Приложение
к ОПОП СПО 09.02.11 Разработка и управление программным обеспечением

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ ДАГЕСТАН

ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
РЕСПУБЛИКИ ДАГЕСТАН «ТЕХНИЧЕСКИЙ КОЛЛЕДЖ ИМЕНИ Р.Н. АШУРАЛИЕВА»

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

междисциплинарного курса МДК 02.01 Разработка программных модулей

Специальность 09.02.11 Разработка и управление программным обеспечением

Квалификация программист

Махачкала, 2026г.

ОДОБРЕНО

предметной (цикловой) комиссией УГС 09.00.00. Информатика и вычислительная техника и 10.00.00 Информационная безопасность

Председатель П(Ц)К

 Ш.М. Мусаева

Протокол № 8 от 30 апреля 2026 г.

Фонд оценочных средств междисциплинарного курса МДК.02.01 Разработка программных модулей разработан на основе требований:

- Приказа Минпросвещения России от 24.02.2025 N 138 "Об утверждении федерального государственного образовательного стандарта среднего профессионального образования по специальности 09.02.11 Разработка и управление программным обеспечением" (Зарегистрировано в Минюсте России 31.03.2025 N 81696);
- Рабочей программы междисциплинарного курса МДК.02.01 Разработка программных модулей

с учетом:

- Примерной образовательной программы «Профессионалитет» по специальности 09.02.11 Разработка и управление программным обеспечением, утвержденная протоколом ФУМО в системе СПО по УГПС 09.00.00 Информатика и вычислительная техника от 29.08.2025 № 6/2025, зарегистрирована в государственном реестре примерных образовательных программ – регистрационный номер 18/2025. Реквизиты утверждающего документа –приказ ФГБОУ ДПО ИРПО № 01-09-581/2025 от 13.10.2025;

Разработчик:

- Азизова Лилия Насруллаховна, к.п.н., преподаватель ГБПОУ РД «Технический колледж имени Р.Н. Ашуралиева»

© Азизова Лилия Насруллаховна, 2026

© ГБПОУ РД «Технический колледж имени Р.Н. Ашуралиева», 2026

СОДЕРЖАНИЕ

1. ОБЩИЕ ПОЛОЖЕНИЯ 4
2. ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ ОСВОЕНИЯ МЕЖДИСЦИПЛИНАРНОГО КУРСА 4
3. ФОРМЫ И МЕТОДЫ КОНТРОЛЯ И ОЦЕНИВАНИЯ....9
4. ПАСПОРТ ФОНДА ОЦЕНОЧНЫХ СРЕДСТВ МЕЖДИСЦИПЛИНАРНОГО КУРСА 12
5. ОЦЕНОЧНЫЕ СРЕДСТВА ТЕКУЩЕГО КОНТРОЛЯ УСПЕВАЕМОСТИ....16
5.1. Типовые задания для оценки знаний и умений....16
5.2. Критерии оценивания 48
5.2.1. Критерии оценивания устного ответа 48
5.2.2. Критерии оценивания выполнения заданий на лабораторных и практических занятиях....48
5.2.3. Критерии оценивания тестовых заданий 49
5.2.4. Общая классификация ошибок 49
6. ОЦЕНОЧНЫЕ СРЕДСТВА ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ....49
6.1. Вопросы к экзамену по МДК....49
6.2. Критерии оценивания ответов на экзамене....52
7. УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ 52

1. ОБЩИЕ ПОЛОЖЕНИЯ

Фонд оценочных средств (далее – ФОС) междисциплинарного курса МДК. 02.01. Разработка программных модулей является частью программы подготовки специалистов среднего звена специальности 09.02.11 Разработка и управление программным обеспечением.

ФОС позволяет оценить достижение, запланированных по междисциплинарному курсу, результатов обучения.

ФОС включают оценочные средства для проведения текущего контроля успеваемости и промежуточной аттестации по междисциплинарному курсу.

Текущий контроль успеваемости осуществляется с целью регулярного наблюдения за ходом поэтапного освоения обучающимися междисциплинарного курса, оптимизации управления образовательной деятельностью обучающихся, своевременной корректировки персональных образовательных результатов обучающихся педагогическими средствами.

Текущему контролю успеваемости подлежат все обучающиеся, осваивающие междисциплинарный курс.

Текущий контроль проводится в пределах учебного времени, отведенного на изучение междисциплинарного курса традиционными и инновационными методами с использованием современных технологий.

Результаты текущего контроля успеваемости обучающихся в виде оценки в балльном выражении («5», «4», «3», «2») записываются в журнале учебных занятий.

Текущий контроль освоения обучающимися программного материала междисциплинарного курса может иметь следующие виды: оперативный и рубежный контроль.

Оперативный контроль проводится с целью объективной оценки качества освоения программы междисциплинарного курса, а также стимулирования учебной работы обучающихся, мониторинга результатов образовательной деятельности, подготовки к промежуточной аттестации и обеспечения максимальной эффективности учебно-воспитательного процесса.

Рубежный контроль является контрольной точкой и проводится с целью комплексной оценки уровня освоения программного материала.

Промежуточная аттестация по междисциплинарному курсу проводится с целью оценки уровня освоения теоретических знаний, умений, приобретенного практического опыта.

Формы и периодичность промежуточной аттестации по междисциплинарному курсу определяются учебным планом образовательной программы: экзамены в 5 и 6 семестрах.

Экзамен проводится непосредственно после завершения освоения междисциплинарного курса, в сроки, установленные календарным учебным графиком. Экзамен проводится в день, освобожденный от других форм учебной нагрузки.

Экзаменационные вопросы и задания составляются на основе рабочей программы междисциплинарного курса. Экзаменационные вопросы и задания должны соответствовать проверяемым результатам обучения и доводятся до сведения обучающихся в течение первых двух месяцев от начала обучения.

2. ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ ОСВОЕНИЯ МЕЖДИСЦИПЛИНАРНОГО КУРСА

ПМ.02 Разработка и интеграция модулей программного обеспечения направлен на формирование общих и профессиональных компетенций.

Освоение междисциплинарного курса должно способствовать формированию общих компетенций:

– ОК 01. Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам.

– ОК 02. Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности.

– ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста.

– ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках.

Освоение междисциплинарного курса должно способствовать формированию профессиональных компетенций:

– ПК 2.1. Проектировать модули программного обеспечения.

– ПК 2.2. Разрабатывать модули программного обеспечения.

– ПК 2.5. Осуществлять документирование программных модулей программного обеспечения.

В результате освоения междисциплинарного курса обучающийся должен овладеть навыками:

– Проектирования модулей ПО с учетом требований заказчика.

– Создания архитектурных диаграмм и спецификаций модулей.

– Определения интерфейсов и взаимодействия модулей.

– Создания модулей программного обеспечения на различных языках программирования.

