Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования «Ростовский государственный экономический университет (РИНХ)»
Финансово-экономический колледж

УТВЕРЖДАЮ
Директор
_____ Р. А. Сычев
«__» _____ 2026г.

Рабочая программа МДК
Технологии выполнения работ по должности «Программист»

Специальность
09.02.12 ТЕХНИЧЕСКАЯ ЭКСПЛУАТАЦИЯ И СОПРОВОЖДЕНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ

Форма обучения очная

Часов по учебному плану 182
в том числе:
аудиторные занятия 100
самостоятельная работа 76

Ростов-на-Дону
2026 г.

Распределение часов дисциплины по семестрам

Семестр (<Курс>.<Семестр на курсе>)	**5 (3.1)**		Итого	
Недель	10			
Вид занятий	УП	РП	УП	РП
Лекции	30	30	30	30
Практические	70	70	70	70
Итого ауд.	100	100	100	100
Контактная работа	100	100	100	100
Сам. работа	76	76	76	76
Часы на контроль	6	6	6	6
Итого	182	182	182	182

ОСНОВАНИЕ

Федеральный государственный образовательный стандарт среднего профессионального образования по специальности 09.02.12 ТЕХНИЧЕСКАЯ ЭКСПЛУАТАЦИЯ И СОПРОВОЖДЕНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ (Приказ Министерства просвещения Российской Федерации от 10 марта 2025г № 184).

Рабочая программа составлена по образовательной программе 09.02.12 ТЕХНИЧЕСКАЯ ЭКСПЛУАТАЦИЯ И СОПРОВОЖДЕНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ для набора 2026 года.
программа среднего профессионального образования

Учебный план утвержден учёным советом вуза от 03.03.2026 протокол № 9

Программу составил(и): Преподаватель, Журавлёв Д.Г.

Председатель ЦМК: Ламин В.А.

Рассмотрено на заседании ЦМК от 06.03.2026 протокол № 7

1. ЦЕЛИ ОСВОЕНИЯ ДИСЦИПЛИНЫ	
1.1	Формирование у обучающихся системных знаний и практических навыков, необходимых для выполнения работ программиста: формализация задач, разработка алгоритмов, написание программного кода, работа с базами данных, использование систем контроля версий, отладка и оформление кода в соответствии с требованиями.

2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ	
Цикл (раздел) ООП:	МДК.03
2.1	**Требования к предварительной подготовке обучающегося:**
2.1.1	Настройка и обеспечение работоспособности программных и аппаратных средств устройств инфокоммуникационных систем
2.1.2	Проектирование и разработка информационных систем
2.1.3	Разработка информационных систем
2.1.4	Тестирование и эксплуатация информационных систем
2.1.5	Базы данных
2.1.6	Основы алгоритмизации и программирования
2.2	**Дисциплины и практики, для которых освоение данной дисциплины (модуля) необходимо как предшествующее:**
2.2.1	Автоматизация процессов тестирования программного обеспечения
2.2.2	Демонстрационный экзамен
2.2.3	Обеспечение качества программного обеспечения

3. ТРЕБОВАНИЯ К РЕЗУЛЬТАТАМ ОСВОЕНИЯ ДИСЦИПЛИНЫ

3.1 Знать

ПК. 3.1 Проверка работоспособности и рефакторинг кода программного обеспечения

- Стандарты кодирования для целевого языка программирования и среды разработки.
- Методы моделирования программного обеспечения с применением объектных моделей.
- Подходы к разработке процедур проверки работоспособности и измерения характеристик программного обеспечения.
- Правила работы с открытыми библиотеками и управление ими.

ПК 3.2: Интеграция программных модулей и компонентов и проверка работоспособности выпусков программного продукта

- Методы и стратегии интеграции программных модулей и критерии выбора подхода в соответствии с техническим заданием.
- Стандарты кодирования, регламентирующие оформление кода при интеграции.
- Принципы построения объектных моделей для описания взаимодействия модулей, зависимостей и потоков данных.
- Технологии создания анимационных эффектов в целевой среде разработки.

ПК 3.3 Разработка и отладка программного кода

- Стандарты оформления технического задания и требования к его структуре.
- Методы формализации поставленных задач.
- Базовые алгоритмические конструкции и принципы построения эффективных алгоритмов для последующей реализации в программном коде.
- Правила логически и технически грамотного изложения разделов технического задания.

3.2 Уметь

ПК. 3.1 Проверка работоспособности и рефакторинг кода программного обеспечения

- Разрабатывать процедуры проверки работоспособности, которые корректно функционируют в полном соответствии с техническим заданием и охватывают все функциональные требования.
- Реализовывать процедуры измерения характеристик программного обеспечения в целевой среде программирования.
- Выполнять предварительное моделирование приложения, применяя объектные модели для описания структуры и поведения системы до написания кода.
- Подключать и использовать открытые библиотеки для решения прикладных задач приложения.

ПК 3.2: Интеграция программных модулей и компонентов и проверка работоспособности выпусков программного продукта

- Разрабатывать процедуры интеграции программных модулей, которые корректно функционируют в полном соответствии с техническим заданием.
- Выполнять предварительное моделирование приложения с применением объектных моделей для проектирования структуры и взаимодействия интегрируемых модулей.
- Реализовывать анимационные эффекты, интегрируя их в общую архитектуру приложения без нарушения логики работы модулей.
- Оформлять программный код в полном соответствии со стандартами кодирования.

ПК 3.3 Разработка и отладка программного кода

- Выполнять полную формализацию задачи, выделяя все сущности, ограничения и критерии успешности.
- Разрабатывать детальные алгоритмы и блок-схемы для каждого функционального требования, пригодные для прямой трансляции в программный код.
- Оформлять техническое задание в полном соответствии с установленными стандартами.
- Логически выстраивать разделы технического задания, обеспечивая непротиворечивость, полноту и однозначность трактовки требований.

3.3 Владеть

ПК. 3.1 Проверка работоспособности и рефакторинг кода программного обеспечения

- Созданием объектных моделей для проектирования веб-приложения по предложенному техническому заданию.
- Разработкой программного кода, оформленный полном соответствии со стандартами кодирования.
- Интеграцией процедуры проверки и измерения характеристик в приложение с возможностью их автоматического или полуавтоматического запуска.
- Анализом результатов проверки работоспособности и измерений характеристик на предмет соответствия техническому заданию и выявления узких мест производительности.

ПК 3.2: Интеграция программных модулей и компонентов и проверка работоспособности выпусков программного продукта

- Построением и использованием объектных моделей для планирования интеграции модулей до начала программирования.
- Созданием процедур интеграции, обеспечивающих корректную сборку, инициализацию и взаимодействие всех модулей системы согласно техническому заданию.
- Применением анимационных эффектов как составной части интегрированного приложения с учётом производительности и совместимости модулей.
- Проверкой корректности работы процедур интеграции, включая выявление и устранение конфликтов интерфейсов, типов данных и последовательностей вызовов.

ПК 3.3 Разработка и отладка программного кода

- Навыками применением стандартизированных шаблонов и средств оформления технического задания с соблюдением всех требований к форме и содержанию.
- Алгоритмизацией и структурированием задач любой сложности для последующей разработки программного кода без пробелов в логике.
- Технически грамотной формулировкой требований к программному продукту.
- Критической проверкой технического задания на соответствие стандартам, полноту, однозначность и технологичность реализации.

4. СТРУКТУРА И СОДЕРЖАНИЕ ДИСЦИПЛИНЫ (МОДУЛЯ)

Код занятия	Наименование разделов и тем /вид занятия/	Семестр / Курс	Часов	Компетен-ции	Литература	Примечание
	Раздел 1. Формализация и алгоритмизация поставленных задач для разработки программного кода.					
1.1	Понятие формализации задачи. Выделение входных, выходных данных, ограничений и	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	

	допущений. Методы алгоритмизации: блок-схемы, псевдокод, структурные схемы. /Лек/					
1.2	Формализация задачи: по текстовому описанию выделить входы, выходы, ограничения. Построение блок-схем алгоритмов (линейные, разветвляющиеся, циклические). /Пр/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
1.3	Формализация задачи: по текстовому описанию выделить входы, выходы, ограничения. Построение блок-схем алгоритмов (линейные, разветвляющиеся, циклические). /Пр/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
1.4	Формализация задачи: по текстовому описанию выделить входы, выходы, ограничения. Построение блок-схем алгоритмов (линейные, разветвляющиеся, циклические). /Ср/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
1.5	Формализация задачи: по текстовому описанию выделить входы, выходы, ограничения. Построение блок-схем алгоритмов (линейные, разветвляющиеся, циклические). /Ср/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
1.6	Основные алгоритмические конструкции: следование, ветвление, циклы. Сложность алгоритмов, понятие асимптотики (О-нотация). /Лек/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
1.7	Запись алгоритмов на псевдокоде. Оценка временной сложности простых алгоритмов (поиск, сортировка). /Пр/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
1.8	Запись алгоритмов на псевдокоде. Оценка временной сложности простых алгоритмов (поиск, сортировка). /Пр/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
	Раздел 2. Написание программного кода с использованием языков программирования,					

	определения и манипулирования данными в базах данных.					
2.1	Синтаксис и семантика выбранного языка программирования. Типы данных, переменные, операторы, управляющие конструкции. /Лек/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
2.2	Функции, модули, классы. Основы обьектно-ориентированного программирования. Взаимодействие с базами данных: SQL. /Лек/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
2.3	Функции, модули, классы. Основы обьектно-ориентированного программирования. Взаимодействие с базами данных: SQL. /Ср/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
2.4	Функции, модули, классы. Основы обьектно-ориентированного программирования. Взаимодействие с базами данных: SQL. /Ср/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
2.5	Подключение к СУБД из приложения: строки подключения, выполнение запросов, обработка результатов. ORM: понятие, преимущества, примеры. /Лек/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
2.6	Написание программы линейной структуры. Разработка программы с условными операторами. Реализация циклических алгоритмов. Создание функций и вызов их из основной программы. /Пр/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
2.7	Написание программы линейной структуры. Разработка программы с условными операторами. Реализация циклических алгоритмов. Создание функций и вызов их из основной программы. /Пр/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
2.8	Проектирование простой базы данных и написание скриптов SQL. Подключение к БД из приложения и выполнение SELECT-запроса.	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	

