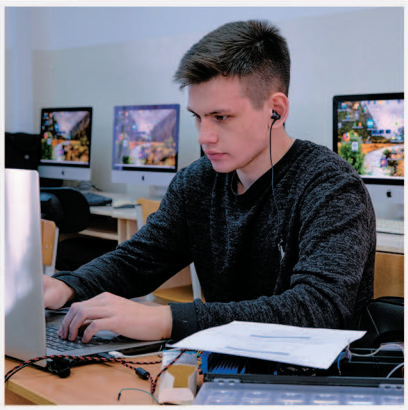
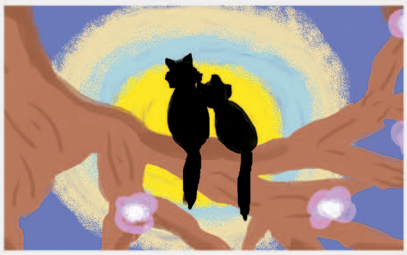
НАУЧНО-ПРАКТИЧЕСКИЙ ЖУРНАЛ 0+ ИЗДАЕТСЯ С 2002 ГОДА

ИНФОРМАТИКА В ШКОЛЕ

№ 6'2023

ISSN 2221-1993 www.infojournal.ru





30.01.2024
—
31.01.2024

XXIV МЕЖДУНАРОДНАЯ
НАУЧНО-ПРАКТИЧЕСКАЯ КОНФЕРЕНЦИЯ

НОВЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В ОБРАЗОВАНИИ

ОСНОВНЫЕ НАПРАВЛЕНИЯ РАБОТЫ:

1. Развитие технологической платформы и прикладных решений 1С, как инструментария для гипераватоматизации процессов создания новых ценностей.
2. Экосистема 1С для поддержки педагогической деятельности в цифровой среде образовательной организации.
3. Развитие форм практикоориентированного обучения и компетентностного подхода в образовании. Участие представителей экосистемы 1С в системе общего, профессионального и дополнительного образования.
4. Применение передовых технологий и методик обучения при подготовке высококвалифицированных специалистов в области цифровизации бизнеса.

В 2023 году в конференции приняли участие более 7400 человек.

Подробнее о тематиках конференции и условиях участия см. сайт educonf.1c.ru

МЕРОПРИЯТИЯ В РАМКАХ КОНФЕРЕНЦИИ:

- Пленарные и секционные заседания
- Мастер-классы по использованию программных продуктов фирмы «1С»
- Вернисаж программных и методических разработок

Традиционно 31 января 2024 года проводится 1С:День студента. Это отличный шанс для студентов оценить свои силы и попасть на стажировку в ИТ-компанию!

Участие бесплатное для всех сотрудников образовательных организаций и органов управления образованием.

Обязательная предварительная регистрация открыта до 29 января 2024 года на сайте educonf.1c.ru

ФИРМА «1С»
Оргкомитет конференции:
Тел./факс: +7 (495) 688-90-02
Email: npk@1c.ru. Web: educonf.1c.ru

1С® ФИРМА "1С"

ИНФОРМАТИКА В ШКОЛЕ

ИЗДАЕТСЯ С СЕНТЯБРЯ 2002 ГОДА

№ 6 (185) декабрь 2023

Содержание

Свидетельство о регистрации средства массовой информации ПИ № 77-12068 от 11 марта 2002 г.

Издатель ООО «Образование и Информатика»
119261, Россия, г. Москва, Ленинский пр-т, д. 82/2, комн. 6
Телефон: +7 (495) 140-19-86
E-mail: readinfo@infojournal.ru
Сайт издательства: http://infojournal.ru/
Почтовый адрес: 119270, Россия, г. Москва, а/я 15

Подписано в печать 15.12.23.
Формат 60×90 1/8. Усл. печ. л. 11,0
Тираж 2000 экз. Заказ № 4409.
Отпечатано в типографии «ЛАЙДЕР ПРИНТ»
142104, Россия, Московская область, г. Подольск, ул. Свердлова, д. 26, тел.: +7 (495) 212-91-99, +7 (926) 204-49-31, e-mail: info@book-expert.ru

НАУЧНО-ПРАКТИЧЕСКИЙ ЖУРНАЛ

«ИНФОРМАТИКА В ШКОЛЕ»

УЧРЕДИТЕЛЬ:

ИЗДАТЕЛЬСТВО
«ОБРАЗОВАНИЕ И ИНФОРМАТИКА»

ISSN 2221-1993

Журнал входит в Перечень российских рецензируемых научных изданий ВАК, в которых должны быть опубликованы основные научные результаты диссертаций на соискание ученых степеней доктора и кандидата наук

Контакты

Главный редактор
akulll@mail.ru

Редакция
readinfo@infojournal.ru

Отдел распространения
info@infojournal.ru

Телефон:
+7 (495) 140-19-86

Почтовый адрес
119270, Россия, г. Москва, а/я 15

Сайт журнала
http://school.infojournal.ru

Уважаемые коллеги! Дорогие друзья!

Вот и подошел к концу 2023 год. В этом году в нашу жизнь прочно вошли аббревиатуры ФООП, ФОП и ФРП. Обозначаемые ими федеральные основные общеобразовательные программы, федеральные образовательные программы и федеральные рабочие программы — важный шаг к единому образовательному пространству, формирование которого — одна из ключевых задач, стоящих перед нашей школой. Важными инструментами формирования единого образовательного пространства, обеспечивающего доступность ресурсов, равенство условий и возможностей для обучающихся, являются федеральный перечень учебников и федеральный перечень электронных образовательных ресурсов. Решению этой же задачи подчинено и введение ФГИС «Моя школа», которая обеспечивает единый доступ к образовательным сервисам и цифровым учебным материалам для учеников, родителей и учителей и представляет собой яркое проявление процесса цифровой трансформации системы образования.

Как не утратить профессиональную индивидуальность? Как сохранить вариативность образовательного процесса? Ответы на эти вопросы, дорогие друзья, нам предстоит найти в наступающем году.

Еще одна задача на ближайшую перспективу связана с активно заявившим о себе в этом году генеративным искусственным интеллектом, способным генерировать тексты, изображения, видео по запросам пользователя. Мы используем учебные материалы, созданные системами искусственного интеллекта, наши ученики прибегают к помощи искусственного интеллекта при выполнении домашних заданий. Наша задача — понять и принять перемены, происходящие в современном мире, научиться пользоваться новыми возможностями, которые предоставляет искусственный интеллект.

Учитель был и остается ключевой фигурой образовательного процесса, без его инициативы и творчества невозможно решение задач, стоящих перед современной школой. Редакционная коллегия журнала «Информатика в школе» ждет от вас, наши уважаемые авторы, новые материалы, готова публиковать их на своих страницах для вас, дорогие наши читатели.

Желаем вам встретить наступающий 2024 год с прекрасным настроением в кругу родных и друзей, полными творческих замыслов и сил для их воплощения.

Испытывая большой соблазн обратиться за помощью к искусственному интеллекту для составления оригинального новогоднего поздравления, мы тем не менее решили завершить это обращение к вам, дорогие наши читатели, поэтическими строками великого А. С. Пушкина — фрагментом романа в стихах «Евгений Онегин»:

Идет волшебница зима.
Пришла, рассыпалась; клоками
Повисла на суках дубов;
Легла волнистыми коврами
Среди полей, вокруг холмов;
Брега с недвижною рекою
Сравняла пухлой пеленою;
Блеснул мороз. И рады мы
Проказам матушки зимы.

С Новым годом, дорогие друзья!

Искренне ваша,
редакция журнала
«Информатика в школе»

DOI: 10.32517/2221-1993-2023-22-6-4-17

А. А. Марко, С. А. Лакомкин, А. С. Барабанов
Московский городской педагогический университет, г. Москва, Россия

ИТ-ВЕРТИКАЛЬ: КОНЦЕПЦИЯ И РЕАЛИЗАЦИЯ ИЗУЧЕНИЯ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ В ОСНОВНОЙ ШКОЛЕ

Аннотация

В статье представлено описание концепции и модели изучения информационных технологий на уровне основного общего образования в условиях реализации ФГОС. На примере осуществления проекта в VII—IX классах общеобразовательных организаций города Москвы раскрыто содержание образовательной траектории подростка с учетом запроса на раннюю профилизацию по направлению «Информатика и информационные технологии». Особенности образовательной траектории реализованы через специальный курс билингвального программирования, новое содержание курса технологии (с включением специальных разделов по мехатронике, технологиям создания цифровых двойников, электронике и схемотехнике) и экосистему курсов внеурочной деятельности в формах, специфических для разработчиков программных продуктов. Особенностью изучения информатики и физики на углубленном уровне является наличие межпредметных кейсов с использованием цифровых лабораторий. Представлены механизмы и инструменты интеграции в образовательную программу специальных знаний по актуальным вопросам ИТ-отрасли. Отдельное место занимают вопросы создания необходимой инфраструктуры для реализации проекта: материально-технической базы, подготовки кадров и методического сопровождения проекта. Представлены уникальные функциональные и дидактические возможности цифровых инструментов для деятельностного погружения школьников в процесс освоения содержания образовательной программы. Приведены примеры авторских заданий для формирования умений и навыков школьников, диагностики и контроля полученных знаний.

Ключевые слова: ИТ-вертикаль, билингвальное программирование, информационные технологии, цифровые лаборатории, виртуальные лаборатории.

1. Введение

Сегодня мы являемся свидетелями множества инициатив, связанных с совершенствованием системы общего образования на всех ее уровнях. Эти инициативы во многом вызваны запросами экономики на формирование основы технологического суверенитета страны. Очевидно, что наибольшую поддержку и дальнейшее продвижение имеют проекты, направленные на трансляцию лучших практик профориентационной работы [11] и проекты,

Контактная информация

Марко Антон Александрович, канд. физ.-мат. наук, зам. директора, Институт развития профильного обучения, Московский городской педагогический университет, г. Москва, Россия; *адрес:* 129226, Россия, г. Москва, 2-й Сельскохозяйственный пр-д, д. 4, корп. 1; *e-mail:* markoaa@mgpu.ru

Лакомкин Сергей Александрович, ст. методист, Институт развития профильного обучения, Московский городской педагогический университет, г. Москва, Россия; *адрес:* 129226, Россия, г. Москва, 2-й Сельскохозяйственный пр-д, д. 4, корп. 1; *e-mail:* lakomkinsa@mgpu.ru

Барабанов Алексей Сергеевич, ст. методист, Институт развития профильного обучения, Московский городской педагогический университет, г. Москва, Россия; *адрес:* 129226, Россия, г. Москва, 2-й Сельскохозяйственный пр-д, д. 4, корп. 1; *e-mail:* barabanov@mgpu.ru

A. A. Marko, S. A. Lakomkin, A. S. Barabanov
Moscow City University, Moscow, Russia

IT VERTICAL: CONCEPT AND IMPLEMENTATION OF STUDYING INFORMATION TECHNOLOGIES IN BASIC SCHOOL

Abstract

The article presents a description of the concept and model for the study of information technologies at the level of basic general education in the context of the implementation of the Federal State Educational Standards. Based on the example of the implementation of the project in grades VII—IX of general education organizations in Moscow, the content of the educational trajectory of a teenager is revealed, taking into account the request for early profiling in the field of "Informatics and information technologies". Features of the educational trajectory are implemented through a special bilingual programming course, new content of the course of technology (with the inclusion of special sections on mechatronics, technologies for creating digital twins, electronics and circuit design) and an ecosystem of extracurricular activities courses in forms specific to software developers. A feature of studying informatics and physics at an advanced level is the presence of interdisciplinary cases using digital laboratories. Mechanisms and tools for integrating specialized knowledge on current issues in the IT industry into the educational program are presented. A special place is occupied by the issues of creating the necessary infrastructure for the implementation of the project: material and technical base, personnel training and methodological support of the project. The unique functional and didactic capabilities of digital tools for active immersion of schoolchildren in the process of mastering the content of the educational program are presented. Examples of author's assignments are given for developing the skills of schoolchildren, diagnosing and monitoring acquired knowledge.

Keywords: IT vertical, bilingual programming, information technologies, digital laboratories, virtual laboratories.

направленные на формирование у школьников специальных знаний и умений для решения отраслевых задач.

Одним из флагманов таких инициатив является проект системы столичного образования «Предпрофессиональные классы в московских школах», который реализуется в Москве с 2016 года. На основе обобщения опыта реализации проектов предпрофессионального образования, базирующихся на технологическом профиле ФГОС [1, 8], — «ИТ-класс в московской школе» [2, 6] (с 2019 года) и «Инженерный класс в московской школе» [4] (с 2016 года) — стала очевидной необходимость более ранней профориентации школьников в рамках технологического профиля в сочетании с пропедевтикой специальных знаний и умений как по традиционным школьным предметам — физика, математика, информатика, так и по предметам, специфическим для современной инженерии и информационно-технологической отрасли, — робототехника, моделирование и прототипирование, электроника и схемотехника, информационная безопасность, программирование. При этом в каждом из направлений рассматриваются различные уровни подготовки — как общеобразовательные, так и предпрофессиональные. Также большое внимание уделяется подготовке для соревнований команд, способных оперативно решать сложные задачи, в том числе в области робототехники [7].

Целью данной статьи является *описание опыта столичной системы образования в ранней предпрофессиональной подготовке школьников в области информационных технологий на примере городского проекта «ИТ-вертикаль»*.

2. О проекте «ИТ-вертикаль»

«ИТ-вертикаль» [5] — проект предпрофильной подготовки учащихся VII—IX классов, направленный на формирование у них знаний и прикладных умений в области информационных технологий для решения теоретических и практико-ориентированных задач.

Проект предполагает подготовку обучающегося к профессиональному выбору: выявляются его интересы и склонности, отслеживается его отношение к ситуации выбора, определяются мотивы выбора будущей профессии.

Практико-ориентированный характер предпрофильной подготовки позволяет подростку овладеть эффективными приемами использования цифровых инструментов для решения задач, получения новых знаний, исследований, а также для решения профессиональных задач современной инженерии.

На основе приобретенного опыта **у обучающегося появляется возможность определить одно из трех ключевых направлений своей дальнейшей самореализации:**

1) использование информационных технологий и инструментов в качестве пользователя (*среднее профессиональное образование*);
2) использование информационных технологий как инструмента для решения инженерных, научных, экономических и иных задач (*инженерный класс, предпринимательский класс, академический класс*);
3) создание и развитие ИТ-инструментов для различных отраслей экономики (*ИТ-класс*).

Решение задачи самоопределения возможно за счет реализации **двух направлений предпрофильной подготовки, объединенных в образовательную траекторию обучающегося проекта «ИТ-вертикаль»:**

1) *освоение системы академических знаний* по физике, математике и информатике на углубленном уровне;
2) *профессиональная ориентация, присвоение специальных знаний* в области информационных технологий и их применения для решения задач науки и техники через профессиональные пробы в приоритетных направлениях ИТ-отрасли: технологии искусственного интеллекта, мехатроника и робототехника, электроника и микропроцессорная техника, создание цифровых двойников, информационная безопасность, технологии связи.

3. Структура образовательной траектории школьника в проекте «ИТ-вертикаль»

3.1. Математика, информатика, физика

Изучение на углубленном уровне системообразующих предметов — математики, информатики и физики — включает в себя не только повышение учебной нагрузки по дисциплинам за счет увеличения количества учебных часов, но и решение межпредметных учебных задач (например, моделирование физического эффекта путем построения математической модели и ее визуализации средствами информационных технологий).

Содержательно обучение информатике и физике в классах проекта «ИТ-вертикаль» на углубленном уровне основывается на федеральных рабочих программах основного общего образования по данным предметам (углубленный уровень) [10, 13, 15]. Метапредметные результаты обучения включат в себя формирование ИТ-компетенций у обучающихся за счет использования ИТ-инструментария, в том числе уникальной экосистемы виртуальных лабораторий Библиотеки Московской электронной школы [3].

Например, при изучении темы «Законы постоянного тока» для сборки экспериментальных установок используются не традиционные учебные наборы кабинета физики, а беспаечные макетные платы и радиотехнические детали в сочетании с предварительной сборкой электрических цепей в виртуальных лабораториях «Электродинамика» или «Использование микроконтроллеров».

Законы механического движения рекомендуется изучать, в числе прочего, с использованием цифровых лабораторий по физике, состоящих из ультразвуковых датчиков расстояния и программного обеспечения для обработки результатов проводимого эксперимента.

Для обработки массивов данных, полученных в результате экспериментов по физике, используются табличные процессоры, позволяющие провести анализ данных эксперимента и их визуализацию.

3.2. Программирование

Обязательный учебный курс «Программирование» реализуется в VII—IX классах на двух языках про-

граммирования — Python и C++. Объем курса на период обучения в классе проекта «ИТ-вертикаль» — 102 часа.

Изучение алгоритмических конструкций и перенос этих конструкций на языки программирования помогают школьникам ***развить логическое мышление и аналитические навыки:*** при написании кода необходимо проводить анализ задачи, разбивать ее на более мелкие части и определять логическую последовательность действий для ее решения.

Одновременное обучение программированию на Python и C++ — это ***современный билингвальный подход***, благодаря которому учащийся в процессе освоения курса изучает сразу два языка программирования среди самых популярных в индустрии информационных технологий. Знание этих языков будет полезно в работе программиста, разработчика, аналитика данных, а также во многих других профессиях.

Неотъемлемым спутником процесса изучения программирования является постоянное ***развитие творческого мышления***. Ведь для поиска оптимального, а иногда и единственного решения обучающимся приходится постоянно экспериментировать и применять творческий подход, так как практически для каждой задачи есть несколько возможных путей ее решения. В процессе творчества и экспериментов при решении задач у школьников происходит не только первичное закрепление изученного материала, но и разносторонняя отработка практических навыков.

Учебный курс «Программирование» помогает учащимся ***развить навыки решения проблем***. При программировании часто возникают ошибки и сложности, и ученикам приходится искать способы исправления ошибок, преодоления сложностей, что развивает у них умение находить пути решения сложных задач. А использование двух языков, Python и C++, помогает расширить набор инструментов для решения. С одной стороны, в начале изучения курса билингвальный подход и сам создает повышенную сложность, школьники допускают много ошибок, но, с другой стороны, в процессе освоения курса этот подход позволяет раскрыть перед обучающимися особенности каждого языка, в том числе особенности синтаксиса, и поиск ошибок будет происходить уже в конкретных местах в зависимости от используемого для решения задачи языка программирования.

Python и C++ являются достаточно универсальными языками программирования, они могут быть использованы для создания различных типов программ. Изучение этих языков помогает учащимся лучше понять основы программирования и создания алгоритмов, что облегчит им в будущем и изучение смежных языков.

Тренды ИТ-отрасли сейчас таковы, что для анализа данных и оперативной разработки используется Python, а при разработке программ для конечных устройств и различных микроконтроллеров — C++ или другие языки, сходные с ним по синтаксису, алгоритмическим конструкциям, способам организации памяти и другим характеристикам. Поэтому ***учебный курс «Программирование» можно рассматривать как одно из звеньев цепи решения задачи профессиональной ориентации школьников***. Изучение сразу двух языков позволит учащемуся понять, какое из направлений программной разработки ему интереснее: работа с данными и разработка программных продуктов (Python) или программирование устройств (C++), что также позволит сделать осознанный выбор между инженерией, ИТ-разработкой и аналитикой данных (профессией Data Scientist).

Курс «Программирование» носит практико-ориентированный характер с опорой на решение отраслевых кейсов: анализ массива реальных данных, программирование движения робота, разработка управляющего кода для 3D-принтера или станка лазерной резки.

В курсе ***большое внимание уделено инструментарию программирования***. Изучение инструментов технологий программирования дает возможность школьникам познакомиться с технологией разработки, отладки и внедрения программного обеспечения. Например, использование системы контроля позволяет учащимся глубоко погрузиться в проблематику разработки программных продуктов, найти применение своих сильных сторон в командной разработке и увидеть процесс создания программного обеспечения с фрагментацией задач.

В процессе преподавания курса используется ***технология «образовательного реверса»***, состоящая в том, что обучающийся изначально не пишет код программы «с нуля», а разбирает действующий код и исследует возможности его дальнейшей модификации. Это позволяет снизить порог входа в сложный курс и познакомить школьников с реверсивными технологиями (закрепить знание о реверсе ученики смогут в курсе технологии при работе с 3D-сканерами).

3.3. Технология

Профессиональная ориентация в структуре образовательной траектории проекта реализуется за счет **расширения содержания предмета «Технология»** (табл. 1).

Проект «ИТ-вертикаль» при его запуске в 2022/2023 учебном году привнес новизну в предмет «Технология», предполагая обязательное включение в курс обучения модулей «Электроника и микропроцессорная техника», «Мехатроника и робототехника», «Моделирование и прототипирование». В настоящее время федеральная рабочая программа предмета «Технология» на уровне основного общего образования предполагает изучение модулей «Робототехника», «3D-моделирование, прототипирование, макетирование», «Компьютерная графика. Черчение» [14].

Предмет «Технология» с расширенным содержанием в VII—IX классах позволяет решать задачи по формированию знаний и умений школьников в вопросах создания цифровых двойников (моделирование и прототипирование), электронных устройств с управляющими контроллерами (электроника и схемотехника), мехатронных и роботизированных устройств (мехатроника и робототехника).

За весь период обучения в проекте «ИТ-вертикаль» учащийся знакомится со всеми тремя модулями предмета «Технология». Каждый год школьник изучает новое направление, получает базовые знания по нему и применяет их на практике. Практическая ориентированность курса позволяет подросткам освоить выполнение сложных задач через решение специальных учебных кейсов, без освоения множества теоретических вопросов.

Таблица 1

Содержание модулей предмета «Технология»

№ п/п	Модуль	Содержание модуля
1	Основы микропроцессорной техники	• Основы схемотехники. Аналоговая электроника. • Цифровая электроника. • Программирование микроконтроллеров
2	Основы моделирования и прототипирования	• Основы черчения. • Основы трехмерного моделирования. • Основы прототипирования. • Основы лазерных фрезерных технологий. • Основы реверсивного инжиниринга
3	Основы мехатроники и робототехники	• Введение в программирование на С-подобных языках. • Основы работы с исполнительными устройствами и датчиками. • Алгоритмы управления робототехническими устройствами. • Базовые задачи мобильной робототехники. • Основы мехатроники и промышленной робототехники. • Основы пилотирования мобильных роботов и квадрокоптеров. • Творческие задания на конструирование и программирование автоматических устройств

Такой подход позволяет осуществить ***профессиональные пробы*** и дает возможность школьнику понять сферу его интересов для дальнейшего развития: пользование готовыми инструментами и самосовершенствование в этом направлении (среднее профессиональное образование — высокотехнологичные рабочие специальности) или развитие и создание новых инструментов и программных решений (дальнейшее получение теоретических знаний в предпрофессиональных классах и в вузах).

Отсутствие процедуры итоговой государственной аттестации по предмету «Технология», а также необходимость проведения промежуточного контроля достижения образовательных результатов по введенным модулям потребовали создания ***инструментов и механизмов независимой диагностики***. Неотъемлемым компонентом образовательного проекта «ИТ-вертикаль» стал инструмент диагностики усвоения знаний обучающихся по предмету «Технология» на весь период обучения — «ИТ-марафон».

«ИТ-марафон» [16] — серия диагностических мероприятий, которые дают участникам проекта «ИТ-вертикаль» равные возможности для демонстрации результатов освоения нового содержания образовательной программы, повышают интерес и мотивацию школьников к изучению информационных технологий через соревновательную механику, а также способствуют развитию и совершенствованию диагностических материалов по проверке ИТ-компетенций на базе виртуальных лабораторий московской электронной школы.

4. Содержание модулей курса «Технология»

Остановимся подробнее на содержании обязательных модулей курса «Технология» и инструмента мониторинга эффективности их реализации.

4.1. Основы микропроцессорной техники

На первом этапе реализации модуля «Основы микропроцессорной техники» — «**Основы схемотехники. Аналоговая электроника**» — обучающиеся знакомятся с основами схемотехники и изучают аналоговую электронику. Задача данного этапа — формирование умений по сборке простейших электрических цепей на основе технического описания.

Обучающиеся, получив базовые теоретические знания об основных электронных компонентах, переходят к практике сборки электрических цепей на реальном оборудовании или на оборудовании виртуальной лаборатории МЭШ «Физика. Электродинамика».

В качестве диагностики усвоения знаний на первом этапе «ИТ-марафона» обучающимся предлагается выполнить серию заданий для самостоятельного решения. В «ИТ-марафоне» 2022/2023 учебного года обучающиеся реализовывали задания «Два ключа», «Три яркости светодиода» и «Капризные лампы» (рис. 1).

Второй этап освоения модуля «Основы микропроцессорной техники» предполагает изучение темы **«Цифровая электроника»**. Здесь формируются умения учащихся по чтению и сборке схем с использованием логических элементов.

Обучающиеся после знакомства с основами цифровой логики изучают логические элементы схемотехники: И, ИЛИ, НЕ, И-НЕ, ИЛИ-НЕ.

Полученные знания применяются на практике путем сборки логических схем как на реальном оборудовании, так и на оборудовании виртуальной лаборатории МЭШ «Технология. Построение логических схем».

В качестве диагностики на втором этапе «ИТ-марафона» обучающимся предлагается выполнить задания, предусматривающие сборку логических схем с использованием различных источников данных: таблицы истинности, диаграммы Венна и др. (рис. 2).

Цель изучения третьей темы модуля «Основы микропроцессорной техники», **«Программирование микроконтроллеров», —** формирование навыков сборки и программирования устройств с использованием микроконтроллеров.

Обучающиеся знакомятся:

- с платой Arduino;

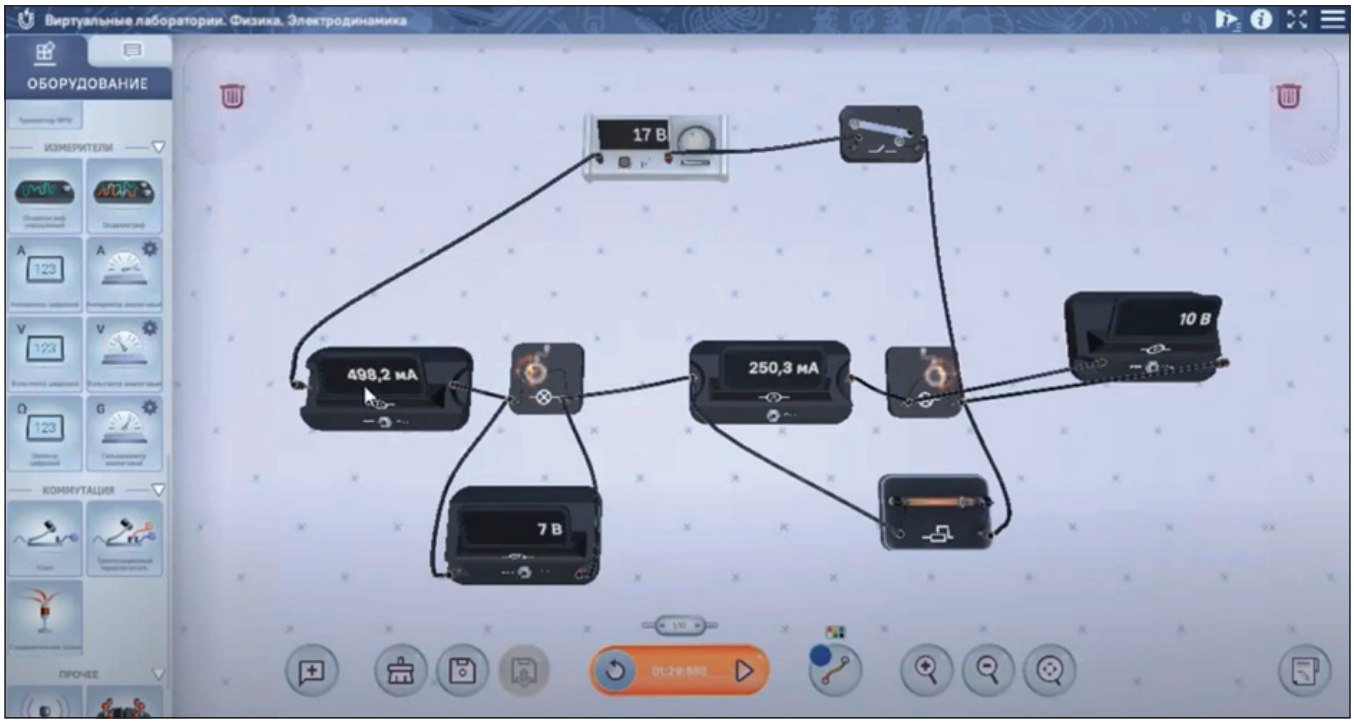


Рис. 1. Задание «Капризные лампы»

- с различными инструментами получения данных из внешней среды (датчиками, клавиатурами и др.);
- с основами программирования микроконтроллеров;
- с управлением внешними устройствами (сервоприводом, двигателем).

Практическое применение знаниям обучающиеся находят в реализации проектов с использованием реального оборудования, а также оборудования виртуальной лаборатории «Технология. Программирование микроконтроллеров».

В качестве диагностики усвоения знаний по теме «Программирование микроконтроллеров» «ИТ-марафон» в 2022/2023 учебном году предполагал реализацию двух проектов: системы из пешеходно-автомобильного

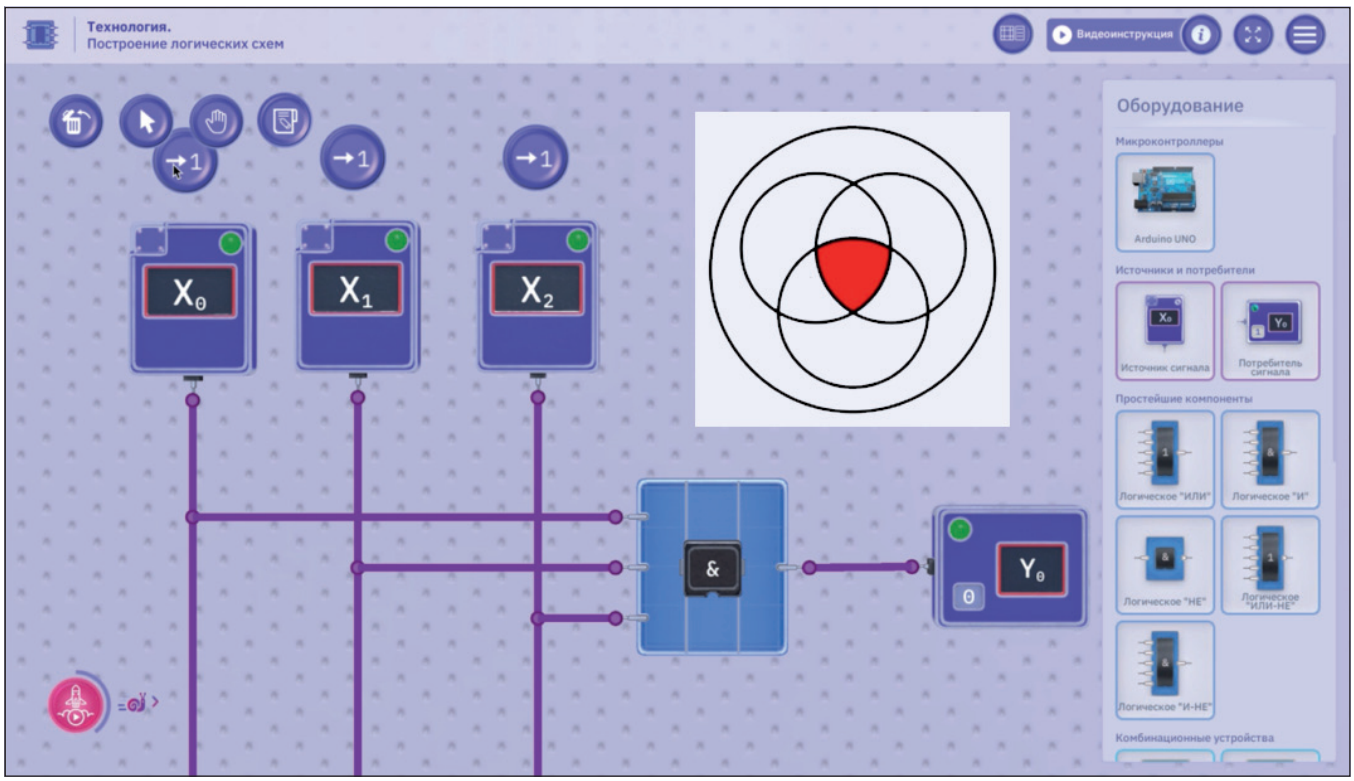


Рис. 2. Построение логической схемы на основе диаграммы Венна

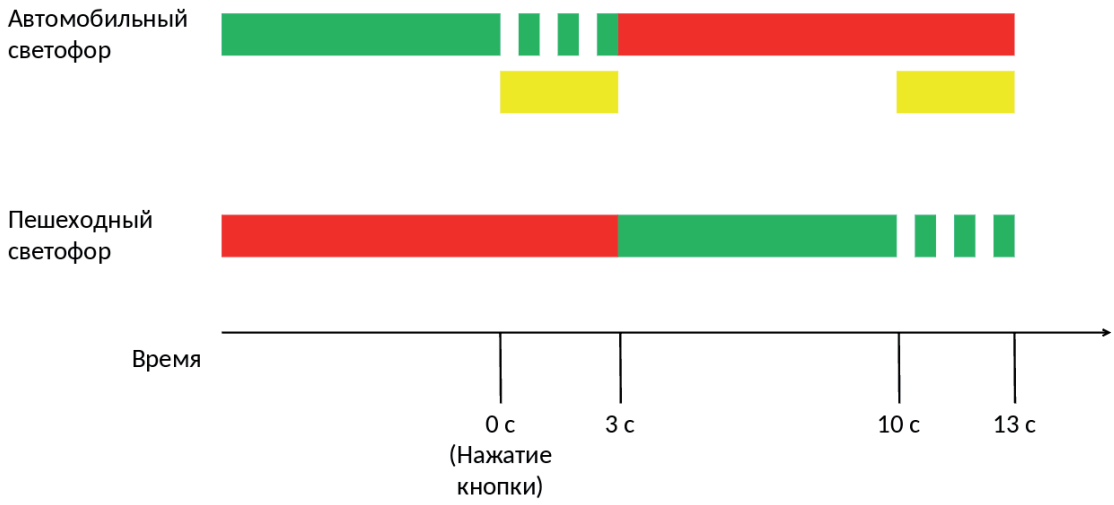


Рис. 3. Диаграмма работы пешеходно-автомобильного светофора

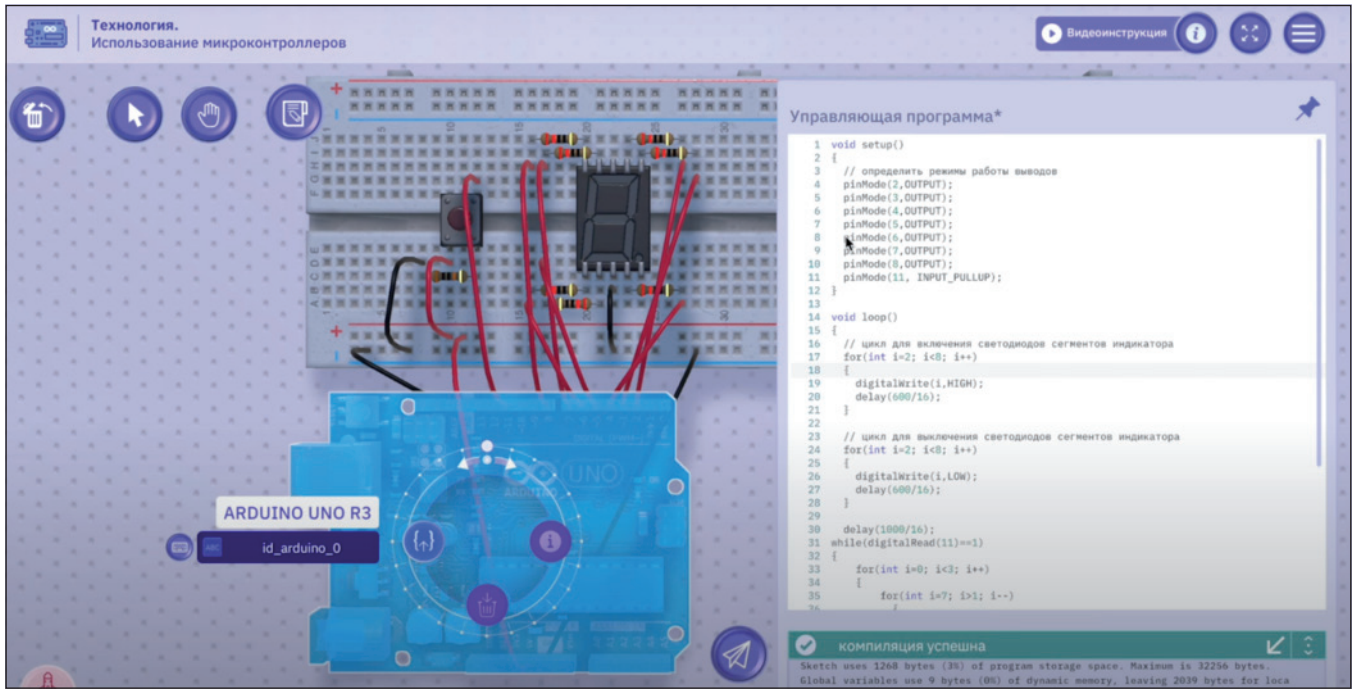


Рис. 4. Реализация пешеходного светофора с семисегментным индикатором

светофора и кнопки (рис. 3) и системы из пешеходного светофора с семисегментным индикатором (рис. 4).

4.2. Основы моделирования и прототипирования

В модуле «Основы моделирования и прототипирования» первая тема — **«Основы черчения»** — включает в себя знакомство с основами черчения и направлена на формирование базовых навыков черчения. Для отработки полученных навыков обучающимся предлагается использовать виртуальную лабораторию «Технология. Черчение».

В качестве диагностики задания «ИТ-марафона» включают в себя отработку полученных навыков и предполагают построение заданного плоского контура, нанесение размеров и заполнение основной надписи на чертеже (рис. 5) в рамках первого этапа, а также построение трех видов изометрической проекции, нанесение размеров и заполнение основной надписи на чертеже (рис. 6) на втором этапе.

Вторая тема модуля «Основы моделирования и прототипирования» — **«Основы трехмерного моделирования»** — посвящена изучению построения трехмерных моделей.

Для диагностики освоения этой темы в заданиях третьего этапа «ИТ-марафона» по направлению «Моделирование и прототипирование» обучающимся предлагается построить трехмерную модель объекта по изометрической проекции. Для выполнения данного задания ученики используют виртуальную лабораторию «Технология. 3D-моделирование. Силаэдр» или другую систему автоматизированного проектирования (КОМПАС-3D, OpenScad, FreeCad, Blender) (рис. 7).

На четвертом этапе «ИТ-марафона» по направлению «Моделирование и прототипирование» обучающимся

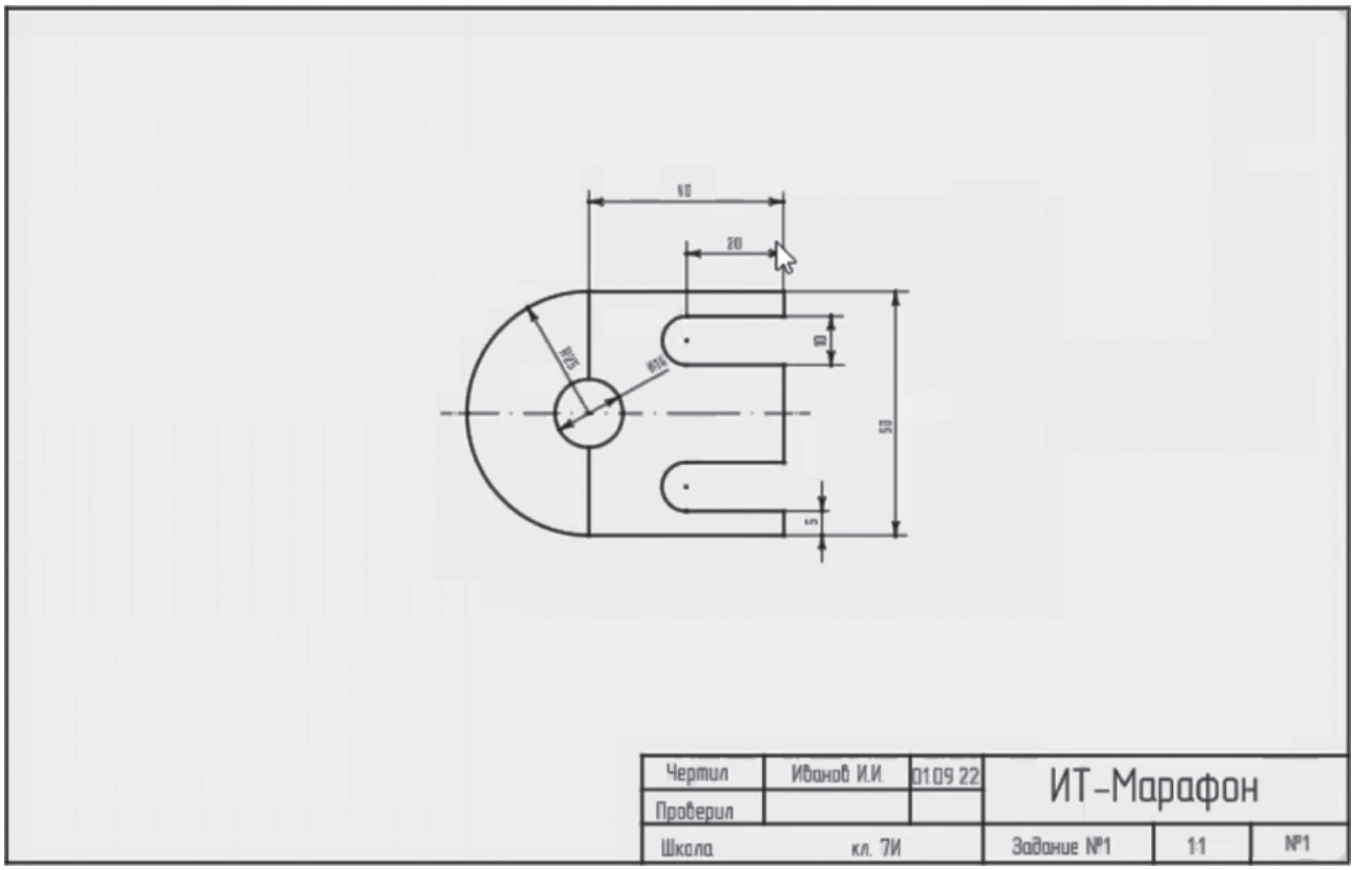


Рис. 5. Пример задания первого этапа «ИТ-марафона» по направлению «Моделирование и прототипирование»

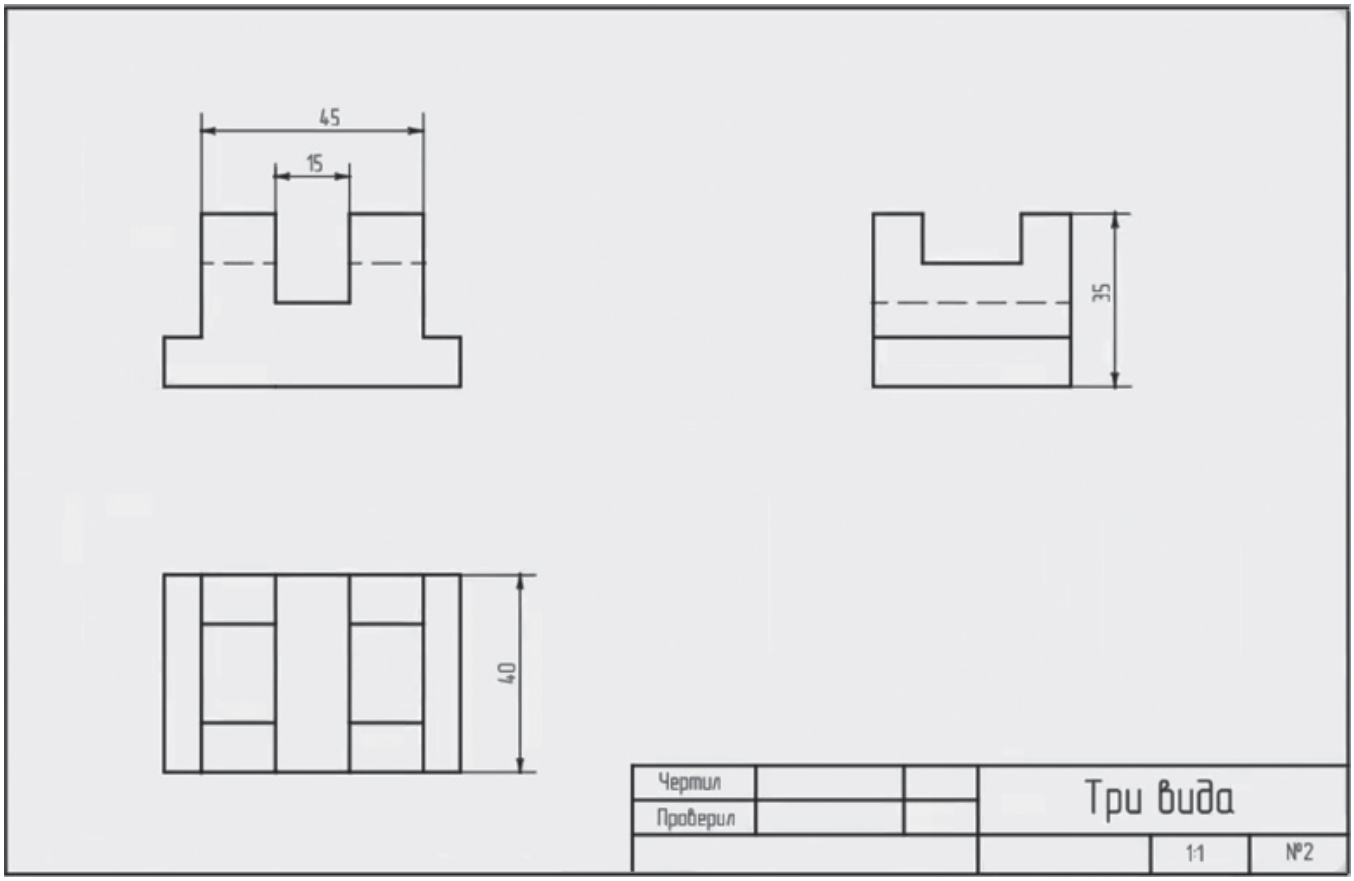
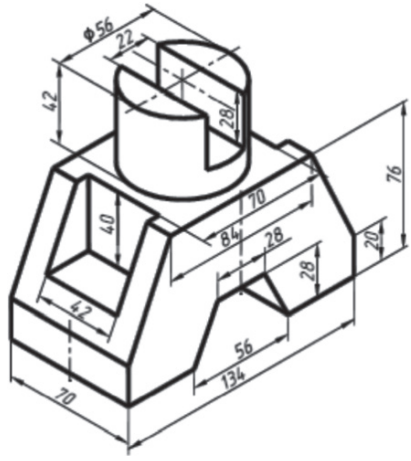
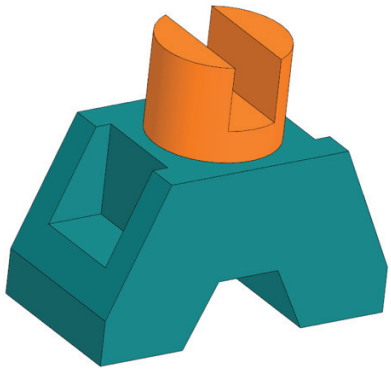


Рис. 6. Пример задания второго этапа «ИТ-марафона» по направлению «Моделирование и прототипирование»

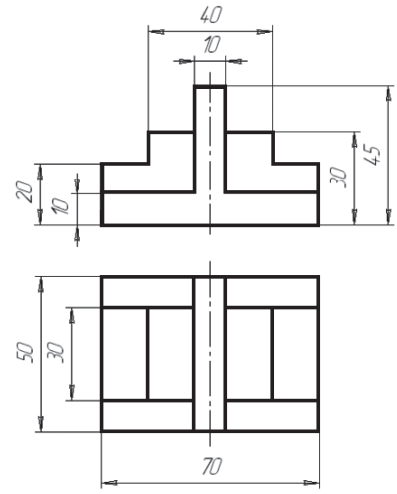


а

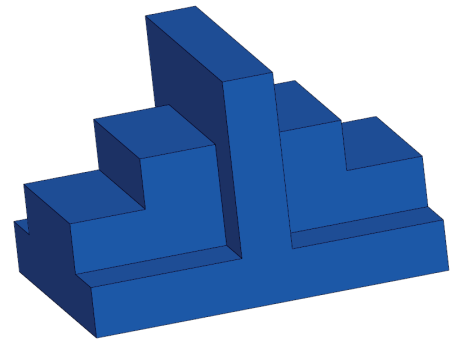


б

Рис. 7. Пример задания (а) и результата выполнения задания (б) третьего этапа «ИТ-марафона» по направлению «Моделирование и прототипирование»



а



б

Рис. 8. Пример задания (а) и результата выполнения задания (б) четвертого этапа «ИТ-марафона» по направлению «Моделирование и прототипирование»

предлагается построить трехмерную модель объекта по двум видам (рис. 8).

