

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ**
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Воронежский государственный технический университет»

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ
«Интеллектуальные технологии разработки программного
обеспечения»

Направление подготовки 09.04.01 Информатика и вычислительная техника

Профиль Управление программным инжинирингом

Квалификация выпускника магистр

Нормативный период обучения 2 года

Форма обучения очная

Год начала подготовки 2026

Воронеж 2026

1. ЦЕЛИ И ЗАДАЧИ ДИСЦИПЛИНЫ

1.1. Цели дисциплины

Целью дисциплины «Интеллектуальные технологии разработки программного обеспечения» является формирование у магистрантов профессиональных компетенций в области применения интеллектуальных инструментов и технологий на всех этапах жизненного цикла программного обеспечения: проектирование, кодирование, тестирование, ревью и сопровождение продукта, с использованием современных ИИ-ассистентов разработчика (Cursor, GigaCode, GitHub Copilot), техник промтинга и интеллектуальных инструментов автоматизации тестирования.

1.2. Задачи освоения дисциплины

В результате освоения дисциплины студент должен решать следующие задачи:

- освоить принципы работы современных ИИ-ассистентов разработчика: Cursor, GigaCode, JetBrains AI Assistant и аналогичных инструментов;
- сформировать навыки эффективного промтинга для задач генерации кода, рефакторинга, документирования и проектирования архитектуры;
- научиться применять ИИ-инструменты для автоматизации тестирования: генерация unit-тестов, интеграционных тестов, анализ покрытия кода и выявление дефектов;
- освоить методологии проведения code review с участием ИИ: автоматический анализ кода, поиск уязвимостей, рефакторинг с помощью ИИ-моделей;
- сформировать навыки интеграции ИИ-инструментов в CI/CD-пайплайны и процессы DevOps;
- изучить методы работы с RAG-системами, LLM-агентами и интеллектуальными API для автоматизации задач разработки ПО;
- оценить эффективность, ограничения и риски применения ИИ-ассистентов в корпоративной разработке ПО.

2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ОПОП

Дисциплина «Интеллектуальные технологии разработки программного обеспечения» относится к дисциплинам обязательной части блока Б1.

3. ПЕРЕЧЕНЬ ПЛАНИРУЕМЫХ РЕЗУЛЬТАТОВ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ

Процесс изучения дисциплины «Интеллектуальные технологии разработки программного обеспечения» направлен на формирование следующих компетенций:

ОПК-2 - Способен разрабатывать оригинальные алгоритмы и программные средства, в том числе с использованием современных интеллектуальных технологий, для решения профессиональных задач;

ОПК-5 - Способен разрабатывать и модернизировать программное и аппаратное обеспечение информационных и автоматизированных систем;

Компетенция	Результаты обучения, характеризующие сформированность компетенции
ОПК-2	знать: принципы работы ИИ-ассистентов разработчика (Cursor, GigaCode, GitHub, JetBrains AI Assistant); методы и паттерны промтинга для задач кодогенерации, рефакторинга, документирования и архитектурного проектирования; основы работы крупных языковых моделей (LLM) применительно к задачам разработки ПО; ограничения и риски генеративного ИИ в программной инженерии.
	уметь: эффективно взаимодействовать с ИИ-ассистентами для генерации, объяснения, рефакторинга и документирования кода; составлять структурированные промты для решения конкретных задач разработки ПО; применять ИИ-инструменты для ускорения цикла «задача — реализация — тест»; критически оценивать качество кода, сгенерированного ИИ.
	владеть: навыками разработки оригинальных программных решений с применением ИИ-ассистентов в качестве инструмента усиления продуктивности; техниками prompt engineering для профессиональных задач разработки ПО; методами верификации и улучшения кода, созданного с участием ИИ.
ОПК-5	знать: современные ИИ-инструменты автоматизации тестирования ПО (генерация unit-тестов, интеграционных тестов, нагрузочных сценариев); принципы применения LLM для статического анализа кода, выявления уязвимостей и технического долга; методы интеграции ИИ-инструментов в CI/CD-пайплайны и процессы DevOps; подходы к code review с применением ИИ.
	уметь: автоматически генерировать и совершенствовать тестовые сценарии с помощью ИИ-инструментов; проводить ИИ-ассистированный анализ и ревью программного кода; встраивать интеллектуальные инструменты анализа качества кода в пайплайны непрерывной интеграции; использовать RAG-системы и LLM-агентов для автоматизации рутинных задач разработки.

	владеть: навыками проектирования и сопровождения ПО с применением полного спектра ИИ-инструментов жизненного цикла разработки; методами оценки и контроля качества ПО с использованием интеллектуального тестирования; практиками безопасного и этичного применения ИИ-ассистентов в профессиональной разработке.
--	--

4. ОБЪЕМ ДИСЦИПЛИНЫ

Общая трудоемкость дисциплины «Интеллектуальные технологии разработки программного обеспечения» составляет 5 з.е.

