Л. Л. Босова,
Московский педагогический государственный университет

КАК УЧАТ ПРОГРАММИРОВАНИЮ В XXI ВЕКЕ: ОТЕЧЕСТВЕННЫЙ И ЗАРУБЕЖНЫЙ ОПЫТ ОБУЧЕНИЯ ПРОГРАММИРОВАНИЮ В ШКОЛЕ

Аннотация

В статье рассмотрены подходы к обучению программированию в российской школе, сложившиеся за годы существования школьного курса информатики, обозначены имеющиеся в настоящее время достоинства и недостатки в обучении школьников программированию. Проанализировано современное состояние обучения программированию школьников в США, странах Европы и Азии. Выявлены основные международные тенденции в обучении школьников программированию: раннее начало, обязательность и непрерывность обучения программированию, разнообразие современных сред для программирования, использование облачных сервисов, проектный подход и командная форма работы в методике обучения программированию.

Ключевые слова: обучение школьников программированию, тенденции в обучении программированию, алгоритмическое мышление, раннее обучение программированию, информатика в школе.

Контактная информация

Босова Людмила Леонидовна, доктор пед. наук, доцент, зав. кафедрой теории и методики обучения информатике Московского педагогического государственного университета; *адрес:* 119991, г. Москва, ул. Малая Пироговская, д. 1, стр. 1; *телефон:* (495) 438-79-77; *e-mail:* akulll@mail.ru

L. L. Bosova,
Moscow Pedagogical State University

HOW TO TEACH PROGRAMMING IN THE 21ST CENTURY: DOMESTIC AND FOREIGN EXPERIENCE IN TEACHING PROGRAMMING AT SCHOOL

Abstract

The article considers approaches to teaching programming in the Russian school, developed over the years of the school informatics course; the current advantages and disadvantages of teaching programming are indicated. The modern state of teaching programming for schoolchildren in the United States, Europe and Asia is analyzed. The main international trends in teaching programming are revealed: early start, compulsion and continuity of programming training, a variety of modern programming environments, the use of cloud services, project approach and team form of work in the methodology of teaching programming.

Keywords: teaching programming, trends in teaching programming, algorithmic thinking, early learning programming, informatics at school.

«Программирование — вторая грамотность!» Эта метафора, сформулированная и произнесенная академиком А. П. Ершовым в далеком 1981 году, стала девизом появившегося в скором времени во всех школах нашей страны нового предмета «Основы информатики и вычислительной техники» (ОИВТ), предыстория которого восходит к 60-м годам прошлого века: именно тогда начались попытки обучения школьников программированию и научно-педагогического осмысления целей и методов такого обучения. Конкретное содержание своей метафоры А. П. Ершов раскрыл в речи на 3-й Всемирной конференции ИФИП (Международной федерации по обработке информации) и ЮНЕСКО по применению ЭВМ в обучении, состоявшейся 27–31 июля 1981 года в Лозанне (Швейцария). Приведем несколько цитат из этого выступления, опубликованного в работе [3].

Прежде всего, А. П. Ершов обращал внимание на то, что и «грамотность, как способность человека воспринять и выразить знание в текстовой форме», и программирование, как механизм перехода от знания к действию, подход «к формированию исполнительных механизмов человека», «являются выражением органической способности человека, т. е. способности, подготовленной организацией его нервной системы и присущей человеку во всех его социальных функциях», и если «грамотность — это не только умение читать, но и воспитание интеллигентного человека», то «вторая грамотность — это не только умение писать команды для машин, но и воспитание человека, решительного и предусмотрительного вместе». Развивая идею «фундаментализации программирования», А. П. Ершов акцентировал внимание на тесной связи «операционного знания и алгоритмического мышления с другими компонентами образования», на том, что «законы программирования смыкаются с математическим образованием, образуя единый, но еще не построенный фундамент воспитания операционного и комбинаторного мышления, способности к абстракции, рассуждению и действию». Подчеркивая «необходимость актуализировать в виде программ информационную модель мира», А. П. Ершов обращал внимание на то, что «мы живем в мире программ, и сами постоянно программируем, не сознавая этого»; по сути, мир программ — это «огромный запас операционного знания, накопленный человечеством и теперь лишь актуализируемый вычислительными машинами, роботами, автоматическими устройствами», а активная жизненная позиция — не что иное, как «способность выработать программу действия и следовать ей». Вместе с тем «способность человека к передаче знаний машине безнадежно отстает от способности создать эту машину», что делает формирование навыков программирования необходимым

компонентом содержания общего образования. При этом речь идет не о том, чтобы «навязать детям новые, не свойственные им навыки и знания», а о том, чтобы «учить детей способности планировать свои действия и их последствия», чтобы «проявить и сформулировать те стороны мышления и поведения, которые реально существуют, но формируются стихийно, неосознанно».

На всех этапах существования отечественного курса школьной информатики линия «Алгоритмизация и программирование» неизменно является одной из его основных содержательных линий; методическая сторона этого вопроса детально разработана в исследованиях М. П. Лапчика, А. Г. Гейна, И. Г. Семакина и др. отечественных педагогов.

