МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное учреждение высшего образования
«КРАСНОЯРСКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.П. АСТАФЬЕВА»
(КГПУ им. В.П. Астафьева)

Институт Математики, физики и информатики

Кафедра информатики и информационных технологий в образовании

Типишкина Валерия Евгеньевна

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Система заданий для автоматизированного контроля учебных достижений обучающихся старших классов по программированию

Направление подготовки / специальность: 44.03.05 Педагогическое образование

Направленность(профиль) образовательной программы: Математика и Информатика

ДОПУСКАЮ К ЗАЩИТЕ

Зав.кафедрой: д-р пед. наук, профессор
Пак Н.И.
_____
(дата, подпись)

Руководитель: канд. пед. наук, доцент
Яшина И.А.
_____
(дата, подпись)

Дата защиты 23.06.2026

Обучающийся: Типишкина В. Е.
_____
(дата, подпись)

Оценка хорошо

Красноярск 2026

Содержание

Введение

Актуальность темы. Современный этап развития общества, характеризующийся стремительной цифровизацией большинства сфер жизни, выдвигает новые требования к качеству школьного образования, в особенности в области информатики и программирования. Программирование становится ключевым элементом функциональной грамотности, формирующим алгоритмическое мышление, креативность и навыки проектной деятельности.

В связи с этим возникает острая необходимость в эффективном контроле учебных достижений обучающихся, который должен быть не только регулярным, но и объективным, своевременным и диагностичным. Традиционные формы контроля, такие как устные опросы и письменные работы, зачастую не справляются с этими задачами в контексте обучения программированию, так как не позволяют оперативно проверить практические навыки написания, отладки и анализа кода у всего класса одновременно.

Проведенный анализ существующей практики выявил ряд проблем:

1. Низкая оперативность: ручная проверка программных кодов отнимает у педагога значительное время.
2. Субъективность оценки: критерии оценки часто размыты и сильно зависят от экспертного мнения учителя.
3. Отсутствие персонализации: сложно организовать контроль с учетом индивидуального темпа и уровня подготовки каждого обучающегося.
4. Дефицит качественного инструментария: несмотря на наличие различных тестовых сред и онлайн-курсов, наблюдается недостаток целостных систем заданий, ориентированных именно на школьную программу и позволяющих комплексно оценивать как теоретические знания, так и практические умения в области программирования.

Разрешение этих противоречий видится в разработке и внедрении системы заданий для автоматизированного контроля. Это определяет актуальность

данного исследования, которая обусловлена социальным заказом на повышение качества IT-образования и объективной потребностью школьной практики в эффективных инструментах педагогического измерения.

Объект исследования: процесс контроля учебных достижений обучающихся старших классов по программированию.

Предмет исследования: система заданий для автоматизированного контроля учебных достижений обучающихся старших классов по программированию на базе плагина CodeRunner.

Цель исследования: теоретически обосновать, разработать и апробировать систему заданий для автоматизированного контроля учебных достижений старшеклассников по программированию на основе связки LMS Moodle и плагина CodeRunner.

Задачи исследования:

1. Проанализировать психолого-педагогическую и методическую литературу по проблемам контроля и оценки образовательных результатов в области программирования.
2. Выявить особенности и определить структуру учебных достижений обучающихся старших классов по программированию.
3. Изучить и проанализировать существующие программные средства и платформы для автоматизированного контроля знаний и умений по программированию.
4. Разработать модель системы заданий, включающую классификацию типов заданий, критерии оценки и требования к их автоматизированной проверке.
5. Реализовать систему заданий на базе плагина CodeRunner, апробировать его в учебном процессе и оценить его эффективность на основе опроса студентов и преподавателей.

Методы исследования:

1. *Теоретические:* анализ научной литературы, учебно-методической документации, систематизация и обобщение, моделирование.
2. *Эмпирические:* педагогическое наблюдение

3. *Статистические:* методы математической обработки результатов эксперимента.

Методологические основы исследования:

1. Системный подход, позволивший рассмотреть контроль учебных достижений как целостную систему.
2. Деятельностный подход, в рамках которого знания и умения формируются и проявляются в деятельности.
3. Теория поэтапного формирования умственных действий, являющаяся основой для проектирования заданий разного уровня сложности.

Апробация и база исследования. Апробация разработанной системы заданий проводилась в ходе учебного процесса. Так как внедрение нового инструмента затрагивает обе стороны образовательного процесса, для оценки эффективности разработанной системы нами был организован и проведен опрос. В опросе приняли участие студенты 1-го курса и преподаватели информатики.

Опрос студентов позволил оценить понятность формулировок задач, удобство работы с плагинов CodeRunner, а так же то, насколько понятны и полезны для них сообщения об ошибках. Опрос преподавателей был направлен на выявление того, насколько разработанная система заданий снижает их нагрузку при проверке работ, соответствует ли она школьной программе и удобно ли ей пользоваться для отслеживания успеваемости класса. Полученные ответы легли в основу выводов о практической значимости разработанной модели системы заданий.

Практическая значимость исследования заключается в том, что разработанная система заданий для плагина CodeRunner может быть непосредственно использована учителями информатики в их профессиональной деятельности для организации текущего, рубежного и итогового контроля.

ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ КОНТРОЛЯ УЧЕБНЫХ ДОСТИЖЕНИЙ В ОБУЧЕНИИ ПРОГРАММИРОВАНИЮ СТАРШЕКЛАССНИКОВ

Современный этап развития общества, характеризующийся стремительной цифровизацией большинства сфер жизни людей, выдвигает новые требования к качеству школьного образования, в особенности в области информатики и программирования. Программирование становится ключевым элементом функциональной грамотности, формирующим алгоритмическое мышление, креативность и навыки проектной деятельности [13]. В условиях глобальной информатизации образования традиционные подходы к организации учебного процесса переживают значительные изменения, которые затрагивают все его компоненты: цели, методы, средства, содержание и формы организации процесса обучения. Однако контроль учебных достижений остается одним из наиболее консервативных. Именно контроль обеспечивает обратную связь, позволяющую корректировать образовательный маршрут, оценивать эффективность применяемых методик и фиксировать результаты освоения образовательной программы.

В связи с этим возникает необходимость в эффективном контроле учебных достижений обучающихся, который должен быть регулярным, но в то же время объективным, своевременным и диагностичным. Традиционные формы контроля, такие как устный и письменный опрос, чаще не справляются с этими задачами в контексте уроков информатики, так как не позволяют оперативно проверить практические навыки написания, отладки и анализа кода у всего класса одновременно. Проведенный анализ существующей практики контроля знаний на уроках информатики выявил ряд проблем, среди которых низкая оперативность ручной проверки, субъективность оценки, отсутствие персонализации и дефицит качественного инструментария, ориентированного на школьную программу.

Решение выявленных противоречий видится в разработке и внедрении системы заданий для автоматизированного контроля. Однако контроль не может быть сведен лишь к технической процедуре проверки кода. Он должен быть

педагогически обоснованным инструментом развития личности школьника. Именно поэтому первая глава ВКР посвящена комплексному теоретическому анализу проблематики контроля учебных достижений в условиях цифровой образовательной среды, специфике диагностики компетенций в области программирования и требованиям к программным платформам, внедренных в школьный процесс.

1.1. Педагогические подходы к оценке образовательных результатов в цифровой среде

Без контроля учебных достижений не обходится ни один педагогический процесс. Это тот самый стержень, что скрепляет этапы обучения, не давая образовательной системе рассыпаться на фрагменты. Контроль здесь - не галочка в журнале и не бюрократический ритуал. Традиционная дидактика рассматривает его куда шире: как целый комплекс процедур по проверке, оцениванию и учёту знаний, умений и навыков.

По сути, речь идёт о сложном механизме обратной связи. Он замыкает контур «учитель - ученик». Без него преподаватель действует вслепую: непонятно, что из пройденного материала осело в головах, а что прошло мимо. Где возникли пробуксовки? В каком месте коллективная или индивидуальная мысль дала сбой? Ответы на эти вопросы контроль и призван давать.

Игнорировать эту функцию - значит лишиться самого рычага управления обучением. Корректировать программу на лету, менять темп, возвращаться к сложным темам - всё это становится возможным только тогда, когда педагог получает от класса внятный сигнал. Нет сигнала - нет и осмысленного движения вперёд. Просто имитация процесса. Именно поэтому контроль нельзя сводить к примитивному выставлению отметок, это живой инструмент настройки всего учебного механизма.

Отечественная педагогическая традиция, опирающаяся на фундаментальные труды её классиков, выделяет несколько ключевых функций контроля, совокупность которых обеспечивает системное влияние на качество образовательного процесса.

Первой и базовой среди них является функция контролирующая. Её суть - в установлении актуального состояния знаний обучающихся: фиксация уровня усвоения материала и степени его соответствия нормативным предписаниям образовательного стандарта. Объективная значимость данной функции обусловлена тем, что при отсутствии точной и своевременной информации о текущей подготовке обучающихся любое планирование последующей работы утрачивает фактологическую опору. Вместе с тем сведение всего многообразия контрольных процедур к одной лишь этой функции представляет собой концептуальное упрощение, укоренённое в авторитарной педагогической модели. В такой парадигме оценка инструментализируется: она начинает обслуживать не развитие, а селекцию - отбор и распределение учащихся по формально замеренным показателям успешности.

Второй по значимости выступает обучающая функция. Её специфика раскрывается в том, что сам процесс контроля при грамотной методической организации становится формой обучения. Готовясь к проверочному испытанию или выполняя тестовые задания, обучающийся вынужден повторять пройденное, закреплять материал, выстраивать смысловые связи между разрозненными понятиями. Происходит активизация познавательных механизмов: память извлекает ранее усвоенное, мышление систематизирует его в новых контекстах. Методически выверенное контрольное задание, таким образом, перерастает узкие рамки проверки и превращается в инструмент закрепления знаний - при условии, разумеется, что оно составлено не формально, а с опорой на принципы развивающего обучения.

Третья функция - диагностическая. В отличие от контролирующей, которая лишь констатирует факт наличия или отсутствия знания, диагностическая функция нацелена на вскрытие причинной механики успехов и неудач. Она призвана ответить не на вопрос «знает или не знает?», а на куда более сложный: «в чём именно состоит пробел, какого типа допущена ошибка и чем она обусловлена?». Предметом анализа здесь становится структура незнания. Такой подход принципиально меняет стратегию коррекции: вместо механической

пересдачи незачтённого раздела педагог получает возможность адресно воздействовать на первопричину затруднения - будь то несформированность понятийного аппарата, методический просчёт преподавателя или когнитивные особенности конкретного обучающегося.

Четвёртая функция - прогностическая. Опираясь на массив данных, полученных в ходе контроля, педагог получает основания для построения вероятностного прогноза: как будет развиваться обучающийся в дальнейшем, каков его познавательный потенциал и в каких точках учебного маршрута с наибольшей вероятностью возникнут препятствия. Прогноз этот не носит фатального характера, однако служит ориентиром для превентивных педагогических мер.

Пятая функция - развивающая. Корректно организованный контроль не угнетает, а стимулирует познавательную активность. Он создаёт ситуации, требующие волевого усилия, мобилизации ответственности и проявления самостоятельности. Через систематическое переживание ситуации оценивания у обучающегося формируются регулятивные навыки, имеющие ценность далеко за пределами конкретной учебной дисциплины.

Шестая функция - ориентирующая. Результаты контроля дают обучающемуся материал для рефлексивной самооценки: он получает возможность увидеть контуры собственного знания, выделить сильные и слабые зоны своей подготовки и на этой основе скорректировать распределение усилий при подготовке к последующим этапам обучения. Контроль здесь работает как своеобразное зеркало, возвращающее ученику объективированный образ его учебных достижений.

Наконец, седьмая функция - воспитательная. Систематическое участие в контрольных процедурах формирует у обучающегося определённое ценностное отношение к учебной деятельности в целом. Вырабатываются дисциплинированность, честность, способность к объективной оценке собственных результатов. Контроль исподволь приучает к тому, что за результаты своей деятельности - будь то успех или неудача - необходимо нести личную

ответственность. Данная функция выходит за рамки собственно дидактики и смыкается с задачами нравственного становления личности.

Сложившийся в отечественной образовательной практике арсенал традиционных форм контроля включает четыре основных инструмента: устный опрос, письменную контрольную работу, практическое задание и тестирование. Каждый из них обладает собственной дидактической спецификой, и каждому присущи как выраженные достоинства, так и неустранимые в рамках данной формы ограничения.

Устный опрос даёт педагогу возможность оценить не поверхностное воспроизведение заученного, а глубину понимания материала. Именно в живом диалоге обнаруживается, способен ли обучающийся аргументированно отстаивать свою позицию, выстраивать логические цепочки рассуждений, реагировать на встречные вопросы. Речь идёт о диагностике мышления в процессе его развёртывания - качество, практически недоступное прочим формам контроля. Плата за эту глубину - существенные временные издержки: за один академический час через процедуру устного опроса удаётся охватить лишь малую часть класса. Остальные обучающиеся в этот момент остаются пассивными наблюдателями, что создаёт известную проблему рационального использования урочного времени.

Письменная контрольная работа решает ровно противоположную задачу: она позволяет единовременно проверить знания всего класса. Фронтальность здесь является главным организационным преимуществом. Однако оборотной стороной становится отсроченность обратной связи. Проверка письменных работ требует от учителя значительных временных затрат, и промежуток между выполнением задания и получением результата может растягиваться на несколько дней. За это время острота учебной ситуации притупляется: обучающийся успевает частично забыть и само задание, и ход своих рассуждений, что снижает коррекционный эффект от выставленной оценки.

Практические задания наиболее органично вписываются в логику предметов деятельностного цикла - таких, как информатика, где знание и умение

неразрывно слиты в акте выполнения конкретной операции. Достоинство этой формы в её приближенности к реальным условиям применения знаний. Слабое же звено кроется в процедуре оценивания: критерии зачастую размыты, и доля субъективности при вынесении суждения о качестве выполненной практической работы остаётся весьма высокой.

Тестирование, наконец, обеспечивает два трудносовместимых в иных формах свойства - объективность и оперативность. Стандартизированная процедура и автоматизированная обработка результатов сводят к минимуму влияние личности проверяющего. Проблема, однако, заключается в том, что традиционные тесты с закрытым перечнем вариантов ответа фиксируют главным образом результат, оставляя за скобками сам путь его получения. Глубина понимания материала, способность к нешаблонному рассуждению и уровень сформированности практических навыков при такой форме контроля ускользают от измерения.

При всех своих достоинствах классические методы контроля обнаруживают ряд существенных ограничений, и особенно рельефно эти ограничения проступают в свете современных требований к образовательному процессу. Прежде всего обращает на себя внимание эпизодический характер традиционных контрольных процедур. Проверка знаний осуществляется дискретно - от среза к срезу, тогда как сам процесс усвоения материала разворачивается непрерывно. Между контрольными точками педагог фактически лишён систематической информации о ходе познавательной деятельности обучающихся, что создаёт зоны информационной неопределённости.

Второе ограничение носит организационно-временной характер. Значительная часть усилий учителя поглощается рутинной проверкой выполненных работ, и эта нагрузка возрастает пропорционально числу обучающихся. Ресурс времени, который мог бы быть направлен на методическую подготовку или индивидуальную работу, расходуется на техническую операцию оценивания. Как следствие - возникает третья проблема: запаздывание обратной связи. Обучающийся получает информацию о допущенных ошибках с отсрочкой

в несколько дней, когда изучение соответствующей темы уже завершено и класс перешёл к новому материалу. Коррекционная работа в таких условиях утрачивает значительную долю своей действенности: возвращаться назад, не сбавляя общего темпа программы, крайне затруднительно.

Нельзя обойти вниманием и концептуальный сдвиг, произошедший в современной педагогике. Акцент принципиально сместился: если прежде контроль мыслился преимущественно как инструмент констатации - фиксации наличия или отсутствия знаний на определённый момент, то теперь во главу угла ставится процессуальная сторона. Оценивается не столько результат, сколько ход учебной деятельности, динамика познавательного продвижения. В этой парадигме контроль перестаёт быть механизмом селекции и аттестации и трансформируется в средство поддержки обучения. Функция его не в том, чтобы отсеять или рассортировать, а в том, чтобы вовремя указать направление коррекции и предоставить обучающемуся ресурсы для преодоления возникших трудностей.

Именно в обозначенном контексте при наложении выявленных ограничений традиционных методов на новые целевые установки педагогики цифровая образовательная среда начинает рассматриваться не как факультативное дополнение, а как пространство, открывающее качественно иные возможности для совершенствования оценочной деятельности. Речь идёт о потенциале автоматизации рутинных процедур, непрерывности сбора данных об учебном поведении и, как следствие, о перспективе перехода от эпизодического контроля к пролонгированному мониторингу, встроенному в саму ткань образовательного процесса.