Распределение трудоемкости дисциплины по видам занятий
очная форма обучения

Виды учебной работы	Всего часов	Семестры
		1
Аудиторные занятия (всего)	108	108
В том числе:		
Лекции	36	36
Лабораторные работы (ЛР)	72	72
Самостоятельная работа	45	45
Курсовой проект	+	+
Часы на контроль	27	27
Виды промежуточной аттестации - экзамен	+	+
Общая трудоемкость:		
академические часы	180	180
зач.ед.	5	5

5. СОДЕРЖАНИЕ ДИСЦИПЛИНЫ (МОДУЛЯ)

5.1 Содержание разделов дисциплины и распределение трудоемкости по видам занятий

очная форма обучения

№ п/п	Наименование темы	Содержание раздела	Лекц	Лаб. зан.	СРС	Всего, час
1	ИИ-ассистенты в современной разработке ПО: обзор, архитектура, возможности	Обзор экосистемы ИИ-инструментов для разработчика: Cursor, GigaCode (Сбер), GitHub Copilot, JetBrains AI Assistant, Tabnine, Amazon CodeWhisperer. Архитектура LLM-ассистентов: языковые модели, контекстное окно, инструкционный фэйнтюнинг. Режимы работы: автодополнение, диалоговый чат, агентный режим (Cursor Agent, Agentic IDE). Встраивание ИИ-инструментов в рабочий процесс разработчика. Ограничения, галлюцинации, вопросы безопасности и конфиденциальности кода. Отечественные альтернативы: GigaCode и GigaChat API.	6	12	6	24

2	Промтинг для задач разработки программного обеспечения	Основы prompt engineering: структура промта (роль, контекст, задача, формат, ограничения). Паттерны промтинга для разработчика: zero-shot, few-shot, chain-of-thought, role prompting. Промтинг для кодогенерации: генерация функций, классов, модулей по спецификации. Промтинг для рефакторинга: улучшение читаемости, применение паттернов проектирования, оптимизация производительности. Промтинг для документирования: генерация docstring, README, OpenAPI-спецификаций. Промтинг для архитектурного проектирования: выбор паттернов, оценка решений, сравнение подходов. Работа с системными промтами и .cursorrules / .github/copilot-instructions.md.	6	12	8	26
3	ИИ-ассистированное тестирование программного обеспечения	Применение ИИ на этапе тестирования: автоматическая генерация unit-тестов (pytest, JUnit, NUnit) с помощью LLM-ассистентов. Генерация интеграционных и end-to-end тестов. ИИ-инструменты для тестирования: CodiumAI (Qodo), GitHub Copilot Tests, Cursor в режиме генерации тестов. Анализ покрытия кода с помощью ИИ и выявление непокрытых ветвей. Генерация граничных случаев и негативных сценариев. Мутационное тестирование с поддержкой ИИ. Автоматический поиск дефектов: статический анализ с DeepCode/Snyk Code, SonarQube AI. Prompt-driven тест-дизайн: составление тестовых сценариев по требованиям.	6	12	8	26
4	ИИ в code review, анализе качества и управлении техническим долгом	Автоматизированный code review с применением ИИ: CodeRabbit, GitHub Copilot Code Review, Cursor Chat. Обнаружение code smells, антипаттернов и уязвимостей с помощью LLM. ИИ-ассистированный рефакторинг: автоматическое предложение улучшений, применение паттернов SOLID, DRY, KISS. Анализ и визуализация технического долга: SonarQube, DeepSource. Генерация объяснений и комментариев к легаси-коду. Работа с большими кодовыми базами: индексация и поиск по репозиторию в Cursor, JetBrains AI, GitHub Copilot Workspace. Метрики продуктивности разработчика при использовании ИИ-ассистентов.	6	12	8	26
5	LLM-агенты, RAG-системы и ИИ-автоматизация в DevOps	LLM-агенты в разработке ПО: принцип работы агентного режима (Cursor Agent, Devin, OpenHands). Инструменты агентов: вызов функций, работа с файловой системой, запуск тестов, веб-поиск. RAG (Retrieval-Augmented Generation) для работы с документацией и кодовой базой: индексация, векторные хранилища, запросы. Интеграция ИИ-инструментов в CI/CD: автоматический анализ pull requests, генерация changelog, проверка безопасности. GitHub Actions + ИИ: автоматические ревью, генерация release notes. Интеллектуальный мониторинг и отладка: ИИ-ассистированный анализ логов и трейсов. Оценка ROI от внедрения ИИ-инструментов в команду разработки.	6	12	8	26
6	Практика применения ИИ-инструментов в полном цикле разработки ПО	Сквозной проект: разработка программного продукта с применением ИИ-ассистентов на всех этапах. ИИ в планировании и декомпозиции задач: генерация user stories, оценка сложности. ИИ в проектировании: генерация ERD, UML, архитектурных схем из описания. Разработка с Cursor/GigaCode: сессия живого кодирования с ИИ-ассистентом. ИИ в	6	12	7	25

		тестировании: полный покрывающий набор тестов по написанному коду. ИИ в документировании: автогенерация технической документации. Этические и правовые аспекты: авторское право на ИИ-сгенерированный код, политики использования ИИ в компаниях. Оценка и защита курсового проекта.				
Итого			36	72	45	153

5.2 Перечень лабораторных работ

1. Настройка и сравнение ИИ-ассистентов разработчика: Cursor, GigaCode.
2. Промтинг для кодогенерации: генерация модулей по спецификации.
3. Промтинг для рефакторинга с применением паттернов проектирования.
4. Генерация unit-тестов с помощью ИИ (CodiumAI / Cursor).
5. ИИ-ассистированный code review и обнаружение дефектов.
6. Работа с .cursorrules: настройка корпоративного стиля кодирования.
7. Автогенерация технической документации (README, docstring, OpenAPI).
8. RAG-система для навигации по кодовой базе и документации.
9. Агентный режим: автоматизация многошагового сценария разработки.
10. Интеграция ИИ-инструментов в CI/CD-пайплайн на GitHub Actions.