– Отладки и тестирования разработанных модулей.

– Применения структурного и объектно-ориентированного программирования.

– Оптимизации кода и алгоритмов программных модулей для увеличения производительности.

– Мониторинга и анализа производительности приложений.

– Создания технической документации для модулей.

– Документирования кода, API и интерфейсов.

– Работы со специализированным ПО по документированию программного кода.

В результате освоения междисциплинарного курса обучающийся должен **уметь:**

Код умения	**Название умения**
У1	Проектировать модули, соответствующие бизнес-задачам.
У2	Создавать архитектурные диаграммы и документацию.
У3	Определять структуру и интерфейсы модулей.
У4	Анализировать требования к модулю и определять его функциональность.
У5	Проектировать архитектуру модуля, включая выбор подходящих паттернов проектирования и структуры данных.
У6	Создавать диаграммы классов, последовательностей и прочих диаграмм для визуализации проектируемого модуля.
У7	Выбирать подходящие языки программирования и технологии для реализации модуля.
У8	Проектировать интерфейсы программного обеспечения для взаимодействия с другими модулями и системами.
У9	Учитывать требования к масштабируемости, производительности и безопасности при проектировании модуля.
У10	Проводить анализ и оптимизацию проектируемого модуля для повышения его эффективности и качества.
У11	Разрабатывать модули программного обеспечения с использованием различных языков программирования и технологий.
У12	Применять паттерны проектирования и структуры данных для создания эффективных и масштабируемых модулей.
У13	Анализировать требования и определять функциональность модуля.

У14	Создавать интерфейсы для взаимодействия с другими модулями и системами.
У15	Обеспечивать безопасность, производительность и масштабируемость при разработке модулей.
У16	Оптимизировать проектируемые модули для повышения их эффективности и качества.
У17	Работать с системой контроля версий.
У18	Улучшать производительность модулей, выявляя и устраняя узкие места.
У19	Проводить анализ и мониторинг производительности приложений.
У20	Применять инструменты для рефакторинга и оптимизации программного кода.
У21	Описывать функциональность модулей в документации.
У22	Создавать диаграммы для иллюстрации работы модулей.
У23	Программировать с использованием комментариев для документирования кода.
У24	Использовать специальные метки/теги для отметки важных частей кода в документации.
У25	Вести журнал изменений и фиксировать обновления программных модулей.
У26	Разбивать модули на логические блоки и описывать каждый блок отдельно.
У27	Включать в документацию особенности модулей, такие как ограничения, уязвимости или оптимальные настройки.
У28	Проводить регулярное обновление документации при изменении модулей или добавлении нового функционала.

В результате освоения междисциплинарного курса обучающийся должен **знать:**

Код знания	**Название знания**
З1	Основные принципы проектирования модулей программного обеспечения.
З2	Языки программирования и технологии для реализации модулей.
З3	Паттерны проектирования и структуры данных для создания эффективных и масштабируемых модулей.
З4	Методы анализа требований и способов определения функциональности модуля.
З5	Принципы создания интерфейсов для взаимодействия с другими модулями и системами.
З6	Принципы обеспечения безопасности, производительности и масштабируемости при проектировании модулей.
З7	Методы анализа и оптимизации проектируемых модулей для повышения их эффективности и качества.
З8	Язык программирования, основные конструкции, синтаксис.
З9	Паттерны проектирования.
З10	Структуры данных.
З11	Принципы создания интерфейсов для взаимодействия с другими модулями и системами, таких как REST API, SOAP.
З12	Работа с инструментальным программным обеспечением.
З13	Методы оптимизации кода и алгоритмов.
З14	Эффективные алгоритмы и структуры данных для повышения производительности.
З15	Многопоточность в программных модулях.
З16	Методы оптимизации сетевых протоколов для ускорения обмена данными.
З17	Кэширование данных.
З18	Управление памятью.
З19	Техники повышения производительности программного обеспечения.
З20	Стандарты технической документации.
З21	Принципы документирования программного обеспечения.
З22	Инструменты для создания технической документации и комментирования кода.

Общие компетенции:

Код компетенции	Формулировка компетенции	Знания, умения
ОК 01	Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам	**Умения:** распознавать задачу и/или проблему в профессиональном и/или социальном контексте, анализировать и выделять ее составные части; определять этапы решения задачи, составлять план действия, реализовывать составленный план, определять необходимые ресурсы; выявлять и эффективно искать информацию, необходимую для решения задачи и/или проблемы; владеть актуальными методами работы в профессиональной и смежных сферах; оценивать результат и последствия своих действий (самостоятельно или с помощью наставника)
		Знания: актуальный профессиональный и социальный контекст, в котором приходится работать и жить; структура плана для решения задач, алгоритмы выполнения работ в профессиональной и смежных областях; основные источники информации и ресурсы для решения задач и/или проблем в профессиональном и/или социальном контексте; методы работы в профессиональной и смежных сферах; порядок оценки результатов решения задач профессиональной деятельности
ОК 02	Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности	**Умения:** определять задачи для поиска информации, планировать процесс поиска, выбирать необходимые источники информации; выделять наиболее значимое в перечне информации, структурировать получаемую информацию, оформлять результаты поиска; оценивать практическую значимость результатов поиска; применять средства информационных технологий для решения профессиональных задач; использовать современное программное обеспечение в профессиональной деятельности; использовать различные цифровые средства для решения профессиональных задач
		Знания: номенклатура информационных источников, применяемых в профессиональной деятельности; приемы структурирования информации; формат оформления результатов поиска информации; современные средства и устройства информатизации, порядок их применения; программное обеспечение в профессиональной деятельности, в том числе цифровые средства; психологические основы деятельности коллектива
ОК 05	Осуществлять устную и письменную коммуникацию на государственном языке с учетом особенностей социального и культурного контекста.	**Умения:** грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке проявлять толерантность в рабочем коллективе
		Знания: правила построения устных сообщений; особенности социального и культурного контекста
ОК 09	Пользоваться профессиональной документацией на государственном и иностранном языках	**Умения:** понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы участвовать в диалогах на знакомые общие и профессиональные темы; строить простые высказывания о себе и о своей профессиональной деятельности; кратко обосновывать и объяснять свои действия (текущие и планируемые) писать простые связные сообщения на знакомые или интересующие профессиональные темы

		Знания: правила построения простых и сложных предложений на профессиональные темы; основные общеупотребительные глаголы (бытовая и профессиональная лексика); лексический минимум, относящийся к описанию предметов, средств и процессов профессиональной деятельности; особенности произношения; правила чтения текстов профессиональной направленности

Профессиональные компетенции:

