Теория и методика обучения и воспитания

УДК 372.800.4
ББК 74.263.2
EDN: WBFYFU

Python или КуМир как первый язык обучения — выбор стратегии

Python or KuMir as a first programming language — choosing a strategy

Пелих О. А.
Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский педагогический государственный университет»
Институт математики и информатики,
техник 1 категории кафедры теории и методики обучения математике и информатике
Москва
E-mail: oa.pelikh@mpgu.su

O. A. Pelikh
Federal State Budgetary Educational Institution of Higher Education "Moscow Pedagogical State University"
Institute of Mathematics and Informatics,
Category 1 Technician,
Department of Theory and Methodology of Mathematics and Informatics Education
Moscow
E-mail: oa.pelikh@mpgu.su

Аннотация
В статье рассматривается проблема обновления содержания раздела «Алгоритмизация и программирование» в курсе информатики 8–9-х классов в контексте низкой учебной мотивации и необходимости ранней профориентации. Особое внимание уделяется психологическому комфорту учащихся при переходе к текстовому программированию. Обосновывается стратегия выбора Python в качестве первого и единственного языка программирования, исключающая необходимость промежуточного этапа изучения специализированных учебных сред. На основе анализа современных педагогических исследований предлагается методический подход, основанный на введении алгоритмических понятий через псевдокод и визуализацию с их немедленной реализацией на Python. Подход ориентирован на формирование устойчивой мотивации через быстрое достижение практических результатов, минимизацию когнитивной нагрузки и создание профориентационно значимых учебных проектов.
Ключевые слова: Python, алгоритмизация, программирование, психологический комфорт, мотивация, ФГОС, основная школа, КуМир, стратегия обучения.

Abstract
The article examines the problem of updating the content of the "Algorithms and Programming" section in the 8–9th grade computer science curriculum, considering the issues of low learning motivation and the need for early career guidance. Special attention is paid to students' psychological comfort during the transition to text-based programming. A strategy is justified for choosing Python as the first and only programming language, eliminating the need for an intermediate stage of learning specialized educational environments. Based on an analysis of contemporary pedagogical research, a methodological approach is proposed, introducing algorithmic concepts through pseudocode and visualization with their immediate implementation in Python. The approach focuses on building sustainable motivation through rapid achievement of practical results, minimizing cognitive load, and creating career-guidance-oriented educational projects.
Keywords: Python, algorithmization, programming, psychological comfort, motivation, Federal State Educational Standards (FSES), secondary school, KuMir, teaching strategy.

Введение

Современные вызовы цифровой трансформации образования требуют пересмотра традиционных подходов к преподаванию программирования в основной школе [1]. Ключевой проблемой, как отмечают исследователи, является низкая учебная мотивация и психологический барьер, который возникает у обучающихся при переходе от блочного программирования к текстовому или на начальном этапе изучения текстового кодирования [2; 3].

В методике преподавания информатики на протяжении последних лет ведется активная дискуссия о выборе первого языка программирования для основной школы [1; 3; 12]. С одной стороны, сохраняется позиция, согласно которой обучение должно строиться

по принципу постепенного усложнения — от визуальных и учебных алгоритмических сред к промышленным языкам программирования [4–7]. С другой стороны, все более распространенной становится точка зрения, предполагающая целесообразность прямого погружения учащихся в текстовое программирование на универсальных языках, прежде всего Python [1; 2; 11].

Аргументы сторон данной дискуссии затрагивают не только вопросы дидактики, но и более широкие аспекты — формирование алгоритмического мышления, психологический комфорт обучающихся, устойчивость мотивации и профориентационную направленность школьного курса информатики.

Целью настоящей статьи является анализ указанных подходов в контексте обучения программированию в 8–9-х классах и обоснование стратегии выбора Python в качестве первого и единственного языка программирования с учетом аргументов сторонников поэтапного перехода и возможных рисков утраты фундаментальных алгоритмических основ.

