Приложение
к ОПОП по специальности
09.02.11 Разработка и управление
программным обеспечением

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ
ПО ПРОФЕССИОНАЛЬНОМУ МОДУЛЮ
ПМ.02 РАЗРАБОТКА И ИНТЕГРАЦИЯ МОДУЛЕЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

для специальности

09.02.11 РАЗРАБОТКА И УПРАВЛЕНИЕ ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ

Базовая подготовка

среднего профессионального образования

(год начала подготовки 2026)

СОДЕРЖАНИЕ

1. ОБЩИЕ ПОЛОЖЕНИЯ

Фонд оценочных средств (ФОС) разработан с целью установления соответствия образовательных достижений студентов требованиям программы подготовки специалистов среднего звена по профессиональному модулю ПМ.02 Разработка и интеграция модулей программного обеспечения для компьютерных систем.

ФОС включают контрольные материалы для проведения текущего контроля и промежуточной аттестации.

ФОС текущего контроля используется для оперативного и регулярного управления учебной деятельностью студентов.

ФОС промежуточной аттестации студентов по профессиональному модулю предназначен для оценки степени достижения запланированных результатов обучения по завершению изучения междисциплинарных курсов профессионального модуля, экзамена (квалификационного) по завершению изучения профессионального модуля в целом.

ФОС разработан на основании:

– программы подготовки специалистов среднего звена по специальности СПО 09.02.11 Разработка и управление программным обеспечением;

– рабочей программы профессионального модуля ПМ.02 Разработка и интеграция модулей программного обеспечения;

– учебного плана по специальности СПО 09.02.11 Разработка и управление программным обеспечением.

2. РЕЗУЛЬТАТЫ ОСВОЕНИЯ ПРОФЕССИОНАЛЬНОГО МОДУЛЯ, ПОДЛЕЖАЩИЕ ПРОВЕРКЕ

Результатом в рамках освоения профессионального модуля *ПМ.02 Разработка и интеграция модулей программного обеспечения* является овладение студентами вида профессиональной деятельности *Разработка и интеграция модулей программного обеспечения*, в том числе профессиональными (ПК) и общими (ОК) компетенциями:

Код	Наименование результата обучения
ОК 1	Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 2	Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности
ОК 3	Планировать и реализовывать собственное профессиональное и личностное развитие, предпринимательскую деятельность в профессиональной сфере, использовать знания по правовой и финансовой грамотности в различных жизненных ситуациях
ОК 4	Эффективно взаимодействовать и работать в коллективе и команде
ОК 5	Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 6	Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных российских духовно-нравственных ценностей, в том числе с учетом гармонизации межнациональных и межрелигиозных отношений, применять стандарты антикоррупционного поведения
ОК 7	Содействовать сохранению окружающей среды, ресурсосбережению, применять знания об изменении климата, принципы бережливого производства, эффективно действовать в чрезвычайных ситуациях
ОК 8	Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности
ОК 9	Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1	Проектировать модули программного обеспечения
ПК 2.2	Разрабатывать модули программного обеспечения
ПК 2.3	Выполнять интеграцию модулей и компонентов программного обеспечения
ПК 2.4	Выполнять тестирование и отладку программного обеспечения
ПК 2.5	Осуществлять документирование программных модулей программного обеспечения

3. ОЦЕНКА УРОВНЕЙ ОСВОЕНИЯ ПРОФЕССИОНАЛЬНОГО МОДУЛЯ

Оценивание уровней сформированности профессиональных и общих компетенций проводится в рамках текущего и промежуточного контроля.

В результате освоения профессионального модуля ПМ.02 Разработка и интеграция модулей программного обеспечения студенты демонстрируют три уровня сформированности профессиональных компетенций: пороговый, базовый и повышенный.

Для каждого конкретного этапа формирования компетенции определены категории «знать», «уметь», «практический опыт», в которые вкладывается следующий смысл:

«приобрести практический опыт» – решать усложненные задачи на основе приобретенных умений и навыков, с их применением в профессиональных деятельности;

«уметь» – решать типичные задачи на основе воспроизведения стандартных алгоритмов решения;

«знать» - воспроизводить и объяснять учебный материал с требуемой степенью научной точности и полноты.

Код ОК, ПК	**Уметь**	**Знать**	**Владеть навыками**
ОК.01Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам	Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам	актуальный профессиональный и социальный контекст, в котором приходится работать и жить; основные источники информации и ресурсы для решения задач и проблем в профессиональном и/или социальном контексте; алгоритмы выполнения работ в профессиональной и смежных областях; методы работы в профессиональной и смежных сферах; структуру плана для решения задач; порядок оценки результатов решения задач профессиональной деятельности	-
ОК.02 Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной	Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной	номенклатура информационных источников, применяемых в профессиональной деятельности; приемы структурирования информации; формат оформления	-

деятельности	деятельности	результатов поиска информации, современные средства и устройства информатизации; порядок их применения и программное обеспечение в профессиональной деятельности в том числе с использованием цифровых средств	
ОК.03 Планировать и реализовывать собственное профессиональное и личностное развитие, предпринимательскую деятельность в профессиональной сфере, использовать знания по правовой и финансовой грамотности в различных жизненных ситуациях	Планировать и реализовывать собственное профессиональное и личностное развитие, предпринимательскую деятельность в профессиональной сфере, использовать знания по правовой и финансовой грамотности в различных жизненных ситуациях	содержание актуальной нормативно-правовой документации; современная научная и профессиональная терминология; возможные траектории профессионального развития и самообразования; основы предпринимательской деятельности; основы финансовой грамотности; правила разработки бизнес-планов; порядок выстраивания презентации; кредитные банковские продукты	-
ОК.04 Эффективно взаимодействовать и работать в коллективе и команде	Эффективно взаимодействовать и работать в коллективе и команде	психологические основы деятельности коллектива, психологические особенности личности; основы проектной деятельности	-
ОК.05 Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста	Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного	особенности социального и культурного контекста; правила оформления документов и построения устных сообщений	-

	контекста		
ОК.06 Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных российских духовно-нравственных ценностей, в том числе с учетом гармонизации межнациональных и межрелигиозных отношений, применять стандарты антикоррупционного поведения	Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных российских духовно-нравственных ценностей, в том числе с учетом гармонизации межнациональных и межрелигиозных отношений, применять стандарты антикоррупционного поведения	сущность гражданско-патриотической позиции, общечеловеческих ценностей; значимость профессиональной деятельности по специальности; стандарты антикоррупционного поведения и последствия его нарушения	-
ОК.07 Содействовать сохранению окружающей среды, ресурсосбережению, применять знания об изменении климата, принципы бережливого производства, эффективно действовать в чрезвычайных ситуациях	Содействовать сохранению окружающей среды, ресурсосбережению, применять знания об изменении климата, принципы бережливого производства, эффективно действовать в чрезвычайных ситуациях	правила экологической безопасности при ведении профессиональной деятельности; основные ресурсы, задействованные в профессиональной деятельности; пути обеспечения ресурсосбережения; принципы бережливого производства; основные направления изменения климатических условий региона	-
ОК.08 Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности	Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности	роль физической культуры в общекультурном, профессиональном и социальном развитии человека; основы здорового образа жизни; условия профессиональной деятельности и зоны риска физического здоровья для специальности; средства	-

		профилактики перенапряжения	
ОК.09 Пользоваться профессиональной документацией на государственном и иностранном языках	Пользоваться профессиональной документацией на государственном и иностранном языках	правила построения простых и сложных предложений на профессиональные темы; основные общеупотребительные глаголы (бытовая и профессиональная лексика); лексический минимум, относящийся к описанию предметов, средств и процессов профессиональной деятельности; особенности произношения; правила чтения текстов профессиональной направленности	-
ПК 2.1 Проектировать модули программного обеспечения	− проектировать модули, соответствующие бизнес-задачам; − создавать архитектурные диаграммы и документацию; − определять структуру и интерфейсы модулей; − анализировать требования к модулю и определять его функциональность; − проектировать архитектуру модуля, включая выбор подходящих паттернов проектирования и структуры данных; − создавать диаграммы классов, последовательностей и прочих диаграмм для визуализации проектируемого модуля;	− основные принципы проектирования модулей программного обеспечения; − языки программирования и технологии для реализации модулей; − паттерны проектирования и структуры данных для создания эффективных и масштабируемых модулей; − методы анализа требований и способов определения функциональности модуля; − принципы создания интерфейсов для взаимодействия с другими модулями и системами; − принципы обеспечения безопасности, производительности и масштабируемости	− проектирования модулей ПО с учетом требований заказчика; − создания архитектурных диаграмм и спецификаций модулей; определения интерфейсов и взаимодействия модулей в системе.

	− выбирать подходящие языки программирования и технологии для реализации модуля; − проектировать интерфейсы программного обеспечения для взаимодействия с другими модулями и системами; − учитывать требования к масштабируемости, производительности и безопасности при проектировании модуля; проводить анализ и оптимизацию проектируемого модуля для повышения его эффективности и качества	при проектировании модулей; методы анализа и оптимизации проектируемых модулей для повышения их эффективности и качества.	
ПК 2.2 Разрабатывать модули программного обеспечения	− разрабатывать модули программного обеспечения с использованием различных языков программирования и технологий; − применять паттерны проектирования и структуры данных для создания эффективных и масштабируемых модулей; − анализировать требования и определять функциональность модуля; − создавать интерфейсы для взаимодействия с другими модулями и системами; − обеспечивать	− язык программирования, основные конструкции, синтаксис; − паттерны проектирования; − структуры данных; − принципы создания интерфейсов для взаимодействия с другими модулями и системами, таких как REST API, SOAP; − работу с инструментальным программным обеспечением; − методы оптимизации кода и алгоритмов; − эффективные алгоритмы и структуры данных для повышения производительности;	− создания модулей программного обеспечения на различных языках программирования; − отладки и тестирования разработанных модулей; − применения структурного и объектно-ориентированного программирования; − оптимизации кода и алгоритмов программных модулей для увеличения производительности; мониторинга и анализа производительности приложений.

	безопасность, производительность и масштабируемость при разработке модулей; − оптимизировать проектируемые модули для повышения их эффективности и качества; − работать с системой контроля версий; − улучшать производительность модулей, выявляя и устраняя узкие места; − проводить анализ и мониторинг производительности приложений; применять инструменты для рефакторинга и оптимизации программного кода.	− многопоточность в программных модулях; − методы оптимизации сетевых протоколов для ускорения обмена данными; − кэширование данных; − управление памятью; техники повышения производительности программного обеспечения	
ПК 2.3 Выполнять интеграцию модулей и компонентов программного обеспечения	− интегрировать модули и компоненты, обеспечивая их взаимодействие; − работать с API и устанавливать соединения между компонентами; − отслеживать и устранять конфликты и ошибки интеграции; − анализировать и определять зависимости между модулями и компонентами; работать с различными форматами данных и протоколами передачи данных	− общие принципы функционирования аппаратных, программных и программно-аппаратных средств администрируемой информационно-коммуникационной системы; − международные стандарты локальных вычислительных сетей; − методы и подходы к интеграции модулей и компонентов; − принципы версионирования и управления изменениями при интеграции; принципы безопасности при интеграции модулей и	− интеграции программных модулей и компонентов в единое программное решение; − работы с API и веб-сервисами для взаимодействия между модулями; − работы с интеграционными платформами и инструментами; обеспечения совместимости и стабильности системы

		компонентов	
ПК 2.4 Выполнять тестирование и отладку программного обеспечения	− анализировать требования к программному обеспечению и составлять планы тестирования; − создавать тестовые сценарии и тест-кейсы для проверки функциональности и соответствия требованиям; − выполнять тестирование программного обеспечения вручную и автоматизировать процесс тестирования; − анализировать результаты тестирования и документировать найденные ошибки; − разрабатывать стратегии отладки и исправлять ошибки в программном обеспечении; − выполнять модульные тесты с использованием инструментов тестирования, в том числе автоматизированного тестирования; − использовать системы контроля дефектов ПО; составлять отчет о выполнении тестирования ПО	− принципы и методы тестирования программного обеспечения; − основы программирования и архитектуры программного обеспечения; − основы баз данных и SQL-запросов; − инструменты для автоматизации тестирования; − основы разработки и отладки программного обеспечения на разных языках программирования; − понятие дефекта программного обеспечения; − критерии качества ПО; − виды и типы тестирования ПО; − техники ручного тестирования; − техники автоматизированного тестирования; − жизненный цикл дефекта ПО; − принципы работы в системе контроля дефектов; основные понятия о качестве ПО	− отладки программного обеспечения на уровне программных модулей; − тестирования программного обеспечения; − формирования тестовых сценариев; − подготовки тестовых платформ (установка операционной системы, дополнительного ПО и другого по необходимости); − оценки объема тестирования ПО с целью определения необходимых ресурсов для его выполнения; − настройки тестовой среды и аппаратных средств для выполнения тестирования ПО в соответствии с заданием на тестирование в пределах своей компетенции; − формирования и представления отчетности о подготовке к выполнению задания на тестирование ПО в соответствии с установленными регламентами; выполнения тестовых процедур на тестовых данных
ПК 2.5 Осуществлять документирование программных модулей	− описывать функциональность модулей в	− стандарты технической документации;	− создания технической документации для

программного обеспечения	документации; − создавать диаграммы для иллюстрации работы модулей; − программировать с использованием комментариев для документирования кода; − использовать специальные метки/теги для отметки важных частей кода в документации; − вести журнал изменений и фиксировать обновления программных модулей; − разбивать модули на логические блоки и описывать каждый блок отдельн; − включать в документацию особенности модулей, такие как ограничения, уязвимости или оптимальные настройки; проводить регулярное обновление документации при изменении модулей или добавлении нового функционала.	− принципы документирования программного обеспечения; инструменты для создания технической документации и комментирования кода	модулей; − документирования кода, API и интерфейсов; работы со специализированным ПО по документированию программного кода

ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ПО ПРОФЕССИОНАЛЬНОМУ МОДУЛЮ ПМ.02 РАЗРАБОТКА И ИНТЕГРАЦИЯ МОДУЛЕЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ (С ПРАВИЛЬНЫМИ ОТВЕТАМИ)

№ п/п	Содержание вопроса	Правильный ответ	Проверяемые компетенции
1.	Какой принцип ООП обеспечивает защиту внутренних данных модуля от несанкционированного изменения и как его правильно применить при разработке?	Инкапсуляция	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
2.	Как полиморфизм позволяет расширять функциональность модуля обработки отчётов без изменения его ядра?	Через использование интерфейсов/абстрактных классов и переопределение методов в наследниках	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
3.	Какую структуру данных целесообразно выбрать для реализации кэша часто используемых записей в модуле с жёстким ограничением по объёму памяти?	Хеш-таблица с политикой вытеснения LRU (Least Recently Used), реализуемая через связный список + хеш-карту	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
4.	Какой алгоритм поиска оптимален для модуля, обрабатывающего динамически поступающие данные с частыми обращениями по уникальному ключу?	Хеш-поиск (на основе хеш-таблицы)	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
5.	Какие метрики связности (cohesion) и зацепления (coupling) являются оптимальными при проектировании программного модуля, и как их достичь?	Высокая связность, низкое зацепление	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
6.	Какой паттерн проектирования следует применить для модуля, который должен уведомлять несколько компонентов об изменении состояния данных, не создавая жёстких связей?	Паттерн «Наблюдатель» (Observer) или событийно-ориентированная шина (Event Bus)	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
7.	Как спроектировать модуль обработки пользовательского ввода, чтобы обеспечить отказоустойчивость, защиту от инъекций и понятную обратную связь?	Многоуровневая валидация (клиентская + модульная) с санитизацией, параметризованными запросами и локализованными сообщениями об ошибках	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9

8.	Какие принципы необходимо учесть при создании модуля графического интерфейса для обеспечения адаптивности и доступности (a11y)?	Responsive design, семантическая разметка, поддержка клавиатурной навигации, контрастность, ARIA-атрибуты.	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
9.	Как реализовать модуль взаимодействия с реляционной БД для выполнения CRUD-операций, обеспечив защиту от SQL-инъекций и оптимизацию запросов?	Использовать параметризованные запросы (prepared statements) или ORM, применять индексы, транзакции и пул соединений.	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
10.	Какие виды тестирования обязательны для программного модуля перед передачей в интеграционную среду, и как организовать автоматизированные модульные тесты?	Модульное (unit), тестирование граничных значений и негативных сценариев. Организация через фреймворки (JUnit, pytest, NUnit) с моками внешних зависимостей.	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
11.	Какой тип интеграционного взаимодействия предпочтителен для связи модулей, работающих в гетерогенной среде (разные языки, ОС), при необходимости строгой типизации контрактов и минимальных накладных расходах?	API на основе gRPC с использованием Protocol Buffers (Protobuf)	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
12.	Как обеспечить согласованность данных между независимыми модулями при выполнении распределённой бизнес-операции, если использование распределённых транзакций (2PC) невозможно?	Паттерн Saga (компенсирующие транзакции) с eventual consistency	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
13.	Какие метрики и механизмы необходимы для оперативного выявления деградации производительности в цепочке взаимосвязанных модулей?	Распределённая трассировка (OpenTelemetry) + метрики RED/USE + health-check эндпоинты	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
14.	Как спроектировать систему логирования, чтобы обеспечить воспроизводимость инцидентов в распределённой среде без блокировки основного потока выполнения?	Асинхронное структурированное логирование с ротацией, уровнями важности и централизованным агрегатором	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
15.	Какие механизмы аутентификации и авторизации обеспечивают безопасное межмодульное взаимодействие в	OAuth 2.0 / OIDC + JWT для внешних API, mTLS +	ПК 2.1 ПК 2.2 ПК 2.3

	микросервисной или модульной архитектуре?	короткоживущие сертификаты для внутренних сервисов	ПК 2.4 ПК 2.5 ОК 1-ОК 9
16.	Как обеспечить защиту интегрированной системы от инъекций и некорректных данных при использовании SQL для синхронизации или поиска информации между модулями?	Параметризованные запросы (prepared statements), ORM с валидацией схем, санитизация входных данных, принцип Zero Trust.	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
17.	Какие архитектурные приёмы позволяют горизонтально масштабировать интегрированную систему при росте пиковой нагрузки без переписывания бизнес-модулей?	Stateless-архитектура, балансировка нагрузки, асинхронные очереди сообщений, кэширование горячих данных.	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
18.	Как оптимизировать модуль, активно использующий SQL-запросы для интеграции с реляционной базой, при увеличении объёма транзакций и числа concurrent-подключений?	Пул соединений, индексация по критичным полям, пакетные операции (batching), анализ планов выполнения (EXPLAIN), денормализация для чтения	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
19.	Какую роль выполняет API Gateway в архитектуре интегрированных модулей и какие кросс-каттинг функции целесообразно вынести на этот компонент?	Единая точка входа, маршрутизация, аутентификация, rate limiting, трансформация протоколов, агрегация ответов, circuit breaker	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
20.	Какие практики CI/CD и контрактного тестирования необходимо внедрить, чтобы гарантировать стабильность интеграции модулей при частых обновлениях?	Consumer-Driven Contract Testing (Pact), семантическое версионирование, автоматизированные интеграционные тесты, feature flags, canary-развёртывание	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
21.	Какие характеристики качества по стандарту ISO/IEC 25010 наиболее критичны для модуля поддержки данных, и как их количественно измерить на этапе эксплуатации?	Надёжность, производительность, сопровождаемость и безопасность.	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
22.	Как статический анализ кода и	Выявляют дефекты	ПК 2.1

	автоматизированные проверки в CI/CD влияют на долгосрочное качество модуля, работающего с данными?	до компиляции/запуска, предотвращают регрессии, обеспечивают единые стандарты кодирования.	ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
23.	Какой метод отладки наиболее эффективен для выявления причин периодического «зависания» модуля при выполнении длительных SQL-операций?	Профилирование + анализ планов выполнения запросов + трассировка блокировок и ожиданий	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
24.	Как организовать безопасную пошаговую отладку модуля, взаимодействующего с внешней БД и API, без риска повреждения продуктивных данных?	Изолированное тестовое окружение, моки/стабы зависимостей, условные точки останова, логирование уровня DEBUG	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
25.	Какие артефакты необходимо собрать для диагностики intermittent-сбоев (прерывистых ошибок) в модуле обработки транзакций?	Сквозные идентификаторы запросов, полные трейсы, дампы состояния, логи БД (slow query, deadlocks), метрики ресурсов в момент сбоя	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
26.	Как правильно спроектировать обработку исключений в модуле CRUD-операций через SQL, чтобы обеспечить стабильность и понятную обратную связь?	Классификация исключений, retry-политика для transient-ошибок, логирование на границе модуля, пользовательские сообщения без технических деталей	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
27.	Почему перехват общего исключения (catch (Exception e)) в модуле работы с данными считается антипаттерном и как его заменить?	Скрывает конкретные причины, нарушает гарантированное освобождение ресурсов, усложняет диагностику. Заменяется на конкретные типы + using/finally	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
28.	Какие виды тестирования обязательны для модуля, формирующего динамические SQL-	Unit-тесты (генерация/валидаци	ПК 2.1 ПК 2.2

	запросы на основе пользовательского ввода, и как их автоматизировать?	я), интеграционные тесты (реальная БД), security-тесты (SQL-инъекции), нагрузочные тесты	ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
29.	Как организовать тестирование модуля поддержки и обновления схем БД (миграций) без потери тестовых данных и с возможностью автоматического отката?	Версионирование миграций, транзакционные тесты с rollback, фикстуры/сиды, инструменты Flyway/Liquibase	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
30.	Как оценить достаточность тестового покрытия модуля обработки данных и какие метрики, помимо процентного покрытия кода, необходимо анализировать?	Анализ покрытия ветвей/условий, мутационное тестирование, покрытие критических бизнес-путей, качество assertions	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
31.	Какие этапы включает процесс построения математической модели для оптимизации распределения вычислительных ресурсов в корпоративной ИТ-инфраструктуре?	Постановка задачи → формализация → построение модели → верификация → валидация → анализ результатов → внедрение и сопровождение	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
32.	В каких случаях для планирования задач в ИТ-проектах целесообразно применять линейное программирование, и какие математические условия должны выполняться?	При линейной целевой функции и линейных ограничениях; решается симплекс-методом или внутренними точками	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
33.	Почему линейные модели неприменимы для оптимизации энергопотребления дата-центров или обучения нейросетей, и какие методы НЛП используются в таких задачах?	Из-за нелинейной зависимости затрат от нагрузки; применяются градиентные методы, Лагранж, эвристики при невыпуклости	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
34.	Как принцип оптимальности Беллмана позволяет решать задачи многоэтапного планирования ресурсов с учетом изменяющихся приоритетов?	Разбиение на перекрывающиеся подзадачи, рекуррентные соотношения, хранение промежуточных результатов	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9

		(мемоизация/табуляция)	
35.	Как определить критический путь в проекте разработки ПО и какие резервы времени позволяют оптимизировать график без срыва дедлайнов?	Критический путь — максимальная по длительности цепочка задач без резервов; резервы: полный и свободный	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
36.	Как рассчитать вероятность блокировки запросов и среднее время отклика в веб-сервисе с ограниченным числом рабочих потоков?	Через модели M/M/c или M/G/1, формулы Эрланга B/C и Литтла	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
37.	Как применить теорию игр для моделирования стратегий распределения ограниченных вычислительных ресурсов между конкурирующими модулями в мультитенантной среде?	Поиск равновесия Нэша в некооперативной игре, анализ платежных матриц, проверка доминирующих стратегий	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
38.	В каких случаях имитационное моделирование предпочтительнее аналитических методов, и как обеспечить статистическую достоверность результатов при моделировании сложных ИТ-процессов?	При высокой нелинейности, стохастичности и отсутствии замкнутого решения; достоверность достигается методом Монте-Карло, верификацией и калибровкой	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
39.	Как спроектировать схему реляционной БД для хранения параметров моделей, результатов оптимизации и истории прогонов с целью автоматизации принятия решений?	Нормализованные таблицы: model_configs, input_datasets, optimization_results, run_logs; API-слой для связи модели и БД	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
40.	Какие процедуры необходимы для подтверждения адекватности математической модели реальным процессам и как обеспечить её актуальность при изменении бизнес-условий?	Верификация (правильность кода/реализации), валидация (соответствие реальности), периодическая калибровка, мониторинг data drift	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
41.	Почему при реализации численных алгоритмов в программных модулях критично учитывать абсолютную и относительную	Погрешности накапливаются при вычислениях; выбор	ПК 2.1 ПК 2.2 ПК 2.3

	погрешности, и как это влияет на выбор типа данных?	типа данных (float/double/decimal) определяет точность и устойчивость алгоритма	ПК 2.4 ПК 2.5 ОК 1-ОК 9
42.	В каких случаях метод Ньютона предпочтительнее метода простой итерации для решения нелинейных уравнений в модуле расчёта параметров системы?	При наличии аналитической производной и хорошем начальном приближении — метод Ньютона обеспечивает квадратичную сходимость	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
43.	Как обеспечить гарантированную сходимость численного метода решения уравнения при отсутствии априорной информации о поведении функции?	Использовать комбинированный метод (например, Ньютона + дихотомии) с автоматическим переключением при нарушении условий сходимости	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
44.	Почему при программной реализации метода Гаусса необходимо использовать схему с выбором главного элемента, и как это влияет на устойчивость решения?	Выбор главного элемента (по строке/столбцу) минимизирует влияние ошибок округления и предотвращает деление на малые числа	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
45.	В каких случаях для решения больших разреженных СЛАУ целесообразно применять итерационные методы (Якоби, Зейделя, сопряжённых градиентов) вместо прямых?	При большой размерности ($n > 10^4$) и разреженности матрицы — итерационные методы экономят память и время, требуя только умножения матрицы на вектор	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
46.	Какой метод интерполяции выбрать для построения прогнозной модели на основе дискретных измерений, и как оценить погрешность экстраполяции?	Для гладких данных — сплайны (кубические); для зашумлённых — сглаживающие сплайны или МНК. Погрешность экстраполяции растёт экспоненциально за	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9

		пределами узлового интервала	
47.	Как выбрать шаг интегрирования в методе Симпсона для расчёта площади под кривой с заданной точностью, и как адаптировать алгоритм для неравномерных данных из БД?	Использовать правило Рунге для оценки погрешности и адаптивное измельчение шага; для неравномерных данных — кусочно-полиномиальная аппроксимация	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
48.	Почему метод Рунге-Кутты 4-го порядка часто используется в модулях моделирования динамических систем, и как контролировать глобальную погрешность?	Обеспечивает баланс точности ($O(h^4)$) и вычислительной сложности; глобальная погрешность контролируется через шаг и сравнение с методом меньшего порядка	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
49.	Как выбрать шаг обучения в градиентном спуске для минимизации функции потерь в модуле машинного обучения, и как ускорить сходимость?	Использовать адаптивные методы (Adam, RMSProp) или поиск по золотому сечению на линии спуска; ускорение — через импульс и нормализацию признаков	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
50.	Как спроектировать модуль, который автоматически запускает численный расчёт при поступлении новых данных в БД и сохраняет результаты с метаданными о точности?	Использовать триггеры/очереди событий + фоновый воркер + транзакционную запись результатов с полями error_estimate, convergence_status, computation_time	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
51.	Какие три базовых принципа информационной безопасности (триада CIA) должны учитываться при проектировании модуля обработки персональных данных, и как их реализовать на архитектурном уровне?	Конфиденциальность, Целостность, Доступность	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
52.	На каких этапах жизненного цикла разработки ПО необходимо внедрять практики безопасности, и почему подход «Shift-Left» снижает стоимость устранения уязвимостей?	На всех этапах SDLC. Shift-Left — перенос проверок безопасности на ранние стадии	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5

		(требования, дизайн, код)	ОК 1-ОК 9
53.	Почему конкатенация строк при формировании SQL-запросов является критической уязвимостью, и какой механизм разработки гарантированно предотвращает SQL-инъекции?	Конкатенация позволяет внедрить произвольный SQL-код. Защита: параметризованные запросы (prepared statements) и ORM	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
54.	Почему для хранения паролей пользователей в БД нельзя использовать обычные хеш-функции (MD5, SHA-256), и какой алгоритм следует применять?	MD5/SHA оптимизированы под скорость и уязвимы к радужным таблицам/брутфорсу. Применять адаптивные функции: Argon2, bcrypt, scrypt, PBKDF2	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
55.	В каких сценариях разработки модулей применяется симметричное, а в каких асимметричное шифрование, и почему их часто комбинируют?	Симметричное — для шифрования больших объёмов данных. Асимметричное — для обмена ключами, ЭЦП, аутентификации. Комбинация решает проблему безопасной передачи симметричного ключа.	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
56.	Какие правила хранения и ротации криптографических ключей необходимо соблюдать в программном модуле, и почему хардкод ключей в коде недопустим?	Ключи хранить в защищённых хранилищах (KMS, Vault, HSM), ротировать по расписанию/инциденту, никогда не хардкодить.	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
57.	Как принцип наименьших привилегий (PoLP) применяется при настройке подключения модуля к реляционной БД, и какие риски он снижает?	Выделение учётной записи БД с минимально необходимыми правами только для конкретных объектов и операций.	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
58.	Почему подробные стектрейсы и технические детали ошибок не должны попадать в ответ клиенту, и как правильно организовать логирование для расследования инцидентов?	Технические детали раскрывают архитектуру и уязвимости	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4

		(information disclosure). Клиенту — понятные сообщения, логам — полные детали в безопасном хранилище.	ПК 2.5 ОК 1-ОК 9
59.	Как гарантировать, что конфигурационные файлы или обновления модуля не были изменены злоумышленником при передаче или хранении?	Использовать цифровую подпись (ЭЦП) или HMAC для проверки целостности и аутентичности	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
60.	Какие методы шифрования данных «в покое» (в БД) применимы для модулей обработки чувствительной информации, и как они влияют на производительность и поиск?	Transparent Data Encryption (TDE) на уровне СУБД или шифрование на уровне приложения (полевое/колонковое). TDE не влияет на поиск, полевое — требует дешифровки для WHERE/JOIN	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9

ТЕСТ
(с правильными ответами)

