МРНТИ 14.35.09

10.51889/2959-5894.2023.84.4.025

Е.А. Киселева[1], Г.А. Абдулкаримова[1]*

[1]Казахский национальный педагогический университет имени Абая, г. Алматы, Казахстан
[*]e-mail: kisseleva@gmail.com

ГЕНЕРАЦИЯ УПРАЖНЕНИЙ ПО ОБЪЕКТНО-ОРИЕНТИРОВАННОМУ ПРОГРАММИРОВАНИЮ С ПОМОЩЬЮ CHATGPT

Аннотация

Статья посвящена изучению возможности разработки эффективных упражнений по объектно-ориентированному программированию с использованием ChatGPT, которые обеспечивали бы глубокое понимание и применение концепций объектно-ориентированного программирования, а также определения технологических этапов процесса их генерации. Используя OpenAI Copilot в качестве большой языковой модели, мы создаем упражнения по программированию (включая примеры решений и тестовые примеры), оценивая их качественно и количественно. Результаты исследования показывают, что большая часть автоматически генерируемого контента является одновременно новым и корректно сформулированным, а в некоторых случаях полностью готовым к использованию. Для генерации упражнений были разработаны прайминги в качестве входных данных для Copilot. Они представляют собой шаблоны, описывающие концепции объектно-ориентированного программирования, структуры данных и ключевые слова для формулировки условий задач. Наш анализ показывает, что массовые модели генеративного машинного обучения представляют значительную ценность в качестве инструмента для преподавателей, хотя по-прежнему существует необходимость в некотором контроле для обеспечения качества сгенерированного контента до того, как он будет предоставлен учащимся.

Ключевые слова: ChatGPT, объектно-ориентированное программирование, генеративный искусственный интеллект, компетенции**.**

Е.А. Киселева[1], Г.А. Абдулкаримова[1]
[1]Абай атындағы Қазақ ұлттық педагогикалық университеті, Алматы қ., Қазақстан

CHATGPT КӨМЕГІМЕН ОБЪЕКТІЛІ-БАҒЫТТАЛҒАН ПРОГРАММАЛАУ БОЙЫНША ЖАТТЫҒУЛАР ГЕНЕРАЦИЯСЫ

Аңдатпа

Мақалада ChatGPT көмегімен объектілі-бағытталған программалау концепцияларын терең түсінуді және қолдануды, сондай-ақ оларды генерациялау процесіндегі технологиялық қадамдарды анықтауды қамтамасыз ететін тиімді объектілі-бағытталған программалау жаттығуларын әзірлеу мүмкіндігі қарастырылады. OpenAI Copilot-ті үлкен тіл үлгісі ретінде пайдалана отырып, біз оларды сапалы және сандық тұрғыдан бағалай отырып, программалау жаттығуларын (үлгі шешімдер мен сынақ жағдайларын қоса) жасаймыз. Зерттеу нәтижелері көрсеткендей, автоматты түрде генерация жасалған мазмұнның көпшілігі жаңа және жақсы тұжырымдалған, ал кейбір жағдайларда пайдалануға толығымен дайын. Жаттығуларды генерациялау үшін Copilot бағдарламасына кіріс ретінде прайминғтер әзірленді. Олар объектілі-бағытталған программалау тұжырымдамаларын, деректер құрылымдарын және проблемалық мәлімдемелерді құрастыруға арналған түйінді сөздерді сипаттайтын шаблондар. Біздің талдауымыз көрсеткендей, негізгі генеративті машиналық оқыту үлгілері оқытушыларға арналған құрал ретінде маңызды құндылықты ұсынады, дегенмен жасалған мазмұнның студенттерге жеткізілуіне дейін оның сапасын қамтамасыз ету үшін әлі де біршама қадағалау қажет.

Түйін сөздер: ChatGPT, объектілі-бағытталған программалау, генеративті жасанды интеллект, құзыреттер.

E.A. Kiseleva[1], G.A. Abdulkarimova[1]
[1] Abai Kazakh National Pedagogical University, Almaty, Kazakhstan

GENERATING OBJECT-ORIENTED PROGRAMMING EXERCISES USING CHATGPT

Abstract

This article is devoted to the study of the possibility of developing effective exercises in object-oriented programming using ChatGPT, which would provide a deep understanding and application of OOP concepts, as well as determining the technological stages of the process of their generation. Using OpenAI Copilot as a large language model, we create programming exercises (including sample solutions and test cases), evaluating them qualitatively and quantitatively. Our results show that most of the automatically generated content is both new and correctly formulated, and in some cases completely ready for use. To generate exercises, primings were developed as input data for Copilot. They are templates describing the concepts of object-oriented programming, data structures and keywords for the formulation of task conditions. Our analysis shows that mass models of generative machine learning are of considerable value as a tool for teachers, although there is still a need for some control to ensure the quality of generated content before it is provided to students.