Код и формулировка компетенции	**Показатели освоения компетенции**
ПК 2.1. Проектировать модули программного обеспечения	**Навыки:** Проектирования модулей ПО с учетом требований заказчика. Создания архитектурных диаграмм и спецификаций модулей. Определения интерфейсов и взаимодействия модулей в системе.
	Умения: Проектировать модули, соответствующие бизнес-задачам. Создавать архитектурные диаграммы и документацию. Определять структуру и интерфейсы модулей. Анализировать требования к модулю и определять его функциональность. Проектировать архитектуру модуля, включая выбор подходящих паттернов проектирования и структуры данных. Создавать диаграммы классов, последовательностей и прочих диаграмм для визуализации проектируемого модуля. Выбирать подходящие языки программирования и технологии для реализации модуля. Проектировать интерфейсы программного обеспечения для взаимодействия с другими модулями и системами. Учитывать требования к масштабируемости, производительности и безопасности при проектировании модуля. Проводить анализ и оптимизацию проектируемого модуля для повышения его эффективности и качества.
	Знания: Основные принципы проектирования модулей программного обеспечения. Языки программирования и технологии для реализации модулей. Паттерны проектирования и структуры данных для создания эффективных и масштабируемых модулей. Методы анализа требований и способов определения функциональности модуля. Принципы создания интерфейсов для взаимодействия с другими модулями и системами. Принципы обеспечения безопасности, производительности и масштабируемости при проектировании модулей. Методы анализа и оптимизации проектируемых модулей для повышения их эффективности и качества.
ПК 2.2. Разрабатывать модули программного обеспечения	**Навыки:** Создания модулей программного обеспечения на различных языках программирования. Отладки и тестирования разработанных модулей. Применения структурного и объектно-ориентированного программирования. Оптимизации кода и алгоритмов программных модулей для увеличения производительности. Мониторинга и анализа производительности приложений.
	Умения: Разрабатывать модули программного обеспечения с использованием различных языков программирования и технологий. Применять паттерны проектирования и структуры данных для создания эффективных и масштабируемых модулей. Анализировать требования и определять функциональность модуля. Создавать интерфейсы для взаимодействия с другими модулями и системами. Обеспечивать безопасность, производительность и масштабируемость при разработке модулей. Оптимизировать проектируемые модули для повышения их эффективности и качества;

	Работать с системой контроля версий. Улучшать производительность модулей, выявляя и устраняя узкие места. Проводить анализ и мониторинг производительности приложений. Применять инструменты для рефакторинга и оптимизации программного кода.
	Знания: Язык программирования, основные конструкции, синтаксис. Паттерны проектирования. Структуры данных. Принципы создания интерфейсов для взаимодействия с другими модулями и системами, таких как REST API, SOAP. Работу с инструментальным программным обеспечением. Методы оптимизации кода и алгоритмов. Эффективные алгоритмы и структуры данных для повышения производительности; Многопоточность в программных модулях. Методы оптимизации сетевых протоколов для ускорения обмена данными; Кэширование данных. Управление памятью. Техники повышения производительности программного обеспечения.
ПК 2.5. Осуществлять документирование программных модулей программного обеспечения.	**Навыки:** Создания технической документации для модулей. Документирования кода, API и интерфейсов. Работы со специализированным ПО по документированию программного кода.
	Умения: Описывать функциональность модулей в документации. Создавать диаграммы для иллюстрации работы модулей. Программировать с использованием комментариев для документирования кода. Использовать специальные метки/теги для отметки важных частей кода в документации. Вести журнал изменений и фиксировать обновления программных модулей. Разбивать модули на логические блоки и описывать каждый блок отдельно. Включать в документацию особенности модулей, такие как ограничения, уязвимости или оптимальные настройки. Проводить регулярное обновление документации при изменении модулей или добавлении нового функционала.
	Знания: Стандарты технической документации. Принципы документирования программного обеспечения. Инструменты для создания технической документации и комментирования кода.

3. ФОРМЫ И МЕТОДЫ КОНТРОЛЯ И ОЦЕНИВАНИЯ

Формы текущего контроля по междисциплинарному курсу:

- устный опрос (фронтальный, индивидуальный, комбинированный);
- тестирование (письменное или компьютерное);
- письменная проверка (ответы на вопросы, решение задач и примеров, составление тезисов, рефератов, выполнение схем, выполнение заданий для самостоятельной работы и др.);
- практическая проверка (при проведении практических и лабораторных занятий);
- самоконтроль и взаимопроверка.

Возможны и другие формы текущего контроля успеваемости, в том числе инновационные на основе информационно-коммуникационных технологий.

Преподаватель на одном учебном занятии может использовать одну или несколько форм текущего контроля.

Код и наименование профессиональных и общих	**Критерии оценки результата (показатели освоенности компетенций)**	**Формы контроля и методы оценки**

компетенций, формируемых в рамках модуля		
ПК 2.1. Проектировать модули программного обеспечения	Анализ проектной и технической документации. Проектирования модулей ПО с учетом требований заказчика. создания архитектурных диаграмм и спецификаций модулей. определения интерфейсов и взаимодействия модулей в системе.	Экспертная оценка деятельности в ходе выполнения практических работ, практической подготовки, интерпретация результатов собеседования и наблюдения, решение производственных задач. Текущий контроль: - защита отчетов по практическим работам; - оценка заданий для самостоятельной работы - экспертная оценка демонстрируемых умений, выполняемых действий в процессе практических занятий Промежуточная аттестация: - экспертная оценка выполнения практических заданий на экзамене по МДК;
ПК 2.2. Разрабатывать модули программного обеспечения	Разрабатывать модули программного обеспечения с использованием различных языков программирования и технологий применять паттерны проектирования и структуры данных для создания эффективных и масштабируемых модулей анализировать требования и определять функциональность модуля создавать интерфейсы для взаимодействия с другими модулями и системами обеспечивать безопасность, производительность и масштабируемость при разработке модулей оптимизировать проектируемые модули для повышения их эффективности и качества работать с системой контроля версий улучшать производительность модулей, выявляя и устраняя узкие места проводить анализ и мониторинг производительности приложений применять инструменты для рефакторинга и оптимизации программного кода	
ПК 2.5. Осуществлять документирование программных модулей программного обеспечения	Использование выбранной системы контроля версий. Использование методов для получения кода с заданной функциональностью и степенью качества. Выявление ошибок в системных компонентах на основе спецификаций.	
ОК 01. Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам	Обоснованность планирования учебной и профессиональной деятельности; соответствие результата выполнения профессиональных задач эталону (стандартам, образцам, алгоритму, условиям, требованиям или ожидаемому результату); степень точности выполнения поставленных задач.	
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности	полнота охвата информационных источников; скорость нахождения и достоверность информации; обновляемость и пополняемость знаний, влияющих на результаты учебной и производственной деятельности.	

ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке с учетом особенностей социального и культурного контекста	демонстрация навыков грамотно общения и оформление документации на государственном языке Российской Федерации, принимая во внимание особенности социального и культурного контекста	
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках	демонстрация умений понимать тексты на базовые и профессиональные темы; - составлять необходимую документацию на государственном и иностранном языках	

4. ПАСПОРТ ФОНДА ОЦЕНОЧНЫХ СРЕДСТВ МЕЖДИСЦИПЛИНАРНОГО КУРСА

МДК 02.01 Разработка программных модулей

№	Наименование темы	Результаты обучения (освоенныеумения и знаний)	ПК, ОК	Текущий контроль успеваемости	Промежуточная аттестация
	МДК 02.01 Разработка программных модулей				
	Раздел 1. Технология разработки программного обеспечения				
	Тема 1.1. Разработка прикладного программного обеспечения				
	Введение. Понятие ЖЦ ПО. Модели жизненного цикла ПО	З4, З20, З21	ОК 01, 05, 09 ПК 2.1, 2.2, 2.5	Устный опрос, тестирование	Экзамен
	Компонентно-ориентированный подход при разработке ПО	З1, З2, З5, З9, З11		Устный опрос, тестирование	
	Этапы разработки ПО	З2, З4, З12, З20, З21		Устный опрос, тестирование	
	Анализ предметной области, разработка и оформление технического задания	У4, У13, У21, У28		Устный опрос, тестирование	
	Изучение работы в системе контроля версий	У17		Устный опрос, тестирование	
	Тема 1.2. Структурное, объектно-ориентированное и событийно-управляемое программирование				
	Понятие структурное программирование	З8, З10	ОК 01, 05, 09 ПК 2.1, 2.2, 2.5	Устный опрос, тестирование	
	Оценка сложности алгоритма	З13, З14		Устный опрос, тестирование	
	Документирование алгоритмов программного обеспечения	З20, З21, З22		Устный опрос, тестирование	
	Основы объектно-ориентированного программирования	З2, З8, З9, З10		Устный опрос, тестирование	
	Статические методы и поля	З8		Устный опрос, тестирование	
	Виртуальные и динамические методы. Полиморфизм	З8, З9		Устный опрос, тестирование	
	Понятие метаклассов	З8		Устный опрос, тестирование	
	Использование метаклассов в программировании	З8		Устный опрос, тестирование	
	Основные принципы событийно-управляемое программирование	З8, З9, З10		Устный опрос, тестирование	
	Элементы управления	З8, З9, З12		Устный опрос, тестирование	
	Диалоговые окна	З8, З9, З12		Устный опрос, тестирование	
	Обработчики событий. Введение в графику	З8, З9, З10, З13		Устный опрос, тестирование	
	Использование инструментальных средств документирования алгоритмов программ	У21, У22, У23, У24, У26		Отчет по лабораторному занятию	
	Определение сложности алгоритмов сортировки, поиска, рекурсивных и эвристических алгоритмов	У10, У16, У18		Отчет по лабораторному занятию	
	Работа с классами и перегрузка методов	У11, У12		Отчет по лабораторному занятию	
	Определение операций в классе	У11, У12		Отчет по лабораторному занятию	
	Создание наследованных классов	У11, У12, У5		Отчет по лабораторному занятию	
	Работа с объектами через интерфейсы	У3, У8, У14		Отчет по лабораторному занятию	
	Использование стандартных интерфейсов	У8, У14		Отчет по лабораторному занятию	
	Работа с типом данных структура, использование коллекций	У5, У12		Отчет по лабораторному занятию	
	Операции со списками	У11, У12		Отчет по лабораторному занятию	
	Разработка приложения с использованием текстовых компонентов	У7, У11		Отчет по лабораторному занятию	
	Разработка приложения с несколькими формами	У7, У11		Отчет по лабораторному занятию	

	Разработка приложения с не визуальными компонентами	У7, У11		Отчет по лабораторному занятию	
	Разработка игрового приложения	У7, У11, У15		Отчет по лабораторному занятию	
	Разработка программных модулей для работы с датой и временем. Работа через систему контроля версий	У11, У17, У25		Отчет по лабораторному занятию	
	Разработка программных модулей с использованием паттернов проектирования. Работа через систему контроля версий	У5, У12, У17, У25		Отчет по лабораторному занятию	
	Навигация по файловой системе. Чтение и запись файлов. Работа с потоками. Работа с изолированным хранилищем	У7, У9, У11, У12		Отчет по лабораторному занятию	
	Работа с большими объемами текста. Кодирование и декодирование строк. Построение регулярных выражений. Чтение и запись файлов в разных кодировках	У9, У11, У12, У16		Отчет по лабораторному занятию	
	Тема 1.3. Модульный принцип разработки ПО. Основы работы с базами данных				
	Основные критерии оптимизации модулей	З6, З7, З13-З19	ОК 01, 05, 09 ПК 2.1, 2.2, 2.5	Устный опрос, тестирование	
	Информационная закрытость	З1, З3, З5		Устный опрос, тестирование	
	Связность. Виды связности	З1		Устный опрос, тестирование	
	Сцепление. Типы сцепления	З1		Устный опрос, тестирование	
	Специальные библиотеки	З2, З12		Устный опрос, тестирование	
	Базовый синтаксис SQL	З2, З8, З10		Устный опрос, тестирование	
	Создание таблицы, работа с данными	З2, З8, З10		Устный опрос, тестирование	
	DB-API модули	З2, З8, З12		Устный опрос, тестирование	
	Объектно-реляционное отображение (ORM)	З2, З8, З12		Устный опрос, тестирование	
	Разработка приложения с функционально связанными модулями	У1, У3, У5, У9, У11		Отчет по лабораторному занятию	
	Разработка приложения с информационно связанными модулями	У1, У3, У5, У9, У11		Отчет по лабораторному занятию	
	Разработка приложения с коммуникативно связанными модулями	У1, У3, У5, У9, У11		Отчет по лабораторному занятию	
	Разработка приложения с различными типами сцепления модулей	У1, У3, У5, У9, У11		Отчет по лабораторному занятию	
	Определение меры сопряжения между модулями	У1, У3, У9, У10		Отчет по лабораторному занятию	
	Анализ базового синтаксиса SQL, создание таблицы	У7, У11		Отчет по лабораторному занятию	
	Создание приложения с БД	У7, У9, У11, У15		Отчет по лабораторному занятию	
	Создание запросов и хранимых процедур к БД	У7, У11, У16		Отчет по лабораторному занятию	
	Разработка стилей для приложения для улучшения взаимодействия с пользователем	У8, У14		Отчет по лабораторному занятию	
	Разработка модулей для работы с изображениями	У7, У11, У12		Отчет по лабораторному занятию	
	Разработка модулей для представления информации в виде графиков и диаграмм	У7, У11, У22		Отчет по лабораторному занятию	
	Разработка модулей для работы аудио и видео	У7, У11, У12		Отчет по лабораторному занятию	
	Реализация загрузки данных из интернета в фоновом режиме без блокировки основного потока приложения.	У7, У9, У11, У15		Отчет по лабораторному занятию	
	Разработка формы регистрации с элементами ввода и проверкой корректности введенных данных.	У7, У9, У11, У15		Отчет по лабораторному занятию	
	Тема 1.4. Конструирование ПО				
	Правила разработки интерфейсов пользователя. Требования интерфейса. Анализ интерфейса	З5, З11	ОК 01, 05, 09 ПК 2.1, 2.2, 2.5	Устный опрос, тестирование	
	Назначение и виды паттернов. Порождающие паттерны. Структурные паттерны. Поведенческие паттерны	З3, З9		Устный опрос, тестирование	

