УДК 004.42:373.5.016

GOOGLE COLAB КАК ИНСТРУМЕНТ РАЗВИТИЯ АЛГОРИТМИЧЕСКОГО МЫШЛЕНИЯ НА УРОКАХ ИНФОРМАТИКИ

Л. А. Крицук

Минский городской институт развития образования, Беларусь

Аннотация. В статье исследуется дидактический потенциал облачной среды разработки Google Colab для формирования алгоритмического мышления учащихся. Сравнительный анализ с традиционными подходами к обучению программированию выявил ключевые преимущества платформы: минимальные затраты на начало работы, объединение теоретического материала, кода и результатов в одном документе, встроенная поддержка совместной работы. Разработана трехэтапная модель интеграции Google Colab в уроки информатики. Предложены методические рекомендации для учителей информатики.

Ключевые слова: Google Colab, алгоритмическое мышление, облачные технологии, методика преподавания, Python, теоретическая модель.

GOOGLE COLAB AS A TOOL FOR DEVELOPING ALGORITHMIC THINKING IN COMPUTER SCIENCE LESSONS

L. A. Kritsuk

Minsk City Institute of Education Development, Belarus

Abstract. The article explores the didactic potential of the Google Colab cloud development environment for developing students' algorithmic thinking. A comparative analysis with traditional approaches to teaching programming revealed the key advantages of the platform: minimal setup costs, integration of theoretical material, code, and results in a single document, built-in support for collaboration. A three-stage model for integrating Google Colab into computer science lessons has been developed. Methodological recommendations for computer science teachers are proposed.

Keywords. Google Colab, algorithmic thinking, cloud technologies, teaching methodology, Python, theoretical model.

Введение

В современных условиях развития образования алгоритмическое мышление стало фундаментальной компетенцией. Оно определяется как способность формализовать задачу, построить четкую последовательность шагов для ее решения и проверить результаты [1; 2]. Целенаправленное развитие этого мышления – одна из ключевых задач при изучении информатики.

Традиционные подходы к обучению программированию, основанные на локальных средах разработки, сталкиваются с трудностями: разнородное программное обеспечение в компьютерных классах, затраты времени на установку и настройку [3]. Эти факторы создают дополнительную нагрузку на учащихся, отвлекая их от содержания алгоритмизации на технические проблемы.

Перспективной альтернативой выступают облачные среды разработки, в частности Google Colab. Хотя их используют в высшем образовании и науке [4], потенциал этой платформы для развития алгоритмического мышления школьников в условиях стандартных учебных программ требует дополнительного изучения.

Цель исследования – разработать и обосновать модель использования среды Google Colab на уроках информатики для развития алгоритмического мышления с учетом дидактических возможностей и ограничений платформы.

Методологическая основа исследования включает:

Компаративный анализ – сравнение Google Colab с традиционными средами программирования по критериям доступности, визуализации и другим параметрами.

Метод моделирования – создание трехэтапной педагогической модели интеграции платформы в учебный процесс.

Теоретический анализ – изучение психолого-педагогических основ формирования алгоритмического мышления в контексте теории поэтапного формирования умственных действий П. Я. Гальперина и деятельностного подхода.

Эмпирическую базу составили: анализ педагогического опыта использования облачных сред; результаты апробации модели в рамках педагогической практики.

Для систематизации данного потенциала обратимся к теоретическим основам формирования алгоритмического мышления и роли облачных сред в этом процессе.

Теоретические аспекты использования облачных сред в образовании

Алгоритмическое мышление – часть более широкого понятия «вычислительное мышление» [2; 5]. Оно включает умение разбивать задачу на части, находить

закономерности и строить эффективные последовательности действий. Чтобы развивать его, нужно создать среду, где технические сложности не мешают ученику думать.

Облачные технологии, и, в частности, интерактивные блокноты, помогают создать такую среду [3; 6]. Google Colab объединяет исполняемый код, пояснения, графики и формулы в одном документе [4]. Возможность выполнять код по частям, сразу видеть результат и вносить правки помогает понять алгоритм и соответствует идее поэтапного формирования умственных действий [5].