В контексте данной дискуссии альтернатива «Python или КуМир (Комплект учебных миров)» представляет собой выбор не просто инструмента, а принципиально разных образовательных стратегий. Классическая стратегия, предполагающая постепенное движение от визуальных сред к текстовым, исторически сложилась в отечественной методике преподавания информатики [5]. Однако современные условия — а именно: принципиально иной цифровой опыт поколения, растущие требования к практической и профориентационной ценности школьного курса в условиях ограниченного времени, а также беспрецедентная доступность образовательных ресурсов по Python — диктуют необходимость рассмотреть альтернативную стратегию целенаправленного выбора этого языка в качестве первого и единственного, что позволяет исключить психологически сложный переход между средами и максимально использовать профориентационный потенциал курса. Для понимания полноты контекста необходимо также рассмотреть иные методические подходы к введению программирования в школе, включая использование блочных сред (Scratch, Blockly), специализированных учебных языков (таких как Pascal или учебный алгоритмический язык) и прямое погружение в промышленные языки.

Сравнительный анализ методических подходов к введению программирования

Традиционно выделяются три основных методических подхода к начальному обучению программированию:

1. Визуально-блочный подход (Scratch, Blockly, Kodu). Основан на манипуляции графическими блоками, что позволяет полностью абстрагироваться от синтаксиса и сосредоточиться на логике алгоритмов. Данный подход минимизирует когнитивную нагрузку и подходит для пропедевтики в начальной школе или для учащихся без технической подготовки. Однако его критикуют за создание «защищенной» среды, которая не готовит к реальному текстовому программированию и может формировать упрощенное представление о разработке.

2. Подход на основе специализированных учебных сред и языков (КуМир, учебный алгоритмический язык, PascalABC). Этот подход, занимающий промежуточное положение, предлагает формализованный, но учебно-ориентированный синтаксис, часто на рус-

ском языке. Его цель — плавный переход от визуального представления к текстовому коду, с акцентом на алгоритмическое мышление.

Система «КуМир», в частности, предлагает исполнение алгоритмов на русском языке и визуализацию работы исполнителей, что позволяет абстрагироваться от синтаксиса и сконцентрироваться на логике [7]. Однако дальнейшее развитие вычислительного мышления требует погружения в среду, синтаксис которой приближен к промышленным стандартам. Это создает необходимость перехода от КуМира к текстовому языку, что является критическим моментом, сопровождающимся ростом тревожности и демотивации [2; 3]. Таким образом, стратегия, основанная на КуМире, неизбежно содержит в себе точку «разрыва» — переход к новой среде, требующей освоения нового синтаксиса и парадигмы.

3. Подход прямого погружения в промышленный язык (Python, JavaScript, C#). Данный подход предполагает изучение программирования с первого урока на языке, широко используемом в индустрии. Его основное преимущество — отсутствие необходимости в последующем переходе и возможность сразу формировать профессионально значимые навыки. Главными вызовами являются первоначальная сложность синтаксиса и необходимость сразу работать с текстовым редактором.

Каждый из подходов имеет свои дидактические цели и целевую аудиторию. Визуально-блочный подход служит для формирования первичных представлений, специализированные среды — для углубления алгоритмического мышления, а прямое погружение — для ранней профессионализации. Выбор стратегии зависит от образовательных целей, возрастной группы обучающихся и ресурсного обеспечения.

В современной методике преподавания информатики под глубинным пониманием алгоритмических основ понимается не воспроизведение синтаксических конструкций, а способность обучающегося анализировать задачу, выделять алгоритмическую структуру, осуществлять декомпозицию, выбирать адекватную модель решения и переносить сформированные способы действия в новые контексты [12]. В зарубежных исследованиях данное качество описывается через концепцию вычислительного мышления (computational thinking), включающего такие компоненты, как абстракция, алгоритмизация, обобщение и перенос решений [13].

Python или КуМир: сравнительный анализ в контексте выбора стратегии

Проведем детальное сравнение двух анализируемых вариантов в рамках выбора стратегии для основной школы (8–9-е классы).

