

УДК

## РАЗВИТИЕ КРИТИЧЕСКОГО МЫШЛЕНИЯ ЧЕРЕЗ ОБУЧЕНИЕ ПРОГРАММИРОВАНИЮ: КОГНИТИВНЫЙ ФУНДАМЕНТ И ДИДАКТИЧЕСКИЕ СТРАТЕГИИ

*\*Агадилова М.А.<sup>1</sup>, Минулин А.А.<sup>2</sup>*

*<sup>1, 2</sup> Филиал НАО «Республиканская физико-математическая школа» в г. Астана, Астана, Казахстан*

*\*(E-mail: meruet.agadilova@fizmat.kz<sup>1</sup>, arnold.minulin@fizmat.kz<sup>2</sup>)*

### *Аннотация*

В статье рассматривается проблема развития критического мышления в процессе обучения программированию в условиях цифровой трансформации образования. Актуальность исследования обусловлена возрастающей потребностью в подготовке учащихся, способных не только применять алгоритмы и программные средства, но и анализировать информацию, оценивать различные способы решения задач и принимать обоснованные решения. Цель исследования заключается в выявлении влияния когнитивно ориентированных дидактических стратегий на формирование критического мышления школьников. Использованы методы педагогического эксперимента, наблюдения, анкетирования и сравнительного анализа результатов до и после внедрения методики. Теоретическую основу составляют работы по когнитивной нагрузке, вычислительному мышлению и критическому мышлению, раскрывающие особенности организации познавательной деятельности учащихся при изучении программирования. В исследовании приняли участие 20 учащихся 8 класса. В образовательный процесс были внедрены задачи Парсона, сократическое вопрошание, парное программирование и рефакторинг кода, а также элементы использования генеративного искусственного интеллекта в качестве средства когнитивной поддержки. Полученные результаты показали значительное улучшение показателей анализа кода, выявления логических ошибок, аргументации решений и самостоятельности учащихся при решении учебных задач (рост до 24–27 %). Полученные данные подтверждают эффективность предложенных дидактических стратегий в развитии критического мышления и свидетельствуют о положительном влиянии когнитивно ориентированного обучения на формирование метакогнитивных навыков. Подтверждено, что программирование выступает эффективной средой развития критического и метакогнитивного мышления. Выводы свидетельствуют о необходимости перехода от обучения синтаксису к формированию культуры осмысленного программирования, основанной на анализе, рефлексии, аргументации и самостоятельном принятии решений.

*Ключевые слова:* критическое мышление, программирование, когнитивная нагрузка, метакогнитивность, вычислительное мышление, парное программирование, искусственный интеллект.

### **Введение**

Активное внедрение искусственного интеллекта в образовательную, экономическую и социальную сферы требует пересмотра целей школьного образования. Особую актуальность приобретает развитие у учащихся способности самостоятельно анализировать информацию, оценивать последствия принимаемых решений и адаптироваться к быстро меняющимся технологическим условиям. Сегодня недостаточно владеть исключительно техническими навыками программирования; важным становится умение анализировать последствия технологических решений, критически оценивать информацию и принимать обоснованные решения в условиях неопределенности[1].

Программирование постепенно трансформируется из узкопрофессиональной дисциплины в универсальный инструмент интеллектуального развития. Оно выступает своеобразной «ментальной лабораторией», в которой происходит интеграция вычислительного и критического мышления [2]. Работа с кодом требует не только алгоритмической точности, но и способности к анализу, интерпретации, оценке и рефлексии. В связи с этим обучение программированию приобретает особое значение как средство формирования когнитивной культуры учащихся.

### ***Теоретические основания критического мышления***

Проблема определения критического мышления остается дискуссионной в современной педагогике и когнитивной психологии. Одним из наиболее содержательных подходов является концепция когнитивного психолога Дэниела Уиллингема, который выделяет три ключевых признака критического мышления: оригинальность, самонаправленность и эффективность.

