут включать разработку игровой стратегии, игр или цифровых историй [84].

Looking Glass, разработка Университета Вашингтона в Сент-Луисе, — это хороший пример платформы, помогающей детям создавать идеи при помощи «шаблонов сообществ». Эти шаблоны помогают ребенку начать свою историю и постепенно раскрывают его творческий потенциал. Онлайн-сообщества также могут помочь в формировании идей, показывая проекты пользователей [83].

Помимо Looking Glass, Scratch планирует решить проблему формирования идей, создав подборку *микромиров по интересам*, где будет представлена упрощенная версия языка Scratch с ограниченным набором средств программирования. Например, один микромир предназначен для создания интерактивных игр, скажем, футбола. После того, как дети освоятся в Scratch с помощью упрощенных микромиров, они смогут переносить свои проекты в полноценную Scratch-среду, благодаря чему процесс совершенствования их навыков не будет прерван [6].

3.1.4. Выбор линии поведения и моральная устойчивость

В технологической среде моральная устойчивость определяется как «осознание и уважение неприкосновенности личности, моральных и социальных ценностей при ответственном использовании технологий и умение выразить себя через технологии». Аспект моральной устойчивости в ПРТ подразумевает избегание моделей поведения, которые могут негативно повлиять на других пользователей. Программирование, как и детская площадка, дает возможность детям понять сложность нравственного выбора и столкнуться с последствиями своих действий. Процесс выбора формирует моральный облик ребенка [79].

Например, компания *Wonder Workshop*, занимающаяся разработкой учебно-образовательных технологий, выпустила школьные наборы *Dash & Dot* — обучающие детские комплекты для занятий робототехникой. Для классов численностью более 30 учеников в комплект входят всего 12 роботов. В ситуациях, когда не каждый ребенок получит предварительно собранный комплект, дети учатся играть с роботом по очереди, помогать другим в игре и договариваться, чтобы получить желаемое. Использование ограниченных ресурсов в сообществе помогает детям выработать внутренние качества и чувствовать ответственность за свои действия [85].

Интересный пример формирования морального облика можно увидеть на уроке робототехники для учеников 4 класса в Республике Корея [33]. Ученики искали решение реальных проблем из жизни, например, помощь при стихийных бедствиях, строительство сейсмостойких домов,

утилизация отходов и решение проблемы нехватки продовольствия в будущем. В качестве примера также можно привести Сингапур, где ученики собирали данные о качестве воды в местных прудах при помощи спроектированных и созданных подводных роботов Sea Perch, затем учились анализировать данные, собранные роботами http://seaperch.mit.edu/index.php [86].

3.1.5. Коммуникация и связь

Коммуникация помогает детям обмениваться идеями, выражать свои мысли и укреплять связи между сверстниками или между детьми и взрослыми. Коммуникация может принимать разные формы:

- Активная работа с совместными проектами в онлайн-сообществах. Во многих инструментах, описанных в этом разделе, как только ребенок завершает проект, он может поделиться им с онлайн-сообществом и взаимодействовать с другими проектами через комментарии, лайки, добавление в избранное и оценки. Примером может служить активное сообщество игровой платформы *Gamestar Mechanic*, созданной для обучения детей программированию и принципам гейм-дизайна в творческой среде.

- Традиционные текстовые форумы. Традиционные текстовые форумы позволяют детям задавать вопросы, делиться идеями и получать комментарии или ответы от других членов сообщества. Например, один из самых активных текстовых форумов блочного конструктора 2D-игр Stencyl. Его раздел «Задать вопрос» насчитывает десятки тысяч постов.

- Группы соцсетей. Некоторые платформы дорабатывают традиционные текстовые форумы и создают каналы соцсетей для обмена информацией с пользователями, где пользователи могут общаться друг с другом. Такой подход использует бесплатная онлайн-платформа обучения программированию на языке С в России *Young Coder*. Пользователи могут общаться друг с другом и разработчиком платформы в группе соцсети ВКонтакте.

Коммуникация и связь очень важны, когда речь идет о разработке программного обеспечения взрослыми, и это один из аспектов ПРТ, где учитель или помощник может быть очень полезен. Сейчас стало модно использовать фразу «учиться учиться», но как претворить эту идею в жизнь? Один из способов — научиться задавать вопросы так, чтобы максимально увеличить вероятность получения помощи. Умение отвечать на вопросы явно выходит за рамки простого технического решения проблемы с кодом. Некоторые веб-сайты, такие как Stack Exchange, просят автора вопроса привести «самый простой пример», чтобы потенциальный помощник мог представить проблему и помочь. Сами помощники, как правило, являются частью саморегулируемой системы создания репутации — недостаточно знать возможный ответ на поставленный вопрос, важен также способ подачи ответа. Те пользователи, чьи ответы понятны, также должны оценить другие ответы. Специалист может укрепить свою репутацию полным объяснением более простых, быстрых или лучших решений.

3.1.6. Сотрудничество и забота

«Использование ремиксов стало одним из главных механизмов распространения идей в Scratch-сообществе... Как объяснил один скретчер: «Классная это штука, возьми любой проект, и обязательно чему-нибудь научишься».

- Митчел Резник [6]

Сотрудничество подразумевает получение или оказание помощи в рамках проекта, совместное программирование или совместную работу над общей задачей. В классе сотрудничество можно организовать с помощью парного программирования, когда двое детей работают за одним компьютером. «Пилот» пишет код, а «штурман» проверяет и дает рекомендации по ходу

написания кода. Дети периодически меняются ролями. Парное программирование способствует сотрудничеству и обучению, поскольку ученики находят лучшие решения сообща и менее склонны в отчаянии бросать программирование [2].

В онлайн-среде основным видом сотрудничества являются ремиксы. Пользователи создают новый проект на основании существующих проектов, которые другие пользователи ранее опубликовали в сообществе. Например, более 27% всех Scratch-проектов представляют собой ремиксы других проектов, и пользователи уже выложили более 27,7 миллионов проектов в Scratch-сообществе Community [87]. Сообщество, использующее ремиксы для обучения программированию, опирается на принципы теории конструкционизма, согласно которой мы учимся быстрее, проектируя и создавая «объекты», и это обучение становится еще более значимым и эффективным в социальном контексте [88].

Принципы теории конструкционизма получили развитие в Scratch-среде и могут быть переформулированы следующим образом: обучение происходит наиболее эффективно, если учащийся занимается созданием продукта деятельности, который другие участники могут обсудить, оценить и использовать для разработки новых объектов [89]. Чтобы участие в предметно-ориентированном сотрудничестве в сетевой среде было эффективным, участник должен иметь возможность оценивать и обсуждать объекты, созданные другими участниками, принимать оценки и суждения других участников, наблюдать за их действиями, классифицировать объекты, созданные другими участниками.

Помимо обучения и укрепления связей, у ремиксов есть еще одно важное преимущество: мы хотим, чтобы дети продолжали изучать более сложные принципы программирования. Однако многие дети быстро достигают определенного уровня и останавливаются на простых принципах. Такие пользователи, как правило, создают простые последовательные программы, содержащие множество повторяющихся команд. Немногие пользователи сами узнают, как использовать другие концепции программирования для создания впечатляющих и интересных анимаций. На этом этапе в игру вступают ремиксы: большинство пользователей могут самостоятельно использовать более сложные принципы программирования в своих программах, создав ремикс из программ, содержащих аналогичные конструкции [90].

Ремиксы также ставят под сомнение общепринятую идею о том, что ученики должны сами делать свою работу и что использование чужих проектов является жульничеством. Напротив, Scratch стремится создать культуру, в которой участники будут испытывать гордость, а не огорчение, от того, что их проекты используют другие пользователи. Например, на главной странице Scratch выложены «Лучшие ремиксы», что подчеркивает престижность проектов, используемых для создания ремиксов. Таким образом, дети с раннего возраста становятся членами открытого сообщества накопления знаний.

Однако недостатком ремиксов является то, что начинающим программистам будет сложно определить код, отвечающий за какое-либо действие, что ограничивает возможности для обучения через создание полных программных ремиксов. Но некоторые инструменты, такие как *Looking Glass*, решают эту проблему с помощью функции Play & Explore, позволяющей пользователям соотносить действия программы с соответствующей строкой или строками кода, что помогает пользователям понять и внести изменения в повторно используемый код [91].

3.1.7. Создание сообщества и вклад в его развитие

Стратегии создания сообщества помогают детям внести свой вклад в развитие учебной среды и сообщества. Программирование дает им возможность представить, как сделать мир лучше при помощи компьютеров. Доказательством этому служат некоторые приложения, которые дети разработали при помощи *MIT App Inventor*. Например, десятиклассник из Ченнаи, Индия,

создал несколько актуальных приложений, например, приложение для координации деятельности по оказанию помощи после наводнения в этом регионе и приложение, позволяющее родителям следить за учениками, которые ездят на автобусах с неточным расписанием.

Некоторые из наиболее эффективных видов обучения подразумевают наличие социального контекста, когда задачу решают несколько человек. Поскольку программирование считается сложным занятием, и дети зачастую учатся более эффективно в группах, социальный контекст обучения помогает организовать учебный процесс [83]. Scratch-сообщество представляет собой пример успешной социальной среды обучения. Оно делает программирование более интересным, превращая его в общественную деятельность, когда пользователи могут загружать свои проекты, комментировать другие проекты и участвовать в существующих проектах. В сообществе представлены все четыре стадии участия в сообществах любителей фан-фикшн, определенных Дженкинсом [92]:

- **Пассивный потребитель**. Оценивает сообщество, чтобы понять его ценности и идеи. Просматривает разные категории и взаимодействует со Scratch-проектами, созданными другими пользователями.

- **Активный потребитель**. Участвует в сообществе, создавая метаданные. Делится своими идеями о чужих проектах, комментируя, помечая и оценивая проекты.

- **Пассивный поставщик**. Создает проекты, вдохновленный проектами других пользователей, которые он нашел в сообществе, но не обязательно чувствует необходимость или готовность поделиться ими с сообществом.

- **Активный поставщик**. Не только использует, но и добавляет проекты. Комментирует проекты других пользователей, вдохновляется ими и вдохновляет других пользователей.

Создание продуктивного онлайн-сообщества с уважительным отношением к пользователям — не простая автоматическая задача, поскольку грубые негативные комментарии могут помешать формированию сообщества. Если дети боятся, что члены сообщества будут высмеивать их проекты, они вряд ли будут делиться своими проектами или идеями. Напротив, когда дети чувствуют, что их окружают вежливые, заботливые сверстники, готовые внести свой вклад в их проекты, они готовы пробовать что-то новое и проявлять творческий подход. Чтобы сформировать такое вежливое и благосклонно настроенное онлайн-сообщество, создатели Scratch сделали следующее: во-первых, установили ряд простых, но действенных правил сообщества. Но одних только правил недостаточно. Во-вторых, команда Scratch постоянно подает пример своими комментариями и поведением на сайте и отслеживает комментарии. Если участники нарушают правила, модераторы пишут комментарии, делают предупреждения и дают советы, даже применяют санкции, если нарушения повторяются. И, наконец, онлайн-лагери Scratch не только помогают детям освоить новые навыки программирования, но и учат их давать конструктивную обратную связь друг другу. Благодаря этим стратегиям создатели Scratch смогли сформировать среду сплоченного сообщества для своих членов.

3.2. Список инструментов программирования

Ниже представлен перечень обучающих сред и инструментов программирования, составленный в результате поиска в сети Интернет с использованием терминов «инструменты развития вычислительного мышления для детей» и «программирование для детей». Список не претендует на полноту, поскольку рынок инструментов программирования и средств обучения программированию постоянно развивается. Некоторые из перечисленных здесь инструментов сегодня уже недоступны, но они имели важное историческое значение. Мы старались избе-

жать оценочных суждений и не оценивали точность и обоснованность заявлений создателей обучающих средств. Здесь представлен справочный список, который заинтересованные читатели смогут изучить, исходя из своих интересов. В следующем разделе мы также рассмотрим различные способы категоризации обучающих продуктов. Список составлен в алфавитном порядке.

AgentSheets и AgentCubes
AgentSheets — это многоагентная среда визуального программирования, разработанная для поддержки освоения вычислительного мышления в процессе создания игр и моделирования. Это среда для моделирования, состоящая из графических агентов (в виде иконок), которые существуют в мире, покрытом сетью координат. AgentCubes — это обучающий язык программирования, позволяющий погрузиться в трехмерное вычислительное мышление. Онлайн-сервисы AgentSheets и AgentCube основаны на педагогической концепции Репеннинга под названием зоны ближайшего потока, объединяющей идею Выготского о зоне ближайшего развития и идею Чиксентмихайи о потоковом состоянии. Предшественниками AgentSheets являются Prolog, электронные таблицы, клеточные автоматы и игры наподобие SimCity. AgentSheets предоставляет язык визуального программирования для разработки и совместного использования экранных моделей в духе SimCity [93]. Среда AgentSheets включена в учебную программу «Масштабируемый дизайн игр» для средних и старших классов, доступную на вики-странице Scalable Game Design Wiki. На основе AgentSheets были разработаны LEGOsheets, Cocoa и KidSim. В среде AgentSheets была впервые представлена форма блочного программирования, которая впоследствии была позаимствована для Squeak eToys, Alice и Scratch. AgentCubes перешел от 2D к 3D-дизайну, включив технологию 3D-моделирования под названием Inflatable Icons (надувные иконки). Эта технология позволяет пользователям сначала нарисовать 2D-версию объекта, затем постепенно довести его до 3D-объекта [94–96].
http://www.agentsheets.com/
https://agentcubesonline.com/

AlgoBlock
В 1994 году авторы AlgoBlock захотели создать активное учебное сообщество, в котором дети могли бы делиться заметками и приемами и учиться друг у друга. В AlgoBlock каждый набор блоков соответствует простой команде в Logo. Блоки можно физически связать друг с другом и составить программы, которые управляют движением подводной лодки в лабиринте. Блоки можно пощупать, они достаточно большие, чтобы их можно было разместить на столе, за которым работают несколько учеников. Это позволяет ученикам работать с блоками в социальном контексте, учиться друг у друга и озвучивать то, что они изучают. Материальность блоков позволяет детям по очереди работать с ними и сообщать о правильном расположении блоков. Проект AlgoBlock показывает, что дети могут работать вместе над созданием программ в подходящей среде. Однако блоки позволяют работать с ограниченным числом конструкций программирования; дети не могут изучить такие понятия, как процедуры, параметры или управляющие конструкции [97].

Alice
Alice — это среда программирования для создания виртуальных 3D-миров, коротких анимационных фильмов или игр. В Alice пользователи создают программы, перетаскивая графические команды и выбирая параметры из выпадающих меню. Когда пользователь перетаскивает команду, система автоматически переходит в меню, где пользователь может выбрать подходящие параметры для выбранной анимации. Таким образом, Alice помогает ученикам научиться работать со всеми стандартными конструкциями, которые проходят на первых уроках программирования, в среде, которая не дает делать синтаксические ошибки.
https://www.alice.org/

App Inventor
MIT App Inventor — это среда визуального программирования, которая позволяет детям создавать приложения для смартфонов и планшетов. App Inventor, выпущенная в 2012 году как платформа с открытым исходным кодом, доступна на нескольких языках, включая русский. У программы больше шести миллионов зарегистрированных пользователей, от учеников средней школы до студентов университетов. Пользователи могут создавать приложения, не зная досконально синтаксис Java и платформу Android. Кроме того, «живой» характер системы позволяет пользователям вносить изменения в код и сразу же видеть результат в приложении, которое они создают.
http://appinventor.mit.edu/explore/

Basic
Язык Basic (универсальная система символического программирования для начинающих) был разработан в 1960-х годах для обучения студентов-гуманитариев Дартмутского колледжа работе с компьютерами через программирование с использованием небольшого набора команд с упрощенным синтаксисом. Все команды начинаются с английского слова, чтобы новичкам было проще вникнуть в язык. Популярные в то время языки FORTRAN и ALGOL были громоздкими и сложными. Используя синтаксис поэтапно, учащиеся могли осваивать небольшую часть языка за раз, что позволяло им тратить больше времени на решение задач, а не на запоминание особенностей языка. Сайт для тех, кто с ностальгией вспоминает один из первых языков для начинающих программистов: http://www.quitebasic.com/

Blockly
Blockly — это библиотека с открытым исходным кодом, которая позволяет разработчикам добавлять кодирование с управляющей графикой в приложения или веб-страницы. Библиотека была выпущена в мае 2012 года и по состоянию на конец 2018 активно используется при разработке новых языков визуального программирования. Blockly представляет собой пользовательский интерфейс редактора блоков и платформу для генерации кода на текстовых языках. Готовый пакет включает генераторы JavaScript, Lua, PHP, Dart и Python, но можно создать собственные генераторы для других текстовых языков. Корневая библиотека написана на JavaScript и может быть интегрирована в любой сайт или встроена в WebView. Также доступны версии Blockly для Android и iOS, которые содержат набор функций для создания мобильных приложений.
https://developers.google.com/blockly/

Boxer
Boxer — одна из первых обучающих сред для начинающих программистов на основании плана Андреа ДиСесса. Представляет собой иерархический мир, состоящий из блоков, содержащих другие блоки. Boxer включала три типа блоков: стандартные блоки, содержащие текст или программный код, блоки данных, содержащие строковые литералы для использования в программах, и графические блоки, содержащие отображение графической информации. Состав блоков имел значение: он показывал, что подпроцедуры входили в состав процедур, а записи — в состав баз данных. Для начинающего программиста блоки играли роль механизма для абстрагирования элементов программ и данных. Boxer была основана на принципе наивного реализма: каждый объект в системе имел графическое представление на экране, которое можно было проверить, изменить и расширить. Например, переменные в Boxer были не просто именами. Чтобы создать переменную, на экране создавался блок с именем, который соответствовал этой переменной.

Codemoji
Учебная программа Codemoji Computer Science для школ позволяет ученикам 1-8 классов изучить основы веб-разработки и кодирования на HTML, CSS и JavaScript. В Codemoji изучить эти текстовые языки можно с помощью эмодзи и онлайн-курсов начального уровня. Преподаватели точных наук могут делиться курсами Codemoji в своих аккаунтах в Google Classroom.
https://www.codemoji.com/

CraftStudio
CraftStudio — это платформа для создания 3D-игр в режиме реального времени. CraftStudio используют в школах в качестве инструмента прототипирования, для творческого взаимодействия с детьми и обучения учащихся мыслить категориями 3D-пространства. CraftStudio позволяет нескольким пользователям одновременно управлять одним проектом (подобно тому, как это происходит в Google Docs).
https://sparklinlabs.itch.io/craftstudio

Роботы Edison
Роботы Edison являются более дешевыми аналогами серии роботов Mindstorms, поскольку стоят они порядка 50 евро. В комплект робота Edison входят моторчики, микрофон, датчик расстояния, датчик света и пульт дистанционного управления. Робота можно усовершенствовать с помощью обычных деталей Lego. Однако датчики и моторчики роботов Edison встроены, поэтому их нельзя усовершенствовать, вставив другие датчики или моторчики. Роботы Edison также умеют считывать штрих-коды. Среда программирования роботов Edison представляет собой программирование с управляющей графикой, аналогичное среде Mindstorms. Роботы могут работать как с компьютерами, так и с интеллектуальными устройствами, такими как iPad или телефоны Android. Существует также среда программирования роботов на основе Python.
https://meetedison.com/

Electronic Blocks
Electronic Blocks — это детали Lego для детей младшего возраста (3-8 лет). Дети могут строить башни, которые загораются, когда они разговаривают, или машинки, которые начинают двигаться, когда светишь на них фонариком. Предусмотрено три типа блоков: блоки датчиков, которые реагируют на свет, звук и касание, логические блоки, которые понимают команды AND, NOT, TOGGLE и DELAY, и блоки действий, которые отвечают за свет, звук и движение. Чтобы заработал синтаксис Electronic Blocks, каждый стек должен включать блок датчиков и блок действия, при этом блок действия должен находиться снизу [98,99].

Etoys
Squeak Etoys разработали для того, чтобы дети могли работать с моделями, созданными другими, или создавать собственные модели. В Etoys есть множество готовых объектов, от простых форм до мусорных баков, и инструмент для рисования, с помощью которого учащиеся могут создавать собственные объекты. У всех объектов есть контейнеры, которые содержат информацию об объекте, а также фрагменты, которые учащиеся могут перетащить из контейнера для создания программ, которые управляют поведением объекта. Программы могут менять положение, ориентацию, размер и внешний вид объектов, а также воспроизводить звуки. Пользователи могут создавать простые операторы if в своей программе, при этом в системе Etoys нет других стандартных управляющих конструкций. Пользователи могут проигрывать объекты различными комбинациями кнопок мыши. Проигрывание можно запускать и останавливать с помощью горячих клавиш, которые пользователи могут добавить в свои модели.
http://www.squeakland.org/

GameSalad
GameSalad — это среда разработки 2D-игр для iOS и Android. Ключевой особенностью GameSalad является то, что игры можно создавать в режиме drag-and-drop, изменяя свойства объектов. Создатели игры не видят «код», весь код остается за кадром. Учебная программа GameSalad, доступная в Интернете, включает учебные пособия, занятия, проекты и системы оценивания.
https://gamesalad.com /

Gamestar Mechanic
Gamestar Mechanic — это онлайн-платформа для разработки игр в виде игры о создании игр со встроенной учебной программой для развития дизайн-мышления, созданной в рамках сотрудничества профессиональных гейм-дизайнеров с учеными-исследователями из Университета Висконсин-Мэдисон, в частности, с исследователем Дж. П. Джи, который в настоящее время работает в Университете штата Аризона. Учебная программа Gamestar Mechanic встроена в приключенческую игру, в которой игроки исследуют виртуальный мир с помощью череды «игровых заданий» в стремлении стать более искусными и популярными «игровыми механиками». Задания — это деятельность, в рамках которой игроки: а) играют и выигрывают в играх, разработанных по определенным шаблонам; б) исправляют нефункционирующие игры с использованием шаблонов, изученных в процессе игры; в) разрабатывают новые игры с нуля и делятся ими с другими игроками, получая экспертную оценку и критические замечания. В процессе разработки игр учащиеся развивают навыки, связанные с точными и гуманитарными науками, креативным решением задач и системным мышлением.
https://gamestarmechanic.com/

Globaloria
Globaloria — это среда обучения, в которой ученики средних школ изучают точные науки, разрабатывая компьютерные игры на платформе Adobe Flash. Globaloria — это система социального обучения и образовательная программа под руководством Идит Харель, известной благодаря книге, посвященной конструкционизму, которую она создавала вместе с Сеймуром Пейпертом [5]. Уроки Globaloria разработаны на основе принципов конструкционистской педагогики и включают проектно-ориентированную учебную программу на основе Web 2.0 и онлайн-сообщество учителей, преподавателей и профессиональных гейм-дизайнеров. Globaloria направлена на развитие вычислительных навыков в рамках структурированной онлайн-программы обучения с упором на разработку Flash-игр в области точных наук на основе технологий Web 2.0 (вики, онлайн-видео, интерактивные учебные пособия).
https://globaloria.com/

GP
GP — это онлайн-язык программирования с управляющей графикой, похожий на Scratch. Библиотека классов и среда программирования GP написаны на самом GP, поэтому весь код можно просматривать, редактировать и отлаживать в виде блоков. До сих пор разработчики GP использовали обычные текстовые редакторы текстового представления кода GP как способ загрузки системы.
https://gpblocks.org/

Greenfoot
Greenfoot предназначена для обучения детей языку программирования Java, сочетает особенности программирования с управляющей графикой и без нее. Использует стандартный компилятор и виртуальную машину Java для обеспечения полного соответствия текущим спецификациям Java. Экземпляры классов можно сбросить в сценарии и управлять ими с помощью доступных методов или методов, написанных пользователем. В Greenfoot также есть сообщества пользователей и преподавателей на базе онлайн-общения.
https://www.greenfoot.org/door

Hopscotch
Hopscotch — это язык визуального программирования, позволяющий начинающим программистам разрабатывать игры. В приложении Hopscotch используется редактор, похожий на редактор в Scratch. В Hopscotch пользователи перетаскивают блоки для создания скриптов, которые можно воспроизводить при активации. С 2018 года Hopscotch доступен только для iPad или iPhone.
https://www.gethopscotch.com/

Hypercard
Разработанная Аткинсоном в 1980-х годах, Hypercard позволяла пользователям создавать стеки, содержащие изображения, текст и кнопки. Кнопки запускали визуальные изменения, издавали звуки или отображали новую карту. Hypertalk — скриптовый язык, позволяющий пользователям расширять функционал созданных ими стеков. Разговорный английский повлиял на язык Hypertalk, введя такие конструкции, как первая карта и последняя карта, дескрипторы, которые были понятны большинству пользователей. При разработке системы Аткинсон сделал упор на поддержку первых проектов пользователей, созданных с помощью Hypercard, и постепенное раскрытие возможностей. Начинающий пользователь сначала учится создавать карточки и использовать инструменты редактирования текста, затем переходит к редактированию графики. Пользователь учится использовать окно сообщения в качестве блока для вычислений, затем начинает размещать значения в полях.

Josef
Josef был создан в 1980-х годах и уже не используется. Как и Робот Karel, он предназначен для изучения программирования новичками в придуманном мире. Josef жил в «Вулфвилле», который был представлен в виде карты ASCII, но пользователи могли заменить карту. Робот умел поворачивать налево и направо и двигаться вперед. Пользователь мог установить скорость движения робота. В отличие от Karel, Josef мог говорить и воспринимать текстовые строки, запускающие программы ввода-вывода. Кроме того, он мог выдавать текстовые маркеры (например, строку «cat»), похожие на бипер Karel, в любой точке своего мира. В отличие от Karel, Josef был рассчитан на полный семестр изучения программирования для специальностей, не относящихся к компьютерным наукам, поэтому он включал несколько конструкций программирования, таких как параметры, переменные и рекурсия [100,101].

Karel
Робот Karel был одним из наиболее широко используемых мини-языков. Его разработал Ричард Паттис в 1980-х годах для студентов университетов, проходящих ознакомительный курс программирования [102,103]. Karel — это робот, который живет в простом сетевом мире с улицами с востока на запад и проспектами с севера на юг. В мире Karel также могут быть неподвижные стены и биперы. Karel может двигаться, поворачивать, отключаться, чувствовать стены за полквартала от него и биперы на углу, на котором находится он сам. Karel позволяет студентам наблюдать за работой их программ. К Karel прилагалось краткое руководство, чтобы учителя могли включить Karel в свои уроки. Идея миниязыкового подхода состояла в том, чтобы разработать простой язык для начала изучения программирования. Язык включал небольшой набор команд, которые мог выполнять объект, а также ряд запросов с возвратом значений и управляющих конструкций. Большинство мини-языков включают все основные управляющие конструкции (условные операторы, циклы, рекурсию и т.д.) и механизм для создания своего рода подпрограммы. Karel содержал основные управляющие конструкции в стиле Pascal для изучения базовых понятий, включая последовательное исполнение, процедурную абстракцию, условные операторы и повторения. Полноценные высокоуровневые языки программирования были упрощены, поскольку в Karel нет переменных, типов или выражений.
http://karel.sourceforge.net/

Karel++
Робот Karel J, J. Karel и Karel ++ являются версиями робота Karel, которые помогают учащимся подготовиться к объектно-ориентированному, но не процедурному программированию. Робот Karel J и J. Karel используют синтаксис в стиле Java, Karel++ использует синтаксис в стиле C++. Вместо того, чтобы создавать процедуры, чтобы научить Karel поворачивать направо, учащиеся создают подкласс основного робота для создания робота, поворачивающего направо. Эти системы используют преимущества оригинального робота Karel для ознакомления учащихся с основными понятиями объектно-ориентированного программирования.
https://homepage.cs.uri.edu/faculty/wolfe/tutorials/csc101/karel++/

Kodu
Kodu позволяет пользователям создавать собственные видеоигры, проектируя мир, выбирая персонажей и программируя их с использованием языка визуального программирования. Язык Kodu полностью событийно-ориентирован. Программы Kodu представляют собой постраничные своды правил WHEN-DO. Программирование выполняется путем размещения элементов программирования в правильном порядке для формирования условия и действия по каждому правилу. Kodu включает полную «среду разработки» 3D-игр, в том числе редактор местности, инструменты компоновки, меню персонажей и другие механизмы, которые позволяют конечным пользователям создавать «данные» мира, с которыми будет работать их код.
https://www.kodugamelab.com/

КуМир
Основная философия КуМир — это возможность расширить компактное ядро путем загрузки одного или нескольких отдельно подготовленных модулей с новыми микромирами и действующими объектами или даже новыми числовыми пакетами и абстрактными типами данных. Такой подход позволяет осуществлять настройку. Применение в реальных условиях показало, что КуМир с Роботом, Чертежником и другими действующими объектами может быть полезным инструментом для изучения программирования школьниками 12-16 лет. Математический факультет МГУ использовал КуМир, чтобы студенты могли решать простые задачи: в течение первого семестра студенты тратят около 20 часов на решение 90 задач по программированию, расчетам и алгебре.

Lego Mindstorms
Lego Mindstorms — это образовательная робототехническая платформа, созданная компанией Lego. EV3 является самой последней серией Lego Mindstorms. Есть большое сообщество пользователей, в которое пользователи могут обратиться за помощью, если у них есть вопросы. Роботов EV3 программируют в среде визуального программирования с использованием drag-and-drop интерфейса. Однако более опытные пользователи могут программировать EV3 с использованием популярных языков программирования с текстовым синтаксисом, таких как Java и Python.
https://www.lego.com/en-us/mindstorms

LegoSheets
LegoSheets, разработанная в 1990-х годах, стала попыткой введения в программирование для Programmable Brick от Массачусетского технологического института, начиная с ручного управления элементами и постепенно переходя к написанию программ. Пользователям была представлена версия Programmable Brick, в которой можно управлять частями. Пользователи могли менять скорость двигателя, подключенного к кирпичику, указав значение или используя кнопки со стрелками для увеличения или уменьшения значения. Освоив управление значениями двигателей и понаблюдав за значениями датчиков, реагирующих на различные действия, пользователи могли дважды щелкнуть мышью на изображении двигателя или датчика и вызвать редактор правил для этого объекта. Редактор правил отображал кнопки для добавления условных или начальных значений для управления кирпичиком.

Lightbot
Lightbot — это онлайн игра-головоломка для детей 9-11 лет, которая знакомит детей с понятиями программирования и кодирования. Дети управляют виртуальным роботом, чтобы подсвечивать блоки и решать головоломки разного уровня сложности. Пользователи решают головоломки, управляя роботом с помощью команд в конечном пространстве команд, называемом Main Method. Задания становятся все более сложными, и для решения головоломки требуются более сложные комбинации команд, такие как функции. Есть также упрощенная версия Lightbot — Lightbot Jr — для детей младшего возраста от 6 до 8 лет.
http://lightbot.com/

Logo
Язык программирования Logo был разработан в середине 1960-х годов Уолли Фюрцайгом и Дэнни Бобровым из BBN Labs при участии Сеймура Пейперта из Массачусетского технологического института, чтобы помочь детям изучить разные темы — от математики и естествознания до языка и музыки — с помощью программирования. Logo представляет собой диалект Lisp, из которого убрали большую часть знаков препинания, чтобы синтаксис стал понятен детям. Самой известной частью Logo является Черепашка — первая роботизированная черепаха, которая могла рисовать. Позже ее заменили объектом, который мог двигаться, поворачивать и оставлять следы в двухмерном мире. Направления движения черепашки объектно-центрированы: если ребенок говорит черепашке «вперед 10», черепашка будет двигаться вперед, а не в направлении, определяемом экраном. Многие дети начали программировать, заставляя черепашку рисовать простые картинки. Язык Logo имеет более широкий спектр возможностей, включая музыкальные программы и программы, которые переводят языки. Logo — это интерпретируемый язык с описательными сообщениями об ошибках. Задачей Logo было заставить учащихся думать о своем мышлении, выражая себя через программы, затем отлаживая эти программы, пока они не начнут работать. Сторонники Logo утверждали, что в процессе отладки учащиеся могут отлаживать свое мышление и приобретать навыки мышления более высокого порядка.
http://el.media.mit.edu/logo-foundation/resources/software_hardware.html

Logo Blocks
Logo Blocks — это язык визуального программирования, разработанный в 1996 году на основе Logo. Он был двухмерным и предусматривал процедуры и вызовы процедур. Интерфейс Logo Blocks напоминал красочный пазл. Одним из ограничений Logo Blocks, которое также является проблемой для физического программирования, был фиксированный размер блоков, что иногда мешало собирать определенные программы без блоков-заполнителей. В Logo Blocks графические формы представляли команды в Brick Logo, дополнении к Logo, которое обеспечивало команды для Programmable Brick. Среда Programmable Brick в свою очередь была предшественником системы Lego Mindstorms. Logo Blocks поддерживал процедуры: пользователи могли подключать команды к фиолетовым «процедурным блокам» и давать названия своим процедурам.

LogoWriter
LogoWriter объединил текстовый редактор, способный работать как с графикой, так и с текстом, с интерпретатором Logo. Черепашка LogoWriter может работать как курсор, меняя буквы под собой или выводя графику на странице. Ученики могут совершать языковые манипуляции и создавать смешанные программы из текста и графики так же, как в другом прикладном ПО.

Looking Glass
Looking Glass, разработанная в 2010 году для учеников средних школ, представляет собой 3D-среду программирования для создания анимационных историй и игр. Исследование Alice, предшественника Looking Glass, показало, что сочинение историй может мотивировать учеников средних школ изучать программирование. Пользователи Looking Glass создают виртуаль-

ные истории с 3D-персонажами в drag-and-drop интерфейсе. Looking Glass позволяет пользователям видеть связи между результатом и кодом, который за ним стоит. Пользователи могут присоединиться к онлайн-сообществу, просматривать сайт и отмечать понравившиеся программы. Они также могут создавать ремиксы этих программ, используя интерфейс создания ремиксов.
https://lookingglass.wustl.edu/

Micro:bit
Micro:bit — это плата размером с ладонь с 25 светодиодами, которые можно запрограммировать на отображение букв, цифр и других форм. Она дает возможность детям в возрасте от 14 до 16 лет разобраться с электроникой в процессе программирования. Micro:bit можно запрограммировать с использованием Blocks, JavaScript или Python.
https://microbit.org/

MOOSE Crossing
MOOSE Crossing — это сетевая среда программирования, созданная для детей. Это адаптированное текстовое многопользовательское подземелье, в котором дети могут использовать объектно-ориентированный скриптовый язык для создания пространств и персонажей, которые населяют текстовый мир [104].