С точки зрения деятельностного подхода [7] Google Colab – это инструмент для проектной работы. Создание мини-проектов с использованием библиотек Python помогает достигать как предметных, так и метапредметных образовательных результатов.

Переходя от теоретических предпосылок к практической оценке инструмента, проведем сравнительный анализ дидактического потенциала Google Colab. Для оценки места Google Colab в учебном процессе проведен сравнительный анализ с традиционными подходами к обучению программированию. Система критериев разрабатывалась с учетом релевантных требований к формированию алгоритмического мышления и институциональных условий преподавания информатики (табл. 1).

На основе проведенного сравнения сформулируем ключевые дидактические возможности и ограничения платформы:

- Снижение когнитивной нагрузки на начальном этапе за счет отсутствия необходимости установки и настройки программного обеспечения (ПО).
- Повышение наглядности обучения через формат блокнота, объединяющего теорию, код и результат выполнения.
- Стимулирование учебной кооперации с помощью встроенных механизмов совместной работы.
- Обеспечение равных стартовых условий благодаря независимости от мощности домашнего компьютера.

Ограничения платформы:

- Зависимость от интернет-инфраструктуры.
- Неполное соответствие олимпиадным и экзаменационным средам.
- Ограниченность инструментов отладки.

С учетом выявленных возможностей и ограничений была разработана трехэтапная модель интеграции Google Colab в учебный процесс, направленная на поэтапное развитие алгоритмического мышления.

Таблица 1. – Сравнительный анализ сред программирования

Критерий	Локальные IDE (PascalABC.NET, Idle)	Облачная среда (Google Colab)
Технические требования и доступность	Требуют установки и конфигурации на каждом компьютере.	Не требуют установки. Доступ с любого устройства с браузером и интернетом.
Стартовый барьер для ученика	Высокий (интерфейс IDE, настройка интерпретатора).	Низкий. Интерфейс интуитивен, среда готова к работе после входа в аккаунт.
Визуализация	Зависит от среды; требует дополнительных настроек	Интеграция кода, текста и графиков по умолчанию
Инструменты коллаборации	Отсутствуют или ограничены (системы контроля версий сложны для школьников).	Встроенные инструменты совместного редактирования в реальном времени, с комментированием.
Соответствие средам проведения олимпиад и ЦТ	Полное. Учащиеся работают в регламентированной среде.	Частичное. Требуется адаптация учащихся к олимпиадным IDE на этапе подготовки.
Автономность работы	Полная. Не зависит от наличия интернет-соединения.	Отсутствует. Работа невозможна при нестабильном или отсутствующем интернете.
Возможности отладки	Мощные. Включают пошаговое выполнение, точки останова, просмотр переменных.	Базовые. Отладка ведется преимущественно через отладочный вывод (print).
Интеграция с учебным процессом	Проверена годами практики, имеет методическую поддержку.	Требует разработки новых методических подходов и материалов.

Теоретическая модель интеграции Google Colab в образовательный процесс

С учетом выявленных возможностей и ограничений разработана трехэтапная модель интеграции Google Colab в учебный процесс для развития алгоритмического мышления.

Этап 1. Ознакомительный (7-8 классы).

Цель: сформировать базовые представления о программировании и принципах работы в облачной среде.

Содержание: знакомство с интерфейсом Google Colab. Основы синтаксиса Python. Решение элементарных задач на арифметические операции, вывод данных, работу со строками.

Методические особенности: акцент на оперативном получении результата. Задания носят исследовательский характер (например, создание специализированного калькулятора, генерация текстовых паттернов).

Этап 2. Базовый алгоритмический (8-9 классы).

Цель: освоить основные алгоритмические конструкции и принципы структурного программирования.

Содержание: реализация и анализ линейных, разветвляющихся и циклических алгоритмов. Работа с базовыми структурами данных (списки, строки, словари). Построение графиков для иллюстрации работы алгоритмов.