	Рефакторинг. Определение, причины и цели	З7, З13		Устный опрос, тестирование	
	Упрощение вызовов методов. Решение задач обобщения	З3, З7, З13		Устный опрос, тестирование	
	Экстремальное конструирование ПО	З4, З12		Устный опрос, тестирование	
	Обратный инжиниринг. Определение, цели проведения	З7		Устный опрос, тестирование	
	Методики проведения обратного инжиниринга. Реинжиниринг	З7, З13		Устный опрос, тестирование	
	Проектирование интерфейса пользователя	У2, У6, У8, У9, У14		Отчет по лабораторному занятию	
	Разработка интерфейса пользователя	У7, У11, У14		Отчет по лабораторному занятию	
	Проектирование с использованием паттернов	У2, У5, У9, У12		Отчет по лабораторному занятию	
	Рефакторинг приложения при наличии дублирования кода	У10, У16, У18, У20		Отчет по лабораторному занятию	
	Рефакторинг приложения при наличии большого класса и жадных функций	У10, У16, У18, У20		Отчет по лабораторному занятию	
	Рефакторинг приложения при наличии избыточных временных переменных	У10, У16, У18, У20		Отчет по лабораторному занятию	
	Рефакторинг приложения при наличии несгруппированных данных	У10, У16, У18, У20		Отчет по лабораторному занятию	
	Реинжиниринг приложения	У10, У16, У18, У20		Отчет по лабораторному занятию	
	Самостоятельная работа обучающихся:				
73.	Изучить теоретический материал и составить тезисы (краткий конспект) по следующим темам:				
	Модульная архитектура построения приложений. Принципы. Преимущества. Примеры приложений.	З1, З3, З6	ОК 01-09 ПК 2.1, 2.2, 2.5	Конспект, устный опрос	
	Архитектурные шаблоны, применяемые при разработке программных модулей (MVC, MVVM, MVP).	З3, З9			
	Инструменты разработки приложений с модульной архитектурой. Системы контроля версий.	З12			
	Алгоритмы сортировки и поиска. Основы рекурсии: примеры, преимущества и недостатки.	З10, З13, З14			
	Хеш-таблица и хеш-функция. Коллизии и разрешение коллизий. Методы хеширования и сжатия данных. Эффективность и применение хеш-структур.	З10, З13, З14			
	Деревья и графы. Представление графов и деревьев. Поиск в глубину и ширину. Минимум затратный путь (алгоритм Дейкстры). Деревья поиска и обхода.	З10, З14			
	Основные принципы проектирования модулей программного обеспечения. Методы анализа требований и способов определения функциональности модуля. Методы анализа и оптимизации проектируемых модулей для повышения их эффективности и качества.	З1, З4, З7			
	Применение диаграмм классов при проектировании требований к внутренней структуре программного модуля.	З20, З21, З22			
	Применение диаграмм компонентов для визуализации организации компонентов проектируемого модуля.	З20, З21, З22			
	Технологии и инструменты разработки графического пользовательского интерфейса.	З2, З12			

	Компоненты графического пользовательского интерфейса. Типы элементов управления. Компоновка элементов управления. События. Обработчики событий.	З5, З8, З11			
	Основные понятия: безопасность программного обеспечения, производительность модулей, масштабируемость архитектуры. Методы обеспечения безопасности.	З6			
	Факторы, влияющие на производительность. Техники повышения производительности программного обеспечения	З6, З13, З14, З19			
	Масштабируемость: горизонтальная и вертикальная масштабируемость; принципы проектирования для масштабируемости; использование облачных технологий для масштабирования.	З1, З3, З6			
	Метрики безопасности (например, количество уязвимостей). Инструменты для мониторинга производительности. Подходы к нагрузочному тестированию.	З6, З7, З12			
	Понятие оптимизации кода. Основные цели оптимизации: повышение скорости выполнения, снижение потребления памяти, улучшение читаемости и поддержки кода. Методы улучшения алгоритмов. Профилирование и отладка производительности. Специфичные методы оптимизации для разных языков программирования.	З7, З13, З14, З15, З18, З19			

5. ОЦЕНОЧНЫЕ СРЕДСТВА ТЕКУЩЕГО КОНТРОЛЯ УСПЕВАЕМОСТИ

МДК 02.01 Разработка программных модулей

5.1.Типовые задания для оценки знаний и умений

Раздел 1. Технология разработки программных модулей

Тема 1.1. Разработка прикладного программного обеспечения

1. Задание для устного опроса по темам

1. Дайте определение жизненного цикла программного обеспечения. Какие основные фазы (этапы) традиционно выделяют в ЖЦ ПО? Кратко охарактеризуйте каждую фазу.
2. Опишите каскадную (водопадную, waterfall) модель ЖЦ. Назовите ее ключевые особенности, преимущества и основные недостатки. В каких проектах применение этой модели оправдано, а в каких – нет?
3. В чем суть итеративного подхода к разработке ПО? Чем итеративная модель отличается от каскадной? Приведите примеры итеративных моделей (например, спиральная модель, модель с промежуточными версиями).
4. Спиральная модель ЖЦ: основные принципы, этапы каждого витка. Какие риски учитываются в спиральной модели и как они управляются? Почему эта модель считается риск-ориентированной?
5. Что такое гибкие методологии (Agile)? Назовите основные ценности и принципы Agile-манифеста. Чем Agile-подход принципиально отличается от традиционных (каскадных) моделей?
6. Сравните каскадную и итеративную модели ЖЦ по следующим критериям: гибкость изменений, взаимодействие с заказчиком, управление рисками, сроки поставки. В каких случаях выбор той или иной модели будет предпочтительнее?
7. Какие внешние и внутренние факторы влияют на выбор модели жизненного цикла для конкретного проекта? Приведите примеры, когда выбор модели может быть определен размером команды, сложностью системы или требованиями к надежности.
8. Дайте определение компонентно-ориентированного программирования (КОП) и компонентно-ориентированной разработки. Каковы основные цели и идеи этого подхода?
9. Опишите преимущества компонентно-ориентированного подхода перед монолитной разработкой. Какие проблемы он решает (например, сложность сопровождения, повторное использование кода, распределенная разработка)?
10. Назовите известные стандарты и технологии компонентной разработки (COM/DCOM, CORBA, JavaBeans / EJB, .NET-компоненты). Кратко охарактеризуйте один из них: на чем основан, как обеспечивается взаимодействие компонентов?
11. Какие недостатки и ограничения присущи компонентно-ориентированному подходу? С какими сложностями сталкиваются разработчики при проектировании компонентных систем (например, версионность, совместимость, производительность)?