NetLogo
NetLogo — это бесплатная мультиагентная программируемая среда моделирования с открытым исходным кодом. По данным сайта NetLogo используют тысячи учеников, преподавателей и исследователей во всем мире. NetLogo написан на Java и Scala. Среда программирования NetLogo включает большую продуманную коллекцию моделей (Damaceanu 2011; Wilensky and Rand 2015).
https://ccl.northwestern.edu/netlogo/

NetsBlox
NetsBlox — это язык визуального программирования и облачная среда, которая позволяет начинающим программистам создавать сетевые программы, например, многопользовательские игры. Его визуальное представление основано на Scratch. Он использует открытый исходный код JavaScript на базе Snap!. Общедоступные научные и другие источники данных NetsBlox позволяют создавать проекты в сфере точных наук, такие как отображение сейсмической активности в любой точке Земли с использованием Google Maps. NetsBlox поддерживает совместное редактирование программ аналогично Google Docs.
https://netsblox.org/

Ozobot
Ozobot — это маленький робот, который ученики начальной и средней школы могут запрограммировать с помощью редактора кода с управляющей графикой или просто цветами. В системе цветового кода дети используют цвета и рисунки для управления движениями Ozobot. Ozobot может следовать этим рисункам, используя цветовой датчик, прикрепленный к его подошве и предварительно запрограммированный на смену действий при обнаружении других цветов. Помимо цветов и рисунков, Ozobot также можно запрограммировать с помощью смартфонов или планшетов с использованием редактора кода с управляющей графикой. https://ozobot.com/

Pascal
Первая версия Pascal была создана в 1970 году, чтобы профессора могли преподавать структурированное программирование своим студентам в рамках их первого курса, а именно системного программирования. Хотя он был разработан для целей обучения, доработки языка

отражали общие усовершенствования языков программирования. Например, Pascal устранил некоторую двусмысленность интерпретации вложенных операторов if. Кроме того, он добавил новые базовые типы и возможность определять специальные типы с помощью операторов структуры.
http://www.pascal-programming.info/index.php

Pencil Code
Pencil Code — это инструмент визуального программирования для рисования, воспроизведения музыки и создания игр. Редактор Pencil Code Block основан на языках визуального программирования Scratch и Blockly, однако его можно в любой момент переключить с блоков на текст. Вместо естественноязыкового текста и иконок в каждом блоке есть названия функций, фактические параметры и синтаксис текстового языка CoffeeScript. Блоки располагаются не в двух измерениях, а помещены в одну основную программу в линейном порядке. В программе есть нумерация строк. Такая структура призвана помочь учащимся перейти от языка визуального программирования к языку программирования с текстовым синтаксисом, поскольку начинающие программисты могут создать программу из блоков и одновременно изучить язык с текстовым синтаксисом.
https://pencilcode.net/

PictoMir
Свободно распространяемая многоплатформенная образовательная и игровая система PictoMir была разработана в 2014 году ФГУ ФНЦ НИИСИ РАН по запросу Российской Академии Наук. PictoMir позволяет дошкольникам в возрасте от 5 до 7 лет изучать понятия программирования, включая подпрограммы, повторители, обратную связь, команды и запросы, ветвление и счетчики. PictoMir внедряют в программу изучения алгоритмического программирования в дошкольных учреждениях и начальных классах школ. Следующая часть программы для средних и старших классов — школьный алгоритмический язык, система КуМир [105]. https://piktomir.ru/

Pocket Code
Pocket Code — это приложение для мобильных устройств Android. Оно позволяет подросткам создавать собственные игры, анимацию, видео и некоторые другие приложения. Pocket Code использует язык визуального программирования. Для его использования не требуется опыт программирования, поэтому он подходит новичкам, которые хотят изучать процессы выполнения программы и понятия логики.
https://www.catrobat.org/#pocketcode

Prolog
Prolog, разработанный в 1970-х годах, был популярным языком программирования на базе правил даже среди начинающих программистов. Его интерфейс был похож на командную строку. В Prolog пользователь мог излагать факты о мире для выполнения команд. Prolog избежал некоторых сложностей, с которыми сталкивались учащиеся, таких как определенные циклы и условные операторы.

Robbo
Российская компания Robbo предлагает наборы для занятий робототехникой. Первый набор Lab представляет собой плату с датчиками и кнопками. Комплект Lab можно использовать со средой программирования Scratch, когда он подключен к компьютеру через USB кабель или Bluetooth. Поскольку комплект Lab оснащен микроконтроллерами Arduino, более опытные ученики также могут запрограммировать его на языке Arduino.
https://www.robbo.world/products/

Robocode
Robocode, разработанная в 2000 году, — это игра для программистов, предназначенная для изучения Java. Пользователи пишут код и создают боевой танк-робот, который соревнуется с другими роботами. Хотя основным языком для программирования в Robocode является Java, также используются C# и Scala. В Robocode предусмотрено обучающее руководство, есть группа Google и совместный сайт под названием RoboWiki, где пользователи могут узнать о Robocode и общаться друг с другом.
https://robocode.sourceforge.io/

ScratchJr
В приложении ScratchJr дети в возрасте от 5 до 7 лет могут научиться программировать собственные игры и интерактивные истории при помощи языка визуального программирования. Дети соединяют блоки программирования, чтобы их персонажи могли двигаться, прыгать, танцевать и петь. В основе ScratchJr лежит язык программирования Scratch.
https://www.scratchjr.org/

Smalltalk
Первая версия Smalltalk была создана в 1971 году компанией Xerox PARC как язык для KiddyKomputer (оригинальное название Алана Кея для портативного компьютера, созданного для детей). В основу Smalltalk положены три идеи: (1) все является объектом; (2) объекты обладают памятью в виде других объектов; и (3) объекты могут связываться друг с другом посредством сообщений. Smalltalk был разработан в процессе создания Dynabook — компьютера, предназначенного для обучения посредством создания и исследования широкого спектра носителей, совместимых с компьютером [106].

Snap!
Snap! (ранее BYOB — Build Your Own Blocks) — это язык визуального программирования с перетаскиванием. Это расширенное воплощение Scratch. Основное отличие от Scratch — наличие списков первого уровня, процедур первого уровня и продолжений. Эта расширенная версия позволяет пользователям создавать другие структуры данных, такие как дерево, неупорядоченный массив данных, хеш-таблица и словарь, благодаря чему Snap! подходит для относительно продвинутого введения в компьютерные науки среди старшеклассников или студентов ВУЗов. Snap! работает в веб-браузере. Он создан с использованием JavaScript, благодаря чему на программу и даже проекты других людей не влияет ПО на базе браузеров.
https://snap.berkeley.edu/

StarLogo
StarLogo, выпущенная в 2006 году как среда моделирования с открытым исходным кодом, была разработана, чтобы помочь учащимся моделировать реальные явления, например, дорожные пробки и рост населения. Среда была основана на языке программирования Logo. В Logo пользователи могли контролировать движение одной черепашки, а в StarLogo можно было контролировать тысячи черепашек. Восприятие черепашек StarLogo было усовершенствовано: они могли обнаружить черепашек рядом и чувствовать запахи в мире [107]. После StarLogo разработчики создали StarLogo TNG.
http://web.mit.edu/mitstep/starlogo/

StarLogo TNG (StarLogo Nova)
StarLogo TNG — это среда программирования на основе 3D-графики с использованием drag-and-drop блоков программирования вместо языков программирования с текстовым синтаксисом. StarLogo TNG — это клиентское ПО для моделирования [108,109]. Это помощник в создании и понимании моделирования сложных систем. StarLogo Nova предоставляется бесплатно, работает в браузере и дает возможность учащимся делиться своими проектами и играть в игры, созданные другими. Учителя также могут использовать модели, разработанные сообществом

StarLogo, для объяснения сложных понятий. Пользователи могут создавать простые или масштабные модели, используя существующую библиотеку агентов или импортируя свои звуки и 3D-модели в формате Collada (Collada или Collaborative Design Activity — это формат обмена файлами, позволяющий передавать 3D-решения в другие приложения).
https://www.slnova.org/

Stencyl
Stencyl — это среда разработки 2D-игр путем перетаскивания. Она работает на Flash, iOS, Windows, Mac и Android. Американская компания Stencyl, LLC, специализирующаяся на образовательных технологиях, выпустила Stencyl в 2011 году. Пользователи создают свои игры при помощи функций Stencyl, таких как создание и редактирование кода игры в модульных элементах, создание и редактирование игровых объектов и игровых уровней. Дополнительной функцией Stencyl являются пользовательские блоки, с помощью которых пользователи могут создавать свои блоки кода с функциями, входящими параметрами и исходящими возвращаемыми значениями [110].
http://ww.stencyl.com

Story Blocks
Со Story Blocks пользователи создают программы, собирая физические блоки, которые представляют персонажей и действия, в интерактивные истории. Затем пользователь может просмотреть эту программу с помощью камеры мобильного устройства. Программа выполнена в виде аудио-истории на мобильном устройстве. Таким образом, Story Blocks включает как инструменты для создания программ, так и универсальное устройство вывода. Создание более доступных инструментов программирования может помочь детям с различными способностями в изучении компьютерных наук. В существующем прототипе блоки выполнены из недорогих материалов с использованием 3D-принтера или лазерного резака. Каждый блок сделан так, чтобы его можно было различить по форме и цвету, поэтому слепые дети также могут их использовать. Как и в других физических языках программирования, форма каждого блока подсказывает, как его можно соединить с другими блоками [111].

Tern
Tern — это физический язык программирования из деревянных блоков. В собранном виде эти блоки могут программировать таких роботов, как LEGO Mindstorms RCX или iRobot Create. Форма блоков создает физический синтаксис, который не даст создать неверные программы, а код идентифицируется и обрабатывается машинным зрением.
http://hci.cs.tufts.edu/tern/

Thimble (Mozilla)
Thimble — это онлайн-редактор кода, с помощью которого пользователи могут создавать и выкладывать веб-страницы, изучая HTML, CSS и JavaScript. Это проект с открытым исходным кодом, созданный Mozilla Foundation совместно с Центром развития открытых технологий колледжа Сенека. Он работает в соответствующем веб-браузере и доступен на нескольких языках, включая русский. Пользователи могут создавать ремиксы проектов, доступных на сайте Thimble, для собственных сайтов, затем делиться проектами с друзьями. На сайте представлены примеры проектов, из которых новые пользователи могут черпать идеи. В процессе написания кода пользователи сразу видят изменения на своих сайтах.
https://thimble.mozilla.org/

Thunkable
Thunkable — это платформа для разработки приложений для Android или iOS. На сайте пользователи могут создавать собственные мобильные приложения, используя среду программирования с функцией drag-and-drop. Thunkable похожа на MIT App Inventor. Дети в возрасте от 12 до 18 лет могут использовать Thunkable в образовательных целях, однако другие пользователи,

не относящиеся к этой возрастной группе, могут использовать платформу для создания коммерческих или бесплатных приложений.
https://thunkable.com

Tynker
Tynker — это образовательная программная платформа, предназначенная для обучения детей созданию игр и программ. Визуальный дизайн и принципы работы основаны на языке программирования с управляющей графикой, однако писать код можно и на текстовых языках программирования JavaScript и Python. Tynker можно использовать в браузере без плагинов, а также на планшетах и смартфонах.
https://www.tynker.com/

3.3. Пять значимых аспектов: путеводитель для растерявшихся

Учитывая разнообразие представленных на рынке инструментов, непросто выбрать инструменты, подходящие для учителя, родителя или другого помощника. Детям также сложно сделать самостоятельный выбор. С концептуальной точки зрения идея позитивного развития технологий (ПРТ), озвученная Мариной Умащи Берс, включает полезные критерии. В этом разделе мы также рассмотрим некоторые практические средства разделения разных инструментов по пяти аспектам.

3.3.1. Аспект 1. Тип языка программирования

Основные языки программирования, используемые разработчиками программного обеспечения, имеют довольно сложную структуру, множество команд и правил синтаксиса, а также логику, которой необходимо придерживаться, чтобы программа «заработала». Выполнение программы подразумевает компиляцию (обычно в асинхронном режиме вне экрана компьютера) в машиночитаемый набор команд, который впоследствии реализуется или выполняется. Программа с неправильным синтаксисом не будет компилироваться и выполняться. Программа может компилироваться, но не будет выполнять свою функцию из-за других ошибок. Параллельное освоение вычислительных концепций и изучение программирования с использованием стандартного текстового языка, как правило, подразумевает знание ряда команд для овладения базовым синтаксисом — если программа не работает, учащийся не понимает, связана ли проблема с синтаксисом, логикой или другим вычислительным элементом.

Визуальное программирование

Визуальное программирование было создано для решения проблемы освоения учениками сложного синтаксиса программы. На рисунке 3.1 показано, как визуальное программирование упрощает интерфейс, устраняет связанные с синтаксисом проблемы и позволяет новичку сосредоточиться на развитии вычислительного мышления. Абстрагирование от синтаксиса оказалось мощным стимулом. Теперь новички могут с первых дней писать сложные и эффективные программы, что является одним из требований концепции «низкий пол, высокий потолок» для преподавания программирования детям. Языки визуального программирования, такие как Scratch и Alice, являются практической альтернативой сложным языкам программирования с текстовым синтаксисом. Они существенно упрощают синтаксис и делают команды больше похожими на разговорный язык. Они заменяют слова картинками или блоками. Например, в Scratch блоки, похожие на кирпичики Lego, представляют несколько управляющих конструкций. «Программирование» осуществляется путем перетаскивания и составления этих блоков друг с другом, как в пазлах [103]. Как и кирпичики Lego, блоки соединяются друг с другом только в правильном порядке. Блоки также разбиты по категориям цветовым кодом, благодаря

Рисунок 3.1. Условные утверждения if-else с использованием текстовых (JavaScript) или визуальных (Scratch) программ

Составные утверждения

Пара фигурных скобок { } и вложенная последовательность утверждений представляют составное утверждение, которое можно использовать аналогично обычному утверждению.

If ... else

```
if (expr) {
  //statements;
} else if (expr2) {
  //statements;
} else {
  //statements;
}
```

Условный (тернарный) оператор

Тернарный оператор создает выражение, которое оценивается как одно из двух выражений в зависимости от условия. Схож с обычной конструкцией if, которая выполняет одно из двух утверждений в зависимости от условия. Т.е. тернарный оператор для выражений — то же, что if для утверждений.

```
result = condition ? expression : alternative;
```

аналогично:

```
if (condition) {
result = expression;
} else {
result = alternative;
}
```

Источник: синтаксис JavaScript в Wikipedia
https://en.wikipedia.org/wiki/JavaScript_syntax

Условия

В программировании ***условие*** — это то, что должно быть истинным, чтобы произошло какое-либо действие.

Пример такого блока:

Представленная конструкция известна как конструкция if-then. С ее помощью мы можем дать команду спрайту поздороваться, только если пользователь, скажем, нажмет кнопку мыши:

Похожая конструкция называется конструкция if-else:

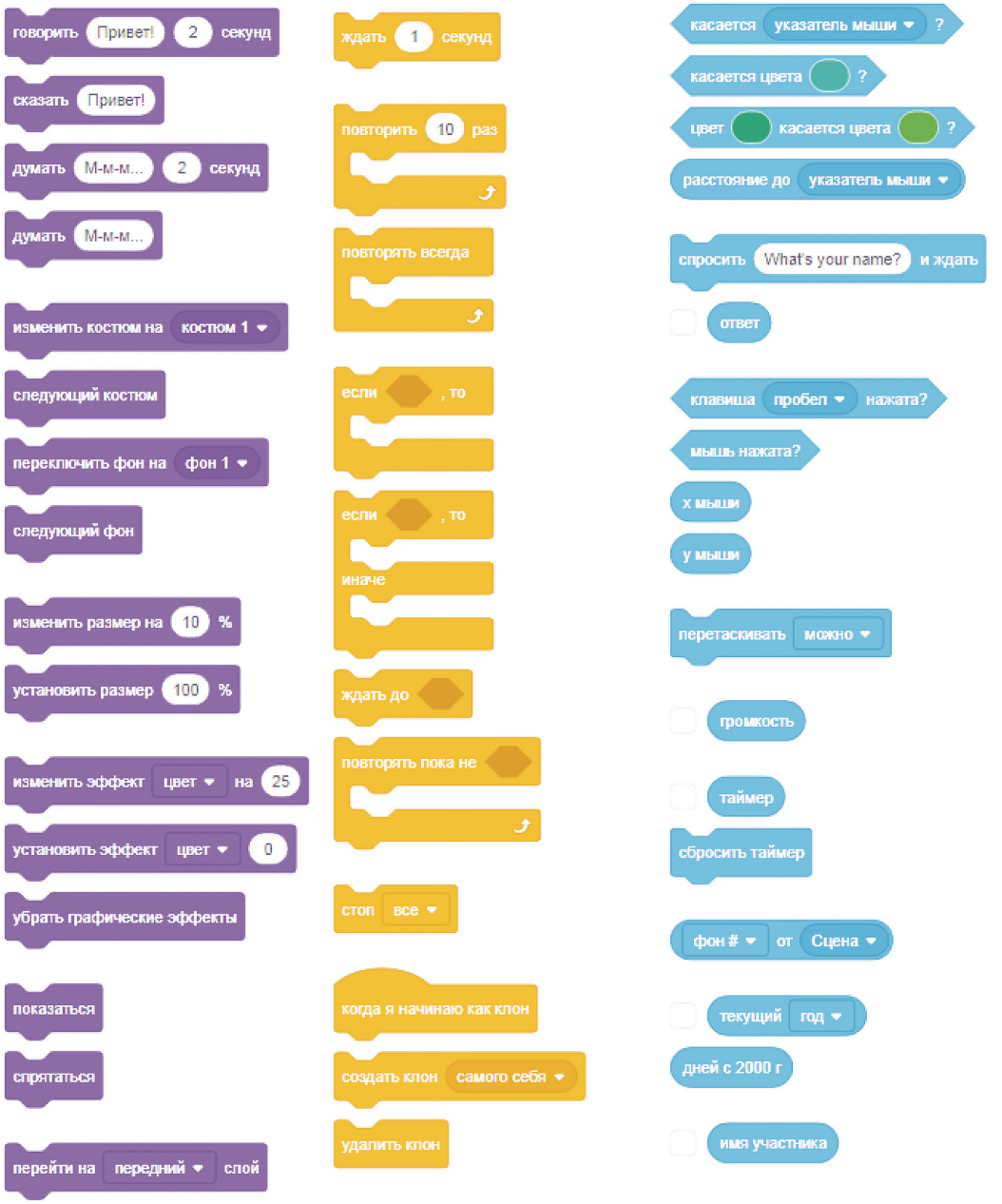
С ее помощью мы можем дать команду спрайту поздороваться или попрощаться в зависимости от того, нажал ли пользователь кнопку мыши:

Источник: отрывок из работы «Scratch для начинающих программистов» Дэвида Малана из Гарвардского университета
https://cs.harvard.edu/malan/scratch/conditions.php

чему пользователи могут найти соответствующие блоки (см. рисунок 3.2). Синтаксис утверждений закодирован в блоках, и учащимся нужно всего лишь распознать блоки, что не позволит им создать утверждение с синтаксическими ошибками.

Исследователи изучили относительную эффективность текстовых языков по сравнению с языками с управляющей графикой для развития навыков вычислительного мышления [112,113]. Старшеклассники отдали предпочтение блокам, потому что: (i) их проще читать; (ii) формы и цвет блоков способствуют пониманию; (iii) их проще собирать; и (iv) они помогают запоминать основные понятия вычислительного мышления [113].

Рисунок 3.2. Scratch-блоки с программированием цветом и формой, что облегчает сборку



Осязаемое программирование

Вместо слов или блоков / иконок пользователи группируют и соединяют физические объекты, например, деревянные блоки, для создания компьютерных программ. Преимуществом языков осязаемого програмирования являются физические свойства объектов, такие как размер, форма и материал, заменяющие синтаксис. Осязаемым программированием, как правило, занимаются дети самого младшего возраста, поскольку оно дает возможность создавать код простыми командами без использования гаджетов.

Термин «осязаемый язык программирования» придумали Сузуки и Като для описания своей среды совместного программирования для детей AlgoBlock [97]. Уникальной особенностью, которая отличает AlgoBlock от других сред программирования, является понятный интерфейс. Вместо того чтобы управлять виртуальными объектами на экране компьютера, пользователи AlgoBlock группировали физические блоки на столе для коммуникации с компьютером. «Осязаемое программирование» означает деятельность по упорядочению блоков для построения (в отличие от «написания») компьютерных программ. Группа исследователей DevTech из Университета Тафтса разработала робот Kibo, представленный на рисунке 3.1, для детей 4-7 лет [12]. Дети группировали блоки, затем при помощи встроенной камеры робота Kibo изучали и обрабатывали код (см. рис. 3.3).

Рисунок 3.3. Робот Kibo — физическое программирование, разработка группы DevTech из Университета Тафтса

Strawbies / Osmo и Tica — это интерфейсы физического программирования для iPad и мобильных устройств Android, соответственно, считывающие физический код, созданный путем составления физических фрагментов вместо визуальных блоков. Как и в случае с визуальными блоками, для физических фрагментов предусмотрены разъемы, которые не дают сделать синтаксические ошибки [114]. По словам разработчиков Strawbies, они использовали принципы проектирования, которые будет полезно знать, поскольку они широко применяются для развития навыков вычислительного мышления [115]. Strawbies — это игра с использованием iPad, световых сенсоров и физических элементов, из которых собирается код программы. Цель игры состоит в том, чтобы соединить фрагменты в код для персонажа по имени Awbie, чтобы он мог перемещаться по экрану iPad и выполнять действия в соответствии с указаниями программы. Используются следующие принципы проектирования: (i) игра должна вызывать интерес — во время тестовых занятий дети замечали объекты игры Strawbies с другого конца комнаты; (ii) изучение в игровой и ничем не ограниченной форме — Awbie должен вырастить и съесть клубнику, но дети могут исследовать и другие возможности; (iii) простое + сложное или метафора «низкий пол, высокий потолок», хотя потолок кажется не таким уж высоким, так как основную аудиторию составляют дети дошкольного возраста; (iv) гибкость и реакция на действия пользователей — использование светочувствительных датчиков и iPad позволяет осу-

ществлять взаимодействие в режиме реального времени без дорогостоящего оборудования, такого как камера для сканирования фрагментов; (v) подходит для развития — было установлено в ходе нескольких раундов тестирования; (vi) педагогическая выверенность — подходит для представления дошкольникам понятий вычислительного мышления в знакомой культурной форме с разнообразными сенсомоторными возможностями; (vii) возможность социального взаимодействия — физические объекты могут использовать разные пользователи, и дети могут взаимодействовать друг с другом, решая, кто и чем будет заниматься; и (viii) адаптивность — дети разных возрастов и с разными способностями могут играть вместе, например, ребенок, не владеющий грамотой, может использовать значки «вверх» или «вниз», а также попросить ребенка старшего возраста помочь ему прочитать непонятную команду.

В качестве примеров осязаемого программирования также можно назвать: Dr. Wagon [116], Robo-Blocks, Tern [117], Twinkle [118], Project Bloks, Story Blocks [111], TanPro-Kit, TanProRobot, Topobo and ToonTalk.

Смешанное программирование

Большинство обучающих инструментов используют один из трех видов языков программирования, однако некоторые инструменты используют смешанный подход, который позволяет детям переключаться с одного вида языка на другой. Смешанный подход позволяет детям работать с несколькими представлениями [119]. Например, ребенок может начать изучать программирование с визуального языка, а затем в процессе работы перейти на текстовый язык.

Примерами таких смешанных текстовых и визуальных языков являются Greenfoot и Pencil Code.

3.3.2. Аспект 2. Создание контента (назначение инструмента)

Создание контента связано с назначением инструмента и соотносится с компонентом «Создание контента и компетенция» концепции позитивного развития технологий.

Прежде всего, инструменты, облегчающие создание контента для детей, можно разделить на две группы: 1) инструменты, которые учат детей программированию, и 2) инструменты, направленные на развитие основных навыков вычислительного мышления без программирования. Дальнейшая классификация этих категорий представлена ниже.

Инструменты, которые учат детей программированию, можно разделить на восемь групп:

1. Инструменты для сочинения историй при помощи программирования.
2. Инструменты для разработки игр.
3. Инструменты, которые учат программированию в процессе игры.
4. Инструменты для разработки приложений.
5. Инструменты для создания художественных работ при помощи программирования.
6. Инструменты для программирования учебных роботов.
7. Инструменты для программирования носимых технологий.
8. Инструменты, которые учат текстовым языкам традиционными способами.

Инструменты, предназначенные для развития основных навыков вычислительного мышления без программирования, можно разделить на две категории:

1. При помощи игр с минимальным использованием кода или без кода.
2. В рамках занятий без гаджетов.

Подробное описание десяти инструментов (восемь инструментов для программирования и два инструмента без программирования) представлено ниже.

Инструменты, которые учат детей программированию

Большинство стратегий обучения детей программированию при помощи различных вариантов создания контента, было описаны в разделе 3.1. (сочинение историй, разработка игр, разработка приложений, создание художественных работ, учебные комплекты для занятий робототехникой, программирование носимых устройств).

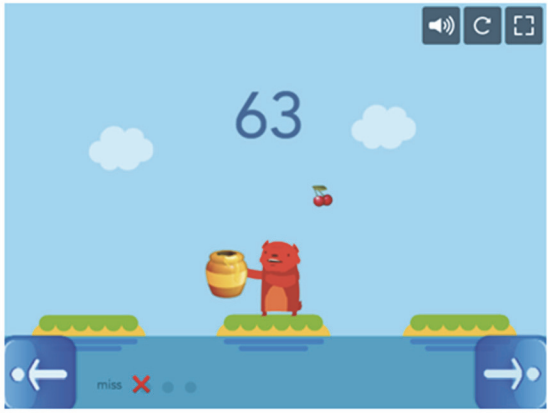
1. Сочинение историй. Одним из наиболее распространенных инструментов, с помощью которых дети могут сочинять креативные истории и создавать анимации, является Scratch. Это среда визуального программирования, в которой пользователи в возрасте от 8 до 16 лет могут создавать интерактивные анимационные истории и игры и делиться ими в процессе изучения компьютерного программирования. Это бесплатная программа была запущена в 2007 году в Media Lab Массачусетского технологического института, и сегодня более 23,5 миллионов зарегистрированных пользователей по всему миру используют Scratch для создания проектов. Scratch помогает детям выражать себя творчески таким же образом, как они бы рисовали картины или строили замки из кирпичиков LEGO. Кроме Scratch есть множество других инструментов, с помощью которых дети могут сочинять истории. Другие известные платформы для сочинения историй: Alice (Storytelling Alice, Looking Glass), PageCraft [120], Story Blocks [111].

2. Разработка игр. Инструменты для разработки игр поддерживают концепцию «обучения путем создания игр». Эти инструменты предназначены для совершенствования вычислительного мышления и навыков программирования и помогают детям создавать собственные игры. Для разработки игр необходим высокий уровень навыков создания контента.

3. Игра в игры. Вышеупомянутая разработка игр направлена на развитие вычислительного мышления и навыков программирования у детей при помощи создания собственных игр. Однако для создания игр нужен творческий подход и серьезные навыки создания контента. Поэ-

Рисунок 3.4. Примеры игр, разработанных детьми с помощью Hopscotch и Kodu

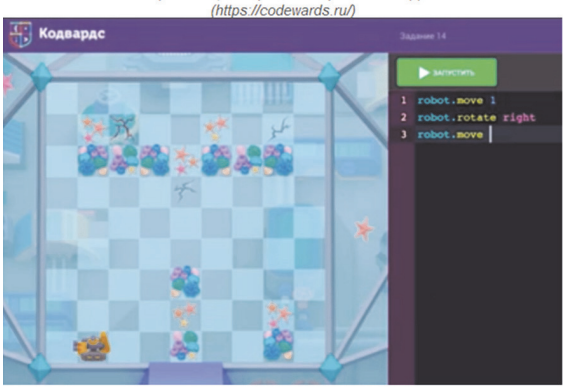
Hopscotch: игра Cherry Pot, разработанная ребенком с использованием Hopscotch - визуальный язык программирования для мобильных устройств, помогающий детям 7-13 лет создавать свои игры *(https://community.gethopscotch.com/projects/xl5e12x97)*



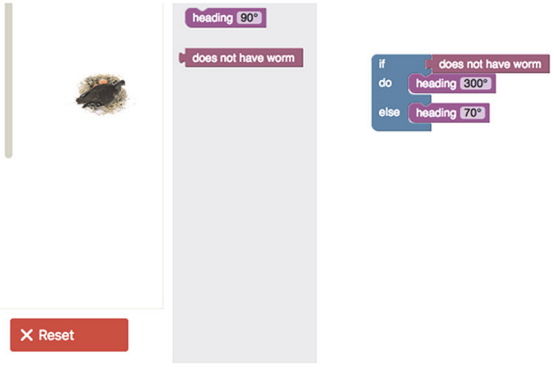
Kodu: дети от 9 до 14 лет разрабатывают 3D-игры на Kodu и выкладывают их в Интернет, чтобы пользователи могли играть в них

Рисунок 3.5. Примеры инструментов, которые учат детей программированию во время игры

Codewards: в этой игре, разработанной на русском языке, ребенок перемещает виртуального робота из одного места в другое, вставляя код. Существует несколько уровней сложности по мере того, как ребенок решает задачи.
(https://codewards.ru/)



Blockly Games: в этой игре, разработанной на английском языке, ребенок перемещает виртуальную птицу в ее виртуальное гнездо, перетаскивая визуальные команды. В игре есть несколько уровней.
(https://blockly-games.appspot.com)

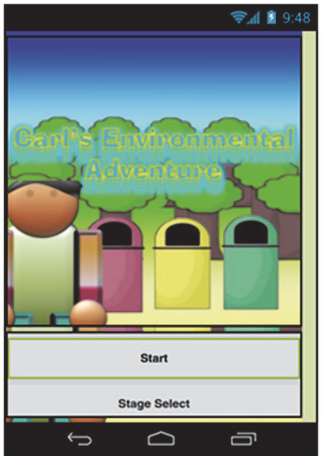


тому некоторые продукты учат детей программированию простым способом — во время игры. Играя в игры с такими продуктами, ребенок дает команды персонажам игры путем программирования и проявляет творческий подход, перемещая персонажей для достижения цели.

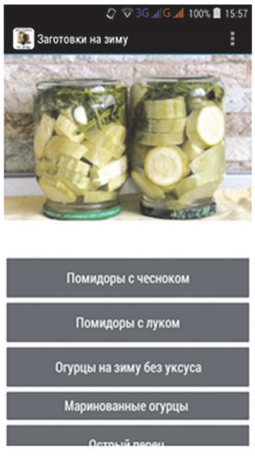
4. Разработка приложений. Дети могут даже создавать приложения для смартфонов и планшетов, используя специальные инструменты, такие как MIT App Inventor и Thunkable. Содержание этих приложений варьируется от простых игр до проектов, отмеченных наградами, например, группа молодых женщин из Молдовы создала краудсорсинговое приложение, чтобы помочь жителям своей страны получить доступ к источникам безопасной питьевой воды http://appinventor.mit.edu/explore/about-us.html.

Рисунок 3.6. Примеры приложений для смартфонов, разработанных с использованием MIT App Inventor и Thunkable

App Inventor Массачусетского технологического института: приложение семиклассника из Филиппин - образовательная игра, в которой игрок помогает Карлу в его миссии по спасению Земли.
(http://ai2.appinventor.mit.edu/?galleryId=5573847883776004504036775837696)

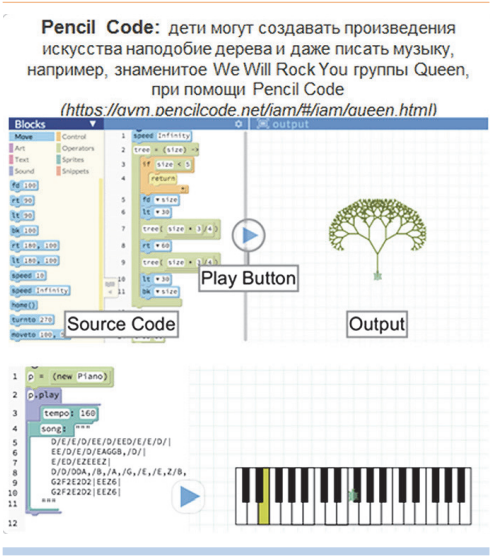


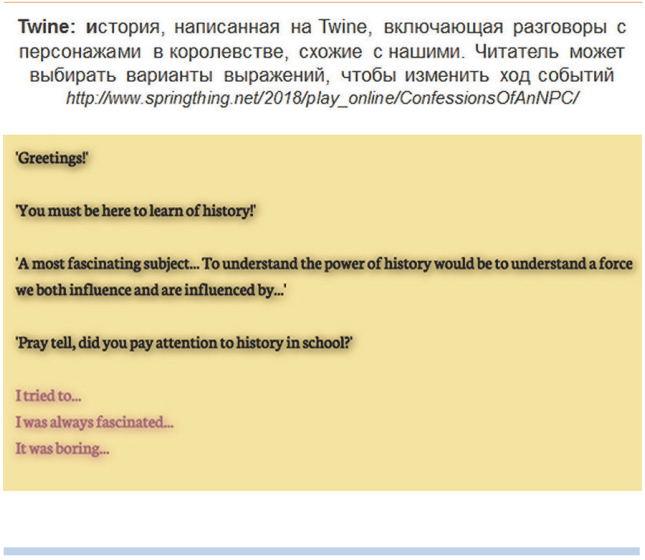
Thunkable: приложение, разработанное в России при помощи Thunkable - «Любимые рецепты закаток» - содержит советы шеф-повара по приготовлению закаток от кимчи до квашеной капусты.
(https://play.google.com/store/apps/details?id=com.thunkable.android.arhipovad66.zagotovkinazimu)



5. Создание художественных работ. С помощью детских инструментов для развития вычислительного мышления и обучения программированию можно рисовать, сочинять музыку и даже рассказы. В этом отношении в креативном программировании не только используются уже существующие технологии, но и создаются новые технологии для развития творческого потенциала, что делает его идеальным инструментом для образования путем развития креативности [121]. Бесплатный инструмент для программирования Pencil Code, доступный на pencilcode.net, — хороший пример того, как дети могут рисовать или сочинять музыку при помощи программирования [91,122,123]. Инструмент с открытым исходным кодом Twine для сочинения интерактивных нелинейных текстовых историй помогает детям и подросткам сочинять интерактивные истории, используя JavaScript и CSS.

Рисунок 3.7. Примеры создания художественных работ с использованием Pencil Code и Twine



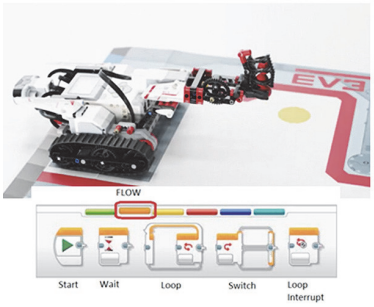


6. Программирование учебных комплектов для занятий робототехникой. Используя такие комплекты, ребенок работает с роботизированным артефактом или создает его с нуля, затем программирует его поведение при помощи неограниченных комбинаций кода. С помощью этих комплектов дети могут «оживить» робота. Для написания кода можно использовать компьютер, а также физические объекты, например, совмещаемые деревянные или пластиковые блоки. Комплекты для занятий робототехникой помогают детям развивать мелкую моторику и зрительно-моторную координацию и вовлекают их в занятия, способствующие совместной и командной работе. За последние годы количество учебных комплектов значительно увеличилось благодаря работе новых образовательных стартапов и исследовательских лабораторий. Примерами использования комплектов для занятий робототехникой для создания различных видов контента являются Lego Mindstorms, Ozobot и Code-a-pillar, представленные на рисунке 3.8.

7. Программирование носимых устройств. В последние годы носимые технологии не обошли стороной и рынок для детей. В США несколько компаний, занимающихся обучающими технологиями, начали разрабатывать портативные устройства для детей, которые можно запрограммировать на проявление нескольких функций. Пока эти продукты еще не очень распространены (мы нашли только два примера, оба в виде браслета, разработанного в США), но они могут послужить вдохновением для других новаторов в сфере образования.