№ пп	Содержание вопроса	Правильный ответ	Проверяемые компетенции
1.	**Какой принцип объектно-ориентированного программирования обеспечивает сокрытие внутренней реализации класса и доступ к его данным только через публичные методы?** А) Наследование Б) Полиморфизм В) Инкапсуляция Г) Абстракция	**В**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
2.	**Какое преимущество даёт полиморфизм при разработке расширяемых программных модулей?** А) Уменьшает объём занимаемой оперативной памяти Б) Позволяет обрабатывать объекты разных классов через единый интерфейс без изменения кода ядра В) Автоматически оптимизирует алгоритмы сортировки Г) Гарантирует потокобезопасность при многозадачности	**Б**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
3.	**Какая комбинация характеристик считается эталонной при проектировании независимых программных модулей?** А) Низкая связность и высокое зацепление Б) Высокая связность и низкое зацепление В) Высокая связность и высокое зацепление	**Б**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9

	Г) Низкая связность и низкое зацепление		
4.	**Какая структура данных обеспечивает среднее время поиска, вставки и удаления элементов O(1) при работе с парами «ключ-значение»?** А) Двусвязный список Б) Массив фиксированной длины В) Хеш-таблица Г) Сбалансированное бинарное дерево	**В**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
5.	**Какой алгоритм поиска является наиболее эффективным для отсортированного по возрастанию массива?** А) Линейный поиск Б) Бинарный поиск В) Поиск в глубину Г) Интерполяционный поиск	**Б**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
6.	**Какой архитектурный паттерн разделяет модуль на три слоя: модель (данные), представление (интерфейс) и контроллер (обработка ввода)?** А) MVC (Model-View-Controller) Б) Singleton В) Factory Method Г) Observer	**А**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
7.	**Почему в модулях пользовательского интерфейса рекомендуется выполнять длительные операции (запросы к БД, сетевые вызовы) асинхронно?** А) Чтобы уменьшить объём кода Б) Чтобы предотвратить блокировку основного потока и «зависание» интерфейса В) Чтобы автоматически повысить точность вычислений Г) Чтобы исключить необходимость валидации входных данных	**Б**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
8.	**Какая практика является обязательной при разработке модулей обработки пользовательского ввода?** А) Автоматическое приведение всех типов данных к строке Б) Сохранение всех введённых данных в открытом виде для аудита В) Отключение кнопок интерфейса на 5 секунд после каждого нажатия Г) Многоуровневая валидация (клиентская + серверная/модульная)	**Г**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
9.	**Какая структура данных наиболее подходит для реализации механизма «Отменить последнее действие» (Undo) в графическом редакторе?** А) Очередь (Queue) Б) Стек (Stack) В) Двоичная куча Г) Хеш-множество (HashSet)	**Б**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
10.	**Какой принцип SOLID требует, чтобы классы модуля зависели от абстракций (интерфейсов), а**	**В**	ПК 2.1 ПК 2.2

	не от конкретных реализаций? А) Принцип единственной ответственности (SRP) Б) Принцип подстановки Лисков (LSP) В) Принцип инверсии зависимостей (DIP) Г) Принцип разделения интерфейса (ISP)		ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
11.	**Какой тип интеграционного взаимодействия предполагает, что модуль-отправитель помещает задачу в очередь и не ожидает мгновенного ответа от модуля-получателя?** А) Синхронный вызов REST/SOAP API Б) Асинхронная интеграция через брокер сообщений В) Прямое обращение к базе данных получателя Г) Интеграция через общие текстовые файлы	**Б**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
12.	**Что является основным элементом, гарантирующим совместимость и однозначное понимание данных при взаимодействии независимых программных модулей, разработанных на разных платформах?** А) Контракт API (спецификация интерфейса и форматов данных) Б) Единая версия языка программирования В) Использование одной операционной системы Г) Наличие общего конфигурационного файла в репозитории	**А**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
13.	**Для чего в логах распределённой системы используется сквозной идентификатор запроса (Correlation ID)?** А) Для шифрования передаваемых данных между модулями Б) Для объединения всех записей логов, относящихся к одному пользовательскому сценарию В) Для автоматического масштабирования контейнеров Г) Для проверки цифровых подписей ответов	**Б**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
14.	**Какой тип проверки работоспособности (Health Check) сообщает оркестратору, что модуль запущен, но ещё не готов обрабатывать входящий трафик (например, инициализирует кэш или подключение к БД)?** А) Liveness probe Б) Security probe В) Startup probe Г) Readiness probe	**Г**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
15.	**Какой механизм наиболее эффективно защищает внутренние API-вызовы между микросервисами от подмены источника и перехвата данных в корпоративной сети?** А) Передача логинов и паролей в заголовках HTTP-запросов Б) Взаимная аутентификация по сертификатам (mTLS) В) Отключение брандмауэра для ускорения обмена	**Б**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9

	Г) Хранение токенов доступа в localStorage браузера		
16.	**Какое правило обязательно должно соблюдаться при обработке данных, поступающих от внешних модулей или пользовательского интерфейса?** А) Доверять данным, если запрос пришёл из внутренней сети Б) Игнорировать проверку типов для ускорения обработки В) Валидировать и санитизировать все входные данные на границе модуля Г) Сохранять исходный формат данных без изменений	**В**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
17.	**Оптимизация и масштабируемость интегрированных решений** **Вопрос: Какое архитектурное свойство программного модуля позволяет без ограничений запускать дополнительные его копии на новых серверах для обработки возросшей нагрузки?** А) Stateful (хранение сессий и состояния в памяти модуля) Б) Stateless (отсутствие сохраняемого состояния между запросами) В) Монолитная структура с жёсткими связями Г) Привязка к конкретному MAC-адресу сетевого интерфейса	**Б**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
18.	**Оптимизация и масштабируемость интегрированных решений** **Вопрос: Какую проблему в интегрированной системе решает паттерн «Circuit Breaker» (Автоматический выключатель)?** А) Предотвращение каскадных отказов при недоступности зависимого модуля Б) Автоматическое шифрование данных в реляционной БД В) Оптимизацию SQL-запросов к внешней базе данных Г) Балансировку нагрузки по алгоритму Round Robin	**А**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
19.	**Какой компонент интеграционной архитектуры позволяет временно накапливать пиковый поток запросов и передавать их модулям-обработчикам в комфортном темпе?** А) Кэш Redis Б) Очередь сообщений (Message Queue) В) Шлюз API Gateway Г) Балансировщик нагрузки L4	**Б**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
20.	**Какой показатель методологии RED/USE наиболее точно отражает качество обслуживания пользователей в веб-модуле?** А) Utilization (Утилизация CPU) Б) Saturation (Насыщенность дисковой подсистемы) В) Error Rate (Частота ошибок в ответах) Г) Queue Length (Длина очереди сообщений)	**В**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9

21.	**Какой международный стандарт описывает модель качества ПО, включая такие характеристики, как функциональная пригодность, надёжность, производительность, сопровождаемость и безопасность?** А) ISO/IEC 12207 Б) ISO/IEC 25010 В) IEEE 829 Г) ITIL v4	**Б**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
22.	**Что означает принцип «Shift-Left» в контексте обеспечения качества программного обеспечения?** А) Встраивание проверок качества, безопасности и тестирования на ранние этапы жизненного цикла Б) Перенос всех тестов на этап приёмки заказчиком В) Отказ от автоматизации в пользу ручных проверок Г) Сокращение количества этапов разработки до двух	**А**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
23.	**Какой инструментальный подход наиболее эффективен для диагностики прерывистых (intermittent) сбоев в распределённой системе, которые сложно воспроизвести локально?** А) Пошаговое выполнение в отладчике IDE Б) Увеличение таймаута сервера до бесконечности В) Добавление операторов вывода в консоль (print/console.log) Г) Анализ структурированных логов с использованием Correlation ID и распределённой трассировки	**Г**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
24.	**Для чего в интегрированной среде разработки используются условные точки останова (conditional breakpoints)?** А) Для автоматического исправления синтаксических ошибок Б) Для приостановки выполнения программы только при выполнении заданного логического условия В) Для шифрования значений переменных во время отладки Г) Для ускорения процесса компиляции кода	**Б**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
25.	**Почему перехват базового исключения catch (Exception e) в начале цепочки вызовов считается антипаттерном?** А) Он значительно ускоряет выполнение программы Б) Он автоматически создаёт резервные копии данных В) Он скрывает конкретные причины ошибок, усложняет диагностику и препятствует адресному восстановлению Г) Он предотвращает все виды утечек памяти	**В**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
26.	**Какая стратегия обработки временных сбоев (transient errors) при вызове внешних сервисов или БД обеспечивает максимальную устойчивость модуля?**	**Б**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4

	А) Мгновенный повторный вызов без задержки Б) Политика повторных попыток с экспоненциальной задержкой (Exponential Backoff) и случайным смещением (jitter) В) Игнорирование ошибки и возврат значения по умолчанию Г) Блокировка потока до ручного перезапуска сервиса		ПК 2.5 ОК 1-ОК 9
27.	**Какой вид тестирования проверяет корректность работы изолированного метода или класса, заменяя внешние зависимости искусственными объектами?** А) Интеграционное тестирование Б) Системное тестирование В) Модульное (Unit) тестирование Г) Нагрузочное тестирование	**В**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
28.	**Что показывают результаты мутационного тестирования (Mutation Testing) в отличие от традиционного процентного покрытия кода?** А) Скорость выполнения тестов в секундах Б) Способность существующих тестов обнаруживать искусственно внесённые дефекты В) Количество написанных строк кода разработчиком Г) Соответствие интерфейса дизайн-макетам	**Б**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
29.	**Какой метрикой покрытия кода следует руководствоваться для гарантии проверки всех возможных ветвлений логических условий (if/else, тернарные операторы)?** А) Покрытие ветвей (Branch Coverage) Б) Покрытие строк (Line Coverage) В) Покрытие операторов (Statement Coverage) Г) Покрытие комментариев	**А**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
30.	**Какую ключевую задачу решает автоматический запуск регрессионных тестов в пайплайне непрерывной интеграции (CI) перед слиянием кода?** А) Генерацию пользовательской документации Б) Предотвращение внесения дефектов в уже работающую функциональность В) Оптимизацию алгоритмов сортировки Г) Ручную приёмку продукта менеджером	**Б**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
31.	**Какой этап математического моделирования предполагает проверку соответствия результатов модели реальному объекту или процессу с использованием экспериментальных данных?** А) Формализация Б) Верификация В) Валидация	**В**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9

	Г) Оптимизация		
32.	**В задаче линейного программирования целевая функция и все ограничения должны быть...** А) Квадратичными Б) Линейными В) Логарифмическими Г) Дискретными	**Б**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
33.	**Какой алгоритм является классическим универсальным методом решения задач линейного программирования в стандартной форме?** А) Симплекс-метод Б) Метод Ньютона В) Метод Монте-Карло Г) Градиентный спуск	**А**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
34.	**В каком случае для оптимизации целевой функции НЕЛЬЗЯ применять методы линейного программирования и необходимо использовать нелинейное программирование?** А) Когда переменные принимают только целые значения Б) Когда целевая функция или ограничения содержат квадраты, произведения или экспоненты переменных В) Когда количество ограничений превышает количество переменных Г) Когда задача решается на графическом процессоре	**Б**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
35.	**Какой фундаментальный принцип лежит в основе динамического программирования для решения многошаговых задач оптимизации?** А) Принцип случайного поиска Б) Принцип наименьших квадратов В) Принцип оптимальности Беллмана Г) Принцип максимума энтропии	**В**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
36.	**Что представляет собой «критический путь» в сетевом графике проекта (метод CPM/PERT)?** А) Маршрут с минимальным количеством задач Б) Последовательность задач с максимальным общим временем выполнения, определяющая минимальную продолжительность проекта В) Путь, по которому передаются данные между серверами Г) Набор задач, которые можно выполнять параллельно без ограничений	**Б**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
37.	**Какая характеристика определяет интенсивность нагрузки (ρ) в простейшей одноканальной СМО и что происходит при $\rho \geq 1$?**	**Г**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4

	А) $\rho = \mu/\lambda$; при $\rho \geq 1$ очередь быстро уменьшается Б) $\rho = \lambda - \mu$; при $\rho \geq 1$ обслуживание прекращается мгновенно В) $\rho = \lambda + \mu$; при $\rho \geq 1$ система переходит в спящий режим Г) $\rho = \lambda/\mu$; при $\rho \geq 1$ очередь неограниченно растёт, система перегружена		ПК 2.5 ОК 1-ОК 9
38.	**Что такое «равновесие Нэша» в некооперативной игре?** А) Состояние, при котором один игрок максимизирует выигрыш за счёт проигрыша другого Б) Ситуация, в которой ни одному игроку не выгодно в одиночку изменять свою стратегию при неизменных стратегиях остальных В) Решение, гарантирующее максимальный выигрыш всем участникам одновременно Г) Стратегия, основанная на случайном выборе ходов	**Б**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
39.	**В каком случае имитационное моделирование предпочтительнее аналитических математических методов?** А) Когда система обладает высокой сложностью, нелинейностью, стохастичностью и не имеет замкнутого аналитического решения Б) Когда система описывается линейными уравнениями с постоянными коэффициентами В) Когда требуется найти точный корень квадратного уравнения Г) Когда объём данных не превышает 100 записей	**А**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
40.	**Какой метод оценки достоверности результатов имитационной модели предполагает многократный запуск с различными случайными генераторами для построения доверительных интервалов?** А) Метод наименьших квадратов Б) Метод Монте-Карло В) Симплекс-метод Г) Метод Ньютона	**Б**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
41.	**Что характеризует относительная погрешность приближённого числа?** А) Разность между точным и приближённым значением Б) Отношение абсолютной погрешности к модулю точного значения В) Количество верных знаков после запятой Г) Максимально возможное отклонение в единицах	**Б**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9

	измерения		
42.	**Какой метод решения нелинейных уравнений гарантирует сходимость при выполнении условия $f(a)\cdot f(b) < 0$ на отрезке [a, b]?** А) Метод Ньютона Б) Метод простой итерации В) Метод дихотомии (половинного деления) Г) Метод секущих	**В**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
43.	**Зачем при реализации метода Гаусса используется выбор главного элемента?** А) Для уменьшения количества арифметических операций Б) Для минимизации накопления ошибок округления и повышения устойчивости алгоритма В) Для возможности решения только квадратных матриц Г) Для автоматического определения ранга матрицы	**Б**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
44.	**Какой метод интерполяции предпочтительнее для гладких функций при большом количестве узлов, чтобы избежать явления Рунге?** А) Интерполяционный полином Лагранжа высокой степени Б) Кубические сплайны В) Линейная интерполяция Г) Интерполяция Ньютона с конечными разностями	**Б**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
45.	**Какой метод численного интегрирования имеет порядок точности $O(h^4)$?** А) Метод прямоугольников Б) Метод трапеций В) Метод Симпсона Г) Метод Монте-Карло	**В**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
46.	**Какой метод решения обыкновенных дифференциальных уравнений является одношаговым и имеет четвёртый порядок точности?** А) Метод Эйлера Б) Метод Рунге-Кутты 4-го порядка В) Метод Адамса Г) Метод прогноза-коррекции	**Б**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
47.	**Какой метод численной оптимизации использует информацию о градиенте функции для поиска минимума?** А) Метод золотого сечения Б) Градиентный спуск В) Метод Нелдера-Мида	**Б**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9

	Г) Генетический алгоритм		
48.	**В чём основное преимущество метода Ньютона перед методом простой итерации?** А) Не требует вычисления производной функции Б) Гарантирует сходимость при любом начальном приближении В) Обеспечивает квадратичную скорость сходимости вблизи корня Г) Применим только к линейным уравнениям	**В**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
49.	**В каком случае итерационные методы (Якоби, Зейделя) предпочтительнее прямых методов для решения СЛАУ?** А) Когда матрица системы имеет малую размерность ($n < 10$) Б) Когда матрица системы является разреженной и большой размерности В) Когда требуется точное решение без погрешностей округления Г) Когда матрица системы является вырожденной	**Б**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
50.	**Для чего используется правило Рунге в численном интегрировании?** А) Для оценки погрешности и адаптивного выбора шага интегрирования Б) Для выбора оптимального метода интегрирования В) Для преобразования пределов интегрирования Г) Для замены подынтегральной функции полиномом	**А**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
51.	**Какая триада принципов составляет основу информационной безопасности программного обеспечения?** А) Скорость, Надёжность, Масштабируемость Б) Конфиденциальность, Целостность, Доступность (CIA) В) Аутентификация, Авторизация, Учёт (AAA) Г) Простота, Эффективность, Удобство (PEU)	**Б**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
52.	**Что означает подход «Shift-Left» в контексте безопасной разработки программного обеспечения?** А) Перенос тестирования безопасности на этап промышленной эксплуатации Б) Встраивание практик безопасности на ранние этапы жизненного цикла разработки (требования, дизайн, код) В) Отказ от использования сторонних библиотек в пользу самописного кода Г) Сокращение количества разработчиков в команде	**Б**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9

	безопасности		
53.	**Какой механизм гарантированно предотвращает атаку типа «SQL-инъекция» при формировании запросов к реляционной базе данных?** А) Экранирование специальных символов вручную Б) Использование параметризованных запросов (prepared statements) или ORM В) Ограничение длины пользовательского ввода до 100 символов Г) Сохранение пользовательского ввода в лог-файл перед выполнением	**Б**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
54.	**Почему для хранения паролей пользователей НЕЛЬЗЯ использовать быстрые криптографические хеш-функции (MD5, SHA-256)?** А) Они занимают слишком много места в базе данных Б) Они уязвимы к атакам перебором (брутфорс) и радужным таблицам из-за высокой скорости вычисления В) Они не поддерживают кодировку Unicode Г) Они автоматически удаляют соль после первого использования	**Б**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
55.	**В каком сценарии целесообразно применять асимметричное шифрование (RSA, ECC)?** А) Для шифрования больших объёмов данных в базе данных Б) Для безопасной передачи симметричного ключа или создания электронной подписи В) Для сжатия файлов перед отправкой по сети Г) Для генерации псевдослучайных чисел в игровом модуле	**Б**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
56.	**Почему хранение криптографических ключей прямо в исходном коде приложения (хардкод) является критической уязвимостью?** А) Это увеличивает размер исполняемого файла Б) Ключ может быть скомпрометирован при утечке репозитория, что приведёт к расшифровке всех защищённых данных В) Это замедляет процесс компиляции приложения Г) Ключи нельзя изменить без переписывания всего кода	**Б**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
57.	**Что требует принцип наименьших привилегий (PoLP) при настройке учётной записи приложения для работы с базой данных?**	**Б**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4

	А) Предоставление прав sa/root для универсальности Б) Выделение минимально необходимых прав только для конкретных операций и объектов БД В) Использование одной учётной записи для всех модулей системы Г) Отключение проверки пароля для ускорения подключения		ПК 2.5 ОК 1-ОК 9
58.	**Почему в ответе клиенту не следует возвращать детальный стектрейс или текст исключения базы данных?** А) Это увеличивает время отклика сервера Б) Техническая информация может раскрыть структуру системы и помочь злоумышленнику в проведении атаки (information disclosure) В) Это нарушает требования к кодировке ответов Г) Стектрейсы занимают слишком много места в логах	**Б**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
59.	**Какой механизм позволяет одновременно гарантировать целостность данных и подтвердить их авторство при передаче между модулями?** А) Контрольная сумма CRC32 Б) Цифровая подпись на основе асимметричной криптографии В) Сжатие данных алгоритмом ZIP Г) Кодирование данных в Base64	**Б**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9
60.	**Какое преимущество даёт использование Transparent Data Encryption (TDE) на уровне СУБД?** А) Автоматическое исправление уязвимостей в коде приложения Б) Прозрачное шифрование файлов БД и бэкапов на диске без изменений в коде приложения В) Ускорение выполнения запросов за счёт аппаратного шифрования Г) Полную защиту от SQL-инъекций	**Б**	ПК 2.1 ПК 2.2 ПК 2.3 ПК 2.4 ПК 2.5 ОК 1-ОК 9

4. МАТЕРИАЛЫ ДЛЯ ПРОВЕДЕНИЯ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ПО ПРОФЕССИОНАЛЬНОМУ МОДУЛЮ

4.1. Типовые задания для оценки освоения МДК.02.01 Разработка программных модулей

Вопросы для подготовки к экзамену/зачёту

1. Модульная архитектура построения приложений. Принципы. Преимущества. Примеры приложений
2. Архитектурные шаблоны, применяемые при разработке программных модулей (MVC, MVVM, MVP)
3. Инструменты разработки приложений с модульной архитектурой. Системы контроля версий.
4. Работа с библиотеками (применение стандартных библиотек, создание библиотек). Базовые принципы работы с массивами, коллекциями, строками. Работа с датой и временем.
5. Паттерны проектирования: отношения между классами и объектами (наследование, реализация, ассоциация, композиция, агрегация), интерфейсы, абстрактные классы, порождающие паттерны, паттерны поведения, структурные паттерны, поведенческие паттерны, паттерны объектов.
6. Система ввода-вывода, средства доступа к файлам и папкам файловой системы, чтения/записи, сжатия потоков и механизмов изолированного хранения.
7. Работа со строками, регулярными выражениями, кодирование/декодирование текста.
8. Асинхронная модель программирования. Пул потоков. Шаблон асинхронного вызова методов. Синхронизация вызывающего потока. Передача и прием специальных данных состояния.
9. Параллельное программирование. Создание задачи. Методы ожидания выполнения задачи. Лямбда-выражения в качестве задачи. Создание продолжения задачи. Возврат значений из задачи. Отмена задачи.
10. Алгоритмы и структуры данных. Оценка сложности алгоритмов. Понятие асимптотической оценки. Большие О-нотации. Временная сложность алгоритма. Пространственная сложность алгоритма. Анализ худшего, лучшего и среднего случаев.
11. Основные структуры данных (массив, связный список, стек, очередь; операции вставки, поиска и удаления; представление данных в памяти).
12. Алгоритмы сортировки и поиска. Основы рекурсии: примеры, преимущества и недостатки.
13. Хеш-таблица и хеш-функция. Коллизии и разрешение коллизий. Методы хеширования и сжатия данных. Эффективность и применение хеш-структур.
14. Деревья и графы. Представление графов и деревьев. Поиск в глубину и ширину. Минимум затратный путь (алгоритм Дейкстры). Деревья поиска и обхода.
15. Жадные алгоритмы и динамическое программирование. Основные идеи динамического программирования.
16. Алгоритмы работы с текстовыми данными. Операции над строками. Поиск подстроки (наивный алгоритм поиска, алгоритм Кнута-Морриса-Пратта, алгоритм Бойера-Мура). Проблемы на строках (Задача о рюкзаке, редакционное расстояние). Алгоритмы с использованием хеширования (хеш-функции для строк, алгоритм Рабина-Карпа). Строки и структуры данных (операции с динамическими строками, триальные деревья)
17. Кучи и очереди. Очереди с приоритетом и кучи. Куча и ее применение.
18. Основные принципы проектирования модулей программного обеспечения. Методы анализа требований и способов определения функциональности модуля. Методы анализа и оптимизации проектируемых модулей для повышения их эффективности и качества. Декомпозиция задачи на подзадачи. Создание спецификаций модуля.

19. Принципы обеспечения безопасности, производительности и масштабируемости при проектировании модулей

20. Принципы проектирования классов. Проектирование классов с учётом инкапсуляции. Использование наследования: создание иерархий классов. Полиморфизм: перегрузка методов и интерфейсов.

21. Применение диаграмм классов при проектировании требований к внутренней структуре программного модуля.

22. Применение диаграмм компонентов для визуализации организации компонентов проектируемого модуля

23. Виды пользовательского интерфейса (командная строка, графический, речевой). Основные этапы и принципы разработки графического пользовательского интерфейса.

24. Технологии и инструменты разработки графического пользовательского интерфейса.

25. Компоненты графического пользовательского интерфейса. Типы элементов управления. Компоновка элементов управления. События. Обработчики событий.

26. Работа с окнами. Основные методы работы с окнами. Создание окна: функции и классы. Открытие и закрытие окон. Взаимодействие с окнами (например, передача данных). Примеры валидации (проверка формата ввода). Сообщения об ошибках и уведомления пользователя. Использование регулярных выражений для валидации.

27. Многопоточность и асинхронная работа окон. Многопоточность в GUI-приложениях. Проблемы синхронизации потоков. Использование асинхронных вызовов для долго выполнимых операций.

28. Значение стиля в UX/UI дизайне. Основы теории цвета. Работа с цветом и шрифтами. Стилизация.

29. Работа с текстом, изображениями. Построение графиков и диаграмм. Библиотеки для построения графиков и диаграмм. Работа с мультимедиа

Примеры практических заданий

1. Разработка программных модулей для работы с массивами. Работа через систему контроля версий.

2. Разработка программных модулей для работы с коллекциями. Работа через систему контроля версий.

3. Разработка программных модулей для работы с датой и временем. Работа через систему контроля версий.

4. Разработка программных модулей с использованием паттернов проектирования. Работа через систему контроля версий.

5. Навигация по файловой системе. Чтение и запись файлов. Работа с потоками. Работа с изолированным хранилищем.

6. Работа с большими объемами текста. Кодирование и декодирование строк. Построение регулярных выражений. Чтение и запись файлов в разных кодировках.

7. Организация асинхронного вызова методов

8. Создание программного модуля, который будет выполнять методы в рамках параллельных задач

9. Оценка сложности алгоритмов

10. Применение рекурсивных алгоритмов

11. Работа с алгоритмами сортировки и поиска

12. Создание хеш-таблиц и их использование для ускорения поиска данных

13. Нахождение кратчайших путей в графах с использованием алгоритма Дейкстры

14. Решение задачи о рюкзаке с использованием метода динамического программирования

15. Реализация строковых алгоритмов

16. Реализация приоритетных очередей для планирования задач

17. Анализ требований к модулю и определение его функциональности

18. Создание спецификации программного модуля

19. Проектирование требований к внутренней структуре программного модуля средствами диаграмм классов. Применение паттернов проектирования

20. Проектирование требований к организации компонентов модуля средствами диаграммы компонентов

21. Проектировать интерфейсы программного обеспечения для взаимодействия с другими модулями и системами

22. Анализ и оптимизация проектируемого модуля для повышения его эффективности и качества

23. Проектирование главного окна приложения с несколькими панелями и элементами управления.

24. Разработка модулей многооконного приложения

25. Разработка стилей для приложения для улучшения взаимодействия с пользователем

26. Разработка модулей для представления текстовой информации

27. Разработка модулей для работы с изображениями

28. Разработка модулей для представления информации в виде графиков и диаграмм

29. Разработка модулей для работы аудио и видео

30. Реализация загрузки данных из интернета в фоновом режиме без блокировки основного потока приложения.

31. Разработка формы регистрации с элементами ввода и проверкой корректности введенных данных.

4.2. Типовые задания для оценки освоения МДК.02.02 Осуществление интеграции программных модулей

Вопросы для подготовки к экзамену/зачёту

1. Разработка REST API. Клиент-серверное взаимодействие. Особенности передачи информации по HTTP протоколу. Структура HTTP запроса. HTTP методы: GET, POST, DELETE, PUT, PATCH. HTTP заголовки. Тело запроса.

2. Маршрутизация запросов. Группировка маршрутов. Статические ресурсы.

3. Обработка запросов пользователя. Path, Query параметры. Обработка содержимого body: raw, objects, forms, multipart. Валидация данных.

4. Формирование и отправка ответов: object, file. Параметры ответов: статус код, тип содержимого, заголовки, cookies. Перенаправления. Сериализация/десериализация объектов.

5. Создание и управление фоновыми задачами.

6. Аутентификация и авторизация. oAuth, JWT, forms. Сессии. Ролевое разграничение доступа к ресурсам.

7. Разработка WebSocket API. Взаимодействие клиента и сервера по WebSocket протоколу. Настройки соединения. Открытие и закрытие соединения. Передача сообщения серверу.

8. Разработка микросервисов. Микросервисная и монолитная архитектура.

9. Синхронное (REST, gRPC) и асинхронное (брокеры сообщений) взаимодействие между микросервисами.

10. Настройка конфигурации и сборки приложения.

11. Логирование событий. Конфигурация логирования. Уровни логирования. Логирование в файлы различного формат.

12. Мониторинг приложения: нагрузка, ошибки, сбор статистики. Внедрение сборщика метрик.

13. Инструменты контейнеризации. Контейнеризация приложения. Средства доставки и средства развертывания решения.

14. Протоколы с использованием безопасного соединения: HTTPS, WSS (WebSocket Secure).

15. Предотвращение угроз безопасности: SQL инъекции, CSRF, XSS. Хеширование сенситивных данных, применение алгоритмов хеширования паролей с солью.

16. Анализ уязвимостей. Регулярные аудиты безопасности. Применение лучших практик защиты информации.

17. Масштабирование интегрированных решений. Горизонтальное и вертикальное масштабирование.

18. Оптимизации производительности. Кэширование данных. Оптимизация запросов к базам данных.

19. Профилирование кода. Уменьшение времени отклика.

Примеры практических заданий

1. Создание клиентского приложения для работы с публичным API
2. Создание REST API приложения с реализацией: добавления, удаления, изменения и создания данных (от 3 - 4 сущностей)
3. Расширение функционала REST API приложения: работа с удаленным источником данных
4. Расширение функционала REST API приложения: работа со статическими изображениями (ресурсами) - загрузка, передача, удаление.
5. Расширение функционала REST API приложения: обработка path и query параметров
6. Расширение функционала REST API приложения: обработка ошибок, передача сообщений об ошибке пользователю
7. Расширение функционала REST API приложения: валидация полученных данных
8. Расширение функционала REST API приложения: добавление фоновых задач
9. Расширение функционала REST API приложения: добавление аутентификации и авторизации, создание ролевой системы
10. Создание клиентского приложения для работы с публичным WebSocket.
11. Создание серверного приложения для работы по websocket протоколу.
12. Создание микросервисного приложения с взаимодействием по REST
13. Создание микросервисного приложения с взаимодействием по gRPC
14. Создание микросервисного приложения с взаимодействием через брокера приложений (consumer, producer)
15. Настроить конфигурацию rest api приложения (порт, хост, данные для подключения к источнику данных, приватные ключи).
16. Внедрить логирование в rest api приложение.
17. Упаковка rest api приложения в контейнер и доставка на другое устройство.
18. Добавление SSL сертификата в приложение
19. Настройка конфигурации безопасности приложения.
20. Реализация кэширования данных в rest api приложение
21. Оптимизация производительности rest api через профилирование

4.3. Типовые задания для оценки освоения МДК.02.03 Поддержка и тестирование программных модулей

Вопросы для подготовки к экзамену/зачёту

1. Определение качества программного модуля. Метрики качества программных модулей (статические метрики: количество строк кода, цикломатическая сложность, коэффициент связности и сцепленной: динамические метрики: покрытие кода тестами, частота отказов, время отклика). Принципы проектирования качественных модулей.
2. Стандарты и модели качества программных модулей. Применение моделей качества. Инструменты для оценки качества. Практические аспекты повышения качества.
3. Понятие отладки. Понятия ошибки, дефекта, сбоя, отказа. Типы ошибок. Инструменты для отладки. Процесс пошаговой отладки (установка точек останова, шаг за шагом выполнение кода, просмотр состояния переменных, выполнение отдельных частей кода).

