**МЕТОДОЛОГИЯ И ТЕХНОЛОГИЯ ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ/METHODOLOGY AND TECHNOLOGY OF VOCATIONAL EDUCATION**

**DOI: https://doi.org/10.60797/PED.2025.8.3**

# ГЕНЕРАТИВНЫЙ ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ И НАЧАЛЬНОЕ ОБУЧЕНИЕ ПРОГРАММИРОВАНИЮ В УНИВЕРСИТЕТАХ: СОДЕРЖАНИЕ, ЗАДАНИЯ, ОЦЕНИВАНИЕ

Обзор

**Долинский М.С.[1, *]**
[1] ORCID : 0000-0002-3057-4051;
[1] Гомельский государственный университет имени Ф. Скорины, Гомель, Беларусь

* Корреспондирующий автор (dolinsky[at]gsu.by)

**Аннотация**

В данной обзорной работе кратко излагается содержание, задания и методология обучения программированию во вводных университетских предметах, традиционно используемые инструменты обучения, системы и способы оценивания. А затем приводится описание направлений использования генеративного искусственного интеллекта (GenAI) для повышения качества обучения программированию и изучения программирования во вводных университетских предметах. Особенное внимание акцентировано на формирующем оценивании, главная цель которого помочь студенту быстро и качественно учиться на своих ошибках. Работа может послужить отправной точкой разработки новых учебных программ, методик и средств обучения программированию во вводных университетских курсах, учитывающих появление генеративного искусственного интеллекта.

**Ключевые слова:** генеративный искусственный интеллект, начальное обучение программированию, университет, содержание, задания, оценивание.

# GENERATIVE ARTIFICIAL INTELLIGENCE AND PRIMARY PROGRAMMING EDUCATION AT UNIVERSITIES: CONTENT, ASSIGNMENTS, GRADING

Review article

**Dolinsky M.S.[1, *]**
[1] ORCID : 0000-0002-3057-4051;
[1] Francisk Skorina Gomel State University, Gomel, Belarus

* Corresponding author (dolinsky[at]gsu.by)

**Abstract**

This review article summarises the content, assignments and methodology of teaching programming in introductory university subjects, traditionally used teaching tools, systems and ways of grading. It then describes the directions of using Generative Artificial Intelligence (GenAI) to improve the quality of programming education and learning in introductory university subjects. Special attention is emphasised on formative assessment, the main aim of which is to help the student to learn quickly and efficiently from their mistakes. The paper can serve as a starting point for the development of new curricula, methods and tools for teaching programming in introductory university courses, taking into account the emergence of generative artificial intelligence.

**Keywords:** generative artificial intelligence, primary programming education, university, content, assignments, grading.

## Введение

Программирование — чрезвычайно востребованная отрасль знаний, потребность в которых стремительно растёт. Одновременно растёт количество студентов, изучающих программирование. Параллельно растёт дифференциация в уровне знаний и мотивации студентов, приступающих к изучению программирования. По экономическим соображениям количество преподавателей не может увеличиваться пропорционально количеству студентов. Всё это вместе приводит к ухудшению качества обучения и увеличению количества студентов, не сумевших закончить начальный курс программирования. Такое положение не может удовлетворять ни студентов, ни преподавателей, ни администрацию вузов, ни общество. Поэтому непрерывно ведутся исследования и разработки для решения указанной проблемы. Одним из мощнейших инструментов-помощников на сегодня видится генеративный искусственный интеллект (GenAI). В данной статье далее излагается содержание, задания и методология обучения программированию во вводных университетских предметах, традиционно используемые инструменты обучения, системы и способы оценивания. А затем приводится описание направлений использования GenAI для повышения качества обучения программированию и изучения программирования во вводных университетских предметах. Особенное внимание акцентировано на формирующем оценивании, главная цель которого помочь студенту быстро и качественно учиться на своих ошибках.

## Содержание и методология начального обучения программированию

В работе [1] представлено описание методологии начального обучения программированию в Northeastern State University (USA). На занятия отводится 3 часа лекции и 1 час лабораторных работ в неделю. Во время лекции инструктор демонстрирует программы в Visual Studio Code — коды заданий из учебника и упражнений в конце главы. На лабораторных работах используются специальные короткие упражнения и решения для независимой практики.

Курс включает 6 индивидуальных заданий на программирование и финальный экзамен с вопросами с множественным выбором. Книга по курсу (в бумажном и электронном форматах) содержит решения всех упражнений.

Темы занятий:

1. Ввод чисел, вычисление суммы и среднего
2. Циклы и форматирование чисел и таблиц
3. Чтение файлов, частичные суммы
4. Генерация случайных чисел, чтение/запись файлов, вложенные циклы
5. Функции
6. Массивы, сортировка, поиск

В работе [2] приводятся особенности обучения в Lappeenranta–Lahti University of Technology (Finland). Курс имеет 7 недель, каждую неделю 3–5 заданий, всего 31 задание, причём последнее необязательное. Вопросы по темам также выпускаются каждую неделю. Описана система генерации и проверки маленьких заданий на программирование, а также управления ими. Новые задания делаются для противодействия такой форме плагиата, когда студенты, ранее завершившие курс, передают свои решения новым поколениям студентов. Был также разработан инструмент управления маленькими заданиями. Чтобы можно было выбрать/создать множество заданий для курса этого года, отличающихся от заданий прошлых лет, но поддерживающих обучение той же теме каждую лекционную неделю. В 2023 году были сделаны новые задания. Большинство этих заданий основывалось на заданиях прошлых лет и сделаны как вариации заданий прошлых лет модификацией данных, текстов, последовательностей выполнения.