Современный этап развития общества характеризуется глубокой цифровизацией всех сфер жизни, включая образование. Цифровая образовательная среда представляет собой совокупность информационных технологий, цифровых ресурсов, сетевой инфраструктуры и педагогических практик, обеспечивающих доступ к образованию и взаимодействие участников образовательного процесса [23]. Это не просто набор компьютеров и

программного обеспечения, а сложная социотехническая система, изменяющая способы взаимодействия учителя и обучающегося, способы представления знаний и способы контроля результатов обучения.

Внедрение цифровых технологий влечёт за собой кардинальную трансформацию всего облика оценочной деятельности. Меняются инструментарий, организационные схемы, скорость и плотность информационных потоков между субъектами образовательного процесса. Цифровая среда не просто добавляет новые технические возможности к существующей модели контроля она перестраивает саму архитектуру оценивания, порождая формы и процедуры, немыслимые в рамках доцифровой дидактики.

Вместе с тем было бы методологической ошибкой полагать, будто технологические новации отменяют фундаментальные основания педагогики. Базовые психолого-педагогические закономерности процесса обучения - механизмы усвоения, забывания, переноса, роль мотивации и рефлексии, сохраняют свою силу независимо от того, ведётся ли учёт результатов в бумажном журнале или в электронной системе. Технология не переписывает законы когнитивного развития; она создаёт новую среду, в которой эти законы проявляют себя иначе и, при грамотном подходе, могут быть использованы с большей эффективностью.

Сама же трансформация разворачивается по нескольким ключевым направлениям, каждое из которых заслуживает отдельного анализа. В их числе: переход от эпизодического контроля к непрерывному мониторингу, автоматизация проверки и мгновенная обратная связь, накопление цифрового следа обучающегося и появление принципиально новых форм заданий, недоступных вне электронной среды. Рассмотрение этих векторов по отдельности позволит составить более рельефное представление о масштабе происходящих изменений.

Первым направлением является автоматизация процессов проверки. Цифровые системы позволяют автоматизировать рутинные операции по проверке заданий, освобождая время учителя для индивидуальной работы с обучающимся.

В контексте обучения программированию это имеет критическое значение, так как проверка кода вручную является трудоемким процессом, требующим от учителя высокой квалификации и концентрации. Автоматизированные системы могут мгновенно запускать код обучающегося на тестовых данных и сравнивать результат с эталоном, что невозможно сделать вручную для каждого обучающегося в реальном времени [9].

Вторым направлением является оперативность обратной связи. В цифровой среде результат может быть получен мгновенно. Это позволяет обучающемуся сразу увидеть ошибку и исправить ее, не дожидаясь проверки учителем. Мгновенная обратная связь является мощным педагогическим инструментом, так как она связывает действие и результат в единый процесс, усиливая ассоциативные связи и способствуя быстрому усвоению правильных образцов поведения. Обучающийся не забывает контекст задачи, пока ждет оценку, и может сразу же применить полученную информацию для коррекции своих действий.

Третьим направлением выступает накопление данных и аналитика. Цифровые системы фиксируют историю действий обучающегося, время выполнения заданий, количество попыток, траекторию движения по учебному материалу. Это позволяет строить детальные профили учебных достижений и выявлять скрытые закономерности. Учитель получает доступ к аналитическим информационным панелям, где видит не только итоговую оценку, но и процесс ее получения. Это позволяет выявлять обучающихся, которые испытывают трудности, еще до того, как они получат неудовлетворительную оценку, и оказывать им своевременную поддержку.

Четвертым направлением является персонализация обучения. На основе данных контроля система может адаптировать сложность заданий под конкретного обучающегося, предлагая индивидуальные траектории развития. Если обучающийся легко справляется с базовыми задачами, система может предложить ему усложненные варианты. Если обучающийся испытывает трудности, система может предложить дополнительные разъясняющие материалы

или упрощенные задания для отработки навыков. Это реализует принцип дифференцированного подхода в массовом обучении, который трудно реализовать силами одного учителя в классе из 20-30 человек.

Пятым направлением является изменение роли учителя. Учитель превращается из единственного источника оценки в регулятора образовательного процесса, аналитика данных и наставника. Его функция смещается от трансляции знаний и контроля к сопровождению индивидуального развития обучающегося. Цифровые инструменты берут на себя рутинную часть работы, позволяя учителю сосредоточиться на творческих и воспитательных аспектах обучения.

Признавая преобразующий потенциал цифровых технологий, следует одновременно отдавать себе отчёт в сопутствующих им рисках. Наиболее серьёзный из них - опасность подмены педагогического содержания техническими возможностями. Ситуация, при которой инструментальная сторона заслоняет содержательную, в образовательной практике не нова, однако цифровая среда многократно усиливает этот соблазн. Обилие данных, автоматизированная аналитика, наглядные дашборды - всё это способно создавать иллюзию глубокой и объективной оценки, тогда как за эффектным интерфейсом может скрываться педагогически бессодержательная процедура.

Технология сама по себе качество обучения не гарантирует. Этот тезис, при всей своей кажущейся очевидности, нередко упускается из виду при форсированном внедрении цифровых решений. Эффективность автоматизированного контроля определяется не мощностью программного обеспечения и не изощрённостью алгоритмов, а тем, насколько органично данный инструмент вписан в ткань педагогического процесса. Решающее значение приобретает методическая обоснованность: каковы цели контроля, на каком этапе усвоения он применяется, как его результаты будут интерпретированы и использованы для коррекции обучения. Если ответы на эти вопросы отсутствуют или формальны, цифровой контроль рискует выродиться в ритуал - процедуру, соблюдаемую ради самой процедуры и не оказывающую сколько-нибудь заметного влияния на действительное качество знаний.

Отдельного упоминания заслуживают риски антропологического характера. Чрезмерное увлечение электронными формами взаимодействия способно провоцировать формирование цифровой зависимости и, что не менее тревожно, вести к постепенному угасанию коммуникативных навыков. Учебный диалог, непосредственное обсуждение, живая реакция педагога на затруднение - все эти элементы образовательной коммуникации не могут быть без потерь переведены в цифровой формат. При неконтролируемой экспансии электронных средств существует опасность, что обучающийся начнёт воспринимать учебный процесс как последовательность безличных транзакций с программной оболочкой, а не как взаимодействие с педагогом и учениками.

Сказанное подводит к выводу, что внедрение цифровых инструментов контроля не может осуществляться стихийно или по административному шаблону. Оно требует тщательного педагогического проектирования, в рамках которого технологическое решение оценивается не само по себе, а сквозь призму решаемых дидактических задач. Не менее императивным представляется сохранение баланса между инновационными и традиционными формами работы. Цифра не должна вытеснять живое педагогическое общение; её задача - дополнять и усиливать то, что составляет суть образовательного процесса, а не подменять эту суть техническим инструментарием.

При проектировании систем автоматизированного контроля важно учитывать, что оценка должна фиксировать не только актуальный уровень знаний, но и потенциал развития обучающегося. Здесь необходимо обратиться к фундаментальным теориям отечественной психологии и педагогики, которые сохраняют свою эвристическую ценность и в условиях цифровизации. Несмотря на то, что технологии меняются, законы человеческого обучения и развития остаются устойчивыми, и игнорирование их при создании цифровых образовательных продуктов ведет к снижению их эффективности [29].

Ключевым понятием для понимания роли контроля в развитии является концепция «Зоны ближайшего развития». Эта концепция, разработанная Львом Семеновичем Выготским, утверждает, что обучение должно опережать развитие,

направляя его. Как отмечал Л. С. Выготский в своем фундаментальном труде: «Обучение не есть просто развитие, но оно пробуждает к жизни те функции, которые находятся в стадии созревания, в зоне ближайшего развития» [24]. Это утверждение имеет принципиальное значение для организации контроля в цифровой среде.

В контексте автоматизированного контроля это означает, что система не должна ограничиваться фиксацией того, что обучающийся уже знает. Она обязана предоставлять инструменты для развития. Традиционный контроль часто работает как фильтр: он отсеивает тех, кто не знает материал, и пропускает тех, кто знает. Однако развивающий контроль должен работать как лифт, поднимая ученика на следующий уровень. Механизмы обратной связи, реализуемые в системах автоматизированного контроля, создают условия для многократного прохождения пути от «не могу» к «могу» в рамках решения одной задачи. Таким образом, автоматизированная проверка кода становится не техническим фильтром, а педагогическим средством поддержки обучающегося в зоне его ближайшего развития.

Если система просто пишет «Ошибка», она фиксирует уровень актуального развития. Если же система подсказывает, где именно ошибка, предлагает аналогичный пример или дает наводящий вопрос, она работает в зоне ближайшего развития, помогая обучающемуся подняться на следующий уровень компетенции. Именно поэтому качество обратной связи в системах типа CodeRunner является критическим фактором их педагогической эффективности [31]. Система должна быть способна не только сказать «неверно», но и объяснить «почему неверно» и «как исправить», выступая в роли интеллектуального наставника.

Реализация поддержки в цифровой среде напрямую зависит от качества проектирования учебных заданий. Система контроля должна не просто фиксировать ошибку, но и предоставлять обучающемуся четкое направление для её исправления. Важность этого аспекта подчеркивал Петр Яковлевич Гальперин, создатель теории поэтапного формирования умственных действий. Он писал:

«Ориентировочная основа действия (ООД) — это система условий, на которые субъект ориентируется при выполнении действия. Полнота и обобщённость ООД определяют успешность формирования умения» [4].

В контексте обучения программированию роль ООД выполняют формулировка задачи, набор тестовых данных, примеры ввода-вывода и тексты ошибок, выдаваемые системой. Если эти элементы составлены методически грамотно, обучающийся формирует устойчивое умение отладки кода и алгоритмического мышления. Если же ООД неполная, автоматизированная проверка становится для обучающегося препятствием, а не помощью, сводясь к механическому подбору ответов. Например, если задача сформулирована нечетко, обучающийся тратит время не на решение алгоритмической проблемы, а на догадки о том, чего хочет система. Это снижает учебную мотивацию и формирует неправильные навыки.

При этом важно учитывать, что формирование умения программировать не происходит моментально: оно требует последовательного прохождения этапов - от работы с готовым кодом и заполнения пропусков до самостоятельного проектирования алгоритмов. Именно такая организация учебного процесса позволяет сделать формирование умений управляемым и предсказуемым. Как отмечал П.Я. Гальперин: «Поэтапное формирование умственных действий позволяет управлять процессом усвоения, обеспечивая высокую эффективность обучения» [26].

Применительно к обучению программированию реализация поэтапного подхода требует выстраивания иерархически организованной системы заданий, каждое из которых соотносится с определённой стадией формирования умственного действия. Логика развёртывания этой иерархии повторяет траекторию интериоризации - постепенного перехода от внешних, материализованных опор к внутренним, свёрнутым умственным операциям.

На первой стадии - стадии материализованного действия - учебная задача решается с развёрнутой внешней поддержкой. Обучающийся опирается на визуальные схемы алгоритмов, готовые фрагменты программного кода либо

пошаговые инструкции, задающие ориентировочную основу действия. Внешние ориентиры здесь выполняют функцию scaffolding - временных опор, без которых самостоятельное выполнение задания пока невозможно. Автоматизированная система контроля на данном этапе решает относительно простую задачу: она верифицирует соответствие структурных элементов создаваемого кода предъявленному шаблону, фиксируя отклонения от заданного образца.

Вторая стадия - стадия внешне-речевого действия. Обучающийся переходит от манипуляции с графическими схемами к написанию программы по словесному описанию алгоритма. Существенным компонентом деятельности на этом этапе становится проговаривание - развёрнутое речевое сопровождение хода решения, позволяющее зафиксировать логику рассуждения. Цифровая среда, сообразуясь с психологической природой данной стадии, может предъявлять требование обязательного комментирования кода: комментарии выступают внешней проекцией речевого плана, и их содержательный анализ даёт педагогу материал для диагностики понимания.

Третья стадия - стадия внутренней речи - знаменует качественный скачок в формировании умения. Внешние опоры, как материальные, так и речевые, сворачиваются; проектирование алгоритма и написание кода осуществляются «в уме», и лишь готовый продукт этой внутренней работы предъявляется вовне. Содержание контроля на данном этапе закономерно смещается: проверке подлежат не промежуточные шаги и не наличие комментариев, а итоговая функциональность созданной программы и степень оптимальности предложенного решения. Акцент переносится с процесса на результат — но такой перенос оправдан именно тем, что процесс к этому моменту уже интериоризирован и недоступен прямому внешнему наблюдению.

Четвёртая стадия - стадия автоматизированного умения - представляет собой завершающий этап, на котором сформированное действие приобретает свойства навыка: оно выполняется быстро, безошибочно и без развёрнутого сознательного контроля. Диагностировать достижение данной стадии позволяют задания особого типа: оптимизация уже написанного кода, отладка программ,

содержащих скрытые ошибки, критический анализ чужого решения. Обучающийся здесь выступает не столько в роли автора, сколько в роли эксперта, способного оценить качество программного продукта и предложить обоснованные улучшения. Успешное выполнение заданий этого уровня свидетельствует о полной сформированности программистского умения как такового.

Автоматизированная система контроля предоставляет уникальную возможность сопровождать каждый из этих этапов адресной обратной связью. Например, на начальных этапах плагин CodeRunner может проверять не только итоговый результат, но и соответствие кода заданной структуре, выдавая подсказки при отклонении от ООД. На более высоких уровнях система оценивает эффективность алгоритма, корректность обработки граничных условий, что способствует переходу от репродуктивного выполнения к творческому применению знаний.

Таким образом, теория поэтапного формирования умственных действий становится методологическим основанием для проектирования дифференцированной системы заданий, где каждое последующее задание не просто сложнее предыдущего, но и требует от обучающегося большей степени самостоятельности в ориентировке и исполнении. Это позволяет реализовать принцип «управляемого усвоения», о котором писал П.Я. Гальперин, в условиях массового школьного обучения. Без учета этой теории автоматизированный контроль рискует стать хаотичным набором задач, не обеспечивающим системного развития навыков.

При этом важно учитывать, что формирование умения программировать - это не механическое накопление отдельных навыков, а становление целостной учебной деятельности, которая имеет свою внутреннюю логику, мотивацию и динамику развития. Именно такой системный взгляд на процесс обучения позволяет проектировать задания, которые не просто тренируют, но и развивают. Как подчёркивал А.Н. Леонтьев: «Деятельность - это не реакция и не

совокупность реакций, а система, имеющая строение, свои внутренние переходы и превращения, своё развитие» [27].

Применительно к автоматизированному контролю по программированию сказанное выше означает необходимость принципиального пересмотра структуры контрольно-измерительных материалов. Система заданий не может строиться как механический набор изолированных тестов или разрозненных задач, никак не связанных между собой содержательно и процессуально. Она должна быть спроектирована в логике целостной учебной деятельности, обладающей внутренней психологической структурой. В этой структуре отчётливо выделяются четыре взаимосвязанных компонента.

Первый - мотивационный компонент. Он реализуется через постановку практической задачи, смысл которой прозрачен для обучающегося и выходит за рамки сугубо учебной ситуации. Второй - ориентировочный компонент, представленный инструкциями, примерами выполнения и, что особенно значимо, ориентировочной основой действия, задающей обобщённый способ решения задач данного класса. Третий - исполнительский компонент, в границах которого разворачивается непосредственное написание и отладка программного кода. Четвёртый - контрольно-оценочный компонент, включающий как автоматизированную обратную связь от системы, так и рефлексивную самооценку обучающимся полученного результата и способа его достижения.

Только при соблюдении полноты этой четырёхкомпонентной структуры автоматизированный контроль перестаёт восприниматься как узкотехнический инструмент проверки и трансформируется в полноценное средство развития учебных достижений. В противном случае он рискует остаться на периферии реального образовательного процесса - процедурой, которую обучающиеся проходят, но которая не оказывает заметного формирующего воздействия.

Особого внимания заслуживает мотивационный компонент как наиболее уязвимый для формализации. Задание, сформулированное вне какого-либо жизненного или практического контекста, например, абстрактное требование «написать программу, выводящую сумму чисел», закономерно воспринимается

обучающимся как искусственная учебная формальность. Мотивационный потенциал такого задания близок к нулю: оно не отвечает на вопрос «зачем?» и не включает механизмы смыслообразования. Совершенно иная картина наблюдается, когда та же самая вычислительная задача помещается в осмысленный контекст. Формулировка «разработать программу для расчёта бюджета школьного мероприятия» задаёт практическую рамку, в которой написание кода обретает инструментальную ценность. Задача встраивается в структуру деятельности обучающегося не как навязанное извне требование, а как звено, связывающее учебное действие с реальной жизненной ситуацией. Именно такое включение практического смысла превращает автоматизированное задание из средства контроля в средство развития.