В образовательной практике оригинальность проявляется тогда, когда учащийся не ограничивается воспроизведением готового алгоритма, а предлагает собственный способ решения задачи или обоснованно модифицирует существующее решение. Самонаправленность связана с отказом от механического следования инструкциям и предполагает автономность мыслительного процесса. Эффективность означает подчинение рассуждений логическим правилам и процедурам проверки достоверности выводов.

В работах Р. Пола и Л. Элдер критическое мышление рассматривается как процесс анализа и оценки аргументов [3]. П. Фасионе подчёркивает его связь с логическим обоснованием решений [4].

Особое значение имеет феномен «аппетита к когнитивной работе», то есть готовности субъекта включаться в сложную мыслительную деятельность там, где большинство склонно действовать по привычке. В контексте программирования данный феномен проявляется через развитие таких компонентов критического мышления, как интерпретация, анализ, оценка, инференция, объяснение и саморегуляция.

### ***Соотношение вычислительного и критического мышления***

Несмотря на тесную взаимосвязь, вычислительное и критическое мышление представляют собой различные когнитивные конструкции. В рамках школьного курса информатики вычислительное мышление проявляется через способность учащихся разбивать сложную задачу на последовательность более простых действий и представлять решение в алгоритмической форме. Его основная цель заключается в поиске эффективного способа достижения результата.

В рамках обучения программированию критическое мышление проявляется в способности учащегося оценивать корректность выбранного алгоритма, выявлять ограничения решения и прогнозировать возможные результаты его применения. Если вычислительное мышление отвечает на вопрос «как решить задачу?», то критическое мышление ставит вопрос «насколько целесообразно данное решение и каковы его последствия?».

Таким образом, критическое мышление дополняет вычислительное уровнем оценки и этической рефлексии. Особенно ярко это проявляется при анализе алгоритмов искусственного интеллекта, систем автоматизации и цифровых платформ, влияние которых выходит далеко за рамки технической эффективности.

### ***Когнитивные механизмы обучения программированию***

Изучение программирования связано с высокой когнитивной нагрузкой, обусловленной ограниченностью рабочей памяти человека [5]. Возможности рабочей памяти ограничены, поэтому начинающие программисты часто испытывают трудности при одновременном анализе синтаксиса, логики алгоритма и структуры данных.

С точки зрения когнитивной психологии на успешность обучения влияет не только сложность изучаемого материала, но и способ его представления. Часть интеллектуальных ресурсов расходуется непосредственно на решение задачи, а часть — на обработку сопутствующей информации и организацию деятельности [6]. Внутренняя нагрузка определяется сложностью самой задачи. Внешняя нагрузка связана с неэффективной организацией учебного материала, сложным синтаксисом или перегруженным интерфейсом среды программирования. Уместная нагрузка направлена на формирование устойчивых когнитивных схем и способствует обучению.

Одной из ключевых задач преподавателя является минимизация внешней когнитивной нагрузки. Это позволяет учащимся сосредоточить внимание не на механическом запоминании синтаксиса, а на понимании логики решения. Данный подход согласуется с культурно-исторической теорией обучения Л. С. Выготского, согласно которой развитие мышления происходит в процессе организованной педагогической поддержки и постепенного перехода от внешних форм деятельности к внутренним интеллектуальным действиям [7]. Эффективным инструментом в данном контексте выступает техника «chunking», предполагающая объединение отдельных элементов кода в концептуальные блоки. Формирование таких блоков способствует освобождению ресурсов рабочей памяти и создает условия для более глубокого анализа программных структур.

### ***Дидактические стратегии развития критического мышления***

#### ***1. Задачи Парсона***

В ходе эксперимента наибольший интерес учащихся вызвали задачи Парсона, предполагающие восстановление логики программы из набора перемешанных строк кода. Данный подход позволяет снизить влияние синтаксических трудностей и сосредоточить внимание на семантической структуре алгоритма.