Рисунок 3.8. Создание контента с использованием учебных комплектов для занятий робототехникой

Lego Mindstorm EV3: дети собирают робота из деталей Lego, включая датчики и двигатели, а затем управляют его поведением с помощью кода на компьютере



Ozobot: дети могут управлять маленьким роботом Ozobot, используя систему цветового кода: когда Ozobot обнаруживает уникальные последовательности цветов, он совершает разные действия и меняет свое поведение

Code-a-pillar: дети собирают и разбирают легко соединяемые сегменты робота-гусеницы в разных сочетаниях, чтобы построить путь для передвижения управляемой гусеницы Code-a-pillar

Рисунок 3.9. Примеры носимых устройств, которые дети могут запрограммировать

Jewelbots: программируемые «браслеты дружбы» для девочек: браслеты соединяются друг с другом через Bluetooth и могут быть запрограммированы (например, загораются радужные огни, когда три браслета оказываются в одной комнате)
(https://youtu.be/wCh9dwHSWcQ)

Mover Kit: Mover Kit - это браслет, в котором дети могут писать коды визуального программирования, чтобы менять цвета / фигуры в ответ на физические движения (например, радужные огни в течение 2 минут, пока ребенок чистит зубы)
(https://www.techwillsaveus.com/shop/mover-kit/)

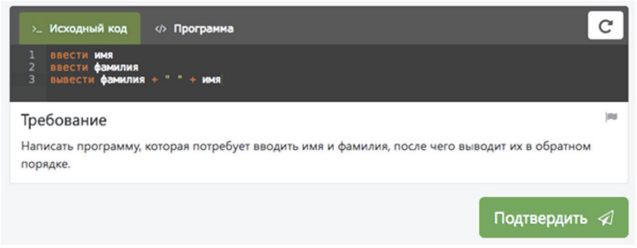
8. Разработка программ при помощи языков с текстовым синтаксисом. Такие продукты обучают традиционным языкам программирования с текстовым синтаксисом, таким как HTML, CSS, JavaScript и т. д., в онлайн-среде, то есть в форме онлайн-курсов. Эти инструменты не включают в себя функции создания «творческого» контента, такие как разработка игр и сочинение историй. Вместо этого ребенок, закончив урок, переходит на другой уровень. Однако, поскольку ребенок учится программировать на традиционных языках с текстовым синтаксисом, конечные продукты могут быть намного сложнее и разнообразнее по сравнению с продуктами, созданными на языках визуального программирования. Например, ребенок может создать сайт, калькулятор, веб-приложение и другие сложные продукты, хорошо изучив язык с текстовым синтаксисом.

Инструменты, предназначенные для развития основных навыков вычислительного мышления без программирования

Инструменты, представленные на рынке, не обязательно должны учить программированию, чтобы развивать навыки вычислительного мышления у детей. Есть множество других доступных инструментов, для работы с которыми не требуется писать код на компьютере / планшете, но которые все же могут внести существенный вклад в развитие навыков вычислительного мышления у детей. Наше исследование показало, что эти инструменты развивают навыки де-

Рисунок 3.10. Примеры инструментов программирования, обучающих традиционным языкам программирования с текстовым синтаксисом

LinguaCode: разработанная армянским стартапом на русском языке, платформа LinguaCode учит программировать на Python, Pascal и JavaScript посредством онлайн-уроков, **от базового до продвинутого уровня**
(https://linguacode.me/)



Codemoji: платформа, которая обучает детей HTML, CSS, JavaScript с использованием emojis и понятных курсов
(https://codemoji.com/)

тей, вовлекая их в игровую среду или занятия вне экрана, например, учебные комплекты для занятий робототехникой, настольные игры или занятия без гаджетов. В следующей части работы мы приводим описание обоих типов и их потенциала в создании контента, что в итоге будет содействовать развитию навыков вычислительного мышления.

1. Игры с минимальным использованием кода или без кода. Вместо того чтобы писать код, дети могут играть в игры на компьютере / планшете, используя простые команды, которые развивают навыки вычислительного мышления. Основное отличие этих игр от простейших игр состоит в том, что они включают основные компоненты вычислительного мышления, такие как планирование, тестирование, отладка, процедуры и циклы. Поиграв в такие игры, ребенок сможет перейти к более сложным играм, которые включают программирование, и в итоге сможет разрабатывать собственные игры.

Рисунок 3.11. Пример игры без программирования

ПиктоМир: позволяет детям создать простую программу с пиктограммами для выполнения определенных задач в игре с участием виртуального робота *(https://piktomir.ru/online/)*

2. Занятия без компьютера и экрана. Занятия вне экрана, как правило, ориентированы на детей младшего возраста от 4 до 9 лет. Для таких занятий не требуется компьютер, планшет или другая форма «экранного времени», что может решить актуальную проблему длительного нахож-

дения детей у экрана [124]. Хотя такого рода инструменты не подразумевают написание кода на компьютере / планшете, они развивают навыки вычислительного мышления творчески, привлекая детей к созданию и программированию роботов, настольным играм и другим занятиям без гаджетов. Большинство инструментов такого рода, доступных на рынке, представлены в виде учебных комплектов для занятий робототехникой. Чтобы управлять поведением робота, ребенок использует «физическое программирование», например, совмещает деревянные / пластиковые блоки, которые управляют физическим роботом или физическими кнопками на роботе. Физическое программирование не такое сложное, как программирование учебных роботов с помощью кода на компьютере, поэтому оно больше подходит детям младшего возраста. С помощью внеэкранных учебных комплектов дети могут создавать контент разными способами. Например, комплект Kibo, разработанный группой исследователей DevTech в Университете Тафтса штата Массачусетс, США, оснащен несколькими блоками для разработки креативного контента, например, «Танцы мира». В этом блоке дети программируют свои KIBO на исполнение любимого танца любой страны мира [12]. Другим видом безкомпьютерных продуктов являются настольные игры, направленные на развитие вычислительного мышления. В этих играх необходимо перемещать объект по доске для достижения определенной цели. Например, в игре Robot Turtles (http://www.robotturtles.com/) карточку с черепашкой перемещают по доске, чтобы добраться до «сокровища», используя карты с командами, например, поворот направо / поворот налево / ход вперед. Существует множество способов переместить черепашку из одной точки в другую, поэтому ребенок должен мыслить креативно, чтобы разработать стратегию.

Двумя яркими примерами развития вычислительного мышления без гаджетов являются CS Unplugged с бесплатными учебными занятиями для изучения компьютерных наук в виде игр и головоломок с использованием карт, строк и цветных карандашей и книга «Hello, Ruby» с занятиями для детей, обучающими основным концепциям вычислительного мышления, таким как установление последовательности, разбиение задачи и распознавание паттернов http://www.helloruby.com/about.

3.3.3. Аспект 3. Ремикс и сообщество

Философия ремиксов

Одним из важных феноменов эпохи цифровой связности является появление и последующее стремительное развитие понятия открытых инноваций — приветствуется обмен новыми идеями, а не утаивание идей в интересах оригинального разработчика. Возможность повторно использовать работу другого человека стала залогом успеха программного обеспечения с открытым исходным кодом, Википедии, открытого образования, открытых данных и многих совместных инициатив по сочинению музыки, историй и созданию фильмов. Примеры проектирования с целью многократного использования кода можно найти в таких областях, как инженерия, образование, графический дизайн, музыка, видеоигры и т. д. В современной культуре многие авторы акцентируют внимание на новых возможностях при изменении цифровых объектов и создании разнообразных литературных, музыкальных, видео и программных ремиксов [125–127]. Э. Навас назвал ремикс переосмыслением уже существующей песни, имея в виду, что «удивительная аура» оригинала будет преобладать в переработанной версии.

Культура ремиксов оказала существенное влияние на образование и новые образовательные стратегии [128–130]. Стремительно увеличивается количество, улучшается качество и доступность цифровых объектов для повторного использования в образовательных целях. Одной из наиболее перспективных технологий среди обучающих инструментов нового поколения является технология повторного использования цифровых объектов. Цифровой объект — это объект, состоящий из упорядоченной последовательности байтов, которой присвоено имя, уникальный идентификатор и атрибуты, описывающие его свойства. Цифровые объекты обычно

определяют как редактируемые, интерактивные, открытые, дублируемые и распределенные [131]. Возможность повторного применения означает, что цифровой объект можно повторно использовать в новом учебном контексте, и его не нужно привязывать к дисциплине или предмету, для которого он был создан. Это позволяет креативно использовать основные цифровые объекты, перенося их из одной области в другую. Одни и те же цифровые объекты — рисунки, звуки, фотографии и текстовые описания — можно включить в детские энциклопедии и отзывы учеников летних школ, использовать в качестве начальных блоков для мультимедийных цифровых историй в среде Scratch. П. Бликштейн отмечает, что идея продуктивной деятельности (цифрового производства) как способности учеников создавать не только собственные компьютерные программы, но и другие цифровые объекты, постепенно выходит за пределы среды обучения ИТ и находит свое отражение в других образовательных средах [132]. Простые в использовании и легко создаваемые цифровые объекты открывают новые горизонты для образования с применением творческих способностей и открытий.

Что касается российских разработок, связанных с творческими концепциями в компьютерных средах, следует упомянуть вклад В.Ф. Турчина, описавшего в своей книге «Феномен науки. Кибернетический подход к эволюции» [133], как информационные технологии, наука и механизация производства устанавливают все более высокие стандарты творческой деятельности, и М.Ю. Лотмана, чей значительный эпистемологический вклад включал понятие семиосферы и механизма формирования нового контента [134]. Согласно модели Лотмана, в рамках которой творчество и смыслообразование рассматривается как феномен коммуникации, приращение знаний, происходящее при переводе из одного жанра в другой, из одной области знаний в другую, очень похоже на концепцию ремикса.

Как создавать ремиксы и использовать блоки повторно?

Сравнение образовательных сообществ, близких к конструкционистской теории, показывает, что практически все они используют идею цикла или спирали действий, выполняемых участниками с объектами сотрудничества. Scratch, Gamestar Mechanic, Agentcubes, Kodu, Pocket Code, NetLogo, StarLogo (TNG & Nova), Globaloria, Greenfoot, Alice (Looking Glass) создают сайты с открытым исходным кодом, на которых молодые люди могут поделиться своими разработками и внести свой вклад. Эти сайты воспользовались опытом MOOSE Crossing, который был в то время уникальным ресурсом, с помощью которого дети изучали программирование в сетевом сообществе.

MOOSE Crossing — это текстовый виртуальный мир, где дети сообща создают пространство, в котором они взаимодействуют. В MOOSE Crossing дети придумывают новые объекты и пишут программы, чтобы управлять их поведением [104]. Эми Брукман внесла несколько изменений в язык для MOOSE, основываясь на многолетнем опыте работы в Logo и своем стремлении сделать язык доступным для начинающих программистов [135] Она сосредоточилась на принципах создания эффективных онлайн-сообществ:

- Максимизируйте возможности творческого самовыражения и активного участия каждого человека.
- Исходите из того, что обычные люди умнее и креативнее, чем вы думаете.
- Поощряйте пользователей создавать контент и поддерживайте качество, применяя минимальный набор стандартов сообщества.
- Разработайте инфраструктуру для поддержки сообщества в процессе обучения.

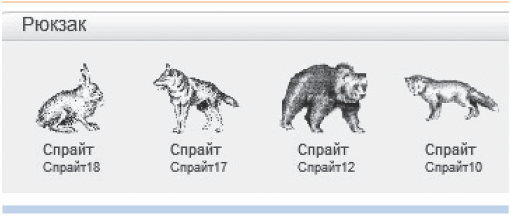
В рамках исследований MOOSE Crossing были определены способы, которыми сообщество может поддерживать процессы проектирования и конструирования. Одной из мотиваций создания MOOSE Crossing было вовлечение детей в другой образ мышления [136]. По мнению Резника, «коллекция примеров MOOSE Crossing заметно отличается и является более экологичной».

Члены MOOSE Crossing могут «заглянуть внутрь» объекта, чтобы увидеть компьютерный код, обуславливающий поведение, создать новую версию (ремикс) объекта со слегка измененным кодом.

Scratch: в сети социального обучения Scratch [137] учащиеся выполняют следующие действия со Scratch-проектами: Представь -> Создай -> Играй -> Поделись. Для успешного участия в субъектно-ориентированном взаимодействии в сетевой среде субъекту совместной деятельности необходимо уметь оценивать и обсуждать объекты, созданные другими субъектами совместной деятельности; принимать оценки и суждения других участников совместной деятельности; отслеживать действия других субъектов; классифицировать объекты, созданные другими субъектами совместной деятельности. Все эти навыки свидетельствуют о социальной компетентности субъекта совместной сетевой деятельности. Ясмин Кафай использует термин «вычислительное участие», подчеркивая что «объекты, помогающие думать» становятся «объектами для совместного использования» [4]. Поскольку код выступает как социальный объект, программирование из когнитивного навыка вычислительного мышления превращается в социальный и культурный навык, необходимый для участия в совместной сетевой деятельности. Создать ремикс в Scratch просто. У Scratch есть сайт сообщества, на котором пользователи могут выкладывать свои проекты. Когда скретчеры открывают проект на сайте и «заглядывают» в основной код, они сразу могут создать ремикс с помощью кнопки в верхнем правом углу. После нажатия на кнопку проект повторно размещается на сайте в качестве ремикса. Карен Бреннан отмечает, что пользователи часто сталкиваются с ремиксами как со способом изучения [46].

Эффективное участие во всех учебных сообществах означает, что ученики самостоятельно распределяют обязанности между членами проектной команды. В настоящее время в сообществе Scroll доступны следующие основные инструменты для совместной работы: студия для сбора и хранения проектов и проектных материалов; возможность сделать копию проекта другого участника и дополнить / улучшить копию; и возможность делать копии отдельных спрайтов и скриптов из чужих проектов в отдельном пространстве и использовать эти объекты в собственном проекте (рис. 3.12).

Рисунок 3.12. Спрайты в рюкзаке



Gamestar Mechanic. В Gamestar Mechanic игроки становятся «игровыми механиками» — персонажами, которые попадают в фантастический мир, где экономика, культура и образ жизни зависят от функционирующих игр. В Галерее игр игроки могут просматривать или комментировать игры друг друга и писать полезные критические отзывы в рамках итерационного проектирования (рис. 3.13). У этих обзоров есть рейтинги и простые заголовки для управления обратной связью. Игроки могут поделиться игрой, прописав для нее ссылку, отправив по электронной почте, разместив на другом сайте или добавив в избранное. Общий доступ к игре делает ее доступной даже для пользователей, не авторизованных в Gamestar Mechanic. Игроки могут показывать свои игры родителям и друзьям, даже если у них нет аккаунтов в Gamestar Mechanic. Игры в разделе «Шаблоны» можно скопировать себе с предварительно созданной конфигурацией, которую вы можете отредактировать по своему усмотрению.

Рисунок 3.13. Галерея игр

AgentCubes — это обучающий язык программирования для детей, позволяющий создавать 3D и 2D онлайн-игры и модели. Пользователи могут играть, проектировать или редактировать существующий проект. Проекты-ремиксы существуют в среде AgentCubes в виде клонов.

Kodu — это новый язык визуального программирования, созданный специально для разработки игр. Kodu поддерживает полный цикл создания ремиксов, включая загрузку существующей программы, ее изменение и повторное совместное использование [138,139].

Pocket Code — это обучающее приложение для мобильных устройств. Это приложение позволяет подросткам создавать собственные игры и множество других приложений. Pocket Code очень похож на язык программирования Scratch [140,141]. Для использования инструмента не требуется опыт программирования, поэтому он подходит новичкам для изучения процессов выполнения программ и понятий логики. В Pocket Code приветствуется создание ремиксов и, как и в Scratch, можно посмотреть на дерево ремиксов.

NetLogo. В NetLogo Commons разработчики моделей совершают с ними следующие действия: Создай -> Запусти -> Поделись -> Прокомментируй -> Измени -> Создай варианты [142]. Автор модели может пригласить других участников, чтобы изменить свою модель совместными усилиями. При этом участники могут выполнять различные роли в команде.

StarLogo (TNG & Nova). В сети StarLogo TNG существуют два взаимосвязанных цикла действий, которые разработчики совершают с моделью как с объектом деятельности.
Цикл исследования: Наблюдай / Собирай данные -> Формулируй вопросы -> Проверяй / Экспериментируй.
Цикл проектирования: Проектируй -> Создавай -> Проверяй / Экспериментируй -> Проектируй [109].

Globaloria. В сети Globaloria учащиеся выполняют действия с компьютерными играми: Играй -> Планируй -> Разрабатывай прототип -> Программируй -> Публикуй [109]. Созданный объект можно обсудить на специализированном вики-портале и использовать в дальнейшем при создании программ других участников.

Greenfoot — это среда изучения объектно-ориентированного программирования [143]. У Greenfoot есть традиционный онлайн-форум и галерея Greenfoot — общедоступный сайт, на котором можно размещать Greenfoot-проекты. Любой пользователь Greenfoot может выложить свои воспроизводимые Greenfoot-проекты. Опубликование проекта может включать полный исходный код проекта по усмотрению автора проекта. Опубликование исходного кода поддерживает идею обучения на примере.

Looking Glass. В сети Looking Glass создатели 3D-моделей совершают следующие действия с моделями Alice: Создай -> Анимируй -> Выдели модуль -> Поделись [144]. Платформа позволяет создать ремикс поведения отдельного объекта/персонажа и затем передать это поведение другому персонажу. Looking Glass обеспечивает поддержку пользователей, чтобы они могли связать между собой результат и код, который привел к такому результату. Пользователи могут просматривать сайт и добавлять в закладки понравившиеся программы, после чего открывается возможность внедрять ремиксы этих программ в свои программы в интегрированной среде разработки с использованием интерфейса создания ремиксов.

Ремиксы относятся к компоненту «Коммуникация и связь» концепции позитивного развития технологий. Представленные на рынке инструменты облегчают коммуникацию пользователей четырьмя различными способами:

a) Дают возможность детям делиться / показывать свои проекты. Эти платформы позволяют детям делиться своим контентом с другими пользователями в Интернете. Совместное использование проектов может способствовать развитию компетенции ребенка, а просмотр других проектов может стать подспорьем в генерировании идей.

b) Дают возможность детям создавать ремиксы совместно используемых проектов. Как упоминалось ранее, с помощью ремиксов пользователи могут создавать новые проекты на основе существующих проектов, которыми другие пользователи поделились в сообществе. Создание ремиксов — это следующий шаг в совместном использовании проектов. Они позволяют детям взаимодействовать с проектами других пользователей.

c) Дают возможность детям задавать вопросы и делиться своими идеями на традиционных текстовых форумах / порталах сообщества. Дети используют текстовые форумы в качестве ресурса, с помощью которого можно задавать вопросы и отвечать на вопросы других пользователей. Форумы / порталы сообщества могут поддержать детей в стремлении продолжить изучение программирования, если они столкнулись с проблемами, которые не могут решить самостоятельно.

d) Дают возможность детям взаимодействовать друг с другом на самом активном уровне. У некоторых инструментов есть интерактивные онлайн-сообщества, где дети могут комментировать общие проекты, ставить им лайки или добавлять их в избранное. Интерактивные онлайн-сообщества обеспечивают максимальное взаимодействие пользователей. Хотя возможность делиться проектами и создавать ремиксы также способствует взаимодействию, она не гарантирует высокого уровня взаимодействия в сообществе.

Внеэкранные инструменты не были упомянуты ни на одном из четырех уровней взаимодействия с сообществом, поскольку невозможно обеспечить взаимодействие с детьми, которые физически находятся в разных местах и используют внеэкранные инструменты.

3.3.4. Аспект 4. Поддержка учителей / родителей

Дети и преподаватели могут использовать большинство инструментов, представленных на рынке, в разных средах с разными уровнями свободы воли и порядка. Бреннан определяет свободу воли как «способность ученика определять цели обучения и достигать их, что позволяет ему играть определенную роль в саморазвитии, адаптации и самообновлении с учетом требований времени» и определяет порядок как «явные и предполагаемые правила, роли и ресурсы в учебных средах» (K. A. Brennan 2013, p. 24). Многие продукты в области программирования / вычислительного мышления используются как дома детьми для самостоятельного изучения (среда с высоким уровнем свободы воли и низким уровнем порядка), так и учителями в рамках школьного образования для обучения учеников (среда с низким уровнем свободы воли и высоким уровнем порядка).

Известно, что слишком строгий порядок может ограничить свободу воли ученика; однако не следует забывать, что отсутствие порядка не равно свободе воли. На данном этапе нам стоит рассматривать свободу воли и порядок не как отдельные, а как взаимодополняющие понятия. Так мы сможем понять, как совместить школьные и неформальные занятия. Таким образом, значительная часть доступных на рынке инструментов дают учителям возможности использовать их наиболее эффективно на своих занятиях. Чтобы добиться идеального сочетания свободы воли и порядка, учителям нужна поддержка для переосмысления своей роли в ключе реализации изменений, которые приносит XXI век, включая классную, институциональную и эмоциональную поддержку.

Классная и институциональная поддержка, которую обеспечивают учителям инструменты для изучения программирования, может проявляться в разных формах:

- **Планы уроков и занятия** — наиболее распространенные ресурсы, предоставляемые учителям. Они помогают использовать инструменты для обучения программированию / освоения вычислительного мышления во время занятий с детьми в классе. Некоторые платформы предлагают множество ресурсов, разделяя их по опыту и потребностям учителей. Например, Scratch-платформа для преподавателей ScratchED предоставляет огромное количество ресурсов для преподавателей, исходя из этапа обучения, типа контента, учебной программы и языка (нет определения ScratchEd). При этом некоторые платформы предлагают очень практичный «комплекс» для учителей с пошаговыми инструкциями для использования в любом классе. Например, Stencyl [110] предоставляет бесплатный учебный комплекс, включающий уроки, рассчитанные на две недели, занятия для учеников, финальный проект и руководство по установке: http://community.stencyl.com.

- **Онлайн-курсы для обучения преподавателей использованию инструмента**. Некоторые компании-разработчики инструментов организуют онлайн-курсы, в основном бесплатные, для обучения учителей. Например, у Kodu есть онлайн-курс «Введение в Kodu» для учителей, являющихся членами сообщества для педагогов Microsoft Educator Community.

- **Программы аттестации для преподавателей**. Хотя такие платформы не получили широкого распространения, некоторые платформы, используемые для углубленного изучения программирования, организуют курсы и программы аттестации для учителей. Например, Stencyl проводит три различных аттестационных экзамена для преподавателей и школ, желающих разработать упорядоченные занятия по гейм-дизайну и программированию. Выдается четыре свидетельства: специалист Stencyl по гейм-дизайну, специалист Stencyl по программированию игр, эксперт Stencyl по гейм-дизайну и эксперт Stencyl по программированию игр. Еще одним примером является программа Blue Ribbon Educator на платформе Tynker. Программа ежегодно отбирает определенное количество препо-

давателей для проведения эксклюзивного обучения, чтобы помочь им стать экспертами в области программирования, и выдает свидетельство о повышении квалификации (https://www.tynker.com/school/training/).

- **Педагогические семинары**. Некоторые разработчики инструментов программирования проводят индивидуальные обучающие семинары по запросу. Игровая платформа для детей Gamefroot, разработанная новозеландской компанией, проводит семинары для учителей, посвященные использованию ее возможностей в образовательной среде и включению Gamefroot в планы уроков.

- **Инструменты организации учебного процесса**. Инструменты организации учебного процесса, как правило, платные, но они предоставляют механизмы оценивания, с помощью которых учителя могут отслеживать успеваемость учеников, сравнивать успехи отдельных учеников со средними показателями класса и показателями других учеников при необходимости. Tynker, Codemonkey и Gamestar Mechanic предоставляют некоторые инструменты организации учебного процесса и учебные материалы для преподавателей.

Помимо описанных выше ресурсов классной и институциональной поддержки, некоторые платформы также предлагают «эмоциональную поддержку» учителям, создавая сообщества. Эти сообщества, как правило, делятся на два типа:

- **Форумы сообществ для учителей**. Такие форумы дают возможность преподавателям и учителям делиться историями, обмениваться ресурсами и задавать вопросы. Scratch, Stencyl, MIT Inventor и Greenfoot являются одними из инструментов с форумами сообществ для учителей.

- **Почтовые группы для учителей**. Почтовые группы можно использовать в качестве альтернативы форумам сообществ, в рамках которых учителя могут задавать вопросы и общаться друг с другом. Например, у Alice и Bitsbox есть почтовые группы для учителей.

Наряду с поддержкой преподавателей некоторые инструменты для программирования также обеспечивают поддержку для родителей. Они, как правило, помогают родителям следить за успехами своих детей. Среди инструментов, которые мы изучили, Tynker предоставляет Parent Dashboard, где родители могут просматривать успехи своих детей, созданные ими проекты и освоенные понятия. В Parent Dashboard родители могут даже делиться проектами своих детей с друзьями и членами семьи.

3.3.5. Аспект 5. Особые соображения

Четыре перечисленных аспекта помогают нам понять основные характеристики продуктов, предназначенных для развития навыков программирования и вычислительного мышления у детей на практике. Кроме этого, мы обнаружили некоторые «особые соображения», которые учителя, родители и педагоги должны учитывать при выборе подходящего инструмента для своих учеников или детей, о чем пойдет речь дальше.

Доступ

Чтобы дети могли использовать технологические инструменты в разных средах, в том числе дома, в школе и в рамках неформального образования, эти инструменты должны быть доступны с точки зрения их стоимости и совместимости с множеством широко используемых аппарат-

ных средств. Поэтому аспект доступа включает два соображения: 1) является ли платформа общедоступной или требуется платное членство; и 2) нужно ли особое аппаратное / программное обеспечение для работы инструмента или он совместим с общедоступными инструментами.

В Руководстве по выбору продуктов, представленном в Приложении, указана информация о том, какие инструменты являются бесплатными / содержат открытый исходный код, а какие инструменты требуют плату за доступ. Некоторые инструменты отмечены в Руководстве как «бесплатные» и «платные». Эти инструменты условно-бесплатны, то есть они предлагают ограниченный контент бесплатно, при этом за расширенный контент придется заплатить.

С точки зрения совместимости с аппаратным и программным обеспечением, мы проверяли, будет ли инструмент работать на любых аппаратных средствах (например, на ноутбуке, настольном компьютере, планшете, телефоне) и с любыми программными системами (iOS, Windows, Linux и т.д.). Большинство инструментов совместимы с любым аппаратным / программным обеспечением, однако некоторые работают только на определенной платформе. Например, приложение Hopscotch доступно только для iPad и iPhone.

Доступность на русском языке

Некоторые изученные инструменты были доступны на нескольких языках, включая русский. Поскольку основной целью этого исследования был вклад в развитие навыков кодирования у детей в России, мы рассмотрели «доступность на русском языке» отдельной задачей. Кроме того, мы изучили инструменты, которые были изначально разработаны в русскоязычных странах. На момент написания этой книги из всего списка продуктов многие из инструментов, изначально разработанных на английском языке, были также доступны на русском языке. Это дает возможность использовать в России целый ряд инструментов в дополнение к 20 продуктам, которые уже были разработаны в России или русскоязычных странах.

Гендерный аспект

Гендерный разрыв в сфере точных наук вызывает беспокойство во всем мире. Остро стоит проблема недостаточной представленности девочек в образовании в области точных наук, которая ограничивает прогресс в достижении целей устойчивого развития [145]. Для решения этой проблемы некоторые исследователи и компании, разрабатывающие учебно-образовательные технологии, начали разрабатывать инструменты, чтобы сделать программирование интересным для девочек. Например, исследователь Кейтлин Келлехер и ее команда разработали новую версию Alice — Storytelling Allice — для анализа гендерного аспекта инструмента [144]. В Storytelling Alice программирование представлено как способ завершить историю. Среда была разработана таким образом, чтобы истории вызывали отклик у учениц средней школы. При этом просто Alice представляет конечной целью программирование и дает возможность пользователям писать программы, которые управляют движениями объектов в 3D-среде.

В ходе экспериментального исследования в США с участием 88 учениц средней школы выяснилось, что девочки, которые использовали Storytelling Alice, уделяли программированию на 42% больше времени, чем девочки, которые использовали просто Alice, и в три раза чаще уделяли программированию дополнительное время. Результаты исследования показали, что акцент на повествовании сделал обучение программированию более интересным для учениц средней школы [144].

Учитывая потенциальное влияние дизайна на привлекательность инструментов для девочек, в рамках гендерного аспекта можно выделить инструменты, ориентированные на девочек. В ходе исследования мы определили несколько инструментов, предназначенных для девочек, преимущественно в англоязычных странах, в основном в США; при этом в России мы пока не

нашли инструменты, разработанные с учетом гендерных аспектов. Согласно данным Международного исследования в области компьютерной и информационной грамотности 2013 года, которое является единственной доступной международной оценкой успеваемости учеников в области компьютерной и информационной грамотности, в России восьмиклассницы получили более высокие оценки в сфере компьютерной и информационной грамотности, чем их одноклассники. При этом их уверенность в продвинутости навыков в сфере ИКТ была ниже, чем у мальчиков [146]. Соответственно, недостаток ресурсов для программирования, предназначенных для девочек, может стать проблемой, которую необходимо решать в российской системе образования, чтобы стимулировать девочек изучать программирование и повышать их уверенность в своих знаниях.

Из 71 инструмента, которые мы рассмотрели на англоязычном рынке, семь (около 10%) инструментов разработаны специально для того, чтобы стимулировать девочек изучать программирование. Наиболее известными из них являются Storytelling Alice, Looking Glass, Pocket Code. Однако на российском рынке мы не смогли найти инструменты для программирования, которые бы явно учитывали гендерный аспект.

Возрастной аспект

Дункан и его коллеги составили список из 47 инструментов, которые можно использовать для обучения программированию, и назвали эти инструменты начальными средами обучения (НСО). Они разделили НСО по уровням от 0 до 4 [147].

Уровень 0 — от 2 до 7 лет. Принцип drag-and-drop или простые решения. Обучает только планированию (последовательности). Не требует абстрагирования. Не прибегает к широкому использованию: функций, переменных, итераций, индексированных структур данных, условного выполнения. Примеры: Daisy the Dinosaur, Lightbot Junior, Scratch Junior, ПиктоМир.

Уровень 1 — от 5 до 10 лет. Принцип drag-and-drop. Не требует абстрагирования (или незначительное абстрагирование). Не содержит или содержит несколько: функций, переменных, итераций, структур индексируемых данных, условных операторов. Примеры: Scratch Junior, ПиктоМир, Lightbot, Kodu, Blockly, HopScotch, Scratch, Tynker.

Уровень 2 — от 8 до 14 лет. Принцип drag-and-drop или текстовые управляющие конструкции. Требуется некоторое абстрагирование. Содержит несколько или большинство: функций, переменных, итераций, структур индексируемых данных, условных операторов. Примеры: Kodu, Blockly, HopScotch, Scratch, Tynker, NetLogo, Alice, AppInventor, Snap!, Looking Glass.

Уровень 3 — 12 лет и старше. Принцип drag-and-drop или текстовые управляющие конструкции. Требуется абстрагирование. Содержит все: функции, переменные, итерации, структуры индексируемых данных, условные операторы. Примеры: Scratch, Tynker, NetLogo, StarLogo Nova, Snap!, Alice, AppInventor, Looking Glass, StarLogo Nova, BlueJ, KidsRuby.

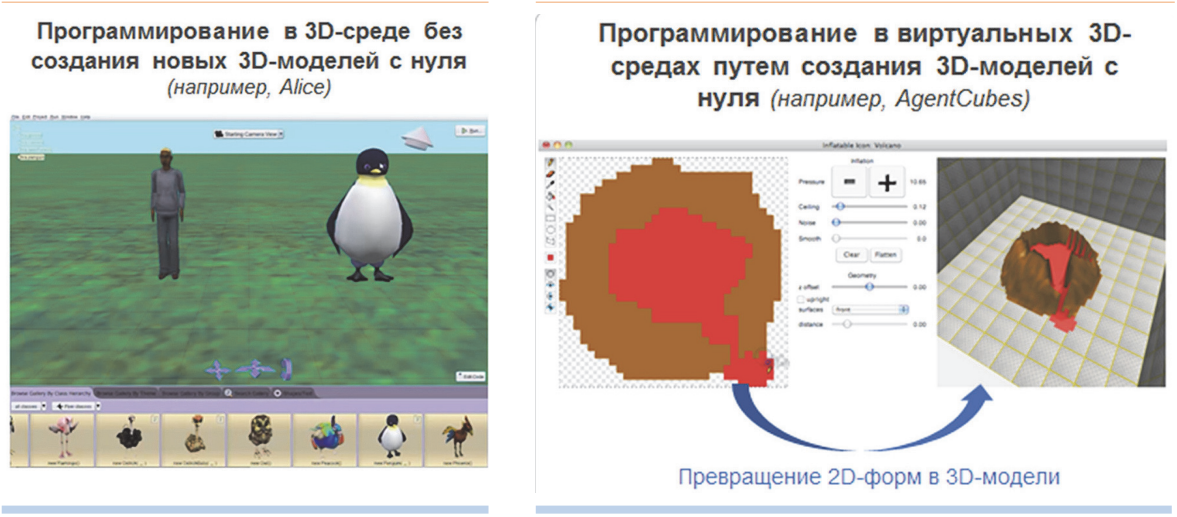
Инструменты, которые могут улучшить владение 3D-технологиями

3D-анимация помогает детям визуализировать объекты и осмысливать реальность, позволяя им «увидеть» написанный код [148]. Несмотря на это дополнительное преимущество 3D-анимации, 3D-объекты и среды, включенные в инструмент программирования, могут стать дополнительной проблемой для детей, поскольку они начинают переживать о внешнем виде, размере, ориентации и расположении объектов в 3D-пространстве или о том, как анимировать объекты в движении [95]. 3D-среды, как правило, повышают уровень сложности процесса.

- *Первый уровень* дает возможность детям программировать в виртуальных 3D-средах без использования встроенных инструментов моделирования для создания новых 3D-моделей с нуля. Например, Alice позволяет начинающим программистам создавать анимации и игры в 3D-мире без возможности создавать собственные 3D-модели.

- *Второй уровень* позволяет пользователям создавать виртуальные 3D-среды / игры путем создания 3D-моделей с нуля. Например, AgentCubes является ведущим инструментом на рынке, с помощью которого пользователи могут превращать 2D-объекты в 3D-объекты. Таким образом дети могут поэтапно переходить от 2D- к 3D-приложениям. Несмотря на увлекательность и позитив этого уровня, ученики могут слишком увлечься созданием 3D-форм в ущерб программированию [95,96].

Рисунок 3.14. Два уровня владения 3D-технологиями, поддерживаемые инструментами программирования



Поддержка генерации идей

Как уже упоминалось в описании компонента «Креативность и уверенность» концепции позитивного развития технологий, ученики, которые делают первые шаги в кодировании, обычно не представляют, что они хотят создать, какие у них есть возможности или насколько сложным может быть программирование. Чтобы избежать разочарований, некоторые инструменты оказывают дружескую поддержку для генерации идей в виде шаблонов или готовых игр. Поскольку это соображение может иметь существенное значение, особенно для начинающих программистов, поддержка генерации идей была включена в число особых соображений в нашем Руководстве по выбору продуктов.

3.4. Руководство по выбору продуктов

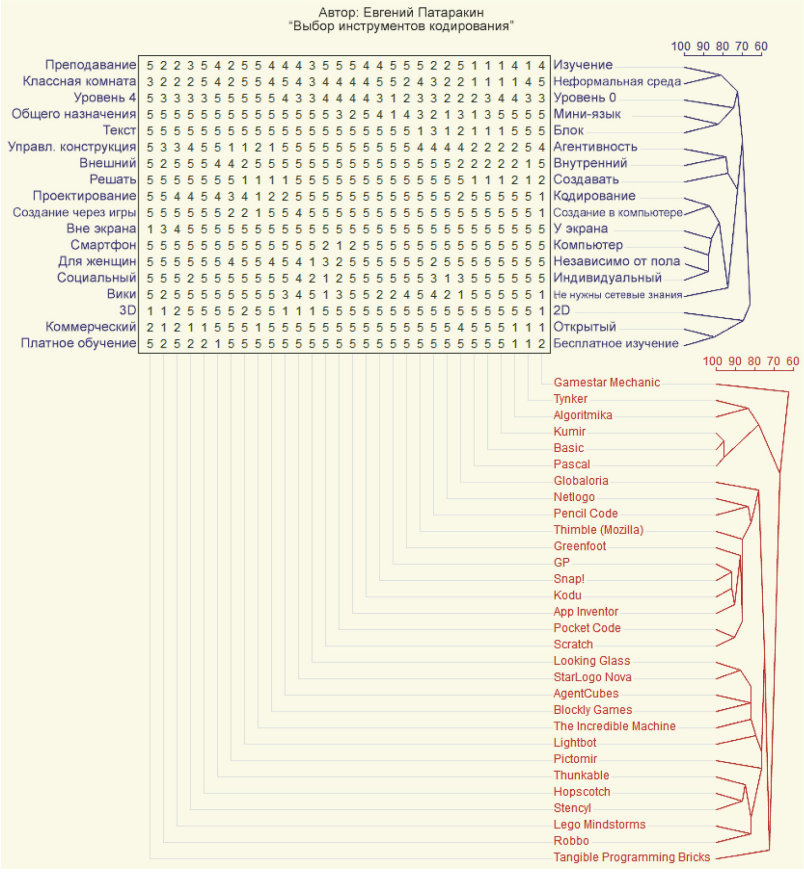
В Руководстве по выбору продуктов, представленном в приложении, все продукты / инструменты указаны с учетом пяти аспектов. Список продуктов не является полным или исчерпывающим. Рынок продуктов для программирования и вычислительного мышления весьма динамичен, поскольку для выхода на этот рынок нет никаких преград. Благодаря оперативной обратной связи, получаемой от соцмедиа, некоторые продукты очень быстро становятся популярными. Изучив такие сайты для финансирования стартапов, как http://www.kickstarter.com и http://www.indiegogo.com, можно выделить текущие кампании по сбору средств для продуктов,

направленных на обучение детей программированию и вычислительному мышлению. Некоторые из этих продуктов получают больше средств, чем требуется, другие не могут собрать достаточное количество средств. В Руководстве по выбору продуктов, представленном в Приложении 3, наглядно представлены виды продуктов, существовавшие в 2018 году, и предлагаемые функции. Заинтересованные читатели могут также ознакомиться с отличным обзором бесплатных продуктов для программирования и занятия робототехникой, доступным в Интернете по адресу https://www.intechopen.com/books/simulation-and-gaming/an-evaluation-of-open-digital-gaming-platforms-for-developing-computational-thinking-skills [149]. Документ, представленный на конференции «Интерактивный дизайн для детей» в 2018 году https://dl.acm.org/citation.cfm?id=3202738 [150], был разработан для детей в возрасте до 7 лет.