КуМир как инструмент начальной алгоритмизации. Система «КуМир» традиционно рассматривается как оптимальный инструмент для начальной алгоритмизации [6; 7]. Ее бесспорным преимуществом является исполнение алгоритмов на русском языке и визуализация работы исполнителей (Робот, Черепаха, Чертежник), что позволяет абстрагироваться от синтаксиса и сконцентрироваться на логике [7]. Среда предоставляет пошаговое выполнение, трассировку переменных и наглядное отображение состояния исполнителя, что существенно облегчает отладку и понимание потока выполнения. Это создает «безопасную» учебную среду с низким порогом входа.

Данные дидактические качества могли бы рассматриваться как преимущество при начале изучения алгоритмизации в более раннем возрасте. Однако в контексте системати-

ческого курса 8–9-х классов, где согласно ФГОС происходит первое целенаправленное знакомство с программированием, ключевыми становятся иные приоритеты: минимизация «точек разрыва» в учебной траектории, формирование практически значимых навыков и ранняя профориентация [9]. Именно с этих позиций необходимо оценивать стратегическую целесообразность использования КуМира на данном этапе.

Однако ключевым ограничением КуМира является его изолированность от современных промышленных стандартов. Язык системы является учебным и не применяется за пределами образовательного контекста. Это приводит к ситуации, когда успешное освоение КуМира не гарантирует легкого перехода к языку вроде Python или C#. Обучающимся приходится заново изучать синтаксис, структуру программы, принципы работы со средой разработки (IDE), что создает двойную нагрузку и часто приводит к фрустрации [2; 3]. Таким образом, стратегия, основанная на КуМире, содержит в себе запланированный «разрыв» учебной траектории, который может негативно сказаться на мотивации.

Вместе с тем сторонники поэтапной модели обучения справедливо указывают на риск формального усвоения программирования при прямом погружении в Python. Отмечается, что учащиеся, начинающие обучение сразу с промышленного языка, нередко демонстрируют способность воспроизводить шаблонные конструкции, но испытывают затруднения при решении задач, требующих осознанного анализа алгоритма, декомпозиции и переноса решения в новую ситуацию [3; 12]. Подобные затруднения особенно проявляются при выполнении контрольных и экзаменационных заданий, ориентированных не на знание синтаксиса, а на понимание логики алгоритма.

Представляется важным подчеркнуть, что указанные проблемы не являются следствием выбора Python как первого языка программирования, а обусловлены преимущественно методикой его преподавания. Как показывают исследования эволюции школьного курса информатики, утрата алгоритмической составляющей возможна при использовании любого инструмента — как учебного, так и промышленного — при смещении акцента с анализа алгоритма на механическое воспроизведение кода [3; 12]. Таким образом, принципиальным является не сам язык программирования, а способ введения алгоритмических понятий и последовательность их освоения.

Исследования, посвященные использованию учебных алгоритмических сред, показывают, что визуализация исполнения алгоритмов и работа с формализованным, но семантически прозрачным языком действительно способствуют формированию первичных представлений об управлении исполнителем, последовательности действий и причинно-следственных связях в алгоритме [6; 7]. В ряде работ отмечается, что на этапе начального обучения такие среды снижают уровень тревожности и позволяют сосредоточиться на логике решения без отвлечения на синтаксические ошибки [4].

Вместе с тем эмпирические данные указывают, что сформированное в рамках учебных сред понимание не всегда автоматически переносится на задачи, требующие самостоятельного проектирования алгоритма в новой языковой и инструментальной среде [3; 12]. Это проявляется в трудностях переноса знаний при переходе к промышленным языкам программирования, особенно в задачах, не имеющих визуального исполнителя.

Python как альтернатива прямого погружения. Альтернативой является стратегия прямого погружения в текстовое программирование на Python. Данный язык, включенный

в Федеральную рабочую программу по информатике [8] и рекомендованный ФГОС [9], обладает минималистичным и читабельным синтаксисом. Ключевым его преимуществом является возможность получения визуально наблюдаемых результатов уже на первом уроке (например, с помощью команд “print()” или графической библиотеки “turtle”). Это обеспечивает немедленную обратную связь, которая является ключевым фактором поддержания мотивации [2]. Ученик с самого начала работает в профессионально значимой среде, что стирает искусственную границу между «учебным» и «настоящим» программированием.