Особую роль играют дистракторы — лишние или ошибочные строки кода. Их использование стимулирует сравнительный анализ альтернативных решений и способствует развитию навыков критической оценки.

#### ***2. Сократическое вопрошание и использование искусственного интеллекта***

Существенным инструментом формирования критического мышления выступает сократическое вопрошание. Развитие критического мышления достигается через вопросы, стимулирующие анализ и оценку решений.

Вместо прямого предоставления готового решения преподаватель организует процесс через систему вопросов, побуждающих учащегося к самостоятельному анализу. Такой подход соответствует идее зоны ближайшего развития, в рамках которой педагог создает условия для перехода учащегося на более высокий уровень познавательной деятельности.

Например, вопросы типа «Каковы преимущества и ограничения использования массива в данной задаче?» развивают навыки оценки, а вопросы «Что произойдет при передаче пустого значения?» способствуют формированию инференции и прогнозирования поведения программы.

В условиях распространения генеративного искусственного интеллекта особую актуальность приобретает проблема «когнитивного обхода», при котором учащийся делегирует ИИ весь процесс решения задачи. В образовательном процессе искусственный интеллект должен выполнять функцию когнитивных «лесов» (scaffolding), поддерживая процесс понимания и отладки, но не заменяя самостоятельную интеллектуальную деятельность учащегося.

#### ***3. Парное программирование***

Парное программирование способствует вербализации мышления и развитию аргументации [8]. Представляет собой форму совместной деятельности, в которой один участник выполняет роль «водителя», непосредственно работающего с кодом, а другой — роль «штурмана», отвечающего за стратегическое планирование и контроль.

По наблюдениям авторов, совместная работа особенно эффективно проявлялась при поиске логических ошибок, когда учащиеся аргументировали выбор различных вариантов реализации алгоритма. Подобная организация взаимодействия способствует вербализации мыслительных процессов и формированию когнитивного конфликта. Решения одного участника немедленно подвергаются критическому анализу со стороны другого, что снижает вероятность хаотичного программирования методом проб и ошибок.

#### *4. Рефакторинг и когнитивная сложность*

Современные образовательные технологии всё чаще ориентируются на снижение когнитивной сложности программного кода. В частности, системы рефакторинга, основанные на когнитивных принципах, помогают учащимся улучшать структуру программ, уменьшать цикломатическую сложность и повышать читаемость кода [9, 10].

Исследования показывают, что подобные подходы способны повысить понимание функциональной структуры программы более чем на 30 %, что свидетельствует о значительном потенциале когнитивно ориентированного обучения программированию.

#### *5. Эффект «дальнего переноса» навыков*

Одним из наиболее значимых вопросов является проблема переноса навыков, сформированных в процессе программирования, на другие виды интеллектуальной деятельности. Результаты метаанализа, проведенного Шерером и соавторами в 2019 году, подтверждают наличие умеренного эффекта переноса. Исследования показывают умеренный эффект переноса навыков программирования на другие когнитивные области [11].

Наиболее выраженное влияние программирование оказывает на развитие креативного мышления, математических способностей, пространственного воображения и метакогнитивных навыков. Учащиеся, систематически занимающиеся программированием, демонстрируют более высокий уровень способности к планированию, мониторингу и саморегуляции собственной деятельности.

В то же время влияние программирования на общую грамотность и читательские способности оказывается менее значительным, что указывает на специфику формируемых когнитивных механизмов.

#### *6. Проблемы инклюзии и системные барьеры*

Несмотря на высокий образовательный потенциал программирования, интеграция критического мышления в учебный процесс сталкивается с рядом трудностей. Многие учащиеся предпочитают задачи с заранее известным алгоритмом решения и испытывают дискомфорт в ситуациях неопределенности.

Особую проблему представляет так называемая «зона средней сложности», в которой успешность решения начинает зависеть не столько от усилий учащегося, сколько от объема его рабочей памяти. Для преодоления данной проблемы необходимо раннее обучение стратегиям декомпозиции и структурирования информации.