6. ПРИМЕРНАЯ ТЕМАТИКА КУРСОВЫХ ПРОЕКТОВ (РАБОТ) И КОНТРОЛЬНЫХ РАБОТ

В соответствии с учебным планом освоение дисциплины предусматривает выполнение курсового проекта в 1 семестре для очной формы обучения, в 4 семестре для заочной формы обучения.

Примерная тематика курсового проекта: «Разработка программного продукта с применением ИИ-ассистентов разработчика на всех этапах жизненного цикла»

Задачи, решаемые при выполнении курсового проекта:

- Постановка задачи и выбор предметной области — формулировка требований к разрабатываемому программному продукту в виде user stories с применением ИИ-ассистента; декомпозиция задачи на модули и оценка трудоёмкости.
- Настройка рабочей среды и ИИ-инструментов — выбор и обоснование ИИ-ассистента; настройка файла .cursorrules / системного промта под стиль и стек проекта; подключение индексации кодовой базы.
- Проектирование архитектуры с применением промтинга — разработка архитектурных схем (UML, ERD) и выбор паттернов проектирования с использованием ИИ-ассистента; обоснование принятых архитектурных решений.

- Реализация программного продукта с ИИ-ассистентом — написание кода с применением техник промтинга (кодогенерация, рефакторинг, объяснение); документирование кода (docstring, README, OpenAPI-спецификация) с помощью ИИ; применение агентного режима для автоматизации рутинных задач разработки.
- ИИ-ассистированное тестирование — автоматическая генерация unit-тестов и интеграционных тестов с помощью CodiumAI / Cursor; анализ покрытия кода; выявление и исправление дефектов с применением ИИ-инструментов статического анализа.
- Code review с применением ИИ — проведение автоматизированного ревью кода выявление уязвимостей, антипаттернов и технического долга; рефакторинг по результатам ревью.
- Анализ эффективности применения ИИ-инструментов — сравнение производительности разработки с ИИ-ассистентом и без него; оценка качества сгенерированного кода; выводы об ограничениях и рисках применённых инструментов.
- Оформление пояснительной записки и защита — подготовка технической документации; представление результатов с демонстрацией применённых ИИ-инструментов и промт-стратегий.

Курсовой проект включают в себя графическую часть и расчетно-пояснительную записку.

7. ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ДЛЯ ПРОВЕДЕНИЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ОБУЧАЮЩИХСЯ ПО ДИСЦИПЛИНЕ

7.1. Описание показателей и критериев оценивания компетенций на различных этапах их формирования, описание шкал оценивания

7.1.1 Этап текущего контроля

Результаты текущего контроля знаний и межсессионной аттестации оцениваются по следующей системе:

«аттестован»;

«не аттестован».

Компетенция	Результаты обучения, характеризующие сформированность компетенции	Критерии оценивания	Аттестован	Не аттестован
ОПК-2	знать: принципы работы ИИ-ассистентов разработчика (Cursor, GigaCode, GitHub, JetBrains AI Assistant); методы и паттерны промтинга для задач кодогенерации, рефакторинга,	Наличие конспекта лекций; демонстрация понимания архитектуры и возможностей ИИ-ассистентов разработчика; активность на занятиях	Выполнение работ в срок, предусмотренный в рабочих программах	Невыполнение работ в срок, предусмотренный в рабочих программах

	документирования и архитектурного проектирования; основы работы крупных языковых моделей (LLM) применительно к задачам разработки ПО; ограничения и риски генеративного ИИ в программной инженерии.			
	уметь: эффективно взаимодействовать с ИИ-ассистентами для генерации, объяснения, рефакторинга и документирования кода; составлять структурированные промты для решения конкретных задач разработки ПО; применять ИИ-инструменты для ускорения цикла «задача — реализация — тест»; критически оценивать качество кода, сгенерированного ИИ.	Своевременное и корректное выполнение лабораторных работ по промтингу и работе с ИИ-ассистентами	Выполнение работ в срок, предусмотренный в рабочих программах	Невыполнение работ в срок, предусмотренный в рабочих программах
	владеть: навыками разработки оригинальных программных решений с применением ИИ-ассистентов в качестве инструмента усиления продуктивности; техниками prompt engineering для профессиональных задач разработки ПО; методами верификации и улучшения кода, созданного с участием ИИ.	Выполнение лабораторных работ в полном объёме; демонстрация навыков составления эффективных промтов и критической оценки результатов ИИ	Выполнение работ в срок, предусмотренный в рабочих программах	Невыполнение работ в срок, предусмотренный в рабочих программах
ОПК-5	знать: современные ИИ-инструменты автоматизации тестирования ПО (генерация unit-тестов, интеграционных тестов, нагрузочных сценариев); принципы применения LLM для статического анализа кода, выявления уязвимостей и технического долга; методы интеграции ИИ-инструментов в CI/CD-пайплайны и процессы	Наличие конспекта лекций; знание ИИ-инструментов автоматизации тестирования и code review; активность на занятиях	Выполнение работ в срок, предусмотренный в рабочих программах	Невыполнение работ в срок, предусмотренный в рабочих программах

