24. *Neuman, W.* Social research methods : Qualitative and quantitative approaches / W. Neuman. — Pearson : Pearson Education Limited, 2014. — 640 p.

25. *Shahmohammadi, N.* Content analysis of elementary science text books based on the achievement motivation constructs / N. Shahmohammadi // Procedia — Social and Behavioral Sciences. — 2013. — № 84. — P. 426–430.

26. Woven Words. Textbook in English for Class XI (Elective Course) / National Council of Educational Research and Training. — URL: https://www.amazon.com/Woven-Words-Textbook-English-Class/dp/8174505148 (date of the application: 09.04.2024).

УДК 159.99,37.025
DOI: 10.51944/20738528_2024_2_268
EDN: VZSCXN

А. В. Попова, Ю. А. Тюменева

Новый метод оценки алгоритмического мышления на непрограммистских задачах: разведывательное исследование[1]

A. V. Popova, Yu. A. Tyumeneva

New Method for Assessing Algorithmic Thinking in Non-Programmer Tasks: Exploratory Study[2]

Алгоритмическое мышление в основном оценивается через стандартизированные тесты и профессиональные задачи, что сужает возможности выявления специфики алгоритмизации решения у людей с разным опытом программирования. С целью уточнить гипотезу о такой специфике мы предложили альтернативный метод оценки, требующий алгоритмизации, но не специальных знаний в программировании. По протоколам решений разработанной нами задачи выделены параметры качества алгоритмизации, допускающие надежную кодировку. Результаты показали, что предложенный метод дает требуемую вариативность решений и позволяет создавать новые варианты задач на основе выделенных параметров. Также уточнена гипотеза о специфике алгоритмизации у людей с разным опытом программирования, хотя ее проверка требует специальных исследований.

Ключевые слова: решение задач, программирование, алгоритмическое мышление, инструкция.

Подходы к оценке мышления программистов

Программистская деятельность заключается в создании программ — алгоритмических решений разнообразных задач. Согласно профес-

[1] Исследование реализовано при поддержке факультета социальных наук Национального исследовательского университета «Высшая школа экономики».

[2] The research was conducted with the support of the Faculty of Social Sciences at the National Research University Higher School of Economics.

сиональному стандарту [4] деятельность программиста направлена на разработку и отладку кода, включая формализацию и алгоритмизацию поставленных задач: написание программного кода, его проверку и отладку. Поэтому работа с задачей: анализом условий, структурированием имеющихся данных, пониманием ограничений, оценкой качества решения и выбора оптимального решения — напрямую касается профессиональной компетенции программиста. Неудивительно поэтому, что во множестве исследований оценивались эффекты обучения программированию в отношении самых разных видов когнитивных способностей, имеющих отношение к решению задач: например, дивергентного мышления [8], метакогнитивных способностей [7], способности разрешать проблемы [13; 20], навыков рассуждения [14; 20], креативности [9], интеллекта [2].

Если же ставится задача исследовать не эффекты обучения программированию, а специфические особенности решения задач программистами, то в литературе выделяются две методические возможности для такого исследования: во-первых, стандартизированные тесты, во-вторых, тестовые «программистские» задачи, когда, например, предлагается написать какой-то код или найти в готовом решении ошибки. Однако ни один подход не позволяет выявить искомую специфику решения задач. Так, при стандартизированном тестировании исследователи имеют дело с некоторым результатом когнитивной деятельности, выраженным числом правильно решенных тестовых заданий. Хотя такой подход позволяет сравнивать *результаты* решения задачи программистами, сам процесс размышления, связывающий постановку задачи с итоговым ответом, остается скрытым.

В отличие от стандартизированных тестов, использование программистских задач позволяет оценить изменения в подходах к их решению и специфике размышления. При этом такого рода задача может быть поставлена только людям, которые уже имеют профессиональную подготовку, и, таким образом, не дает возможности описать специфику алгоритмического мышления программистов в сравнении с непрограммистами. Поэтому при всей полезности исследований с использованием профессиональных задач вопрос о методе сравнения алгоритмического мышления программистов остается открытым.

Основная цель данной работы — разработать надежный метод оценки специфики алгоритмического мышления программистов, достаточно чувствительный к возможным индивидуальным вариантам этой специфики. Дополнительная цель — на основе полученных результатов описать характерные черты алгоритмического мышления программистов в виде гипотезы для ее последующей проверки.

Что известно о мышлении программистов?

Основной подход к исследованию программистского мышления, как уже было указано, — определить отличия в решении программистских задач у людей с разным уровнем профессиональной подготовки.

В области программирования традиционные методы сбора данных, такие как анкетирование и интервью, долгое время были наиболее распространенными среди исследователей. Тем не менее в последние годы наблюдается увеличенный интерес к новым методам сбора данных о когнитивных процессах, в частности отслеживании движений глаз (Eye Tracking). Этот подход становится все более популярным среди исследователей в программировании, позволяя анализировать и понимать разнообразные задачи, такие как процессы восприятия и отладки кода [6; 12; 19]. Данные отслеживания движений глаз могут служить уникальной метрикой для сравнения различий между экспертами и новичками, обеспечивая понимание процессов внимания [11; 23].

В результате использования отслеживания движений глаз при исправлении ошибок в коде было установлено, что эксперты и новички различаются в поведении при чтении кода: эксперты фокусируются на меньшем количестве элементов кода, а новички более детально читают методы, переменные и ключевые слова — увеличение опыта программирования только усиливает эти различия [6].

В другом исследовании были продемонстрированы похожие результаты, а именно: эксперты и новички различаются в естественном поведении при отладке кода, например, эксперты делают более короткие фиксации и меньше переходов между кодом и выводом [15].