В работе [3] описывается авторский экспериментальный подход к начальному обучению программированию на языке C, ориентированный на Outcome-Based Education (обучение, ориентированное на результат), применяемый с 2021 года в Gansu University of Political Science and Law (China). В 2021 году 123 студента взялись изучать этот курс. Никто из этих студентов никогда не изучал программирование:

53% мало пользовались компьютером

41% имели хорошую математическую подготовку

85% имели достаточную математическую подготовку для изучения программирования. Предложенный материал для изучения:

- ввод-вывод с клавиатуры на экран
- операторы, выражения
- последовательные, условные, циклические структуры
- массивы чисел, символов, строк
- функции стандартные, пользовательские
- битовая обработка
- работа с файлами

Автор придерживался также концепции Conceive–Design–Implement–Operate (Задумать–Проектировать–Реализовать–Эксплуатировать) детализированной следующими ключевыми процессами:

- project leading — важную роль в обучении играла разработка проекта
- flipping teaching — «перевёрнутое» обучение, когда студент является активным действующим лицом обучения, в разработке собственного проекта, задавая вопросы и инициируя обсуждение возникающих при разработке проблем
- personality training — индивидуальное обучение, ориентированное на интересы и проблемы студента
- task driving — обучение, движущей силой которого являются поставленные перед студентом задачи
- pair programming — при решении задач и выполнении проектов программы пишутся парами студентов, что позволяет им самостоятельно справляться с большим количеством проблем, чем если бы они работали по одиночке.
- process assessment — перманентное оценивание процесса обучения, позволяющее оперативно реагировать на проблемы, возникающие у студентов.

Было создано огромное количество обучающих ресурсов, включая микро-видео, задания и соответствующее их оценивание. Содержание курса было организовано в 12 модулей. Каждому модулю прилагалось высококачественное видео по 6–8 мин. Студенты могли просматривать эти видео с мобильных устройств в удобное для них время. В классе преподаватель отвечал на вопросы студентов по необходимости. Было установлено более 800 вопросов и 92 проекта в 6 категориях.

В работе [4] описывается алгоритм генерации пакетов индивидуальных заданий каждому студенту из накопленного большого банка заданий, обеспечивающий примерно равную сложность и представленность всех изучаемых тем. Такой подход к составлению заданий предложен авторами, чтобы исключить ситуации, когда студенты при изучении вместо самостоятельного выполнения заданий берут решения у студентов, изучавших данный предмет в прошлые годы.

Работа [5] содержит предложения по изменению начального обучения программированию, реализованные в Tan Trao University (Vietnam), включая: улучшение программы обучения, семинары по улучшению кодирования, индивидуальные программы обучения, внедрение проектов, связанных с промышленностью, интеграцию в программы обучения современных языков программирования и промышленных практик, сотрудничество с промышленными экспертами при разработке программ обучения, инвестирование в современные обучающие ресурсы, включая средства разработки и онлайн-платформы, внедрение мотивационных стратегий, таких как геймификация, соревнования по программированию, хакатоны.

В работе [6] приводится обзор австралийских программ обучения разработке программного обеспечения, подчёркивается, что начальное обучение программированию должно стать надёжным фундаментом для изучения указанных последующих разделов.

Программирование: объектно-ориентированное программирование, программирование систем, операционные системы, машинное обучение, искусственный интеллект, компьютерная графика, архитектура компьютеров, обработка естественного языка, фул-стек разработка, WEB-разработка, разработка мобильных приложений, обработка изображений, параллельные вычисления, блок-чейн, глубокое обучение.

Математика: дискретная математика, статистика, моделирование и анализ, наука о данных, исчисления, линейная алгебра.

Данные и алгоритмы: алгоритмы, структуры данных, база данных, обработка данных.

Soft-transferable skills: решение проблем, коммуникация, рефлексия, абстракция, инновации.

Компьютерные системы и аппаратное обеспечение: разработка встроенных систем, разработка компьютерных систем, электроника, цифровые системы.

Процесс разработки программного обеспечения: улучшение процесса разработки программного обеспечения, сложные системы, системный анализ и моделирование, качество программного обеспечения и тестирование.

Сетевая обработка: компьютерные сети, распределённые системы и сетевое программирование

Кибербезопасность: основы кибербезопасности, безопасность программного обеспечения, ИТ-криминалистика и облачные вычисления.

**Инструменты обучения**

В работе [7] описан проект среды разработки, предназначенный для упрощения обучения основам программирования в контексте огромных классов начинающих. Самая большая проблема для учителей в таких классах — процесс проверки заданий на кодирование. Чтобы решить эту проблему, авторы предлагают платформу, специально спроектированную для уменьшения нагрузки на учителя, давая ему одновременно и общий обзор класса, и способ быстро определить проблемные коды. Платформа базируется на множестве метрик и тестах и обеспечивает учителю значительную статистку и данные. Платформа была протестирована в реальных условиях класса из 500 студентов и показала свою эффективность.