Дополнительным вызовом является цифровое неравенство. Исследования показывают, что результаты тестов на вычислительное мышление коррелируют с социально-экономическим статусом семьи и уровнем культурного капитала. В этих условиях задача педагога заключается в создании инклюзивной образовательной среды, обеспечивающей доступ к высокоуровневому мышлению для всех категорий учащихся.

Важную роль здесь могут играть «бескомпьютерные» методы обучения (unplugged activities), позволяющие развивать алгоритмическое и критическое мышление без необходимости постоянного доступа к цифровым устройствам.

### **Методология исследования**

Для проверки теоретических положений было проведено педагогическое исследование на базе филиала НАО «РФМШ» в г. Астана. В исследовании приняли участие 20 учащихся 8 класса, изучающих программирование на языке Python в рамках предмета «Информатика». Выбор данной возрастной группы обусловлен тем, что учащиеся уже обладают базовыми навыками алгоритмизации и способны выполнять задания, требующие анализа и оценки различных способов решения.

Исследование проводилось по схеме «до внедрения методики – после внедрения методики» в течение одного учебного семестра.

На первом этапе была проведена первичная диагностика уровня развития критического мышления учащихся. Диагностика включала задания на анализ программного кода,

выявление логических ошибок, сравнение альтернативных алгоритмов и обоснование выбора наиболее эффективного решения.

На втором этапе в образовательный процесс были внедрены дидактические стратегии, рассмотренные в теоретической части исследования:

- задачи Парсона для формирования навыков анализа структуры алгоритма;
- сократическое вопрошание при разборе программных решений;
- элементы парного программирования;
- рефакторинг программного кода;
- использование генеративного искусственного интеллекта в качестве инструмента

поддержки поиска решений и отладки программ.

Особое внимание уделялось не получению правильного ответа, а аргументации выбора решения, анализу альтернативных вариантов и рефлексии над процессом решения задачи.

На завершающем этапе была проведена повторная диагностика по тем же критериям.

Для сбора и анализа данных использовались следующие методы:

- педагогическое наблюдение;
- анализ учебной деятельности учащихся;
- выполнение диагностических заданий;
- анкетирование;
- сравнительный анализ результатов до и после внедрения методики.

В качестве показателей развития критического мышления были определены:

1. способность анализировать программные конструкции;
2. умение выявлять ошибки и противоречия;
3. способность сравнивать альтернативные решения;
4. уровень аргументации собственных выводов;
5. степень самостоятельности при решении задач.

Диагностика включала 10 комплексных практических заданий, каждое из которых оценивалось по 100-балльной шкале на основе критериев правильности логики, полноты анализа и точности аргументации. Значения в таблице (например, 57 % по показателю «Анализ программного кода») отражают средний процент правильно выполненных заданий и элементов анализа среди всех учащихся класса, а не долю успешно справившихся учеников. Данный формат оценки позволил точно зафиксировать глубину понимания программного материала и оценить качественные изменения когнитивных навыков до и после внедрения методики.

### **Результаты и их обсуждение**

Проведенное исследование позволило выявить положительную динамику в развитии критического мышления учащихся после применения когнитивно ориентированных методов обучения программированию.

Первичная диагностика показала, что большинство учащихся успешно воспроизводили изученные конструкции языка программирования, однако испытывали затруднения при необходимости анализа альтернативных решений и объяснения причин выбора конкретного алгоритма.

Наиболее распространенными затруднениями являлись:

- ориентация исключительно на получение правильного результата;
- недостаточное внимание к эффективности алгоритма;
- затруднения при поиске причин ошибок;
- отсутствие аргументированного обоснования принимаемых решений.

После внедрения описанных дидактических стратегий наблюдались существенные изменения в характере учебной деятельности учащихся. Динамика показателей изменения критического мышления до и после внедрения представлены в таблице 1.