Проектируя систему автоматизированного контроля, нельзя упускать из виду более широкую – надпредметную - задачу школьного образования. Учебная деятельность не исчерпывается формированием предметных знаний и умений; её фундаментальная миссия состоит в развитии личности обучающегося. Применительно к старшей школе это положение приобретает особую весомость: обучение программированию хронологически совпадает с сенситивным периодом активного личностного становления. Именно в эти годы интенсивно вызревают такие качества, как самостоятельность в принятии решений, ответственность за результат собственных действий, способность к осознанному целеполаганию и развитая рефлексия - умение оценивать не только продукт, но и сам способ своей деятельности.

Систематическое включение старшеклассника в полноценную учебную деятельность с её развёрнутой структурой, ясными ориентирами и осмысленной обратной связью, создаёт питательную среду для качественных личностных сдвигов. Механизм этих сдвигов раскрывается через категорию присвоения: обучающийся, действуя в организованном образовательном пространстве, постепенно интериоризирует социальные нормы и обобщённые способы действия, выработанные культурой. То, что первоначально существовало как

внешнее требование или образец, становится внутренним достоянием личности, её собственным инструментом ориентации в мире.

Данный тезис имеет глубокие корни в отечественной психологической традиции. В работах А.Н. Леонтьева он выражен с предельной отчётливостью: «Личность есть особое, сверхприродное качество, которое приобретается индивидом в обществе, в целокупности отношений, общественных по своей природе» [27]. Формула эта задаёт методологический вектор для педагогического проектирования. Если личность действительно формируется не изнутри, по биологической программе, а в пространстве общественных отношений и деятельностей, то образовательная среда, включая её цифровой сегмент, должна целенаправленно выстраиваться как система таких отношений. Автоматизированный контроль, будучи вписанным в эту систему, перестаёт быть лишь инструментом проверки кода. Он становится одним из звеньев, опосредствующих личностное развитие: через него обучающийся осваивает нормы объективной оценки, опыт ответственного отношения к результату своего труда, практику рефлексивного анализа собственных достижений и ошибок.

Применительно к реалиям современной школы цифровая образовательная среда перестаёт быть лишь вспомогательной инфраструктурой и приобретает статус одного из центральных пространств, в которых разворачиваются те самые «общественные по своей природе» отношения, о которых писал А.Н. Леонтьев. Это уже не просто канал доставки учебного контента и не техническая оболочка для автоматизированной проверки. Речь идёт о формировании полноценной социальной плоскости, где протекают ключевые процессы учебного взаимодействия.

Учитель и обучающийся, оставаясь субъектами образовательного процесса, всё в большей степени взаимодействуют через посредство цифровых инструментов. Обратная связь, некогда существовавшая исключительно в форме устного комментария или пометки в тетради, теперь обретает вид данных в электронном журнале, автоматически сгенерированных рекомендаций, визуализированных траекторий продвижения. Оценка достижений, в свою

очередь, перестаёт быть единичным актом и трансформируется в непрерывно пополняемый массив информации, доступный для анализа как педагогу, так и самому обучающемуся. Все эти сдвиги не являются чисто техническими, они затрагивают саму ткань педагогических отношений, а потому требуют специального, методологически выверенного осмысления.

Более того, цифровая среда начинает функционировать как своеобразный микросоциум со своими, пока ещё только формирующимися нормами. В этом пространстве у обучающегося складываются представления о допустимом и недопустимом в цифровом поведении, вырабатывается осознание ответственности за оставляемый им цифровой след, кристаллизуются зачатки этики взаимодействия с интеллектуальными системами. Всё это - элементы того самого личностного новообразования, которое не может быть передано прямой трансляцией, а выращивается только через опыт реального, пусть и опосредованного технологией социального взаимодействия. Педагогическое проектирование цифровой среды, таким образом, обязано учитывать не только дидактическую результативность инструментов контроля, но и их латентное влияние на становление личности, формирующейся в пространстве, где технология всё плотнее срастается с социальной тканью образовательного процесса.

Именно этот запрос - потребность в систематическом осмыслении происходящих трансформаций, вызвал к жизни формирование новой отрасли педагогического знания. А.Л. Семёнов и А.Ю. Уваров фиксируют её появление со всей определённостью: «Цифровая дидактика - это отрасль педагогики, изучающая закономерности и принципы организации обучения в условиях цифровой образовательной среды» [28]. Сама формулировка показательна: речь идёт не о частной методике и не о наборе технологических рекомендаций, а о полноценной научной дисциплине со своим предметом, понятийным аппаратом и исследовательской программой.

Возникновение цифровой дидактики не было произвольным. Оно продиктовано объективной необходимостью: традиционные дидактические

принципы, сложившиеся в иной технологической реальности, нуждаются в переосмыслении применительно к условиям электронной образовательной среды. Механический перенос классических положений на новую почву без учёта специфики цифрового инструментария оказался бы непродуктивным. Требуется именно адаптация - сохранение сущностного ядра принципа при одновременной модификации способов его практического воплощения.

Наглядность может служить здесь выразительной иллюстрацией. В докомпьютерной дидактике она реализовывалась преимущественно через статичные изображения, таблицы и натуральные объекты. Цифровая среда радикально раздвигает эти границы: принцип наглядности обретает форму интерактивных моделей, динамических симуляций, трёхмерных визуализаций, с которыми обучающийся может манипулировать, извлекая из объекта не только его внешний облик, но и скрытые функциональные связи. Аналогичным образом трансформируется принцип доступности. В традиционной трактовке он предполагал адаптацию учебного материала к возрастным и индивидуальным особенностям на этапе его создания. Цифровая же среда делает доступность динамической характеристикой: адаптивные интерфейсы подстраивают уровень сложности и формат подачи под конкретного обучающегося в режиме реального времени, а персонализированные образовательные траектории позволяют варьировать темп, последовательность и глубину освоения содержания. Принцип не отменяется, он обретает новое инструментальное воплощение, и именно такого рода метаморфозы составляют предметное поле цифровой дидактики как складывающейся научной отрасли.

Цифровизация, однако, не сводится к простому переносу традиционных методов в новую технологическую оболочку. Она ставит перед теорией и практикой педагогического измерения качественно иные задачи, требующие не косметической адаптации, а содержательного пересмотра устоявшихся подходов. Автоматизированный контроль, адаптивные образовательные траектории, мгновенная обратная связь - все эти возможности, появившиеся благодаря цифровой среде, меняют саму логику оценивания. Измерение перестаёт быть

дискретной процедурой, вынесенной за скобки учебного процесса, и превращается в его имманентную составляющую, непрерывно сопровождающую продвижение обучающегося.

А.Л. Семёнов формулирует эту мысль с необходимой прямотой: «Цифровая трансформация образования требует пересмотра традиционных подходов к контролю и оценке учебных достижений» [29]. Существенно, что пересмотру подлежат не одни лишь инструменты - тесты вместо устных опросов, автоматическая проверка вместо ручной, но и сами критерии, по которым выносится оценочное суждение. В доцифровой парадигме ценность представлял преимущественно конечный результат: правильный ответ, работающая программа, верно решённая задача. Цифровая среда, фиксирующая процессуальные характеристики деятельности, позволяет разглядеть ценность в том, что прежде оставалось невидимым. Траектория движения к ответу, количество предпринятых попыток, частота и характер использования подсказок, время реакции на предъявляемое задание - все эти параметры, аккумулируемые системой, складываются в объёмную картину, несводимую к бинарной оппозиции «верно-неверно».

Методологическая роль цифровой дидактики в этом контексте раскрывается как роль связующего звена - моста между классической педагогической теорией и современными технологическими возможностями. Она не отбрасывает фундаментальные положения предшественников, а интегрирует их в новую реальность, задавая принципы проектирования такого контроля, который ориентирован не на селекцию и фиксацию, а на развитие.

Так, положение Л.С. Выготского о зоне ближайшего развития о том, что обучение должно опережать актуальный уровень и опираться на формирующиеся, ещё не созревшие функции, получает инструментальное подкрепление в виде обширного анализа цифрового следа. Система, отслеживающая поведенческие паттерны обучающегося, способна с известной точностью локализовать границу между тем, что он уже умеет делать самостоятельно, и тем, что пока доступно лишь при дозированной помощи. Теория П.Я. Гальперина об ориентировочной

основе действия, в свою очередь, находит практическое воплощение в интерактивных подсказках, встроенных в интерфейс: ориентиры не просто предъявляются однократно в виде инструкции, а могут актуализироваться системой именно в тот момент, когда обучающийся сталкивается с затруднением. Деятельностный подход А.Н. Леонтьева, наконец, смыкается с проектными методами, реализуемыми в цифровой среде: учебная деятельность приобретает характер полноценной - мотивированной, целенаправленной, рефлексивно завершаемой, а технология обеспечивает для неё адекватное инструментальное оснащение.

Таким образом, цифровая дидактика не замещает классические теории, а предоставляет им новую площадку для развёртывания. Её назначение - перевести концептуальные построения в плоскость конкретных проектных решений, сохраняя при этом педагогическую сущность контроля, не позволяя технологическому инструментарию вытеснить развивающую направленность образовательного процесса.

Проведённый теоретический анализ даёт основания для формулирования ряда обобщающих выводов, фиксирующих методологические позиции, с которых следует подходить к проектированию контроля учебных достижений в цифровой среде.

Первое. Педагогические подходы к оцениванию в условиях цифровизации не могут строиться на голом эмпиризме или на заимствовании технологических решений без их теоретического осмысления. Методологическим фундаментом здесь призваны служить классические отечественные концепции культурно-историческая теория Л.С. Выготского, теория поэтапного формирования умственных действий П.Я. Гальперина, деятельностный подход А.Н. Леонтьева. Их эвристическая ценность не только не утрачена с приходом цифровых инструментов, но, напротив, раскрывается с новой отчётливостью: каждая из этих теорий предлагает объяснительные схемы и проектные ориентиры, позволяющие осмысленно встраивать технологию в педагогический процесс, а не наоборот.

Второе. Цифровая дидактика, формирующаяся как самостоятельная отрасль педагогического знания, выполняет роль методологического моста. Она связывает корпус классической теории с современными технологическими возможностями, не допуская ни высокомерного игнорирования нового инструментария, ни растворения педагогической сущности в техническом функционале. Применительно к контролю это означает смену вектора: от фиксации результата к сопровождению развития. Именно цифровая дидактика задаёт принципы, на которых такой контроль может быть спроектирован, не утратив своей педагогической природы.

Третье. Система заданий для автоматизированного контроля по программированию не должна представлять собой механический набор проверочных единиц. Она обязана выстраиваться как целостная учебная деятельность, в структуре которой отчётливо выделены мотивационный, ориентировочный, исполнительский и контрольно-оценочный компоненты. Необходима поэтапная градация сложности, воспроизводящая логику интериоризации - от действий с внешними опорами к полностью самостоятельному программированию. Обратная связь, генерируемая системой, должна быть выверена так, чтобы удерживать обучающегося в зоне его ближайшего развития: не подменять собой самостоятельное усилие, но и не оставлять наедине с непосильной трудностью.

Сформулированные выше положения в совокупности образуют методологический фундамент, позволяющий подойти к организации контроля в цифровой среде не как к технической, а как к собственно педагогической задаче. Вместе с тем программирование как учебный предмет обладает выраженной спецификой, отличающей его от многих других дисциплин школьного цикла. Характер формируемых умений, типология типичных ошибок, сама природа программистской деятельности требуют отдельного аналитического рассмотрения уже не столько в общедидактическом, сколько в частнометодическом ключе. Этому и посвящён следующий этап исследования.

1.2. Специфика формирования и диагностики учебных достижений в области программирования

Обучение программированию в старшей школе представляет собой сложный процесс, выходящий за рамки простого изучения синтаксиса языка. Программирование становится ключевым элементом функциональной грамотности, формирующим алгоритмическое мышление, креативность и навыки проектной деятельности [13]. Это утверждение подчеркивает, что цель обучения не в том, чтобы подготовить профессиональных разработчиков ПО, а в том, чтобы сформировать у обучающихся универсальные навыки решения задач с помощью вычислительных средств.

Структура компетенций в области программирования может быть представлена несколькими уровнями, которые необходимо учитывать при проектировании системы контроля.

Первый уровень - концептуальный. Он включает понимание базовых концепций информатики: алгоритм, структура данных, переменная, цикл, условие, функция. Контроль на этом уровне может осуществляться через тестовые задания, вопросы на соответствие и терминологические диктанты. Однако ограничиваться этим уровнем нельзя, так как знание терминов не гарантирует умения программировать.

Второй уровень - технологический. Он предполагает владение синтаксисом конкретного языка программирования, умение использовать среду разработки, отладчик. Контроль здесь требует проверки работоспособности кода.

Третий уровень - алгоритмический. Это способность разработать алгоритм решения задачи, выбрать оптимальную структуру данных, оценить сложность алгоритма. Диагностика этого уровня наиболее сложна, так как одну и ту же задачу можно решить множеством способов, и система контроля должна уметь распознавать корректность различных подходов.

Четвертый уровень - практический. Это умение написать работающий код, протестировать его, найти и исправить ошибки. Именно этот уровень является

приоритетным для автоматизированного контроля, так как он наиболее формализуем.

Пятый уровень - рефлексивный. Это способность анализировать чужой код, оптимизировать собственное решение, документировать работу. Контроль этого уровня часто требует экспертной оценки учителя или взаимной оценки обучающихся, так как автоматике сложно оценить качество комментариев или архитектурную красоту решения.

Специфика формирования этих компетенций заключается в их взаимосвязи. Невозможно сформировать алгоритмическое мышление без практической реализации алгоритмов в коде. В то же время, механическое заучивание синтаксиса без понимания алгоритмической сути не приводит к формированию полноценной компетенции. А.Г. Гейн в своей методике подчеркивает, что «Обучение программированию должно быть построено на основе деятельности по исследованию окружающей действительности, которая выполняется в процессе решения задач» [5]. Это означает, что задания по программированию не должны быть абстрактными математическими упражнениями. Они должны иметь контекст, быть приближенными к реальным ситуациям, что повышает мотивацию обучающихся и способствует переносу знаний в новые условия. Например, вместо задачи «найдите сумму элементов массива», более целесообразно поставить задачу «рассчитайте средний балл класса по журналу оценок». Такая формулировка включает обучающегося в деятельность по исследованию действительности, о которой писал Гейн. Система автоматизированного контроля должна поддерживать эту специфику, позволяя формулировать задачи с контекстом и проверять их решение не только по числовому ответу, но и по логике обработки данных.

Ключевым системообразующим фактором, определяющим структуру автоматизированного контроля по программированию, является необходимость охвата полного набора тем школьного курса. Согласно требованиям ФГОС среднего общего образования [30] и примерной основной образовательной программы [29], раздел «Алгоритмы и программирование» включает строго

определенный круг содержательных линий, каждая из которых должна быть представлена в системе заданий. В базовый набор тем, подлежащих контролю, входят: базовый ввод-вывод и арифметические операции, условный оператор и логическое ветвление, циклы с предусловием и постусловием (while), циклы с известным числом повторений (for), строки и посимвольная обработка, списки и операции над ними, множества и словари как специализированные структуры данных, функции, модульность и рекурсия. Каждая из этих тем представляет собой самостоятельную дидактическую единицу со своей спецификой диагностики, однако все они объединены единой логикой формирования алгоритмического мышления.

Вторым системообразующим фактором выступает последовательное использование тем в системе контроля. Программирование - предмет с жесткой внутренней иерархией: невозможно полноценно диагностировать умение работать со списками, если обучающийся не освоил базовые циклы, а рекурсия не может быть оценена без предварительного формирования понятия функции. Последовательность в системе заданий должна отражать не только логическую структуру языка программирования, но и психологическую логику формирования умственных действий по П.Я. Гальперину [26]. Это означает, что каждая новая тема вводится в контроль только после того, как предыдущая освоена на уровне автоматизированного навыка, а усложнение происходит не за счет введения нового синтаксиса, а за счет снижения объема ориентировочной основы действия.

Третьим системообразующим фактором является разнообразие типов заданий, используемых в системе автоматизированного контроля. Эффективная система должна включать задания на заполнение пропусков в готовом коде (материализованная форма ООД), задания на исправление ошибок в программе (диагностика навыков отладки), задания на написание кода по готовому алгоритмическому описанию (внешне-речевая форма), задания на самостоятельное проектирование алгоритма и его реализацию (внутренняя речь) и, наконец, задания на оптимизацию и анализ уже написанного кода (автоматизированный уровень). Такое разнообразие типов заданий обеспечивает

диагностику компетенций на всех уровнях - от технологического до рефлексивного - и позволяет избежать ситуации, когда система оценивает лишь один аспект умения программировать.