Методические особенности: применение библиотек для визуализации; компаративный анализ эффективности алгоритмов. Сравнение эффективности различных алгоритмов решения одной задачи через замеры времени.

Этап 3. Проектный и исследовательский (9-11 классы).

Цель: сформировать умение применять алгоритмические знания для решения прикладных задач.

Содержание: разработка учебных проектов с использованием специализированных библиотек. Коллективная работа над проектами.

Методические особенности: организация групповой работы средствами платформы. Акцент на этапах проектирования: постановка задачи, выбор алгоритмов, реализация, тестирование, представление результатов.

Для иллюстрации практической реализации модели рассмотрим пример урока на тему «Алгоритм сортировки пузырьком».

Цель: сформировать у учащихся понимание принципа работы алгоритма сортировки пузырьком, умение его реализовывать на Python и анализировать эффективность.

Структура учебного блокнота:

1. Импорт библиотек: matplotlib.pyplot, numpy, random, time.
2. Markdown-ячейки с теоретическим материалом.
3. Вспомогательные функции (генерация случайных списков, тестирование).
4. Заготовка для реализации функции bubble_sort.
5. Визуализация процесса сортировки и анализ производительности.

Шаблон учебного блокнота Google Colab (фрагмент с заданиями)

```python
# УРОК: АЛГОРИТМ СОРТИРОВКИ ПУЗЫРЬКОМ
# Сохраните копию этого блокнота на своем Google Диске!
# 1. Импорт библиотек
import matplotlib.pyplot as plt
import random
import time
# 2. Теоретическая часть (в виде Markdown-ячеек)
# ... (Ученики читают о принципе работы алгоритма) ...
# 3. ПРАКТИЧЕСКАЯ ЧАСТЬ: РЕАЛИЗАЦИЯ АЛГОРИТМА
# ЗАДАНИЕ 1 (Базовый уровень): Реализуйте базовую версию сортировки пузырьком.
# Заполните пропуски в коде (обозначены как `...`).
def bubble_sort_basic(arr):
    """
    Базовая реализация алгоритма сортировки пузырьком.
    """
    n = len(arr)
    for i in range(n):
        for j in range(0, ...):  # ПОДСКАЗКА: до n-i-1
            if arr[j] > arr[...]:  # ПОДСКАЗКА: j+1
                # Поменяйте элементы arr[j] и arr[j+1] местами
                ...
    return arr
# ЗАДАНИЕ 2 (Продвинутый уровень): Реализуйте оптимизированную версию с флагом.
def bubble_sort_optimized(arr):
    """
    Оптимизированная реализация с флагом swapped.
    """
    n = len(arr)
    for i in range(n):
        swapped = ...
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:
```

```
                arr[j], arr[j+1] = arr[j+1], arr[j]
                swapped = ...
        # Если обменов не было, выходим из цикла
        if ...:
            break
    return arr
# 4. ТЕСТИРОВАНИЕ
print("=== ТЕСТИРОВАНИЕ ===")
test_array = [64, 34, 25, 12, 22, 11, 90]
print("Исходный массив:", test_array)
# Протестируйте свои функции здесь
# sorted_array = bubble_sort_basic(test_array.copy())
# print("Отсортированный массив (базовый):", sorted_array)
# 5. ВИЗУАЛИЗАЦИЯ И АНАЛИЗ ПРОИЗВОДИТЕЛЬНОСТИ
# ... (Код для визуализации и замера времени предоставлен учителем) ...
```