Другим методом изучения когнитивных особенностей с точки зрения нейронауки выступает изучение электроэнцефалограммы. С использованием программистских задач было установлено, что эксперты, по сравнению с новичками, обладают более высоким уровнем понимания кода и, следовательно, выделяются в цифровом кодировании, кратковременной памяти (решении простых программ за короткое время) и эффекте последующей памяти (способности запоминать функции программ после продолжительного времени) [10].

Более традиционными опросными и поведенческими методами было установлено, что при анализе условий программистских задач программисты-эксперты сначала вычленяют алгоритм ее решения, а новички анализируют специфические особенности, указанные в условии [24]; программисты-эксперты, по сравнению с новичками, используют более широкий диапазон метакогнитивных стратегий в ходе решения программистских задач [1]; у экспертов и новичков в программировании различаются подходы к организации и пониманию написанного кода и его запоминание. Эксперты способны вспомнить значительные объемы программы, даже когда общий смысл программы нарушен [5; 16; 25], эксперты запоминают короткие исходные коды в виде смысловых единиц, в то время как начинающие программисты запоминают их построчно [22]. Интересно и то, что эксперты сохраняют преимущество даже при воспоминании случайного материала; этот эффект навыка не ограничивается одной конкретной областью [21]. Кроме того, у экспертов лучше абстрактное понимание кода по сравнению с новичками [26], большинство начинающих программистов не осознают процесс

решения задач, используемый для написания кода на языке программирования. Утверждается, что этот навык, называемый метакогнитивной осведомленностью, крайне важен для обучения новичков [17].

Также на материале профессиональных программистских задач было показано, что для новичков в программировании одной из главных проблем является не только сложность синтаксиса языка программирования, но и разрыв между интуитивным способом мышления и способом мышления, необходимым для программирования. Для последнего необходимы четкие шаги решения и охват всех возможных сценариев для обработки команд, что требует нового способа мышления от новичков. Норман и Дрейпер [18] выдвинули предложения по устранению разрыва между разными способами мышления: приблизить пользователя к системе/среде программирования или приблизить систему/среду программирования к пользователю.

Уточнение гипотезы о специфике решения задач программистами

Упомянутые выше исследования процесса и способов решения задач программистами преимущественно ограничивались задачами профессиональными: написанием, оптимизацией или исправлением имеющихся программ на языке программирования, то есть задачами, написанными и решаемыми с помощью кода. Однако реальная деятельность программиста осуществляется в пространстве между обыденным языком поставленной задачи и абстрактным языком кода программы и всегда включает перевод задачи с обыденного языка на абстрактный. Это означает, что в качестве обязательной части своей деятельности программист вынужден репрезентировать возможные решения изначальной задачи как последовательность действий, которую нужно отразить в коде. Иными словами, для программистской деятельности и для обучения программированию центральными компонентами являются анализ поставленной задачи и алгоритмизация решения. Поэтому мы полагаем, что опыт программирования должен сказываться на способах анализа задачи и алгоритмизации, причем это должно касаться любых задач, не только программистских.

Таким образом, мы уточняем гипотезу о специфике мышления программистов: у программистов решение задачи должно выражаться в детализации шагов, их прописывании и последовательности, экспликации условий перехода между шагами, а также в учете максимально возможного числа сценариев развития событий.

Написание инструкции как метод изучения алгоритмизации решения

Когда оценивается алгоритмическое мышление у программистов, то написанный код является главным предметом оценки, поскольку он в сжатом виде и есть такой алгоритм. Что может стать предметом оценки у непрограммистов, если написание кода им недоступно? Что

вместо кода может репрезентировать алгоритмизацию решения и сделать итоговый алгоритм доступным для внешней оценки? Мы предполагаем, что есть один тип задач, который невозможно решить без алгоритмизации. Это задача написания инструкции. Инструкция (или руководство) представляет собой свод правил, порядок и способ действий, необходимых и достаточных для решения строго определенной задачи. По сути, инструкция — это и есть алгоритмизированное решение задачи или проблемы.

Важно, что написание инструкции, в отличие от написания кода, не ставит программистов в более благоприятные условия по сравнению с непрограммистами.

Мы исходили из того, что навыки алгоритмизации, обычно требующиеся в деятельности программиста для написания кода, структурно схожи с теми, что необходимы для написания инструкции. В случае программирования написанный код может рассматриваться как аналог инструкции, выданной не человеку, а машине и, соответственно, написанной не на обыденном языке, а на языке программирования. Написание инструкции, так же как и написание кода, подразумевает: учет того, что уже известно исполнителю, декомпозицию большой задачи на ряд подзадач (шагов или этапов), выстраивание последовательности шагов, выбор оптимальной последовательности и пр. Таким образом, задание составить инструкцию предположительно оставляло возможность респондентам с разным опытом и без опыта программирования приблизиться к алгоритмизации решения. И это позволило бы анализировать итоговые решения-инструкции как алгоритмы.

Следующим вопросом стал вопрос о содержании самой задачи. Она должна была подходить для индивидуального и группового предъявления (в том числе онлайн), ограниченного времени исследования, быть доступной для решения всеми участниками, а не только программистами. Задача должна была допускать несколько вариантов решения, разные комбинации элементов, разную детализацию шагов решения, а также позволять делать ошибки, например пропускать шаги, не доводить до конца решения и пр.

В итоге разработанная нами задача включала описание обыденной проблемной ситуации с действующими персонажами, которых нужно было проинструктировать, как разрешить стоящую перед ними проблему. Таким образом, инструкция, написание которой требовалось от участников нашего исследования, была инструкцией для персонажей вымышленной истории. Сама инструкция и была алгоритмом решения задачи.