3.4.1. Представление многообразия предложений на рынке с использованием репертуарной решетки

В этом разделе представлена экспериментальная концептуальная карта (рис. 3.15) рынка продуктов для программирования. Чтобы эта концептуальная карта была читаемой, мы сократили список до тридцати продуктов, информация о которых доступна в Интернете, представлена в рекламных материалах разработчиков или получена в ходе научных исследований, связанных с использованием этих продуктов. Карта была составлена в целях ознакомления, чтобы предоставить исследователям возможность проводить дальнейшую работу по отслеживанию изменений на этом динамичном рынке.

Рисунок 3.15. Анализ тематического кластера инструментов для программирования



Метод, используемый для создания концептуальной карты, основан на технике репертуарных решеток, разработанной Джорджем Келли в 1950-х годах в рамках теории личностных конструктов, которая связана с психологией конструкционизма [151]. Концептуальные решетки первоначально использовались как метод представления концептуальной модели определенной области путем выявления биполярных конструктов, посредством которых индивид дифференцирует значимые элементы в этой области. Применение репертуарных решеток стало популярным с появлением доступного программного обеспечения, в рамках этого раздела мы использовали программное обеспечение WebGrid5 [152]. Метод Келли широко применяется в сфере стратегического менеджемента. Репертуарные решетки признаны эффективным инструментом для получения знаний от специалистов в определенной области. В настоящее время такие решетки используют в различных областях, от клинической психологии до маркетинга.

Фокусный алгоритм репертуарных решеток сортирует строки и столбцы решетки, сближая похожие элементы и конструкты, а также отражает иерархию сходств, полученную в результате такой сортировки (рис. 3.15). Существует восемнадцать биполярных подаспектов или конструктов, по которым каждый из тридцати продуктов был оценен по шкале от одного до пяти на основе информации, собранной в процессе составления Руководства по выбору продуктов. Фокусный алгоритм определяет приближенные конструкты на основе корреляции оценок [153]. Кластеры конструктов (обозначены синим цветом вверху) на рисунке 3.15 разделены на четыре основные группы. Существует связь между «Внутренней мотивацией», «Агентивностью» и «Созданием» по отношению к «Внешней мотивации», «Управляющей конструкции» и «Решению задач». Четвертый кластер указывает на интуитивно угадываемую связь между «Открытым программным обеспечением» и «Бесплатным изучением» по отношению к «Коммерческому программному обеспечению» и «Платному обучению». Кластеры продуктов (обозначены красным цветом внизу рисунка 3.15) также разделены на четыре группы.

3.4.2. Представление сред обучения с использованием Semantic MediaWiki

Расширение Semantic MediaWiki позволяет добавлять семантические аннотации к вики-страницам, тем самым превращая вики в семантическую вики. На российском сайте http://letopisi.org пользователи могут создавать новые страницы, содержащие дополнительную информацию о средах обучения, образовательных практиках, образовательных стандартах и других темах. Описания учебных сред для обучения программированию представлены в виде отдельных вики-страниц в категории «Среда обучения» — http://letopisi.org/index.php/Category:LearningEnvironment.

Описание языков программирования было отредактировано с использованием формы, в которой перечислены необходимые свойства. Например, для языка Alice в форме указаны следующие свойства:

Возраст:	14
Область деятельности:	Расширение возможностей учащихся, вычислительное мышление, инновационный дизайнер
Формат:	3D
Цель:	Мини-язык
Визуальный текст:	Визуальные блоки
Открытость:	Открытый
Адрес:	http://www.alice.org/
Предшественники:	
Потомки:	Storytelling Alice, Looking Glass
Сейчас используется:	Да
Создание ремиксов:	Нет

Описание языка программирования с использованием его свойств позволяет пользователям конкретизировать семантические запросы на страницах.[13]

Выводы главы 3. Многообразие предложений на рынке

- **Позитивное развитие технологий (ПРТ): концепция для анализа**. Концепция разработана группой исследователей DevTech под руководством Марианы Умащи-Берс из Университета Тафтса в Бостоне, США и дает представление о связи между технологиями и развитием ценностей и навыков. Концепция подтверждает мнение о том, что продукт или инструмент нельзя полноценно оценить в отрыве от контекста, — от способа использования инструмента зависит его эффективность. Концепция устанавливает взаимосвязь между моделями поведения, например, созданием контента, в результате которых формируются ценные качества, например, уверенность.

- **Инструменты ПТР для детей, среды и помощников**. Группа DevTech разработала комплект из двух инструментов, которые можно использовать для определения степени задействования концепции ПРТ в определенном контексте в рамках исследований по данным наблюдений. Преимущество использования этого вопросника, представленного в приложении, состоит в том, что его структура уже была утверждена в рамках программы практических исследований, но проведение экспериментов и модификация могут быть полезны при условии, что будут проведены аналогичные исследования. Вопросник, касающийся детей, предназначен для определения того, как часто «дети наблюдают и / или используют работы друг друга» или «дети играют друг с другом или разговаривают друг с другом». Вопросник для условий деятельности предназначен для определения того, в какой мере «организация пространства способствует совместной работе детей над одним проектом». Независимо от выбранной платформы или продукта, концепция ПРТ позволит определить правильность применения — следует отметить, что во время традиционных занятий в классах детям нельзя наблюдать и / или использовать работы друг друга.

- **Пять аспектов оценки продуктов и онлайн-баз данных продуктов**. Мы используем пять аспектов, ориентированных на специалистов-практиков: (i) визуальное, текстовое или смешанное программирование; (ii) характер контента (сочинение историй, разработка игр, художественные работы и т.д.); (iii) возможности создания ремиксов — особенно важны для привлечения детей, которые уже погружены в то, что Дэвид Шапиро называет глобальной агорой; (iv) поддержка для учителей и родителей; и (v) особые соображения — в том числе наличие русскоязычных версий, учет гендерных аспектов, инклюзивный подход и возможности для мышления в формате 3D-дизайна. Предполагается, что база данных продуктов, представленная в приложении, с соответствующими текстовыми комментариями будет предоставлена в режиме онлайн в качестве базы данных с возможностью поиска.

[13] Семантический поиск — это поиск, основанный на значении, а не обычный поиск по совпадениям слов или структур слов безотносительно значения запроса. Семантический поиск учитывает контекст, например, географию. Синтаксис семантического поиска, используемого здесь, можно изучить по ссылке https://www.semantic-mediawiki.org/wiki/Help:Inline_queries. Например, если пользователь хочет узнать, какие языки лучше всего использовать для обучения детей в возрасте 7 лет дизайну, он может указать следующий семантический вопрос на странице. {#ask: [[Category:Learning Environment]] [[Age::<8]] [[Activity::Creative Designer]]}.

4. КРЕАТИВНОЕ ПРОГРАММИРОВАНИЕ В РОССИИ

4.1. Истоки конструкционизма в Российской Федерации

Андрей Петрович Ершов, дальновидный ученый в области компьютерных наук, имеющий доступ к рычагам власти в Кремле в 1970–80-х годах, выдвинул идею программирования как «второй грамотности», опередив время на полвека (А. Ершов, 1981). Изучение развития науки и научных открытий в бывшем Советском Союзе представляет собой обширную область современных исследований, и изучение мыслей и действий Андрея Ершова является важной частью этой области. Возможно, это связано с высокой производительностью Ершова, его узнаваемостью и контактами с западными учеными, а также его привычкой все записывать [154].
Большая часть литературы, посвященная изучению советской науки, является узкоспециализированной и не совсем актуальна для современной вычислительной науки в России, тем не менее, сперва следует рассмотреть исторический контекст. В доступных материалах об «информатизации в Советском Союзе» подробно описано, как стратегия развития вычислительного мышления у всех учащихся, разработанная Андреем Ершовым, была поддержана Горбачевым на волне перестройки [19]. Г. Афиногенов отмечает значимость развития кибернетики — управления системами, например, общественной системой с плановой экономикой, какой и был Советский Союз. Афиногенов рассказывает, как Ершов сначала пытался привлечь внимание общественности и обеспечить поддержку реформы системы образования, направленной на включение предмета, известного как «информатика», но потерпел неудачу, а затем пошел другим путем:

«Хотя его усилия по продвижению эволюционной модели школьной информатики получили поддержку государственных и партийных властей, к 1985 году Ершову стало ясно, что «инфильтрационная модель», как он ее называл, потерпела неудачу: «Оказалось, что эта стратегия неспособна преодолеть инертность и недостаточность общественного сознания, а также позднее выявленный «тормозной механизм» общественного развития в целом». Теперь требовалась «лобовая атака». Как бы там ни было, в конце 1984 года (после того, как Ершов стал действительным членом Академии) Министерство образования после значительных усилий Ершова и других распорядилось уже в следующем учебном году ввести курс «Основы информатики и вычислительной техники» в девятых и десятых классах всех школ по всему Советскому Союзу. Однако этого было недостаточно: возникла необходимость принять «твердое» и «резкое» «политическое решение». Ершов обратился в Политбюро, которое 9 марта 1985 года приняло постановление об углубленном изучении информатики в школах. Поручение дал сам Горбачев».

- G. Afinogenov, 2013

В своей статье 2014 года Е.К. Хеннер и И.К. Семакин писали, что в 1985 году в 9–10 классах всех школ Советского Союза был введен курс под названием «Основы информатики» с целью развития алгоритмического мышления и компьютерной грамотности учащихся. Были определены следующие составные части курса:

1. Понятие алгоритма и его свойства, средства и методы описания алгоритмов, программа как форма представления алгоритма для компьютеров.
2. Основы программирования на языке программирования.
3. Практические навыки работы с компьютером.
4 Принципы работы компьютера и его основные элементы.
5 Применение компьютеров, их роль в разных сферах деятельности человека.

Источник: [155]

Афиногенов пишет, что Ершов был соавтором учебника, предназначенного для обучения «алгоритмическому мышлению», но на программе сказалась острая нехватка компьютеров в школах, даже в Москве, не говоря уже о сельской местности и менее благополучных республиках. Несмотря на то, что некоторые заявления о всеобщем изучении программирования нашли свое отражение в идеологических терминах общественного контроля и интеллектуального творчества в рамках очень абстрактного представления о взаимодействии человек-машина, с течением времени идеи Ершова претерпели изменения. Если раньше он считал программистов особой группой людей с высоким статусом, ценной для советских предприятий, то, по мнению Афиногенова, появление персональных компьютеров заставило Ершова пересмотреть свои взгляды на концепцию массовой грамотности.

Интересно отметить значимость идей Сеймура Пейперта и компьютерного языка Logo в первые годы обучения детей программированию. Интересные материалы о революционных идеях Пейперта и противодействии, которое они породили, — «оскомина для административных работников системы образования» — можно найти в работе [156]. Пейперт хотел провести революцию в системе образования, внедрив язык Logo, — «грандиозная идея» во многом провалилась в своем первоначальном воплощении, и только с появлением Интернета ученики Пейперта смогли воплотить в жизнь свою идею о детях, использующих компьютер, а не компьютерах, использующих детей. Знания о первых годах обучения детей компьютерному программированию в России показывают, насколько схожим был опыт по обе стороны «железного занавеса». В случае с Россией ситуация усугублялась нехваткой компьютеров в школах.

Идеи и материалы, разработанные Пейпертом, повлияли на образование во всем мире, включая Российскую Федерацию, хотя не исключено, что влияние в России было сильнее, чем во многих других странах. Мы опишем влияние Пейперта в двух аспектах. Первый аспект — это различие между чувством зависимости и чувством субъектности для всех деятелей системы образования. Второй аспект связан с различием между индивидуальной деятельностью и совместной деятельностью.

4.1.1. Черепашка Logo как инструмент контроля учащихся

Анализ текста «Программирование, вторая грамотность», в котором Ершов дважды ссылается на Пейперта, — самый простой способ отследить влияние идей Пейперта на Ершова. В обоих случаях Ершов упоминал Пейперта, когда обсуждал проблему субъектности учащихся. В первом случае Ершов полагался на авторитет Пейперта в подтверждение решительного заявления о том, что «обучение» = «способность что-то сделать» = «программирование».

«Даже обучение, то есть приобретение знаний или, скорее, способности что-то сделать, — это программирование. Лет десять назад профессор Массачусетского технологического института Сеймур Пейперт, один из первых психологов и педагогов, взявший на вооружение концепции программирования, в серии своих работ убедительно показал, что ребенок научается что-то делать только после того, как он поймет, как это делается. Только после выработки такого понимания повторная тренировка достигает успеха. Заметим, что это касается не только программ, представляющих собой цепочки логических реакций на заранее известные стимулы, но и программ реального поведения, включая всяческую моторику (спорт, музыка, игры и т.п.)».

- А. Ершов [7]

Во втором случае Ершов, ссылаясь на Пейперта, указывает на рост субъектности учащихся как положительный побочный эффект от распространения компьютеров.

«Профессор Пейперт предсказывает всестороннее вторжение ЭВМ в мир ребенка, когда машина станет интеллектуальным орудием, применяемым ребенком с той же непосредственностью, с какой он использует перо и карандаш, но с гораздо большим разнообразием. Интерпретируя наблюдения Ж. Пиаже над тем, что ребенок совершает большинство своих интеллектуальных открытий самостоятельно при условии, что окружающий его фон достаточно богат, — профессор Пейперт показал, что компьютеризация этого фона создаст новую, невиданную ранее операционную обстановку, которая потребует новых представлений в психологии развития. В качестве примеров им было приведено значительное ускорение овладения алфавитным языком и более раннее развитие комбинаторных способностей, что позволяет детям овладеть этими фундаментальными умениями практически до вступления в подростковый возраст. Одним из положительных последствий этого изменения может стать преодоление инфантилизма и чувства зависимости, столь характерных для современного городского общества».

- А. Ершов [7]

Ершов писал в первую очередь об ученике как о программисте. Чтобы понять значимость этого сравнения, следует вспомнить глубочайшее уважение, которое Ершов питал к профессии программиста. Ершов писал в 1972 году:

«Применяя метафору троицы, программист ощущает себя и отцом-создателем программы, и сыном-братом этой машины, и носителем святого духа — вложенного в нее интеллекта. Это торжество интеллекта, наверное, самая сильная и самая специфическая сторона программирования».

- (Andrei P. Ershov 1972, 504)

Во втором случае Ершов противопоставляет субъектность ученика чувству зависимости. И здесь он справедливо выделяет субъектность как ядро конструкционизма, потому что в этом отношении продуктивная субъектность сродни конструкционизму [47]. Это также связано с тем, что в культурологии называют «неагентивностью», в русской культуре — это склонность к смирению и покорности, отсутствие средства, выделяющего человека как «деятеля», особенно в материальном, «внешнем» смысле. Неагентивность — это чувство, что люди не управляют своей жизнью и что их способность контролировать события весьма ограничена; склонность к фатализму, смирению, покорности; отсутствие средства, выделяющего человека как самостоятельного субъекта, «деятеля» и контролера событий [157].

Пейперт участвовал в совместном проекте Массачусетского технологического института и компании Bolt Beranek & Newman, в рамках которого был создан управляемый компьютером робот, названный на определенном этапе Черепашка. Знания и опыт в области детской психологии и педагогики помогли Пейперту понять, насколько важен для обучения такой простой объект с предсказуемым поведением. Важно, что простейшие команды были похожи на команды, адресованные человеку, — вперед, назад, вправо, влево, повтори и т.д. Со временем эти простые процедурные эквиваленты кирпичиков Lego образовали более сложные микрокосмы, которые моделировали физические, химические или биологические явления. Но простота и понятность основных команд и процедур сделали мир обучения более доступным.

Язык Logo не был разработан специально для обучения программированию. Однако графика черепашки Logo оказала огромное влияние на формирование направления мини-языков, первым из которых была графика черепашки Logo. У мини-языков скромный синтаксис и простая семантика. Поэтому ученик, даже очень маленький, может овладеть всем мини-языком и использовать его, получая интересные результаты. Второе важное преимущество состоит в том, что мини-языки построены на интересных и визуально привлекательных метафорах. В каком-то смысле все языки с графическим исполнителем являются потомками Logo. Как писал Петр Брусиловский, используя большинство существующих мини-языков, «ученик узнает,

что такое программирование, изучая, как управлять действующим субъектом, которым может быть черепаха, робот или любой другой активный агент, действующий в микромире» [158]. Хотя действующим агентом может быть и физическое устройство, ученик обычно имеет дело с программной моделью такого устройства и наблюдает за поведением исполнителя на экране. Действующий агент может выполнять небольшой набор команд и отвечать на несколько запросов, возвращающих значения. Обычно ученик управляет агентом сначала посредством отдельных команд, затем посредством написания небольших программ на специальном мини-языке программирования.

Несколько мини-языков были разработаны с использованием Logo и широко использовались в СССР, например, Путник, Turingal и Turtoise [101]. Множество учебных проблем было связано с задачами управления простым агентом «Путником», перемещающимся по отмеченному полю со стенами. Формальная концепция исполнителя была разработана российскими исследователями Г.А. Звенигородским и А.Г. Кушниренко. Исполнитель включает агент, действующий в микромире, и небольшой набор команд и запросов к агенту, возвращающих значения. Концепция исполнителя делает возможным использование нескольких агентов в рамках одного языка программирования. Философия КуМир заключается в том, что язык имеет компактное ядро, которое можно динамически расширять путем загрузки одного или нескольких отдельно подготовленных модулей, то есть исполнителей с новыми микромирами и субъектами, или даже новых числовых пакетов и абстрактных типов данных. Такой подход позволяет легко настроить систему с несколькими исполнителями [158].

В странах Восточной Европы появились многочисленные потомки Черепашки Logo. Logo Comenius, мощная по своим возможностям версия Logo, была написана в Словацкой Республике [159]. KIDLOGO, компактный микромир для использования в детских садах и для специального образования, был разработан в Венгрии [160]. Geomland — версия Logo, включающая инструментарий эвклидовой геометрии, была разработана в Болгарии [161]. IconLogo, разработанный доктором Сергеем Сопруновым и его группой в Институте новых технологий в образовании (Москва), включал текстовый процессор, графический редактор и музыкальный синтезатор [162]. Другие российские программы, предназначенные для обучения алгоритмам и программированию, такие как МикроМир, Алгоритмика и Роботландия [163], также использовали концепцию исполнителя.

Как писал Клод Леви-Стросс, «природные виды выбирают не за то, что они «хороши, чтобы есть», а за то, что они «хороши, чтобы с ними думать» [164]. Сегодня мы можем сказать, что все многочисленные потомки Черепашки Logo были созданы не только потому, что их «хорошо программировать», но потому что они «хороши для управления».

4.1.2. Проект Logo как объект для совместной работы и обмена

Дальнейшие усилия Пейперта и его группы (Media MIT Lab) были направлены на перенос управляемого исполнителя Черепашки в компьютерную среду и разработку высокоуровневого языка для обучения — Logo, Logo for Children и LogoWriter. LogoWriter был первым видом потомков Logo, который действительно изменил среду программирования учащихся; пользовательский интерфейс стал более интуитивно-понятным, а пакет был переведен на многие языки (Газдиал, 2004). LogoWriter представлял собой сочетание языка программирования, текстового процессора, графики и анимации. LogoWriter, безусловно, был хорошим инструментом для реализации проектно-ориентированного подхода. В России его продвигал и поддерживал Институт новых технологий в образовании [162].

Интересной особенностью инструмента была возможность делиться работами и объединять страницы LogoWriter в совместный проект. Для дальнейшего обсуждения стоит рассмотреть конкретный пример построения концепций гиперсреды у детей в рамках программы летней

компьютерной школы в 1990 году, который послужит контекстом для сравнения творческого потенциала различных сред. В рамках программы летней школы ученики работали в специальной среде микромира, созданного на языке LogoWriter и призванного помочь детям перейти с простого текста к мультипликационным фильмам. Работая в этой среде, каждый ученик отвечал за создание одностраничной истории об одном из персонажей микромира. Участникам заранее сказали, что персонажи, разрабатываемые другими учениками, смогут посетить те страницы LogoWriter, которые каждый ученик разрабатывал для своего персонажа. В ходе первоначального осуществления этой деятельности было обнаружено, что ученики разрабатывали более интересный и глубокий контент на своих страницах LogoWriter каждый раз, когда речь шла о других персонажах микромиров. С другой стороны, когда страница LogoWriter была посвящена только описанию определенного персонажа — без связи с другими персонажами, — страницы LogoWriter получались какими-то мертвыми без ссылок на другие страницы. В результате изменилось отношение учеников к текстам. Каждый участник стремился обогатить свою историю, включив контент общего проекта, в результате чего ученики стали внимательнее относиться к работе своих сверстников. Такой прием смешения и слияния страниц LogoWriter использовался в «Проекте десять» (1994-1996 гг.). Участники создавали животных с целью сформировать устойчивое сообщество в ограниченном пространстве — на острове, столе или экране компьютера. Творческая деятельность помогла детям получить общее представление об экологическом равновесии и сложных взаимосвязях между различными видами. Каждый участник сделал следующее: нарисовал воображаемый живой организм (растение или животное), слепил животное или растение из пластилина, написал рассказ об отношениях между различными несуществующими видами, сделал страницу LogoWriter с текстом и анимацией [165].

Формирование сетевого сообщества российский преподавателей, использующих в своей практике язык Logo и его разновидности, такие как LogoWriter, Splash LogoWriter, LogoExpress и MSWindows Logo, началось в начале 1990-х годов и было связано с организацией летних компьютерных школ и информационно-образовательной сети «Uchcom», созданной в 1992 году Институтом программных систем Российской Академии Наук. Uchcom специализировался на таких аспектах образования, как физика, математика, когнитивная психология, мультимедиа. Однако Logo Media была самой успешной работой в Uchcom, потому что LogoWriter активно использовался в телекоммуникационных проектах: Космонавт (была реализована идея «чата»), Компьютерный театр (реализована идея телеконференции), Микромир (реализована идея переписки), Фантастический мир (реализована идея упоминания) [166]. Целью проекта было исследование образовательных возможностей телекоммуникаций в среде Logo и создание телекоммуникационной ассоциации русских пользователей Logo. В июле 1993 года участники проекта из разных городов России (Москва, Санкт-Петербург, Череповец, Ярославль, Нижний Новгород, Саранск, Омск, Норильск) приняли участие в двухнедельном семинаре Международной компьютерной школы в Переславле-Залесском. Школа имела телекоммуникационный доступ в Интернет и могла распространять свои материалы через образовательную сеть «ТВ-ИНФОРМ».

Телекоммуникации Logo и LogoNet — это термины, используемые для описания возможности общения групп учеников и преподавателей в разных местах по Интернету через программный интерфейс LogoExpress и LogoNet [167]. Участники сети обменивались статическими текстовыми, графическими и цифровыми видеоизображениями LogoWriter, отформатированными в виде файлов LogoWriter. Таким образом, ученики в Переславле-Залесском могли «общаться» с учениками в Омске и делиться друг с другом историями, графикой и картинками [167].

Logo оказал влияние на российское образование, не только представив отдельный язык программирования и его потомков, но и сформировав международную культуру Logo, в которой можно делиться радостью от обучения, использовать свои ошибки в творческом контексте, работать с учениками и учителями как с партнерами в научно-исследовательской группе. В конце

двадцатого века Черепашка Logo была интересным граничным объектом [168], который объединил людей из разных стран и разных профессиональных сообществ. Из всего множества существующих примеров взаимодействия мы остановимся на двух, любезно предоставленных Е. Сендовой. В начале 1980-х годов во время посещения школы N° 119 в Софии, Болгария, русский педагог и социальный философ Симон Соловейчик попросил Евгению Сендову описать культуру Logo и эксперимент исследовательской группы по образованию одним предложением, не используя слово «компьютер». Она сказала: «Наши дети любят ходить в школу». На что сразу последовал ответ: «Спасибо, этого вполне достаточно» [169]. Еще один великий российский педагог и психолог Василий Давыдов посетил одну из школ, использующих Logo, в городе Благоевград, Болгария, на день раньше, чем было объявлено. Заходя в школу, он улыбнулся и сказал: «Я знаю, что гостю вы покажете самое лучшее, но мне нужна реалистичная картина». Ответ записан ниже:

*«Учительница посмотрела на меня немного удивленно, но я попросила ее следовать плану на день — на уроке они изучали процедуры рисования спиралей с параметрами (для угла поворота черепашки и размера начального сегмента). Ввиду самоподобия конструкции, ученики 5 класса, естественно, создали рекурсивную процедуру. Экспериментируя с разными значениями параметра УГОЛ, ученики нашли интересные паттерны, они смогли создать спирали в форме правильных многоугольников с определенным количеством ветвлений, уходящих влево или вправо. Они изменили процедуру, сначала введя параметр для приращения РАЗМЕРА, затем изменив правило его прибавления (РАЗМЕР * 2). Затем ученики решили проверить, что произойдет, если они оставят РАЗМЕР фиксированным и увеличат УГОЛ. Эта идея появилась благодаря учительнице, которая таким образом готовила почву для экспериментов в науке с процессами, зависящими от нескольких параметров. Профессор Давыдов был искренне удивлен и признал, что он бы подозревал, что работа шла по предварительно подготовленному сценарию, не измени он дату своего визита. Вряд ли он тогда предполагал, что такая творческая атмосфера была обычным явлением на занятиях по языку и математике».*

- Е. Сендова [169]

Историю языка Logo и его потомков, из которых Scratch является наиболее популярным и успешным, можно рассматривать как созданную сеть конструкционизма, число участников которой постепенно растет. Международное сообщество Logo получило информационную поддержку за счет печатных периодических изданий (LogoExchange, LogoUpdate) и дистанционной очной конференции EuroLogo, на которой каждый раз были представлены многочисленные работы из России: Дублин (1987), Гент (1989), Парма (1991), Анависсос (1993), Бирмингем (1995), Будапешт (1997), София (1999), Дублин (2001). Таким образом, появление и распространение языка Scratch было ожидаемым событием в России, к которому многие преподаватели и исследователи были готовы. Прежде чем мы перейдем к описанию дальнейшего развития Scratch, следует изучить контекст, сформированный действующими федеральными стандартами и официальными правилами.

4.2. Информатика: внедрение федеральных стандартов

В 1990-х годах был пересмотрен предмет «Информатика» и его содержание. В это время информатику как основной предмет перенесли из старших классов в средние. Также частично перешли от изучения программирования к изучению работе на ПК в качестве пользователя.

Наконец, в 2010-2012 гг. в Российской Федерации вышла новая редакция Федеральных образовательных стандартов [155]. Согласно этим стандартам, информатика стала обязательным предметом в средних классах, и любая школа вправе ввести информатику в программу старших классов с изучением на начальном или углубленном уровне. В своей статье 2014 года Е. Хеннер и И. Семакин подсчитали, что в то время около 50% старшеклассников изучали инфор-

матику на базовом уровне, и около 10% проходили углубленный курс информатики. В начальной школе некоторые понятия информатики преподают в рамках таких основных предметов, как математика и технология. Кроме того, каждая начальная школа вправе включить информатику в свою учебную программу [155]. По окончании школы число желающих сдать единый государственный экзамен по информатике в 2017 году составило 12%, а в 2018 году увеличилось до 14% (http://gia.edu.ru). Таблица 4.1 показывает состояние образования в области информатики в Российской Федерации по данным Хеннера и Семакина.

Таблица 4.1. Текущее состояние (по данным за 2014 г.) образования в области информатики в Российской Федерации

Уровень	Описание
Младшие классы (от 7 до 10 лет)	Изучение понятий информатики обязательно в рамках таких предметов, как математика и технологии
Средние классы (от 11 до 15 лет)	Обязательный предмет «Информатика»
Старшие классы (от 16 до 17 лет)	Факультативный предмет «Информатика» (базовый или углубленный уровень изучения)
Единый государственный экзамен	Обязателен для поступления в ВУЗы по программам, связанным с информатикой

Адаптация рисунка 1 из работы Хеннера и Семакина, 2014

4.2.1. Федеральные образовательные стандарты, касающиеся информационно-коммуникационных технологий

Федеральная система образования показывает, как высоко она ценит компьютерные науки, выделяя время на изучение информатики в школьных программах и разделах государственных образовательных стандартов. В рамках действующих федеральных образовательных стандартов особое внимание уделяется вычислительной деятельности учеников: стандарты начального общего образования (1-4 классы), основного общего образования (5-9 классы) и полного общего образования (9-11 классы). Разделы, посвященные информационно-коммуникационным технологиям (ИКТ), представлены ниже. Этот термин используется как более общий по сравнению с термином «вычислительные науки». Вычислительное мышление в российских образовательных стандартах пока не упоминается.

ФЕДЕРАЛЬНЫЙ ГОСУДАРСТВЕННЫЙ ОБРАЗОВАТЕЛЬНЫЙ СТАНДАРТ НАЧАЛЬНОГО ОБЩЕГО ОБРАЗОВАНИЯ (утвержден Приказом N° 1897 Министерства образования и науки России, издан 6 октября 2009 года за N° 373; с изменениями от 26 ноября 2010 года N° 1241 и от 22 сентября 2011 года N° 2357)

12.2. Математика и информатика:

1 использование начальных математических знаний для описания и объяснения окружающих предметов, процессов, явлений, а также оценки их количественных и пространственных отношений;
2) овладение основами логического и алгоритмического мышления, пространственного воображения и математической речи, измерения, пересчета, прикидки и оценки, наглядного представления данных и процессов, записи и выполнения алгоритмов;
3) приобретение начального опыта применения математических знаний для решения учебно-познавательных и учебно-практических задач;

4) умение выполнять устно и письменно арифметические действия с числами и числовыми выражениями, решать текстовые задачи, умение действовать в соответствии с алгоритмом и строить простейшие алгоритмы, исследовать, распознавать и изображать геометрические фигуры, работать с таблицами, схемами, графиками и диаграммами, цепочками, совокупностями, представлять, анализировать и интерпретировать данные;
5) приобретение первоначальных представлений о компьютерной грамотности.

Хотя в Федеральном государственном образовательном стандарте начального общего образования прямо не прописана необходимость обучения программированию, современное представление о компьютерной грамотности подразумевает самовыражение посредством программирования цифровых историй и анимации. В настоящее время для таких занятий существует множество действующих программируемых агентов, которыми могут управлять маленькие дети. Компьютерная грамотность играет особую связующую роль для результатов междисциплинарного обучения в образовательном стандарте основного общего образования. Можно утверждать, что навык вычислительного участия очень важен с точки зрения способности организовать совместное обучение и совместную работу с учителем и сверстниками. Однако было бы опрометчиво ограничивать компьютерную грамотность только базовым использованием таких программных приложений, как Microsoft Word, Excel и PowerPoint.

ФЕДЕРАЛЬНЫЙ ГОСУДАРСТВЕННЫЙ ОБРАЗОВАТЕЛЬНЫЙ СТАНДАРТ ОСНОВНОГО ОБЩЕГО ОБРАЗОВАНИЯ (утвержден Приказом N° 1897 Министерства образования и науки России, издан 17 декабря 2010 года за N° 1897)

10. Трансдисциплинарные (или метапредметные) результаты освоения основной образовательной программы основного общего образования должны отражать:
<...>
7) умение создавать, применять и преобразовывать знаки и символы, модели и схемы для решений учебных и познавательных задач;
9) умение организовывать учебное сотрудничество и совместную деятельность с учителем и сверстниками; работать индивидуально и в группе: находить общее решение и разрешать конфликты на основе согласования позиций и учета интересов; формулировать, аргументировать и отстаивать свое мнение;
11) формирование и развитие компетентности в области использования информационно-коммуникационных технологий (далее ИКТ-компетенции).

В ходе обсуждения Федерального государственного образовательного стандарта начального общего образования специалисты подчеркнули, что знание языка программирования имеет значение не само по себе, а в связи с изучением всех тем школьного курса информатики. Кроме того, умение программировать значительно расширяет круг решаемых задач по физике, химии, биологии и другим школьным предметам.

11.3. Математика и информатика:
<...>
10) формирование информационной и алгоритмической культуры; формирование представления о компьютере как универсальном устройстве обработки информации; развитие основных навыков и умений использования компьютерных устройств;
11) формирование представления об основных изучаемых понятиях: информация, алгоритм, модель — и их свойствах;
12) развитие алгоритмического мышления, необходимого для профессиональной деятельности в современном обществе; развитие умений составить и записать алгоритм для конкретного исполнителя; формирование знаний об алгоритмических конструкциях, логических значениях и операциях; знакомство с одним из языков программирования и основными алгоритмическими структурами — линейной, условной и циклической;

13) формирование умений формализации и структурирования информации, умения выбирать способ представления данных в соответствии с поставленной задачей — таблицы, схемы, графики, диаграммы, с использованием соответствующих программных средств обработки данных;
14) формирование навыков и умений безопасного и целесообразного поведения при работе с компьютерными программами и в Интернете, умения соблюдать нормы информационной этики и права.

Среди результатов обучения, указанных в Федеральном государственном образовательном стандарте среднего (полного) общего образования, наибольшее внимание уделяется способности создавать и использовать компьютерные модели. Ученики, способные создавать и использовать компьютерные модели для анализа данных и проверки гипотез, имеют возможность участвовать в сложной работе, которую раньше отводили только профильным специалистам.

ФЕДЕРАЛЬНЫЙ ГОСУДАРСТВЕННЫЙ ОБРАЗОВАТЕЛЬНЫЙ СТАНДАРТ СРЕДНЕГО (ПОЛНОГО) ОБЩЕГО ОБРАЗОВАНИЯ (утвержден Приказом N° 413 Министерства образования и науки России, издан 17 апреля 2012 года за N° 413)

Информатика (базовый уровень) — требования к предметным результатам освоения базового курса информатики должны отражать:
<...>
5) сформированность представлений о компьютерно-математических моделях и необходимости анализа соответствия модели и моделируемого объекта (процесса); о способах хранения и простейшей обработке данных; понятия о базах данных и средствах доступа к ним, умений работать с ними;

Информатика (углубленный уровень) — требования к предметным результатам освоения углубленного курса информатики должны включать требования к результатам освоения базового курса и дополнительно отражать:
<...>
9) владение опытом построения и использования компьютерно-математических моделей, проведения экспериментов и статистической обработки данных с помощью компьютера, интерпретации результатов, получаемых в ходе моделирования реальных процессов; умение оценивать числовые параметры моделируемых объектов и процессов, пользоваться базами данных и справочными системами.