Российская школа имеет в своем арсенале богатый опыт **обучения младших школьников азам алгоритмизации и программирования с использованием интерактивных сред с виртуальными управляемыми объектами** (учебно-методический комплекс «Роботландия» с его алгоритмическими этюдами; виртуальная лаборатория «Алгоритмика» с исполнителями Кузнечик, Директор строительства, Водолей, Ханойская башня, Чертежник, Черепаха, Робот; учебно-методические комплексы «ПервоЛого» и «ЛогоМиры», система программирования «КуМир» с исполнителями Чертежник, Робот и др.). На важность наличия таких сред указывал академик А. П. Ершов, отмечавший следующее: «Вопрос о том, как учить детей способности планировать свои действия и их последствия, какая операционная обстановка при этом нужна, очень далек от тех методических альтернатив, которые мы обсуждаем, например, при профессиональном обучении программированию. С одной стороны, мы должны сделать эту обстановку естественной для ребенка, с другой стороны, она должна быть достаточно богатой для того, чтобы он мог, как говорят психологи, сам создавать теорию познаваемого явления» [3].

В последние годы в работе с младшими школьниками все более широкое распространение получают современные интерактивные среды для раннего обучения алгоритмизации и программированию: Scratch, Blockly и др. Эти визуальные среды программирования позволяют учащимся избегать синтаксических ошибок (ошибок при вводе команд), которые неизбежно возникают при работе с традиционными текстовыми языками. Данные среды могут использоваться школьниками достаточно долго, так как с их помощью можно создавать достаточно сложные программы, игры, приложения и анимации. Методика применения интерактивных сред для обучения младших школьников программированию, а также основные характеристики учебного модуля «Пропедевтика программирования со Scratch» представлены в работе [2].

Тревогу вызывает лишь то обстоятельство, что курс информатики, включающий в себя алгоритмизацию и программирование, не является обязательным для изучения **в начальной школе**. Что же касается такого компонента ИКТ-компетентности выпускника начальной школы, как «планирование деятельности, управление и организация», с достаточно емкими планируемыми результатами (создавать движущиеся модели и управлять ими в компьютерно управляемых средах (создание простейших роботов); определять последовательность выполнения действий, составлять инструкции (простые алгоритмы) в несколько действий, строить программы для компьютерного исполнителя с использованием конструкций последовательного выполнения и повторения; планировать несложные исследования объектов и процессов внешнего мира) [7], то его формирование в процессе изучения разных предметов скорее декларируется, чем реализуется на практике.

В старшей школе на базовом уровне изучения информатики, ориентированном на общую функциональную грамотность, получение компетентностей для повседневной жизни и общего развития, учащиеся должны научиться «читать и понимать несложные программы, написанные на выбранном для изучения универсальном алгоритмическом языке высокого уровня; выполнять пошагово (с использованием компьютера или вручную) несложные алгоритмы управления исполнителями и анализа числовых и текстовых данных; создавать на алгоритмическом языке программы для решения типовых задач базового уровня из различных предметных областей с использованием основных алгоритмических конструкций». **Углубленный уровень** изучения информатики, ориентированный на получение компетентностей для последующей профессиональной деятельности как в рамках данной предметной области, так и в смежных с ней областях, должен обеспечивать умения «создавать собственные алгоритмы для решения прикладных задач на основе изученных алгоритмов и методов; <...> выполнять отладку и тестирование программ в выбранной среде программирования; использовать при разработке программ стандартные библиотеки языка программирования и внешние библиотеки программ; создавать многокомпонентные программные продукты в среде программирования» [9].

Согласно действующим в настоящее время федеральным государственным образовательным стандартам, изучение информатики является обязательным только **в основной школе** — по одному часу в неделю в VII—IX классах, всего 102 часа. Раздел «Алгоритмы и элементы программирования» — важнейший раздел действующей в настоящее время примерной программы по информатике для основной школы [8], при его изучении рассматриваются следующие темы: «Исполнители и алгоритмы. Управление исполнителями», «Алгоритмические конструкции», «Построение алгоритмов и программ», «Анализ алгоритмов», «Робототехника», «Математическое моделирование».

Курс информатики основной школы ориентирован на то, чтобы научить школьников:

- составлять алгоритмы для решения учебных задач различных типов;
- выражать алгоритм решения задачи различными способами (словесным, графическим, в том числе и в виде блок-схемы, с помощью формальных языков и др.);
- определять оптимальный способ выражения алгоритма для решения конкретных задач (словесный, графический, с помощью формальных языков);
- определять результат выполнения заданного алгоритма или его фрагмента;
- использовать термины «исполнитель», «алгоритм», «программа», а также понимать разницу между употреблением этих терминов в обыденной речи и в информатике;
- выполнять без использования компьютера («вручную») несложные алгоритмы управления исполнителями и анализа числовых и текстовых

данных, записанные на конкретном языке программирования с использованием основных управляющих конструкций последовательного программирования (линейная программа, ветвление, повторение, вспомогательные алгоритмы);
- составлять несложные алгоритмы управления исполнителями и анализа числовых и текстовых данных с использованием основных управляющих конструкций последовательного программирования и записывать их в виде программ на выбранном языке программирования;
- выполнять эти программы на компьютере;
- использовать величины (переменные) различных типов, табличные величины (массивы), а также выражения, составленные из этих величин;
- использовать оператор присваивания;
- анализировать предложенный алгоритм, например, определять, какие результаты возможны при заданном множестве исходных значений;
- использовать логические значения, операции и выражения с ними;
- записывать на выбранном языке программирования арифметические и логические выражения и вычислять их значения.