Итоговые инструкции-решения оценивались по тому, насколько они поддаются формализованной и унифицированной оценке их алгоритмичности: в них можно было выделить параметры, общие для всех инструкций, и оценки по этим параметрам варьировали по выборке. Вариации в оценке параметров могли быть связаны в том числе и с разным программистским опытом (разнообразие которого обеспечивалось составом выборки).

Подчеркнем, что фокусом работы являлись достижение вариативности инструкций и их надежная оценка. На этом этапе у нас не было задачи провести количественные сравнения, например, программистов с непрограммистами. Поскольку исследование направлено на разработку и оценку метода, переход к любым оценкам выборки был бы методологической ошибкой. Единственное, что мы сделали в этом направлении, — описали наиболее вероятные параметры инструкций, по которым можно ожидать различий между группами людей с разной подготовкой в области программирования. Однако к этим результатам стоит относиться именно как к предварительным.

Разработка задачи

В качестве задачи, для решения которой требовалось написать инструкцию, использовалась слабоструктурированная[1] задача с обыденным контекстом, не имеющая одного верного решения. Стимульный материал был сформулирован с использованием международного исследования Programme for International Student Assessment (PISA). В частности, была использована третья часть задания «Визит» [3]. На ее основе мы создали комплексную слабоструктурированную задачу:

«Никита, Рита и Борис — близкие друзья. Они учатся в одном классе. В их школу прибыли школьники из другой страны.

В субботу в 11:00, выходя с Борисом из школы, Никита узнает от Риты, которая не в школе, так как болеет, что Джону, одному из зарубежных гостей, нужно срочно вернуться в свою страну. Семья, которая его принимает, должна была привезти вещи, забрать его у главного входа в школу и отвезти в аэропорт, но она не сможет этого сделать. Также выясняется, что Джон где-то оставил свой мобильный телефон, поэтому не знает о том, что семья не сможет его отвезти.

Джон сейчас ищет свой телефон, который может быть либо в интернет-кафе, либо в ресторанном дворике. Интернет-кафе и ресторанный дворик равноудалены от школы.

Билет на Джона куплен. Рейс в 16:00, но нужно быть в аэропорту по крайней мере за 2 часа до вылета. До аэропорта можно добраться на автобусе или электричке. Чтобы прибыть в аэропорт вовремя, на автовокзале нужно быть в 13:00. Дорога от школы до мест посадки на электричку или автобус занимает 1 час.

Джон не может улететь без вещей и мобильного телефона. Привезти вещи Джона в школу займет 1 час».

Для сохранения слабой структуры мы не уточняли множество деталей, например расписание транспорта. Также мы не накладывали ограничений на формат и вид представления данных. Это могло не только провоцировать уточняющие вопросы (что важно само себе), но и позволить добиться большей вариативности решений.

[1] Слабоструктурированная задача — это задача или проблема, которая не имеет жесткой и однозначной структуры или ясно определенного решения. Она характеризуется неопределенностью, отсутствием четких правил и пути решения. В отличие от структурированных задач, где имеются четко определенная последовательность шагов и конкретные критерии для достижения результата, в слабоструктурированных задачах решение может зависеть от множества факторов, таких как контекст, предпочтения, опыт и интуиция. В них часто требуются анализ и оценка различных альтернатив, чтобы прийти к наилучшему возможному решению.

Стимульный материал содержал рисунок, уточняющий расположение упоминаемых в задаче объектов (рис. 1).

Рис. 1. Рисунок из стимульного материала

Инструкция предъявлялась в следующем виде:

«Ваша задача — проинструктировать ребят, как им действовать в этой ситуации, чтобы доставить Джона в аэропорт оптимальным способом».

В случае возникновения вопросов по сюжетной информации, которой нет в условии, мы предлагали респондентам представить или придумать возможные варианты ситуации ответа и учесть их в инструкциях.

Таким образом, задача респондентов свелась к написанию инструкции в любой удобной форме.

Метод

Выборка

В исследовании приняли участие 54 человека, из которых 36 мужчин и 19 женщин. Из них 27 человек — студенты второго и третьего курсов колледжа, обучающиеся по следующим направлениям: «Прикладная информатика», «Информационные системы в программировании» и «Программирование в компьютерных системах». Возраст респондентов от 17 до 30 лет (средний возраст — 20 лет). Среднее количество часов (в неделю), затрачиваемое на программирование, по выборке от 0 до 70 ч (медиана — 4 ч). Опыт (в годах) программирования варьировался от 0 до 10 лет (медиана — 1 год). Доля (в процентах) рабочего времени, затрачиваемого на организационную работу или планирование, от 0 до 90 % (медиана — 20 %). Выборка формировалась методом «снежного

кома», но так, чтобы в числе участников оказались люди с разнообразным опытом обучения и работы в области программирования.

Пять наблюдений были исключены из первоначальной выборки из-за отсутствия решения. Итоговая выборка составила 49 человек.

Процедура

Сбор данных осуществлялся через онлайн-платформу Online Test Pad. Перед заполнением формы респонденты были проинформированы о том, что исследование является анонимным и результаты будут анализироваться в обобщенной форме.

Анализ

Анализ протоколов решений происходил по следующим характеристикам:

1. Количество шагов.

Количество совершаемых персонажами действий (при условии решения задачи) рассматривалось нами как сумма шагов в алгоритме. В типичных случаях при сравнении двух алгоритмов решения одной и той же задачи при прочих равных предпочтение отдается более короткому — экономичному — алгоритму. Поэтому эта характеристика была включена в анализ.

Действия могли быть или *коммуникацией* между персонажами, или их *перемещением* из точки в точку.

1.1. Коммуникация.

Данный критерий включает любые виды коммуникации, которые фиксировали респонденты в своих решениях. Например, «Рита сообщила Никите и Борису...» или «дети звонят в интернет-кафе и ресторанный дворик».

1.2. Перемещения.

