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
Тестирование и эксплуатация информационных систем

Специальность
09.02.12 ТЕХНИЧЕСКАЯ ЭКСПЛУАТАЦИЯ И СОПРОВОЖДЕНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ

Форма обучения	очная
Часов по учебному плану	108
в том числе:	
аудиторные занятия	72
самостоятельная работа	30

Ростов-на-Дону
2026 г.

Распределение часов дисциплины по семестрам

Семестр (<Курс>.<Семестр на курсе>)	**4 (2.2)**		Итого	
Недель	18			
Вид занятий	УП	РП	УП	РП
Лекции	36	36	36	36
Практические	36	36	36	36
Итого ауд.	72	72	72	72
Контактная работа	72	72	72	72
Сам. работа	30	30	30	30
Часы на контроль	6	6	6	6
Итого	108	108	108	108

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 2025 г. № 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, Дадашов Г.Х

Председатель ЦМК: Ламин В.А.

Рассмотрено на заседании ЦМК от 06.03.2026 протокол № 7

1. ЦЕЛИ ОСВОЕНИЯ ДИСЦИПЛИНЫ	
1.1	Освоение вида деятельности «Тестирование и эксплуатация информационных систем».

2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ	
Цикл (раздел) ООП:	МДК.01
2.1	**Требования к предварительной подготовке обучающегося:**
2.1.1	Основы алгоритмизации и программирования
2.1.2	Операционные системы и среды
2.1.3	Базы данных
2.1.4	Информатика
2.2	**Дисциплины и практики, для которых освоение данной дисциплины (модуля) необходимо как предшествующее:**
2.2.1	Учебная практика
2.2.2	Производственная практика
2.2.3	Демонстрационный экзамен
2.2.4	Защита дипломного проекта (работы)

3. ТРЕБОВАНИЯ К РЕЗУЛЬТАТАМ ОСВОЕНИЯ ДИСЦИПЛИНЫ

3.1 Знать

ПК.1.3 Осуществлять написание программного кода информационных систем в соответствии с техническим заданием

Основы современных СУБД. Теорию баз данных. Основы программирования. Современные объектно- ориентированные языки программирования. Современные структурные языки программирования. Языки современных бизнес-приложений. Современные методики тестирования разрабатываемых ИС: инструменты и методы модульного тестирования. Методы верификации программного обеспечения. Источники информации, необходимой для профессиональной деятельности в рамках технической поддержки процессов создания (модификации) и сопровождения ИС. Лучшие практики создания (модификации) и сопровождения ИС в экономике.

ПК. 1.4 Выполнять тестирование информационных систем (верификацию) в соответствии с техническим заданием.

Языки программирования и работы с базами данных. Основы современных операционных систем. Основы современных СУБД. Устройство и функционирование современных ИС. Основы архитектуры мультиарендного программного обеспечения. Основы ИБ организации. Теорию баз данных. Системы хранения и анализа баз данных. Современные методики тестирования разрабатываемых ИС. Инструменты и методы модульного тестирования. Источники информации, необходимой для профессиональной деятельности в рамках технической поддержки процессов создания (модификации) и сопровождения ИС. Лучшие практики создания (модификации) и сопровождения ИС в экономике. Культуру речи. Правила деловой переписки

ПК. 1.5 Исправлять дефекты и несоответствия в коде информационных систем и документации к информационным системам

Основы управления изменениями в проектах в области информационных технологий. Основы современных СУБД. Основы ИБ организации. Теорию баз данных. Основы программирования. Современные объектно- ориентированные языки программирования. Современные структурные языки программирования. Языки современных бизнес-приложений. Современные методики тестирования разрабатываемых ИС: инструменты и методы модульного тестирования. Источники информации, необходимой для профессиональной деятельности в рамках технической поддержки процессов создания (модификации) и сопровождения ИС. Лучшие практики создания (модификации) и сопровождения ИС в экономике

3.2 Уметь

ПК.1.3 Осуществлять написание программного кода информационных систем в соответствии с техническим заданием

Кодировать на языках программирования ИС в рамках технической поддержки процессов создания (модификации) и сопровождения ИС. Тестировать результаты разработки ИС в рамках

технической поддержки процессов создания (модификации) и сопровождения ИС.

ПК. 1.4:Выполнять тестирование информационных систем (верификацию) в соответствии с техническим заданием.

Кодировать на языках программирования ИС. Тестировать результаты разработки ИС. Работать с записями по качеству (в том числе с корректирующими действиями, предупреждающими действиями, запросами на исправление несоответствий) при выполнении технической поддержки процессов создания (модификации) и сопровождения ИС.

ПК. 1.5 Исправлять дефекты и несоответствия в коде информационных систем и документации к информационным системам

Кодировать на языках программирования ИС в рамках технической поддержки процессов создания (модификации) и сопровождения ИС. Тестировать результаты разработки ИС в рамках технической поддержки процессов создания (модификации) и сопровождения ИС. Работать с типовой ИС в рамках технической поддержки процессов создания (модификации) и сопровождения ИС. Работать с записями по качеству (в том числе с корректирующими действиями, предупреждающими действиями, запросами на исправление несоответствий) при выполнении технической поддержки процессов создания (модификации) и сопровождения ИС

3.3 Владеть

ПК.1.3 Осуществлять написание программного кода информационных систем в соответствии с техническим заданием

Разработкой кода ИС и баз данных ИС в соответствии с трудовым заданием в рамках технической поддержки процессов создания (модификации) и сопровождения ИС. Верификации кода ИС и баз данных ИС относительно дизайна ИС и структуры баз данных ИС в соответствии с трудовым заданием в рамках технической поддержки процессов создания (модификации) и сопровождения ИС. Устранения обнаруженных несоответствий в коде ИС в соответствии с трудовым заданием в рамках технической поддержки процессов создания (модификации) и сопровождения ИС.

ПК. 1.4 Выполнять тестирование информационных систем (верификацию) в соответствии с техническим заданием.

Проведением тестирования разрабатываемого модуля ИС в соответствии с трудовым заданием в рамках технической поддержки процессов создания (модификации) и сопровождения ИС. Устранения обнаруженных несоответствий в ИС в рамках технической поддержки процессов создания (модификации) и сопровождения ИС. Фиксирования результатов тестирования разрабатываемого модуля ИС в системе учета организации

ПК. 1.5 Исправлять дефекты и несоответствия в коде информационных систем и документации к информационным системам

Воспроизведением зафиксированных в системе учета дефектов и несоответствий в коде ИС и документации к ИС согласно трудовому заданию в рамках технической поддержки процессов создания (модификации) и сопровождения ИС. Установления причин возникновения дефектов и несоответствий в коде ИС и документации к ИС в рамках технической поддержки процессов создания (модификации) и сопровождения ИС. Устранения дефектов и несоответствий в коде ИС и документации к ИС в рамках технической поддержки процессов создания (модификации) и сопровождения ИС

4. СТРУКТУРА И СОДЕРЖАНИЕ ДИСЦИПЛИНЫ (МОДУЛЯ)						
Код занятия	**Наименование разделов и тем /вид занятия/**	**Семестр / Курс**	**Часов**	**Компетен-ции**	**Литература**	**Примечание**
	Тема 3.1. Отладка и тестирование информационных систем					
1.1	Качество информационных систем. Метрики качества (статические метрики: количество строк кода, цикломатическая сложность, коэффициент связности и сцепленной: динамические	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	

	метрики: покрытие кода тестами, частота отказов, время отклика). /Лек/					
1.2	Нормативно-технические материалы по вопросам испытания и тестирования информационных систем. /Лек/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.3	Анализ и оценка качества информационной системы с использованием метрик качества. /Пр/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.4	Понятие процесса тестирования программного обеспечения. Этапы процесса тестирования программного обеспечения. /Лек/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.5	Техники ручного тестирования и автоматизированного тестирования. /Лек/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.6	Использование статического анализа кода для выявления дефектов. /Пр/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.7	Виды тестирования (функциональное тестирование, нефункциональное тестирование, статическое и динамическое тестирование). /Лек/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.8	Типы тестирования (модульное тестирование, интеграционное тестирование, системное тестирование, приемочное тестирование, нагрузочное тестирование, стресс-тестирование). /Лек/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.9	Анализ и оценка качества индивидуальной информационной системы с использованием метрик. /Ср/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.10	Анализ и оценка качества индивидуальной информационной системы с использованием метрик. /Ср/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.11	Анализ и оценка качества индивидуальной информационной системы с использованием метрик. /Ср/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.12	Разработка стратегии отладки и исправление ошибок в программном обеспечении. /Пр/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.13	Тестирование юзабилити: виды, этапы. Методы и инструменты юзабилити тестирования. /Лек/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.14	Анализ требований к программному обеспечению и составление планов тестирования. Использование	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	

	систем контроля дефектов программного обеспечения. /Пр/					
1.15	Анализ требований к программному обеспечению и составление планов тестирования. Использование систем контроля дефектов программного обеспечения. /Пр/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.16	Разработка стратегии отладки и исправление ошибок по заданному программному коду. /Ср/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.17	Разработка стратегии отладки и исправление ошибок по заданному программному коду. /Ср/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.18	Тестирование интеграции: цели, этапы. Практики и инструменты интеграционного тестирования. /Лек/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.19	Разработка тестовых сценариев. /Пр/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.20	Понятие отладки. Понятия ошибки, сбоя, отказа. Типы ошибок. /Лек/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.21	Инструменты для отладки. Процесс пошаговой отладки (установка точек останова, шаг за шагом выполнение кода, просмотр состояния переменных, выполнение отдельных частей кода). /Лек/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.22	Стратегии поиска ошибок (метод половинного деления, метод исключения, проверка граничных условий, поиск паттернов повторяющихся ошибок). Документирование процесса отладки. /Лек/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.23	Поиск и документирование дефектов, используя системы контроля дефектов программного обеспечения. /Пр/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.24	Поиск и документирование дефектов, используя системы контроля дефектов программного обеспечения. /Пр/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.25	Анализ и документирование дефектов, с использованием систем контроля версии. /Ср/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.26	Анализ и документирование дефектов, с использованием систем контроля версии. /Ср/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.27	Анализ и документирование	4	2	ПК 1.3.	Л1.1Л2.1	