Таблица 1. Динамика показателей критического мышления учащихся (%)

| Показатель                      | До внедрения методики | После внедрения методики |
|---------------------------------|-----------------------|--------------------------|
| Анализ программного кода        | 57 %                  | 81 %                     |
| Выявление ошибок и противоречий | 60 %                  | 84 %                     |
| Поиск альтернативных решений    | 52 %                  | 79 %                     |
| Аргументация выводов            | 55 %                  | 82 %                     |
| Самостоятельность решения задач | 58 %                  | 85 %                     |

Полученные результаты свидетельствуют о положительном влиянии используемых методик на развитие компонентов критического мышления и согласуются с выводами о влиянии программирования на когнитивное развитие. Полученный средний прирост показателей критического мышления (25,8%) и метакогнитивных навыков хорошо согласуется с данными метаанализа R. Scherer et al. (2019), зафиксировавшего значимый эффект влияния обучения программированию на развитие высших когнитивных функций. Как и в выводах исследователей, в нашем эксперименте наибольшая динамика проявилась в компонентах самостоятельности, планирования и саморегуляции (рост до 85%). Это подтверждает, что когнитивно ориентированные дидактические стратегии эффективно стимулируют средний и высокий уровень переноса навыков аналитического мышления.

Наиболее значительные изменения были зафиксированы по показателям самостоятельности решения задач и выявления ошибок в программном коде. По наблюдениям авторов, учащиеся стали чаще использовать стратегии самопроверки и рефлексии до обращения за помощью к учителю.

Особенно заметный эффект продемонстрировало использование задач Парсона. Работа с перемешанными фрагментами кода позволяла учащимся концентрироваться на логике алгоритма, не отвлекаясь на синтаксические детали. В процессе обсуждения решений школьники активно сравнивали различные варианты организации программы и аргументировали свой выбор.

Положительное влияние оказало и применение сократического вопрошания. Вместо прямых подсказок учащимся предлагались вопросы, побуждающие самостоятельно анализировать последствия принимаемых решений. Это способствовало развитию навыков прогнозирования и оценки эффективности алгоритмов.

Интересные результаты были получены при использовании элементов парного программирования. В ходе совместной работы учащиеся чаще проговаривали собственные рассуждения, критически оценивали идеи партнера и предлагали альтернативные способы решения поставленной задачи. Подобное взаимодействие способствовало развитию рефлексии и аргументации.

В ходе педагогических наблюдений отмечалось, что использование элементов парного программирования способствовало усилению когнитивного взаимодействия учащихся. В процессе совместной работы школьники чаще вербализовали собственные рассуждения, критически оценивали идеи партнера и предлагали альтернативные способы решения поставленной задачи. Кроме того, регулярное выполнение заданий на рефакторинг привело к улучшению структурного понимания кода: учащиеся стали более осознанно подходить к оптимизации алгоритмов и повышению их читаемости.

Отдельного внимания заслуживает использование инструментов искусственного интеллекта. Первоначально часть учащихся стремилась использовать ИИ для получения готовых решений. Однако после внедрения практики анализа и обсуждения предложенных вариантов произошло смещение акцента с копирования ответов на их критическую оценку. Учащиеся стали чаще задавать уточняющие вопросы, проверять корректность предложенного кода и анализировать ограничения полученных решений.

Результаты анкетирования показали, что:

- 86 % учащихся отметили повышение уверенности при решении новых задач;
- 82 % стали чаще анализировать несколько вариантов решения;
- 79 % сообщили о более осознанном отношении к поиску и исправлению ошибок;

– 75 % отметили, что научились критически оценивать ответы, полученные от систем искусственного интеллекта.

Полученные результаты подтверждают положения когнитивной психологии о том, что развитие критического мышления происходит наиболее эффективно в условиях активной интеллектуальной деятельности, предполагающей анализ, оценку и рефлексию.