Под перемещением понимается любое действие, которое ведет к изменению текущего местоположения любого из персонажей. Например, «Рита едет туда, куда ей ближе...», «Борис идет в интернет-кафе» или «Джон приезжает в аэропорт и проходит регистрацию на рейс».

2. Логические ошибки.

В случае алгоритма соблюдение логики — важная характеристика, поскольку обеспечивает корректную работу всей программы. Поэтому при анализе инструкций мы фиксировали все логические ошибки. К ошибкам мы отнесли противоречивую информацию по отношению к условию задачи и описанным шагам внутри инструкции. Характеристики, относящиеся к нарушению логики, включали:

2.1. Безальтернативные допущения.

Поскольку мы не вводили никаких дополнительных ограничений, участники могли допускать возможности, изначально не прописанные в инструкции. Однако если такие допущения принимались как единственно возможные, то это ухудшало качество инструкции, потому что не учитыва-

ло факт, что сделанное допущение может быть неверным. Например, «Так как Рита единственная сейчас не в школе, потому что она “болеет”, то она едет за вещами Джона». При этом допускается, что болезнь Риты позволяет ей уезжать из дома, и не обсуждается альтернативный вариант.

Поэтому случаи, когда участник сделал допущение, но не рассмотрел альтернативу, мы считали ухудшающими качество инструкции.

2.2. Разрывы в последовательности шагов.

Инструкции части респондентов содержали изолированные друг от друга шаги, связь между которыми была неясна. Например, в решении могло отсутствовать описание перемещения Бориса после того, как он забрал вещи Джона; из дальнейшего описания можно было только догадываться, что он вернулся к школе.

2.3. Нарушения условий задачи.

Инструкции части респондентов содержали измененные входные условия, ведущие к их упрощению. Условия могли меняться из-за невнимательности респондентов, связанной, очевидно, с большим объемом информации и, возможно, со слабой мотивацией участия респондентов в исследовании. Как пример — включение на первом шаге решения действия Джона, который по условию находится не в школе, отправной точке маршрута, и, следовательно, он не может быть действующим лицом.

Другим примером является решение, в котором респондент пишет, что Борис и Никита разделяются и тот, кто первый находит телефон, идет на автовокзал, то есть один из них перестает участвовать в решении. Однако далее указывается, что Джон, забрав свои вещи, встречается с двумя мальчиками.

3. Логические высказывания.

Мы фиксировали эксплицитное использование логических высказываний, например «либо — либо», «если — то». Данный критерий чаще всего можно обнаружить после того, как респондент предлагает персонажам разделиться в поисках телефона и Джона, о точном местоположении которых в условии задачи не сказано. Таким образом, мы считали количество исходов предложенного респондентом действия. Например, «В зависимости от того, где был Джон относительно телефона и кто его в итоге нашел, каждый из двух ребят может найти либо телефон, либо Джона, либо их обоих, либо ничего не найти».

4. Персонажи.

4.1. Неизвестные персонажи.

Неизвестные персонажи, введенные в инструкцию, расценивались как «deus ex machina» — прием, позволяющий обойти необходимость продумывать непротиворечивую последовательность шагов, что приводило к «нелегитимному» упрощению решения. Для работающего алгоритма такие действия недопустимы. Поэтому мы фиксировали количество упомянутых в решении персонажей, отсутствующих в условии. Например, «В этот же момент Никита и Борис объясняют охраннику или вахтеру в школе случившуюся ситуацию...».

4.2. Количество персонажей.

По данному параметру мы фиксировали количество персонажей, задействованных в решении. Для данной задачи оптимальным количеством действующих персонажей было три или четыре. Если использовался один или два персонажа, то общее затрачиваемое время для решения задачи (доставки Джона в аэропорт) неизбежно удлинялось. Иными словами, написанная инструкция игнорировала часть возможностей, которые предоставляли условия задачи, и вела к неоптимальному решению.

Некоторые из указанных характеристик сами по себе могли не быть показательными с точки зрения качества алгоритма в инструкции. Такие характеристики имело смысл совместить в комплексные показатели.

5. Комплексные показатели качества инструкции.

5.1. Отношение количества разрывов в последовательности шагов к общему количеству шагов.

Характеристика дает дополнительную информацию о качестве инструкции, так что более длинные последовательности действий могли быть выше коротких по качеству, если последние содержали большую долю «разрывов».

5.2. Отношение количества безальтернативных допущений к общему количеству шагов.

Характеристика также дает дополнительную информацию о качестве инструкции, позволяя сравнивать длинные, но аккуратные последовательности действий с более короткими, но полагающими не всегда обоснованные допущения.

Таким образом, каждый из протоколов анализировался по 11 характеристикам.

Кодировка.

Мы использовали кодировку, присваивая единицу в случае, если инструкция содержала одну из описанных характеристик. Для подсчета частоты мы проводили анализ каждого протокола. Пример закодированного протокола представлен на рис. 2.

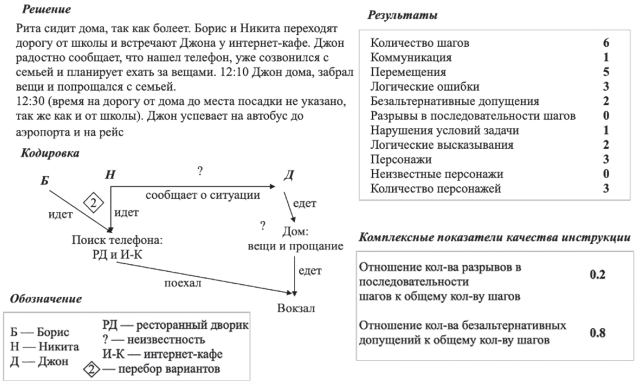

Рис. 2. Пример закодированного решения респондента

Результаты

Оценка выделенных характеристик и их межиндивидуальная вариативность

Во-первых, надо отметить, что все указанные характеристики могли быть объективно и непротиворечиво закодированы для всех протоколов. Это значит, что они соответствуют требованиям формализации и унификации. Во-вторых, в результате анализа закодированных протоколов мы увидели, что практически по каждой из выделенных характеристик достигается необходимая вариативность (кроме характеристики «Неизвестные персонажи») (таблица).