Работа [8] представляет систему автоматической проверки решений по тестам. Решение студента запускается на множестве тестов, подготовленных автором задачи. Студент оперативно получает вердикт о проверке его решения.

В работе [9] представлена программа SeliotM, которая упрощает обучение и повышает его качество за счёт визуализации очередей, стеков и связных списков для программ, написанных на C++. Данные структуры динамически отображаются на экране во время исполнения программы, что существенно повышает понимание студентами алгоритмов их исполнения.

В работе [10] представлена Европейская статистика: среднее количество студентов на преподавателей в 2018 году было 15.3 (в Греции максимум — 38.7, в Люксембурге — минимум — 4.4.). Там же указывается, что оптимальным для обучения считается значение от 5 до 8.

Предлагается активнее использовать студенческое самооценивание! При обучении программированию, в частности, само оцениванию подлежат программы или их фрагменты, по следующим трём излагаемым авторами причинам: такой подход ближе к реальности (по сравнению с ответами на вопросы), стимулируется желание студентов качественно учиться программированию, ведётся подготовка к аналогичным экзаменам.

Работа [11] предлагает смелее и активнее использовать микрообучение — метод обучения, при котором учебный материал разбивается на маленькие порции (обычно не более 10–15 минут), которые можно легко усвоить за короткое время. Содержание курса должно быть представлено в самых разных формах. Например, сочетание презентаций, заданий, игр, дискуссий вовремя вебинаров, видеороликов, тестов, соревнований, пиктограмм, текста. Достоинства микрообучения для студентов: экономия времени, быстрый результат, вовлеченность. Достоинства микрообучения для преподавателя: гибкость — можно при необходимости легко заменить информационные блоки без необходимости перестраивать весь курс.

**Системы и способы оценивания**

В работе [12] излагаются общие подходы к оцениванию заданий. Авторы проанализировали 121 статью за период с 2017 по 2021 годы и категорировали их по подходам, языкам программирования, степени автоматизации и технике оценивания. Предметы, обучающие программированию на начальной стадии, выросли значительно в последнее время как по количеству предметов, так и по количеству студентов, выбравших такие предметы. Это привело к соответствующему увеличению количества проверяемых работ. Время на проверку короткое, а своевременная обратная связь увеличивает мотивацию студентов. Возникает опасность неполного оценивания и снижение качества обратной связи. Увеличение количества преподавателей, проверяющих работы — первый шаг к решению проблемы. Следующий шаг — внедрение систем автоматической проверки решений. Важнейший последующий шаг обеспечить формирующее оценивание, которое обеспечит улучшение качества обучения, а не только проверку правильно или нет выполнено задание. Для автоматической проверки преподавателям необходимо подготовить множество тестов для каждого проверяемого задания, а студентам необходимо строго соблюдать спецификации (например, формат ввода и вывода или имена функций и переменных). Существуют также средства автоматической статической проверки на соответствие решения студента авторскому шаблону. Некоторые преподаватели предпочитают в своих курсах большие проектные задания, которые труднее поддаются автоматическому оцениванию или даже делают его практически невозможным. Авторы предлагают оценивать задания на разработку программных проектов по четырём критериям: корректность, возможность поддерживать развитие, читабельность, документирование.

В работе [13] приводится обзор представлений учителей о способах оценки вычислительного мышления. На одном фланге стоят методы, базирующиеся на индивидуальном общении со студентом, во время разработки им алгоритма решения задачи. Такой подход требует существенных затрат времени учителя. Поэтому на другом фланге стоят методы автоматического оценивания программ. Затраты времени учителя для оценивания сводятся к нулю.

Однако в этом случае работа оценивается по принципу правильно/неправильно, не принимая во внимание качество решения задачи, кроме того, студент не получает обучающей обратной связи. Поэтому предлагаются подходы, основанный на интеграции автоматического оценивания с обучающей обратной связью. Существенным помощником может оказаться генеративный искусственный интеллект.

Работа [14] описывает предпочитаемые студентами методы оценивания. Авторы провели опрос со студентами, один из вопросов был такой: «Какой метод оценки Вы рассматриваете как самый справедливый и почему?» Студенты ответили, что предпочитают письменные экзамены, практические проекты, практические экзамены.

В работе [15] описывается система проверки присылаемых программ на плагиат. В заданиях на программирование, предлагаемых ежегодно по одному и тому же предмету или предлагаемых студентам одного и того же потока, часто встречаются ситуации, когда решение может быть взято у однокурсников, студентов прошлых лет, или в Интернете. Поэтому авторы разработали систему сравнения присылаемых на проверку программ на наличие плагиата, чтобы стимулировать самостоятельную работу студентов.

Работа [16] представляет статическую оценку программ — без исполнения — по уровню соответствия студенческой программы программе преподавателя. К достоинствам такого подхода можно отнести уменьшение временных затрат преподавателя на проверку, с поддержкой относительно качественной обратной связи, помогающей студенты понять свои ошибки. К недостаткам можно отнести всё-таки не 100%-ую проверку программы, поскольку она не исполнялась.

В работе [17] описывается коллегиальная оценка. Группы студентов выполняют проекты, по завершению которых наступает коллегиальная оценка (Peer assessment), состоящая из 3 фаз: презентация, оценка, коммуникация. Развитие этого опыта — представление онлайн, обсуждение онлайн, заключение на основе обсуждения.