	дефектов, с использованием систем контроля версии. /Ср/			ПК 1.4. ПК 1.5.	Э1	
1.28	Тестирование методами белого и черного ящика. /Пр/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.29	Тестирование методами белого и черного ящика. /Пр/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.30	Тестирование методами белого и черного ящика. /Пр/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.31	Чек-листы: требования, процесс создания. Тест-кейсы: цели написания, жизненный цикл, свойства. Наборы тест -кейсов: классификация, принципы построения. /Лек/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.32	Чек-листы: требования, процесс создания. Тест-кейсы: цели написания, жизненный цикл, свойства. Наборы тест -кейсов: классификация, принципы построения. /Лек/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.33	Разработка модульных тестов. Тестирование производительности. Тестирование документации и требований. Тестирование юзабилити. Тестирование интеграции. /Пр/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.34	Автоматизация тестирования. Возможности автоматизации тестирования. Недостатки и риски автоматизации тестирования. /Лек/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.35	Документирование результатов тестирования. /Пр/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.36	Понятие дефекта программного обеспечения. Жизненный цикл дефекта программного обеспечения. /Лек/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.37	Работы, выполняемые при поддержке программного обеспечения. Исправление дефектов. /Лек/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.38	Модель работы с дефектами. Принципы работы в системе контроля дефектов. /Лек/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.39	Ревьюирование кода. Рефакторинг кода. Оптимизация кода. Цели и принципы рефакторинга. Типичные техники рефакторинга. Инструменты	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	

	рефакторинга. /Лек/					
1.40	Работа с системой автоматизированного тестирования. /Пр/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.41	Работа с системой автоматизированного тестирования. /Пр/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.42	Ревьюирование кода, рефакторинг кода, оптимизация кода на конкретном модуле проекта. /Ср/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.43	Ревьюирование кода, рефакторинг кода, оптимизация кода на конкретном модуле проекта. /Ср/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.44	Ревьюирование кода, рефакторинг кода, оптимизация кода на конкретном модуле проекта. /Ср/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.45	Ревьюирование, рефакторинг и оптимизация кода. /Пр/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.46	Ревьюирование, рефакторинг и оптимизация кода. /Пр/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.47	Работа с системой автоматизированного тестирования(Selenium, JUnit/TestNG, NUnit, Allure. /Ср/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.48	Работа с системой автоматизированного тестирования(Selenium, JUnit/TestNG, NUnit, Allure. /Ср/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.49	Работа с системой автоматизированного тестирования(Selenium, JUnit/TestNG, NUnit, Allure. /Ср/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.50	Анализ логов и отчетов об ошибках. /Пр/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.51	Автоматизация тестирования с использованием CI/CD. /Ср/	4	2	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	
1.52	Экзамен /Экзамен/	4	6	ПК 1.3. ПК 1.4. ПК 1.5.	Л1.1Л2.1 Э1	

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 понимается под качеством информационной системы? Перечислите и охарактеризуйте основные метрики качества (статические и динамические).
2. Что такое цикломатическая сложность, коэффициент связности и сцепленности? Как они влияют на качество программного обеспечения?
3. Дайте определение статическим метрикам качества ИС. Что измеряют количество строк кода,

цикломатическая сложность, коэффициенты связности и сцепления?
4. Охарактеризуйте динамические метрики качества ИС: покрытие кода тестами, частота отказов, время отклика системы.
5. Какие нормативно-технические документы используются при тестировании и испытании информационных систем? Назовите ГОСТы, стандарты (ISO 25010, IEEE 829) и отраслевые руководства.
6. Что такое процесс тестирования программного обеспечения? Перечислите и опишите его основные этапы.
7. Сравните ручное и автоматизированное тестирование: техники, преимущества и ограничения.
8. В чём различие между функциональным и нефункциональным тестированием? Приведите примеры.
9. Дайте характеристику статическому и динамическому тестированию. В каких случаях применяется каждый из видов?
10. Охарактеризуйте основные типы тестирования: модульное, интеграционное, системное и приёмочное.
11. Что такое нагрузочное и стресс-тестирование? Какие показатели оцениваются при их проведении?
12. Что такое тестирование юзабилити? Опишите его виды, этапы, методы и используемые инструменты.
13. Каковы цели и этапы интеграционного тестирования? Какие подходы и инструменты применяются?
14. Дайте определения понятиям: ошибка, сбой, отказ. Перечислите типы ошибок и методы их поиска.
15. Опишите процесс отладки программного обеспечения: этапы, инструменты и стратегии поиска ошибок.
16. Что такое чек-лист и тест-кейс? Опишите требования к их разработке и жизненный цикл тест-кейсов.
17. В чём заключается автоматизация тестирования? Оценка целесообразности, риски и технологии автоматизации.
18. Что такое дефект программного обеспечения и его жизненный цикл? Опишите процессы исправления дефектов, ревьюирования и рефакторинга кода.
19. Каков порядок проведения анализа и оценки качества информационной системы с использованием метрик качества (лабораторная работа №1)?
20. Как используется статический анализ кода для выявления дефектов? Какие инструменты статического анализа вы знаете (SonarQube, ESLint, PVS-Studio)?
21. Опишите процесс разработки стратегии отладки и исправления ошибок в программном обеспечении. Какие этапы включает данная стратегия?
22. Как проводится анализ требований к программному обеспечению при составлении планов тестирования? Какие разделы должен содержать план тестирования?
23. Какие системы контроля дефектов программного обеспечения вы знаете (Jira, Redmine, YouTrack, Bugzilla)? Опишите их основные возможности.
24. Как осуществляется разработка тестовых сценариев (test scenarios) на основе требований и вариантов использования (use cases)?
25. Опишите процесс поиска и документирования дефектов с использованием системы контроля дефектов. Какие поля обязательны для заполнения в баг-репорте?
26. В чем суть тестирования методами «белого ящика»? Какие техники тест-дизайна используются (покрытие операторов, ветвей, путей)?

27. В чем суть тестирования методами «черного ящика»? Какие техники тест-дизайна используются (классы эквивалентности, анализ граничных значений, таблицы решений, попарное тестирование)?
28. Как разрабатываются модульные тесты (unit tests)? Какие фреймворки для модульного тестирования существуют (JUnit, PyTest, NUnit)?
29. Как проводится тестирование производительности? Какие метрики фиксируются (время отклика, пропускная способность, потребление ресурсов)?
30. Опишите процесс тестирования документации и требований. Какие типы дефектов могут быть найдены в документации?

31. Какие методы и инструменты используются при тестировании юзабилити (опросы, eye-tracking, think-aloud протоколы, Maze, UserTesting)?
32. Как организовано тестирование интеграции: какие подходы (Big Bang, инкрементальный, «сэндвич») и инструменты применяются?
33. Как правильно документировать результаты тестирования? Какие артефакты создаются (протоколы, отчеты, сводки по дефектам)?
34. Как организована работа с системой автоматизированного тестирования? Назовите популярные инструменты автоматизации (Selenium, Cypress, Postman для API, JMeter для нагрузки).
35. Опишите процессы ревьюирования, рефакторинга и оптимизации кода. В чем их различие и взаимосвязь?
36. Как проводится анализ логов и отчетов об ошибках? Какие инструменты для анализа логов вы знаете (ELK Stack, Graylog, Splunk)?
37. Что такое пошаговая отладка (step-by-step debugging)? Опишите операции: установка точек останова, шаг с заходом, шаг с обходом, шаг с выходом, просмотр состояния переменных.
38. Какие стратегии поиска ошибок существуют: метод половинного деления, метод исключения, проверка граничных условий, поиск паттернов повторяющихся ошибок?
39. Зачем нужно документирование процесса отладки? Какую информацию следует фиксировать при отладке сложного дефекта?
40. Какие требования предъявляются к чек-листам? Опишите процесс создания чек-листа для тестирования конкретной функции.
41. Каковы цели написания тест-кейсов? Опишите жизненный цикл тест-кейса от создания до архивации.
42. Какими свойствами должен обладать хороший тест-кейс (атомарность, воспроизводимость, ожидаемый результат)?
43. Что такое наборы тест-кейсов (test suites)? Как классифицируются наборы и по каким принципам они строятся (регрессионные, смоук, критический путь)?
44. Каковы возможности автоматизации тестирования? В каких случаях автоматизация дает максимальную выгоду?
45. Назовите недостатки и риски автоматизации тестирования (ложные срабатывания, поддержка тестов, высокая первоначальная стоимость).
46. Как проводится оценка применимости и выгоды от автоматизации тестирования (ROI автоматизации)? Какие факторы учитываются?
47. Какие технологии автоматизации тестирования существуют (запись/воспроизведение, data-driven, keyword-driven, BDD- подход)?
48. Какие работы выполняются при поддержке программного обеспечения после его передачи в эксплуатацию? Опишите модель работы с дефектами в этой фазе.
49. Каковы цели и принципы рефакторинга кода? Назовите типичные техники рефакторинга (выделение метода, замена магического числа константой, устранение дублирования).
50. Какие инструменты рефакторинга существуют в современных IDE (Renaming, Extract Method, Inline, Change Signature)? Как рефакторинг связан с ревьюированием кода и оптимизацией?
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 более одной ошибки по изложению фактов или более двух-трёх недочетов 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 МДК

6. УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ

6.1. Рекомендуемая литература

6.1.1. Основная литература				
	Авторы,	Заглавие	Издательство, год	Колич-во
Л1.1	Зараменских Е. П.	Информационные системы: управление жизненным циклом: учебник и практикум для СПО: текст электронный	Юрайт, 2023	https://urait.ru/bcode/530571 – неограниченный доступ зарегистрированным пользователям

6.1.2. Дополнительная литература				
	Авторы, составители	Заглавие	Издательство, год	Колич-во

	Авторы,	Заглавие	Издательство, год	Колич-во
Л2.1	Грекул В. И., Коровкина Н. Л., Левочкина Г. А.	Проектирование информационных систем: учебник и практикум для СПО: текст электронный	Юрайт, 2023	https://urait.ru/bcode/518749 – неограниченный доступ зарегистрированным пользователям

6.2. Перечень ресурсов информационно-телекоммуникационной сети "Интернет"	
Э1	Этапы разработки проекта: реализация, тестирование, эксплуатация и сопровождение - https://www.interface.ru

6.3. Перечень программного обеспечения	
6.3.1	Офисный пакет LibreOffice

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.01.03 Тестирование и эксплуатация информационных систем