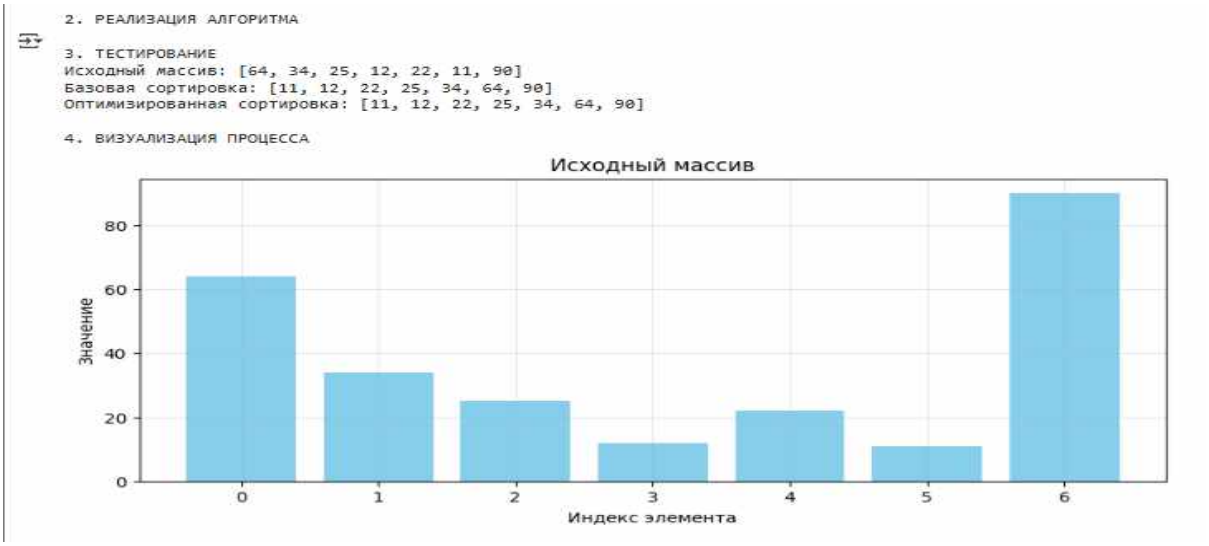

Рисунок 1 – Визуализация процесса работы алгоритма сортировки пузырьком

```
5. АНАЛИЗ ПРОИЗВОДИТЕЛЬНОСТИ
Размер: 100 | Базовая: 0.0006с | Оптимизированная: 0.0004с
Размер: 500 | Базовая: 0.0122с | Оптимизированная: 0.0162с
Размер: 1000 | Базовая: 0.0501с | Оптимизированная: 0.0513с

6. ВЫВОДЫ
• Алгоритм пузырьковой сортировки прост для понимания
• Оптимизация с флагом swapped значительно ускоряет сортировку почти отсортированных массивов
• Временная сложность: O(n²) в худшем случае, O(n) в лучшем случае
• Алгоритм эффективен только для небольших массивов
```

Рисунок 2 – Сравнение времени выполнения базового и оптимизированного алгоритмов

Ход урока

Актуализация (5 мин): Демонстрация проблемы неупорядоченного списка. Ученики открывают шаблон блокнота и сохраняют копию на своем Google Диске.

Изучение нового материала (15 мин): Совместное прохождение Markdown-ячеек с теорией. Акцент на ключевых идеях: множественные проходы, сравнение соседей, «всплывание» максимального элемента.

Практическая работа (20 мин): реализация функции bubble_sort с дифференциацией:

Базовый уровень: код с пропусками.

Продвинутый уровень: самостоятельная реализация с оптимизацией (флаг swapped).

Первичное закрепление (10 мин): тестирование реализаций, обсуждение типичных ошибок.

Визуализация и углубление (10 мин): поэтапная визуализация процесса сортировки в виде столбчатой диаграммы. Обсуждение временной сложности $O(n^2)$.

Критерии оценивания (10-балльная система):

1-3 балла: функция содержит синтаксические ошибки, не проходит базовые тесты.

4-5 баллов: функция корректно реализована, проходит базовые тесты.

6-7 баллов: реализация оптимизирована (например, с использованием флага), функция не изменяет исходный массив, предоставлен базовый анализ операций.

8-9 баллов: выполнены все условия на 6-7 баллов, а также реализована визуализация процесса сортировки или проведено сравнительное исследование производительности на разных наборах данных.