Стратегии поиска ошибок (метод половинного деления, метод исключения, проверка граничных условий, поиск паттернов повторяющихся ошибок). Документирование процесса отладки.

4. Понятие исключения. Типы исключений. Механизм обработки исключений. Логика работы с исключениями. Методы отладки кода с использованием исключений и логирования.
5. Понятие процесса тестирования программного обеспечения. Этапы процесса тестирования программного обеспечения. техники ручного тестирования и автоматизированного тестирования
6. Модель работы с дефектами. Принципы работы в системе контроля дефектов.
7. Виды тестирования (функциональное тестирование, нефункциональное тестирование, статическое и динамическое тестирование).
8. Типы тестирования (модульное тестирование, интеграционное тестирование, системное тестирование, приемочное тестирование, нагрузочное тестирование, стресс-тестирование)
9. Тестирование по белому ящику. Метод покрытия операторов. Метод покрытия условий.
10. Тестирование по белому ящику. Метод комбинаторного покрытия условий.
11. Тестирование по черному ящику. Метод классов эквивалентности.
12. Тестирование по черному ящику. Метод граничных значений.
13. Модульные тесты. Тестирование интеграции. Методы и инструменты для тестирования интегрированных решений.

Примеры практических заданий

1. Анализ и оценка качества программного модуля с использованием метрик качества программных модулей
2. Использование статического анализа кода для выявления дефектов
3. Разработка и применение процессов обеспечения качества в жизненном цикле разработки программных модулей
4. Разработка стратегии отладки и исправление ошибок в программном обеспечении
5. Код-ревью и парное программирование
6. Основные конструкции для обработки исключительных ситуаций
7. Практическое использование исключений в реальной задаче
8. Обработка ошибок и исключение в RESTful API.
9. Анализ требований к программному обеспечению и составление планов тестирования. Использование систем контроля дефектов программного обеспечения
10. Тестирование методами белого ящика. Метод покрытия операторов. Метод покрытия условий.
11. Тестирование методами белого ящика. Метод комбинаторного покрытия условий.
12. Тестирование по черному ящику. Метод классов эквивалентности.
13. Тестирование по черному ящику. Метод граничных значений.
14. Тестирование по черному ящику. Анализ причинно-следственных связей.
15. Разработка модульных тестов.
16. Разработка модульных тестов с проверкой результатов тестирования с учетом погрешности.
17. Разработка модульных тестов для отдельно компилируемых модулей.
18. Разработка модульных тестов для проверки коллекций.
19. Тестирование интеграции. Написание и выполнение тестов для проверки взаимодействия между модулями
20. Тестирование RESTful API
21. Тестирование производительности
22. Разработка через тестирование.

4.4. Типовые задания для оценки освоения МДК.02.04 Математическое моделирование

Вопросы для подготовки к экзамену/зачёту

1. Понятие модели. Классификация моделей. Понятие математической модели. Типы математических моделей. Принципы построения математических моделей. Основные этапы математического моделирования.
2. Каноническая задача линейного программирования. Основные определения. Графический метод решения задач линейного программирования. Симплексный метод решения задач линейного программирования. Транспортная задача. Задача о назначениях. Целочисленное программирование.
3. Основные понятия и определения нелинейного программирования. Методы решения задач нелинейного программирования.
4. Основные понятия и определения динамического программирования. Задачи, решаемые методами динамического программирования.
5. Основные понятия и определения теории графов. Нахождение кратчайшего пути. Дерево решений. Сетевые графики. Расчет временных параметров.
6. Марковский случайный процесс. Системы массового обслуживания: основные понятия, классификация. Схема гибели и размножения
7. Предмет и задачи теории игр. Основные понятия теории игр. Матричные игры. Биматричные игры. Игры в развернутой форме
8. Основные понятия имитационного моделирования. Примеры имитационных моделей. Методы имитационного моделирования. Инструментальные средства имитационного моделирования

Примеры практических заданий

1. Построение простейших математических моделей
2. Решение задач линейного программирования симплексным методом
3. Решение транспортной задачи
4. Решение задачи о назначениях
5. Применение инструментальных средств для решения задач линейного программирования
6. Решение задач нелинейного программирования
7. Решение задач оптимального распределения ресурсов, о замене оборудования
8. Решение задач определения оптимального пути, оптимального резервирования
9. Решение задач на применение методов сетевого планирования
10. Расчет характеристик простейших систем массового обслуживания
11. Решение игровых задач с нулевой суммой.
12. Решение задач в развернутой форме
13. Разработка простейшей имитационной модели
14. Решение задач массового обслуживания методами имитационного моделирования

4.5. Типовые задания для оценки освоения МДК.02.05 Численные методы

Вопросы для подготовки к экзамену/зачёту

1. Способы хранения чисел в памяти компьютера. Абсолютная погрешность, относительная погрешность.
2. Верные, сомнительные, значащие цифры.
3. Погрешности арифметических действий. Оценка погрешностей значений функции. Отделение корней. Метод половинного деления. Метод простой итерации.
4. Методы Ньютона: метод хорд, касательных.
5. Методы Ньютона: комбинированный метод хорд и касательных. Сравнение методов вычислений по скорости сходимости итерационного процесса. Решение систем линейных алгебраических уравнений методом Гаусса. Применение метода Гаусса для вычисления определителей и нахождения обратной матрицы.

6. Метод простой итераций. Метод Зейделя. Сравнение методов вычислений по скорости сходимости итерационного процесса. Понятие интерполяции. Интерполяционный многочлен Лагранжа. Интерполяционные формулы Ньютона.

7. Интерполяция сплайнами.

8. Экстраполяция функций. Квадратурные формулы Ньютона-Котеса.

9. Квадратурная формула Гаусса. Сравнение методов численного интегрирования_Метод Эйлера. Уточненная схема Эйлера.

10. Метод Рунге – Кутта. Сравнение методов. Методы минимизации функции одной переменной: метод дихотомии, метод золотого сечения. Методы минимизации функции двух переменных: покоординатный спуск, наискорейший спуск

Примеры практических заданий

1. Вычисление погрешностей приближенных значений. Вычисление погрешностей результатов арифметических действий.

2. Решение алгебраических и трансцендентных уравнений приближенными методами (метод половинного деления, метод простых итераций)

3. Решение алгебраических и трансцендентных уравнений приближенными методами (методы Ньютона)

4. Мониторинг и анализ производительности разработанных приложений для численного решения уравнений.

5. Решение систем линейных алгебраических уравнений методом Гаусса. Вычисление определителя. Нахождение обратной матрицы

6. Решение систем линейных алгебраических уравнений методом простой итерации, методом Зейделя

7. Мониторинг и анализ производительности разработанных приложений для численного решения систем линейных алгебраических уравнений.

8. Составление интерполяционных формул Лагранжа и Ньютона. Интерполяция сплайнами.

9. Экстраполирование функций

10. Вычисление интегралов при помощи формул Ньютона – Котеса

11. Вычисление интегралов при помощи формул Гаусса.

12. Нахождение решений обыкновенных дифференциальных уравнений при помощи формул Эйлера.

13. Нахождение решений обыкновенных дифференциальных уравнений методом Рунге – Кутта.

14. Нахождение экстремумов функций одной переменной приближенными методами

15. Нахождение экстремумов функций двух переменных приближенными методами

4.6. Типовые задания для оценки освоения МДК.02.06 Безопасность программного обеспечения

Вопросы для подготовки к экзамену/зачёту

1. Введение в кибербезопасность и уязвимости ПО.
2. Модели угроз и анализ рисков.
3. Уязвимости веб-приложений: OWASP Top 10.
4. Безопасная аутентификация и авторизация.
5. Криптография для разработчиков.
6. Принципы безопасного проектирования архитектуры.
7. Криптографические протоколы и их реализация.
8. Криптография в мобильных приложениях.
9. Криптография в веб-приложениях.
10. Криптография в облачных средах.

Примеры практических заданий

1. Анализ кода на наличие уязвимостей - ручной review 1000 строк кода
2. SQL инъекции - эксплуатация и защита уязвимого приложения
3. XSS атаки - создание и предотвращение межсайтового скриптинга
4. CSRF защита - реализация токенов и проверки Origin/Referer
5. Составление модели угроз для типового веб-приложения
6. Настройка безопасной аутентификации с JWT и refresh токенами
7. Реализация RBAC системы с разделением привилегий
8. Шифрование данных с использованием AES и RSA
9. Хэширование паролей с salt и adaptive functions (bcrypt, Argon2)
10. Анализ сетевого трафика с помощью Wireshark
11. Сканирование уязвимостей OWASP ZAP и Burp Suite
12. Настройка HTTPS и создание самоподписанных сертификатов
13. Защита от brute-force атак с ограничением попыток входа. Безопасная работа с файлами
14. Реализация безопасной десериализации данных
15. Аудит логов безопасности и выявление подозрительной активности
16. Настройка CORS политик для веб-приложений
17. Защита от DDOS атак с помощью rate limiting
18. Безопасная работа с памятью в приложениях. Создание безопасного API с валидацией всех входных данных
19. Реализация end-to-end шифрования для мессенджера на Signal Protocol
20. Настройка TLS 1.3 с perfect forward secrecy и современными cipher suites
21. Создание secure OAuth 2.0 провайдера с PKCE и защитой от атак
22. Имплементация JWE (JSON Web Encryption) для защищённых токенов
23. Разработка безопасного voting system с homomorphic encryption
24. Создание cryptocurrency wallet с ECDSA и hierarchical deterministic keys. Реализация secure password manager с client-side encryption
25. Настройка HSM эмулятора для аппаратной защиты ключей
26. Разработка secure file storage с encryption at rest и in transit
27. Имплементация zero-knowledge proof для аутентификации без пароля
28. Создание blockchain smart contract с защитой от reentrancy attacks. Реализация secure multi-party computation для совместных вычислений
29. Настройка quantum-resistant cryptography с lattice-based алгоритмами.
30. Разработка secure API gateway с JWT verification и rate limiting
31. Создание hardware-backed key storage для мобильного приложения
32. Имплементация digital signature system с timestamping
33. Настройка certificate transparency logs для мониторинга SSL сертификатов
34. Разработка secure session management с защитой от hijacking. Создание cryptographically secure RNG (random number generator)

Вопросы для устного опроса по разделам

по МДК.02.01 Разработка программных модулей

1. Объясните суть модульной архитектуры приложений. Какие принципы лежат в её основе и какие преимущества она даёт по сравнению с монолитной архитектурой? Приведите пример реального приложения с модульной структурой.
2. Сравните архитектурные шаблоны MVC, MVVM и MVP: в чём их ключевые различия, какие компоненты входят в каждый шаблон и в каких сценариях целесообразно применять каждый из них?

3. Опишите роль систем контроля версий (Git, SVN) в разработке модульных приложений. Какие практики ветвления и слияния кода вы знаете и как они влияют на командную разработку?

4. Что такое библиотека в контексте разработки ПО? Опишите процесс подключения стандартной библиотеки и создания собственной. Какие преимущества даёт повторное использование кода через библиотеки?

5. Объясните разницу между наследованием, композицией и агрегацией. В каких случаях целесообразно использовать каждый из этих типов отношений между классами? Приведите примеры.

6. Опишите основные порождающие паттерны проектирования (Factory, Singleton, Builder). В каких практических задачах они применяются и какие проблемы решают?

7. Что такое инкапсуляция и как она реализуется на уровне классов? Почему инкапсуляция важна для поддержания целостности данных и упрощения тестирования модулей?

8. Как диаграммы классов UML помогают в проектировании программного модуля? Какие элементы диаграммы вы используете для отображения связей между классами и их ответственности?

9. Что такое асимптотическая сложность алгоритма? Объясните смысл нотации O-большое и приведите примеры алгоритмов с разной временной сложностью (O(1), O(n), O(n^2), O(log n)).

10. Сравните массив и связный список: как они представлены в памяти, какие операции выполняются быстрее в каждой структуре и в каких задачах целесообразно использовать каждую из них?

11. Опишите принцип работы хеш-таблицы. Что такое коллизии, какие методы их разрешения вы знаете (цепочки, открытая адресация) и как выбрать хорошую хеш-функцию?

12. Объясните разницу между поиском в глубину (DFS) и поиском в ширину (BFS) в графах. В каких задачах применяется каждый алгоритм? Приведите пример использования алгоритма Дейкстры.

13. В чём суть динамического программирования? Опишите основные шаги применения ДП на примере задачи о рюкзаке или вычисления чисел Фибоначчи.

14. Сравните алгоритмы сортировки: быстрая сортировка, сортировка слиянием и пузырьковая сортировка. Какова их временная сложность в лучшем, среднем и худшем случаях?

15. Опишите алгоритм Рабина-Карпа для поиска подстроки. Как используется хеширование в этом алгоритме и в каких случаях он эффективнее наивного поиска?

16. Что такое куча (heap) и как она реализуется? Объясните применение очереди с приоритетом на примере планировщика задач или алгоритма Дейкстры.

17. Опишите основные операции работы с файловой системой: чтение, запись, сжатие потоков. Какие классы и методы вы используете для безопасной работы с файлами?

18. Для чего применяются регулярные выражения? Приведите примеры шаблонов для валидации email, телефона и даты. Какие преимущества и ограничения есть у этого подхода?

19. Объясните разницу между многопоточностью и асинхронным программированием. В каких сценариях целесообразно использовать пул потоков, а в каких — async/await?

20. Что такое лямбда-выражения и как они используются при создании задач в параллельном программировании? Приведите пример передачи лямбды в метод для выполнения в отдельном потоке.

21. Как обеспечить синхронизацию потоков при доступе к общим ресурсам? Опишите механизмы блокировок (lock, mutex, semaphore) и потенциальные проблемы (deadlock, race condition).

22. Как реализовать отмену длительной задачи в асинхронном коде? Опишите механизм CancellationToken (или аналога) и обработку исключения при отмене.

23. Какие стратегии кодирования/декодирования текста вы знаете (UTF-8, ASCII, Base64)? В каких случаях необходимо явно указывать кодировку при работе со строками и файлами?

24. Сравните командный интерфейс (CLI), графический (GUI) и речевой (VUI): в каких сценариях каждый из них предпочтителен и какие требования предъявляются к их разработке?

25. Опишите жизненный цикл окна в графическом приложении: создание, отображение, обработка событий, закрытие. Как передавать данные между окнами и обеспечивать валидацию ввода?

26. Какие проблемы возникают при выполнении долгих операций в главном потоке GUI-приложения? Как использовать асинхронные вызовы или фоновые потоки для сохранения отзывчивости интерфейса?

27. Объясните принцип работы обработчиков событий в графическом интерфейсе. Как связать элемент управления (кнопка, поле ввода) с методом-обработчиком и передать контекст выполнения?

28. Какие принципы теории цвета и типографики важны при проектировании пользовательского интерфейса? Как обеспечить доступность (accessibility) и читаемость интерфейса для разных групп пользователей?

29. Какие библиотеки вы знаете для построения графиков и диаграмм в приложениях? Опишите процесс визуализации данных: подготовка данных, выбор типа диаграммы, настройка отображения.

30. Как обеспечить безопасность, производительность и масштабируемость на уровне модуля? Опишите практики: валидация входных данных, логирование, кэширование, разделение ответственности и тестирование модулей.

<u>по МДК.02.02 Осуществление интеграции программных модулей</u>

1. Опишите структуру HTTP-запроса. В чём принципиальная разница между методами GET, POST, PUT, PATCH и DELETE? Приведите сценарии корректного применения каждого метода в REST API.

2. Какую роль играют HTTP-заголовки в клиент-серверном взаимодействии? Приведите примеры заголовков, отвечающих за аутентификацию, кэширование, согласование контента и управление сессиями.

3. Что такое маршрутизация в веб-фреймворках? Объясните принципы группировки маршрутов и механизм отдачи статических ресурсов (CSS, JS, изображения) отдельно от динамических эндпоинтов.

4. Чем отличаются Path-параметры от Query-параметров? Как сервер обрабатывает различные форматы тела запроса: raw-данные, JSON-объекты, URL-формы и multipart-запросы?

5. Зачем необходима серверная валидация входящих данных? Опишите типичные подходы к валидации (схемы, аннотации, middleware) и последствия её отсутствия для безопасности и целостности данных.

6. Как формируется HTTP-ответ сервера? Объясните смысл статус-кодов (2xx, 3xx, 4xx, 5xx), заголовка Content-Type, механизма cookies и процессов сериализации/десериализации объектов.

7. В каких случаях сервер должен возвращать файл, а не JSON? Как корректно настроить заголовки для скачивания, потоковой отдачи и безопасного перенаправления клиента?

8. Какие задачи целесообразно выносить в фоновое выполнение? Опишите жизненный цикл фоновой задачи, способы отслеживания её статуса и обработки ошибок без блокировки основного потока.

9. В чём разница между аутентификацией и авторизацией? Сравните подходы на основе сессий, JWT и OAuth 2.0: архитектуру, безопасность, statefulness и типичные сценарии применения.

10. Как реализовать ролевое разграничение доступа (RBAC) в REST API? Опишите процесс проверки прав пользователя на уровне маршрутизатора или middleware и обработку случаев отсутствия прав.

11. Чем протокол WebSocket отличается от классического HTTP? Опишите процесс установления соединения (handshake), двусторонний обмен сообщениями и корректное закрытие канала связи.

12. В каких сценариях WebSocket предпочтительнее REST? Как обрабатывать разрывы соединения, гарантировать доставку сообщений и масштабировать WebSocket-серверы при росте нагрузки?

13. Сравните монолитную и микросервисную архитектуры. При каких условиях переход на микросервисы оправдан, а когда он создаёт избыточную сложность и операционные издержки?

14. Как микросервисы взаимодействуют между собой синхронно и асинхронно? Сравните REST/gRPC с брокерами сообщений (RabbitMQ, Kafka) по надёжности, идемпотентности и отказоустойчивости.

15. Зачем выносить конфигурацию приложения из кода? Опишите принципы управления переменными окружения, секретами, профилями сборки (dev/stage/prod) и безопасного хранения чувствительных параметров.

16. Какие уровни логирования вы знаете (DEBUG, INFO, WARN, ERROR и др.)? Как правильно настраивать логирование для production-среды, выбирать форматы вывода и избегать избыточной записи логов?

17. Что включает в себя мониторинг современного веб-приложения? Опишите ключевые метрики (нагрузка, latency, error rate, saturation) и роль систем сбора/визуализации метрик (например, Prometheus + Grafana).

18. Что такое контейнеризация и как она упрощает доставку и развёртывание? Опишите базовую структуру Dockerfile, принцип изоляции процессов, работу с образами и реестрами контейнеров.

19. Почему важно использовать HTTPS и WSS? Опишите процесс TLS-рукопожатия, роль сертификатов и то, как обеспечивается конфиденциальность и целостность данных при передаче.

20. Что такое SQL-инъекции, XSS и CSRF? Какими методами (параметризованные запросы, экранирование, SameSite cookies, CSRF-токены) разработчик защищает приложение от каждой из этих угроз?

21. Почему пароли нельзя хранить в открытом виде или простом хеше? Объясните принцип хеширования с солью (salt), работу алгоритмов bcrypt/Argon2 и необходимость настройки стоимости вычисления.

22. Как проводятся регулярные аудиты безопасности? Опишите этапы статического и динамического анализа кода (SAST/DAST), применение OWASP Top 10 и внедрение практик Security-by-Design.

23. В чём разница между горизонтальным и вертикальным масштабированием? При каких условиях каждый подход эффективен, какие ограничения имеет и как влияет на архитектуру приложения?

24. Как балансировка нагрузки влияет на отказоустойчивость системы? Опишите стратегии маршрутизации запросов (round-robin, least connections, IP hash) и обработку состояния сессий при горизонтальном масштабировании.

25. Какие стратегии кэширования данных вы знаете (на уровне приложения, БД, reverse-proxy, CDN)? Как управлять инвалидацией кэша и избегать проблем с устаревшими или несогласованными данными?

26. Как оптимизировать запросы к реляционной базе данных? Объясните роль индексов, анализ планов выполнения запросов, влияние нормализации/денормализации и работу с соединениями (joins).

27. Что такое профилирование кода и зачем оно нужно в production? Какие инструменты и метрики (CPU, memory, I/O, GC) помогают выявить узкие места и исключить неочевидные утечки ресурсов?

28. Как уменьшить время отклика сервера на стороне бэкенда? Опишите подходы к оптимизации алгоритмов, использованию пулов соединений, асинхронной обработке и сокращению размера передаваемых данных.

29. Как обеспечить безопасную и надёжную передачу пользовательских данных в REST API? Опишите полный цикл: от валидации запроса и аутентификации до защиты от атак, логирования и формирования корректного HTTP-ответа.

30. Как спроектировать отказоустойчивую и масштабируемую систему из микросервисов? Опишите взаимодействие сервисов, обработку сбоев (retry, circuit breaker, dead-letter queues), мониторинг и стратегию бесшовного развёртывания.

по МДК.02.03 Поддержка и тестирование программных модулей

1. Что понимается под качеством программного модуля? Различите статические и динамические метрики качества, приведите по два примера каждой и объясните, как они влияют на принятие решений в процессе разработки.

2. Объясните понятия цикломатической сложности и коэффициентов связности/сцепления. Почему высокие значения этих метрик часто указывают на архитектурные проблемы и как их снижение улучшает сопровождаемость кода?

3. Какие стандарты и модели качества ПО (например, ISO/IEC 25010, CMMI, SPICE) вы знаете? Как их практическое применение помогает формализовать требования к качеству и оценить зрелость процессов?

4. Какие инструменты автоматической оценки качества кода (линтеры, статические анализаторы, метрические дашборды) вы применяли? Как их интеграция в CI/CD-конвейер предотвращает деградацию кодовой базы?

5. Какие принципы проектирования качественных модулей (SOLID, DRY, KISS, YAGNI, принцип единой ответственности) вы считаете наиболее критичными? Как они соотносятся с динамическими метриками, такими как время отклика и частота отказов?

6. Дайте строгие определения понятиям «ошибка», «дефект», «сбой» и «отказ». Как они связаны причинно-следственными связями и на каком этапе жизненного цикла ПО каждый из них обычно выявляется?

7. Опишите пошаговый процесс отладки в современной IDE. Какие функции отладчика (точки останова, пошаговое выполнение, просмотр стека вызовов, изменение переменных на лету) вы используете и для решения каких задач?

8. Сравните стратегии поиска ошибок: метод половинного деления, метод исключения, проверка граничных условий и поиск паттернов. В каких сценариях каждая из них наиболее эффективна? Приведите пример применения метода половинного деления.

9. Зачем документировать процесс отладки? Какая минимальная информация должна фиксироваться в отчёте об отладке для обеспечения воспроизводимости и передачи знаний внутри команды?

10. Что такое исключение в программировании? Различите проверяемые и непроверяемые исключения. Как правильно выстраивать архитектуру обработки исключений, чтобы не нарушать принцип fail-fast и избегать «проглатывания» ошибок?

11. Как сочетание механизма исключений и логирования ускоряет отладку в production-среде? Какие правила вы применяете к логированию, чтобы сохранить информативность, но не снизить производительность и не засекретить чувствительные данные?

12. Опишите основные этапы процесса тестирования ПО. Чем ручное тестирование отличается от автоматизированного и при каких условиях автоматизация становится экономически нецелесообразной?

13. Что такое жизненный цикл дефекта? Перечислите типичные статусы бага в системе трекинга (например, New, Open, In Progress, Fixed, Verified, Closed, Reopened) и объясните ответственность ролей на каждом этапе.

14. Какие принципы эффективной работы в системе контроля дефектов вы знаете? Как корректно оформить баг-репорт, чтобы минимизировать время на воспроизведение и ускорить исправление?

15. Как метрики дефектов (плотность дефектов, среднее время обнаружения, коэффициент повторного открытия) используются для оценки готовности релиза и улучшения инженерных практик команды?

16. В чём принципиальная разница между функциональным и нефункциональным тестированием? Приведите примеры нефункциональных требований (производительность, безопасность, usability) и опишите, как их верифицируют на практике.

17. Сравните статическое и динамическое тестирование. Какие техники статического анализа (код-ревью, парное программирование, статические анализаторы, инспекции) позволяют находить дефекты до компиляции или запуска программы?

18. Опишите концепцию «пирамиды тестирования». Как соотносятся модульное, интеграционное, системное и приёмочное тестирование по количеству тестов, стоимости поддержки и скорости выполнения?

19. В каких бизнес-сценариях применяются нагрузочное и стресс-тестирование? Какие показатели (RPS, latency, throughput, error rate, utilization CPU/RAM) вы мониторите и как интерпретируете «точку деградации» системы?

20. Что такое тестирование по «белому ящику»? В чём его ключевые преимущества перед «чёрным ящиком» и какие ограничения оно накладывает на тестировщика?

21. Объясните метод покрытия операторов (statement coverage). Как рассчитать эту метрику и почему достижение 100% покрытия операторов не гарантирует отсутствие логических дефектов?

22. Что такое покрытие условий (condition coverage)? Приведите пример составного логического выражения и покажите, какие тестовые наборы необходимы для достижения полного покрытия условий.

23. В чём суть метода комбинаторного покрытия условий? В каких отраслях (авионика, медицина, финансы) он является нормативным требованием и как балансируют между полнотой покрытия и стоимостью тестирования?

24. Объясните метод классов эквивалентности. Как он позволяет сократить количество тестовых случаев без потери эффективности проверки функционала? Приведите пример разбиения входного диапазона на валидные и невалидные классы.

25. Опишите метод граничных значений. Почему статистически большинство дефектов проявляется именно на границах классов эквивалентности и как правильно выбирать значения для тестирования (минимум, максимум, чуть выше/ниже)?

26. Как комбинировать методы классов эквивалентности и граничных значений для тестирования сложной формы ввода с несколькими полями? Какие дополнительные техники чёрного ящика (таблицы решений, диаграммы состояний/переходов, попарное тестирование) вы знаете?

27. Что такое модульное тестирование? Опишите принципы FIRST для юнит-тестов и объясните, зачем и когда применяются моки (mocks), стабы (stubs) и фейки (fakes) для изоляции зависимостей.

28. В чём отличие интеграционного тестирования от модульного? Какие подходы (снизу вверх, сверху вниз, big bang, сэндвич) вы знаете и какие проблемы межкомпонентного взаимодействия оно помогает выявить?

29. Какие фреймворки и инструменты для автоматизации юнит- и интеграционного тестирования вы использовали? Как настроить запуск тестов в CI-конвейере, анализировать отчёты о покрытии и блокировать мерж при падении критических тестов?

30. Как обеспечить качество интегрированных решений в распределённых системах? Опишите роль контрактного тестирования (Consumer-Driven Contracts), тестовых контейнеров (Testcontainers) и симуляторов внешних API при проверке интеграций.

по МДК.02.04 Математическое моделирование

1. Что такое модель и по каким критериям оценивается её адекватность? Приведите классификацию моделей по форме представления, степени детализации и цели использования.
2. В чём принципиальное отличие математической модели от физической, концептуальной или компьютерной? Перечислите основные типы математических моделей (детерминированные/стохастические, непрерывные/дискретные, статические/динамические) и приведите примеры.
3. Опишите основные этапы математического моделирования. На каком этапе наиболее вероятно возникновение ошибки адекватности и какие методы используются для её выявления и минимизации?
4. Какие фундаментальные принципы лежат в основе построения математических моделей (принцип подобия, инвариантности, сохранения)? Как они влияют на выбор управляемых переменных, параметров и ограничений?
5. Сформулируйте каноническую задачу линейного программирования. Каковы математические условия существования, единственности и неограниченности оптимального решения?
6. Опишите графический метод решения задач ЛП для двух переменных. Как визуально определяются допустимая область, линии уровня целевой функции и точка оптимума? В чём его практические ограничения?
7. В чём заключается геометрическая и алгебраическая идея симплекс-метода? Объясните переход от одного базисного допустимого решения к другому, критерий оптимальности и признаки зацикливания.
8. Сравните транспортную задачу и задачу о назначениях. Как специфическая структура матрицы ограничений позволяет применять специализированные алгоритмы (метод потенциалов, венгерский алгоритм) вместо общего симплекс-метода?
9. В каких прикладных задачах необходимо применять целочисленное программирование? Чем метод ветвей и границ отличается от симплекс-метода и почему задачи ЦП в общем случае считаются NP-трудными?
10. Чем задачи нелинейного программирования принципиально отличаются от задач ЛП? Какие математические свойства (выпуклость, дифференцируемость, условия Куна-Таккера) гарантируют нахождение глобального оптимума?
11. Опишите метод множителей Лагранжа для решения задач условной оптимизации с ограничениями-равенствами. В чём его геометрический смысл и как он обобщается на неравенства (KKT-условия)?
12. Сравните градиентные методы первого порядка (наискорейшего спуска, сопряжённых градиентов) и методы второго порядка (Ньютона, квазиньютоновские). Когда целесообразно применять каждый из них с учётом вычислительной сложности и сходимости?
13. Сформулируйте принцип оптимальности Беллмана. Какие два ключевых свойства задачи (оптимальная подструктура и перекрывающиеся подзадачи) делают динамическое программирование применимым?
14. Сравните подходы «сверху вниз» (мемоизация с рекурсией) и «снизу вверх» (табуляция, итеративный) в ДП. В каких случаях предпочтителен каждый из них с точки зрения потребления памяти, скорости и удобства отладки?
15. Приведите пример классической задачи, решаемой методом ДП (задача о рюкзаке, кратчайший путь в многоступенчатом графе, управление запасами). Постройте рекуррентное соотношение, определите базовые случаи и опишите порядок вычислений.
16. Что такое граф и какие основные способы его представления в памяти компьютера вы знаете? Сравните матрицу смежности и списки смежности по требованиям к памяти, скорости обхода и модификации.
17. Сравните алгоритмы Дейкстры и Беллмана-Форда для поиска кратчайшего пути. В каких случаях каждый из них применим, как обрабатываются отрицательные веса рёбер и как детектируется отрицательный цикл?