В теме **«Основы прототипирования»** модуля «Основы моделирования и прототипирования» школьники изучают технологии 3D-печати, виды пластиков, особенности работы с 3D-принтерами.

Тема **«Основы лазерных фрезерных технологий»** модуля «Основы моделирования и прототипирования» предполагает изучение основ лазерных и фрезерных технологий. Обучающиеся изучают особенности работы станков лазерной резки и станков ЧПУ, алгоритмы их калибровки и использования.

Изучение модуля «Основы моделирования и прототипирования» завершается темой **«Основы реверсивного инжиниринга»**. В результате изучения данной темы у обучающихся формируются навыки 3D-сканирования и обработки результатов сканирования.

4.3. Основы мехатроники и робототехники

Изучение модуля «Основы мехатроники и робототехники» начинается с тем **«Введение в программирование на С-подобных языках»**, **«Основы работы с исполнительными устройствами и датчиками»** и **«Алгоритмы управления робототехническими устройствами»**. Для изучения исполнительных устройств используется плата Arduino с комплектом двигателей и датчиков, а также джойстиком.

Четвертая тема модуля «Основы мехатроники и робототехники» — **«Базовые задачи мобильной робототехники»** — посвящена решению указанных задач. Обучающимся предлагается освоить алгоритмы движения вдоль линии с одним и двумя датчиками, движения вдоль стены, движения по лабиринту.

Использование технологий реверсивного инжиниринга позволяет на данном этапе опустить особенности программирования роботов и углубиться в изучение принципа работы исполнительных устройств. Для практической отработки навыков используется виртуальная лаборатория МЭШ «Технология. Моделирование роботов с использованием механических элементов».

Для диагностики освоения указанных выше тем используются задания «ИТ-марафона». Все этапы «ИТ-марафона» по направлению «Робототехника» реализуются в рамках работы с мобильными роботами.

Умения по сборке мобильного робота, а также по написанию программного кода для прохождения роботом испытательного полигона отрабатываются на первом этапе «ИТ-марафона» по направлению «Робототехника» (рис. 9).

На втором этапе участникам предлагается сконструировать и запрограммировать мобильного робота с использованием датчика дистанции и сервопривода для решения задач с их применением (рис. 10).

Третий этап «ИТ-марафона» по направлению «Робототехника» ставит задачу освоения дистанционного

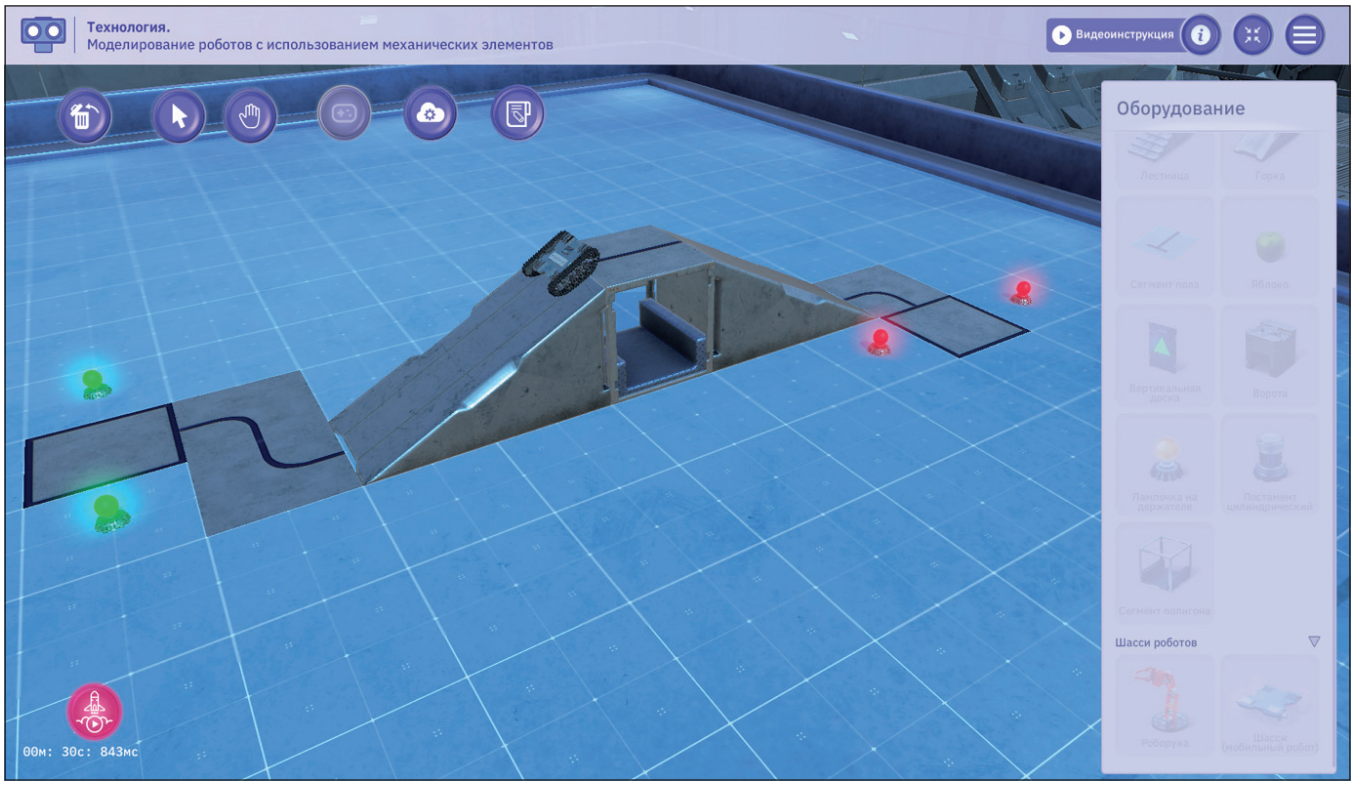


Рис. 9. Фрагмент выполнения задания первого этапа «ИТ-марафона» по направлению «Робототехника»

управления роботом и работы с инфракрасным приемником. Самостоятельно собранный и запрограммированный обучающимся робот с дистанционным управлением должен пройти заданную трассу (рис. 11).

Четвертый этап «ИТ-марафона» по направлению «Робототехника» предлагает обучающимся осуществить комбинированное управление роботом: необходимо реализовать управление мобильным роботом с самостоятельно сконструированным манипулятором с использованием ИК-приемника.

В теме **«Основы мехатроники и промышленной робототехники»** обучающимся предлагается сконстру-

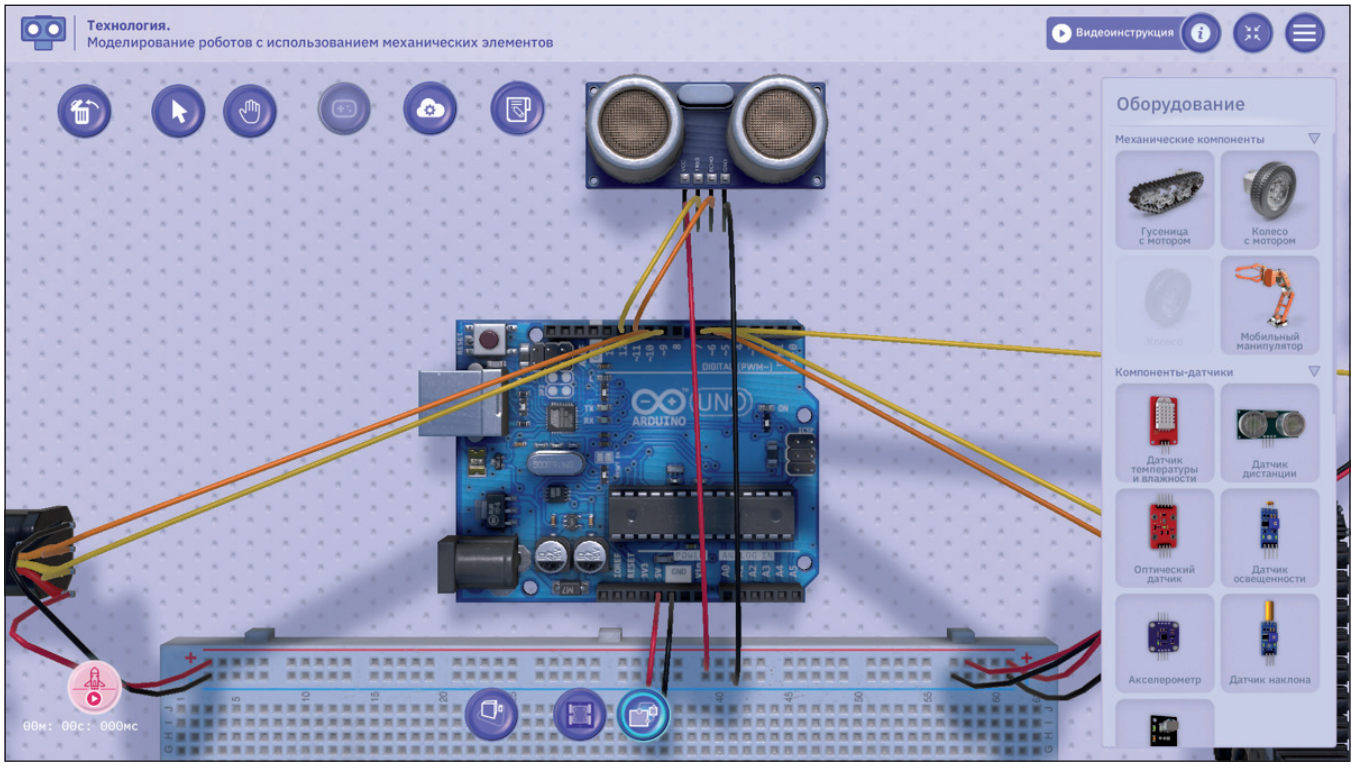


Рис. 10. Конструирование робота с датчиком дистанции

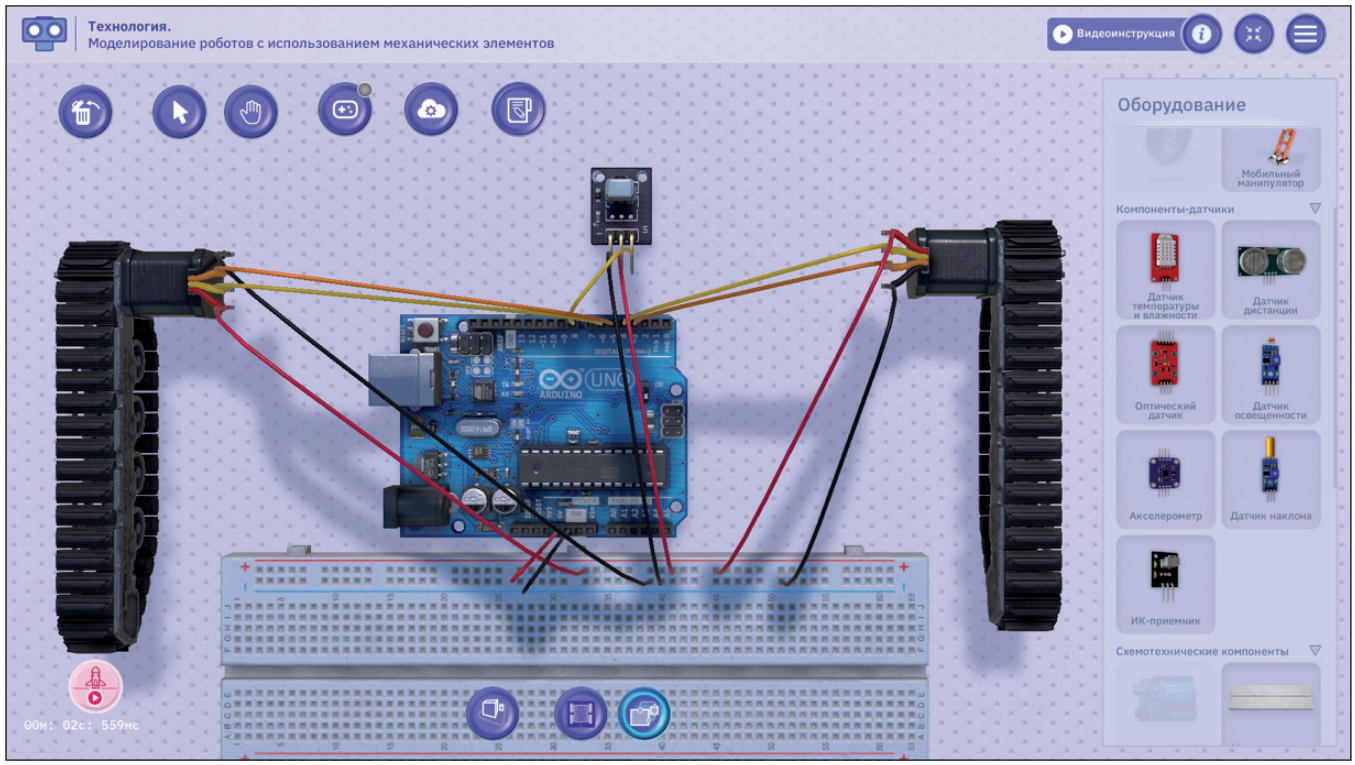


Рис. 11. Конструирование робота с ИК-приемником

ировать модели промышленных автоматизированных систем. Сборка моделей лифта, токарного станка, станка с ЧПУ, роборуки и промышленного робота позволяет школьникам изучить основы современной промышленной робототехники.

Изучение темы **«Основы пилотирования мобильных роботов и квадрокоптеров»** позволяет учащимся освоить управление различными роботизированными устройствами на основе знаний их конструктивных особенностей. Отдельный блок занятий посвящен управлению беспилотниками. Все полетные упражнения выполняются с использованием технологий FPV (First Person View), передающих оператору беспилотника окружающую устройство реальность с помощью технологий виртуальной реальности.

Особое место в курсе уделяется выполнению **творческих заданий на конструирование и программирование автоматических устройст**в. Обучающимся предлагается выполнить проект, в котором автоматическое устройство будет сконструировано и запрограммировано согласно техническому заданию. Реализация проектов в условиях ограниченных ресурсов развивает инженерные навыки учащихся.

5. Внеурочная деятельность

Внеурочная деятельность в рамках проекта является неотъемлемой частью образовательной программы и обеспечивает для школьника широту горизонта его понимания использования знаний и умений, полученных на обязательных учебных курсах и при освоении предметов, для решения задач в различных отраслях науки, техники, медицины, экономики и креативных индустрий [14].

Одна из таких задач была поставлена обучающимся на хакатоне «ИТ в Музее Победы»: решение этой задачи подразумевало создание продукта, который в связке с мобильным устройством является путеводителем по интерактивному историко-патриотическому маршруту образовательного квеста в Музее Победы.

Разработан рекомендованный перечень программ внеурочной деятельности с возможным вариантом распределения часов с учетом преемственности и возрастных особенностей школьников вертикали (табл. 2).

Выбор программ курсов внеурочной деятельности предлагает соединить знания и навыки обучающихся, полученные при изучении информатики, технологии и программирования, в практическом применении для решения прикладных задач.

Экосистема курсов внеурочной деятельности делает акценты на таких направлениях использования информационных технологий, как: информационная безопасность, операционные системы, технологии дополненной и виртуальной реальности, технологии связи, веб-программирование, веб-дизайн, разработка мобильных приложений.

Реализация программ внеурочной деятельности предполагает **использование особых форм организации работы школьников, специфичных для ИТ-отрасли:**

- хакатоны по созданию ИТ-решений (мобильных приложений, объектов AR/VR, сайтов);
- турниры по информационной безопасности, имитирующие возникновение и ликвидацию киберугроз (CTF);
- чемпионаты профессионального мастерства по настройке и защите сетевой инфраструктуры и передаче данных.

Таблица 2

Рекомендованный перечень программ внеурочной деятельности

№ п/п	Программа внеурочной деятельности	Классы		
		VII	VIII	IX
1	Мир информационной безопасности CTF*	–	+	+
2	Инструменты компьютерной математики	+	+	+
3	Разработка мобильных приложений	–	+	+
4	Веб-дизайн	+	–	–
5	Цифровые двойники AR/VR	–	+	–
6	Спортивное программирование	+	+	+
7	Спортивная робототехника	+	+	+
8	Технологии связи	+	–	–
9	Операционные системы: первые шаги	–	–	+

* CTF (Capture The Flag — «захват флага») — индивидуальные или командные соревнования по практико-ориентированной информационной безопасности. Команды решают прикладные задачи в области кибербезопасности, чтобы найти уязвимость (или решить какую-то другую прикладную задачу) и получить уникальную комбинацию символов (флаг).

6. Инфраструктура проекта «ИТ-вертикаль»

Реализация проекта требует создания системы необходимых и достаточных условий. Созданная инфраструктура проекта «ИТ-вертикаль» представлена следующими **ключевыми компонентами**, которые тесно взаимосвязаны друг с другом и динамично откликаются на изменения в быстро меняющейся ИТ-отрасли:

- методическое сопровождение образовательной траектории;
- материально-техническая база проекта;
- программные продукты для решения образовательных задач;
- кадровое обеспечение образовательной траектории.

6.1. Методическое сопровождение образовательной траектории

Информационно-методическое сопровождение образовательной траектории осуществляется проектным офисом проекта «ИТ-вертикаль». Структура и особенности реализации проекта закреплены в Стандарте проекта «ИТ-вертикаль» [9].

Достижение результатов предпрофильного образования с использованием инструментария расширенного содержания становится возможным благодаря тренингам и курсам повышения квалификации, проводимым оператором проекта «ИТ-вертикаль». В курсах повышения квалификации особое внимание уделяется особенностям содержания и процесса обучения в каждом профильном предмете и в специальном курсе [5].

Проектным офисом и партнерами проекта разработаны и продолжают разрабатываться учебные и методические материалы для педагогов и школьников, в том числе материалы для Библиотеки МЭШ.

6.2. Материально-техническая база проекта

Реализация проекта предполагает наличие материально-технической базы в образовательной организации.

Материально-техническая база обеспечивает возможность реализации практико-ориентированного обучения на высокотехнологичном оборудовании в групповом и индивидуальном режимах благодаря широкой линейке учебного оборудования специальных лабораторий по направлениям: прототипирование, электроника и схемотехника, мехатроника и робототехника.

При формировании материально-технической базы следует уделить внимание месту использования комплектов оборудования. Так, **в модулях «Основы мехатроники и робототехники» и «Основы микропроцессорной техники» можно выделить два вида комплектов используемого оборудования**.

Учебно-лабораторное оборудование предназначено для проведения практических экспериментов, включает в себя методическое обеспечение содержания и проведения экспериментов. Данный вид оборудования подходит, например, для изучения логических элементов в теме «Цифровая электроника», однако не позволяет организовать творческую деятельность обучающихся в связи с ограниченными возможности конструирования.

Для развития творческих способностей обучающихся и организации проектной деятельности по темам «Основы мехатроники и робототехники» и «Основы микропроцессорной техники» используются ***образовательные наборы***, включающие в себя большое количество компонентов для сборки и проектирования и не ограничивающие обучающихся возможностями лабораторных платформ. Важной составляющей наборов является большой спектр электронных компонентов

с различными характеристиками: диоды, конденсаторы, резисторы, транзисторы, датчики, двигатели и т. д., программируемые микроконтроллеры, а также макетные платы и макетные провода для отладки собранных устройств.

Материально-техническое обеспечение направления «Основы моделирования и прототипирования» включает в себя комплект 3D-принтеров (работающих на технологиях 3D-печати FMD и DLP), станок лазерной резки, фрезерный станок с ЧПУ и 3D-сканер.

6.3. Программные продукты для решения образовательных задач

В качестве дополнения к материально-технической базе разработаны программные продукты для обеспечения решения образовательных задач каждым школьником.

В числе программных продуктов — уникальная **экосистема виртуальных лабораторий** по информатике, технологии и физике [3].

Виртуальная лаборатория по информатике включает в себя методические материалы по изучению языков программирования с поддержкой программирования в самой виртуальной лаборатории. В лаборатории также содержатся материалы для подготовки к экзаменам и олимпиадам по программированию: демонстрационные материалы, теоретическое объяснение материала.

Комплекс виртуальных лабораторий по технологии включает в себя лаборатории по робототехнике, черчению, 3D-моделированию, цифровой электронике, программированию микроконтроллеров.

Шесть виртуальных лабораторий по физике позволяют обучающимся выполнить эксперименты по всем разделам школьного курса физики.

6.4. Кадровое обеспечение образовательной траектории

Кадровый состав педагогов проекта формируется из учителей профильных предметов, которые:

- имеют высокий или экспертный уровень владения предметной областью, который подтверждается диагностиками Московского центра качества образования;
- обладают специальными знаниями по ИТ-направлениям и навыками использования материально-технической базы проекта и программных продуктов для организации различных видов учебной деятельности; эти знания и навыки подтверждаются **сертификацией**, которая представляет собой специальную процедуру выявления готовности педагогов к реализации образовательной траектории проекта «ИТ-вертикаль».

Для успешной реализации образовательной траектории педагогами образовательных организаций проектным офисом была разработана и апробирована **система диагностики специальных ИТ-компетенций** учителей информатики, технологии, физики и математики.

Задачей сертификации являются не только необходимость произвести диагностику знаний педагогов, но и демонстрация направлений развития педагогических навыков для успешной реализации проекта.

Сертификация проводится в два этапа.

Первый — теоретический — этап сертификации направлен на оценку:

- профессиональных компетенций педагогических работников в области организации проектной и исследовательской деятельности на основе знаний и умений в предпрофильном образовании;
- профессиональных компетенций владения инструментами МЭШ, необходимых для реализации городских образовательных проектов.

Второй — практический — этап сертификации направлен на оценку:

- специальных профессиональных компетенций педагогических работников;
- навыков работы с оборудованием в соответствии с направленностью проекта.

Учитель физики, работающий в классах проекта «ИТ-вертикаль», *должен обладать следующими компетенциями:*

- владеть предметной областью;
- владеть приемами работы с цифровыми лабораториями;
- владеть инструментами цифровой дидактики;
- владеть средствами ИКТ;
- владеть приемами обработки и визуализации данных.

Практический этап сертификации учителя физики включает в себя пять заданий для проверки описанных компетенций. Каждое задание подразумевает выполнение физического эксперимента и обработку его результатов.

Три задания разработаны для выполнения в виртуальных лабораториях МЭШ. Помимо виртуальных лабораторий по каждому из разделов физики может быть представлено задание в виртуальной лаборатории «Обработка результатов эксперимента».

Два задания выполняются с использованием реального оборудования и цифровых лабораторий.

Учитель математики, работающий в классах проекта «ИТ-вертикаль», *должен обладать такими компетенциями, как:*

- владение предметной областью;
- владение инструментами цифровой дидактики;
- владение элементами компьютерной алгебры;
- владение средствами ИКТ;
- владение приемами обработки и визуализации данных.

Практический этап сертификации учителя математики включает в себя пять заданий с использованием различных виртуальных лабораторий, табличных процессоров и средств компьютерной алгебры.

Виртуальные лаборатории, используемые учителями математики: «Построение логических схем», «Графики функций», «Математический конструктор».

Отдельно выделены задания с табличными процессорами. В качестве таких заданий предполагается выполнение статистической обработки массива данных, определение распределения данных и вычисление других параметров массива. Работа с табличными процессорами подразумевает построение графиков функций и их обработку.

Практический этап сертификации **учителя информатики** включает в себя выполнение 10 заданий, по которым определяется *наличие следующих компетенций:*

- умение читать, составлять и редактировать программные коды на языках программирования Python и C++;
- умение определять и реализовывать стратегии безопасной работы с информационными ресурсами и программным обеспечением;
- умение использовать средства обучения и воспитания для решения задач в областях робототехники, микропроцессорной техники, моделирования и прототипирования, технологий связи;
- умение использовать цифровые дидактические инструменты, в том числе виртуальные лаборатории, для решения задач в областях робототехники, микропроцессорной техники, моделирования и прототипирования, технологий связи;
- умение реализовывать математические методы обработки данных с использованием программных продуктов и языков программирования;
- умение работать в различных операционных системах: Windows, LINUX, Android.

Учитель технологии, преподающий в проекте «ИТ-вертикаль», *должен обладать следующими компетенциями:*

- высокий уровень технической грамотности;
- умение выполнять чертежи и создавать модели в CAD-системах;
- умение использовать средства обучения и воспитания для решения задач, относящихся к современным технологиям производства, — пайка, 3D-печать, лазерная резка и др.;
- умение читать принципиальные электрические схемы, а также производить сборку и настройку электронных аналоговых и цифровых устройств по представленной схеме;
- умение читать, анализировать, создавать и редактировать программные коды, написанные на блочных и С-подобных языках программирования;
- умение использовать средства обучения и воспитания для решения задач в областях робототехники, микропроцессорной техники, моделирования и прототипирования, технологий связи;
- умение использовать цифровые дидактические инструменты, в том числе виртуальные лаборатории, для решения задач в областях робототехники, микропроцессорной техники, моделирования и прототипирования, технологий связи;
- владение навыками пилотирования различных устройств и управления работой манипулятора.

Практический этап сертификации учителя технологии включает в себя выполнение 10 заданий. Эти задания затрагивают все направления предмета «Технология», которые обучающиеся проходят за весь период обучения. Предлагаются как задания для выполнения на реальном оборудовании, так и задания в виртуальных лабораториях.

Ознакомиться с **демонстрационными вариантами оценочных средств сертификации** педагогических работников проектов «ИТ-вертикаль» и «ИТ-класс в московской школе» можно на сайте ГАОУ ДПО «Корпоративный университет» [12].

7. Заключение

В заключение отметим, что проект «ИТ-вертикаль» реализуется в московских школах с 2022 года. В настоящее время 125 образовательных организаций реализуют проект в VII и VIII классах, и 51 образовательная организация начала реализацию проекта в VII классах в 2023/2024 учебном году. Общий охват школьников в проекте составляет более 7000 человек.

Список источников

1. *Босова Л. Л.* Информатика и обновленный ФГОС: ключевые изменения, требования, возможности // Актуальные проблемы методики обучения информатике и математике в современной школе. Материалы международной научно-практической интернет-конференции (Москва, 18–24 апреля 2022 года). М.: МПГУ, 2022. С. 10–17. EDN: OLHUYH.

2. *Босова Л. Л., Павлов Д. И., Ткач Т. В., Бутарев К. В.* О содержании «Базового курса информатики» в проекте «ИТ-класс в московской школе» // Информатика в школе. 2021. № 10. С. 9–18. EDN: FVCHQS. DOI: 10.32517/2221-1993-2021-20-10-9-18

3. Виртуальные лаборатории // Лаборатории предпрофессионального образования. https://labpredprof.ru/vl/

4. Инженерный класс в московской школе. О проекте // Портал «Городские проекты». https://profil.mos.ru/inj/o-proekte.html

5. ИТ-вертикаль // Портал «Городские проекты». https://profil.mos.ru/it-vert/

6. ИТ-класс в московской школе. О проекте // Портал «Городские проекты». https://profil.mos.ru/it/o-proekte.html

7. *Мырадов М. В., Павлов Д. И.* Особенности подготовки к олимпиадам школьников по соревновательной робототехнике // Актуальные проблемы обучения математике в школе и вузе: от науки к практике. К 80-летию со дня рождения В. А. Гусева. Материалы VII Международной научно-практической конференции (Москва, 18–19 ноября 2022 года). М.: МПГУ, 2022. С. 615–622. EDN: MMCERV.

8. *Павлов Д. И.* Изменения в реализации курса информатики на уровне основного общего образования в свете введения новой редакции ФГОС ООО // Актуальные проблемы методики обучения информатике и математике в современной школе. Материалы международной научно-практической интернет-конференции (Москва, 18–24 апреля 2022 года). М.: МПГУ, 2022. С. 240–260. EDN: WHVFUD.

9. Приказ Департамента образования и науки города Москвы от 04.03.2022 № 120 «Об утверждении Стандарта городского образовательного проекта “IT-вертикаль” в государственных образовательных организациях, подведомственных Департаменту образования и науки города Москвы». https://www.mos.ru/donm/documents/normativnye-pravovye-akty/view/266007220/?ysclid=lq4cw0dlqe936307074

10. Приказ Министерства просвещения Российской Федерации от 18.05.2023 № 370 «Об утверждении федеральной образовательной программы основного общего образования». http://publication.pravo.gov.ru/document/0001202307140040?ysclid=lq4cqwbije65863942

11. Профориентационный минимум (единая модель профориентационной деятельности) // Минпросвещения России. https://edu.gov.ru/career_guidance?ysclid=lq4fzcrx3o747993400

12. Сертификация педагогических работников в рамках реализации городских образовательных проектов «IT-вертикаль», «ИТ-класс в московской школе» // Корпоративный

университет московского образования. https://corp-univer.ru/center-assessment-and-certification/certification/it_vertical/it/#1676554075014-b7c9015b-7cb1

13. Федеральная рабочая программа основного общего образования. Информатика (углублённый уровень) (для 7–9 классов образовательных организаций) // Единое содержание общего образования. https://edsoo.ru/wp-content/uploads/2023/08/16_ФРП_Информатика_7-9-классы_угл.pdf

14. Федеральная рабочая программа основного общего образования. Технология (для 5–9 классов образовательных организаций) // Единое содержание общего образования. https://edsoo.ru/wp-content/uploads/2023/08/29_ФРП-_Технология_5-9-классы.pdf

15. Федеральная рабочая программа основного общего образования. Физика (углублённый уровень) (для 7–9 классов образовательных организаций) // Единое содержание общего образования. https://edsoo.ru/wp-content/uploads/2023/08/21_ФРП_Физика_7-9-классы_угл.pdf

16. IT-марафон // Лаборатории предпрофессионального образования. https://labpredprof.ru/ITmarafon/

DOI: 10.32517/2221-1993-2023-22-6-18-30

Е. А. Еремин
Пермский государственный гуманитарно-педагогический университет, г. Пермь, Россия

ТРИ ЭТЮДА ПО КОМПИЛЯЦИИ, ИЛИ КАК ИСПОЛЬЗОВАТЬ ЯЗЫК JULIA ДЛЯ ИЗУЧЕНИЯ ГЕНЕРИРУЕМОГО МАШИННОГО КОДА

Аннотация

Статья демонстрирует интересную для образования возможность языка программирования Julia выводить ассемблерный текст для любой написанной на Julia функции. Подробно рассмотрен исполняемый код для трех наглядных примеров: вычисление по формуле, разветвляющийся и циклический фрагменты. Показано, что компилятор Julia действительно генерирует тщательно оптимизированную программу, стараясь использовать выполняемые максимально быстро инструкции процессора Intel. Он успешно пытается обойтись без трудоемких арифметических операций (если удается, то значения вычисляются заранее, умножение заменяется сдвигом и т. п.). Особенно впечатляют результаты для целочисленных данных. Удается увидеть целый ряд оптимизационных приемов, в том числе учитывающих особенности самого алгоритма: например, компилятор способен предсказать результаты простейшего цикла и исключить этот цикл из итоговой программы. Обсуждается также проблема контроля переполнения в программе.

Проведенное рассмотрение убедительно подтверждает полезность теоретических знаний для разработки эффективно работающей программы. Описанная методика анализа представляет интерес для образования, поскольку важность глубокого понимания современными специалистами сути процесса программирования трудно переоценить. Материал может быть полезен при обучении идеям программирования на разных уровнях. Детальность изложения делает статью доступной для чтения без специальной подготовки.

Ключевые слова: программирование, язык Julia, компиляция, оптимизация, эффективность, исполняемый код, машинный код, обучение программированию, методики обучения.

Контактная информация

Еремин Евгений Александрович, канд. физ.-мат. наук, доцент, доцент факультета информатики и экономики, Пермский государственный гуманитарно-педагогический университет, г. Пермь, Россия; *адрес:* 614990, Россия, г. Пермь, ул. Сибирская, д. 24; *e-mail:* evg_eremin@mail.ru

E. A. Eremin
Perm State Humanitarian Pedagogical University, Perm, Russia

THREE ETUDES ON COMPILING, OR HOW TO USE THE JULIA LANGUAGE TO STUDY GENERATED MACHINE CODE

Abstract

The article demonstrates the ability of the Julia programming language to output an assembly text for any function written in Julia, which is interesting for education. The executable code for three illustrative examples is examined in detail: calculation by formula, branching and cyclic fragments. It is shown that Julia compiler really generates a carefully optimized program, trying to use the fastest executing instructions of the Intel processor. It successfully tries to do without time-consuming arithmetic operations (if possible, the values are calculated in advance, multiplication is replaced by shift and so on). The results for integer data are especially impressive. It is possible to see a whole range of optimization techniques, including those that take into account the features of the algorithm itself: for example, the compiler is able to predict the results of the simplest loop and exclude this loop from the final program. The problem of overflow control in a program is also discussed.

The consideration carried out convincingly confirms the usefulness of theoretical knowledge for developing an effectively working program. The described analysis technique is interesting for education, because the importance of a deep understanding by modern specialists of the programming process essence cannot be overestimated. The material can be useful in teaching programming ideas at different levels. The detailed presentation makes the article readable without special preparation.

Keywords: programming, Julia language, compilation, optimization, efficiency, executable code, machine code, programming training, teaching methods.

> И только потом я понял, что отличает Julia от других языков. Julia разрушает барьер между высокоуровневым и ассемблерным кодом. Julia не просто позволяет писать код, который работает так же быстро, как код на C, но и дает возможность посмотреть на LLVM-представление функций и их сгенерированный ассемблерный код. И все это прямо в интерактивной среде.
>
> *Э. Миллер [35]*

1. Julia — новый язык программирования

Язык программирования Julia [39] появился относительно недавно: он стал общедоступным в феврале 2012 года [20]. Его создатели [25, 26] из Массачусетского технологического института (Massachusetts Institute of Technology — MIT) первоначально планировали применить язык для целей научного программирования (в первую очередь как быстродействующую замену пакета MATLAB). Однако результат превзошел ожидания создателей: родился не просто эффективный язык для вычислений и построения графиков, но общецелевой и многоплатформенный язык программирования. Автор недавно вышедшей книги [22] Э. Энгхейм, будучи профессиональным разработчиком ПО, написал об этом так: «Для того чтобы использовать Julia, вовсе не нужно быть гением или ученым. Julia — замечательный язык общецелевого программирования для всех! Я не ученый, и мне нравится использовать его уже более девяти лет. Вы обнаружите, что с Julia вы сможете решать задачи быстрее и элегантнее, чем раньше. И вишенкой на торте является то, что вычислительно емкий исходный код будет исполняться невероятно быстро. <...> С тех пор я испробовал много других языков программирования, но так и не приблизился к тому, чего добился на языке Julia. <...> Язык Julia помог мне вернуть радость от программирования» [22], Немалое число программистов разделяют такую эмоциональную симпатию к языку. Сошлемся, в частности, на результаты опроса 2021 года, в котором приняли участие свыше 83 тысяч разработчиков из 181 страны мира. Чтобы определить «самый любимый» язык, специалисты известного сайта Stack Overflow спросили у разработчиков, какой язык они использовали в прошлом году и на каком языке они хотели бы писать в следующем. Язык Julia по результатам опроса занял почетное пятое место (данные взяты из [10]).

Автор убедительно просит читателей не считать вводный абзац восхвалением нового «суперъязыка» программирования! Хотя и во многом согласен с приведенной выше оценкой.

Знакомство с литературой на английском языке подтверждает внимание мировой аудитории к языку. Приведем несколько примеров книг по Julia [36–38], в названиях которых обращает на себя внимание упоминание наиболее передовых областей современной информатики. Из публикаций на русском языке хотелось бы отметить статью обзорного характера [3], где помимо общей характеристики языка особое внимание уделено возможностям уже написанных математических библиотек. Подчеркнем, что, согласно базовым идеям создателей Julia [25], язык должен работать настолько быстро, чтобы позволять писать все библиотеки непосредственно на нем, а не подключать результаты трансляции с другого языка, например, с C или FORTRAN.

В работах [14, 19] дается характеристика нового языка, причем в статье [14] приводятся примеры решения на Julia простых практических задач. Статьи [11, 31] посвящены применению Julia в математике, а [4, 17, 30] демонстрируют полезность языка в других областях (от машинного обучения до биологии).

Особо хотелось бы отметить, что наши ведущие вузы уже обучают своих студентов языку Julia. И, что еще приятнее, делятся в Сети своими учебно-методическими пособиями по курсам данной тематики (см., например, [1, 2, 21]).

Таким образом, по имеющимся источникам можно сделать вывод, что язык Julia эффективно применяется в самых разнообразных областях, в том числе в образовании. Об особенностях и достоинствах языка можно написать отдельную статью. Но в данной публикации речь пойдет только об одном аспекте Julia — о возможности очень легко посмотреть, как выглядит машинный код для несложных фрагментов на языке программирования высокого уровня. По мнению автора, изучение данного вопроса полезно по нескольким причинам.

Во-первых, можно на собственном опыте проверить, действительно ли Julia генерирует эффективный код, как это везде утверждается.

Во-вторых, некоторое представление о том, как выглядит одна и та же программа для человека и для компьютера, весьма полезно и должно быть хотя бы раз продемонстрировано при обучении. Julia позволяет обойтись без вспомогательного программного обеспечения, которое приходилось для этих целей специально разрабатывать [6, 29].

Примечание. Возможно, данный тезис требует обоснования. Сошлемся как на образец на систематический курс [16], показывающий тесную взаимосвязь всех уровней работы компьютера. А в известных статьях [23, 28] утверждается, что необходимо рассматривать материал минимум на один уровень глубже, чем это требуется в данном учебном курсе.

Наконец, в-третьих, изучать поставленную проблему средствами Julia можно без особой предварительной подготовки, так что автор не может не поделиться с окружающими своим мнением о том, насколько это легко и приятно.

Завершая вводную часть, поясним, что многозначное слово ***этюд*** (от *франц.* étude, буквально — изучение, упражнение) в статье понимается следующим образом. Во-первых, как «*небольшое по объему произведение (научное, критическое), посвященное частному вопросу*» [13]. Во-вторых, как «*упражнение, служащее для развития и совершенствования техники какого-либо искусства*» [8]. И, наконец, в-третьих (что, возможно, не очень скромно), известны случаи, когда удачный этюд представляет самостоятельный интерес.

2. Как Julia оптимизирует программу

Одна из целей, поставленных при разработке языка Julia, заключалась в генерации эффективного машинного кода в момент выполнения программы. Для решения поставленной задачи была предложена следующая схема

трансляции [24], в ходе которой оптимизация возможна на нескольких этапах.

Сначала *исходный текст* на Julia анализируется и превращается в *абстрактные синтаксические деревья* (Abstract Syntax Tree — AST). Последние далее преобразуются в некоторое *внутреннее представление*, что позволяет проводить оптимизацию на уровне языка Julia.

Полученный результат транслируется в некоторый *промежуточный код* под названием LLVM IR [33, 34]. Расшифровывается это сочетание букв так. Речь идет о промежуточном представлении (**I**ntermediate **R**epresentation; первоначально также использовался термин *битовый код* [12], чтобы не путать с байткодом Java). Код предназначен для LLVM; сначала это называлось виртуальной машиной низкого уровня — **L**ow **L**evel **V**irtual **M**achine, но позднее от идеи виртуальной машины проект отказался, оставив только старое сокращенное название. На уровне LLVM IR также производится оптимизация, уже не зависящая от языка высокого уровня. Заметим, что текстовое представление LLVM-кода Julia может показать при вызове функции **code_llvm()**.

Наконец, на завершающем этапе LLVM IR стандартным образом преобразуется в *машинный код* конкретного процессора.

В Julia компиляция выполняется при вызове функции, причем результат запоминается и повторная компиляция уже не требуется.

Познакомившись в общих чертах с идеями компиляции, перейдем к экспериментам. Дальнейшее изложение предполагает, что язык Julia уже установлен на вашем компьютере. С этим не должно возникать проблем, поскольку установка среды и базовые принципы общения с ней неоднократно описаны (см., например, [1]). В ОС Windows (32-битная версия) автору даже не потребовалось проводить инсталляцию — оказалось достаточно разархивировать скачанный с официального сайта **julialang.org** дистрибутив (версия 1.9.2) и в традиционном каталоге **bin** найти исполняемый файл **julia.exe**. Для проведения описанных в данной публикации экспериментов можно больше никакое дополнительное программное обеспечение не использовать.

Итак, полагаем, что вы запустили указанный выше файл и получили приглашение к диалогу в виде текста «**julia>**» в начале строки. Можно набирать команду! Что именно вводить, описывается далее в этюдах 1–3. Отметим, что количество этюдов выбрано не случайно: существуют три базовые алгоритмические структуры (следование, ветвление (развилка), цикл), и каждую из них мы опробуем.

3. Этюд 1. Оптимальное вычисление арифметических выражений

Рассмотрим для начала, как компилятор Julia оптимизирует вычислительные формулы. При анализе нам могут потребоваться некоторые дополнительные сведения, которые, чтобы не отвлекаться от основного изложения, имеет смысл обсудить предварительно. Подготовленные читатели могут пропустить знакомый материал.

3.1. Этюд 1: предварительные сведения

3.1.1. Функция code_native()

Для просмотра машиной программы в Julia предусмотрена замечательная функция code_native(), которая выводит на экран текст исполняемого («родного») кода процессора. При обращении надо всегда указывать имя интересующей нас функции и тип аргумента (аргументов), для которого следует построить ее код. А далее в качестве последнего параметра приходится явно задавать синтаксис ассемблера: хочется видеть общепринятый синтаксис Intel, а тестируемая версия Julia по умолчанию устанавливает ассемблер AT&T (American Telephone and Telegraph; компания известна своей связью с разработкой ОС Unix).

Примечание. Между прочим, такой же выбор синтаксиса необходимо делать и при использовании встроенного ассемблера в среде Free Pascal.

Приведем пример. Пусть у нас имеется функция одной переменной $f(x)$ и мы хотим посмотреть ее код для аргумента типа Int32 (можно задать и другой, в том числе вещественный). Тогда необходимо ввести следующую команду:

```
code_native(f, (Int32,); syntax=:intel)
```

Поясним дополнительно, что у функции *f* мог быть не один аргумент; тогда второй параметр code_native() должен вмещать в себя несколько значений (такой набор значений называют *кортежем*). Запись (Int32,), которая выглядит несколько непривычно, как раз и описывает кортеж из одного значения.

3.1.2. Некоторые необходимые для анализа команды

Каждая программа для компьютера — это последовательность машинных инструкций (команд), которая представляется в двоичном (*машинном*) коде. Именно в таком виде машина, а точнее, ее процессор, выполняет программу. Чтобы облегчить программирование на таком низком уровне, используется специальный язык — *ассемблер*, где для удобства человека инструкции записываются в текстовой форме.

Любая инструкция ассемблера состоит из кода операции (его часто называют *мнемоникой*), который показывает, какое действие требуется выполнить, а также описания, где взять исходные данные (*операнды*) и куда поместить результат.

Рассмотрим несколько примеров, которые пригодятся нам при изучении кода этюда 1.

Пример 1.

```
mov ebp, esp
```

Приведенная инструкция **копирует** (переносит — move) содержимое из регистра **esp** в регистр **ebp** (по сути, это низкоуровневый эквивалент оператора присвоения **ebp** = **esp**).

Если операнд заключен в квадратные скобки, то он является **адресом памяти**, откуда необходимо прочитать данные.

Пример 2.

```
mov eax, dword ptr [ebp + 8]
```

Здесь в регистр **eax** копируется число из памяти по адресу, равному содержимому регистра **ebp** плюс 8. Запись задает только начальный адрес памяти, и по нему невозможно догадаться, сколько байт надо прочитать. Поэтому в команду приходится добавлять текст ***dword ptr***, который задает размер данных. Сокращение **dword** (double word — двойное слово) указывает на четырехбайтовые данные, потому что обычное слово (word) занимает два байта.

Пример 3.

```
shl eax, 5
```

А эта инструкция сдвигает (sh — начало слова shift) влево (буква «l» в конце мнемоники) на 5 двоичных разрядов содержимое **eax**. Результат заносится в тот же самый регистр. В итоге получается результат, эквивалентный умножению на $2^5 = 32$ (см. п. 3.1.4 ниже).

Пример 4.

```
or eax, 1
```

Данная инструкция выполняет логическую операцию ИЛИ (**or**) между содержимым регистра **eax** и константой 1. Правила этой операции хорошо известны. Отметим здесь лишь то, что младший бит регистра в результате операции всегда будет установлен в единицу, в то время как все остальные останутся без изменения.

3.1.3. Команды SSE

Технология *SSE* (***S**treaming **S**IMD **E**xtensions* — потоковое SIMD-расширение процессора; SIMD — **S**ingle **I**nstruction, **M**ultiple **D**ata, т. е. одна инструкция — множество данных) впервые была представлена в процессорах Pentium III. Ее основное назначение состоит в организации параллельного выполнения операций над несколькими вещественными числами [18]. В процессор добавлен набор специализированных регистров, которые обозначаются **xmm**. Каждый регистр имеет емкость 128 бит, что одновременно позволяет обрабатывать четыре 32-битных значения с плавающей запятой одинарной точности (*single*) или два 64-битных значения двойной точности (*double*). Параллельно обрабатываемые данные называются *упакованными* (packed, к мнемонике команды добавляется буква «p»). Допускается действие только над одним вещественным числом — такая инструкция называется *скалярной* (scalar, добавляется «s»). В зависимости от того, какая точность чисел используется — одинарная или двойная, — к обозначению команды приписывается еще одна буква — «s» или «d» соответственно. Например, сочетание «sd» говорит о скалярной обработке одного вещественного числа двойной точности.