В работе [18] описывается эволюция автоматизированной системы тестирования Edgar, включением в неё интерактивного обучающего модуля. Этот модуль может оценивать решения на языках программирования SQL, Java, C, Python и выдавать персональные рекомендации по улучшению ошибочных решений.

Работы [19] и [20] описывают практику автоматической проверки решений студентов по тестам. Студенты посылают свои C++ файлы для заданных упражнений, используя выданные шаблоны кодов. Сервер компилирует студенческий код и проверяет его на 10 тестах, подготовленных преподавателем. Оценка соответствует количеству пройденных тестов. Кроме того, студент получает обратную связь с указанием на ошибки. Студент может делать неограниченное количество повторных отсылок для улучшения оценки. Автоматические средства оценивания могут помочь учителю в инспектировании и оценке программистских упражнений, а также помочь осуществлять формирующую обратную связь студентам в реальном времени.

Адаптивное формирующее оценивание в разных аспектах рассматривается в работах [21], [22], [23].

В работе [21] объясняется, что формирующее оценивание обеспечивает немедленную обратную связь с указанием на ошибки. Адаптивное оценивание учитывает предыдущее взаимодействие со студентом.

В работе [22] описывается система ProGrader проверки решений на тестах со встроенным контролем плагиата — от студента требуется писать строку за строкой в системе (копирование не срабатывает). После отсылки студент получает оценку и обратную связь. Поддерживается множество языков программирования, встроены средства отладки и профилирования. Обеспечен простой интерфейс для студентов и преподавателей.

В работе [23] описывается система автоматического формирующего оценивания ArTeMis, обеспечивающая быструю и полезную студентам обратную связь, что обеспечивает результативность системы. Еженедельно студентам открывается несколько заданий на программирование. Они должны решать задачи с помощью системы. В конце недели проводится видео-конференция по обсуждению заданий с участием преподавателя.

В работе [24] описывается новая дружественная к пользователю система для создания и поддержки автоматического оценивания. Центральным компонентом системы является интерфейс между преподавателем и автоматически оцениваемым кодом. Существенно сокращено время и трудозатраты на создание заданий для оценивания. За 3 месяца было создано 100 заданий и сэкономлено более 170 часов преподавателя.

В работе [25] представлена система оценивания претендентов на учительство в штате Иллинойс, США. Для получения лицензии на педагогическую деятельность, необходимо сдать два экзамена:

Educative Teacher Performance Assessment (edTPA) — проверка как педагога

Illinois Licensure Testing System (ILTS) — проверка знаний по специальности

Имеются сотни различных по содержанию ILTS-экзаменов для различных специальностей.

**Развитие обучения с помощью GenAI**

В работе [26] отмечается перспективность создания заданий на программирование с помощью GenAI. Работы может выполняться в следующих направлениях: генерация различных текстовых описаний одного формального задания; генерация вариаций задания, предложенного преподавателем. Такие автоматизированные подходы к формированию заданий существенно экономят время преподавателей, что особенно критично для «больших» потоков студентов, когда одновременно предмет изучают несколько сотен студентов.

В работе [27] описаны три экспериментальных подхода к генерации упражнений на программирование с помощью ChatGPT для большого курса по Python преподавателями University of Turku (Finland):

1. Новые упражнения с указанием темы.

2. Варианты существующих упражнений изменяя тему или содержание.

3. Гибридные упражнения на основе имеющихся добавляя темы или другие упражнения Все три подхода имеют потенциал и ограничения.

В работе [28] представлен опыт преподавателя Undergraduate Physics Students University of Mataram (Indonesia). На начальной стадии были использованы различные AI-модели, чтобы сгенерировать множество программ от простых до сложных. Затем эти программы были проверены с использованием онлайн Паскаль компилятора (gdb). Программы без

ошибок были интегрированы в процесс обучения. Во время лекции студентам рассказывались теоретические основы вычислительной физики и приводились сгенерированные Паскаль-программы как практические примеры. Эти программы служили для студентов основой, на которой они разрабатывали собственные программы. Задания последовательно усложнялись в течение семестра.

Работа [29] исследует отношение и опыт 208 студентов Киевского технического университета, которые по собственной инициативе использовали ChatGPT. Студенты поделились на 3 группы: те, кому работа с GenAI существенно помогла в изучении предмета; те, кому работа с GenAI помогала эпизодически; те, кому работа с GenAI не помогла совсем. По мнению авторов, фактически эти группы коррелируют с уровнем общей и специальной предварительной подготовки студентов. В первую группу попали наиболее подготовленные студенты, в третью — наименее подготовленные студенты.

Работа [30] анализирует опыт использования GenAI студентами по рекомендациям преподавателей. В этом случае наименее подготовленным студентам становится проще воспользоваться помощью GenAI, поскольку фактически преподаватели указывают, что и для чего нужно спрашивать у GenAI.

**Формирующее оценивание с помощью GenAI**

Работа [31] вводит общие понятия формирующего оценивания с помощью GenAI.

Слово «формирующее» здесь указывает на то, что в результате прочтения результатов такого оценивания у студента формируется более качественное представление об изучаемом предмете и его отдельных деталей, что служит основой к упрощению последующего обучения.