Python позволяет решать те же дидактические задачи, что и КуМир (формирование понятий переменной, условия, цикла, процедуры), но в контексте универсального инструмента. При этом отпадает необходимость в психологически сложном и ресурсозатратном переходе между средами. Учащиеся с первого дня накапливают опыт, актуальный для реальной разработки, что усиливает профориентационную значимость.

Современные исследования показывают, что обучение программированию на Python при корректно выстроенной методике может обеспечивать формирование вычислительного мышления и глубинного алгоритмического понимания, сопоставимого с результатами, достигаемыми в специализированных учебных средах [11; 13–15]. Отмечается, что минималистичный синтаксис Python снижает когнитивную нагрузку, связанную с формальными аспектами языка, позволяя сосредоточиться на структуре алгоритма и логике решения [2; 14].

В ряде эмпирических исследований показано, что учащиеся, изучающие программирование сразу на Python, демонстрируют более высокую способность к переносу алгоритмических знаний при решении нестандартных задач по сравнению с обучающимися, прошедшими этап специализированного учебного языка [13; 15]. Это объясняется тем, что работа в универсальной среде изначально требует явной декомпозиции задачи и осознанного построения алгоритма без опоры на поведение заранее заданного исполнителя.

Предлагаемая стратегия: Python как первый и единственный язык обучения

На основании проведенного анализа предлагается стратегия, в которой Python выбирается в качестве первого и единственного языка программирования. Данный подход реализуется через метод параллельного конструирования, состоящий из трех взаимосвязанных этапов, и призван компенсировать отсутствие специализированной учебной среды за счет методических приемов.

Этап 1: Алгоритмизация на уровне абстракции. На первом этапе происходит введение алгоритмической конструкции на уровне псевдокода или блок-схемы. Учитель объясняет логику конструкции — будь то линейный алгоритм, ветвление или цикл, абстрагируясь от синтаксиса конкретного языка, используя общепринятые способы записи алгоритмов. Это позволяет учащимся сосредоточиться на понимании логики без дополнительной когнитивной нагрузки, связанной с изучением синтаксиса. Данный этап функционально аналогичен работе в КуМире, но использует более универсальные и переносимые средства описания алгоритмов.

Этап 2: Немедленная реализация на Python. Второй этап предполагает немедленную демонстрацию и отработку изученной конструкции на Python. Сразу после теоретиче-

ского объяснения учащиеся знакомятся с синтаксисом данной конструкции в Python и выполняют практические задания. Такой подход обеспечивает прямую связь между теоретическим пониманием алгоритма и его практической реализацией, закрепляя абстрактное понятие через конкретный код. Это устраняет разрыв между знанием и умением.

Этап 3: Визуализация результата для подкрепления понимания. Третий этап направлен на визуализацию результата для подкрепления понимания. Для обеспечения наглядности, аналогичной той, что предоставляет КуМир, активно используются различные инструменты:

- Пошаговое исполнение и отладка: использование режима отладки в средах разработки (IDLE, Thonny, PyCharm Edu) для наблюдения за изменением переменных и потоком выполнения.
- Графические библиотеки: использование библиотек “turtle” или “tkinter” для создания визуального результата выполнения алгоритма (рисование фигур, создание простых интерфейсов).
- Специализированные инструменты визуализации: применение онлайн-симуляторов алгоритмов (например, PythonTutor) или генераторов блок-схем по готовому коду для рефлексии и анализа.

Например, при изучении темы «Циклы» учитель сначала объясняет принцип работы цикла на примере блок-схемы, затем демонстрирует его реализацию в коде на Python с использованием библиотеки “turtle” для рисования геометрических фигур. Такой подход позволяет достичь тех же дидактических целей, что и классический подход с КуМиром — формирование алгоритмического мышления и обеспечение наглядности, — но исключает психологически сложный этап перехода между средами, сразу погружая ученика в практическую, востребованную деятельность. Для пропедевтики объектно-ориентированного подхода эффективно использование визуальных сред [10], что может быть реализовано через те же библиотеки Python.