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. 1.3 Осуществлять написание программного кода информационных систем в соответствии с техническим заданием.			
Знать: Основы современных СУБД. Теорию баз данных. Основы программирования. Современные объектно-ориентированные языки программирования. Современные структурные языки программирования. Языки современных бизнес-приложений. Современные методики тестирования разрабатываемых ИС: инструменты и методы модульного тестирования. Методы верификации программного обеспечения. Источники информации, необходимой для профессиональной деятельности в рамках технической поддержки процессов создания (модификации) и сопровождения ИС. Лучшие практики создания (модификации) и сопровождения ИС в экономике.	**Получение систематических знаний** Получение систематических знаний в области теории баз данных, современных СУБД, основ и парадигм программирования (структурного и объектно-ориентированного), языков разработки бизнес-приложений, а также методик и инструментов тестирования программного обеспечения, включая модульное тестирование и верификацию, и источников профессиональной информации для сопровождения и эксплуатации информационных систем.	**Уровень знаний** Уровень знаний теоретических основ баз данных и программирования, современных языков разработки, методик и инструментов тестирования и верификации программного обеспечения, а также принципов поиска и использования профессиональной информации и лучших практик сопровождения информационных систем.	**Т 1-40**
Уметь: Кодировать на языках программирования ИС в рамках технической поддержки процессов создания (модификации) и сопровождения ИС. Тестировать результаты разработки ИС в рамках	**Сформировать систематическое умение** Формирование систематических умений разработки программного кода на языках программирования в рамках создания, модификации и	**Уровень умения** Уровень умений разработки программного кода, тестирования и анализа результатов разработки	**Т 1-40 ПЗ 1-12**

технической поддержки процессов создания (модификации) и сопровождения ИС.	сопровождения информационных систем, а также умений тестирования результатов разработки с применением современных методов и инструментов тестирования.	информационных систем, включая оценку качества и корректности функционирования программного обеспечения в процессе его сопровождения и эксплуатации.	
Владеть: Разработкой кода ИС и баз данных ИС в соответствии с трудовым заданием в рамках технической поддержки процессов создания (модификации) и сопровождения ИС. Верификации кода ИС и баз данных ИС относительно дизайна ИС и структуры баз данных ИС в соответствии с трудовым заданием в рамках технической поддержки процессов создания (модификации) и сопровождения ИС. Устранения обнаруженных несоответствий в коде ИС в соответствии с трудовым заданием в рамках технической поддержки процессов создания (модификации) и сопровождения ИС.	**Сформировать систематическое владение** Формирование систематического владения навыками разработки программного кода и баз данных информационных систем, верификации и тестирования кода на соответствие проектным решениям и структуре баз данных, а также навыками выявления и устранения ошибок и несоответствий в процессе создания, модификации и сопровождения информационных систем.	**Уровень владения навыками** Уровень владения практическими навыками разработки, верификации, тестирования и отладки программного кода и баз данных информационных систем, включая устранение дефектов и обеспечение соответствия программного обеспечения установленным требованиям и проектной документации.	**Т 1-40 ПЗ 1-12**
ПК. 1.4:Выполнять тестирование информационных систем (верификацию) в соответствии с техническим заданием.			
Знать: Языки программирования и работы с базами данных. Основы современных операционных систем. Основы современных СУБД. Устройство и функционирование современных ИС. Основы архитектуры мультиарендного программного обеспечения. Основы ИБ организации. Теорию баз данных. Системы хранения	**Получение систематических знаний** Получение систематических знаний в области языков программирования и технологий работы с базами данных, основ современных операционных систем и СУБД, архитектуры и функционирования информационных систем, включая мультиарендные решения, а также основ	**Уровень знания** Уровень знаний языков программирования, технологий работы с базами данных, операционных систем, архитектуры и функционирования информационных систем, включая аспекты информационной безопасности, а	**Т 1-40**

и анализа баз данных. Современные методики тестирования разрабатываемых ИС. Инструменты и методы модульного тестирования. Источники информации, необходимой для профессиональной деятельности в рамках технической поддержки процессов создания (модификации) и сопровождения ИС. Лучшие практики создания (модификации) и сопровождения ИС в экономике. Культуру речи. Правила деловой переписки	информационной безопасности. Освоение теории баз данных, систем хранения и анализа данных, современных методик и инструментов тестирования (в том числе модульного), источников профессиональной информации, лучших практик разработки и сопровождения ИС, а также норм деловой коммуникации и культуры речи.	также методик и инструментов тестирования программного обеспечения, принципов работы с профессиональной информацией и правил деловой коммуникации.	
Уметь: Кодировать на языках программирования ИС. Тестировать результаты разработки ИС. Работать с записями по качеству (в том числе с корректирующими действиями, предупреждающими действиями, запросами на исправление несоответствий) при выполнении технической поддержки процессов создания (модификации) и сопровождения ИС.	**Сформировать систематическое умение** Формирование систематических умений разработки программного кода на языках программирования информационных систем, тестирования результатов разработки, а также работы с записями по качеству, включая оформление корректирующих и предупреждающих действий и обработку запросов на устранение несоответствий в процессе создания, модификации и сопровождения информационных систем.	**Уровень умения** Уровень умений разработки и тестирования программного обеспечения информационных систем, а также ведения и анализа документации по качеству, включая регистрацию, обработку и контроль устранения выявленных несоответствий.	**Т 1-40 ПЗ 1-12**
Владеть: Проведением тестирования разрабатываемого модуля ИС в соответствии с трудовым заданием в рамках технической поддержки процессов создания (модификации) и сопровождения ИС. Устранения обнаруженных несоответствий в ИС в рамках технической поддержки процессов создания (модификации) и	**Сформировать систематическое владение** Формирование систематического владения навыками проведения тестирования модулей информационных систем, выявления и устранения несоответствий в программном обеспечении, а также фиксации и документирования результатов	**Уровень владения** Уровень владения практическими навыками тестирования программного обеспечения, регистрации и анализа результатов тестирования, устранения выявленных	**Т 1-40 ПЗ 1-12**

сопровождения ИС. Фиксирования результатов тестирования разрабатываемого модуля ИС в системе учета организации	тестирования в системах учета в процессе создания, модификации и сопровождения информационных систем.	дефектов и ведения отчетности в системах учета, обеспечивающих контроль качества информационных систем.	
ПК. 1.5:Исправлять дефекты и несоответствия в коде информационных систем и документации к информационным системам			
Знать: Основы управления изменениями в проектах в области информационных технологий. Основы современных СУБД. Основы ИБ организации. Теорию баз данных. Основы программирования. Современные объектно-ориентированные языки программирования. Современные структурные языки программирования. Языки современных бизнес-приложений. Современные методики тестирования разрабатываемых ИС: инструменты и методы модульного тестирования. Источники информации, необходимой для профессиональной деятельности в рамках технической поддержки процессов создания (модификации) и сопровождения ИС. Лучшие практики создания (модификации) и сопровождения ИС в экономике	**Сформировать систематическое умение** Получение систематических знаний в области управления изменениями в проектах информационных технологий, основ программирования и современных языков разработки (структурных, объектно-ориентированных и языков бизнес-приложений), теории баз данных и современных СУБД, а также основ информационной безопасности. Освоение современных методик и инструментов тестирования информационных систем, включая модульное тестирование, источников профессиональной информации и лучших практик создания, модификации и сопровождения информационных систем.	**Уровень владения** Уровень знаний основ управления изменениями в ИТ-проектах, программирования и языков разработки, теории и технологий баз данных, принципов обеспечения информационной безопасности, а также методик и инструментов тестирования и сопровождения информационных систем.	**Т 1-40 ПЗ 1-12**
Уметь: Кодировать на языках программирования ИС в рамках технической поддержки процессов	**Сформировать систематическое умение** Формирование систематических умений разработки программного кода	**Уровень владения** Уровень умений разработки и тестирования программного	**Т 1-40 ПЗ 1-12**

создания (модификации) и сопровождения ИС. Тестировать результаты разработки ИС в рамках технической поддержки процессов создания (модификации) и сопровождения ИС. Работать с типовой ИС в рамках технической поддержки процессов создания (модификации) и сопровождения ИС. Работать с записями по качеству (в том числе с корректирующими действиями, предупреждающими действиями, запросами на исправление несоответствий) при выполнении технической поддержки процессов создания (модификации) и сопровождения ИС	на языках программирования информационных систем, тестирования результатов разработки, работы с типовыми информационными системами, а также ведения и обработки записей по качеству, включая корректирующие и предупреждающие действия и управление запросами на устранение несоответствий в процессе создания, модификации и сопровождения информационных систем.	обеспечения, эксплуатации типовых информационных систем, а также ведения, анализа и сопровождения документации по качеству, включая контроль устранения выявленных несоответствий.	
Владеть: Воспроизведением зафиксированных в системе учета дефектов и несоответствий в коде ИС и документации к ИС согласно трудовому заданию в рамках технической поддержки процессов создания (модификации) и сопровождения ИС. Установления причин возникновения дефектов и несоответствий в коде ИС и документации к ИС в рамках технической поддержки процессов создания (модификации) и сопровождения ИС. Устранения дефектов и несоответствий в коде ИС и документации к ИС в рамках технической	**Сформировать систематическое умение** Формирование систематического владения навыками воспроизведения зарегистрированных дефектов и несоответствий в коде и документации информационных систем, анализа причин их возникновения, а также устранения выявленных дефектов и приведения программного обеспечения и документации в соответствие с установленными требованиями в процессе создания, модификации и сопровождения информационных систем.	**Уровень владения** Уровень владения практическими навыками воспроизведения, анализа и устранения дефектов в программном коде и документации информационных систем, включая выявление причин ошибок и обеспечение соответствия программного обеспечения установленным требованиям и стандартам качества.	**Т 1-40 ПЗ 1-12**

поддержки процессов создания (модификации) и сопровождения ИС			

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. К статическим метрикам относится:

a) время отклика
b) покрытие тестами
c) цикломатическая сложность
d) частота отказов

2. Количество строк кода (LOC) используется, для:

a) оценки сложности интерфейса
b) оценки объема программы
c) оценки производительности
d) тестирования безопасности

3. К динамическим метрикам относится:

a) сцепленность
b) связность
c) покрытие тестами
d) глубина вложенности

4. Что измеряет время отклика системы:

a) объем кода
b) скорость реакции системы
c) количество ошибок
d) структуру программы

5. Первый этап тестирования:

a) выполнение тестов
b) анализ требований
c) написание отчета
d) исправление дефектов