При изучении машинного кода этюда 1 нам встретятся следующие **команды SSE:**

- `movss xmm0, dword ptr [ebp + 8]`
  Эта инструкция загружает одно скалярное число одинарной точности в регистр **xmm0**. Исходное число берется из памяти, о чем говорят квадратные скобки в записи команды, из адреса, вычисляемого как содержимое регистра **ebp** плюс 8. Запись ***dword ptr***, как мы уже знаем из раздела 3.1.2, указывает в явном виде, что размер числа равен 4 байтам.
- `addss xmm0, dword ptr [.LCPI0_1]`
  Здесь к такому же по точности числу в **xmm0** прибавляется (**add**) из памяти «техническая» константа с именем **LCPI0_1**. Аналогичным образом следующая команда описывает операцию умножения (**mul**).
- `mulss xmm0, dword ptr [.LCPI0_0]`

3.1.4. Умножение с помощью сдвига

Пусть у нас есть число 12 в привычной десятичной системе счисления. Для умножения его на 10 достаточно формально приписать к нему справа ноль — получится 120. Для умножения на $10^2 = 100$ нужно приписать два нуля (1200), и т. д. Если внимательно посмотреть, то в данной операции исходное число фактически сдвигается влево на количество разрядов, равное показателю степени, а «освободившиеся» при этом младшие разряды заполняются нулями.

В двоичной системе все работает точно так же, только множители надо брать кратными не 10, а 2 (т. е. кратными основанию системы счисления). Например, для «быстрого» умножения числа $101_2 = 5_{10}$ на $32 = 2^5$ достаточно сдвинуть код на 5 разрядов влево: $10100000_2 = 160_{10}$ (см. пример 3 в разделе 3.1.2).

Подчеркнем, что все сказанное относится только к целым числам. Для умножения вещественного числа на степень двойки (фактически для этого достаточно просто увеличить порядок числа) существует специальная инструкция **fscale**. Но из-за хранения вещественных чисел в виде двух компонентов (порядок и значащая часть) к указанной инструкции потребуется добавить несколько дополнительных команд. Как (длинно) это будет выглядеть, можно посмотреть на с. 92—93 методического пособия [9].

3.2. Этюд 1: постановка задачи

Перейдем, наконец, непосредственно к содержанию этюда.

Пусть мы хотим посмотреть, как компилятор Julia раскрывает арифметические выражения. Для этого возьмем простую функцию

$$f(x) = 3 + 32x - 2.$$

Легко заметить, что в целях проверки способностей компилятора в ней заложены два «подводных камня». Во-первых, можно заранее вычислить $3 - 2 = 1$ и исключить одну из арифметических операций. Во-вторых, переменная x умножается на степень двойки, что, как мы видели в разделе 3.1.4, позволяет заменить длинную операцию умножения быстрой инструкцией сдвига. Посмотрим, сумеет ли компилятор преодолеть искусственно созданные нами трудности.

3.3. Этюд 1: решение задачи

Начинаем работу. Запускаем Julia и набираем следующие интуитивно легко понятные команды (листинг 1).

Тут все ясно без особых пояснений. Обратим только внимание на два обстоятельства. Во-первых, Julia, стремясь обеспечить максимальный комфорт математикам, позволяет опускать знак умножения в записи $32x$ (языки программирования этого обычно не разрешают). Во-вторых, функция прекрасно работает и для целых, и для вещественных чисел.

команда 1 ответ (готово)	`julia> f(x)=3+32x-2` `f (generic function with 1 method)`
команда 2 ответ (Int)	`julia> f(2)` `65`
команда 3 ответ (Float)	`julia> f(2.0)` `65.0`

Листинг 1. Описание и вызов функции f(x)

А теперь попытаемся заглянуть внутрь нашей подопытной функции. Для этого аккуратно наберем команду вызова функции **code_native()**, описанную в разделе 3.1.1, и увидим на экране следующее (см. третий столбец таблицы в листинге 2; первые два — это вспомогательные комментарии).

Важное примечание. Код для 64-битной версии ОС будет несколько отличаться. 32-битная архитектура IA32 была выбрана автором, так как по ней больше литературы, чем по Intel 64.

На первый взгляд, сложно, но давайте не спеша разберемся.

Разобьем полученный текст программы на четыре фрагмента F0 — F3. (Сразу о хорошем: собственно программа есть самый короткий фрагмент F2.)

Фрагмент F0 содержит некоторый стандартный заголовок. Он практически неизменен и не содержит особо интересной для нас сейчас информации.

Входной фрагмент F1 служит для подготовки извлечения значения аргумента *x*. Параметры в машинных программах, как правило, передаются через стековую память, доступ к которой осуществляется через специальный регистр процессора **esp**. Как следует из строки с номером 2, содержимое этого регистра копируется

Команда		`julia> code_native(f, (Int32,);syntax=:intel)`

F0 — заголовок		`.text`
		`.file "f"`
		`.globl julia_f_152 # -- Begin function julia_f_152`
		`.p2align 4, 0x90`
		`.type julia_f_152,@function`

F1 — вход		`julia_f_152: # @julia_f_152`
		`; ┌ @ REPL[1]:1 within `f``
		`.cfi_startproc`
		`# %bb.0: # %top`

	1	`push ebp`
		`.cfi_def_cfa_offset 8`
		`.cfi_offset ebp, -8`
	2	`mov ebp, esp`
		`.cfi_def_cfa_register ebp`
	3	`and esp, -16`
	4	`sub esp, 16`
F2 — **код программы**	5	`mov eax, dword ptr [ebp + 8]`
		`; │┌ @ int.jl:88 within `*``
	6	`shl eax, 5`
		`; │└`
		`; │┌ @ int.jl:86 within `-``
	7	`or eax, 1`
		`; │└`
F3 — выход	8	`mov esp, ebp`
	9	`pop ebp`
	10	`ret`
		`.Lfunc_end0:`
		`.size julia_f_152, .Lfunc_end0-julia_f_152`
		`.cfi_endproc`
		`; └`
		`# -- End function`
		`.section ".note.GNU-stack","",@progbits`

Листинг 2. Этюд 1, аргумент — Int32

в регистр **ebp**; с его помощью позднее в F2 будет прочитано значение аргумента.

Фрагмент F2 наиболее интересен для нас, так что подробно обсудим его чуть позднее.

И, наконец, последний фрагмент F3 восстанавливает первоначальное состояние регистров и обеспечивает выход из функции по команде **ret**. Заметим, что в F3 нет команд возврата результата функции через стек; это значит, что ответ возвращается через регистр, где был сосчитан, т. е., как мы увидим, через **eax**.

А теперь вернемся к центральному фрагменту F2. Если не обращать внимания на комментарии, которые пишутся после знака «;», остается всего три строки, и в каждой из них — по одной машинной команде. При их расшифровке полезно заглядывать в раздел 3.1.2.

В строке под номером 5 значение аргумента копируется в 32-битный регистр **eax**. Далее с ним будут производиться вычисления по нашей тестовой формуле. Возможно, у вас возник вопрос, зачем к адресу из регистра прибавляется 8. Все дело в том, что в первых 8 байтах стека лежит информация для возврата из функции по команде **ret**, так что их надо пропустить.

Строка 6 свидетельствует о «грамотности» компилятора: он «заметил», что множитель есть степень двойки и заменил операцию умножения командой сдвига (см. разделы 3.1.2 и 3.1.4). Последняя команда находится в строке 7. Здесь тоже видна оптимальность реализации: вместо сложения и вычитания будет выполняться только одна операция. Но почему мы не видим ни сложения (**add**), ни вычитания (**sub**)? Дело в том, что прибавить надо 1, а текущее значение самого младшего бита, полученное в результате сдвига, всегда 0. 0 + 1 = 1 (перенос отсутствует). Очевидно, что выполнять сложение не требуется, а достаточно просто занести в младший бит 1. Что и делает команда логического ИЛИ (**or**). Подчеркнем, что логические операции всегда выполняются чуть быстрее, чем арифметические, поскольку в них не требуется обеспечивать перенос между разрядами.

Итак, мы имеем замечательный результат! Оба «подводных камня» компилятор успешно обошел, а всю формулу сумел вычислить всего за две машинные команды, причем даже не арифметические. Браво, Julia!!!

Попробуем посмотреть, как обстоит дело для других типов данных. Для этого при вызове функции **code_native()** вместо Int32 достаточно написать любой интересующий вас тип данных (Int8-Int128, BigInt, UInt8-UInt128, Float16-Float64 или BigFloat).

Начнем с Int8 и Int16, разрядность которых меньше базовой. Сравнивая вывод с листингом 2, с удивлением обнаружим, что все отличия сосредоточены в строке с номером 5. Напомним, что там извлекается значение аргумента. Сравним команды для Int32 и Int8:

```
mov eax, dword ptr [ebp + 8]
```

и

```
movsx eax, byte ptr [ebp + 8]
```

Ожидаемо различается размер числа (dword и byte). Но еще вместо **mov** используется команда **movsx**. Посмотрев документацию [32] и литературу (например, [5, 15, 18]), понимаем, что в последнем случае операнд со знаком меньшего размера (у нас байт) копируется в регистр большего (двойное слово). При этом старшие 24 бита формально не заданы. Поскольку типы Int могут содержать отрицательные значения, то эти неопределенные биты команда **movsx** устанавливает по следующему правилу: они обнуляются для положительных значений операнда и устанавливаются в единицу для отрицательных. По сути, процессор копирует во все старшие биты знаковый разряд исходного числа. Данный процесс принято называть *расширением знака*.

В результате вместо 8- или 16-битного аргумента получаем 32-разрядный. Все дальнейшие вычисления и результат тоже будут 32-разрядными.

Обратим здесь внимание на еще одно важное обстоятельство. Как мы видим, компилятор по умолчанию не принимает специальных мер для анализа *переполнения* (т. е. получения слишком большого результата, выходящего за пределы отведенного количества битов). Скажем, для нашей функции при Int8, когда $x = 10$, должно получаться $y = 321$, что явно выходит за границы байта. Julia здесь предотвращает ошибку путем возвращения 32-битного результата. (Напомним, что в экспериментах использовалась 32-разрядная ОС.) С практической точки зрения, это, возможно, и выход, но с академической — выглядит несколько неестественно.

Если мы теперь протестируем беззнаковые целочисленные типы UInt8 и UInt16, то увидим в обсуждаемой команде вместо **movsx** мнемонику **movzx**: все значения положительны, а значит, старшие биты всегда устанавливаются нулевыми.

Подчеркнем, что арифметика для чисел со знаком и без знака не отличается.

Завершая обсуждение ситуации с целыми числами, проверим бóльшую разрядность Int64. Здесь (в случае 32-битной ОС) мы обнаружим, что при считывании числа его младшая часть помещается в регистр **ecx**, а старшая — в **edx**. Для совместного сдвига обеих «половинок» используется пара инструкций **shld edx, ecx, 5** и **shl ecx, 5**. Детали для краткости здесь обсуждать не будем.

Если еще нарастить разрядность (Int128 или BigInt), то компилятор ради экономии кода начнет использовать служебные подпрограммы. Их вызывают командой **call**. Анализ такого кода усложняется, так что для первого знакомства он не очень подходит.

Итак, для целочисленных типов компилятор Julia продемонстрировал очень даже оптимальные результаты. Посмотрим теперь, как обстоят дела для вещественных чисел Float32 (листинг 3; фрагменты F1 и F3 сокращены).

Во фрагменте F0 мы замечаем вещественные константы 32, 3 и –2 (строки 1–6). Настроение сразу ухудшается, поскольку вместо двух последних значений хотелось бы видеть их сумму.

Кодовый фрагмент F2 подтверждает пессимистические ожидания. Заглянув в раздел 3.1.3, расшифруем команды 7–12. В строке 7 аргумент загружается в регистр **xmm0** математического сопроцессора. Далее в строках 8–10 выполняются арифметические действия в строгом соответствии с исходной формулой: значение x умножается на 32, а затем тупо выполняются два сложения (вычитание заменено сложением с отрицательной константой –2).

F0 — заголовок		`.text`
		`.file "f"`
		`.section .rodata.cst4,"aM",@progbits,4`
		`.p2align 2 # -- Begin function julia_f_117`

	1	`.LCPI0_0:`
	2	`.long 0x42000000 # float 32`
	3	`.LCPI0_1:`
	4	`.long 0x40400000 # float 3`
	5	`.LCPI0_2:`

	6	`.long 0xc0000000 # float -2`
		`.text`
		`.globl julia_f_117`
		`.p2align 4, 0x90`
		`.type julia_f_117,@function`

F1 — вход		`julia_f_117: # @julia_f_117` …
F2 — **код программы**	7	`movss xmm0, dword ptr [ebp + 8] #xmm0 = mem[0],zero,zero,zero`
		`; ┌ @ promotion.jl:411 within `*` @ float.jl:410`
	8	`mulss xmm0, dword ptr [.LCPI0_0]`
		`; └`
		`; ┌ @ promotion.jl:410 within `+` @ float.jl:408`
	9	`addss xmm0, dword ptr [.LCPI0_1]`
		`; └`
		`; ┌ @ promotion.jl:412 within `-` @ float.jl:409`
	10	`addss xmm0, dword ptr [.LCPI0_2]`
		`; └`
	11	`movss dword ptr [esp + 4], xmm0`
	12	`fld dword ptr [esp + 4]`

F3 — выход		…
		`# -- End function`
		`.section ".note.GNU-stack","",@progbits`

Листинг 3. Этюд 1, аргумент — Float32

Таким образом, в случае вещественного аргумента оптимальность кода, увы, теряется. Во всяком случае, два соседних сложения с константами так и просятся на объединение!

4. Этюд 2. Реализация ветвлений

Подавляющее большинство программ не просто производят последовательные вычисления, но в зависимости от получаемых результатов могут выбирать, какие действия выполнять, а какие нет. Вы, конечно, догадались, что сейчас речь пойдет о ветвлениях и о том, как они реализуются в Julia.

Нам снова потребуются некоторые дополнительные сведения, которые мы предварительно обсудим.

4.1. Этюд 2: предварительные сведения

4.1.1. Анализ условий в процессоре

При построении разветвляющихся и циклических программ используются так называемые *условные* команды, которые в зависимости от результата проверки указанного в них условия либо выполняются, либо нет. Чаще всего это условные переходы, которые всегда работают по следующему алгоритму: если анализируемое условие истинно, то переход происходит; в противном случае он игнорируется и выбирается следующая команда.

Чтобы определить, справедливо ли требуемое условие, любая условная команда обращается к специальному регистру процессора, который называется *регистром флагов*. *Флаги* — это биты, которые устанавливаются в зависимости от результата выполнения инструкции. Например, один из битов, отвечающий за анализ знака числа, устанавливается в 1, если результат отрицателен, и сбрасывается в 0 в противном случае (проще говоря, в него копируется знаковый бит числа).

Установка флагов в процессоре Intel — это тонкий вопрос. Некоторые инструкции на флаги не воздействуют совсем (например, постоянно используемая команда **mov**), а некоторые изменяют только часть флагов; наконец, есть арифметические и некоторые другие операции (в том числе специализированная команда сравнения двух чисел), которые влияют практически на все флаги. Чтобы не ошибиться, полезно проверять сведения о воздействии на флаги по справочнику.

Таким образом, вырисовывается следующая схема реализации ветвления в программе. В ходе вычислений инструкции влияют на состояние регистра флагов. В том месте программы, где нужно произвести ветвление, ставится условная команда, которая анализирует текущее состояние. Рассмотрим пример.

Пример 5.

```
cmp ax, 2
jne label
```

Инструкция **cmp** сравнивает два своих операнда путем вычитания. Полученная разность не сохраняется, но будет проанализирована с целью установки флагов. В нашем примере надо проследить за тем из них, который фиксирует факт равенства/неравенства чисел (а более точно, равенства их разности нулю). Следующая далее инструкция условного перехода по неравенству **jne** (**j**ump if **n**ot **e**qual) анализирует указанный флаг и, если его состояние соответствует неравенству, передает управление на участок программы, помеченный меткой **label**; в противном случае никаких действий не производится и выполнение программы продолжается.

Примечание. Обсуждение логики анализа условий возможно и без упоминания флагов (сравниваем содержимое регистра **ax** с константой 2 и в случае неравенства результата переходим...). Тем не менее понимание механизма работы флагов иметь полезно. Достаточно, например, сказать, что у процессора есть некоторые специальные команды сброса или установки флагов.

По аналогичной логике работают и другие условные команды. Проанализируем такой фрагмент программы:

Пример 6.

```
xor ecx, ecx
cmp eax, 2
sete cl
```

Первая инструкция — исключающее ИЛИ (**xor**) — имеет неочевидное, на первый взгляд, назначение. Как известно, эта логическая операция, по своей сути, позволяет сравнивать двоичные коды: если соответствующие биты операндов совпадают, то результирующий бит сбрасывается в 0, а в противном случае устанавливается в 1. Поскольку двоичное содержимое регистра не может не совпасть «с самим собой» (оба операнда в рассматриваемой команде одинаковы), то результат *всегда* будет нулем и, таким образом, в регистр будет занесен ноль.

Примечание. Можно применить для обнуления регистра инструкцию **mov ecx, 0**. Но, поскольку нулевая константа будет включена в код команды, полученная инструкция будет занимать в памяти больше места и дольше выполняться. Именно поэтому профессиональные программисты (а тем более оптимизирующие компиляторы!) предпочитают вариант с **xor**.

Итак, в начале рассматриваемого фрагмента регистр **ecx** будет обнулен. Далее следует обычная инструкция сравнения значения **eax** с константой 2, которая устанавливает флаги для последующего анализа. Наконец, инструкция в конце фрагмента проверяет флаг, ответственный за равенство (последняя буква в мнемонике операции «e», что означает if **e**qual). Если состояние флага оказалось подходящим, то инструкция выполняется, иначе — игнорируется. А выполнение инструкции состоит в установке (**set**) младшего бита регистра **cl** (причем **cl** — это младшая часть регистра **ecx**). Если же команда «не сработала», то там сохранится прежнее нулевое значение.

В итоге в результате работы рассмотренного фрагмента при **eax** = 2 получается **ecx** = 1; во всех остальных случаях устанавливается **ecx** = 0.

4.1.2. Вызов подпрограмм и передача им параметров

Для вызова подпрограммы используется команда **call**. В отличие от обычного перехода, по команде **call** аппаратным образом сохраняется текущий адрес для той точки основной программы, в которую необходимо вернуться после завершения подпрограммы. Адрес возврата заносится в специально организованную память, которая называется *стеком*. Доступ в стек, как мы уже видели в листинге 2, регулируется регистром **esp** — его еще называют *указателем стека*.

Команда **ret** обеспечивает возврат из подпрограммы по адресу, занесенному в стек командой **call**.

Перед вызовом подпрограммы в стек часто записывают значения входных параметров. Для выходных параметров стек также можно использовать, но компилятор Julia, если удается, применяет более простой способ — результат возвращается через определенный регистр.

При занесении данных в стек по инструкции **push** значение **esp** автоматически уменьшается, а при извлечении (команда **pop**) — растет. Допускается (аккуратное и продуманное) изменение указателя стека **esp**: этим приемом компилятор Julia часто пользуется (см., например, строку 4 в листинге 2).

Более подробно о работе стека (в том числе и при обслуживании подпрограмм) можно почитать в публикации автора [7].

4.1.3. Некоторые необходимые для анализа команды

В программах, которые будут рассмотрены в данном этюде, появится еще несколько новых (для нашего обсуждения) инструкций.

- `jmp label`
  Кроме описанных в разделе 4.1.1 условных переходов существует еще безусловный, который всегда (без всякого анализа флагов) передает управление на указанную метку. Интересно, что в языках высокого уровня безусловный переход принято считать вредным. Так, классик теории программирования Э. В. Дейкстра писал: «Я убежден в том, что оператор go to должен быть отменен в языках программирования “высокого уровня” (т. е. отовсюду, кроме, возможно, простого машинного кода)» [27]. Зато написать полную развилку на машинном языке без **jmp** не удается!
- `inc eax`
  Эта инструкция прибавляет к операнду 1 (инкрементирует его). По сравнению с обычным сложением **add eax, 1**, инструкция **inc** сохраняет значения некоторых флагов. В частности, флага переноса, что упрощает организацию арифметики для «длинных» чисел.
  Для уменьшения значения на 1 предусмотрена инструкция **dec**.
- `cwde`
  Данная экзотическая инструкция позволяет преобразовать 16-битное (w) число со знаком, находящееся в регистре **ax**, в 32-битное (d). Результат помещается в регистре **eax**, младшие байты которого есть не что иное, как **ax**. По сути дела, число в **eax** расширяет свою разрядность

с учетом знака. Очень похожа на нее описанная в этюде 1 инструкция **movsx**, с той лишь разницей, что в **cwde** нельзя изменить операнды.

- `lea eax, [ecx + eax + 1]`
  Инструкция с мнемоникой **lea** (**l**oad **e**ffective **a**ddress) предназначена для определения адреса данных в ОЗУ. Вычисления ведутся по формуле:

  адрес = база + M ∗ индекс + C,

  где база и индекс — это значения в указанных регистрах, C — константа (возможно, отрицательная), а M — множитель, описывающий размер данных (1, 2, 4 или 8 байт). Попутно заметим, что такой способ адресации в процессоре Intel принято называть *масштабируемым*. С помощью описываемой инструкции можно вычислить адрес переменной; особенно это ценно при работе с массивами или строками.
  Можно смотреть на **lea** как на «незаконченную» команду **mov**: когда адрес сосчитан, она выдает **его** в качестве результата вместо того, чтобы прочитать данные из ОЗУ.
  Компиляторы (в том числе и языка Julia) часто используют **lea** как комплексную вычислительную операцию, которая к тому же не портит флаги. Указанная выше команда прибавит к содержимому **eax** значение из регистра **ecx** и к полученной сумме еще добавит 1. И все это одной командой.

4.2. Этюд 2: постановка задачи

Представим себе, что на пустую шахматную доску в горизонтальный ряд с номером i ставится белая пешка. Для простоты будем считать, что $1 < i < 9$, т. е. имеет корректное значение. Программа должна вывести все возможные ходы пешки.

Для тех, кто не силен в шахматных правилах, напомним, что белая пешка двигается по доске на одну клетку вверх, т. е. номер горизонтали увеличивается на единицу. Единственным исключением служит случай начального положения пешки ($i = 2$), когда ее можно двинуть не только на одну, но и при желании на две клетки вверх.

Таким образом, мы имеем дело с простой задачей на ветвление.

4.3. Этюд 2: решение задачи

Обратимся к листингу 4, где приведено решение нашего несложного этюда на языке Julia.

В функции **pawn1()** использована естественная логика: если $i = 2$, то возможны два хода, во всех остальных случаях (**else**) — один. Алгоритмически не слишком красиво то, что один ход будет всегда, независимо от выполнения условия. А это значит, что лучше было бы написать по-другому: если $i = 2$, то возможен дополнительный ход, а далее во всех случаях (вне оператора **if**) один ход реализуется стандартным (безусловным!) образом. Видно, что это хорошо хотя бы потому, что отпадает необходимость в ветви **else**. Но для наших экспериментов интереснее именно приведенная выше версия.

Теперь посмотрим, как устроена функция **pawn1()** для типа Int16. В листинге 5 приводится только собственно программный код, а все вспомогательные фрагменты (они практически не отличаются от показанных ранее в листинге 2) опущены.

Перед нами — классическая реализация развилки.

В строке 1 из стека в 32-битный регистр **eax** считывается 16-битное число. Заметим, что в следующей команде использоваться будет только младшая часть регистра, которую в ассемблере принято маркировать **ax**. В строке 2 полученное значение сравнивается с 2, и если нет совпадения (т. е. $i \neq 2$), то в строке 3 следует переход на метку **LBB0_2** (строка 9) к ветви **else**. Подчеркнем, что в программе было написано $i == 2$, а переход используется противоположный — jne (if **n**ot **e**qual). В случае равенства переход «не срабатывает» и программа выполняется дальше.

В строке 4 в стек заносится аргумент для вывода на экран (см. раздел 4.1.2), равный константе 3. Вспомним, что для этой ветви равенство выполнилось (т. е. значение i равнялось 2). Мы видим, что «мудрый» компилятор, стараясь уменьшить число арифметических действий в программе, заранее сосчитал 2 + 1 и подставил в программу ответ 3. Далее в строке 5 вызывается служебная подпрограмма Julia для вывода целого числа на экран.

Строки 6 и 7 устроены аналогично, только выводимая константа теперь равна 4 (компилятор опять предусмотрительно вычислил следующее значение).

команда 1
(ввод функции)
 	
ответ (готово)
	julia> function pawn1(i::Integer)
	  if i==2
	  i=i+1; println(i)
	  i=i+1; println(i)
	  else
	  i=i+1; println(i)
	  end
	  end
	pawn1 (generic function with 1 method)

команда 2 (тест 1) ответы команда 3 (тест 2) ответ	julia> pawn1(2) 3 4 julia> pawn1(4) 5

*Листинг 4. Описание **полного** варианта функции и ее вызов*

F2 — **код программы**

```
1           movzx   eax, word ptr [ebp + 8]
            ; | @ REPL[2]:2 within `pawn1`
            ; |┌ @ promotion.jl:449 within `==` @ promotion.jl:499
2           cmp     ax, 2
            ; |└
3           jne     .LBB0_2
            # %bb.1:                                # %L4
            ; | @ REPL[2]:3 within `pawn1`
4           mov     dword ptr [esp], 3
5           call    j_println_108@PLT
            ; | @ REPL[2]:4 within `pawn1`
6           mov     dword ptr [esp], 4
7           call    j_println_108@PLT
8           jmp     .LBB0_3
9           .LBB0_2:                                # %L10
            ; | @ REPL[2]:6 within `pawn1`
            ; |┌ @ int.jl:1040 within `+`
            ; ||┌ @ int.jl:523 within `rem`
10          cwde
            ; ||└
            ; || @ int.jl:1042 within `+` @ int.jl:87
11          inc     eax
            ; |└
12          mov     dword ptr [esp], eax
13          call    j_println_108@PLT
14          .LBB0_3:                                # %common.ret
```

*Листинг 5. Этюд 2, **полный** вариант, аргумент — Int16*

После этого в строке 8 следует безусловный переход на метку **LBB0_3** (строка 14), что обеспечивает обход альтернативной ветви **else**. Подчеркнем, что именно здесь то самое место, где безусловный переход незаменим.

Переходим к анализу кода ветви **else**. В строке 10 аргумент *i* из 16-битного регистра **ax** преобразуется в 32-битный и результат заносится в **eax** (см. раздел 4.1.3). В этом месте, как нам кажется, компилятор недоработал: если бы в строке 1 вместо **movzx** стояло **movsx** (команды обсуждались ранее в этюде 1), то строка 10 не потребовалась бы!

В строке 11 текущее значение **eax** (т. е. *i*) увеличивается на 1. Результат в строке 12 заносится в стек в качестве аргумента, и в следующей строке следует вызов служебной подпрограммы вывода.

Строка 14 служит выходом из развилки: в нее мы попадаем как при выполнении, так и при невыполнении условия $i = 2$ (вспомним про переход в строке 3).

А теперь, с точки зрения изучения оптимальности кода, интересно проделать еще один эксперимент. Удалим из функции **pawn1()** все операторы вывода **println(i)**. Чтобы как-то увидеть результат, добавим в конец функции (перед последним **end**) оператор **return i**, который будет возвращать значение *i* в качестве ответа. Автор надеется, что читатели без труда самостоятельно внесут перечисленные изменения в листинг 4.

Машинный код для сокращенной программы получится совсем коротким и очень необычным (листинг 6). Бросается в глаза полное отсутствие переходов. Как же это работает?

Строка 1 считывает из стека в 32-битный регистр **eax** 16-битный аргумент, расширяя при этом его знак инструкцией **movsx**. Фрагмент в строках 2–4 вырабатывает в **ecx** значение либо 1, когда **eax** (т. е. *i*) = 2, либо 0 в противном случае; работа кода подробно описана в разделе 4.1.1. Именно здесь и сосредоточена развилка, которая реализована за счет условной инструкции **sete**.

Наконец, в строке 5 результат вычисляется по формуле

$$eax = eax + 1 + ecx,$$

причем последнее слагаемое равно 1 при $i = 2$ и 0 в остальных случаях. Иными словами, результирующее значение переменной *i* увеличивается либо на 2, либо на 1.

Заметим, что отсутствие какого-либо дополнительного кода, связанного с реализацией оператора **return i**, косвенно свидетельствует о том, что результат вычислений будет взят непосредственно из регистра **eax**.

F2 — **код программы**

```
1           movsx   eax, word ptr [ebp + 8]
2           xor     ecx, ecx
3           cmp     eax, 2
4           sete    cl
            ; | @ REPL[1]:2 within `w_pawn`
5           lea     eax, [ecx + eax + 1]
```

*Листинг 6. Этюд 2, **сокращенный** вариант, аргумент — Int16*

5. Этюд 3. Реализация циклов

Чтобы наши упражнения-этюды охватили все базовые алгоритмические конструкции, осталось посмотреть, как реализуются циклы. Тут нас ждет приятный сюрприз: оказывается, организация кода для циклов во многом похожа на то, как это делалось для развилки. Если вдуматься, то логика построения неполной (без **else**) развилки действительно имеет много общего со структурой цикла с предусловием **while**. В связи с этим автор во многом надеется на самостоятельность читателей при разборе данного этюда.

5.1. Этюд 3: постановка задачи

Изучим реализацию программы, которая выводит на экран «обратный отсчет»: последовательность уменьшающихся положительных чисел, которая начинается с заданного значения *k* и заканчивается 1.

5.2. Этюд 3: решение задачи

Решение нашего несложного этюда на языке Julia приведено в листинге 7.

Фрагмент F2 эквивалентного машинного кода, составленного компилятором, размещен в листинге 8. Кратко прокомментируем его.

Строка 1 считывает 32-битное (dword) значение аргумента в регистр **esi**. В строке 2 полученное значение тестируется (инструкция **test** эквивалентна **and** без записи результата; ее цель — установить флаги в соответствии с текущим значением операнда). По результатам анализа в случае отрицательного или нулевого аргумента по команде **jle** (**j**ump if **l**ess or **e**qual — переход по меньше или равно) происходит обход цикла, так как предусловие сразу оказывается не выполнено, и цикл игнорируется.

Строки 5–6 узнаваемо выводят текущее значение переменной *k* из **esi**, а в строке 7 переменная уменьшается на 1.

А далее ради оптимизации кода компилятор «совершает подлог»: вместо того чтобы сделать безусловный переход на проверку *предусловия* (строки 2–3), в строке 8 он вставляет *постусловие*. Оптимизация здесь заключается в том, что нет необходимости специально проверять на равенство нулю текущее значение переменной, поскольку инструкция **dec** это и так уже сделала! В итоге, если счетчик не дошел до 0, цикл повторяется, в противном случае перехода нет и цикл завершается.

Примечание. Попутно заметим, что цикл с постусловием всегда имеет более короткий код, чем цикл с предусловием. Печально, что в Julia такой цикл, похоже, вообще не предусмотрен.

команда 1
(ввод функции)
 	
ответ (готово)
	`julia> function countdown(k)`
	`  while k>0`
	`  println(k)`
	`  k=k-1`
	`  end`
	`  end`
	`countdown (generic function with 1 method)`

команда 2 (тест)
результаты
	`julia> countdown(3)`
	`3`
	`2`
	`1`

Листинг 7. Описание и вызов функции

F2 — **код программы**		
	1	`  mov  esi, dword ptr [ebp + 8]`
		`; │ @ REPL[19]:2 within `countdown``
		`; │┌ @ operators.jl:369 within `>``
		`; ││┌ @ int.jl:83 within `<``
	2	`  test  esi, esi`
		`; │└└`
	3	`  jle  .LBB0_2`
		`  .p2align  4, 0x90`
	4	`.LBB0_1:  # %L5`
		`  # =>This Inner Loop Header: Depth=1`
		`; │ @ REPL[19]:3 within `countdown``
	5	`  mov  dword ptr [esp], esi`
	6	`  call  j_println_172@PLT`
		`; │ @ REPL[19]:2 within `countdown``
		`; │┌ @ operators.jl:369 within `>``
		`; ││┌ @ int.jl:83 within `<``
	7	`  dec  esi`
		`; │└└`
	8	`  jne  .LBB0_1`
	9	`.LBB0_2:  # %L8`

Листинг 8. Этюд 3, аргумент — Int32

Те, кто хочет заглянуть поглубже, могут в листинге 7 удалить **println** (подобно тому как мы делали в этюде 2) и, добавив еще строку **return k**, посмотреть генерируемый код. Вероятно, вы удивитесь, но цикла для целого аргумента в этом случае совсем не будет! Будет только ветвление: если аргумент $k \le 0$, то он и является ответом (моделируется игнорирование цикла), либо сразу 0 в противном случае (зачем «прокручивать» цикл, если его результат заранее известен?!).

Любопытно, что для вещественных аргументов такой оптимизации не будет и цикл сохранится.

6. Заключение

Надеюсь, предложенные вашему вниманию эксперименты вызвали интерес. Какие из них следует сделать выводы?

- Действительно, с помощью Julia легко «увидеть программу изнутри», т. е. анализировать эквивалентный ей исполняемый машинный код (на наш взгляд, анализ простейших случаев доступен даже школьнику). Возможно, потребуются материалы по системе команд процессора, но для семейства Intel (невзирая даже на санкционные ограничения, принятые сейчас на официальном сайте компании) это не проблема.
- Проведенные эксперименты подтверждают, что генерируемый компилятором Julia код вполне эффективен.
- Оптимизация для целочисленных данных заметно лучше, чем для вещественных. Для целых чисел компилятор, как мы увидели, очень успешно старается обойтись без трудоемких арифметических операций (если удается, то значения вычисляются заранее, умножение заменяется сдвигом и т. п.). Компилятор даже способен предсказать результаты простейшего цикла и исключить сам цикл из программы.
- Для вещественных чисел компилятор активно использует инструкции новой технологии SSE. Поэтому интересно в будущих экспериментах посмотреть на организацию параллельных вычислений для векторных операций.
- Ради краткости кода компилятор по умолчанию не принимает мер по фиксации переполнения. Перед нами еще один аргумент в пользу важности понимания того, как компьютер считает и когда у него могут возникать вычислительные трудности.
- Проведенное рассмотрение кода учит, что тщательный анализ алгоритма и знания в области теории программирования могут помочь существенно ускорить работу программы. На наш взгляд, в подготовке современных специалистов значение подобного опыта трудно переоценить.
- К языку Julia стоит присмотреться повнимательнее!

Список источников

1. *Антонюк В. А.* Язык Julia как инструмент исследователя. М.: Физ. фак. МГУ имени М. В. Ломоносова, 2019. 48 с. https://cmp.phys.msu.ru/sites/default/files/VA_Antonyk_Julia_2019.pdf

2. *Белов Г. В.* Краткое описание языка программирования Julia и примеры его использования. М.: МГТУ им. Н. Э. Баумана, 2023. 105 с. http://ihed.ras.ru/~thermo/Julia/Brief description of Julia language.pdf

3. *Белов Г. В., Аристова Н. М.* О возможностях использования языка программирования Julia для решения научных и технических задач // Вестник МГТУ им. Н. Э. Баумана. Серия «Приборостроение». 2020. № 2 (131). С. 27–43. EDN: XODPOB. DOI: 10.18698/0236-3933-2020-2-27-43

4. *Белов Г. В., Аристова Н. М., Мальцев М. А.* Использование языка программирования Julia для расчета равновесного состава многокомпонентной газофазной системы // Вестник Объединенного института высоких температур. 2022. Т. 7. № 1. С. 52–55. EDN: OIBSEI.

5. *Гук М. Ю.* Процессоры Intel: от 8086 до Pentium II. СПб.: Питер, 1997. 224 с.

6. *Еремин Е. А.* Компилятор? Это довольно просто! Пермь: Изд-во ПРИПИТ, 1998. 120 с.

7. *Еремин Е. А.* Стек // Информатика. 2008. № 2. С. 37–38; 2008. № 3. С. 42–44; 2008. № 4. С. 41–44; 2008. № 6. С. 43–45; 2008. № 8. С. 43–45; 2008. № 9. С. 39–42. http://emc.orgfree.com/theory/raznoe/d4.html

8. *Ефремова Т. Ф.* Самый полный толковый словарь русского языка. В 3 т. М.: АСТ, 2015. 3312 с.

9. *Иванова Е. М.* Программная модель процессора и базовые инструкции процессоров Intel и MIPS. М.: НИУ ВШЭ, 2019. 139 с. https://publications.hse.ru/pubs/share/direct/317732672.pdf

10. ИТ-шники всего мира назвали самый любимый и самый ненавистный язык программирования // CNews, 4 августа 2021. https://www.cnews.ru/news/top/2021-08-03_programmisty_vsego_mira

11. *Кулябов Д. С., Королькова А. В.* Компьютерная алгебра на Julia // Программирование. 2021. № 2. С. 44–50. EDN: HETTIQ.

12. *Лопес Б. К., Аулер Р.* LLVM: инфраструктура для разработки компиляторов. М.: ДМК Пресс, 2015. 342 с.

13. *Ожегов С. И., Шведова Н. Ю.* Толковый словарь русского языка. М.: Азбуковник, 1997. 944 с.

14. *Оконешникова Е. А.* Новый язык программирования в учебном процессе и научных исследованиях // Наука. Информатизация. Технологии. Образование. Материалы XIII международной научно-практической конференции (Екатеринбург, 24–28 февраля 2020 г.). Екатеринбург: Российский государственный профессионально-педагогический университет, 2020. С. 123–129. EDN: WVFCHN.

15. *Смит Б. Э., Джонсон М. Т.* Архитектура и программирование микропроцессора INTEL 80386. М.: Конкорд, 1992. 334 с.

16. *Таненбаум Э. С.* Архитектура компьютера. СПб.: Питер, 2003. 704 с.

17. *Устинин Д. М.* Использование языка научных вычислений Julia для моделирования внутриклеточных процессов методом броуновской динамики // Доклады Международной конференции «Математическая биология и биоинформатика». 2018. № 7. С. e29.1–e29.4. EDN: YPPYWL. DOI: 10.17537/icmbb18.93

18. *Шагурин И. И., Бердышев Е. М.* Процессоры семейства Intel P6. Архитектура, программирование, интерфейс. М.: Горячая линия — Телеком, 2000. 248 с.

19. *Шадрин Ю. А., Грюмова Е. А., Гумирова А. И.* Обзор языка программирования Julia // Технологии инженерных и информационных систем. 2019. № 3. С. 43–52. EDN: IIUXHP.

20. *Шеррингтон М.* Осваиваем язык Julia. Совершенствование мастерства в области аналитики и программирования. М: ДМК-Пресс, 2017. 416 с.

21. *Шиндин А. В.* Язык программирования математических вычислений Julia. Базовое руководство. Нижний Новгород: ННГУ, 2016. 24 с. http://www.unn.ru/books/met_files/JULIA_tutorial.pdf

22. *Энгхейм Э.* Julia в качестве второго языка. М.: ДМК Пресс, 2023. 446 с.

23. *Ben-Ari M.* Constructivism in computer science education // Journal of Computers in Mathematics and Science Teaching. 2001. Vol. 20. No. 1. P. 45–73.

24. *Bezanson J., Chen J., Chung B., Karpinski S., Shah V. B., Vitek J., Zoubritzky L.* Julia: Dynamism and performance reconciled by design // Proceedings of the ACM on Programming Languages. 2018. Vol. 2 (OOPSLA). P. 120.1–120.23. DOI: 10.1145/3276490

25. *Bezanson J., Edelman A., Karpinski S., Shah V. B.* Julia: A fresh approach to numerical computing // SIAM Review. 2017. Vol. 59. No. 1. P. 65–98. DOI: 10.1137/141000671

26. *Bezanson J., Karpinski S., Shah V. B., Edelman A.* Julia: A fast dynamic language for technical computing // arXiv preprint. arXiv:1209.5145. 2012. DOI: 10.48550/arXiv.1209.5145

27. *Dijkstra E. W.* Go to statement considered harmful // Communications of the ACM. 1968. Vol. 11. No. 3. P. 147–148. DOI: 10.1145/362929.362947. (Русский перевод: http://hosting.vspu.ac.ru/~chul/dijkstra/goto/goto.htm)

28. *Du Boulay B.* Some difficulties of learning to program // Journal of Educational Computing Research. 1986. Vol. 2. No. 1. P. 57–73. DOI: 10.2190/3LFX-9RRF-67T8-UVK9

29. *Eremin E. A.* Educational Pascal compiler into MMIX code // Proceedings of the 6th Baltic Sea Conference on Computing Education Research, Koli Calling. 2007. P. 143–144. EDN: MWQNJR.

30. *Gao K., Mei G., Piccialli F., Cuomo S., Tu J., Huo Z.* Julia language in machine learning: Algorithms, applications, and open issues // Computer Science Review. 2020. Vol. 37. P. 100254.1–100254.13. DOI: 10.1016/j.cosrev.2020.100254

31. *Gevorkyan M. N., Korolkova A. V., Kulyabov D. S.* Julia language features for processing statistical data // Discrete and Continuous Models and Applied Computational Science. 2023. Vol. 31. No. 1. P. 5–26. EDN: VNJCSU. DOI:10.22363/2658-4670-2023-31-1-5-26

32. Intel 64 and IA-32 Architectures Software Developer's Manual. 2023. 5066 p. http://asmcourse.cs.msu.ru/wp-content/uploads/2022/01/325383-sdm-vol-2abcd.pdf

33. *Lattner C.* LLVM // The Architecture of Open Source Applications. 2011. Vol. 1. P. 155–170. (Русский перевод: https://rus-linux.net/MyLDP/BOOKS/Architecture-Open-Source-Applications/Vol-1/llvm-1.html)

34. *Lattner C., Adve V.* LLVM: a compilation framework for lifelong program analysis & transformation // Proceedings of the International Symposium on Code Generation and Optimization (CGO 2004). San Jose, CA, USA, 2004. P. 75–86. DOI: 10.1109/CGO.2004.1281665

35. *Miller E.* Why I'm betting on Julia. https://www.evanmiller.org/why-im-betting-on-julia.html. (Русский перевод: https://habr.com/ru/articles/210298/)

36. *Nazarathy Y., Klok H.* Statistics with Julia: Fundamentals for data science, machine learning and artificial intelligence. Springer Series in the Data Sciences. Springer Nature, 2021. 527 p. DOI: 10.1007/978-3-030-70901-3

37. *Salceanu A.* Julia Programming Projects: Learn Julia 1.x by building apps for data analysis, visualization, machine learning, and the web. Birmingham: Packt Publishing Ltd, 2018. 500 p.

38. *Sengupta A., Edelman A.* Julia High Performance: Optimizations, distributed computing, multithreading, and GPU programming with Julia 1.0 and beyond. 2nd Edition. Birmingham: Packt Publishing Ltd, 2019. 218 p.

39. The Julia Language. V.1.9.2. The Julia Project. July 6, 2023. 1644 p. https://docs.julialang.org/en/v1/

DOI: 10.32517/2221-1993-2023-22-6-31-40

А. М. Кочигина, П. А. Корнилов
Ярославский государственный педагогический университет им. К. Д. Ушинского, г. Ярославль, Россия

УЧЕБНАЯ ПРОГРАММНАЯ СРЕДА ДЛЯ ИЗУЧЕНИЯ АБСТРАКТНЫХ ВЫЧИСЛИТЕЛЬНЫХ МАШИН ТЬЮРИНГА И ПОСТА*

Аннотация
Федеральная рабочая программа среднего общего образования по информатике на углубленном уровне предполагает знакомство школьников с машинами Тьюринга и Поста как универсальными моделями вычислений. Использование в учебном процессе интерпретаторов машин Тьюринга и Поста позволяет преодолеть затруднения, связанные с обучением, опирающимся на абстрактно-логическое мышление. В статье описаны возможности разработанной авторами учебной программной среды, в которой для каждой вычислительной машины содержатся три раздела. В разделе «Теория» учащиеся могут познакомиться с особенностями работы машин Поста и Тьюринга. Раздел «Тренажер» позволяет пользователю составить собственные примеры программ или проанализировать готовые, в данном разделе есть возможность выполнить программу по шагам, при этом каждый шаг будет сопровождаться комментариями. Кроме того, в этом разделе можно выполнить лабораторную работу, задания которой генерируются случайным образом согласно шаблону. С помощью раздела «Контроль» учитель может проверить знания школьников по данной теме с помощью серии задач — каждое задание выбирается случайным образом из заготовленной базы задач. Описанная в статье учебная программная среда позволяет организовать как фронтальную, так и индивидуальную работу учащихся.
Ключевые слова: абстрактные вычислительные машины, машина Тьюринга, машина Поста, учебная программная среда.

1. Введение

Федеральная рабочая программа среднего общего образования по информатике на углубленном уровне предполагает изучение школьниками формализации понятия алгоритма, их знакомство с машинами Тьюринга и Поста как универсальными моделями вычислений [10].

Первые попытки предложить школьникам рассмотрение абстрактных вычислительных моделей были предприняты в середине 80-х годов прошлого века [8]. В своей статье [9] И. Н. Фалина и Е. Л. Радченко отмечают, что начальным этапом изучения теории алгоритмов и программирования является изучение машины Поста. Несмотря на свою простоту, машина Поста является универсальным исполнителем, поскольку на ней можно записать любой алгоритм.

Выполнение отмеченной в федеральной рабочей программе практической работы на составление простой про-

* Материалы к статье можно скачать на сайте ИНФО: http://infojournal.ru/journals/school/school_06-2023/

Контактная информация

Кочигина Анна Михайловна, студентка 5-го курса кафедры теории и методики обучения информатике физико-математического факультета, Ярославский государственный педагогический университет им. К. Д. Ушинского, г. Ярославль, Россия; *адрес:* 150000, Россия, г. Ярославль, ул. Республиканская, д. 108; *e-mail:* kociginaanna@gmail.com

Корнилов Петр Анатольевич, канд. физ.-мат. наук, доцент, зав. кафедрой теории и методики обучения информатике, Ярославский государственный педагогический университет им. К. Д. Ушинского, г. Ярославль, Россия; *адрес:* 150000, Россия, г. Ярославль, ул. Республиканская, д. 108; *e-mail:* kornilovpa@yandex.ru

A. M. Kochigina, P. A. Kornilov
Yaroslavl State Pedagogical University named after K. D. Ushinsky, Yaroslavl, Russia

EDUCATIONAL SOFTWARE ENVIRONMENT FOR THE STUDY OF ABSTRACT TURING AND POST COMPUTING MACHINES

Abstract
The Federal Work Program of Secondary General Education in Informatics at an in-depth level involves familiarization of schoolchildren with Turing and Post machines as universal models of computing. The use of Turing and Post machines interpreters in the educational process makes it possible to overcome the difficulties associated with learning based on abstract logical thinking. The article describes the possibilities of the educational software environment developed by the authors, which contains three sections for each computer. In the "Theory" section, students can get acquainted with the features of the Post and Turing machines. The "Simulator" section allows the user to create their own examples or analyze ready-made ones, in this section it is possible to run the program step by step, with each step accompanied by comments. In addition, in this section, you can perform laboratory work, the tasks of which are randomly generated according to a template. With the help of the "Control" section, the teacher can test students' knowledge on this theme by solving a series of tasks. Each task is randomly selected from the prepared database of tasks. The principle of gradualness and sequence of tasks is observed. The educational software environment described in the article allows you to organize both front-end and individual work of students.
Keywords: abstract computing machines, Turing machine, Post machine, educational software environment.