В полном соответствии с целями и задачами общего образования **рассмотрение вышеперечисленных вопросов подается в основной школе с метапредметной точки зрения**, демонстрируя обучающимся методологию решения широкого спектра жизненных задач, в том числе связанных с их учебно-познавательной деятельностью, предполагающей наличие умений:
- самостоятельно определять цели своего обучения;
- самостоятельно планировать пути достижения целей, в том числе альтернативные, осознанно выбирать наиболее эффективные способы решения учебных и познавательных задач;
- соотносить свои действия с планируемыми результатами, осуществлять контроль своей деятельности в процессе достижения результата, определять способы действий в рамках предложенных условий и требований, корректировать свои действия в соответствии с изменяющейся ситуацией;
- оценивать правильность выполнения учебной задачи, собственные возможности ее решения;
- самоконтроля, самооценки, принятия решений и осуществления осознанного выбора в учебной и познавательной деятельности.

В таблице представлена **информация по количеству учебного времени, запланированного в разных УМК на изучение раздела «Алгоритмы и элементы программирования»**.

«Алгоритмы и элементы программирования» — важный, но не единственный раздел чрезвычайно насыщенного содержания отечественного курса школьной информатики. В разных УМК на его изучение отводится от 15 до 27,5 % учебного времени. Соотнесение представленных выше планируемых результатов изучения раздела и реального учебного времени, которое предусмотрено на его освоение, позволяет усомниться в реалистичности намеченных планов.

Жесткие временны́е рамки тесно связаны с еще одной проблемой реализации раздела «Алгоритмы и элементы программирования» в основной школе. «В школьном курсе информатики существует разрыв между изучением основ программирования, основных методов и конструкций, используемых на практике, и построением законченных приложений <...> изучение программирования остается на некотором начальном, логически незавершенном уровне: оно не ведет к исследованию научных задач или решению проблем, встречающихся на практике» [6]. Действительно, традиционные подходы к формированию фундаментальных навыков программирования предполагают приоритетное рассмотрение базовых алгоритмических конструкций, разработку на этой основе простейших программ, решающих элементарные задачи. После этого было бы логично знакомить учащихся с имеющимися стандартными библиотеками языка программирования, предлагать им более сложные задачи, решение которых может быть сконструировано из имеющихся в библиотеке функций. Во многом именно дефицит учебного времени не позволяет реализовать эту идею.

Следующая проблема связана с **выбором языка для обучения школьников программированию**. Следует отметить, что и ранее, и в настоящее время нормативные документы, регламентирующие образование, никогда не определяли тот язык программирования, который должен в обязательном порядке изучаться в школе. В частности, задания ОГЭ по информатике содержат примеры программ на таких языках программирования, как Basic, Pascal, школьный алгоритмический язык (КуМир), C++ и Python. Учебники информатики для основной школы ориентированы преимущественно на Pascal и школьный алгоритмический язык, что связано с традициями и развитым учебно-методическим обеспечением для данных языков.

Что касается современных школьников, особенно тех, кто интересуется программированием, то их такое положение дел не вполне устраивает. Школьники

Таблица

Алгоритмы и элементы программирования в УМК по информатике для основной школы

№ п/п	Авторы и название УМК	Темы (по классам)	Кол-во часов
1	Л. Л. Босова, А. Ю. Босова. УМК по информатике для VII—IX классов	VIII класс. Основы алгоритмизации VIII класс. Начала программирования IX класс. Алгоритмизация и программирование	10 10 8
2	Н. Д. Угринович. УМК по информатике для VII—IX классов	IX класс. Основы алгоритмизации и объектно-ориентированного программирования	15
3	И. Г. Семакин и др. УМК по информатике для VII—IX классов	IX класс. Управление и алгоритмы IX класс. Введение в программирование	12 15

заинтересованы в изучении на уроках информатики языка, являющегося востребованным в промышленном программировании и подходящего для разработки современных приложений. Этим требованиям в полной мере отвечает язык программирования Python, интерес к которому растет не только в нашей стране, но и за рубежом. Педагоги, использующие Python, отмечают следующие его достоинства [5]:

- не нужны инструкции, не имеющие непосредственного отношения к алгоритму;
- как правило, программа на языке Python записывается короче, чем на C++, Pascal, Basic, но при этом без ущерба для понятности кода;
- свободная и кроссплатформенная реализация;
- интерпретируемая архитектура языка упрощает написание, запуск и отладку программ;
- современность языка, наличие в нем высокоуровневых структур данных (списки, множества, ассоциативные массивы, длинная арифметика);
- наличие средств объектно-ориентированного и функционального программирования;
- наличие богатой библиотеки, позволяющей легко разрабатывать графические приложения, веб-приложения и т. д.

Все вышеперечисленные качества делают возможным использование языка Python как для углубленного изучения информатики в старшей школе, так и при освоении программирования в курсе информатики основной школы.

Современный урок программирования допускает и параллельную работу учащихся с различными языками, ставшую возможной благодаря наличию облачных сервисов (например, http://rextester.com), позволяющих:

- изучать программирование без установки программного обеспечения на свой компьютер;
- изучать программирование с использованием мобильного телефона.

В целом важно не то, какой именно язык программирования используется на уроках информатики конкретным учителем в конкретной школе, а то, как именно происходит обучение программированию, в какой степени достигаются заявленные во ФГОС образовательные результаты.