Keywords: ChatGPT, object-oriented programming, generative artificial intelligence, competencies**.**

Введение

Объектно-ориентированное программирование (ООП) является ключевым компонентом в современном программировании, и его понимание критически важно для разработчиков. Однако обучение ООП может быть сложным из-за его абстрактной природы и сложности концепций, таких как наследование, полиморфизм и инкапсуляция. Одной из основных проблем обучения программированию является создание актуальных, стимулирующих и увлекательных заданий, которые студенты должны выполнить. Традиционно задачи по программированию создаются преподавателем вручную, что требует глубоких знаний и солидного опыта в этой области. При создании таких заданий необходимо учитывать множество факторов, таких как начальный уровень навыков студентов, уровень навыков учащихся по мере прохождения курса и соответствие разработанных упражнений модели компетенций в ООП, чрезвычайно важных для успешного обучения на последующих курсах. Эта задача очень сложная, и требует существенных затрат времени.

В традиционной академической среде значительные усилия, необходимые для разработки высококачественных заданий по объектно-ориентированному программированию, часто приводят к тому, что преподаватели курса используют один набор типовых заданий для всех студентов, стремясь с их помощью оценить степень усвоения материала и сформированности необходимых компетенций. Но такой подход позволяет некоторым студентам выполнять задания без должного понимания концепций ООП. Они заучивают решения, не вникая в то как оно получено или даже копируют работы своих сокурсников [1]. Несмотря на то, что распространенность обмана сильно варьируется в зависимости от большого количества факторов, случаи обмана, как сообщается, тревожно высоки, достигая 83% в определенных случаях [2]. Студенты, прибегающие к копированию чужих работ вместо того, чтобы получить ценную практику, будут в дальнейшем испытывать серьезные затруднения в процессе своего обучения. Это является одной из самых главных причин неудач при изучении языков программирования [3].

С появлением искусственного интеллекта (ИИ) и машинного обучения, образовательная сфера стала свидетелем революции в методах преподавания и обучения. Одним из таких прорывов является ChatGPT [4], модель генерации текста, разработанная OpenAI, которая использует технологию трансформеров для создания связного и информативного текста на основе предоставленного ввода. Очевидный успех таких больших языковых моделей в генерации текста привел к росту интереса к их использованию для широкого круга приложений, включая генерацию задач для программирования. Использование ChatGPT для создания практических заданий и упражнений представляет собой привлекательную перспективу, поскольку он может обеспечить высокий уровень вариативности и креативности

в задачах, одновременно демонстрируя эффективность времени и масштабируемость. Исходя из этого, генерация задач по программированию, содержащих схожие понятия, но с индивидуальными вариациями, может принести педагогические преимущества и одновременно снизить риск академической нечестности. При предоставлении всем ученикам уникальных заданий возможность дословного копирования становится невозможной и поощряется реальное обучение.

Однако, несмотря на эти преимущества, использование ChatGPT также вызывает определенные проблемы, поскольку студенты тоже могут использовать ChatGPT для получения готовых ответов на задания. Это вызывает необходимость формулировать задачи и упражнения таким образом, чтобы можно было преодолеть возможности ChatGPT.

Таким образом, цель данного исследования состоит в определении технологических этапов процесса разработки эффективных упражнений по ООП с использованием ChatGPT, которые обеспечивали бы глубокое понимание и применение концепций ООП, минимизируя при этом возможность недобросовестного использования искусственного интеллекта.

Материалы и методы

В рамках исследования была проведена оценка сгенерированных упражнений по программированию смешанными методами: качественного и количественного анализа. Для качественного анализа была сформирована случайная выборка из 80 упражнений, которую проверили на логичность формулировок, новизну и готовность к использованию в учебном процессе. При оценивании описания предметной области задачи анализировалась доступность для понимания учащимися, насколько постановка задачи отражает практическую задачу с методами решения которой учащиеся знакомы. Для выявления новизны, проводился поиск в Google и в GitHub с целью выявления существующих аналогичных упражнений или дословных формулировок упражнений. Также рассматривалась адекватность сгенерированных упражнений для применения различных концепций, заданных в прайминге. Оценка готовности к использованию включала анализ объема ручной работы, которую требуется выполнить преподавателю для реализации упражнений, а также создания образцов решений и тестов, связанных с ними. Для качественного анализа сгенерированные упражнения были проверены в среде программирования на соответствие требованиям. Результаты качественного и количественного анализа были использованы для формирования выводов исследования, обеспечивая надежность и объективность наших исследовательских данных.