Например, как указано ранее, понятие цикла сначала вводится на примере блок-схемы, а затем сразу реализуется в коде:

```
import turtle
t = turtle.Turtle()
for i in range(4): # Цикл для рисования квадрата
  t.forward(100)
  t.right(90)
```

Данный код обеспечивает тот же визуальный результат, что и аналогичный алгоритм в КуМире, но делает это в рамках профессионального языка.

Таким образом, утверждение о том, что учащиеся, начинающие обучение с Python, «умеют программировать, но не понимают основ», представляется методически некорректным обобщением. Исследования показывают, что поверхностное усвоение

возможно при любой стратегии обучения — как с использованием КуМира, так и без него — при доминировании репродуктивных заданий и отсутствии этапов анализа и рефлексии [12; 14]. В то же время при систематическом акценте на проектирование алгоритма объяснение решений и перенос знаний Python оказывается эффективным инструментом формирования глубинного понимания алгоритмизации [11; 13].

Условия реализации стратегии обучения программированию на Python

Предложенная в статье стратегия предъявляет определенные требования к ресурсному обеспечению образовательного процесса. Анализ этих условий необходим для оценки возможности ее внедрения в массовой школе.

1. Кадровое обеспечение и готовность педагога.

Ключевым фактором успеха является уровень подготовки учителя. Преподавание Python в рамках описанной методики требует от педагога: уверенного владения базовым синтаксисом Python и стандартными библиотеками; понимания принципов отладки кода и работы в интегрированных средах разработки; способности адаптировать сложные темы под уровень восприятия учащихся 8–9-х классов, используя метод параллельного конструирования.

Отсутствие указанных компетенций может быть компенсировано за счет прохождения программ повышения квалификации, ориентированных на современные методики преподавания программирования. Важно отметить, что переход на Python не требует от учителя глубоких знаний в области Data Science или веб-разработки, достаточно уверенного владения школьным уровнем языка.

2. Материально-техническое и программное обеспечение.

Вопреки распространенному мнению, данная стратегия не требует значительного обновления компьютерного парка школ. Требования к оборудованию остаются на уровне типовых компьютерных классов, соответствующих действующим СанПиН.

- Программная платформа: стратегия может быть реализована как в операционных системах семейства Windows, так и в среде Linux (альт Linux, Astra Linux и др., используемых в российских школах), что обеспечивает ее универсальность.
- Инструментарий: на начальном этапе рекомендуется использовать среду разработки Thonny, которая изначально разрабатывалась для обучения программированию. Она включает встроенный интерпретатор Python, не требует сложной настройки и предоставляет наглядные средства отладки (просмотр значений переменных на каждом шаге). По мере усложнения курса возможен переход на более профессиональные редакторы (Visual Studio Code, PyCharm Edu).
- Учебно-методические материалы: необходимым условием является наличие доступа к цифровым образовательным ресурсам (ЦОР), содержащим банк практико-ориентированных задач, а также к инструментам визуализации, таким как PythonTutor.

Таким образом, критическим фактором является не столько дорогостоящее оборудование, сколько методическая и цифровая компетентность педагога и грамотный подбор программного инструментария, соответствующего возрастным особенностям обучающихся.

Методические аспекты обеспечения быстрого результата для повышения вовлеченности

Как отмечает Д. А. Понамарев [11], вариативность изучения Python позволяет учителю гибко подходить к построению курса. Предлагаемая стратегия полностью использует это преимущество. Важнейшим аспектом является возможность постепенного наращивания сложности решаемых задач без необходимости кардинального изменения синтаксиса [1].

Ключевым методическим принципом является «быстрый успех». Первые уроки должны быть построены вокруг задач, результат выполнения которых очевиден и визуализирован:

1. Вывод информации: использование “print()” для приветствий, простых расчетов, форматированного вывода.
2. Простая графика: рисование базовых фигур с помощью “turtle” для иллюстрации линейных алгоритмов и циклов.
3. Интерактивные программы: создание простейших диалогов с использованием “input()” и условных операторов (калькулятор, викторина).