12. Как в компонентно-ориентированной системе обеспечивается слабая связанность (loose coupling) и высокая связанность (cohesion)? Приведите примеры интерфейсных контрактов и механизмов их реализации.
13. Что включает этап «Анализ требований»? Какие виды требований выделяют (функциональные, нефункциональные, ограничения)? Какие методы сбора и анализа требований вы знаете?
14. Опишите этап проектирования ПО. В чем различие между архитектурным (высокоуровневым) и детальным (низкоуровневым) проектированием? Какие артефакты создаются на этом этапе (диаграммы, спецификации, прототипы)?
15. Какую роль играет тестирование в процессе разработки ПО? Назовите уровни тестирования (модульное, интеграционное, системное, приемочное) и объясните, на каких этапах ЖЦ они применяются.
16. Что такое «сопровождение ПО»? Перечислите виды сопровождения (корректирующее, адаптивное, совершенствующее, профилактическое) и приведите примеры каждого.
17. Как этапы разработки ПО могут меняться при использовании различных моделей ЖЦ? Например, в каскадной модели этапы строго последовательны, а в итеративной – цикличны. Сравните организацию этапов для двух разных моделей.

Тестовые вопросы

Тема 1.1. Разработка прикладного программного обеспечения

1. Что понимается под жизненным циклом программного обеспечения?
А) Процесс создания программного кода
Б) Период времени от начала разработки до окончания эксплуатации и снятия с эксплуатации
В) Процесс тестирования ПО
Г) Набор методов управления проектами

2. Какая модель ЖЦ предполагает последовательное выполнение этапов, где каждый следующий начинается только после завершения предыдущего?
А) Спиральная
Б) Каскадная (водопадная)
В) Итеративная
Г) Agile

3. В какой модели жизненного цикла основное внимание уделяется управлению рисками и повторению циклов до достижения приемлемого уровня?
А) Каскадная
Б) V-образная
В) Спиральная
Г) Инкрементная

4. Какой из принципов НЕ относится к Agile-манифесту?
А) Люди и взаимодействие важнее процессов и инструментов
Б) Работающий продукт важнее исчерпывающей документации
В) Строгое следование плану важнее реагирования на изменения
Г) Сотрудничество с заказчиком важнее контрактных условий

5. Что такое итерация в контексте разработки ПО?
А) Один полный цикл разработки, который приводит к созданию рабочей версии продукта
Б) Этап тестирования
В) Сбор требований
Г) Внедрение системы

6. Какая модель ЖЦ предполагает поставку продукта частями (инкрементами), где каждая часть добавляет новую функциональность?
А) Каскадная
Б) Инкрементная
В) Спиральная
Г) Чистая вода

7. Какой из факторов НЕ влияет на выбор модели жизненного цикла?
А) Сложность проекта
Б) Квалификация команды
В) Язык программирования
Г) Требования к надежности

8. В каскадной модели возврат на предыдущий этап…
А) Разрешен в любой момент
Б) Запрещен или крайне затруднен
В) Обязателен после каждого этапа
Г) Осуществляется только при тестировании
9. Что такое компонент в компонентно-ориентированном программировании?
А) Класс, реализующий определенный интерфейс
Б) Независимая, заменяемая единица программного обеспечения с четко определенным интерфейсом
В) Библиотека функций
Г) Исполняемый файл

10. Какое преимущество дает компонентно-ориентированный подход?
А) Повышение производительности за счет оптимизации кода
Б) Возможность повторного использования компонентов в разных проектах
В) Упрощение алгоритмов
Г) Снижение требований к тестированию

11. Какой стандарт компонентной разработки используется в платформе .NET?
А) COM
Б) JavaBeans
В) .NET-сборки (Assembly)
Г) CORBA

12. В чем отличие компонента от модуля?
А) Компонент не имеет интерфейса, а модуль имеет
Б) Компонент обладает свойством независимого развертывания и замены, а модуль – нет
В) Модуль всегда меньше компонента
Г) Различий нет

13. Какой принцип компонентного проектирования означает скрытие внутренней реализации и предоставление только четко определенного интерфейса?
А) Инкапсуляция

Б) Полиморфизм
В) Наследование
Г) Абстракция

14. Какая технология является примером компонентной модели для Java?
А) COM
Б) CORBA
В) JavaBeans (и EJB)
Г) ActiveX

15. Что такое слабая связанность (loose coupling) компонентов?
А) Компоненты сильно зависят друг от друга
Б) Компоненты минимально зависят друг от друга, взаимодействуя через стандартные интерфейсы
В) Компоненты используют общие глобальные переменные
Г) Компоненты не могут общаться между собой

16. Какое из перечисленных НЕ является преимуществом компонентного подхода?
А) Упрощение разработки за счет повторного использования
Б) Легкость обновления отдельных частей системы
В) Увеличение производительности за счет уменьшения числа вызовов
Г) Возможность распределенной разработки

17. Какой этап разработки ПО включает сбор, анализ и документирование потребностей пользователей?
А) Проектирование
Б) Кодирование
В) Анализ требований
Г) Тестирование

18. На каком этапе создается архитектура системы и определяются компоненты и их взаимодействие?
А) Анализ требований
Б) Проектирование (архитектурное)
В) Реализация
Г) Сопровождение

19. Какие виды тестирования проверяют взаимодействие между отдельными модулями системы?
А) Модульное тестирование
Б) Интеграционное тестирование
В) Системное тестирование
Г) Приемочное тестирование

20. Этап ввода системы в эксплуатацию, установка на рабочие места пользователей и переход на новую систему называется:
А) Сопровождение
Б) Внедрение (развертывание)
В) Проектирование
Г) Тестирование

21. Какой вид сопровождения связан с исправлением ошибок, обнаруженных после сдачи в эксплуатацию?