Оценка возможностей GenAI для формирующего оценивания с помощью GenAI приводится в работах [32], [33].

В работе [32] описывается процесс и результат проверки способности GenAI выполнять работу вместо студентов, по 10 инженерным предметам, один из которых — программирование. В общем это перспективный способ оценки возможностей различных GenAI — предлагать им выполнять задания для студентов и оценивать результаты, теми же подходами, которыми оцениваются работы студентов. Это важно в том числе и для того, чтобы можно было понимать в перспективе в какой степени можно полагаться на GenAI в качестве ассистента учителя.

Работа [33] констатирует, что развитие GenAI поставило вопрос о будущем компьютерного образования. Первая реакция была противоречивой — от «конец программирования» до «преждевременный некролог программированию». От «надо обнаруживать написанное GenAI и наказывать», до «Ну и что, если это написал GenAI?». Авторы рассматривают проблему с трёх ракурсов:

1. Собственно, разработки программного обеспечения.
2. Компьютерного образования в целом.
3. Сравнение компьютерного образования с практикой разработки программного обеспечения.

Главный вывод — разрыв в отношении к GenAI между компьютерным образованием и практикой разработки программного обеспечения.

Работа [34] агитирует за развитие самооценивания с помощью GenAI, настаивая на том, что надо учить студентов

- делать запросы;
- мыслить критически, анализируя ответы;
- отлаживать программы, полученные от GenAI, самостоятельно или с его помощью.

Непосредственное использование GenAI для формирующего оценивания представлено в работах [35], [36].

В работе [35] указывается, поддерживать формирующее оценивание для большого количества студентов практически невозможно, набор дополнительных ассистентов преподавателей ограничен бюджетами вузов, поэтому весьма перспективным выглядит использование GenAI для выполнения формирующего оценивания работ студентов.

Авторы работы [36] отмечают, что GenAI продемонстрировал возможности завершать код, создавать документацию по коду, поддерживать отладку. Они исследовали возможности GenAI по обратной связи на студенческие решения, направляющей студентов на улучшение написанного кода. Для этого они разработали промпты к GPT4, которые

1) обучают, что сделали студенты правильно или неправильно;
2) контролируют, как студенты учитывают предыдущие ответы при новой посылке.

В работе [37] предлагается использовать GenAI для анализа исходных текстов решений студентов, поскольку учитель не может обеспечить Code Review всем студентам сразу. Системы автоматической проверки только указывают на наличие ошибки. GenAI можно использовать для пояснения, в чём заключается ошибка. Одновременно можно использовать его для проверки не было ли попытки обмана со стороны студента (написания проверяемого кода с помощью GenAI). Чтобы обеспечить хорошую обратную связь, были разработаны специальные препромпты, обеспечивающие ответам GenAI точность, полезность, дружелюбный тон, обучающий эффект. Система обеспечивает студентов банком заданий на программирование. После того как студент выбрал задание и послал решение на проверку, система показывает корректность кода и code review — комментарии на полученный код. Обучение ведётся на языке программирования Python. При генерации обратной связи используется авторское решение. Оно посылается GenAI вместе с условием задачи (включающим пример ввода-вывода) и решением студента.

В работе [38] описывается среда разработки программ со встроенными запросами/ответами к ChatGPT. Авторы предлагают обучать студентов в низкоуровневой среде кодогенерации, развивая не только их умение кодировать, но также декомпозировать и делать запросы к GenAI. Ключевая идея предлагаемого подхода — использовать интеллектуальную разработку запросов, чтобы обучать студентов алгоритмическому мышлению и декомпозиции, комбинируя их с кодогенерацией и собственно кодированием.

Сначала студенты изучают основы декомпозиции задачи, так что они могут работать с программой на уровне функций. Затем для данной задачи студент описывает необходимый алгоритм на естественном языке, используя интеллектуальную разработку запроса. Имеется ввиду, что запрос может содержать код, если студент уже знаком с

некоторыми концепциями программирования. Студент может сгенерировать по запросу свою программу для частично описанного блока и увидеть текущую версию сгенерированного кода в секции черновика с указанием на ошибки в ней. Важно, что ошибки подсвечиваются в обоих блоках и запроса, и кода. Студент может исправлять ошибки в любой из секций. Наконец, студент может выполнить проверку своего решения, сгенерировав полностью код и добавив тесты к задаче. Для пилотного проекта авторы использовали язык программирования Kotlin, интегрируя своё решение в IDE-плагин JetBrains Academy.

В работе [39] указывается на важность и полезность развития критического мышления при создании программ с помощью GenAI. На сегодня GenAI не всегда справляется с предлагаемыми студентам заданиями, и поэтому важно, чтобы студенты, получая ответ от GenAI оценивали его критически, а в случае неудовлетворительности результата, формулировали эффективные новые запросы, продвигающие к получению правильного конечного результата.

Работа [40] посвящена сравнению возможностей различных GenAI в поиске ошибок и рекомендациях по их исправлению. Авторы указывают, что сегодня при обучении программированию уже используются инструменты, уточняющие сообщения об ошибках, помогающие вести отладку в реальном времени, объясняющие код, рекомендующие следующие шаги в разработке.