18. Что такое дерево решений и в каких задачах принятия решений оно применяется? Как оценивается ожидаемая ценность узлов, как учитываются вероятности исходов и как выбирается оптимальная стратегия?

19. Опишите метод критического пути (CPM) в сетевом планировании проектов. Как рассчитываются ранние/поздние сроки выполнения работ, полные и свободные резервы времени, и как определяется критический путь?

20. Что такое марковский случайный процесс и в чём заключается его основное свойство (отсутствие последействия)? В каких прикладных областях (финансы, телеком, надёжность) применяются цепи Маркова?

21. Опишите основные элементы и классификацию систем массового обслуживания. Что характеризуют входящий поток заявок, поток обслуживания и дисциплина очереди? Как система обозначается в нотации Кендалла (A/B/m/K/N/D)?

22. Что такое схема гибели и размножения? Как она связана с системами дифференциальных уравнений Колмогорова, как находятся стационарные вероятности состояний и при каких условиях существует стационарный режим СМО?

23. В чём заключается предмет теории игр? Дайте строгие определения ключевых понятий: игрок, стратегия (чистая/смешанная), функция выигрыша, равновесие Нэша, антагонистические и кооперативные игры.

24. Как решаются матричные игры с нулевой суммой? Объясните концепцию максимина/минимакса, седловой точки и необходимость перехода к смешанным стратегиям. Как оптимальные смешанные стратегии находятся через задачу ЛП?

25. Чем биматричные игры и игры в развёрнутой форме отличаются от матричных? Как информационные множества и деревья решений помогают анализировать последовательные ходы, неполную информацию и динамические стратегии?

26. В чём суть имитационного моделирования и когда оно предпочтительнее аналитических методов оптимизации? Приведите примеры сложных систем, где аналитическое решение невозможно или требует неприемлемых упрощений.

27. Сравните три основных парадигмы имитационного моделирования: дискретно-событийное, агентное и системной динамики. Какие типы систем (логистика, социальные сети, макроэкономика) лучше всего описываются каждым подходом?

28. Какие этапы включает жизненный цикл имитационной модели (верификация, валидация, калибровка, анализ чувствительности, экспериментирование)? Как доказать, что модель адекватно отражает поведение реальной системы, а не содержит программные ошибки?

29. Как обоснованно выбрать метод моделирования или оптимизации для конкретной прикладной задачи? Приведите критерии выбора между ЛП, НЛП, ДП, теорией игр, сетевыми методами и имитационным моделированием в зависимости от структуры задачи, данных и требований к точности.

30. Как современные программные инструменты (Python+PuLP/SciPy, Gurobi, CPLEX, AnyLogic, MATLAB) интегрируют рассмотренные методы? Опишите типичный рабочий процесс исследователя: от формализации задачи и подготовки данных до вычислений, валидации результатов и принятия управленческих решений.

по МДК.02.05 Численные методы

1. Как представляются вещественные числа в памяти компьютера (формат IEEE 754)? Что такое машинный эпсилон, машинный ноль и как они ограничивают точность вычислений?

2. Дайте строгие определения абсолютной и относительной погрешности. В каких практических ситуациях относительная погрешность является более информативной характеристикой, чем абсолютная?

3. Что такое верные, сомнительные и значащие цифры? Как по известной абсолютной погрешности результата определить количество верных знаков в его десятичной записи?

4. Как погрешности исходных данных распространяются при выполнении базовых арифметических операций? Объясните, почему вычитание двух близких по величине чисел считается численно неустойчивой операцией и как её избегать.

5. Как оценивается погрешность значения функции нескольких переменных, если известны погрешности её аргументов? Приведите формулу через частные производные и объясните её геометрический смысл.

6. Что значит «отделить корень» нелинейного уравнения? Какие достаточные условия теоремы о существовании и единственности корня на отрезке вы знаете?

7. Опишите алгоритм метода половинного деления (бисекции). Какова его скорость сходимости, априорная оценка погрешности и в каких случаях его применение наиболее оправдано?

8. В чём заключается суть метода простой итерации для решения уравнения $f(x)=0$? Как преобразовать исходное уравнение к виду $x = \varphi(x)$ и какое условие на $|\varphi'(x)|$ гарантирует сходимость итерационного процесса?

9. Сравните геометрическую интерпретацию метода хорд и метода касательных (Ньютона). В каких случаях метод Ньютона может расходиться, давать комплексные значения или попадать в точку перегиба?

10. Объясните идею комбинированного метода хорд и касательных. Как одновременное использование двух приближений (с разных сторон от корня) повышает надёжность и скорость сходимости по сравнению с отдельными методами?

11. Опишите алгоритм метода Гаусса для решения систем линейных алгебраических уравнений. В чём разница между прямым и обратным ходом, и как применяется выбор главного элемента для повышения устойчивости?

12. Как с помощью модификации метода Гаусса вычислить определитель матрицы и найти обратную матрицу? Какова вычислительная сложность этих операций и в каких случаях их применение нецелесообразно?

13. В чём принципиальное различие между прямыми (Гаусс) и итерационными методами решения СЛАУ? При каких размерах систем и структуре матриц итерационные методы становятся предпочтительнее?

14. Опишите метод простой итерации (Якоби) для СЛАУ. Какое достаточное условие сходимости (например, строгое диагональное преобладание) необходимо выполнять для гарантированного получения решения?

15. Чем метод Зейделя отличается от метода Якоби? Как использование уже вычисленных на текущей итерации компонент вектора влияет на скорость сходимости, расход памяти и параллелизуемость?

16. Как оценивается скорость сходимости итерационных процессов? Что такое спектральный радиус матрицы перехода и как он связан с порядком линейной сходимости методов?

17. В чём заключается классическая задача интерполяции? Сформулируйте теорему о существовании и единственности интерполяционного многочлена степени n для $n+1$ узла.

18. Объясните структуру интерполяционного многочлена Лагранжа. В каких случаях его вычислительная реализация становится неудобной (например, при добавлении новых узлов) и почему?

19. Как строятся интерполяционные формулы Ньютона (для равноотстоящих и неравноотстоящих узлов)? Какое архитектурное преимущество они дают при последовательном наращивании степени многочлена?

20. Что такое интерполяционный сплайн и зачем он заменяет высокостепенные многочлены? Объясните идею кубического сплайна, условия непрерывности производных и как он подавляет осцилляции (феномен Рунге).

21. Чем экстраполяция принципиально отличается от интерполяции? Какие риски (неустойчивость, лавинообразное накопление погрешности) сопровождают экстраполяцию за пределы узлов и как их можно минимизировать?

22. В чём заключается идея квадратурных формул Ньютона-Котеса? Приведите формулы средних прямоугольников, трапеций и Симпсона, объяснив их геометрический смысл и порядок точности.

23. Опишите принцип построения квадратурной формулы Гаусса. Почему оптимальный выбор узлов (корни ортогональных многочленов) позволяет достичь точности степени 2n+1 при n узлах?

24. Сравните методы численного интегрирования по точности, вычислительной стоимости и устойчивости к шумам в подынтегральной функции. Как выбрать метод в зависимости от гладкости функции и требуемой точности?

25. Как оценивается погрешность численного интегрирования на практике? Объясните правило Рунге для автоматического выбора шага и адаптивного измельчения сетки.

26. Опишите метод Эйлера для решения задачи Коши. Каков его порядок точности, почему он считается методом первого порядка и в каких инженерных задачах его применение допустимо?

27. В чём заключается «уточнённая схема Эйлера» (метод Эйлера-Коши / модифицированный Эйлер)? Как использование прогноза и коррекции на промежуточной точке повышает порядок точности до второго?

28. Объясните структуру классического метода Рунге–Кутта 4-го порядка. Почему он является отраслевым стандартом для нежёстких ОДУ, какова его вычислительная стоимость на шаг и как контролируется локальная погрешность?

29. Сравните методы минимизации функции одной переменной (дихотомия, золотое сечение) и методы для двух переменных (покоординатный спуск, наискорейший спуск/градиентный). Какие факторы (наличие производных, выпуклость, рельеф поверхности) определяют выбор алгоритма?

30. Как обоснованно выбрать численный метод для прикладной задачи? Опишите алгоритм принятия решения: от анализа условий (гладкость, размерность, жёсткость, требования к точности/скорости) до верификации результатов и оценки вычислительной устойчивости.

<u>по МДК.02.06 Безопасность программного обеспечения</u>

1. Что такое кибербезопасность в контексте разработки ПО? Объясните триаду безопасности CIA (Confidentiality, Integrity, Availability) и приведите примеры нарушений каждого компонента в реальных приложениях.

2. Какие основные классы уязвимостей ПО вы знаете (переполнение буфера, инъекции, неправильная обработка авторизации и др.)? Как уязвимости возникают на разных этапах жизненного цикла разработки (SDLC)?

3. Что такое модель угроз (threat model) и зачем она нужна? Опишите этапы построения модели угроз: идентификация активов, определение границ системы, выявление угроз, оценка рисков, выбор контрмер.

4. Как проводится качественный и количественный анализ рисков? Объясните формулу Risk = Likelihood × Impact, методы оценки (FAIR, DREAD, STRIDE) и как результаты влияют на приоритизацию исправлений.

5. В чём разница между уязвимостью (vulnerability), угрозой (threat) и инцидентом (incident)? Приведите пример цепочки: уязвимость → эксплуатация → инцидент → последствия и опишите роль разработчика на каждом этапе.

6. Что представляет собой проект OWASP и как используется рейтинг OWASP Top 10? Назовите текущие критические категории уязвимостей и объясните, почему они остаются актуальными несмотря на известность.

7. Опишите механизм и последствия инъекционных атак (SQLi, NoSQLi, Command Injection). Какие практики защищённого кодирования (параметризованные запросы, ORM, валидация, принцип наименьших привилегий) предотвращают эти уязвимости?

8. Что такое межсайтовый скриптинг (XSS) и какие его типы вы знаете (отражённый, хранимый, DOM-based)? Как контекстно-зависимое экранирование, Content Security Policy (CSP) и

безопасные фреймворки снижают риск XSS?

9. В чём суть уязвимостей нарушенного контроля доступа (Broken Access Control)? Как реализовать надёжную проверку прав на уровне бизнес-логики, маршрутизации и доступа к данным, избегая распространённых ошибок?

10. Какие риски несёт использование компонентов с известными уязвимостями (A06:2021)? Как настроить автоматический мониторинг зависимостей (SCA-инструменты), управление версиями и экстренное обновление в продакшене?

11. В чём фундаментальная разница между аутентификацией, авторизацией и учётом (accounting)? Как эти три компонента взаимодействуют в системе контроля доступа?

12. Какие требования предъявляются к безопасному хранению паролей? Объясните принцип хеширования с солью, работу адаптивных алгоритмов (bcrypt, Argon2, scrypt) и необходимость настройки стоимости вычисления.

13. Сравните подходы к управлению сессиями: серверные сессии, JWT, opaque tokens. Какие преимущества и риски несёт каждый подход с точки зрения масштабируемости, отзыва токенов и защиты от кражи?

14. Как реализовать безопасную многофакторную аутентификацию (MFA)? Какие факторы вы знаете (знание, владение, биометрия), как защитить процесс регистрации устройств и восстановить доступ при потере фактора?

15. Что такое принцип наименьших привилегий (PoLP) и как он применяется при проектировании ролевой (RBAC) и атрибутной (ABAC) моделей авторизации? Приведите пример эскалации привилегий при нарушении этого принципа.

16. В чём разница между симметричным и асимметричным шифрованием? Приведите примеры алгоритмов (AES, ChaCha20, RSA, ECC), их типичные параметры и сценарии применения в прикладной разработке.

17. Зачем нужны криптографические хеш-функции и коды аутентификации сообщений (MAC, HMAC)? Как отличить хеш от шифра, когда применять каждый механизм и почему нельзя использовать обычный хеш для защиты паролей?

18. Что такое вектор инициализации (IV) и nonce? Почему их повторное использование с одним ключом ломает безопасность режимов шифрования (CBC, GCM) и как генерировать их криптографически стойким способом?

19. Как правильно управлять криптографическими ключами в приложении? Опишите жизненный цикл ключа: генерация, хранение (HSM, KMS, env-variables), ротация, отзыв, архивирование и уничтожение.

20. Какие типичные ошибки разработчиков при использовании криптобиблиотек вы знаете (самописные алгоритмы, ECB-режим, хардкод ключей, игнорирование аутентификации шифротекста)? Как их предотвратить через code review и статический анализ?

21. Опишите процесс установления защищённого соединения по TLS 1.3. Какие этапы рукопожатия обеспечивают аутентификацию сервера/клиента, согласование шифров и выработку сеансового ключа?

22. В чём разница между шифрованием данных в покое (at rest), в передаче (in transit) и в использовании (in use)? Приведите примеры технологий для каждого состояния (TDE, TLS, confidential computing) и их комбинации.

23. Как работают протоколы аутентифицированного шифрования (AEAD), например AES-GCM или ChaCha20-Poly1305? Почему важно проверять тег аутентификации перед расшифровкой и какие атаки предотвращает этот порядок?

24. Что такое прямая секретность (Forward Secrecy) и почему она критична для долгосрочной защиты переписки? Как её обеспечивают эфемерные ключи в протоколах (DHE, ECDHE) и какие компромиссы по производительности это влечёт?

25. Какие дополнительные требования предъявляются к криптографии в мобильных приложениях (iOS/Android)? Как защитить ключи от извлечения (Keychain, Keystore, Secure Enclave), обнаружить рут/джейлбрейк и предотвратить отладку?

26. Как обеспечить безопасность криптоопераций в веб-приложениях? Опишите роль

Web Crypto API, необходимость защиты от XSS для доступа к ключам, и почему криптография на клиенте не заменяет защиту на сервере.

27. Какие особенности криптографии в облачных средах (AWS KMS, Azure Key Vault, GCP KMS)? Как реализовать шифрование на стороне клиента (client-side encryption) и разделить ответственность между провайдером и разработчиком?

28. Что такое управление секретами (secrets management) в распределённых системах? Сравните подходы: переменные окружения, vault-сервисы, sidecar-агенты — по безопасности, удобству развёртывания и аудиту.

29. Какие принципы безопасного проектирования архитектуры вы знаете (защита по умолчанию, минимизация поверхности атаки, эшелонированная защита, безопасные умолчания)? Как они применяются на уровне микросервисов, API-шлюзов и баз данных?

30. Как построить процесс безопасной разработки (Secure SDLC) в команде? Опишите интеграцию threat modeling, SAST/DAST-сканирования, зависимого мониторинга, пентестов и реагирования на инциденты в CI/CD-конвейер.

Критерии оценки:

Отметка «5»: ответ полный и правильный на основании изученных теорий; материал изложен в определенной логической последовательности, литературным языком. Ответ самостоятельный.

Отметка «4»: ответ полный и правильный на основании изученных теорий; материал изложен в определенной логической последовательности, при этом допущены две-три несущественные ошибки, исправленные по требованию преподавателя.

Отметка «3»: ответ полный, но при этом допущена существенная ошибка, или неполный, несвязный.

Фонд тестовых заданий

по МДК.02.01 Разработка программных модулей

Тип 1: Один правильный ответ (выбор из 4 вариантов)

1. **Какой принцип модульной архитектуры требует, чтобы каждый модуль решал одну конкретную задачу?**

А) Слабая связанность
Б) Монолитность
В) Высокая связность (High Cohesion)
Г) Полиморфизм

2. **Какая коллекция в большинстве ООП-языков гарантирует порядок вставки элементов и позволяет обращаться к ним по индексу?**

А) HashSet
Б) List / ArrayList
В) Dictionary / Map
Г) Queue

3. **Какой режим открытия файла позволяет читать данные, но запрещает запись?**

А) r (read)
Б) w (write)
В) a (append)
Г) x (create new)

4. **Ключевое слово, которое приостанавливает выполнение асинхронного метода до завершения ожидаемой задачи, не блокируя UI-поток полностью:**

А) async
Б) await
В) yield
Г) task

5. **Какая асимптотическая сложность соответствует линейному поиску в неотсортированном массиве в худшем случае?**
А) O(1)
Б) O(log n)
В) O(n) Г)
$O(n^2)$
6. **В какой структуре данных вставка элемента происходит в конец, а удаление — из начала (FIFO)?** А)
Стек Б)
Очередь
В) Связный список
Г) Дерево
7. **Алгоритм поиска кратчайшего пути во взвешенном графе с неотрицательными весами рёбер:**
А) Поиск в глубину (DFS)
Б) Алгоритм Краскала
В) Топологическая сортировка
Г) Алгоритм Дейкстры
8. **Какой метод разрешения коллизий использует цепочку связанных списков в каждой ячейке хеш-таблицы?**
А) Открытая адресация
Б) Метод цепочек (Chaining)
В) Двойное хеширование
Г) Рехеширование
9. **Алгоритм поиска подстроки, использующий префикс-функцию для эффективного пропуска несовпадающих символов:**
А) Кнута-Морриса-Пратта (KMP)
Б) Наивный
В) Бойера-Мура
Г) Рабина-Карпа
10. **В максимальной куче (Max-Heap) родительский узел всегда:**
А) Меньше или равен дочерним
Б) Больше или равен дочерним
В) Равен сумме дочерних
Г) Не имеет отношения к дочерним
11. **Какой принцип безопасности предполагает предоставление модулю минимально необходимых прав для выполнения его функций?**
А) Принцип наименьших привилегий
Б) Принцип максимальной открытости
В) Принцип полного доверия
Г) Принцип изоляции сред
12. **Какая UML-диаграмма используется для отображения статической структуры системы, включая классы, их атрибуты, методы и отношения?**
А) Диаграмма последовательностей
Б) Диаграмма развёртывания
В) Диаграмма классов
Г) Диаграмма состояний
13. **Что происходит при срабатывании события «Click» на кнопке в GUI-приложении?**
А) Компьютер перезагружается
Б) Вызывается привязанный обработчик события (метод)
В) Закрывается приложение
Г) Очищается память

14. **Какой механизм чаще всего используется для безопасного обновления элементов GUI из фонового потока?**
А) Прямое изменение свойств контрола
Б) Механизм диспетчера/контекста синхронизации (Dispatcher / Invoke)
В) Перезапуск приложения
Г) Очистка кэша
15. **Алгоритм сортировки, стабильный и гарантирующий сложность O(n log n) даже в худшем случае:**
А) Быстрая сортировка (QuickSort)
Б) Сортировка пузырьком
В) Сортировка слиянием (MergeSort)
Г) Сортировка вставками

№	Ответ	№	Ответ
1	В	11	А
2	Б	12	В
3	А	13	Б
4	Б	14	Б
5	В	15	В
6	Б		
7	Г		
8	Б		
9	А		
10	Б		

Тип 2: Несколько правильных ответов (выбор из 5 вариантов)
16. **Выберите два преимущества модульной архитектуры:**
А) Упрощение тестирования отдельных частей
Б) Увеличение размера исполняемого файла
В) Возможность параллельной разработки разными командами
Г) Жёсткая зависимость от одной платформы
17. **Какие из перечисленных инструментов являются системами контроля версий?**
А) Docker
Б) Git
В) Jenkins
Г) SVN
Д) Figma
18. **Выберите паттерны, относящиеся к порождающим (Creational):**
А) Singleton
Б) Observer
В) Factory Method
Г) Strategy
Д) Builder
19. **Какие операции для статического массива имеют сложность O(1)?**
А) Доступ по индексу
Б) Вставка в начало
В) Удаление из середины
Г) Изменение значения по известному индексу
20. **Какие кодировки являются универсальными (поддерживают символы большинства языков мира)?**
А) ASCII
Б) UTF-8

В) UTF-16
Г) KOI8-R
Д) Windows-1251

21. **Выберите верные утверждения о бинарном дереве поиска (BST):**

А) В левом поддереве все узлы меньше корня
Б) Допускаются дубликаты в любом узле без ограничений
В) В правом поддереве все узлы больше корня
Г) Обход в глубину всегда возвращает отсортированный массив

22. **Какие этапы входят в процесс проектирования программного модуля?**

А) Сбор и анализ требований
Б) Декомпозиция задачи
В) Написание маркетингового отчёта
Г) Создание спецификации
Д) Развёртывание в продакшене без тестов

23. **Какие механизмы относятся к полиморфизму в ООП?**

А) Переопределение методов (Overriding)
Б) Реализация интерфейсов
В) Использование модификатора static
Г) Перегрузка методов (Overloading)
Д) Наследование от sealed-класса

24. **Выберите элементы управления, предназначенные для ввода текстовых данных пользователем:**

А) TextBox / TextField
Б) Label В) ComboBox (в режиме ввода)
Г) RadioButton
Д) PasswordBox

25. **Какие инструменты/библиотеки предназначены для визуализации данных (построения графиков и диаграмм)?**

А) Matplotlib
Б) Chart.js
В) OpenCV
Г) D3.js
Д) FFmpeg

№	Правильные варианты
16	А, В
17	Б, Г
18	А, В, Д
19	А, Г
20	Б, В
21	А, В
22	А, Б, Г
23	А, Б, Г
24	А, В, Д
25	А, Б, Г

Тип 3: Установление соответствия

26. **Установите соответствие между архитектурным шаблоном и его компонентами:**

1.	MVC	А) Модель, Представление, Модель представления
2.	MVVM	Б) Модель, Представление, Контроллер
3.	MVP	В) Модель, Представление, Презентер

27. **Установите соответствие между типом отношения классов и его описанием:**

1.	Наследование	А) Отношение «часть-целое», где часть может существовать
2.	Композиция	отдельно от целого
3.	Агрегация	Б) Отношение «является» (is-a)
		В) Отношение «часть-целое», где часть не существует без целого

1. **Установите соответствие между методом задачи (Task) и его назначением:**

1.	Task.Wait()	А) Запускает задачу в фоновом потоке из пула
2.	Task.ContinueWith()	Б) Блокирует вызывающий поток до завершения задачи
3.	Task.Run()	В) Выполняет указанный код после завершения исходной задачи

1. **Установите соответствие между алгоритмом и его средней временной сложностью:**

1.	Бинарный поиск	А) $O(n^2)$
2.	Быстрая сортировка (QuickSort)	Б) O(log n)
3.	Сортировка пузырьком	В) O(n log n)

1. **Установите соответствие между строковым алгоритмом/структурой и его назначением:**

1.	Расстояние Левенштейна	А) Поиск подстроки с использованием хеширования
2.	Алгоритм Рабина-Карпа	Б) Измерение минимального количества правок для
3.	Триальное дерево (Trie)	превращения одной строки в другую
		В) Эффективное хранение и поиск строк с общим префиксом

1. **Установите соответствие между принципом ООП и его определением:**

1.	Инкапсуляция	А) Способность объектов с одинаковой спецификацией иметь
2.	Полиморфизм	различную реализацию
3.	Наследование	Б) Механизм создания новых классов на основе существующих
		В) Сокрытие внутренней реализации и защита данных через методы доступа

1. **Установите соответствие между типом окна и его поведением:**

1.	Модальное окно	А) Позволяет пользователю переключаться между ним и
2.	Немодальное окно	родительским окном
3.	Диалоговое окно	Б) Блокирует взаимодействие с родительским окном до своего закрытия
		В) Специализированное окно для получения ответа или выбора параметров

№	Ответ
26	1-Б, 2-А, 3-В
27	1-Б, 2-В, 3-А
28	1-Б, 2-В, 3-А
29	1-Б, 2-В, 3-А
30	1-Б, 2-А, 3-В
31	1-В, 2-А, 3-Б
32	1-Б, 2-А, 3-В

Тип 4: Установление последовательности

33. **Установите правильную последовательность этапов работы с задачей (Task) в параллельном программировании:**

1. Запуск задачи
2. Создание задачи (инициализация)
3. Ожидание результата / настройка продолжения

34. Получение результата или обработка исключения **Установите последовательность действий при обработке пользовательского ввода в GUI:**

1. Вызов обработчика события
2. Генерация события (например, клик мыши)
3. Обновление интерфейса / модели
4. Регистрация обработчика на элементе управления

1. **Установите последовательность шагов алгоритма Дейкстры:**

1. Выбор вершины с минимальным расстоянием из непосещённых
2. Инициализация расстояний (0 для старта, ∞ для остальных)
3. Обновление расстояний до соседей выбранной вершины
4. Отметка вершины как посещённой

№	Правильный порядок
33	2 → 1 → 3 → 4
35	4 → 2 → 1 → 3
35	2 → 1 → 3 → 4

Тип 5: Верно / Неверно

36. **В шаблоне MVC Контроллер отвечает за обработку пользовательского ввода и обновление Модели.**

○ Верно
○ Неверно

Верно

37. **Стандартные библиотеки в современных языках не требуют явного подключения (import/using) перед использованием.**

○ Верно
○ Неверно

Неверно

38. **Пул потоков (Thread Pool) создаёт новый поток для каждой новой задачи, что гарантирует максимальную производительность.**

○ Верно
○ Неверно

Неверно *(Он переиспользует существующие потоки)*

39. **Хеш-функция всегда должна возвращать уникальное значение для каждого возможного ключа.**

○ Верно
○ Неверно

Неверно *(Коллизии неизбежны при конечном размере таблицы)*

40. **Динамическое программирование всегда работает быстрее жадных алгоритмов.**

○ Верно
○ Неверно

Неверно *(Зависит от задачи; ДП часто имеет больше накладных расходов)*

41. **Менеджер компоновки (Layout Manager) в GUI-фреймворках позволяет элементам интерфейса автоматически адаптироваться к изменению размера окна.**

○ Верно
○ Неверно

Верно

42. **Длительные вычисления в GUI-приложении следует выполнять в основном (UI) потоке, чтобы интерфейс оставался отзывчивым.**

○ Верно
○ Неверно

Неверно *(Их необходимо выносить в фоновые потоки или async-задачи)*

43. **Масштабируемость приложения означает возможность увеличения нагрузки без пропорционального роста затрат или потери производительности.**

○ Верно

○ Неверно

Верно

№	Ответ
36	Верно
37	Неверно
38	Неверно *(Он переиспользует существующие потоки)*
39	Неверно *(Коллизии неизбежны при конечном размере таблицы)*
40	Неверно *(Зависит от задачи; ДП часто имеет больше накладных расходов)*
41	Верно
42	Неверно *(Их необходимо выносить в фоновые потоки или async-задачи)*
43	Верно

Тип 6: Заполнение пропуска / Краткий ответ

44. **Команда в Git, создающая новую ветку и сразу переключающаяся на неё: git _____ <имя_ветки>**

45. **Регулярное выражение ^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}$ используется для валидации _____.**

46. **Основная идея динамического программирования — разбиение сложной задачи на _____, решения которых запоминаются для повторного использования.**

47. **Диаграмма _____ показывает физическую организацию программного обеспечения в виде компонентов (библиотек, модулей, файлов) и зависимостей между ними.**

48. **Как называется процесс разделения сложной системы на более мелкие, управляемые и относительно независимые части?**

49. **Как называется ситуация, когда рекурсивная функция вызывает саму себя бесконечно из-за отсутствия условия остановки?**

50. **Какое регулярное выражение проверяет, что строка состоит ровно из 6 цифр (например, почтовый индекс)?**

№	Ответ
44	checkout -b (допускается также switch -c)
45	email (электронной почты)
46	перекрывающиеся подзадачи *(допускается просто подзадачи)*
47	Компонентов **^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}$** Разберём, как работает это регулярное выражение — по частям: ^ — обозначает начало строки. Гарантирует, что проверка начинается с самого первого символа. [A-Za-z0-9._%+-]+ — описывает локальную часть адреса (до символа @): [A-Za-z0-9] — латинские буквы (верхнего и нижнего регистра) и цифры; ._%+- — разрешённые специальные символы: точка, подчёркивание, процент, плюс, дефис; + — квантификатор, означающий «один или более» повторений. То есть локальная часть не может быть пустой. @ — обязательный символ, разделяющий локальную часть и домен. [A-Za-z0-9.-]+ — описывает имя домена (после @, но до последней точки): [A-Za-z0-9] — латинские буквы и цифры; .- — точка и дефис (допустимы внутри доменного имени);

	+ — снова означает «один или более» символов. \. — экранированная точка. Соответствует литеральной точке перед доменом верхнего уровня (например, перед com, org). Без экранирования точка означала бы «любой символ», что недопустимо. [A-Z|a-z]{2,} — домен верхнего уровня (TLD): [A-Z|a-z] — латинские буквы (в выражении использован |, что технически некорректно для класса символов; правильнее было бы [A-Za-z]); {2,} — означает «два или более» символов. Это соответствует реальным TLD, которые имеют минимум 2 буквы (com, ru, info и т.д.). $ — обозначает конец строки. Гарантирует, что после TLD нет никаких других символов.
48	Декомпозиция
49	Бесконечная рекурсия *(или переполнение стека)*
50	^\d{6}$ **Объяснение** *В этом выражении:* \d - обозначает любую цифру. {6}` - указывает, что цифр должно быть шесть. *Выражение записано между символами / и $/, где:* ^ - указывает на начало строки; $ - указывает на конец строки.