Рис. 1. Главное меню учебной программной среды «Основы теории алгоритмов»

граммы для машины Тьюринга может быть осуществлено без использования компьютера, однако в работе А. С. Чухнова, С. Н. Позднякова [11] отмечено, что построение машины Тьюринга с помощью некоторого интерпретатора дает лучшие результаты, чем ее построение на бумаге; кроме того, успешное выполнение конструктивных задач с помощью интерпретатора влечет более успешное выполнение теоретических задач по этой же тематике.

И. Д. Колдунова отмечает, что использование интерпретаторов машин Тьюринга и Поста позволяет преодолеть затруднения, связанные с обучением, опирающимся на абстрактно-логическое мышление. Визуализация учебной информации позволяет обеспечить интенсификацию процесса обучения, активизировать познавательную деятельность школьников, способствует формированию и развитию образного представления знаний и учебных действий [2].

В настоящее время можно найти большое количество интерпретаторов машин Тьюринга и Поста [3–5, 12]. Однако в них заложена лишь возможность выполнять программы и получать результат. Для эффективной организации изучения темы целесообразно использовать обучающую учебную среду, в которой школьники могли бы познакомиться с описанием исполнителей, проанализировать работу готовых алгоритмов, самостоятельно создавать программы и осуществлять самоконтроль изученного материала.

Учебная программная среда «Основы теории алгоритмов» позволяет решить все эти педагогические задачи, дает возможность организовать изучение темы как комплексно, так и с использованием лишь отдельных ее элементов.

Главное меню программной среды предлагает выбор одного из двух исполнителей — «Машина Тьюринга» или «Машина Поста» (рис. 1). Для каждого из исполнителей пользователю предлагается последовательно пройти три учебных блока (рис. 2):

1) «Теория»;
2) «Тренажер»;
3) «Контроль».

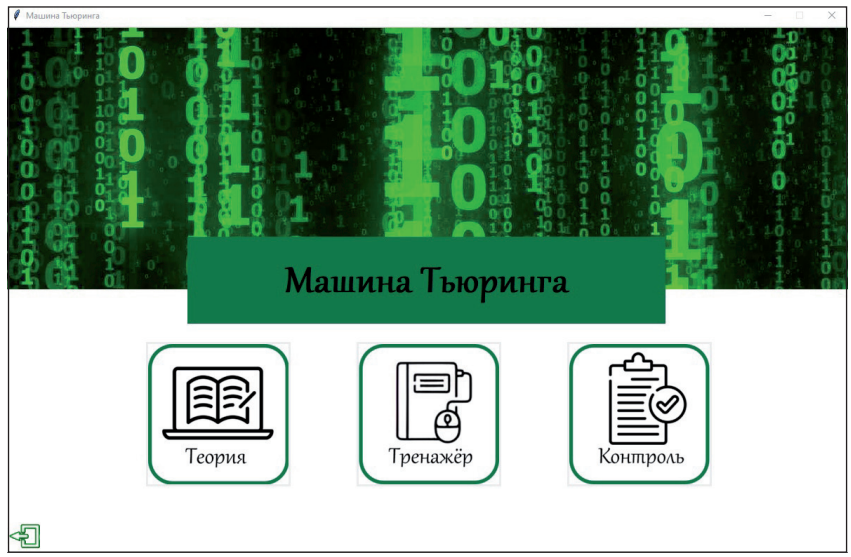


Рис. 2. Изучение универсального исполнителя «Машина Тьюринга»

2. Блоки учебной программной среды для работы с исполнителем «Машина Тьюринга»

Блок «Теория» (рис. 3) знакомит учащегося с процессом составления алгоритма для машины Тьюринга. В этом разделе учащиеся узнают, что такое бесконечная лента, какие у нее составляющие, как выглядит таблица правил [7, 8].

Блок «Тренажер» (рис. 4) позволяет пользователю приобрести опыт разработки собственных программ для машины Тьюринга.

Интерфейс этой части программы содержит несколько кнопок, бесконечную ленту, таблицу правил и поля с комментариями. Ленту можно двигать в двух направлениях, а также выбирать для каждого ее элемента свой символ из представленного алфавита. По умолчанию алфавит состоит из трех символов: ноль,

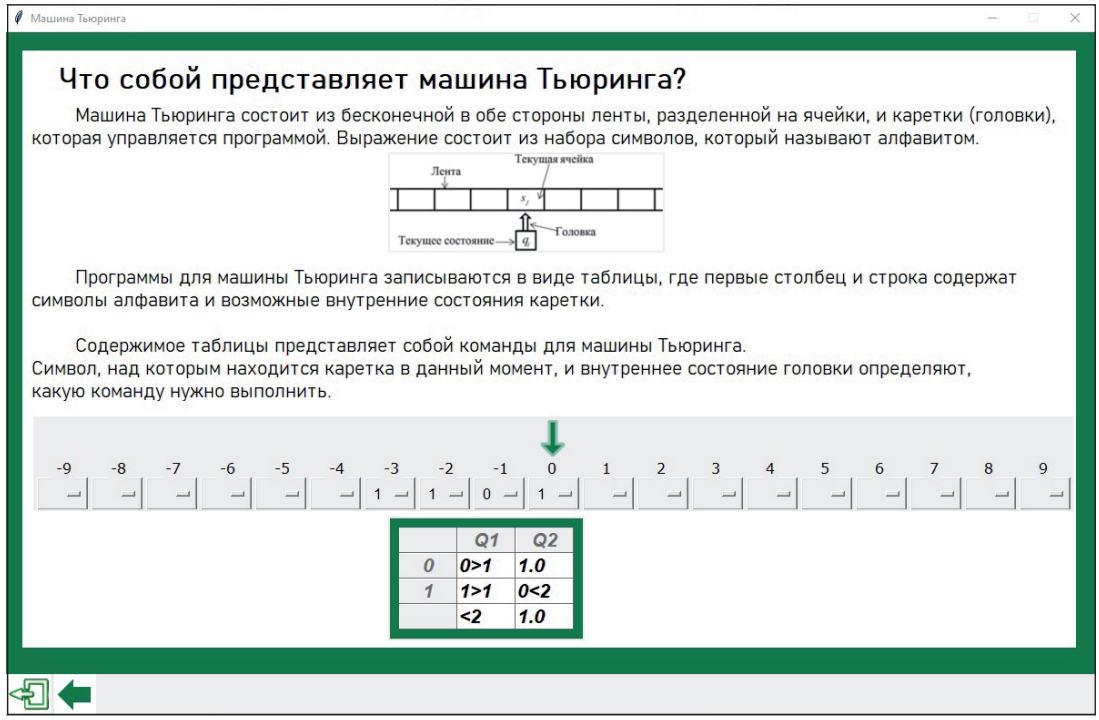


Рис. 3. Блок «Теория» для исполнителя «Машина Тьюринга»

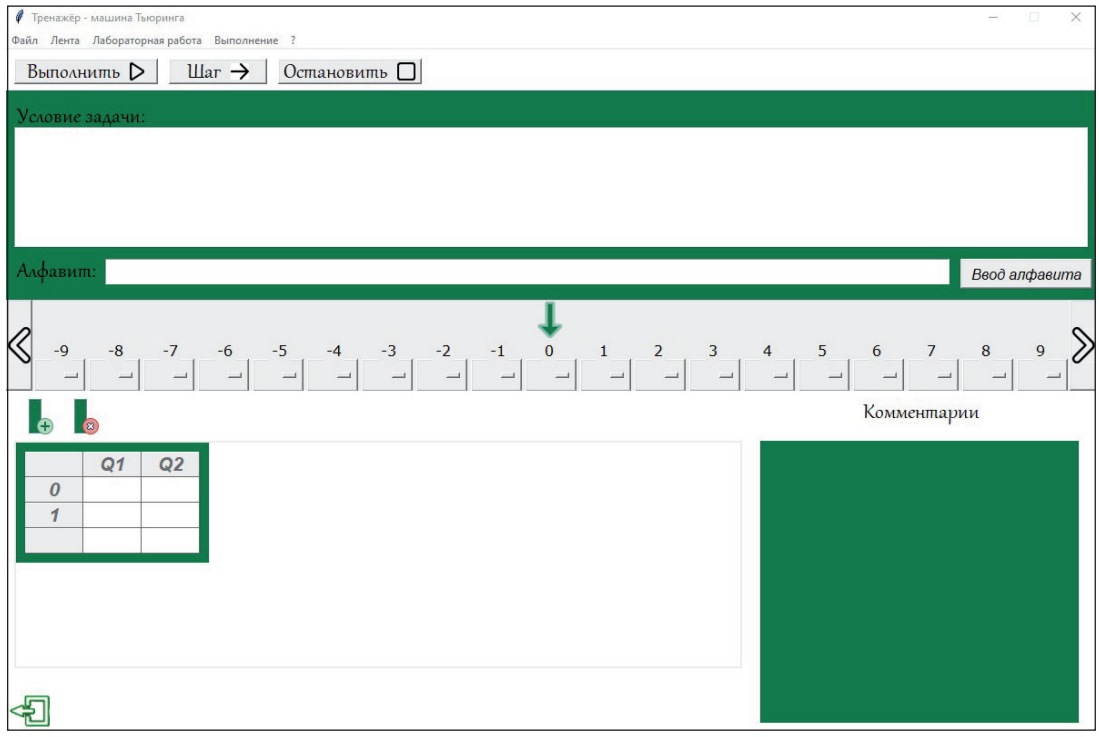


Рис. 4. Блок «Тренажер» для исполнителя «Машина Тьюринга»

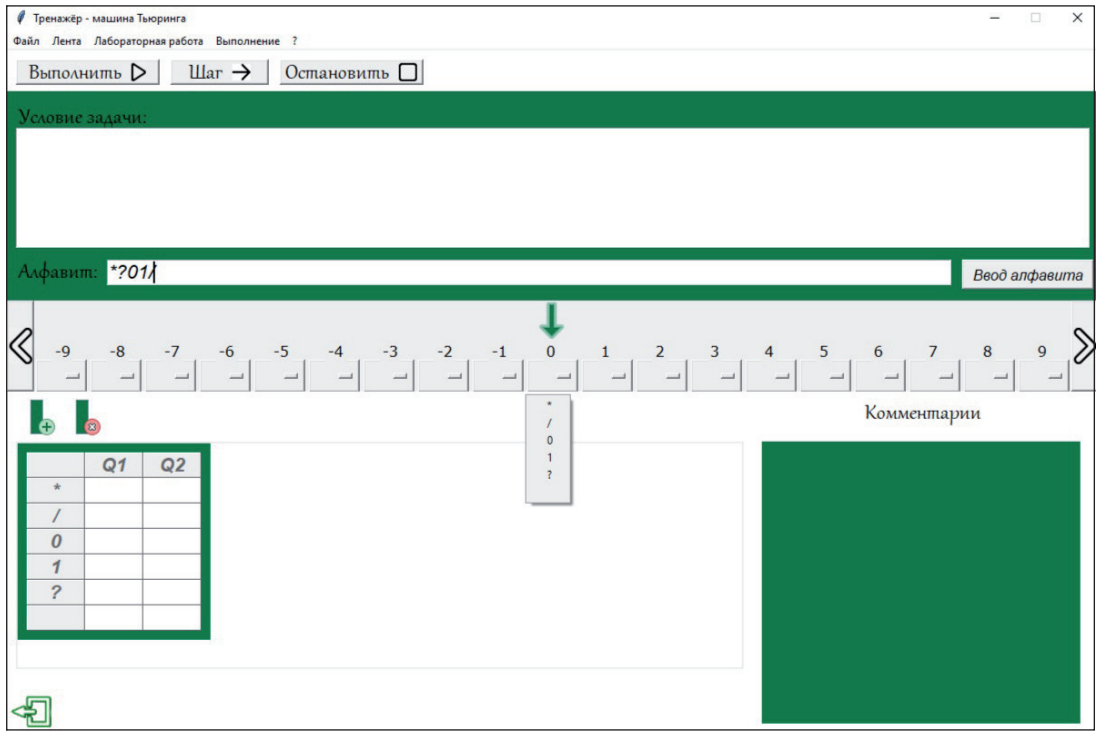


Рис. 5. Ввод алфавита в блоке «Тренажер» для исполнителя «Машина Тьюринга»

единица и пустой. Пользователь может изменить алфавит, воспользовавшись соответствующими полем ввода и кнопкой. В таком случае допустимые символы для элементов ленты изменятся. Также автоматически изменится и таблица правил, расположенная в левом нижнем углу (рис. 5). С помощью кнопок, расположенных над таблицей, можно добавлять и удалять столбцы, которые обозначают состояния.

Пользователь может запустить созданный алгоритм в двух режимах. При нажатии на кнопку *Выполнить* на экране будет видно измененную согласно правилам ленту. С помощью кнопки *Шаг* можно запустить пошаговое выполнение алгоритма, при этом на каждом шаге будут видны текущее правило, текущий элемент и состояние ленты. Стоит отметить, что бесконечная лента двигается самостоятельно, если это подразумевает

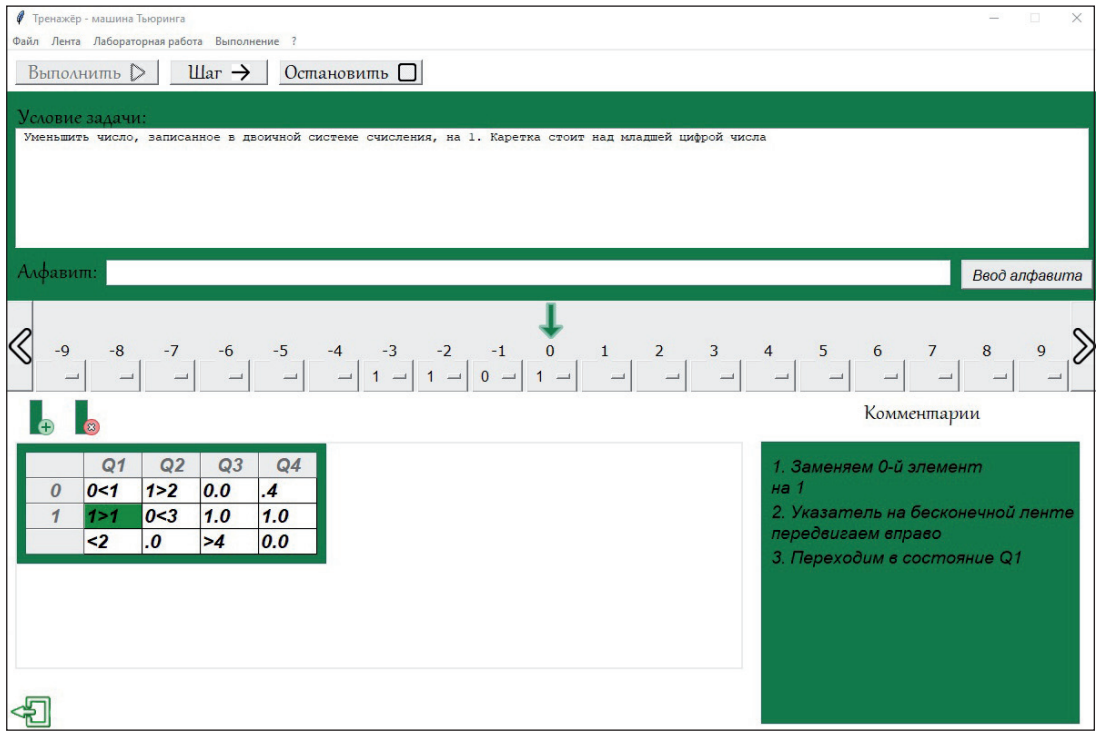


Рис. 6. Пошаговое выполнение алгоритма (блок «Тренажер» для исполнителя «Машина Тьюринга»)

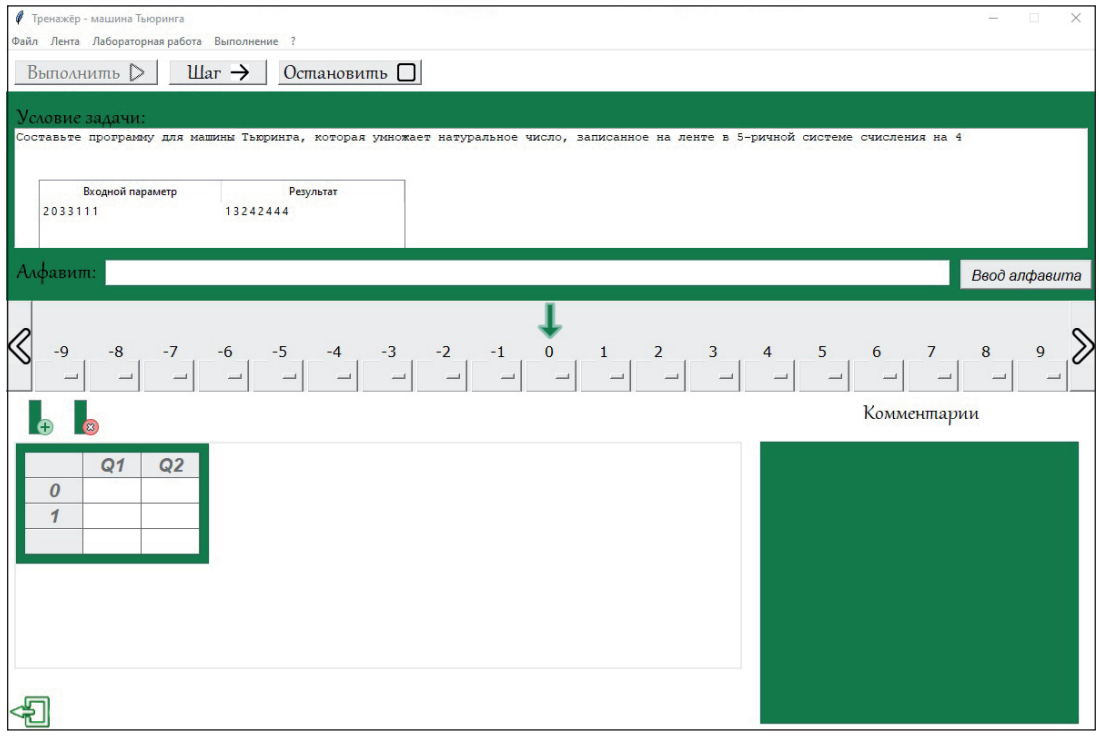


Рис. 7. Раздел «Лабораторная работа» (пример задания 2) для исполнителя «Машина Тьюринга»

правило. Каждый шаг сопровождается комментариями, расположенными в правом нижнем углу (рис. 6).

Пункт верхнего меню *Лента* позволяет сохранять и восстанавливать бесконечную ленту. При восстановлении изменяются все составляющие: выражение, текущий элемент, положение ленты.

В разделе «Лабораторная работа» содержатся пять типов заданий. Задачи генерируются случайным образом на основе шаблонов. Учащимся необходимо написать программу и проверить ее работу на представленных тестах (рис. 7, 8).

Пункт меню *Файл* позволяет открывать имеющиеся примеры, сохранять собственные в отдельный файл, а также создавать новые. С помощью подпункта *Новый* можно очистить все поля: условие задачи, алфавит, бесконечную ленту и таблицу правил. Подпункт *Открыть* дает воз-

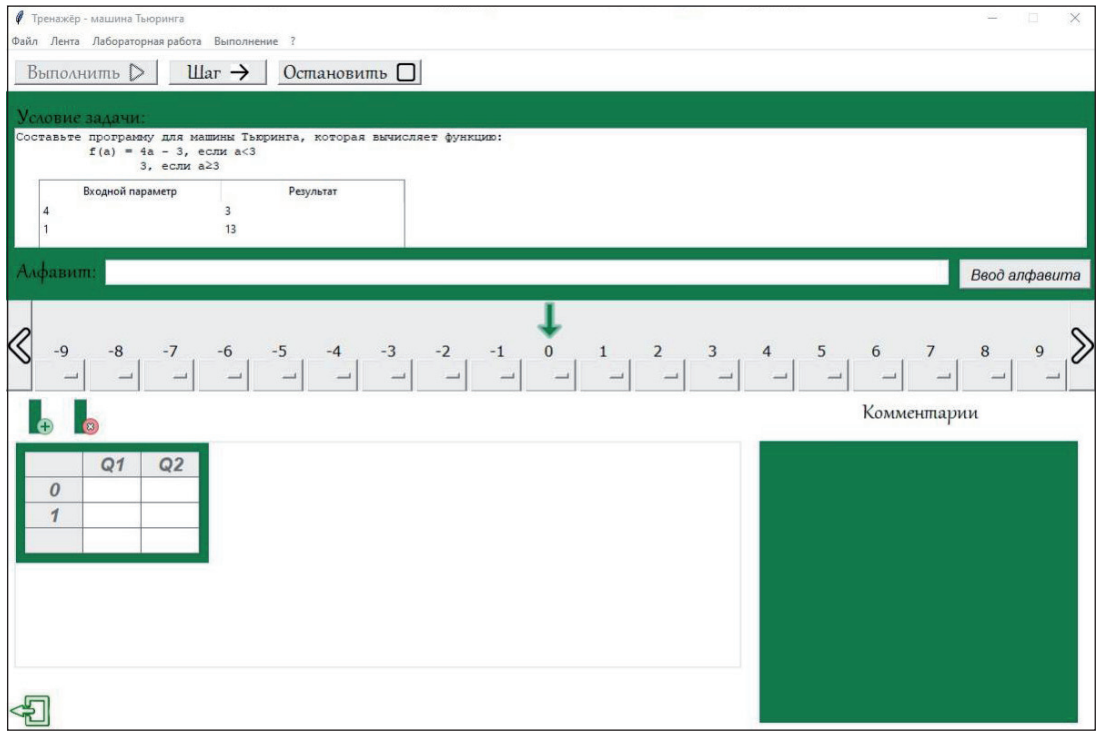


Рис. 8. Раздел «Лабораторная работа» (пример задания 5) для исполнителя «Машина Тьюринга»

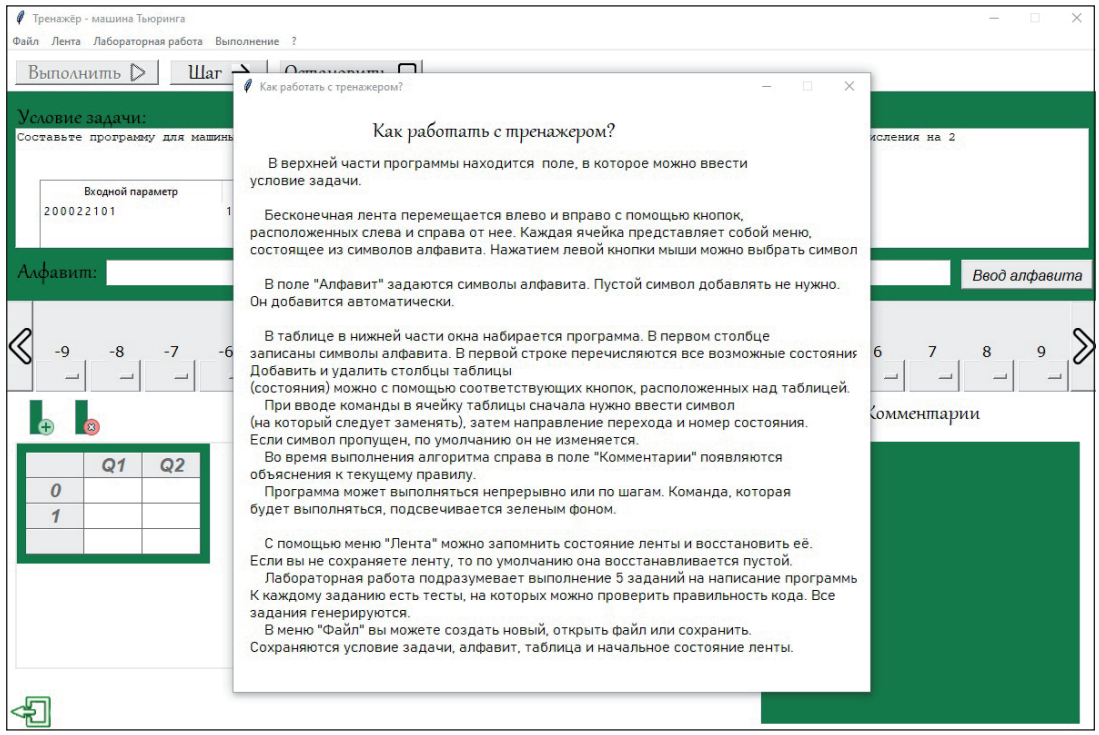


Рис. 9. Справка в блоке «Тренажер» для исполнителя «Машина Тьюринга»

можность загрузить в тренажер готовые примеры, взятые из [1, 5, 6, 11]. Программа позволяет открыть файлы только определенного формата, для данного раздела программы был создан формат Turing Examples (*.tur). Сохранять примеры в отдельный файл можно, используя подпункт *Сохранить как*. Файлы также сохраняются в формате Turing Examples (*.tur). Следует отметить, что все файлы записываются с помощью текстового формата JSON.

С помощью пункта меню *Справка* можно узнать особенности работы с тренажером (рис. 9).

Блок «Контроль» содержит тест из пяти заданий разного вида и уровня сложности. Первое и второе задания предполагают ввод ответа в отдельном поле (рис. 10). Третье задание проверяет правильность составления правила в таблице. В четвертом задании обучающимся следует выбрать один правильный ответ

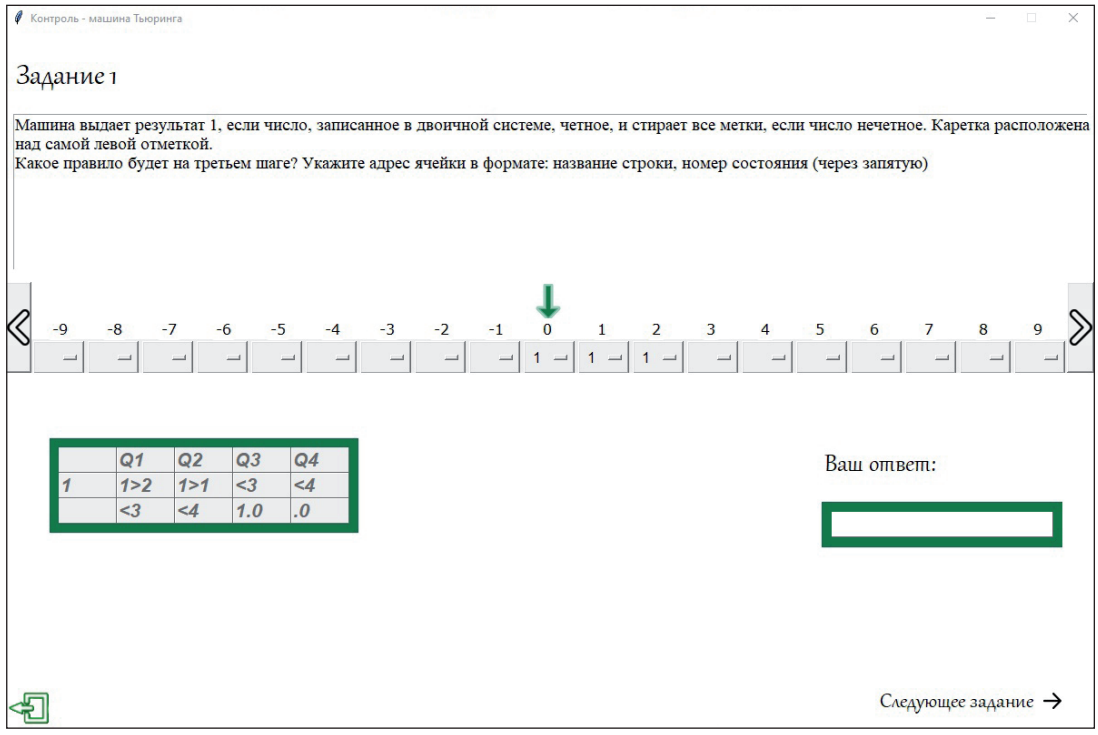


Рис. 10. Пример задания 1 блока «Контроль» для исполнителя «Машина Тьюринга»

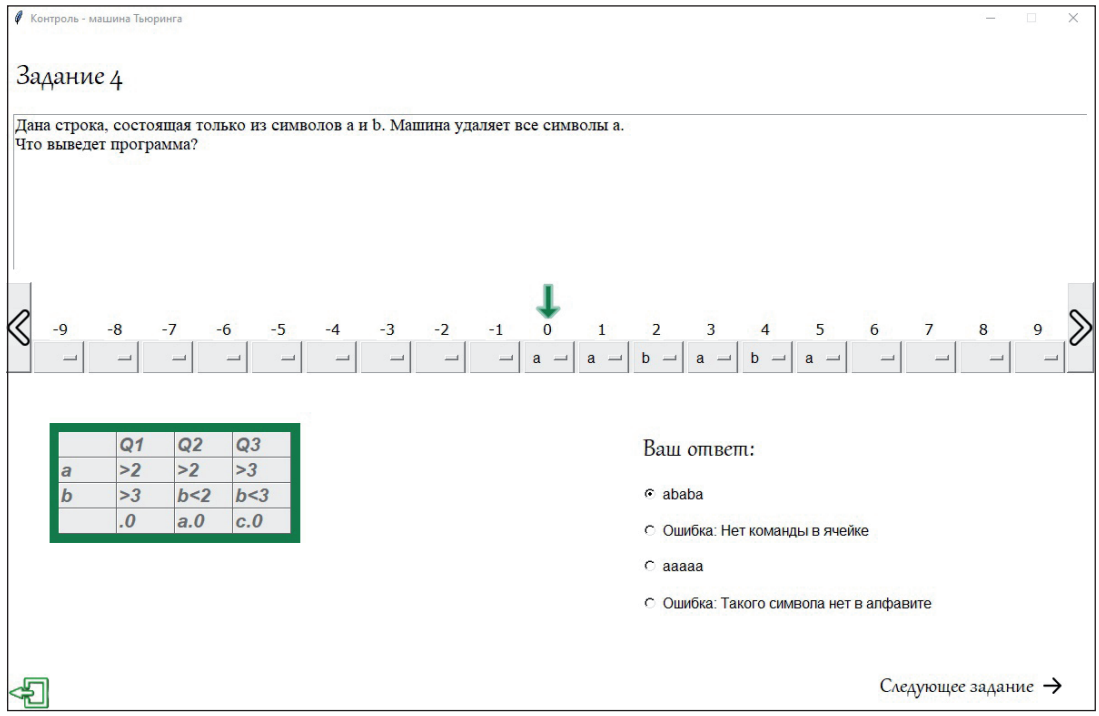


Рис. 11. Пример задания 4 блока «Контроль» для исполнителя «Машина Тьюринга»

из предложенных (рис. 11). Последнее задание является самым сложным, в нем необходимо самостоятельно выполнить алгоритм и написать результат его выполнения (рис. 12).

В конце теста пользователь может узнать свою оценку. Программа позволяет проходить тест несколько раз, однако задания каждый раз будут отличаться. Все задания выбираются случайным образом из базы. База содержит пять задач для каждого задания.

3. Блоки учебной программной среды для изучения универсального исполнителя «Машина Поста»

Раздел «Машина Поста» также содержит три блока.

В блоке «Теория» пользователь знакомится с описанием машины Поста (рис. 13).

Интерфейс **блока «Тренажер»** включает несколько управляющих кнопок, бесконечную ленту и таблицу пра-

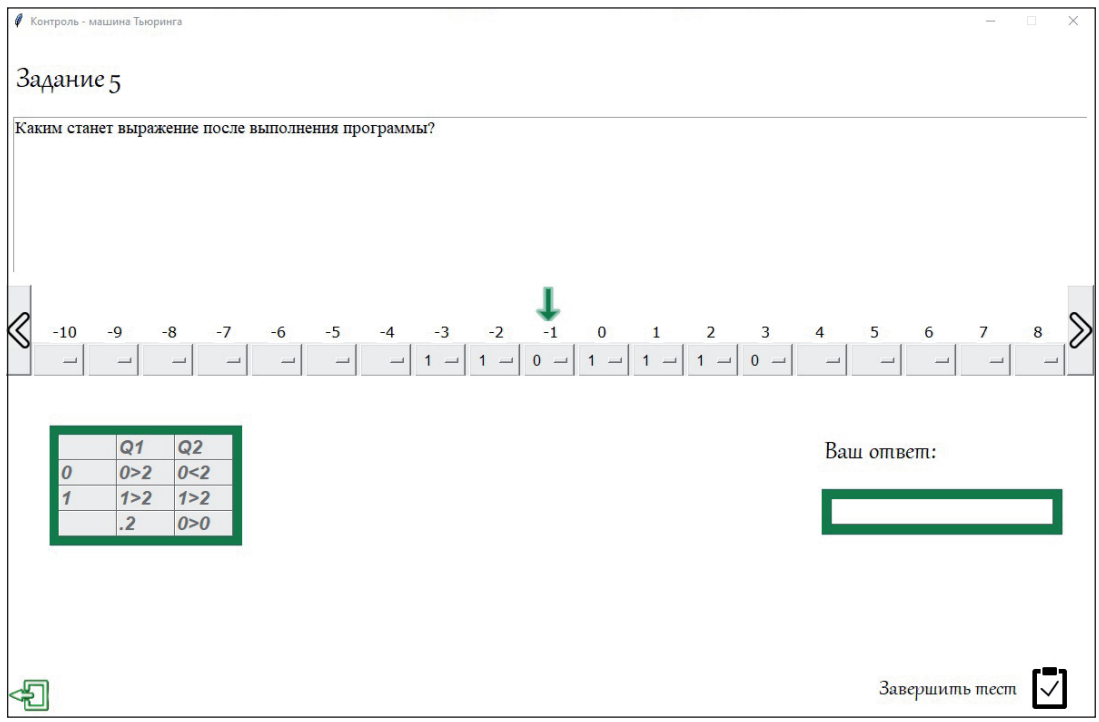


Рис. 12. Пример задания 5 блока «Контроль» для исполнителя «Машина Тьюринга»

вил. Стрелка над лентой указывает на текущий элемент (рис. 14). Работа с бесконечной лентой не отличается от работы с бесконечной лентой в машине Тьюринга, однако есть различия в алфавите. В классической (двузначной) машине Поста ячейка имеет два состояния: отмечена и не отмечена. В трехзначной машине Поста всего три символа: ноль, единица и пустой.

Пользователь может изменить алфавит, воспользовавшись соответствующими пунктами верхнего меню (рис. 15).

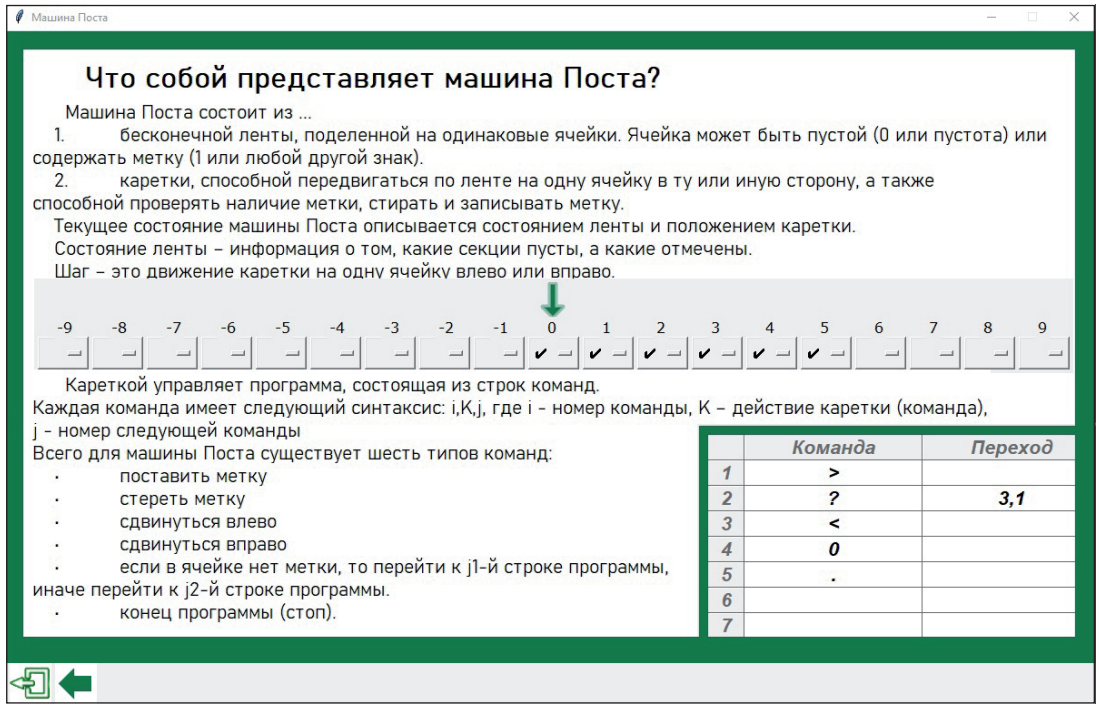


Рис. 13. Блок «Теория» для исполнителя «Машина Поста»

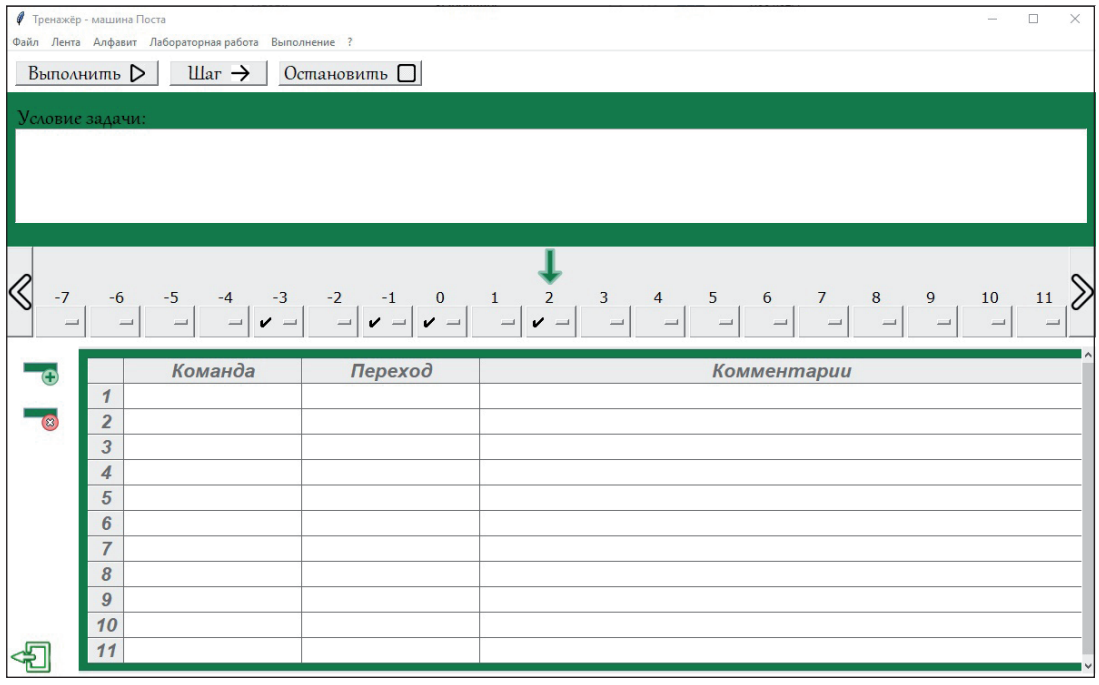


Рис. 14. Работа с бесконечной лентой в блоке «Тренажер» для исполнителя «Машина Поста»

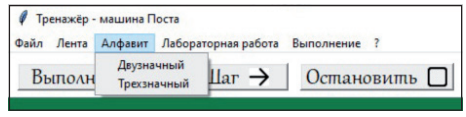


Рис. 15. Изменение алфавита в блоке «Тренажер» для исполнителя «Машина Поста»

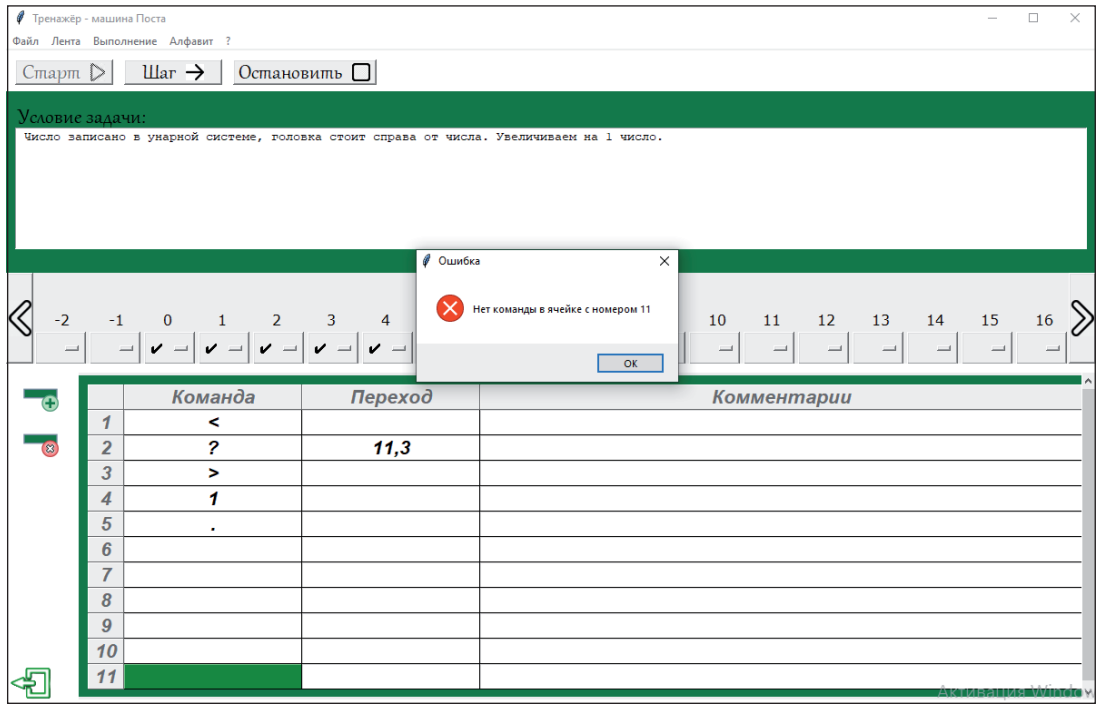


Рис. 16. Пример ошибки в блоке «Тренажер» для исполнителя «Машина Поста»

Когда пользователь выбирает другой алфавит, то бесконечная лента и таблица правил очищаются. Стоит отметить, что для каждого алфавита прописана своя валидация строк таблицы. С помощью кнопок, расположенных слева от таблицы, можно добавлять и удалять строки.

В данном тренажере также есть возможность пошагового выполнения алгоритма. Пользователь имеет возможность сохранять и восстанавливать бесконечную ленту, при этом используется формат Post Examples (*.post).

Необходимо отметить, что учебная программная среда реагирует на ошибки, связанные с неправильным заполнением правил и появлением цикла (рис. 16).

Блок «Контроль» состоит из пяти заданий разного вида и уровня сложности. Первое и второе задания предполагают ввод ответа в отдельном поле. Третье задание проверяет правильность составления правила в таблице (рис. 17). В четвертом задании обучающимся следует выбрать один правильный ответ из предложенных. Последнее задание является самым сложным, необходимо

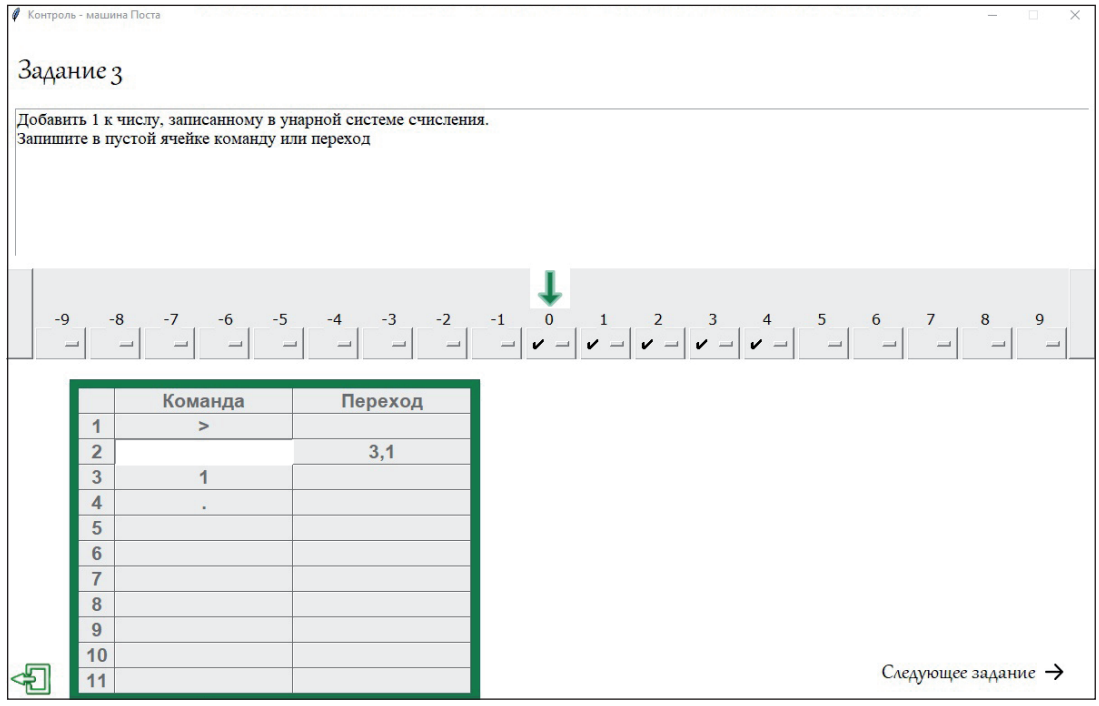


Рис. 17. Пример задания 3 блока «Контроль» для исполнителя «Машина Поста»

самостоятельно выполнить алгоритм и написать результат выполнения.

Контролирующий тест можно пройти несколько раз. Задания каждый раз выбираются случайным образом.

4. Заключение

Представленная учебная программная среда повышает эффективность изучения материала. Обучающиеся в большей степени увлечены изучением, нежели при чтении параграфа учебника или при выполнении заданий в рабочей тетради. Каждый из них может работать в удобном для него темпе. Школьники могут самостоятельно обратиться к теоретическому материалу, пройти обучение заново, потренироваться. Каждый ученик работает индивидуально, что снижает вероятность нежелательной групповой работы. Также упрощается проведение проверки полученных знаний.