Исследование также показало, что формирование критического мышления в процессе обучения программированию связано не столько с изучением новых языковых конструкций, сколько с организацией учебной деятельности, стимулирующей осмысленное принятие решений и аргументированное обсуждение различных подходов к решению задач.

К ограничениям исследования следует отнести небольшой объем выборки и проведение эксперимента в рамках одной образовательной организации. В дальнейшем представляется целесообразным расширение выборки и проведение сравнительных исследований в различных типах школ.

### **Заключение**

Развитие критического мышления через обучение программированию представляет собой переход от обучения синтаксису к формированию культуры осмысленного мышления посредством кода. Программирование становится не только инструментом технической подготовки, но и средством развития интеллектуальной автономии, рефлексии и способности к анализу сложных систем.

Эффективное формирование критического мышления требует системного подхода, включающего определение микронавыков мышления, подбор содержательного материала, постепенное усложнение задач и регулярное возвращение к ключевым концепциям. Особое значение приобретает организация учебного процесса таким образом, чтобы учащийся не воспроизводил готовые решения, а осмысленно конструировал собственные стратегии мышления.

Только в этом случае программирование сможет выполнять свою подлинную образовательную функцию — формировать специалистов, способных не только создавать технологии, но и осознанно управлять их влиянием на общество и будущее человечества.

### **СПИСОК ЛИТЕРАТУРЫ**

- 1 Willingham, D. T. *Why Don't Students Like School? A Cognitive Scientist Answers Questions About How the Mind Works and What It Means for the Classroom.* — 2nd ed. — San Francisco: Jossey-Bass, 2021. — 320 p.
- 2 Wing, J. M. Computational Thinking // *Communications of the ACM.* — 2006. — Vol. 49, No. 3. — P. 33–35.
- 3 Paul, R., Elder, L. *Critical thinking: Tools for taking charge of your learning and your life.* — New York: Pearson, 2020. — 348 p.
- 4 Facione, P. A. *Critical Thinking: What It Is and Why It Counts.* — Millbrae, CA: Insight Assessment, 2015. — 30 p.
- 5 Sweller, J. *Cognitive Load Theory.* — New York: Springer, 2011. — 274 p.
- 6 Sweller, J., Ayres, P., Kalyuga, S. *Cognitive Load Theory.* — New York: Springer, 2019. — 402 p.
- 7 Выготский, Л. С. *Педагогическая психология.* — М.: Юрайт, 2023. — 510 с.
- 8 Williams, L., Kessler, R. *Pair Programming Illuminated.* — Boston: Addison-Wesley, 2003. — 224 p.
- 9 Martin, R. C. *Clean Code: A Handbook of Agile Software Craftsmanship.* — Boston: Prentice Hall, 2008. — 464 p.
- 10 McCabe, T. J. A Complexity Measure // *IEEE Transactions on Software Engineering.* — 1976. — Vol. 2, No. 4. — P. 308–320.

11 Scherer, R., Siddiq, F., Sánchez Viveros, B. The Cognitive Benefits of Learning Computer Programming: A Meta-Analysis of Transfer Effects // Journal of Educational Psychology. — 2019. — Vol. 111, No. 5. — P. 764–792.