по МДК.02.02 Осуществление интеграции программных модулей

Тип 1: Один правильный ответ (выбор из 4 вариантов)

1. **Какой HTTP-метод является идемпотентным и используется для полного обновления ресурса?**

А) POST
Б) PUT
В) PATCH
Г) DELETE

2. **Код состояния HTTP 201 Created возвращается сервером в ответ на:**

А) Успешный GET-запрос
Б) Ошибку аутентификации
В) Успешное создание нового ресурса
Г) Перенаправление на другой URL

3. **Какой заголовок HTTP указывает браузеру, как интерпретировать тело ответа?**

А) Accept
Б) Authorization
В) Content-Type
Г) Host

4. **Для отправки данных формы вместе с файлами в теле запроса используется кодировка:**

А) application/json
Б) application/x-www-form-urlencoded
В) text/plain
Г) multipart/form-data

5. **Основное отличие WebSocket от классического HTTP-запроса заключается в:**

А) Использовании только метода GET
Б) Односторонней передаче данных от сервера к клиенту
В) Постоянном двустороннем соединении после установки
Г) Отсутствии заголовков в запросе

6. **Какой стандарт авторизации использует токены в формате JSON с криптографической подписью и не требует хранения сессии на сервере?**
А) Session/Cookie
Б) OAuth 2.0 (Authorization Code)
В) JWT (JSON Web Token)
Г) Basic Auth
7. **Вертикальное масштабирование приложения предполагает:**
А) Добавление новых серверов в кластер
Б) Увеличение мощности существующего сервера (CPU, RAM, SSD)
В) Разделение монолита на микросервисы
Г) Нагрузочное тестирование перед деплоем
8. **Какой механизм защиты предотвращает атаки, когда злоумышленник пытается выполнить несанкционированные действия от имени авторизованного пользователя?**
А) Хеширование паролей
Б) CSRF-токены
В) Индексирование БД
Г) GZIP-сжатие
9. **Уровень логирования, используемый исключительно для отладки и не рекомендуется к включению в production:**
А) ERROR
Б) WARN
В) INFO
Г) DEBUG
10. **Какой инструмент является стандартом де-факто для контейнеризации приложений?**
А) VirtualBox
Б) Docker
В) Postman
Г) Jenkins
11. **Алгоритм хеширования паролей, специально разработанный для защиты от перебора и использующий «соль»:**
А) MD5
Б) SHA-256
В) bcrypt / Argon2
Г) Base64
12. **Профилирование кода позволяет:**
А) Автоматически исправлять синтаксические ошибки
Б) Выявить «узкие места» по времени выполнения и потреблению памяти
В) Сгенерировать документацию к API
Г) Зашифровать исходный код

№	Ответ	№	Ответ
1	Б	7	Б
2	В	8	Б
3	В	9	Г
4	Г	10	Б
5	В	11	В
6	В	12	Б

Тип 2: Несколько правильных ответов (выбор из 5 вариантов)
13. **Выберите HTTP-методы, которые НЕ должны изменять состояние ресурса на сервере:**
А) GET
Б) POST

В) PUT
Г) HEAD
Д) OPTIONS

14. **Какие из перечисленных компонентов входят в структуру JWT-токена?**

А) Header (заголовок)
Б) Payload (полезная нагрузка)
В) Signature (подпись)
Г) Cookie (файл куки)
Д) Session ID (идентификатор сессии)

15. **Преимущества микросервисной архитектуры перед монолитной:**

А) Независимое развертывание отдельных сервисов
Б) Возможность использования разных стеков технологий
В) Отсутствие сетевых задержек между компонентами
Г) Упрощение отладки распределённых транзакций
Д) Локализация отказов (падение одного сервиса не роняет всё приложение)

16. **Меры защиты от XSS-атак (межсайтовый скриптинг):**

А) Экранирование (escaping) выводимых данных
Б) Использование заголовка Content-Security-Policy
В) Валидация и санитизация пользовательского ввода
Г) Отключение JavaScript в браузере клиента
Д) Хранение всех данных в куки с флагом HttpOnly

17. **Способы оптимизации медленных запросов к реляционной БД:**

А) Создание индексов на полях фильтрации и сортировки
Б) Выборка только необходимых полей (SELECT id, name вместо SELECT *)
В) Использование LIMIT для постраничной выборки
Г) Отключение механизма транзакций
Д) Хранение всех данных в одной таблице без связей

18. **Какие метрики критичны для мониторинга работоспособности REST API?**

А) Время отклика (Response Time)
Б) Количество запросов в секунду (RPS)
В) Процент ошибок 4xx и 5xx
Г) Количество строк в исходном коде
Д) Загрузка CPU и памяти хоста

19. **Инструменты и практики CI/CD, используемые при сборке и доставке приложения:**

А) GitLab CI / GitHub Actions
Б) Docker Compose / Kubernetes
В) Artifacts Registry (реестр образов)
Г) Ручное копирование файлов по FTP
Д) Автоматические тесты перед деплоем

№	Правильные варианты
13	А, Г, Д
14	А, Б, В
15	А, Б, Д
16	А, Б, В
17	А, Б, В
18	А, Б, В, Д
19	А, Б, В, Д

Тип 3: Установление соответствия

20. **Соотнесите HTTP-код и его значение:**

1.	400	А) Запрещено: доступ отклонён, несмотря на аутентификацию
2.	401	Б) Плохой запрос: сервер не может обработать синтаксис
3.	403	В) Не авторизован: отсутствуют или невалидны учётные данные
4.	502	Г) Bad Gateway: шлюз получил невалидный ответ от вышестоящего сервера

21. **Соотнесите уровень логирования и тип события:**

1.	FATAL	А) Критическая ошибка, приводящая к остановке приложения
2.	INFO	Б) Ошибка выполнения операции, требующая внимания разработчика
3.	WARN	В) Потенциальная проблема или нештатная ситуация, но работа
4.	ERROR	продолжается
		Г) Информационное сообщение о штатном ходе работы

1. **Соотнесите протокол и характеристику безопасности:**

1.	HTTP	А) Данные передаются в открытом виде, уязвимы к перехвату
2.	HTTPS	Б) Двустороннее соединение без шифрования
3.	WS	В) HTTP с шифрованием транспортного уровня (TLS/SSL)
4.	WSS	Г) WebSocket с шифрованием транспортного уровня

1. **Соотнесите тип уязвимости и механизм атаки:**

1.	SQL-инъекция	А) Внедрение вредоносного JS-кода, выполняемого в браузере
2.	XSS	жертвы
3.	CSRF	Б) Выполнение произвольных SQL-команд через невалидированный ввод
		В) Принуждение браузера авторизованного пользователя выполнить действие без его ведома

1. **Соотнесите тип взаимодействия микросервисов и пример технологии:**

1.	Синхронное	А) REST API / gRPC
2.	Асинхронное (брокер сообщений)	Б) RabbitMQ / Apache Kafka

№	Ответ
20	1-Б, 2-В, 3-А, 4-Г
21	1-А, 2-Б, 3-В, 4-Г *(Примечание: в задании опечатка нумерации, исправлено на 1-FATAL, 2-INFO, 3-WARN, 4-ERROR)*
22	1-А, 2-В, 3-Б, 4-Г
23	1-Б, 2-А, 3-В
24	1-А, 2-Б

Тип 4: Верно / Неверно

25. **REST API обязательно должен использовать JSON в качестве единственного формата обмена данными.**

○ Верно
○ Неверно

26. **При использовании JWT серверу не нужно хранить сессию пользователя в БД или памяти для валидации каждого запроса.**

○ Верно
○ Неверно

подписи)

27. **Переменные окружения (.env) рекомендуется хранить в публичном Git-репозитории для удобства команды.**

○ Верно
○ Неверно

28. **Логирование уровня ERROR полностью заменяет необходимость в мониторинге метрик производительности и алертинге.**

○ Верно
○ Неверно

29. **Контейнер виртуализирует операционную систему, а виртуальная машина виртуализирует оборудование.**
○ Верно
○ Неверно
30. **Хеширование без «соли» (salt) уязвимо к атакам с использованием радужных таблиц (rainbow tables).**
○ Верно
○ Неверно
31. **Горизонтальное масштабирование REST API требует обязательного использования балансировщика нагрузки (Load Balancer).**
○ Верно
○ Неверно
32. **Кэширование данных (Redis, Memcached) всегда устраняет проблему медленных запросов к БД без побочных эффектов.**
○ Верно
○ Неверно

№	Ответ
25	**Неверно** *(REST допускает XML, YAML, Protobuf, plain text и др.)*
26	**Верно** *(JWT stateless, проверка производится по криптографической подписи)*
27	**Неверно** *(Секреты никогда не коммитятся в репозиторий)*
28	**Неверно** *(Логи показывают причины сбоев, метрики показывают состояние системы в реальном времени)*
29	**Верно**
30	**Верно**
31	**Верно**
32	**Неверно** *(Возможны проблемы инвалидации кэша, потребление памяти, устаревшие данные)*

Тип 5: Заполнение пропуска / Краткий ответ

33. **Заголовок HTTP, в который клиент помещает токен авторизации в формате Bearer <token>: _____.**
34. **Параметры URL, передаваемые после символа ? и разделяемые &, называются _____ параметрами.**
35. **Протокол, обеспечивающий защищённое двустороннее соединение для WebSocket: _____.**
36. **Механизм проверки входных данных перед их обработкой или сохранением в БД называется _____.**
37. **Файл конфигурации, описывающий этапы сборки контейнерного образа, называется _____.**
38. **Архитектурный стиль взаимодействия, при котором клиент и сервер обмениваются сообщениями без ожидания немедленного ответа, называется _____.**
39. **Процесс преобразования объекта памяти (например, JSON) в байтовый поток для передачи по сети или сохранения называется _____.**

№	Ответ
33	Сериализация
34	Query (или query string)
35	WSS (WebSocket Secure)
36	Валидация
37	Dockerfile
38	Асинхронное взаимодействие (или Event-driven / Message Queue)

39	Authorization

Тип 6: Установление последовательности

40. **Установите порядок обработки REST-запроса сервером:**

1. Маршрутизатор определяет контроллер/обработчик
2. Клиент формирует HTTP-запрос
3. Выполняется бизнес-логика и запрос к БД
4. Сервер возвращает HTTP-ответ с кодом состояния
5. Происходит валидация и десериализация входных данных

41. **Этапы аутентификации с использованием JWT:**

1. Клиент сохраняет токен локально
2. Сервер проверяет учётные данные
3. Клиент отправляет логин и пароль
4. Сервер проверяет подпись и срок действия токена в заголовке Authorization
5. Сервер генерирует и отдаёт JWT

1. При запросе к защищённому ресурсу клиент прикрепляет токен **Последовательность действий при контейнеризации и запуске приложения:**

1. Запуск контейнера из образа (docker run / docker-compose up)
2. Написание Dockerfile с инструкциями сборки
3. Сборка образа (docker build)
4. Публикация образа в Registry (docker push)
5. Локальное тестирование образа

№	Правильный порядок
40	**2 → 1 → 5 → 3 → 4**
41	**2 → 3 → 5 → 4 → 1**
42	**3 → 2 → 5 → 1 → 6 → 4**

Тип 7: Ситуационные задачи / Анализ

43. **Разработчик отправляет POST /api/users с телом {"name":"Ivan"}, но не указывает заголовок Content-Type. Сервер настроен на строгую валидацию. Какой HTTP-статус вернёт сервер?**

А) 200 OK
Б) 404 Not Found
В) 400 Bad Request или 415 Unsupported Media Type
Г) 500 Internal Server Error

44. **В приложении наблюдается проблема N+1 запросов к БД при получении списка пользователей с их профилями. Какое решение наиболее эффективно устраняет проблему?**

А) Увеличить размер кэша Redis
Б) Использовать JOIN или eager loading для выборки связанных данных одним запросом
В) Перевести БД на NoSQL
Г) Добавить индекс на все поля таблицы

45. **При тестировании API обнаружено, что злоумышленник может получить доступ к данным других пользователей, меняя user_id в URL (GET /api/users/101/profile → /api/users/102/profile). Как называется эта уязвимость и как её исправить?**

А) XSS. Исправить экранированием вывода.
Б) IDOR (Insecure Direct Object Reference). Исправить проверкой прав доступа к ресурсу на сервере.
В) SQLi. Исправить параметризованными запросами.
Г) CSRF. Исправить токенами в форме.

46. **WebSocket-соединение неожиданно разрывается каждые 30 секунд при бездействии. Какая настройка обычно решает проблему?**

А) Увеличение max_request_size
Б) Отключение валидации заголовков
В) Переход на HTTP/2
Г) Настройка Ping/Pong (heartbeat) для поддержания соединения

47. **В микросервисной архитектуре сервис оплаты зависает, ожидая ответа от сервиса уведомлений. Какой паттерн следует внедрить, чтобы избежать каскадного отказа?**
А) Circuit Breaker (Автоматический выключатель) + Timeout
Б) Long Polling
В) Монолитная сборка
Г) Синхронные gRPC-вызовы без лимитов

48. **При нагрузочном тестировании API время ответа растёт экспоненциально. Профилировщик показывает, что 80% времени тратится на сериализацию больших JSON-объектов. Какое решение оптимизирует ситуацию?**
А) Пагинация данных и выборка только необходимых полей
Б) Отключение логирования В) Перевод всех запросов в POST
Г) Увеличение размера оперативной памяти сервера в 4 раза **Для защиты от брутфорс-атак на эндпоинт /login администратор хочет ограничить количество попыток входа. Какой механизм следует применить?**
А) Хеширование паролей
Б) Rate Limiting (ограничение частоты запросов по IP или аккаунту)
В) Включение CORS
Г) Сжатие GZIP

49. **Разработчик конфигурирует маршрутизацию: GET /users возвращает список, GET /users/{id} возвращает профиль. При запросе GET /users/new сервер возвращает 404. Почему?**
А) Маршрутизатор интерпретирует new как {id}, но в БД нет пользователя с таким ID. Валидация формата ID должна отсекать такие запросы на уровне роутинга.
Б) Забыт заголовок Accept
В) Сервер не поддерживает GET-запросы
Г) Конфликт с файлом new.html в статике

№	Ответ
43	В
44	Б
45	Б
46	Г
47	А
48	А
50	Б

по МДК.02.03 Поддержка и тестирование программных модулей

Тип 1: Один правильный ответ (выбор из 4 вариантов)

1. **Метрика, показывающая количество независимых путей выполнения в модуле и используемая для оценки сложности тестирования:**
А) Глубина наследования
Б) Цикломатическая сложность
В) Плотность комментариев
Г) Индекс сопровождаемости

2. **Международный стандарт, описывающий модель качества программного обеспечения (функциональность, надежность, удобство и т.д.):**
А) ISO 9001
Б) ISO/IEC 25010
В) IEEE 802.11

Г) RFC 793

3. **Несоответствие программы требованиям или спецификации, обнаруженное в исходном коде, называется:**

А) Сбой (Failure)

Б) Отказ

В) Дефект (Bug)

Г) Инцидент

4. **Инструмент, позволяющий приостановить выполнение программы, пошагово выполнять строки и отслеживать значения переменных:**

5. А) Компилятор

6. Б) Отладчик (Debugger)

7. В) Линтер

8. Г) Сборщик мусора

9. **Блок обработки исключений, который выполняется всегда, независимо от того, произошло исключение или нет:**

А) try

Б) catch

В) finally

Г) throw

10. **Этап тестирования, на котором проверяется корректность взаимодействия двух и более программных модулей:**

А) Модульное

Б) Интеграционное

В) Системное

Г) Приемочное

11. **Метод тестирования по белому ящику, гарантирующий выполнение каждой строки кода хотя бы один раз:**

А) Покрытие условий

Б) Покрытие операторов

В) Покрытие путей

Г) Комбинаторное покрытие

12. **Тестирование, направленное на проверку соответствия ПО бизнес-требованиям заказчика перед выпуском в эксплуатацию:**

А) Модульное

Б) Нагрузочное

В) Приемочное

Г) Стресс-тестирование

13. **Динамическое тестирование отличается от статического тем, что:**

А) Проводится без запуска кода

Б) Выполняется во время работы программы с реальными или тестовыми данными

В) Используется только на этапе проектирования

Г) Требует обязательного использования ИИ

14. **Метод граничных значений применяется для выбора тестовых данных:**

А) На границах классов эквивалентности

Б) Случайным образом из всего диапазона

В) Исключительно для проверки циклов

Г) Только для UI-элементов

15. **При стресс-тестировании система проверяется на:**

А) Удобство пользовательского интерфейса

Б) Работу в условиях превышения нормативной нагрузки

В) Соответствие требованиям информационной безопасности

Г) Корректность расчетов бизнес-логики

16. **Какая из перечисленных метрик относится к динамическим показателям качества?**
А) Цикломатическая сложность
Б) Покрытие кода тестами (Code Coverage)
В) Количество строк кода
Г) Глубина вложенности условий
17. **Стратегия поиска ошибки, при которой диапазон кода делится пополам для быстрой локализации дефекта:**
А) Метод исключения
Б) Метод половинного деления
В) Проверка граничных условий
Г) Метод золотого сечения
18. **Статус дефекта Resolved в трекере задач означает, что:**
А) Ошибка подтверждена тестировщиком
Б) Разработчик исправил ошибку и передал сборку на повторную проверку
В) Задача отклонена как «не баг»
Г) Ошибка отложена до следующего релиза **Тестирование по черному ящику подразумевает, что тестировщик:**
А) Имеет полный доступ к исходному коду
Б) Проверяет систему по внешним спецификациям, не зная внутренней реализации
В) Пишет только модульные тесты
Г) Анализирует только производительность базы данных

№	Ответ	№	Ответ
1	Б	11	Б
2	Б	12	В
3	В	13	Б
4	Б	14	А
5	Б	15	Б
6	В	16	Б
7	Б	17	Б
8	Б	18	Б
9	В	19	Б
10	Б		

Тип 2: Несколько правильных ответов (выбор из 5 вариантов)
16. **Выберите динамические метрики качества ПО:**
А) Время отклика
Б) Цикломатическая сложность
В) Частота отказов в production
Г) Количество строк кода
Д) Покрытие кода тестами
17. **Принципы качественного проектирования модулей включают:**
А) Высокая связность (High Cohesion)
Б) Слабое зацепление (Low Coupling)
В) Монолитность бизнес-логики
Г) Принцип единственной ответственности (SRP)
Д) Активное использование глобальных переменных
18. **Основные типы ошибок в программном обеспечении:**
А) Логические
Б) Синтаксические
В) Временные (таймингговые/состояния гонки)
Г) Эстетические

Д) Аппаратные/средовые

19. **Блоки, входящие в стандартный механизм обработки исключений:**

А) try

Б) catch / except

В) finally

Г) loop

Д) switch

20. **Этапы процесса тестирования ПО:**

А) Планирование и анализ требований

Б) Разработка тестовой документации

В) Написание продакшн-кода

Г) Выполнение тестов и фиксация дефектов

Д) Закрытие тестирования и составление отчета

21. **Инструменты статического анализа качества кода:**

А) SonarQube

Б) ESLint / Pylint

В) JUnit / pytest

Г) Checkstyle

Д) Postman

22. **Виды нефункционального тестирования:**

А) Нагрузочное

Б) Тестирование безопасности

В) Тестирование бизнес-логики

Г) Тестирование юзабилити

Д) Тестирование совместимости

23. **Преимущества автоматизированного тестирования:**

А) Быстрое выполнение регрессионных проверок

Б) Исключение человеческого фактора при рутинных сценариях

В) Полная замена ручного тестирования

Г) Интеграция в CI/CD пайплайн

Д) Автоматическое исправление найденных багов

24. **Методы структурного покрытия кода (белый ящик):**

А) Покрытие операторов

Б) Покрытие решений (ветвей)

В) Покрытие условий

Г) Покрытие классов эквивалентности

Д) Покрытие путей

25. **Функциональность системы контроля дефектов (Bug Tracker):**

А) Регистрация и классификация багов

Б) Назначение ответственных разработчиков

В) Отслеживание статуса исправления

Г) Автоматическая генерация unit-тестов

Д) Ведение истории изменений и комментариев

№	Правильные варианты
16	А, В, Д
17	А, Б, Г
18	А, Б, В, Д
16	А, Б, В
20	А, Б, Г, Д
21	А, Б, Г
22	А, Б, Г, Д

23	А, Б, Г
24	А, Б, В, Д
25	А, Б, В, Д

Тип 3: Установление соответствия

26. **Соотнесите термин отладки и его описание:**

1.	Точка останова (Breakpoint)	А) Выполнение текущей строки с заходом внутрь вызываемой функции
2.	Step Over	Б) Список переменных, значения которых отслеживаются в реальном времени
3.	Step Into	В) Строка кода, при достижении которой выполнение приостанавливается
4.	Watch List	Г) Выполнение текущей строки без захода в вызываемые функции

27. **Соотнесите тип тестирования и его основную цель:**

1.	Модульное	А) Проверка готовности продукта к внедрению заказчиком
2.	Интеграционное	Б) Изолированная проверка отдельной функции/класса
3.	Системное	В) Проверка взаимодействия модулей и передачи данных
4.	Приемочное	Г) Проверка работы всей системы в условиях, близких к боевым

1. **Соотнесите метод черного ящика и пример применения:**

1.	Классы эквивалентности	А) Тестирование поля возраста: проверка 17, 18, 60, 61 при допустимом 18-60
2.	Граничные значения	Б) Тестирование логики: "Если скидка > 10% И сумма > 5000 → применить бонус"
3.	Таблица решений	В) Выбор одного валидного и одного невалидного значения для поля email

1. **Соотнесите уровень критичности дефекта и описание:**

1.	Critical / Blocker	А) Опечатка в интерфейсе, не влияющая на функционал
2.	Major	Б) Ошибка, полностью блокирующая работу с системой
3.	Minor	В) Некорректный расчет суммы, но есть обходной путь
4.	Trivial / Cosmetic	Г) Неверное сообщение об ошибке, основной функционал работает

1. **Соотнесите инструмент и его назначение:**

1.	JUnit / pytest	А) Система управления задачами и дефектами
2.	Selenium / Playwright	Б) Фреймворк для модульного тестирования
3.	SonarQube	В) Автоматизация UI-тестирования
4.	JIRA / YouTrack	Г) Платформа статического анализа и контроля качества кода

№	Ответ
26	1-Б, 2-В, 3-Г, 4-А
27	1-Б, 2-В, 3-Г, 4-А
28	1-В, 2-А, 3-Б
29	1-Б, 2-В, 3-Г, 4-А
30	1-В, 2-Г, 3-А, 4-Б

Тип 4: Верно / Неверно

31. **Статическое тестирование выполняется без запуска исполняемого кода программы.**

○ Верно

○ Неверно

32. **Цикломатическая сложность равна 1 для модуля, не содержащего ветвлений и циклов.**

○ Верно

○ Неверно

33. **Обработка исключений (try-catch) должна использоваться для управления обычным потоком выполнения (например, валидации ввода вместо if-else).**
○ Верно
○ Неверно
34. **Метод покрытия операторов гарантирует нахождение всех логических ошибок в составных условиях.**
○ Верно
○ Неверно
35. **Интеграционное тестирование «сверху вниз» (Top-Down) использует заглушки (stubs) для имитации нижележащих модулей.**

№	Ответ
31	**Верно**
32	**Верно**
33	**Неверно** *(Исключения предназначены для исключительных, а не штатных ситуаций)*
34	**Верно**
35	**Неверно** *(Требуется покрытие условий/решений или MC/DC)*

Тип 5: Заполнение пропуска / Краткий ответ
36. **Ситуация, когда программа выдает некорректный результат или аварийно завершает работу из-за внутреннего дефекта, называется _____.**
37. **Метод тестирования, разбивающий входные данные на группы, где поведение системы должно быть идентичным, называется методом _____.**
38. **Документ, фиксирующий описание ошибки, шаги воспроизведения, фактический и ожидаемый результат, называется _____. Тестирование, при котором разработчик проверяет работоспособность отдельной функции/класса с использованием фреймворков вроде JUnit или NUnit, называется _____ тестированием.**
39. **Процесс замены реальных зависимостей на упрощенные имитации при модульном тестировании называется использованием _____.**

№	Ответ
36	Моков (или Заглушек / Стабов)
37	Классов эквивалентности
38	Баг-репорт (Отчет об ошибке)
39	Модульным (Unit)
40	Сбой (или Отказ / Failure)

Тип 6: Установление последовательности
41. **Порядок обработки исключения в коде:**
1. Возникновение ошибки в блоке try
2. Передача управления в соответствующий catch
3. Логирование/обработка ошибки
4. Выполнение блока finally
5. Продолжение работы или завершение программы
42. **Стандартный жизненный цикл дефекта при успешном исправлении:**
1. New (Новый)
2. Open / In Progress (В работе)
3. Fixed / Resolved (Исправлен)
4. Verified / Closed (Проверен и закрыт)
1. **Последовательность шагов паттерна Arrange-Act-Assert в модульном тесте:**
1. Подготовка тестовых данных и окружения
2. Вызов тестируемой функции/метода

3. Проверка результата с помощью утверждений (assert)
4. Очистка временных ресурсов (если требуется)

1. **Этапы нагрузочного тестирования:**

1. Определение целей и метрик (KPI)
2. Создание сценариев нагрузки
3. Выполнение тестов и сбор метрик
4. Анализ результатов и выявление узких мест
5. Подготовка отчета

1. **Порядок поиска ошибки методом половинного деления:**

1. Определение диапазона кода, где проявляется сбой
2. Разделение диапазона на две части
3. Проверка работы первой половины
4. Сужение диапазона до локализации дефекта
5. Устранение ошибки

№	Правильный порядок
41	1 → 2 → 3 → 4 → 5
42	1 → 2 → 3 → 4 → 5
43	1 → 2 → 3 → 4
44	1 → 2 → 3 → 4
45	1 → 2 → 3 → 4 → 5

Тип 7: Ситуационные задачи / Анализ

46. **Разработчик написал функцию divide(a, b). Тест divide(10, 2) прошел, но при divide(5, 0) программа падает. Какой подход наиболее эффективен для предотвращения такой ситуации на этапе разработки?**

А) Нагрузочное тестирование
Б) Модульное тестирование с проверкой граничных и исключительных условий
В) Тестирование пользовательского интерфейса
Г) Стресс-тестирование

47. **При тестировании поля ввода «Возраст» (допустимо от 18 до 65 лет включительно) методом граничных значений тестировщик должен проверить значения:**

А) 0, 18, 65, 100
Б) 10, 50, 70, 90
В) 18, 30, 45, 65
Г) 17, 18, 65, 66

48. **В системе логирования ежедневно создаются файлы app_log_20241001.txt размером 500 МБ, что быстро заполняет диск сервера. Какой принцип работы с логами нарушен?**

А) Валидация входных данных
Б) Ротация и архивирование логов
В) Использование структурных исключений
Г) Покрытие кода тестами

49. **Тестировщик обнаружил, что при быстром последовательном нажатии кнопки «Оплатить» транзакция проводится дважды. К какому типу дефекта это относится?**

А) Логическая ошибка / Отсутствие идемпотентности и обработки состояния
Б) Ошибка верстки
В) Синтаксическая ошибка
Г) Ошибка конфигурации сервера

50. **Для тестирования API микросервиса используется Postman и автоматические скрипты. Какой уровень тестирования это покрывает в первую очередь?**

А) Модульное
Б) Интеграционное / Контрактное

В) Системное UI
Г) Стресс-тестирование

№	Ответ
46	Б
47	Г *(Проверяются значения на границе и за её пределами)*
48	Б
49	А
50	Б

по МДК.02.04 Математическое моделирование

Тип 1: Один правильный ответ (выбор из 4 вариантов)

1. **Что такое математическая модель?**

А) Физический макет объекта в уменьшенном масштабе
Б) Совокупность математических соотношений, описывающих свойства и процессы объекта
В) Компьютерная программа с графическим интерфейсом
Г) Чертеж или схема устройства

2. **Графический метод решения задачи линейного программирования применим, когда число переменных равно:**

А) Одной
Б) Двум
В) Трём и более
Г) Любому целому числу

3. **Метод последовательного перебора вершин многогранника допустимых решений называется:**

А) Методом Ньютона
Б) Методом Гаусса
В) Методом Монте-Карло
Г) Симплекс-методом

4. **Процесс, в котором будущее состояние системы зависит только от текущего состояния и не зависит от предыстории, называется:**

А) Детерминированным
Б) Марковским
В) Циклическим
Г) Стохастическим с памятью

5. **Алгоритм Дейкстры предназначен для:**

А) Построения минимального остовного дерева
Б) Топологической сортировки
В) Нахождения кратчайших путей от одной вершины до всех остальных в графе с неотрицательными весами
Г) Раскраски графа

6. **Игра, в которой сумма выигрышей всех участников постоянна (обычно равна нулю), называется:**

А) Кооперативной
Б) Игрой с нулевой суммой
В) Многошаговой
Г) Статической

7. **Основной принцип динамического программирования, позволяющий разбивать сложную задачу на оптимальные подзадачи, сформулировал:**

А) Р. Беллман
Б) Дж. фон Нейман
В) Л. Канторович

Г) Дж. Данциг

8. **Если коэффициент загрузки канала СМО $\rho = \lambda/\mu > 1$, то:**

А) Система работает стабильно

Б) Очередь будет неограниченно расти

В) Канал будет большую часть времени простаивать

Г) Заявки будут обслуживаться мгновенно

9. **Метод, основанный на многократной генерации случайных чисел для оценки характеристик сложной системы, называется:**

А) Методом наименьших квадратов

Б) Методом потенциалов

В) Градиентным методом

Г) Методом Монте-Карло

10. **В канонической форме задачи линейного программирования все ограничения должны быть записаны в виде:**

А) Неравенств типа $\leq$

Б) Равенств с неотрицательными переменными

В) Модульных неравенств

Г) Произвольных функций

11. **Граф, в котором между любыми двумя вершинами существует ровно один простой путь, называется:**

А) Циклом

Б) Деревом

В) Полным графом

Г) Ориентированным графом

12. **Стратегия, гарантирующая игроку выигрыш не меньше определённого значения независимо от действий противника, называется:**

А) Минимаксной стратегией

Б) Равновесием Нэша

В) Доминирующей стратегией

Г) Смешанной стратегией

13. **Имитационное моделирование наиболее эффективно применяется, когда:**

А) Существует точная аналитическая формула

Б) Система слишком сложна или стохастична для аналитического описания

В) Требуется только статический расчёт

Г) Отсутствуют любые данные о системе

14. **Схема «гибели и размножения» описывает процессы с дискретными состояниями, в которых переходы возможны:**

А) В любое произвольное состояние

Б) Только в соседние состояния (+1 или -1)

В) Только в начальное состояние

Г) Только при наличии внешнего управления

15. **Целочисленное программирование отличается от обычного линейного тем, что:**

А) Переменные могут принимать только целочисленные значения

Б) Целевая функция обязательно квадратичная

В) Ограничения отсутствуют

Г) Решается исключительно графическим методом

№	Ответ	№	Ответ
1	Б	11	Б
2	Б	12	А
3	Г	13	Б
4	Б	14	Б

5	В	15	А
6	Б		
7	А		
8	Б		
9	Г		
10	Б		

Тип 2: Несколько правильных ответов (выбор из 5 вариантов)

16. **Признаки, по которым классифицируются математические модели:**

А) Характер отражаемых свойств (структурные, функциональные)
Б) Способ представления (аналитические, имитационные, графические)
В) Характер изменения во времени (статические, динамические)
Г) Цвет оформления чертежей
Д) Язык программирования реализации

17. **К характерным чертам транспортной задачи относятся:**

А) Линейная целевая функция (минимизация затрат)
Б) Ограничения в виде балансовых уравнений по поставщикам и потребителям
В) Переменные означают объёмы перевозок между пунктами
Г) Наличие нелинейных штрафов за простой
Д) Обязательное использование целочисленных переменных