Список источников

1. *Ефремов Е. Л.* Алгоритмы вычисления частичных функций: учебно-методическое пособие. Владивосток: Изд-во Дальневосточного федерального университета, 2021. 48 с.

2. *Колдунова И. Д.* Визуализация процесса обучения теории алгоритмов // Решетневские чтения. 2014. Т. 3. С. 90–93. EDN: TRPBSJ.

3. Машина Поста // Преподавание, наука и жизнь: сайт Константина Полякова. https://kpolyakov.spb.ru/prog/post.htm

4. Машина Поста // Лаборатория Паскаля. http://pyabc.ru/post.php

5. Машина Тьюринга // Преподавание, наука и жизнь: сайт Константина Полякова. https://kpolyakov.spb.ru/prog/turing.htm

6. *Пильщиков В. Н., Абрамов В. Г., Вылиток А. А., Горячая И. В.* Машина Тьюринга и алгоритмы Маркова. Решение задач. М.: МГУ, 2016. 72 с.

7. *Поляков К. Ю., Еремин Е. А.* Информатика. Углубленный уровень: учебник для 11 класса: в 2 ч. Ч. 2. М.: БИНОМ. Лаборатория знаний, 2013. 304 с.

8. *Успенский В. А., Семенов А. Л.* Теория алгоритмов: основные открытия и приложения. М.: Наука, 1987. 288 с.

9. *Фалина И. Н., Радченко Е. Л.* Изучение машины Поста в школьном курсе информатики // Информатика. 2007. № 1. С. 31–36.

10. Федеральная рабочая программа среднего общего образования. Информатика (углублённый уровень) (для 10–11 классов образовательных организаций) // Единое содержание общего образования. https://edsoo.ru/wp-content/uploads/2023/08/22_ФРП_Информатика-10-11-классы_угл.pdf

11. *Чухнов А. С., Поздняков С. Н.* Конструктивные задачи по дискретной математике: сравнительный анализ экзамена с компьютером и без компьютера // Компьютерные инструменты в образовании. 2022. № 1. С. 57–84. EDN: YEKZVK. DOI: 10.32603/2071-2340-2022-1-57-84

12. Эмулятор машины Тьюринга // Сообщество Programforyou. https://programforyou.ru/calculators/turing-machine-emulator

DOI: 10.32517/2221-1993-2023-22-6-41-47

Е. А. Склярова
Средняя общеобразовательная школа № 41 имени В. В. Сизова, г. Курск, Россия

УРОК-ИГРА «В ПОИСКАХ ЗАТЕРЯННЫХ БАЙТОВ»*

Аннотация

В статье описывается опыт разработки и проведения урока-игры по информатике, на котором в форме командного соревнования повторяются и закрепляются первоначальные знания учащихся о компьютере, видах информации, действиях с информацией, а также активизируется логическое мышление. Представлены все этапы командной игры с формированием у учащихся универсальных учебных действий и достигаемых предметных результатов обучения. Содержание урока построено таким образом, чтобы, пройдя все конкурсы, учащиеся смогли закрепить предметные компетенции и продолжить формировать метапредметные. Ход урока основан на том, что обучающиеся поочередно выполняют задания, работают в команде. Каждый этап урока сопровождается демонстрацией заданий и вопросов на интерактивной доске с музыкальным сопровождением, что обеспечивает положительный настрой на игру. Жюри выставляет баллы командам, используя оценочные листы. По завершении урока-игры жюри озвучивает результаты, командам присуждаются места в соревновании. Педагогом заранее подготавливаются наградные материалы в виде грамот и медалей.

Ключевые слова: информатика, урок-игра, информация, компьютер, файл, объект.

Интеллектуально-познавательная игра по информатике «В поисках затерянных байтов» представляет собой нестандартный урок, где в игровой форме учащиеся повторяют и систематизируют теоретический материал по информатике, закрепляют полученные знания. Урок содействует развитию сообразительности, смекалки учащихся, воспитанию коммуникативных навыков, организует работу в команде, учит слушать товарищей.

При проведении игры используется классно-урочная форма работы (внеурочная деятельность), рассчитанная на 40–45 минут (один урок).

Тема урока: В поисках затерянных байтов.

Цели урока:

- актуализировать, обобщить и систематизировать теоретические знания и практические умения учащихся по темам «Информация», «Компьютер», «Информационные процессы»;
- в игровой форме закрепить ранее изученный материал указанных тем.

Задачи урока:

- *дидактическая:* систематизация и углубление знаний учащихся по информатике;
- *развивающие:* развитие творчества, сообразительности, смекалки учащихся, их познавательной активности;

* Материалы к статье можно скачать на сайте ИНФО: http://infojournal.ru/journals/school/school_06-2023/

Контактная информация

Склярова Елена Александровна, учитель информатики, средняя общеобразовательная школа № 41 имени В. В. Сизова, г. Курск, Россия; *адрес:* 305025, Россия, г. Курск, Магистральный пр-д, д. 20; *e-mail:* kursk41@yandex.ru

E. A. Sklyarova
School 41, Kursk, Russia

LESSON-GAME "IN SEARCH OF LOST BYTES"

Abstract

The article describes the experience of developing and conducting a lesson-game in informatics, in which, in the form of a team competition, students' initial knowledge about the computer, types of information, actions with information is repeated and consolidated, and logical thinking is also activated. All stages of team game are presented with the formation of universal educational actions in students and the achieved subject learning results. The content of the lesson is structured in such a way that, after passing all competitions, students will be able to consolidate subject competencies and continue to develop meta-subject competencies. The course of the lesson is based on the fact that students take turns performing tasks and working in a team. Each stage of the lesson is accompanied by a demonstration of tasks and questions on an interactive board with musical accompaniment, which ensures a positive mood for the game. The jury assigns points to teams using score sheets. At the end of the lesson-game, the jury announces the results, and the teams are awarded places in the competition. The teacher prepares award materials in the form of certificates and medals in advance.

Keywords: informatics, lesson-game, information, computer, file, object.

- *воспитательные:* воспитание чувства коллективизма, ответственности, самоконтроля; формирование/приобретение интереса к предмету «Информатика» [3].

Планируемые образовательные результаты:

личностные:
- понимание роли фундаментальных знаний как основы современных информационных технологий;
- способность увязать учебное содержание с собственными жизненным опытом и знаниями;
- освоение обучающимися социального опыта, основных социальных ролей, соответствующих ведущей деятельности возраста, норм и правил общественного поведения, форм социальной жизни в группах и сообществах, в том числе в виртуальном пространстве;

метапредметные:
- навыки анализа различных объектов;
- умение определять понятия, создавать обобщения, устанавливать аналогии, классифицировать, самостоятельно выбирать основания и критерии для классификации, устанавливать причинно-следственные связи, строить логические рассуждения, делать умозаключения и выводы;
- владение основами самоконтроля, самооценки, принятия решений и осуществления осознанного выбора в учебной и познавательной деятельности;

предметные:
- умение называть основные компоненты персональных компьютеров и мобильных устройств, объяснять их назначение;
- понимание содержания понятий «программное обеспечение», «операционная система», «файл», «информатика», «информация» [6].

Тип урока: урок обобщения и закрепления учебного материала.

Вид урока: урок проводится в форме командного соревнования.

Возраст учащихся: пятый класс.

Методы работы на уроке:
- словесный;
- наглядный.

Формы работы на уроке:
- фронтальная;
- групповая — в группах по 8–10 человек.

Оснащение урока:
- интерактивная (сенсорная) доска;
- мультимедийная презентация, подготовленная в PowerPoint;
- листы с заданиями конкурсов для команд;
- медали, грамоты.

Оформление доски: на доске написан эпиграф: «Кто владеет информацией, тот владеет миром. Н. М. Ротшильд».

План урока.
- Конкурс 1. Порт Приветствие.
- Конкурс 2. Бухта Загадка.
- Конкурс 3. Порт Вассерман.
- Конкурс 4. Остров Логические приключения.
- Конкурс 5. Остров Ребусово.
- Подведение итогов урока-игры. Награждение команд.

Ход урока

Урок сопровождается демонстрацией слайдов мультимедийной презентации*.

Ведущий. Дорогие ребята! Сегодня вас ждет удивительное информационное путешествие по волнам данных в море информации к берегам знаний в поисках затерянных байтов. Все ваше удивительное путешествие по морю от бухт к причалам вас будет сопровождать опытный проводник — Елена Александровна Склярова. Желаем вам удачи!

Конкурс 1. Порт Приветствие

Учитель. Уважаемые команды, прежде чем мы отправимся в далекое путешествие, нам нужно познакомиться с командами и проследовать в порт Приветствие (рис. 1).

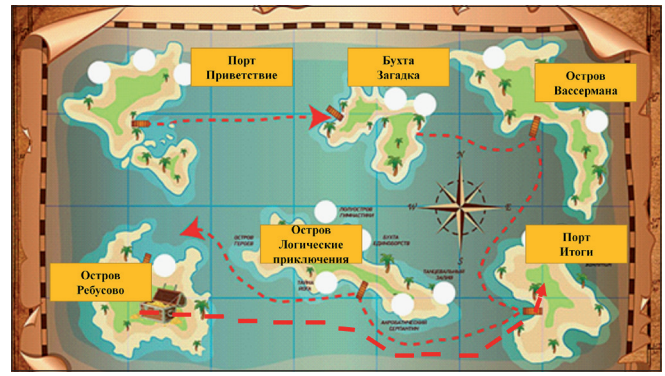


Рис. 1. Слайд с картой маршрута путешествия

Капитаны команд поочередно выходят в центр кабинета, знакомят присутствующих с членами своих команд, озвучивают название команды; участники каждой команды хором произносят ее девиз (рис. 2).

Рис. 2. Слайд-приветствие одной из команд

В ходе игры члены жюри оценивают участие команд в конкурсах и выставляют баллы в оценочные листы.

* Презентацию можно скачать на сайте ИНФО: http://infojournal.ru/journals/school/school_06-2023/

Конкурс 2. Бухта Загадка

Учитель. Вам отлично покорился порт Приветствие, и сейчас вы доплыли до бухты Загадка.

Вам предстоит по очереди назвать цифру с вопросом (рис. 3).

Когда на экране появится вопрос, вы должны ответить на него в течение 15 секунд (рис. 4). Если у вас не будет верного ответа, то вопрос переходит следующей команде, вам не присваивается ни одного балла, а команда, ответившая правильно, получает 10 баллов. И так, пока не закончатся все вопросы на данном этапе.

В таблице 1 представлены все вопросы конкурса «Бухта Загадка» [8, 9].

Учитель. Отлично! Вы прекрасно справляетесь с преградами на пути к заветной цели. Но сейчас у вас очень сложное задание, и, преодолев его, вы станете ближе к цели на целый порт.

Рис. 3. Слайд с номерами вопросов конкурса «Бухта Загадка»

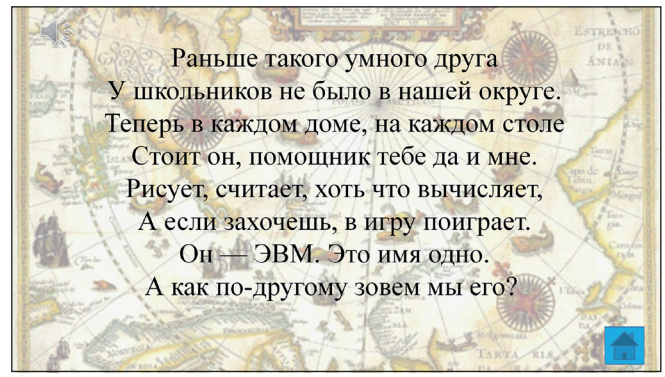


Рис. 4. Слайд с вариантом вопроса конкурса «Бухта Загадка»

Таблица 1

Вопросы для конкурса «Бухта Загадка»

№ п/п	Формулировка вопроса	Ожидаемый ответ
1	Раньше такого умного друга У школьников не было в нашей округе. Теперь в каждом доме, на каждом столе Стоит он, помощник тебе да и мне. Рисует, считает, хоть что вычисляет, А если захочешь, в игру поиграет. Он — ЭВМ. Это имя одно. А как по-другому зовем мы его?	Компьютер
2	Бывает струйный, лазерный бывает. Его всегда печатать заставляют. Он на бумагу распечатает что нужно. Печатник этот всем нам очень нужен.	Принтер
3	На компьютерном столе Помогает она мне. Колесиком и кнопкой Я управляю ловко.	Мышка
4	Компьютер будет молчалив, Коль нет с ним рядом дев таких. А если есть, он говорит, Поет, играет и пищит. Стоят над ним в сторонке Близняшки две —	Колонки
5	Корпус компьютера. В нем то хранится, Что компьютеру пригодится. Корпус из пластика, стали, стекла, В нем материнская плата жила, А также процессор, ОЗУ, дисковод... Что это за корпус, скажите, народ?	Системный блок

Окончание табл. 1

№ п/п	Формулировка вопроса	Ожидаемый ответ
6	Много кнопок, цифры, буквы, Enter, Shift, F2, F5.	Клавиатура
	На английском и на русском Можно, дети, с ней писать. Пальцами стучу по ней. Кто она? Скажи скорей!	
7	На нем информацию можно читать, Картинки смотреть и в игры играть.	Монитор
8	Всемирная сеть иль, еще, паутина, Найдешь в ней про все — про людей, про машины. Каких только сведений разных в ней нет! Зовется она, знаешь ты, ...	Интернет
9	Если мой компьютер «заболеет», Вылечить его я сам сумею. Не боюсь вредоносных программ, Повредить ничего им не дам. Как вредители те называются, Что заразны и вмиг размножаются?	Вирусы

Конкурс 3. Порт Вассерман

Учитель. А вот и порт Вассерман [2, 4, 5]. Дорогие друзья, вам предстоит отвечать на вопросы из области информатики в разных категориях по принципу телевизионной игры «Своя игра» (неоднократным победителем которой является Анатолий Александрович Вассерман, по имени которого и назван данный конкурс). Выбирайте категорию и стоимость вопроса (рис. 5, 6).

Внимание! Если вы даете верный ответ на выбранный вопрос, то получаете такое количество баллов, сколько стоит этот вопрос. Если вы даете неверный ответ, на вопрос отвечает другая команда, и в случае удачного ответа она получает заявленные баллы. Если вам попадается вопрос «кот в мешке», вы выбираете, какой из команд-соперниц отдать вопрос. Если команда справится с заданием, то она получает заявленное количество баллов.

В таблице 2 представлены все вопросы конкурса «Порт Вассерман».

Рис. 5. Слайд для выбора вопросов конкурса «Порт Вассерман»

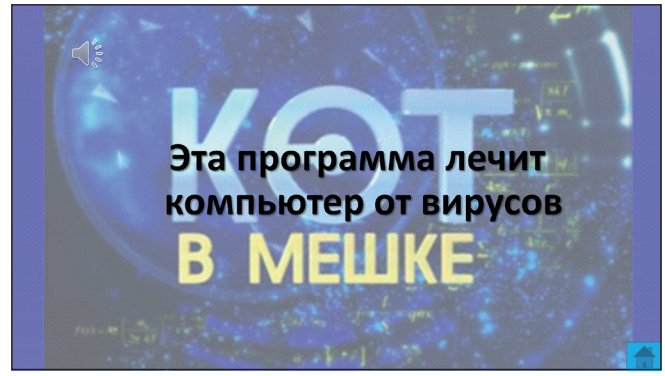


Рис. 6. Слайд с вариантом вопроса конкурса «Порт Вассерман»

Таблица 2

Вопросы конкурса «Порт Вассерман»

Категория вопроса	Баллы	Формулировка вопроса
Компьютер	50	*Кот в мешке.* Устройство, предназначенное для просмотра изображений, фильмов, текста
	100	С их помощью мы можем слушать звуки

Окончание табл. 2

Категория вопроса	Баллы	Формулировка вопроса
	150	С ее помощью можно набирать слова и цифры
	200	Она помогает управлять компьютером. С ее помощью можно рисовать
	250	В этой части компьютера расположены все микросхемы и важные устройства
	300	*Кот в мешке.* Эта программа лечит компьютер от вирусов
Файлы и папки	50	Файл — это ...
	100	Из чего состоит имя файла?
	150	На что указывает расширение файла?
	200	*Кот в мешке.* Какие символы нельзя включать в имя файла и папки?
	250	Назовите точное определение понятия «исполнитель»
	300	Какие операции можно совершать с файлами? Каких действий следует избегать при работе с файлами?
Информация	50	Информация — это ...
	100	*Кот в мешке.* Перечислите виды информации по способам записи
	150	Назовите минимальную единицу измерения информации
	200	Дата рождения, номер телефона — какой это вид информации?
	250	Назовите известные вам источники информации
	300	*Кот в мешке.* Бóльшую часть информации человек воспринимает с помощью именно этого органа
Объекты	50	Объект — это ...
	100	Назовите два формальных исполнителя
	150	Приведите примеры объектов-предметов
	200	*Кот в мешке.* Что такое естественная классификация?
	250	Алгоритм — это ...
	300	Изобразите графически при помощи кругов Эйлера пересечение и приведите примеры

Конкурс 4. Остров Логические приключения

Учитель. Ребята, я вас поздравляю! Вы идете верным курсом. Наконец мы добрались до острова Логические приключения (рис. 7).

Вам предстоит по очереди отвечать на логические вопросы (рис. 8).

За каждый верный ответ вам будут присуждены 10 баллов. Вы готовы? Тогда мы начинаем.

Рис. 7. Слайд-заставка конкурса «Остров Логические приключения»

Вы опередили лыжника, который находился на второй позиции. Какое место теперь занимаете вы?

Рис. 8. Слайд с вариантом вопроса конкурса «Остров Логические приключения»

Таблица 3

Вопросы конкурса «Остров Логические приключения» [1]

№ п/п	Формулировка вопроса	Ожидаемый ответ
1	Вы опередили лыжника, который находился на второй позиции. Какое место теперь занимаете вы?	Второе
2	Из какой посуды не едят?	Из пустой
3	Что с пола за хвост не поднимешь?	Клубок ниток
4	В 12-этажном доме есть лифт. На первом этаже живут всего два человека, от этажа к этажу количество жильцов увеличивается вдвое. Какая кнопка в лифте этого дома нажимается чаще других?	Кнопка первого этажа
5	Ученые придумали прибор, с помощью которого можно проходить сквозь стены. Как ученые назвали данное устройство?	Дверь
6	У арфы их четыре, у домбры и у гитары — шесть. А сколько их у пианино?	Семь. Букв
7	Какая деталь книжного шкафа состоит из половины согласной буквы?	Пол Ка

В таблице 3 представлены вопросы конкурса «Остров Логические приключения».

Конкурс 5. Остров Ребусово

Учитель. Дорогие путешественники, вы на финишной прямой — мы добрались до острова Ребусово (рис. 9)! И это очень позитивная новость.

Сейчас на экране вам будут последовательно представлены ребусы. Вам предстоит разгадать каждый ребус, записать ответ на листе и по моей команде поднять лист с ответом.

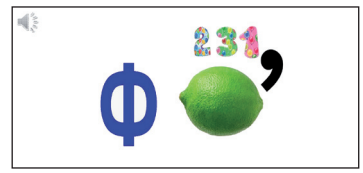
За каждый разгаданный ребус команде присваивается 50 баллов. После демонстрации ответов участников на экран выводится верный ответ [4, 10].

Рис. 9. Слайд-заставка конкурса «Остров Ребусово»

В таблице 4 представлены вопросы конкурса «Остров Ребусово».

Таблица 4

Вопросы конкурса «Остров Ребусово»

№ п/п	Ребус	Ответ	№ п/п	Ребус	Ответ
1		Дисплей	4		Файл
2		Пиксель	5		Клавиатура
3		Компьютер	6		Процессор

Подведение итогов урока-игры. Награждение команд

После завершения последнего конкурса команды отдыхают. Появляется последний слайд презентации «Подведение итогов», на котором звучит праздничная мелодия (рис. 10), в это время жюри совещается и подсчитывает баллы. Один из членов жюри озвучивает итоги игры.

Рис. 10. Слайд подведения итогов

Учитель. Вот мы и добрались до крайней точки нашего маршрута — порта Итоги и нашли наши байты. Обобщив и повторив материал по информатике, мы определили победителей игры. Ими стала команда ... (*называет команду-победителя*).

Награждаются победители и призеры игры, вручаются грамоты и медали.

Список источников

1. *Авдошин С. М., Максименкова О. В., Ахметсафина Р. З.* Информатика. Логика и алгоритмы. Эффективные методы решения задач. М.: Просвещение, 2013. 174 с.

2. *Босова Л. Л., Босова А. Ю.* Информатика. 5–6 классы: методическое пособие. 2-е изд., перераб. М.: БИНОМ. Лаборатория знаний, 2017. 384 с.

3. *Босова Л. Л., Босова А. Ю.* Информатика. Программа для основной школы. 5–6 классы. 7–9 классы. М.: БИНОМ. Лаборатория знаний, 2015. 88 с.

4. *Босова Л. Л., Босова А. Ю.* Информатика: учебник для 5 класса. М.: БИНОМ. Лаборатория знаний, 2020. 184 с.

5. *Босова Л. Л., Босова А. Ю.* Информатика: учебник для 6 класса. М.: БИНОМ. Лаборатория знаний, 2020. 213 с.

6. *Босова Л. Л., Босова А. Ю.* Примерная рабочая программа. 5–6 классы. М.: БИНОМ. Лаборатория знаний, 2016. 22 с.

7. *Горячев А. В., Суворова Н. И.* Логика и алгоритмы. М.: Баласс, 2015. 32 с.

8. *Куличкова А. Г.* Информатика. 2–11 классы. Внеклассные мероприятия. Неделя информатики. Волгоград: Учитель, 2011. 152 с.

9. *Рудченко Т. А.* Методические рекомендации. 5–6 классы. М.: Просвещение, 2022. 204 с.

10. *Семенов А. Л., Рудченко Т. А.* Информатика. 5 класс. М.: Просвещение, 2022. 145 с.

DOI: 10.32517/2221-1993-2023-22-6-48-53

А. А. Францкевич
Белорусский государственный педагогический университет имени Максима Танка, г. Минск, Беларусь
О. Ю. Простак
Гимназия № 2, г. Солигорск, Минская область, Беларусь

ОПЫТ ИСПОЛЬЗОВАНИЯ ВИЗУАЛИЗИРОВАННОЙ СРЕДЫ ПРОГРАММИРОВАНИЯ SCRATCH ДЛЯ ОБУЧЕНИЯ ОСНОВАМ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ В VI—VIII КЛАССАХ ШКОЛ БЕЛАРУСИ

Аннотация

«Алгоритмизация и программирование» — важная содержательная линия школьного курса информатики. Однако многие ученики испытывают трудности в ее изучении из-за недостаточного развития абстрактного мышления и отсутствия практического опыта программирования. Визуальный язык программирования Scratch может стать пропедевтическим инструментом изучения данной содержательной линии, позволяющим преодолеть указанные трудности и сделать изучение программирования более доступным и интересным для школьников.

В статье представлены примеры заданий для выполнения в визуализированной среде программирования Scratch, которые предлагаются учащимся VI—VIII классов школ Беларуси для освоения содержательной линии «Основы алгоритмизации и программирования» учебного предмета «Информатика». Каждое задание представлено на трех уровнях: репродуктивном, продуктивном и творческом.

Материал будет полезен учителям информатики, работающим с учащимися, которым трудно осваивать понятия алгоритмизации и программирования с использованием текстовых языков программирования (Pascal, Python, C++ и т. д.).

Ключевые слова: визуализированная среда программирования, визуальный язык программирования, Scratch, основы алгоритмизации и программирования, школьный курс информатики, основное общее образование.

Контактная информация

Францкевич Александр Александрович, канд. пед. наук, декан физико-математического факультета, Белорусский государственный педагогический университет имени Максима Танка, г. Минск, Беларусь; *адрес:* 220030, Республика Беларусь, г. Минск, ул. Советская, д. 18; *e-mail:* frantskevich@live.ru

Простак Оксана Юрьевна, учитель информатики, гимназия № 2, г. Солигорск, Минская область, Беларусь; *адрес:* 223700, Республика Беларусь, Минская область, г. Солигорск, ул. Богомолова, д. 22а; *e-mail:* oksana.prostak14@gmail.com

A. A. Frantskevich
Belarusian State Pedagogical University named after Maxim Tank, Minsk, Belarus
O. Yu. Prostak
Gymnasium 2, Soligorsk, Minsk Region, Belarus

EXPERIENCE OF USING THE VISUALIZED PROGRAMMING ENVIRONMENT SCRATCH FOR TEACHING THE BASICS OF ALGORITHMIZATION AND PROGRAMMING IN GRADES VI—VIII IN SCHOOLS OF BELARUS

Abstract

"Algorithmization and programming" is an important content line of the school informatics course. However, many students have difficulty learning it due to insufficient development of abstract thinking and lack of practical experience in programming. The visual programming language Scratch can become a propaedeutic tool for studying this content line, allowing one to overcome these difficulties and make the study of programming more accessible and interesting for schoolchildren.

The article presents examples of tasks to be performed in the visualized programming environment Scratch, which are offered to students of the VI—VIII grades of schools in Belarus to master the content line "Fundamentals of algorithmization and programming" of the academic subject "Informatics". Each task is presented at three levels: reproductive, productive and creative.

The material will be useful to informatics teachers working with students who find it difficult to master the concepts of algorithmization and programming using text-based programming languages (Pascal, Python, C++, etc.).

Keywords: visualized programming environment, visual programming language, Scratch, basics of algorithmization and programming, school informatics course, basic general education.

1. Введение

В школах Беларуси учебный предмет «Информатика» изучается с VI класса. Занятия по программированию в рамках содержательной линии «Основы алгоритмизации и программирования» начинаются в последней четверти VI класса с изучения среды программирования PascalABC.NET и языка программирования Pascal. Многим ученикам, тем, которые до этого не сталкивались с программированием, сложно воспринимать и анализировать информацию на уроках. Для помощи таким учащимся были введены два факультатива с использованием визуального языка программирования Scratch.

Пропедевтический факультатив «Пропедевтика основ алгоритмизации и программирования в визуальной среде программирования SCRATCH» для V—VI классов [11] позволяет учащимся познакомиться с алгоритмизацией и программированием за год до начала изучения соответствующей содержательной линии в школьном курсе информатики.

Для учеников VII—VIII классов разработан факультатив «Основы алгоритмизации и программирования в визуальной среде программирования SCRATCH» [10], который позволяет учителю углублять знания учащихся, полученные ими на уроках информатики.

Scratch — это визуальный язык программирования, разработанный специально для обучения школьников и студентов основам алгоритмизации и программирования [16]. Он имеет простой и понятный интерфейс, позволяет создавать анимацию, игры и интерактивные проекты без необходимости написания кода. Визуализированная среда программирования предлагает множество готовых блоков и шаблонов, которые можно использовать для создания собственных проектов, что дает возможность разнообразить процесс выполнения учебных заданий и делает Scratch удобным инструментом для обучения алгоритмизации и программированию.

Стоит отметить, что *под **визуальным языком программирования** мы понимаем язык программирования, который состоит из визуальных выражений (блоков, скриптов) и установления связей между ними* [14]. *Под **визуализированной средой программирования** понимается среда программирования, в которой составление программы происходит в интерактивном режиме при помощи использования визуального языка программирования* [13].

В данной статье мы рассмотрим возможности изучения основ алгоритмизации и программирования в VI—VIII классах с использованием визуализированной среды программирования Scratch.

2. Требования к учащимся в усвоении содержательной линии «Основы алгоритмизации и программирования»

Рассмотрим **основные цели и задачи изучения учебного предмета «Информатика» в рамках содержательной линии «Основы алгоритмизации и программирования»:**

- формирование компьютерной грамотности;
- развитие логического и алгоритмического мышления;
- формирование теоретических знаний и практических умений в области алгоритмизации и программирования.

Основные требования к учащимся в усвоении содержательной линии «Основы алгоритмизации и программирования» за VI—VIII классы следующие: учащиеся должны знать понятия алгоритма и исполнителя алгоритма, способы записи алгоритмов, алгоритмические конструкции «ветвление», «повторение», понятие переменной [7–9].

Проведенный нами **анализ ряда работ отечественных и зарубежных авторов по программированию в среде Scratch** ([1, 2, 6, 12, 15] и др.) позволил выделить отдельные проекты, которые можно применить на факультативных занятиях по информатике. Но среди рассмотренных работ нет ни одной, которую можно было бы взять за системную основу при проведении факультативных занятий. Кроме того, терминологический аппарат, используемый в данных изданиях, отличается от принятого в учебном предмете «Информатика». В связи с этим *актуальной становится разработка дидактических материалов для сопровождения факультативных занятий в белорусской школе, которые соответствовали бы требованиям содержательной линии «Основы алгоритмизации и программирования»*.

3. Пример задания в Scratch для изучения основ алгоритмизации и программирования в VI классе

Нами **разработана база заданий, соответствующих содержательной линии «Основы алгоритмизации и программирования».**

Учащимся, в зависимости от результатов их учебной деятельности на первых занятиях в среде Scratch, **предлагается одно и то же задание, но на разных уровнях усвоения знаний:**

- ***на репродуктивном уровне*** учащимся предоставляются:
  - формулировка задания;
  - изображение готовой программы;
  - краткое решение в текстовом формате;
  - готовые скрипты, которые необходимо собрать в алгоритм;
- ***на продуктивном уровне*** учащимся предлагаются:
  - формулировка задания;
  - словесное описание скриптов;
  - краткое решение в текстовом формате;
- ***на творческом уровне*** учащимся предоставляются:
  - формулировка задания;
  - изображение готовой программы.

Рассмотрим первое задание «Слон», при выполнении которого учащиеся изучат понятия алгоритма и исполнителя алгоритма. С помощью расширения «Перо» в визуализированной среде программирования Scratch можно реализовать подобие исполнителя Чертежник среды PascalABC.NET, которую учащиеся изучают в VI классе, используя учебное пособие «Информатика» [5].

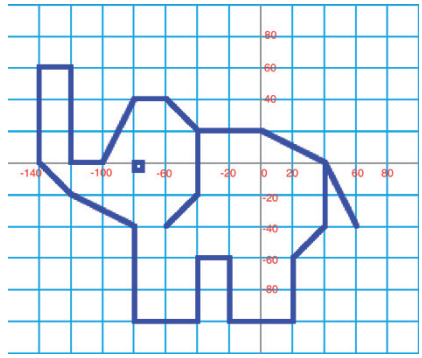
Задание «Слон».

Общая формулировка:

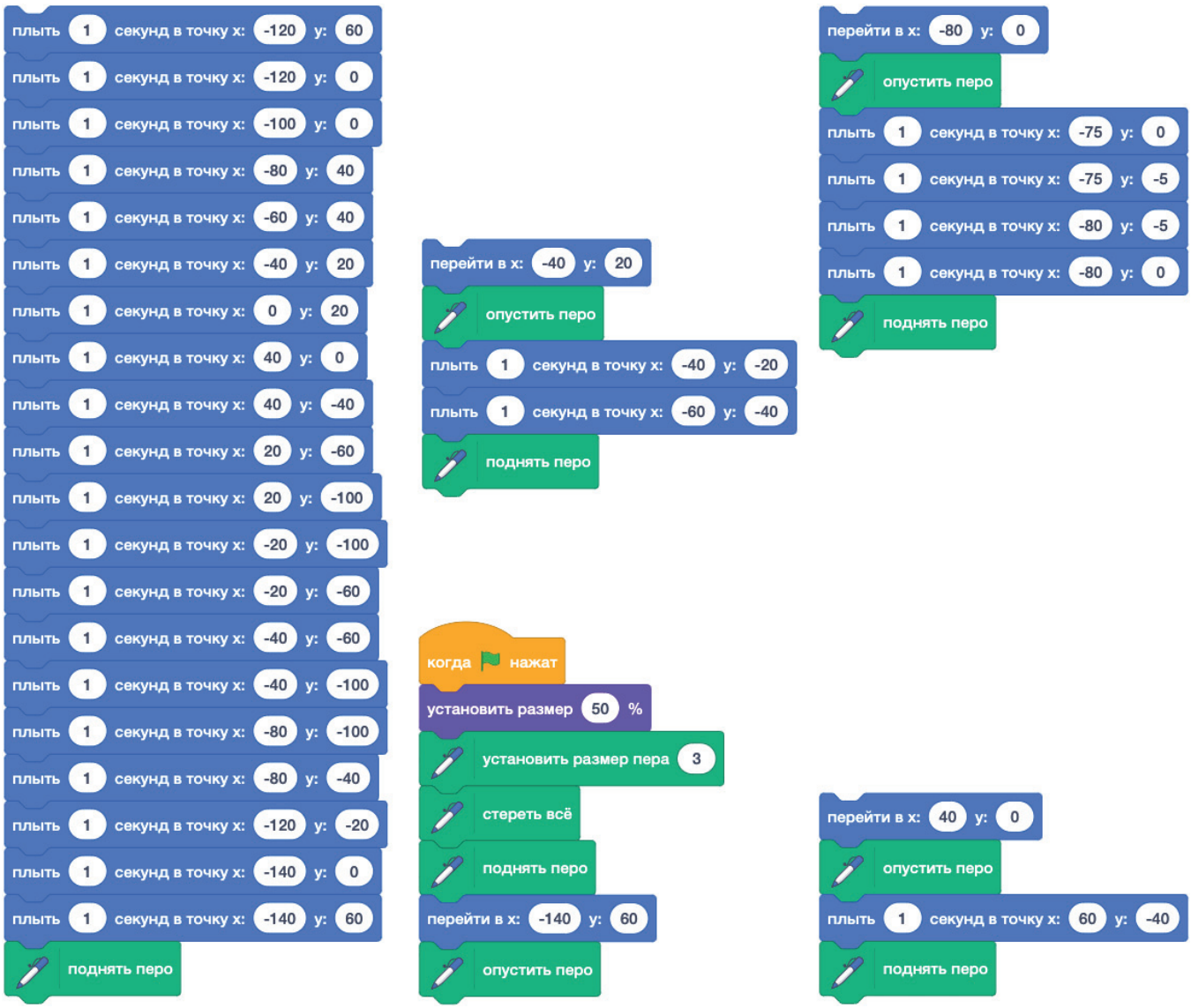
- Используя расширение *Перо* визуализированной среды программирования Scratch, изобразить представленного на рисунке слона (рис. 1, *а*).

На репродуктивном уровне учащимся предлагаются готовые блоки скриптов программы (рис. 1, *б*), которые необходимо соединить в правильном порядке.

На продуктивном уровне учащимся дополнительно к рисунку задания предоставляются справочный материал и краткое словесное описание решения.



а



б

Рис. 1. Задание «Слон»

Справочный материал может быть представлен, например, в таком виде:

- используй скрипты палитры «Перо»: установить размер пера; стереть все; поднять перо; опустить перо;
- используй скрипты палитры «Движение»: перейти в x y; плыть 1 секунд в точку x y;
- используй скрипты палитры «Внешний вид»: установить размер; команды *События*: когда флаг нажат.

Пример словесного описания решения:

- Для решения задачи выбираем спрайт Pencil и фон Xy-grid-20px. Устанавливаем центр карандаша во вкладке *костюмы*, задаем его размер 50 % и начальное положение. Переходим из точки в точку согласно заданным координатам.

На творческом уровне формулировка задания выглядит следующим образом:

- В визуализированной среде программирования Scratch изобразить слона по указанным на рисунке координатам.

4. Пример задания в Scratch для изучения основ алгоритмизации и программирования в VII классе

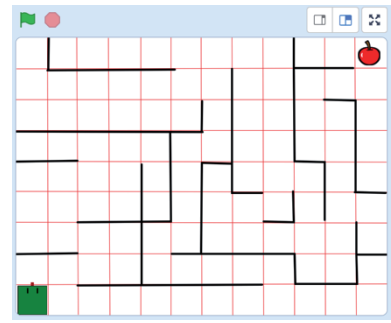
Рассмотрим задание «Лабиринт», которое можно соотнести с исполнителем Робот среды PascalABC.NET. Изучение этого исполнителя является ключевым в VII классе в учебном пособии «Информатика» [4]. Визуализированная среда программирования Scratch не позволяет добавить расширение, которое было бы похоже на Робота из Pascal, как это было с расширением «Перо» в задании «Слон». Но мы можем предложить учащимся разработать своего исполнителя Робот с помощью создания «своих» блоков скриптов (рис. 2).

Рис. 2. Блоки для исполнителя Робот в среде Scratch

Задание «Лабиринт».

Общая формулировка:

- Нарисовать свой фон в виде лабиринта. Нарисовать свой спрайт, который будет выполнять движение по лабиринту с помощью блоков исполнителя Робот. Добавить спрайт Apple, который исчезнет при соприкосновении с нарисованным спрайтом, при этом проиграется звук Chomp (рис. 3, *а*).



а

б

Рис. 3. Задание «Лабиринт»

На репродуктивном уровне учащимся предлагаются готовые блоки скриптов программы (рис. 3, *б*), которые необходимо соединить в правильном порядке.

На продуктивном уровне учащимся предлагается *справочный материал*:

- используй скрипты исполнителя Робот: вправо; влево; вверх;
- используй скрипты палитры «Движение»: повернуться в направлении ...; перейти в x ... y ...;
- используй скрипты палитры «События»: когда флаг нажат; когда клавиша «стрелка вниз» нажата; когда клавиша «стрелка влево» нажата; когда клавиша «стрелка вправо» нажата; когда клавиша «стрелка вверх» нажата;
- используй скрипты палитры «Управления»: ждать до ...; если ... то ...; повторить ... раз; ждать ... секунд;
- используй скрипты палитры «Звук»: играть звук до конца;
- используй скрипты палитры «Сенсоры»: касается.

Пример словесного описания решения:

- Выбираем спрайт Apple из категории «Еда» библиотеки. Устанавливаем его положение в верхнем правом углу. Второй спрайт рисуем и устанавливаем в левом нижнем углу. Рисуем фон согласно условию. Создаем скрипты для спрайтов, используя справочный материал.

На творческом уровне учащимся предлагается только общая формулировка задачи.

5. Пример задания в Scratch для изучения основ алгоритмизации и программирования в VIII классе

Следующее задание связано с созданием игры «Собери яйца». При выполнении этого задания учащиеся знакомятся с алгоритмическими конструкциями «ветвление» и «повторение», с понятием переменной. Данные понятия изучаются по учебному пособию «Информатика» в VIII классе [3].

Задание «Собери яйца».

Общая формулировка:

- Создать игру «Собери яйца», суть которой состоит в следующем. Яйца падают сверху вниз из разных положений относительно оси *Ox*. Нужно собрать 10 яиц в корзину. Движение корзины осуществляется при помощи стрелок «вправо» и «влево». Создать три жизни (т. е. можно пропустить менее трех яиц). Игра заканчивается в случае победы или проигрыша. Вывести соответствующее сообщение. На рисунке 4, *а* показана одна из возможных реализаций игры.

На репродуктивном уровне учащимся предлагаются готовые блоки скриптов программы (рис. 4, *б*), которые необходимо соединить в правильном порядке.

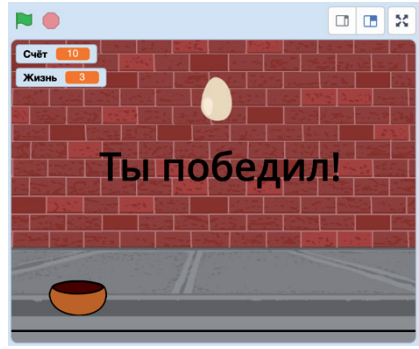
На продуктивном уровне учащимся дополнительно предлагаются справочный материал и краткое описание решения.

Справочный материал:

- используй скрипты палитры «Движение»: перейти на случайное положение; изменить x на; изменить y на;
- используй скрипты палитры «События»: когда флаг нажат; когда клавиша «стрелка влево» нажата; когда клавиша «стрелка вправо» нажата;
- используй скрипты палитры «Управления»: повторять всегда; ждать до ...; если ..., то ...;
- используй скрипты палитры «Переменные»: задать переменной значение; изменить переменную на ...;
- используй скрипты палитры «Звук»: играть звук до конца;
- используй скрипты палитры «Сенсоры»: касается;
- используй скрипты палитры «Внешнего вида»: спрятаться; показаться.

Пример словесного описания решения:

- Добавляем фон Wall 1 и спрайт Bowl. Создаем скрипты движения спрайта Bowl. Спрайт Bowl перемещается с помощью стрелок клавиатуры вправо и влево по фону Wall 1. Добавляем спрайт Egg и задаем скрипт перемещения по фону. Спрайт



а

б

Рис. 4. Задание «Собери яйца»

Egg все время перемещается в случайное положение и падает вниз. Создаем переменную счета и изменяем ее на единицу при касании двух спрайтов друг с другом. Создаем переменную жизни. Добавляем спрайты победы и проигрыша. Добавляем звук касания и условия окончания игры.

На творческом уровне учащимся предлагается только общая формулировка задачи.

6. Заключение

В данной статье были рассмотрены некоторые возможности визуализированной среды программирования Scratch для обучения основам алгоритмизации и программирования учащихся VI—VIII классов.

Результаты внедрения разработанных нами материалов на репродуктивном, продуктивном и творческом уровнях усвоения знаний показывают, что использование визуализированной среды программирования в учебном процессе может способствовать повышению мотивации учащихся, улучшению понимания ими основ алгоритмизации и программирования, а также развитию критического и творческого мышления. Однако для эффективного использования среды программирования Scratch в обучении необходимо разработать методически обоснованные подходы и учебные материалы, учитывающие возрастные особенности учащихся и специфику предмета.

Визуализированная среда программирования Scratch обладает всеми необходимыми инструментами для изучения школьниками содержательной линии «Основы алгоритмизации и программирования» в VI—VIII классах. В связи с этим актуальным становится разработка базы дидактических материалов для сопровождения факультатива «Основы алгоритмизации и программирования в визуализированной среде программирования Scratch», которые бы полностью обеспечивали изучение содержательной линии «Основы алгоритмизации и программирования» и закладывали необходимую базу знаний для изучения языков программирования Python, C++, C# или Java.

Список источников

1. *Алейникова Т. Г., Оганджанян О. П.* Задачник по программированию в Scratch. Витебск: ВГУ имени П. М. Машерова, 2018. 44 с.

2. *Вордерман К., Вудкок Д., Макаманус Ш.* Программирование для детей. Иллюстрированное руководство по языкам Scratch и Python. М.: Манн, Иванов и Фербер, 2017. 224 с.

3. *Котов В. М., Быкадоров Ю. А., Лапо А. И., Войтехович Е. Н.* Информатика: учебное пособие для 8 класса учреждений общего среднего образования с русским языком обучения. Минск: Народная асвета, 2018. 168 с. https://uchebniki.by/media/download/files/narodnaya_asveta/informatika_8kl_kotov_rus_2018.pdf

4. *Котов В. М., Лапо А. И., Войтехович Е. Н.* Информатика: учебное пособие для 7 класса учреждений общего среднего образования с русским языком обучения. Минск: Народная асвета, 2017. 176 с. https://uchebniki.by/media/download/files/narodnaya_asveta/informatika-kotov-7kl-rus-2017.pdf

5. *Макарова Н. П., Лапо А. И., Войтехович Е. Н.* Информатика: учебное пособие для 6 класса учреждений общего среднего образования с русским языком обучения. Минск: Народная асвета, 2018. 168 с. https://uchebniki.by/media/download/files/narodnaya_asveta/informatika_6kl_makarova_rus_2018.pdf

6. *Торгашева Ю. В.* Программирование для детей. Учимся создавать игры на Scratch. СПб.: Питер, 2018. 128 с.

7. Учебная программа по учебному предмету «Информатика» для VI класса учреждений образования, реализующих образовательные программы общего среднего образования с русским языком обучения и воспитания. Утв. Постановлением Министерства образования Республики Беларусь 07.07.2023 № 190. https://adu.by/images/2023/08/matem/up_inf_6_rus_1.docx

8. Учебная программа по учебному предмету «Информатика» для VII класса учреждений образования, реализующих образовательные программы общего среднего образования с русским языком обучения и воспитания. Утв. Постановлением Министерства образования Республики Беларусь 07.07.2023 № 190. https://adu.by/images/2023/08/matem/up_inf_7_rus_1.docx

9. Учебная программа по учебному предмету «Информатика» для VIII класса учреждений образования, реализующих образовательные программы общего среднего образования с русским языком обучения и воспитания. Утв. Постановлением Министерства образования Республики Беларусь 07.07.2023 № 190. https://adu.by/images/2023/08/matem/up_inf_8_rus_1.docx

10. Учебная программа факультативного занятия «Основы алгоритмизации и программирования в визуальной среде программирования SCRATCH» для VII—VIII классов учреждений образования, реализующих образовательные программы общего среднего образования. Утв. Постановлением Министерства образования Республики Беларусь 28.07.2020 № 208. https://adu.by/images/2023/inform/fz-Osnovi-algoritmizacii-i-program-v-srede-SCRATCH-VII-VIII-kl.pdf

11. Учебная программа факультативного занятия «Пропедевтика основ алгоритмизации и программирования в визуальной среде программирования SCRATCH» для V—VI классов учреждений образования, реализующих образовательные программы общего среднего образования. Утв. Постановлением Министерства образования Республики Беларусь 16.06.2020 № 131. https://adu.by/images/2023/inform/fz_propedevtika_osnov_5-6kl.pdf

12. *Уэйнрайт М.* Программируем на Scratch. Приключение в джунглях. М.: Клевер-Медиа-Групп, 2018. 31 с.

13. *Францкевич А. А.* Визуализированные среды как средство повышения эффективности обучения школьников основам алгоритмизации и программирования: автореф. дис. ... канд. пед. наук: 13.00.02. Минск: БГУ, 2020. 28 с.

14. *Францкевич А. А.* Результаты педагогического эксперимента по внедрению методики обучения школьников основам алгоритмизации и программирования с использованием визуализированных сред программирования // Весці БДПУ. Серыя 3. Фізіка. Матэматыка. Інфарматыка. Біялогія. Геаграфія. 2019. № 4. С. 58–68.

15. *Хайлэнд М.* Программируем с детьми. Создайте 10 веселых игр на Scratch. М.: Эксмо, 2021. 176 с.

16. *Resnick M., Maloney J., Monroy-Hernández A., Rusk N., Eastmond E., Brennan K., Millner A., Rosenbaum E., Silver J., Silverman B., Kafai Ya.* Scratch: Programming for all // Communications of the ACM. 2009. Vol. 52. No. 11. P. 60–67. DOI: 10.1145/1592761.1592779

DOI: 10.32517/2221-1993-2023-22-6-54-62

Л. М. Коренюгина
Сибирский федеральный университет, г. Красноярск, Россия;
Средняя общеобразовательная школа № 63 Кировского района Красноярска, Россия

МОДЕЛИРОВАНИЕ ДВИЖЕНИЯ ОБЪЕКТОВ ПРИ СТОЛКНОВЕНИИ С ПРЕПЯТСТВИЯМИ НА ЯЗЫКЕ PASCALABC.NET В ПРОЕКТНОЙ ДЕЯТЕЛЬНОСТИ ШКОЛЬНИКОВ

Аннотация
Для решения методической задачи расширения содержания школьных учебников информатики в части практических заданий могут быть использованы задания для проектной работы, которые выполняются учащимися индивидуально или в группах. Выполнение проектной работы, предполагающей применение знаний из других предметных областей, способствует формированию у учащихся метапредметных умений. Использование групповой работы как основной формы выполнения практических заданий направлено на достижение личностных результатов обучения. Проектно-деятельностный подход помогает достижению планируемых результатов обучения в рассматриваемом разделе курса информатики. В статье рассмотрены задания по разделу «Начала программирования» курса информатики VIII класса, которые могут быть предложены для проектной деятельности учащихся по созданию графических и информационных моделей и при проведении компьютерного эксперимента. На основе выполнения постепенно усложняющихся примеров учащиеся подводятся к заданиям проектного типа — моделированию движения объектов при столкновении с препятствиями. Программная реализация моделей осуществляется на языке PascalABC.NET.
Ключевые слова: информатика, программирование, PascalABC.NET, проектная деятельность, компьютерное моделирование, информационное моделирование.