Варьирование частоты проявления характеристик

Характеристика	Минимум	Максимум
1. Количество шагов		
1.1 Коммуникация	0	7
1.2 Перемещения	1	8
2. Логические ошибки		
2.1 Безальтернативные допущения	0	10
2.2 Разрывы в последовательности шагов	0	4
2.3 Нарушения условий задачи	0	3
3. Логические высказывания	0	13
4. Персонажи		
4.1 Неизвестные персонажи	0	1
4.2 Количество персонажей	1	5

Судя по этим данным, межиндивидуальная вариативность инструкций по выбранным характеристикам была достигнута.

Сравнение инструкций по выделенным характеристикам для нескольких случаев

Напомним, что целью этого исследования была оценка самого метода исследования алгоритмического мышления, а не сравнительная оценка групп с помощью этого метода. Тем не менее мы сделали попытку определить те параметры, по которым можно ожидать значимых различий между группами людей с разной подготовкой в области программирования. Эта задача осложнялась тем, что на этапе формирования выборки мы стремились добиться максимального разнообразия программистского опыта участников. Более того, мы не имели информации о программах обучения программированию и специфике рабочего опыта участников. В итоге мы были вынуждены отказаться от сравнения между программистами и непрограммистами, потому что нам не удалось образовать хоть сколько-нибудь представительных по численности групп, члены которых были бы существенно различны по опыту программирования между группами и схожи внутри групп. Поэтому для демонстрации качественных межиндивидуальных различий

мы ограничились несколькими случаями, весьма показательными в отношении алгоритмизации инструкции.

Сначала рассмотрим решение респондента, имеющего опыт программирования (рис. 3).

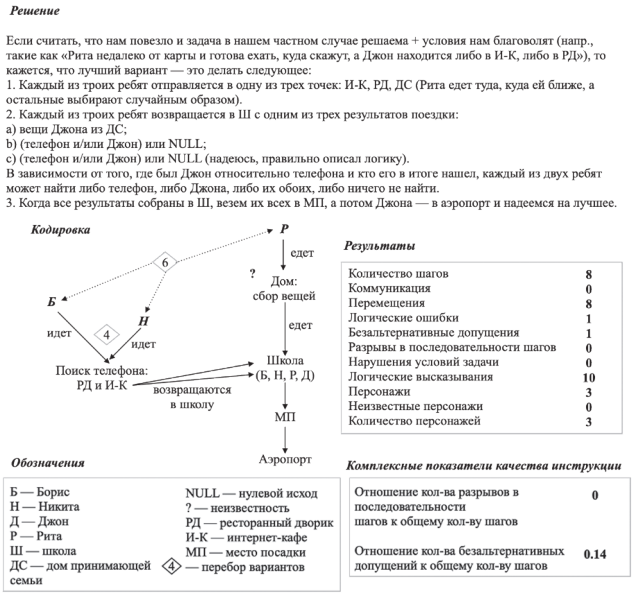

Рис. 3. Пример решения респондента с опытом программирования

Помимо представленного на рис. 3 решения респондент также показал свои размышления, добавив **описание неизвестных переменных**. Например, «Является ли “расстояние” симметричной функцией (например, верно ли, что dist Ш — И-К == dist И-К — Ш)?», где *dist* — расстояние, *Ш* — школа, *И-К* — интернет-кафе. Респондент учел потенциальное множество решений в предложенной ситуации и написал: «Существуют такие множества условий, при которых решения задачи не существует. Например, если рейс в 16:00 вчерашнего дня. Или, например, если dist И-К — Ш > 2 часов, а рейс сегодня». Таким образом, мы видим, что респондент учитывает инструкцию о включении ответов на свои вопросы в решении, а также пытается выяснить все неизвестные условия.

Из рисунка 3 мы видим, что решение респондента содержит минимальное количество баллов, набранных по характеристикам, относящимся к **логическим ошибкам** решения. В частности, одно допущение, отсутствие разрывов и нарушений логики. При этом, говоря о **безальтернативных допущениях**, мы не присвоили дополнительный балл за включение Риты как действующего персонажа в решение, несмотря на ее болезнь. Поскольку респондент понимал, что может нарушать поставленное условие, он добавил следующий комментарий в решение: «Если считать, что нам повезло и задача в нашем частном случае ре-

шаема + условия нам благоволят (напр., такие как “Рита недалеко от карты и готова ехать, куда скажут...”)». Другими словами, респондент принимал это допущение не как единственно возможное.

Относительно другого **безальтернативного допущения**, которому был присвоен балл, респондент неявным образом предполагает, что возможность забрать вещи у ребят есть. Об этом не сказано в условии, а также не зафиксировано в самом решении, в отличие от других сомнений и комментариев, возникших в результате анализа условий задачи.

Также из рис. 3 можно заключить, что респондент видит альтернативные исходы описанных шагов. Таким образом, мы смогли зафиксировать наличие в решении характеристик, относящихся к **логическим высказываниям**. В частности, он говорит о рандомизации выбора точек, в которые могут отправиться ребята, и вариативности результатов поиска Джона и телефона. Отметим, что для визуализации мы зафиксировали один из возможных случаев отправки персонажей в интернет-кафе, ресторанный дворик и дом принимающей семьи.

Количество персонажей в рассматриваемом случае было оптимальным.