Результаты и обсуждение

Литературный обзор. Анализ литературных источников показывает, что ChatGPT может быть эффективным инструментом для преподавания, в том числе и ООП. В частности, он может быть использован для объяснения концепций ООП. ChatGPT может генерировать текст, который объясняет сложные концепции ООП простым и понятным языком. Также его можно применять для предоставления обратной связи студентам по их работе, что может помочь им улучшить свои навыки программирования. ChatGPT может быть использован для проведения практических занятий по ООП, таких как разработка небольших типовых программ.

Исследования показывают, что ChatGPT может быть эффективным инструментом для генерации учебных заданий по ООП. В частности, он может быть использован для генерации заданий, которые соответствуют уровню подготовки студентов, основываясь на их предыдущих работах. Это может помочь обеспечить посильность предлагаемых упражнений. Они должны быть достаточно сложными, чтобы стимулировать обучение, но не настолько сложными, чтобы они стали невыполнимыми. В исследовании Al-Jabri и Al-Khalifa [5] было обнаружено, что использование ChatGPT для генерации учебных заданий по ООП привело к повышению успеваемости студентов. В исследовании сделан вывод, что студенты, которые использовали задания, сгенерированные ChatGPT, получали более высокие оценки, чем

студенты, которые использовали задания, разработанные преподавателями. Также ChatGPT может генерировать задания, которые охватывают широкий спектр компетенций в сфере ООП. Это может помочь студентам изучить все основные концепции ООП. Cetin и Turkmen [6] и Kumar и Kaushik [7] предоставили данные о том, что студенты, которые использовали задания, сгенерированные ChatGPT, сообщали о более высокой удовлетворенности обучением и вовлеченности в процесс обучения, чем студенты, которые использовали типовые задания.

В целом, исследования показывают, что использование ChatGPT для генерации учебных заданий по ООП может быть эффективным инструментом для повышения успеваемости, удовлетворенности обучением и вовлеченности студентов. Однако ChatGPT не является заменой преподавателя. Преподаватели должны продолжать играть активную роль в разработке и оценке учебных заданий, сгенерированных ChatGPT.

Важно отметить, что при разработке упражнений по программированию необходимо учитывать модель компетенций будущего специалиста. Для того, чтобы эффективно программировать на ООП, необходимо обладать определенными компетенциями, которые включают в себя как базовые понятия и концепции ООП, так и более сложные навыки, такие как проектирование и реализация сложных программных систем.

В области ООП существуют различные модели компетенций, которые помогают определить и измерить навыки и знания, необходимые для успешного выполнения задач программирования. Одной из таких моделей является “Структурная модель компетенций объектно-ориентированного программирования”, предложенная в проекте COMMOOP [8]. Эта модель была разработана на основе обзора существующей литературы по моделированию компетенций в других предметных областях, а также на основе анализа теоретических и эмпирических исследований по обучению и психологическим аспектам в области ООП.

В результате были выявлены четыре потенциальных измерения компетенций:

– *знание и навыки ООП*, это измерение включает в себя знания и навыки, связанные с базовыми понятиями и концепциями ООП, такими как классы, объекты, интерфейсы, наследование, полиморфизм, абстракция, инкапсуляция и композиция;

– *освоение представления,* это измерение включает в себя знания и навыки, связанные с представлением ООП, такими как язык, синтаксис, семантика;

– *когнитивные процессы,* это измерение включает в себя когнитивные процессы, связанные с решением проблем на основе ООП, такие как понимание проблемы, определение способа решения проблемы, перевод проблемы в программу на компьютерном языке, тестирование и отладка программы;

– *метакогнитивные процессы*, это измерение включает в себя метакогнитивные процессы, связанные с обучением ООП, такие как управление временем, планирование, саморегуляция и рефлексия.

Другой моделью компетенций ООП являются рекомендации CC2020 (Common Core Computing Curriculum) [9]. Эти рекомендации определяют следующие компетенции в области ООП:

– *Основы ООП*, понимание базовых понятий и концепций ООП, таких как классы, объекты, интерфейсы, наследование, полиморфизм, абстракция, инкапсуляция и композиция.

– *Проектирование ООП*, понимание принципов ООП, таких как принцип единого назначения, принцип открытости/закрытости, принцип инкапсуляции, принцип полиморфизма и принцип суперкласса.

– *Реализация ООП*, навыки разработки классов и объектов на основе базовых понятий и концепций ООП, а также навыков использования наследования и полиморфизма для разработки сложных программных систем.

На основе рекомендаций CC2020 компетенции в области ООП можно разделить на следующие уровни сложности:

Основной уровень – компетенции, связанные с базовыми понятиями и концепциями ООП.

Продвинутый уровень – компетенции, связанные с принципами объектно-ориентированного проектирования и реализацией классов и объектов.

Мастерский уровень – компетенции, связанные с использованием наследования и полиморфизма для разработки сложных программных систем и тестированием и сопровождением программных систем, разработанных на ООП.