10 баллов: выполнены все условия на 8-9 баллов, а также предложена и реализована дополнительная оптимизация алгоритма или проведен его углубленный анализ (например, сравнение с другим алгоритмом сортировки).

На основе проведенного анализа сформулированы рекомендации для интеграции Google Colab на уроках информатики:

Принцип дополнительности. Использовать Google Colab как дополнение к локальным средам на начальных и средних этапах обучения, а также для проектных работ.

Фокус на развитии метапредметных навыков. Максимально использовать коллаборативные возможности платформы для организации групповых проектов.

Акцент на визуализации и исследовании. Активно задействовать графические библиотеки Python для превращения абстрактных алгоритмов в наглядные процессы.

Дифференциация и персонализация. Использовать формат блокнотов для создания индивидуальных траекторий обучения.

Заключение

Проведенное исследование позволяет сделать вывод о дидактической целесообразности интеграции облачной среды разработки Google Colab на уроках информатики в качестве эффективного инструмента развития алгоритмического мышления.

Ключевые преимущества платформы – минимизация технических барьеров, интерактивность и наглядность формата «живого» блокнота, а также поддержка учебной коллаборации – делают ее перспективным ресурсом для модернизации образовательного процесса.

Основным ограничением является зависимость от интернет-инфраструктуры и необходимость параллельного формирования навыков работы в локальных средах, используемых при итоговой аттестации. Перспективы дальнейших исследований связаны с разработкой учебно-методических комплексов на базе Google Colab для ключевых тем школьной информатики и их последующей педагогической апробацией.

Список библиографических ссылок (на языке оригинала)

1. Паутова Н.Г. Формирование алгоритмического мышления школьников в процессе обучения информатике. *Информатика и образование*. 2019; 5:12-18.

2. Wing J.M. Computational Thinking. *Communications of the ACM*. 2006; 49(3):33-35.

3. Роберт И.В. Теория и методика информатизации образования (психолого-педагогический и технологический аспекты). Москва: БИНОМ. Лаборатория знаний, 2014. 398 с.

4. Kluyver T., et al. Jupyter Notebooks – a publishing format for reproducible computational workflows. *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. 2016. P. 87-90.

5. Гальперин П.Я. Лекции по психологии. Москва: Книжный дом «Университет», 2002. 400 с.

6. Цветкова М.С., Богомолова О.Б. Информатика и ИКТ: примерная рабочая программа: 7-9 классы. Москва: БИНОМ. Лаборатория знаний, 2022. 112 с.

7. Леонтьев А.Н. Деятельность. Сознание. Личность. Москва: Смысл, Academia, 2005. 352 с.

References (на английском языке)

1. Pautova N.G. Formirovanie algoritmicheskogo myshleniya shkol'nikov v protsesse obucheniya informatike [Formation of algorithmic thinking in schoolchildren in the process of teaching computer science]. *Informatika i obrazovanie*. 2019; 5:12-18. (In Russian)

2. Wing J. M. Computational Thinking. *Communications of the ACM*. 2006; 49(3):33-35.

3. Robert I.V. Teoriya i metodika informatizatsii obrazovaniya (psikhologo-pedagogicheskiy i tekhnologicheskiy aspekty). M.: BINOM. Laboratoriya znaniy, 2014. 398 p. (In Russian)

4. Kluyver T. et al. Jupyter Notebooks -- a publishing format for reproducible computational workflows. *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. 2016. P. 87-90.

5. Gal'perin P.Ya. Lektsii po psikhologii. M.: Knizhnyy dom «Universitet», 2002. 400 p. (In Russian)

6. Tsvetkova M.S., Bogomolova O.B. Informatika i IKT: primernaya rabochaya programma: 7-9 klassы. M.: BINOM. Laboratoriya znaniy, 2022. 112 p. (In Russian)

7. Leont'ev A.N. Deyatel'nost'. Soznanie. Lichnost'. M.: Smysl, Academia, 2005. 352 p. (In Russian)