ТЕХНИЧЕСКИЕ НАУКИ

УДК 004.4

Обзор платформ для обучения программированию: функциональные возможности, методы автоматизированной проверки и безопасность выполнения кода

Д.В. Голосуев, В.В. Долгов
Донской государственный технический университет, г. Ростов-на-Дону, Российская Федерация

Аннотация
Проведён обзор существующих платформ для обучения программированию и проверки навыков разработчиков. Рассматриваются их функциональные возможности, преимущества и ограничения. Особое внимание уделено методам автоматизированной проверки решений задач. Изучены подходы к безопасному выполнению недоверенного кода, а также перспективы реализации совместного редактирования программного кода. Предложенные решения и выводы направлены на оптимизацию процессов обучения и повышения безопасности платформ.

Ключевые слова: обучение программированию, проверка навыков программирования, автоматизированная проверка решений задач, автоматизированное тестирование, выполнение недоверенного кода, контейнеризация, Docker, изоляция в песочнице, совместное редактирование, CRDT, WebRTC, WebSocket

Для цитирования. Голосуев Д.В., Долгов В.В. Обзор платформ для обучения программированию: функциональные возможности, методы автоматизированной проверки и безопасность выполнения кода. *Молодой исследователь Дона.* 2025;10(2):29–32.

Overview of Platforms for Learning Programming: Functionality, Automated Verification Methods and Secure Code Execution

Danil V. Golosuev, Vasilii V. Dolgov
Don State Technical University, Rostov-on-Don, Russian Federation

Abstract
The paper provides an overview of existing platforms for learning programming and assessing developers' skills. Functional capabilities, advantages and limitations of such platforms were studied. Special attention was paid to the methods of automated verification of problem solving. Approaches to secure execution of untrusted code, as well as prospects for leveraging collaborative code editing, were investigated. The proposed solutions and conclusions aim at optimising the study process and improving security of the platforms.

Keywords: learning programming, assessment of programming skills, automated verification of problem solving, automated testing, untrusted code execution, containerization, Docker, sandbox-based isolation, collaborative editing, CRDT, WebRTC, WebSocket

For Citation. Golosuev DV, Dolgov VV. Overview of Platforms for Learning Programming: Functionality, Automated Verification Methods and Secure Code Execution. *Young Researcher of Don*. 2025;10(2):29–32.

Введение. Интерактивные образовательные системы обучения и проверки навыков предоставляют эффективный и увлекательный способ развития навыков программирования. Такие платформы позволяют участникам получать мгновенную обратную связь и анализировать свои ошибки, что ускоряет процесс обучения. Интерактивные платформы создают сообщества, где пользователи могут обмениваться опытом, решать задачи и совершенствоваться вместе. Платформы могут предлагать задачи различных уровней сложности и направлений, охватывая широкий круг интересов и специализаций. Решение алгоритмических задач развивает навыки, востребованные в профессиональной разработке, и помогает в подготовке к собеседованиям. Таким образом, разработка образовательных систем по программированию остаётся актуальной.

Цель исследования заключается в проведении обзора источников, касающегося проектирования и разработки интерактивной информационной системы обучения и проверки навыков программирования.

Сравнение существующих аналогов. Существует множество информационных систем для обучения программированию с автоматизированной проверкой решений задач; однако во время анализа можно выявить недостатки существующих решений. Именно они создают предпосылки для разработки нового решения, более полно соответствующего потребностям пользователей.

LeetCode [1] — одна из самых популярных платформ для изучения алгоритмов и структур данных. Её сильной стороной являются удобный и красивый интерфейс, подходящий для начинающих, встроенный редактор кода с подсветкой синтаксиса и наличие теоретических материалов. Сервис позволяет тестировать программы на своих входных данных, хотя возможность создания собственных задач не предоставляется. Это отличная платформа для изучения алгоритмов, но её функционал сосредоточен исключительно на этой области.

CodeWars [2] выделяется возможностью создавать свои задачи, что делает её особенно привлекательной для пользователей, которые ценят творческий подход и хотят делиться своими идеями с сообществом. Несмотря на отсутствие теоретических материалов, платформа имеет красивый интерфейс и удобна для новичков. Встроенный редактор кода с подсветкой синтаксиса дополняется системой тестирования программ через автотесты, что является особенностью данной платформы среди рассматриваемых.

Информатикс [3] — русскоязычный сервис, но его устаревший и неудобный интерфейс снижает привлекательность. Код приходится писать и отлаживать локально, так как встроенного редактора нет, а решения загружаются в виде файлов. Навигация по задачам усложнена, поиск осуществляется по идентификаторам. Теоретические материалы присутствуют в большом объёме. Добавлять задачи на сайт могут только некоторые пользователи через специальную заявку, что ограничивает гибкость работы с платформой. Собственные тестовые данные указать нельзя.