Авторов интересовали ответы на 2 вопроса:

1. Сравнение результативности открытых и закрытых языковых моделей в объяснении ошибок и предложениях по их исправлению.

2. В какой степени модели с открытым и закрытым исходным кодом могут оценить качество обратной связи по программированию, генерируемой другими моделями, по сравнению с экспертным человеческим мнением?

Чтобы ответить на первый вопрос, они сгенерировали объяснение ошибок и предложения по их исправлению с помощью пяти моделей с открытыми исходниками и двух моделей с закрытыми исходниками. Авторы вручную проанализировали объяснения и предложения по исправлению ошибок. Чтобы ответить на второй вопрос, авторы использовали ответы, полученные при ручной оценке. Модели с открытыми исходниками (бесплатные) показали сравнимое качество с проприетарными моделями.

Авторы использовали данные из Socratic guidance benchmark, который состоит из 57 заданий студентам по написанию функций. Каждое задание снабжено тестами для проверки и уникальным некорректным студенческим решением, описанием ошибок в нём и требуемых исправлений, а также нескольких бесед между «придуманным» студентом и преподавателем.

Конечная цель этого бенчмарка — оценивать способность моделей помогать студентам, используя сократический метод, т.е. направляя студента серией вопросов, помогая ему размышлять.

Наконец, работа [41] посвящена сокращению количества запросов к GenAI при массовом обучении. Авторы использовали FalconCode. Это набор из нескольких тысяч решений (и правильных, и неправильных) задач на Python. Каждая задач снабжена тестами, на которых проверялись решения, также метаданными типа уровня сложности задания. Авторы взяли только последние неправильные решения каждого студента за один семестр. Получилось 370 программ на 44 задачи. По каждому решению был отправлен такой запрос (промпт) к GPT, содержащий условие задачи, решение студента и запрос типа: Вы преподаватель вводного курса по программированию на Python, ниже представлено условие задачи и некорректное её решение студентом (т.е. решение не прошло все тесты). Ваша задача — дать короткое объяснение, какой следующий шаг должен предпринять студент, чтобы исправить свой код. Релевантность советов была оценена самим же GPT4 с помощью запросов такого вида: Вы преподаватель вводного курса по программированию на Python, ниже представлено условие задачи и некорректное её решение студентом (т.е. решение не прошло все тесты), также приведено короткое объяснение, какой следующий шаг должен предпринять студент, чтобы исправить свой код. Ваша задача – оценить полезность объяснения числом от 0 до 2.

- 0 — бесполезный совет;
- 1 — частично полезный совет;
- 2 — полезный совет.

Советы, признанные полезными, включены в банк данных и выдаются студентам без обращения к GenAI. Если же совет не помог, включается обращение к GenAI.

**Заключение**

В данной статье изложено содержание, задания и методология обучения программированию во вводных университетских предметах, традиционно используемые инструменты обучения, системы и способы оценивания. А затем приводится описание направлений использования генеративного искусственного интеллекта (GenAI) для повышения качества обучения программированию и изучения программирования во вводных университетских предметах. Особенное внимание акцентировано на формирующем оценивании, главная цель которого помочь студенту быстро и качественно учиться на своих ошибках. Цель выполненного исследования — помочь вузовским преподавателям оптимально модифицировать содержание, задания и методологию обучения в своих курсах начального обучения программирования с учётом мирового опыта и тенденций развития и использования GenAI. Результат исследования — максимально полный обзор статей по соответствующей тематике.

**Конфликт интересов**

Не указан.

**Рецензия**

Все статьи проходят рецензирование. Но рецензент или автор статьи предпочли не публиковать рецензию к этой статье в открытом доступе. Рецензия может быть предоставлена компетентным органам по запросу.

**Conflict of Interest**

None declared.

**Review**

All articles are peer-reviewed. But the reviewer or the author of the article chose not to publish a review of this article in the public domain. The review can be provided to the competent authorities upon request.

**Список литературы на английском языке / References in English**

1. Bekkering T.E. A Comparison of Generative AI Solutions and Textbook Solutions in an Introductory Programming Course / T.E. Bekkering, P. Harrington // Information Systems Education Journal. — 2025. — № 23 (1). — P. 4–22. — DOI: 10.62273/YQWP1758.

2. Saarivuori R. Creation and Management of Small Programming Assignments to Combat Plagiarism Master's thesis / R. Saarivuori. — Lappeenranta, 2023. — 72 p.

3. Yuan X. Multi-method integrated experimental teaching reform of a programming course based on the OBE-CDIO model under the background of engineering education / X. Yuan, J. Wan, D. An [et al.] // Sci Rep. — 2024. — № 14. — P. 16623. — DOI: 10.1038/s41598-024-67667-6.

4. Sychev O. Searching Questions and Learning Problems in Large Problem Banks: Constructing Tests and Assignments on the Fly / O. Sychev // Computers. — 2024. — Vol. 13. — № 6. — P. 144. — DOI: 10.3390/computers13060144.

5. Dung T.H. Some Solutions to Improve Programming Skills for Information Technology Students at Tan Trao University / T.H. Dung // Eur. J. Appl. Sc. Eng. Technol. — 2024. — Vol. 2 (3). — P. 205–213. — DOI: 10.59324/ejaset.2024.2(3).19.