Важным направлением возможного развития раздела «Алгоритмы и элементы программирования» является включение в него **элементов робототехники**. Идея использования учебных конструкторов с возможностью создания управляющих программ не нова для российской школы — можно вспомнить концепцию «LEGO+Logo», имеющую достаточно продолжительное и широкое применение в начальных школах. Что касается основной школы, то там до сих пор основной упор делался на работу с виртуальными (экранными) управляемыми объектами. В настоящее время, в условиях развития современного высокотехнологичного общества, все более актуальным становится вопрос о подкреплении визуализации «виртуальной» визуализацией «реальной», основанной на работе с роботами, действующими в реальном физическом мире. Это позволяет в рамках школьной программы совершить переход от создания алгоритмов детерминированных процессов с заранее заданным конечным результатом (построение изображения, выход из лабиринта) к разработке алгоритмов взаимодействия с окружающей средой, управлению с обратной связью. При этом важно понимать, что курс робототехники не может быть альтернативой курсу информатики, что робототехника имеет непосредственное отношение к информатике именно в части программирования робота.

Серьезное внимание обучению школьников программированию уделяется в последние годы и в других странах. Рассмотрим некоторые примеры.

Англия, Уэльс, Северная Ирландия.

С 2014 года развитие вычислительного (компьютерного) мышления и первых навыков программирования уже в начальной школе стало одним из основных направлений развития британской школы [10]. При этом сущность вычислительного мышления определяется через умения разбивать сложные задачи на мелкие подзадачи (декомпозиция), сравнивать с задачами, решенными ранее (распознавание паттернов), отбрасывать несущественные детали (абстрагирование), определять и прорабатывать шаги для достижения результата (алгоритмизация), совершенствовать вышеперечисленные этапы (отладка).

С учащимися I—II классов (5–7 лет) обсуждаются вопросы необходимости использования алгоритмов в жизни, рассматриваются способы записи алгоритмов на псевдокоде и с помощью блок-схем.

После этого (III—VI классы, 7–11 лет) ученики приступают к знакомству с базовыми алгоритмическими структурами, рассматривают простые алгоритмы, содержащие следование, ветвление и циклы, работают с переменными и различными формами ввода и вывода.

Освоив основы алгоритмов, учащиеся (VII—IX классы, 11–14 лет) начинают знакомство с программированием. Для этого рекомендуется использовать языки Python, Java, C++, Basic, Scratch. Ученики рассматривают способы создания программы из алгоритма, учатся записывать циклы и ветвления на выбранном языке программирования, знакомятся с записью и применением логических операций. Далее следует знакомство с такими структурами данных, как массивы и списки; разъясняются вопросы, касающиеся разработки модульных программ, которые используют процедуры и функции.

В X—XII классах учащиеся получат возможность развивать и применять свои навыки в области программирования.

Франция.

До 2012 года информатика во Франции ограничивалась обучением использованию основных программных продуктов. В 2012 году во Франции учебная программа по информатике была введена по выбору для IX—XII классов. В 2016 году впервые во Франции ввели обязательное преподавание информатики в начальной и средней школе [14]; при этом информатика не стала отдельным предметом, а была интегрирована сразу в два предмета — математику и технологию. Ответственность за обучение программированию возложена на учителей математики, в помощь которым разработаны сайты, содержащие методические разработки компьютерных и бескомпьютерных занятий. Учащиеся знакомятся с программированием, разрабатывая несколько простых программ в рамках проектного подхода, не ставя своей целью исчерпывающее знание определенного языка программирования или программного обеспечения. В проектах учащиеся используют соответствующие цифровые инструменты (организуют, ищут, проектируют,

производят, планируют, моделируют) и разрабатывают всю программу или ее часть, компилируют и запускают ее. Среди предлагаемых школьникам задач — разработка, настройка и программирование компьютерных приложений для мобильных устройств; изменение существующей программы с целью улучшения ее характеристик. Создавая программу, учащиеся осваивают методы программирования, имеют возможность познакомиться с понятиями переменной и функции, посмотреть на них с точки зрения, отличной от математической.

Ирландия.

В 2016 году Департаментом образования Ирландии были разработаны программы обязательных для изучения дисциплин «Краткий курс кодирования» для младшего цикла (12–15 лет) и «Курс кодирования» для старшего цикла (16–18 лет) средней школы [15]. Особенностью программ является то, что акцент в них сделан не только на предметных результатах, но и на формировании важнейших ключевых навыков (Key skills): грамотность, креативность, коммуникабельность, способность управлять собой, работать с другими и т. д. Для этого в процессе обучения поощряется командная работа: ученики сотрудничают, дают друг другу разъяснения, ищут информацию, обеспечивают обратную связь и размышляют над своей работой. Теоретические вопросы, рассматриваемые в рамках курса, подкрепляются решением практических задач и выполнением проектов. При организации практических работ предпочтение отдается бесплатному программному обеспечению с открытым исходным кодом, что позволяет сделать программные инструменты доступными для обучающихся и предоставляет им возможность изучения исходного кода используемых инструментов.

Предполагается, что в результате изучения дисциплины «Краткий курс кодирования» учащиеся будут:

- разрабатывать алгоритм и записывать код коротких программ с использованием операторов присваивания, арифметических операций, логических операций и операций сравнения;
- решать короткие задачи программирования с использованием основных линейных структур данных (одномерные массивы, списки);
- использовать функции и/или процедуры (определение и вызов);
- документировать программы и представлять документированный код друг другу в небольших группах;
- анализировать программный код для определения его назначения и выявления явных или потенциальных ошибок и т. д.

Финляндия.

В соответствии с национальной учебной программой в 2016/2017 учебном году программирование стало обязательной сквозной темой финского школьного курса математики начиная с первого класса [13]. Программирование рассматривается как инструмент формирования вычислительного (компьютерного) мышления, как средство для стимулирования творчества, повышения мотивации в целом и интереса к STEM, а также для развития логического мышления. Фактически навыки программирования позиционируются в качестве нового навыка, приобретаемого в ходе обучения, в дополнение к чтению, письму, рисованию и вычислительным навыкам.