А) Адаптивное сопровождение
Б) Корректирующее сопровождение
В) Совершенствующее сопровождение
Г) Профилактическое сопровождение

22. Что такое нефункциональные требования?
А) Требования к функциям, которые система должна выполнять
Б) Требования к качеству системы, таким как производительность, безопасность, надежность
В) Требования к интерфейсу пользователя
Г) Требования к аппаратному обеспечению

23. На каком этапе создаются диаграммы классов, диаграммы последовательности и другие модели?
А) Анализ требований
Б) Проектирование (детальное)
В) Реализация
Г) Тестирование

24. Какой этап включает написание исходного кода программы?
А) Проектирование
Б) Реализация (кодирование)
В) Тестирование
Г) Внедрение

25. Что такое приемочное тестирование?
А) Тестирование отдельных функций программы
Б) Тестирование всей системы в целом
В) Тестирование, проводимое с участием заказчика для подтверждения соответствия требованиям
Г) Тестирование производительности

26. Какой этап ЖЦ выполняется после внедрения и включает доработки под изменяющиеся условия?
А) Тестирование
Б) Сопровождение
В) Проектирование
Г) Кодирование

27. В какой модели ЖЦ тестирование начинается после завершения кодирования, а не параллельно?
А) В итеративной
Б) В каскадной
В) В Agile
Г) В спиральной

28. Какой этап предшествует этапу проектирования в классическом ЖЦ?
А) Тестирование
Б) Сбор требований (анализ)
В) Внедрение
Г) Сопровождение

29. Что такое «прототипирование» в контексте этапов разработки?
А) Создание рабочей модели для уточнения требований и проверки решений
Б) Финальное тестирование
В) Написание технической документации
Г) Оптимизация кода

30. Какой вид сопровождения связан с модернизацией системы и добавлением новых функций?
А) Корректирующее
Б) Адаптивное
В) Совершенствующее (эволюционное)
Г) Профилактическое

КЛЮЧ К ТЕСТУ

№ вопроса	Правильный ответ	№ вопроса	Правильный ответ
1	Б	16	В
2	Б	17	В
3	В	18	Б
4	В	19	Б
5	А	20	Б
6	Б	21	Б
7	В	22	Б
8	Б	23	Б
9	Б	24	Б
10	Б	25	В
11	В	26	Б
12	Б	27	Б
13	А	28	Б
14	В	29	А
15	Б	30	В

Лабораторные занятия № 1-2:

1. Анализ предметной области. Разработка и оформление технического задания.
2. Изучение работы в системе контроля версий

Тема 1.2. Структурное, объектно-ориентированное и событийно-управляемое программирование

1. Задание для устного опроса по темам

1. Дайте определение структурного программирования. Назовите его основные принципы (последовательность, ветвление, циклы) и главное ограничение (отказ от оператора безусловного перехода goto).
2. В чем заключаются преимущества структурного подхода перед неструктурированным (спагетти-код)? Приведите пример, когда использование goto могло бы усложнить понимание программы.
3. Можно ли считать структурное программирование полностью изолированным от других парадигм? Как оно сочетается с модульным программированием?
4. Приведите примеры алгоритмов с константной, линейной, квадратичной и логарифмической сложностью. Объясните, как выбор алгоритма влияет на производительность при увеличении объема входных данных.
5. Какие факторы, помимо теоретической оценки, могут влиять на фактическое время

выполнения алгоритма (архитектура процессора, язык реализации, оптимизации)?

6. Перечислите основные способы документирования алгоритмов (блок-схемы, псевдокод, диаграммы, комментарии, формальные спецификации). В каких ситуациях каждый из них наиболее уместен?
7. Какие требования предъявляются к качественной документации алгоритма? Как документирование помогает при сопровождении и модификации ПО?
8. Чем отличается документирование алгоритмов для разработчика от документирования для конечного пользователя? Приведите примеры.
9. Перечислите основные принципы ООП (инкапсуляция, наследование, полиморфизм, абстракция). Раскройте суть каждого принципа и приведите примеры из реального мира.
10. Чем класс отличается от объекта? Как связаны понятия «экземпляр класса» и «состояние объекта»?
11. Какие преимущества дает ООП при разработке крупных программных систем по сравнению с процедурным подходом?
12. Что такое статические (классовые) поля и методы? В чем их принципиальное отличие от нестатических (экземплярных) членов?
13. Когда целесообразно использовать статические методы и поля? Приведите примеры типовых ситуаций (например, счетчики экземпляров, фабричные методы, утилитные функции).
14. Какие ограничения накладываются на статические члены (невозможность обращения к нестатическим членам, особенности инициализации, потоко-безопасность)?
15. Объясните механизм виртуальных методов и позднего связывания (динамический полиморфизм). Как он позволяет вызывать переопределенный метод через ссылку на базовый класс?
16. В чем разница между переопределением (override) и сокрытием (hide) методов? Как это связано с модификаторами virtual, override, new (в C#) или аналогичными в других языках?
17. Что такое абстрактные классы и интерфейсы? Как они поддерживают полиморфизм и в каких случаях предпочтительнее использовать каждый из них?
18. Дайте определение метакласса. Чем метакласс отличается от обычного класса? В каких языках программирования метаклассы реализованы на уровне языка?
19. Какова роль метакласса в процессе создания классов? Какие задачи можно решить с помощью метаклассов (например, автоматическое добавление атрибутов, валидация структуры класса)?
20. В чем состоит различие между метаклассом и декоратором класса? Когда использование метакласса предпочтительнее?
21. Опишите, как можно создать метакласс в Python (на примере наследования от type и переопределения __new__ или __init__).
22. Приведите практический пример использования метакласса: например, для автоматической регистрации всех подклассов в словаре, для проверки наличия обязательных методов или для реализации синглтона.
23. Какие риски и сложности возникают при использовании метаклассов (сложность отладки, конфликты, снижение читаемости) и как их минимизировать?
24. Что такое событийно-управляемое программирование? В чем его отличие от

традиционного процедурного (последовательного) подхода?

25. Опишите модель событийного цикла (event loop) и роль очереди событий. Как обработчики событий связываются с источниками событий?
26. Каким образом событийно-управляемый подход поддерживает асинхронную обработку и неблокирующий ввод-вывод? Приведите примеры из GUI-фреймворков или веб-разработки.
27. Дайте определение элемента управления (виджета) в графическом интерфейсе. Приведите примеры основных элементов (кнопка, текстовое поле, список, флажок, переключатель) и опишите их назначение.
28. Какие ключевые свойства (например, текст, размер, состояние) и события (щелчок, изменение текста, выбор элемента) характерны для элементов управления? Как происходит их настройка?
29. В чем различие между контейнерными элементами (панели, группы) и простыми элементами управления? Как организуется иерархия вложенности?
30. Что такое диалоговое окно? Какие существуют виды диалогов: модальные и немодальные, стандартные (открытия/сохранения файла) и настраиваемые пользовательские?
31. Как организован обмен данными между диалоговым окном и основной программой? Опишите способы: возврат значения через результат (модальный), использование ссылок на данные, обратные вызовы (колбэки) или события.
32. Когда следует использовать модальное диалоговое окно, а когда — немодальное? Приведите примеры типичных сценариев применения каждого вида.
33. Что такое обработчик события (event handler)? Как связать событие элемента управления с кодом обработчика?
34. В каких ситуациях используют анонимные обработчики (лямбда-выражения) вместо именованных методов? Каковы преимущества и недостатки каждого подхода?
35. В чем заключаются основы работы с графикой в программировании? Назовите базовые примитивы и операции. Как обычно организуется отрисовка?