6. McKenzie S. Integrating Human-Centric Approaches into Undergraduate Software Engineering Education: A Scoping Review and Curriculum Analysis in the Australian Context / S. McKenzie, X. Lui. — 2024.

7. Gianinazzi M. Designing a Framework to Support the Teaching of Programming Basics to Large Numbers of Novices / M. Gianinazzi, P. Weidmann, L. Moccozet // Proceedings of the 16th International Conference on Computer Supported Education (CSEDU 2024). — 2024. — Vol. 2. — P. 597–604 — DOI: 10.5220/0012723100003693.

8. Perez-Rojas D. Adaptive mentoring with immediate feedback for the development of programming skills / D. Perez-Rojas, R. Paredes-Juarez, C. Perez-Lezama [et al.] // EDULEARN24 Proceedings. — 2024. — P. 4341–4348. — DOI: 10.21125/edulearn.2024.1086.

9. Mtaho A.B. Developing a CeliotM programming learning tool to facilitate teaching and learning data structure concepts in C++ for novice programmers / A.B. Mtaho, M.M. Masoud, L.J. Mselle // Information Technologies and Learning Tools. — 2024. — Vol. 101. — № 3. — P. 42–70. — DOI: 10.33407/itlt.v101i3.5567.

10. Rodriguez-Vidal J. C-programming self-assessment exercises versus final exams: 12 years of experience / J. Rodriguez-Vidal, R. Martinez, A. Garcia-Beltran // Comput. Appl. Eng. Educ. — 2023. — № 31. — P. 1272–1288. — DOI: 10.1002/cae.22639.

11. Sankaranarayanan R. Exploring the role of a microlearning instructional approach in an introductory database programming course: an exploratory case study / R. Sankaranarayanan, M. Yang, K. Kwon // J Comput High Educ. — 2024. — DOI: 10.1007/s12528-024-09408-2.

12. Messer M. Automated Grading and Feedback Tools for Programming Education: A Systematic Review / M. Messer, N.C.C. Brown, M. Kolling [et al.]. — 2023. — 43 p.

13. Ukkonen A. Teachers' understanding of assessing computational thinking / A. Ukkonen, K. Pajchel, L. Mifsud // Computer Science Education. — 2024. — P. 1–26. — DOI: 10.1080/08993408.2024.2365566.

14. Petrescu M. The Perceived Learning Behaviors and Assessment Techniques of First-Year Students in Computer Science: An Empirical Study / M. Petrescu, T. Mihoc // Proceedings of the 16th International Conference on Computer Supported Education (CSEDU 2024). — 2024. — Vol. 2. — P. 405–412. — DOI: 10.5220/0012674000003693.

15. Mammadli M. Analysis and Evaluation of the Contestant's Progress in Real-time Coding Contests / M. Mammadli, N. Mammadli, J. Hasanov // Olympiads in Informatics. — 2024. — Vol. 18. — P. 51–62.

16. Xiang C. Graph semantic similarity-based automatic assessment for programming exercises / C. Xiang, Y. Wang, Q. Zhou [et al.] // Sci Rep. — 2024. — № 14. — P. 10530. — DOI: 10.1038/s41598-024-61219-8.

17. Wang X.-M. Promoting students' creative self-efficacy, critical thinking and learning performance: An online interactive peer assessment approach guided by constructivist theory in maker activities / X.-M. Wang, X.T. Huang, Y.-H. Han, [et al.] // Thinking Skills and Creativity. — 2024. — Vol. 52. — DOI: 10.1016/j.tsc.2024.101548.

18. Mekterovic I. Interactive Programming Tutorials in Automated Programming Assessment System Edgar / I. Mekterovic, L. Brkic, M. Fertalj [et al.] // 2024 47th MIPRO ICT and Electronics Convention (MIPRO). — Opatija, 2024. — P. 218–223. — DOI: 10.1109/MIPRO60963.2024.10569406.

19. Barczak A.L.C. Automated assessment system for programming courses: a case study for teaching data structures and algorithms / A.L.C. Barczak, A. Mathrani, B. Han [et al.] // Education Tech Research. — 2023. — Dev. 71. — P. 2365–2388. — DOI: 10.1007/s11423-023-10277-2.

20. Chuang Y.-T. Analyzing novice and competent programmers' problem-solving behaviors using an automated evaluation system / Y.-T. Chuang, H.-Y. Chang // Science of Computer Programming. — 2024. — Vol. 237. — DOI: 10.1016/j.scico.2024.103138.

21. Thangaraj J. Adaptive Formative Assessment For Teaching Novices in Introductory Programming / J. Thangaraj // Proceedings of the 2023 Conference on United Kingdom & Ireland Computing Education Research (UKICER '23). — New York: Association for Computing Machinery, 2023. — Art. 30. — 1 p. — DOI: 10.1145/3610969.3610971.

22. Nafa F. Improving Educational Outcomes: Developing and Assessing Grading System (ProGrader) for Programming Courses / F. Nafa, L. Sreeramareddy, S. Mallapuram [et al.] // Mori, H., Asahi, Y. (eds) Human Interface and the Management of Information. HCII 2023. Lecture Notes in Computer Science / Ed. by H. Mori, Y. Asahi. — 2023. — Vol. 14016. — DOI: 10.1007/978-3-031-35129-7_24.