Знакомство с программированием в I—II классах начинается с рассмотрения пошаговых инструкций — однозначных приказов, которые ученики научатся отдавать друг другу. Важно, чтобы ученики поняли, что только точные инструкции приводят к точным действиям.

В III—VI классах начинается разработка программ в среде языка графического программирования. Используемый инструмент преднамеренно еще не является реальным языком программирования — в выбранной среде визуального программирования (например, в Scratch) ученики перетаскивают готовые блоки, а не записывают текстовые программы.

В VII—IX классах начинается изучение реального языка программирования. При этом нет указания на то, какой именно язык программирования должен изучаться; главное, чтобы ученики могли понимать написанный программный код. Рекомендуется для решения задач из курса математики самостоятельно разрабатывать компьютерные программы или использовать готовые.

Китай.

С 2012 года в Китае реализуется непрерывный курс информационных технологий, охватывающий всех учащихся на всех ступенях школьного образования. Линия алгоритмизации и программирования является, по сути, сквозной в школьном курсе информационных технологий Китая, где она представлена следующими модулями [1]:

- начальная школа — дополнительный модуль «Знакомство с разработкой алгоритмов и программированием» (36 ч);
- средняя школа — дополнительный модуль «Разработка алгоритмов и программирование» (36 ч);
- старшая школа — вариативный модуль «Разработка алгоритмов и программирование» (36 ч).

Модуль «Знакомство с разработкой алгоритмов и программированием» ориентирован на учеников V—VI классов. Его основная идея состоит в признании приоритетности развития алгоритмического мышления школьников; программирование не сводится к тренировке в написании кода — гораздо важнее понимание его сущности как особого метода обработки информации. В рамках рассматриваемого модуля школьники:

- учатся переходить от описания жизненных ситуаций на естественном языке к их представлению с помощью блок-схем;
- рассматривают линейные, разветвляющиеся и циклические алгоритмы из повседневной жизни;
- знакомятся со средой визуального программирования;
- разрабатывают простые алгоритмы, содержащие базовые алгоритмические конструкции, пишут соответствующие им программы в среде визуального программирования.

Содержание модуля «Разработка алгоритмов и программирование», ориентированного на учеников VIII—IX классов, включает сведения об истории возникновения, классификации и тенденциях развития языков программирования, достаточно полное знакомство с объектно-ориентированным языком программирования.

В начальной и средней школе опыт программирования приобретается учащимися также в рамках дополнительных модулей «Знакомство с роботом» (36 ч) и «Проектирование и создание роботов» (36 ч).

В модуле «Разработка алгоритмов и программирование» для старшей школы четко прослеживается направленность курса на решение практических задач, содержание которых так или иначе связано с реальной жизнью. Сами задачи имеют умеренный уровень сложности, хотя их решение предполагает знание и использование некоторых методов и алгоритмов (аналитический метод, метод полного перебора, рекурсивный метод, алгоритмы последовательного и бинарного поиска, алгоритмы сортировки и т. д.).

Южная Корея.

Это одна из самых развитых по степени информатизации и компьютеризации стран в мире, имеющая многолетние традиции преподавания информатики в школе.

В курсе информатики для начальной школы (I—VI классы) изучаются тематические разделы: «Информационное общество», «Прикладное программное обеспечение», «Представление информации в компьютере», «Алгоритмы» [4]. Изучение программирования начинается в среде исполнителя, для этого используют современную среду визуального программирования Scratch.

В основной школе (VII—IX классы) информатика изучается в рамках предмета «Технология и домашняя экономика». С точки зрения продолжения линии программирования интерес представляет включение в программу таких тем, как «Программирование контроллера», «Разработка приложений в App Inventor», «Программирование приложений для мобильных устройств».

Япония.

Япония — один из мировых лидеров в области высоких технологий. Тем не менее в настоящее время в начальной школе Японии изучение информатики не предусмотрено; в основной школе ученики осваивают компьютерную грамотность и базовое программирование роботов в рамках предмета «Технология и экономика»; программирование появляется в средней школе, причем только в рамках профиля обучения «Наука». В целях обеспечения промышленной конкурентоспособности Япония планирует сделать обязательным образование в области программирования для начальной, основной и средней школы с 2020, 2021 и 2020 годов соответственно [12].

Младшие школьники будут осваивать основы программирования на уроках математики, разрабатывая программы для управления персонажами компьютерных игр. Считается, что такой подход будет способствовать развитию логического мышления и творческих способностей детей.

В основной школе развитие навыков программирования планируется продолжать в рамках курса «Информационные технологии».

В средней школе предполагается введение базового (Информатика I) и углубленного (Информатика II) уровней изучения предмета, инвариантной частью которого будет модуль «Компьютер и программирование», предполагающий изучение таких вопросов, как «Представление данных внутри компьютера», «Основы программирования», «Моделирование».

США.

Информатика не включена в учебный план американских школ в качестве обязательного предмета. В стране нет единого стандарта в отношении изучения информатики, равно как нет его и в большинстве штатов; есть прекрасные примеры подготовки школьников по информатике, но они не отражают общего состояния дел. Тем не менее в последние годы констатируются значимые успехи в изучении информатики, достигнутые младшими школьниками. Это связано, в первую очередь, с использованием специально разработанных для обучения визуальных языков программирования, в частности Scratch. Есть общенациональная программа и профессиональные усилия по оказанию помощи учителям начальной школы в преподавании программирования (например, Code.org, 2016).