4.2.2. Подготовка учителей

В Российской Федерации учителей информатики готовят тремя способами:

- **Дипломы педагогических институтов по специальности «Информатика»**. Учителя могут выбрать один из двух вариантов: i) четырехлетнюю программу бакалавриата с одним профилем образования в области информатики — Педагогическое образование (информатика) — код программы 44.03.01; или ii) пятилетнюю программу бакалавриата с двумя профилями образования, например, Педагогическое образование (математика и информатика, информатика и технологии) — код программы 44.03.05.

- **Обучение специалистов**. Многие выпускники других программ начинают преподавать информатику, пройдя курсы повышения квалификации на базе различных педагогических институтов. Среди таких учебных курсов можно назвать программу Intel «Обучение для будущего» и «Обучение основам». Программа Intel «Обучение для будущего» (2004-2014 гг. http://www.iteach.ru/) предназначена для повышения квалификации учителей пу-

тем внедрения технологических средств для обеспечения проектно-ориентированного обучения. Вики-платформа «Летописи», поддерживающая совместные проекты преподавателей и учеников, работает с 2006 года и доступна на сайте letopisi.org.

- **Подготовка на месте работы**. Курсы повышения квалификации для учителей информатики организуются для штатных преподавателей на базе педагогических институтов, университетов и специальных институтов повышения квалификации учителей. Платформа «Летописи», продвигающая использование новых технологий, основала «хабы» в более чем 30 региональных институтах повышения квалификации учителей (E. Patarakin and Visser 2012).

4.3. Scratch

В рамках направления языков Лого были созданы среды, в которых ученики могли создавать и управлять многочисленными агентами. В середине 1990-х годов Митч Резник и его команда разработали продукт StarLogo, в котором действовало множество черепашек [107,170]. Продукт был нацелен на изучение закономерностей и явлений, в которые вовлечено множества агентов. В результате синтеза идей StarLogo и возможностей языка Squeak был разработан язык Scratch. Scratch наследует и развивает идеи, которые Пейперт сформулировал в среде Logo. Scratch учит нас создавать проекты из кирпичиков и делиться результатами с другими людьми. Эти навыки важны не только для специальных сред программирования, но и для современных онлайн-сообществ. Единство процессов создания, поиска и хранения информационных кирпичиков все чаще можно наблюдать на страницах современных сайтов, созданных по стандарту Web 2.0. Метафора строительных блоков, из которых дети и взрослые могут собрать простые и очень сложные конструкции, присутствует не только в учебных проектах, но и в большинстве современных сетевых сервисов формата Web 2.0, предназначенных для поддержки организаций и онлайн-сообществ обмена знаниями.

Метафора кирпичиков, из которых каждый ученик собирает структуры собственного мышления, активно использовалась в конструкционизме — и таких обучающих средах, как Logo и Lego-Logo. Многочисленные многопользовательские обучающие миры также основывались на простых объектах и простых действиях, которые могли совершать участники [104,171]. Все эти эффективные педагогические идеи на какое-то время были забыты в рамках первой волны развития Всемирной паутины, когда возможность разместить в сети учебные материалы на какое-то время отодвинула на задний план важность продуктивной деятельности самих учащихся. Но с развитием сервисов Web 2.0 все вернулось на круги своя, и мы видим, какие интересные идеи и технологии параллельных действий и обмена информацией оказались доступны для сферы образования.

4.3.1. Инструменты совместной работы в среде Scratch

В Scratch можно создать новый проект онлайн при помощи блоков в окошке браузера. Не нужно скачивать проект себе на компьютер, чтобы посмотреть, как он устроен, из каких спрайтов он состоит и какой код определяет поведение каждого спрайта. Чтобы просмотреть внутреннюю организацию, достаточно нажать кнопку «Заглянуть внутрь», и можно увидеть все спрайты и коды управляющих программ. При просмотре кода проекта у читателя есть возможность создать свою версию этого проекта. Для этого нужно просто нажать на кнопку «Ремикс». Среда Scratch способствует созданию ремиксов. Например, в разделе «Часто используемое» — http://scratch.mit.edu/latest/remixed — собраны проекты, которые чаще других использовались участниками социальной сети в качестве основы для новых проектов. В Scratch-сети такие проекты ценятся даже больше тех, что понравились большинству участников.

Среда Scratch дает авторам возможность самим организовывать группы-студии по интересам.

В таких студиях участники могут собирать проекты по темам. Например, это могут быть проекты, связанные с определенными играми, тематическими моделями и историями и т.д. Создатель Scratch-студии автоматически становится ее руководителем и может не только добавлять проекты в студию, но и разрешать другим участникам добавлять проекты, предоставив им права куратора студии.

Принцип создания и повторного использования можно сформулировать следующим образом: «создавай объекты — спрайты и проекты, — которые другие люди будут не только просматривать и изучать, но и использовать». С другой стороны, в социальной сети действует и обратное правило: «можно и нужно брать и использовать цифровые объекты — спрайты и проекты, созданные другими людьми». Эта практика повторного использования цифровых материалов достойна всяческой похвалы. В социальной сети Scratch 2.0 это правило повторного использования получило дополнительную поддержку. В системе появилась возможность не только просматривать чужие проекты, но и выбирать из них отдельные понравившиеся спрайты и перетаскивать их в виртуальный рюкзак. Рюкзак (временное хранилище кода) — еще один инструмент совместной работы, доступный на Scratch-портале. С помощью рюкзака при просмотре проекта можно заимствовать нужный спрайт или код этого спрайта. Просматривая проекты, пользователь может перетащить заинтересовавшие его спрайты или скрипты в виртуальный рюкзак, затем при редактировании собственного проекта достать эти компоненты из рюкзака и использовать их по собственному усмотрению. Рюкзак помогает организовать совместную работу и позволяет встраивать различные скрипты в общей студии, когда партнеры работают над разными частями проекта и создают спрайты и скрипты, которые в конце будут собраны в рюкзак и составят финальную версию проекта.

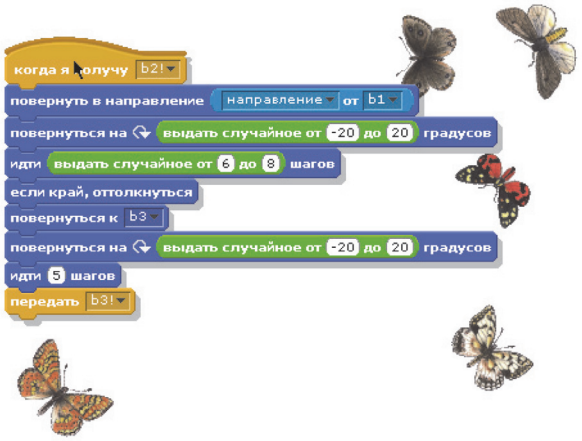
Все члены сообщества делятся своими проектами и наборами команд. Нет никаких секретных наборов команд, все наборы команд доступны всем. Каждый набор команд можно использовать и изменять. Каждый зарегистрированный член сообщества может опубликовать на сервере сообщества свой проект. При публикации автор добавляет к проекту краткое описание и ключевые слова — теги. Другие члены сообщества могут отметить проект как понравившийся, отметить проект своими тегами, оставить комментарий к проекту, добавить проект в галереи сходной тематики. Члены сообщества могут скачать проект, если хотят просмотреть его скрипты на своем компьютере. Скачанный проект можно изменить и дополнить. После этого можно вновь опубликовать проект на сервере как свой собственный. При этом ПО сервера распознает проект как производный от другого проекта и добавит в описание проекта «Этот проект создан на основе» □ Ссылка на родительский проект. Автоматические ссылки на родительский проект позволяют анализировать связи между проектами и отслеживать повторное использование проектов, при этом другие члены сообщества могут запрограммировать поведение и изобразить внешний вид отдельных спрайтов. У нас в программе есть уровень поведения для отдельных спрайтов. И у нас есть механизм создания ремиксов и студия для сбора материалов для проекта. В целом, у нас есть команда для создания нового блока, и есть блоки, созданные членами сообщества.

4.3.2. История сети скретчеров в России

В рамках развития международного Scratch-сообщества среда Scratch и материалы сайта со Scratch-проектами были русифицированы уже в 2006 году. Кроме того, эта среда была дополнена объектами, знакомыми российским школьникам, с помощью которых они могли создавать игры и истории с персонажами, представляющими природу, историю и литературу России. То есть теперь они могут не просто посмотреть и почитать, но и поиграть с объектами и использовать их для своих проектов. В частности, дети могут осваивать цифровые коллекции российских университетов, что чрезвычайно полезно. Например, для создания спрайтов и декораций использовались коллекции Астрономического музея Нижегородского педагогического университета.

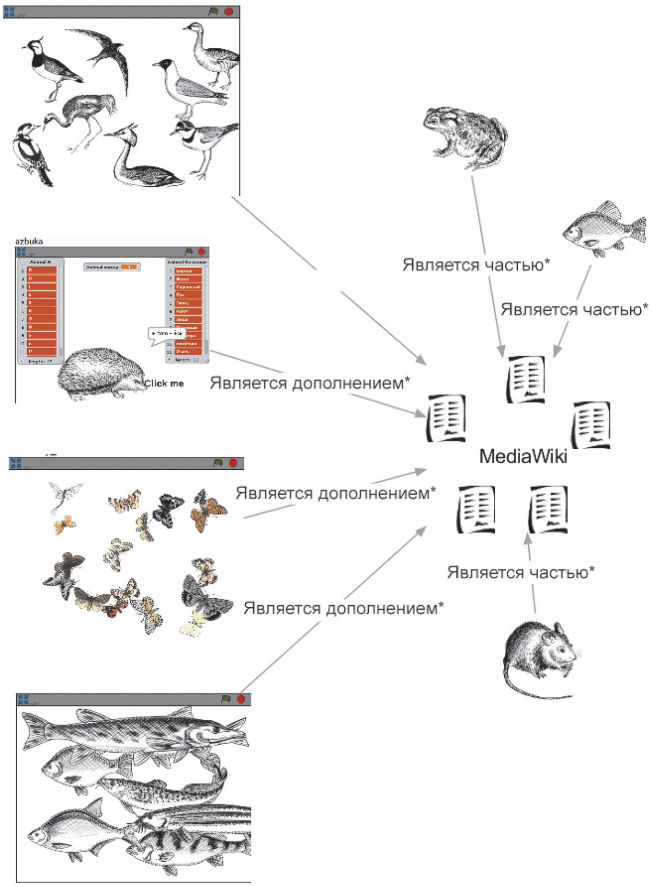
Рисунок 4.1. Спрайты для коллекции музея

Рисунок 4.2. Бабочки из коллекции насекомых



Многие растения и животные попали в российские Scratch-библиотеки из определителей и цифровых Красных книг, созданных экологами из Нижнего Новгорода. После того как объекты коллекций встраивают в открытые Scratch-галереи, учащиеся и школьники могут использовать их для создания новых презентаций, мультимедийных историй и игр.

Рисунок 4.3. Связь вики-статей и Scratch-проектов в Letopisi.org



Scratch учит нас создавать проекты из кирпичиков и делиться своими результатами с другими людьми. Эти навыки важны не только в специальных средах программирования, но и в современных онлайн-сообществах. Особенностью сети Scratch-программистов или скретчеров в России была тесная связь с вики-Scratch. Так получилось, что российский вики-проект с международным участием Letopisi.org поддерживал русских скретчеров с самого начала. Вики-среда Letopisi.org была настроена таким образом, чтобы участники могли сохранять там свои проекты и отдельные спрайты. Изображения растений и животных были включены в вики-статьи и одновременно использовались как часть спрайтов Scratch-проекта (рис. 4.3).

Летние компьютерные школы воспользовались преимуществами параллельной деятельности и возможностью сохранять промежуточные результаты Scratch-проектов в вики [172].

В 2008 и 2009 году в летних школах в деревне Старая Пустынь, работаю-

щих по модели «1 ученик: 1 компьютер», Scratch был основным инструментом создания ученических проектов и конкурировал с PowerPoint в представлении результатов проектов[14]. Язык Scratch имеет важную особенность, которая помогает упорядочить такую работу, — отдельные объекты, то есть спрайты, можно сохранять отдельными файлами. Затем эти файлы можно объединить в общий проект. В 2010 году обмен между вики и Scratch был расширен с помощью MediaWiki, что позволило встраивать Scratch-проекты в тексты вики-статей. Сегодня проект «Летописи» поддерживает русских скретчеров и помогает разрабатывать русскоязычные Scratch Wiki, связанные со Scratch-платформой. См. http://www.mediawiki.org/wiki/Extension:Scratch и https://scratch-ru.info/wiki/.

Обе платформы используют wiki-scratchblocks, — https://github.com/tjvr/wiki-scratchblocks — которые автоматически трансформируют текст Scratch-команд в визуальные блоки. Например, следующий текстовый блок команд будет трансформирован и отображен на вики-странице в виде графики:

```
<scratchblocks>
when [up arrow v] key pressed
if on edge, bounce
move (n1) steps
</scratchblocks>
```

Рисунок 4.4. Scratch-блоки на вики-страницах

Эта особенность помогает учителям не только обсуждать формирующиеся варианты учебной деятельности на вики-страницах, но и создавать ремиксы учебной деятельности с включением конкретных примеров кода Scratch-программы.

4.3.3. Инициативы Intel Corporation и Google (Alphabet Corporation)

В 2008 году в России впервые прошел конкурс индивидуальных проектов, написанных учениками в среде Scratch. Конкурс стартовал 20 января 2008 года. Члены жюри рассматривали два вида работ: игры и видеоролики. В первом (заочном) туре приняли участие более 500 человек. После первого тура прошли выездные полуфиналы в Нижнем Новгороде, Москве, Новосибирске и Санкт-Петербурге. В выездных полуфиналах приняли участие более 80 ребят. Победители региональных полуфиналов из Новосибирска, Ангарска, Нижнего Новгорода, Сарова, Магнитогорска, Москвы, Ярославской области и Якутии стали участниками следующего заключительного этапа, состоявшегося 20 июня 2008 года в Санкт-Петербурге. В течение четырех часов юные программисты соревновались в умении создавать компьютерные игры в среде Scratch. Подросткам были предоставлены различные скрипты и специальная библиотека объектов и фонов[15].

В 2010 году программа Intel «Обучение для будущего» организовала конкурс анимированных историй «Жили-Были» для школьных команд из разных регионов России и стран СНГ (в состав команд входили учителя и дети). В каждом крае и каждом регионе есть свои уникальные истории — легенды, мифы, предания, сказания и присказки. Это те истории, которые передаются из поколения в поколение, уникальные истории именно этого региона или уголка мира. Организаторы конкурса предложили командам рассказать эти истории в форме анимации. Из 195 ра-

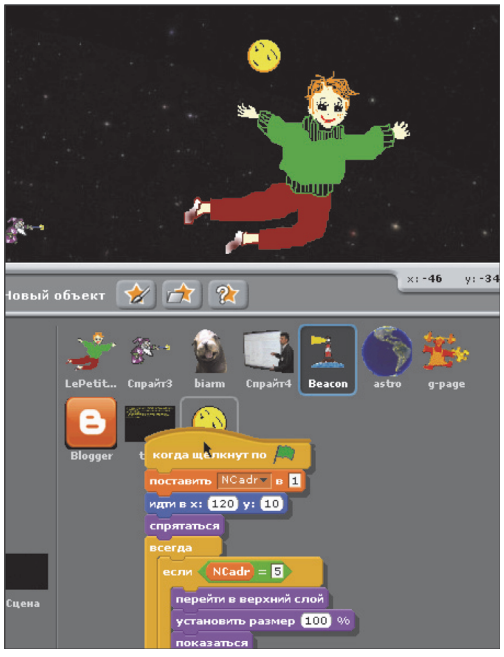
[14] Фотографии из летней школы доступны по адресу https://www.flickr.com/photos/patarakin/albums/72157617516398180. В 2009 году группа школьников разработала проект Bouns https://scratch.mit.edu/projects/652601/ как группа настоящих программистов, в которой каждый участник отвечал за свою часть работы и стремился внести свой вклад.
[15] Работы победителей представлены в галерее https://scratch.mit.edu/studios/17369/

Рисунок 4.5. Галерея победителей

бот, представленных на конкурс, 29 проектов были выполнены в формате цифровой истории в среде Scratch. Ссылки и описания этих проектов доступны в вики-Летописи http://letopisi.org/index.php/Scratch:Once_upon_a_time.

Учитывая растущий интерес учителей к возможностям новой обучающей среды, весной 2010 года была организована дистанционная школа Scratch, в которую были зачислены более 100 преподавателей из разных регионов России. Все материалы школы были открытыми и были опубликованы в нескольких форматах: блог, вики и Google Docs — https://groups.google.com/forum/#!forum/scratch_ru.

Рисунок 4.6. Ремикс проекта «Маленький принц»



Наряду с обучением, представлением материалов и мониторингом навыков, Scratch-школа задумывалась как один из шагов по построению российской сети, в которой участники смогут обмениваться результатами своей работы, совместно собирать и выстраивать общие конструкции. По правилам школы участники должны были зарегистрироваться на сайте Scratch и в группе Google. Выполненные задания выкладывали на сайте, а проблемы, с которыми сталкивались учителя при выполнении заданий, обсуждались в рабочей группе. Далеко не все смогли найти время и силы для выполнения всех заданий, но большинство планировало использовать навыки, полученные в рамках школьных занятий. Дистанционная школа выполнила свою основную задачу по привлечению новых участников в сеть детских инструментов проектирования. Следующий шаг предполагает продолжение общей деятельности, в рамках которой учителя смогут обмениваться опытом и методическими разработками. В качестве площадки для такого обмена выбрана среда созда-

ния коллективной книги, которая позволяет встраивать в страницы не только изображения проектов или скрипты, но и готовые проекты. Соавторы книги могут добавлять к текстам уроков выполненные задания.

Российские учителя — участники профессионального сообщества «Образовательная галактика Intel» — приняли участие в нескольких сетевых проектах, направленных на формирование и изучение культуры ремиксов. Первый сетевой проект назывался «Путешествие маленького принца», в нем приняли участие 12 учителей. Идея проекта заключалась в создании интерактивной игры-квеста, главный герой которого перемещался в виртуальном пространстве и должен был найти и посетить заданное количество космических планет.

Участников проекта попросили создать планеты и их обитателей. Каждый участник мог взять одну планету и обустроить ее по собственному усмотрению. В проекте была использована идея простого сотрудничества, когда каждый участник добавлял в проект новых персонажей, за счет чего игра становилась все более насыщенной. Конечный результат получился довольно интересным, но сам проект был слишком сложным и, выиграв соревнование, мы не смогли достичь тактических целей, связанных с развитием отношений и связей между участниками и цифровыми объектами. График ремиксов красноречиво свидетельствовал о том, что организаторы постоянно помогали и поддерживали участников, не давая им делать все самостоятельно.

Сетевой проект «День Scratch в России 2012» был менее сложным. Его действие развивалось в пространстве одной сцены, в которую участники добавляли свои изображения, тексты и аудиозаписи[16]. Простая структура проекта позволила решать сетевые задачи — объединить как можно больше людей и связать их с инструментами, которые они в дальнейшем будут использовать в своей учебной деятельности. Проект был реализован более успешно — и по количеству участников (28), и по количеству созданных ремиксов (22). График ремиксов свидетельствует о том, что участники успешно передавали проект друг другу и устанавливали связи с другими участниками и цифровыми объектами.

Следующий сетевой проект был связан с созданием совместного букваря и включал следующие этапы:

- Участникам дали проект, над которым они должны были работать. Это был проект по сценарию «Дом, который построил Джек».
- Участники обсудили возможные варианты разработки проекта.
- Участники выдвинули несколько предложений и представили собственные прототипы проекта, с которых они могли начать совместную деятельность.
- Участники самостоятельно выбрали проект по созданию совместного букваря, в котором каждый участник добавлял свой спрайт, связанный с новой буквой алфавита.
- Участники столкнулись с проблемой дублирования — у одной буквы было несколько спрайтов. Проблему обсудили и успешно решили.
- Проект был завершен, и в окончательном варианте были представлены все буквы. К исходному проекту было добавлено более 30 ремиксов.

Рисунок 4.7. Ремиксы проекта по созданию букваря

[16] https://scratch.mit.edu/projects/2466633/

Окончательный вариант представлен на рисунке 4.7. По завершении проекта состоялось его критическое обсуждение, в ходе которого участники проанализировали свои достижения и проблемы, с которыми им пришлось столкнуться. При обсуждении по теме проекта было представлено более 200 комментариев. Участники отметили успешную самоорганизацию, возможность выбрать тему и скрипт проекта, установить рабочие отношения в группе без посторонней помощи. Среди навыков, которые использовались и отрабатывались в рамках сетевого взаимодействия, были отмечены следующие:

- навыки работы в команде, навыки ведения переговоров, способность распределять обязанности;
- умение использовать возможности сетевых коммуникаций для оперативного взаимодействия;
- креативное мышление;
- способность понимать, принимать и развивать чужие идеи;
- способность работать в заданном ключе в рамках единой стратегии;
- базовые навыки программирования, способность создавать простейшие алгоритмы и скрипты.

Среди проблем, которые были выявлены в процессе разработки проекта, были отмечены следующие:

- Проблема последовательности действий. Было сложно спланировать последовательность действий, необходимую при сетевом сотрудничестве, когда участники совершают действия с цифровыми материалами один за другим.

- Проблема, связанная со стремлением сделать как можно больше и добавить в общий проект излишнее количество собственных цифровых материалов. Такое стремление отмечалось у многих участников, которые нарушали правила и договоренности.

- Нежелание использовать социальные возможности сети Scratch — обсуждение проектов и установление связей между участниками осуществлялось на портале «Образовательная галактика Intel», а не в сети Scratch.

- Проблема использования открытого контента, доступного в галереях социальной сети Scratch. Множество спрайтов, которые уже есть в социальной сети, остались невостребованными.

Судя по результатам проекта, учителя пока не знают о том, что сама социальная сеть Scratch может служить простым и открытым источником объектов для повторного использования. Участники проигнорировали образовательную стратегию проектирования социальной сети Scratch, когда можно и нужно пользоваться цифровыми объектами, созданными внутри сети. Как следствие, многие из представленных в проекте цифровых объектов сделаны на основе персонажей зарубежных мультфильмов. Очень жаль, ведь в социальной сети Scratch представлены коллекции проектов, в которых есть птицы, звери, рыбы, змеи и насекомые, обитающие в России. Авторы этих коллекций были бы рады поделиться с участниками своими цифровыми объектами, но участники не искали легких путей.

Все представленные проекты были очень простыми с точки зрения программирования и работы с цифровыми материалами. Сложность заключается в организации коллективных действий и сотрудничества. Во всех проектах использовалась одна тактика: участникам предлагалась задача для совместного решения и возможные способы решения этой задачи с использованием исходного цифрового объекта. В ходе совместного решения поставленной задачи про-

исходит включение участников в социальную сеть, куда добавляются новые объекты, между участниками и цифровыми объектами возникают новые связи, между участниками формируются отношения, основанные на участии в совместной проектной деятельности.

4.4. Scratch Collab 2018

В 2018 году Московский городской университет и Национальное общество технологий в образовании объединили усилия с командой Всемирного банка в рамках российской сети для проведения экспериментов с новыми видами учебной деятельности, формирующими навыки XXI века. Стоит отметить, что организаторы сети обладали многолетним опытом организации общественных и образовательных проектов на стыке образовательной политики и учебной деятельности. Этот проект был нацелен на изучение цифровой грамотности в сфере всех навыков XXI века. В рамках цифровой грамотности акцент был сделан на навыки совместной работы. Организаторы решили найти и разработать учебную деятельность, которая поможет сформировать способность и желание учащихся делиться результатами своей деятельности, работать в команде и распределять задания. Они выбрали Scratch как среду, в которой уже есть инструменты для организации и мониторинга такой деятельности.

4.4.1. Создание ремиксов учебной деятельности

Следует отметить, что среда Scratch повлияла на стратегию сбора методических материалов для проектов в форме Scratch-проектов. Учебная деятельность — это взаимодействие между учащимся и средой (включая ресурсы контента, средства и инструменты, компьютерные системы и сервисы, события и объекты реального мира), которое происходит в ответ на задание с предполагаемым результатом. Учебная деятельность включает ряд задач, выполняемых учеником в определенном образовательном контексте для достижения заранее определенных результатов обучения. Примеры учебной деятельности: поиск и обобщение Интернет-ресурсов, обработка данных в электронной таблице, написание совместного отчета в среде вики, создание цифровой истории в Scratch, создание компьютерной 3D-игры в StarLogo TNG или Alice. Новые виды учебной деятельности являются предметом педагогических разработок как общественного управления совместной деятельностью с использованием образовательных инструментов. С начала XXI века активно разрабатываются и уточняются международные стандарты описания и представления учебной деятельности. Существуют разные способы представления учебной деятельности (описания, схемы, диаграммы и т.д.). Последовательность действий в учебной деятельности можно перевести в Scratch-логику (рис. 4.8). В рамках проекта были объединены Scratch-хакатоны и соревнования для школьников Collaborative Challenge. Для проведения хакатонов и соревнований были созданы шаблоны — прототипы сценариев учебной деятельности. Первый образовательный сценарий — это описание учебной деятельности, которая включает последовательность действий. Каждое действие описывается в формате ответа на вопрос «Как сделать ...?»:

Как создать и использовать аккаунт учителя?
Как зарегистрировать участников?
Как создать студию для работы над общим проектом?
Как пригласить в студию членов своей команды?
Как создать новый спрайт для нового персонажа?
Как заставить персонажа перемещаться по экрану?
Как посчитать, сколько шагов сделал персонаж?
Как заставить персонажей взаимодействовать?
Как создать ремикс проекта?
Как использовать дерево ремиксов?
Как правильно описать проект?

Как самостоятельно оценить качество кода?
Как использовать Dr. Scratch?
Как оценить совместность деятельности?

В этом сценарии уже заложен Scratch-проект, который автор сценария учебной деятельности выполняет от лица учеников, — это задача, которую он ставит перед учениками, пытается решить от их лица и понять, насколько правильно организован процесс.

Рисунок 4.8. Образовательная практика в виде Scratch-проекта / руководство в виде Scratch-проекта

Было создано несколько исходных авторских сценариев учебной деятельности. Все они были опубликованы на сайте Letopisi.org в разделе ScratchHackathon. Сценарий учебной деятельности — это не просто отвлеченное описание; он всегда описывает конкретный объект, то есть Scratch-проект[17]. На следующем этапе организаторы хакатонов обсуждали и дорабатывали исходные сценарии учебной деятельности. Некоторые блоки, из которых выстраивается учебная деятельность, были изменены или заменены новыми блоками. Например, в сценарии сказки о Колобке тестировали и обсуждали принцип взаимодействия персонажей, затем его полностью изменили на более простой принцип, управляемый на основе анализа значения переменной, обозначающей число шагов главного героя. Таким образом, в категории, включающей сценарии учебной деятельности, появлялись новые блоки, которые можно было использовать для создания измененных сценариев учебной деятельности. Работа на этом этапе повторяет создание ремиксов Scratch-проектов. Когда вносят изменения в сценарий учебной деятельности, меняется и Scratch-проект, вокруг которого выстраивается этот сценарий. Все записи ремикса, включая фрагменты кода и ссылки на Scratch-проекты, были объединены в один компонент — сборник методических материалов по организации учебной деятельности[18]. На третьем эта-

[17] Описание сценариев и перечень ссылок на сценарии опубликованы в обобщающей статье по адресу http://letopisi.org/index.php/Scratch/HowTo/Hackathon

[18] Этот сборник доступен в сети в формате вики-статьи и PDF: http://letopisi.org/index.php/Scratch/HowTo/Hackathon/Manual.

пе учителя-организаторы школьных семинаров и хакатонов получили готовые методические материалы в форме документа и сборника вики-статей. При подготовке школьных и внешкольных мероприятий некоторые учителя проводили дополнительное тестирование предложенных сценариев обучения и создавали собственные ремиксы учебной деятельности.

Когда мы проводили занятия с помощниками и обсуждали, как можно работать с детьми разного уровня подготовки и помогать тем, кто первый раз программирует на Scratch, мы поняли, что очень важно посадить самих педагогов за компьютеры и проработать все шаги, которые должны будут сделать дети для создания проекта. Педагоги объединились по три человека, попробовали все сами, и тогда они обратили внимание на проблемы, с которыми могут столкнуться дети. Например, для маленьких детей может оказаться очень сложным ждать друг друга, и поэтому необходимо учитывать подобные моменты при составлении критериев для разных возрастных групп.

- Из интервью с Оксаной Петровой, руководителем группы организаторов в Пскове

В Москве и других городах России прошли хакатоны, в ходе которых ученики осваивали новые виды учебной деятельности, направленные на формирование навыков совместной сетевой деятельности. Формат хакатонов был выбран как наиболее соответствующий поиску инновационных решений. Всего было проведено более 90 хакатонов, в которых приняли участие более 4 000 школьников. В рамках хакатонов внимание уделялось навыкам и инструментам совместной деятельности. В ходе тестирования учителя смогли определить недостатки и неопределенности предлагаемой учебной деятельности и внести изменения в шаблон. Эта работа проводилась большей частью на местах — в школах и центрах дополнительного образования. Наибольшее внимание в ходе семинаров и хакатонов уделялось использованию инструментов совместной деятельности.

Учителя в основном создавали ремиксы в Google-документах или на региональных вики-страницах, а не на Scratch-платформе, что сказалось на проведении исследования и измерении данных. Однако организаторы Scratch-хакатонов рассказали о многочисленных примерах совместной деятельности, включая использование студий и Scratch-проекты, созданные на основе сценариев учебной деятельности, разработанных учителями.

В конце хакатона мы сделали доску для обратной связи, на которую ребята клеили стикеры. Мы написали несколько вопросов на доске, например: Что было сложно? Получили ли они все, что хотели, от этого хакатона? Ребята отвечали на стикерах, которые мы затем размещали на доске. В итоге вся доска быстро заполнилась стикерами. Было очень интересно потом посмотреть на эти ответы. После хакатона у меня появилось желание продолжить проводить похожие мероприятия в регионе и, возможно, даже сделать что-то более масштабное. Многие школы заинтересовались подобным опытом. Пятьдесят человек только у меня познакомились со Scratch, и практически всем было интересно работать с данной платформой.

- Из обсуждения специфики организации учебной деятельности в Новосибирске с Михаилом Молчановым

Несколько фрагментов кода из проектов, созданных учителями для организации хакатонов, были перенесены из Scratch в вики в форме наглядного кода и включены в общий сборник методических материалов по организации учебной деятельности. В ходе хакатонов и семинаров школьники переключались на свою учебную деятельность, которая заимствует некоторые элементы образовательного сценария и тестирует его в реальных условиях. Полностью завершить работу и реализовать всю последовательность учебной деятельности удалось 339 командам участников. При этом учителя, организующие совместную деятельность команд, пришли к выводу, что соотношение команд, сумевших закончить работу и представить ее на кон-

курс, к общему числу участников составило примерно 60%. Такой показатель был, в основном, обусловлен сложностью командной работы, участникам было непросто договариваться друг с другом, распределять роли и обязанности.

— А были ли дети, которые не хотели взаимодействовать с другими ребятами, и все стремились сделать сами?
— Конечно. И в итоге у них ничего не получалось. В таких случаях результатом являлся проект с одним ремиксом, где все сделал создатель. Подобные проекты тоже не были отправлены на конкурс. Если посмотреть студию Новосибирского хакатона, там, я думаю, процентов 60 проектов было отправлено на конкурс. Самым интересным оказалось то, что дети не умеют договариваться. Для меня это была первая трудность, с которой я столкнулся: как научить их взаимодействовать между собой. С этой целью изначально мы сделали план, по которому они работали. Я старался, чтобы они всегда смотрели в этот план и в соответствии с ним действовали. Только так и получалось их усмирять.

- Из интервью с Михаилом Молчановым, организатором командной работы из Новосибирска

4.4.2. Оценивание Scratch-проектов

На финальном этапе конкурса жюри оценивало не только сам окончательный проект, но и материалы студии, деревья ремиксов, то есть то, насколько деятельность членов команды соответствовала разработанному сценарию учебной деятельности.

При оценивании проектов эксперты пользовались таблицей оценивания по критериям (таблица 4.2).

Таблица 4.2. Таблица критериев для оценивания проектов

Адрес проекта				
КРИТЕРИЙ (см. описание критериев ниже)	**СООТВЕТСТВИЕ**			**ДОПОЛНИТЕЛЬНЫЕ КОММЕНТАРИИ ИЛИ ПОЯСНЕНИЯ**
	Низкий (1,2,3)	**Средний (4,5,6)**	**Высокий (7,8,9)**	
ОРИГИНАЛЬНОСТЬ (собственные спрайты, фоны, музыка, идея)				
КАЧЕСТВО КОДА (проект должен грамотно использовать язык Scratch)				
СОВМЕСТНОСТЬ ДЕЯТЕЛЬНОСТИ (команда использовала все инструменты совместной деятельности — студия, рюкзак, ремикс)				
ОБЩЕСТВЕННОЕ ПРИЗНАНИЕ (как команда привлекала внимание к своему проекту)				
ОКОНЧАТЕЛЬНЫЙ ИТОГ	ОБЩИЙ БАЛЛ (макс. 36)			
	Рекомендовать или не рекомендовать как победителя (поставьте X в соответствующей ячейке)			
	Да			Нет

Таблица оценивания по критериям была дополнена следующими пояснениями по использованию критериев:

Оригинальность проекта

Оригинальность означает оригинальные спрайты, фоны, музыку и идею. При оценивании обратите внимание на библиотеки готовых спрайтов, фонов и звуков — участники могли просто взять готовые материалы из Scratch-библиотек или других цифровых хранилищ. Или могли рисовать, фотографировать и делать аудиозаписи. Идея проекта могла быть полностью или частично заимствована из известной игры или сказки, а могла быть придумана самостоятельно.

- Низкий уровень, если команда использовала Scratch-библиотеки готовых спрайтов, фонов и звуков или нашла готовые картинки и звуки в сети.
- Средний уровень, если команда изменила готовые картинки в Scratch-редакторе или другом редакторе.
- Высокий уровень, если авторы сами создали новые спрайты, фоны и звуки.

Как оценивать оригинальность? В описании проекта авторы часто указывают, какие картинки они нарисовали сами, какие картинки и анимации им помогли сделать мамы и папы, а какие картинки они нашли в сети. Картинку нашли в сети и оставили без изменений — 1–3. Картинку анимировали своими руками и с помощью мамы — 4–6. Картинку нарисовали — 7–9. Проверяем оригинальность спрайтов и фонов в режиме «Войти в проект».

Качество кода

Качество кода (контроль, представление данных, абстракция, интерактивное взаимодействие, синхронизация, параллельные действия и логические операторы). При оценивании обратите внимание на использование циклов и условий, хранение данных в переменных и списках, создание собственных блоков и клонов, использование управляющих событий, обмен сообщениями между спрайтами и использование логических операторов.

- Низкий уровень, если программа состоит из одного длинного блока кода, в котором собраны все команды.
- Средний уровень, если в программе использованы блоки повторений, логические операторы, управление через обмен сообщениями.
- Высокий уровень, если в программе есть блоки и клоны, управление поведением спрайтов происходит через проверку условий.

Как оценивать качество кода?

Оцениваем код в режиме «Войти в проект» и отмечаем, какие конструкции использовали авторы.

Совместная деятельность

Совместность (команда использовала все инструменты совместной деятельности —студию, рюкзак, ремикс). При оценивании обратите внимание, есть ли у команды студия, в которой авторы могли собирать материалы для своего проекта. Посмотрите на количество ремиксов и графическое представление дерева ремиксов. Почитайте комментарии, которыми обменивались члены команды при разработке и доработке промежуточных версий проекта.

- Низкий уровень, если работа представлена одним проектом, ремиксы и студия отсутствуют.