По **комплексным показателям качества** инструкции мы видим, что один из них равен нулю, а другой приближается к нулевому значению. Это также указывает на качество представленной инструкции — отсутствие неаккуратных последовательностей действий.

В заключение отметим, что интересным было и то, что респондент добавил в решение блок с рефлексией. В частности, им были описаны возможные «точки расширения» данного решения. Другими словами, респондент указывал на внедрение дополнительных условий, не изменяющих структуру решения, а оптимизирующих время. Например, он пишет: «Можно ли купить вещи и телефон в аэропорту? Тогда сразу ищем Джона и едем туда». В этом решении также встречаются вопросы, уточняющие условия и предлагающие новые пути решения. Например, «А можно ли обменять билет на самолет и вернуться в другое время? Так, можно не решать текущую проблему, а создать себе сразу рабочую конфигурацию».

Теперь рассмотрим решение другого респондента, не имеющего опыт программирования (рис. 4).

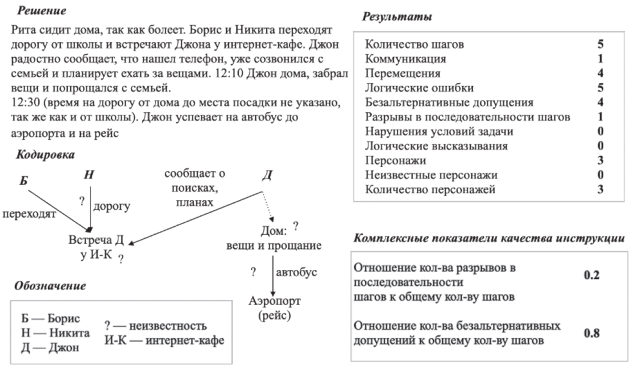

Рис. 4. Пример решения респондента без опыта программирования

Первое, на что хочется обратить внимание, — это отсутствие дополнительных блоков с уточнениями и рефлексией. Респондент представил только решение, которое имеет несколько допущений, а также упрощающее предложенную ситуацию. Поговорим о каждой показательной характеристике подробнее.

Из рисунка 4 мы видим, что респондент использует небольшое **количество шагов**. Напомним, что у нас не было эталонного количества шагов. При этом решение выглядит линейным, без включения всей мощности потенциальных возможностей. Под линейным решением мы понимаем отсутствие декомпозиции решения. Другими словами, решение респондента не содержит особенность присущей профессиональной деятельности программистов. Респондент не снижает сложность предложенной ситуации, а также не пытается оптимизировать время, затраченное на решение.

Относительно **логических ошибок** в решении мы зафиксировали некоторую неаккуратность, ведущую к менее качественному решению, в отличие от респондентов, имеющих опыт программирования. В частности, мы видим четыре **безальтернативных допущения** и один **разрыв в последовательности шагов**. Опишем каждую из **логических ошибок**.

Первые два **безальтернативных допущения**, которые видно из рис. 4, — предположения о встрече персонажей и нахождении интернет-кафе на противоположной стороне от школы. Несмотря на то что оба допущения встречаются на первом шаге, мы присвоили каждому из них по одному баллу.

Третье **безальтернативное допущение**, которое встретилось у многих респондентов вне зависимости от наличия или отсутствия программистского опыта, — предположение о возможности забрать вещи из дома. В то же время мы можем предположить из описания решения, что Джон обговорил планы с принимающей семьей и они ждут его дома. Поскольку это неявно читается в решении, мы засчитали это как допущение, сделанное респондентом.

И наконец, мы присвоили балл за **безальтернативное допущение**, предполагающее доступность транспорта от дома семьи. В отличие от этого решения, мы могли видеть в решениях респондентов с опытом программирования отдельный шаг про уточнение расписания в месте посадки или в процессе выполнения других задач ребятами, где Рита является помощником, не участвующим в перемещениях. Это делает решение более прозрачным и полным, то есть то, что мы называем «качественным» решением.

Говоря о **разрывах в последовательности шагов**, мы обратили внимание, что респондент обрывает ветку решения, в котором могли бы участвовать Борис и Никита, то есть они остаются в интернет-кафе и неясно, что с их дальнейшими действиями.

Поскольку респондент упростил свое решение, нам также не удалось обнаружить в написанной инструкции **логических высказываний**. Это соответствует нашим ожиданиям относительно

проявления данной характеристики в группе респондентов с опытом программирования.

Помимо прочего, в решении мы также не обнаружили наличия **неизвестных персонажей**. В целом данная характеристика не позволяла дифференцировать респондентов. Поэтому мы планируем исключить эту опцию в будущих исследованиях с предложенным нами методом.

Количество персонажей на первый взгляд, кажется, попадает в диапазон оптимального. Однако, по сути, решение основано на действиях одного человека. При этом мы не могли не посчитать других персонажей, которые совершают действие — переходят дорогу. Поэтому, несмотря на зафиксированное количество персонажей, реальное количество сводится к одному.

По **комплексным показателям качества инструкции** мы видим, что обе характеристики отличны от нуля. Это также указывает на минимальное качество представленной инструкции — наличие неаккуратных последовательностей действий. Отметим, что, несмотря на потенциальное доказательство некачественности решения, в рамках данного исследования и метода мы не можем делать выводы относительно разности в показателях. Сейчас он бинарный: значения, равные нулю, указывающие на качество решения, и отличные от нуля, ведущие к некачественным решениям.

Таким образом, даже на этих двух примерах мы видим отличия в качестве решений и работоспособности почти всех выделенных показателей. Суммируя, мы обнаружили, что в решении респондента без опыта программирования, в отличие от респондента с опытом, чаще наблюдались допущения, что вело к менее понятному изложению инструкции, неоптимальному количеству действий, а также к отсутствию переборов, то есть рассмотрению альтернативных решений. Важно отметить, что в инструкциях респондента без опыта программирования оба показателя по комплексным характеристикам отличны от нуля.