Представление о том, как должно в идеале выглядеть изучение информатики в школе, изложено в материалах Американской ассоциации учителей информатики (Computer Science Teachers Association — CSTA), опубликовавшей в 2016 году рассчитанную на 12 лет обучения рамочную образовательную программу по информатике [11].

Программа построена вокруг пяти основных содержательных линий:

1) вычислительные системы;
2) сети и интернет;
3) данные и анализ;
4) алгоритмы и программирование;
5) влияние информационных технологий.

Для каждой из содержательных линий сформулированы своего рода требования к знаниям и умениям учеников к концу II, V, VIII и XII классов.

Авторы рамочной программы особо подчеркивают роль линии «Алгоритмы и программирование». Приведем несколько соответствующих цитат:

С раннего возраста детей учат читать и писать, чтобы они могли воспринимать чужие и выражать собственные мысли.

Большинство школьников учат использовать («читать»), а не создавать («писать») компьютерные продукты.

Вместо того чтобы быть пассивными потребителями компьютерных технологий, школьники могут стать их активными производителями и создателями.

В наш цифровой век вы можете «программировать или быть запрограммированными».

Умения планирования деятельности особенно важны в наше время, когда все больше рутинных операций можно поручить роботизированным комплексам.

Понимание того, как переложить на компьютер работы, с которыми до этого справлялись только люди; понимание того, с какими трудностями при этом предстоит столкнуться.

Основное содержание линии «Алгоритмы и программирование» структурировано по следующим блокам:

- алгоритмы;
- переменные;
- управление;
- модульность;
- разработка программ.

Проследим, как разворачивается содержание в каждом из этих блоков к концу II, V, VIII и XII классов.

Алгоритмы.

Конец II класса. Учащиеся:

- приводят примеры алгоритмов, которым они следуют в повседневной жизни;

- знают, что люди могут разрабатывать и записывать с помощью специальных языков алгоритмы, которые будут понимать и исполнять компьютеры.

Конец V класса. Учащиеся:
- понимают, что одного и того же результата можно достичь с помощью разных алгоритмов, при этом одни алгоритмы будут более подходящими для конкретной ситуации, чем другие;
- знают, что алгоритмы могут быть записаны на языках, не связанных с компьютером, в том числе с помощью естественного языка, блок-схемы или псевдокода.

Конец VIII класса. Учащиеся:
- понимают, что алгоритмы определяют характер взаимодействия человека и компьютера; что целесообразно разрабатывать алгоритмы, применимые ко многим ситуациям, обеспечивающие различные выходные сигналы для широкого диапазона входных сигналов; приводят примеры таких алгоритмов;
- знают, что алгоритмы тестируют и отлаживают; тестирование алгоритма требует использования данных (входных сигналов), которые отражают все возможные условия для оценки их точности и надежности.

Конец XII класса. Учащиеся:
- понимают, что для решения конкретной задачи алгоритмы выбираются с учетом скорости их выполнения, количества памяти для хранения данных, простоты реализации на конкретном языке программирования, большей универсальности;
- знают типовые алгоритмы, используемые для поиска и сортировки данных в различных приложениях, для защиты данных, а также алгоритмы сжатия;
- проводят простейшую оценку производительности (количество операций, количество памяти) алгоритмов без привлечения сложного математического аппарата.

Переменные.

Конец II класса. Учащиеся:
- понимают, что данные из реального мира (числа, тексты, цвета, изображения) могут быть обработаны с помощью компьютерных программ; тип данных определяет действия, которые с ними могут быть совершены (например, числа можно складывать или вычитать, изображения перекрашивать или обрезать и т. д.).

Конец V класса. Учащиеся:
- понимают, что в языках программирования для хранения различных типов данных (чисел, текстов) используются переменные.

Конец VIII класса. Учащиеся:
- понимают, что переменная — это контейнер с именем; содержимое контейнера может меняться, а имя (идентификатор) — нет. Соглашение по именованию переменных и продуманный выбор идентификаторов улучшают читаемость программы. Программисты создают переменные для хранения значений данных определенных типов. Доступ и выполнение операций над значением переменной осуществляется по ее имени. Использование переменных позволяет применить один алгоритм для обработки различных наборов данных и получения различных выходных данных;
- понимают различие между переменной в программировании и переменной в математике. Переменная в программировании — это место (область памяти), в котором хранится значение, а также имя, используемое для доступа к значению; переменной присваивается значение, и это значение может изменяться во время выполнения программы. В математике не делается различия между переменной и именем переменной; под переменной в математике понимают каждый элемент некоторого множества, называемого областью изменения переменной.

Конец XII класса. Учащиеся:
- понимают, что для управления сложностью программ, создания высокопроизводительных программ, эффективно работающих с памятью, используются структуры данных;
- знают, что список — это структура данных, которая используется для обеспечения эффективного хранения, упорядочения и извлечения значений и различных других операций с его элементами. Знания сложных структур данных, таких как стеки, список ожиданий (очередь), древовидное представление и хэш-таблицы, не требуется. Рассмотрение определяемых пользователем типов и объектно-ориентированного программирования на этом этапе обучения не является обязательным.

Управление.

Конец II класса. Учащиеся:
- понимают, что компьютеры точно (буквально) следуют последовательности команд. Выполнение инструкций в программе не всегда бывает последовательным. Для инициирования тех или иных инструкций используются определенные события, например нажатие клавиши. Для повторяющихся инструкций могут быть использованы циклы; отличать различные типы циклов на этом этапе обучения не требуется.