	Реализация CRUD-операций через параметризованные запросы. Использование ORM для работы с данными. /Пр/					
2.9	Проектирование простой базы данных и написание скриптов SQL. Подключение к БД из приложения и выполнение SELECT-запроса. Реализация CRUD-операций через параметризованные запросы. Использование ORM для работы с данными. /Пр/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
2.10	Проектирование простой базы данных и написание скриптов SQL. Подключение к БД из приложения и выполнение SELECT-запроса. Реализация CRUD-операций через параметризованные запросы. Использование ORM для работы с данными. /Ср/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
2.11	Проектирование простой базы данных и написание скриптов SQL. Подключение к БД из приложения и выполнение SELECT-запроса. Реализация CRUD-операций через параметризованные запросы. Использование ORM для работы с данными. /Ср/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
2.12	Проектирование простой базы данных и написание скриптов SQL. Подключение к БД из приложения и выполнение SELECT-запроса. Реализация CRUD-операций через параметризованные запросы. Использование ORM для работы с данными. /Ср/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
2.13	Разработка простого модульного. консолького	5	2	ПК 3.1. ПК 3.2. ПК	Л1.1 Л1.2 Л1.3 Л2.1	

	приложения с использованием ООП. /Пр/			3.3.	Л2.2 Э1 Э2	
2.14	Разработка простого модульного. консолького приложения с использованием ООП. /Пр/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
2.15	Доработка приложения с хранением данных в БД и с использованием ООП. Интеграция консольного интерфейса с БД. /Пр/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
2.16	Доработка приложения с хранением данных в БД и с использованием ООП. Интеграция консольного интерфейса с БД. /Пр/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
2.17	Доработка приложения с хранением данных в БД и с использованием ООП. Интеграция консольного интерфейса с БД. /Ср/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
2.18	Доработка приложения с хранением данных в БД и с использованием ООП. Интеграция консольного интерфейса с БД. /Ср/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
2.19	Доработка приложения с хранением данных в БД и с использованием ООП. Интеграция консольного интерфейса с БД. /Ср/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
2.20	Разработка графического интерфейса приложения. /Пр/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
2.21	Разработка графического интерфейса приложения. /Пр/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
2.22	Разработка графического интерфейса приложения. /Ср/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
2.23	Разработка графического интерфейса приложения. /Ср/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
2.24	Разработка графического интерфейса приложения. /Ср/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
	Раздел 3. Работа с системой управления версиями программного кода.					
3.1	Назначение систем контроля версий (VCS). Основы Git. Основные команды. /Лек/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
3.2	Работа с удалёнными	5	2	ПК 3.1. ПК	Л1.1 Л1.2	

	репозиториями. Ветвление и слияние: branch, checkout, merge, разрешение конфликтов. Игнорирование файлов. Стратегии ветвления. /Лек/			3.2. ПК 3.3.	Л1.3 Л2.1 Л2.2 Э1 Э2	
3.3	Работа с удалёнными репозиториями. Ветвление и слияние: branch, checkout, merge, разрешение конфликтов. Игнорирование файлов. Стратегии ветвления. /Ср/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
3.4	Работа с удалёнными репозиториями. Ветвление и слияние: branch, checkout, merge, разрешение конфликтов. Игнорирование файлов. Стратегии ветвления. /Ср/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
3.5	Работа с удалёнными репозиториями. Ветвление и слияние: branch, checkout, merge, разрешение конфликтов. Игнорирование файлов. Стратегии ветвления. /Ср/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
3.6	Установка Git, настройка имени пользователя и email. Создание локального репозитория, добавление файлов, первый коммит. Просмотр истории коммитов, изменение последнего коммита. Создание и переключение между ветками. Слияние веток. /Пр/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
3.7	Установка Git, настройка имени пользователя и email. Создание локального репозитория, добавление файлов, первый коммит. Просмотр истории коммитов, изменение последнего коммита. Создание и переключение между ветками. Слияние веток. /Пр/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
3.8	Создание конфликта и его разрешение. Регистрация на GitHub/GitLab, создание удалённого репозитория. Отправка локального репозитория на удалённый и клонирование. Работа с	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	

	.gitignore. /Пр/					
3.9	Создание конфликта и его разрешение. Регистрация на GitHub/GitLab, создание удалённого репозитория. Отправка локального репозитория на удалённый и клонирование. Работа с .gitignore. /Пр/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
3.10	Создание конфликта и его разрешение. Регистрация на GitHub/GitLab, создание удалённого репозитория. Отправка локального репозитория на удалённый и клонирование. Работа с .gitignore. /Ср/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
3.11	Создание конфликта и его разрешение. Регистрация на GitHub/GitLab, создание удалённого репозитория. Отправка локального репозитория на удалённый и клонирование. Работа с .gitignore. /Ср/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
3.12	Создание конфликта и его разрешение. Регистрация на GitHub/GitLab, создание удалённого репозитория. Отправка локального репозитория на удалённый и клонирование. Работа с .gitignore. /Ср/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
3.13	Выполнение pull request. Командная работа. /Пр/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
3.14	Выполнение pull request. Командная работа. /Пр/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
	Раздел 4. Отладка программного кода.					
4.1	Типы ошибок: синтаксические, логические, времени выполнения. Понятие отладки. Инструменты отладки в IDE. /Лек/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
4.2	Точки останова (breakpoints), пошаговое выполнение (step over, step into, step out). Просмотр и изменение переменных в процессе отладки. /Лек/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
4.3	Запуск отладчика в IDE, установка точек останова. Пошаговое выполнение	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	

	программы с анализом изменения переменных. Поиск и исправление логической ошибки в небольшом фрагменте кода. Обработка исключений (try-catch-finally) с выводом стека вызовов. /Пр/					
	Запуск отладчика в IDE, установка точек останова. Пошаговое выполнение программы с анализом изменения переменных. Поиск и исправление логической ошибки в небольшом фрагменте кода. Обработка исключений (try-catch-finally) с выводом стека вызовов. /Пр/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
4.4	Запуск отладчика в IDE, установка точек останова. Пошаговое выполнение программы с анализом изменения переменных. Поиск и исправление логической ошибки в небольшом фрагменте кода. Обработка исключений (try-catch-finally) с выводом стека вызовов. /Ср/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
4.5	Запуск отладчика в IDE, установка точек останова. Пошаговое выполнение программы с анализом изменения переменных. Поиск и исправление логической ошибки в небольшом фрагменте кода. Обработка исключений (try-catch-finally) с выводом стека вызовов. /Ср/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
4.6	Запуск отладчика в IDE, установка точек останова. Пошаговое выполнение программы с анализом изменения переменных. Поиск и исправление логической ошибки в небольшом фрагменте кода. Обработка исключений (try-catch-finally) с выводом стека вызовов. /Ср/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
4.7	Логирование: уровни логирования, настройка логов в приложении. Использование assert,	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	

	трассировка стека. /Лек/					
4.8	Настройка логирования в консоль и в файл. Анализ дампа памяти или логов при падении приложения. Использование условных точек останова. Отладка многопоточного приложения. /Пр/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
4.9	Настройка логирования в консоль и в файл. Анализ дампа памяти или логов при падении приложения. Использование условных точек останова. Отладка многопоточного приложения. /Пр/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
4.10	Настройка логирования в консоль и в файл. Анализ дампа памяти или логов при падении приложения. Использование условных точек останова. Отладка многопоточного приложения. /Ср/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
4.11	Настройка логирования в консоль и в файл. Анализ дампа памяти или логов при падении приложения. Использование условных точек останова. Отладка многопоточного приложения. /Ср/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
4.12	Настройка логирования в консоль и в файл. Анализ дампа памяти или логов при падении приложения. Использование условных точек останова. Отладка многопоточного приложения. /Ср/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
4.13	Применение профилировщика для поиска узких мест. Исправление ошибок в чужом коде. Написание юнит- тестов для проверки корректности функций. Интеграция логирования в приложение для отслеживания действий пользователя. /Пр/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
4.14	Применение профилировщика для поиска узких мест. Исправление ошибок в	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	

	чужом коде. Написание юнит- тестов для проверки корректности функций. Интеграция логирования в приложение для отслеживания действий пользователя. /Пр/					
4.15	Применение профилировщика для поиска узких мест. Исправление ошибок в чужом коде. Написание юнит- тестов для проверки корректности функций. Интеграция логирования в приложение для отслеживания действий пользователя. /Ср/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
4.16	Применение профилировщика для поиска узких мест. Исправление ошибок в чужом коде. Написание юнит- тестов для проверки корректности функций. Интеграция логирования в приложение для отслеживания действий пользователя. /Ср/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
4.17	Применение профилировщика для поиска узких мест. Исправление ошибок в чужом коде. Написание юнит- тестов для проверки корректности функций. Интеграция логирования в приложение для отслеживания действий пользователя. /Ср/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
	Раздел 5. Оформление программного кода в соответствии с установленными требованиями.					
5.1	Стандарты оформления кода: PEP 8 (Python), Code Conventions for Java, .NET Naming Guidelines. Правила именования переменных, функций, классов, констант. Форматирование: отступы, пробелы, максимальная длина строки. Документирование кода: комментарии, docstring,	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	