## REFERENCES

- 1 Willingham, D. T. Why Don't Students Like School? A Cognitive Scientist Answers Questions About How the Mind Works and What It Means for the Classroom. — 2nd ed. — San Francisco: Jossey-Bass, 2021. — 320 p.
- 2 Wing, J. M. Computational Thinking // Communications of the ACM. — 2006. — Vol. 49, No. 3. — P. 33–35.
- 3 Paul, R., Elder, L. Critical thinking: Tools for taking charge of your learning and your life. — New York: Pearson, 2020. — 348 p.
- 4 Facione, P. A. Critical Thinking: What It Is and Why It Counts. — Millbrae, CA: Insight Assessment, 2015. — 30 p.
- 5 Sweller, J. Cognitive Load Theory. — New York: Springer, 2011. — 274 p.
- 6 Sweller, J., Ayres, P., Kalyuga, S. Cognitive Load Theory. — New York: Springer, 2019. — 402 p.
- 7 Vygotsky, L. S. Pedagogical Psychology. Moscow: Yurait, 2023. — 510 p.
- 8 Williams, L., Kessler, R. Pair Programming Illuminated. — Boston: Addison-Wesley, 2003. — 224 p.
- 9 Martin, R. C. Clean Code: A Handbook of Agile Software Craftsmanship. — Boston: Prentice Hall, 2008. — 464 p.
- 10 McCabe, T. J. A Complexity Measure // IEEE Transactions on Software Engineering. — 1976. — Vol. 2, No. 4. — P. 308–320.
- 11 Scherer, R., Siddiq, F., Sánchez Viveros, B. The Cognitive Benefits of Learning Computer Programming: A Meta-Analysis of Transfer Effects // Journal of Educational Psychology. — 2019. — Vol. 111, No. 5. — P. 764–792.

## БАҒДАРЛАМАЛАУ АРҚЫЛЫ СЫНИ ОЙЛАУДЫ ДАМУ: КОГНИТИВТІК НЕГІЗДЕР МЕН ДИДАКТИКАЛЫҚ СТРАТЕГИЯЛАР

*Агадилова М.А.<sup>1</sup>, Минулин А.А.<sup>2</sup>*

*<sup>1, 2</sup> Астана қ. «Республикалық физика-математика мектебі» КеАҚ филиалы, Астана, Қазақстан*

### *Аңдатпа*

Мақалада білім беруді цифрлық трансформациялау жағдайында бағдарламалауды оқыту үдерісінде оқушылардың сыни ойлауын дамыту мәселесі қарастырылады. Зерттеудің өзектілігі алгоритмдер мен бағдарламалық құралдарды қолдана алатын ғана емес, сонымен қатар ақпаратты талдап, есептерді шешудің әртүрлі тәсілдерін бағалай алатын және негізделген шешімдер қабылдайтын оқушыларды даярлау қажеттілігінің артуымен айқындалады. Зерттеудің мақсаты – когнитивтік бағытталған дидактикалық стратегиялардың мектеп оқушыларының сыни ойлауын қалыптастыруға ықпалын анықтау. Зерттеуде педагогикалық эксперимент, бақылау, сауалнама жүргізу және әдістемені енгізгенге дейінгі және енгізгеннен кейінгі нәтижелерді салыстырмалы талдау әдістері қолданылды. Теориялық негізін когнитивтік жүктеме, есептеулік ойлау және сыни ойлау салаларындағы, бағдарламалауды оқыту барысында оқушылардың танымдық іс-әрекетін ұйымдастыру ерекшеліктерін ашып көрсететін ғылыми еңбектер құрайды. Зерттеуге 8-сыныптың 20 оқушысы қатысты. Оқу үдерісіне Парсон тапсырмалары, Сократтық сұрақ қою әдісі, жұптық бағдарламалау және кодты рефакторингтеу, сондай-ақ генеративті жасанды интеллектіні когнитивтік қолдау құралы ретінде пайдалану элементтері енгізілді. Алынған нәтижелер кодты талдау, логикалық қателерді анықтау, шешімдерді дәлелдеу және оқу тапсырмаларын орындаудағы оқушылардың дербестік көрсеткіштерінің айтарлықтай жақсарғанын көрсетті (өсім 24–27 %). Алынған деректер ұсынылған дидактикалық стратегиялардың сыни ойлауды дамытудағы тиімділігін және когнитивтік бағытталған оқытудың метатанымдық дағдыларды қалыптастыруға оң әсерін дәлелдейді. Бағдарламалау сыни және метатанымдық ойлауды дамытудың тиімді ортасы екендігі расталды. Зерттеу қорытындылары бағдарламалау тілінің синтаксисін меңгертуден талдауға,

рефлексияға, дәлелдеуге және өз бетінше шешім қабылдауға негізделген саналы бағдарламалау мәдениетін қалыптастыруға көшудің қажеттілігін көрсетеді.