Конец V класса. Учащиеся:
- понимают, что для определения последовательности исполнения инструкций используются управляющие конструкции, включая циклы, обработчики событий, и условные конструкции. Для того чтобы повторить инструкции несколькими способами в зависимости от ситуации, используются различные типы циклов. События — это щелчки мышью, ввод с клавиатуры, столкновение между объектами. Обработчики событий — это последовательности команд, которые связаны с конкретными событиями. Условные конструкции избирательно выполняют или пропускают инструкции при различных условиях. Условная конструкция состоит из логического условия, которое определяет действия в зависимости от того, является условие истинным или ложным. В логических условиях могут использоваться связки И, ИЛИ, НЕ.

Конец VIII класса. Учащиеся:
- понимают, что программисты выбирают и комбинируют управляющие структуры (циклы, обработчики событий и условные конструкции) при создании более сложных программ;

- знакомятся с составными условными операторами и вложенными условными структурами, приводят соответствующие примеры из повседневной жизни;
- понимают, что различные типы управляющих структур, такие как циклы и условные конструкции, могут быть объединены друг с другом.

Конец XII класса. Учащиеся:

- понимают, что при выборе и комбинации управляющих структур программисты должны идти на компромиссы, связанные с реализацией, читаемостью и производительностью программы. Реализация включает в себя выбор языка программирования, который влияет на время и усилия, необходимые для создания программы. Читаемость означает то, насколько программа будет понятна другим программистам. Оценка производительности на данном этапе обучения ограничивается теоретическим пониманием времени выполнения и требований к памяти. Управляющие структуры на этом уровне могут включать в себя условные операторы, циклы, обработчики событий и рекурсию. Рекурсия обязательной для изучения не является.

Модульность.

Конец II класса. Учащиеся:

- понимают, что сложная задача может быть разбита на более простые задачи, в свою очередь, некоторые из этих задач также могут быть разбиты на части. Ученики приводят примеры разделения (декомпозиции) сложной задачи на более простые задачи из повседневной жизни. Ученики понимают, что путем комбинации (объединения) известных решений небольших (простых) задач можно получить решение сложной задачи; приводят соответствующие примеры из повседневной жизни.

Конец V класса. Учащиеся:

- понимают, что разбиение программы на части позволяет облегчить ее разработку: можно сосредоточиться на тестировании той или иной части; разные люди одновременно могут работать над различными частями программы. В разрабатываемую программу также могут быть включены фрагменты ранее созданных программ.

Конец VIII класса. Учащиеся:

- понимают, что, для того чтобы скрыть детали программы, сделать код проще и обеспечить возможность его повторного использования, применяют процедуры. Процедура представляет собой модуль (группу инструкций в программе), который выполняет конкретную задачу. Процедуры вызываются, чтобы повторить группы инструкций. Процедуры, заданные с помощью параметров, применимы во многих ситуациях: они будут производить разные выходные данные, соответствующие широкому диапазону входных данных (аргументов).

Конец XII класса. Учащиеся:

- понимают, что современные программы — это системы взаимосвязанных модулей, разработка таких программ требует системного подхода. Например, в объектно-ориентированном программировании программа — это совокупность модулей, называемых объектами, соединяющих в себе данные с методами. Каждый модуль играет определенную роль, обеспечивающую достижение общей цели. Эти модули могут быть процедурами в рамках программы, комбинацией данных и процедур, быть независимыми, но взаимосвязанными программами. Модули позволяют лучше управлять сложными задачами.

Разработка программ.

Конец II класса. Учащиеся:

- понимают, что люди (программисты и пользователи) работают вместе, чтобы создавать (планировать, разрабатывать и тестировать) программы совместно и с определенной целью, например, для выражения идей или решения проблем. Программирование используется в качестве инструмента для создания продуктов, отражающих широкий круг интересов.

Конец V класса. Учащиеся:

- понимают, что создание программы — это итерационный процесс, включающий в себя этапы планирования, разработки и тестирования, каждый из которых может быть дополнительно разбит на подэтапы. Этап проектирования осуществляется до написания кода; это этап планирования, в рамках которого программисты собирают информацию о задаче и намечают решение. На этапе разработки планируемая структура проекта (замысел) выражается в языке программирования: создается код, который можно запустить на вычислительном устройстве. Разработка часто включает в себя повторное использование уже существующего кода или использование доработанных версий существующего кода, созданного другими. На этапе тестирования осуществляется проверка на предмет соблюдения требований программы, корректность и удобство использования. Результаты тестирования могут потребовать возвращения на этап разработки и, возможно, проектирования и внесения изменений; это демонстрирует итерационный характер процесса. Люди постоянно отслеживают, работают ли программы так, как и ожидалось; они исправляют или отлаживают части, которые этому не соответствуют. Повторение этих шагов позволяет людям улучшать и совершенствовать программы.

Конец VIII класса. Учащиеся:

- понимают, что люди (команды разработчиков) разрабатывают значимые для других (ориентированные на пользователя) решения путем определения критериев и ограничений задачи, тщательно рассматривая различные потребности и желания сообщества и проверяя, были ли соблюдены критерии и ограничения.