Особенность контроля знаний по программированию заключается в необходимости проверки не только теоретических знаний, но и практических навыков написания, отладки и анализа кода [22]. Традиционные тестовые задания с выбором ответа часто не позволяют оценить глубину понимания алгоритмических конструкций. Обучающийся может угадать правильный ответ или запомнить шаблон, не понимая логики работы программы. Поэтому наиболее эффективным инструментом диагностики является выполнение практических заданий на написание кода.

Однако проверка такого рода заданий вручную требует от учителя высокой квалификации и значительных временных затрат. Учителю необходимо не только запустить код и проверить результат, но и проанализировать стиль программирования, эффективность алгоритма, наличие лишних операций, читаемость кода. В условиях наполняемости классов в 30 и более человек качественная ручная проверка становится практически невозможной без ущерба для других видов учебной деятельности. В этом контексте автоматизированные инструменты оценки могут помочь преподавателям в проверке, маркировке и оценке упражнений по программированию, а также в предоставлении обучающимся оперативной обратной связи.

Системы автоматической проверки (например, на основе тестовых наборов данных) позволяют мгновенно подтвердить корректность работы программы на множестве входных данных, включая граничные случаи, которые обучающийся мог не предусмотреть. Это повышает объективность оценки и снижает нагрузку на учителя. Однако автоматизированная диагностика имеет свои ограничения. Она хорошо проверяет функциональность (что делает программа), но хуже оценивает структурность и оптимальность (как делает программа). Поэтому эффективная система контроля должна комбинировать автоматическую проверку результатов с экспертной оценкой учителя или взаимной оценкой обучающихся

для аспектов, трудно поддающихся автоматизации (например, качество именования переменных, комментарии, архитектурные решения).

Диагностика навыков отладки представляет собой задачу ещё более высокого порядка сложности. Специфика отладки как деятельности заключается в том, что она плохо поддаётся прямой объективации: в отличие от итогового программного продукта, сам процесс поиска и устранения ошибок не оставляет видимого следа в финальном коде. Между тем в реальной профессиональной практике умение находить и квалифицированно исправлять ошибки ценится зачастую выше, чем способность писать безошибочный код с первой попытки. Программа редко рождается чистой; умение же работать с её несовершенством - маркер зрелого специалиста.

Для диагностики данного навыка методически оправдано использование заданий особого типа. Речь идёт о предъявлении обучающемуся фрагмента кода, содержащего скрытые дефекты, с инструкцией «Найти ошибку» либо о более развёрнутой постановке: «Исправить программу таким образом, чтобы она проходила весь набор предъявленных тестов». В отличие от задач на написание кода с нуля, здесь акцент сознательно смещён с продуктивной деятельности на аналитическую. От обучающегося требуется не столько породить собственное решение, сколько реконструировать чужое, выявив в нём точки отказа.

Автоматизированная среда открывает в этом отношении возможности, недоступные при традиционной организации контроля. Система способна фиксировать параметры, выступающие косвенными индикаторами уровня сформированности отладочного навыка: количество предпринятых попыток компиляции, характер и величина вносимых изменений между последовательными версиями кода, общее время, затраченное на достижение корректного решения. Каждый из этих показателей, взятый по отдельности, не обладает диагностической достаточностью, однако их совокупность позволяет строить обоснованные предположения о стратегии, которой руководствуется обучающийся.

Особую настороженность должны вызывать поведенческие паттерны, указывающие на отсутствие системного подхода к отладке. Если система регистрирует двадцать и более попыток компиляции с минимальными, хаотическими правками, это с высокой вероятностью свидетельствует о применении тактики «случайного тыка» - бессистемного перебора вариантов без предварительного анализа причин неработоспособности программы. Подобная стратегия не просто неэффективна в конкретной учебной ситуации, но и сигнализирует о несформированности самой ориентировочной основы отладочной деятельности. В таких случаях автоматизированная система может выполнять не только контролирующую, но и сигнальную функцию: фиксируя тревожный паттерн, она оповещает учителя, создавая фактическую основу для своевременного педагогического вмешательства - индивидуальной беседы, направленной на коррекцию самого способа действия, а не только его результата.

Важно также учитывать проблему плагиата и использования генеративных моделей ИИ. Современные системы контроля должны включать механизмы анализа схожести кода и выявления признаков использования сторонних решений [8]. Это требует постоянного обновления методов диагностики и включения в систему контроля заданий, которые сложно решить копированием готового ответа (например, задачи с уникальными параметрами, генерируемыми для каждого обучающегося).

Становление личности в процессе обучения программированию не может быть понято как некий побочный эффект, сопутствующий освоению предметного содержания. Напротив, сама природа программистской деятельности такова, что она с неизбежностью задействует, а значит, и развивает целый комплекс когнитивных и личностных структур. В первую очередь речь идёт о логическом мышлении, способности к планированию собственных действий и рефлексии получаемых результатов. Написание программы требует от обучающегося последовательного развёртывания замысла от общей идеи к конкретной реализации, предвосхищения возможных состояний системы, анализа причин расхождения между ожидаемым и действительным поведением кода. Всё это

формирует привычку к упорядоченному, дисциплинированному мышлению, не терпящему приблизительности.

Не менее значимым оказывается и требование к концентрации внимания, к способности удерживать в уме одновременно несколько уровней абстракции от синтаксиса конкретного оператора до общей архитектуры решаемой задачи. Абстрагирование вообще составляет сердцевину программистского мышления: работа с переменными, типами данных, функциями и классами по самой своей сути есть оперирование формальными, отвлечёнными от конкретики сущностями. В этом смысле программирование выступает не просто как ещё одна учебная дисциплина, но как своего рода тренажёр абстрактного мышления.

Период старшего школьного возраста 15–17 лет характеризуется качественными сдвигами в когнитивной сфере, и сдвиги эти входят в резонанс с теми требованиями, которые предъявляет обучение программированию. Именно в данном возрастном интервале завершается формирование формально-логического мышления: подросток обретает способность к гипотетико-дедуктивным рассуждениям, к оперированию высказываниями, истинность или ложность которых не задана непосредственным опытом, а устанавливается через логический вывод. Эта способность составляет когнитивную предпосылку для полноценного освоения программирования, но что важнее в контексте обсуждаемой темы само программирование создаёт естественную, предметно-насыщенную среду, в которой данная способность может упражняться и оттачиваться. Возникает своего рода синергия: возрастная готовность к формальным операциям встречается с учебным содержанием, требующим именно таких операций, и в этой встрече когнитивный потенциал подростка получает возможность реализоваться с наибольшей полнотой. Отсюда следует, что контроль учебных достижений по программированию не вправе ограничиваться проверкой предметных результатов: он должен быть чувствителен и к тем личностно-когнитивным новообразованиям, которые вызревают в процессе обучения.

Написание программы - это по сути построение логической модели решения задачи. Любая ошибка в логике приводит к неработоспособности программы, что дает объективную и немедленную обратную связь о качестве мышления обучающегося. Компьютер не жалеет и не ошибается в оценке синтаксиса, что воспитывает точность и аккуратность. Кроме того, программирование воспитывает настойчивость и трудолюбие. Процесс отладки кода часто сопряжен с неудачами. Обучающийся учится воспринимать ошибку не как поражение, а как информацию для улучшения решения. Это формирует «мышление роста», важное для современной личности.

Также стоит отметить развитие навыков проектной деятельности. Современные курсы программирования часто включают создание полноценных проектов (игры, сайты, анализ данных). Это требует умения ставить цели, планировать этапы работы, распределять ресурсы и презентовать результат. Эти навыки являются универсальными и востребованы в любой профессиональной деятельности. Таким образом, диагностика учебных достижений по программированию не должна ограничиваться узкопредметными метриками. Она должна учитывать и метапредметные результаты: развитие алгоритмического мышления, навыков самостоятельной работы, способности к самообучению.

Автоматизированные системы могут собирать статистику, позволяющую судить о динамике развития этих качеств (например, снижение количества ошибок со временем, увеличение сложности решаемых задач). Однако важно, чтобы эта статистика использовалась не для наказания, а для поддержки. Учитель должен видеть не только кто «отстал», но и кто «вырос», и поощрять прогресс. В этом смысле автоматизированный контроль становится инструментом педагогической поддержки личностного развития, а не просто механизмом отбора.

1.3. Требования к системам автоматизированного контроля в школьной практике

Использование любых инструментов контроля в школе должно соответствовать нормативно-правовой базе Российской Федерации. Требования к

предметным результатам освоения программ среднего общего образования по информатике включают умение создавать и отлаживать программы, использовать алгоритмические конструкции, работать с данными [5]. Это означает, что система контроля должна обеспечивать возможность проверки именно этих умений. Нельзя сводить контроль к тестам на знание терминов. Обязательным элементом является практическое программирование.

Нормативную опору внедрение автоматизированных систем контроля обретает в положениях Федерального государственного образовательного стандарта. Пункт 19.3 ФГОС фиксирует требования к структуре образовательной программы, прямо предусматривая использование цифровых инструментов для оценивания результатов обучения [30]. Данное положение не носит рекомендательного характера оно закрепляет правовую основу, легитимирующую применение в учебном процессе таких программных комплексов, как, например, система Moodle с интегрированным плагином CodeRunner. Школа, таким образом, не просто вправе обращаться к цифровому инструментарию оценивания она объективно поставлена перед необходимостью его освоения. Цифровая трансформация образования переводит вопрос из плоскости допустимости в плоскость императива: способность образовательной организации обеспечить выпускнику конкурентоспособный уровень подготовки всё теснее увязывается с масштабом и качеством используемых ею цифровых средств, включая средства контроля.

Вместе с тем нормативная база не исчерпывается положениями, открывающими технологические возможности. Она одновременно задаёт и систему ограничений, пренебрежение которыми способно свести на нет позитивный эффект цифровизации. Три группы требований заслуживают в этом контексте особого внимания.

Первая группа связана с защитой персональных данных обучающихся. Информация об учебных достижениях, будучи аккумулированной в цифровой системе, приобретает свойство лёгкой распространяемости и ровно в той же мере становится уязвимой для несанкционированного доступа. Законодательство

недвусмысленно требует, чтобы подобные данные не оказывались в общем доступе без документально оформленного согласия родителей или законных представителей. Для проектировщиков автоматизированной системы это означает необходимость встраивания механизмов разграничения прав доступа и обеспечения конфиденциальности на уровне архитектуры платформы.

Вторая группа требований касается доступности цифровой среды для лиц с ограниченными возможностями здоровья. Интерфейс системы контроля не может проектироваться в расчёте на некоего абстрактного, лишённого индивидуальных особенностей пользователя. Он должен быть адаптирован к различным типам восприятия информации визуальному, аудиальному, предусматривать возможность масштабирования текста, совместимость с программами экранного доступа, альтернативные способы ввода. Инклюзивность не является факультативной опцией; это прямое нормативное требование, несоблюдение которого закрывает части обучающихся доступ к процедуре оценивания на равных условиях.

Третья группа - санитарно-гигиенические нормы, регламентирующие продолжительность непрерывной работы с компьютерной техникой. СанПиН устанавливает жёсткие временные рамки, дифференцированные по возрастным группам. Данное ограничение оказывает непосредственное влияние на проектирование контрольных процедур: длительность автоматизированной контрольной работы не может превышать установленных лимитов, а структура заданий должна учитывать необходимость переключения активности, снижающего зрительное и статическое напряжение. Игнорирование этих норм при внедрении системы контроля чревато не только административными последствиями, но и прямым ущербом для здоровья обучающихся.

Таким образом, нормативное поле, в которое погружена разработка автоматизированной системы контроля, имеет двойственный характер. С одной стороны, ФГОС создаёт легитимное пространство для технологической модернизации оценочных процедур. С другой, требования к защите данных, доступности и санитарной безопасности очерчивают границы, в пределах

которых эта модернизация только и может считаться допустимой. Учёт обеих сторон в их взаимосвязи составляет обязательное условие педагогически грамотного проектирования.

На современном рынке образовательных технологий существует множество платформ для обучения программированию. Среди них можно выделить специализированные системы (Stepik) и системы управления обучением (LMS) с функционалом проверки кода (Moodle). Для школьной практики наиболее целесообразным является использование LMS Moodle с плагином CodeRunner. CodeRunner - это бесплатный подключаемый модуль с открытым исходным кодом для Moodle, который может запускать программный код, представленный обучающимся, и оценивать его по серии тестов [31]. Выбор именно этой связки обусловлен рядом преимуществ.

Во-первых, интеграция. Плагин работает внутри привычной для многих школ среды Moodle, не требует отдельной регистрации на внешних сервисах. Это упрощает администрирование и обеспечивает единое пространство для всех учебных предметов. Во-вторых, гибкость. Поддерживает множество языков программирования (Python, Java, PHP и др.), что позволяет использовать его на разных этапах обучения и в разных классах. В-третьих, безопасность. Код выполняется в изолированной среде, что предотвращает вредоносное воздействие на сервер. Это критически важно для школьной сети, где безопасность данных является приоритетом. В-четвертых, адаптивность. Плагин позволяет использовать адаптивный режим: обучающиеся могут исправлять код и повторно отправлять его, получая немедленную обратную связь [32].

Данный подход согласуется с принципами широкоформатного оценивания: первая ошибка не влечёт за собой штрафных санкций, а становится точкой роста, обучающийся получает возможность извлечь из неё учебный эффект. Существуют и альтернативные инструменты, в частности встроенные средства платформы Stepik. Однако подобные решения, как правило, проектировались под задачи массовых открытых онлайн-курсов, и их гибкость применительно к внутренней школьной оценке оказывается ограниченной. Учителю же в классе

требуется иное - детальная аналитика в разрезе конкретного учебного коллектива. Moodle в этом отношении выигрывает: педагог сохраняет полный контроль над журналом оценок и механизмами доступа к учебным материалам.

Несмотря на очевидные преимущества, внедрение автоматизированных систем в общеобразовательной школе сопряжено с рядом трудностей. Использование платформ в общеобразовательной школе часто затруднено из-за избыточной сложности, ориентации на вузовскую аудиторию или профильных олимпиад [32]. Многие системы предполагают высокий уровень самостоятельности обучающихся, который не всегда сформирован у старшеклассников массовой школы. Интерфейсы могут быть перегружены техническими деталями, отвлекающими от сути задачи. Поэтому при внедрении необходимо проводить адаптацию интерфейса и упрощение инструкций.

Технологическая зависимость является еще одним существенным ограничением. Образовательные платформы требуют стабильного интернет-соединения и современного оборудования, что может ограничивать их доступность в некоторых регионах [17]. В случае сбоя сети или оборудования учебный процесс может быть парализован. Поэтому важно иметь резервные варианты проведения контроля. Школа должна обладать необходимой инфраструктурой, включая локальные серверы или кэширование материалов, чтобы минимизировать риски зависимости от внешнего канала.

Нельзя сбрасывать со счетов и риск «натаскивания» на тесты. Если автоматизированная проверка сводится исключительно к сравнению вывода программы с эталонным образцом, у обучающихся возникает соблазн подбирать решение под ожидаемый результат, минуя понимание существа задачи. Противодействие этой тенденции требует тщательного проектирования тестовых наборов: использование случайно генерируемых входных данных, проверка не только результата, но и структуры кода. Уникальные тестовые сценарии, формируемые для каждого запуска или индивидуально для каждого обучающегося, способны перекрыть обходные пути. Наряду с этим в систему

контроля целесообразно включать задания, требующие не только предъявления кода, но и развёрнутого объяснения хода решения.

Эффективность автоматизированного контроля зависит не от технологии самой по себе, а от методически выверенного проектирования заданий и педагогического сопровождения их использования [18]. Технология - это лишь инструмент. Если инструмент используется неумело, результат будет низким независимо от мощности системы. Поэтому ключевым фактором успеха является методическая грамотность учителя-разработчика заданий.

Для школьной практики необходимы задания, соотнесённые с программой, с чёткими критериями оценки и возможностью расположения по уровню сложности [9]. Задания не должны выходить за рамки школьного курса, но должны покрывать все ключевые темы. Критерии оценки должны быть прозрачны для обучающегося: он должен знать, за что получает баллы. Дифференциация позволяет работать с классом разного уровня подготовки, предлагая базовый уровень всем и повышенный уровень тем, кто справляется быстрее.

Основные методические требования к заданиям в системе автоматизированного контроля включают четкость формулировки. Условия задачи должны быть однозначными. Не должно быть места для двоякого понимания требований к вводу и выводу. Наличие примеров обязательно: каждая задача должна сопровождаться примерами ввода и вывода, иллюстрирующими логику работы программы. Система тестов должна быть надежной: набор тестов должен включать базовые тесты (проверка основной логики), граничные тесты (минимальные и максимальные значения, пустые вводные данные) и стресс-тесты (проверка производительности на больших объемах данных).