23. Sauerwein C. Towards a Success Model for Automated Programming Assessment Systems Used as a Formative Assessment Tool / C. Sauerwein, T. Antensteiner, S. Oppl [et al.] // Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1 (ITiCSE 2023). — Turku; New York, 2023. — 7 p. — DOI: 10.1145/3587102.3588848.

24. Muuli E. Reflecting on Simplification of the Creation and Maintenance of Automated Assessments for Programming Tasks / E. Muuli, M. Lepp, T. Palts [et al.] // 2024 IEEE Global Engineering Education Conference (EDUCON). — Kos Island, 2024. — P. 1–3. — DOI: 10.1109/EDUCON60312.2024.10578575.

25. Drenckpohl D. Analyzing Illinois Licensure Testing System ( Ilts ) Content Exam and Pearson's Educative Teacher Performance Assessment ( Edtpa ) Performance in Teacher Candidates Who Study Abroad / D. Drenckpohl. — 2024.

26. Alrifai R. Using Generative AI to Design Programming Assignments in Introduction to Computer Science / R. Alrifai // J. Comput. Sci. Coll. — 2024. — Vol. 39. — № 6. — P. 103–106. — DOI: 10.5555/3665464.3665476.

27. Rytilahti J. Exploring AI-driven programming exercise generation / J. Rytilahti, O. Weerakoon, E. Kaila // 2024 20th International CDIO Conference, ESPRIT. — Tunisia, 2024.

28. Taufik M. AI-Driven Optimization of Pascal Programming Instruction for Undergraduate Physics Students at University of Mataram / M. Taufik, M. Zuhdi, S. Ayub [et al.] // International Journal of Contextual Science Education. — 2024. — № 1 (3). — P. 85–88.

29. Chugai O. ChatGPT: Attitudes and experiences of technical university in Ukraine / O. Chugai, K. Havrylenko // Information Technologies and Learning Tools. — 2024. — Vol 101. — № 3. — DOI: 10.33407/itlt.v101i3.5559.

30. Khavasi A. Reimagining entrepreneurship education with G-AI: a Finnish vocational university case study / A. Khavasi, H. Paananen // EDULEARN24 Proceedings. — 2024. — P. 3877–3882. — DOI: 10.21125/edulearn.2024.0985.

31. Wiliam D. What is assessment for learning? / D. Wiliam // Studies in Educational Evaluation. — 2011. — № 37 (1). — P. 3–14. — DOI: 10.1016/j.stueduc.2011.03.001.

32. Nikolic S. ChatGPT, Copilot, Gemini, SciSpace and Wolfram versus higher education assessments: an updated multi-institutional study of the academic integrity impacts of Generative Artificial Intelligence (GenAI) on assessment, teaching and learning in engineering / S. Nikolic, C. Sandison, R. Haque [et al.] // Australasian Journal of Engineering Education. — 2024. — P. 1–28. — DOI: 10.1080/22054952.2024.2372154.

33. Sengul C. Software Engineering Education in the Era of Conversational AI: Current Trends and Future Directions / C. Sengul, R. Nevkova, G. Destefanis // Frontiers in Artificial Intelligence. — 2024. — Vol. 7. — DOI: 10.3389/frai.2024.1436350.

34. Reeves B.N. Prompts First, Finally / B.N. Reeves, J. Prather, P. Denny [et al.]. — 2024. — DOI: 10.48550/arXiv.2407.09231.

35. Zhai X. AI and formative assessment: The train has left the station / X. Zhai, R.H. Nehm // Journal of Research in Science Teaching. — 2023. — № 60 (6). — P. 1390–1398. —DOI: 10.1002/tea.21885.

36. Nguyen H. Comparing Feedback from Large Language Models and Instructors: Teaching Computer Science at Scale / H. Nguyen, N. Stott, V. Allan // Proceedings of the Eleventh ACM Conference on Learning @ Scale (L@S '24). — Atlanta; New York, 2024. — 5 p. — DOI: 10.1145/3657604.3664660.

37. Dong-Kyu L. A GPT-based Code Review System for Programming Language Learning / L. Dong-Kyu. — 2024.

38. Potriasaeva A. Using a Low-Code Environment to Teach Programming in the Era of LLMs / A. Potriasaeva, K. Dzialets, Y. Golubev [et al.] // ACM Conference on International Computing Education Research V.2 (ICER '24 Vol. 2). — Melbourne; New York, 2024. — 2 p.

39. Styve A. Developing Critical Thinking Practices Interwoven with Generative AI Usage in an Introductory Programming Course / A. Styve, O.T. Virkki, U. Naeem // IEEE Global Engineering Education Conference (EDUCON). — Kos Island, 2024. — P. 01–08. — DOI: 10.1109/EDUCON60312.2024.10578746.

40. Koutcheme C. Evaluating Language Models for Generating and Judging Programming Feedback / C. Koutcheme [et al.]. — 2024. — DOI: 10.48550/arXiv.2407.04873.

41. Koutcheme C. Propagating Large Language Models Programming Feedback / C. Koutcheme, A. Hellas // Proceedings of the Eleventh ACM Conference on Learning @ Scale (L@S '24). — Atlanta; New York, 2024. — 5 p. — DOI: 10.1145/3657604.3664665.