	DevOps; подходы к code review с применением ИИ.			
	уметь: автоматически генерировать и совершенствовать тестовые сценарии с помощью ИИ-инструментов; проводить ИИ-ассистированный анализ и ревью программного кода; встраивать интеллектуальные инструменты анализа качества кода в пайплайны непрерывной интеграции; использовать RAG-системы и LLM-агентов для автоматизации рутинных задач разработки.	Своевременное и корректное выполнение лабораторных работ по ИИ-тестированию и анализу кода	Выполнение работ в срок, предусмотренный в рабочих программах	Невыполнение работ в срок, предусмотренный в рабочих программах
	владеть: навыками проектирования и сопровождения ПО с применением полного спектра ИИ-инструментов жизненного цикла разработки; методами оценки и контроля качества ПО с использованием интеллектуального тестирования; практиками безопасного и этичного применения ИИ-ассистентов в профессиональной разработке.	Выполнение лабораторных работ в полном объеме; демонстрация навыков интеграции ИИ-инструментов в процессы разработки и тестирования ПО	Выполнение работ в срок, предусмотренный в рабочих программах	Невыполнение работ в срок, предусмотренный в рабочих программах

7.1.2 Этап промежуточного контроля знаний

Результаты промежуточного контроля знаний оцениваются в 1 семестре для очной формы обучения, 4 семестре для заочной формы обучения по четырехбалльной системе:

- «отлично»;
- «хорошо»;
- «удовлетворительно»;
- «неудовлетворительно».

Компетенция	Результаты обучения, характеризующие сформированность компетенции	Критерии оценивания	Отлично	Хорошо	Удовл.	Неудовл.
ОПК-2	знать: принципы работы ИИ-ассистентов разработчика	Тест	Выполнение теста на 90-100%	Выполнение теста на 80-90%	Выполнение теста на 70-80%	В тесте менее 70% правильных ответов

	(Cursor, GigaCode, GitHub, JetBrains AI Assistant); методы и паттерны промтинга для задач кодогенерации, рефакторинга, документирования и архитектурного проектирования; основы работы крупных языковых моделей (LLM) применительно к задачам разработки ПО; ограничения и риски генеративного ИИ в программной инженерии.					
	уметь: эффективно взаимодействовать с ИИ-ассистентами для генерации, объяснения, рефакторинга и документирования кода; составлять структурированные промты для решения конкретных задач разработки ПО; применять ИИ-инструменты для ускорения цикла «задача — реализация — тест»; критически оценивать качество кода, сгенерированного ИИ.	Решение стандартных практических задач	Задачи решены в полном объеме и получены верные ответы	Продемонстрирован верный ход решения всех, но не получен верный ответ во всех задачах	Продемонстрирован верный ход решения в большинстве задач	Задачи не решены
	владеть: навыками разработки оригинальных программных решений с применением ИИ-ассистентов в качестве инструмента усиления продуктивности; техниками prompt engineering для профессиональных задач разработки ПО; методами верификации и улучшения кода, созданного с участием ИИ.	Решение прикладных задач в конкретной предметной области	Задачи решены в полном объеме и получены верные ответы	Продемонстрирован верный ход решения всех, но не получен верный ответ во всех задачах	Продемонстрирован верный ход решения в большинстве задач	Задачи не решены

ОПК-5	знать: современные ИИ-инструменты автоматизации тестирования ПО (генерация unit-тестов, интеграционных тестов, нагрузочных сценариев); принципы применения LLM для статического анализа кода, выявления уязвимостей и технического долга; методы интеграции ИИ-инструментов в CI/CD-пайплайны и процессы DevOps; подходы к code review с применением ИИ.	Тест	Выполнение теста на 90-100%	Выполнение теста на 80-90%	Выполнение теста на 70-80%	В тесте менее 70% правильных ответов
	уметь: автоматически генерировать и совершенствовать тестовые сценарии с помощью ИИ-инструментов; проводить ИИ-ассистированный анализ и ревью программного кода; встраивать интеллектуальные инструменты анализа качества кода в пайплайны непрерывной интеграции; использовать RAG-системы и LLM-агентов для автоматизации рутинных задач разработки.	Решение стандартных практических задач	Задачи решены в полном объеме и получены верные ответы	Продемонстрирован верный ход решения всех, но не получен верный ответ во всех задачах	Продемонстрирован верный ход решения в большинстве задач	Задачи не решены
	владеть: навыками проектирования и сопровождения ПО с применением полного спектра ИИ-инструментов жизненного цикла разработки; методами оценки и контроля качества ПО с использованием интеллектуального тестирования; практиками безопасного и	Решение прикладных задач в конкретной предметной области	Задачи решены в полном объеме и получены верные ответы	Продемонстрирован верный ход решения всех, но не получен верный ответ во всех задачах	Продемонстрирован верный ход решения в большинстве задач	Задачи не решены

	этичного применения ИИ-ассистентов в профессиональной разработке.					
--	---	--	--	--	--	--

7.2 Примерный перечень оценочных средств (типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности)

7.2.1 Примерный перечень заданий для подготовки к тестированию

1. Какой из перечисленных инструментов является ИИ-ассистентом разработчика, встроенным непосредственно в IDE?