18. **Особенности решений задач нелинейного программирования:**

А) Оптимум может находиться внутри области допустимых решений
Б) Может существовать несколько локальных оптимумов
В) Всегда гарантируется единственность решения
Г) Целевая функция или ограничения нелинейны
Д) Графический метод применим для любого числа переменных

19. **Основные элементы системы массового обслуживания (СМО):**

А) Входной поток заявок
Б) Очередь (буфер ожидания)
В) Каналы обслуживания
Г) Выходной поток обслуженных/отказавших заявок
Д) Генератор случайных паролей

20. **Элементы матричной игры с нулевой суммой:**

А) Два игрока
Б) Множество стратегий для каждого игрока
В) Платёжная матрица
Г) Вероятности состояний природы
Д) Дерево решений

21. **Преимущества имитационного моделирования:**

А) Возможность экспериментировать без риска для реального объекта
Б) Учёт случайных факторов, нелинейностей и ограничений ресурсов
В) Автоматическая генерация точного аналитического решения
Г) Наглядность динамики процессов во времени
Д) Отсутствие необходимости в статистической обработке результатов

22. **Задачи, решаемые методами динамического программирования:**

А) Задача о рюкзаке
Б) Управление запасами во времени
В) Поиск оптимального маршрута в многоступенчатой сети
Г) Решение системы линейных уравнений методом Гаусса
Д) Построение графика функции

№	Правильные варианты

16	А, Б, В
17	А, Б, В *(Д также часто верно, но А,Б,В — базовые признаки)*
18	А, Б, Г
16	А, Б, В, Г
20	А, Б, В
21	А, Б, Г
22	А, Б, В

Тип 3: Установление соответствия

23. **Соотнесите тип модели и её описание:**

1. Натурная модель А) Описание на естественном языке, схемах, алгоритмах
2. Информационная модель Б) Физический макет или уменьшенная копия объекта
3. Математическая модель В) Система уравнений, формул, неравенств

24. **Соотнесите термин линейного программирования и определение:**

1. Целевая функция А) Множество точек, удовлетворяющих всем
2. Область допустимых решений (ОДР) ограничениям Б) Решение, соответствующее вершине ОДР
3. Опорное решение В) Показатель эффективности, требующий максимизации или минимизации

1. **Соотнесите параметр сетевого графика и его смысл:**

1. Ранний срок начала работы А) Предельное время завершения работы без срыва
2. Поздний срок окончания работы общего срока проекта
3. Полный резерв времени Б) Наименьшее возможное время, когда работа может начаться

В) Максимальное время, на которое можно задержать работу без влияния на критический путь

1. **Соотнесите термин теории игр и описание:**

1. Равновесие Нэша А) Выбор одного конкретного действия без рандомизации
2. Чистая стратегия Б) Распределение вероятностей на множестве чистых стратегий
3. Смешанная стратегия В) Ситуация, где ни один игрок не может увеличить свой выигрыш, односторонне меняя стратегию

1. **Соотнесите метод оптимизации и область применения:**

1. Симплекс-метод А) Задачи нелинейного программирования
2. Градиентный спуск Б) Задачи целочисленного программирования
3. Метод ветвей и границ В) Задачи линейного программирования

1. **Соотнесите вид СМО и его особенность:**

1. СМО с отказами А) Заявка уходит, если все каналы заняты
2. СМО с неограниченной очередью Б) Заявка встаёт в очередь и ждёт освобождения канала
3. СМО с ограниченной очередью В) Очередь имеет максимальную длину, при превышении заявки теряются

№	Ответ
23	1-Б, 2-А, 3-В
24	1-В, 2-А, 3-Б
25	1-Б, 2-А, 3-В
26	1-А, 2-Б, 3-В
27	1-В, 2-А, 3-Б
28	1-В, 2-А, 3-Б

Тип 4: Верно / Неверно

29. **Адекватность модели означает её полное физическое и математическое тождество с оригиналом.**

○ Верно

○ Неверно

30. **В задаче о назначениях каждый исполнитель может выполнять более одной работы одновременно.**

○ Верно

○ Неверно

31. **Для решения задачи нелинейного программирования всегда применим графический метод.**

○ Верно

○ Неверно

32. **В динамическом программировании оптимальное управление на текущем шаге зависит только от текущего состояния системы, а не от истории переходов.**

○ Верно

○ Неверно

33. **Критический путь в сетевом графике имеет минимальную продолжительность среди всех путей.**

○ Верно

○ Неверно

34. **В биматричной игре выигрыши игроков задаются двумя матрицами, так как игра не обязательно с нулевой суммой.**

○ Верно

○ Неверно

35. **Результаты имитационного эксперимента требуют статистической обработки для оценки достоверности и погрешности.**

○ Верно

○ Неверно

№	Ответ
29	**Неверно** *(Адекватность означает соответствие модели целям исследования и отражение существенных свойств)*
30	**Неверно** *(Каждый исполнитель назначается ровно на одну работу)*
31	**Неверно** *(Только при двух переменных)*
32	**Верно**
33	**Неверно** *(Имеет максимальную продолжительность)*
34	**Верно**
35	**Верно**

Тип 5: Заполнение пропуска / Краткий ответ

36. **Метод _____ применяется для решения задач целочисленного линейного программирования путём последовательного разбиения области допустимых решений.**

37. **Условие, при котором любой локальный минимум выпуклой функции совпадает с глобальным, называется свойством _____.**

38. **СМО, в которой заявка получает отказ при занятости всех каналов обслуживания, называется системой с _____.**

39. **Принцип, согласно которому игроки действуют рационально, стремясь максимизировать свой ожидаемый выигрыш, называется принципом _____.**

40. **Дерево _____ используется для визуализации последовательных решений, вероятностных исходов и расчёта ожидаемой полезности.**

41. **Интенсивность потока заявок в теории массового обслуживания традиционно обозначается греческой буквой _____.**

42. **Процесс проверки соответствия имитационной модели концептуальному описанию системы называется _____.**

№	Ответ
36	**ветвей и границ**
37	**выпуклости**
38	**отказами**
39	**рационального поведения**
40	**решений**
41	**λ (лямбда)**
42	**Верификацией**

Тип 6: Установление последовательности

43. **Этапы математического моделирования:**

1. Построение математической модели
2. Формулировка цели и постановка задачи
3. Анализ результатов и корректировка модели
4. Проведение вычислительных экспериментов

44. **Порядок решения задачи динамического программирования:**

1. Условная оптимизация (расчёт с конца к началу)
2. Безусловная оптимизация (восстановление оптимальной траектории с начала к концу)
3. Определение шагов и состояний системы
4. Запись рекуррентных уравнений Беллмана

1. **Этапы имитационного моделирования:**

1. Верификация и валидация модели
2. Постановка задачи и сбор данных
3. Проведение серий экспериментов
4. Разработка алгоритма и программирование
5. Анализ результатов и выработка рекомендаций

1. **Последовательность расчёта параметров сетевого графика:**

1. Расчёт ранних сроков начала и окончания работ (прямой ход)
2. Определение критического пути
3. Расчёт поздних сроков начала и окончания работ (обратный ход)
4. Вычисление резервов времени

№	Правильный порядок
43	2 → 1 → 4 → 3
44	3 → 4 → 1 → 2
45	2 → 4 → 1 → 3 → 5
46	1 → 3 → 4 → 2

Тип 7: Ситуационные задачи / Анализ

47. **При решении задачи ЛП графическим методом область допустимых решений оказалась неограниченной, а целевая функция стремится к бесконечности в направлении роста. Какой вывод сделает специалист?**

А) Задача имеет единственное оптимальное решение

Б) ОДР пуста, задача неразрешима

В) Целевая функция не ограничена на ОДР, оптимального решения нет

Г) Решение лежит в начале координат

48. **В СМО интенсивность входящего потока λ = 4 заявки/час, интенсивность обслуживания μ = 2 заявки/час. Каков коэффициент загрузки ρ и каково состояние системы?**

А) ρ = 0.5, система стабильна

Б) ρ = 2, очередь будет расти неограниченно

В) ρ = 1, система работает на пределе

Г) ρ = 6, заявки обслуживаются мгновенно

49. **При тестировании логистической компании выяснилось, что время доставки зависит от случайных факторов: пробок, погоды, загруженности складов. Аналитическая модель слишком сложна. Какой инструмент исследования операций наиболее целесообразно применить?**

А) Симплекс-метод

Б) Имитационное моделирование (метод Монте-Карло или дискретно-событийное)

В) Теорию игр с нулевой суммой

Г) Графический метод ЛП

50. **В сетевом графике проекта найден путь продолжительностью 45 дней. Все остальные пути имеют длину ≤ 40 дней. Что означает этот путь для проекта?**

А) Это критический путь, определяющий минимальный срок выполнения проекта

Б) Это резервный путь, на который можно перенести ресурсы

В) Это путь с максимальным резервом времени

Г) Это фиктивная работа, не влияющая на сроки

№	Ответ
47	В
48	Б *($\rho = \lambda/\mu = 4/2 = 2 > 1$)*
49	Б
50	А

по МДК.02.05 Численные методы

Тип 1: Один правильный ответ (выбор из 4 вариантов)

1. **Какое представление чисел в памяти компьютера позволяет работать с очень большими и очень малыми величинами с заданной точностью?**

А) Фиксированная запятая

Б) Плавающая запятая

В) Двоично-десятичное кодирование

Г) Целочисленное представление

2. **Относительная погрешность приближённого числа вычисляется по формуле:**

А) |X_прибл – X_точн| / |X_точн|

Б) |X_точн – X_прибл| × 100

В) X_прибл / X_точн

Г) |X_прибл + X_точн| / 2

3. **Цифра числа называется значащей, если:**

А) Она стоит после запятой

Б) Она не равна нулю или ноль стоит между ненулевыми цифрами либо после запятой в дробной части

В) Она равна нулю

Г) Она является первой цифрой числа

4. **Метод половинного деления гарантированно находит корень уравнения f(x)=0, если:**

А) Функция непрерывна на отрезке [a,b] и $f(a)\cdot f(b) < 0$

Б) Производная функции положительна

В) Отрезок [a,b] содержит более одного корня

Г) Функция является многочленом

5. **Метод касательных (Ньютона) для нахождения корня уравнения требует вычисления:**

А) Только значения функции

Б) Значения функции и её первой производной

В) Значения функции и второй производной

Г) Интеграла функции

6. **Прямой ход метода Гаусса при решении СЛАУ предназначен для:**
А) Приведения расширенной матрицы к треугольному виду
Б) Нахождения обратной матрицы
В) Вычисления определителя методом разложения
Г) Проверки совместности системы
7. **Отличие метода Зейделя от метода простой итерации для систем линейных уравнений заключается в:**
А) Использовании уже вычисленных на текущем шаге значений неизвестных
Б) Отсутствии условия сходимости
В) Применении только к симметричным матрицам
Г) Использовании матрицы Гессе
8. **Интерполяционный многочлен Лагранжа используется для:**
А) Аппроксимации функции, заданной таблично, в узлах и между ними
Б) Решения дифференциальных уравнений
В) Численного интегрирования
Г) Минимизации нелинейных функций
9. **Кубический сплайн на каждом отрезке между узлами представляет собой:**
А) Линейную функцию
Б) Квадратичный полином
В) Кубический полином
Г) Экспоненциальную функцию
10. **Экстраполяция в отличие от интерполяции предполагает:**
А) Восстановление значений внутри интервала известных узлов
Б) Оценку значений функции за пределами интервала узлов
В) Сглаживание шумов в данных
Г) Нахождение производных в узлах
11. **Квадратурная формула Ньютона-Котеса при n=2 называется:**
А) Правило прямоугольников
Б) Правило трапеций
В) Формула Симпсона (парабол)
Г) Формула Гаусса
12. **Формула численного интегрирования Гаусса отличается от формул Ньютона-Котеса тем, что:**
А) Узлы интегрирования выбираются оптимально (корни ортогональных полиномов), а не равноотстоящими
Б) Использует только концы отрезка
В) Применима только к периодическим функциям
Г) Требует вычисления производных подынтегральной функции
13. **Метод Эйлера для решения задачи Коши y' = f(x,y) имеет порядок точности:**
А) 1
Б) 2
В) 4
Г) 0
14. **Метод Рунге-Кутта 4-го порядка на каждом шаге требует вычисления правой части ОДУ:**
А) 1 раз
Б) 2 раза
В) 4 раза
Г) 8 раз
15. **Метод дихотомии поиска экстремума функции одной переменной на каждом шаге:**
А) Делит отрезок неопределённости пополам и отбрасывает половину
Б) Строит касательную

В) Использует вторую производную
Г) Применяет градиентный спуск

№	Ответ	№	Ответ
1	Б	11	В
2	А	12	А
3	Б	13	А
4	А	14	В
5	Б	15	А
6	А		
7	А		
8	А		
9	В		
10	Б		

Тип 2: Несколько правильных ответов (выбор из 5 вариантов)

16. **Источники погрешностей в численных вычислениях включают:**

А) Погрешность исходных данных
Б) Погрешность метода (усечения)
В) Погрешность округления
Г) Погрешность шрифтового отображения
Д) Погрешность компиляции программы **А, Б, В**

17. **При умножении и делении приближённых чисел:**

А) Абсолютные погрешности складываются
Б) Относительные погрешности складываются
В) Результат округляется до количества значащих цифр наименее точного множителя
Г) Погрешность всегда уменьшается
Д) Используется только правило трапеций **Б, В**

18. **Преимущества комбинированного метода хорд и касательных:**

А) Ускоренная сходимость по сравнению с отдельными методами
Б) Возможность получения двусторонней оценки корня
В) Не требует вычисления производной
Г) Применим только к линейным функциям
Д) Позволяет контролировать погрешность на каждом шаге **А, Б, Д**

19. **Условие сходимости метода простой итерации для системы СЛАУ выполняется, если:**

А) Матрица системы имеет строгое диагональное преобладание
Б) Норма матрицы перехода меньше единицы
В) Все элементы матрицы положительны
Г) Определитель равен нулю
Д) Система имеет единственное решение **А, Б**

20. **Интерполяционные формулы Ньютона удобны тем, что:**

А) При добавлении нового узла не требуется пересчитывать все коэффициенты
Б) Используют конечные разности, которые легко вычисляются
В) Гарантируют минимум осцилляций
Г) Применимы только для равноотстоящих узлов
Д) Требуют решения СЛАУ для нахождения коэффициентов **А, Б, Г**

21. **Преимущества сплайн-интерполяции перед глобальным полиномом Лагранжа:**

А) Отсутствие эффекта Рунге при большом числе узлов
Б) Локальное изменение коэффициентов при изменении одного узла
В) Обеспечение гладкости (непрерывности производных)
Г) Автоматическая экстраполяция за пределы узлов
Д) Простота вычисления производных в узлах **А, Б, В**

22. **К методам численного минимизации функции двух переменных без вычисления производных относятся:**

А) Метод покоординатного спуска
Б) Симплекс-метод Нелдера-Мида
В) Метод золотого сечения
Г) Метод наискорейшего спуска
Д) Метод Ньютона для оптимизации **А, Б**

23. **При применении метода Гаусса для нахождения обратной матрицы:**

А) Справа от исходной матрицы записывается единичная матрица
Б) Прямой и обратный ход применяются к расширенной матрице
В) В результате слева получается единичная матрица, а справа – обратная
Г) Требуется вычисление определителя на каждом шаге
Д) Метод применим только к вырожденным матрицам **А, Б, В**

№	Правильные варианты
16	А, Б, В
17	Б, В
18	А, Б, Д
16	А, Б
20	А, Б, Г
21	А, Б, В
22	А, Б
23	А, Б, В

Тип 3: Установление соответствия

24. **Соотнесите вид погрешности и её определение:**

1. Абсолютная погрешность	А) Отношение абсолютной погрешности к модулю точного значения
2. Относительная погрешность	Б) Модуль разности приближённого и точного значений
3. Предельная погрешность	В) Верхняя граница возможной погрешности, гарантирующая точность

1-Б, 2-А, 3-В

25. **Соотнесите метод решения нелинейных уравнений и его геометрическую интерпретацию:**

1. Метод хорд	А) Проведение касательной к графику в текущей точке до пересечения с осью OX
2. Метод касательных	Б) Проведение секущей между концами отрезка до пересечения с OX
3. Метод простой итерации	В) Построение итерационной последовательности через пересечение $y=x$ и $y=\varphi(x)$

1-Б, 2-А, 3-В

1. **Соотнесите метод численного интегрирования и порядок точности:**

1. Метод прямоугольников	А) $O(h^2)$
2. Метод трапеций	Б) $O(h^4)$
3. Формула Симпсона	В) $O(h)$

1-В, 2-А, 3-Б

1. **Соотнесите алгоритм минимизации и его ключевую особенность:**

1. Метод золотого сечения	А) Движение поочерёдно вдоль осей координат
2. Метод наискорейшего спуска	Б) Использование отношения отрезков, равного золотому сечению (~0.618)
3. Метод покоординатного спуска	В) Движение в направлении антиградиента функции

1-Б, 2-В, 3-А *(Примечание: в задании нумерация сдвинута, исправлено на 1-Золотое сечение, 2-Наискорейший, 3-Покоординатный)*

1. **Соотнесите понятие и математическое выражение/свойство:**

1.	Интерполяция	А) Приближение функции на заданном отрезке с минимизацией погрешности (без обязательного прохождения через узлы)
2.	Экстраполяция	Б) Восстановление функции внутри интервала известных узлов
3.	Аппроксимация	В) Определение значений функции за пределами интервала узлов

1-Б, 2-В, 3-А

1. **Соотнесите этап метода Гаусса и его действие:**

1.	Прямой ход	А) Последовательное исключение переменных, приведение к треугольному виду
2.	Обратный ход	Б) Нахождение значений неизвестных начиная с последнего уравнения

1-А, 2-Б

1. **Соотнесите условие и метод отделения корней:**

1.	$f(a)\cdot f(b) < 0$	А) Гарантирует монотонность функции на отрезке
2.	$f'(x)$ сохраняет знак на $[a,b]$	Б) Гарантирует выпуклость/вогнутость графика
3.	$f''(x)$ сохраняет знак на $[a,b]$	В) Указывает на наличие хотя бы одного корня на отрезке

1-В, 2-А, 3-Б

1. **Соотнесите свойство сплайна и его математическую формулировку:**

1.	Непрерывность $S(x)$	А) Гладкость стыка без изломов
2.	Непрерывность $S'(x)$	Б) Плавность изменения кривизны
3.	Непрерывность $S''(x)$	В) Отсутствие разрывов в узловых точках

1-В, 2-А, 3-Б

№	Ответ
24	**1-Б, 2-А, 3-В**
25	**1-Б, 2-А, 3-В**
26	**1-В, 2-А, 3-Б**
27	**1-Б, 2-В, 3-А** *(Примечание: в задании нумерация сдвинута, исправлено на 1-Золотое сечение, 2-Наискорейший, 3-Покоординатный)*
28	**1-Б, 2-В, 3-А**
29	**1-А, 2-Б**
30	**1-В, 2-А, 3-Б**
31	**1-В, 2-А, 3-Б**

Тип 4: Верно / Неверно

32. **Сомнительной считается цифра, если абсолютная погрешность числа превышает половину разряда, в котором она стоит.**

○ Верно
○ Неверно

33. **При сложении приближённых чисел абсолютная погрешность суммы равна сумме абсолютных погрешностей слагаемых.**

○ Верно
○ Неверно

34. **Метод простой итерации сходится быстрее метода Зейделя при одинаковых начальных приближениях.**

○ Верно
○ Неверно

35. **Интерполяционный многочлен Лагранжа степени n проходит через все n+1 узловые точки.**

○ Верно

○ Неверно

36. **Экстраполяция, как правило, даёт меньшую погрешность, чем интерполяция.**

○ Верно

○ Неверно

37. **Метод Гаусса применим только к системам с ненулевым определителем матрицы коэффициентов.**

○ Верно

○ Неверно

38. **Формула Гаусса для численного интегрирования использует равноотстоящие узлы на отрезке.**

○ Верно

○ Неверно

№	Ответ
32	**Верно**
33	**Верно**
34	**Неверно** *(Метод Зейделя обычно сходится быстрее, так как использует обновлённые значения)*
35	**Верно**
36	**Неверно** *(Экстраполяция менее надёжна и погрешность быстро растёт за пределами узлов)*
37	**Верно**
38	**Неверно** *(Узлы выбираются неравномерно, как корни полиномов Лежандра)*

Тип 5: Заполнение пропуска / Краткий ответ

39. **Скорость сходимости итерационных методов оценивается по величине нормы матрицы _____ или по разности последовательных приближений.**

40. **Если при решении СЛАУ методом Гаусса на каком-то шаге все элементы столбца ниже главного элемента равны нулю, а правая часть не ноль, то система _____.**

41. **Метод Рунге-Кутта более высокого порядка требует меньшего шага интегрирования для достижения той же точности по сравнению с методом Эйлера.**

42. **В методе золотого сечения отношение длин отрезков на каждом шаге сохраняется постоянным и равно _____.**

43. **Процесс нахождения промежутка, на котором уравнение имеет ровно один корень, называется _____ корней.**

№	Ответ
39	**перехода (или итерационной матрицы)**
40	**несовместна (не имеет решений)**
41	**Неверно** *(Наоборот: позволяет брать больший шаг при той же точности)*
42	**$\tau \approx 0.618$ (или $(\sqrt{5}-1)/2$)**
43	**Отделением**

Тип 6: Установление последовательности

44. **Этапы решения нелинейного уравнения f(x)=0 численными методами:**

1. Уточнение корня итерационным методом
2. Отделение корней (локализация)
3. Проверка критерия остановки (|x_{n+1} - x_n| < ε)
4. Выбор начального приближения

45. **Порядок вычислений при использовании уточнённого метода Эйлера:**

1. Вычисление прогнозного значения y* = y_n + h·f(x_n, y_n)
2. Вычисление производной в начальной точке f(x_n, y_n)

3. Вычисление производной в прогнозной точке f(x_{n+1}, y*)
4. Уточнение значения y_{n+1} = y_n + (h/2)·[f(x_n, y_n) + f(x_{n+1}, y*)]

1. **Шаги метода покоординатного спуска для минимизации f(x,y):**
1. Фиксация y, минимизация по x → x_{k+1}
2. Проверка условия остановки
3. Фиксация x_{k+1}, минимизация по y → y_{k+1}
4. Переход к следующей итерации

№	Правильный порядок
44	2 → 4 → 1 → 3
45	2 → 1 → 3 → 4
46	1 → 3 → 2 → 4 *(или 1-3-2, если проверка в конце цикла)*

Тип 7: Ситуационные задачи / Анализ

47. **При численном интегрировании функции f(x)=e^x на [0,1] методом трапеций с шагом h=0.5 получена погрешность ε. Чтобы уменьшить погрешность в 4 раза, шаг h необходимо:**
А) Увеличить в 2 раза
Б) Уменьшить в 2 раза
В) Уменьшить в 4 раза
Г) Оставить без изменений

48. **Студент применяет метод Ньютона к уравнению $x^3 - x - 1 = 0$ с начальным приближением $x_0 = 0.5$. График функции в этой точке имеет точку перегиба, а производная близка к нулю. Какой риск возникает?**
А) Метод гарантированно найдёт корень за 1 итерацию
Б) Погрешность округления станет нулевой
В) Последующая итерация может «улететь» далеко от корня или метод не сойдётся
Г) Метод автоматически переключится на метод хорд

№	Ответ
47	Б *(Погрешность метода трапеций пропорциональна h^2, значит ε/4 достигается при h/2)*
48	В

по МДК.02.06 Безопасность программного обеспечения

Тип 1: Один правильный ответ (выбор из 4 вариантов)

1. **Что такое уязвимость программного обеспечения?**
А) Ошибка в документации
Б) Слабое место в системе, которое может быть использовано злоумышленником для нарушения политики безопасности
В) Временная задержка в работе сервера
Г) Лицензионное ограничение продукта

2. **Какая уязвимость стабильно занимает первое место в актуальном списке OWASP Top 10?**
А) Injection
Б) Cryptographic Failures
В) Broken Access Control (Нарушение контроля доступа)
Г) Security Misconfiguration

3. **Принцип наименьших привилегий (Least Privilege) означает:**
А) Предоставление пользователю максимальных прав для удобства работы
Б) Выдача ровно тех прав, которые необходимы для выполнения конкретной задачи
В) Отключение всех учётных записей по умолчанию
Г) Использование одного пароля для всех систем

4. **Какой тип шифрования использует один и тот же ключ для шифрования и расшифрования?**

А) Асимметричное
Б) Одностороннее
В) Симметричное
Г) Квантовое

5. **Протокол TLS обеспечивает:**

А) Только сжатие передаваемых данных
Б) Конфиденциальность, целостность и аутентификацию канала связи
В) Автоматическое резервное копирование
Г) Балансировку нагрузки между серверами

6. **Где безопаснее всего хранить криптографические ключи в мобильном приложении на Android?**

А) В файле assets проекта
Б) В строковых ресурсах (strings.xml)
В) В Android Keystore System (аппаратно или ОС-изолированно)
Г) В зашифрованном виде в базе данных SQLite

7. **Какой HTTP-заголовок защищает веб-приложение от XSS-атак, ограничивая допустимые источники скриптов?**

А) Cache-Control
Б) Content-Security-Policy (CSP)
В) Access-Control-Allow-Origin
Г) X-Frame-Options

8. **Что такое «Envelope Encryption» (конвертное шифрование) в облачных средах?**

А) Шифрование всего жёсткого диска сервера
Б) Шифрование данных ключом данных (DEK), который сам зашифрован главным ключом (KEK)
В) Архивирование данных перед отправкой в облако
Г) Использование одного ключа для всех клиентов

9. **SQL-инъекция возникает преимущественно из-за:**

А) Неправильной обработки пользовательского ввода и конкатенации строк в SQL-запросах
Б) Медленного сетевого соединения
В) Отсутствия HTTPS
Г) Использования IPv6

10. **Какой алгоритм хеширования рекомендован современными стандартами для безопасного хранения паролей?**

А) MD5
Б) SHA-1
В) bcrypt / Argon2 / PBKDF2
Г) Base64

11. **Для чего предназначен протокол OAuth 2.0?**

А) Для шифрования электронной почты
Б) Для делегированного доступа к ресурсам без передачи учётных данных пользователя
В) Для защиты от DDoS-атак
Г) Для сжатия JSON-ответов

12. **Принцип Security by Design предполагает:**

А) Внедрение средств защиты после завершения разработки и тестирования
Б) Интеграцию мер безопасности на этапах проектирования архитектуры и разработки
В) Покупку готового антивируса
Г) Отказ от использования сторонних библиотек

13. **Что такое Nonce в криптографических протоколах?**

А) Одноразовое случайное число, предотвращающее атаки повторного воспроизведения (replay)
Б) Постоянный секретный ключ

В) Хеш-сумма файла
Г) Идентификатор сессии

14. **Флаг куки HttpOnly защищает от:**

А) Перехвата по незашифрованному каналу
Б) Доступа к куки через клиентский JavaScript (снижает риск кражи сессии при XSS)
В) Отправки куки на сторонние домены
Г) Автоматического удаления куки браузером

15. **Модель угроз (Threat Model) в первую очередь помогает:**

А) Ускорить компиляцию кода
Б) Систематизировать активы, выявить потенциальные угрозы и определить приоритеты защиты
В) Написать документацию к API
Г) Оптимизировать запросы к базе данных

№	Ответ	№	Ответ
1	Б	11	Б
2	В	12	Б
3	Б	13	А
4	В	14	Б
5	Б	15	Б
6	В		
7	А		
8	Б		
9	А		
10	В		

Тип 2: Несколько правильных ответов (выбор из 5 вариантов)

16. **Выберите компоненты классического анализа рисков:**

А) Идентификация активов
Б) Оценка угроз и уязвимостей
В) Расчёт вероятности и потенциального ущерба
Г) Выбор стратегии обработки риска
Д) Увеличение бюджета на маркетинг

17. **Какие меры эффективно защищают от SQL-инъекций?**

А) Параметризованные запросы / Prepared Statements
Б) Использование ORM-фреймворков
В) Валидация и санитизация входных данных
Г) Принцип наименьших привилегий для БД-пользователя
Д) Отключение логирования ошибок

18. **Факторы аутентификации в многофакторной защите (MFA) включают:**

А) Знание (пароль, PIN)
Б) Владение (смартфон, аппаратный токен)
В) Биометрия (отпечаток, лицо)
Г) Местоположение пользователя
Д) Версия браузера

19. **Свойства криптографически стойкой хеш-функции:**

А) Необратимость (прообраз невосстановим)
Б) Устойчивость к коллизиям
В) Детерминированность (одинаковый ввод → одинаковый вывод)
Г) Лавинный эффект (малое изменение входа кардинально меняет хеш)
Д) Возможность расшифровки

20. **Принципы безопасной архитектуры программного обеспечения:**

А) Глубокая защита (Defense in Depth)

Б) Минимальная поверхность атаки
В) Разделение обязанностей (Separation of Duties)
Г) Отказ в безопасном состоянии (Fail Secure)
Д) Скрытие всех ошибок от логов

21. **Уязвимости OWASP Top 10, связанные с обработкой данных и криптографией:**
А) Cryptographic Failures
Б) Insecure Design
В) Security Misconfiguration
Г) Vulnerable and Outdated Components
Д) Unvalidated Redirects

22. **Типичные риски криптографии в мобильных приложениях:**
А) Хардкодинг ключей в исходном коде
Б) Использование устаревших алгоритмов (DES, MD5, RC4)
В) Отсутствие проверки целостности кода (tampering)
Г) Хранение чувствительных данных в незашифрованном виде в логах
Д) Обязательное использование квантового шифрования

23. **Механизмы защиты сессий в веб-приложениях:**
А) Флаги Secure, HttpOnly, SameSite для cookies
Б) Ротация идентификатора сессии после аутентификации
В) Жёсткие таймауты неактивности
Г) Привязка сессии к User-Agent и IP (с осторожностью)
Д) Хранение токенов в localStorage без защиты

24. **Преимущества аппаратных модулей безопасности (HSM) в облаке:**
А) Физическая и логическая изоляция ключей от ОС и администраторов
Б) Сертификация по стандартам (FIPS 140-2/3, Common Criteria)
В) Аппаратное ускорение криптографических операций
Г) Автоматическое удаление данных при попытке взлома корпуса
Д) Бесплатное использование без ограничений