Существует множество различных моделей компетенций ООП, которые могут быть использованы для определения и измерения степени усвоения материала курса. Для эффективной разработки программного обеспечения необходимо обладать компетенциями как на базовом, так и на продвинутом и мастерском уровнях. При правильном подходе к разработке тренировочных и оценочных заданий можно эффективно проверить компетенции студентов в области ООП и помочь им улучшить свои навыки программирования.

При этом недобросовестное использование ИИ студентами в процессе выполнения заданий ООП является актуальной проблемой, которая может привести к снижению качества образования. Оценке рисков при использовании ChatGPT посвящено множество публикаций [10-15]. Для минимизации недобросовестного использования ИИ студентами при выполнении заданий по ООП предлагается принимать следующие меры:

– Прозрачность оценивания, т.е. студенты должны знать, что ИИ может использоваться для проверки их работ. Это поможет им понять, что использование ИИ для недобросовестного выполнения заданий может быть обнаружено.

– Оценка навыков, которые студенты должны приобрести в ходе обучения, а не просто их способность генерировать код. Например, задание может оценивать способность студентов анализировать проблему, проектировать решение и реализовывать его.

– Индивидуализация, задание должно быть адаптировано к уровню подготовки конкретного студента. Это поможет предотвратить ситуацию, когда студент может использовать ИИ для выполнения задания, которое он не в состоянии выполнить самостоятельно.

Исследования показывают, что использование этих подходов может быть эффективным для минимизации недобросовестного использования ИИ студентами при выполнении заданий по ООП. Однако важно отметить, что эти подходы не являются универсальными и должны адаптироваться к конкретным условиям обучения.

Использование ChatGPT. Для генерации упражнений нами использовались чат-боты GPT-3,5 от OpenAI, Copilot от Microsoft и Bard от Google. Доступ к ним осуществлялся через веб-интерфейс. Пользователь предоставляет прайминг, т.е. подсказку, чат-боту в качестве входных данных, и чат-бот генерирует новый контент в качестве выходных данных на основе заданного прайминга. Например, при наличии описания желаемого поведения на естественном языке чат-бот может сгенерировать исходный код для программы, предоставляющей эту функциональность. Поскольку прайминг, предоставляемый чат-боту, подготавливает шаблон упражнения, которое должно быть сгенерировано, мы можем настроить чат-бот, задав ему в качестве образца существующее упражнение по объектно-ориентированному программированию. Кроме того, есть возможность описать словами новую предметную область для формулировки условия задачи. Это помогает чат-боту сгенерировать задание, похожее на шаблонное упражнение, но с заданным контекстом (рисунок 1).

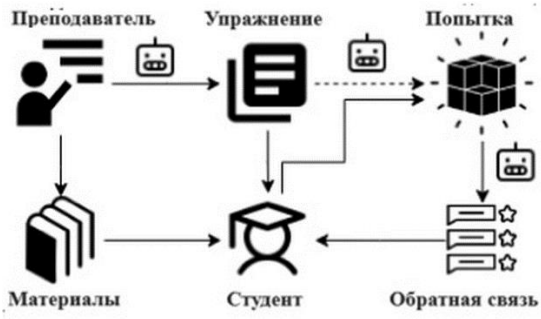

Рисунок 1. Жизненный цикл упражнения по программированию

Создание упражнений по объектно-ориентированному программированию.

Выбор входных данных для ChatGPT. Одной из основных задач нашего исследования является определение, разграничение и уточнение компетенций для объектно-ориентированного программирования. Связь между компетенциями, их компонентами и задачами (элементами) имеет существенное значение для валидации компетенций. Поэтому мы проанализировали структуру типовых заданий и примеры программных решений студентов, чтобы определить необходимые навыки с предметно-познавательной точки зрения. Анализ включал только навыки, необходимые для программирования конкретной типовой задачи. Во многих случаях существуют альтернативные решения (например, итерационное решение вместо рекурсивного или наоборот). Такие решения будут считаться ошибочными.

Курс «Объектно-ориентированное программирование» является обязательным в пятом семестре для программ бакалавриата 6В01507-«Информатика». В нем рассматриваются темы и формируются соответствующие компетенции, представленные в Таблице 1.