Практико-ориентированный подход реализуется через решение задач, имеющих личностную значимость для учащихся: автоматизация рутинных расчетов для математики, создание простых игр (угадай число, крестики-нолики в консоли) или анализ текстовых данных (подсчет слов, частотный анализ) [11]. Такой подход не только обеспечивает быстрое достижение результата, но и демонстрирует практическую ценность изучаемого материала, формируя осознанную мотивацию к обучению. Учитель выступает в роли наставника, который помогает превратить абстрактное знание в конкретный, работающий продукт, пусть и небольшой.

Профориентационный аспект изучения Python и анализ востребованности

Стратегия выбора Python в качестве первого языка имеет мощный профориентационный эффект, который становится одним из центральных аргументов в ее пользу. Раннее знакомство с профессиональной средой разработки способствует формированию осознанного выбора IT-специальности [1]. В отличие от учебных сред, Python является языком, на котором создаются реальные веб-приложения (бэкенд на Django/Flask), программы для анализа данных (с использованием Pandas, NumPy), инструменты автоматизации, игры и даже программное обеспечение для искусственного интеллекта (библиотеки Scikit-learn, TensorFlow).

Анализ открытых статистических материалов Федерального государственного бюджетного научного учреждения «Федеральный институт педагогических измерений» (ФИПИ) демонстрирует устойчивую тенденцию роста доли обучающихся, выбирающих Python при выполнении заданий ОГЭ по информатике, что может рассматриваться как косвенный показатель его доступности и методической прозрачности для школьников. Этот выбор обусловлен не только простотой синтаксиса, но и широкой доступностью учебных материалов, сообществ поддержки и потенциальными карьерными перспективами.

Изучая Python с первых шагов, школьник использует тот же инструмент, что и профессиональные разработчики, что позволяет накапливать опыт в актуальной среде и способствует осознанному выбору будущей специальности [2; 4]. Он знакомится с понятиями библиотек, фреймворков, систем управления пакетами (pip), работой в командной

строке — всем тем, что составляет повседневную практику IT-специалиста. Это создает прочный фундамент для дальнейшего углубленного изучения программирования в старшей школе (профильный уровень) или в учреждениях среднего профессионального и высшего образования.

Заключение

Проведенное исследование позволяет утверждать, что стратегия выбора Python в качестве первого и единственного языка программирования в основной школе является жизнеспособной и педагогически обоснованной альтернативой классическому пути через КуМир. Она не отвергает дидактические принципы наглядности и пошагового изучения, а переосмысливает их, используя современный инструментарий.

Данная стратегия предлагает современное решение ключевых проблем мотивации и профориентации. Она позволяет:

- Обеспечить психологический комфорт за счет исключения травмирующего перехода между средами;
- Сформировать устойчивую положительную мотивацию через немедленное достижение практических результатов в профессиональной среде;
- Реализовать профориентационный потенциал курса через раннее и полное погружение в промышленный стандарт;
- Сформировать у учащихся единую, непрерывную траекторию обучения программированию, от базовых конструкций к сложным проектам, без смены парадигмы и инструментов.

Таким образом, ответ на вопрос, вынесенный в заголовок, «Python или КуМир?» может быть сформулирован как целенаправленный выбор в пользу Python, основанный на стратегии прямого погружения в актуальную и практическую программистскую деятельность. Этот выбор соответствует как тенденциям развития цифровой экономики, так и потребностям современных школьников в осмысленном, прикладном и мотивирующем обучении.

Сравнительный анализ двух стратегий — с использованием специализированной учебной среды КуМир и стратегии прямого погружения в Python — позволяет сделать вывод, что обе они потенциально способны обеспечить формирование алгоритмического мышления. Однако различие заключается в характере формируемого опыта. Стратегия с КуМиром эффективна на этапе первичного знакомства с алгоритмами, но содержит риск фрагментации учебной траектории. Стратегия прямого погружения в Python при методически выверенной организации обучения обеспечивает не только сопоставимый уровень глубинного понимания, но и более высокий потенциал переноса знаний и профориентационной значимости.