Качество сообщений об ошибках также критически важно. Сообщения системы должны быть понятны обучающемуся. Вместо «Runtime Error» лучше писать «Программа завершилась аварийно, проверьте деление на ноль или выход за границы массива». Это реализует принцип полноты ООД по Гальперину. Поэтапность заданий должна соответствовать теории формирования умственных

действий: от заполнения пропусков до написания с нуля. Возможность частичного начисления баллов позволяет мотивировать обучающихся, даже если они не решили задачу полностью: если задача состоит из нескольких подпунктов, система должна позволять начислить баллы за выполненные части, даже если полное решение не прошло все тесты.

Важно также предусмотреть методическое сопровождение для учителя. Учитель должен уметь анализировать отчеты системы, выявлять типичные ошибки класса и корректировать учебный план на основе полученной аналитики. Автоматизация не освобождает учителя от педагогической работы, а меняет ее характер, делая его более аналитическим и управленческим. Учитель становится дизайнером образовательной траектории, используя данные системы для принятия решений.

Выводы по главе 1:

В первой главе нами был проведен комплексный анализ психолого-педагогической и методической литературы, а также изучены нормативно-правовые аспекты использования цифровых инструментов в школе.

На основе проведенного теоретического исследования можно сделать следующие основные выводы:

Во-первых, в условиях цифровой образовательной среды контроль учебных достижений перестает быть просто административной процедурой выставления оценок. Он становится мощным инструментом педагогической обратной связи. Автоматизация рутинных процессов проверки кода позволяет высвободить время учителя для индивидуальной работы с обучающимися и обеспечивает мгновенную реакцию на ошибки ученика, что критически важно для поддержания его мотивации.

Во-вторых, проектирование систем автоматизированного контроля не должно сводиться лишь к техническим возможностям платформы. Как показал анализ теорий Л.С. Выготского, П.Я. Гальперина и А.Н. Леонтьева, система должна работать в «зоне ближайшего развития» обучающегося. Это означает, что обратная связь от системы должна быть не просто констатацией факта ошибки, а

предоставлять обучающемуся полноценную ориентировочную основу действия для ее самостоятельного исправления.

В-третьих, обучение программированию требует выхода за рамки традиционных тестов. Диагностика должна охватывать не только концептуальный, но и практический, алгоритмический и рефлексивный уровни. При этом важно учитывать, что процесс написания и отладки кода способствует развитию алгоритмического мышления, дисциплины ума и навыков проектной деятельности, что полностью соответствует требованиям ФГОС СОО.

В-четвертых, для условий общеобразовательной школы наиболее оптимальной является связка LMS Moodle и плагина CodeRunner. Она отвечает требованиям безопасности, интегрируется в привычную школьную среду и поддерживает адаптивный режим работы. Однако сама по себе платформа не гарантирует качества знаний. Ключевым фактором успеха является методическая грамотность учителя при проектировании заданий: условия задач должны быть однозначными, система тестов - надежной (включая граничные случаи), а градация сложности - соответствовать теории поэтапного формирования умственных действий.

В-пятых, системообразующими факторами эффективной системы заданий выступают: полный охват тем школьного курса (от базового ввода-вывода до рекурсии и вложенных структур данных), строгая последовательность использования тем в соответствии с логикой формирования умственных действий и разнообразие типов заданий (от заполнения пропусков до самостоятельного проектирования алгоритма).

Таким образом, теоретический анализ позволил нам обосновать необходимость создания с помощью плагина CodeRunner специализированной системы заданий для автоматизированного контроля и сформулировать жесткие методические требования к ее компонентам.

ГЛАВА 2. РАЗРАБОТКА И МЕТОДИЧЕСКОЕ ОБОСНОВАНИЕ СИСТЕМЫ ЗАДАНИЙ ДЛЯ АВТОМАТИЗИРОВАННОГО КОНТРОЛЯ УЧЕБНЫХ ДОСТИЖЕНИЙ ПО ПРОГРАММИРОВАНИЮ

На основе теоретического анализа, проведённого в первой главе, был сделан вывод о том, что эффективность автоматизированного контроля определяется не столько технической мощностью платформы, сколько методической грамотностью проектирования заданий. В связи с этим вторая глава ВКР посвящена решению практической задачи: разработке целостной системы заданий для автоматизированного контроля учебных достижений обучающихся старших классов по программированию на языке Python, её методическому обоснованию и апробации в учебном процессе.

2.1. Структура и содержание системы заданий по разделам школьного курса

Разработанная система заданий охватывает ключевые разделы школьного курса «Алгоритмы и программирование» и построена по принципу поэтапного усложнения ориентировочной основы действия. Каждая тема включает набор задач, методически направленных на формирование конкретного уровня компетенций: от технологического, связанного с синтаксисом и вводом-выводом, до алгоритмического, связанного с выбором структуры данных, оптимизацией и рекурсией.

Система заданий была разработана для восьми тем, представленных в разделе «Программирование» школьного курса информатики. Общий банк вопросов состоит из шестидесяти пяти заданий, распределённых по темам следующим образом.

Тема «Введение в Python», включающая операторы ввода, вывода и типы данных, содержит пять заданий.

Тема «Арифметические операции» также включает пять заданий.

Тема «Условный оператор» представлена девятью заданиями.

Тема «Цикл while» содержит тринадцать заданий.

Тема «Цикл for» включает одиннадцать заданий.

Тема «Списки» представлена двенадцатью заданиями.

Тема «Строки» содержит пять заданий.

Тема «Функции» включает пять заданий.

Дополнительно в систему включены задания по темам «Множества и словари» и «Рекурсия», что позволяет охватить все ключевые содержательные линии школьного курса.

Ниже представлено описание системы заданий по основным темам раздела с указанием методического фокуса каждой темы.

Тема 1. Базовый ввод-вывод и арифметические операции.

В рамках данной темы разработаны задачи «Дележ яблок», «Электронные часы», «Следующее и предыдущее» и «Парты». Методический фокус этих заданий заключается в формировании первичной ориентировочной основы действия работы с числовыми типами данных, отработке преобразования типов с использованием функции int (), понимании различий между делением с плавающей точкой, целочисленным делением и взятием остатка. Задания содержат скрытые тесты на отрицательные и нулевые входные значения, что позволяет диагностировать понимание области допустимых значений. Предварительная загрузка содержит только каркас ввода, оставляя обучающемуся пространство для самостоятельной реализации вычислительной логики.

Задача «Дележ яблок» требует от обучающегося понимания операций целочисленного деления и взятия остатка. n школьников делят k яблок поровну, неделящийся остаток остаётся в корзинке. Программа получает на вход числа n и k и должна вывести искомое количество яблок, доставшихся каждому школьнику, и количество яблок, оставшихся в корзинке. Эта задача методически направлена на формирование понимания различий между обычным делением, целочисленным делением и операцией взятия остатка.

Задача «Электронные часы» требует понимания работы с остатками и целочисленным делением в более сложном контексте. Дано число n. С начала суток прошло n минут. Необходимо определить, сколько часов и минут будут показывать электронные часы в этот момент. Программа должна вывести два

числа: количество часов от 0 до 23 и количество минут от 0 до 59. При этом число n может быть больше, чем количество минут в сутках, что требует от обучающегося понимания циклической природы времени.

Задача «Следующее и предыдущее» направлена на отработку базовых арифметических операций и форматирования вывода. Необходимо написать программу, которая считывает целое число и выводит текст, аналогичный приведённому в примере. Эта задача формирует навык работы с операторами сложения и вычитания, а также навык точного форматирования строкового вывода.

Задача «Парты» требует применения целочисленного деления в практическом контексте. В некоторой школе решили набрать три новых математических класса и оборудовать кабинеты для них новыми партами. За каждой партой может сидеть два учащихся. Известно количество учащихся в каждом из трёх классов. Необходимо вывести наименьшее число парт, которое нужно приобрести. Эта задача формирует понимание того, как целочисленное деление применяется для решения практических задач округления в большую сторону.

В качестве примера реализации задания в системе CodeRunner рассмотрим задачу «Дележ яблок». При создании данного вопроса настраивается тип вопроса CodeRunner с выбором языка python3. В общих настройках вопроса выбирается категория из банка вопросов, вводится название и текст вопроса. В поле ответа вводится правильный ответ, представляющий собой эталонный код решения. В поле предварительной загрузки ответа вводится шаблон для обучающегося, содержащий только каркас ввода. В поле тестовые случаи вводится несколько тестовых случаев, причём первый берётся за пример, видимый обучающемуся. Остальные тесты являются скрытыми и проверяют корректность решения на граничных значениях.

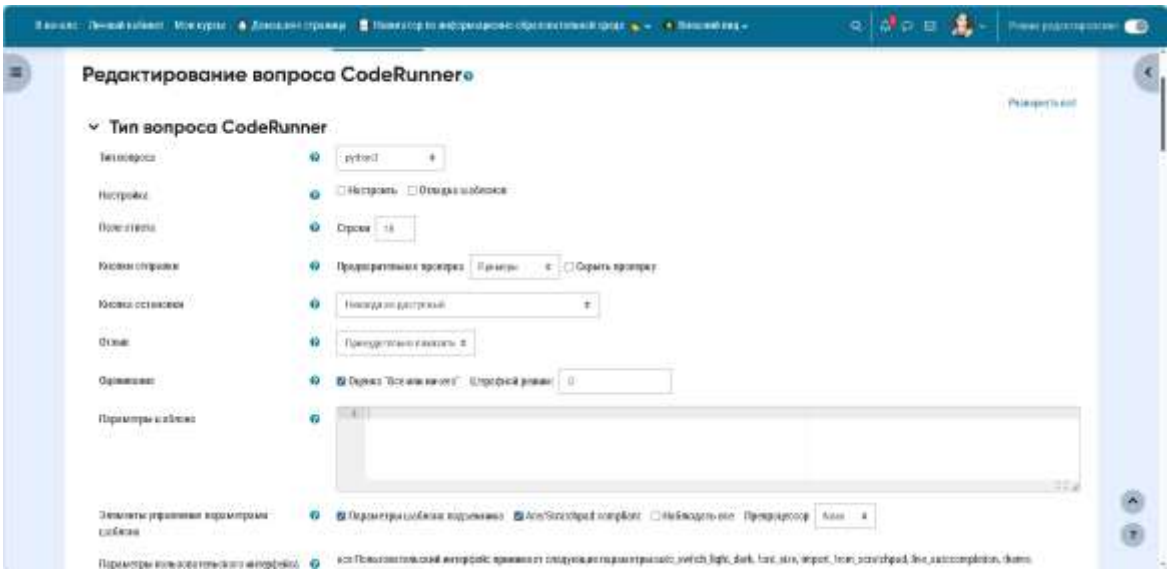


Рисунок 1. Настройки типа вопроса

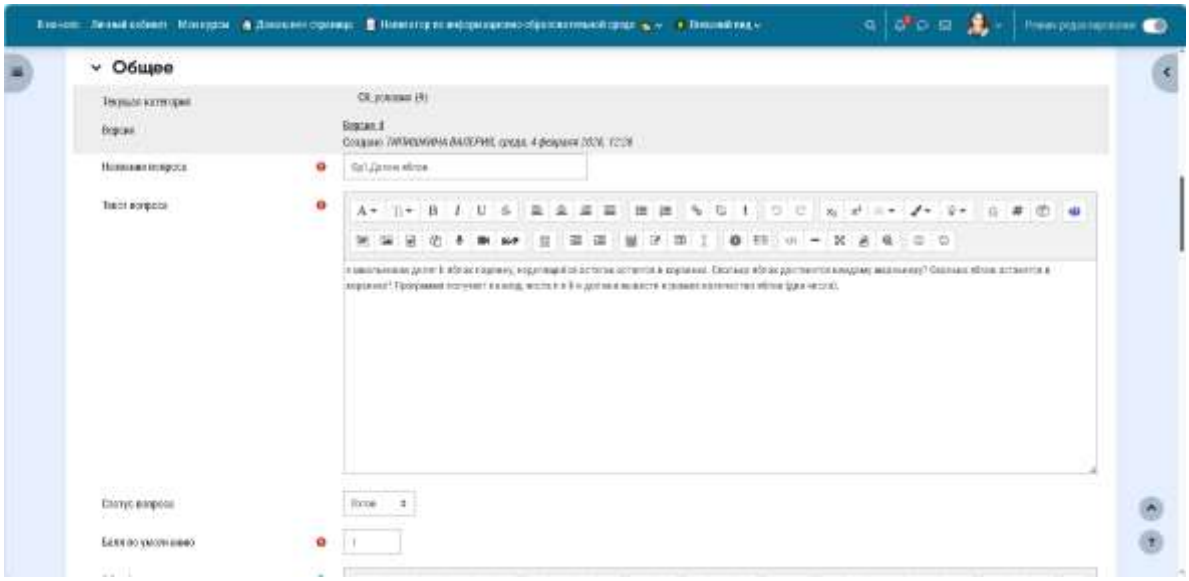


Рисунок 2. Общие настройки вопроса

Рисунок 3. Настройка поля ответ

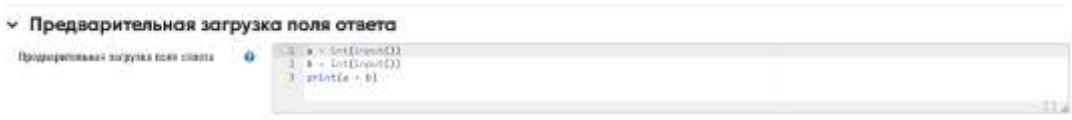


Рисунок 4. Настройка поля предварительная загрузка ответа

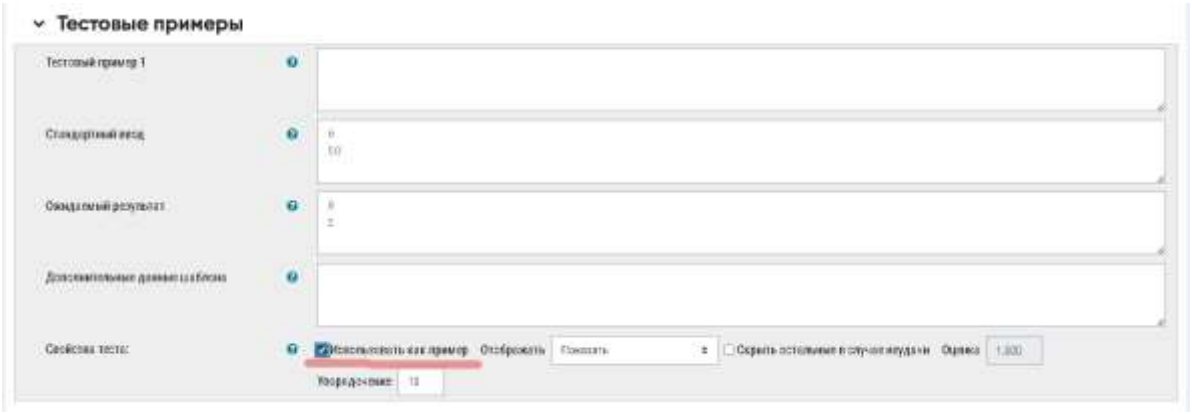


Рисунок 5. Настройка поля тестовые примеры

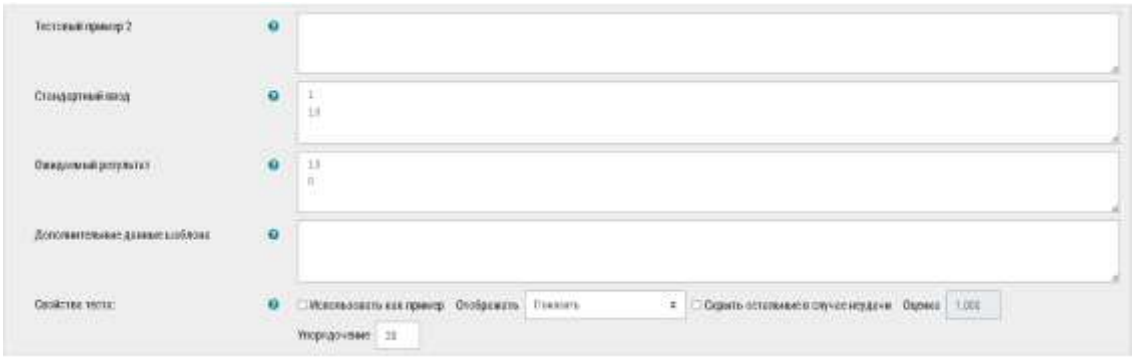


Рисунок 6. Настройка поля тестовые примеры

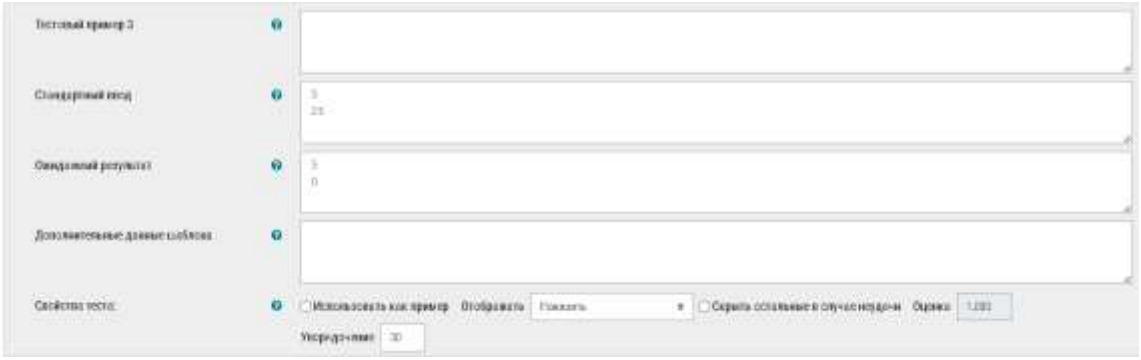


Рисунок 7. Настройка поля тестовые примеры

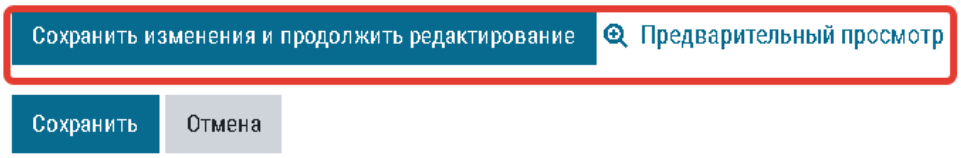


Рисунок 8. Сохранение и проверка вопроса

Тема 2. Условный оператор и логическое ветвление.