Таблица 1. Структура компетенций курса

Темы	*Компетенции*
Принципы ООП	*Понимание концепции классов и их использования в объектно-ориентированном программировании.* *Способность объяснить, как классы обеспечивают модульность и повторное использование кода.*
Классы, объекты, экземпляры классов в Python	*Способность определить и создать классы и объекты.* *Понимание того, как экземпляры классов представляют конкретные объекты в коде.*
Инкапсуляция. Методы и атрибуты классов	*Понимание того, как атрибуты класса представляют состояние объекта.* *Способность определить и использовать атрибуты класса в коде.* *Способность использовать конструкторы для инициализации экземпляров классов.* *Понимание различия между атрибутами класса и атрибутами экземпляра.*
Геттеры и сеттеры, property атрибуты	*Понимание, как использовать геттеры и сеттеры для управления доступом к атрибутам класса.* *Способность правильно использовать атрибут property для создания геттеров и сеттеров в коде.*
Наследование	*Понимание принципа наследования: возможности одному классу наследовать атрибуты и методы другого класса.* *Способность применять наследование в коде, создавая иерархии классов и используя наследование для повторного использования кода*
Переопределение методов	*Понимание механизма переопределения методов, того, когда и как следует переопределять методы.*

	Способность переопределять методы в производных классах, корректно использовать функции super для вызова методов базового класса
Полиморфизм	*Понимание, что такое полиморфизм, какие виды полиморфизма существуют в Python.* *Умение использовать полиморфизм в соответствии с заданной задачей.* *Способность анализировать код, в котором используется полиморфизм.*
Исключения	*Понимание концепции исключений в Python.* *Способность объяснить, какие виды исключений бывают.* *Способность применять механизм исключений на практике.*

Создание упражнений по программированию. С целю отработки технологии генерации упражнений по объектно-ориентированному программированию с использованием ChatGPT мы ограничились тремя типами упражнений, позволяющих сформировать основные компетенции в ООП.

Задание 1 типа

--Формулировка условия задачи--

Описать класс со следующими характеристиками:

- конструктор с параметрами по умолчанию
- деструктор, который выводит сообщение об уничтожении объекта
- метод для вычисления на основе значений атрибутов
- метод для формирования строки, которая представляет информацию об объекте.

В основной программе

- должен быть создан список из нескольких экземпляров этого класса
- должна быть продемонстрирована работа всех методов класса
- динамически определить новый атрибут для экземпляра класса и продемонстрировать его использование
- текст задания не должен содержать названий классов, атрибутов и методов (студент должен придумать их сам)
- условие задачи должно описывать реальную проблему (кейс)
- текст задачи должен быть связный в одном абзаце
- ключевое слово для кейса: "футбол| музыка| здоровье| хоккей| книги| кулинария| животные| растения| университет| хобби| космос"

[Повышение сложности задачи:

- описать еще один метод
- добавить статический атрибут класса
- продемонстрировать их в основной программе.]

--Пример решения на Python--

--Автоматические тесты--

Задание 2 типа

--Формулировка условия задачи--

Описать класс со следующими характеристиками:

- публичные, приватные атрибуты и методы
- геттеры
- сеттеры
- декоратор property
- вычисляемые свойства
- содержит приватный атрибут, который хранит закрытую информацию, доступ к которой ограничен

- метод, который возвращает закрытую информацию, при выполнении определенного условия, иначе вызывает исключение
- продемонстрировать их в основной программе, выведите закрытую информацию
- текст задания не должен содержать названий классов, атрибутов и методов (студент должен придумать их сам)
- условие задачи должно описывать реальную проблему (кейс)
- текст задачи должен быть связный в одном абзаце
- ключевое слово для кейса: "поход| рыбалка| семья| футбол| музыка| здоровье| хоккей| книги| кулинария| животные| растения| университет| хобби| космос"

--Пример решения на Python--

--Автоматические тесты--

Задание 3 типа

--Формулировка условия задачи--

Описать иерархии классов со следующими характеристиками:
- описать две иерархии классов, которые взаимодействуют друг с другом
- один из атрибутов должен быть: "словарем| множеством| кортежем| списком"
- описать расширение класса каким-то методом
- переопределить какой-нибудь метод при наследовании
- предусмотреть делегирование для некоторого метода используя функцию super()

В основной программе
- должны быть созданы списки из нескольких экземпляров разных классов каждой иерархии
- должна быть продемонстрирована работа всех методов класса
- написать внешнюю функцию, которой передаются в качестве параметров экземпляры классов из первой и второй иерархии
- добавить: взаимодействие между объектами| композицию| агрегацию| расширить иерархию классов| использовать полиморфизм| обработку исключений
- Текст задания не должен содержать названий классов, атрибутов и методов. Студент должен придумать их сам.
- Условие задачи должно описывать реальную проблему (кейс). Стиль описания должен быть художественный.
- ключевое слово для кейса: " университет| хобби| космос| музыка| здоровье| хоккей| книги| кулинария| поход| рыбалка| семья| футбол".