1. Введение

В настоящее время актуально решение методической задачи расширения содержания школьных учебников информатики в части практических заданий, в том числе предложение учащимся заданий для проектной работы. Практико-ориентированный метод обучения реализуется в процессе групповой и индивидуальной проектной деятельности учащихся по созданию графических и информационных моделей и при проведении компьютерного эксперимента.

В данной статье представлены дополнительные материалы к разделу «Начала программирования» учебника информатики для VIII класса Л. Л. Босовой, А. Ю. Бо-

* Материалы к статье можно скачать на сайте ИНФО: http://infojournal.ru/journals/school/school_06-2023/

Контактная информация

Коренюгина Людмила Михайловна,
ст. преподаватель, Институт космических и информационных технологий, Сибирский федеральный университет, г. Красноярск, Россия; *адрес:* 660041, Россия, г. Красноярск, Свободный пр-т, д. 79;
учитель информатики, средняя общеобразовательная школа № 63 Кировского района Красноярска, Россия; *адрес:* 660059, Россия, г. Красноярск, ул. Вавилова, д. 49Б;
e-mail: korenugina71@yandex.ru

L. M. Korenyugina
Siberian Federal University, Krasnoyarsk, Russia;
School 63, Krasnoyarsk, Russia

MODELING OBJECTS MOVEMENT WHEN COLLISING WITH OBSTACLES IN PASCALABC.NET LANGUAGE IN SCHOOLCHILDREN'S PROJECT ACTIVITIES

Abstract
To solve the methodological problem of expanding the content of school informatics textbooks in terms of practical tasks, assignments for project work can be used, which are completed by students individually or in groups. Carrying out project work that involves the application of knowledge from other subject areas contributes to the development of meta-subject skills in students. The use of group work as the main form of performing practical tasks is aimed at achieving personal learning results. The project-activity approach helps to achieve the planned learning outcomes in the section of the informatics course under consideration. The article discusses assignments in the "Beginnings of programming" section of the VIII grade informatics course, which can be offered for students' project activities to create graphical and information models and when conducting a computer experiment. Based on the implementation of gradually more complex examples, students are introduced to project-type tasks — modeling the movement of objects when collising with obstacles. The software implementation of the models is carried out in the PascalABC.NET language.
Keywords: informatics, programming, PascalABC.NET, project activity, computer modeling, information modeling.

совой [5]. В ходе проектной деятельности учащиеся выполняют задания по моделированию движения объектов по прямой и при столкновении с препятствиями. В процессе имитационного моделирования физических явлений с применением знаний, полученных при изучении тем «Программирование линейных алгоритмов», «Программирование разветвляющихся алгоритмов», «Различные варианты программирования циклического алгоритма», реализуется системно-деятельностный подход к обучению.

Проекты, основу которых составляет имитационное моделирование физических процессов [6, 10], могут быть реализованы на разных языках программирования. В данной статье предложены примеры и задания, предполагающие написание программ на языке Pascal [7, 8, 11, 14, 15].

В процессе разработки методики обучения информатике с использованием проектной деятельности в компьютерном моделировании физических явлений в качестве программного средства была выбрана среда программирования PascalABC.NET, так как она обладает такими свойствами, как ранний контроль ошибок, четкая структура программ, высокая скорость выполнения программ [7, 11, 15]. Немаловажно и то, что в школьных учебниках присутствуют соответствующие теоретические сведения [3–5, 12, 13].

Практика использования PascalABC.NET как средства программной реализации и организации проектной деятельности учащихся позволяет закрепить знания следующих тем раздела «Начала программирования» курса информатики VIII класса [5]:

- Язык программирования. Основные сведения о языке программирования Pascal: структура программы; правила представления данных; правила записи основных операторов (ввод, вывод, присваивание, ветвление, цикл).
- Решение задач по созданию программ в среде программирования Pascal.

В ходе выполнения заданий проектного типа учащиеся:

- программируют линейные алгоритмы, предполагающие вычисление арифметических, строковых и логических выражений;
- разрабатывают программы, содержащие операторы ветвления, в том числе с использованием логических операций;
- разрабатывают программы, содержащие операторы цикла.

2. Актуализация знаний. Пробное учебное действие с предварительной демонстрацией

Прежде чем приступать к выполнению заданий проектного типа, следует повторить основные моменты темы «Имитация движения графических объектов по прямой». При компьютерном моделировании движения объектов на экране меняются экранные координаты (рис. 1).

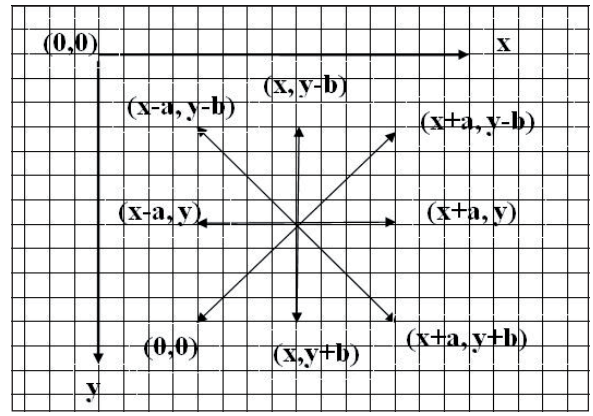


Рис. 1. Схема изменения координат движущегося объекта

Рассмотрим **несколько примеров задач** [9], которые могут быть предложены учащимся после изучения тем «Программирование циклов с заданным условием окончания работы», «Программирование циклов с заданным числом повторений», «Различные варианты программирования циклического алгоритма» раздела «Начала программирования» курса информатики VIII класса.

Пример 1.

Перемещение круга в цикле слева направо (т. е. в направлении увеличения координаты x).

Решение.

В библиотеке GraphABC есть процедуры LockDrawing и Redraw, которые позволяют передать сформированное изображение в буфер обмена и затем показать пользователю. В цикле рисуем движущийся круг.

Текст программы приведен в листинге 1, результат работы программы представлен на рисунке 2.

Как мы видим, необходимо стирать прежнее изображение, так как иначе за кругом будет оставаться след. Про-

```
program Kolobok;  //название программы
uses GraphABC;    //подключение графического модуля
var i: integer;   //объявление переменной для реализации цикла
begin             //начало программы
  setwindowtitle('Колобок слева направо-ФАМИЛИЯ');
  for i:=1 to 500 do begin   //начало цикла
    lockdrawing;             //передать изображение в буфер обмена
    setpencolor(clyellow);   //выбор цвета карандаша рисунка
    circle(100+i,200,40);    //окружность с изменяемыми координатами
    sleep(20);               //задержка выполнения в миллисекундах
    redraw;                  //вывести изображение из буфера обмена на экран
  end;                       //конец цикла
end.              //конец программы
```

Листинг 1

стейший способ стереть старое изображение — очистить все графическое окно, вызвав его метод ClearWindow.

Пример 2.

Изменить направление движения шарика, модифицировав код предыдущей программы.

Решение.

Текст программы приведен в листинге 2, результат работы программы представлен на рисунке 3.

Пример 3.

Представить имитационную компьютерную модель движения нескольких шаров разного цвета, внеся изменения в программу примера 2.

Моделируем одновременное движение в разных направлениях четырех шаров разного цвета.

Решение.

Текст программы приведен в листинге 3, результат работы программы представлен на рисунке 4.

```
program Circles;
uses GraphABC;
var i: integer;
begin
  SetWindowSize(500,500);  //Размеры экрана вывода результата
  i:=10;
  repeat
    ClearWindow;  //очистка экрана
    SetBrushColor (clrandom);
    circle(10+2*i,10+i,50);
    sleep(100);
    i:=i+10;
    until i>600;
end.
```

Листинг 2

```
program circles2;
uses GraphABC;
var i: integer;
begin
  SetWindowSize(500,500); //размеры экрана вывода результата
  i:= 10;
  randomize;
  repeat
    ClearWindow;
    SetBrushColor(clrandom);  //выбор случайного цвета заливки шара
    circle(10+2*i,10+3*i,50);
    SetBrushColor(clRed);     //выбор красного цвета заливки шара
    circle(500-2*i,10+i,50);
    SetBrushColor(clYellow);  //выбор желтого цвета заливки шара
    circle(500-4*i,500-i,50);
    SetBrushColor(clGreen);   //выбор зеленого цвета заливки шара
    circle(10+i,500-i,50);
    sleep(100);
    i:=i+10;
    until i>600;
end.
```

Листинг 3

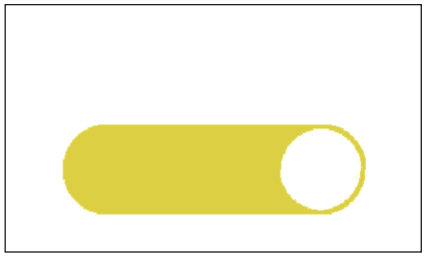


Рис. 2. Результат работы программы движения шарика слева направо

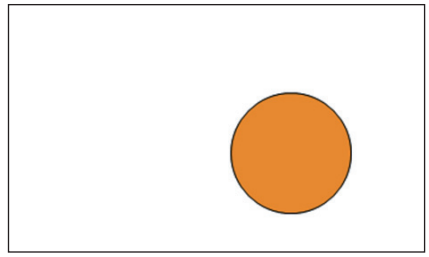


Рис. 3. Результат работы программы движения шарика справа налево

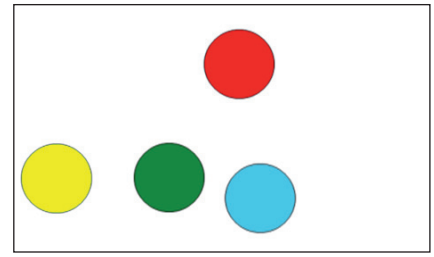
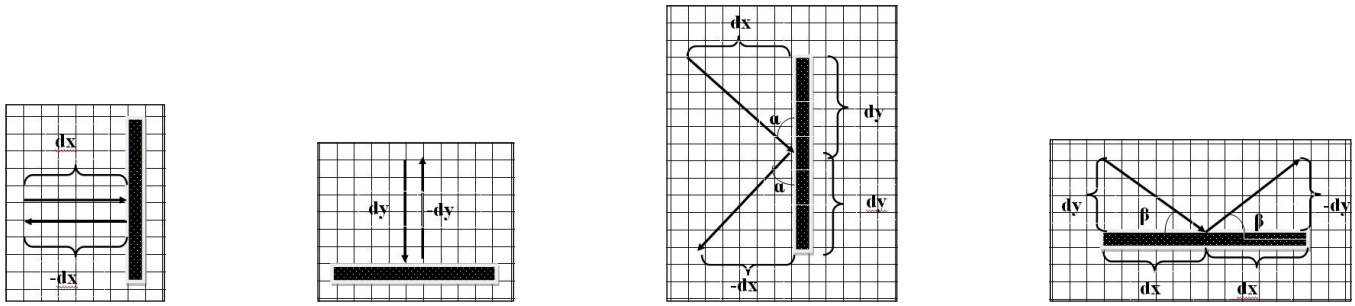


Рис. 4. Результат работы программы движения нескольких шаров



а *б* *в* *г*

Рис. 5. Схемы разных видов взаимодействия движущегося объекта и препятствия

3. Составление математической и графической моделей движения объектов при столкновении с препятствиями

При постановке задачи проекта мы применяем объяснительно-иллюстративный метод обучения. Строим графическую модель изменения траектории движения объекта, движущегося по прямой, после столкновения с препятствием:

- на рисунках 5, *а, б* показано изменение координат при столкновении с препятствием, когда исходная траектория движения была перпендикулярна к препятствию;
- на рисунках 5, *в*, *г* показано изменение координат при столкновении с препятствием под некоторым углом.

Для моделирования используем правило: угол падения равен углу отражения. При вертикальном расположении траектории падения знак приращения только одной координаты меняется на противоположный. При падении под углом меняются оба приращения координат. Данные свойства и изменения координат наглядно демонстрируются на рисунке 5.

Для описания реализации модели используются алгоритмические конструкции «ветвление» и «повторение».

Практическое применение представленной модели. Данная модель может быть использована для написания учебных проектов компьютерных игр («Футбол», «Теннис» и т. п.), для имитационного моделирования движения объектов на дорогах («Перекресток» и т. п.).

4. Программная реализация моделей

Пример 4.

Красный шар ударяется в синий квадрат и отскакивает от него в противоположную сторону до невидимой преграды. Движения повторяются в цикле while.

В примере меняется знак приращения координат объекта на противоположный после того, как расстояние от центра шара до препятствия становится меньше радиуса шара.

Решение.

Текст программы приведен в листинге 4, результат работы программы представлен на рисунке 6.

```
Program Stolknovenie;
uses GraphABC;
var i,x,y,dx,r,x0,y0: integer;
begin
  SetWindowsize(600,500);
  x0:=100; y0:=260; r:=60;  //координаты центра и радиус шара
  x:=400; y:=200;
  dx:=20;
  LockDrawing;
  while i<300 do begin
    ClearWindow;
    SetBrushColor(clred);  //выбор красного цвета заливки шара
    circle(x0,y0,r);
    setpencolor(clblue);   //выбор синего цвета квадрата
    rectangle(x,y,x+120,y+120);
    floodfill(x+1,y+1,clblue);
    sleep(100);
    if (x0>=x-r) or (x0<r) then dx:=-dx;
    x0:=x0+dx;
    Redraw;
  end;
end.
```

Листинг 4

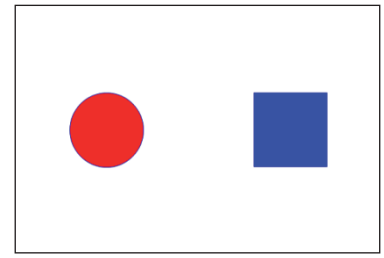


Рис. 6. Результат работы программы столкновения шара и квадрата

```
program Football;
uses graphabc;
var x0, y0, x1, y1, dx, dy, i, r: integer;
begin
  setwindowsize(900,600);  //задание размеров экрана
  x0:=50;y0:=50;           //левые верхние координаты игрового поля
  r:=30;                   //радиус шарика
  x1:=150+random(50);      //начальные координаты шарика по горизонтали
  y1:=150+random(70);      //начальные координаты шарика по вертикали
  dx:=10;
  dy:=6;
  LockDrawing;
  while i<300 do begin
    ClearWindow;
    rectangle(x0,y0,x0+800,y0+500);       //внешний борт игрового поля в виде прямоугольника
    rectangle(x0+50,y0+50,x0+750,y0+450); //внутренний борт игрового поля
    circle(x1+dx,y1+dy,r);                //окружность , изображающая мячик
    floodfill(x1+dx,y1+dy,clred);         //заливка красного цвета
    sleep(30);
    if (x1>=750) or (x1<150) then dx:=-dx;
    if (y1>=450) or (y1<150) then dy:=-dy;
    x1:=x1+dx;
    y1:=y1+dy;
    Redraw;
  end;
end.
```

Листинг 5

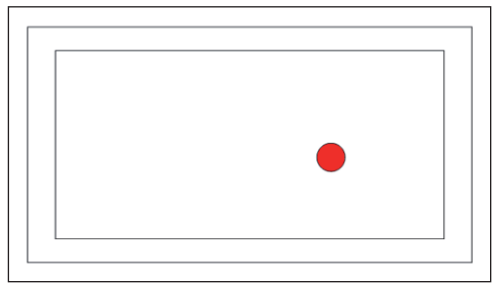


Рис. 7. Результат работы программы «Футбол»

Пример 5. Игра «Футбол».

Шарик возникает в любом месте экрана на прямоугольном поле. Он движется в произвольном направлении и сталкивается с бортами игрового поля. Согласно схеме взаимодействия с препятствиями он меняет направление своего движения.

Решение приведено в листинге 5, результат работы программы представлен на рисунке 7.

5. Задания для самостоятельной работы

Задание 1.

Написать программу движения машины слева направо.

Этапы выполнения задания.

1) Начертить проект в тетради в клеточку.

2) Написать программу.

3) Запустить программу на выполнение и протестировать ее на наличие ошибок.

Пример выполненного проекта приведен на рисунках 8, 9 и в листинге 6.

Задание 2.

Написать программу движения улитки слева направо.

Текст программы приведен в листинге 7, пример работы программы — на рисунке 10.

Задание 3.

Написать программу имитации столкновения двух машин.

Текст программы приведен в листинге 8, пример работы программы — на рисунке 11.

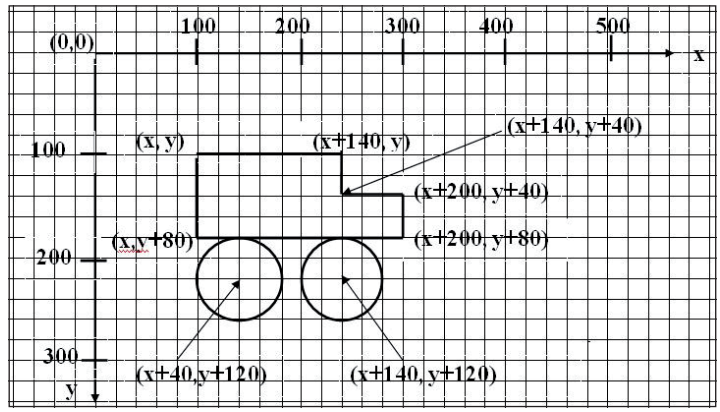


Рис. 8. Пример выполнения первого этапа задания 1

```
program Car;
uses GraphABC;
var x, y, i: integer;
begin
  SetWindowSize(800,500);
  SetWindowTitle('Машина едет слева направо-ФАМИЛИЯ');
  x:=100;
  y:=200;
  LockDrawing;
  for i:=1 to 300 do begin
    ClearWindow;
    SetPenColor(clgray);
    line(x,y,x+140,y);
    line(x+140,y,x+140,y+40);
    line(x+140,y+40,x+200,y+40);
    line(x+200,y+40,x+200,y+80);
    line(x+200,y+80,x,y+80);
    line(x,y+80,x,y);
    SetPenColor(clred);
    circle(x+40,y+120,40);
    circle(x+160,y+120,40);
    x:=x+3;
    sleep(20);
    Redraw;
  end;
end.
```

Листинг 6

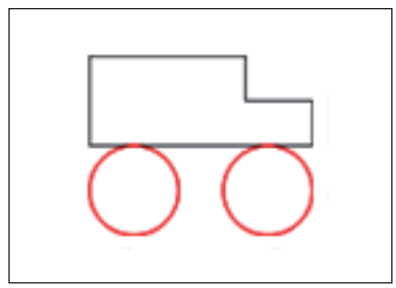


Рис. 9. Пример работы программы задания 1

Задание 4.

Написать программу имитации столкновения двух персонажей, после которого выводится надпись «Ой!» (или любая другая надпись).

Задание 5.

Написать программу имитации падения мяча и его отскока от горизонтальной поверхности.

```
program Ulitka;
uses GraphABC;
var x, y, i: integer;
begin
  SetWindowSize(940,680);
  SetWindowTitle('Улитка-ФАМИЛИЯ');
  SetPenWidth(2);
  x:=100;
  y:=100;
  LockDrawing;
  for i:=1 to 50 do begin
    ClearWindow;
    setpencolor (clblack);
    circle(x,y,50);
    circle(x,y,35);
    circle(x,y,20);
    circle(x+90,y-10,20);
    circle(x+80,y-15,3);
    circle(x+100,y-15,3);
    line(x+85,y,x+95,y);
    line(x+90,y-30,x+105,y-45);
    line(x+90,y-30,x+75,y-45);
    line(x+5,y+50,x+90,y+10);
    line(x+45,y+50,x+90,y+10);
    line(x+5,y+50,x+45,y+50);
    x:=x+i;
    sleep(100);
    Redraw;
  end;
end.
```

Листинг 7

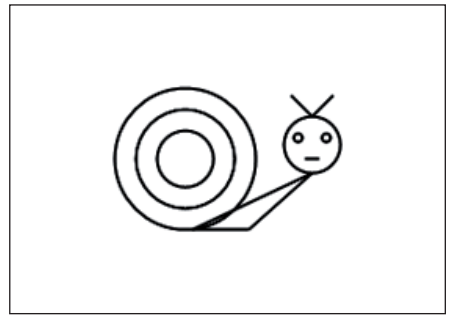


Рис. 10. Пример работы программы задания 2

6. Заключение

Использование в обучении информатике заданий проектного типа позволяет обогатить содержание тем раздела «Начала программирования» курса информатики VIII класса. Данный подход нацелен на изменение соотношения частей курса, направленных на освоение знаний и формирование практических предметных умений. Базовым принципом отбора содержания, помимо классических принципов, становится деятельностный подход. Выполнение проектной работы, предполагающей применение знаний из других предметных областей, способствует формированию у учащихся метапредметных умений. Использование групповой работы как основной формы выполнения практических заданий направлено на достижение личностных результатов обучения. Проектно-деятельностный подход помогает достижению планируемых результатов обучения в рассматриваемом разделе курса информатики. Ввод в систему обучения заданий проектного типа отвечает познавательному интересу учащихся, позволяет увеличить количество ребят, вовлеченных в разработку учебных проектов.

Использование компьютерного моделирования в рамках проектной деятельности учащихся при обучении информатике с применением проектного метода отвечает требованиям практической ориентированности, развития межпредметных связей, соответствия курса информатики современным тенденциям построения содержания, обогащающего базовый курс информатики [1, 2].

```
program Cars;
uses GraphABC;
var x1, x2, y1, y2, y0: integer;

procedure Per;
begin
  SetPenColor(clblack);
  line(0,y1,windowwidth,y1);
  line(0,y2,windowwidth,y2);
  line(windowwidth div 2-30,0,windowwidth div 2-30,windowheight);
  line(windowwidth div 2+30,0,windowwidth div 2+30,windowheight);
  SetPenColor(clwhite);
  rectangle(windowwidth div 2-30,y1,windowwidth div 2+31,y2+1);
end;

procedure Auto2(x,y,k: integer);
begin
  SetPenColor(clblack);
  rectangle(x-k*10,y-8,x,y+8);
  rectangle(x-k*10-k*30,y-10,x-k*10,y+10);
end;

begin
  y0:=windowheight div 2;
  y1:=y0-30;
  y2:=y0+30;
  x1:=40;
  x2:=windowwidth-40;
  Per;
  Auto2(x1,y2-15,1);
  Auto2(x2,y1+15,-1);
  LockDrawing;
  repeat
    ClearWindow;
    Per;
    Auto2(x1,y2-15,1);
    Auto2(x2,y1+15,-1);
    sleep(50);
    x1:=x1+2;
    x2:=x2-4;
    redraw;
  until x1>=windowwidth;
end.
```

Листинг 8

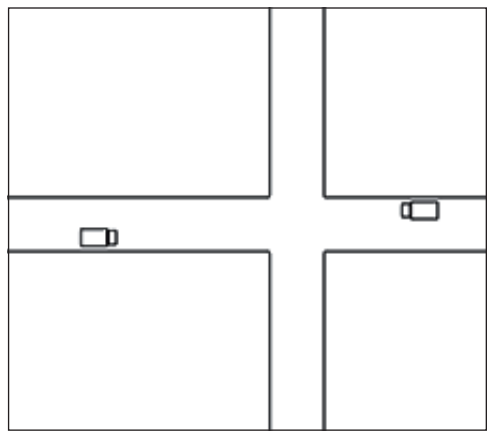


Рис. 11. Пример работы программы задания 3

Список источников

1. *Босова Л. Л.* Как учат программированию в XXI веке: отечественный и зарубежный опыт обучения программированию в школе // Информатика в школе. 2018. № 6. С. 3–11. EDN: XZOOJV.

2. *Босова Л. Л.* Международные тенденции развития школьного образования в области информатики и ИКТ // Материалы Шестнадцатой открытой Всероссийской конференции «Преподавание информационных технологий в Российской Федерации» (Москва, 14–15 мая 2018 года). М.: МГТУ имени Н. Э. Баумана, 2018. С. 350–352. EDN: XUJFSX.

3. *Босова Л. Л., Босова А. Ю.* Информатика. 7–9 классы: методическое пособие. М.: БИНОМ. Лаборатория знаний, 2016. 464 с. https://lbz.ru/metodist/iumk/informatics/files/bosova-7-9-met.pdf

4. *Босова Л. Л., Босова А. Ю.* Информатика: рабочая тетрадь для 8 класса: в 2 ч. Ч. 2. М.: БИНОМ. Лаборатория знаний, 2019. 88 с.

5. *Босова Л. Л., Босова А. Ю.* Информатика: учебник для 8 класса. М.: БИНОМ. Лаборатория знаний, 2013. 155 с.

6. *Гейн А. Г.* Изучение информационного моделирования как средство реализации межпредметных связей информатики с дисциплинами естественнонаучного цикла: дис. ... д-ра пед. наук: 13.00.02. 2000. 300 с.

7. *Долинер Л. И.* Основы программирования в среде PascalABC.NET: учебное пособие. Екатеринбург: Изд-во Уральского университета, 2014. 128 с. https://elar.urfu.ru/bitstream/10995/28702/1/978-5-7996-1260-3_2014.pdf

8. *Комарова Е. С.* Практикум по программированию на языке Паскаль: учебное пособие. 2-е изд., стер. Ч. 1. М.; Берлин: Директ-Медиа, 2019. 86 с.

9. *Коренюгина Л. М.* Информатика и программирование (PascalABC.NET): методические указания по подготовке к практическим занятиям, в том числе в интерактивной форме, и самостоятельному изучению дисциплины для школьников 8 классов общеобразовательных средних учебных заведений (базовый уровень). Красноярск: МБОУ СОШ 63, 2022. 103 с. http://suroklxj.bget.ru/ege_informatika/МУ_ИНФОРМАТИКА_ПРОГРАММИРОВАНИЕ_ПАСКАЛЬ_Коренюгина_2020.pdf

10. *Кравченко К. В., Кузьмичева Е. А., Шахбазян Я. А.* Компьютерное моделирование как средство обучения в среднем и общем образовании // Информационные технологии в образовательном процессе вуза и школы. Материалы XV Всероссийской научно-практической конференции. Воронеж: Воронежский государственный педагогический университет, 2021. С. 227–230. EDN: ZRGRSX.

11. *Осипов А. В.* PascalABC.NET: выбор школьника. Ч. 1. Ростов-на-Дону; Таганрог: Изд-во Южного федерального университета, 2020. 148 с. http://pascalabc.net/downloads/OsipovBook/StudentChoice.pdf

12. *Поляков К. Ю., Еремин Е. А.* Информатика. 8 класс: учебник. М.: БИНОМ. Лаборатория знаний, 2019. 256 с.

13. *Семакин И. Г., Шестаков А. П.* Основы программирования: учебник для студентов учреждений среднего профессионального образования. 2-е изд., стер. М.: Академия, 2020. 304 с.

14. *Сырицына В. Н., Кадеева О. Е.* Модернизация языка программирования Pascal // Инновации в науке. 2017. № 6 (67). С. 18–20. EDN: YJBHYJ.

15. *Цветков А. С.* Язык программирования PASCAL. Система программирования ABC Pascal: учебное пособие для школьников 7–9 классов. СПб.: Павловск, 2015. 46 с.

DOI: 10.32517/2221-1993-2023-22-6-63-70

А. Г. Степанов, В. М. Космачев, О. И. Москалева
Санкт-Петербургский государственный университет аэрокосмического приборостроения, Санкт-Петербург, Россия

ФОРМИРОВАНИЕ НАВЫКОВ И УМЕНИЙ ОБУЧАЮЩИХСЯ С ИСПОЛЬЗОВАНИЕМ ТЕСТОВОГО ЗАДАНИЯ ТИПА «ФОРМУЛЫ» В MOODLE

Аннотация
В статье обсуждаются средства противодействия академическому мошенничеству при проведении компьютерного тестирования. Особое внимание уделяется борьбе с составлением обучающимися таблиц правильных ответов на вопрос тестового задания. Предлагается использовать имеющуюся в Moodle технологию автоматизации генерации вариантов тестовых заданий. Рассматриваются задания типа «Вычисляемый» и «Формулы». Описывается метод создания вопроса в форме, допускающей его неоднократное применение в различных случайных вариантах формулировки как на этапе обучения с целью формирования навыков и умений, так и в процессе промежуточного контроля. На примере, связанном с обучением программированию, демонстрируется возможность создания фрагментов программ со случайным набором идентификаторов, что заставляет обучаемого понимать смысл анализируемого фрагмента. Приводится пример решения задачи по физике, позволяющий показать возможность контроля знаний стандартных формул и умения проводить расчеты с их использованием.

Ключевые слова: академическое мошенничество, компьютерное тестирование, генерация тестов, вычисляемый вопрос, задание типа Формулы, Moodle.

1. Введение

Компьютерный тест — современный педагогический инструмент, который может быть с успехом использован в условиях как очного, так и дистанционного обучения. Несмотря на множество явных преимуществ по сравнению с традиционными средствами обучения и контроля, он имеет и ряд существенных недостатков, связанных с встречающимися на практике случаями академического мошенничества, в частности, с подсказыванием и списыванием. Если первая причина устраняется за счет проведения контрольных мероприятий в очном режиме с соответствующей организацией испытаний, то бороться со списыванием оказывается гораздо сложнее в силу большого количества уже придуманных обучающимися вариантов организации этого процесса. Один из них связан с повсеместным составлением таблиц правильных ответов на вопросы компьютерного теста в процессе обучения. Затем эти таблицы используются во время контрольных мero-

Контактная информация

Степанов Александр Георгиевич, доктор пед. наук, доцент, профессор кафедры бизнес-информатики и менеджмента, Санкт-Петербургский государственный университет аэрокосмического приборостроения, Санкт-Петербург, Россия; *адрес:* 190000, Россия, г. Санкт-Петербург, ул. Большая Морская, д. 67; *e-mail:* georgich_spb@mail.ru

Космачев Валентин Михайлович, канд. тех. наук, доцент, профессор кафедры бизнес-информатики и менеджмента, Санкт-Петербургский государственный университет аэрокосмического приборостроения, Санкт-Петербург, Россия; *адрес:* 190000, Россия, г. Санкт-Петербург, ул. Большая Морская, д. 67; *e-mail:* kvm@aanet.ru

Москалева Ольга Ильинична, ст. преподаватель кафедры бизнес-информатики и менеджмента, Санкт-Петербургский государственный университет аэрокосмического приборостроения, Санкт-Петербург, Россия; *адрес:* 190000, Россия, г. Санкт-Петербург, ул. Большая Морская, д. 67; *e-mail:* o.i.moskaleva@gmail.com

A. G. Stepanov, V. M. Kosmachev, O. I. Moskaleva
Saint-Petersburg State University of Aerospace Instrumentation, Saint Petersburg, Russia

FORMATION OF STUDENTS' SKILLS AND ABILITIES USING FORMULAS-TYPE TEST TASKS IN MOODLE

Abstract
The means of countering academic fraud in computer testing are discussed in the article. Special attention is paid to the fight against students compiling a table of correct answers to a test assignment question. It is proposed to use the technology available in Moodle to automate the generation of test task variants. Tasks of the type "Calculated" and "Formulas" are considered. The method of creating a question in a form that allows its repeated use in various random versions of the formulation both at the learning stage in order to form skills and abilities, and in the process of intermediate control is described. Using an example related to programming training, the possibilities of creating program fragments with a random set of identifiers are demonstrated, which forces the student to understand the meaning of the analyzed fragment. An example of solving a physics problem is given, which allows us to show the possibility of controlling knowledge of standard formulas and the ability to perform calculations using them.

Keywords: academic fraud, computer testing, test generation, computed question, Formulas type question, Moodle.

приятий. Ситуация отягощается тем, что, если тестирование проводится с помощью компьютера, вероятно использование его же для поиска правильных ответов с помощью обычной фильтрации заготовленного текста. Поэтому, с одной стороны, преподаватели вынуждены принимать жесткие меры по ограничению возможностей доступа обучающихся к тестовым заданиям до начала контрольных мероприятий, с другой стороны, мы не можем не согласиться с высказанным в работе [2] мнением ее авторов: «Существует пять важнейших элементов по использованию оценивания: закрепление и повторение материала, подготовка к аттестации, проектное обучение, персонализация, мотивация и удержание внимания, подготовка к итоговой аттестации по курсу. Чем более регулярная и планомерная итоговая аттестация по курсу, тем более высокий уровень подготовки и формируемых компетенций демонстрируют обучающиеся. Задача преподавателя состоит в том, чтобы делать оценивание регулярным, максимально автоматизировав эту технологию».

Возникает проблема, связанная с тем, что, с одной стороны, преподаватель заинтересован максимально возможно автоматизировать формирование у обучаемого представления о содержании дисциплины, а с другой стороны, необходимо предотвратить создание базы ответов на контрольные задания. Необходимо настолько затруднить использование такой базы во время контрольных испытаний, чтобы учащимся было проще выучить материал и отказаться от мошенничества в виде списывания. Как следствие, приходится искать компромисс.

Причины академического мошенничества связаны с низким уровнем учебной мотивации и собственным представлением обучаемого о честности окружающей его среды [8]. Как справедливо отмечает Д. Л. Еськин, часто мошенничество «рассматривается обучающимися как самый простой путь, позволяющий достигнуть требуемого результата в виде положительной оценки с минимальными на то затратами сил и времени. И если в качестве конечной цели обучения выбирается для себя исключительно получение документа об образовании, а не овладение системой знаний, умений и навыков, то вероятность того, что обучаемый будет прибегать к различным формам академического мошенничества, стремится к единице» [3]. Как следствие, отношение к контролю результатов обучения во многом определяет и общие результаты образовательной деятельности [5].

2. Компьютерное тестирование как средство контроля знаний

Традиционно компьютерное тестирование рассматривается как средство контроля знаний, под которыми, согласно [1], понимается «субъективный образ реальности в форме понятий и представлений».

Обычно для компьютерного тестирования используются так называемые **закрытые вопросы**. Простейший, наименее защищенный вопрос такого вида представляет собой некую фразу (например, определение понятия), на которую испытуемый должен дать ответ типа «Верно/Неверно».

Лучше от случайного угадывания правильного ответа защищены **многоальтернативные вопросы** типа «Выбор пропущенных слов», «Перетаскивание в текст», а также вопросы типа «Множественный выбор», которые предусматривают несколько вариантов правильного ответа.

Для проверки знаний могут использоваться и так называемые **открытые вопросы** «Числовой ответ» и «Короткий ответ». Они предусматривают клавиатурный ввод соответственно числа или строки.

Общим у упомянутых заданий является их **низкая защищенность от проявления академического мошенничества** как раз в виде использования заранее подготовленных таблиц правильных ответов. Возможно, что именно это обстоятельство поддерживает в педагогическом сообществе отрицательное отношение к компьютерному тестированию.

3. Компьютерное тестирование как средство контроля навыков и умений

Контроль навыков («действие, сформированное путем повторения и доведения до автоматизма» [1]) и умений («освоенный субъектом способ выполнения действия, обеспечиваемый совокупностью приобретенных знаний и навыков, который формируется путем упражнений и создает возможность выполнения действия в стандартных ситуациях» [1]) требует использования других технических и программных средств. В зависимости от предметной области ими могут быть реальные физические устройства, тренажеры, лабораторные стенды и т. п. Тем не менее во многих практических задачах обучения такие устройства могут быть заменены их математическими моделями. В этом случае контроль навыков, умений и владений осуществляется средствами вычислительной техники и может быть сведен к проверке понимания выбора обучаемым используемой модели, правильности формирования исходных данных и правильной интерпретации результатов моделирования. Как следствие, *для проверки результатов обучения возникает необходимость в разработке компьютерных тестов, использующих в своей работе математические модели, и появляется возможность создания компьютерных тренажеров по самой разнообразной тематике*.

В работе [6] нами были рассмотрены возможности использования в Moodle вопроса типа «Вычисляемый». В основе его программирования лежит возможность создания дискретного алгебраического выражения и генерации случайных входных данных. Общее количество создаваемых вариантов тестовых билетов весьма велико, и это обстоятельство может быть использовано для формирования определенного навыка или умения. Кроме этого рассмотренный метод снижает и вероятность академического мошенничества. Недостатком обсуждаемого типа тестового задания являются ограниченные возможности программирования правильного ответа. С помощью вопроса типа «Вычисляемый» можно проверить достаточно сложные алгоритмы вычислений, включающие условные операторы, однако отсутствуют возможность работы с циклами, а также средства обработки текстовых сообщений.

4. Задания типа «Формулы» в Moodle как средство контроля навыков и умений

В Moodle существует еще один тип заданий со случайными значениями параметров вопроса и несколькими полями ответов под названием «Формулы» (рис. 1). Поля ответов можно разместить в любом месте текста вопроса. Существует возможность создавать вопросы, включающие различные структуры ответов (координаты, полином, матрица). Доступна проверка единиц измерения [9].

Для вычисления правильного ответа на задание используется собственный язык программирования, включающий комментарии (символ «#;»), присваивания, циклы, списки, генераторы случайных чисел, встроенные функции обработки чисел и строк и многое другое. В качестве разделителя операторов выступает символ «;» [10]. Программирование ответа на вопрос «Формулы» осуществляется в разделе «Переменные», который содержит два окна: «Случайные переменные» и «Глобальные переменные» (рис. 2). Первое используется для объявления случайных переменных и задания параметров их изменения. Второе — собственно для программирования действий над переменными. Для обозначения переменных используются идентификаторы, а тип данных, который может быть числом, строкой, алгебраической переменной, списком чисел или строк, определяется автоматически.

Формулировка задания, отображаемого ученику, заносится в раздел «Основной вопрос» в окно «Текст вопроса». Кроме обычного текста в это окно может быть занесен так называемый заполнитель. Его признаком является пара фигурных скобок. Если заполнитель представляет собой имя переменной, то в тексте вопроса при его генерации отобразится текущее значение переменной. Если требуется выполнить вычисление в тексте вопроса, то текст заполнителя можно оформить в виде выражения, которое должно начинаться с символа равенства «=» и может включать в себя переменные и допустимые операции.

Ответ испытуемого программируется в разделе, например «Часть 1» (рис. 3). Там, в частности, имеется окно «Текст части» («Part's text»), которое можно использовать для генерации текста, обрамляющего поле ответа испытуемого. Сам ответ может быть предусмотрен в одном из четырех возможных режимов, задаваемых раскрывающимся списком «Тип ответа»: «Число», «Числовой», «Числовая формула» и «Алгебраическая формула». По умолчанию в созданном билете поле для ответа размещается после этого текста или, при необходимости, может быть установлено в любом месте результирующей формы.

В своей практике мы используем компьютерное тестирование в Moodle, в частности, для обучения языку программирования C++.

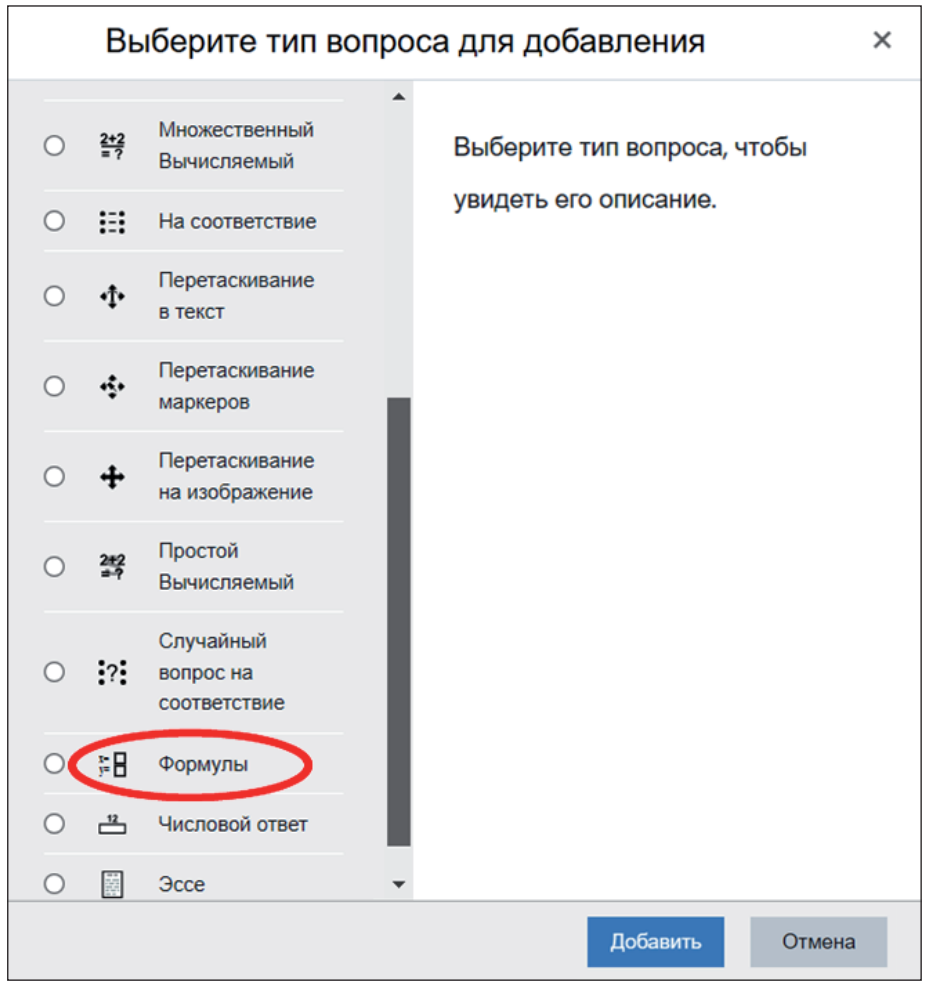


Рис. 1. Выбор типа вопроса

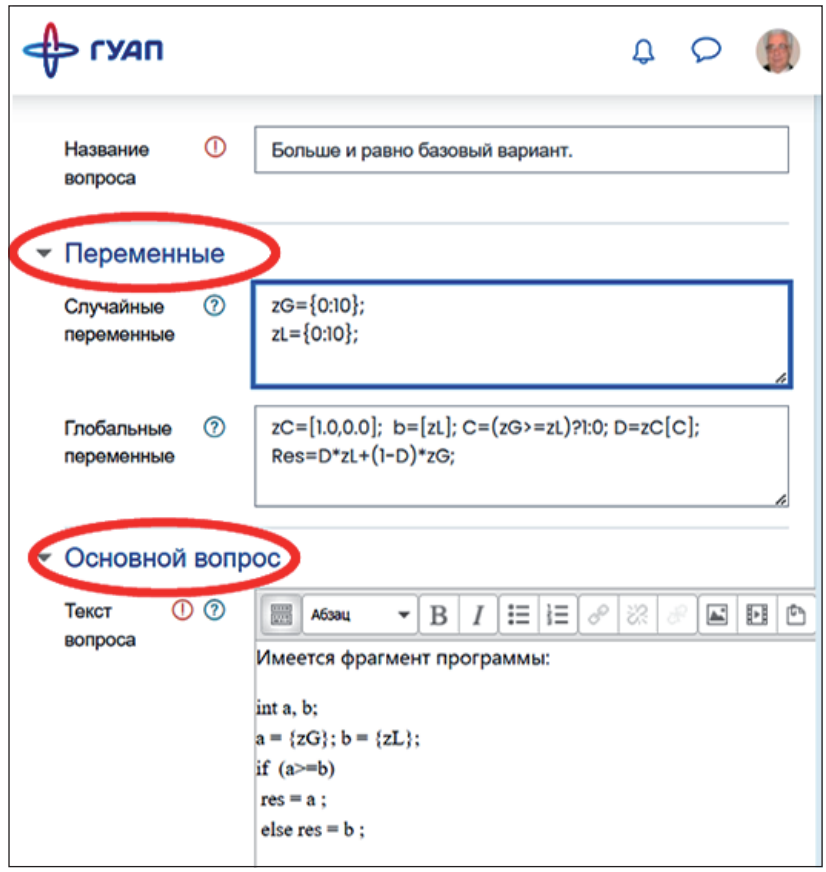


Рис. 2. Программирование ответа на вопрос «Формулы»

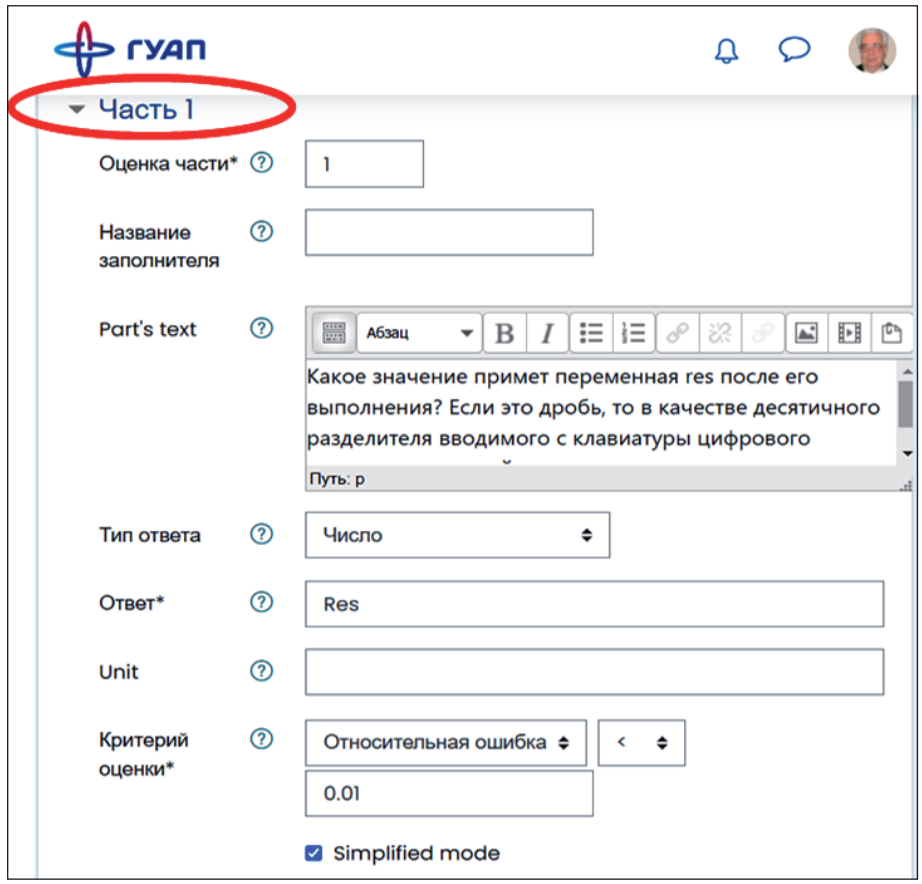


Рис. 3. Программирование ответа «Часть 1»

5. Пример использования вопроса типа «Формулы» в Moodle при формировании навыков программирования

Для примера рассмотрим предлагаемую технологию использования вопроса типа «Формулы» для формирования навыка программирования условного оператора и умения читать чужие программы с условным оператором, написанные на языке C++ и отличающиеся идентификаторами и некоторыми другими мелкими деталями. Предполагается, что обучающийся уже ознакомлен с правилами оформления условного оператора if в тексте программы.