6. Ручное тестирование — это:

a) выполнение тестов программой
b) тестирование без участия человека
c) тестирование человеком без скриптов
d) автоматическое тестирование

7. Автоматизированное тестирование:

a) выполняется вручную
b) выполняется с помощью инструментов
c) не требует сценариев

d) исключает ошибки

8. Функциональное тестирование проверяет:
a) скорость
b) интерфейс
c) функциональность
d) нагрузку

9. Нефункциональное тестирование включает:
a) проверку логики
b) тестирование производительности
c) тестирование алгоритмов
d) написание кода

10. Статическое тестирование:
a) требует запуска программы
b) не требует запуска программы
c) проводится после релиза
d) выполняется только автоматически

11. Динамическое тестирование:
a) без выполнения кода
b) при выполнении программы
c) только на бумаге
d) без данных

12. Модульное тестирование проверяет:
a) систему целиком
b) отдельные модули
c) пользователей
d) документацию

13. Интеграционное тестирование проверяет:
a) отдельные функции
b) взаимодействие модулей
c) дизайн
d) требования

14. Системное тестирование:
a) тест одного модуля
b) тест всей системы
c) тест интерфейса
d) тест документации

15. Приемочное тестирование выполняется:
a) разработчиком
b) заказчиком
c) системой
d) автоматически

16. Нагрузочное тестирование:

a) тест интерфейса
b) тест под нагрузкой
c) тест кода
d) тест документации

17. Стресс-тестирование:
a) нормальная нагрузка
b) экстремальная нагрузка
c) тест дизайна
d) тест требований

18. Юзабилити — это:
a) безопасность
b) удобство использования
c) производительность
d) логика

19. Ошибка — это:
a) корректная работа
b) отклонение от ожидаемого результата
c) команда
d) интерфейс

20. Сбой — это:
a) ошибка в коде
b) проявление ошибки при выполнении
c) документ
d) пользователь

21. Отказ — это:
a) невозможность выполнения функции
b) часть программы
c) тест
d) метод

22. Отладка — это:
a) Тестирование
b) поиск и исправление ошибок
c) разработка
d) анализ требований

23. Breakpoint нужен для:
a) ускорения
b) остановки выполнения
c) удаления кода
d) тестирования

24. Пошаговая отладка позволяет:
a) ускорить код
b) выполнять код по шагам
c) удалить ошибки

d) протестировать интерфейс

25. Метод исключения используется для:
a) поиска ошибок
b) тестирования UI
c) анализа требований
d) разработки

26. Граничные условия проверяются для:
a) поиска ошибок
b) анализа дизайна
c) тестирования пользователей
d) анализа логов

27. Тест-кейс — это:
a) ошибка
b) сценарий проверки
c) программа
d) интерфейс

28. Чек-лист — это:
a) список проверок
b) код
c) ошибка
d) тест

29. Набор тест-кейсов:
a) один тест
b) группа тестов
c) ошибка
d) алгоритм

30. Автоматизация тестирования:
a) всегда обязательна
b) ускоряет тестирование
c) заменяет разработку
d) убирает ошибки

31. Недостаток автоматизации:
a) высокая скорость
b) сложность поддержки
c) точность
d) стабильность

32. Дефект — это:
a) правильная работа
b) ошибка системы
c) интерфейс
d) метод

33. Жизненный цикл дефекта включает:
a) только создание
b) создание и исправление
c) только удаление

d) тестирование интерфейса

34. Система контроля дефектов используется для:
a) написания кода
b) учета ошибок
c) тестирования
d) разработки

35. Ревью кода — это:
a) тест
b) проверка кода
c) выполнение программы
d) запуск системы

36. Рефакторинг — это:
a) изменение логики
b) улучшение структуры
c) удаление тестов
d) добавление функций

37. Оптимизация кода направлена на:
a) ухудшение читаемости
b) повышение эффективности
c) удаление логики
d) тестирование

38. Анализ логов помогает:
a) писать код
b) находить ошибки
c) тестировать интерфейс
d) разрабатывать дизайн

39. Тестирование «белого ящика» основано на:
a) интерфейсе
b) внутренней структуре кода
c) пользователях
d) документации

40. Тестирование «черного ящика» основано на:
a) коде
b) требованиях
c) структуре
d) алгоритмах

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:

Практическое занятие №1
АНАЛИЗ И ОЦЕНКА КАЧЕСТВА ИНФОРМАЦИОННОЙ СИСТЕМЫ С ИСПОЛЬЗОВАНИЕМ МЕТРИК КАЧЕСТВА

Качество ИС– совокупность характеристик информационной системы, которые определяют её способность удовлетворять установленные и предполагаемые потребности пользователей.

Основные метрики качества ИС по стандарту ISO 25010:

1. Функциональность – способность ИС выполнять заданные функции
2. Производительность – быстродействие системы при заданных условиях
3. Надёжность – способность системы работать без сбоев
4. Удобство использования – простота освоения и работы с системой
5. Сопровождаемость – лёгкость внесения изменений и исправлений
6. Переносимость – возможность переноса на другие платформы

Примеры метрик:

- Время отклика системы (сек)
- Количество ошибок на 1000 строк кода
- Процент успешно выполненных функций (%)
- Время наработки на отказ (часы)

ПРАКТИЧЕСКОЕ ЗАДАНИЕ

Цель работы: Научиться анализировать качество информационной системы с использованием базовых метрик.

Оборудование: ПК, калькулятор, MS Excel или аналог.

Задание:

Вам предоставлены данные о работе информационной системы "Электронная библиотека колледжа" за месяц:

Показатель	Значение
Время работы системы за месяц	720 часов
Время простоя из-за сбоев	12 часов
Среднее время отклика на запрос	2,5 сек
Количество обнаруженных ошибок	8
Количество пользователей, освоивших систему за 1 день	25 из 30

Рассчитайте следующие метрики:

1. Функциональная полнота = (Кол-во работающих функций / Общее кол-во функций) × 100%

2. Коэффициент готовности = (Время работы / Общее время) × 100%

3. Показатель удобства использования = (Освоили за 1 день / Всего пользователей) × 100%

4. Оценка производительности:

- Если время отклика < 2 сек – отлично (5 баллов)
- 2-3 сек – хорошо (4 балла)
- 3-5 сек – удовлетворительно (3 балла)
- более 5 сек – неудовлетворительно (2 балла)

5. Сделайте общий вывод о качестве ИС и предложите 2-3 рекомендации по улучшению.

Практическое занятие №2

ИСПОЛЬЗОВАНИЕ СТАТИЧЕСКОГО АНАЛИЗА КОДА ДЛЯ ВЫЯВЛЕНИЯ ДЕФЕКТОВ

Статический анализ кода – метод проверки программного кода без его выполнения, направленный на выявление потенциальных ошибок, уязвимостей и нарушений стандартов кодирования.

Основные типы дефектов, выявляемых статическим анализом:

1. Синтаксические ошибки – нарушение правил языка программирования
2. Логические ошибки – неверная логика работы программы
3. Уязвимости безопасности – потенциальные точки атаки
4. Нарушения стиля кодирования – несоблюдение стандартов
5. "Запахи кода" (code smells) – признаки плохой архитектуры

Преимущества статического анализа:

- Обнаружение ошибок на ранних этапах
- Снижение стоимости исправления дефектов
- Улучшение читаемости и поддерживаемости кода
- Выявление уязвимостей безопасности

Инструменты: SonarQube, ESLint, Pylint, Checkstyle, FindBugs

ПРАКТИЧЕСКОЕ ЗАДАНИЕ

Цель работы: Освоить базовые принципы статического анализа кода на примере выявления типовых дефектов.

Оборудование: ПК, текстовый редактор.

Задание:

Вам предоставлен фрагмент кода на Python для функции расчета скидки в интернет-магазине. В коде содержатся несколько типовых дефектов.

```
def calculate_discount(price, customer_type):
    if price > 1000:
        discount = 0.15
    elif price > 500:
        discount = 0.1
    elif price > 100:
        discount = 0.05

    if customer_type == "VIP":
        discount = discount + 0.05

    final_price = price - (price * discount)

    return final_price
```

Задачи:

1. Проведите ручной статический анализ кода и найдите как минимум 4 дефекта/проблемы.
2. Классифицируйте найденные дефекты по типам:
    - Логическая ошибка
    - Потенциальная ошибка выполнения
    - Нарушение стиля/читаемости
    - Другое (указать)
3. Исправьте код, устранив все найденные дефекты.
4. Добавьте обработку граничных случаев:
    - Цена меньше или равна 0
    - Неизвестный тип клиента
    - Скидка не должна превышать 30%

Подсказки для поиска дефектов:

- Что произойдет, если цена ≤ 100?
- Все ли переменные инициализированы перед использованием?
- Есть ли проблемы с типами данных?
- Достаточно ли читаемы имена переменных?

Форма отчёта:

1. Название работы, цель
2. Таблица найденных дефектов: (Описание дефекта, Тип дефекта, Строка)
3. Исправленный код
4. Выводы о важности статического анализа

Практическое занятие №3

РАЗРАБОТКА СТРАТЕГИИ ОТЛАДКИ И ИСПРАВЛЕНИЕ ОШИБОК В ПРОГРАММНОМ ОБЕСПЕЧЕНИИ

Часть 1. Освоение отладчика

Создайте файл task.py и вставьте код:

```
def calculate_discount(price, percent):
    final_price = price - price * percent / 100
    return round(final_price, 2)


def process_orders():
    orders = [1000, 2500, 3000, 1500]
    discount = 10

    total = 0
    for order in orders:
        total += calculate_discount(order, discount)
        discount += 5

    return total
```

Запустите программу обычным способом. Посмотрите на результат. Затем подумайте: логично ли, что скидка увеличивается для каждого заказа? Это соответствует бизнес-логике?

Теперь начните отладку:

Поставьте точку останова на строку total += calculate_discount(order, discount).

Запустите режим отладки.

Используйте функции шаг с заходом, шаг с обходом и тд.

Следите за значением переменной discount.

В отчёте вы должны описать:

как изменяется значение discount,

почему результат работы функции может быть неверным,

как это может повлиять на расчёты в интернет-магазине.

После анализа исправьте программу так, чтобы скидка была фиксированной.

Часть 2. Отладка ошибок изменяемых объектов

Создайте файл task2.py:

```
def add_item(item, items=[]):
    items.append(item)
    return items


list1 = add_item("apple")
list2 = add_item("banana")
```

Запустите программу.