--Пример решения на Python--

--Автоматические тесты

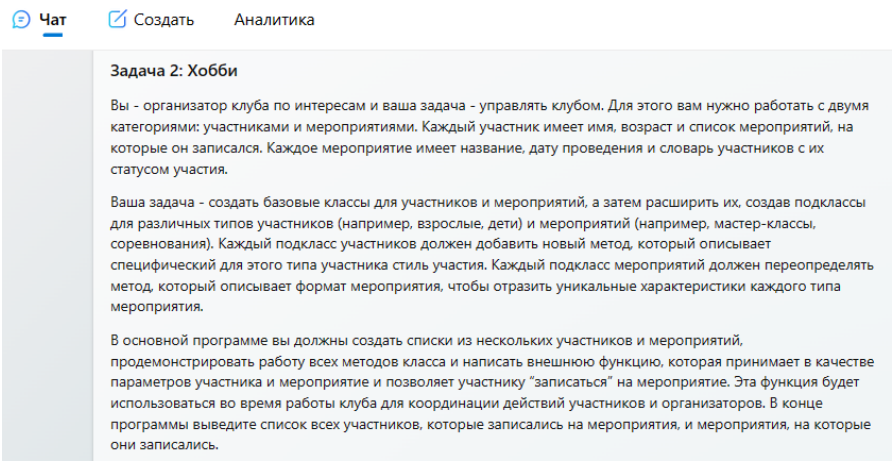

Рисунок 2. Сгенерированная в Copilot формулировка условия задачи 3 типа

Разработанная структура прайминга для генерации упражнений по объектно-ориентированному программированию содержит описание задачи, пример ее решения на Python и автоматические тесты. Мы предусмотрели возможность варьировать в прайминге концепции, связанные со структурами данных (например, словари или множества), взаимодействие между объектами (композицию, агрегацию и др.) и тематическую область задачи (например, университет, хобби, космос). С помощью описанных праймингов в Copilot были сгенерированы наборы заданий трех типов. В общей сложности было разработано 12×10×3 = 360 различных комбинаций входных данных (контекстуальные понятия × наборы понятий, связанных с программированием, × прайминги упражнений). Пример сгенерированного упражнения по объектно-ориентированному программированию приведен на рисунках 2-5.

```python
class Participant:
    def __init__(self, name, age):
        self.name = name
        self.age = age
        self.events = []

    def participate(self, event):
        self.events.append(event)
        event.participants[self.name] = 'Not confirmed'

class Adult(Participant):
    def confirm_participation(self, event):
        if event in self.events:
            event.participants[self.name] = 'Confirmed'

class Child(Participant):
    def ask_parent(self, event):
        if event in self.events:
            event.participants[self.name] = 'Waiting for parent confirmation'

class Event:
    def __init__(self, title, date):
        self.title = title
        self.date = date
        self.participants = {}

class MasterClass(Event):
    def format_event(self):
        return f"This is a master class on {self.title} happening on {self.date}"

class Competition(Event):
    def format_event(self):
        return f"This is a competition on {self.title} happening on {self.date}"

def register(participant, event):
    participant.participate(event)
```

Рисунок 3. Сгенерированное в Copilot описание классов

```python
# Test the classes
p1 = Adult('John', 30)
p2 = Child('Timmy', 10)

e1 = MasterClass('Cooking', '2023-12-01')
e2 = Competition('Chess', '2023-12-05')

register(p1, e1)
register(p2, e2)

p1.confirm_participation(e1)
p2.ask_parent(e2)

print(p1.events[0].participants)
print(p2.events[0].participants)
```

Рисунок 4. Основная программа, сгенерированная Copilot для демонстрации функционала описанных классов

Оценка упражнений по программированию проводилась в виде исследования смешанными методами, где упражнения оценивались как качественно, так и количественно.

```
def test_classes():
    p1 = Adult('John', 30)
    p2 = Child('Timmy', 10)

    e1 = MasterClass('Cooking', '2023-12-01')
    e2 = Competition('Chess', '2023-12-05')

    register(p1, e1)
    register(p2, e2)

    p1.confirm_participation(e1)
    p2.ask_parent(e2)

    assert p1.events[0].participants == {'John': 'Confirmed'}
    assert p2.events[0].participants == {'Timmy': 'Waiting for parent confirmation'}

test_classes()
```

Рисунок 5 Сгенерированные в Copilot автоматические тесты

В качественном анализе мы сосредоточились на случайной выборке из 80 упражнений. Задачи проверялись на логичность формулировок, новизну и готовность к использованию созданных упражнений в учебном процессе. При оценке адекватности описания предметной области задачи анализировалось доступна ли она для понимания учащимися – описывает ли постановка задачи практическую задачу, с методами решения которой учащиеся знакомы. При оценке новизны осуществлялся поиск как в Google, так и в GitHub на предмет существования дословной формулировки упражнения или аналогичной сгенерированному. В качестве новизны также рассматривалась адекватность сгенерированных упражнений на применение различных концепций, заданных в прайминге. При оценке готовности к использованию учитывался объем ручной работы, которую преподаватель должен будет проделать для выполнения упражнений и связанных с ними образцов решения и тестов.