Конец XII класса. Учащиеся:

- понимают, что коллектив разработчиков, состоящий из людей разных типов, опираясь на сильные стороны своих членов, проявляемые в разных ролях, способен создавать востребованные программы. При создании программ неизбежны компромиссы. Так называемый «треугольник проекта» гласит: «Быстро / Качественно / Дешево: Выберите два». Мы не можем иметь всё и сразу; руководитель проекта может выбрать только два из трех представленных критериев. Поскольку программы становятся все более сложными, все

более важным становится вопрос выбора используемых ресурсов. Эти ресурсы включают в себя библиотеки, интегрированные среды разработки и отладки. Системный анализ предполагает тестирование производительности и функциональности программы, а затем тестирование с позиций конечного пользователя. Систематический анализ имеет решающее значение для определения последствий оставшихся ошибок.

* * *

Завершая рассмотрение зарубежного опыта обучения школьников программированию, **сформулируем несколько отчетливо проявившихся в последние годы тенденций, которые целесообразно широко использовать и в отечественной школе:**

- раннее начало, обязательность и непрерывность обучения программированию;
- разнообразие современных сред для программирования, использование облачных сервисов;
- проектный подход и командная форма работы в методике обучения программированию.

Список использованных источников

1. *Босова Л. Л.* Школьная информатика в Китае: идеи, которые могут быть нам полезны // Наука и школа. 2016. № 1.

2. *Босова Л. Л., Сорокина Т. Е.* Методика применения интерактивных сред для обучения младших школьников программированию // Информатика и образование. 2014. № 7.

3. *Ершов А. П.* Программирование — вторая грамотность // Проблемы информатики. Институт вычислительной математики и математической геофизики Сибирского отделения РАН (Новосибирск). 2015. № 4 (29).

4. *Каракозов С. Д., Маняхина В. Г.* Обучение информатике в Южной Корее: анализ учебников для младшей и средней школы // Информатика и образование. 2016. № 1.

5. *Кириенко Д. П.* Использование языка Python для обучения программированию // Новые информационные технологии в образовании. Сборник научных трудов 13-й Международной научно-практической конференции «Новые информационные технологии в образовании» (Технологии «1С» для эффективного обучения и подготовки кадров в целях повышения производительности труда), 29–30 января 2013 года. Ч. 2 / под общ. ред. Д. В. Чистова. М.: 1С-Паблишинг, 2013.

6. *Петрусевич Д. А.* Трансформация изучения программирования в школе: от знания основных конструкций к решению сложных задач // Материалы XXIX международной конференции «Современные информационные технологии в образовании» (26 июня 2018 года, Троицк — Москва). М., 2018. http://ito2018.bytic.ru/uploads/materials/conf_2018.pdf

7. Примерная основная образовательная программа начального общего образования. Одобрена решением федерального учебно-методического объединения по общему образованию (протокол от 8 апреля 2015 г. № 1/15). http://fgosreestr.ru/wp-content/uploads/2015/08/primernaja-osnovnaja-obrazovatelnaja-programma-nachalnogo-obshchego-obrazovanija.pdf

8. Примерная основная образовательная программа основного общего образования. Одобрена решением федерального учебно-методического объединения по общему образованию (протокол от 8 апреля 2015 г. № 1/15). http://fgosreestr.ru/wp-content/uploads/2017/03/primernaja-osnovnaja-obrazovatelnaja-programma-osnovogo-obshchego-obrazovanija.pdf

9. Примерная основная образовательная программа среднего общего образования. Одобрена решением федерального учебно-методического объединения по общему образованию (протокол от 28 июня 2016 г. № 2/16). http://fgosreestr.ru/wp-content/uploads/2015/07/Primernaya-osnovnaya-obrazovatelnaya-programma-srednego-obshhego-obrazovaniya.pdf

10. All subjects. Learning resources for adults, children, parents and teachers organised by subject. Bitesize. https://www.bbc.co.uk/education/subjects

11. K–12 Computer Science Framework. 2016. http://www.k12cs.org

12. *Kanemune S., Shirai S., Tani S.* Informatics and Programming Education at Primary and Secondary Schools in Japan // Olympiads in Informatics. 2017. Vol. 11.

13. Perusopetuksen opetussuunnitelman perusteet 2014. http://www.oph.fi/saadokset_ja_ohjeet/opetussuunnitelmien_ja_tutkintojen_perusteet/perusopetus

14. Programmes d'enseignement du cycle des apprentissages fondamentaux (cycle 2), du cycle de consolidation (cycle 3) et du cycle des approfondissements (cycle 4). http://cache.media.education.gouv.fr/file/MEN_SPE_11/35/1/ BO_SPE_11_26-11-2015_504351.pdf

15. Short Course Coding. Specification for Junior Cycle. http://www.curriculumonline.ie/getmedia/cc254b82-1114-496e-bc4a-11f5b14a557f/NCCA-JC-Short-Course-Coding.pdf

ПРАВИЛА ДЛЯ АВТОРОВ

Уважаемые коллеги!

Статьи для публикации в журналах «Информатика и образование» и «Информатика в школе» должны отправляться в редакцию **только через электронную форму на сайте ИНФО (раздел «Авторам → Отправка статьи»)**:

http://infojournal.ru/authors/send-article/

Обращаем ваше внимание, что для отправки статьи необходимо предварительно зарегистрироваться на сайте ИНФО (или авторизоваться — для зарегистрированных пользователей).

С требованиями к оформлению представляемых для публикации материалов можно ознакомиться на сайте ИНФО в разделе **«Авторам»**:

http://infojournal.ru/authors/

Дополнительную информацию можно получить в разделе **«Авторам → Часто задаваемые вопросы»**:

http://infojournal.ru/authors/faq/

а также в редакции ИНФО:

e-mail: readinfo@infojournal.ru

телефон: (495) 140-19-86