Список литературы

1. Городняя Л. В. Выбор решений для языка учебного программирования / Л. В. Городняя // Научный сервис в сети Интернет. 2024. № 26. С. 57–72. DOI 10.20948/abrau-2024–3. EDN RVDVWH.
2. Пасихин А. И. Язык программирования Python в повседневной работе учителя // Педагогическое образование в России. 2023. № 5. С. 45–52. URL: https://www.elibrary.ru/item.asp?id=82595523 (дата обращения: 15.10.2024).
3. Вострокнутов И. Е., Шегурова И. Г. Эволюция содержания школьного курса информатики и ИКТ в области использования школьного алгоритмического языка программирования // Педагогические

чтения в ННГУ: сборник научных статей, Нижний Новгород — Арзамас, 10–11 декабря 2015 года. Нижний Новгород — Арзамас: Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского, Арзамасский филиал, 2015. С. 500–503. EDN WYMLYV.4.

4. Леонов А. Г. Система КуМир в непрерывном школьном курсе информатики // Ярославский педагогический вестник. 2012. Т. 3, № 4. С. 28–44. EDN PXMMVL.
5. Леонов А. Г., Первин Ю. А. Элементы программирования в непрерывном курсе школьной информатики // Ярославский педагогический вестник. 2013. Т. 3, № 1. С. 45–50. EDN REOPCR.
6. Семтина Е. А., Проценко С. И. Обучение основам алгоритмизации на базе системы кумир в основной школе // Информационные технологии. Проблемы и решения. 2021. № 4(17). С. 140–144. EDN COTSKN.
7. Козлов С. В., Быков А. А. Обучение школьников выполнению и анализу простейших алгоритмов управления исполнителями в среде кумир // Современные наукоемкие технологии. 2023. № 10. С. 123–128. DOI 10.17513/snt.39803. EDN KICPUL.
8. Федеральная рабочая программа основного общего образования. Информатика (базовый уровень) (для 7–9 классов образовательных организаций) / Минпросвещения России, Институт содержания и методов обучения им. В.С. Леднева. М., 2025. — URL: https://edsoo.ru/wp-content/uploads/2025/07/2025_ooo_frp_informatika-7–9_baza.pdf (дата обращения: 10.03.2026).
9. Об утверждении федерального государственного образовательного стандарта начального общего образования [Электронный ресурс]: приказ Министерства просвещения Российской Федерации от 31.05.2021 № 286 (зарегистрирован 05.07.2021 № 64100). URL: http://publication.pravo.gov.ru/document/0001202107050028 (дата обращения: 10.03.2026).
10. Павлов Д. И., Бутарев К. В. Пропедевтика объектно-ориентированного программирования с использованием среды Greenfoot. Опыт разработки // Актуальные проблемы обучения математике и информатике в школе и вузе: материалы IV Междунар. науч. конф. (Москва, 04–05 дек. 2018 г.). В 2 ч. Ч. 1. М.: АКФ «Политоп», 2018. С. 176–180.
11. Пономарев Д. А. Варианты изучения языка программирования Python в школьном курсе информатики // Образование. Технологии. Качество: Материалы III Всероссийской научно-практической конференции, Саратов, 29–30 марта 2019 года. Саратов: Издательство «Перо», 2019. С. 130–136. EDN SRDTFI.
12. Кузнецов А. А., Захарова Т. Б. Школьная информатика: вчера, сегодня, завтра // Информатика и образование. 2014. № 10(259). С. 3–6. EDN TBRVDN.
13. Wing J. M. Computational Thinking // Communications of the ACM. 2006. Vol. 49, No. 3. P. 33–35.
14. Grover S., Pea R. Computational Thinking in K-12: A Review of the State of the Field // Educational Researcher. 2013. Vol. 42, No. 1. P. 38–43.
15. Sentance S., Waite J. Teaching Computer Programming in Schools: A Review of Approaches and Tools // ACM Transactions on Computing Education. 2017. Vol. 17, No. 3.