	XML- комментарии. Инструменты автоматического форматирования и проверки стиля (black, flake8, checkstyle). Создание README.md, лицензии, CHANGELOG.md. /Лек/					
5.2	Стандарты оформления кода: PEP 8 (Python), Code Conventions for Java, .NET Naming Guidelines. Правила именования переменных, функций, классов, констант. Форматирование: отступы, пробелы, максимальная длина строки. Документирование кода: комментарии, docstring, XML- комментарии. Инструменты автоматического форматирования и проверки стиля (black, flake8, checkstyle). Создание README.md, лицензии, CHANGELOG.md. /Лек/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
5.3	Анализ «плохого» кода (неправильные имена, отсутствие отступов, магические числа). Переписать по стандарту. Применение автоформаттера к своему коду. /Пр/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
5.4	Написание docstring для функции. Генерация документации из docstring. /Пр/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
5.5	Написание docstring для функции. Генерация документации из docstring. /Ср/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
5.6	Написание docstring для функции. Генерация документации из docstring. /Ср/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
5.7	Проверка кода с помощью линтера. Исправление замечаний. Оформление README.md для проекта. /Пр/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
5.8	Создание файла .gitignore с правилами для выбранной платформы. Добавление лицензии (MIT, GPL,	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	

	Apache) в репозиторий. /Пр/					
5.9	Создание файла .gitignore с правилами для выбранной платформы. Добавление лицензии (MIT, GPL, Apache) в репозиторий. /Ср/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
5.10	Создание файла .gitignore с правилами для выбранной платформы. Добавление лицензии (MIT, GPL, Apache) в репозиторий. /Ср/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
5.11	Создание файла .gitignore с правилами для выбранной платформы. Добавление лицензии (MIT, GPL, Apache) в репозиторий. /Ср/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
5.12	Оформление кода с комментариями на уровне модуля и класса. Рефакторинг с сохранением функциональности. /Пр/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
5.13	Оформление кода с комментариями на уровне модуля и класса. Рефакторинг с сохранением функциональности. /Ср/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
5.14	Оформление кода с комментариями на уровне модуля и класса. Рефакторинг с сохранением функциональности. /Ср/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
	Раздел 6. Комплексная разработка.					
6.1	Этапы разработки программного проекта. Сборка и упаковка приложения (exe, jar, wheel, Docker). Основы работы с Docker. /Лек/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
6.2	Релиз. Подготовка к релизу. Типичные ошибки при релизе проектов. /Лек/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
6.3	Составление технического задания на проект. Проектирование базы данных (ER-диаграмма, схема). /Пр/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
6.4	Разработка приложения с использованием всех предыдущих навыков. Отладка и тестирование приложения. /Лек/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
6.5	Разработка приложения с использованием всех предыдущих навыков.	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	

	Отладка и тестирование приложения. /Ср/					
6.6	Разработка приложения с использованием всех предыдущих навыков. Отладка и тестирование приложения. /Ср/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
6.7	Разработка приложения с использованием всех предыдущих навыков. Отладка и тестирование приложения. /Ср/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
6.8	Подготовка репозитория (GitHub/GitLab): README, документация, инструкция по запуску. Сборка исполняемого файла или настройка Docker-контейнера. /Пр/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
6.9	Подготовка репозитория (GitHub/GitLab): README, документация, инструкция по запуску. Сборка исполняемого файла или настройка Docker-контейнера. /Пр/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
6.10	Рецензирование проекта коллег (peer review). /Пр/	5	2	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	
6.11	Экзамен. /Экзамен/	5	6	ПК 3.1. ПК 3.2. ПК 3.3.	Л1.1 Л1.2 Л1.3 Л2.1 Л2.2 Э1 Э2	

5. ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

5.1. Фонд оценочных средств для проведения промежуточной аттестации

Промежуточная аттестация проходит в форме экзамена. Перечень вопросов к экзамену:

1. Что такое формализация задачи Приведите пример формализации.
2. Какие элементы обязательно должны быть выделены при формализации задачи
3. Что такое входные и выходные данные задачи Приведите пример.
4. Какие виды ограничений могут присутствовать в постановке задачи
5. Что такое алгоритм Перечислите основные свойства алгоритма.
6. Какие способы записи алгоритмов вы знаете
7. Что такое блок-схема алгоритма Какие основные блоки используются
8. Какие алгоритмические конструкции являются базовыми (структурное программирование)
9. Чем отличается цикл с предусловием от цикла с постусловием
10. Что такое сложность алгоритма Что означает O-нотация (Big O notation)
11. Оцените временную сложность пузырьковой сортировки в худшем случае.
12. Что такое рекурсия Приведите пример рекурсивного алгоритма.
13. Чем отличается итеративный алгоритм от рекурсивного
14. Как построить блок-схему для алгоритма с вложенными условиями
15. Какие методы формализации используются для сложных систем
16. Назовите основные типы данных в выбранном языке программирования.
17. Чем отличается компиляция от интерпретации
18. Что такое переменная Правила именования переменных.

19. Какие операторы ветвления существуют Приведите синтаксис.
20. Какие виды циклов есть в языке Когда какой лучше использовать
21. Что такое функция Как передать аргументы в функцию
22. Чем отличается передача по значению от передачи по ссылке
23. Что такое область видимости переменной
24. Что такое рекурсивная функция Условие выхода из рекурсии.
25. Что такое структура данных Назовите встроенные структуры.
26. В чём разница между массивом и списком
27. Что такое класс и объект Основные принципы ООП.
28. Что такое инкапсуляция Приведите пример.
29. Что такое наследование Зачем оно нужно
30. Что такое полиморфиз Пример полиморфизма.
31. Что такое SQL Назначение языка.
32. Какие основные операторы SQL вы знаете
33. Какой оператор используется для выборки данных
34. Как выполнить выборку с условием
35. Что такое JOIN Какие типы JOIN существуют
36. Как вставить новую запись в таблицу
37. Как обновить данные в таблице
38. Как удалить запись Чем DELETE отличается от TRUNCATE
39. Что такое первичный ключ Внешний ключ
40. Как подключиться к базе данных из приложения Напишите пример строки подключения.
41. Что такое параметризованный запрос Почему он безопаснее конкатенации строк
42. Что такое ORM Назовите известные ORM.
43. Преимущества и недостатки использования ORM.
44. Как выполнить запрос к БД через SQLAlchemy (или другой ORM) без написания сырого SQL
45. Разработайте простую схему БД для «Список задач» (таблицы, поля, связи).
46. Что такое система контроля версий Для чего она нужна
47. Что такое Git Основные понятия.
48. Как создать локальный репозиторий Git
49. Какие состояния могут быть у файла в Git
50. Какие команды используются для добавления файлов в индекс и фиксации изменений
51. Что такое коммит Что содержит коммит
52. Как просмотреть историю коммитов
53. Как отменить последний коммит (не удаляя изменения)
54. Что такое ветка (branch) Как создать новую ветку
55. Как переключиться на другую ветку
56. Что такое слияние (merge) Типы слияния.
57. Что такое конфликт при слиянии Как его разрешить
58. Что такое удалённый репозиторий? Как связать локальный репозиторий с удалённым
59. Какие команды используются для отправки изменений на удалённый репозиторий и получения изменений
60. Что такое pull request Как он работает на GitHub/GitLab
61. Что такое .gitignore Приведите пример содержимого.
62. Что такое Git Flow Назовите основные ветки.
63. Чем отличается git merge от git rebase
64. Как просмотреть изменения, сделанные в рабочей директории (diff)
65. Как вернуть файл к состоянию последнего коммита
66. Какие типы ошибок в программах вы знаете
67. Что такое отладка (debugging) Какие инструменты используются
68. Что такое точка останова (breakpoint)
69. Какие действия можно выполнять при остановке на точке останова
70. Что такое пошаговое выполнение
71. Как просмотреть значение переменной во время отладки

72. Что такое условная точка останова Как её установить
73. Что такое логирование Уровни логирования.
74. Какая библиотека для логирования используется в Python
75. Как настроить вывод логов в файл и в консоль одновременно
76. Что такое исключение Как обработать исключение в коде
77. Что такое трассировка стека (stack trace) Как её прочитать
78. Что такое профилирование (profiling) Для чего используется
79. Как найти утечку памяти в приложении
80. Какие инструменты отладки существуют в выбранной вами IDE
81. Что такое стандарт оформления кода Приведите примеры.
82. Какие правила именования переменных и функций существуют в PEP 8
83. Как правильно именовать классы в Python. В Java. В C#
84. Что такое магические числа Как с ними бороться
85. Какая максимальная длина строки рекомендуется в PEP 8
86. Как оформляются отступы в коде
87. Что такое docstring Как её оформить и для чего
88. Как сгенерировать документацию из docstring
89. Какие инструменты автоматического форматирования кода вы знаете
90. Что такое линтер (linter) Назовите популярные линтеры.
91. Как настроить проверку кода с помощью flake8 или pylint
92. Что должно быть в файле README проекта
93. Как оформить лицензию в проекте Какие лицензии бывают
94. Что такое CHANGELOG.md Какая информация туда заносится
95. Как правильно комментировать сложные участки кода
96. Перечислите основные этапы разработки программного проекта.
97. Что должно содержать техническое задание на разработку
98. Как подготовить презентацию проекта для релиза
99. Какие разделы должны быть в отчёте по проекту
100. Какие критерии используются для оценки комплексного проекта

Критерии оценивания:

5 баллов выставляется студентам за полный и правильный ответ на все вопросы билета с логическим обоснованием аргументов, в ответе нет ошибок.

4 балла выставляется студентам, если вопросы билета раскрыты полностью, но обоснования доказательства недостаточны, при этом допущены две-три несущественные ошибки, исправленные по требованию преподавателя.

3 балла ставится студентам за правильный ответ на вопросы билета, при этом допущено одной ошибки по изложению фактов или более двух-трёхнедочетов в ответе.