HackerRank [4] ориентирован на алгоритмические задачи и поддерживает добавление пользовательских задач, что способствует разнообразию контента. Оценка решений на платформе может быть кастомизирована с помощью скриптов на языках Python, Java и C++. HackerRank не предоставляет теоретических материалов, фокусируясь исключительно на практике. Платформа поддерживает множество языков программирования и предоставляет сертификаты по различным навыкам, которые можно добавить в профессиональное портфолио или профиль в LinkedIn.

Методы проверки решений. На платформах Информатикс и HackerRank проверка решений задач осуществляется на основе обработки консольного ввода и вывода. Пользовательская программа запускается с заранее подготовленными тестовыми данными, которые подаются на её консольный ввод. Программа выводит результат работы в консольный вывод, и система тестирования сравнивает полученный вывод с эталонным. Для проверки программы на других тестовых наборах данных осуществляется её повторный запуск.

На платформе LeetCode проверка решений задач осуществляется на основе передачи тестовых данных в текстовом формате. Входные данные десериализуются и преобразуются в аргументы для вызова пользовательской функции, реализующей вычисления. Оценка корректности выполняется путём сравнения возвращаемого функцией результата с эталонным значением. Такой подход исключает необходимость перезапуска программы, что позволяет более точно измерять время выполнения алгоритма.

На платформе CodeWars проверка решений задач осуществляется с использованием автоматизированных тестов (автотестов). Автор задачи пишет код, который выполняет проверку пользовательского решения на корректность. Такой подход способствует изучению не только алгоритмов и структур данных, где ключевым является получение правильных выходных данных для заданных входных, но и развитию концепций объектно-ориентированного программирования, включая проектирование классов, реализацию методов с заданными сигнатурами, использование наследования, полиморфизма и других принципов ООП. Корректность выполнения программ оценивается с помощью набора автотестов, проверяющих их поведение в различных сценариях.

Запуск недоверенного кода. Для проверки решений задач требуется запуск пользовательского кода, при этом любой пользовательский код, отправляемый на платформу, является недоверенным и потенциально вредоносным. Недоверенный код нельзя выполнять на хост-системе, поэтому для его выполнения применяется концепция песочницы или sandboxing. Изучены литературные данные по теме применения sandboxing в олимпиадах по информатике. В [5] описывается вариант «песочницы» на основе Linux control group и namespaces. Эти же механизмы используются в системе контейнеризации Docker, которую автор применил в качестве «песочницы». В [6] затрагиваются вопросы безопасности системы оценки задач. Авторы адаптировали требования к системе контейнеризации Docker [7]. Если обобщить, основным требованием является всяческое ограничение возможностей для недоверенного кода.

Для проверки решений задач требуется выполнение пользовательского кода. Однако любой пользовательский код, отправляемый на платформу, считается недоверенным и потенциально вредоносным. Исполнение недоверенного кода на хост-системе недопустимо, поэтому для его обработки применяется концепция песочницы (sandboxing). В литературе изучены различные аспекты использования песочниц на олимпиадах по информатике. В источнике [5] описывается вариант реализации песочницы с использованием Linux control group и namespaces. Эти механизмы также активно применяются в системе контейнеризации Docker, которая используется автором как основа для создания песочницы. Исследование, представленное в [6], поднимает вопросы безопасности системы оценки задач. Авторы адаптировали требования к системе контейнеризации Docker из [7]. Основное требование заключается в максимальном ограничении возможностей для выполнения недоверенного кода, чтобы минимизировать потенциальные угрозы.

По умолчанию Docker создает изолированную файловую систему для каждого контейнера. Если используется файловая система btrfs, overlay2, windowsfilter или zfs, размер может быть ограничен параметром --storage-opt size = 120G. Еще одним вариантом ограничения размера является монтирование временной файловой системы tmpfs с параметром --tmpfs /usr/app:size = 32M. В дополнение к этому можно ограничить количество операций ввода-вывода с помощью параметров --device-read-IOPS и --device-write-IOPS, хотя ограничение размера файловой системы в оперативной памяти может не всегда иметь практический смысл. Также возможно создать файловую систему только для чтения с использованием параметра --read-only.

Docker предоставляет возможность ограничить ресурсы контейнера, такие как объем оперативной памяти и процессорное время. Даже если конкретное задание по программированию не предусматривает ограничение по памяти или процессору, важно задать эти лимиты, чтобы предотвратить влияние на другие контейнеры и хостовые процессы. Это можно реализовать с помощью параметров --memory 128M и --cpus 1.0.