Рассмотрим пример программирования тестового задания типа «Формулы», реализующего проверку знаний и умений использования оператора if в языке C++.

Окно «Случайные переменные» раздела «Переменные» заполнено следующим образом:

```
zG={-10:10}; zL={-10:10}; zX={-30:30};
ID={0:5}; IZ={0:6};
```

Здесь:

- zG, zL, zX, ID, IZ — имена случайных переменных, а выражения в фигурных скобках задают диапазон их изменения;
- три первых имени используются для генерации числовых данных синтезируемого тестового задания (программы на языке C++);
- ID — случайный номер варианта комбинации идентификаторов программы C++;
- IZ — вид логической операции условного оператора.

Окно «Глобальные переменные», где, собственно говоря, и осуществляется основное программирование, содержит следующий текст программы:

```
id1 = ["arg1", "px", "fa", "mx", "hz"];
id2 = ["arg2", "py", "ta", "nx", "tz"];
id3 = ["stp", "pz", "ha", "sx", "fz"];
id4 = ["rrr", "res", "result", "rx", "rz"];
zn = [">=", ">", "<", "<=", "==", "!="];
zC = [1, 0];
C = (zG >= zL && IZ == 0)?1:
    (zG > zL && IZ == 1)?1:
    (zG < zL && IZ == 2)?1:
    (zG <= zL && IZ == 3)?1:
    (zG == zL && IZ == 4)?1:
    (zG != zL && IZ == 5)?1:0;
D = zC[C];
Res = C*(zG+zX) + D*(zL-zX);
```

Здесь:

- списки id1—id4 содержат варианты имен, используемых в сформированном билете идентификаторов;
- zn — случайный знак операции;
- далее следует код программы, моделирующей итоговый результат (символ "?" — признак условного оператора собственного языка программирования задания «Формулы» (рис. 4)).

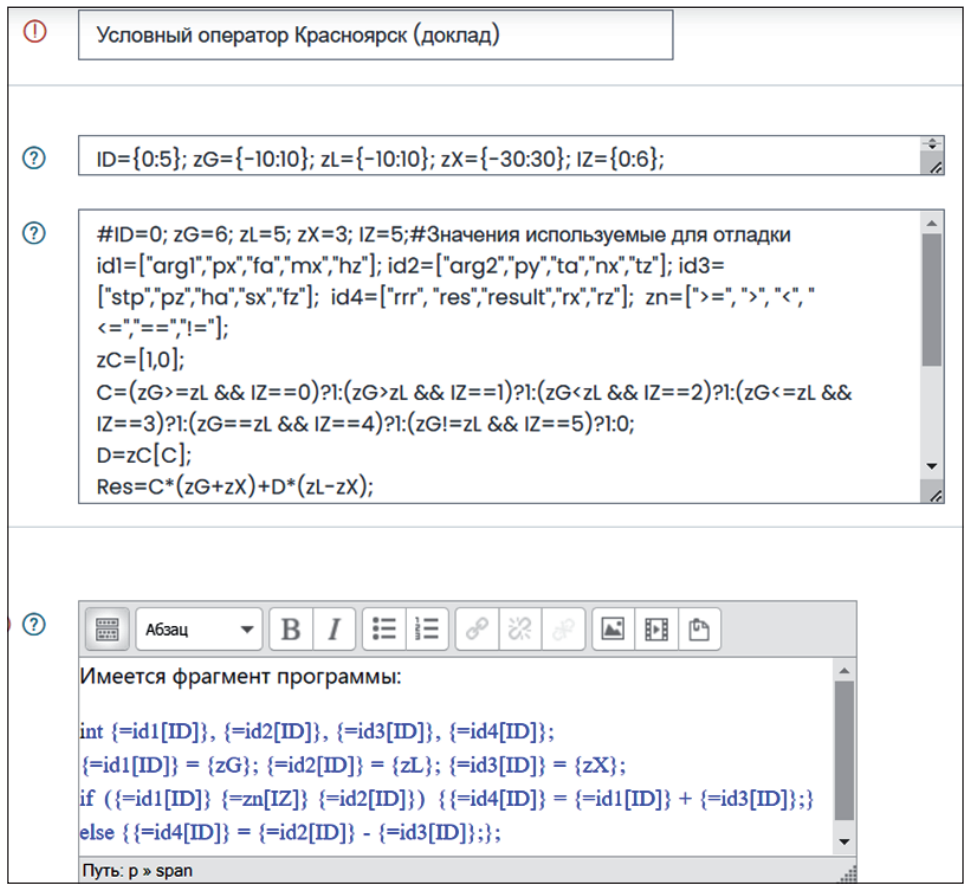


Рис. 4. Пример программирования тестового задания

Заготовка для синтеза отображаемого текста вопроса размещена в разделе «Основной вопрос» в окне «Текст вопроса» и имеет следующий вид:

```
Имеется фрагмент программы:
int {=id1[ID]}, {=id2[ID]}, {=id3[ID]},
    {=id4[ID]};
{=id1[ID]} = {zG};
{=id2[ID]} = {zL};
{=id3[ID]} = {zX};
if   ({=id1[ID]} {=zn[IZ]} {=id2[ID]})
     {{=id4[ID]} = {=id1[ID]} + {=id3[ID]};}
else {{=id4[ID]} = {=id2[ID]} - {=id3[ID]};};
```

Наконец, в разделе «Часть 1» в окне «Текст части» («Part's text») записан следующий текст (рис. 5):

```
Какое значение примет переменная {=id4[ID]}
после его выполнения?
Введите ответ с клавиатуры:
```

Как результат, при запуске теста на выполнение система генерирует задание в виде, показанном на рисунке 6.

Другие примеры генерации заданий для тестирования показаны на рисунках 7, 8.

6. Заключение

За последние несколько лет нами была создана база вопросов компьютерного тестирования по дисциплине, связанной с программированием на языке C++. В ней присутствуют все перечисленные выше типы заданий. В ее состав входят и задания типа «Вычисляемый». Общее количество вариантов генерируемых тестовых билетов превысило 7000, а использование заданий типа «Формулы» сделало их количество практически бесконечным. Это позволило применять их в процессе контроля знаний в течение обучения параллельно с теоретическими занятиями, не опасаясь академического мошенничества в виде составления и использования

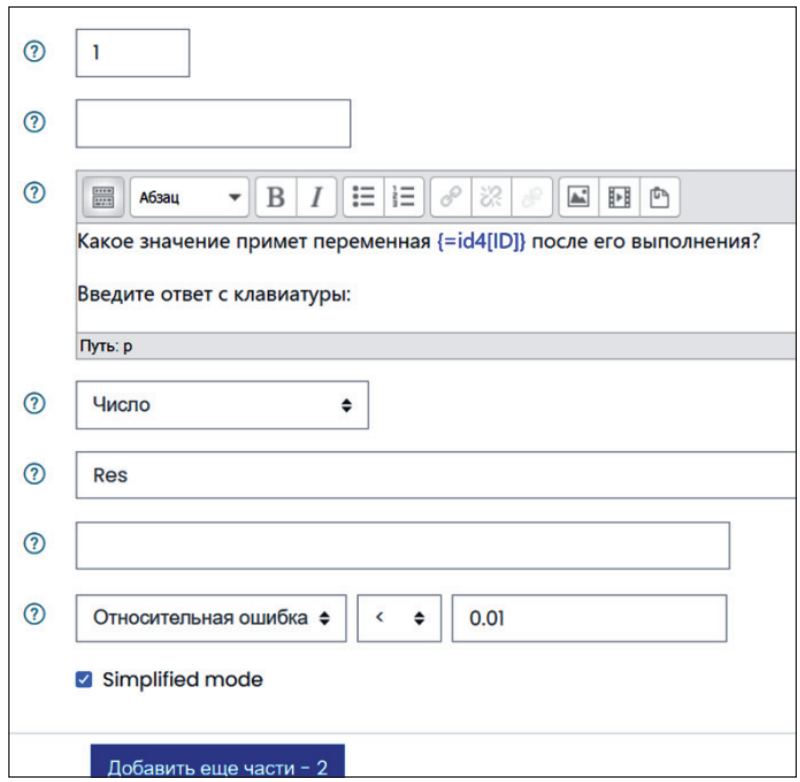


Рис. 5. Пример записи текста в разделе «Часть 1»

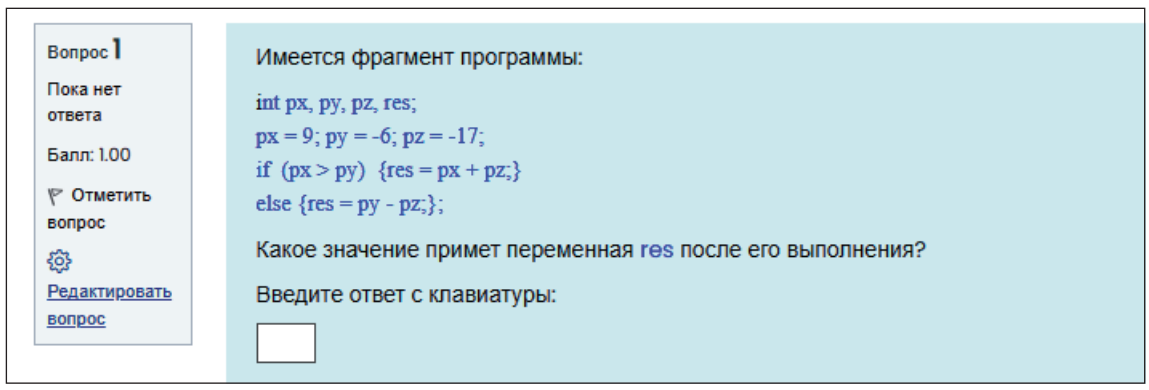


Рис. 6. Пример 1 сформированного задания для тестирования

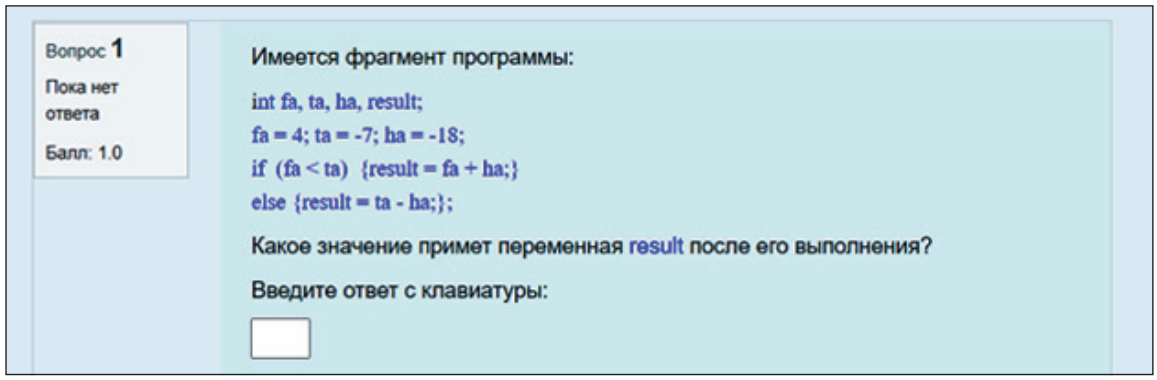


Рис. 7. Пример 2 сформированного задания для тестирования

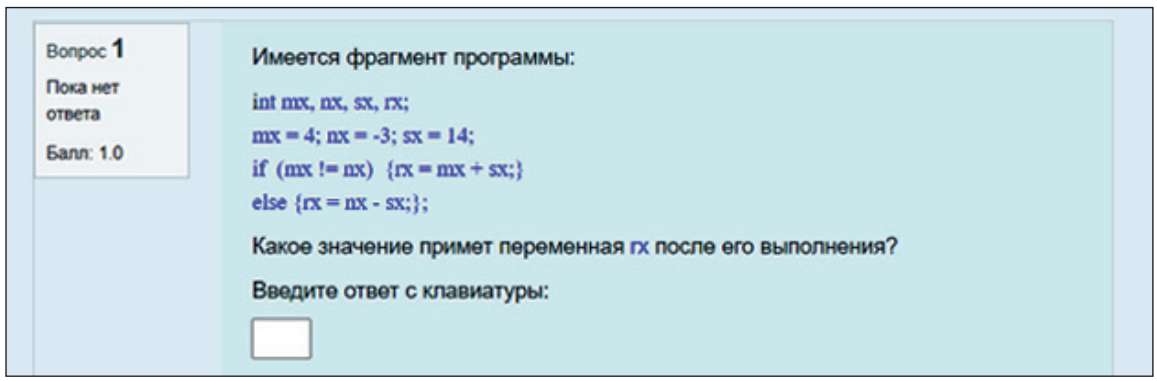


Рис. 8. Пример 3 сформированного задания для тестирования

таблиц правильных ответов. Как следствие, обучающиеся многократно встретились с типовыми заданиями, посвященными конкретному разделу дисциплины в разных формах их представлений. В результате удалось сформировать навыки и умения решения типовых задач программирования. В то же время составление и использование таблиц правильных ответов стало для обучающихся бессмысленным из-за ее гигантских размеров.

В качестве еще одного примера нами была рассмотрена задача из школьной программы X класса, размещенная на сайте [4], — первый раздел «Примеры решения задач по теме “Равномерное прямолинейное движение”». Окно «Случайные переменные» раздела «Переменные» заполнено следующим образом:

```
t1={1,2,4,8}; x1={-10:10};x2={-20:20};
```

Здесь:

- t1, x1 и x2 — имена случайных переменных, причем для упрощения устного счета переменная t1 предусматривает всего четыре варианта случайных значений.

Окно «Глобальные переменные» содержит текст вычисляемой формулы в виде выражения, записанного на внутреннем языке программирования:

```
Res = (x2 - x1)/t1;
```

В разделе «Основной вопрос» окно «Текст вопроса» имеет следующий вид:

```
Определите модуль и направление скорости
точки, если при равномерном движении
вдоль оси OX ее координата за время t1={t1}c
изменилась
```

В разделе «Часть 1» в окне «Текст части» («Part’s text») записан следующий текст:

```
t1={t1}c изменилась от x1={x1}м до x2={x2}м.
Модуль равен {_0}.
В качестве десятичного разделителя
используйте символ точка.
```

Результат программирования показан на рисунке 9.

Наконец, если смущает непривычная запись математических переменных, то можно воспользоваться

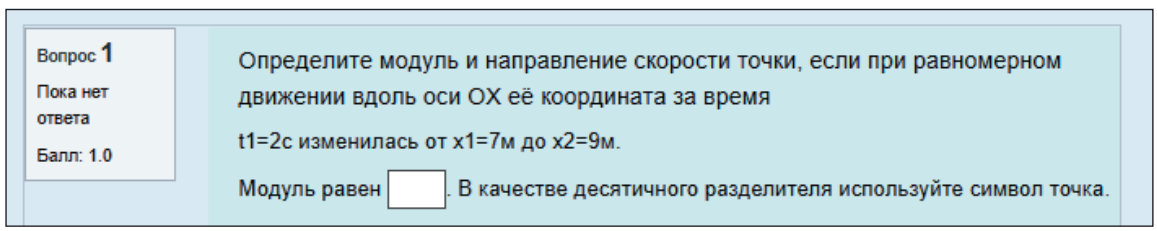


Рис. 9. Пример сформированного задания для тестирования из школьной программы

возможностями редактора TeX [7]. Для этого в разделе «Часть 1» в окне «Текст части» («Part's text») записывается та же формула в нотации TeX:

```
$$\small{t_1={t1}}$$с изменилась от $$\
small{x_1={x1}}$$м до $$\small{x_2={x2}}$$м.
```

Изложенное позволяет утверждать, что компьютерные системы тестирования могут быть использованы не только для контроля знаний, но и для формирования умений и навыков, в частности, навыков программирования.

Список источников

1. *Анисимова М. А., Бляхеров И. С., Масленников А. В., Моржов А. В.* К вопросу о проектировании оценочных средств сформированности компетенций // Высшее образование в России. 2013. № 4. С. 106–112. EDN: PZCJXP.

2. *Боцоева А. В., Гучетль С. К., Лазаренко Л. А.* К вопросу об инновационных подходах к контролю и интерактивному оцениванию знаний обучающихся в цифровой образовательной среде вуза // Проблемы современного педагогического образования. 2021. № 73-1. С. 343–346. EDN: WXLYJN.

3. *Еськин Д. Л.* Проблема академического мошенничества при оценке результатов обучения с использованием дистанционных образовательных технологий // Современные проблемы науки и образования. 2022. № 6-1. С. 68. EDN: LBHWV. DOI: 10.17513/spno.32322

4. Класс!ная физика. http://class-fizika.ru/10_a6.html

5. *Москалева О. И., Степанов А. Г.* Исследование результатов тестовой формы контроля знаний // Актуальные проблемы экономики и управления. 2017. № 2 (14). С. 84–89. EDN YUPIRZ.

6. *Степанов А. Г., Космачев В. М., Москалева О. И.* Вычисляемый вопрос в Moodle как средство проверки знаний и умений учащегося // Информатизация образования и методика электронного обучения: цифровые технологии в образовании. Материалы VI Международной научной конференции: в 3 ч. (г. Красноярск, 20–23 сентября 2022 года). Ч. 3. Красноярск: Красноярский государственный педагогический университет им. В. П. Астафьева, 2022. С. 346–350. EDN: VVQRVZ.

7. *Шмелева Е. Д., Семенова Т. В.* Академическое мошенничество студентов: учебная мотивация vs образовательная среда // Вопросы образования. 2019. № 3. С. 101–129. EDN: DRSCI. DOI: 10.17323/1814-9545-2019-3-101-129

8. Moodle: справочник. Формулы в TeX. СЭО ТГАСУ. https://ido.tsuab.ru/mod/book/view.php?id=62&chapterid=23

9. Moodle: Тип вопроса Formulas. https://www.uchiurok.ru/blog/uchitelyam/uchitelyam-informatiki/moodle-tip-voprosa-formulas.html

10. moodle-coordinate-question — Documentation.wiki: Google Code Archive. https://code.google.com/archive/p/moodle-coordinate-question/wikis/Documentation.wiki

DOI: 10.32517/2221-1993-2023-22-6-71-77

М. А. Червонный, О. В. Савина, А. И. Карпенко
Томский государственный педагогический университет, г. Томск, Россия

ART-МАТЕМАТИКА С ИСПОЛЬЗОВАНИЕМ 3D-ТЕХНОЛОГИЙ

Аннотация

В статье рассмотрены способы визуализации математических объектов с использованием инструментария 3D-технологий (3D-принтера, 3D-ручки). Наглядное представление объектов в обучении математике школьников и будущих учителей строится на основе применения STEAM-подхода (от *англ.* Science — естественные науки, Technology — технологии, Engineering — инженерия, the Arts — искусства, гуманитарные науки, Mathematics — математика). Компонент Arts позволяет исследовать возможности STEAM-подхода для раскрытия семиотического потенциала математики для обучения, визуализировать нетрадиционные объекты искусства, творчества, в которых явно и неявно есть математические основы. Исследование начинается с изучения и печати готовых моделей, а затем школьники и студенты разрабатывают собственные модели для 3D-печати. Таким образом у каждого обучающегося (школьника и студента — будущего педагога) последовательно формируется инженерная книга. Инженерная книга педагога содержит серию математических моделей, позволяющих поддерживать разный уровень освоения математики. Результаты педагогического эксперимента позволяют говорить о высокой вовлеченности детей в работу на уроках математики после применения 3D-технологий.

Ключевые слова: STEAM, 3D-моделирование, 3D-печать, обучение математике.

1. Введение

3D-моделирование и 3D-печать предоставляют современные уникальные возможности сделать математику более визуализируемой, а ее уравнения — более понятными для учащихся. Изучением этих возможностей занимаются как ученые [15], так и педагоги — методисты и учителя, обучающие школьников математике [19]. Занятия в этом направлении позволяют также вооружить выпускников школы знаниями по применению богатства математического аппарата в инженерии. В превращениях абстрактных моделей в осязаемые физические формы таится огромный потенциал для обучения и дальнейших исследований, проводимых педагогами и школьниками. Эффективным образовательным подходом, синтезирующим достижения 3D-технологий и математическое моделирование, является STEAM-образование, которое выступает одной из модификаций метода проектов, широко распространяющейся в зарубежных странах, таких как Япония, Китай, Сингапур, Финляндия, Канада и др. [4, с. 4026; 5, с. 323; 10, с. 236–239]. Характерно, что для такой страны, как Китай, обладающей значительной долей доходов и инвестиций от всей мировой экономики, рост инвестиций в 2017 году в образовательные технологии был связан с принятием концепции STEAM-образования и ее реализацией в обучении [9, с. 51–52].

Контактная информация

Червонный Михаил Александрович, доктор пед. наук, доцент, профессор кафедры физики и методики обучения физике, Томский государственный педагогический университет, г. Томск, Россия; *адрес:* 634061, Россия, г. Томск, ул. Киевская, д. 60; *e-mail:* mach@tspu.edu.ru

Савина Ольга Владимировна, магистрант физико-математического факультета, Томский государственный педагогический университет, г. Томск, Россия; *адрес:* 634061, Россия, г. Томск, ул. Киевская, д. 60; *e-mail:* olga.ssavi@gmail.com

Карпенко Анастасия Игоревна, магистрант физико-математического факультета, Томский государственный педагогический университет, г. Томск, Россия; *адрес:* 634061, Россия, г. Томск, ул. Киевская, д. 60; *e-mail:* karpenkoanastasia83@gmail.com

M. A. Chervonny, O. V. Savina, A. I. Karpenko
Tomsk State Pedagogical University, Tomsk, Russia

ART-MATHEMATICS USING 3D TECHNOLOGIES

Abstract

The article discusses ways to visualize mathematical objects using 3D technology tools (3D printer, 3D pen). The visual representation of objects in teaching mathematics to schoolchildren and future teachers is based on the use of the STEAM approach (Science, Technology, Engineering, the Arts — arts, humanities, Mathematics). The Arts component allows you to explore the possibilities of the STEAM approach to reveal the semiotic potential of mathematics for teaching, to visualize non-traditional objects of art and creativity, which explicitly and implicitly have mathematical foundations. The research begins with studying and printing ready-made models, and then schoolchildren and students develop their own models for 3D printing. Thus, each student (schoolchild and student — future teacher) consistently develops an engineering book. The teacher's engineering book contains a series of mathematical models that allow you to support different levels of mastery of mathematics. The results of the pedagogical experiment suggest that children are highly involved in work at mathematics lessons after using 3D technologies.

Keywords: STEAM, 3D modeling, 3D printing, teaching mathematics.

Математическое образование, поддержанное компьютерным моделированием и напечатанными образцами, визуализирующими такие объекты, как уравнения, геометрические фигуры, математические решения и т. п., **интегрирует несколько компонентов STEAM-подхода**, а именно:

- *S (Science)* — естественные науки. Этот компонент позволяет рассматривать с учащимися разработки новых математических моделей процессов, явлений и объектов, изучаемых в естественных науках.
- *T (Technology)* — технологии. В рамках этого компонента мы используем знания обучающихся в таких областях, как: основы программирования; работа в специальных компьютерных программах, поддерживающих технологические процессы, операционные действия на конкретных устройствах (3D-принтере, 3D-сканере, 3D-ручке); процессы применения того или иного вида пластика.
- *E (Engineering)* — инженерия. В рамках этого компонента актуализируются инженерные мыслительные операции и действия, включающие: экспериментирование, изобретательность, умение оценивать значимость инженерного решения для общества и экономики, проводить экономическое обоснование, оценку и дизайн инженерного объекта.
- *A (Arts)* — искусства, гуманитарные дисциплины. Компонент обеспечивает творческое оформление 3D-моделей, также содействует лучшему дизайнерскому решению, выступает основой для понимания красоты математики, содействует формированию у школьников более широких семиотических представлений о математике.
- *M (Mathematics)* — математика. Выступает одновременно как предмет изучения для школьников, в котором расширяются когнитивные знания и эмоционально-образные представления о математическом аппарате при помощи 3D-моделей, и как развивающийся инструмент, который позволяет математически смоделировать и физически исполнить сложные объекты науки, промышленности, быта.

Разработанные и напечатанные математические объекты можно применять на различных занятиях по математике (на уроках, внеурочных занятиях, досуговых мероприятиях) с целью повышения уровня знаний учащихся, формирования у них ясного образного представления о математических объектах, развития пространственного мышления.

Обратим внимание, что выделение компонента Arts в STEAM-подходе как творческого, художественного компонента [20] расширяет возможности применения этого подхода в математическом образовании. Например, проявляется интерес к математике у детей, которые в большей степени увлечены гуманитарными, социальными науками, предметами художественно-эстетического направления. Так, для определенного нами предмета изучения Art-математики с использованием 3D-технологий компонент Arts позволяет исследовать возможности STEAM-подхода для раскрытия семиотического потенциала математического образования [3, 18], визуализировать нетрадиционные объекты искусства, творчества, в которых есть математические основы.

Таким образом, базируясь на STEAM-подходе в образовании и достижениях 3D-технологий, в качестве **цели нашей работы** мы определили следующую: *представление математики в наглядно-образном виде с использованием 3D-технологий для развития и формирования пространственного мышления школьников*.

Ценность развития пространственного мышления у школьников определяется его социальной значимостью. Его роль значительно возросла в настоящее время в связи с высоким уровнем использования в быту, профессиональной деятельности, науке и технике графического моделирования, позволяющего наглядно описывать исследуемые теоретические зависимости, представлять результаты практической деятельности.

Из собственного опыта мы можем сделать вывод о том, что у современных школьников наблюдаются проблемы с пространственным мышлением, а именно, у учеников X класса наблюдаются проблемы с задачами на сечение, а у учащихся среднего звена возникают сомнения в правдивости изложенного геометрического материала.

Развитие пространственного мышления, как наиболее сложного вида образного мышления, выступает необходимым условием интеллектуального развития и учебной успешности обучающихся в школе и вузе [12, 13, 22]. Особую роль в развитии пространственного мышления школьников отводят математике [7, 13, 16]. Роль пространственного мышления в овладении различными видами деятельности значительно возросла в связи с высоким использованием в науке и технике графического моделирования, позволяющего наглядно описывать исследуемые теоретические зависимости или представлять результаты творческой практической деятельности [1, 6, 8, 14]. Пространственное мышление признается базовой когнитивной способностью, позволяющей школьникам осваивать программы STEAM-образования [21].

Эффективным инструментом развития пространственного мышления в урочной и внеурочной деятельности могут выступить методики и технологии 3D-моделирования математических объектов.

Для достижения поставленной цели были обозначены следующие **задачи:**

- разработка и печать математических 3D-моделей;
- разработка сценария занятия с применением 3D-технологий;
- применение разработок на занятиях по математике (на уроках, в кружках);
- рефлексия проведенной деятельности, апробация результатов в профессиональных сообществах.

2. Материалы и методы

В нашем исследовании использовались следующие методы:

- поэтапное тестирование учащихся;
- метод трехмерного моделирования с помощью программы Autodesk 123D design;
- методы генерации G-кода для различных моделей 3D-принтеров с помощью программы Ultimaker Cura, в частности моделирование методом послойного наплавления.

3. Исследование

Описанные выше методы были использованы для 3D-моделирования и 3D-печати математических объектов. Моделирование и печать выполнялись на уроках математики в V—VI классах школы № 49 г. Томска, а также на занятиях по общеобразовательной общеразвивающей программе дополнительного образования на базе центра дополнительного физико-математического и естественно-научного образования Томского государственного педагогического университета.

Педагогический эксперимент проводился в несколько этапов:

1) поиск со школьниками идей математического моделирования и готовых компьютерных моделей;
2) обсуждение идей (мозговой штурм) с учащимися;
3) поиск оригинальных 3D-моделей для печати или обсуждение собственных разработок учащихся;
4) компьютерное моделирование и печать 3D-моделей;
5) создание учащимися инженерных книг по выбранной тематике;
6) обсуждение со студентами — будущими учителями комплекса сценариев занятий для школьников по разработанным моделям;
7) рефлексия проведенных занятий будущими учителями.

Например, на этапе 3 — поиска готовых и моделирования собственных математических объектов — были взяты готовые модели и разработаны новые модели куба. В частности, была использована готовая модель тессеракта — четырехмерного гиперкуба — аналога обычного трехмерного куба в четырехмерном пространстве. Этот куб был напечатан в дальнейшем на занятиях с использованием 3D-принтера (фото 1).

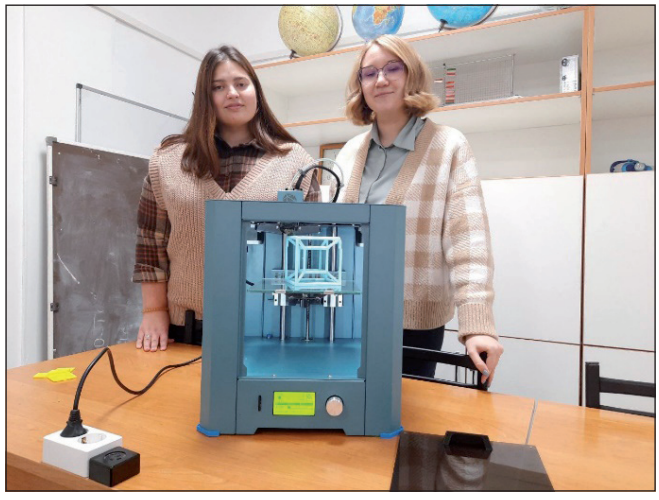


Фото 1. Тессеракт, напечатанный на 3D-принтере по готовой модели

Был разработан и изготовлен кубик Блума с заданиями на гранях для изучения и закрепления свойств куба (рис. 1).

При этом школьники обсудили и проверили различные варианты инженерного решения для прототипа кубика Блума. В частности, обсуждались решения: переворот кубика в зависимости от его размеров, размер шрифта, размещение надписей на соответствующих сторонах модели, материал для печати и др.

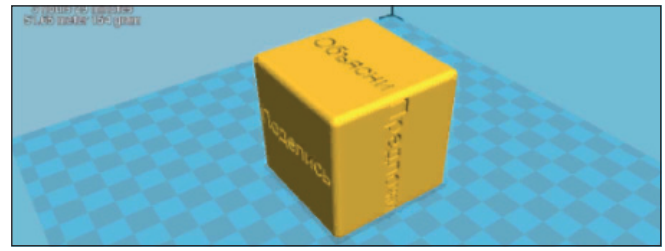


Рис. 1. Модель кубика Блума с заданиями на гранях

С помощью 3D-ручки были изготовлены кубики, в которых проводились различные сечения (фото 2, 3).

Фото 2. Моделирование математических разверток с помощью 3D-ручки учениками V класса

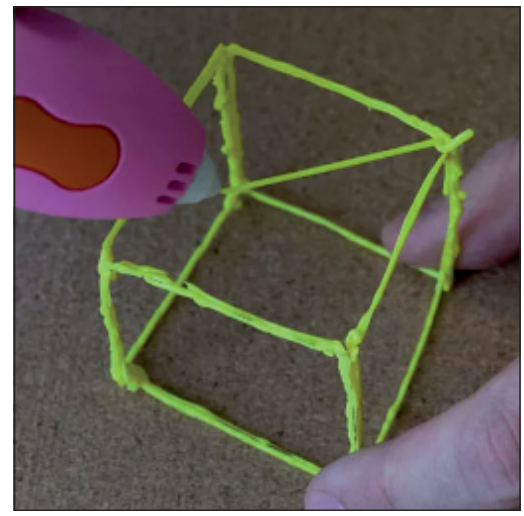


Фото 3. Создание школьником диагонального сечения куба с помощью 3D-ручки

На этапе 5 — создания инженерных книг — создавались книги как школьниками непосредственно по математическим объектам, выбранным ими для моделирования и печати, так и студентами — будущими учителями, которые создавали более сложные инженерные решения, разрабатывая серию моделей, расширяющую

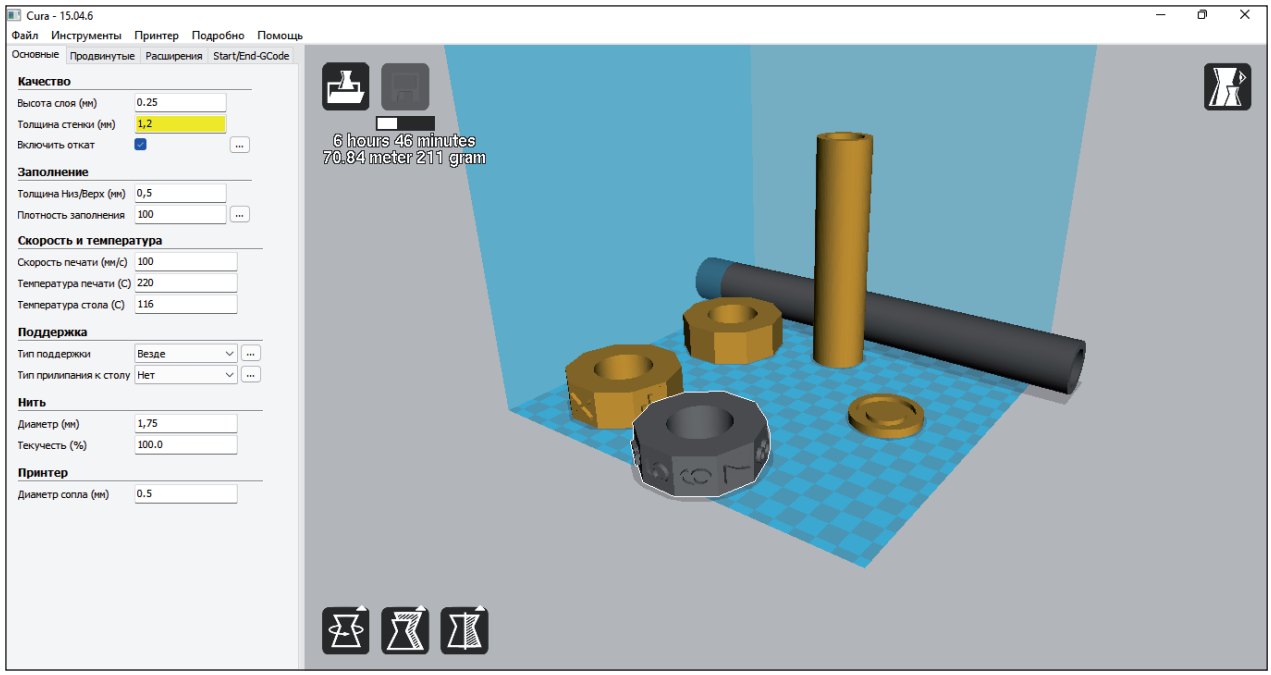


Рис. 2. Подготовка компьютерной модели математического калькулятора

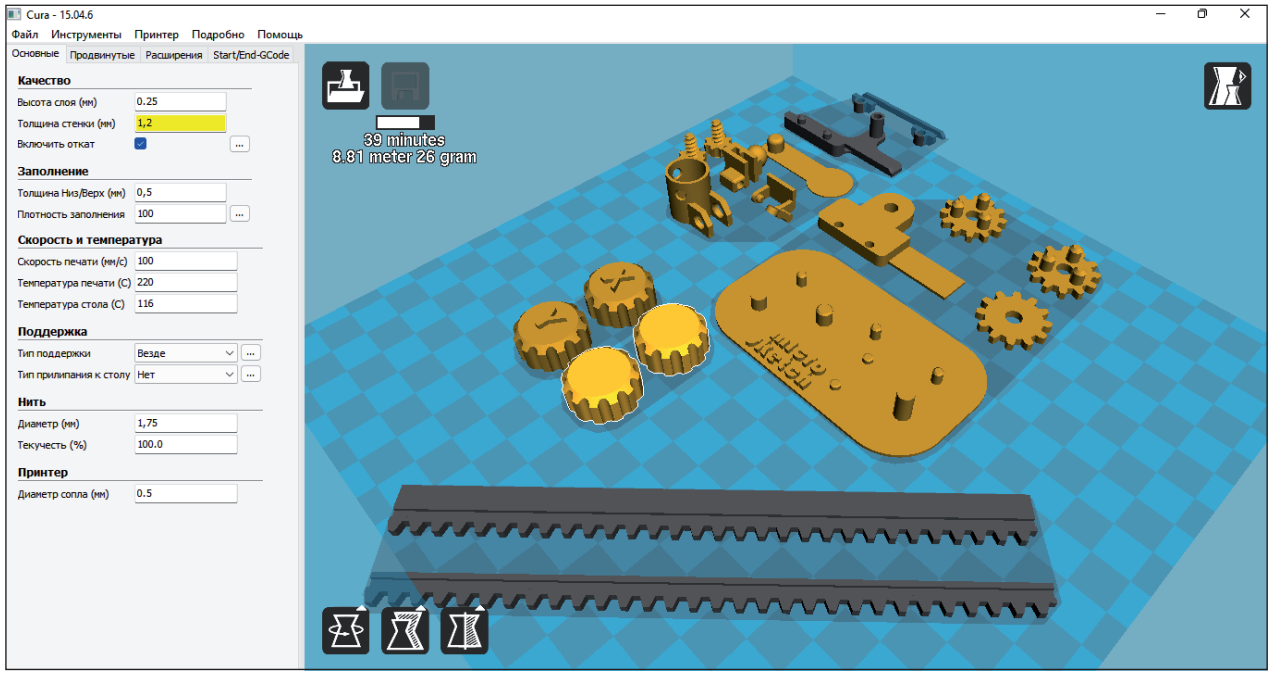


Рис. 3. Подготовка компьютерной модели доски для координатного черчения

представление о математическом объекте. В последнем случае обеспечивалась визуализация математических решений разного уровня сложности, что важно для реализации сценариев разных занятий.

В ходе занятий со школьниками были подготовлены и напечатаны модели: математического калькулятора (рис. 2), доски для координатного черчения (рис. 3), теоремы Пифагора (фото 4), кривых постоянной ширины [2, с. 19–26] и др.

При осуществлении определенных видов деятельности на занятиях со школьниками выявились некоторые проблемы у обучающихся, для решения которых были разработаны предложения по их устранению (см. табл.).

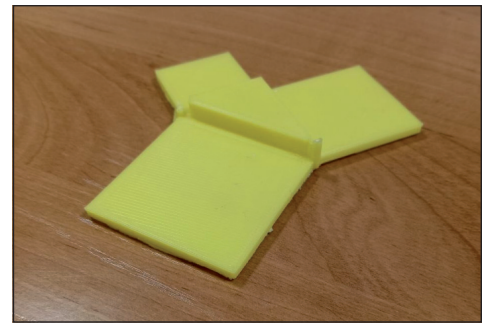


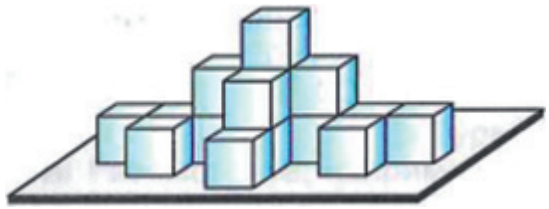
Фото 4. Напечатанная на 3D-принтере модель теоремы Пифагора

Таблица

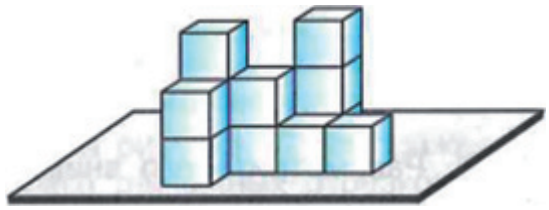
Проблемы обучающихся, выявленные при определенных видах деятельности на уроках

Проводимая на занятиях деятельность с учащимися	Выявленные в ходе работы проблемы	Предложения по устранению проблем
Обсуждение идей (мозговой штурм) с учащимися. Постановка проблем по визуализации математических объектов. Демонстрация интересных готовых моделей, напечатанных ранее учителем или другими школьниками	Быстрая увлекаемость учащихся, желание поскорее повторить самые сложные модели без предварительной подготовки: изучения свойств объекта, построения компьютерной модели, изучения специальных программ	Корректировка деятельности учащихся, плавный перевод на обсуждение проблем, с которыми можно столкнуться при построении компьютерной модели. Рассказ о поступательной деятельности инженера и правилах построения инженерной книги
Изучение объемных фигур	У обучающихся возникали сложности с переносом моделей в реальную жизнь, в частности, проявлялись проблемы пространственного представления модели	Выполнение упражнений, направленных на развитие пространственного мышления (построение видов модели спереди, сверху, слева) (рис. 4)
Изготовление при помощи 3D-ручки развертки объемных фигур	При моделировании обучающиеся с трудом представляли, как должна выглядеть деталь в конечном итоге	Применение компьютерных программ по геометрическому конструированию
3D-моделирование и печать на 3D-принтере (математического калькулятора, призм, пирамид и др.)	У обучающихся возникали сложности с моделированием объемных фигур	Моделирование изображений именно изображений? в различных программах, а также создание объемных моделей из бумаги при помощи ножниц и клея (фото 5)

Для устранения проблем у учащихся в пространственных представлениях применялись упражнения, направленные на развитие пространственного мышления [17]. Так, со школьниками выполняли упражнения, в которых разбирали построение вида спереди, сверху, слева для фигур из кубиков, представленных на рисунке (рис. 4) [17, с. 38].



а

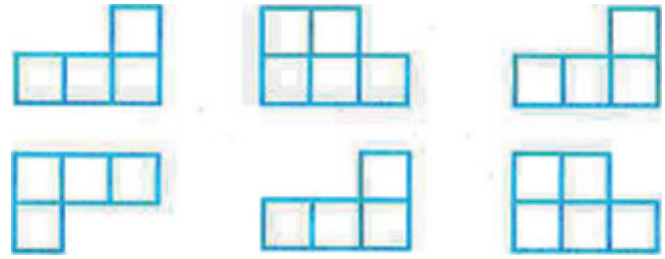


б

Рис. 4. Фигуры из кубиков для построения вида спереди, сверху, слева

Также учащимся предлагалось рассмотреть конструкции из кубиков, представленные в двух видах — вид спереди и вид сверху (рис. 5), и изобразить для каждой конструкции вид слева [14, с. 39]. Для проверки можно было собрать конструкции из кубиков у себя на парте.

Для устранения дефицита итогового пространственного представления деталей применялись учебные программы — так называемые информационные источники сложных структур по геометрическому конструированию в плоскости и пространстве для V класса, расположенные в Единой коллекции цифровых образовательных ресурсов [11].



а *б* *в*

Рис. 5. Конструкции, представленные в двух видах — вид спереди и вид сверху

Для устранения проблемы со сложностью моделирования объемных фигур применялись разные программы для моделирования (Blender, FreeCAD, Tinkercad), а также разные способы создания промежуточных моделей при помощи бумаги, ножниц и клея (фото 5).

Таким образом, для достижения поставленной нами цели мы **решили следующие организационные и дидактические задачи:**

- педагогический поиск и анализ в научной и научно-популярной литературе и интернет-источниках готовых математических моделей, а также разработка собственных математических 3D-моделей и их последующая печать для использования на соответствующих этапах уроков;
- разработка педагогических сценариев занятий по математике по различным темам с применением 3D-технологий;

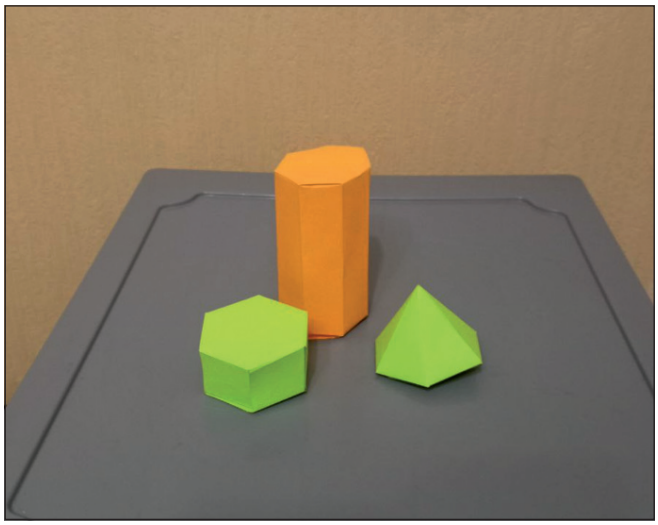


Фото 5. Модели объемных геометрических фигур, изготовленные школьниками из бумаги

- реализация разработок сценариев и использование моделей на уроках и внеурочных занятиях в школе, а также в кружках по проектной деятельности и математике в центре дополнительного образования;
- рефлексия проведенного педагогического эксперимента, апробация результатов в профессиональных сообществах.

4. Результаты

При проверке вовлеченности детей в работу на уроках математики до и после применения 3D-технологий получены результаты, представленные на рисунке 6.

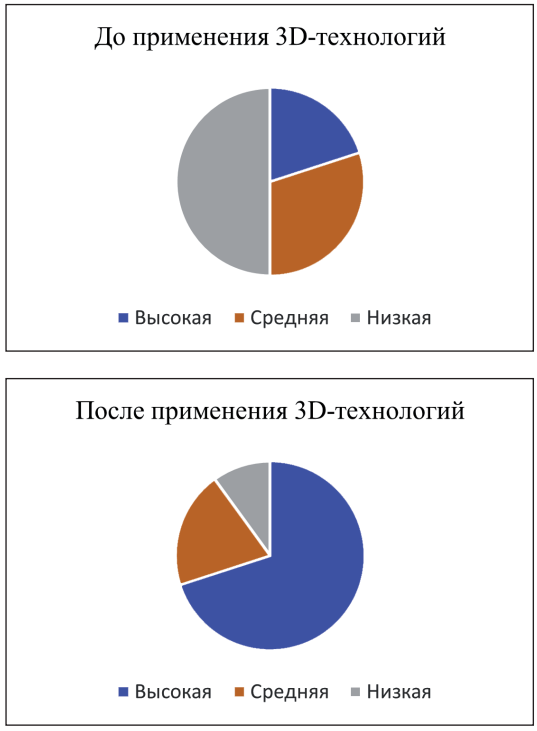


Рис. 6. Анализ вовлеченности учащихся в работу на занятиях по математике до и после применения 3D-технологий.

5. Выводы

В математике, особенно на уроках геометрии, нельзя обойтись без наглядного материала. Визуализация на уроках служит таким наглядным материалом, который поясняет различные теоремы и определения, а также позволяет закрепить образами в памяти полученные математические сведения. Распечатанные модели позволяют иллюстрировать понятия из различных областей математики (представление функций, тригонометрических и логарифмических выражений и т. д.). Совместная работа студентов — будущих педагогов со школьниками над созданием моделей посредством инструментария 3D-технологий и последующая печать разработанных моделей решают несколько образовательных задач:

- обучение 3D-технологиям — освоение процесса 3D-моделирования и 3D-печати (на 3D-принтере и с помощью 3D-ручки), создание дизайна модели;
- повышение мотивации к изучению компонентов STEAM, понимание связей между естественными науками, точными науками, технологиями, инженерией и искусством (гуманитарными науками);
- развитие инженерных способностей (изобретательности, умения проводить оценку инженерного решения — социальную, экономическую, дизайнерскую);
- развитие у школьников когнитивных способностей;
- повышение интереса учащихся к задачам, которые решают ученые-математики.