2 балла ставится студентам, если допущены существенные ошибки, показавшие, что обучающийся не обладает обязательными умениями по данной теме в полной мере.

5.2. Фонд оценочных средств для проведения текущего контроля

Представлен в Приложении 1 к рабочей программе дисциплины.

6. УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ

6.1. Рекомендуемая литература

6.1.1. Основная литература

	Авторы,	Заглавие	Издательство,	Колич-во
Л1.1	Паронджанов, В. Д.	Алгоритмические языки и программирование: ДРАКОН : учебник для СПО: текст электронный	Москва: Юрайт, 2026	https://urait.ru/bcode/588183 неограниченный доступ зарегистрированным пользователям
Л1.2	Трофимов, В. В.	Основы алгоритмизации и программирования: учебник для СПО: текст электронный	Москва: Юрайт, 2026	https://urait.ru/bcode/563861 неограниченный доступ зарегистрированным пользователям
Л1.3	Казарин, О. В.	Программно-аппаратные средства защиты	Москва: Юрайт, 2026	https://urait.ru/bcode/588246 неограниченный доступ

		информации. Защита программного обеспечения : учебник и практикум для СПО: текст электронный		зарегистрированным пользователям
6.1.2. Дополнительная литература				
	Авторы, составители	Заглавие	Издательство, год	Колич-во
Л2.1	Станкевич, Л. А.	Интеллектуальные системы и технологии : учебник и практикум для СПО: текст электронный	Москва: Юрайт, 2026	https://urait.ru/bcode/587749 неограниченный доступ зарегистрированным пользователям
Л2.2	Черпаков, И. В.	Алгоритмизация и программирование в Python : учебник для СПО: текст электронный	Москва: Юрайт, 2026	https://urait.ru/bcode/582413 неограниченный доступ зарегистрированным пользователям
6.2. Перечень ресурсов информационно-телекоммуникационной сети "Интернет"				
Э1	Открытая научная электронная библиотека https://cyberleninka.ru/			
Э2	Общероссийский портал для поиска научной информации по математике, физике, информационным технологиям и смежным наукам https://www.mathnet.ru/			
6.3. Перечень программного обеспечения				
6.3.1	Офисный пакет – LibreOffice.			
6.3.2	Веб-браузер – Chromium.			
6.4 Перечень информационных справочных систем				
6.4.1	ИСС «КонсультантПлюс»			
6.4.2	ИСС «Гарант»			

7. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ (МОДУЛЯ)	
7.1	Помещения для проведения всех видов работ, предусмотренных учебным планом, укомплектованы необходимой специализированной учебной мебелью и техническими средствами обучения.

8. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ
Методические указания по освоению дисциплины представлены в Приложении 2 к рабочей программе МДК

Приложение 1

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

МДК 03.01 Технологии выполнения работ по должности «Программист»

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

1.1 Показатели и критерии оценивания компетенций:

УУД, составляющие компетенцию	Показатели оценивания	Критерии оценивания	Средства оценивания
ПК 3.1: Проверка работоспособности и рефакторинг кода программного обеспечения			
Знать: - Стандарты кодирования для целевого языка программирования и среды разработки. - Методы моделирования программного обеспечения с применением объектных моделей. - Подходы к разработке процедур проверки работоспособности и измерения характеристик программного обеспечения. - Правила работы с открытыми библиотеками и управление ими.	**Получение систематических знаний** о стандартах кодирования для целевого языка программирования и среды разработки; методах моделирования программного обеспечения с применением объектных моделей; подходах к разработке процедур проверки работоспособности и измерения характеристик программного обеспечения; правилах работы с открытыми библиотеками и управлении ими.	**Уровень знаний:** Знает на уровне понимания и воспроизведения: перечисляет стандарты, объясняет суть объектного моделирования, описывает подходы к тестированию и измерению характеристик, называет правила подключения и использования открытых библиотек.	**Т (1-110)** **ПЗ (1-25)**
Уметь: - Разрабатывать процедуры проверки работоспособности, которые корректно функционируют в полном соответствии с техническим заданием и охватывают все функциональные требования. - Реализовывать процедуры измерения характеристик программного обеспечения в целевой среде программирования. - Выполнять предварительное	**Сформировать систематическое умение** разрабатывать процедуры проверки работоспособности, корректно функционирующие в полном соответствии с техническим заданием и охватывающие все функциональные требования; реализовывать процедуры измерения характеристик программного обеспечения в целевой среде программирования;	**Уровень умения** Умеет самостоятельно применять полученные знания для создания тестовых процедур, измерительных модулей, объектных моделей и интеграции внешних библиотек при решении типовых профессиональных задач.	**Т (1-110)** **ПЗ (1-25)**

моделирование приложения, применяя объектные модели для описания структуры и поведения системы до написания кода. - Подключать и использовать открытые библиотеки для решения прикладных задач приложения.	выполнять предварительное моделирование приложения, применяя объектные модели для описания структуры и поведения системы до написания кода; подключать и использовать открытые библиотеки для решения прикладных задач.		
Владеть: - Созданием объектных моделей для проектирования веб-приложения по предложенному техническому заданию. - Разработкой программного кода, оформленный полном соответствии со стандартами кодирования. - Интеграцией процедуры проверки и измерения характеристик в приложение с возможностью их автоматического или полуавтоматического запуска. - Анализом результатов проверки работоспособности и измерений характеристик на предмет соответствия техническому заданию и выявления узких мест производительности.	**Сформировать систематическое владение** созданием объектных моделей для проектирования веб-приложения по предложенному техническому заданию; разработкой программного кода, оформленного в полном соответствии со стандартами кодирования; интеграцией процедур проверки и измерения характеристик в приложение с возможностью их автоматического или полуавтоматического запуска; анализом результатов проверки работоспособности и измерений характеристик на предмет соответствия техническому заданию и выявления узких мест производительности.	**Уровень владения:** Владеет устойчивыми навыками практического применения объектного моделирования, стандартов кодирования, тестирования и анализа производительности при разработке и сопровождении программных продуктов.	**Т (1-110)** **ПЗ (1-25)**
ПК 3.2: Интеграция программных модулей и компонентов и проверка работоспособности выпусков программного продукта			
Знать: - Методы и стратегии интеграции программных модулей и критерии выбора подхода в соответствии с техническим заданием. - Стандарты кодирования,	**Получение систематических знаний** о методах и стратегиях интеграции программных модулей и критериях выбора подхода в соответствии с техническим	**Уровень знания:** Знает на уровне понимания и воспроизведения: перечисляет стратегии интеграции, объясняет критерии выбора, описывает стандарты оформления интегрируемого кода,	**Т (1-110)** **ПЗ (1-25)**

регламентирующие оформление кода при интеграции. - Принципы построения объектных моделей для описания взаимодействия модулей, зависимостей и потоков данных. - Технологии создания анимационных эффектов в целевой среде разработки.	заданием; стандартах кодирования, регламентирующих оформление кода при интеграции; принципах построения объектных моделей для описания взаимодействия модулей, зависимостей и потоков данных; технологиях создания анимационных эффектов в целевой среде разработки.	принципы объектного моделирования взаимодействий и технологии анимации.	
Уметь: - Разрабатывать процедуры интеграции программных модулей, которые корректно функционируют в полном соответствии с техническим заданием. - Выполнять предварительное моделирование приложения с применением объектных моделей для проектирования структуры и взаимодействия интегрируемых модулей. - Реализовывать анимационные эффекты, интегрируя их в общую архитектуру приложения без нарушения логики работы модулей. - Оформлять программный код в полном соответствии со стандартами кодирования.	**Сформировать систематическое умение** разрабатывать процедуры интеграции программных модулей, которые корректно функционируют в полном соответствии с техническим заданием; выполнять предварительное моделирование приложения с применением объектных моделей для проектирования структуры и взаимодействия интегрируемых модулей; реализовывать анимационные эффекты, интегрируя их в общую архитектуру приложения без нарушения логики работы модулей; оформлять программный код в полном соответствии со стандартами кодирования.	**Уровень умения:** Умеет самостоятельно выбирать и применять подходящие стратегии интеграции, проектировать взаимодействие модулей с помощью объектных моделей, добавлять анимационные эффекты и оформлять код в соответствии со стандартами.	**Т (1-110)** **ПЗ (1-25)**
Владеть: - Построением и использованием объектных моделей для планирования интеграции модулей до начала программирования. - Созданием процедур интеграции,	**Сформировать систематическое владение** построением и использованием объектных моделей для планирования интеграции модулей до начала программирования; созданием процедур	**Уровень владения** Владеет устойчивыми навыками проектирования интеграционных решений, объектного моделирования, реализации анимации и отладки взаимодействия модулей при сборке	**Т (1-110)** **ПЗ (1-25)**

обеспечивающих корректную сборку, инициализацию и взаимодействие всех модулей системы согласно техническому заданию. - Применением анимационных эффектов как составной части интегрированного приложения с учётом производительности и совместимости модулей. - Проверкой корректности работы процедур интеграции, включая выявление и устранение конфликтов интерфейсов, типов данных и последовательностей вызовов.	интеграции, обеспечивающих корректную сборку, инициализацию и взаимодействие всех модулей системы согласно техническому заданию; применением анимационных эффектов как составной части интегрированного приложения с учётом производительности и совместимости модулей; проверкой корректности работы процедур интеграции, включая выявление и устранение конфликтов интерфейсов, типов данных и последовательностей вызовов.	сложных программных систем.	
ПК 3.3 Разработка и отладка программного кода			
Знать: - Стандарты оформления технического задания и требования к его структуре. - Методы формализации поставленных задач. - Базовые алгоритмические конструкции и принципы построения эффективных алгоритмов для последующей реализации в программном коде. - Правила логически и технически грамотного изложения разделов технического задания.	**Получение систематических знаний** стандартах оформления технического задания и требованиях к его структуре; методах формализации поставленных задач; базовых алгоритмических конструкциях и принципах построения эффективных алгоритмов для последующей реализации в программном коде; правилах логически и технически грамотного изложения разделов технического задания.	**Уровень знания:** Знает на уровне понимания и воспроизведения: перечисляет требования к ТЗ, объясняет методы формализации, описывает алгоритмические конструкции и правила изложения разделов ТЗ.	**Т (1-110)** **ПЗ (1-25)**
Уметь: - Выполнять полную формализацию задачи, выделяя все сущности, ограничения и критерии успешности. - Разрабатывать детальные алгоритмы и	**Сформировать систематическое умение** выполнять полную формализацию задачи, выделяя все сущности, ограничения и критерии успешности;	**Уровень умения:** Умеет самостоятельно формализовать задачи любой сложности, создавать алгоритмы и блок-схемы, разрабатывать и оформлять ТЗ в	**Т (1-110)** **ПЗ (1-25)**