Ожидали ли вы, что оба списка будут одинаковыми?

Теперь включите отладчик:

Поставьте точку останова внутри функции.

Запустите отладчик.

В окне переменные(Variables) посмотрите:

id(items)

значение списка перед append

значение списка после append

Добавьте в код строки:

print(id(items))

Ответьте письменно:

почему список не создаётся заново при каждом вызове функции,

как это может привести к труднообнаружимой ошибке в реальном проекте,

почему такую ошибку сложно заметить при поверхностном тестировании.

Исправьте функцию корректным способом.

Часть 3. Анализ стека вызовов

Создайте файл task3.py:

```
def level3(x):
    return 10 / x


def level2(x):
    return level3(x - 2)


def level1(x):
    return level2(x - 3)
```

Запустите программу.

Произойдёт ошибка деления на ноль.

Теперь:

Поставьте точку останова на строку return 10 / x.

Запустите отладчик.

Используйте стек вызовов.

Посмотрите, какие функции вызвали текущую.

Запишите значения x на каждом уровне.

В отчёте нужно объяснить:

как формируется стек вызовов,

в какой момент параметр становится равным нулю,

почему ошибка возникает не там, где её логично ожидать.

Часть 4. Работа с памятью и ссылками

Создайте файл task4.py:

```
a = [1, 2, 3]
b = a
c = a.copy()

a.append(4)

print("a:", a)
print("b:", b)
print("c:", c)
```

Перед запуском предположите результат.

Теперь:

Поставьте точку останова после a.append(4).

Запустите отладчик.

Посмотрите id(a), id(b), id(c).

Объясните, какие переменные ссылаются на один и тот же объект.

Затем усложните пример:

```
a = [[1, 2], [3, 4]]
b = a.copy()

a[0].append(99)
```

Проанализируйте:

почему copy() не создаёт независимую структуру,

чем отличается поверхностное копирование от глубокого,

какие риски это создаёт в больших проектах.

Практическое занятие №4

АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ И СОСТАВЛЕНИЕ ПЛАНОВ ТЕСТИРОВАНИЯ. ИСПОЛЬЗОВАНИЕ СИСТЕМ КОНТРОЛЯ ДЕФЕКТОВ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

1. Условие задания

Вам предоставлен фрагмент требований к информационной системе «Электронный журнал колледжа».

Ваша задача — провести анализ качества требований, выявить дефекты и показать, как такие требования могут быть протестированы.

Работа выполняется индивидуально или в группе до 3 человек.

2. Исходные требования (обязательны для всех)

Фрагмент требований к ИС «Электронный журнал колледжа»

Система должна обеспечивать удобную работу преподавателя.

Пользователь может быстро войти в систему.

Данные студентов должны быть защищены.

Преподаватель может выставлять оценки студентам.

Система должна работать стабильно.

В случае сбоя данные не должны теряться.

Доступ к системе осуществляется по логину и паролю.

Пароль должен быть сложным.

Студент может просматривать свои оценки.

Система должна поддерживать большое количество пользователей.

3. Задания к работе

Задание 1. Классификация требований

Для каждого требования:

определить тип:

функциональное;

нефункциональное;

указать роль требования (безопасность, производительность, пользовательский функционал и т.д.).

Задание 2. Анализ качества требований

Для каждого требования:

Определить, является ли оно тестируемым в текущем виде;

Выявить дефекты (если есть);

Указать тип дефекта, например:

неоднозначность;

непроверяемость;

неполнота;

субъективность;

абстрактность.

Задание 3. Улучшение требований

Выберите не менее 4 проблемных требований и перепишите их так, чтобы они:

были однозначными;

содержали измеримые критерии;

могли быть проверены тестированием.

Задание 4. Тестирование требований

Для двух улучшенных требований:

разработайте минимум 2 тест-кейса на каждое:

один позитивный;

один негативный.

Формат тест-кейса свободный, но должен включать:

предусловия;

шаги;

ожидаемый результат.

Пример выполненной работы (образец)

Задание 1–2. Анализ требований

№	Требование	Тип	Тестируемость	Дефект	Обоснование
1	Удобная работа преподавателя	НФ	Нет	Субъективность	Не определены критерии удобства
2	Преподаватель может выставлять оценки	Ф	Частично	Неполнота	Не указано кому, какие оценки

Задание 3. Улучшенные требования

Исходное требование 2:

Пользователь может быстро войти в систему.

Улучшенное требование:

Система должна обеспечивать вход пользователя в систему не более чем за 5 секунд при корректно введённых логине и пароле.

Исходное требование 8:

Пароль должен быть сложным.

Улучшенное требование:

Пароль пользователя должен содержать не менее 8 символов, включая минимум одну заглавную букву, одну строчную букву, одну цифру и один специальный символ.

Задание 4. Тест-кейсы

Тест-кейс 1 (позитивный)

Требование: Пароль должен содержать не менее 8 символов…

Предусловие: Пользователь находится на странице регистрации

Шаги:

1. Ввести пароль Qwerty1!
2. Подтвердить регистрацию

Ожидаемый результат:

Пароль принят, регистрация продолжается

Тест-кейс 2 (негативный)

Шаги:

1. Ввести пароль qwerty
2. Подтвердить регистрацию

Ожидаемый результат:

Система выдаёт сообщение о несоответствии пароля требованиям безопасности

Практическая работа №5
РАЗРАБОТКА ТЕСТОВЫХ СЦЕНАРИЕВ

Цель: получить навыки разработки тестовых сценариев.

Теоретические вопросы

Оценка стоимости и причины ошибок в программном обеспечении. Виды и методы тестирования.

Понятие теста.

Требования к разработке тестовых сценариев. Правила разработки тестовых сценариев.

Задание № 1. Написать программу решения квадратного уравнения $ax^2 + bx + c = 0$.

Задание № 2. Найти минимальный набор тестов для программы нахождения вещественных корней квадратного уравнения $ax^2 + bx + c = 0$. Решение представлено в таблице.

Таким образом, для этой программы предлагается минимальный набор функциональных тестов, исходя из 7 классов выходных данных.

Заповеди по отладки программного средства, предложенные Г. Майерсом.

Заповедь 1. Считайте тестирование ключевой задачей разработки ПС, поручайте его самым квалифицированным и одаренным программистам, нежелательно тестировать свою собственную программу.

Заповедь 2. Хорош тот тест, для которого высока вероятность обнаружить ошибку, а не тот, который демонстрирует правильную работу программы.

Заповедь 3. Готовьте тесты как для правильных, так и для неправильных данных.

Заповедь 4. Документируйте пропуск тестов через компьютер, детально изучайте результаты каждого теста, избегайте тестов, пропуск которых нельзя повторить. Заповедь 5. Каждый модуль подключайте к программе только один раз, никогда не изменяйте программу, чтобы облегчить ее тестирование.

Заповедь 6. Пропускайте заново все тесты, связанные с проверкой работы какой-либо программы ПС или ее взаимодействия с другими программами, если в нее были внесены изменения (например, в результате устранения ошибки).

Задание № 6. Разработайте набор тестовых сценариев (как позитивных, так и негативных) для следующей программы:

Имеется консольное приложение (разработайте самостоятельно). Ему на вход подается 2 строки. На выходе приложение выдает число вхождений второй строки в первую. Например:

Набор тестовых сценариев запишите в виде таблицы, приведенной выше.

Задание № 3. Оформите отчет.

Практическое занятие №6

ПОИСК И ДОКУМЕНТИРОВАНИЕ ДЕФЕКТОВ, ИСПОЛЬЗУЯ СИСТЕМЫ КОНТРОЛЯ ДЕФЕКТОВ ПО

Цель: Научиться находить дефекты в программном обеспечении, правильно оформлять баг-репорты и работать с тест-трекером (на примере баг-трекера).

Краткая теория

Дефект (баг) — это ошибка в программе, из-за которой она работает не так, как ожидается.

Система контроля дефектов (баг-трекер) — программа, где хранятся все найденные ошибки. Примеры: Bugzilla, Jira, Redmine, YouTrack.

Обязательные поля в баг-репорте:

1. Краткое описание (суть проблемы одной фразой).
2. Шаги воспроизведения (что сделать, чтобы увидеть баг).
3. Фактический результат (что происходит на самом деле).
4. Ожидаемый результат (как должно работать правильно).
5. Серьезность (Severity):
   - Blocker — нельзя работать дальше.
   - Critical — крах программы, потеря данных.
   - Major — важная функция не работает.
   - Minor — мелкая ошибка, не мешает работе.
   - Trivial — опечатка, не влияет на логику.
6. Приоритет (Priority):

- High — исправить срочно.

- Medium — исправить в ближайшую очередь.

- Low — исправить в последнюю очередь.

Задание

Представьте, что вы тестировщик. Вам дали тестовый сайт-тренажер:

http://www.seleniumframework.com/PracticeForm/

(откроется в браузере — это учебная форма для тестирования).

Нужно найти 3 реальных или возможных дефекта в работе этой формы (интерфейс, логика, поведение кнопок, поля ввода).

Что делать:

1. Вручную протестировать форму:

- Попробуйте отправить пустую форму.

- Введите буквы в поле для цифр.

- Нажмите кнопки, проверьте, что происходит.

- Посмотрите на внешний вид — есть ли неровности, наложения текста.

2. На каждый дефект завести «ручной баг-репорт» (можно в тетради или текстовом файле) по шаблону ниже.

3. Указать для каждого бага:

- Серьезность (Severity)

- Приоритет (Priority)

Шаблон для одного баг-репорта (заполняется для каждого из 3 дефектов)

Поле	**Значение**
ID (номер)	1
Краткое описание|	«Поле Email принимает текст без собаки @»)
Шаги воспроизведения	1. Открыть форму. 2. В поле Email ввести "test". 3. Нажать Submit.
Фактический результат |	Форма отправляется, хотя email некорректен. |
Ожидаемый результат	Выдается сообщение: «Введите корректный email».
Серьезность (Severity)	Major |
| Приоритет (Priority)	Medium |

Практическая работа №7

«ТЕСТИРОВАНИЕ МЕТОДАМИ БЕЛОГО И ЧЕРНОГО ЯЩИКА.»

Тестирование «Черным ящиком»

Цель работы: изучить метод тестирования «Черным ящиком».