Обсуждение

В целом можно сказать, что написание инструкций предоставляет богатый материал для оценки алгоритмического мышления, причем оценки объективной и в большинстве случаев непротиворечивой. Однако поскольку в этой работе использовалась только одна задача, для решения которой нужно было написать инструкцию, то встает вопрос о том, насколько этот метод будет эффективным для других задач.

Мы полагаем, что данный метод применим в отношении широкого круга задач, решение которых принципиально алгоритмизируемо. По большому счету любые инструкции — инструкции эксплуатации каких-либо механизмов, должностные инструкции, инструкции поведения в кризисных или опасных ситуациях, инструкции для подготовки специалистов — подчиняются одним и тем же нормам. Например, инструкция должна:

— охватить типичные или наиболее вероятные сценарии развития событий и действия участвующих лиц;
— предложить оптимальную последовательность действий;
— учесть необходимые и достаточные условия для перехода на следующий шаг или на другой сценарий;
— оценить достижение промежуточных или конечных результатов.

В этом смысле инструкции для персонажей истории, которые были написаны нашими респондентами, следуют тем же нормам. Понятно, что инструкции, написанные для решения одной и той же задачи, могут различаться по качеству: учета имеющихся ресурсов, логичности и последовательности описанных шагов, достаточности предлагаемых шагов и их детализации и учитываемых в инструкции сценариев развития событий и т. д. В этом смысле любая задача с несколькими известными вариантами решения может служить стимульным материалом для оценки алгоритмического мышления через написание инструкции к ее решению.

Говоря о полученных результатах, необходимо подчеркнуть их предварительный характер. Прежде всего необходимо убедиться в надежности предлагаемого метода на других выборках и задачах. Пока неизвестно, насколько сильно качество инструкций будет различаться у одного и того же человека при алгоритмизации решений разных задач. Важно также убедиться, что по мере накопления опыта работы с написанием алгоритмов (в нашем случае — опыта программирования) будут улучшаться и инструкции. Наконец, пока неизвестно, какие из выделенных параметров зарекомендуют себя как показательные для разных уровней развития алгоритмического мышления.

В отношении продолжения исследования имеет смысл рассмотреть написание инструкции как самостоятельную задачу. То есть вместо непосредственного решения вымышленной истории (слабоструктурированной задачи) респонденту может предлагаться уже готовое ее решение. Такой подход будет выравнивать респондентов по одному основанию, и их итоговые инструкции не будут зависеть от выбранного ими способа решения.

Algorithmic thinking is primarily assessed through standardized tests and professional tasks, limiting the ability to identify the specifics of algorithmization in individuals with varying programming experience. To refine the hypothesis regarding such specificity, we proposed an alternative assessment method that requires algorithmization but not specialized programming knowledge. Using the solution protocols of the task we developed, we identified quality parameters of algorithmization that allow for reliable encoding. The results demonstrated that the proposed method provides the required variability in solutions and enables the creation of new task variants based on the identified parameters. Additionally, the hypothesis about the specificity of algorithmization in individuals with different programming experience was further clarified, although its validation requires specific research.

Keywords: problem-solving, programming, algorithmic thinking, instruction.

Литература

1. *Кукушкина, Ю. А.* Критическое мышление как фактор профессиональной компетентности программистов / Ю. А Кукушкина, В. Ф. Спиридонов // Психология. — 2008. — Т. 5, № 1. — С. 165–174.

Kukushkina, Yu. A. Kriticheskoe my`shlenie kak faktor professional`noj kompetentnosti programmistov / Yu. A. Kukushkina, V. F. Spiridonov // Psixologiya. — 2008. — T. 5, № 1. — S. 165–174.

2. *Орел, Е. А.* Особенности интеллекта профессиональных программистов / Е. А. Орел // Вестник Московского университета. Серия 14, Психология. — 2007. — № 2. — С. 70–79.

Orel, E. A. Osobennosti intellekta professional`ny`x programmistov / E. A. Orel // Vestnik Moskovskogo universiteta. Seriya 14, Psixologiya. — 2007. — № 2. — S. 70–79.

3. Примеры заданий предварительного тестирования PISA 2015 / Федеральный институт оценки качества образования. — OECD, 2015. — 89 с. — (Программа ОЭСР «Международная оценка образовательных достижений учащихся»).

Primery` zadanij predvaritel`nogo testirovaniya PISA 2015 / Federal`ny`j institut ocenki kachestva obrazovaniya. — OECD, 2015. — 89 s. — (Programma OE`SR «Mezhdunarodnaya ocenka obrazovatel`ny`x dostizhenij uchashhixsya»).

4. Российская Федерация. Правительство. Об утверждении федерального государственного образовательного стандарта среднего профессионального образования по специальности 09.02.03 Программирование в компьютерных системах : Приказ Минобрнауки РФ от 28.07.2014 № 804. — URL: https://fgos.ru/fgos/fgos-09-02-03-programmirovanie-v-kompyuternyh-sistemah-804/?ysclid=lus18javkd535678914 (дата обращения: 09.04.2024).

Rossijskaya Federaciya. Pravitel`stvo. Ob utverzhdenii federal`nogo gosudarstvennogo obrazovatel`nogo standarta srednego professional`nogo obrazovaniya po special`nosti 09.02.03 Programmirovanie v komp`yuterny`x sistemax : Prikaz Minobrnauki RF ot 28.07.2014 № 804. — URL: https://fgos.ru/fgos/fgos-09-02-03-programmirovanie-v-kompyuternyh-sistemah-804/?ysclid=lus18javkd535678914 (data obrashheniya: 09.04.2024).