- а) Apache Kafka;
- б) Cursor IDE с интегрированным ИИ-ассистентом;
- в) PostgreSQL;
- г) Docker Compose.

2. Что такое GigaCode?

- а) Отечественный фреймворк для машинного обучения;
- б) Плагин-ассистент разработчика от Сбербанка на основе GigaChat, предоставляющий автодополнение кода и чат-помощь;
- в) Система управления версиями кода;
- г) Инструмент для контейнеризации приложений.

3. Что такое «галлюцинация» ИИ-ассистента в контексте разработки ПО?

- а) Слишком быстрая генерация кода;
- б) Генерация синтаксически корректного, но семантически неверного кода (несуществующие функции, API, библиотеки);
- в) Отказ отвечать на запрос пользователя;
- г) Повторение одного и того же фрагмента кода.

4. Что такое .cursorrules?

- а) Файл конфигурации базы данных;
- б) Системный промт-файл в Cursor IDE, задающий правила, стиль кодирования и контекст для ИИ-ассистента в проекте;
- в) Набор правил линтера для Python;
- г) Файл описания CI/CD-пайплайна.

5. Какой паттерн промтинга предполагает несколько примеров желаемого результата до формулировки задачи?

- а) Zero-shot;
- б) Chain-of-thought;
- в) Few-shot;
- г) Role prompting.

6. Что означает «контекстное окно» (context window) применительно к ИИ-ассистенту?

- а) Визуальное окно предпросмотра кода в IDE;
- б) Максимальный объём текста (кода, промтов, истории диалога), который модель учитывает при генерации ответа;
- в) Список открытых вкладок в редакторе;
- г) Буфер обмена операционной системы.

7. Что такое агентный режим (Agentic mode) в Cursor IDE?

- а) Режим автоматического форматирования кода;
- б) Режим, в котором ИИ самостоятельно выполняет многошаговые задачи: редактирует файлы, запускает тесты и итеративно исправляет ошибки;
- в) Режим совместного редактирования несколькими разработчиками;
- г) Режим автоматической публикации кода в репозиторий.

8. Какой инструмент специализируется на ИИ-генерации и анализе тестов для кода?

- а) Docker;
- б) CodiumAI (Qodo);
- в) Apache Airflow;
- г) Postman.

9. Что такое RAG (Retrieval-Augmented Generation) применительно к разработке ПО?

- а) Метод обучения нейронной сети с подкреплением;
- б) Подход, при котором LLM дополняет генерацию ответа релевантными фрагментами из базы знаний или кодовой базы;
- в) Система автоматического развёртывания приложений;
- г) Протокол передачи данных между микросервисами.

10. Какой инструмент предназначен для автоматизированного ИИ-ревью pull requests?

- а) Jupyter Notebook;
- б) CodeRabbit;
- в) pgAdmin;
- г) Grafana.

7.2.2 Примерный перечень заданий для решения стандартных задач

1. Разработчик генерирует в Cursor функцию парсинга JSON. Наиболее эффективный промт:

- а) «Напиши код»;

- б) «Напиши на Python функцию `parse_json(data: str) -> dict`, обрабатывающую `JSONDecodeError` и возвращающую пустой словарь при ошибке. Добавь docstring и type hints»;
- в) «Сделай парсер»;
- г) «Помоги с Python».

2. ИИ-ассистент вернул код с несуществующей библиотекой. Правильная реакция:

- а) Немедленно закоммитить код;
- б) Проверить библиотеку в официальной документации, скорректировать промт с указанием доступных зависимостей;
- в) Удалить ИИ-ассистент;
- г) Заменить название на похожее без проверки.

3. Для автоматической генерации unit-тестов Python-модуля наиболее подходящий подход:

- а) Писать тесты вручную;
- б) Открыть модуль в Cursor или CodiumAI и запросить тесты с граничными случаями;
- в) Использовать генератор случайных данных;
- г) Скопировать тесты из аналогичного проекта.

4. Для понимания логики легаси-кода без комментариев с помощью ИИ оптимально:

- а) Переписать код заново;
- б) Вставить фрагмент в Cursor Chat: «Объясни пошагово, что делает код, и укажи возможные проблемы»;
- в) Запустить и по ошибкам восстановить логику;
- г) Спросить у коллег.

5. Для автопроверки PR на безопасность и стиль в CI/CD используется:

- а) Только ручное ревью тимлида;
- б) CodeRabbit или аналог, интегрированный в GitHub Actions с автокомментариями к PR;
- в) Только линтер;
- г) Отключение проверок для ускорения деплоя.

6. Чтобы применить паттерн «Стратегия» с ИИ-ассистентом, промт должен содержать:

- а) Только «примени паттерн»;
- б) Исходный код, название паттерна, язык и ожидаемый результат с объяснением;
- в) Только название паттерна;
- г) Запрос на замену всего кода.