Сегодня тестирование – это обязательная часть процесса разработки программного обеспечения (далее – ПО). Это связано с жесткими правилами конкуренции для компаний, производящих программные продукты (ПП).

Раньше таких компаний на рынке было мало, и пользователи программных продуктов были продвинутыми и заменяли тестеров. Если в программе обнаруживались баги, то пользователь звонил или отправлял письмо в компанию, где ошибку исправляли и по почте отправляли дискетку со свежим релизом. Но начиная с 1990 года согласно статистике продажи персональных компьютеров с каждым годом удваивались. И появилась армия пользователей, которая не готова была что-то тестировать. Если что-то не устроило было проще обменять на другой софт, т.к. число компаний, производящих ПО тоже увеличивалось с каждым готом. И у пользователей появился выбор что покупать и чем пользоваться.

Таким образом, тестирование ушло внутрь компаний, и появилась профессия тестировщика.

Рассмотрим определение, которое записано в SWEBOK.

Тестирование ПО – это проверка соответствия между реальным поведением программы и ее ожидаемым поведением на конечном наборе тестов, выбранном определенным образом. [IEEE Guide to Software Engineering Body of Knowledge, SWEBOK, 2004].

Все виды тестирования можно условно разделить на две большие группы:

Статическое тестирование (static testing).

Динамическое тестирование (dynamic testing).

Статическое тестирование – это процесс анализа самой разработки программного обеспечения, т. е. тестирование без запуска программы.

К данной группе можно отнести анализ кода. Данный вид тестирования осуществляется в основном программистами. Проводят тестирование артефактов разработки программного обеспечения, таких как требования, дизайн или программный код, проводимое без исполнения этих артефактов. Например, с помощью рецензирования или статического анализа.

Статический анализ кода (static code analysis) – это анализ исходного кода, производимый без его исполнения.

Динамическое тестирование – это тестовая деятельность, предусматривающая эксплуатацию (запуск) программного продукта.

Динамическое тестирование предполагает запуск программы, выполнение всех ее функциональных модулей и сравнение фактического ее поведения с ожидаемым.

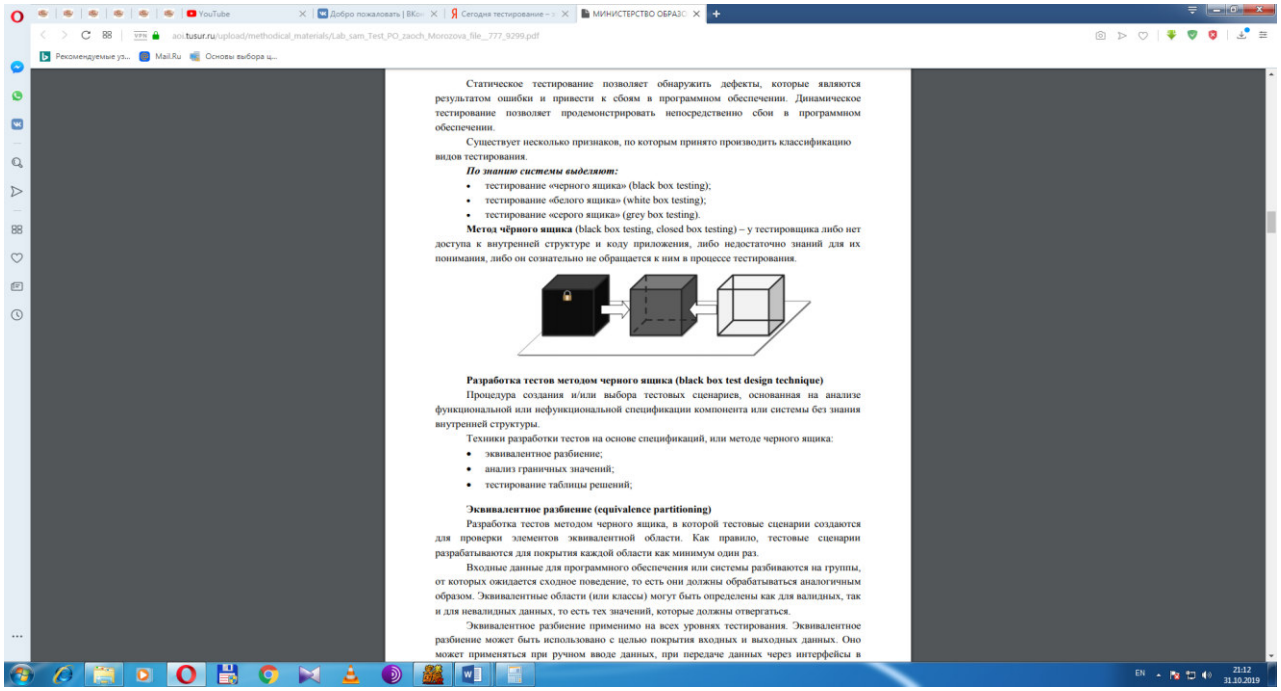
Статическое тестирование позволяет обнаружить дефекты, которые являются результатом ошибки и привести к сбоям в программном обеспечении. Динамическое тестирование позволяет продемонстрировать непосредственно сбои в программном обеспечении.

Существует несколько признаков, по которым принято производить классификацию видов тестирования.

По знанию системы выделяют:

- тестирование «черного ящика» (black box testing);
- тестирование «белого ящика» (white box testing);
- тестирование «серого ящика» (grey box testing).

Метод чёрного ящика (black box testing, closed box testing) – у тестировщика либо нет доступа к внутренней структуре и коду приложения, либо недостаточно знаний для их понимания, либо он сознательно не обращается к ним в процессе тестирования.



Разработка тестов методом черного ящика (black box test design technique)

Процедура создания и/или выбора тестовых сценариев, основанная на анализе функциональной или нефункциональной спецификации компонента или системы без знания внутренней структуры.

Техники разработки тестов на основе спецификаций, или методе черного ящика:

- эквивалентное разбиение;
- анализ граничных значений;
- тестирование таблицы решений;

Эквивалентное разбиение (equivalence partitioning)

Разработка тестов методом черного ящика, в которой тестовые сценарии создаются для проверки элементов эквивалентной области. Как правило, тестовые сценарии разрабатываются для покрытия каждой области как минимум один раз.

Входные данные для программного обеспечения или системы разбиваются на группы, от которых ожидается сходное поведение, то есть они должны обрабатываться аналогичным образом. Эквивалентные области (или классы) могут быть определены как для валидных, так и для невалидных данных, то есть тех значений, которые должны отвергаться.

Эквивалентное разбиение применимо на всех уровнях тестирования. Эквивалентное разбиение может быть использовано с целью покрытия входных и выходных данных. Оно может применяться при ручном вводе данных, при передаче данных через интерфейсы в систему, или при проверке параметров интерфейсов в интеграционном тестировании.

Анализ граничных значений (boundary value analysis): Разработка тестов методом черного ящика, при котором тестовые сценарии проектируются на основании граничных значений.

Граничное значение (boundary value): Входное значение или выходные данные, которое находится на грани эквивалентной области или на наименьшем расстоянии от обеих сторон грани, например, минимальное или максимальное значение области. Анализ граничных значений может применяться на всех уровнях тестирования

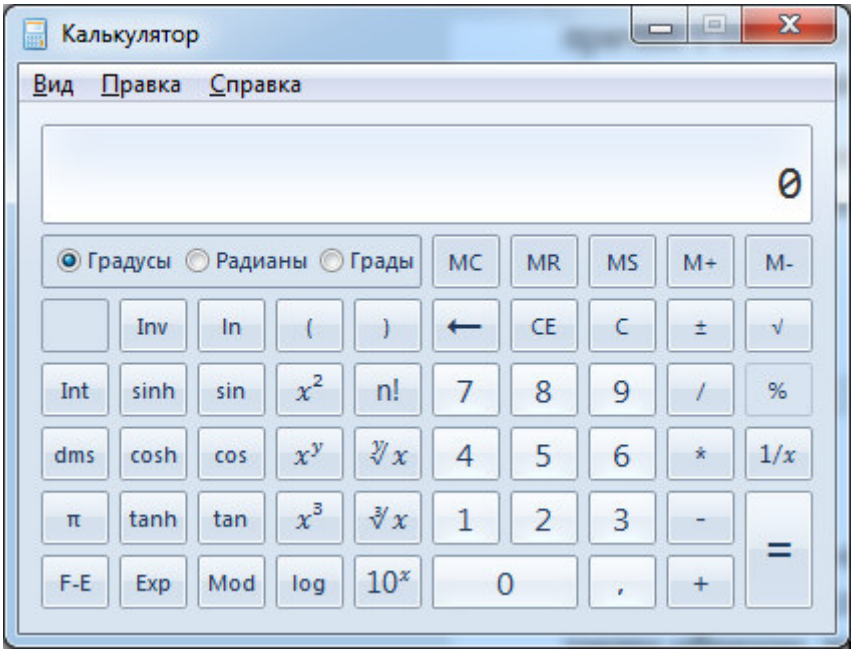
Таблица решений (decision table): Таблица, отражающая комбинации входных данных и/или причин с соответствующими выходными данными и/или действиям (следствиям), которая может быть использована для проектирования тестовых сценариев.

Таблицы решений – хороший метод для сбора системных требований, содержащих логические условия и документирования внутреннего дизайна системы. Они могут использоваться для записи сложных бизнес-правил, которые должна реализовывать система. Анализируются спецификации и определяются условия и действия системы. Входные условия и действия чаще всего формулируются таким образом, чтобы они могли принимать логические значения «истина» или «ложь».

Сильной стороной тестирования таблицы решений является то, что она создает комбинации условий, которые могли бы быть не проверены в ходе тестирования иным способом. Этот метод может быть применен ко всем ситуациям, в которых действие программного продукта зависит от нескольких логических альтернатив.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ И ФОРМА ОТЧЕТНОСТИ:

Задание 1. Написать калькулятор с небольшими багами.



Задание 2. Обменяться программой с другими студентами. Провести тестирование и написать отчет в тетради.

Пример:

Название теста	Описание сценария	Входные данные	Выходные данные	Удачное/ неудачное тестирование	Предложения по исправлению найденных ошибок.	Пожелания пользователей
Функция суммы	Сложение двух положительных чисел; Проверка результата	Первая переменная=3 Вторая переменная=8	Результат=11	Неудачное	-	Поле для ввода значений и вывода, объединить

– при разработке проектов, требующих демонстрации качества и версий системы или продукта через короткий период времени;

– при разработке проектов, для которых необходим подсчет затрат, связанных с оценкой и разрешением рисков.