блок-схемы для каждого функционального требования, пригодные для прямой трансляции в программный код. - Оформлять техническое задание в полном соответствии с установленными стандартами. - Логически выстраивать разделы технического задания, обеспечивая непротиворечивость, полноту и однозначность трактовки требований.	разрабатывать детальные алгоритмы и блок-схемы для каждого функционального требования, пригодные для прямой трансляции в программный код; оформлять техническое задание в полном соответствии с установленными стандартами; логически выстраивать разделы технического задания, обеспечивая непротиворечивость, полноту и однозначность трактовки требований.	соответствии со стандартами, а также проверять его качество.	
Владеть: - Навыками применением стандартизированных шаблонов и средств оформления технического задания с соблюдением всех требований к форме и содержанию. - Алгоритмизацией и структурированием задач любой сложности для последующей разработки программного кода без пробелов в логике. - Технически грамотной формулировкой требований к программному продукту. - Критической проверкой технического задания на соответствие стандартам, полноту, однозначность и технологичность реализации.	**Сформировать систематическое владение** навыками применения стандартизированных шаблонов и средств оформления технического задания с соблюдением всех требований к форме и содержанию; алгоритмизацией и структурированием задач любой сложности для последующей разработки программного кода без пробелов в логике; технически грамотной формулировкой требований к программному продукту; критической проверкой технического задания на соответствие стандартам, полноту, однозначность и технологичность реализации.	**Уровень владения:** Владеет устойчивыми навыками формализации, алгоритмизации, документирования и экспертизы технических заданий, а также написания кода на основе разработанных алгоритмов.	**Т (1-110)** **ПЗ (1-25)**

Т – тестовые задания, ПЗ – практические задания.

2 Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций в процессе освоения образовательной программы

Тестовые задания:

1. Что такое формализация задачи?
    a) Представление задачи в виде строгих математических или логических моделей
    b) Написание программного кода
    c) Тестирование программы
    d) Оформление документации
2. Какие элементы обязательно должны быть выделены при формализации задачи?
    a) Входные данные, выходные данные, ограничения, допущения
    b) Только входные данные
    c) Только выходные данные
    d) Имена переменных
3. Что такое входные данные задачи?
    a) Данные, которые поступают в систему для обработки
    b) Результат решения задачи
    c) Алгоритм решения
    d) Документация
4. Какой вид ограничений определяет допустимые значения входных данных?
    a) Ограничения на входные данные
    b) Временные ограничения
    c) Ресурсные ограничения
    d) Ограничения на выходные данные
5. Что такое алгоритм?
    a) Точная конечная последовательность действий, приводящая от исходных данных к результату
    b) Любая последовательность команд
    c) Описание задачи
    d) Программа на языке высокого уровня
6. Какое свойство алгоритма означает, что он должен приводить к результату за конечное число шагов?
    a) Определённость
    b) Массовость
    c) Результативность
    d) Дискретность
7. Какой способ записи алгоритма использует геометрические фигуры?
    a) Словесный
    b) Блок-схема
    c) Псевдокод
    d) Программа
8. Какая фигура в блок-схеме обозначает начало или конец алгоритма?
    a) Прямоугольник
    b) Овал
    c) Ромб
    d) Параллелограмм
9. Какая конструкция реализует выбор одного из путей в зависимости от условия?
    a) Ветвление
    b) Цикл
    c) Следование
    d) Рекурсия
10. Чем отличается цикл с предусловием от цикла с постусловием?
    a) В цикле с предусловием условие проверяется до выполнения тела цикла
    b) В цикле с предусловием тело выполняется всегда хотя бы один раз
    c) Они ничем не отличаются

d) Цикл с предусловием выполняется бесконечно

11. Что такое сложность алгоритма?
    a) Характеристика, показывающая зависимость времени выполнения от объёма входных данных
    b) Количество строк кода
    c) Количество переменных
    d) Глубина вложенности циклов
12. Что означает O(n) (линейная сложность)?
    a) Время выполнения пропорционально размеру входных данных
    b) Время выполнения не зависит от размера данных
    c) Время выполнения растёт квадратично
    d) Время выполнения растёт экспоненциально
13. Какая временная сложность у пузырьковой сортировки в худшем случае?
    a) O(log n)
    b) $O(n^2)$
    c) O(n)
    d) O(1)
14. Что такое рекурсия?
    a) Вызов функцией самой себя
    b) Циклическое выполнение блока кода
    c) Условный оператор
    d) Объявление переменной
15. Что обязательно должно быть в рекурсивной функции для предотвращения бесконечного выполнения?
    a) Условие выхода
    b) Глобальная переменная
    c) Рекурсивный вызов
    d) Цикл for
16. Какой метод формализации используется для моделирования бизнес-процессов?
    a) Блок-схема
    b) IDEF0
    c) Псевдокод
    d) Асимптотический анализ
17. Что такое псевдокод?
    a) Промежуточный язык между естественным языком и языком программирования
    b) Официальный язык программирования
    c) Машинный код
    d) Байт-код
18. Какой оператор в псевдокоде обычно обозначает ветвление?
    a) for
    b) if-else
    c) while
    d) repeat
19. Что такое инвариант цикла?
    a) Условие, которое остаётся истинным на каждой итерации цикла
    b) Количество итераций цикла
    c) Переменная цикла
    d) Тело цикла
20. Какой подход к разработке алгоритмов использует принцип «разделяй и властвуй»?
    a) Разбиение задачи на подзадачи и рекурсивное решение
    b) Последовательный перебор всех вариантов
    c) Жадный алгоритм
    d) Динамическое программирование
21. Какой тип данных в Python используется для хранения целых чисел?
    a) float
    b) int
    c) str

d) bool

22. Чем отличается компиляция от интерпретации?
    a) Компиляция преобразует код в машинный заранее, интерпретация выполняет построчно
    b) Интерпретация создаёт исполняемый файл
    c) Компиляция выполняется на сервере
    d) Ничем
23. Какое ключевое слово в Python используется для определения функции?
    a) func
    b) define
    c) def
    d) function
24. Какой оператор ветвления в C# позволяет проверить несколько условий?
    a) if-else
    b) for
    c) while
    d) switch
25. Какой цикл выполняется, пока условие истинно, с проверкой в начале?
    a) while
    b) do-while
    c) for
    d) foreach
26. Что такое локальная переменная?
    a) Переменная, объявленная внутри функции и доступная только в ней
    b) Переменная, объявленная вне всех функций
    c) Переменная, хранящаяся в базе данных
    d) Константа
27. Что такое класс в ООП?
    a) Шаблон для создания объектов, описывающий их состояние и поведение
    b) Конкретный экземпляр объекта
    c) Функция
    d) Массив
28. Какой принцип ООП означает сокрытие внутренних деталей реализации?
    a) Наследование
    b) Полиморфизм
    c) Инкапсуляция
    d) Абстракция
29. Что такое наследование в ООП?
    a) Создание нового класса на основе существующего с добавлением новых свойств
    b) Использование одного интерфейса для разных типов
    c) Скрытие данных
    d) Перегрузка операторов
30. Какой оператор SQL используется для выборки данных?
    a) INSERT
    b) UPDATE
    c) SELECT
    d) DELETE
31. Какой оператор SQL добавляет новую запись в таблицу?
    a) INSERT INTO
    b) ADD
    c) CREATE
    d) UPDATE
32. Какой оператор SQL удаляет записи из таблицы?
    a) DROP
    b) DELETE FROM
    c) REMOVE
    d) TRUNCATE
33. Чем DELETE отличается от TRUNCATE?

a) DELETE удаляет записи с условием и логируется, TRUNCATE быстро удаляет все записи
b) DELETE быстрее
c) TRUNCATE можно использовать с WHERE
d) Ничем

34. Что такое первичный ключ (PRIMARY KEY)?
a) Поле, уникально идентифицирующее каждую запись в таблице
b) Поле, ссылающееся на другую таблицу
c) Индекс
d) Тип данных

35. Какой тип JOIN возвращает только совпадающие строки из обеих таблиц?
a) INNER JOIN
b) LEFT JOIN
c) RIGHT JOIN
d) FULL OUTER JOIN

36. Какой метод в Python SQLite выполняет SQL-запрос?
a) execute()
b) query()
c) run()
d) select()

37. Что такое параметризованный запрос?
a) Запрос с плейсхолдерами для подстановки значений, предотвращающий SQL-инъекции
b) Запрос с фиксированными значениями
c) Запрос на хранимой процедуре
d) Запрос без условий

38. Что такое ORM (Object-Relational Mapping)?
a) Технология, связывающая объекты программы с таблицами базы данных
b) Язык запросов
c) Тип базы данных
d) Среда разработки

39. Какой фреймворк является ORM для Python?
a) SQLAlchemy
b) Django
c) Flask
d) NumPy