7. Для генерации OpenAPI-спецификации REST API с помощью ИИ:
- а) Написать спецификацию вручную;
 - б) Передать ИИ-ассистенту код контроллеров и запросить YAML-спецификацию OpenAPI 3.0;
 - в) Использовать только Swagger UI;
 - г) Скопировать спецификацию аналогичного API.
8. В `.cursorrules` для командного проекта целесообразно указать:
- а) Только название проекта;
 - б) Язык, фреймворк, стандарты именования, запрещённые паттерны и правила тестов;
 - в) Пароли от БД;
 - г) Список имён разработчиков.
9. Для навигации по большой кодовой базе в Cursor первый шаг:
- а) Удалить лишние файлы;
 - б) Добавить папку проекта в Cursor и включить индексацию (@Codebase);
 - в) Скопировать весь код в промт;
 - г) Установить отдельную БД для кода.
10. Главное преимущество GigaCode перед зарубежными аналогами:
- а) Более быстрая генерация;
 - б) Российская инфраструктура, соответствие требованиям ИБ, код не передаётся за рубеж;
 - в) Больше поддерживаемых языков;
 - г) Бесплатный доступ без ограничений.

7.2.3 Примерный перечень заданий для решения прикладных задач

1. Команда стартапа выбирает ИИ-ассистент для разработки. Приоритетный критерий:
- а) Самый популярный в соцсетях;
 - б) Бесплатный план, интеграция с IDE, поддержка языка проекта и политика конфиденциальности кода;
 - в) Наибольший размер модели;
 - г) Яркий интерфейс.
2. Разработчик сгенерировал 200 строк кода в Cursor. Следующий обязательный шаг:
- а) Слить в `main`;
 - б) Ревью кода: проверка логики, ошибок, соответствия стандартам, запуск тестов;
 - в) Отправить на проверку только ИИ;
 - г) Отправить заказчику без проверки.

3. В проекте 50 000 строк легаси без документации. Стратегия документирования с ИИ:

- а) Писать вручную для каждой функции;
- б) ИИ генерирует docstring по модулям, пайплайн автогенерации в CI, критическая проверка результата;
- в) Удалить код и начать заново;
- г) Запросить документацию у заказчика.

4. CodiumAI создал 30 тестов, 5 падают. Правильные действия:

- а) Удалить упавшие тесты;
- б) Проанализировать: возможно, выявлены реальные баги; исправить баги или скорректировать тесты;
- в) Отключить тесты до следующего релиза;
- г) Принять результаты ИИ без проверки.

5. Задача Cursor Agent: прочитать spec.md, создать модель БД, CRUD и тесты. Правильная организация:

- а) Один неструктурированный промт;
- б) Чёткие шаги в промте, контекст стека и файлов, итерационная проверка каждого этапа;
- в) Полная передача агенту без контроля;
- г) Отказ от агентного режима.

6. Компания рассматривает GitHub Copilot для 20 разработчиков. Первоочередной риск:

- а) Слишком быстрая генерация;
- б) Передача кода на серверы Microsoft/OpenAI — риск утечки коммерческой тайны;
- в) Несовместимость с клавиатурными сокращениями;
- г) Недостаточная яркость интерфейса.

7. ИИ-ассистент систематически генерирует устаревший API. Эффективное решение:

- а) Отказаться от ИИ-ассистента;
- б) Добавить в .cursorrules указание актуальной версии с примерами корректного API;
- в) Обновить IDE;
- г) Написать постпроцессор вручную.

8. Внедрение ИИ-анализа PR в GitHub Actions. Минимальный набор шагов:

- а) Нанять сотрудника для ревью;
- б) Подключить CodeRabbit через GitHub App, настроить триггер на PR, задать правила в конфиге;
- в) Удалить code review;

г) Дать ИИ право принимать решение о merge.

9. Качество генерации падает на большом файле в Cursor. Причина и решение:

- а) Слабый компьютер;
- б) Исчерпание контекстного окна — разбить на модули, использовать @Codebase с релевантным контекстом;
- в) Длинные имена переменных;
- г) Проблема интернета.

10. Организация требует обработки кода только российскими сервисами. Выбор ИИ-ассистента:

- а) GitHub Copilot (Microsoft/OpenAI);
- б) GigaCode на базе GigaChat (Сбербанк, российская инфраструктура);
- в) Amazon CodeWhisperer;
- г) Tabnine (израильский провайдер).

7.2.4 Примерный перечень вопросов для подготовки к зачету

Не предусмотрено учебным планом

7.2.5 Примерный перечень заданий для подготовки к экзамену

1. Экосистема ИИ-ассистентов разработчика: Cursor, GigaCode, GitHub Copilot, JetBrains AI Assistant.

2. Архитектура и режимы работы Cursor IDE. Настройка .cursorrules.

3. GigaCode и GigaChat API: особенности отечественного ИИ-ассистента.

4. Основы prompt engineering: структура промта, паттерны zero-shot, few-shot, chain-of-thought.

5. Промтинг для кодогенерации: принципы и паттерны.

6. Промтинг для рефакторинга и архитектурного проектирования.

7. ИИ-ассистированное тестирование: генерация unit-тестов и интеграционных тестов.

8. Инструменты ИИ для тестирования: CodiumAI, Cursor, GitHub Copilot Tests.

9. Автоматизированный code review с ИИ: CodeRabbit, GitHub Copilot Code Review.

10. Управление техническим долгом с помощью ИИ.