Практическая работа №8

РАЗРАБОТКА МОДУЛЬНЫХ ТЕСТОВ. ТЕСТИРОВАНИЕ ПРОИЗВОДИТЕЛЬНОСТИ. ТЕСТИРОВАНИЕ ДОКУМЕНТАЦИИ И ТРЕБОВАНИЙ. ТЕСТИРОВАНИЕ ЮЗАБИЛИТИ. ТЕСТИРОВАНИЕ ИНТЕГРАЦИИ.

Цель: Познакомиться с основными видами тестирования ПО, научиться применять их на простых примерах.

Модульное (Unit). Каждый маленький кусок кода (функция, метод) отдельно. Работает ли кнопка «сложение» правильно сама по себе |

Производительности. Скорость работы, нагрузка, время отклика. Не тормозит ли приложение, когда много пользователей.

Документации и требований. Правильно ли написаны инструкции, нет ли противоречий | Сходится ли то, что написано в ТЗ, с тем, что сделано |

Юзабилити (Usability). Удобство для пользователя | Понятно ли, куда нажимать, не путается ли пользователь |

Интеграционное. Как взаимодействуют несколько модулей между собой | Передает ли модуль А данные в модуль Б правильно |

Задание

Часть 1. Модульное тестирование

Дана простая функция (на русском языке):

> Функция «Скидка»: если сумма покупки больше 1000 ₽, то скидка 10%; если больше 5000 ₽ — скидка 20%; иначе скидка 0%.

Задание:

Напишите 3 тест-кейса для этой функции в виде таблицы.

№	Входные данные (сумма)	Ожидаемый результат	Пояснение
1	600	Скидка 0%	Граница до порога
2	1001	Скидка 10%	Чуть выше первого порога
3	5001	Скидка 20%	Второй порог пройден

Часть 2. Тестирование производительности

Сценарий: Представьте, что вы тестируете сайт с расписанием пар. Надо проверить, как он ведет себя при одновременном заходе 10 студентов.

Задание:

1. Назовите 3 показателя производительности, которые надо измерить (например: время загрузки страницы).

2. Предложите простой способ провести нагрузочное тестирование без специальных программ (например: попросить 10 друзей одновременно нажать F5).

Запишите ответ текстом.

Часть 3. Тестирование документации и требований

Дана фраза из документации к расписанию:

«Приложение должно работать быстро и надежно на любых устройствах».

Задание:

1. Почему это плохое требование? (Напишите 1-2 предложения)
2. Перепишите его конкретно (как должно быть в хорошей документации).

Часть 4. Тестирование юзабилити

Дано: описание учебного приложения «Электронный дневник»:

- Кнопка «Выйти» находится рядом с кнопкой «Сохранить».
- Поле «Дата рождения» требует формат `ГГГГ-ММ-ДД`, но подсказки нет.
- Шрифт очень мелкий на кнопках.

Задание (выбрать 2 из 3):

Найдите 2 проблемы юзабилити и предложите, как их исправить. Заполните таблицу:

Проблема	Почему плохо	Как исправить

Часть 5. Тестирование интеграции

Дана система из двух модулей:

- Модуль А — форма ввода логина (логин не должен быть пустым).
- Модуль Б — база данных пользователей (не принимает логин короче 3 символов).

Задание:

Придумайте ситуацию, когда каждый модуль отдельно работает правильно, а вместе —нет. Опишите шаги:

Шаг	Действие	Результат
1	Вместе логин ab (2 символа) в Модуль А	Модуль А пропускает (пустого нет)
2	Передать логин в Модуль Б	Модуль Б отвергает (меньше 3 символов) → Ошибка

Как исправить интеграционную проблему? Запишите одно предложение.

Практическая работа №9

ДОКУМЕНТИРОВАНИЕ РЕЗУЛЬТАТОВ ТЕСТИРОВАНИЯ.

Цель: Научиться составлять итоговый отчет по тестированию ПО, фиксировать результаты, дефекты и выводы.

Краткая теория

Документирование результатов тестирования — это процесс фиксации того, что было проверено, какие баги найдены, а что работает правильно.

Основные документы тестировщика:

Документ	**Краткое описание**
Тест-план	Что, когда и кто тестирует
Тест-кейсы	Конкретные шаги для проверки
Баг-репорт	Отчет об одной ошибке
Отчет о тестировании	Итоговый документ: что прошло, что нет, качество продукта

Что обязательно должно быть в итоговом отчете о тестировании:

1. Название / ID отчета
2. Что тестировалось (объект, версия)
3. Период тестирования
4. Количество тест-кейсов:
   - Пройдено успешно (Passed)
   - Не пройдено (Failed)
   - Заблокировано (Blocked)
5. Количество найденных дефектов (по серьезности)
6. Вывод (можно ли выпускать продукт или нет)

Пример вывода:

«Продукт не готов к релизу. Обнаружено 2 критических дефекта, требующих исправления».

Задание (выполняется за 1 пару)

Вводная: Вы протестировали учебное приложение «Калькулятор оценок студента».

Вот результаты вашего тестирования (выдуманные, но реалистичные):

Исходные данные для отчета

№ тест-кейса	Название	Результат	Примечание
TC-01	Ввод целого числа от 0 до 100	Passed	ОК
TC-02	Ввод отрицательного числа (-5)	Failed	Пропускает, хотя не должно
TC-03	Ввод букв вместо чисел	Passed	Выдает ошибку «Введите число»
TC-04	Расчет среднего балла за 3 предмета	Failed	Результат округляется неверно

TC-05	Кнопка «Сбросить» очищает поля	Passed	OK
TC-06	Сохранение результата в файл	Blocked	Функция не реализована

Найденные дефекты (баг-репорты):

ID	Краткое описание	Серьезность
BUG-1	Программа принимает отрицательные оценки	Critical
BUG-2	Ошибка округления среднего балла (2.7 вместо 2.8)	Major
BUG-3	Функция сохранения отсутствует в сборке	Blocker

Задание для студента

Составить итоговый отчет о тестировании** по шаблону ниже.

Шаблон отчета (заполнить полностью)

Поле	Значение
Название отчета	Тестирование приложения «Калькулятор оценок студента», версия 1.0
Дата тестирования	(указать текущую дату)
Тестировщик	(ФИО студента)
Всего тест-кейсов	6
Passed	(посчитать из таблицы)
Failed	(посчитать)
Blocked	(посчитать)
Всего дефектов	(сколько в списке BUG) |
Blocker / Critical / Major	(распределить по серьезности)
Вывод (нужное подчеркнуть или выделить	Продукт ГОТОВ / НЕ ГОТОВ к выпуску. Почему? (1-2 предложения)

Практическое занятие №10

РАБОТА С СИСТЕМОЙ АВТОМАТИЗИРОВАННОГО ТЕСТИРОВАНИЯ

Автоматизированное тестирование – процесс проверки программного обеспечения с использованием специальных программных инструментов, которые выполняют тесты автоматически без участия человека.

Преимущества автоматизации:

- Быстрое выполнение большого количества тестов
- Возможность многократного использования тестов
- Снижение человеческих ошибок

- Экономия времени при регрессионном тестировании
- Возможность тестирования в нерабочее время

Когда применяется автоматизация:

- Регрессионное тестирование (повторяющиеся тесты)
- Нагрузочное тестирование
- Тестирование на разных платформах/браузерах
- Проверка больших объемов данных

Когда НЕ применяется автоматизация:

- Разовые тесты
- Тестирование UI/UX (удобство использования)
- Исследовательское тестирование
- Тесты, которые часто меняются

Популярные инструменты:

- Selenium – автоматизация веб-приложений
- Postman – тестирование API
- JMeter – нагрузочное тестирование
- pytest, JUnit – модульное тестирование

Структура автоматизированного теста:

1. Arrange (Подготовка) – настройка начальных условий
2. Act (Действие) – выполнение тестируемой операции
3. Assert (Проверка) – сравнение результата с ожидаемым

ПРАКТИЧЕСКОЕ ЗАДАНИЕ

Цель работы: Познакомиться с принципами автоматизированного тестирования на практическом примере.

Оборудование: ПК, MS Excel, калькулятор Windows.

Задание 1. Знакомство с концепцией автоматизации

Вы будете моделировать работу системы автоматизированного тестирования в Excel.

Сценарий: Автоматизация тестирования функции калькулятора (сложение чисел).

Создайте файл Excel "Автотесты_ФИО.xlsx"

Лист 1: "Тестовые данные"

Создайте таблицу с тестовыми данными для проверки операции сложения:

С столбцами:

- № теста
- Описание теста
- Число 1

– Число 2
– Ожидаемый результат
– Фактический результат
– Статус
– Комментарий

Выполните действия:

1. Используя калькулятор Windows, выполните каждую операцию
2. Занесите фактический результат в таблицу
3. В столбце "Статус" укажите:
   - PASS – если фактический = ожидаемому
   - FAIL – если результаты не совпадают
4. Добавьте формулу в столбец "Статус":
   ```
   =ЕСЛИ(D2=E2;"PASS";"FAIL")
   ```
5. Примените условное форматирование:
   - PASS – зеленая заливка
   - FAIL – красная заливка
6. Добавьте итоговую статистику:
   ```
   Всего тестов: =СЧЁТ(A2:A9)
   Пройдено (PASS): =СЧЁТЕСЛИ(F2:F9;"PASS")
   Провалено (FAIL): =СЧЁТЕСЛИ(F2:F9;"FAIL")
   Процент успеха: =PASS/Всего*100
   ```

Задание 2. Создание тест-скрипта (псевдокод)

Лист 2: "Тест-скрипт"

Напишите псевдокод автоматизированного теста для проверки калькулятора:

```
АВТОМАТИЗИРОВАННЫЙ ТЕСТ: Проверка функции сложения

ФУНКЦИЯ test_addition():

 Подготовка (Arrange)
    1. Открыть приложение "Калькулятор"
    2. Убедиться, что калькулятор в режиме "Обычный"
    3. Нажать кнопку "C" для очистки
```

```
    // Действие (Act)
    4. Нажать кнопку "5"
    5. Нажать кнопку "+"
    6. Нажать кнопку "3"
    7. Нажать кнопку "="

    // Проверка (Assert)
    8. Получить результат с экрана калькулятора
    9. Сравнить результат с ожидаемым значением (8)

    ЕСЛИ результат = 8 ТО
        ВЫВОД: "Тест ПРОЙДЕН"
        Статус = PASS
    ИНАЧЕ
        ВЫВОД: "Тест ПРОВАЛЕН"
        ВЫВОД: "Ожидалось: 8, Получено: " + результат
        Статус = FAIL
    КОНЕЦ ЕСЛИ

КОНЕЦ ФУНКЦИИ
```

Ваша задача:

Напишите аналогичный псевдокод для тестов:

1. test_subtraction() – проверка вычитания (10 - 4 = 6)
2. test_multiplication() – проверка умножения (7 × 3 = 21)
3. test_division() – проверка деления (15 ÷ 3 = 5)

Практическое занятие №11

РЕВЬЮИРОВАНИЕ, РЕФАКТОРИНГ И ОПТИМИЗАЦИЯ КОДА.