- Средний уровень, если есть студия и/или число ремиксов, предшествующих финальной версии проекта, составляет 4-5.

- Высокий уровень, если есть студия и высокое и ветвистое дерево ремиксов.

Как оценивать совместную деятельность?

Смотрим, создали ли участники студию, в которой собраны промежуточные проекты. Обращаем внимание на количество ремиксов. Нажимаем на кнопку «Посмотреть дерево ремиксов».

Общественное признание

При оценивании по этому критерию обратите внимание на количество просмотров, лайков и комментариев. Особенно бросаются в глаза лайки на дереве ремиксов.

- Низкий уровень, если есть лайки и комментарии только от членов команды.

- Средний уровень, если у проекта более 20 просмотров и несколько лайков и комментариев от сторонней аудитории, а не от организаторов конкурса.

- Высокий уровень, если аудитория активно комментирует и оценивает проект (больше 5 лайков и комментариев).

Общественное признание оценивают по количеству зрителей, добавивших проект в избранное, количеству зрителей, отметивших проект как понравившийся, и количеству комментариев.

Результаты конкурса

Младшие классы:

- Вторжение https://scratch.mit.edu/projects/216184554/
- Подводный мир https://scratch.mit.edu/projects/214725535/
- Чемпионат мира по футболу FIFA 2018 https://scratch.mit.edu/projects/216199396/
- Времена года https://scratch.mit.edu/projects/213470130/
- Выйти из комнаты https://scratch.mit.edu/projects/219007916/
- Поезд https://scratch.mit.edu/projects/216277723/
- История Дома Гончарова https://scratch.mit.edu/projects/219634556/

Средние классы:

- Энциклопедия по музыке https://scratch.mit.edu/projects/216055569/
- Интерактивная история https://scratch.mit.edu/projects/215946147/
- Музей ретро-автомобилей https://scratch.mit.edu/projects/218295696/
- Windows XP Pro https://scratch.mit.edu/projects/214724397/
- Волк собирает яйца! https://scratch.mit.edu/projects/163288952/
- Меломан https://scratch.mit.edu/projects/211179557/
- Змейка https://scratch.mit.edu/projects/219395515/

Старшие классы:

- Adventure Space https://scratch.mit.edu/projects/218050874/
- Шахтер Бен https://scratch.mit.edu/projects/216166616/
- Защита замка https://scratch.mit.edu/projects/219442699/
- Символ ФИФА 2018 https://scratch.mit.edu/projects/214645815/
- Монополия https://scratch.mit.edu/projects/214737116/
- Red Ball https://scratch.mit.edu/projects/219457475/
- Быки и коровы https://scratch.mit.edu/projects/219454275/

Рисунок 4.9. Скриншот проекта «Подводный мир»

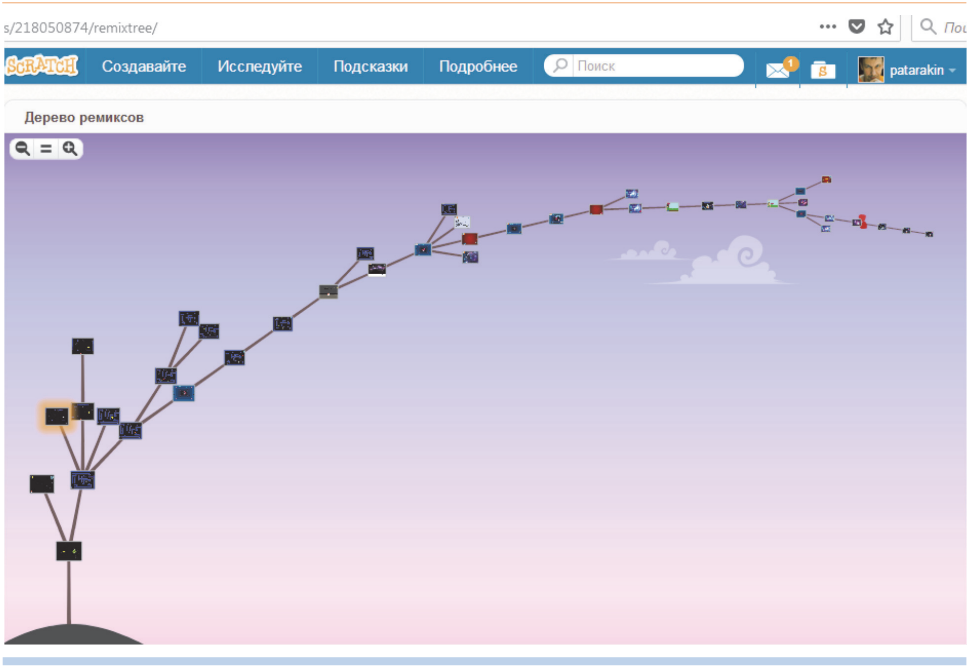
Ребята изначально хотели сделать так, чтобы подлодка во что-то стреляла, затем они решили, что она будет преодолевать препятствия — все эти идеи их. Их мама действительно помогала с анимацией спрайтов. Она сделала в фотошопе костюмы, однако код ребята писали самостоятельно. Причем они учились в процессе, спрашивали, как что делать, и в итоге написали все сами. Когда родители принимали слишком большое участие в проекте, они тем самым оказывали своим детям медвежью услугу, так как те в итоге ничему не научились.

- Из обсуждения специфики организации учебной деятельности в Новосибирске с Михаилом Молчановым

4.4.3. Обсуждение учебной деятельности с лучшими организаторами

Учебную деятельность, которая позволила школьникам добиться лучших результатов в конкурсе, обсудили с учителями, которые организовали такую деятельность. Интересно, что организаторы сосредоточили свое внимание на различных аспектах инструментов совместной деятельности. Например, организаторы работ победителей в старшей возрастной группе из Пскова наибольшее внимание уделили использованию дерева ремиксов. Все псковские старшеклассники в своих работах активно использовали ремиксы и отслеживали работу партнеров при помощи дерева ремиксов. Дерево проекта-победителя представлено на рисунке 4.10.

Рисунок 4.10. Дерево проекта https://scratch.mit.edu/projects/218050874



— Команде старшеклассников из Пскова удалось построить наиболее разветвленное дерево ремиксов. Как вам удалось выстроить с учениками такое замечательное дерево? Насколько активно дети его использовали?
— На мой взгляд, самым главным для учеников были заданные критерии. Старшеклассники очень четко уловили, за счет чего они могут выиграть. На ознакомительной части мы показывали пример дерева ремиксов и рассказывали, как оно строится. Ребята поняли, чтобы получить высокую оценку за «командную работу», нужно строить дерево ремиксов. Однако, это было не так просто, так как каждый делал свою часть, и участнику нужно было дождаться другого члена своей команды, чтобы идти дальше. Дети двигались в своем темпе, им приходилось согласовывать свои действия, несмотря на это они справились и претерпели все неудобства.

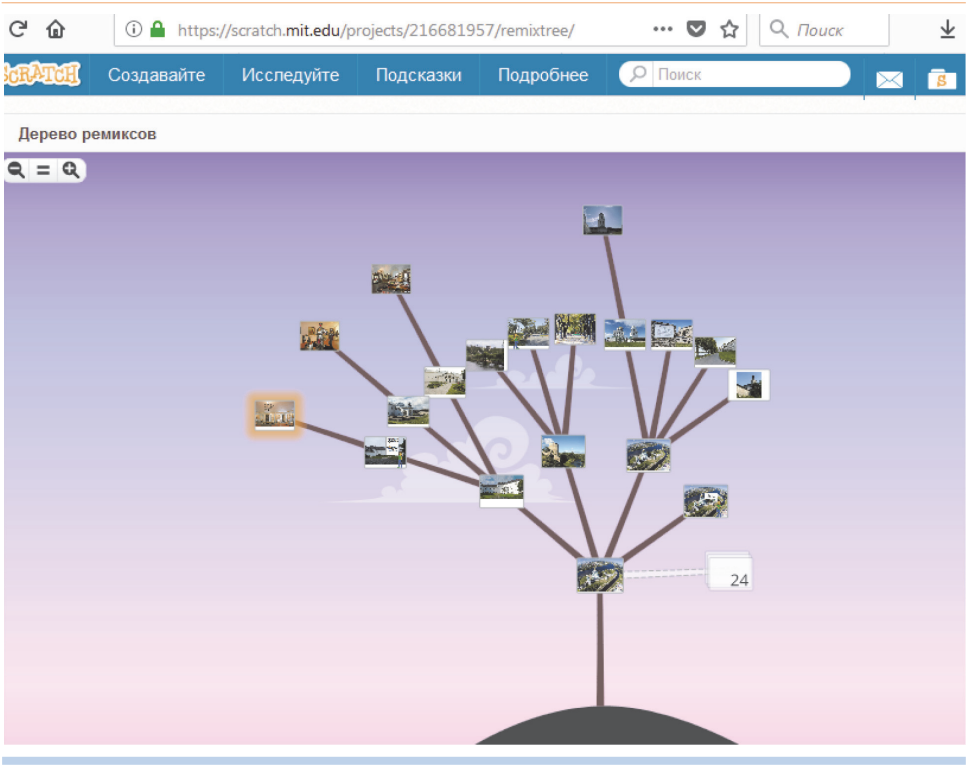
- Из интервью с Оксаной Петровой, организатором командной работы из Пскова

В другом проекте, созданном старшеклассниками из Пскова, было еще более разветвленное дерево ремиксов, но оценки членов жюри были низкими.

— У вас была еще одна команда с более «развесистым» деревом ремиксов, но они не выиграли. Как ребята это пережили?
— Трудно. Потому что на самом деле они тоже были действительно командой, и им очень хотелось победить. Ребята из второй команды были более творческие, на мой взгляд. Они знают, в чем ошиблись. Дело в том, что они выбрали такой проект, в котором очень сложно было выстроить дерево. Относительно командности еще добавлю, что были и дети, которые знают язык Scratch, и те, кто первый раз с ним столкнулись. Чтобы организовать совместную работу, ребята использовали рабочие листы, где они прописывали свои роли, исходя из задач и критериев конкурса, и учитывая способности каждого, таким образом, чтобы члены команды дополняли друг друга. В итоге ребята поддерживали друг друга, чтобы получился красивый результат.

- Из интервью с Оксаной Петровой, организатором командной работы из Пскова

Рисунок 4.11. Дерево проекта 216681957 (Псков)

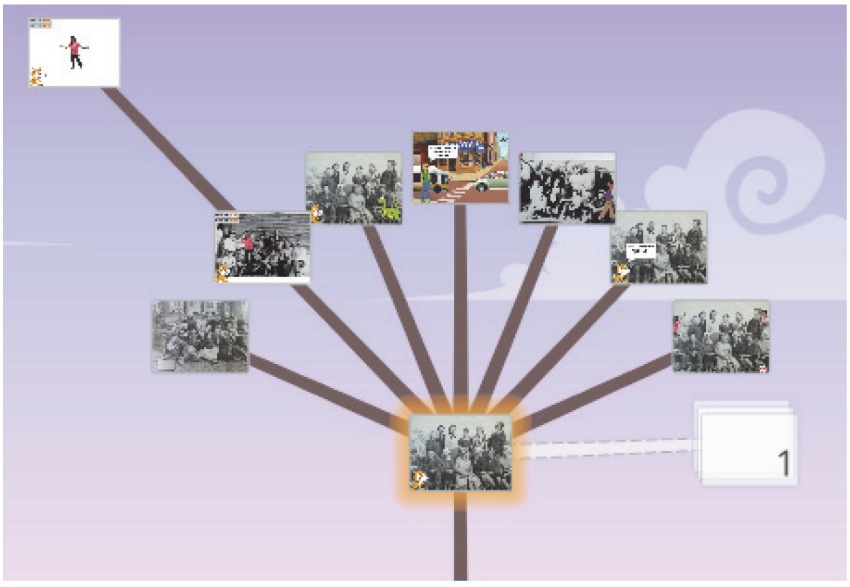


В младшей возрастной группе особое внимание уделялось общей студии, в которой собирались все работы участников. Это считалось важным изменением исходного сценария. Сразу несколько команд из этой студии стали победителями. Для анализа общественной структуры студии использовался модифицированный метод на основе викиграммы — связи между участниками и проектами определялись на основе действий участников, связанных с проектами, собранными в этой студии. Сплошные линии на рисунке означают, что участник является автором проекта. Пунктирные линии означают, что участник комментировал проект.

Чтобы добиться успеха в XXI веке, учащиеся должны иметь базовые знания о сложных сетях, которые позволят им изучить взаимосвязанность нашего мира. Сети охватывают все аспекты жизни: биологический, физический, экономический и социальный, и связи в нашем обществе продолжают укрепляться благодаря использованию инструментов социальных сетей. В силу своей междисциплинарности сетевые науки становятся идеальным инструментом привлечения учеников в сферу точных наук. Компоненты сетей уже много лет используются в качестве инструмента преподавания математики и компьютерных наук. Ученые расширили эту сферу, разработав и предоставив материалы, демонстрирующие, каким образом можно использовать сетевые науки для вовлечения учеников в сферу точных наук [173–175].

Понимание науки о сетях позволяет нам углубленно изучать различные системы. Scratch-сообщество поддерживает процесс групповой рефлексии посредством инструмента дерева ремиксов. Этот инструмент позволяет просматривать разработанные проекты в виде измененных копий каждого отдельного проекта. На рисунке 4.12 представлено дерево ремиксов для проектов студии «Путешествие в (деревню) Сомовку».

Рисунок 4.12. Дерево ремиксов студии «Путешествие в Сомовку»



Субъекты деятельности могут не только совместно изменять объекты, но и записывать все изменения в журнал изменений. Данные этого журнала можно использовать для построения карты совместной сетевой деятельности учащихся. Эта карта позволяет оценивать образовательную социально-технологическую систему по показателям, характеризующим совместную сетевую деятельность. Совместная сетевая деятельность учащихся представляет собой сложную адаптивную систему, для понимания которой необходимы специальные инструменты. Такими инструментами, позволяющими построить карты и схемы связей между субъектами совместной сетевой деятельности, являются сетевые модели, основанные на данных о действиях субъектов и изменении объектов совместной деятельности. Различные типы современных социально-технологических систем, в которых реализуется совместная деятельность участников, вносятся в рабочие журналы участников. В общем виде историю, сохраненную в журнале, можно просмотреть в виде записи игры, состоящей из множества ходов, каждый из которых содержит три обязательных элемента:

Субъект деятельности | *Объект деятельности*| *Вид деятельности*

Различные формы совместной сетевой деятельности (совместное редактирование документов, создание и комментирование записей в блоге, создание и совместное редактирование вики-статей и т.д.) можно свести к одной схеме, позволяющей анализировать и сравнивать деятельность участников. Перефразируя Бруно Латура, можно утверждать, что с каждым редактированием вики-страницы, с каждым ремиксом и модификацией в Scratch, Alice, StarLogo Nova и NetLogo с записью в электронном журнале все учителя получают возможность усовершенствовать процесс обучения. Возможность оценивания совместной сетевой деятельности по критериям при помощи социограмм уже была реализована для образовательной вики-системы Letopisi.org [176]. При разработке матрицы оценивания совместной сетевой деятельности использовались следующие критерии: продуктивность, связанность, сплоченность и устойчивость.

Поскольку все системы продуктивной сетевой деятельности основаны на общих принципах, мы предположили, что можно использовать сходный инструментарий для построения социограмм на базе анализа действий, которые совершили члены Scratch-сообщества (скретчеры) в

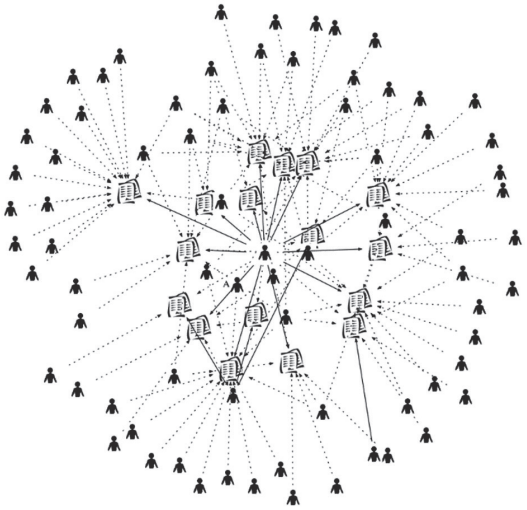
студиях, созданных для совместной работы. При этом мы считаем, что Scratch-студии по своим функциям аналогичны категориям MediaWiki. Критерии и показатели качества групповой деятельности в Scratch-студии представлены в таблице (Таблица 4.3).

Таблица 4.3. Оценивание структуры Scratch-студии по критериям

Критерий	Высокий уровень	Средний уровень	Низкий уровень
Продуктивность	Схема включает множество созданных участниками объектов.	Число объектов примерно равно числу участников.	Число объектов значительно меньше числа участников.
Связанность	Все субъекты и объекты деятельности представлены в одной схеме.	Субъекты и объекты деятельности представлены небольшим (3-4) числом компонентов.	Схема студии разбита на множество несвязанных компонентов.
Сплоченность	В социограмме представлена одна сплоченная группа/команда, в которой все участники связаны друг с другом через страницы.	Участники образуют несколько малочисленных групп.	В социограмме практически нет групп, что говорит об отсутствии взаимодействия.
Устойчивость	В социограмме представлены несколько ключевых игроков, связи которых обеспечивают устойчивость совместной деятельности.	В социограмме представлены 2–3 участника, удаление которых приведет к распаду схемы на несвязанные компоненты.	В социограмме представлен единственный ключевой игрок, через которого идут все информационные процессы. Удаление этой точки разрушит сеть.

В экспериментальных целях мы собрали данные о действиях участников в нескольких Scratch-студиях и на основании этих данных построили социограммы совместной деятельности. Пример такой социограммы приведен на рисунке 4.13. Фигурки людей обозначают участников/скретчеров, страницы кода — проекты. Сплошная линия от скретчера к проекту означает, что этот скретчер является автором проекта. Пунктирная линия означает, что скретчер комментировал проект.

Рисунок 4.13. Социограмма Scratch-студии



Как видно на социограмме, для студии характерна средняя продуктивность, высокая связанность и сплоченность, а также низкая устойчивость. Удаление центрального скретчера и его проектов приведет к распаду схемы студии на несвязанных участников.

Изначально, когда они сформировали команды, я предложил ребятам продумать план. Мы расписали детально, что они хотят сделать, какую игру получить в конце, и прописали количество спрайтов, которые у них будут в игре. Заготовку они составляли самостоятельно и затем защищали свой план мне: показывали, что будут делать, сколько будет спрайтов, какой будет интерфейс у игры (т.е. они делали небольшие эскизы). После того как они составили план, а план, как вы знаете, это половина работы, они уже приступали к работе с проектом. Таким образом, у них

был поделен фронт работы: например, за определенную часть спрайтов отвечает один программист, за другую – второй, или один занимается только дизайном, а другой – будет писать код. Дети делили обязанности, после чего им было нетрудно работать в студиях.

- Из обсуждения организации учебной деятельности в Новосибирске с Михаилом Молчановым

Выводы главы 4. Креативное программирование в России

- **Истоки конструкционизма в Российской Федерации**. Большая часть литературы, посвященной изучению советской науки, является узкоспециализированной и не совсем актуальна для современной вычислительной науки в России, тем не менее, сперва следует рассмотреть исторический контекст. Идеи и материалы, разработанные Пейпертом, повлияли на образование во всем мире, включая Российскую Федерацию, хотя не исключено, что влияние в России было сильнее, чем во многих других странах. Мы подробно обсудили широкое применение языка Logo и его вариантов на заре обучения детей машинным вычислениям. Даже во время холодной войны специалисты по обе стороны «железного занавеса» активно поддерживали связь друг с другом. Идеи Пейперта были революционными для своего времени и, возможно, до сих пор остаются таковыми, хотя Интернет и коллективное сознание претворили в жизнь некоторые из его самых смелых идей, включая идею обучения в формате школы самбы в Бразилии.

- **ИКТ в федеральных образовательных стандартах**. В то время как идеи Пейперта были поддержаны в реальной жизни и имели множество последователей в ранний период обучения детей программированию, более поздние разработки в Российской Федерации, похоже, были сделаны в традиционной форме с упором на общее использование компьютерных приложений, включая Microsoft Word и PowerPoint, а не программирование и вычислительное мышление. В действующих федеральных стандартах кодирование или программирование не указаны как часть учебной программы для начальной школы, но в стандартах указаны компоненты или элементы, которые составляют вычислительное мышление.

- **Scratch в Российской Федерации**. Язык Scratch, созданный Митчем Резником и другими последователями Сеймура Пейперта в Массачусетском технологическом институте, пользуется большой популярностью во всем мире. Первая версия Scratch была разработана в 2003-2006 годах. По мере развития международного Scratch-сообщества в 2006 году среда Scratch и контент сайта, на котором размещены Scratch-проекты, были русифицированы. Кроме того, в эту среду были добавлены объекты, знакомые русским школьникам, с помощью которых они могут создавать игры и истории с персонажами, представляющими природу, историю и литературу России. Это значит, что они могут не просто видеть и читать, но и играть с объектами и использовать их для своих проектов. В этом разделе представлены разные истории о скретчерах в России.

- **Scratch Collab 2018**. В 2018 году Московский городской университет и Национальное общество технологий в образовании объединили усилия с командой Всемирного банка в рамках российской сети для проведения экспериментов с новыми видами учебной деятельности, формирующими навыки XXI века. Проект объединил Scratch-хакатоны и конкурсы Collaborative Challenge для школьников. Сеть партнеров-организаторов разработала и использовала шаблоны — прототипы сценариев учебной деятельности. Поддержка команды Банка состояла в том, чтобы собственным примером привлечь внимание к идеям, представленным в книге. В общей сложности было проведено более 90 хакатонов, в которых приняли участие более 4 000 школьников. Ссылки на некоторые проекты, созданные участниками, представлены в тексте.

5. ОСНОВНЫЕ РЕКОМЕНДАЦИИ

В этой книге мы осветили множество тем и представили большой объем информации для должностных лиц и специалистов-практиков. В этой заключительной главе представлены ключевые рекомендации, основанные на информации, доступной в первых четырех главах.

5.1. Развитие вычислительного мышления через обучение программированию необходимо сделать ключевой целью образования для детей всех возрастов на всей территории Российской Федерации.

У входа на станцию метро в неблагополучном районе г. Вашингтон, округ Колумбия, местная организация разместила баннер, на котором изображена улыбающаяся девочка со стопкой книг. Вдохновляющий слоган гласит: «Если умеешь читать, можешь отправиться куда угодно!». Не умаляя важности навыка чтения, стоит отметить, что более подходящий баннер для станции метро в любой точке мира от Санкт-Петербурга в Российской Федерации до Крайстчерча в Новой Зеландии в XXI веке может гласить: «Если умеешь читать, можешь отправиться куда угодно. Если умеешь кодировать, можешь делать что угодно!». Это не преувеличение — поскольку повсеместное распространение компьютерных технологий и сетей влияет на все сферы жизни, цифровизация будет и дальше преображать мир. Как писал Джордан Шапиро, нам нужно подготовить детей к «другому образу мышления, другому образу жизни, обучения и любви» [16].

Аргумент в пользу развития вычислительного мышления через программирование как цели образования не является аргументом в пользу рабочих мест для разработчиков ПО. Хотя иногда звучит аргумент о формирующемся разрыве между спросом и предложением в среде программистов, этот аргумент не очень убедителен, так как работа программистов во многом автоматизирована или передана машинам. Вполне возможно, что дети, поступающие сегодня в дошкольные учреждения, будут взрослеть в мире квантовых вычислений, мгновенных соединений и других возможностей, которые сложно себе представить. Трудно сказать, какие навыки владения языком программирования будут востребованы через два года, а уж тем более через десять лет. Тем не менее, ускорение процесса цифровизации не ставится под сомнение, как и растущая значимость понятий вычислительного мышления, таких как алгоритмы и сопоставление с паттерном во всех видах работ. Если необходим аргумент в пользу рабочих мест, чтобы обучение детей программированию стало обязательным, его можно сформулировать так: для любой работы потребуется некоторая форма навыков вычислительного мышления, и люди, обладающие такими навыками, смогут быстрее продвинуться по служебной лестнице. Но ключевой аргумент в пользу обучения программированию касается даже не работы, а совсем базового понятия — грамотности.

Идея программирования как новой грамотности подтверждается исследованиями грамотности в форме чтения и письма. История сама показывает нам, как случилось, что сначала чтение, а потом и письмо стали восприниматься как универсальную потребность для повседневной жизни. Исследование, проведенное Аннет Ви и ее коллегами, показывает параллельное развитие чтения и вычислительной грамотности в два этапа — первый этап, когда компьютеры стали частью инфраструктуры. В Советском Союзе наблюдался сильный рывок к всеобщей компьютерной грамотности со стороны высших органов власти. Идеи мечтателя Сеймура Пейперта и язык Logo нашли отклик в стране, которая была на переднем крае развития математики. Однако в то время рано было говорить об идеалах всеобщей компьютерной грамотности. Теперь, благодаря возможности подключения к Интернету и совместной работы с помощью инструментов с

открытым исходным кодом, идея компьютерной грамотности распространяется все быстрее по мере того, как все больше стран включают программирование в учебные программы и развитие вычислительного мышления становится ключевой целью образования.

Понятие грамотности включает различные практические аспекты, связанные с программированием и вычислительным мышлением. Основными из них являются навыки критического мышления и решения задач, особенно совместного решения задач, когда люди работают в группах. Абстрагирование является важным элементом в развитии мыслительных способностей, обучение компьютерному программированию дает маленьким детям возможность развивать абстрактное мышление — «объекты, помогающие думать», как определил их Сеймур Пейперт, автор педагогической идеи конструкционизма. Можно увидеть воплощение или представление мысли на экране, особенно при помощи платформ визуального программирования, таких как Scratch, и затем мыслью можно легко управлять. Здесь присутствует важный аспект равенства, имеющий серьезные последствия для значения образования. Распространение программирования и вычислительного мышления в начальных классах может породить два сопутствующих последствия для демократизации или выравнивания показателей успеваемости: творческие или языковые дисциплины станут доступными для тех, кого раньше могли считать бесталанным, а благодаря таким понятиям, как адаптивный гейм-дизайн, навыки программирования позволят быстрее и легче понимать математические и научные концепции.

5.2 Обучение программированию должно проходить в духе сотрудничества и на основе проектов в рамках школьных и внешкольных занятий. Реализовать программы в Российской Федерации будет проще в ходе внешкольных занятий.

Эта рекомендация основана на долгой истории исследований и разработок, начиная с новаторской работы Эми Брукман с MOOSE Crossing и включая работу создателей и исследователей Computer Clubhouse и Scratch в рамках проекта «Спираль обучения» Media Lab Массачусетского технологического института под руководством Митчела Резника. Работы таких практиков и исследователей, как Карен Бренан, Рикароуз Роке и Мария Умащи-Берс, упоминались в разных главах этой книги. Лучшее описание мышления представлено в названии и содержании книги Резника «Спираль обучения. 4 принципа развития детей и взрослых». Мы используем предложенную Резником структуру «четырех «П», чтобы объяснить, как следует преподавать программирование.

Цель изучения программирования заключается не просто в том, чтобы научиться использовать язык программирования, такой как Java или C +, Python или Ruby. Как объясняет в своей книге ученый-компьютерщик Мартин Эрвиг, вычисление состоит из двух частей — из алгоритма и языка или языков, используемых для выполнения алгоритма [177]. Важно понимать, что освоение вычислительного мышления через программирование — освоение разработки или создания подходящего алгоритма и последующего его выполнения с помощью кода — отличается от изучения традиционных предметов, преподаваемых учителем традиционным способом. Передача фактов в формате лекции и конспектирования или чтения не позволит учащемуся продвинуться дальше. Кроме того, демонстрации процесса по принципу «делай как я» бесполезны, потому что индивидуальные вариации нельзя ограничивать. Только в этом случае они будут иметь смысл, учащиеся будут понимать, что они делают, и знать, зачем они это делают.

Проектно-ориентированное обучение имеет множество преимуществ, но главное из них — мотивация. Если ученику или группе учеников разрешают выбрать или придумать проект, над которым они будут работать, это оказывает огромное положительное влияние на мотивацию. В этом случае процесс обучения становится самостоятельным, поскольку ученик пытается приобрести навыки или знания, необходимые для достижения прогресса в своем проекте. Само

понятие знаний переворачивается с ног на голову — знания перестают быть чем-то существующим в предопределенных структурах, что ученики должны освоить независимо от того, интересно им это или нет, и ученики сами определяют свои интересы и делают мотивированный выбор. Это усложняет работу учителя, о чем мы поговорим в следующей рекомендации.

Пылкая увлеченность — это всего лишь крайняя форма мотивации, а проектно-ориентированное обучение — это лишь один из аспектов такой мотивации. Медленно, но верно старая модель школьного обучения времен промышленной революции, когда ученики одного возраста сидят рядами и изучают определенную учебную программу, затем массово «переходят» в «следующий класс», уступает место обучению XXI века, в котором будет доминировать самовыражение и совместная деятельность. Самовыражение относится к смыслу, который ребенок находит, и к тому самосознанию, которое ребенок создает, взаимодействуя с «объектами, помогающими думать». Ребенок, участвующий в проекте, в который он может привнести свои любимые в данный момент вещи, будь то динозавры, ведьмы или грузовики, будет учиться в зоне потока, как ее описал Михай Чиксентмихайи.

Партнеры или идея обучения во взаимодействии — третья из четырех «П» Резника. Партнеры выполняют множество функций в психологическом и эмоциональном развитии ребенка. В сфере программирования партнеры играют важнейшую роль на всех этапах достижения цели приобретения когнитивных и некогнитивных навыков, включая помощь в определении, каким проектом хочет заниматься ребенок, какую роль он будет играть в этом проекте, как группа будет решать проблемы и конфликты, когда они возникают и как группа будет отмечать достижения и переходить к следующему проекту.

Процесс игры — последняя из четырех «П», но, возможно, самая важная в том смысле, что процесс игры объединяет три другие «П». Игра подразумевает совместное использование и увлекательное времяпрепровождение. Scratch и такие инструменты, как Alice, Pencil Code и т.д., упомянутые в этой книге, являются «средами программирования с низким порогом входа», что означает, что новички смогут быстро научиться и начнут разрабатывать игры и интерактивные истории, которыми можно делиться друг с другом. По мере взросления дети становятся более терпеливыми, но при этом важна возможность делать что-то лично значимое как можно скорее. Скучному процессу преподавания, при котором нужно что-то учить только потому, что учитель знает, почему это важно, почти нет места в эффективном обучении вычислительному мышлению через программирование.

Хотя нет никакого условия, которое бы исключало совместное проектно-ориентированное обучение из традиционной системы образования, скорее всего, существует сильнейшая инертность, которая усложняет такое изменение. Эту идею целесообразно реализовать в рамках внеклассного обучения, обучения в выходные дни и во время каникул, где эти элементы, как правило, уже присутствуют, особенно в кружках или группах по интересам.

5.3. Проблема учителей состоит в том, что нужно менять мировоззрение, а не организовывать обучение, чтобы стать квалифицированными специалистами в области компьютеров. Лучшим инструментом такой трансформации мировоззрения может стать сообщество целеустремленных и преданных своему делу учителей в Российской Федерации, которые могут поддерживать связь через сеть «Летописи».

Самая большая проблема для реализации эффективной программы обучения детей вычислительному мышлению через программирование — это традиционное мировоззрение учителя.

Идея о том, что учитель — это всезнающим специалист, который должен каким-то образом передать свои знания или навыки ученикам посредством подражания или слушания и осознания. В свете целей образования XXI века многие осознают важность перехода учителя от роли «вещателя с кафедры» к роли «подсказчика со стороны». Однако в отношении традиционных предметов это проще сказать, чем сделать, хотя есть модели, например, реджио-педагогика и подход «новая школа», основанные на взаимном и совместном обучении. Поскольку у преподавания программирования не такая длинная история, вполне возможно, что учить программированию по-новому будет проще.

С распространением инструментов и методов и резким ростом популярности таких сайтов, как http://StackExchange.com, появилось много возможностей найти альтернативные решения этой проблемы. Освоение вычислительного мышления обязательно включает процесс устранения проблем и является важной задачей в обучении учащихся. Учитель должен не просто предоставлять учащемуся решение задачи, а помочь ему изучить стратегии или подходы к ее решению. Обычному учителю трудно смириться с тем, что он уже не «самый умный в классе».

Программирование дает учителю возможность получить доступ к различным новым типам мировоззрения. Он может выбрать идеологию, соответствующую его предпочтениям и сильным сторонам. Среди доступных альтернативных идеологий можно упомянуть следующие:

- «идеология мейкера» — идея о том, что можно сделать все что угодно, проявив творческий подход к дизайну и материалам, открытость к изучению опыта других без страха показаться невеждой, вместо того, чтобы пытаться найти решение самостоятельно;
- «идеология исполнителя» — заразительный энтузиазм, который зависит от способности не сдаваться, столкнувшись с неудачами: иногда возникают неразрешимые проблемы, но научиться их распознавать можно только после череды провалов, при этом меняется само понятие «провала», потому что учитель показывает ученикам, как учиться на ошибках, чтобы подход сработал;
- идеология «экологического мышления» или децентрализованное мышление подразумевает осознание того, что сложные модели поведения или структуры не обязательно определяет центральный директивный орган, они могут быть результатом решений уполномоченных децентрализованных подразделений — как происходит, например, в колониях насекомых, пчелиных ульях и, скоро в вашей округе— роях ремонтных дронов;
- «идеология агентивности» — идея, близкая к экологическому мышлению, согласно которой обеспечение свободы выбора группе людей может быть лучшим способом решения задач при создании условий для самоорганизации — отсутствуют комитеты, которые должны принимать решения, отсутствует иерархии для утверждения, и никакие внешние условия не влияют на интерес учителя — что, в свою очередь, обеспечивает возможность для учеников проявлять собственную субъектность.

Едва ли можно обучить вычислительному мышлению через программирование в рамках традиционной педагогики. Точно так же может быть неэффективной и традиционная модель подготовки учителей. Под традиционной моделью подразумевается идея о том, что учителя, прослушав один или два курса лекций, посвященных обучению вычислительному мышлению, смогут преподавать так, как их научили. Нет ничего плохого в обучающих курсах для учителей, если их проводят, руководствуясь идеями участия и совместной деятельности, которые учителя должны будут преподавать. Однако такой учебный курс вряд ли повлияет на мировоззрение. С этой функцией лучше справится сообщество увлеченных учителей-энтузиастов. В некотором смысле, это и есть воплощение идеологии, которую Резник назвал экологическим мышлением. Учителя являются частью сложной системы со множеством вариантов взаимодействий и аккомодаций между ее элементами. В рамках комплексно-системного подхода, в отличие от

линейного подхода, задача состоит в том, чтобы обеспечить системную поддержку для всех учителей, — в качестве интересного примера такого подхода можно привести школьный округ Сауз Файет под Питсбургом, опыт которого был описан в главе 2.

«Летописи» (http://letopisi.org) — национальный образовательный вики-проект, который стартовал в 2006 году. В 2019 году проект насчитывал более 82 000 зарегистрированных участников. Проект продвигает идею использования вики-технологий и сетевых веб-сервисов открытого доступа в российском образовании и открыл «хабы» в более чем 30 регионах. С помощью таких инструментов, как экспорт и импорт страниц, легко переносить материалы из одной вики в другую. Благодаря этому механизму образовательные вики-сайты могут делиться своими материалами. Первоначально сеть Letopisi.org считали ресурсом для творческого продуктивного обучения, с помощью которого учащиеся могут создавать совместные истории о своих учителях, школах и городах. Однако затем в 2008 году проект поддержал первый конкурс индивидуальных проектов, созданных учащимися в среде Scratch. В 2010 году в рамках проекта «Летописи» было показано собрание анимированных Scratch-историй «Жили-были». Результаты конкурса Scratch Collab Challenge 2018, организованного в рамках подготовки к написанию этой книги, также доступны в «Летописи». За последние несколько лет в «Летописи» были опубликованы сборники образовательных стандартов, описания сред и практик обучения. Благодаря специальным расширениям вики эти материалы можно использовать для проведения мероприятий в сообществах. Letopisi.org включает несколько расширений (Scratch-block, Semantic MediaWiki, CollaborativeDiagram), которые превращают этот сайт в дополненные средства публикации. Подход дополненной публикации сам по себе говорит о значимости программирования для учителей и других пользователей. Любой пользователь создаст собственную более полную книгу намного быстрее, если сможет общаться с системой на языке кода.