40. Какой метод в SQLAlchemy соответствует SELECT?
a) query()
b) insert()
c) update()
d) delete()

41. Что такое миграция базы данных?
a) Управление изменениями схемы БД с возможностью отката
b) Копирование данных
c) Резервное копирование
d) Индексирование

42. Какой тип данных SQL хранит строку переменной длины?
a) CHAR
b) VARCHAR
c) INT
d) DATE

43. Какой оператор SQL изменяет существующие записи?
a) UPDATE
b) ALTER
c) MODIFY
d) CHANGE

44. Что такое внешний ключ (FOREIGN KEY)?
a) Поле, ссылающееся на первичный ключ другой таблицы, обеспечивающее целостность связи

b) Первичный ключ
c) Индекс
d) Уникальное поле

45. Какой метод в Python библиотеки sqlite3 используется для выполнения нескольких запросов в транзакции?
    a) commit() и rollback()
    b) execute()
    c) fetchall()
    d) close()
46. Какой язык используется для написания хранимых процедур в PostgreSQL?
    a) PL/pgSQL
    b) T-SQL
    c) Python
    d) Java
47. Что такое индексы в базах данных?
    a) Структуры для ускорения поиска и сортировки данных
    b) Ограничения целостности
    c) Типы данных
    d) Триггеры
48. Какой тип соединения таблиц возвращает все строки из левой таблицы и совпадающие из правой, а для несовпадающих – NULL?
    a) INNER JOIN
    b) LEFT JOIN
    c) RIGHT JOIN
    d) CROSS JOIN
49. Какой оператор SQL группирует строки с одинаковыми значениями?
    a) GROUP BY
    b) ORDER BY
    c) WHERE
    d) HAVING
50. Какая агрегатная функция SQL подсчитывает количество строк?
    a) SUM
    b) AVG
    c) COUNT
    d) MAX
51. Что такое система контроля версий?
    a) Программа для отслеживания изменений в файлах и совместной работы
    b) База данных
    c) Язык программирования
    d) Операционная система
52. Какая команда Git создаёт локальный репозиторий?
    a) git init
    b) git start
    c) git create
    d) git new
53. Какое состояние файла в Git означает, что файл отслеживается, но ещё не добавлен в индекс?
    a) Staged
    b) Committed
    c) Modified
    d) Untracked
54. Какая команда добавляет изменения в индекс (staging area)?
    a) git add
    b) git commit
    c) git push
    d) git status
55. Что такое коммит (commit)?
    a) Фиксация изменений с сохранением состояния файлов

b) Отправка изменений на сервер
c) Получение изменений с сервера
d) Создание ветки

56. Какой командой просмотреть историю коммитов?
a) git log
b) git status
c) git diff
d) git show

57. Как отменить последний коммит, сохранив изменения в рабочей директории?
a) git reset --hard HEAD~1
b) git reset --soft HEAD~1
c) git revert HEAD
d) git checkout HEAD~1

58. Что такое ветка (branch) в Git?
a) Отдельная линия разработки, позволяющая вести параллельную работу
b) Копия репозитория
c) Тег
d) Удалённый репозиторий

59. Какая команда создаёт новую ветку?
a) git checkout -b / git branch
b) git merge
c) git branch -d
d) git checkout

60. Какая команда переключается на другую ветку?
a) git checkout
b) git switch
c) git branch
d) git merge

61. Что такое слияние (merge) веток?
a) Объединение изменений из одной ветки в другую
b) Удаление ветки
c) Создание новой ветки
d) Отправка на сервер

62. Что такое конфликт при слиянии?
a) Ситуация, когда в разных ветках изменены одни и те же строки файла
b) Ошибка компиляции
c) Проблема с сетью
d) Отсутствие прав доступа

63. Как разрешить конфликт слияния в Git?
a) Вручную отредактировать файл, затем git add и git commit
b) Удалить одну из веток
c) Переустановить Git
d) Игнорировать конфликт

64. Какая команда связывает локальный репозиторий с удалённым?
a) git remote add origin
b) git clone
c) git push
d) git pull

65. Какая команда отправляет локальные коммиты в удалённый репозиторий?
a) git fetch
b) git push
c) git pull
d) git clone

66. Что такое pull request?
a) Запрос на включение изменений из одной ветки в другую в удалённом репозитории
b) Команда git pull
c) Создание коммита

d) Удаление ветки

67. Какой файл указывает Git, какие файлы игнорировать?
    a) .gitignore
    b) .gitconfig
    c) .gitattributes
    d) README.md
68. Что такое Git Flow?
    a) Модель ветвления с основными ветками master, develop, feature, release, hotfix
    b) Графический интерфейс Git
    c) Тип коммита
    d) Удалённый сервер
69. Чем отличается git merge от git rebase?
    a) Rebase переписывает историю, перенося коммиты поверх другой ветки, merge создаёт коммит слияния
    b) Merge переписывает историю
    c) Rebase нельзя отменить
    d) Ничем
70. Какая команда показывает различия между рабочим каталогом и индексом?
    a) git diff
    b) git status
    c) git log -p
    d) git show
71. Какие типы ошибок в программах существуют?
    a) Синтаксические, логические, времени выполнения
    b) Только синтаксические
    c) Только логические
    d) Только компиляции
72. Что такое отладка (debugging)?
    a) Процесс поиска и исправления ошибок в программе
    b) Написание кода
    c) Компиляция
    d) Оптимизация
73. Что такое точка останова (breakpoint)?
    a) Место в коде, где выполнение программы приостанавливается
    b) Конец программы
    c) Ошибка
    d) Переменная
74. Какое действие в отладчике выполняет шаг с заходом внутрь вызываемой функции?
    a) Step Over
    b) Step Into
    c) Step Out
    d) Continue
75. Как просмотреть значение переменной во время остановки в отладчике?
    a) Навести курсор или посмотреть в окне переменных
    b) Запустить программу заново
    c) Перекомпилировать
    d) Изменить код
76. Что такое условная точка останова?
    a) Точка останова, срабатывающая только при выполнении заданного условия
    b) Точка, установленная в условном операторе
    c) Точка, которая всегда срабатывает
    d) Точка после цикла
77. Что такое логирование?
    a) Запись событий программы в журнал
    b) Тестирование
    c) Профилирование
    d) Компиляция

78. Какой уровень логирования означает самую подробную информацию?
    a) ERROR
    b) WARNING
    c) DEBUG
    d) INFO
79. Какой уровень логирования используется для критических ошибок, останавливающих работу?
    a) INFO
    b) WARNING
    c) ERROR
    d) CRITICAL
80. Какая библиотека Python используется для логирования?
    a) logging
    b) log4j
    c) syslog
    d) logger
81. Как обработать исключение в Python?
    a) try-except
    b) try-catch
    c) catch
    d) finally
82. Что такое трассировка стека (stack trace)?
    a) Вывод последовательности вызовов функций, приведших к ошибке
    b) Список переменных
    c) Код ошибки
    d) Имя функции
83. Что такое профилирование (profiling) программы?
    a) Измерение времени выполнения отдельных частей кода, поиск узких мест
    b) Отладка
    c) Логирование
    d) Компиляция
84. Какой инструмент профилирования встроен в Python?
    a) cProfile
    b) profile
    c) pdb
    d) timeit
85. Как найти утечку памяти в приложении?
    a) Использовать профилировщик памяти
    b) Перезапустить приложение
    c) Увеличить память
    d) Никак
86. Какая команда в Python запускает отладчик pdb?
    a) python -m pdb script.py
    b) pdb script.py
    c) debug script.py
    d) python --debug script.py
87. Что такое assert в программировании?
    a) Утверждение, проверяющее условие; при ложности вызывает ошибку
    b) Тип данных
    c) Цикл
    d) Функция
88. Как в Python вывести стек вызовов без остановки программы?
    a) traceback.print_stack()
    b) print(stack)
    c) logging.stack()
    d) sys.stack()
89. Какой метод в C# используется для логирования в файл?
    a) System.Diagnostics.Trace

b) Console.WriteLine
c) Debug.Write
d) File.WriteAllText

90. Что такое юнит-тест (unit test)?
a) Тестирование отдельного модуля (функции, класса)
b) Тестирование всей системы
c) Нагрузочное тестирование
d) Ручное тестирование

91. Какой стандарт оформления кода используется в Python?
a) PEP 8
b) ISO 9001
c) IEEE 802.11
d) ГОСТ 19

92. Какие имена переменных соответствуют стилю snake_case?
a) user_name
b) userName
c) UserName
d) username (тоже, но лучше user_name)

93. Какой стиль именования классов рекомендуется в Python?
a) CamelCase
b) snake_case
c) UPPER_CASE
d) lowercase

94. Что такое магическое число (magic number) в коде?
a) Числовой литерал, значение которого не очевидно без комментария
b) Число Пи
c) Случайное число
d) Константа

95. Какая максимальная длина строки рекомендуется в PEP 8?
a) 79 символов (или 99 для соглашений)
b) 120
c) 50
d) Не ограничена

96. Чем следует отделять блоки кода в Python?
a) Отступами (пробелами)
b) Фигурными скобками
c) Ключевыми словами begin/end
d) Точками с запятой

97. Что такое docstring в Python?
a) Строка документации функции или класса, заключённая в тройные кавычки
b) Комментарий
c) Имя переменной
d) Тип данных

98. Как сгенерировать документацию из docstring?
a) Sphinx, pydoc
b) Doxygen
c) Javadoc
d) Swagger

99. Какой инструмент автоматически форматирует Python-код в соответствии с PEP 8?
a) black
b) flake8
c) pylint
d) autopep8 (тоже верно, но black популярнее)

100. Что такое линтер (linter)?
a) Инструмент статического анализа кода, проверяющий стиль и потенциальные ошибки
b) Компилятор
c) Отладчик

d) Интерпретатор

101. Какой инструмент проверяет код на соответствие PEP 8?
a) flake8 или pylint
b) black
c) pytest
d) mypy