5. *Adelson, B.* Problem solving and the development of abstract categories in programming languages / B. Adelson // Memory and Cognition. — 1981. — Vol. 9. — P. 422–433.

6. *Aljehane, S.* Determining Differences in Reading Behavior Between Experts and Novices by Investigating Eye Movement on Source Code Constructs During a Bug Fixing Task / S. Aljehane, B. Sharif, J. Maletic // ACM Symposium on Eye Tracking Research and Applications (ETRA'21 Short Papers) / Association for Computing Machinery. — New York, 2021. — DOI: 10.1145/3448018.3457424

7. *Bernard, M.* Enhancing the Metacognitive Skill of Novice Programmers Through Collaborative Learning / M. Bernard, E. Bachu // Intelligent Systems Reference Library. — 2015. — Vol. 76. — P. 277–298. — DOI: 10.1007/978-3-319-11062-2_11

8. *Cathcart, W. G.* Effects of Logo Instruction on Cognitive Style / W. G. Cathcart // Journal of Educational Computing Research. — 1990. — Vol. 6, № 2. — P. 231–242.

9. *Clements, D. H.* Enhancement of Creativity in Computer Environments / D. H. Clements // American Educational Research Journal. — 1991. — Vol. 28, № 1. — P. 173–187.

10. Comparing Programming Language Comprehension between Novice and Expert Programmers Using EEG Analysis / S. Lee, A. Matteson, D. Hooshyar [et al.] // IEEE 16th International Conference on Bioinformatics and Bioengineering (BIBE). — Taichung, Taiwan, 2016. — DOI: 10.1109/BIBE.2016.30

11. Developer Reading Behavior While Summarizing Java Methods : Size and Context Matters / N. J. Abid, B. Sharif, N. Dragan [et al.] // IEEE/ACM 41st International Conference on Software Engineering (ICSE). — Montreal, 2019. — DOI: 10.1109/ICSE.2019.00052

12. Eye tracking in computing education / T. Busjahn, C. Schulte, B. Sharif [et al.] // The Tenth Annual Conference on International Computing Education Research. — Glasgow, 2014. — P. 3–10.

13. *Fessakis, G.* Problem solving by 5–6 years old kindergarten children in a computer programming environment: a case study / G. Fessakis, E. Gouli, E. Mavroudi // Computers and Education. — 2013. — Vol. 63. — P. 87–97.

14. *Fox, R. W.* The Effect of Computer Programming Education on the Reasoning Skills of High School Students / R. W. Fox, M. E. Farmer // Instructional Science. — 2011. — Vol. 45, № 5. — P. 583–602.

15. How Experts Adapt Their Gaze Behavior When Modeling a Task to Novices / S. N. Emhardt, E. M. Kok, H. Jarodzka [et al.]. // Cognitive Science. — 2020. — Vol. 44, № 9. — e12893. — DOI: 10.1111/cogs.12893

16. *Koenemann, J.* Expert problem solving strategies for program comprehension / J. Koenemann, S. P. Robertson // The SIGCHI Conference on Human Factors in Computing Systems. — New Orleans, 1991. — P. 125–130.

17. Metacognitive Difficulties Faced by Novice Programmers in Automated Assessment Tools / J. Prather, P. Raymond, M. Kayla, A. Peters // ACM Conference on International Computing Education Research (ICER'18) / Association for Computing Machinery. — New York, 2018. — DOI: 10.1145/3230977.3230981

18. *Norman, D. A.* User centered system design : New perspectives on human-computer interaction / D. A. Norman, S. W. Draper // Cognitive Engineering, Lawrence Erlbaum Associates, Hillsdale. — New Jersey, 1986. — P. 31–61.

19. *Obaidellah, U.* A Survey on the Usage of Eye-Tracking in Computer Programming / U. Obaidellah, M. Haek, P. C.-H. Cheng // ACM Computing Surveys. — 2018. — Vol. 51, № 1. — DOI: 10.1145/3145904

20. *Psycharis, S.* The effects of computer programming on high school students' reasoning skills and mathematical self-efficacy and problem solving / S. Psycharis, M. Kallia // Instructional Science. — 2017. — Vol. 45. — P. 583–602.

21. *Sala, G.* Experts' memory superiority for domain-specific random material generalizes across fields of expertise : A meta-analysis / G. Sala, F. Gobet // Memory and Cognition. — 2017. — Vol. 45, № 2. — DOI: 10.3758/s13421-016-0663-2

22. Understanding the Differences Between Novice and Expert Programmers in Memorizing Source / M. Kramer, M. Barkmin, D. Tobinski, T. Brinda // Tomorrow's Learning : Involving Everyone. Learning with and about Technologies and Computing. WCCE 2017. IFIP Advances in Information and Communication Technology. — 2017. — Vol. 515. — DOI: 10.1007/978-3-319-74310-3_63

23. Using eye tracking to examine expert-novice differences during simulated surgical training : A case study / S. Li, M. C. Duffy, P. L. Susanne [et al.] // Computers in Human Behavior. — 2023. — Vol. 144, № 7. — DOI: 10.1016/j.chb.2023.107720

24. *Weiser, M.* Programming problem representation in novice and expert programmers / M. Weiser, J. Shertz // International Journal of Man-Machine Studies. — 1983. — Vol. 19, № 4. — P. 391–398.

25. *Woodrow, B.* Expert-novice differences for software: implications for problem-solving and knowledge acquisition / B. Woodrow // Behaviour and Information Technology. — 1986. — Vol. 5, № 1. — P. 15–29.

26. *Ye, N.* Expert-novice knowledge of computer programming at different levels of abstraction / N. Ye, G. Salvendy // Ergonomics. — 1996. — Vol. 39, № 3. — P. 461–481.