5.4. Программирование лучше всего вводить в учебную программу, руководствуясь комплексным сквозным подходом. Нет единого языка программирования, такого как Python или JavaScript, который учащиеся должны изучить. Акцент, скорее, следует делать на освоении и применении вычислительного мышления для изучения таких предметов, как физика, история или язык.

Подход к включению освоения вычислительного мышления в другие предметы четко сформулирован в идее «адаптивного гейм-дизайна», хотя другие средства также применимы, например, занятия на базе «мейкинга». Чтобы гейм-дизайн считался образовательной деятельностью не только в смысле освоения навыков вычислительного мышления, он должен обеспечивать перенос навыков учащимися в сферу научных или математических моделей; должен быть основан на автоматически измеримых понятиях, которые легко распознать и использовать [71]. Басавапатна и соавт. приводят примеры того, как игровые концепции можно преобразовать в концепции для изучения наук, — от столкновения агентов до столкновения молекул, от перемещения одного агента другим агентом в игре до моделирования доставки молекул кислорода эритроцитами в разные части тела. Такие инструменты обучения, как AgentSheets и Scratch, позволяют учащимся экспериментировать с кодом, узнавать значение и правдивость научных истин в духе совершения настоящих открытий. Анимация об обратно пропорциональной зависимости давления от объема (закон Бойля-Мариотта), особенно созданная своими руками, обеспечивает личное и запоминающееся представление о законах природы. Как объяснил Энди Ко в своем блоге «Пейперт также утверждал, что ученые и инженеры, применяющие формулу Ньютона (F = ma), не выучили ее, впитав своим умом, а разработали свое понимание значения формулы, основанное на таких предшествующих знаниях, как физический опыт катания на велосипеде или игры в бильярд»[19].

[19] https://medium.com/bits-and-behavior/mindstorms-what-did-papert-argue-and-what-does-it-mean-for-learning-and-education-c8324b58aca4

Связать программирование с наукой и искусством естественным образом можно, создав специализированные микромиры, предназначенные для изучения науки, техники и искусства. Многие из современных сред обучения поддерживают эти функции. Например, существует несколько учебных модулей StarLogo TNG (биология, химия, физика, математика), каждый из которых включает планы уроков, руководство для преподавателей и задания для учащихся — https://education.mit.edu/project/starlogo-tng/. Такой же подход используется в библиотеке моделей NetLogo по физике, химии и биологии — http://www.netlogoweb.org/. В конкурсе Scratch Collab Challenge 2018 использовались Scratch-студии на русском языке по физике, математике и литературе https://scratch.mit.edu/studios/4614548/.

5.5. Подходить к оцениванию вычислительного мышления следует совсем не так, как к традиционному оцениванию. Программирование обеспечивает мгновенную обратную связь, оценивание должно обеспечивать поддержку ученика и учителя в том, что касается наиболее подходящего процесса обучения.

Сама цель оценивания программирования для освоения вычислительного мышления полностью отличается от оценивания других предметов. По традиционным предметам проводят тесты, чтобы проверить, изучили ли или запомнили ли учащиеся определенный материал, а также их способность решить заранее подготовленные задачи. Тесты обычно выполняют внешнюю функцию: они информируют об оцениваемых способностях учащегося, и учащийся может получить обратную связь с некоторой задержкой, например, путем составления рейтинга учащихся и их мотивации учиться усерднее и повышать свой рейтинг. Традиционные тесты проводят с использованием бумаги и ручки или на компьютере, когда учащийся отвечает на ряд вопросов с несколькими вариантами ответов или на открытые вопросы. Сами вопросы имеют различный уровень сложности — в ходе итеративного процесса экзаменатор изучает, как учащиеся отвечают на такие вопросы, отвечают ли они правильно только на простые или также на сложные вопросы, и может дать оценку способности учащегося в тестируемой области. Такой вид тестирования неприемлем для оценивания обучения детей программированию.

Используя терминологию, разработанную Лорной Эрл, мы можем выделить три вида оценивания: оценивание обучения, оценивание для обучения и оценивание как обучение — именно на третьем виде оценивания стоит сосредоточить внимание при обсуждении вычислительного мышления и программирования. Написанный или собранный (как в случае с цветными блоками визуального программирования) фрагмент кода дает мгновенную обратную связь — алгоритм или работает и выполняет свою функцию, или нет. Если алгоритм не работает, учащиеся устраняют ошибки в коде. Если работает, учащийся переходит к следующему шагу. Мотивация уже присутствует в задаче и у учащегося. В рамках оценивания как обучения акцент смещается на учебный процесс — учащемуся самому интересно, какие знания или навыки ему необходимы, чтобы перейти к следующему шагу, и оценивание должно выполнять эту функцию.

Поскольку оценивание длительное время воспринималось как оценивание обучения, проверка освоения материала, это сфера в обучении программированию отмечена либо отсутствием оценивания как такового, либо применением неправильного вида оценивания — например, тесты с выбором нескольких вариантов ответа о следующих этапах в последовательности кода. Автоматизированная оценка — это возможность продвинуться вперед, но для этого необходимы инвестиции в исследования. Такие инструменты, как Dr. Scratch, не дают дополнительной нагрузки на учащегося, но могут обеспечить обратную связь об уровне сложности проекта. В среде «низкого пола, высокого потолка», примером которой является Scratch, такая обратная связь полезна, поскольку она дает учащемуся понять, достиг ли он какого-либо прогресса в ряде проектов. Работа в области анализа паттернов вычислительного мышления (АПВМ) может

поддержать достижения учащегося в «зоне ближайшего потока», но необходимы дальнейшие исследования в разных областях и на разных платформах для обеспечения широкого применения.

Вопросник о задействовании концепции позитивного развития технологий (ПРТ) Марины Умащи-Берс — это еще один вид оценивания, который может быть очень полезен в этой области. Мы упоминаем эти вопросники в рекомендации относительно рынка продуктов. По сути, идея состоит в том, чтобы определить, соответствуют ли используемые средства, тьюторы и процессы тому, что нам известно об эффективном обучении вычислительному мышлению. Вопросник по ПРТ заполняется на основании человеческих наблюдений, однако весьма перспективным является использование автоматических носимых технологий. Командную работу, совместную деятельность и способы ее улучшения можно дополнить оцениванием с помощью таких технологий, как программа Open Badge, разработанная в Media Lab Массачусетского технологического института. Программа способна отражать динамику и качество взаимодействия в группе.

5.6. Родители и другие члены семьи имеют все возможности внести свой вклад в развитие навыков вычислительного мышления — такие центры по внешкольной работе, как кванториумы и технопарки, должны разработать программы, чтобы у родителей были знания и навыки.

Ускорение темпов технологического прогресса вызывает тревогу у родителей, которые понимают, что их дети растут в мире, который во многом отличается от мира их детства, — так происходило на протяжении веков, но стремительный технологический прогресс усилил беспокойство большинства родителей. Родители переживают, что их дети проводят слишком много времени «у экрана» и не уделяют достаточно времени физической активности на свежем воздухе. Некоторые родители, сами зацикленные на своих смартфонах и просмотре ленты в соцсетях, беспокоятся о том, что их дети проводят слишком много времени в видеоиграх, а не в играх без гаджетов. Технофобы и технофилы оттачивают в спорах свои электронные карандаши и «рвут» блоги в поддержку своей позиции.

Некоторые специалисты пытаются найти золотую середину, признавая неотвратимость технологического прогресса. Один из таких специалистов, Девора Хайтнер, приводит различие между детьми потребляющими и детьми творящими — «если мы называем любое использование технологий «проводить время у экрана», мы не можем отличить творчество от потребления» [178]. Она утверждает, что родители могут добиться большего успеха, наставляя своих детей, а не контролируя их, и предлагает другие способы стать «родителем, дружелюбно настроенным к технологиям». Другой специалист, Джордан Шапиро, не придумывает отмазки для детей, играющих в видеоигры. Вместо этого он приводит философию Сократа, нежелающего принимать письменное слово, в доказательство своей правоты — «Платон был бы геймером» [16].

Возможно, специалистам по воспитанию детей было бы проще подчеркнуть необходимость отнестись с пониманием и присоединиться к веселым занятиям детей, но не всем родителям это дается легко. Обучение программированию может происходить в школе, библиотеке, музее, во время внеклассных занятий или в сообществах, которые в рамках педагогического подхода превратили в места для создания цифровых артефактов. Для родителей, у которых нет опыта программирования, игра с детьми, когда те изучают создание анимации в Scratch или похожем инструменте, может стать настоящим открытием. Рикароуз Роке подробно исследовала вопрос участия родителей в программировании в рамках семинаров «Творческое обучение с семьей» [179]. Ее выводы имеют важное значение для кванториумов, технопарков и других мест, где дети учатся кодировать в рамках внеклассных занятий. На основе подробного исследования

Роке были разработаны рекомендации, особенно для родителей с низким уровнем дохода, у которых нет поддержки со стороны родственников или друзей, знакомых с программированием. Эти рекомендации особенно важны, потому что в России родители, как правило, вообще не присутствуют на занятиях по программированию и робототехнике, и эта возможность может быть упущена. Роке считает правильным предоставить возможности для создания семейных проектов, совместного творчества и совместного использования.

5.7. Рынок продуктов для вычислительного мышления переживает бурный рост. Государство может принять меры для поддержания развития рынка посредством институциональной и финансовой поддержки стартапов, а также поддержки ВУЗов в проведении исследований и предоставлении информации потенциальным клиентам.

Сейчас самое время стать предпринимателем, заручившись поддержкой со всех сторон, — будь то высшее образование или бизнес-инкубаторы, которые предоставляют стартапам рабочие места и оказывают юридические и бухгалтерские услуги. Средства из целого ряда источников — от меценатов-инвесторов, венчурных фондов и краудфандинговых платформ — а также традиционные частные инвестиции вливаются в цифровые предприятия для создания таких продуктов, как наборы для занятия робототехникой, и другие платформы, нацеленные на обучение детей программированию. Так как этот рынок динамично развивается и растет, возникает вопрос о возможных действиях со стороны государства — не только с точки зрения пользы, но и с точки зрения отсутствия вреда. Сценарий вмешательства государства осложняется особым явлением, свойственным цифровым продуктам, особенно платформам, которые зависят от связей между людьми, — так называемым эффектом «предпочтительного присоединения» [180]. Когда успех продукта зависит от количества подключенных пользователей (что важно для детей, изучающих программирование, ввиду значения таких моделей, как «используй, меняй, создавай»), очень важно, чтобы продукт быстро распространился в широком масштабе. Особенно в коммерческом плане, когда прибыль с каждого пользователя, как правило, довольно небольшая из-за жесткой конкуренции, единственным способом заработать деньги становится большое число пользователей. Согласно экономической теории, правительству не следует «выбирать победителей» или иным образом вмешиваться в процесс развития рынка. В этом случае для государства возможны три комплекса мер.

Исследования и разработки. Серия прекрасных продуктов, созданных в Media Lab Массачусетского технологического института, была значительно дополнена за счет грантов Национального научного фонда. Поддержка ВУЗов в проведении НИОКР принесет двойную выгоду — поддержка разработки продуктов укрепит потенциал ВУЗов, и у них появятся возможности оказывать поддержку школам и учителям. В случае с Российской Федерацией, где один ВУЗ может быть не таким мощным, как Массачусетский технологический институт, который неизменно входит в пятерку лучших ВУЗов мира, можно предложить оказывать поддержку консорциуму ВУЗов. Государственная поддержка может быть направлена на развитие инициатив по разработке продуктов с открытым исходным кодом, которые позволяют другим работать над их дальнейшим развитием. Такие платформы, как Scratch, смогли привлечь миллионы подписчиков во многом благодаря тому, что они бесплатны для пользователей. Помимо поддержки консорциумов ВУЗов, государство также может поддерживать консорциумы компаний, работающих друг с другом и с ВУЗами. Таким образом, государство помогает обеспечивать общественное благосостояние, не вмешиваясь в развитие рынка. Что касается государственных закупок, следует дать возможность школам и учителям принимать решения на децентрализованном уровне — изменения происходят стремительно, что делает централизованный процесс неэффективным и нерезультативным.

Оценивание. Логическое и осмысленное оценивание успеваемости учащихся — нерешенная задача для детей, изучающих программирование и осваивающих навыки вычислительного мышления. Особенно важно не обременять детей бессмысленными письменными экзаменами, имитирующими тестирование по традиционным предметам, в которых учащиеся из нескольких ответов выбирают правильный. Концепция «оценивание как обучение» все еще находится в зачаточном состоянии, особенно в том, что касается разработки и использования автоматизированного анализа паттернов вычислительного мышления для управления процессом обучения. Так же и носимые вычислительные устройства, такие как открытые бейджи, разработанные в Media Lab Массачусетского технологического института, используются только в рамках отдельных экспериментов, и это та область, в которой ведомства Министерства образования могут оказать поддержку аналогично поддержке традиционного тестирования и разработки тестов.

Информация. Учитывая большой выбор внеклассных занятий, доступных родителям, государство может использовать существующие сервисы, такие как https://bus.gov.ru/pub/home, для предоставления информации об уровне учреждений и сотрудников. Вместо того чтобы предоставлять рейтинги отдельных продуктов, что будет затруднительно ввиду динамичности рынка, органы местного самоуправления могут поддержать разработку рейтингов по пунктам «позитивного развития технологий», представленным в приложении, охватив аспекты коммуникации, сотрудничества, создания сообщества, создания контента, креативности и выбора линии поведения. Федеральные органы власти могут поддержать разработку этих критериев, возможно, совместно с кафедрами ВУЗов, имеющими большой опыт исследований в области развития технологий для детей.

СПИСОК ЛИТЕРАТУРЫ

1. Wing J.M. Computational Thinking // Commun. ACM. 2006. Vol. 49, N° 3. P. 33–35.
2. Krauss J., Prottsman K. Computational Thinking and Coding for Every Student: The Teacher's Getting-Started Guide. Corwin Press, 2016.
3. Balanskat A., Engelhardt K. Computing our future: Computer programming and coding-Priorities, school curricula and initiatives across Europe. European Schoolnet, 2014.
4. Kafai Y.B., Burke Q. Connected Code: Why Children Need to Learn Programming. MIT Press, 2014.
5. Papert S., Harel I. Situating Constructionism // Constructionism. Ablex Publishing Corporation, 1991. P. 193–206.
6. Resnick M. Lifelong Kindergarten: Cultivating Creativity Through Projects, Passion, Peers, and Play. MIT Press, 2017.
7. Ershov A.P. Programming – A Second Literacy (In Russian) // Novosibirsk, USSR Academy of Sciences. 1981.
8. Bogost I. Play anything: The pleasure of limits, the uses of boredom, and the secret of games. Basic Books, 2016.
9. Blau I., Benolol N. Can Designing Self-Representations through Creative Computing Promote an Incremental View of Intelligence and Enhance Creativity among At-Risk Youth? // Interdisciplinary Journal of e-Skills and Lifelong Learning (IJELL). 2016. Vol. 12. P. 267–278.
10. Turkle S., Papert S. Epistemological pluralism and the revaluation of the concrete // Journal of Mathematical Behavior. 1992. Vol. 11, N° 1. P. 3–33.
11. Taleb N.N. Antifragile: how to live in a world we don't understand. Allen Lane London, 2012. Vol. 3.
12. Bers M.U. Coding as a Playground: Programming and Computational Thinking in the Early Childhood Classroom. Routledge, 2017.
13. Scherer R., Siddiq F., Sánchez Viveros B. The cognitive benefits of learning computer programming: A meta-analysis of transfer effects. // Journal of Educational Psychology. 2018.
14. Popat S., Starkey L. Learning to code or coding to learn? A systematic review // Computers & Education. 2019. Vol. 128. P. 365–376.
15. Parandekar S. et al. Enhancing School Quality through Participative and Collaborative Learning. Washington, D.C.: The World Bank, 2017.
16. Shapiro J. The New Childhood: Raising kids to thrive in a digitally connected world. Hodder & Stoughton, 2019.
17. Glendening M.L. From Video Games to Real Life: Tapping into Minecraft to Inspire Creativity and Learning in the Library: Inspiring Creativity and Learning in the Library. ABC-CLIO, 2016.
18. DiSessa A.A. Changing Minds: Computers, Learning, and Literacy. MIT Press, 2001.
19. Afinogenov G. Andrei Ershov and the Soviet Information Age // Kritika: Explorations in Russian and Eurasian History. 2013. Vol. 14, N° 3. P. 561–584.
20. Rushkoff D. Program Or Be Programmed: Ten Commands for a Digital Age. OR Books, 2010. 151 p.
21. Ritzen J. Looking for Eagles: A short guide to bird watching in a educational context // The World Bank. 1999.
22. Page S.E. The Difference: How the Power of Diversity Creates Better Groups, Firms, Schools, and Societies. Princeton University Press, 2007.
23. Anand B. The Content Trap: A Strategist's Guide to Digital Change. Random House Inc., 2016.
24. A is for algorithm // The Economist. 2014. N° April 26.
25. Bocconi S. et al. Developing computational thinking in compulsory education // European Commission, JRC Science for Policy Report. 2016.
26. European Union I. Communication from the Commission to the European Parliament, the Council, the European Economic and Social Committee and the Committee of the Regions. Brussels, 2014.

27. How did Estonia become a leader in technology? // The Economist. 2013.
28. Uzunboylu H., Kinik E., Kanbul S. An Analysis of Countries which have Integrated Coding into their Curricula and the Content Analysis of Academic Studies on Coding Training in Turkey // TEM JOURNAL-TECHNOLOGY EDUCATION MANAGEMENT INFORMATICS. 2017. Vol. 6, N° 4. P. 783–791.
29. The Royal Society. After the Reboot: Computing Education in UK Schools. The Royal Society, 2017.
30. Gee-Spillane S. Participants' views on the effects of digital technologies on their teaching/learning in food and textiles technology education. The University of Waikato, 2018.
31. Gerristen J. Kids to learn how to code before high school. 2017.
32. Blannin J. Coding in the classroom. 2017.
33. Choi H., Lee J. Promoting Computational Thinking and Collaborative Skills in Primary Robotics Classes // Proceedings of the International Conference on Computational Thinking Education 2018. Hong Kong: The Education University of Hong Kong. 2018.
34. Singer N. How Silicon Valley Pushed Coding Into American Classrooms // The New York Times. 2018.
35. Ruberg L.F., Owens A. A Future-Focused Education: Designed to Create the Innovators of Tomorrow // Emerging Research, Practice, and Policy on Computational Thinking / ed. Rich P.J., Hodges C.B. Cham: Springer International Publishing, 2017. P. 367–392.
36. Pan T.-Y. (Tim). Reenergizing CS0 in China // Emerging Research, Practice, and Policy on Computational Thinking / ed. Rich P.J., Hodges C.B. Cham: Springer International Publishing, 2017. P. 351–362.
37. Delcker J., Ifenthaler D. Computational Thinking as an Interdisciplinary Approach to Computer Science School Curricula: A German Perspective // Emerging Research, Practice, and Policy on Computational Thinking / ed. Rich P.J., Hodges C.B. Cham: Springer International Publishing, 2017. P. 49–62.
38. Hiltunen T. Learning and Teaching Programming Skills in Finnish Primary Schools. The Potential of Games. Oulu: University of Oulu, 2016.
39. The Royal Society. Shut Down or Restart? The Way Forward for Computing in UK Schools. 2012.
40. Foxall S., Mist R. Computing Education – 1 year on | Royal Society [Electronic resource]. 2018. URL: https://royalsociety.org/science-events-and-lectures/2018/11/computing-education-1-year-on/ (accessed: 06.03.2019).
41. Dougherty D. The Maker Mindset: Design, Make, Play. Routledge New York, NY, 2013.
42. Brennan K. Social dimensions of computing education // NSF Future Directions in Computing Education Summit. 2014.
43. McMahon M. The Jesuit model of education // EDOCERE: A Resource for Catholic Education. 2004.
44. Trow W.C., Kalachov P.D. Character Education in Soviet Russia. Ann Arbor Press, 1934.
45. Savoia A., Copeland P. Entrepreneurial innovation at Google // Computer. 2011. Vol. 44, N° 4. P. 56–61.
46. Brennan K.A. Best of both worlds: Issues of structure and agency in computational creation, in and out of school: PhD Thesis. Massachusetts Institute of Technology, 2013.
47. Schwartz D.L. The productive agency that drives collaborative learning // In P. Dillenbourg (Ed.), Collaborative learning: Cognitive and computational approaches. NY: Elsevier Science/Permagon, 1999. P. 197–218.
48. Okita S.Y., Schwartz D.L. Learning by teaching human pupils and teachable agents: The importance of recursive feedback // Journal of the Learning Sciences. 2013. Vol. 22, N° 3. P. 375–412.
49. Papert S. Mindstorms: children, computers, and powerful ideas. Basic Books, 1980.
50. Roque R.V. Making together: Creative collaboration for everyone: PhD Thesis. Massachusetts Institute of Technology, 2012.
51. Briscoe G. Digital innovation: The hackathon phenomenon. 2014.
52. Briscoe G., Mulligan C. The hackathon phenomenon. 2014.
53. Abelson H., Sussman G.J., Sussman J. Structure and interpretation of computer programs. Justin Kelly, 1996.

54. Yadav A., Stephenson C., Hong H. Computational Thinking for Teacher Education // Commun. ACM. 2017. Vol. 60, N° 4. P. 55–62.
55. Buss A., Gamboa R. Teacher Transformations in Developing Computational Thinking: Gaming and Robotics Use in After-School Settings // Emerging Research, Practice, and Policy on Computational Thinking / ed. Rich P.J., Hodges C.B. Cham: Springer International Publishing, 2017. P. 189–203.
56. Yadav A. et al. Computational thinking in elementary and secondary teacher education // ACM Transactions on Computing Education (TOCE). 2014. Vol. 14, N° 1. P. 5.
57. Zeeman M. To Scratch or not to Scratch-a Reflection // International Conference on e-Learning. Academic Conferences International Limited, 2013. P. 425.
58. Toikkanen T., Leinonen T. The Code ABC MOOC: Experiences from a Coding and Computational Thinking MOOC for Finnish Primary School Teachers // Emerging Research, Practice, and Policy on Computational Thinking / ed. Rich P.J., Hodges C.B. Cham: Springer International Publishing, 2017. P. 239–248.
59. Davidson C.N. The New Education: How to Revolutionize the University to Prepare Students for a World In Flux. Basic Books, 2017.
60. Kelleher C. Barriers to programming engagement // Advances in gender and education. 2009. Vol. 1, N° 1. P. 5–10.
61. Senghas A., Kita S., Özyürek A. Children creating core properties of language: Evidence from an emerging sign language in Nicaragua // Science. 2004. Vol. 305, N° 5691. P. 1779–1782.
62. Milne L.R., Ladner R.E. Blocks4All: Overcoming Accessibility Barriers to Blocks Programming for Children with Visual Impairments // Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems. ACM, 2018. P. 69.
63. Kane S.K., Bigham J.P. Tracking@ stemxcomet: teaching programming to blind students via 3D printing, crisis management, and twitter // Proceedings of the 45th ACM technical symposium on Computer science education. ACM, 2014. P. 247–252.
64. Hung Y.-X. et al. What do stroke patients look for in game-based rehabilitation: a survey study // Medicine. 2016. Vol. 95, N° 11.
65. Gelfand L., Freed E.C. Sustainable School Architecture: Design for Elementary and Secondary Schools. John Wiley & Sons, 2010.
66. Earl L.M. Assessment as Learning: Using Classroom Assessment to Maximize Student Learning. SAGE Publications, 2012.
67. Akcaoglu M., Koehler M.J. Cognitive outcomes from the Game-Design and Learning (GDL) after-school program // Computers & Education. 2014. Vol. 75. P. 72–81.
68. Román-González M., Moreno-León J., Robles G. Complementary tools for computational thinking assessment // Proceedings of International Conference on Computational Thinking Education (CTE 2017), S. C Kong, J Sheldon, and K. Y Li (Eds.). The Education University of Hong Kong. 2017. P. 154–159.
69. Basawapatna A.R. et al. The zones of proximal flow: guiding students through a space of computational thinking skills and challenges // Proceedings of the ninth annual international ACM conference on International computing education research. ACM, 2013. P. 67–74.
70. Koh K.H. et al. Towards the automatic recognition of computational thinking for adaptive visual language learning // Visual languages and human-centric computing (VL/HCC), 2010 IEEE symposium on. IEEE, 2010. P. 59–66.
71. Basawapatna A. et al. Recognizing computational thinking patterns // Proceedings of the 42nd ACM technical symposium on Computer science education. ACM, 2011. P. 245–250.
72. D'Alba A., Huett K.C. Learning Computational Skills in uCode@ UWG: Challenges and Recommendations // Emerging Research, Practice, and Policy on Computational Thinking. Springer, 2017. P. 3–20.
73. Bales R.F. Interaction Process Analysis: A Method for the Study of Small Groups. University Microfilms, 1965.
74. Gawande A. The Checklist Manifesto: How to Get Things Right. Henry Holt and Company, 2010.
75. Bond T., Fox C.M. Applying the Rasch Model: Fundamental Measurement in the Human Sciences, Third Edition. Taylor & Francis, 2015.

76. D'Alba A., Huett K.C. Learning computational skills in uCode@ UWG: Challenges and recommendations // Emerging research, practice, and policy on computational thinking. Springer, 2017. P. 3–20.
77. Bers M. Positive technological development: Working with computers, children, and the internet // MassPsych. 2007. Vol. 51, N° 1. P. 5–7.
78. Hammond M. What is an affordance and can it help us understand the use of ICT in education? // Education and Information Technologies. 2010. Vol. 15, N° 3. P. 205–217.
79. Bers M., Doyle-Lynch A., Chau C. Positive technological development: The multifaceted nature of youth technology use towards improving self and society // Constructing the self in a digital world. 2012. P. 110–136.
80. Worker S. Bers's Theory of Positive Technological Development // Journal of Youth Development. 2014. Vol. 9, N° 1. P. 170–174.
81. Bers M.U. The TangibleK Robotics Program: Applied Computational Thinking for Young Children // Early Childhood Research & Practice. 2010. Vol. 12, N° 2.
82. Maloney J. et al. The scratch programming language and environment // ACM Transactions on Computing Education (TOCE). 2010. Vol. 10, N° 4. P. 16.
83. Kelleher C., Pausch R. Lowering the Barriers to Programming: A Taxonomy of Programming Environments and Languages for Novice Programmers // ACM Comput. Surv. 2005. Vol. 37, N° 2. P. 83–137.
84. Lye S.Y., Koh J.H.L. Review on teaching and learning of computational thinking through programming: What is next for K-12? // Computers in Human Behavior. 2014. Vol. 41. P. 51–61.
85. Adkins S. The 2016-2021 Global Game-based Learning Market // Serious Play Conference. 2016.
86. Hameed S.S. et al. A School-Wide Approach to Infusing Coding in the Curriculum. // Proceedings of the International Conference on Computational Thinking Education 2018. Hong Kong: The Education University of Hong Kong. 2018.
87. Monroy-Hernández A. Designing for remixing : supporting an Online community of amateur creators: PhD Thesis. 2013.
88. Dasgupta S. et al. Remixing As a Pathway to Computational Thinking // Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing. New York, NY, USA: ACM, 2016. P. 1438–1449.
89. Patarakin Y., Shilova O. Concept of Learning Design for Collaborative Network Activity // Procedia – Social and Behavioral Sciences. 2015. Vol. 214. P. 1083–1090.
90. Harms K.J. et al. Designing a community to support long-term interest in programming for middle school children // Proceedings of the 11th International Conference on Interaction Design and Children. ACM, 2012. P. 304–307.
91. Bau D. et al. Learnable Programming: Blocks and Beyond // Commun. ACM. 2017. Vol. 60, N° 6. P. 72–80.
92. Jenkins H. Convergence Culture: Where Old and New Media Collide. NYU Press, 2008.
93. Repenning A., Ambach J. The agentsheets behavior exchange: Supporting social behavior processing // CHI'97 Extended Abstracts on Human Factors in Computing Systems. ACM, 1997. P. 26–27.
94. Ioannidou A., Repenning A., Webb D.C. AgentCubes: Incremental 3D end-user development // Journal of Visual Languages & Computing. 2009. Vol. 20, N° 4. P. 236–251.
95. Repenning A., Ioannidou A. AgentCubes: Raising the ceiling of end-user development in education through incremental 3D // Visual Languages and Human-Centric Computing, 2006. VL/HCC 2006. IEEE Symposium on. IEEE, 2006. P. 27–34.
96. Repenning A. et al. AgentCubes: Enabling 3D creativity by addressing cognitive and affective programming challenges // EdMedia: World Conference on Educational Media and Technology. Association for the Advancement of Computing in Education (AACE), 2012. P. 2762–2771.
97. Suzuki H., Kato H. Interaction-level support for collaborative learning: AlgoBlock—an open programming language // The first international conference on Computer support for collaborative learning. L. Erlbaum Associates Inc., 1995. P. 349–355.

98. Wyeth P., Purchase H.C. Tangible programming elements for young children // CHI'02 extended abstracts on Human factors in computing systems. ACM, 2002. P. 774–775.
99. Wyeth P., Purchase H.C. Using developmental theories to inform the design of technology for children // Proceedings of the 2003 conference on Interaction design and children. ACM, 2003. P. 93–100.
100. Brusilovsky P.L. Towards an intelligent environment for learning introductory programming // Cognitive models and intelligent environments for learning programming. Springer, 1993. P. 114–124.
101. Brusilovsky P. et al. Mini-languages: a way to learn programming principles // Education and Information Technologies. 1997. Vol. 2, N° 1. P. 65–83.
102. Pattis R.E., Pattis R.P. Karel the Robot: A Gentle Introduction to the Art of Programming. John Wiley & Sons Australia, Limited, 1994.
103. Ruf A., Mühling A., Hubwieser P. Scratch vs. Karel: impact on learning outcomes and motivation // Proceedings of the 9th Workshop in Primary and Secondary Computing Education. ACM, 2014. P. 50–59.
104. Bruckman A.S. MOOSE crossing: construction, community and learning in a networked virtual world for kids. Massachusetts Institute of Technology, 1997.
105. Rogozhkina I., Kushnirenko A. PiktoMir: teaching programming concepts to preschoolers with a new tutorial environment // Procedia-Social and Behavioral Sciences. 2011. Vol. 28. P. 601–605.
106. Kay A., Goldberg A. The dynabook: past, present, and future // A history of personal workstations. ACM, 1988. P. 249–264.
107. Resnick M. StarLogo: an environment for decentralized modeling and decentralized thinking // Conference companion on Human factors in computing systems: common ground. New York, NY, USA: ACM, 1996. P. 11–12.
108. Klopfer E., Scheintaub H. StarLogo TNG: Science in Student-programmed Simulations // Proceedings of the 8th International Conference on International Conference for the Learning Sciences – Volume 3. Utrecht, The Netherlands: International Society of the Learning Sciences, 2008. P. 59–60.
109. Klopfer E. et al. The simulation cycle: Combining games, simulations, engineering and science using StarLogo TNG // E-Learning and Digital Media. 2009. Vol. 6, N° 1. P. 71–96.
110. Liu J. et al. Making games a snap with Stencyl: a summer computing workshop for K-12 teachers // Proceedings of the 45th ACM technical symposium on Computer science education. ACM, 2014. P. 169–174.
111. Koushik V., Kane S.K. Tangibles+ Programming+ Audio Stories= Fun // Proceedings of the 19th International ACM SIGACCESS Conference on Computers and Accessibility. ACM, 2017. P. 341–342.
112. Weintrop D., Killen H., Franke B. Blocks or text? How programming language modality makes a difference in assessing underrepresented populations // Rethinking Learning in the Digital Age: Making the Learning Sciences Count, 13th International Conference of the Learning Sciences (ICLS). 2018.
113. Weintrop D., Wilensky U. To block or not to block, that is the question: students' perceptions of blocks-based programming // Proceedings of the 14th International Conference on Interaction Design and Children. ACM, 2015. P. 199–208.
114. Wilkie J., Good J. Tica: an environment for exploring tangible vs. screen-based programming // 28th Annual Workshop of the Psychology of Programming Interest Group-PPIG 2017: Proceedings. Psychology of Programming Interest Group, 2017. P. 65–75.
115. Hu F. et al. Strawbies: explorations in tangible programming // Proceedings of the 14th International Conference on Interaction Design and Children. ACM, 2015. P. 410–413.
116. Chawla K. et al. Dr. Wagon: a'stretchable'toolkit for tangible computer programming // Proceedings of the 12th international conference on interaction design and children. ACM, 2013. P. 561–564.
117. Horn M.S., Jacob R.J. Tangible programming in the classroom with tern // CHI'07 extended abstracts on Human factors in computing systems. ACM, 2007. P. 1965–1970.
118. Silver J.S., Rosenbaum E. Twinkle: programming with color // Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction. ACM, 2010. P. 383–384.

119. Price T.W. et al. Evaluation of a frame-based programming editor // Proceedings of the 2016 ACM Conference on International Computing Education Research. ACM, 2016. P. 33–42.
120. Budd J. et al. PageCraft: learning in context a tangible interactive storytelling platform to support early narrative development for young children // Proceedings of the 6th international conference on Interaction design and children. ACM, 2007. P. 97–100.
121. Yang H., Zhang L. Promoting Creative Computing: origin, scope, research and applications // Digital Communications and Networks. 2016. Vol. 2, N° 2. P. 84–91.
122. Bau D., Bau D.A. A Preview of Pencil Code: A Tool for Developing Mastery of Programming // Proceedings of the 2nd Workshop on Programming for Mobile & Touch. ACM, 2014. P. 21–24.
123. Heines J.M. Converting MIDI notes to ABC notes in pencil code // ACM Inroads. 2016. Vol. 7, N° 2. P. 84–84.
124. Sullivan A.A., Bers M.U., Mihm C. Imagining, Playing, and Coding with KIBO: Using Robotics to Foster Computational Thinking in Young Children // Siu-cheung KONG The Education University of Hong Kong, Hong Kong. 2017. P. 110.
125. The Routledge Companion to Remix Studies / ed. Eduardo Navas, Owen Gallagher, xtine burrough. Routledge, 2014.
126. Knobel M., Lankshear C. Remix: The Art and Craft of Endless Hybridization // Journal of Adolescent & Adult Literacy. 2011. Vol. 52, N° 1. P. 22–33.
127. Navas E. Remix theory: The aesthetics of sampling. Birkhäuser, 2014.
128. Smith A. et al. Remix as Professional Learning: Educators' Iterative Literacy Practice in CLMOOC // Education Sciences. 2016. Vol. 6, N° 4. P. 12.
129. Liu C.-C. et al. A remix-oriented approach to promoting student engagement in a long-term participatory learning program // Computers & Education. 2017. Vol. 110. P. 1–15.
130. Dodds C. et al. Remix Portal: Connecting Classrooms with Local Music Communities // Proceedings of the 8th International Conference on Communities and Technologies. New York, NY, USA: ACM, 2017. P. 203–212.
131. Kallinikos J., Aaltonen A., Marton A. A theory of digital objects // First Monday. 2010. Vol. 15, N° 6.
132. Blikstein P., Krannich D. The Makers' Movement and FabLabs in Education: Experiences, Technologies, and Research // Proceedings of the 12th International Conference on Interaction Design and Children. New York, NY, USA: ACM, 2013. P. 613–616.
133. Turchin V.F. The phenomenon of science. Columbia University Press New York, 1977.
134. Lotman Y.M. Universe of the Mind: A Semiotic Theory of Culture. I.B.Tauris, 2001.
135. Guzdial M. Programming environments for novices // Computer science education research. 2004. Vol. 2004. P. 127–154.
136. Resnick M. Thinking Like a Tree (and Other Forms of Ecological Thinking) // International Journal of Computers for Mathematical Learning. 2003. Vol. 8, N° 1. P. 43–62.
137. Brennan K., Hernández A.M., Resnick M. Scratch: creating and sharing interactive media // Proceedings of the 9th international conference on Computer supported collaborative learning – Volume 2. International Society of the Learning Sciences, 2009. P. 217–217.
138. Fowler A., Cusack B. Kodu game lab: improving the motivation for learning programming concepts // Proceedings of the 6th International Conference on Foundations of Digital Games. ACM, 2011. P. 238–240.
139. Stolee K.T., Fristoe T. Expressing computer science concepts through Kodu game lab // Proceedings of the 42nd ACM technical symposium on Computer science education. ACM, 2011. P. 99–104.
140. Petri A. et al. Pocket Game Jams: a Constructionist Approach at Schools // Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct. ACM, 2015. P. 1207–1211.
141. Slany W. Pocket code: a scratch-like integrated development environment for your phone // Proceedings of the companion publication of the 2014 ACM SIGPLAN conference on Systems, Programming, and Applications: Software for Humanity. ACM, 2014. P. 35–36.
142. Lerner R.M. Agent-Based Modeling as a Social Activity. Northwestern University, 2014.
143. Kölling M. Introduction to Programming with Greenfoot: Object-Oriented Programming in Java with Games and Simulations. Pearson, 2015.