25. **Модели аутентификации в современных микросервисных приложениях:**
А) JWT (JSON Web Tokens)
Б) OAuth 2.0 / OpenID Connect
В) mTLS (взаимный TLS)
Г) Basic Auth по HTTP
Д) API-ключи в URL

№	Правильные варианты
16	А, Б, В, Г
17	А, Б, В, Г
18	А, Б, В
16	А, Б, В, Г
20	А, Б, В, Г
21	А, Б, В, Г
22	А, Б, В, Г
23	А, Б, В, Г
24	А, Б, В, Г
25	А, Б, В *(Г и Д устарели/небезопасны для production)*

Тип 3: Установление соответствия

26. **Соотнесите угрозу и нарушаемый принцип CIA:**

1. Утечка базы данных клиентов	А) Конфиденциальность
2. Изменение суммы транзакции злоумышленником	Б) Целостность
3. DDoS-атака на веб-сервер	В) Доступность

27. **Соотнесите криптографический алгоритм и его основное назначение:**

1.	AES-256	А) Цифровая подпись и обмен ключами
2.	RSA-2048 / ECDSA	Б) Хеширование и проверка целостности
3.	SHA-256	В) Быстрое симметричное шифрование данных Г) Безопасное
4.	bcrypt	хеширование паролей

1. **Соотнесите категорию OWASP Top 10 и пример уязвимости:**

1.	Broken Access Control	А) Изменение user_id в URL для доступа к чужому профилю
2.	Injection	(IDOR)
3.	Security Misconfiguration	Б) Включение режима отладки в production-окружении
		В) Внедрение вредоносного JavaScript через поле комментария (XSS)

1. **Соотнесите протокол и уровень сетевой модели (TCP/IP):**

1.	IPsec	А) Прикладной уровень
2.	TLS/SSL	Б) Сетевой уровень
3.	S/MIME / OpenPGP	В) Транспортный уровень

1. **Соотнесите тип атаки и механизм защиты:**

1.	Brute Force (перебор паролей)	А) Anti-CSRF токены, SameSite cookies
2.	Session Hijacking	Б) Rate Limiting, Account Lockout, CAPTCHA
3.	Cross-Site Request Forgery (CSRF)	В) HttpOnly/Secure flags, ротация сессии, привязка к контексту

1. **Соотнесите принцип Zero Trust и его реализацию:**

1.	Не доверяй, всегда проверяй	А) Строгая проверка каждого запроса независимо от сети
2.	Минимальный доступ	Б) Изоляция рабочих нагрузок друг от друга на уровне
3.	Микросегментация	сети
		В) Предоставление прав только на время выполнения задачи (JIT)

1. **Соотнесите мобильную платформу и защищённое хранилище:**

1.	iOS	А) Keychain Services / Secure Enclave
2.	Android	Б) Android Keystore System / StrongBox
3.	Кроссплатформенные	В) Абстракции (например, react-native-keychain,
фреймворки		flutter_secure_storage)

1. **Соотнесите вид шифрования данных и состояние данных:**

1.	Encryption at Rest	А) Защита данных при передаче по сети
2.	Encryption in Transit	Б) Защита данных на дисках/в БД/в бэкапах
3.	Encryption in Use	В) Защита данных во время обработки в памяти (конфиденциальные вычисления)

№	Ответ
26	1-А, 2-Б, 3-В
27	1-В, 2-А, 3-Б, 4-Г
28	1-А, 2-В, 3-Б
29	1-Б, 2-В, 3-А
30	1-Б, 2-В, 3-А
31	1-А, 2-В, 3-Б
32	1-А, 2-Б, 3-В
33	1-Б, 2-А, 3-В

Тип 4: Верно / Неверно

34. **Уязвимость, активно эксплуатируемая злоумышленниками до выхода официального патча, называется «0-day».**

○ Верно

○ Неверно

35. **Хранение паролей в виде MD5-хешей без использования «соли» соответствует современным стандартам безопасности.**
○ Верно
○ Неверно
36. **Асимметричное шифрование работает быстрее симметричного и оптимально для шифрования больших объёмов данных.**
○ Верно
○ Неверно
37. **Принцип «Security through Obscurity» (безопасность через неясность) считается надёжной самостоятельной мерой защиты.**
○ Верно
○ Неверно
38. **Заголовок Strict-Transport-Security (HSTS) предписывает браузеру всегда использовать HTTPS для указанного домена.**
○ Верно
○ Неверно
39. **JWT-токены, хранящиеся в localStorage, автоматически защищены от кражи через XSS-атаки.**
○ Верно
○ Неверно
40. **Шифрование на стороне клиента (Client-Side Encryption) гарантирует, что облачный провайдер не имеет доступа к содержимому данных.**
○ Верно
○ Неверно

№	Ответ
34	**Верно**
35	**Неверно** *(MD5 криптографически сломан, отсутствие соли уязвимо к радужным таблицам)*
36	**Неверно** *(Оно значительно медленнее, используется для обмена ключами и цифровых подписей)*
37	**Неверно** *(Скрытие деталей не заменяет криптографию и контроль доступа)*
38	**Верно**
39	**Неверно** *(localStorage доступен через JS, токены уязвимы; безопаснее HttpOnly cookies)*
40	**Верно** *(При корректном управлении ключами клиентом)*

Тип 5: Заполнение пропуска / Краткий ответ

41. **Процесс добавления случайных данных к паролю перед хешированием для защиты от атак по радужным таблицам называется _____.**
42. **Протокол, позволяющий двум сторонам безопасно согласовать общий симметричный ключ по незащищённому каналу связи, называется _____.**
43. **Атака, при которой злоумышленник заставляет браузер авторизованного пользователя выполнить нежелательный запрос к уязвимому веб-приложению, называется _____.**
44. **Количественная оценка риска в классических моделях вычисляется по формуле: Риск = Вероятность × _____.**
45. **Флаг cookie _____ запрещает браузеру отправлять куки при переходе по незашифрованному**
46.

№	Ответ
41	Соль (Salting)
42	Диффи-Хеллман (Diffie-Hellman)
43	CSRF (Cross-Site Request Forgery)
44	Ущерб (Impact / Воздействие)
45	HTTP-соединению. Secure

Тип 6: Установление последовательности

46. **Этапы процесса управления рисками информационной безопасности:**

1. Идентификация активов и угроз
2. Оценка вероятности и потенциального ущерба
3. Выбор стратегии обработки (снижение, принятие, передача, избегание)
4. Внедрение мер контроля и мониторинг

47. **Порядок установления защищённого TLS-соединения (TLS Handshake):**

1. Согласование версий протокола и набора шифров
2. Обмен сертификатами и проверка подлинности сервера
3. Генерация и обмен предмастер-секретом (например, через DH)
4. Вывод сессионных ключей и начало зашифрованного обмена данными

1. **Жизненный цикл секрета (Secret Management) в безопасной разработке:**

1. Генерация с использованием криптографически стойкого ГСЧ
2. Безопасное хранение (Vault, HSM, Keystore)
3. Ротация и обновление по расписанию или при компрометации
4. Аудит использования и безопасное уничтожение

№	Правильный порядок
46	1 → 2 → 3 → 4
47	1 → 2 → 3 → 4
48	1 → 2 → 3 → 4

Тип 7: Ситуационные задачи / Анализ

49. **Разработчик встроил API-ключ для стороннего платежного сервиса прямо в клиентский JavaScript-файл SPA-приложения. Какая критическая угроза возникает и какое архитектурное решение её устраняет?**

А) Угроза: Утечка ключа через исходный код браузера. Решение: Вынести вызовы к API на серверную сторону (Backend-for-Frontend) или использовать прокси с ограничением по домену/IP.
Б) Угроза: Медленная загрузка страницы. Решение: Минификация JS.
В) Угроза: SQL-инъекция. Решение: Использование ORM.
Г) Угроза: Переполнение памяти. Решение: Увеличение RAM сервера. **А**

50. **Корпоративное приложение хранит персональные данные клиентов в облачной БД. Провайдер предлагает шифрование «в покое», но управляет ключами самостоятельно. Аудит требует гарантии конфиденциальности даже от сотрудников облака. Какое решение необходимо внедрить?**

А) Использовать модель BYOK/HYOK (Bring/Host Your Own Key) с клиентским шифрованием или доверенным KMS под полным контролем заказчика.
Б) Отказаться от шифрования для ускорения запросов.
В) Хранить пароли пользователей в открытом виде для удобства поддержки.
Г) Перевести приложение на виртуальную машину без сети. **А**

№	Ответ
47	А
48	А

Критерии оценки:

90-100 баллов «отлично» заслуживает студент, показавший всестороннее систематическое и глубокое знание учебно-программного материала, умение свободно выполнять задания, предусмотренные программой, усвоивший основную и знакомый с дополнительной литературой, рекомендованной программой; как правило, оценка «отлично» выставляется студентам, усвоившим взаимосвязь основных понятий междисциплинарного курса и их значение для приобретаемой профессии, проявившим творческие способности в понимании, изложении и использовании учебно-программного материала.

80-90 баллов «хорошо» заслуживает студент, обнаруживший полное знание учебно-программного материала, успешно выполняющий предусмотренные в программе задания, усвоивший основную литературу, рекомендованную в программе; как правило, оценка «хорошо» выставляется студентам, показавшим систематический характер знаний по дисциплине и способным к их самостоятельному пополнению и обновлению в ходе дальнейшей учебной работы и профессиональной деятельности.

60-80 баллов «удовлетворительно» заслуживает студент, обнаруживший знания основного учебно-программного материала в объеме, необходимом для дальнейшей учебы и предстоящей работы по профессии, справляющийся с выполнением заданий, предусмотренных программой, знакомый с основной литературой, рекомендованной программой; как правило, оценка «удовлетворительно» выставляется студентам, допустившим погрешности в ответе на зачете, но обладающим необходимыми знаниями для их устранения под руководством преподавателя.

Менее 60 баллов «неудовлетворительно» выставляется студенту, обнаружившему проблемы в знаниях основного учебно-программного материала, допустившему принципиальные ошибки в выполнении предусмотренных программой заданий; как правило, оценка «неудовлетворительно» ставится студентам, которые не могут продолжать обучение или приступить к профессиональной деятельности по окончании учебного заведения без дополнительных занятий по соответствующему междисциплинарному курсу.

Кейс-задачи

по МДК.02.01 Разработка программных модулей

Кейс 1. Система логирования и анализа событий

Контекст: Требуется разработать модульную систему сбора, парсинга и агрегации логов от нескольких микросервисов.

Требования:

- Реализовать модульную архитектуру с чётким разделением ответственности.
- Применить поведенческие паттерны (Observer для подписки на события, Strategy для выбора формата парсинга).
- Использовать регулярные выражения для извлечения полей (timestamp, level, service, message).
- Организовать запись в файлы с опциональным сжатием потоков.
- Провести оценку временной и пространственной сложности парсера.

Кейс 2. Асинхронный веб-парсер с дедупликацией

Контекст: Создание высокопроизводительного парсера, собирающего данные о товарах с нескольких источников одновременно.

Требования:

- Реализовать асинхронную модель на базе задач/async-await, использовать пул потоков.
- Реализовать механизмы отмены задач, продолжения и возврата агрегированных значений.
- Применить хеш-таблицу для устранения дубликатов с обработкой коллизий.
- Использовать очередь с приоритетами для лимитирования запросов к хостам.

Кейс 3. Модуль валидации и коррекции пользовательского ввода

Контекст: Разработка компонента проверки данных формы регистрации с автокоррекцией опечаток.

Требования:

- Валидация email, телефона, паролей через регулярные выражения.
- Реализация алгоритмов поиска подстрок (KMP или Бойер-Мура) для проверки чёрных списков.
- Расчёт редакционного расстояния (Левенштейна) для подсказок при опечатках.
- Интеграция с GUI: обработчики событий, уведомления об ошибках, валидация в реальном времени

Кейс 4. Десктопное приложение мониторинга в реальном времени

Контекст: Создание GUI-приложения для отображения метрик системы (CPU, RAM, сеть) с графиками и уведомлениями.

Требования:

- Архитектура MVVM, разделение логики и представления.
- Асинхронное обновление UI без блокировки основного потока (многопоточность в GUI).
- Построение графиков/диаграмм с помощью внешней библиотеки.
- Стилизация интерфейса: цветовая палитра, типографика, адаптивная компоновка.
- Окна настроек с валидацией ввода и передачей данных между окнами.

Кейс 5. Оптимизатор маршрутов доставки

Контекст: Модуль планирования маршрутов курьеров с учётом грузоподъёмности, приоритетов и дорожной сети.

Требования:

- Реализация алгоритма Дейкстры для поиска кратчайшего пути.
- Сравнение жадного подхода и динамического программирования на примере задачи о рюкзаке (выбор груза).
- Декомпозиция задачи на подмодули, создание спецификаций.
- Построение UML-диаграммы компонентов

Кейс 6. Безопасное хранилище документов с версионированием

Контекст: Система управления файлами с шифрованием, сжатием и интеграцией Git/SVN.

Требования:

- Работа с файловой системой: чтение/запись, рекурсивный обход папок.
- Сжатие потоков, кодирование/декодирование данных (Base64/Unicode).
- Применение принципов безопасности и масштабируемости.
- Иерархия классов с инкапсуляцией, полиморфизмом и интерфейсами доступа.
- Интеграция с системой контроля версий (логика коммитов/тегов).

Кейс 7. Анализатор зависимостей кодовой базы

Контекст: Инструмент для анализа связей между модулями проекта, поиска циклических зависимостей и порядка сборки.

Требования:

- Представление графов зависимостей (матрица/список смежности).
- Поиск в глубину/ширину, топологическая сортировка.
- Триальные деревья (Trie) для индексации импортируемых символов.
- Оценка сложности в худшем/среднем случае.
- Построение диаграммы классов для визуализации внутренней структуры.

Кейс 8. Универсальная библиотека статистической обработки данных

Контекст: Создание переиспользуемой библиотеки для работы с коллекциями, расчёта метрик и экспорта отчётов.

Требования:

- Проектирование библиотеки с чётким API, версионированием и документацией.
- Работа с массивами, динамическими коллекциями, датами/временем.
- Инкапсуляция, полиморфизм, перегрузка методов.
- Декомпозиция на подмодули (ввод, обработка, экспорт), создание спецификаций.

Кейс 9. Инструмент сравнения биологических последовательностей

Контекст: Десктопное приложение для анализа текстовых последовательностей (ДНК/белки) с визуализацией результатов.

Требования:

- Динамическое программирование для глобального/локального выравнивания.
- Хеш-функции для строк, алгоритм Рабина-Карпа для поиска мотивов.
- Асинхронная обработка больших файлов без блокировки UI.
- GUI с графиками, стилизацией, валидацией входных данных

Кейс 10. Рефакторинг монолита в модульную масштабируемую систему

Контекст: Проектирование стратегии перехода устаревшего приложения на модульную архитектуру с сохранением обратной совместимости.

Требования:

- Декомпозиция монолита на независимые модули.
- Применение архитектурных шаблонов (MVC/MVVM/MVP).
- Обеспечение безопасности, производительности и горизонтальной масштабируемости.
- CI/CD-пайплайн с интеграцией систем контроля версий.
- Построение диаграмм классов и компонентов.

<u>по МДК.02.02 Осуществление интеграции программных модулей</u>

Кейс 1. REST API системы управления заказами

Контекст: Разработка бэкенда для интернет-магазина с поддержкой CRUD-операций над заказами и ролевым доступом.

Требования:

- Реализовать маршрутизацию с группировкой (/api/v1/orders, /api/v1/users).
- Обрабатывать Path и Query параметры, валидировать тело запроса (JSON/form).
- Формировать ответы с корректными статус-кодами, заголовками Content-Type, сериализацией объектов.
- Внедрить аутентификацию JWT + авторизацию RBAC.
- Настроить структурированное логирование (INFO/WARN/ERROR) с ротацией файлов.

Кейс 2. Сервис уведомлений в реальном времени

Контекст: Система мгновенных оповещений для пользователей платформы.

Требования:

- Реализовать WebSocket API: открытие/закрытие соединения, передача сообщений, обработка отключений.
- Защитить соединение через WSS, валидировать входящие данные.
- Использовать фоновые задачи для отложенных/повторных уведомлений.
- Предусмотреть горизонтальное масштабирование (sticky sessions или Redis Pub/Sub для синхронизации сокетов).
- Применить защиту от CSRF/XSS при передаче токенов.

Кейс 3. Стратегия миграции монолита в микросервисы

Контекст: Декомпозиция устаревшего монолитного приложения на независимые сервисы (Users, Orders, Payments).

Требования:

- Спроектировать границы сервисов, выбрать синхронное (REST/gRPC) и асинхронное (брокер сообщений) взаимодействие.
- Централизовать конфигурацию (env/конфиг-сервер).
- Внедрить единый формат логирования и сбор метрик (

Кейс 4. Защищённый файловый шлюз

Контекст: API для загрузки, обработки и выдачи файлов с гарантией безопасности и производительности.

Требования:

- Обработка multipart/form-data, валидация MIME-типов, размеров, расширения.
- Асинхронная обработка в фоне (генерация превью, сканирование).
- Выдача файлов через подписанные URL, защита от SQL-инъекций в метаданных.
- Хеширование чувствительных данных, обязательное использование HTTPS.
- Оптимизация запросов к БД (индексы, пагинация).

Кейс 5. API-шлюз с кэшированием и профилированием

Контекст: Единая точка входа для внутренних сервисов с акцентом на производительность.

Требования:

- Группировка маршрутов, проксирование, валидация JWT.
- Внедрить кэширование ответов (Redis/In-Memory) с TTL и инвалидацией.
- Профилировать endpoints: замер времени отклика, потребления CPU/RAM, выявление узких мест.
- Снизить latency за счёт оптимизации сериализации и пулов соединений.
- Логирование с уровнями и корреляцией запросов (Request ID).

Кейс 6. Асинхронный пайплайн аналитики

Контекст: Сбор, обработка и агрегация пользовательских событий для бизнес-аналитики.

Требования:

- Приём событий через REST, публикация в брокер (Kafka/RabbitMQ).
- Асинхронные воркеры: фильтрация, обогащение, агрегация.
- Горизонтальное масштабирование потребителей, мониторинг глубины очередей и ошибок.
- Экспорт агрегированных метрик через REST с пагинацией и фильтрами.
- Конфигурация логирования для распределённой трассировки.

Кейс 7. Контейнеризация и CI/CD-пайплайн

Контекст: Подготовка приложения к промышленному развёртыванию с автоматизацией доставки.

Требования:

- Создать многоэтапные Dockerfile, настроить docker-compose/K8s манифесты.
- Управление конфигурацией через environment/секреты.
- Структурированное логирование в файлы и внешние системы (ELK/Loki).
- Интеграция Prometheus для сбора метрик нагрузки и ошибок.
- Обязательное использование HTTPS/WSS, регулярное сканирование образов на уязвимости.

Кейс 8. Высокопроизводительный поисковый API

Контекст: Сервис сложного поиска по каталогу с фильтрами, сортировкой и пагинацией.

Требования:

- Обработка Query-параметров, строгая валидация входных данных.
- Оптимизация запросов к БД (индексы, EXPLAIN, избегание N+1).
- Внедрение кэширования результатов поиска, инвалидация при обновлениях.
- Профилирование hot-путей, минимизация времени отклика.
- Формирование ответов с корректными заголовками и статус-кодами.

Кейс 9. Система аутентификации и управления сессиями

Контекст: Безопасный модуль входа, регистрации и управления доступом.

Требования:

- Поддержка JWT и сессионной аутентификации, механизм refresh-токенов.
- Secure cookies (HttpOnly, Secure, SameSite), защита от CSRF/XSS/SQLi.
- Хеширование паролей с солью (bcrypt/Argon2), ограничение попыток входа.
- Аудит уязвимостей, применение чек-листа лучших практик OWASP.
- Формирование ответов с cookies, редиректами и статус-кодами.

Кейс 10. Дашборд метрик в реальном времени

Контекст: Бэкенд для отображения системных метрик и бизнес-показателей в live-режиме.

Требования:

- WebSocket push данных, управление состоянием соединений, обработка реконнектов.
- Фоновая агрегация метрик, буферизация пакетов.
- WSS, эффективная сериализация JSON/MsgPack, минимизация payload.
- Горизонтальное масштабирование за балансировщиком, мониторинг отвалов клиентов.
- Профилирование WebSocket-обработчиков, оптимизация GC/памяти.

<u>по МДК.02.03 Поддержка и тестирование программных модулей</u>

Кейс 1. Аудит качества и рефакторинг унаследованного модуля

Контекст: В проект передан модуль расчёта скидок с высоким уровнем технической задолженности. Требуется провести оценку качества, выявить узкие места и подготовить модуль к сопровождению.

Требования:

- Рассчитать статические метрики: LOC, цикломатическая сложность, коэффициенты связности (cohesion) и сцепления (coupling).
- Применить стандарты качества (ISO/IEC 25010 или внутренний Quality Gate) и настроить инструменты статического анализа (SonarQube, ESLint, Pylint, Checkstyle).
- Провести отладку критических ветвей логики с использованием точек останова и пошагового выполнения.
- Написать набор модульных тестов, добиться покрытия ≥80%.

Кейс 2. Отладка и обработка исключений в платёжном шлюзе

Контекст: Модуль обработки транзакций периодически падает с неочевидными ошибками при работе с внешними API.

Требования:

- Воспроизвести сбой, применить стратегии поиска ошибок (половинное деление, проверка граничных условий, исключение внешних факторов).
- Внедрить иерархию кастомных исключений, реализовать корректную обработку (try/catch/finally, fallback-логика).

- Настроить логирование исключений с контекстом (стек, входные данные, ID транзакции).
- Задокументировать процесс отладки: гипотезы, шаги, результаты.

Кейс 3. Валидация форм регистрации: чёрный ящик и дефекты

Контекст: Требуется протестировать форму регистрации пользователя перед релизом.

Требования:

- Применить методы классов эквивалентности и граничных значений для полей: email, пароль, возраст, промокод.
- Сформировать матрицу тест-кейсов, выполнить ручное и автоматизированное тестирование.
- Завести дефекты в системе трекинга (Jira/YouTrack/Redmine), описать шаги воспроизведения, приоритет, severity, статус.
- Отследить жизненный цикл дефектов до закрытия.

Кейс 4. Библиотека парсинга конфигураций: white-box и покрытие

Контекст: Разработка ядра для разбора сложных YAML/JSON конфигов с вложенными условиями и флагами.

Требования:

- Реализовать white-box тестирование: покрытие операторов (statement) и условий (branch/condition).
- Для критически безопасных флагов применить метод комбинаторного покрытия условий (MC/DC).
- Снизить цикломатическую сложность до допустимых порогов (≤10 на функцию).
- Интегрировать проверку покрытия в CI-пайплайн.

Кейс 5. Интеграционное тестирование микросервисной корзины

Контекст: Система корзины взаимодействует с сервисами каталога, авторизации и складского учёта.

Требования:

- Спроектировать интеграционные тесты с моками/stub'ами внешних сервисов.
- Проверить сценарии: успешное добавление, блокировка при отсутствии товара, отказ внешнего API, таймауты.
- Применить динамическое и статическое тестирование контрактов (OpenAPI/Swagger validation).
- Настроить изолированную тестовую среду (docker-compose, testcontainers).

Кейс 6. Нагрузочное и стресс-тестирование API-шлюза

Контекст: Перед запуском рекламной кампании необходимо убедиться, что шлюз выдержит пиковую нагрузку.

Требования:

- Спроектировать нагрузочное (steady load) и стресс-тестирование (spike, soak).
- Мониторить динамические метрики: время отклика, частота отказов, утилизация CPU/RAM, ошибки 5xx.
- Выявить деградацию качества при превышении порогов, зафиксировать точки отказа.
- Сформировать рекомендации по оптимизации и масштабированию.

Кейс 7. Модуль расчёта медицинских дозировок: безопасность и отладка

Контекст: Критически важный модуль, где ошибки недопустимы. Требуется строгий контроль качества и трассируемость.

Требования:

- Применить MC/DC для всех условий расчёта дозировок.
- Реализовать обработку исключений с безопасным fallback и аудит-логами.
- Провести отладку граничных случаев (вес 0, аллергия, передозировка, плавающие точки).
- Завести дефекты в системе контроля с обязательным указанием impact analysis.

Кейс 8. Автоматизация приёмочного тестирования CRM-системы

Контекст: Подготовка системы к UAT (User Acceptance Testing) с заказчиком.

Требования:

- Сформулировать acceptance criteria на основе требований.
- Автоматизировать ключевые пользовательские сценарии (создание лида, конвертация, отчёт).
- Применить статическое тестирование требований и динамическое тестирование UI/API.
- Настроить отчётность для стейкхолдеров (процент прохождения, открытые дефекты, риски).

Кейс 9. Quality Gate и CI/CD для библиотеки валидации данных

Контекст: Внедрение автоматических проверок качества в пайплайн сборки библиотеки.

Требования:

- Настроить статический анализ (сложность, дубликаты, уязвимости).
- Задать пороги: покрытие unit-тестами ≥85%, сложность ≤15, 0 blocker/critical дефектов.
- Интегрировать запуск тестов, генерацию отчётов и блокировку мержа при нарушении Quality Gate.
- Документировать процесс и правила работы с дефектами в пайплайне.

Кейс 10. Комплексное QA финансового калькулятора (сквозной кейс)

Контекст: Полный цикл обеспечения качества для модуля расчёта кредитов/инвестиций.

Требования:

- Статические метрики + стандарты качества.
- White-box (statement/condition) + Black-box (EP/BVA).
- Модульные + интеграционные + системные тесты.
- Обработка исключений, отладка формул, документирование.
- Трекинг дефектов, отчёт о готовности к релизу

<u>по МДК.02.04 Математическое моделирование</u>

Кейс 1. Оптимизация производственного плана предприятия

Контекст: Завод выпускает три вида продукции при ограниченных ресурсах (сырьё, время станков, энергопотребление). Себестоимость нелинейно зависит от объёма партии из-за затрат на переналадку.

Требования:

- Сформулировать математическую модель: определить цель, ограничения, переменные. Пройти основные этапы мат. моделирования.
- Привести задачу к каноническому виду ЛП. Решить графическим методом (для сокращённой постановки) и симплекс-методом.
- Учесть целочисленность переменных (целочисленное программирование).
- Решить смежную задачу о назначениях: распределить 5 операторов по 5 участкам с минимальными затратами.

Кейс 2. Оптимизация логистической сети доставки

Контекст: Компания управляет 4 складами и 6 точками продаж. Необходимо минимизировать транспортные расходы при соблюдении пропускной способности дорог и сроков поставки.

Требования:

- Решить транспортную задачу методом потенциалов или транспортной симплекс-таблицы.
- Построить граф логистической сети, найти кратчайшие пути (Дейкстра/Флойд-Уоршелл).
- Составить дерево решений для выбора хаба (строительство нового склада vs аренда).
- Рассчитать временные параметры поставок и построить сетевой график.

Кейс 3. Управление портфелем ИТ-проектов

Контекст: Руководство выбирает и планирует 6 ИТ-проектов с ограничениями бюджета, сроков и взаимозависимостей. ROI зависит от объёма инвестиций нелинейно.

Требования:

- Применить динамическое программирование для выбора оптимального набора проектов при ограниченном бюджете.
- Построить дерево решений для оценки рисков срыва сроков.
- Составить сетевой график проекта, рассчитать ранние/поздние сроки, резервы, критический путь.
- Учесть нелинейную функцию полезности при оптимизации распределения ресурсов.

Кейс 4. Оптимизация работы кол-центра

Контекст: Кол-центр обрабатывает звонки, чаты и тикеты. Необходимо определить число операторов и структуру очередей для выполнения SLA (80% звонков за 20 сек).

Требования:

- Классифицировать СМО, построить марковский случайный процесс.
- Составить схему гибели и размножения, найти стационарные вероятности состояний.
- Рассчитать метрики: среднее время ожидания, длина очереди, загрузка каналов.
- Построить имитационную модель (SimPy/AnyLogic) для проверки аналитических расчётов.

Кейс 5. Конкурентная стратегия выхода на рынок

Контекст: Два ритейлера выбирают цены и объёмы рекламы на новом рынке. Платежи зависят от стратегий обоих игроков, известны матрицы выигрышей.

Требования:

- Сформулировать матричную и биматричную игру, найти седловые точки и равновесие Нэша (чистые/смешанные стратегии).
- Представить игру в развёрнутой форме, провести обратную индукцию.
- Использовать методы нелинейного программирования для нахождения оптимальных смешанных стратегий

Кейс 6. Управление запасами в условиях стохастического спроса

Контекст: Сеть аптек работает с переменным спросом и случайными сроками поставок. Требуется минимизировать издержки хранения и дефицита.

Требования:

- Построить многоэтапную модель динамического программирования для определения оптимального размера заказа на каждом шаге.
- Описать состояния склада как марковскую цепь, рассчитать вероятности дефицита/переполнения.

- Разработать имитационную модель (Монте-Карло) для оценки политики (s, S) при разных сценариях спроса.
- Чётко описать этапы математического моделирования и верификации.

Кейс 7. Балансировка нагрузки в распределённой энергосети

Контекст: Диспетчерская распределяет генерацию между ТЭЦ, ГЭС и ВИЭ. Потери в линиях нелинейны, включение агрегатов дискретно.

Требования:

- Сформулировать задачу нелинейного программирования с учётом потерь и ограничений мощности.
- Применить целочисленное ЛП для бинарных решений (вкл/выкл генераторов).
- Представить сеть как граф, рассчитать резервные пути и критические узлы.
- Провести имитационное моделирование аварийных сценариев (отключение линий, пики нагрузки).

Кейс 8. Оптимизация приёмного отделения стационара

Контекст: Поток пациентов проходит triage, диагностику, распределение по палатам. Очереди формируются на каждом этапе.

Требования:

- Классифицировать многофазную СМО, построить марковский процесс обслуживания.
- Решить задачу о назначениях: оптимальное распределение врачей по потокам пациентов.
- Пройти этапы построения математической модели: формализация, верификация, валидация.
- Создать имитационную модель для тестирования реорганизации потоков (например, выделение отдельной линии скорой помощи).

Кейс 9. Инвестиционный анализ венчурного портфеля

Контекст: Фонд выбирает стартапы для финансирования. Доходы стохастичны, есть конкуренция за раунды и нелинейная зависимость риска от доли в портфеле.