102. Что должно быть в файле README проекта?
a) Описание проекта, установка, запуск, примеры использования
b) Только код
c) Логи
d) Конфигурация

103. Какой файл содержит историю изменений версий?
a) CHANGELOG.md
b) README.md
c) LICENSE
d) CONTRIBUTING.md

104. Какая лицензия позволяет свободное использование, модификацию и распространение при условии сохранения уведомления об авторстве?
a) MIT License
b) GPL
c) Apache 2.0
d) Все перечисленные

105. Как правильно комментировать сложный участок кода?
a) Объяснить «почему», а не «что»
b) Комментировать каждую строку
c) Не комментировать
d) Комментировать только на английском

106. Какие этапы включает разработка программного проекта?
a) Анализ, проектирование, реализация, тестирование, внедрение, сопровождение
b) Только написание кода
c) Только тестирование
d) Только анализ

107. Что должно содержать техническое задание на разработку?
a) Цели, требования к функционалу, ограничения, сроки
b) Только список функций
c) Исходный код
d) Инструкцию по установке

108. Какой инструмент используется для контейнеризации приложений?
a) Docker
b) Git
c) Jenkins
d) VirtualBox

109. Какой инструмент автоматизации сборки часто используется в CI/CD?
a) Jenkins или GitLab CI
b) Visual Studio
c) Notepad++
d) Photoshop

110. Какие критерии используются для оценки комплексного проекта?
a) Полнота реализации, качество кода, работа с Git, документация, защита
b) Только скорость выполнения
c) Только количество строк
d) Только дизайн интерфейса

Критерии оценивания:

- 5 баллов выставляется, если правильные ответы даны на 85-100% тестовых заданий
- 4 балла выставляется студенту, если правильные ответы даны на 65-84% тестовых заданий

- 3 балла выставляется студенту, если правильные ответы даны на 50-64% тестовых заданий

- 2 балла выставляется студенту, если правильные ответы даны на менее 50% тестовых заданий

Практические задания:

Раздел 1. Формализация и алгоритмизация задач

Задание 1

Тема: Формализация задачи и построение блок-схемы

Условие: Дано текстовое описание задачи: «Программа запрашивает у пользователя три числа (длины сторон треугольника). Необходимо определить, существует ли треугольник с такими сторонами, и если да – вычислить его площадь по формуле Герона. Входные данные – три положительных числа. Ограничения: каждое число должно быть больше 0, сумма любых двух сторон больше третьей».

Задание:

1. Выделить входные, выходные данные, ограничения и допущения.
2. Построить блок-схему алгоритма решения.
3. Записать алгоритм на псевдокоде.
4. Оценить временную сложность алгоритма.

Задание 2

Тема: Рекурсивные алгоритмы

Условие: Написать алгоритм вычисления n-го числа Фибоначчи двумя способами: итеративным и рекурсивным.

Задание:

1. Построить блок-схемы для обоих алгоритмов.
2. Сравнить временную сложность.
3. Реализовать оба варианта на выбранном языке программирования (Python/C#/Java).
4. Измерить время выполнения для n=10, 20, 30, сделать вывод.

Задание 3

Тема: Сортировка и поиск

Условие: Дан массив целых чисел. Реализовать алгоритм сортировки выбором и бинарный поиск элемента.

Задание:

1. Формализовать задачу сортировки и поиска.
2. Построить блок-схему алгоритма бинарного поиска.
3. Оценить сложность обоих алгоритмов.
4. Реализовать программу, которая генерирует случайный массив, сортирует его и выполняет поиск заданного числа.

Задание 4

Тема: Формализация задачи с несколькими ограничениями

Условие: Разработать алгоритм для задачи: «Программа получает список товаров (название, цена, количество). Рассчитать общую стоимость, найти самый дорогой товар, вывести товары с ценой выше средней».

Задание:

1. Выделить все сущности, входные/выходные данные, ограничения.
2. Построить блок-схему.
3. Записать алгоритм с использованием структур данных (массив/список).
4. Оценить сложность.

Раздел 2. Написание программного кода и работа с базами данных

Задание 5

Тема: Консольное приложение с ветвлением и циклами

Условие: Написать программу «Калькулятор» с меню: сложение, вычитание, умножение, деление, возведение в степень, выход. Программа должна запрашивать два числа, выполнять операцию и выводить результат. Обрабатывать деление на ноль.

Задание:

1. Реализовать программу на Python (или C#).
2. Использовать функции для каждой операции.

3. Добавить цикл для повторного запроса операций.
4. Оформить код в соответствии с PEP 8 (или стандартом языка).

Задание 6

Тема: Объектно-ориентированное программирование

Условие: Создать класс Book с полями: название, автор, год издания, ISBN. Реализовать методы: вывод информации, сравнение по году, изменение года.

Задание:

1. Создать класс с конструктором и методами.
2. Создать список из нескольких книг.
3. Вывести книги, изданные после заданного года.
4. Добавить статический метод для подсчёта количества книг.

Задание 7

Тема: Создание базы данных и выполнение SQL-запросов

Условие: Спроектировать БД для интернет-магазина: таблицы Products (id, name, price, quantity), Orders (id, date, customer_name), OrderItems (order_id, product_id, quantity).

Задание:

1. Написать SQL-скрипты для создания таблиц.
2. Заполнить тестовыми данными (INSERT).
3. Написать запросы: список товаров с ценой > 1000, общая стоимость каждого заказа, самый популярный товар.
4. Подключиться к БД из программы (SQLite) и вывести результаты.

Задание 8

Тема: CRUD-операции через параметризованные запросы

Условие: Разработать консольное приложение для управления списком книг (таблица Books).

Задание:

1. Реализовать функции: добавить книгу, просмотреть все, редактировать, удалить, поиск по названию.
2. Использовать параметризованные запросы для защиты от SQL-инъекций.
3. Обработать возможные ошибки (отсутствие записи, дублирование).
4. Организовать меню для взаимодействия.

Задание 9

Тема: ORM (SQLAlchemy)

Условие: Используя SQLAlchemy, описать модели User (id, name, email) и Post (id, title, content, user_id).

Задание:

1. Создать базу данных SQLite с этими таблицами.
2. Добавить несколько пользователей и постов.
3. Выбрать все посты с подгрузкой автора (JOIN).
4. Вывести пользователей, у которых есть посты.

Задание 10

Тема: Разработка простого графического интерфейса (GUI)

Условие: Создать окно для ввода данных о книге и сохранения в БД.

Задание:

1. Использовать Tkinter (Python) или Windows Forms (C#).
2. Форма содержит поля: название, автор, год.
3. Кнопка «Добавить» записывает данные в таблицу Books.
4. Кнопка «Показать» выводит все книги в список/таблицу.

Задание 11

Тема: Интеграция GUI и БД с ООП

Условие: Доработать предыдущее приложение: добавить редактирование выбранной книги и удаление.

Задание:

1. При выборе строки из списка поля заполняются текущими данными.
2. Кнопка «Изменить» обновляет запись.
3. Кнопка «Удалить» удаляет запись с подтверждением.
4. Организовать обновление списка после каждого действия.

Задание 12

Тема: Работа с файлами – импорт/экспорт

Условие: Расширить приложение «Книги» возможностью импорта из CSV и экспорта в CSV.

Задание:

1. Экспорт: сохранить все книги в файл books.csv с разделителем «;».
2. Импорт: прочитать CSV и добавить записи в БД (проверить дубликаты).
3. Выводить отчёт о количестве добавленных/пропущенных записей.

Раздел 3. Работа с системой управления версиями Git

Задание 13

Тема: Основы Git: локальный репозиторий

Условие: Создать локальный репозиторий для проекта «Калькулятор».

Задание:

1. Инициализировать репозиторий (git init).
2. Добавить файл calculator.py (код калькулятора) и сделать первый коммит.
3. Создать файл .gitignore, исключив временные файлы (*.pyc, __pycache__/).
4. Просмотреть историю коммитов (git log --oneline).

Задание 14

Тема: Ветвление и слияние

Условие: Разработать новую функцию «История операций» для калькулятора.

Задание:

1. Создать ветку feature/history.
2. Добавить код, сохраняющий каждое вычисление в список и выводящий историю.
3. Сделать несколько коммитов.
4. Переключиться на ветку main, внести исправление (например, улучшить обработку деления).
5. Слить ветку feature/history в main.
6. Если возник конфликт – разрешить его.

Задание 15

Тема: Работа с удалённым репозиторием (GitHub/GitLab)

Условие: Опубликовать проект на GitHub.

Задание:

1. Создать новый репозиторий на GitHub (без README).
2. Привязать локальный репозиторий к удалённому (git remote add origin ...).
3. Отправить изменения (git push -u origin main).
4. Добавить README.md с описанием проекта и инструкцией по запуску.
5. Создать pull request из ветки feature/history (если ещё не слита) через интерфейс GitHub.

Задание 16

Тема: Разрешение конфликтов и совместная работа

Условие: Два студента работают над одним проектом (или имитация через разные ветки).

Задание:

1. Студент А создаёт ветку feature/a и изменяет функцию add.
2. Студент Б создаёт ветку feature/b и изменяет ту же функцию add.
3. Каждый пушит свою ветку.
4. Студент А создаёт pull request и сливает ветку.
5. Студент Б обновляет свою ветку main и пытается слить – возникает конфликт.
6. Разрешить конфликт, сделать коммит слияния.

Раздел 4. Отладка программного кода

Задание 17

Тема: Использование отладчика IDE

Условие: Дан фрагмент кода с ошибкой (например, неправильное вычисление факториала).