В рамках данной темы разработаны задачи «Минимум из двух и трёх чисел», «Високосный год», «Шахматная доска», «Линейное уравнение», «Ход ладьи», «Шоколадка» и «Яша плавает в бассейне». Методический фокус этих

заданий заключается в закреплении конструкций if-elif-else, отработке логических операторов and, or, not, формировании навыка анализа взаимно исключающих условий. Тестовая система покрывает все ветви логики, включая пограничные случаи. Ошибки в условии ветвления мгновенно отражаются в сообщении системы, направляя обучающегося к проверке порядка условий и приоритетов операций.

Задача «Минимум из трёх чисел» требует от обучающегося понимания вложенных условных конструкций. Даны три целых числа. Необходимо вывести значение наименьшего из них. Эта задача формирует навык анализа нескольких взаимно исключающих условий и выбора оптимальной структуры ветвления.

Задача «Високосный год» направлена на отработку сложных логических выражений. Дано натуральное число. Требуется определить, является ли год с данным номером високосным. В соответствии с григорианским календарём, год является високосным, если его номер кратен 4, но не кратен 100, а также если он кратен 400. Эта задача формирует понимание приоритетов логических операций и необходимости тщательной проработки граничных случаев.

Задача «Ход ладьи» требует понимания системы координат и условий попадания на шахматной доске. Шахматная ладья ходит по горизонтали или вертикали. Даны две различные клетки шахматной доски, необходимо определить, может ли ладья попасть с первой клетки на вторую одним ходом. Программа получает на вход четыре числа от 1 до 8 каждое, задающие номер столбца и номер строки сначала для первой клетки, потом для второй клетки. Эта задача формирует навык работы с несколькими переменными и условиями их взаимосвязи.

Задача «Шоколадка» направлена на понимание ограничений и условий выполнимости. Шоколадка имеет вид прямоугольника, разделённого на n на m долек. Шоколадку можно один раз разломить по прямой на две части. Необходимо определить, можно ли таким образом отломить от шоколадки часть, состоящую ровно из k долек. Эта задача формирует понимание того, как несколько условий объединяются в сложное логическое выражение.

Задача «Яша плавает в бассейне» требует понимания минимума из нескольких значений и работы с системой координат. Яша плавал в бассейне размером N на M метров и устал. В этот момент он обнаружил, что находится на расстоянии x метров от одного из длинных бортиков и y метров от одного из коротких бортиков. Какое минимальное расстояние должен проплыть Яша, чтобы выбраться из бассейна на бортик? Эта задача формирует навык работы с функцией min () и понимания того, как из нескольких возможных решений выбрать оптимальное.

Тема 3. Цикл while и итерационные вычисления.

В рамках данной темы разработаны задачи «Утренняя пробежка», «Длина последовательности», «Индекс максимума последовательности», «Второй максимум», «Количество элементов, равных максимуму», «Максимальное число идущих подряд равных элементов», «Количество нулей», «Сумма последовательности», «Лесенка», «Список квадратов», «Обратный порядок», «Потерянная карточка» и «Ряд». Методический фокус этих заданий заключается в отработке циклов for и while, функции range (), аккумуляторов и счётчиков. Задания дифференцированы: часть требует явного указания шага, часть работы с отрицательным шагом или прерывания цикла по условию. Тестовые наборы проверяют корректность при нулевых и единичных значениях, а также на больших объёмах данных, что косвенно диагностирует понимание алгоритмической сложности. Предварительные шаблоны содержат только объявление переменных, что соответствует этапу перехода от материализованной формы ориентировочной основы действия к внутреннему плану действий.

Задача «Утренняя пробежка» требует понимания циклов с условием продолжения. В первый день спортсмен пробежал x километров, а затем он каждый день увеличивал пробег на 10 процентов от предыдущего значения. По данному числу y необходимо определить номер дня, на который пробег спортсмена составит не менее y километров. Эта задача формирует понимание циклов while с условием завершения, зависящим от вычисляемого значения.

Задача «Длина последовательности» направлена на отработку циклов с обработкой последовательности неизвестной длины. Программа получает на вход последовательность целых неотрицательных чисел, каждое число записано в отдельной строке. Последовательность завершается числом 0, при считывании которого программа должна закончить свою работу и вывести количество членов последовательности, не считая завершающего числа 0. Эта задача формирует понимание паттерна обработки последовательности с признаком завершения.

Задача «Индекс максимума последовательности» требует понимания одновременного отслеживания нескольких характеристик последовательности. Последовательность состоит из натуральных чисел и завершается числом 0. Необходимо определить индекс наибольшего элемента последовательности. Если наибольших элементов несколько, необходимо вывести индекс первого из них. Нумерация элементов начинается с нуля. Эта задача формирует навык работы с несколькими аккумуляторами одновременно.

Задача «Второй максимум» направлена на понимание более сложных алгоритмов обработки последовательности. Последовательность состоит из различных натуральных чисел и завершается числом 0. Необходимо определить значение второго по величине элемента в этой последовательности. Гарантируется, что в последовательности есть хотя бы два элемента. Эта задача формирует понимание того, как отслеживать несколько максимальных значений одновременно.

Задача «Количество элементов, равных максимуму» требует двухпроходного алгоритма или одновременного отслеживания максимума и его количества. Последовательность состоит из натуральных чисел и завершается числом 0. Необходимо определить, сколько элементов этой последовательности равны её наибольшему элементу. Эта задача формирует понимание того, как комбинировать несколько задач обработки последовательности в одном цикле.

Задача «Максимальное число идущих подряд равных элементов» направлена на понимание паттерна сравнения соседних элементов. Дана последовательность натуральных чисел, завершающаяся числом 0. Необходимо

определить, какое наибольшее число подряд идущих элементов этой последовательности равны друг другу. Эта задача формирует навык работы с предыдущим и текущим элементами последовательности.

Задача «Количество нулей» требует понимания циклов с известным числом повторений. Дано N чисел: сначала вводится число N, затем вводится ровно N целых чисел. Необходимо подсчитать количество нулей среди введённых чисел и вывести это количество. Важно подсчитать количество чисел, равных нулю, а не количество цифр. Эта задача формирует понимание различий между циклами с известным и неизвестным числом повторений.

Задача «Лесенка» направлена на понимание вложенных циклов. По данному натуральному n, не превышающему 9, необходимо вывести лесенку из n ступенек, i-я ступенька состоит из чисел от 1 до i без пробелов. Эта задача формирует понимание вложенных циклов и зависимости внутреннего цикла от внешнего.

Задача «Потерянная карточка» требует понимания математического подхода к решению задач. Для настольной игры используются карточки с номерами от 1 до N. Одна карточка потерялась. Необходимо найти её, зная номера оставшихся карточек. Дано число N, далее N минус 1 номер оставшихся карточек. Программа должна вывести номер потерянной карточки. При этом массивами и аналогичными структурами данных пользоваться нельзя. Эта задача формирует понимание того, как математические свойства могут быть использованы для упрощения алгоритма.

Тема 4. Строки и посимвольная обработка.

В рамках данной темы разработаны задачи «Второе вхождение», «Делаем срезы», «Замена внутри фрагмента», «Длина слова» и дополнительные задания на форматирование строк. Методический фокус этих заданий заключается в освоении индексации, срезов, методов строк, таких как find(), replace(), count() и других. Задания методически выстроены так, чтобы исключить ручной перебор и приучить к использованию встроенных инструментов языка. Тесты включают строки нулевой длины, строки с одним символом, полные совпадения и

отсутствие искомого элемента. Это формирует культуру написания устойчивого к аномальным данным кода.

Задача «Второе вхождение» требует понимания методов поиска подстрок. Дана строка. Необходимо найти в этой строке второе вхождение буквы f и вывести индекс этого вхождения. Если буква f в данной строке встречается только один раз, необходимо вывести число минус 1, а если не встречается ни разу, необходимо вывести число минус 2. Эта задача формирует понимание методов find() и работы с различными случаями отсутствия искомого элемента.

Задача «Делаем срезы» направлена на комплексную отработку работы со срезами строк. Дана строка. Необходимо выполнить девять различных операций со срезами: вывести третий символ, предпоследний символ, первые пять символов, всю строку кроме последних двух символов, все символы с чётными индексами, все символы с нечётными индексами, все символы в обратном порядке, все символы строки через один в обратном порядке, начиная с последнего, и длину данной строки. Эта задача формирует комплексное понимание всех видов срезов и их параметров.

Задача «Замена внутри фрагмента» требует понимания комбинирования методов строк. Дана строка. Необходимо заменить в этой строке все появления буквы h на букву H, кроме первого и последнего вхождения. Эта задача формирует понимание того, как комбинировать методы find(), rfind() и replace() для решения сложной задачи обработки строк.

Тема 5. Составные типы данных: списки, множества, словари.

В рамках данной темы разработаны задачи «Больше предыдущего», «Количество различных элементов», «Наибольший элемент», «Переставить соседние», «Уникальные элементы», «Четные элементы», «Удаление дубликатов», «Частотный анализ», «Объединение множеств», «Номер появления слова», «Выборы в США», «Кубики» и «Угадай число». Методический фокус этих заданий заключается в формировании умения выбирать оптимальную структуру данных под конкретную задачу. Задания направлены на осознание различий между изменяемыми и неизменяемыми коллекциями, уникальностью

элементов, скоростью поиска. Тесты проверяют корректность обработки пустых коллекций, повторяющихся элементов, регистронезависимого сравнения. Система оценивания покомпонентна: баллы начисляются за правильный тип структуры, корректность алгоритма и прохождение всех тестов.

Задача «Больше предыдущего» требует понимания итерации по списку с доступом к предыдущему элементу. Дан список чисел. Необходимо вывести все элементы списка, которые больше предыдущего элемента. Эта задача формирует понимание паттерна сравнения соседних элементов в списке.

Задача «Количество различных элементов» направлена на понимание работы с упорядоченными списками. Дан список, упорядоченный по не убыванию элементов в нём. Необходимо определить, сколько в нём различных элементов. Эта задача формирует понимание того, как использование свойства упорядоченности упрощает решение задачи.

Задача «Наибольший элемент» требует понимания одновременного отслеживания значения и индекса. Дан список чисел. Необходимо вывести значение наибольшего элемента в списке, а затем индекс этого элемента в списке. Если наибольших элементов несколько, необходимо вывести индекс первого из них. Эта задача формирует навык работы с несколькими характеристиками элемента одновременно.

Задача «Переставить соседние» направлена на понимание работы с индексами и обменом значений. Необходимо переставить соседние элементы списка. Если элементов нечётное число, то последний элемент остаётся на своём месте. Эта задача формирует понимание паттерна обработки элементов с шагом 2 и работы с остатком от деления.

Задача «Уникальные элементы» требует понимания вложенных циклов или использования дополнительных структур данных. Дан список. Необходимо вывести те его элементы, которые встречаются в списке только один раз. Элементы нужно выводить в том порядке, в котором они встречаются в списке. Эта задача формирует понимание различных подходов к подсчёту частоты элементов.

Задача «Выборы в США» направлена на понимание работы со словарями. Как известно, в США президент выбирается не прямым голосованием, а путём двухуровневого голосования. В первой строке дано количество записей. Далее каждая запись содержит фамилию кандидата и число голосов, отданных за него в одном из штатов. Необходимо подвести итоги выборов: для каждого из участника голосования определить число отданных за него голосов. Участников нужно выводить в алфавитном порядке. Эта задача формирует понимание использования словарей для подсчёта частоты и сортировки результатов.

Задача «Кубики» требует понимания операций над множествами. Аня и Боря любят играть в разноцветные кубики, причём у каждого из них свой набор и в каждом наборе все кубики различны по цвету. Необходимо найти три множества: номера цветов кубиков, которые есть в обоих наборах, номера цветов кубиков, которые есть только у Ани, и номера цветов кубиков, которые есть только у Бори. Для каждого из множеств необходимо вывести сначала количество элементов в нём, а затем сами элементы, отсортированные по возрастанию. Эта задача формирует понимание операций пересечения, разности и объединения множеств.

Задача «Угадай число» направлена на понимание использования множеств для отслеживания возможных значений. Август и Беатриса играют в игру. Август загадал натуральное число от 1 до n. Беатриса пытается угадать это число, для этого она называет некоторые множества натуральных чисел. Август отвечает Беатрисе YES, если среди названных ей чисел есть задуманное, или NO в противном случае. Необходимо определить, какие числа мог задумать Август. Эта задача формирует понимание использования операций пересечения и разности множеств для сужения множества возможных решений.

Задача «Номер появления слова» требует понимания использования словарей для подсчёта частоты. В единственной строке записан текст. Для каждого слова из данного текста необходимо подсчитать, сколько раз оно встречалось в этом тексте ранее. Эта задача формирует понимание использования словарей для эффективного подсчёта частоты элементов.

Тема 6. Функции, модульность и рекурсия.

В рамках данной темы разработаны задачи «Возведение в степень», «Факториал», «Фибоначчи», «Модульное разбиение задачи», «Количество делителей», «Первая цифра числа». Методический фокус этих заданий заключается в переходе на алгоритмический уровень компетенций. Задания требуют декомпозиции задачи, выделения подпрограмм, понимания механизма передачи параметров и возврата значений. Рекурсивные задачи включают скрытые проверки на наличие базового случая и предотвращение переполнения стека. Сообщения об ошибках настроены на диагностику типичных проблем, таких как RecursionError, отсутствие return и неправильный порядок аргументов. Это реализует принцип полноты ориентировочной основы действия на этапе автоматизированного контроля.

Задача «Возведение в степень» требует понимания рекуррентных соотношений. Дано действительное положительное число a и целое неотрицательное число n. Необходимо вычислить a в степени n, не используя циклы, возведение в степень через оператор двойной звёздочки и функцию math.pow(), а используя рекуррентное соотношение. Решение необходимо оформить в виде функции power(a, n). Эта задача формирует понимание базового принципа рекурсии и необходимости базового случая.

Задача «Факториал» направлена на понимание итеративного и рекурсивного подходов. Факториалом числа n называется произведение 1 на 2 на ... на n. По данному натуральному n необходимо вычислить значение n!. Пользоваться математической библиотекой math в этой задаче запрещено. Эта задача формирует понимание того, как одна и та же задача может быть решена итеративно и рекурсивно.

Задача «Фибоначчи» требует понимания рекурсии с несколькими базовыми случаями. Необходимо написать функцию fib(n), которая по данному целому неотрицательному n возвращает n-е число Фибоначчи. В этой задаче нельзя использовать циклы, необходимо использовать рекурсию. Эта задача формирует

понимание рекурсии с двумя рекурсивными вызовами и важности базовых случаев.

Задача «Количество делителей» направлена на понимание декомпозиции задачи на подзадачи. Необходимо написать функцию для нахождения количества делителей натурального числа. Используя её, необходимо найти все простые числа в заданном интервале чисел. Эта задача формирует понимание того, как выделение подпрограммы упрощает решение сложной задачи.

В рамках разработанной системы заданий выделены три типа вопросов, реализуемых средствами плагина CodeRunner. Типология опирается на теорию поэтапного формирования умственных действий П.Я. Гальперина и выстраивает последовательный переход обучающегося от репродуктивных операций к самостоятельному алгоритмическому проектированию.