Качественный анализ был проведен несколькими исследователями индивидуально с использованием критериев, описанных в таблице 2, где каждый исследователь оценивал выборку упражнений с помощью утверждений «Да /Нет /Может быть» и добавлял примечания по мере необходимости. Все ответы на вопрос «Может быть» были затем совместно проанализированы, по крайней мере, двумя исследователями, работающими в тандеме, чтобы сформировать единое мнение о том, следует ли рассматривать их как «Да» или «Нет».

Таблица 2. Критерии качественного анализа упражнений

Аспект	*Критерий*	*Оценка*
Логичность формулировок	*Адекватно ли постановка задачи описывает проблему?*	*Да / Нет / Может быть*
Новизна	*Постановка задачи по ООП не находится через онлайн-поиск (Google и GitHub).*	*Да / Нет / Может быть*
Готовность: проблема и решение	*Соответствует ли постановка задачи сгенерированному решению?*	*Да / Нет / Может быть*
Адекватность взаимодействия между объектами	*Соответствуют ли взаимодействия между объектами в постановке задачи запросу описанному в прайминге?*	*Да / Нет / Может быть*
Адекватность структур данных	*Включает ли постановка задачи требуемые в прайминге структуры данных?*	*Да / Нет / Может быть*
Адекватность предметной области	*Соответствует ли тема постановки задачи заданному в прайминге ключевому слову?*	*Да / Нет / Может быть*
Примечания		

Для количественного анализа все сгенерированные упражнения были проверены в среде программирования (таблица 3) на выполнение следующих требований:

1) Можно ли запустить сгенерированные решения?
2) Прошли ли сгенерированные решения автоматизированные тесты?
3) Покрытие заданий автоматическими тестами.

Таблица 3. Критерии количественного анализа упражнений

Аспект «Готовность»	*Критерий*	*Оценка*
Работоспособность	*Можем ли мы запустить пример решения без ошибок?*	*Да / Нет / Нет данных*
Решение и тестирование	*Проходит ли пример решения модульные тесты?*	*Да / Нет / Нет данных*
Тестовое покрытие	*В какой степени модульные тесты охватывают сгенерированное решение?*	*от 0 до 100% / Нет данных*

Результаты и обсуждения

Статистические данные по адекватности, новизне и готовности оцениваемых упражнений представлены в таблице 4. Из них 75,0% были разумными, 81,8% – новыми, а 76,7% имели соответствующий выборочный раствор. В заметках в свободной форме в основном обсуждались вопросы, которые включали наличие избыточной информации, отсутствующую информацию, отсутствующие или неверные значения в выборочных входах и/или выходах (некоторые из формулировок задачи содержали примеры входных и выходных данных), обсуждались несоответствия между постановкой задачи и образцом решения, а также излагались причины, по которым автоматические тесты не проходят.

Таблица 4. Результаты качественного анализа упражнений

Количество упражнений	*Логичность изложения*	*Формулировка*	*Соответствие ключевого слова*	*Соответствие прайменгу*	*Адекватность взаимодействия объектов*	*Адекватность структур данных*
80	*79%*	*82%*	*77%*	*80%*	*76%*	*76%*

Статистические данные по программному анализу, проведенному по всем 360 сгенерированным упражнениям, представлены в таблице 5. Из 360 упражнений по программированию 317 имели пробное решение (88%). 331 упражнение (92%) удалось выполнить (т.е. выполнение кода не привело к ошибкам). В общей сложности 270 упражнений содержали автоматические тесты, а 265 упражнений содержали как образец решения, так и автоматические тесты.

Таблица 5. Результаты количественного анализа упражнений

Аспект «Готовность»	*Есть решение*	*Пример можно запустить*	*Есть тесты*	*Все тесты пройдены*	*Тестовое покрытие*
Проценты	*88%*	*92%*	*75%*	*65%*	*98%*
Количество	*317/360*	*331/360*	*270/360*	*234/360*	*353/360*

В результатах программного анализа готовности, представленных в таблице 5, мы видим, что примерно в 90% случаев сгенерированные примеры решений являются валидным работающим кодом, тесты генерируются и автоматически извлекаются.

Большинство сгенерированных упражнений оказались одновременно корректными и новыми, а также включали в себя образец решения, который компилировался и запускался. Использование ChatGPT продемонстрировало достаточно хорошую производительность при интерактивной работе с Copilot и пошаговой подсказкой.