Помимо создания и реализации программ по математике для школы с использованием 3D-технологий, построенных на традиционном содержании математики, можно создавать новые математические модели, повышающие интерес к математике, а также реализовывать поступление этих моделей в учреждения общего и дополнительного образования. С этой целью происходит встраивание модуля по изучению 3D-печати математических объектов в курсы повышения квалификации учителей, которые сопровождаются методическими материалами, включающими в себя распечатанные модели для визуализации математических решений.

Список источников

1. *Авдоньев Е. Я.* Графические модели — начало науки технического творчества // Строительство и техногенная безопасность. 2014. № 50. С. 25–28.

2. *Акияма Дж., Руис М.-Дж.* Страна математических чудес / пер. с англ. М. И. Бабиковой. М.: Изд-во МЦНМО, 2009. 240 с.

3. *Акуленко И. А.* Семиотический компонент методической компетенции будущего учителя математики // Вестник Брянского госуниверситета. 2013. № 1. С. 74–76.

4. *Алдошина М. И.* Эффективные технологии формирования компетенций в современном университетском образовании // Профессиональное образование в современном мире. 2020. Т. 10. № 3. С. 4022–4030. DOI: 10.15372/PEMW20200312

5. *Анисимова Т. И., Шатунова Ф. М., Сабирова Ф. М.* STEAM-образование как инновационная технология для Индустрии 4.0 // Научный диалог. 2018. № 11. С. 322–332. EDN: YOWSNV. DOI: 10.24224/2227-1295-2018-11-322-332

6. *Баева Е. И.* Графическое моделирование как смыслотехника в учебном процессе // Северо-Кавказский психологический вестник. 2014. № 4. С. 21–27.

7. *Бреус И. А.* Теоретико-методические аспекты проблемы развития пространственного мышления школьников // Проблемы современного педагогического образования. 2018. № 61-4. С. 38–41. EDN: YUKMWD.

8. *Брикалова Е. А., Горюнова И. А., Корягина О. М.* Моделирование как средство развития пространственного мышления // Актуальные проблемы гуманитарных и естественных наук. 2016. № 10-1. С. 51–55. EDN: WWHRTR.

9. *Гулева М. А.* Новые тенденции в развитии образовательной сферы в Китае // Азия и Африка сегодня. 2018. № 6 (731). С. 51–55. EDN: URAYSU. DOI: 10.7868/S0321507518060077

10. *Дорофеева А. С.* Анализ развития STEAM-образования в России и за рубежом // Известия Балтийской государственной академии рыбопромыслового флота: психолого-педагогические науки. 2020. № 4 (54). С. 236–242. EDN: AEVPSI. DOI: 10.46845/2071-5331-2020-4-54-236-242.

11. Единая коллекция цифровых образовательных ресурсов. http://school-collection.edu.ru/catalog/rubr/b33a1431-1b0f-4794-b2a7-83cd3b9d7bca/104715/

12. *Зеппова Н. Н.* Развитие пространственного мышления школьников — залог успешного изучения точных дисциплин в вузе // Вестник Иркутского государственного технического университета. 2012. № 6 (65). С. 231–237.

13. *Коногорская С. А.* Особенности развития компонентов пространственного мышления школьников на разных ступенях общего образования // Ученые записки Российского государственного социального университета. 2019. Т. 18. № 4 (153). С. 91–99. EDN: PONMAK. DOI: 10.17922/2071-5323-2019-18-4-91-99

14. *Марченко М. Н.* Графическая деятельность и компьютерные технологии в профессиональной подготовке будущих дизайнеров // Историческая и социально-образовательная мысль. 2013. № 5. С. 115–118. EDN: RPXOAP.

15. Математические уравнения можно увидеть, благодаря 3D-печати. https://3dtoday.ru/industry/matematicheskie-uravneniya-mozhno-uvidet-blagodarya-3d-pechati.html

16. *Нурмагомедов Д. М., Камилова Ш. Д.* Проблема формирования пространственных представлений у младших школьников при обучении математике // Известия Дагестанского государственного педагогического университета. 2012. № 3 (20). С. 70–74. EDN: QBPFBL.

17. *Панчищина В. А., Гельфман Э. Г., Ксенева В. Н. и др.* Математика. 5–6 классы. Наглядная геометрия. М.: Просвещение. 2023. 177 с.

18. *Тарасенкова Н. А.* Проблемы и перспективы реализации семиотического подхода в математическом образовании // Дидактика математики. 2005. № 24. С. 132–136.

19. *Шаповалов Д. Н., Чернобабова К. В.* Применение 3D-технологий для решения стереометрических задач // Современные инновации. 2019. № 5 (33). С. 32–35.

20. *Шатунова О. В.* STEM- и STEAM-образование: от технологии к искусству // Актуальные направления современной науки, образования и технологий. Материалы Всероссийской научно-практической конференции (г. Чебоксары, 23 апреля 2020 года). Чебоксары: Экспертно-методический центр, 2020. С. 259–263.

21. *Buckley J., Seery N., Canty D.* Investigating the use of spatial reasoning strategies in geometric problem solving // International Journal of Technology and Design Education. 2019. DOI:10.1007/s10798-018-9446-3

22. *Reid C., Buckley J., Seery N., Dunbar R.* Examining the relationship between spatial ability and cognitive load during complex problem solving. PATT 39. 2022. P. 246.

DOI: 10.32517/2221-1993-2023-22-6-78-84

О. М. Корчажкина
Федеральный исследовательский центр «Информатика и управление» Российской академии наук, г. Москва, Россия

АНАЛИЗ ПОВЕДЕНИЯ СИСТЕМЫ АЛГЕБРАИЧЕСКИХ УРАВНЕНИЙ С ПАРАМЕТРОМ МЕТОДАМИ ДИНАМИЧЕСКОЙ ГЕОМЕТРИИ

Аннотация
В статье описываются аналитико-графические способы исследования поведения системы двух алгебраических уравнений с параметром через поиск особых (граничных) точек в области допустимых значений параметра. Эти точки являются границами интервалов, причем внутри каждого интервала система уравнений имеет сходные свойства, а ее поведение подчиняется единым правилам. Таким образом устанавливается полная картина решения задачи, подтверждаемая затем визуальными методами динамической геометрии, которые воспроизводятся в интерактивной творческой среде «1С:Математический конструктор». Предлагаемый прием особых точек может быть использован при выполнении тренировочных упражнений для подготовки к олимпиадам и профильному ЕГЭ по математике как ключ к анализу всех возможных ситуаций, возникающих в конкретной задаче с параметром.
Ключевые слова: особые точки, параметры, задачи с параметром, динамические чертежи, «1С:Математический конструктор».

1. Введение

Обзор результатов сдачи ЕГЭ по профильной математике за последние несколько лет показывает, что лишь 1 % учащихся приступает к решению задач с параметром. Причем только одной десятой части самых отважных старшеклассников из этой группы удается найти правильное решение [10, с. 10], поскольку эти задачи требуют высокого уровня математической подготовки, включающего, прежде всего, умение тщательно анализировать условие задачи, т. е. верно проводить рассуждения, проверки и вспомогательные преобразования.

Основной причиной, по которой учащиеся не рискуют приступать к выполнению задания, включающего параметры, можно считать сложную аналитическую запись условия, содержащую модули, радикалы, тригонометрические, логарифмические или экспоненциальные функции, другие степенные выражения или даже неравенства. При этом учащиеся мысленно по прочно устоявшейся привычке начинают рассматривать возможный путь решения уравнения или системы уравнений «в лоб», т. е. пытаются найти корни, принимая параметр за некую константу. Тогда как вместо этого необходимо проводить тщательный анализ условия задачи, позволяющий увидеть, что скрывается за сложными «наворотами» в аналитической записи, и при «расшифровке» этой записи правильно использовать сам параметр.

2. Общие правила решения алгебраических задач с параметром

Задачи с параметром могут быть представлены разными способами, например:

- найдите решение задачи для любого значения параметра во всей области допустимых значений или в пределах ограниченного множества;

Контактная информация

Корчажкина Ольга Максимовна, канд. тех. наук, ст. научный сотрудник, Институт кибернетики и образовательной информатики им. А. И. Берга, Федеральный исследовательский центр «Информатика и управление» Российской академии наук, г. Москва, Россия; *адрес:* 119333, Россия, г. Москва, ул. Вавилова, д. 44, к. 2; *e-mail:* olgakomax@gmail.com

O. M. Korchazhkina
Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, Moscow, Russia

ANALYZING THE BEHAVIOUR OF A SYSTEM OF ALGEBRAIC EQUATIONS WITH A PARAMETER USING DYNAMIC GEOMETRY METHODS

Abstract
The article proposes analytical and graphic techniques for investigating the behaviour of a system of two algebraic equations with a parameter through the search for singular (i. e. boundary) points in the range of acceptable parameter values. The singular points are the boundaries of the ranges, and within each range, the system has similar properties, as well as its behaviour obeys the same rules. Thus, the method establishes a complete picture of the solution of the problem, which then can be confirmed by visual ways of dynamic geometry reproduced in the interactive creative environment 1C:MathKit. The algorithm of singular points proposed in the paper can be used when performing training exercises to prepare for the Olympiads and the profile USE in mathematics as a key to analyzing all possible situations that arise in a particular task with a parameter.
Keywords: special points, singular points, parameters, tasks with parameter, problems with parameter, dynamic drawings, dynamic figures, 1C:MathKit.

- определите значения параметра для заданного числа решений задачи;
- определите число решений задачи при заданных значениях параметра;
- определите значения параметра, для которых выполняются заданные условия, накладываемые на решения задачи.

Несмотря на такое многообразие вариантов, проблемы понимания (или непонимания), возникающие в процессе анализа условия задачи, одинаковы при любых формулировках задачи с параметром. Чаще всего учащиеся не видят различий между способами определения значений параметра и способами определения неизвестных величин (корней) уравнения или системы уравнений, тогда как **параметры обладают особыми функциями**, отличными от функций неизвестных величин:

- параметры создают условия задачи, т. е. в задаче они играют доминирующую роль, а неизвестные величины — второстепенную;
- параметры могут вносить дополнительные ограничения в условие задачи, влияющие на выбор способа ее решения;
- граничные значения параметра, или *особые точки*, делят область его допустимых значений на интервалы, формируя в каждом интервале из первоначального условия различные задачи, т. е. параметры создают семейство задач, различных по виду (аналитической записи), способам решения и числу корней.

Общие правила решения задач с параметром состоят в следующем:

- задачи с параметром — многовариантные задачи, требующие учета определенных ограничений (условий существования решений), которые накладываются на корни уравнения или системы уравнений конкретным значением параметра;
- решение задачи с параметром следует начинать не с определения корней, а с исследования значений параметра, отвечающих заданным условиям;
- направление поиска решения задачи задается значением параметра при заданных условиях;
- анализ условия задачи с параметром осуществляется путем рассуждений с использованием логических операторов или их синтаксических значений на естественном языке: «если ..., то/тогда ...»; «что если ...»; «и ..., и ...»; «или ..., или ...»; «без ...»; «не ...»; «кроме ...» и пр.

Эти правила лежат в основе всех оптимизационных процедур (процедур упрощения условия задачи), представляющих собой специальные методы решения задач с параметром.

К ним относятся, например, использование свойств монотонности [5, с. 84–90; 8, с. 50–56; 9, с. 109–123], симметрии [5, с. 245–257; 8, с. 85–87], инвариантности [9, с. 147–165], экстремальных значений функций [4; 5, с. 231–243; 8, с. 88–91], аффинных преобразований [5, с. 97–140], свойств квадратичной функции [5, с. 159–212; 8, с. 34–36; 9, с. 88–108], приемов динамической математики [1–3], а также многие другие способы оптимизации.

3. Прием особых точек для анализа поведения системы двух алгебраических уравнений с параметром

Если рассматривать наиболее часто встречающиеся задачи с параметром в виде системы двух алгебраических уравнений, то они обладают одним полезным свойством, использование которого на подготовительном этапе значительно облегчает поиск основного решения. Подвергая оба уравнения различным алгебраическим преобразованиям и рассматривая параметр как неизвестную величину, можно найти все особые точки, разбивающие область допустимых значений параметра на ряд интервалов, где система ведет себя однотипно. Причем переходы через эти граничные точки приводят к принципиальным изменениям в условиях задачи. Выявление особых точек в области допустимых значений параметра помогает составить полную картину поведения системы с параметром, учитывающую все возможные варианты решения задачи.

Таким образом, особые точки являются теми значениями параметра, которые упрощают аналитическое решение задачи. Их можно определить, приравнивая к нулю дискриминант квадратного уравнения, другое подкоренное выражение или свободный член многочлена, содержащие параметр. Кроме того, особые точки могут обнаруживаться в вырожденных случаях, когда, например, окружность стягивается в точку, а гипербола сливается с собственными асимптотами.

Прием особых точек при решении задач с параметром предполагает выбор следующей стратегии:

- в аналитической записи условия задачи выявить признаки, которые могут свидетельствовать о наличии особых точек, определить эти значения параметра и зафиксировать вызванные ими качественные изменения задачной ситуации (например, вид аналитической записи задачи и число решений);
- разделить область допустимых значений параметра на интервалы с учетом особых точек и проследить изменение условий задачи для каждого интервала;
- для проверки правильности результата найти при заданных значениях параметра соответствующие корни системы уравнений, чтобы проверить, действительно ли не нарушены условия задачи;
- для подтверждения проведенного анализа условия задачи, для понимания общей картины поведения параметра, а также для подтверждения полученного аналитического решения задачи *использовать инструменты динамической геометрии*.

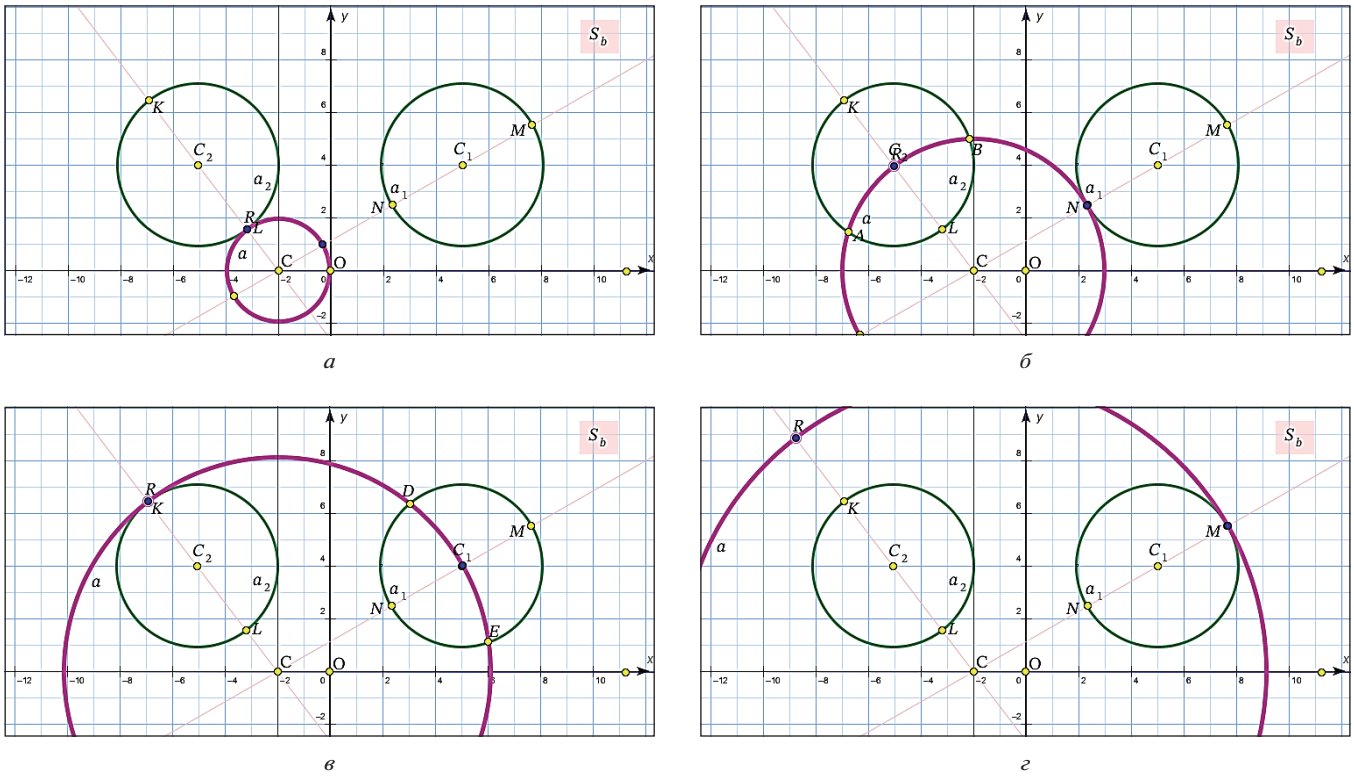
4. Примеры анализа поведения системы двух алгебраических уравнений с параметром

Рассмотрим несколько примеров анализа поведения системы двух алгебраических уравнений с параметром во всей области допустимых значений параметра.

Задача 1.

Исследуйте поведение системы уравнений:

$$\begin{cases} (|x|-5)^2+(y-4)^2=9, \\ (x+2)^2+y^2=p^2 \end{cases} \qquad (1)$$



а *б* *в* *г*

Рис. 1. Расположение особых точек (значений параметра) системы уравнений (1) на координатной плоскости: а) $p = 2$; б) $p \approx 5{,}1$; в) $p = 8$; г) $p \approx 11{,}1$

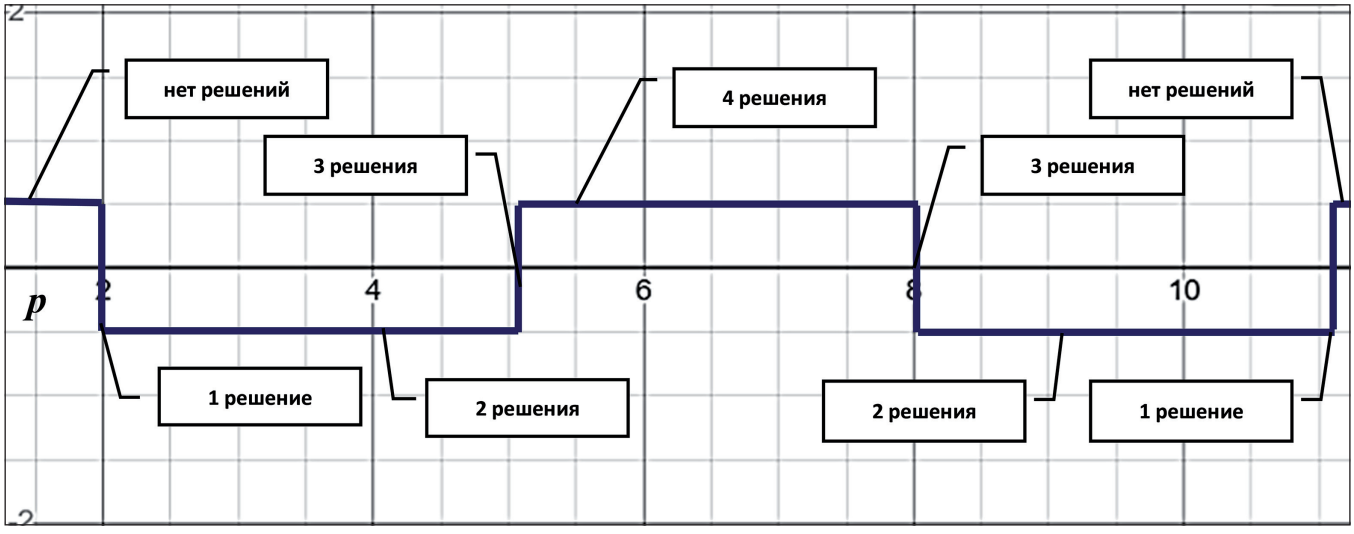


Рис. 2. Шкала изменений параметра p и число корней в каждом интервале

во всей области допустимых значений параметра p (графическое решение задачи см. в [6, с. 19–20], а с помощью динамических чертежей — в интерактивной творческой среде «1С:Математический конструктор» [1–3]).

Решение.

Раскрывая знак модуля, получаем две системы уравнений:

$$\begin{cases}(x-5)^2+(y-4)^2=9,\\(x+2)^2+y^2=p^2;\end{cases} \qquad \begin{cases}(x+5)^2+(y-4)^2=9,\\(x+2)^2+y^2=p^2,\end{cases}$$

в которых второе уравнение задает окружность a радиуса p с центром в точке C (–2; 0), а первые уравнения — две окружности a_1 и a_2 радиуса 3 с центрами в точках C_1 (5; 4) и C_2 (–5; 4) соответственно; эти окружности расположены симметрично относительно оси ординат (рис. 1, *а–г*). Особыми точками системы (1) являются значения параметра p, при которых происходит изменение числа ее решений.

Эти особые точки находят через значения радиусов окружностей a и a_1; a и a_2. Учитывая, что расстояние между центрами окружностей $CC_1 = \sqrt{7^2+4^2} \approx 8{,}1$ и $CC_2 = \sqrt{3^2+4^2} = 5$, получаем (см. рис. 1, *а–г*):

$$p_1 = CC_2 - 3 = 2;$$

$$p_2 = CC_1 - 3 \approx 5{,}1;$$

$$p_3 = CC_2 + 3 = 8;$$

$$p_4 = CC_1 + 3 \approx 11{,}1.$$

С помощью динамических чертежей* могут быть проверены аналитические выкладки и продемонстрирована полная картина поведения системы уравнений (см. рис. 1, *а—г*). На рисунке 2 представлена шкала изменений параметра p с указанием числа корней в каждом интервале его значений.

* Здесь и далее динамические чертежи построены с помощью интерактивной творческой среды «1С:Математический конструктор».

Задача 2.

Исследуйте поведение системы уравнений:

$$\begin{cases} x^2 + y^2 = p^2, \\ xy = p(p-3) \end{cases} \qquad (2)$$

во всей области допустимых значений параметра p.

Решение.

Очевидно, что особыми точками в области допустимых значений параметра будут $p = 0$ и $p = 3$, при которых гипербола, задаваемая вторым уравнением системы (2), вырождается в собственные асимптоты, совпадающие с осями координат. А при $p = 0$ окружность, которая описывается первым уравнением, стягивается в точку, совпадающую с началом координат.

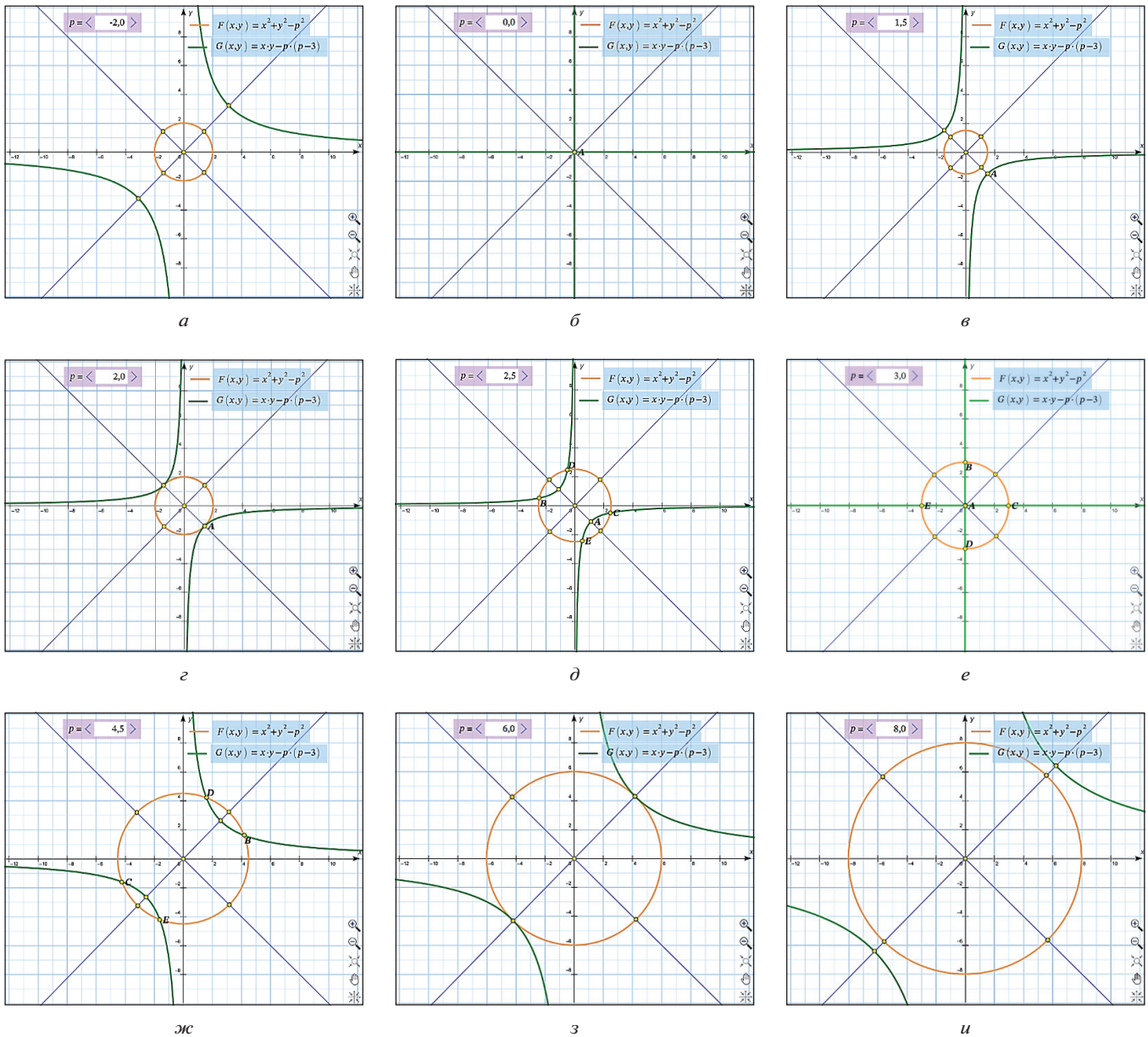


Рис. 3. Поведение системы уравнений (2) в интервалах изменения параметра: а) нет решений при $p < 0$; б) одно решение при $p = 0$; в) нет решений при $0 < p < 2$; г) два решения при $p = 2$; д) четыре решения при $2 < p < 3$; е) четыре решения при $p = 3$; ж) четыре решения при $3 < p < 6$; з) два решения при $p = 6$; и) нет решений при $p > 6$

Чтобы проверить, существуют ли дополнительные особые точки в области допустимых значений параметра, преобразуем систему (2) следующим образом: прибавим к первому уравнению удвоенное второе. При этом получим еще одну особую точку $p = 2$:

$$\begin{cases}(x+y)^2 = 3p(p-2),\\ xy = p(p-3).\end{cases}$$

Если же из первого уравнения вычесть удвоенное второе, то получим четвертую особую точку $p = 6$:

$$\begin{cases}(x-y)^2 = p(6-p),\\ xy = p(p-3).\end{cases}$$

Таким образом, особые точки 0, 2, 3 и 6 разбили область допустимых значений параметра p на пять интервалов: $(-\infty; 0)$, $(0; 2)$, $(2; 3)$, $(3; 6)$ и $(6; +\infty)$, в пределах которых система уравнений (2) имеет однотипное поведение. А сами особые точки служат границами, где происходит качественный скачок в аналитической записи системы и, следовательно, в ее решении.

Итак, несложно установить, что система (2) имеет следующие решения в особых точках:

- при $p_1 = 0$ система:

$$\begin{cases}x^2 + y^2 = 0,\\ xy = 0\end{cases}$$

имеет вырожденное решение (0; 0);

- при $p_2 = 2$ система:

$$\begin{cases}x^2 + y^2 = 4,\\ xy = -2\end{cases}$$

имеет два решения: $(\pm\sqrt{2}; \mp\sqrt{2})$;

- при $p_3 = 3$ система:

$$\begin{cases}x^2 + y^2 = 9,\\ xy = 0\end{cases}$$

имеет четыре решения: $(\pm 3; 0)$ и $(0; \pm 3)$;

- при $p_4 = 6$ система:

$$\begin{cases}x^2 + y^2 = 36,\\ xy = 18\end{cases}$$

имеет два решения: $(\pm\sqrt{18}; \pm\sqrt{18})$.

Если требуется исследовать поведение системы (2) в интервалах изменения параметра между особыми точками, то это можно сделать методом динамических чертежей [1–3], получив при этом результаты, изображенные на рисунке 3.

Исследования показали, что при переходе значения параметра через 0 (когда первое уравнение вырождается в начало координат, а второе — в оси координат) ветви гиперболы меняют свое расположение на координатной плоскости, перемещаясь из I и III квадрантов во II и IV квадранты. При этом два корня (как точки касания гипербол и окружности) появляются при $p = 2$, а затем в области допустимых значений параметра от 2 до 6 система имеет четыре решения, как и при промежуточном значении $p = 3$. Однако в последнем случае центральная окружность остается, а ветви гиперболы вырождаются в оси координат.

Задача 3.

Исследуйте поведение системы уравнений [7, с. 85]:

$$\begin{cases}xy + x + y = p,\\ x^2 + y^2 = 8\end{cases} \qquad (3)$$

во всей области допустимых значений параметра p.

Решение.

Первое уравнение системы (3) описывает гиперболу, а второе — окружность радиуса $2\sqrt{2}$.

Первое, очевидное, ***значение особой точки p* = 0**. Для определения других особых точек прибавим ко второму уравнению системы (3) удвоенное первое. Добавим по 1 к обеим частям полученного выражения:

$$(x + y + 1)^2 = 2p + 9, \text{ или } x + y + 1 = \pm\sqrt{2p+9}. \qquad (4)$$

Приравняв к нулю подкоренное выражение, найдем ***вторую особую точку*: *p* = –4,5**.

Поиск остальных особых точек не столь очевиден, однако, проделав необходимые алгебраические преобразования, можно добиться полной картины допустимых значений параметра, определяющих поведение системы.

Итак, из (4) имеем:

$$x + y = \pm\sqrt{2p+9} - 1,$$

а из первого уравнения (3):

$$xy = p - (x + y). \qquad (5)$$

Рассмотрим для определенности случай:

$$x + y = \sqrt{2p+9} - 1.$$

Тогда:

$$x = \sqrt{2p+9} - 1 - y.$$

Подставляя выражения для $(x + y)$ и x в (5), получим квадратное уравнение относительно y:

$$y^2 + (1 - \sqrt{2p+9})\,y + (p + 1 - \sqrt{2p+9}) = 0.$$

Обозначив $\sqrt{2p+9}$ через t и проведя необходимые преобразования, получим следующее уравнение:

$$t^2 - 2t - 15 = 0.$$

Его корнями являются $t_1 = 5$ и $t_2 = -3$, дающие $p = 8$ и $p = 0$ соответственно.

Таким образом, мы получили еще одну, ***третью, особую точку p* = 8** и подтвердили существование ранее найденной точки $p = 0$ дополнительным способом. Отметим, что случай $x + y = -\sqrt{2p+9} - 1$ приводит при $t_1 = -5$ и $t_2 = 3$ к аналогичному результату.

Найдем четвертую особую точку. Для этого приведем первое уравнение системы (3) для гиперболы к виду для $y = \dfrac{p-x}{1+x}$, где $x \neq -1$ и для $x = \dfrac{p-y}{1+y}$, где $y \neq -1$.

Очевидно, что при $p = -1$ ветви гиперболы вырождаются в собственные асимптоты: $x = -1$ и $y = -1$. Следовательно, ***четвертая особая точка* — это *p* = –1**.

Итак, подставляя в систему (3) найденные значения параметра p, можно получить следующие результаты, которые подтверждаются динамическими чертежами, представленными на рисунке 4:

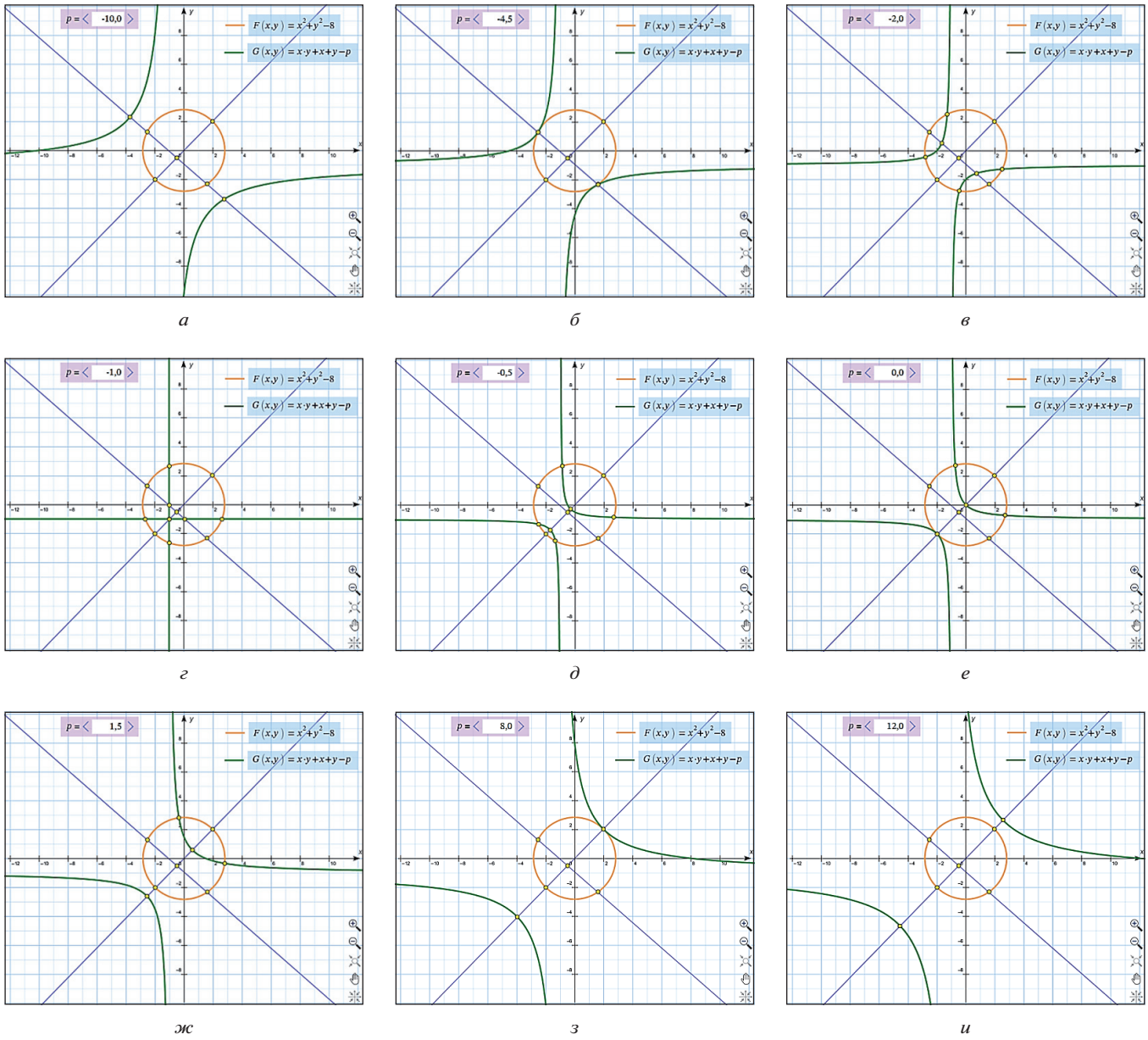


Рис. 4. Поведение системы уравнений (3) в интервалах изменения параметра:
а) нет решений при $p < -4,5$; б) два решения при $p = -4,5$; в) четыре решения при $-4,5 < p < -1$;
г) четыре решения при $p = -1$; д) четыре решения при $-1 < p < 0$; е) три решения при $p = 0$;
ж) два решения при $0 < p < 8$; з) одно решение при $p = 8$; и) нет решений при $p > 8$

- при $p_1 = -4,5$ система:

$$\begin{cases} xy + x + y = -4,5, \\ x^2 + y^2 = 8 \end{cases}$$

имеет два решения:

$$\left(\frac{-1+\sqrt{15}}{2}; \frac{-1-\sqrt{15}}{2}\right) \text{ и } \left(\frac{-1-\sqrt{15}}{2}; \frac{-1+\sqrt{15}}{2}\right);$$

- при $p_2 = -1$ система:

$$\begin{cases} xy + x + y = -1, \\ x^2 + y^2 = 8 \end{cases}$$

имеет четыре решения:

$$(-1; +\sqrt{7}); (+\sqrt{7}; -1); (-\sqrt{7}; -1); (-1; -\sqrt{7});$$

- при $p_3 = 0$ система:

$$\begin{cases} xy + x + y = 0, \\ x^2 + y^2 = 8 \end{cases}$$

имеет три решения:

$$(-2; -2), (1 - \sqrt{3}; 1 + \sqrt{3}) \text{ и } (1 + \sqrt{3}; 1 - \sqrt{3});$$

- при $p_4 = 8$ система:

$$\begin{cases} xy + x + y = 8, \\ x^2 + y^2 = 8 \end{cases}$$

имеет одно решение: (2; 2).

Динамическое поведение системы уравнений (3) в зависимости от значений параметра выглядит следующим образом:

- при $p < -4{,}5$ система (3) не имеет решений (рис. 4, *а*);
- при $p = -4{,}5$ система (3) имеет два решения, образованные точками касания окружности и ветвей гиперболы (рис. 4, *б*);
- при $-4{,}5 < p < 0$ система (3) имеет четыре решения как точки пересечения ветвей гиперболы и окружности (рис. 4, *в*—*д*);
- при переходе параметра через $p = -1$ ветви гиперболы меняют локализацию на координатной плоскости, а при $p = -1$ они вырождаются в свои асимптоты: $x = -1$ и $y = -1$ (рис. 4, *г*);
- при $p = 0$ система (3) имеет три решения: два — как точки пересечения правой ветви гиперболы и окружности и одно — как точка касания левой ветви гиперболы и окружности (рис. 4, *е*);
- при $0 < p < 8$ система (3) имеет два решения — как точки пересечения правой ветви гиперболы и окружности (рис. 4, *ж*);
- при $p = 8$ система (3) имеет одно решение — как точка касания правой ветви гиперболы и окружности (рис. 4, *з*);
- при $p > 8$ система (3) не имеет решений (рис. 4, *и*).

5. Заключение

Описанный в статье прием особых точек помогает учащимся выявить набор «подводных камней» задачи с параметром через исследование поведения системы во всей области допустимых значений параметра, а также понять, чем функции параметра отличаются от роли корней системы уравнений. Таким образом, через общий обзор задачной ситуации складывается полная картина влияния параметра на конечный результат.

Тем не менее следует отметить, что основная трудность при использовании приема особых точек состоит в том, что число этих точек заранее неизвестно. Однако если запись задачи включает квадратичную функцию или имеет модули, то, вероятнее всего, число корней может варьироваться от одного до четырех, что предполагает наличие четырех особых точек, разбивающих область допустимых значений параметра на пять интервалов. В пределах внутренних интервалов задача, как правило, имеет решение, а в пределах правого и левого интервалов — при значениях параметра, уходящих в $\pm\infty$, — решения не существует.

Трудности, с которыми встречается большинство учащихся, полностью снимаются, если при решении подобных нетривиальных алгебраических задач использовать подвижные чертежи, выполняемые с помощью инструментов динамической геометрии «1С:Математический конструктор». Эти инструменты могут работать как автономно — непосредственно для решения задач с параметром в режиме «сериала», демонстрирующего переходные процессы во всей области допустимых значений параметра, так и как вспомогательный способ оценки общей картины задачной ситуации и проверки правильности решения, найденного алгебраическим путем.

Список источников

1. 1С:Математический конструктор 10.0. Интерактивная творческая среда для создания математических моделей. URL: https://obr.1c.ru/mathkit/

2. 1С:Урок — Библиотека интерактивных материалов. Виртуальные лаборатории по математике, 7—11 кл. // 1С:Урок. https://urok.1c.ru/library/mathematics/virtualnye_laboratorii_po_matematike_7_11_kl/

3. 1С:Урок — Главная страница // 1С:Урок. https://urok.1c.ru/share/model/9dd2671fdc61e07c580e3b87194bdb6f/

4. *Бичегкуев М. С., Олисаев Э. Г.* Наименьшее и наибольшее значение функций в задачах с параметром // Математика в школе. 2022. № 8. С. 8—15. EDN: MTFKBE. DOI: 10.47639/0130-9358_2022_8_8

5. *Горнштейн П. И., Полонский В. Б., Якир М. С.* Задачи с параметрами. 3-е изд., допол. и перераб. М.: ИЛЕКСА, 2007. 328 с.

6. Демонстрационный вариант контрольных измерительных материалов единого государственного экзамена 2023 года по математике. Профильный уровень. М.: ФИПИ, 2022. 23 с. https://fipi.ru/ege/demoversii-specifikacii-kodifikatory#!/tab/151883967-2

7. *Ларин С. В.* Использование компьютерной анимации при обучении решению задач с параметрами // Математика в школе. 2022. № 7. С. 43—48. EDN: ORLAOG. DOI: 10.47639/0130-9358_2022_7_43

8. *Шевкин А. В.* Математика. Трудные задания ЕГЭ. Задачи с параметром: профильный уровень: учебное пособие. 2-е изд., стер. М.: Просвещение, 2022. 96 с.

9. *Шестаков С. А.* ЕГЭ 2022. Математика. Задачи с параметром. Задача 17 (профильный уровень) / под ред. И. В. Ященко. М.: МЦНМО, 2022. 288 с.

10. *Ященко И. В., Высоцкий И. Р., Семенов А. В.* Методические рекомендации для учителей, подготовленные на основе анализа типичных ошибок участников ЕГЭ 2021 года по математике // Педагогические измерения. 2021. № 4. С. 3—28. EDN: XMTHLT. https://doc.fipi.ru/zhurnal-fipi/PI-2021-04.pdf

НАПЕЧАТАНО В 2023 ГОДУ

АКТУАЛЬНЫЕ ПРОБЛЕМЫ МЕТОДИКИ ОБУЧЕНИЯ ИНФОРМАТИКЕ В СОВРЕМЕННОЙ ШКОЛЕ

ОБЩИЕ ВОПРОСЫ

КОНКУРС ИНФО-2022

УРОКИ ИНФОРМАТИКИ

ИНТЕГРИРОВАННЫЕ УРОКИ

МЕТОДИЧЕСКАЯ КОПИЛКА

ЗАДАЧИ

ИТОГОВАЯ АТТЕСТАЦИЯ ПО ИНФОРМАТИКЕ

СРЕДСТВА ИКТ В УЧЕБНОМ ПРОЦЕССЕ

ЗАРУБЕЖНЫЙ ОПЫТ

ИНФОРМАТИКА И ИКТ В НАЧАЛЬНОЙ ШКОЛЕ

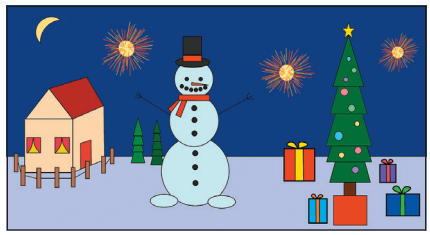
ИСТОРИЯ ИНФОРМАТИКИ

ТОЧКА ЗРЕНИЯ

КОНКУРСЫ

КОНКУРСЫ

В оформлении обложки данного номера журнала «Информатика в школе» использованы работы следующих авторов:



Софья Тишина,
ученица средней общеобразовательной школы № 6, г. Мегион, Ханты-Мансийский автономный округ — Югра, Россия

Н. В. Лукьянова,
преподаватель Барнаульского государственного педагогического колледжа имени Василия Константиновича Штильке, г. Барнаул, Алтайский край, Россия

Елизавета Бабошкина,
ученица средней общеобразовательной школы № 1, р.п. Дергачи, Саратовская область, Россия

Камилла Гаянова,
студентка Мамадышского политехнического колледжа, г. Мамадыш, Республика Татарстан, Россия

Екатерина Афанасьева,
студентка Барнаульского государственного педагогического колледжа имени Василия Константиновича Штильке, г. Барнаул, Алтайский край, Россия

Алина Бабина,
студентка Барнаульского государственного педагогического колледжа имени Василия Константиновича Штильке, г. Барнаул, Алтайский край, Россия

Н. С. Салий,
студентка Мордовского государственного педагогического университета имени М. Е. Евсевьева, г. Саранск, Республика Мордовия, Россия

Н. С. Поздяйкин,
студент Мордовского государственного педагогического университета имени М. Е. Евсевьева, г. Саранск, Республика Мордовия, Россия

С. Г. Пузиновская,
учитель информатики средней школы № 4, г. Дзержинск, Минская область, Беларусь

ПОДПИСКА

Журнал «Информатика в школе»

Индекс подписки
на 1-е полугодие 2024 года
(«Урал-Пресс», «АРЗИ» и другие агентства подписки)

81407

Периодичность выхода: 3 номера в полугодие (февраль, апрель, июнь)
Объем — не менее 88 полос

Редакционная стоимость — 500 руб.

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/

Обратите внимание: требования к оформлению файла рукописи — **разные** для журналов «Информатика и образование» и «Информатика в школе». При подготовке файла рукописи ориентируйтесь на требования для того журнала, в который вы представляете статью. Если вы представляете рукопись в оба журнала (для публикации в одном из изданий — на усмотрение редакции), при ее оформлении следует руководствоваться требованиями к оформлению рукописи в журнал «Информатика и образование».

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: +7 (495) 140-19-86

1С:Оценка качества образования. Школа

Трехуровневая система оценки качества образования

Единые подходы к внутренней и внешней оценке качества образования

Прогнозирование результатов итоговой государственной аттестации

Соответствие актуальным нормативным документам

Оперативное управление качеством образования

Программно-методическая система предназначена для оценки качества освоения образовательной программы на следующих уровнях: оценка индивидуальные достижений обучающихся, внутриклассное и внутришкольное оценивание.

Программа разработана на основе методики ведущего научного сотрудника Института управления образованием РАО, кандидата педагогических наук, доцента Н.Б. Фоминой.

Функциональные возможности

- Оценка индивидуального уровня освоения ФГОС.
- Аналитические расчеты успеваемости учащихся и качества образования.
- Анализ объективности оценивания индивидуальных образовательных достижений обучающихся.
- Персональный контроль профессиональной деятельности педагога с выявлением проблемных компонентов.
- Прогноз повышения качества образования, включая результаты государственных экзаменов (ОГЭ и ЕГЭ).

Преимущества использования

- Обеспечение индивидуализации образования, выявление способностей и предрасположенности каждого учащегося к определенному спектру дисциплин.
- Предоставление педагогам необходимой информации для практической деятельности (корректировка программ, выбор технологий обучения, выявление проблем в обучении).
- Предоставление руководителю данных, необходимых для анализа работы педагогического коллектива.