Первый тип - вопрос с заполнением шаблона. Обучающемуся предъявляется заготовка кода с пропущенными фрагментами, которые требуется восстановить. Данный тип ориентирован на проверку понимания конкретных языковых конструкций - циклов, условных операторов, функций. Методическое достоинство такого формата состоит в снижении когнитивной нагрузки: обучающийся сосредоточен на осмыслении целевой конструкции, тогда как синтаксическая периферия уже задана шаблоном и не отвлекает внимания. В логике теории Гальперина этот тип заданий соответствует материализованной форме ориентировочной основы действия - этапу, на котором учебная операция совершается с развёрнутой внешней опорой.

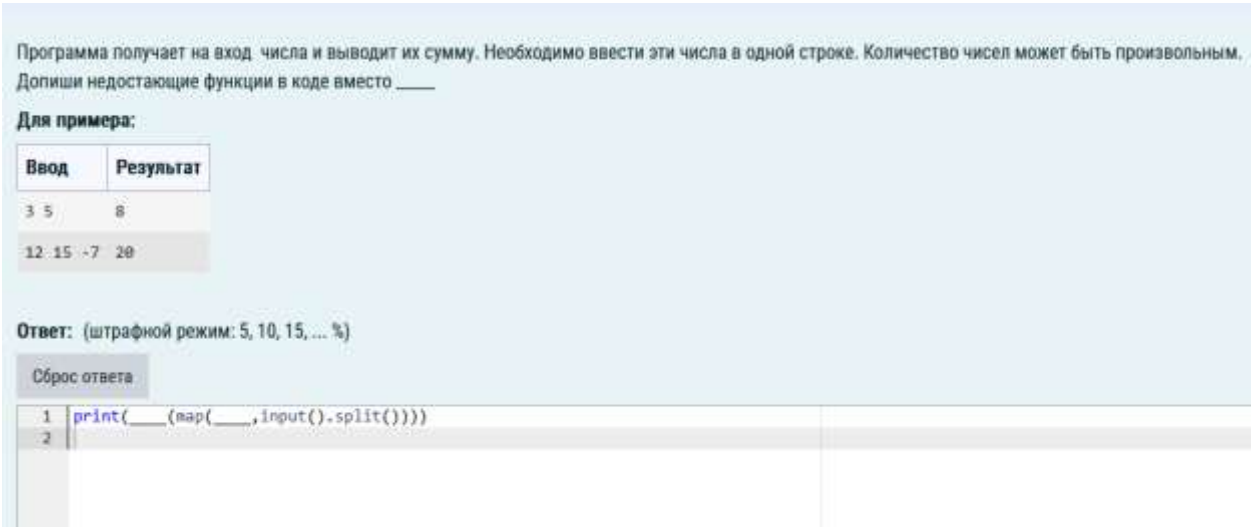


Рисунок 9. Скриншот задания с заполнением шаблона

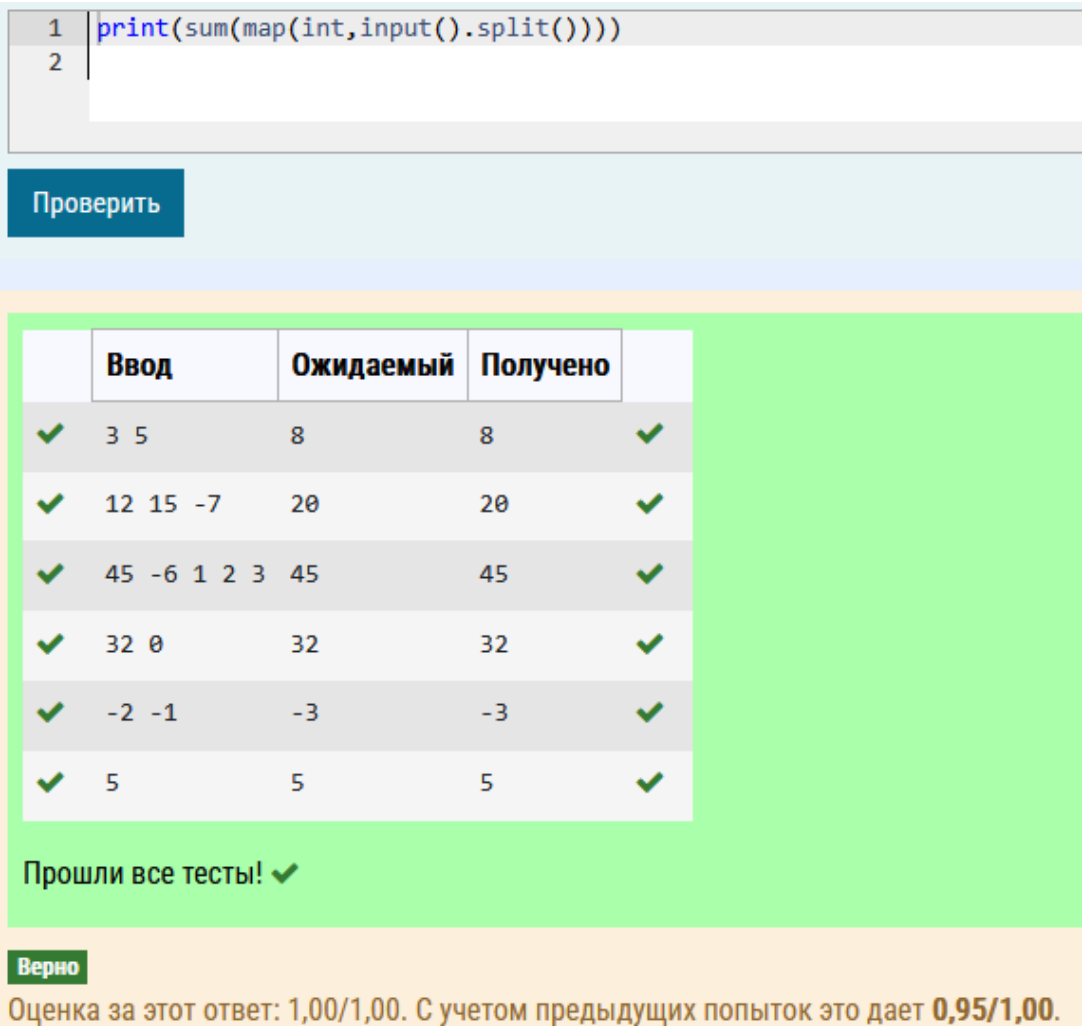
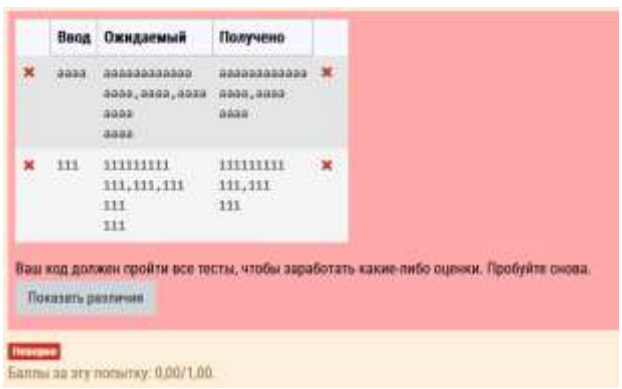


Рисунок 10. Скриншот результата выполнения задания

Второй тип - вопрос с полной проверкой кода. Обучающийся пишет полный код программы с нуля, что проверяет общие навыки программирования. Методическая ценность этого типа заключается в том, что он требует от

обучающегося самостоятельного проектирования алгоритма, выбора структуры данных и реализации решения. Этот тип заданий соответствует этапу внутренней речи, когда обучающийся самостоятельно проектирует алгоритм и пишет код.

Рисунок 11. Скриншот задания с автоматической проверкой, выполняемого во время лекции, изучения операторов ввода - вывода.



При неверном коде необходимо сравнить ожидаемый результат и полученный

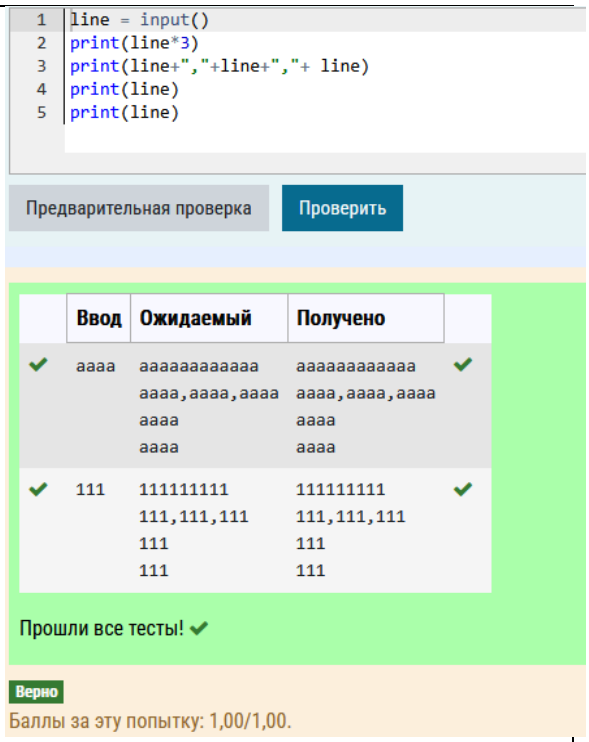


Рисунок 12. Скриншот автоматической проверки неверного/верного решения

Третий тип - вопрос по устранению ошибок и корректировке кода. Данный тип вопроса технически реализуется как тип 1, но его необходимо выделить в отдельный с позиции методики обучения. Обнаружение ошибок, их устранение и рефакторинг кода являются важными умениями, формируемыми в процессе обучения программированию. Методическая ценность этого типа заключается в том, что он формирует навык отладки и понимания чужого кода, что является важной профессиональной компетенцией. Этот тип заданий соответствует автоматизированному уровню, когда обучающийся решает задачи на оптимизацию, отладку и анализ чужого кода.

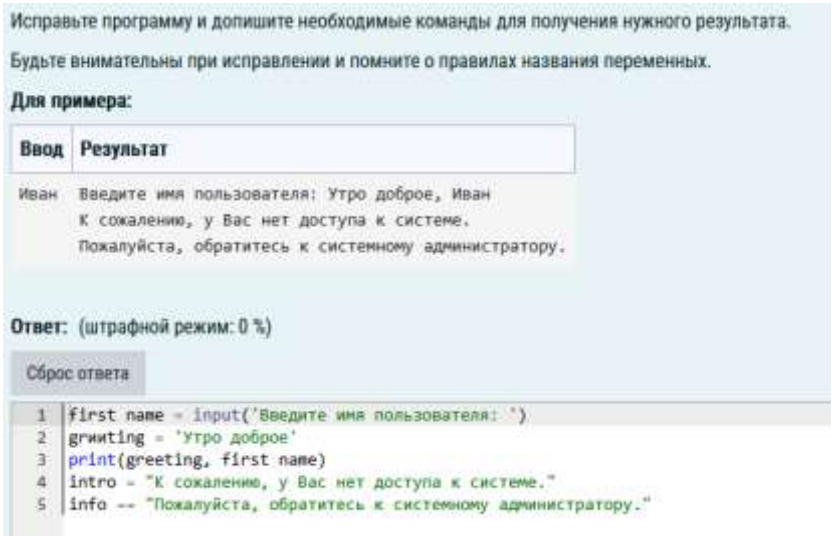


Рисунок 13. Скриншот задания по исправлению кода с автоматической проверкой

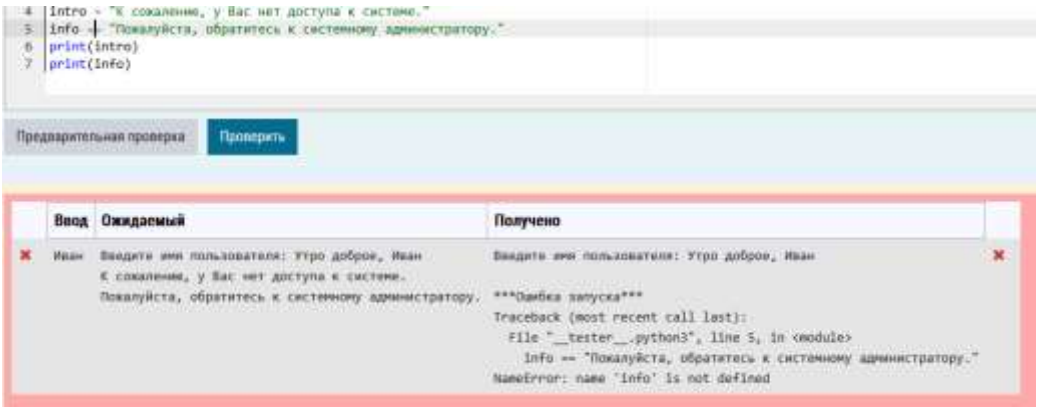


Рисунок 14. Скриншот результата автоматической проверки при синтаксических ошибках

Для закрепления практических навыков написания кода по каждой теме предполагается от четырёх до восьми задач для самостоятельной работы. Задания по заполнению шаблона используются также в тестах в конце лекционных занятий с целью оценки первичного усвоения изученного материала.

Системные характеристики разработанного набора заданий включают прогрессию сложности от задач с готовым каркасом ввода-вывода до полностью самостоятельного проектирования алгоритма, дифференциацию, при которой базовый уровень доступен всем, а повышенный уровень, включающий оптимизацию, рекурсию и работу с большими данными, предлагается обучающимся, освоившим базовую ориентировочную основу действия, прозрачность критериев, при которой система начисления баллов соответствует требованиям ФГОС среднего общего образования и позволяет обучающемуся видеть, за какие подэтапы решения начислены баллы, и педагогическую обратную связь, при которой сообщения об ошибках формулируются не как технические отчёты, а как дидактические подсказки, направляющие к самостоятельному исправлению.

2.2. Апробация системы заданий и оценка её педагогической эффективности

Разработанная система заданий для автоматизированного контроля учебных достижений по программированию на языке Python для обучающихся старших классов была апробирована в курсе «Программирование» для студентов первого курса Института математики, физики и информатики Красноярского государственного педагогического университета им. В.П. Астафьева. Поскольку этот курс является начальным в обучении студентов, результаты апробации могут коррелировать с результатами обучения старшеклассников программированию.

В рамках каждой темы студентам предлагались практические задания, которые выполнялись на аудиторных занятиях и контролировались преподавателем, а также самостоятельная работа с целью закрепления практических навыков, реализованная в виде описываемой системы заданий. В рамках каждой самостоятельной работы студентам необходимо было выполнить от пяти до восьми заданий разных типов с преимуществом заданий с полной

проверкой кода. Кроме того, задания по заполнению шаблона использовались в тестах в конце лекционных занятий с целью оценки первичного усвоения изученного материала.

Для оценки эффективности разработанной системы был составлен опросник, включающий восемь вопросов, направленных на оценку восприятия, эффективности и опыта самостоятельной работы с заданиями CodeRunner. Был проведён опрос сорока трёх студентов первого курса.

Результаты опроса показали:

1. задания с автоматической проверкой помогают учиться программировать самостоятельно (без помощи преподавателя) большинству опрошенных: 23,3% -полностью помогает, 14% студентов оценили в 5 баллов из 5 и 41,9 % оценили в 4 балла из 5;

2. основные трудности при работе с CodeRunner - это непонимание сообщения об ошибках системы (34,9%), слишком много попыток уходит на отладку синтаксиса, а не логики (25,6 %) и не хватает примеров правильных решений для сравнения (27,9%);

3. вопросы данного типа повышают мотивацию и вовлеченность: сильно повышает – хочется «пройти все тесты» как в игре (39,5 %), немного повышает – приятно видеть зелёную галочку (39,5 %);

4. общую пользу заданий с автоматической проверкой для обучения программированию отметили 84% студентов, а 88, 4% хотели бы, чтобы все задания в электронном курсе "Программирование", включая лабораторные работы, были сделаны с автоматической проверкой кода.