Требования:

- Построить дерево решений для последовательного инвестирования с учётом этапов развития стартапов.
- Применить нелинейное программирование для оптимизации риск-доходности (аналог модели Марковица с ограничениями).
- Смоделировать взаимодействие фонда и рынка как матричную игру (стратегии: агрессивное/консервативное финансирование vs рост/спад рынка).
- Провести имитационное моделирование (Monte Carlo) для оценки распределения итоговой доходности.

Кейс 10. Комплексная оптимизация цепочки поставок (сквозной)

Контекст: Полная цепочка: сырьё → производство → логистика → ритейл. Очереди на складах, конкуренция с альтернативными каналами, нелинейные издержки хранения, динамическое планирование на 4 квартала.

Требования:

- Интегрировать методы: ЛП/целочисленное (производство), нелинейное (издержки), ДП (квартальное планирование), графы/сети (маршруты), СМО/Марков (склады/транспорт), теорию игр (конкуренция), имитацию (валидация).
- Обосновать выбор моделей для каждого звена, провести верификацию и калибровку.
- Сравнить эффективность детерминированных, стохастических и имитационных подходов.

по МДК.02.05 Численные методы

Кейс 1. Расчёт параметров орбиты спутника с оценкой точности

Контекст: Требуется определить момент времени, когда спутник достигнет заданной высоты, решая нелинейное уравнение движения. Результаты используются для планирования сеансов связи.

Требования:

- Реализовать методы отделения корней и поиска: половинного деления, простой итерации, Ньютона (касательных), хорд.
- Оценить абсолютную и относительную погрешность на каждой итерации, отслеживать потерю значащих цифр из-за округлений.
- Сравнить методы по скорости сходимости (число итераций, вычислительная стоимость).
- Обосновать выбор метода для встраивания в бортовой компьютер с ограниченной разрядностью.

Кейс 2. Калибровка датчиков: решение СЛАУ и анализ устойчивости

Контекст: Система из 10 датчиков требует калибровки. Коэффициенты калибровки находятся из системы линейных уравнений, составленной по эталонным измерениям.

Требования:

- Решить СЛАУ методом Гаусса с выбором главного элемента, вычислить определитель и обратную матрицу.
- Оценить число обусловленности матрицы, проанализировать влияние погрешностей входных данных на решение.
- Реализовать итерационные методы: простой итерации и Зейделя, сравнить скорость сходимости с прямым методом.
- Исследовать потерю точности при вычислениях с плавающей запятой (единичные округления, накопление ошибок

Кейс 3. Восстановление сигнала по дискретным отсчётам

Контекст: АЦП зафиксировал значения физического процесса в 15 точках. Требуется восстановить непрерывную зависимость для расчёта производных и интегралов.

Требования:

- Построить интерполяционные многочлены Лагранжа и Ньютона, оценить погрешность интерполяции.
- Реализовать кубическую сплайн-интерполяцию (естественные граничные условия), сравнить гладкость и точность с полиномиальной интерполяцией.
- Выполнить экстраполяцию за пределы интервала, оценить риски роста погрешности (феномен Рунге).
- Проанализировать влияние погрешности исходных данных (сомнительные цифры) на результат.

Кейс 4. Расчёт объёма резервуара сложной формы

Контекст: Форма резервуара задана таблично. Требуется вычислить объём жидкости при разных уровнях заполнения для системы учёта.

Требования:

- Применить квадратурные формулы Ньютона-Котеса (трапеции, Симпсона) для численного интегрирования.
- Реализовать квадратурную формулу Гаусса, сравнить точность при одинаковом числе узлов.
- Оценить погрешность методов, использовать правило Рунге для апостериорной оценки.

- Учесть погрешность измерения уровней (сомнительные цифры) и её влияние на итоговый объём.

Кейс 5. Моделирование динамики химического реактора

Контекст: Концентрация реагента описывается ОДУ первого порядка. Требуется смоделировать процесс во времени для подбора режима работы.

Требования:

- Реализовать явный метод Эйлера и уточнённую схему Эйлера (метод Эйлера-Коши).
- Реализовать метод Рунге-Кутты 4-го порядка, сравнить точность и устойчивость методов.
- Оценить глобальную погрешность, использовать адаптивный выбор шага.
- Проанализировать накопление вычислительной погрешности при длительном моделировании.

Кейс 6. Оптимизация параметров антенны

Контекст: Требуется найти минимум функции усиления антенны, зависящей от двух геометрических параметров. Функция вычисляется численно, градиент недоступен.

Требования:

- Для одномерных сечений реализовать методы дихотомии и золотого сечения, сравнить скорость сходимости.
- Реализовать методы покоординатного спуска и наискорейшего спуска для функции двух переменных.
- Оценить влияние погрешности вычисления функции на точность локализации минимума.
- Исследовать устойчивость методов к локальным экстремумам и «овражности» функции.

Кейс 7. Обработка экспериментальных данных: интерполяция + интегрирование

Контекст: Получены зашумлённые данные о скорости потока. Требуется восстановить зависимость и вычислить интегральные характеристики (расход, импульс).

Требования:

- Применить сплайн-интерполяцию для сглаживания и восстановления функции.
- Вычислить определённый интеграл методом Гаусса, оценить погрешность с учётом неопределённости исходных данных.
- Выполнить численное дифференцирование сплайна для анализа ускорений, оценить устойчивость к шуму.
- Проанализировать распространение погрешностей через цепочку операций (интерполяция → интегрирование/дифференцирование).

Кейс 8. Расчёт электрической цепи: СЛАУ + итерационные методы

Контекст: Большая резистивная сеть (50+ узлов) описывается разреженной СЛАУ. Требуется найти токи и напряжения с контролем точности.

Требования:

- Составить СЛАУ по законам Кирхгофа, решить методом Гаусса с анализом заполнения матрицы.
- Реализовать метод Зейделя с предобуславливанием, сравнить эффективность для разреженных систем.
- Оценить погрешность решения, используя невязку и апостериорные оценки.
- Исследовать влияние округлений при вычислениях с одинарной/двойной точностью.

Кейс 9. Моделирование теплопередачи: краевая задача ОДУ

Контекст: Распределение температуры в стержне описывается краевой задачей. Требуется получить численное решение для разных граничных условий.

Требования:

- Свести краевую задачу к задаче Коши, применить метод Рунге-Кутты с адаптивным шагом.
- Реализовать метод стрельбы с использованием метода Ньютона для уточнения начального условия.
- Оценить погрешность решения, сравнить с аналитическим решением (если доступно).
- Проанализировать устойчивость метода при жёстких системах.

Кейс 10. Комплексный расчёт аэродинамического профиля (сквозной)

Контекст: Требуется численно исследовать подъёмную силу профиля: решить нелинейное уравнение для угла атаки, интерполировать экспериментальные данные, проинтегрировать распределение давления.

Требования:

- Найти корень нелинейного уравнения (баланс сил) методом Ньютона с оценкой погрешности.
- Интерполировать табличные данные о давлении сплайнами, вычислить интеграл (подъёмную силу) квадратурой Гаусса.
- Решить вспомогательную СЛАУ для восстановления поля скоростей методом Зейделя.
- На всём протяжении отслеживать абсолютную/относительную погрешность, значащие цифры, накопление ошибок.

по МДК.02.06 Безопасность программного обеспечения

Кейс 1. Аудит и защита веб-приложения электронной коммерции

Контекст: Интернет-магазин после автоматизированного сканирования показал критические уязвимости. Требуется провести ручной анализ и внедрить защиты.

Требования:

- Сопоставить найденные проблемы с OWASP Top 10 (Injection, Broken Access Control, XSS, Insecure Deserialization и др.).
- Построить модель угроз (STRIDE/DREAD) для ключевых процессов: корзина, оплата, личный кабинет.
- Внедрить санитизацию ввода, CSP-заголовки, безопасные флаги cookies (HttpOnly, Secure, SameSite).
- Описать принципы безопасного проектирования (Security by Design, Least Privilege, Fail-Safe Defaults).

Кейс 2. Проектирование системы единого входа (SSO) и ролевого доступа

Контекст: Корпоративный портал объединяет 5 внутренних сервисов. Требуется централизованная аутентификация и гибкая авторизация.

Требования:

- Реализовать OAuth 2.0 / OpenID Connect flow (Authorization Code + PKCE).
- Настроить безопасное хранение паролей (Argon2/bcrypt), механизм refresh-токенов, защиту от Account Takeover.
- Внедрить RBAC/ABAC, валидацию scopes и claims, автоматический отзыв сессий.
- Проанализировать криптографические требования к подписи JWT (JWS) и шифрованию (JWE).

Кейс 3. Безопасная архитектура микросервисов с Zero Trust

Контекст: Финтех-платформа из 10+ сервисов требует защиты внутреннего трафика и минимизации blast radius.

Требования:

- Спроектировать архитектуру с принципами Zero Trust: mTLS между сервисами, изолированные namespaces, сегментация сети.
- Провести анализ рисков (NIST SP 800-30 или аналог), выявить точки компрометации.
- Настроить TLS 1.3, управление сертификатами (rotation, revocation), secure bootstrapping сервисов.
- Документировать secure architecture principles (defense-in-depth, secure defaults, auditability).

Кейс 4. Криптография и защита мобильного банковского приложения

Контекст: iOS/Android приложение для платежей подвержено рискам утечки ключей, insecure storage и reverse engineering.

Требования:

- Реализовать безопасное хранение ключей (Secure Enclave / Android Keystore), запретить экспорт приватных ключей.
- Настроить certificate pinning, защиту от перехвата трафика (MITM), secure communication с бэкендом.
- Применить криптографические примитивы для локального шифрования (AES-GCM, ChaCha20-Poly1305).
- Внедрить защиту от модификации APK/IPA, отладки и эмуляции.

Кейс 5. Реализация протокола защищённого обмена сообщениями

Контекст: Корпоративный мессенджер требует end-to-end шифрования с forward secrecy и защитой от replay-атак.

Требования:

- Спроектировать протокол обмена ключами (Ephemeral Diffie-Hellman или X3DH), обосновать выбор параметров.
- Реализовать шифрование сообщений, генерацию уникальных nonce/IV, проверку целостности (MAC/AEAD).
- Обеспечить forward secrecy, обработку потери пакетов, механизм ротации сеансовых ключей.
- Проанализировать типичные ошибки реализации (nonce reuse, padding oracle, side-channel).

Кейс 6. Защищённое облачное хранилище конфиденциальных данных

Контекст: SaaS-платформа хранит документы клиентов в AWS/GCP/Azure. Требуется обеспечить compliance и криптографическую изоляцию.

Требования:

- Внедрить envelope encryption через Cloud KMS, разделить ключи шифрования данных (DEK) и мастер-ключи (KEK).
- Настроить управление секретами (HashiCorp Vault / Cloud Secrets Manager), автоматическую ротацию, аудит доступа.
- Реализовать шифрование на стороне клиента для особо чувствительных данных, безопасно передавать ключи.
- Применить принципы безопасной облачной архитектуры: IAM least privilege, network isolation, logging & monitoring.

Кейс 7. Анализ рисков и защита публичного API-шлюза

Контекст: REST API подвергается credential stuffing, rate abuse и injection-атакам. Требуется комплексная защита.

Требования:

- Провести оценку рисков (вероятность × impact), приоритизировать угрозы по модели STRIDE.
- Внедрить WAF-правила, rate limiting, IP/device fingerprinting, защиту от OWASP API Security Top 10.
- Настроить валидацию JWT/OAuth токенов, строгую схему запросов (JSON Schema/OpenAPI).
- Организовать мониторинг аномалий и автоматический response на инциденты.

Кейс 8. Безопасная обработка платежных данных (PCI DSS)

Контекст: Интеграция платежного модуля без прямого касания к PAN, с соблюдением требований PCI DSS.

Требования:

- Реализовать токенизацию платежных данных, изоляцию контура обработки, шифрование в транзите и покое.
- Настроить безопасное логирование (маскирование PAN/CVV), контроль доступа, регулярный аудит уязвимостей.
- Применить криптографические стандарты (FIPS 140-2/3 validated modules, TLS 1.2+).
- Сформировать отчёт о соответствии требованиям безопасного проектирования.

Кейс 9. Модернизация legacy-системы: от уязвимостей к secure-by-design

Контекст: Устаревшая ERP использует MD5, plaintext-пароли, открытые порты, не имеет модели угроз и CI/CD security checks.

Требования:

- Провести аудит уязвимостей, построить актуальную модель угроз, определить критические риски.
- Спроектировать безопасную архитектуру: замена слабой криптографии, внедрение современных AuthN/AuthZ, сегментация сети.
- Внедрить secure SDLC: SAST/DAST, dependency scanning, secret detection, code review checklists.
- Составить поэтапный план миграции с backward compatibility и rollback-стратегией.

Кейс 10. Сквозной кейс: Защищённая платформа телемедицины

Контекст: Разработка платформы с видеоконсультациями, хранением медкарт, интеграцией с внешними лабораториями и мобильным приложением для пациентов.

Требования:

- Полный цикл: threat modeling, OWASP для веб/мобайл, OAuth/OIDC, E2E для видео, шифрование БД (TDE), облачный KMS.
- Реализовать криптопротоколы для signalling и media streams (DTLS-SRTP), certificate pinning в мобильных клиентах.
- Обеспечить compliance (ФЗ-152/GDPR/HIPAA), аудит доступа, инцидент-менеджмент, регулярные пентесты.
- Документировать все архитектурные и криптографические решения, провести верификацию реализации.

Критерии оценки:

отметка «5»: Задание выполнено в полном объёме с соблюдением необходимой последовательности. Студент работал полностью самостоятельно.

отметка «4»: Практическое задание выполнено студентом в полном объёме и самостоятельно. Допускается отклонение от необходимой последовательности выполнения, не влияющее на правильность конечного результата. Допускаются неточности и небрежность в оформлении результатов задания.

отметка «3»: Практическое задание выполнено и оформлено студентом с помощью преподавателя или хорошо подготовленных и уже выполнивших на «отлично» данную работу студентов. На выполнение задания затрачено много времени.

отметка «2»: Выставляется в том случае, когда студент оказался неподготовленным к выполнению задания. Полученные результаты не позволяют сделать правильных выводов и полностью расходятся с поставленной целью. Обнаружено плохое знание теоретического материала и отсутствие необходимых умений. Руководство и помощь со стороны преподавателя неэффективны из-за плохой подготовки студента.

Ситуационные задачи для учебной практики

по ПМ.02 «Разработка и интеграция модулей программного обеспечения»

Задание 1. Проектирование модуля по техническому заданию

Ситуация: Заказчик передал ТЗ на разработку модуля «Уведомления пользователей» (email, push, SMS). Требуется спроектировать архитектуру до написания кода.

Задание:

1. Проанализировать ТЗ, выделить функциональные и нефункциональные требования.
2. Спроектировать внутреннюю структуру модуля (классы/компоненты, зависимости).
3. Визуализировать архитектуру (UML-диаграммы компонентов и развертывания).
4. Обосновать выбранные паттерны и принципы проектирования.

Задание 2. Определение интерфейсов и контракт API

Ситуация: Модуль «Уведомления» должен взаимодействовать с модулем «Пользователи» и внешним SMS-шлюзом.

Задание:

1. Спроектировать REST API: эндпоинты, HTTP-методы, коды ответов.
2. Описать форматы запросов/ответов (JSON), заголовки авторизации и валидации.
3. Сформировать спецификацию в формате OpenAPI 3.0.
4. Развернуть mock-сервер для эмуляции ответов.

Задание 3. Создание модуля и пошаговая отладка

Ситуация: При реализации модуля валидации телефонных номеров возникли ошибки: некорректное отбрасывание кодов стран, падение при пустом вводе.

Задание:

1. Реализовать модуль валидации согласно контракту.
2. Воспроизвести ошибки, использовать отладчик (breakpoints, watch, call stack).
3. Исправить логику, добавить обработку граничных случаев.
4. Зафиксировать процесс отладки и принятые решения.

Задание 4. Интеграция с внешним сервисом и брокером сообщений

Ситуация: Модуль уведомлений должен работать асинхронно: принимать задачи из очереди и отправлять их через внешний API SMS-провайдера.

Задание:

1. Настроить подключение к брокеру сообщений (RabbitMQ / Redis Streams / локальный эмулятор).
2. Реализовать producer (отправка задач) и consumer (обработка и вызов внешнего API).
3. Внедрить механизм повторных попыток (retry) и dead-letter queue.
4. Протестировать устойчивость при имитации сбоев сети.

Задание 5. Формирование тестовых сценариев и оценка ресурсов

Ситуация: Перед запуском модуля в тестовую среду необходимо подготовить план тестирования и рассчитать трудозатраты.

Задание:

1. На основе спецификации составить набор тестовых сценариев (позитивные, негативные, граничные).
2. Определить виды тестирования (модульное, интеграционное, нагрузочное).
3. Оценить время выполнения, необходимые данные и человеческие ресурсы.
4. Согласовать план с условным «руководителем проекта».

Задание 6. Подготовка тестовой платформы и выполнение процедур

Ситуация: Тестовая среда «сломана» после предыдущих прогонов. Требуется развернуть чистое окружение и выполнить тесты.

Задание:

1. Развернуть ОС/контейнер, установить зависимости, СУБД, тестовые данные.
2. Настроить конфигурацию окружения (env-файлы, доступы).
3. Выполнить тестовые процедуры по сценариям из Задания 5.
4. Зафиксировать фактические результаты и отклонения.

Задание 7. Формирование отчётности по тестированию

Ситуация: По итогам тестового цикла необходимо представить формализованный отчёт для принятия решения о готовности модуля к релизу.

Задание:

1. Проанализировать результаты прогонов, рассчитать метрики (покрытие, плотность дефектов, pass/fail rate).
2. Классифицировать дефекты по критичности и приоритету.
3. Оформить отчёт в соответствии с корпоративным/учебным регламентом.
4. Сформулировать рекомендации по доработке или выпуску.

Задание 8. Документирование кода и API

Ситуация: Проект проходит аудит. Требуется привести документацию в соответствие с требованиями поддержки и передачи кода другим разработчикам.

Задание:

1. Добавить документационные комментарии в код (docstrings / JSDoc / XML Comments).
2. Сгенерировать документацию API из OpenAPI-спецификации.
3. Создать руководство по использованию модуля для разработчиков.
4. Настроить автогенерацию и публикацию документации.

Задание 9. Рефакторинг и оптимизация существующего модуля

Ситуация: Унаследованный модуль обработки заказов имеет высокую цикломатическую сложность, дублирование кода и плохую тестируемость.

Задание:

1. Провести статический анализ кода, зафиксировать метрики до рефакторинга.
2. Выполнить рефакторинг: выделение методов, устранение дублирования, применение паттернов.
3. Обновить unit-тесты, убедиться в сохранении функциональности.
4. Сравнить метрики до/после, описать принятые архитектурные решения.

Задание 10. Комплексная интеграция и CI-настройка

Ситуация: Необходимо объединить разработанный модуль с внешним сервисом, автоматизировать проверку качества и подготовить пакет для развёртывания.

Задание:

1. Настроить CI-конвейер: сборка → линтинг → unit-тесты → генерация документации → публикация артефактов.
2. Интегрировать модуль с тестовым стендом через API-шлюз или proxy.
3. Выполнить сквозной тест интеграции, зафиксировать результаты.
4. Подготовить итоговый пакет документации и инструкцию по развёртыванию.

Критерии оценки:

отметка «5»: Задание выполнено в полном объёме с соблюдением необходимой последовательности. Студент работал полностью самостоятельно.

отметка «4»: Практическое задание выполнено студентом в полном объёме и самостоятельно. Допускается отклонение от необходимой последовательности выполнения, не влияющее на правильность конечного результата. Допускаются неточности и небрежность в оформлении результатов задания.

отметка «3»: Практическое задание выполнено и оформлено студентом с помощью преподавателя или хорошо подготовленных и уже выполнивших на «отлично» данную работу студентов. На выполнение задания затрачено много времени.

отметка «2»: Выставляется в том случае, когда студент оказался неподготовленным к выполнению задания. Полученные результаты не позволяют сделать правильных выводов и полностью расходятся с поставленной целью. Обнаружено плохое знание теоретического материала и отсутствие необходимых умений. Руководство и помощь со стороны преподавателя неэффективны из-за плохой подготовки студента.

Ситуационные задачи для производственной практики

ПМ.02 «Разработка и интеграция модулей программного обеспечения»

Задание 1. Проектирование архитектуры модуля «Управление заказами»

Ситуация: Заказчик передал техническое задание на разработку модуля обработки клиентских заказов. Требуется спроектировать решение до написания кода, чтобы избежать переделок на поздних этапах.

Задание:

1. Декомпозировать ТЗ: выделить функциональные и нефункциональные требования.
2. Спроектировать внутреннюю структуру модуля (компоненты, слои, зависимости).
3. Визуализировать архитектуру (UML: компонентов, последовательности, развёртывания).
4. Описать контракты взаимодействия между компонентами (форматы данных, протоколы, ошибки).

Задание 2. Разработка ядра модуля и пошаговая отладка

Ситуация: При реализации расчётного алгоритма модуля возникают логические ошибки: некорректный учёт скидок, падение при нулевых значениях, неочевидные побочные эффекты.

Задание:

1. Реализовать функциональное ядро модуля согласно спецификации.
2. Воспроизделить дефекты, использовать отладчик (breakpoints, watch, call stack, conditional breakpoints).
3. Исправить логику, добавить обработку граничных и исключительных ситуаций.
4. Зафиксировать ход отладки в журнале.

Задание 3. Профилирование и оптимизация производительности

Ситуация: Модуль обработки больших выборок данных превышает допустимое время отклика (SLA > 2 сек). Требуется выявить узкие места и оптимизировать код.

Задание:

1. Провести профилирование приложения (CPU, память, I/O, время выполнения функций).
2. Выявить алгоритмические и структурные bottleneck-и.
3. Применить оптимизации (кэширование, замена структур данных, асинхронность, рефакторинг циклов).
4. Провести сравнительный бенчмарк до/после.

Задание 4. Интеграция с внешними API и настройка шины данных

Ситуация: Модуль должен обмениваться данными с внешним платежным шлюзом и службой логистики. Требуется обеспечить надёжную асинхронную доставку сообщений.

Задание:

1. Реализовать клиенты для REST/gRPC сервисов (авторизация, сериализация, обработка ошибок).
2. Настроить интеграционную платформу/брокер сообщений (очереди, retry-политика, dead-letter queue).
3. Обеспечить идемпотентность и логирование транзакций.
4. Протестировать взаимодействие при имитации сбоев сети.

Задание 5. Обеспечение совместимости и стабильности системы

Ситуация: Решение должно стабильно работать в гетерогенной среде: разные версии ОС, СУБД, библиотек зависимостей. Зафиксированы конфликты версий и «падения» при обновлении пакетов.

Задание:

1. Составить матрицу совместимости (ОС, рантаймы, зависимости, СУБД).
2. Настроить изоляцию окружения (venv, venv, контейнеризация, пакетные менеджеры).
3. Внедрить механизмы graceful degradation и fallback-логики.
4. Провести стресс-тест стабильности при длительной работе.

Задание 6. Планирование тестирования и разработка сценариев

Ситуация: Перед передачей модуля в QA-отдел необходимо согласовать стратегию тестирования, оценить трудозатраты и подготовить сценарии.

Задание:

1. Определить виды и уровни тестирования (модульное, интеграционное, регрессионное, нагрузочное).
2. Разработать набор тестовых сценариев (позитивные, негативные, граничные, error-пути).
3. Оценить объём тестирования: количество тест-кейсов, время выполнения, необходимые данные, человеческие ресурсы.
4. Согласовать план с условным руководителем практики.

Задание 7. Развертывание и конфигурирование тестовой платформы

Ситуация: Тестовый стенд не соответствует требованиям релиза. Требуется развернуть чистое окружение, идентичное production, и подтвердить готовность к тестированию.

Задание:

1. Развернуть ОС/контейнер, установить СУБД, веб-сервер, вспомогательное ПО.
2. Настроить конфигурацию (env-файлы, доступы, сети, права доступа, фаерволы).
3. Загрузить эталонные тестовые данные, выполнить миграции.
4. Сформировать акт готовности тестовой среды по регламенту предприятия.

Задание 8. Выполнение тестовых процедур на эталонных данных

Ситуация: Стенд готов. Необходимо выполнить запланированные тестовые процедуры, зафиксировать отклонения и подготовить данные для анализа.

Задание:

1. Запустить автоматизированные и ручные тесты по сценариям из Задания 6.
2. Выполнить процедуры на подготовленных тестовых наборах данных (валидные, невалидные, пограничные).
3. Зафиксировать фактические результаты, оформить дефекты в трекере.
4. Провести ретест после исправлений, обновить статусы тест-кейсов.

Задание 9. Генерация технической документации и автодокументирование

Ситуация: Проект проходит внутренний аудит. Требуется привести документацию в состояние, достаточное для передачи модула команде сопровождения.

Задание:

1. Добавить структурные комментарии в код (docstrings, JSDoc, XML Comments) согласно стандарту компании.
2. Сгенерировать документацию API из спецификации и аннотаций.
3. Создать руководство разработчика и администратора (архитектура, развёртывание, troubleshooting).
4. Настроить автогенерацию и публикацию в корпоративном wiki.

Задание 10. Финальная сборка, интеграционный контроль и передача в эксплуатацию

Ситуация: Модуль прошёл тестирование и документирован. Требуется подготовить релиз-пакет, провести финальную проверку интеграции и передать решение в эксплуатацию.

Задание:

1. Выполнить сборку решения, проверить зависимости, сигнатуры и целостность артефактов.
2. Провести smoke-тестирование на staging-окружении, убедиться в совместной работе всех компонентов.
3. Сформировать релиз-ноты, чек-лист передачи и акт приёмки-передачи.
4. Презентовать решение руководителю практики от организации.

Критерии оценки:

отметка «5»: Задание выполнено в полном объёме с соблюдением необходимой последовательности. Студент работал полностью самостоятельно.

отметка «4»: Практическое задание выполнено студентом в полном объёме и самостоятельно. Допускается отклонение от необходимой последовательности выполнения, не влияющее на правильность конечного результата. Допускаются неточности и небрежность в оформлении результатов задания.

отметка «3»: Практическое задание выполнено и оформлено студентом с помощью преподавателя или хорошо подготовленных и уже выполнивших на «отлично» данную работу студентов. На выполнение задания затрачено много времени.

отметка «2»: Выставляется в том случае, когда студент оказался неподготовленным к выполнению задания. Полученные результаты не позволяют сделать правильных выводов и полностью расходятся с поставленной целью. Обнаружено плохое знание теоретического материала и отсутствие необходимых умений. Руководство и помощь со стороны преподавателя неэффективны из-за плохой подготовки студента.

5. КОНТРОЛЬ И ОЦЕНКА РЕЗУЛЬТАТОВ ОСВОЕНИЯ ПРОФЕССИОНАЛЬНОГО МОДУЛЯ (ВИДА ПРОФЕССИОНАЛЬНОЙ ДЕЯТЕЛЬНОСТИ)

Код ПК, ОК	Критерии оценки результата (показатели освоенности компетенций)	Формы контроля и методы оценки
ОК.01	распознает задачу и/или проблему в профессиональном и/или социальном контексте; анализирует задачу и/или проблему; определяет этапы решения задачи; выявляет и эффективно находит информацию, необходимую для решения задачи и/или проблемы; составляет план действия; определяет необходимые ресурсы; оценивает результат и последствия своих действий (самостоятельно или с помощью наставника)	Контрольные работы, зачеты, квалификационные испытания, защита курсовых и дипломных проектов (работ), учебная и производственная практики, экзамены. Интерпретация результатов выполнения практических и лабораторных заданий, оценка решения ситуационных задач, оценка тестового контроля, результатов наблюдений за деятельностью обучающегося в процессе учебной и производственной практики
ОК.02	определяет задачи для поиска информации; определяет необходимые источники информации; планирует процесс поиска; структурирует полученную информацию; выделяет наиболее значимое в перечне информации; оценивает практическую значимость результатов поиска; оформляет результаты поиска	
ОК.03	определяет актуальность нормативно-правовой документации в профессиональной деятельности; применяет современную научную профессиональную терминологию; определяет и выстраивает траектории профессионального развития и самообразования	
ОК.04	организовывает работу коллектива и команды; взаимодействует с коллегами, руководством, клиентами в ходе профессиональной деятельности	
ОК.05	излагает свои мысли и оформлять документы по профессиональной тематике на государственном языке, проявлять толерантность в рабочем коллективе	
ОК.06	описывает значимость своей специальности	
ОК.07	соблюдает нормы экологической безопасности определять направления ресурсосбережения в рамках профессиональной деятельности по специальности	
ОК.08	чередует смену деятельности; выполняет комплекс лечебной гимнастики с учетом профессиональной деятельности	
ОК.09	понимает общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимает тексты на базовые профессиональные темы участвовать в диалогах на знакомые общие и профессиональные темы; пишет простые связные сообщения на знакомые или интересующие профессиональные темы	
ПК 2.1	проектирует модули программного обеспечения с учетом технического задания; визуализирует и описывает архитектурные решения; определяет интерфейсы и взаимодействие модулей в системе	
ПК 2.2	создает модули программного обеспечения; оптимизирует код и алгоритмы программных модулей для увеличения производительности; мониторит и анализирует	

	производительность приложений	
ПК 2.3	проводит интеграцию программных модулей и компонентов в единое программное решение; работает с API и веб-сервисами для взаимодействия между модулями; работает с интеграционными платформами и инструментами; обеспечивает совместимость и стабильность системы	
ПК 2.4	проводит отладку программного обеспечения на уровне программных модулей; тестирует программное обеспечение; формирует тестовые сценарии; готовит тестовые платформы (устанавливает операционную систему, дополнительное программное обеспечение и другое по необходимости); проводит оценку объема тестирования программного обеспечения с целью определения необходимых ресурсов для его выполнения; настраивает тестовые среды и аппаратные средства для выполнения тестирования программного обеспечения в соответствии с заданием на тестирование в пределах своей компетенции; формирует и предоставляет отчетность о подготовке к выполнению задания на тестирование программного обеспечения в соответствии с установленными регламентами; выполняет тестовые процедуры на тестовых данных	
ПК 2.5	создает техническую документацию для модулей; документирует код, API и интерфейсов; работает со специализированным программным обеспечением по документированию программного кода	