144. Kelleher C., Pausch R. Using Storytelling to Motivate Programming // Commun. ACM. 2007. Vol. 50, N° 7. P. 58–64.
145. UNESCO. Cracking the code: girls' and women's education in science, technology, engineering and mathematics (STEM). UNESCO, 2017.
146. Fraillon J. et al. Preparing for life in a digital age: The IEA International Computer and Information Literacy Study international report. Springer, 2014.
147. Duncan C., Bell T., Tanimoto S. Should your 8-year-old learn coding? // Proceedings of the 9th Workshop in Primary and Secondary Computing Education. ACM, 2014. P. 60–69.
148. Cooper S., Dann W., Pausch R. Teaching objects-first in introductory computer science // ACM SIGCSE Bulletin. ACM, 2003. Vol. 35. P. 191–195.
149. Eguíluz A., Garaizar P., Guenaga M. An Evaluation of Open Digital Gaming Platforms for Developing Computational Thinking Skills // Simulation and Gaming. InTech, 2018.
150. Yu J., Roque R. A survey of computational kits for young children // Proceedings of the 17th ACM Conference on Interaction Design and Children. ACM, 2018. P. 289–299.
151. Raskin J.D. Constructivism in psychology: Personal construct psychology, radical constructivism, and social constructionism // American communication journal. 2002. Vol. 5, N° 3. P. 1–25.
152. Gaines B.R., Shaw M.L.G. Sociocognitive Inquiry // Social Network Mining, Analysis, and Research Trends: Techniques and Applications / ed. Ting I.-H., Hong T.-P., Wang L.S.-L. IGI Global, 2012. P. 35–55.
153. Shaw M.L. Recent advances in personal construct technology. Academic Press, 1981.
154. Krainev I., Cheremnykh N. Academician Andrei Ershov and his Archive // Perspectives on Soviet and Russian Computing. Springer, 2011. P. 117–125.
155. Khenner E., Semakin I. School subject informatics (computer science) in Russia: Educational relevant areas // ACM Transactions on Computing Education (TOCE). 2014. Vol. 14, N° 2. P. 14.
156. Blikstein P. Seymour Papert's Legacy: Thinking about learning, and learning about thinking // Seymour Papert Tribute at IDC 2013. 2013.
157. Wierzbicka A. Semantics, Culture, and Cognition: Universal Human Concepts in Culture-Specific Configurations. Oxford University Press, 1992.
158. Brusilovsky P. et al. Teaching Programming to Novices: A Review of Approaches and Tools. Vancouver, BC, Canada., 1994. P. 103–109.
159. Blaho A., Kalas I., Tomcsanyi P. Comenius Logo: Environment for teachers and Environment for learners // Proceedings of the EUROLOGO. 1993. Vol. 93.
160. Turcsányi-Szabó M. Approaching Arts through Logo // Proceedings of the Sixth European Logo Conference. 1997. P. 284–294.
161. Sendov B., Sendova E. East orWest—GEOMLAND is Best, or Does the Answer Depend on the Angle? // Computers and exploratory learning. Springer, 1995. P. 59–78.
162. Soprunov S., Yakovleva. The Russian School System and the Logo Approach: Two Methods Worlds Apart // Logo Philosophy and Implementation. Logo Computer Systems Incorporated, 1999.
163. Parmentier C., Pervin Y. Les nouvelles technologies informatiques dans l'enseignement, un vecteur de la perestroïka // Bulletin de l'EPI (Enseignement Public et Informatique). 1992. N° 67. P. 209–219.
164. Levi-Strauss C. Totemism. Beacon Press, 2016. 119 p.
165. Patarakin E., Visser Y.L. Creativity and creative learning in the context of electronic communication networks: A framework for analysis of practice and research // ResearchGate. 2016.
166. Patarakin E.D. New educational BBS established in Russia // Internet Society News. 1993. N° 1. P. 14–14.
167. Dickinson S. Logo Telecommunications: Crossing Boundaries and Opening Minds. 1995. P. 16.
168. Wenger E. Communities of Practice: Learning, Meaning, and Identity. Cambridge University Press, 1999. 340 p.
169. Sendova E. You do – you understand, you explore – you invent: the fourth level of the inquiry-based learning // Contructionism and Creativity, Proceedings of the 3rd International Construc-tionism Conference. 2014. P. 103–112.
170. Resnick M. Turtles, termites, and traffic jams: explorations in massively parallel microworlds. MIT Press, 1997.

171. Bruckman A., Jensen C. The Mystery of the Death of MediaMOO // Building Virtual Communities: Learning and Change in Cyberspace. 2002. P. 21.
172. Patarakin E., Yarmakhov B. Wiki for designing games at summer camp // Proceedings of the 7th international conference on Games + Learning + Society Conference. Pittsburgh, PA, USA: ETC Press, 2011. P. 268–271.
173. Harrington H.A. et al. Commentary: Teach network science to teenagers // Network Science. 2013. Vol. 1, N° 2. P. 226–247.
174. Sánchez A., Brändle C. More network science for teenagers // arXiv preprint arXiv:1403.3618. 2014.
175. Sheetz L., Dunham V., Cooper J. Professional development for network science as a multi-disciplinary curriculum tool // Integrated STEM Education Conference (ISEC), 2015 IEEE. IEEE, 2015. P. 178–182.
176. Patarakin E.D. Wikigrams-Based Social Inquiry // Digital Tools and Solutions for Inquiry-Based STEM Learning. IGI Global, 2017. Vol. 1. P. 112–138.
177. Erwig M. Once Upon an Algorithm: How Stories Explain Computing. MIT Press, 2017. 333 p.
178. Heitner D. Screenwise: Helping Kids Thrive (and Survive) in Their Digital World. Taylor & Francis, 2016.
179. Roque R.V. Family creative learning: designing structures to engage kids and parents as computational creators. Massachusetts Institute of Technology, 2016.
180. Barabási A.-L., Albert R. Emergence of scaling in random networks // Science. 1999. Vol. 286, N° 5439. P. 509–512.

ПРИЛОЖЕНИЯ

Приложение 1. Изучение отношения к работе с применением компьютера

Категория	Вопросы
Определение	Вычислительное мышление — это понимание принципов работы компьютеров
	Вычислительное мышление предполагает логическое мышление для решения задач
	Вычислительное мышление предполагает использование компьютеров для решения задач
	Вычислительное мышление предполагает абстрактное восприятие общих принципов и применение их в других ситуациях
Комфорт	Я не думаю, что можно применять знания в области программирования для решения других задач
	Мне не нравится изучать концепции программирования
	Я могу получить хорошие оценки («удовлетворительно» и выше) на компьютерных курсах
	Я могу научиться понимать концепции программирования
	Я не использую навыки работы на компьютере ежедневно
	Я сомневаюсь, что у меня есть навыки для решения задач с применением компьютера
Интерес	Я думаю, что компьютерные науки скучны
	Мне интересно решать задачи с помощью компьютерных наук
	Я думаю, что компьютерные науки интересны
	Я бы добровольно посещал/-ла компьютерные курсы, если бы у меня была возможность
Занятия в классе	Вычислительное мышление можно включить в занятия в классе, используя компьютеры на уроке
	Вычислительное мышление можно включить в занятия в классе, позволив ученикам решать задачи
Карьера	Умение работать на компьютере позволит мне получить лучшую работу
	Для достижения моих карьерных целей мне не нужны навыки работы на компьютере
	Я полагаю, что навыки работы на компьютере помогут мне в достижении моих карьерных целей
	Я надеюсь, что в моей будущей профессиональной деятельности мне придется использовать концепции программирования
	Наличие базовых знаний и понимания компьютерных наук ценно само по себе

Источник: [56].

Created  at Tufts

Приложение 2а. Вопросник о задействовании концепции позитивного развития технологий (ПРТ): дети / ребенок

Что представляет собой вопросник?

Вопросник о задействовании концепции ПРТ основан на теоретическом обосновании концепции позитивного развития технологий (ПРТ). Эта концепция лежит в основе разработки, реализации и оценки образовательных программ, в которых используются новые технологии, способствующие обучению как аспекту позитивного развития молодежи. Концепция ПРТ является естественным продолжением идей компьютерной и технологической грамотности, которые повлияли на мир образования, но добавляет к когнитивным компонентам социально-психологические и этические компоненты. С теоретической точки зрения, концепция ПРТ представляет собой междисциплинарный подход, который объединяет идеи из области связи с использованием компьютеров, обучения при взаимодействии с компьютерной поддержкой и конструкционистской теории обучения, разработанной Сеймуром Пейпертом (1993), и рассматривает их в свете исследований в области прикладного развития наук и позитивного развития молодежи.

В качестве теоретической концепции ПРТ включает шесть моделей позитивного поведения (шесть компонентов), которые следует включить в образовательные программы, использующие новые технологии и инновации, например, пространство для мейкеров в школе Элиот-Пирсон. В число таких моделей поведения входят: коммуникация, сотрудничество, создание сообщества, создание контента, креативность и выбор линии поведения.

Более подробная информация о концепции ПРТ представлена в книге Марины Умащи Берс «Проектирование цифровых впечатлений для позитивного развития молодежи: от детского манежа до детской площадки» (2012).

Как использовать вопросник?

Вопросник о задействовании концепции ПРТ следует использовать в различных ситуациях, в которых дети используют технологии. Вопросник разделен на шесть частей (каждая из которых представляет модель поведения, описанную в концепции ПРТ) и оценивается по 5-балльной шкале Лайкерта. Вопросник предназначен для оценивания группы детей или одного ребенка, который работает в пространстве. Взрослые могут использовать вопросник несколько раз в течение урока или один раз в течение учебного периода. Вопросник ПРТ призван помочь составить представление о том, как дети взаимодействуют с пространством и экспериментируют с моделями поведения, описанными в концепции ПРТ.

 **Коммуникация**	**1 Никогда**	**2 Почти никогда**	**3 Иногда**	**4 Часто**	**5 Всегда**	**Нет данных или наблюдений**
Дети наблюдают и/или используют работу друг друга – Дети смотрят, как другие работают над проектом – Дети прикасаются или играют с проектами друг друга во время работы						
Дети играют друг с другом или разговаривают друг с другом – Дети разговаривают или подписываются друг на друга – Дети спрашивают друг друга, что они делают, просят передать инструменты и т.д.						
Дети вступают в разговор (вербально или невербально) с взрослыми – Дети говорят, кивают головой и т.д., когда взрослые делают паузу, чтобы дать им возможность ответить						
Комментарии:						

 **Сотрудничество**	**1 Никогда**	**2 Почти никогда**	**3 Иногда**	**4 Часто**	**5 Всегда**	**Нет данных или наблюдений**
Дети делятся инструментами / материалами – Дети пользуются материалами и возвращают их по завершении работы – Дети не «собирают» инструменты, которыми они не пользуются						
Дети работают вместе над одним проектом – Дети активно участвуют в одной игре / работе – Дети добавляют элементы в один проект						
Дети обращаются за помощью к сверстникам – Дети просят друг друга держать вещи в одном месте, просят помочь им передвинуть предметы и т.д. – Дети высказывают предложения друг другу, просят друг друга показать им, как они что-то сделали и т.д.						
Комментарии:						

 **Создание сообщества**	**1** **Никогда**	**2** **Почти никогда**	**3** **Иногда**	**4** **Часто**	**5** **Всегда**	**Нет данных или наблюдений**
Дети делятся работой с другими детьми – Дети показывают работу сверстникам или помощникам – Дети размещают работу на видном месте в пространстве (или просят / разрешают помощникам показать их работу)						
Дети работают над проектами, связанными с местной средой – Дети создают роботов, чтобы помочь своей школе, скребки для уборки снега в своем районе и т.д. – Дети читают книги, задают вопросы или устраивают ролевые игры на тему местных праздников, событий или локаций						
Дети относятся друг к другу тепло и дружелюбно – Дети спрашивают друг друга о доме или обсуждают факты биографии друг друга (например, «Я видел твою сестру в коридоре») – Дети смеются и вместе играют						
Комментарии:						

 **Создание контента**	**1** **Никогда**	**2** **Почти никогда**	**3** **Иногда**	**4** **Часто**	**5** **Всегда**	**Нет данных или наблюдений**
Дети собирают инструменты и материалы без посторонней помощи – Дети тянутся к поделкам или включают компьютеры без посторонней помощи – Дети сами открывают ящики или выносят инструменты						
Дети проявляют сосредоточенность и внимательность во время работы – Дети строят башню, разукрашивают и мастерят цветок, программируют историю – Дети стабильно работают над этим проектом в течение нескольких минут						
Дети работают над проектами с помощью итераций – Дети возвращаются к проектам снова и снова, через одно или несколько занятий – Дети вносят изменения в свои артефакты, не приступив к работе над новым артефактом						
Комментарии:						

 **Креативность**	**1 Никогда**	**2 Почти никогда**	**3 Иногда**	**4 Часто**	**5 Всегда**	**Нет данных или наблюдений**
Дети наблюдают, прикасаются или работают с предметами в пространстве – Дети проводят пальцами по плюшевой подушке или рассматривают стеклянные бусы на свету – Дети комментируют свойства объекта (например, «Эта древесина шершавая»)						
Дети используют различные материалы в процессе работы – Дети создают коллаж из проволоки, блесток и ткани – Дети смешивают материалы из разных областей (например, блоки с поделками)						
Дети пробуют разные подходы в рамках одной задачи – Дети пробуют прикрепить бумагу скотчем, скрепками и клеевым пистолетом – Дети пробуют построить башню на столе, стуле и подушке						
Комментарии:						

 **Выбор линии поведения**	**1 Никогда**	**2 Почти никогда**	**3 Иногда**	**4 Часто**	**5 Всегда**	**Нет данных или наблюдений**
Дети с осторожностью обращаются с инструментами / материалами – Дети осторожно держат / пользуются ножницами, стеклом и т.д. – Дети безопасно возвращают опасные инструменты после их использования						
Дети проявляют уважение к пространству – Дети убирают материалы после работы – Дети безопасно используют мебель, технологии и т.д.						
Дети проявляют уважение друг к другу – Дети ждут своей очереди, делятся материалами и дают друг другу место – Дети проявляют черты характера (например, обнимают плачущего, помогают другим навести порядок)						
Комментарии:						

Created  at Tufts

Приложение 2b. Вопросник о задействовании концепции позитивного развития технологий (ПРТ): среда и помощник

Что представляет собой вопросник?

Вопросник о задействовании концепции ПРТ основан на теоретическом обосновании концепции позитивного развития технологий (ПРТ). Эта концепция лежит в основе разработки, реализации и оценки образовательных программ, в которых используются новые технологии, способствующие обучению как аспекту позитивного развития молодежи. Концепция ПРТ является естественным продолжением идей компьютерной и технологической грамотности, которые повлияли на мир образования, но добавляет к когнитивным компонентам социально-психологические и этические компоненты. С теоретической точки зрения, концепция ПРТ представляет собой междисциплинарный подход, который объединяет идеи из области связи с использованием компьютеров, обучения при взаимодействии с компьютерной поддержкой и конструкционистской теории обучения, разработанной Сеймуром Пейпертом (1993), и рассматривает их в свете исследований в области прикладного развития наук и позитивного развития молодежи.

В качестве теоретической концепции ПРТ включает шесть моделей позитивного поведения (шесть компонентов), которые следует включить в образовательные программы, использующие новые технологии и инновации, например, пространство для мейкеров в школе Элиот-Пирсон. В число таких моделей поведения входят: коммуникация, сотрудничество, создание сообщества, создание контента, креативность и выбор линии поведения.

Более подробная информация о концепции ПРТ представлена в книге Марины Умащи Берс «Проектирование цифровых впечатлений для позитивного развития молодежи: от детского манежа до детской площадки» (2012).

Как использовать вопросник?

Вопросник о задействовании концепции ПРТ следует использовать в различных ситуациях, в которых дети используют технологии. Вопросник разделен на шесть частей (каждая из которых представляет модель поведения, описанную в концепции ПРТ) и оценивается по 5-балльной шкале Лайкерта. Вопросник предназначен для оценивания среды обучения и учителя/помощника в процессе работы в пространстве. Взрослые могут использовать вопросник несколько раз в течение урока или один раз в течение учебного периода. Вопросник ПРТ призван помочь составить представление о том, как дети взаимодействуют с пространством и экспериментируют с моделями поведения, описанными в концепции ПРТ.

 **Коммуникация**	**1 Никогда**	**2 Почти никогда**	**3 Иногда**	**4 Часто**	**5 Всегда**	**Нет данных или наблюдений**
Организация пространства позволяет детям видеть работу друг друга – Между пространствами установлены низкие барьеры или их нет вообще						
Организация пространства позволяет детям разговаривать друг с другом – Рабочие места организованы так, чтобы дети могли видеть друг друга – Дети в разных местах общаются, не повышая голос						
Помощник(-и) вовлекает детей в двусторонние беседы – Помощник задает детям открытые вопросы – Помощник делает паузу, чтобы дети могли ответить						
Комментарии:						

 **Сотрудничество**	**1 Никогда**	**2 Почти никогда**	**3 Иногда**	**4 Часто**	**5 Всегда**	**Нет данных или наблюдений**
Организация пространства способствует обмену инструментами / материалами – Материалы размещают в центре, чтобы дети могли воспользоваться ими и положить на место – Инструменты не предназначены для использования одним ребенком (например, тяжелые блоки, для которых нужны два человека, инструменты с множеством деталей или ступеней)						
Организация пространства позволяет нескольким детям работать над одним проектом – Рабочие места занимают большую площадь, оснащены круглыми столами с несколькими стульями и т.д.						
Помощник призывает детей работать вместе – Помощник предлагает детям обратиться за помощью к сверстникам						
Комментарии:						

 Создание сообщества	1 Никогда	2 Почти никогда	3 Иногда	4 Часто	5 Всегда	Нет данных или наблюдений
Дети оставляют свой след в пространстве – Есть изображения детей, использующих пространство – Есть предметы, подписанные именем их создателя, или история о том, как это было сделано						
В пространстве чувствуется влияние местных особенностей или контекста – Есть изображения, карты и т.д., которые служат отсылкой к району, школе или городу, где располагается пространство – Есть материал, представляющий праздники, сезоны или события местной культуры						
У помощника складываются теплые и дружеские отношения с детьми – Помощник ведет личные беседы с детьми (например, спрашивает ученика о его новом питомце) – Помощник оказывает поддержку и поощряет детей в процессе работы						
Комментарии:						

 Создание контента	1 Никогда	2 Почти никогда	3 Иногда	4 Часто	5 Всегда	Нет данных или наблюдений
Инструменты и материалы видны и доступны детям – Материалы находятся в прозрачных контейнерах, низких корзинах или стоят на столах – Материалы хранятся на высоте роста детей и доступны детям						
В пространстве предусмотрены локации для представления или документирования детских работ – Есть фотографии и пояснения к детским работам – Предсмотрено место для размещения проектов в разработке						
Помощник моделирует создание контента – Помощник работает над проектами вместе с детьми или помогает, когда его просят – Помощник открыто делится ошибками и показывает, как их исправить						
Комментарии:						

 **Креативность**	**1 Никогда**	**2 Почти никогда**	**3 Иногда**	**4 Часто**	**5 Всегда**	**Нет данных или наблюдений**
Есть места и материалы в пространстве, поддерживающие идею чуда – В комнате расставлены красивые, красочные или текстурированные предметы – Предусмотрены уютные, мягкие или приватные места для детей						
Есть материалы / инструменты, которые можно использовать по-разному – Есть корзины с разными поделками – Есть доказательства того, что один и тот же материал / инструмент используют по-разному						
Помощник поощряет разнообразные подходы в работе детей – Помощник моделирует, используя критерии для сравнения работ (например, «Я считаю, что этот метод более X, но другая твоя идея более Y») – Помощник позволяет детям менять идеи в процессе работы						
Комментарии:						

 **Выбор линии поведения**	**1 Никогда**	**2 Почти никогда**	**3 Иногда**	**4 Часто**	**5 Всегда**	**Нет данных или наблюдений**
Детям предлагают инструменты / материалы, которые нужно использовать с осторожностью – Ломкие или хрупкие материалы и контейнеры – Инструменты с острыми краями или требующие особого внимания						
В пространстве есть ощущение ценностей тех, кто его использует – Есть проекты или рисунки на стенах, связанные со словами, отражающими ценности (например, уважение, забота) – Есть таблицы с правилами или изображения с вариантами поведения						
Помощник вовлекает детей в дружелюбную беседу о выборе – Помощник сдерживает детей, которые не хотят наблюдать и общаться друг с другом, чтобы найти решение – Помощник вырабатывает у детей характер, например: «Ты из тех девочек, которые знают, как быть добрыми со своими друзьями»						
Комментарии:						

Приложение 3. База данных инструментов для обучения детей программированию

		Общая информация			1. Тип языка программирования			2. Создание контента (назначение)													3. Доступ			4. Поддержка сообществ				5. Особые соображения				
#	Инструмент	Разработчик	Дата выпуска	Возрастная группа	Текстовое программирование	Визуальное программирование	Осязаемое программирование	Сочинение	Изучение кода, играя в игры	Разработка игр	Разработка приложений	Произведения искусства	Программирование учебных роботов	Развитие основных навыков вычислит. Мышления	Изучение традиц. языков кодирования	Программирование носимых технологий	Без экрана	Бесплатно / открытый код	Платно	Доступ с основных аппаратных / операционных систем	Размещение проектов	Создание ремиксов	Традиц. текстовые форумы как онлайн-сообщества	Интерактивное онлайн-сообщество	Доступность на русском	Гендерный аспект	Поддержка учителей родителей	3D-элементы	Поддержка генерации идей			
1	Scratch	Массачусетский технологический институт	2007	8 - 16 лет		X		X										X		X	X	X		X			X					
2	Alice	Университет Карнеги-Мелло	2004	12 - 19 лет		X				X								X		X			X		X			X	X			
3	MIT App Inventor	Массачусетский технологический институт	2012	12 - 19 лет		X					X							X		X					X							
4	Thunkable	Два предпринимателя (студенты Массачусетского технологического института)	2016	12 - 19 лет		X					X							X		X			X								X	
5	GameBlox	Массачусетский технологический институт	2014	12 - 19 лет		X				X								X		X												
6	Hopscotch	Компания-разработчик технологий обучения	2013	7 - 13 лет		X				X								X	X		X	X					X					
7	Snap	Калифорнийский университет в Беркли		15 - 19 лет		X		X										X		X												
8	Stencyl	Компания Stencyl LLC	2011	9 - 19 лет		X				X								X	X	X			X		X		X			X		
9	Gamefroot	Instinct Entertainment – новоз	2012	8 - 17 лет		X				X								X		X	X	X					X					
10	GameMaker Studio	YoYo Games в Шотландии	2012	8 - 19 лет	X	X				X									X	X							X					
11	Codemoji	Codemoji, американская компания	2016	8 - 14 лет	X	X									X			X	X	X							X					
12	Pocket Code	Catrobat – австрийская компания	2013	8 - 14 лет		X					X							X			X	X		X	X		X					
13	GameSalad	Gamesalad, Inc. - американская компания	2010	12 - 19 лет		X				X	X								X	X			X				X					
14	Contruct	Scirra Ltd, английская компания	2011	10 - 19 лет		X				X								X	X	X			X		X		X					
15	WaterBear	Индивидуальные разработчики	2011	10 - 16 лет		X		X										X		X												
16	Gamestar Mechanic	Висконсинский университет в Мадисоне в рамках государственно-частного партнерства	2010	7 - 14 лет		X			X	X								X	X	X	X			X			X					
17	AgentCubes	Университет Колорадо и AgentSheets Inc	2008	8 - 21 лет		X				X									X	X							X	X				
18	Kodu	Microsoft	2009	9 - 14 лет						X				X				X			X		X	X	X		X	X				
19	Move the Turtle	Компания Next is Great	2013	5 - 11 лет										X					X													
20	Daisy the Dinosaur	Hopscotch	2011	6 - 8 лет		X								X				X														

		Общая информация			1. Тип языка программирования			2. Создание контента (назначение)										3. Доступ			4. Поддержка сообществ				5. Особые соображения				
#	Инструмент	Разработчик	Дата выпуска	Возрастная группа	Тактовое программирование	Визуальное программирование	Осязаемое программирование	Сочинение	Изучение кода, играя в игры	Разработка игр	Разработка приложений	Произведения искусства	Программирование учебных роботов	Развитие основных навыков вычислит. Мышления	Изучение традиц. языков кодирования	Программирование носимых технологий	Без экрана	Бесплатно / открытый код	Платно	Доступ с основных аппаратных / операционных систем	Размещение проектов	Создание ремиксов	Традиц. текстовые форумы как онлайн-сообщества	Интерактивное онлайн-сообщество	Доступность на русском	Гендерный аспект	Поддержка учителей / родителей	3D-элементы	Поддержка генерации идей
21	Looking Glass	Университет Вашингтона в Сент-Луисе	2010	10 - 15 лет		X		X										X		X	X	X		X				X	X
22	Starlogo TNC	Группа STEP Массачусетского технологического института	2008	9 - 20 лет		X				X								X		X								X	
23	Greenfoot	Кентский университет и компания Oracle	2006	14 - 19 лет	X	X									X			X		X				X	X		X		
24	Pencil Code	Группа лиц		14 - 18 лет	X	X						X						X		X							X		
25	Blockly Games	Google	2014	8 - 14 лет	X	X			X									X		X					X				
26	AppLab	Code.org	2015	13 - 19 лет	X	X					X							X		X	X	X		X			X		
27	Tynker	Tynker, американская компания	2013	7 лет и старше	X	X			X				X					X	X	X							X		
28	Robot Magic	нет данных	нет данных	7 - 16 лет	X	X			X				X					X	X	X	X	X					X	X	
29	Light Bot	Частное лицо (канадский студент)	2008	4 года и старше										X				X	X	X					X				
30	Cargo-Bot	Австралийская компания и выпускник Массачусетского технологического института по специальности "Информатика"	2012	5 лет и старше										X				X											
31	Sprite Box	LightBot	2017	7 лет и старше										X					X										
32	CodeMonkey	Компания-разработчик технологий обучения	2014	8 - 14 лет	X				X										X								X		
33	RoboMind	Нидерландская компания	2005	9 - 21 лет	X				X										X	X					X		X		
34	Bits Box	Два бывших сотрудника компании Google	2014	6 - 14 лет	X						X								X	X	X						X		X
35	Mozilla Thimble	Фонд Mozilla Foundation	2012	12 лет и старше	X										X			X		X		X			X				X
36	Made with Code	Google	2014	13 - 18 лет		X								X				X		X						X			
37	Storytelling Alice	Исследователь Университета Карнеги-Меллона	2007	13 - 18 лет		X		X										X		X						X		X	
38	JewelBots	Американский стартап Jewelbots	2016	9 - 12 лет	X											X			X				X			X	X		
39	Mover Kit	Английская компания Technology Will Save Us	2016	8 лет и старше		X										X			X		X	X					X		
40	Boolean Box	Американская НКО Boolean Girl	2016	8 - 15 лет	X	X							X						X							X	X		
41	SmartGurlz	Датский стартап	2016	6 - 12 лет		X							X	X					X							X			
42	GoldieBlox	Американская компания	2012	4 года и старше							X		X	X			X		X							X			
43	Lego Mindstorms	Группа Lego	2013	10 лет и старше	X	X							X						X						X		X		
44	Ozobot	Компания-разработчик технологий обучения	2012	7 - 16 лет		X							X						X								X		
45	Wonder Workshop	Компания-разработчик технологий обучения	2013	6 лет и старше							X		X						X								X		

		Общая информация			1. Тип языка программирования			2. Создание контента (назначение)										3. Доступ			4. Поддержка сообществ				5. Особые соображения				
#	Инструмент	Разработчик	Дата выпуска	Возрастная группа	Текстовое программирование	Визуальное программирование	Осязаемое программирование	Сочинение	Изучение кода, играя в игры	Разработка игр	Разработка приложений	Произведения искусства	Программирование учебных роботов	Развитие основных навыков вычислит. Мышления	Изучение традиц. языков кодирования	Программирование носимых технологий	Без экрана	Бесплатно / открытый код	Платно	Доступ с основных аппаратных / операционных систем	Размещение проектов	Создание ремиксов	Традиц. текстовые форумы как онлайн-сообщества	Интерактивное онлайн-сообщество	Доступность на русском	Гендерный аспект	Поддержка учителей / родителей	3D-элементы	Поддержка генерации идей
46	Micro:Bit	BBC	2016	14 - 16 лет	X	X							X						X							X	X			
47	Thymio	Федеральная политехническая школа Лозанны, швейцарский университет	2012	6 - 16 лет	X	X							X						X				X				X			
48	Puzzlets	Американский стартап	2014	6 - 10 лет			X						X						X								X			
49	Osmo Coding Jam	Калифорнийский стартап Tangible Play Inc.	2017	6 - 12 лет			X						X						X								X			X
50	Cubetto (by Primo)	Primo Toys (английская компания)	нет данных	3 - 6 лет			X						X				X		X											
51	AlgoBrix	Образовательная компания Young Engineers	Еще не выпущен	5 - 13 лет			X						X				X		X											
52	Kibo	KinderLab Robotics, американская компания	2014	4 - 7 лет			X						X				X		X								X			
53	Code & Go™ Robot Mouse Activity Set	Learning Resources	2015	4 года и старше			X						X				X		X											
54	Code-A-Pillar	Fisher Price, американская компания	2016	3 - 6 лет			X						X				X		X											
55	WowWee Elmoji	Wowwee - компания в Гонконге	2017	5 - 8 лет										X					X											
56	Lego Boost	Группа Lego	2017	7 to 12		X							X						X								X			
57	Sphero SPRK Edition	Sphero, американская компания	2015	8 лет и старше		X							X						X											
58	Kano	Kano, английская компания	2013	6 лет и старше	X	X							X					X	X			X		X			X			X
59	LittleBits Code Kit	LittleBits, американская компания	2017	8 - 17 лет		X							X						X								X			X
60	Hasbro's Proto Max	Hasbro, мультинациональная компания по производству игрушек	2017	6 лет и старше		X							X	X					X											
61	Cozmo	Anki, американская компания	2017	8 лет и старше		X							X						X											
62	Meccano M.A.X.	Spin Master, канадская компания по производству игрушек	2017	10 лет и старше		X							X						X											
63	Augie	Pai Technology, китайская компания	2017	7 - 12 лет		X			X				X						X											
64	Hello Ruby	Финская компания	2014	4 - 7 лет										X			X	X	X								X			
65	CS Unplugged	Университет Кентербери, Новая Зеландия	нет данных	5 - 14 лет										X			X	X									X			
66	Robot Turtles	ThinkFun, американская компания	2013	3 - 8 лет										X			X		X											
67	Code Monkey Island	Разработчик из Нью-Йорка	2014	8 лет и старше										X			X		X											
68	Koov	Sony	мар.18	7 лет и старше		X							X						X	X				X			X			X
69	Twine	Веб-разработчик	2009	12 - 18 лет	X							X						X		X			X							
70	My Kingdom Books	Английское издательство	2016	6 - 11 лет										X					X											
71	Yibu	Frog, всемирная компания-раработчик	нет данных	нет данных					X				X						X											

	Общая информация				1. Тип языка программирования			2. Создание контента (назначение)													3. Доступ			4. Поддержка сообществ				5. Особые соображения				
#	Инструмент	Разработчик	Дата выпуска	Возрастная группа	Текстовое программирование	Визуальное программирование	Осязаемое программирование	Сочинение	Изучение кода, играя в игры	Разработка игр	Разработка приложений	Произведения искусства	Программирование учебных роботов	Развитие основных навыков вычислит. Мышления	Изучение традиц. языков кодирования	Программирование носимых технологий	Без экрана	Бесплатно / открытый код	Платно	Доступ с основных аппаратных / операционных систем	Размещение проектов	Создание ремиксов	Традиц. текстовые форумы как онлайн-сообщества	Интерактивное онлайн-сообщество	Доступность на русском	Гендерный аспект	Поддержка учителей / родителей	3D-элементы	Поддержка генерации идей			
ИНСТРУМЕНТЫ, ИЗНАЧАЛЬНО РАЗРАБОТАННЫЕ НА РУССКОМ																															
1	Logiclike	Два IT-специалиста из Беларуси	нет данных	5 - 12 лет										X					X	X					X		X					
2	YoungCoder	Аспирант из России	2016	14 лет и старше	X										X			X		X			X		X							
3	CheckiO	Российско-украинская команда	2013	14 лет и старше	X				X									X		X					X							
4	КуМир	Научно-исследовательский институт системных исследований	2018	12 - 17 лет	X										X			X		X					X		X					
5	LinguaCode	Армянский стартап	2015	7 - 17 лет	X										X			X		X	X			X	X							
6	GeekBrains	Российская компания	нет данных	12 лет и старше	X										X			X	X				X		X							
7	ПиктоМир	Научно-исследовательский институт системных исследований	2014	5 - 7 лет										X				X		X					X		X					
8	ПервоЛого 4.0	Разработка Института новых технологий	2013	4 - 10 лет		X								X					X						X							
9	ЛогоМиры	Институт новых технологий	1997	12 - 17 лет	X	X		X											X		X				X							
10	RooboJunior	Павел Фролов, ROBBO	нет данных	5 лет и старше		X		X	X									X							X							
11	Алгоритмика	Algoritmika, российский стартап	2016	7 - 12 лет		X		X		X									X						X		X					
12	Кодвардс	Redmadrobot	2017	7 - 12 лет	X				X									X	X						X		X					
13	Роббо	ROBBO	нет данных	7 - 15 лет		X							X					X	X						X		X					
14	Амперка	Российская компания	нет данных	11 - 15 лет	X	X							X						X						X		X					
15	Роботрек	Brain Development Ltd.	2012	5 - 16 лет		X							X						X						X		X					
16	Robotologica	Robotologia, российская компания	нет данных	9 лет и старше	X	X							X						X						X		X					
17	Lart	Lart, российская компания	нет данных	14 лет и старше	X								X						X						X							
18	Code Adventures	Cyborc Games в Болгарии	2017	6 лет и старше										X					X						X							
19	Битва големов	М. Казанцев and А. Казанцев	2015	6 лет и старше										X			X		X						X							
20	Прогеры	С. Пахоменко	2017	6 лет и старше										X			X		X						X							

ГРУППА ВСЕМИРНОГО БАНКА
Образование