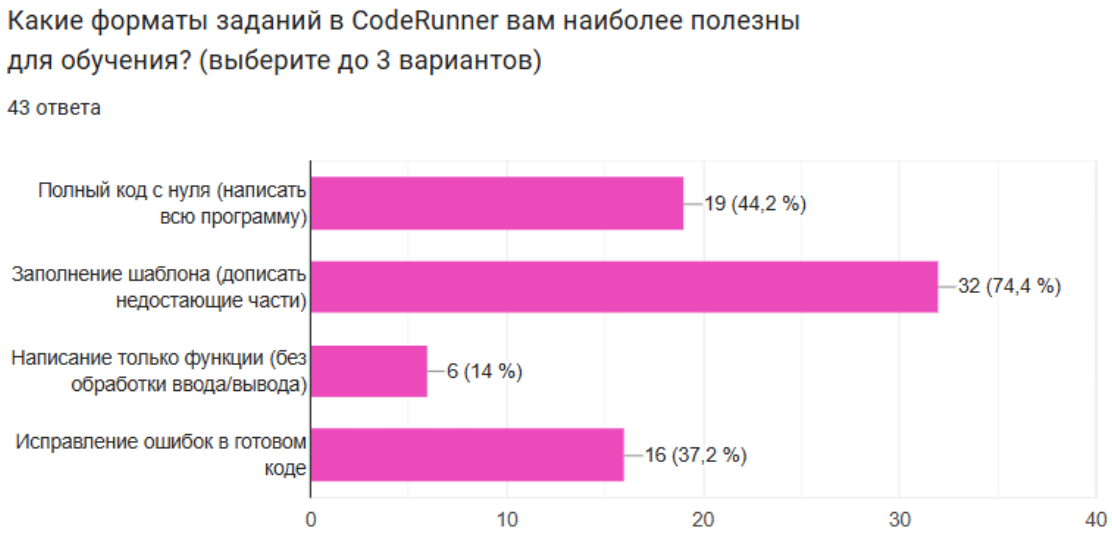


Рисунок 15. Диаграмма ответов на вопрос

Анализ показал, что большинство студентов предпочитают задания с полной проверкой кода, которые позволяют развивать навыки самостоятельного программирования. Задания с заполнением шаблона также пользуются популярностью, особенно на начальных этапах обучения. Задания по устранению ошибок вызывают наибольшие трудности, но именно они формируют наиболее важные профессиональные компетенции.

Выводы по главе 2

Во второй главе нами был проведён сравнительный анализ программных платформ и разработана целостная методика проектирования системы заданий для обучающихся старших классов.

Нами спроектирована и описана система заданий, охватывающая все ключевые разделы школьного курса «Алгоритмы и программирование», от базового ввода-вывода и условных операторов до работы со строками, циклами, составными типами данных и рекурсией. Общий банк вопросов состоит из шестидесяти пяти заданий, распределённых по восьми темам. Система построена на принципе поэтапного усложнения ориентировочной основы действия по П.Я. Гальперину, что обеспечивает плавный переход обучающегося от репродуктивных действий к самостоятельному алгоритмическому проектированию. Выделены три типа заданий, реализованных в CodeRunner: заполнение шаблона для проверки понимания отдельных конструкций, полная

проверка кода для оценки навыков разработки программы с нуля и устранение ошибок и корректировка кода для формирования умений отладки и рефакторинга.

Разработанная система была апробирована в курсе программирования для студентов первого курса Института математики, физики и информатики Красноярского государственного педагогического университета им. В.П. Астафьева. Результаты опроса сорока трёх респондентов показали, что восемьдесят четыре процента студентов отметили общую пользу заданий с автоматической проверкой, восемьдесят восемь целых и четыре десятых процента выразили желание использовать такой формат для всех лабораторных работ, большинство опрошенных, свыше шестидесяти пяти процентов, подтвердили, что автоматическая проверка помогает самостоятельно осваивать программирование и повышает вовлечённость.

Таким образом, вторая глава позволила нам перейти от общих теоретических обоснований к конкретному методическому проектированию. Созданная система заданий полностью отвечает требованиям ФГОС среднего общего образования и психолого-педагогическим принципам, сформулированным в первой главе. Разработанная система заданий трансформирует автоматизированный контроль из технической процедуры проверки кода в педагогически обоснованный инструмент развития алгоритмического мышления и учебной самостоятельности старшеклассников.

Заключение

В ходе выпускной квалификационной работы достигнута поставленная цель - теоретически обоснована и разработана система заданий для автоматизированного контроля учебных достижений старшеклассников по программированию на базе LMS Moodle и плагина CodeRunner.

Анализ литературы, проведённый в первой главе, показал, что в условиях цифровой трансформации контроль перестаёт быть сугубо административной процедурой. С опорой на концепцию зоны ближайшего развития Л.С. Выготского и теорию поэтапного формирования умственных действий П.Я. Гальперина установлено: эффективность автоматизированного контроля прямо зависит от качества педагогического проектирования, а именно от полноты ориентировочной основы действия. Контроль должен выступать средством поддержки обучающегося, а не фильтром отсева.

Определена многоуровневая структура учебных достижений в программировании, включающая концептуальный, технологический, алгоритмический, практический и рефлексивный уровни. Это позволило выстроить диагностику, оценивающую не только верный ответ, но и сформированность навыка написания устойчивого, оптимального кода. Процесс программирования развивает алгоритмическое мышление и дисциплину ума, что соответствует требованиям ФГОС.

Сравнительный анализ платформ показал, что для школы оптимальна связка Moodle и CodeRunner - по критериям информационной безопасности, гибкости многовариантного тестирования и возможности мгновенной обратной связи. Педагогическая ценность платформы раскрывается лишь при наличии методически выверенной системы заданий.

Центральный практический результат - целостная система из шестидесяти пяти заданий по восьми темам школьного курса «Алгоритмы и программирование»: от ввода-вывода и условных операторов до функций и рекурсии. Каждое задание включает однозначную формулировку, надёжный набор тестов, шаблон кода и педагогически ориентированные сообщения об

ошибках. Выделены три типа заданий: заполнение шаблона, полная проверка кода и устранение ошибок, что обеспечивает переход от репродуктивных действий к самостоятельному проектированию в логике теории Гальперина.

Система опирается на три фактора: полный охват тем, строгую последовательность и разнообразие типов заданий. Это устраняет типичные пробелы школьной практики, такие как разрыв между возможностями платформ и содержанием программы, слабую проработку тестов, формальную обратную связь и дефицит методических материалов.

Апробация проведена на первом курсе Института математики, физики и информатики КГПУ им. В.П. Астафьева. Опрос сорока трёх респондентов подтвердил эффективность подхода: 84% отметили пользу заданий, 88,4% выразили желание использовать формат для всех лабораторных работ. Около трети указали на сложности с пониманием сообщений об ошибках и недостаток примеров правильных решений - это направления для доработки.

Практическая значимость состоит в том, что система может непосредственно применяться учителями для текущего, рубежного и итогового контроля. Комплекс заданий с пошаговой технической инструкцией представляет собой готовый к внедрению продукт, не требующий от педагога глубоких знаний программирования.

Гипотеза подтвердилась: автоматизированный контроль становится инструментом развития алгоритмического мышления и учебной самостоятельности лишь при методически грамотном проектировании, опирающемся на фундаментальные психолого-педагогические теории.

Перспективы дальнейших исследований - расширение базы заданий темами «Работа с файлами» и «Объектно-ориентированное программирование», а также изучение возможностей интеграции ИИ для семантического анализа стиля кода и совершенствование обратной связи.

Библиографические списки

1. Андреева Е. В. Преподавание программирования в школе: методическое пособие / Е. В. Андреева. Москва: БИНОМ. Лаборатория знаний, 2018. 216 с. ISBN 978-5-9963-3920-4. URL: https://resources.mgpu.ru/showlibraryurl.php?docid=474518
2. Белоусова Л. И. Становление и развитие личности в процессе обучения программированию: монография / Л. И. Белоусова. Москва: БИНОМ. Лаборатория знаний, 2012. 176 с. ISBN 978-5-9963-0987-0. URL: https://search.rsl.ru/ru/record/01005425299
3. Гальперин П. Я. Лекции по психологии / П. Я. Гальперин. Москва: Изд-во МГУ, 1991. 264 с. ISBN 5-211-01762-8. URL: https://rusneb.ru/catalog/000199_000009_002582971/
4. Гальперин, П. Я. Психология мышления и учение о поэтапном формировании умственных действий / П. Я. Гальперин // Психология: предмет и метод: избранные психологические труды / П. Я. Гальперин. Москва: Издательство Московского университета, 2023. С. 620–657. URL:https://lib.mgppu.ru/OpacUnicode/app/webroot/index.php?url=/notices/index/IdNotice:482967/Source:default#
5. Гейн А.Г. Методика обучения основам программирования. Екатеринбург: УрГПУ, 2010. 128 с.
6. Гущин А. Н. Цифровая дидактика: системные основания и образ будущего / А. Н. Гущин // Педагогика и просвещение. 2022. № 2. С. 100–115. DOI 10.7256/2454-0676.2022.2.35657. EDN MFFOKP. URL: ЦИФРОВАЯ ДИДАКТИКА: СИСТЕМНЫЕ ОСНОВАНИЯ И ОБРАЗ БУДУЩЕГО
7. Жуков, И. А. Модель представления многовариантных заданий для автоматизированного контроля знаний по программированию / И. А. Жуков, Ю. Л. Костюк // Вестник Томского государственного университета. Управление, вычислительная техника и информатика. 2020. № 53. С. 110–117. DOI 10.17223/19988605/53/11. EDN TERGOW. URL: МОДЕЛЬ

ПРЕДСТАВЛЕНИЯ МНОГОВАРИАНТНЫХ ЗАДАНИЙ ДЛЯ АВТОМАТИЗИРОВАННОГО КОНТРОЛЯ ЗНАНИЙ ПО ПРОГРАММИРОВАНИЮ

8. Кадина, Д. И. Автоматизация проверки на плагиат: новый подход к анализу кода в цифровой образовательной платформе / Д. И. Кадина, А. Г. Леонов, Н. С. Мартынов, К. А. Сидоров // Научно-технический вестник информационных технологий, механики и оптики. 2022. Т. 22, № 3. С. 456–467. DOI 10.17586/2226-1494-2022-22-3-456-467. URL: Автоматизация проверки на плагиат: новый подход к анализу кода в цифровой образовательной платформа Мирера | Кадина | Труды НИИСИ
9. Карманова, Е. В. Автоматизированный контроль при обучении программированию на Python с использованием плагина CodeRunner LMS MOODLE / Е. В. Карманова // Наука, информатизация, технологии, образование: материалы XIV Международной научно-практической конференции, Екатеринбург, 01–05 марта 2021 г. Екатеринбург: Российский государственный профессионально-педагогический университет, 2021. С. 102–108. EDN TEDZUF. URL: Электронная библиотека УрГПУ: Автоматизированный контроль при обучении программированию на PYTHON с использованием плагина CODERUNNER LMS MOODLE
10. Кондратьева В. А. Обучение основам программирования на языке Python в школьном курсе информатики // Вестник Московского городского педагогического университета. Серия: Информатика и информатизация образования. 2021. № 1 (55). С. 8–16. DOI 10.25688/2072-9014.2021.55.1.01. URL: ОБУЧЕНИЕ ОСНОВАМ ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ PYTHON В ШКОЛЬНОМ КУРСЕ ИНФОРМАТИКИ
11. Методические рекомендации по преподаванию информатики в условиях цифровой трансформации / Под ред. Минобрнауки России. 68 с. URL: Методические_рекомендации_информатика_2025.pdf
12. Осин, А. В. Мультимедиа в образовании: контекст информатизации / А. В. Осин. – Москва: Агентство «Росвузнаука», 2005. – 216 с. ISBN 5-89686-060-9. URL: https://rusneb.ru/catalog/000199_000009_002439938/

13. Основы алгоритмизации и программирования: учебник для студ. учреждений сред. проф. образования / И. Г. Семакин, А. П. Шестаков. 3-е изд., стер. Москва: Издательский центр «Академия», 2012. 400 с. ISBN 978-5-7695-8957-7. URL: fragment_20547.pdf
14. Перязева, Ю. В. Возможности автоматической проверки заданий в LMS Moodle / Ю. В. Перязева // Современные информационные технологии и ИТ-образование. 2019. Т. 15, № 4. С. 980–987. URL: Возможности автоматической проверки заданий в LMS Moodle
15. Петухова Н.В. «Организация автоматизированного контроля знаний по программированию в профильной школе» // Информатика в школе. 2024. № 2. С. 15–23.
16. Примерная основная образовательная программа среднего общего образования: одобрена решением федерального учебно-методического объединения по общему образованию, протокол от 28.06.2016 № 2/16-з.
URL: *"Примерная основная образовательная программа среднего общего образования" (одобрена решением федерального учебно-методического объединения по общему образованию, протокол от 28.06.2016 N 2/16-з) {КонсультантПлюс}*
17. Приказ Министерства просвещения РФ от 16.11.2022 № 993 «Об утверждении федерального государственного образовательного стандарта среднего общего образования». М., 2022. URL: Приказ Министерства просвещения Российской Федерации от 16.11.2022 № 993 · Официальное опубликование правовых актов
18. Рубинштейн К.И. «Автоматизированные системы оценки программных решений в учебном процессе» // Вестник МГТУ им. Н.Э. Баумана. Серия: Естественные науки. 2021. № 4. С. 89–104.
19. Саукова, Н. М. Использование систем автоматизированного контроля знаний в профессиональной деятельности педагога: учебно-методическое пособие / Н. М. Саукова, Г. Ю. Соколова, С. А. Моркин; под ред. Н. М. Сауковой. Москва: МПГУ, 2013. – 124 с. URL: Использование систем автоматизированного контроля знаний в профессиональной деятельности педагога Н. М. Саукова, Г. Ю. Соколова, С. А. Моркин (книга), скачать бесплатно, читать онлайн | izbe.ru

20. Семакин, И. Г. Информатика. Программы для общеобразовательных учреждений. 7–9, 10–11 классы / И. Г. Семакин, Е. К. Хеннер. Москва: БИНОМ. Лаборатория знаний, 2013. 120 с. ISBN 978-5-9963-1378-5. URL: https://rusneb.ru/catalog/002178_000020_BGUNB-BELG%7C%7C%7CBIBL%7C%7C%7C0000151054/
21. Семёнов А.Л. «Цифровая дидактика: новые вызовы для системы образования» // Педагогика. 2023. № 5. С. 12–24.
22. Семёнов А.Л., Уваров А.Ю. Цифровая дидактика: теория и практика. М.: Педагогика, 2023. 444 с.
23. Соловьёва Л.Ф. Контроль знаний учащихся по информатике. СПб.: БХВ-Петербург, 2008. 453 с.
24. Талызина Н. Ф. Педагогическая психология: учебник для студ. высш. учеб. заведений / Н. Ф. Талызина. 9-е изд., стер. Москва: Академия, 2013. 288 с. ISBN 978-5-7695-9362-1. URL: Талызина Н. Ф. Педагогическая психология. Практикум — изучать онлайн. «Юрайт»
25. Федеральный государственный образовательный стандарт среднего общего образования: приказ Минпросвещения России от 17.05.2012 № 413 (ред. от 12.08.2022). URL: *Приказ Минобрнауки России от 17.05.2012 N 413 (ред. от 12.02.2025) "Об утверждении федерального государственного образовательного стандарта среднего общего образования" {КонсультантПлюс}*
26. Цифровая дидактика: технологии и сервисы : учебное пособие / Н. В. Соловова, Д. О. Ежков, А. Н. Рылов, Н. В. Суханкина. Самара: Издательство Самарского университета, 2025. – 82 с. ISBN 978-5-7883-2194-3. URL: 978-5-7883-2194-3_2025.pdf
27. Деятельность. Сознание. Личность : учеб. пособие для студентов вузов по направлению и спец. "Психология", "Клин. психология" / А. Н. Леонтьев. Москва: Смысл, Academia, 2004. 345, [1] с.: 22 см (Высшее образование, Классическая учебная книга, Classicus).; ISBN 5-89357-153-3. URL: https://rusneb.ru/catalog/000199_000009_002463700/

28. CodeRunner Project. Official Documentation. University of Canterbury, New Zealand. URL: bing.com/ck/a?!&&p=e950004091fe3d5eaaa9c5b331db28784733e04e31237535fbb872b56fa3a9adJmltdHM9MTc4MTc0MDgwMA&ptn=3&ver=2&hsh=4&fclid=09ce1dc9-2512-6e6f-2297-0930249d6f9c&psq=30.%09CodeRunner+Project.+Official+Documentation.++University+of+Canterbury%2c+New+Zealand.&u=a1aHR0cHM6Ly9jb2RlcnVubmVyLm9yZy5uei8&ntb=1
29. Guo, P. J. Online python tutor: embeddable web-based program visualization for cs education / P. J. Guo // SIGCSE '13: Proceedings of the 44th ACM technical symposium on Computer science education. 2013. P. 579–584. DOI 10.1145/2445196.2445368.
30. Hromčik, M. Automatic Assessment of Programming Exercises: A Systematic Review / M. Hromčik, M. Bielikova // ACM Computing Surveys. 2020. Vol. 53, Issue 4. Article 78. P. 1–35. DOI 10.1145/3397537.
31. Ihantola, P. Review of automatic assessment methods for programming courses / P. Ihantola, T. Kaila, M. Vihavainen // IEEE Transactions on Education. 2010. Vol. 53, № 3. P. 410–419. DOI 10.1109/TE.2009.2032392.
32. Lobb, R. CodeRunner question type / R. Lobb // MoodleDocs. 2023. URL: https://docs.moodle.org/en/CodeRunner_question_type.