11. LLM-агенты: принцип работы, инструменты, Cursor Agent, OpenHands.

12. RAG-системы для работы с кодовой базой.

13. Интеграция ИИ-инструментов в CI/CD и DevOps.

14. Правовые и этические аспекты применения ИИ в разработке ПО.

15. Оценка эффективности и ROI от применения ИИ-ассистентов.

7.2.6. Методика выставления оценки при проведении промежуточной аттестации

Экзамен проводится по тест-билетам, каждый из которых содержит 10 вопросов и задачу. Каждый правильный ответ на вопрос в тесте оценивается

1 баллом, задача оценивается в 10 баллов (5 баллов верное решение и 5 баллов за верный ответ). Максимальное количество набранных баллов – 20.

1. Оценка «Неудовлетворительно» ставится в случае, если студент набрал менее 6 баллов.

2. Оценка «Удовлетворительно» ставится в случае, если студент набрал от 6 до 10 баллов

3. Оценка «Хорошо» ставится в случае, если студент набрал от 11 до 15 баллов.

4. Оценка «Отлично» ставится, если студент набрал от 16 до 20 баллов.)

7.2.7 Паспорт оценочных материалов

№ п/п	Контролируемые разделы (темы) дисциплины	Код контролируемой компетенции	Наименование оценочного средства
1	ИИ-ассистенты в современной разработке ПО: обзор, архитектура, возможности	ОПК-2, ОПК-5	Тест, защита лабораторных работ, курсовой работы, защита реферата
2	Промтинг для задач разработки программного обеспечения	ОПК-2, ОПК-5	Тест, защита лабораторных работ, курсовой работы, защита реферата
3	ИИ-ассистированное тестирование программного обеспечения	ОПК-2, ОПК-5	Тест, защита лабораторных работ, курсовой работы, защита реферата
4	ИИ в code review, анализе качества и управлении техническим долгом	ОПК-2, ОПК-5	Тест, защита лабораторных работ, курсовой работы, защита реферата
5	LLM-агенты, RAG-системы и ИИ-автоматизация в DevOps	ОПК-2, ОПК-5	Тест, защита лабораторных работ, курсовой работы, защита реферата
6	Практика применения ИИ-инструментов в полном цикле разработки ПО	ОПК-2, ОПК-5	Тест, защита лабораторных работ, курсовой работы, защита реферата

7.3. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности

Тестирование осуществляется, либо при помощи компьютерной системы тестирования, либо с использованием выданных тест-заданий на бумажном носителе. Время тестирования 30 мин. Затем осуществляется проверка теста экзаменатором и выставляется оценка согласно методики выставления оценки при проведении промежуточной аттестации.

Решение стандартных задач осуществляется, либо при помощи компьютерной системы тестирования, либо с использованием выданных задач на бумажном носителе. Время решения задач 30 мин. Затем осуществляется проверка решения задач экзаменатором и выставляется

оценка, согласно методики выставления оценки при проведении промежуточной аттестации.

Решение прикладных задач осуществляется, либо при помощи компьютерной системы тестирования, либо с использованием выданных задач на бумажном носителе. Время решения задач 30 мин. Затем осуществляется проверка решения задач экзаменатором и выставляется оценка, согласно методики выставления оценки при проведении промежуточной аттестации.

Защита курсовой работы, курсового проекта или отчета по всем видам практик осуществляется согласно требованиям, предъявляемым к работе, описанным в методических материалах. Примерное время защиты на одного студента составляет 20 мин.

8 УЧЕБНО МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ)

8.1 Перечень учебной литературы, необходимой для освоения дисциплины

1. Салмина, Н. Ю. Функциональное программирование и интеллектуальные системы : учебное пособие / Н. Ю. Салмина. — Томск : Томский государственный университет систем управления и радиоэлектроники, 2016. — 100 с. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: <https://www.iprbookshop.ru/72216.html> (дата обращения: 18.06.2026). — Режим доступа: для авторизир. Пользователей

2. Козырева, Г. Ф. Функциональное и логическое программирование : учебно-методическое пособие / Г. Ф. Козырева. — Саратов : Ай Пи Эр Медиа, 2018. — 120 с. — ISBN 978-5-4486-0122-4. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: <https://www.iprbookshop.ru/71596.html> (дата обращения: 18.06.2026). — Режим доступа: для авторизир. пользователей

3. Бутусов, О. Б. Компьютерные методы интеллектуальных информационных систем и дискретной математики : учебное пособие / О. Б. Бутусов, Н. И. Редикульцева. — Москва : Московский гуманитарный университет, 2016. — 156 с. — ISBN 978-5-906822-86-4. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: <https://www.iprbookshop.ru/74726.html> (дата обращения: 18.06.2026). — Режим доступа: для авторизир. пользователей

8.2 Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине, включая перечень лицензионного программного обеспечения, ресурсов информационно-телекоммуникационной сети «Интернет», современных профессиональных баз данных и информационных справочных систем:

8.2.1 Лицензионное программное обеспечение:

Windows Pro Dev UpLic A Each Academic Non-Specific Professional;
Acrobat Pro 2017 Multiple Platforms Russian AOO License TLP.