Задание:

1. Запустить отладчик, установить точку останова на вызове функции.
2. Пошагово выполнить программу, наблюдая значения переменных.
3. Найти логическую ошибку.
4. Исправить код и убедиться в правильности.

Задание 18

Тема: Логирование

Условие: В приложение «Калькулятор» добавить логирование всех действий пользователя.

Задание:

1. Настроить вывод логов в консоль и в файл calculator.log.
2. Установить уровень логирования INFO.
3. Логировать: запуск программы, каждую операцию (какие числа, оператор, результат), закрытие программы.
4. Реализовать просмотр последних 10 строк лога через отдельную команду меню.

Задание 19

Тема: Обработка исключений и трассировка стека

Условие: В программе «Работа с книгами» добавить обработку ошибок.

Задание:

1. Обработать ошибку подключения к БД.
2. При попытке удалить несуществующую запись – выводить сообщение, а не крашиться.
3. При вводе некорректных данных (буквы вместо числа) – перехватывать исключение.
4. Выводить полный стек ошибок в лог (уровень ERROR).

Задание 20

Тема: Юнит-тестирование

Условие: Написать юнит-тесты для функций калькулятора (сложение, вычитание, умножение, деление).

Задание:

1. Использовать unittest (Python) или xUnit (C#).
2. Написать тесты для корректных вводов, граничных случаев, деления на ноль (ожидание исключения).
3. Запустить тесты, убедиться, что все проходят.
4. Добавить тест для новой функции (например, возведение в степень).

Раздел 5. Оформление программного кода

Задание 21

Тема: Рефакторинг и приведение к стандартам

Условие: Дан «плохой» код (плохие имена переменных, отсутствие отступов, магические числа, длинные функции).

Задание:

1. Переименовать переменные в соответствии с PEP 8 (snake_case).
2. Заменить магические числа на именованные константы.
3. Разбить длинную функцию на несколько маленьких.
4. Применить автоформаттер (black).
5. Сделать коммит с сообщением «refactor: code style improvements».

Задание 22

Тема: Документирование кода

Условие: Для класса Book (из задания 6) написать docstring для класса и каждого метода.

Задание:

1. Описать параметры конструктора, возвращаемые значения методов.
2. Добавить пример использования в docstring.
3. Сгенерировать HTML-документацию с помощью Sphinx (или pydoc).
4. Разместить сгенерированную документацию в папке docs.

Задание 23

Тема: Оформление репозитория и README

Условие: Подготовить репозиторий проекта «Управление книгами» к публикации.

Задание:

1. Написать README.md: описание, установка, запуск, примеры, скриншоты.
2. Добавить лицензию (MIT).
3. Создать CHANGELOG.md с историей версий (1.0.0 – первая версия).
4. Настроить .gitignore для Python (или другой платформы).
5. Сделать push в удалённый репозиторий.

Раздел 6. Комплексная разработка и защита проекта

Задание 24

Тема: Комплексный проект «Система учёта задач» (Todo-приложение)

Условие: Разработать приложение с графическим интерфейсом, БД, поддержкой Git, логированием, тестами.

Задание:

1. Формализовать задачу, составить ТЗ.
2. Спроектировать БД (таблица Tasks с полями: id, title, description, due_date, status).
3. Реализовать CRUD-операции через GUI (добавление, редактирование, удаление, просмотр).
4. Добавить поиск по названию и фильтрацию по статусу.
5. Написать юнит-тесты для основных функций.
6. Вести разработку с использованием Git (ветки, коммиты).
7. Настроить логирование действий пользователя.
8. Оформить README, документацию.
9. Подготовить презентацию (5 слайдов).
10. Защитить проект перед преподавателем.

Задание 25

Тема: Рецензирование и сборка проекта

Условие: Обменяться проектами с одногруппником и провести рецензирование.

Задание:

1. Склонировать репозиторий коллеги.
2. Запустить приложение, проверить функциональность.
3. Проверить код на соответствие стандартам (линтер).
4. Оценить качество документации.
5. Написать рецензию (список замечаний и предложений).
6. Создать issue на GitHub с рецензией.
7. Подготовить исполняемую версию (exe или Docker-образ) для демонстрации.

Критерии оценивания:

- 5 баллов выставляется, если правильные ответы даны на 85-100% практических заданий
- 4 балла выставляется студенту, если правильные ответы даны на 65-84% практических заданий
- 3 балла выставляется студенту, если правильные ответы даны на 50-64% практических заданий
- 2 балла выставляется студенту, если правильные ответы даны на менее 50% практических заданий

3. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

Процедура оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций состоит из текущего контроля.

Текущий контроль успеваемости проводится с использованием оценочных средств, представленных в п. 2 данного приложения. Результаты текущего контроля доводятся до сведения студентов до промежуточной аттестации и учитываются при оценивании знаний, умений, навыков и (или) опыта деятельности.

Приложение 2

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ОБУЧАЮЩИХСЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ (МОДУЛЯ)

МДК 03.01 Технологии выполнения работ по должности «Программист»

Методические указания для студентов по освоению дисциплины МДК 03.01 Технологии выполнения работ по должности «Программист» являются частью рабочей программы дисциплины (РПД) (приложением к рабочей программе).

РПД − рабочая программа, утвержденная директором колледжа для изучения дисциплины МДК 03.01 Технологии выполнения работ по должности «Программист». Она определяет цели и задачи дисциплины, формируемые в ходе ее изучения компетенции и их компоненты, содержание изучаемого материала, виды занятий и объем выделяемого учебного времени, а также порядок изучения и преподавания дисциплины МДК 03.01 Технологии выполнения работ по должности «Программист».

Для самостоятельной учебной работы студента важное значение имеют разделы «Структура и содержание дисциплины (модуля)» и «Учебно-методическое и информационное обеспечение дисциплины (модуля)». В первом указываются разделы и темы изучаемой дисциплины МДК 03.01 Технологии выполнения работ по должности «Программист», а также виды занятий и планируемый объем (в академических часах), во втором − рекомендуемая литература и перечень ресурсов информационно-телекоммуникационной сети «Интернет».

Для подготовки к текущему контролю студенты могут воспользоваться оценочными средствами, представленными в Приложении 1 к рабочей программе дисциплины.

1. Описание последовательности действий студента

Приступая к изучению, дисциплины МДК 03.01 Технологии выполнения работ по должности «Программист» необходимо в первую очередь ознакомитьсясодержанием РПД, где в разделе «Структура и содержание дисциплины (модуля)» приведено общее распределение часов аудиторных занятий и самостоятельной работы по темам дисциплины МДК 03.01 Технологии выполнения работ по должности «Программист».

Залогом успешного освоения дисциплины МДК 03.01 Технологии выполнения работ по должности «Программист» является регулярное посещение занятий и выполнение предусмотренных программой заданий. Пропуск одного, а тем более нескольких занятий может осложнить освоение разделов курса.

Лекции имеют целью дать систематизированные основы научных знаний по содержанию дисциплины МДК 03.01 Технологии выполнения работ по должности «Программист». При изучении и проработке теоретического материала необходимо:

− повторить законспектированный на лекционном занятии материал и дополнить его с учетом рекомендованной по данной теме литературы;

− при самостоятельном изучении теоретической темы подготовить конспект, используя рекомендованные в РПД литературные источники и электронные образовательные ресурсы.

Практические занятия проводятся с целью углубления и закрепления знаний, полученных на лекциях и в процессе самостоятельной работы с учебной литературой.

В процессе практического занятия, как вида учебных занятий, обучающиеся выполняют одно или несколько практических заданий под руководством преподавателя в соответствии с изучаемым содержанием учебного материала.

Выполнение обучающимся практических работ проводится с целью:

- систематизации и закрепления полученных теоретических знаний и практических умений;

- углубления теоретических знаний в соответствии с заданной темой;

- формирования умений применять теоретические знания при решении поставленных задач;

- развития профессиональных компетенций у обучающихся;

- развития творческой инициативы, самостоятельности, ответственности и организованности.

Выполнение обучающимися практических заданий направлено на:

- обобщение, систематизацию, углубление, закрепление полученных теоретических знаний по конкретным темам дисциплины;
- формирование умений применять полученные знания на практике, реализацию единства интеллектуальной и практической деятельности;
- выработку при решении поставленных задач таких профессионально значимых качеств, как самостоятельность, ответственность, точность, творческая инициатива.

При подготовке к практическому занятию необходимо изучить или повторить лекционный материал по соответствующей теме.

2. Самостоятельная работа студента

Самостоятельная работа студента – самостоятельная учебная деятельность студента, организуемая колледжем и осуществляемая без непосредственного руководства педагога, но по его заданиям и под его контролем.

Цели самостоятельной работы:

– систематизация и закрепление полученных теоретических знаний и практических умений студентов;
– углубление и расширение теоретических знаний;
– формирование умений использовать нормативную, правовую, справочную документацию и специальную литературу;
– развитие познавательных способностей и активности студентов: творческой инициативы, самостоятельности, ответственности и организованности;

формирование самостоятельности мышления, способностей к саморазвитию, самосовершенствованию и самореализации;

– воспитание самостоятельности, как личностного качества будущего специалиста.

Самостоятельная работа студента по дисциплине выполняется:

– самостоятельно вне расписания учебных занятий;
– с использованием современных образовательных технологий;
– работа со специальной литературой для подготовки к тестовым, практическим заданиям.

3. Рекомендации по работе с литературой и источниками

Работу с литературой следует начинать с анализа РПД, содержащей список основной и дополнительной литературы, а также знакомства с учебно-методическими разработками.

В случае возникновения затруднений в понимании учебного материала следует обратиться к другим источникам, где изложение может оказаться более доступным.

Работа с литературой не только полезна как средство более глубокого изучения дисциплины МДК 03.01 Технологии выполнения работ по должности «Программист», но и является неотъемлемой частью профессиональной деятельности будущего выпускника.