*Түйін сөздер:* сыни ойлау, бағдарламалау, когнитивтік жүктеме, метатаным, есептік ойлау, жұптық бағдарламалау, жасанды интеллект.

## DEVELOPING CRITICAL THINKING THROUGH PROGRAMMING INSTRUCTION: COGNITIVE FOUNDATIONS AND DIDACTIC STRATEGIES

*Agadilova M.A.<sup>1</sup>, Minulin A.A.<sup>2</sup>*

*<sup>1, 2</sup> National School of Physics and Math in Astana, Astana, Kazakhstan*

### Annotation

The article examines the development of critical thinking through programming education in the context of the digital transformation of education. The relevance of the study is determined by the growing need to prepare students who are able not only to apply algorithms and programming tools but also to analyze information, evaluate alternative problem-solving approaches, and make well-reasoned decisions. The aim of the study is to investigate the impact of cognitively oriented instructional strategies on the development of school students' critical thinking. The research employed a pedagogical experiment, classroom observation, questionnaires, and a comparative analysis of the results obtained before and after the implementation of the proposed methodology. The theoretical framework is based on studies of cognitive load theory, computational thinking, and critical thinking, which explain the organization of students' cognitive activity in programming education. The study involved 20 eighth-grade students. The instructional process incorporated Parsons problems, Socratic questioning, pair programming, code refactoring, and the use of generative artificial intelligence as a cognitive scaffolding tool. The findings demonstrated a significant improvement in students' ability to analyze code, identify logical errors, justify their solutions, and solve programming tasks independently, with performance gains ranging from 24% to 27%. The obtained results confirm the effectiveness of the proposed instructional strategies in fostering critical thinking and indicate the positive impact of cognitively oriented programming instruction on the development of metacognitive skills. The study confirms that programming provides an effective environment for developing both critical and metacognitive thinking. The findings highlight the need to move beyond teaching programming syntax toward fostering a culture of mindful programming based on analysis, reflection, evidence-based reasoning, and independent decision-making.

*Keywords:* critical thinking, programming education, cognitive load theory, metacognition, computational thinking, pair programming, artificial intelligence.

## АВТОРЛАР ТУРАЛЫ МӘЛІМЕТ / СВЕДЕНИЯ ОБ АВТОРАХ / INFORMATION ABOUT AUTHORS

**Агадилова Мерует Абибековна** – педагогика магистрі, информатика пәні мұғалімі, Астана қ. «Республикалық физика-математика мектебі» КеАҚ филиалы, Астана, Қазақстан, meruet.agadilova@fizmat.kz

**Агадилова Мерует Абибековна** – магистр педагогика, учитель информатики, филиал НАО «Республиканская физико-математическая школа» в г. Астана, Астана, Казахстан, meruet.agadilova@fizmat.kz

**Agadilova Meruyet Abibekovna** – Master of Pedagogy, Computer Science Teacher, National School of Physics and Math in Astana, Astana, Kazakhstan, meruet.agadilova@fizmat.kz

**Минулин Арнольд Альбертович** – педагогика магистрі информатика пәні мұғалімі, Астана қ. «Республикалық физика-математика мектебі» КеАҚ филиалы, Астана, Қазақстан, arnold.minulin@fizmat.kz

**Минулин Арнольд Альбертович** – магистр педагогика, учитель информатики, филиал НАО «Республиканская физико-математическая школа» в г. Астана, Астана, Казахстан, arnold.minulin@fizmat.kz

**Minulin Arnold Albertovich** – Master of Pedagogy, Computer Science Teacher, National School of Physics and Math in Astana, Astana, Kazakhstan, arnold.minulin@fizmat.kz