Свободно распространяемое программное обеспечение:

Python (<https://www.python.org/>); Git (<https://git-scm.com/>);
Visual Studio Code (<https://code.visualstudio.com/>); Cursor IDE
(<https://www.cursor.com/>);
GigaCode (<https://developers.sber.ru/portal/products/gigacode>);
GitHub Copilot для студентов (<https://github.com/features/copilot>);
LibreOffice (<https://libreoffice.org>);
Docker Desktop (<https://www.docker.com/>).

8.2.2 Ресурсы информационно-телекоммуникационной сети «Интернет», современные профессиональные базы данных и информационные справочные системы:

Образовательный портал ВГТУ: <https://old.education.cchgeu.ru/>;
Электронная библиотека ВГТУ: <https://bibl.cchgeu.ru/catalog/>;
Информационно-справочная система ВГТУ: <https://wiki.cchgeu.ru/>;
ЭБС IPR SMART: <https://www.iprbookshop.ru/>;
ЭБС «Лань»: <https://e.lanbook.com/>;
Научная электронная библиотека: www.elibrary.ru;
Портал разработчиков GigaChat API:
<https://developers.sber.ru/portal/products/gigachat>.

9 МАТЕРИАЛЬНО-ТЕХНИЧЕСКАЯ БАЗА, НЕОБХОДИМАЯ ДЛЯ ОСУЩЕСТВЛЕНИЯ ОБРАЗОВАТЕЛЬНОГО ПРОЦЕССА

Для проведения лекционных занятий используется аудитория или лаборатория, оснащенная проектором для проведения лекций. Для проведения лабораторных (практических работ) используется лаборатория, укомплектованная специализированной мебелью для обучающихся и преподавателя, оборудованная техническими средствами обучения: компьютерами с лицензионным программным для проведения лабораторного практикума, с возможностью подключения к сети «Интернет» и доступом в электронную информационно-образовательную среду университета, мультимедиа-проектором, из следующего перечня:

- Лаборатория цифровых технологий (ауд. 202/2);
 - Компьютерная лаборатория (ауд. 215/2)
 - Лаборатория интеллектуальных систем проектирования ауд 215/2 (24-ауд. 215/2)
- Для самостоятельной работы использовать ауд. 200/2

Лаборатории расположены по адресу: 394018, г. Воронеж, Плехановская, 11 (учебный корпус №2).

10. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ (МОДУЛЯ)

По дисциплине «Интеллектуальные технологии разработки программного обеспечения» читаются лекции, проводятся лабораторные работы, выполняется курсовой проект.

Основой изучения дисциплины являются лекции, на которых излагаются наиболее существенные и трудные вопросы, а также вопросы, не нашедшие отражения в учебной литературе.

Лабораторные работы выполняются на лабораторном оборудовании в соответствии с методиками, приведенными в указаниях к выполнению работ.

Методика выполнения курсового проекта изложена в учебно-методическом пособии. Выполнять этапы курсового проекта должны своевременно и в установленные сроки.

Контроль усвоения материала дисциплины производится проверкой курсового проекта, защитой курсового проекта.

Вид учебных занятий	Деятельность студента
Лекция	Написание конспекта лекций: кратко, схематично, последовательно фиксировать основные положения, выводы, формулировки, обобщения; пометать важные мысли, выделять ключевые слова, термины. Проверка терминов, понятий с помощью энциклопедий, словарей, справочников с выписыванием толкований в тетрадь. Обозначение вопросов, терминов, материала, которые вызывают трудности, поиск ответов в рекомендуемой литературе. Если самостоятельно не удастся разобраться в материале, необходимо сформулировать вопрос и задать преподавателю на лекции или на практическом занятии.
Лабораторная работа	Лабораторные работы позволяют научиться применять теоретические знания, полученные на лекции при решении конкретных задач. Чтобы наиболее рационально и полно использовать все возможности лабораторных для подготовки к ним необходимо: следует разобрать лекцию по соответствующей теме, ознакомиться с соответствующим разделом учебника, проработать дополнительную литературу и источники, решить задачи и выполнить другие письменные задания.
Самостоятельная работа	Самостоятельная работа студентов способствует глубокому усвоению учебного материала и развитию навыков самообразования. Самостоятельная работа предполагает следующие составляющие: - работа с текстами: учебниками, справочниками, дополнительной литературой, а также проработка конспектов лекций; - выполнение домашних заданий и расчетов; - работа над темами для самостоятельного изучения; - участие в работе студенческих научных конференций, олимпиад; - подготовка к промежуточной аттестации.
Подготовка к промежуточной аттестации	Готовиться к промежуточной аттестации следует систематически, в течение всего семестра. Интенсивная подготовка должна начинаться не позднее, чем за месяц-полтора до промежуточной аттестации.

	Данные перед экзаменом, экзаменом три дня эффективнее всего использовать для повторения и систематизации материала.
--	---

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

№ п/п	Перечень вносимых изменений	Дата внесения изменений	Подпись заведующего кафедрой, ответственной за реализацию ОПОП
----------	-----------------------------	----------------------------	--

Лист согласования

Автор	Бредихин Алексей Вячеславович	Подписано	08.07.2026 22:56:40
Заведующий кафедрой	Бредихин Алексей Вячеславович	Подписано	08.07.2026 22:56:55
Руководитель образовательной программы	Олейникова Светлана Александровна	Подписано	09.07.2026 10:31:50
Декан факультета	Бредихин Алексей Вячеславович	Утверждено	09.07.2026 11:48:35