Задание 1. Проведение Code Review

Вам дан программный код. Представьте, что вы участвуете в Code Review и должны проверить код другого разработчика.

Код для анализа:

```
def calc_total(a, b, c, discount):
    total = a + b + c
    if discount > 0:
```

```
        total = total - total * discount
    else:
        total = total
    return total


def find_biggest(x, y, z):
    biggest = x
    if y > biggest:
        biggest = y
    if z > biggest:
        biggest = z
    if biggest == x:
        print("max number is x")
    elif biggest == y:
        print("max number is y")
    else:
        print("max number is z")
    return biggest


def count_even_numbers(n):
    i = 0
    count = 0
    while i <= n:
        if i % 2 == 0:
            count = count + 1
        i = i + 1
    return count


def average_value(a, b, c, d):
    result = a + b + c + d / 4
    return result
```

Задание 2. Найдите проблемы в коде

Проанализируйте код и ответьте на вопросы.

1. Найдите не менее 5 проблем в коде.

Проблемы могут быть:

логические ошибки

плохие названия переменных

лишние операции

неправильные вычисления

нарушение принципов хорошего кода

Запишите найденные проблемы.

Задание 3. Напишите комментарии Code Review

Представьте, что вы оставляете комментарии разработчику.

Напишите минимум 4 комментария review.

Задание 4. Исправьте одну функцию

Выберите одну функцию из кода и перепишите её так, чтобы:

код был понятнее

логика была корректной

названия переменных были более ясными

Практическое занятие №12

АНАЛИЗ ЛОГОВ И ОТЧЕТОВ ОБ ОШИБКАХ

Лог (Log) – текстовый файл, содержащий записи о событиях, происходящих в системе (запуск приложения, ошибки, действия пользователей, системные события).

Уровни логирования:

- FATAL/CRITICAL – критическая ошибка, приложение не может продолжать работу
- ERROR – ошибка в работе, но приложение продолжает функционировать
- WARNING – предупреждение о потенциальной проблеме
- INFO – информационное сообщение о нормальной работе
- DEBUG – детальная информация для отладки

Зачем нужен анализ логов:

- Поиск причин ошибок и сбоев
- Мониторинг работы системы
- Выявление узких мест производительности
- Анализ поведения пользователей
- Расследование инцидентов безопасности

ПРАКТИЧЕСКОЕ ЗАДАНИЕ

Цель работы: Научиться читать и анализировать логи для поиска причин ошибок в информационной системе.

Оборудование: ПК, MS Word или текстовый редактор.

Лог-файл для анализа

Вам предоставлен фрагмент лога веб-приложения "Электронный журнал колледжа":

```
{} text

2024-01-15 08:00:01 INFO System Приложение запущено
2024-01-15 08:00:05 INFO Database Подключение к базе данных успешно
2024-01-15 08:15:23 INFO Auth Пользователь 'ivanov_p' успешно авторизован
2024-01-15 08:16:45 INFO Auth Пользователь 'petrova_m' успешно авторизован
2024-01-15 08:17:12 WARNING Database Медленный запрос (3.5 сек): SELECT * FROM students
2024-01-15 09:23:34 ERROR Auth Неудачная попытка входа: пользователь 'admin', неверный пароль
2024-01-15 09:23:56 ERROR Auth Неудачная попытка входа: пользователь 'admin', неверный пароль
2024-01-15 09:24:15 ERROR Auth Неудачная попытка входа: пользователь 'admin', неверный пароль
2024-01-15 09:24:20 WARNING Security Пользователь 'admin' заблокирован на 15 минут
2024-01-15 10:15:47 INFO Grades Преподаватель 'sidorov_a' выставил оценку студенту ID:1523
2024-01-15 10:16:02 ERROR Grades Ошибка сохранения оценки: студент ID:9999 не найден
2024-01-15 11:45:23 ERROR Database Потеряно соединение с базой данных
2024-01-15 11:45:25 ERROR System Критическая ошибка: невозможно обработать запрос
2024-01-15 11:45:30 INFO Database Попытка переподключения к БД (1/3)
2024-01-15 11:45:35 INFO Database Подключение к базе данных восстановлено
2024-01-15 12:30:15 WARNING Memory Использование памяти: 85%
2024-01-15 13:00:00 INFO System Начало резервного копирования
2024-01-15 13:15:45 INFO System Резервное копирование завершено успешно
2024-01-15 14:22:10 ERROR FileUpload Ошибка загрузки файла: превышен максимальный размер (15MB)
2024-01-15 15:30:22 FATAL Database Невозможно подключиться к базе данных после 3 попыток
2024-01-15 15:30:25 FATAL System Приложение остановлено из-за критической ошибки
```

ЗАДАНИЯ

Задание 1. Подсчет событий по уровням

Прочитайте лог-файл и подсчитайте количество событий каждого уровня:

- Сколько записей уровня FATAL?
- Сколько записей уровня ERROR?
- Сколько записей уровня WARNING?
- Сколько записей уровня INFO?
- Какой процент составляют ошибки (ERROR + FATAL) от общего количества записей?

Задание 2. Анализ ошибок по компонентам

Выпишите все ошибки (ERROR и FATAL) и определите, в каких компонентах системы они произошли:

- Перечислите все ошибки компонента Database
- Перечислите все ошибки компонента Auth
- Перечислите все ошибки компонента System
- Какой компонент системы работает наиболее нестабильно?

Задание 3. Описание инцидентов

Найдите и опишите три серьезных инцидента из лога:

Инцидент 1: Попытка несанкционированного доступа

- Когда произошло?
- Что именно случилось?
- Какие действия предприняла система?
- Какие могут быть последствия?

Инцидент 2: Проблема с базой данных (11:45)

- Когда началась проблема?
- Что произошло?

- Как система отреагировала?
- Решилась ли проблема? Как?

Инцидент 3: Остановка приложения

- Когда произошла остановка?
- Что стало причиной?
- Какие события предшествовали остановке?
- Какие последствия для пользователей?

Задание 4. Временной анализ

Проанализируйте, как события распределены по времени:

- В какой период времени произошло больше всего ошибок?
- Есть ли время, когда система работала без ошибок?
- В какое время произошла критическая ошибка, остановившая систему?

Задание 5. Поиск признаков проблем

Найдите в логе признаки следующих проблем:

1. Проблемы производительности:
   - Есть ли записи о медленной работе?
   - Есть ли предупреждения о ресурсах системы?
2. Проблемы безопасности:
   - Есть ли попытки подбора пароля?
   - Как система защищается от этого?
3. Проблемы с данными:
   - Есть ли ошибки при работе с данными?
   - Что именно пошло не так?

Задание 6. Создание отчета об ошибке

Выберите самую критичную ошибку из лога и составьте по ней подробный отчет:

- ID ошибки (придумайте, например BUG-001)
- Дата и время обнаружения
- Серьезность (Critical/Major/Minor)
- Приоритет (High/Medium/Low)
- Краткое описание проблемы
- Точная запись из лога
- Что происходило до ошибки (2-3 предыдущих события)
- Последствия ошибки
- Ваши рекомендации по исправлению

Задание 7. Рекомендации по улучшению**

На основе анализа лога предложите минимум 5 рекомендаций для улучшения работы системы:

1. Что нужно сделать с базой данных?

2. Как улучшить безопасность?

3. Что сделать для предотвращения критических сбоев?

4. Какие дополнительные проверки добавить?

5. Как улучшить мониторинг системы?

Задание 8. Общая статистика

Подготовьте краткую статистическую сводку:

- Общее количество записей в логе

- Количество успешных операций

- Количество предупреждений

- Количество ошибок
- Количество критических ошибок
- Самый проблемный час работы системы
- Время работы системы без сбоев (от запуска до первой ошибки)

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое лог-файл и для чего он нужен?
2. Чем отличается уровень ERROR от уровня WARNING?
3. Какая запись в логе самая опасная и почему?
4. Как логи помогают найти причину сбоя системы?
5. Почему важно анализировать не только ошибки, но и обычные INFO-записи?
6. Какие инструменты можно использовать для анализа логов?
7. Как часто нужно проверять логи производственной системы?
8. Что делать, если в логе обнаружены попытки несанкционированного доступа?

Критерии оценивания:

«5 баллов» выставляется студентам за полностью выполненную работу с логическим обоснованием аргументов, в ответе нет исторических ошибок.

«4 балла» выставляется студентам, если работа выполнена полностью, но обоснования доказательства недостаточны, при этом допущены две-три несущественные ошибки, исправленные по требованию преподавателя.

«3 балла» ставится студентам за полный и правильный ответ, при этом допущено более одной ошибки по изложению исторических фактов или более двух-трёх недочетов в ответе.

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

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.01.03 Тестирование и эксплуатация информационных систем

Методические указания для студентов по освоению дисциплины МДК.01.03 Тестирование и эксплуатация информационных систем являются частью рабочей программы дисциплины (РПД) (приложением к рабочей программе).

РПД − рабочая программа, утвержденная директором колледжа для изучения дисциплины МДК.01.03 Тестирование и эксплуатация информационных систем. 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.

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, 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 необходимо в первую очередь ознакомиться содержанием РПД, где в разделе «Структура и содержание дисциплины (модуля)» приведено общее распределение часов аудиторных занятий и самостоятельной работы по темам дисциплины.

Залогом успешного освоения дисциплины 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. 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, но и является неотъемлемой частью профессиональной деятельности будущего выпускника.