Заключение

Таким образом, результаты данного эксперимента подтверждают выводы о том, что большие языковые модели обучаются с нуля [18] и с небольшим количеством попыток [19]. Мы убедились в том, что они хорошо справляются с заданиями, даже если в качестве входных данных не дается ни одного или всего несколько примеров в качестве образца. Современные модели машинного обучения, такие как OpenAI Copilot, предоставляют множество возможностей разработчикам образовательных курсов по программированию, хотя не следует игнорировать потенциальные проблемы, связанные с их использованием [20]. Проведенный нами анализ показал хорошие результаты при генерации новых и корректно сформулированных упражнений по программированию с готовыми примерами решений и автоматизированными тестами, несмотря на наличие некоторых проблем с точностью и качеством (которые могут быть легко исправлены вручную). Мы предвидим, что доступность генеративных моделей для практики компьютерного образования и научных исследований со временем будет только возрастать по мере дальнейшего развития этих технологий.

Список использованной литературы:

1. D. Radosevic, T. Orehovacki and Z. Stapic. Automatic on-line generation of student's exercises in teaching programming, vol. 1, Sep. 2010.

2. S. Davis, C. Grover, A. Becker and L. Mcgregor. Academic dishonesty: Prevalence, determinants, techniques, and punishments, Teaching of Psychology - TEACH PSYCHOL, vol. 19, pp. 16–20, Feb. 1992. DOI: 10.1207/ s15328023top1901_3.

3. B. Ozmen Yagiz and A. Altun. Undergraduate students' experiences in programming: Difficulties and obstacles, Turkish Online Journal of Qualitative Inquiry, vol. 5, Mar. 2014. DOI: 10.17569/tojqi.20328.

4. OpenAI, Openai chatgpt blog post, https://openai.com/blog/chatgpt/, Accessed: 2023-11-10.

5. Al-Jabri, O., & Al-Khalifa, M. (2022). Using ChatGPT for teaching object-oriented programming. Journal of Educational Technology & Society, 25(3), 1-12.

6. Cetin, Y., & Turkmen, S. (2021). A study on the effectiveness of ChatGPT on object-oriented programming education. In 2021 13th International Conference on Informatics Education (ICIE) (pp. 211-216). IEEE.

7. Kumar, S., & Kaushik, S. (2022). ChatGPT: A tool for teaching object-oriented programming. In 2022 IEEE 9th International Conference on Computing, Communication and Automation (ICCCA) (pp. 107-112). IEEE.

8. M. Kramer, P. Hubwieser and T. Brinda, "A Competency Structure Model of Object-Oriented Programming," 2016 International Conference on Learning and Teaching in Computing and Engineering (LaTICE), Mumbai, India, 2016, pp. 1-8, doi: 10.1109/LaTiCE.2016.24.

9. Computing Curricula 2020. URL: https://www.acm.org/binaries/content/assets/education/curricularecommendations/cc2020.pdf, 2020 (дата обращения: 23.10.2023).

10.Chen, Y., & Wang, Q. (2021). Using artificial intelligence to detect plagiarism in object-oriented programming assignments. In 2021 IEEE 59th Conference on Decision and Control (CDC) (pp. 4076-4081). IEEE.

11.Kim, T., & Kim, J. (2022). A method for preventing plagiarism in object-oriented programming assignments using artificial intelligence. In 2022 IEEE 7th International Conference on Educational Data Mining (EDM) (pp. 393-398). IEEE.

12.Li, H., Li, C., & Chen, H. (2022). A novel approach for detecting plagiarism in object-oriented programming assignments using deep learning. In 2022 IEEE 46th Annual Computer Software and Applications Conference (COMPSAC) (pp. 1351-1357). IEEE.

13.Abramson, J. S., & Taylor, T. (2022). The ethics of AI in education. In Ethics of artificial intelligence: A primer (pp. 129-146). Springer.

14.Bates, A. W., & Sangra, A. (2021). Addressing the challenges of artificial intelligence in education. In Artificial intelligence for education: Designing and deploying effective learning experiences (pp. 3-18). Springer.

15.Fischer, F., & Heerink, M. (2022). Addressing the risks of artificial intelligence in education. In Ethics of artificial intelligence: A primer (pp. 147-164). Springer.

16.Khor, K., & Yusoff, M. (2022). A systematic review of chatbot applications in education. Education and Information Technologies, 27(2), 1035-1059.

17.Zhao, Y., & Zhang, J. (2022). A survey of chatbots in education: Applications, challenges, and opportunities. Educational Technology Research and Development, 70(2), 333-360.

18.Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large Language Models are Zero-Shot Reasoners. arXiv preprint arXiv:2205.11916 (2022).

19.Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In Advances in neural information processing systems. 1877–1901.

20.James Finnie-Ansley, Paul Denny, Brett A Becker, Andrew Luxton-Reilly, and James Prather. 2022. The Robots Are Coming: Exploring the Implications of OpenAI Codex on Introductory Programming. In Australasian Computing Education Conference. 10–19.