Сетевое взаимодействие также может быть ограничено параметром --network = none. Избыточное количество процессов в контейнере может перегружать планировщик задач. Поэтому рекомендуется ограничить количество запущенных процессов с помощью параметра --ulimit nproc = 64, а также лимитировать количество файловых дескрипторов с параметром `--ulimit nofile = 512. На рис. 1 представлена команда для запуска контейнера с недоверенным кодом.

```
docker run -it --rm \
  --read-only \
  --tmpfs /usr/app:size=32M \
  --memory 128M \
  --cpus 1.0 \
  --ulimit nproc=64 --ulimit nofile=512 \
  --network=none \
  gcc:13.2.0
```

Рис. 1. Команда запуска Docker контейнера

Совместное редактирование кода. На платформе планируется реализовать функционал совместного редактирования кода, что позволит пользователям учиться программированию или проверять свои навыки вместе с другими. Для этого необходимо синхронизировать содержимое редактора в реальном времени, включая положение курсоров и выделенные области текста. В качестве редактора кода можно использовать Monaco Editor или CodeMirror.

Синхронизация состояния редактора может быть выполнена с использованием CRDT (Conflict-free replicated data type) или механизмов операционной трансформации (OT, operational transformation). В качестве протокола обмена данными можно выбрать между WebRTC и WebSocket. WebRTC обеспечивает организацию прямого соединения между браузерами пользователей, что минимизирует задержки и снижает нагрузку на сервер. С другой стороны, использование WebSocket может быть более простым вариантом, так как предполагает применение центрального сервера для обмена данными. Для реализации данного функционала можно воспользоваться библиотекой Yjs, которая уже включает в себя типы CRDT [8]. Yjs предлагает интеграцию с протоколами WebRTC и WebSocket, а также поддерживает редакторы кода Monaco Editor и CodeMirror.

Заключение. В ходе работы были изучены существующие образовательные платформы и их методы проверки решений задач. Проведен анализ материалов по безопасному запуску недоверенного кода с применением системы контейнеризации Docker. Также выявлены возможности для реализации функционала совместного редактирования кода. Обзор источников позволит перейти к следующему этапу — проектированию интерактивной образовательной системы.

Список литературы

1. *Платформа LeetCode.* URL: https://leetcode.com/ (дата обращения: 21.01.2025).
2. *Платформа CodeWars.* URL: https://www.codewars.com/ (дата обращения: 21.01.2025).
3. *Платформа Информатикс.* URL: https://informatics.msk.ru/ (дата обращения: 21.01.2025).
4. *Платформа HackerRank.* URL: https://www.hackerrank.com/ (дата обращения: 21.01.2025).
5. Mareš M, Blackham B. A New Contest Sandbox. *Olympiads in Informatics*, 2012;6:100–109.
6. Mareš, M. (2021). Security of Grading Systems. Olympiads in Informatics, 15, 37–52.
7. *Система контейнеризации Docker.* URL: https://docs.docker.com/engine (дата обращения: 21.01.2025).
8. *Бесконфликтные реплицированные типы данных CRDT.* URL: https://en.wikipedia.org/wiki/Conflict-free_replicated_data_type (дата обращения: 21.01.2025).

Об авторах:

Данил Витальевич Голосуев, магистрант направления программной инженерии Донского государственного технического университета (344003, Российская Федерация, г. Ростов-на-Дону, пл. Гагарина, 1), danil2003.2043@gmail.com

Василий Валерьевич Долгов, кандидат технических наук, доцент, заведующий кафедрой программного обеспечения вычислительной техники и автоматизированных систем Донского государственного технического университета (344003, Российская Федерация, г. Ростов-на-Дону, пл. Гагарина, 1), bdolgov@donstu.ru

Конфликт интересов: **авторы заявляют об отсутствии конфликта интересов.**

Все авторы прочитали и одобрили окончательный вариант рукописи.

About the Authors:

Danil V. Golosuev, Master's Degree Student in Software Engineering Specialism, Don State Technical University (1, Gagarin Sq., Rostov-on-Don, 344003, Russian Federation), danil2003.2043@gmail.com

Vasilii V. Dolgov, Cand.Sci. (Engineering), Head of the Computer Engineering and Automated Systems Department, Don State Technical University (1, Gagarin Sq., Rostov-on-Don, 344003, Russian Federation), bdolgov@donstu.ru

Conflict of Interest Statement: **the authors declare no conflict of interest.**

All authors have read and approved the final manuscript.