Тестовые вопросы

Тема 1.2. Структурное, объектно-ориентированное и событийно-управляемое программирование

Вариант 1

1. Какой из принципов НЕ относится к структурному программированию?
А) Последовательное выполнение операторов
Б) Использование циклов и ветвлений
В) Активное применение оператора безусловного перехода goto
Г) Разбиение программы на подпрограммы (функции)

2. Какой асимптотической оценкой характеризуется алгоритм бинарного поиска в отсортированном массиве?
А) O(1)
Б) O(log n)
В) O(n)
Г) O(n log n)

3. Что из перечисленного является способом документирования алгоритмов на уровне исходного кода?
А) Блок-схема
Б) Псевдокод
В) Комментарии в коде
Г) Диаграмма классов

4. Какое понятие в ООП означает скрытие внутренних деталей объекта и предоставление доступа только через открытые методы?
А) Наследование
Б) Инкапсуляция
В) Полиморфизм
Г) Абстракция

5. Статическое поле класса в языке программирования (например, C# или Java) создается:
А) Для каждого экземпляра класса отдельно
Б) В единственном экземпляре для всего класса и разделяется всеми объектами
В) Только внутри статических методов
Г) Только для хранения констант

6. Виртуальные методы в ООП позволяют реализовать:
А) Статическое связывание (раннее)
Б) Динамическое связывание (позднее) и полиморфизм
В) Перегрузку операторов
Г) Множественное наследование

7. Что такое метакласс?
А) Класс, который не может иметь экземпляров
Б) Класс, экземплярами которого являются другие классы
В) Класс, содержащий только статические методы
Г) Абстрактный класс

8. Для чего могут использоваться метаклассы в программировании?
А) Для автоматической регистрации всех подклассов
Б) Для создания пользовательских интерфейсов
В) Для обработки событий мыши
Г) Для работы с графикой

9. Какой из перечисленных языков поддерживает метаклассы на уровне синтаксиса?
А) C++
Б) Python
В) Java (без изменений байт-кода)
Г) C#

10. В событийно-управляемом программировании основным механизмом взаимодействия является:
А) Прямой вызов процедур
Б) Цикл событий (event loop) и обработчики
В) Потоки без синхронизации
Г) Блокирующие операции ввода-вывода

11. Элемент управления (виджет) в графическом интерфейсе – это:
А) Метод для обработки данных
Б) Объект, обеспечивающий взаимодействие пользователя с программой (кнопка, поле ввода и т.п.)

В) База данных
Г) Файл конфигурации

12. Какой тип диалогового окна блокирует работу с родительским окном до своего закрытия?
А) Немодальное
Б) Модальное
В) Всплывающее
Г) Системное

13. Обработчик события – это:
А) Функция или метод, вызываемый при возникновении события
Б) Свойство элемента управления
В) Тип данных
Г) Графический примитив

14. Какой из примитивов относится к базовым средствам рисования в графических библиотеках?
А) Диалоговое окно
Б) Линия
В) Событие мыши
Г) Обработчик

15. В структурном программировании циклы реализуются с помощью:
А) Оператора goto
Б) Операторов for, while, do-while
В) Рекурсивных вызовов
Г) Макросов

16. Оценка сложности алгоритма сортировки пузырьком в среднем и худшем случае составляет:
А) O(n)
Б) O(n log n)
В) $O(n^2)$
Г) $O(2^n)$

17. Что такое псевдокод в контексте документирования алгоритмов?
А) Компилируемый код на языке низкого уровня
Б) Описание алгоритма на естественном языке с элементами программирования, не требующее компиляции
В) Блок-схема
Г) Тестовый набор данных

18. Какое из утверждений о статических методах верно?
А) Они могут обращаться к нестатическим полям класса напрямую
Б) Они могут быть вызваны без создания экземпляра класса
В) Они переопределяются как виртуальные
Г) Они всегда приватны

19. Что такое абстрактный класс?
А) Класс, который нельзя наследовать
Б) Класс, содержащий хотя бы один абстрактный метод (без реализации) и предназначенный для наследования
В) Класс, все методы которого статические
Г) Класс, который не может создавать объекты

20. В каком случае используется ключевое слово override **(в C#) или** @Override **(в Java)?**
А) Для перегрузки оператора
Б) Для переопределения виртуального метода из базового класса
В) Для создания нового метода с тем же именем, но другим списком параметров
Г) Для вызова конструктора базового класса

21. Событийно-управляемая модель особенно эффективна для:
А) Научных расчетов с большим объемом данных
Б) Приложений с графическим интерфейсом пользователя (GUI)
В) Пакетной обработки транзакций
Г) Компиляции кода

22. Какой из элементов управления обычно используется для выбора одного варианта из нескольких предложенных?
А) Кнопка (Button)
Б) Флажок (CheckBox)
В) Переключатель (RadioButton)
Г) Текстовое поле (TextBox)

23. Какое диалоговое окно предоставляет пользователю возможность выбрать файл для открытия?
А) MessageBox
Б) OpenFileDialog
В) SaveFileDialog
Г) ColorDialog

24. В графических системах обычно для рисования используется объект, называемый:
А) Канва (Canvas) или контекст устройства (Graphics)
Б) Событие
В) Обработчик
Г) Менеджер компоновки

25. Какой из принципов является основой структурного программирования?
А) Модульность и использование трех базовых управляющих структур
Б) Отказ от использования подпрограмм
В) Обязательное применение объектно-ориентированных подходов
Г) Использование только рекурсии

Вариант 2

1. В структурном программировании не рекомендуется использование:
А) Условных операторов (if-else)
Б) Циклов
В) Оператора goto
Г) Подпрограмм (функций)

2. Оценка сложности O(n!) соответствует:
А) Поиску минимального элемента
Б) Сортировке пузырьком
В) Перебору всех перестановок
Г) Бинарному поиску

3. Какой вид документирования наиболее нагляден для представления блок-схемы алгоритма?
А) Текстовое описание на естественном языке
Б) Графическая схема с блоками и стрелками

В) Спецификация на формальном языке
Г) Javadoc-комментарии

4. Какой принцип ООП обеспечивает возможность использования одного и того же имени метода для разных типов объектов?
А) Инкапсуляция
Б) Наследование
В) Полиморфизм
Г) Абстракция

5. Статический метод может быть вызван:
А) Только через экземпляр класса
Б) Только из другого статического метода
В) Без создания экземпляра, по имени класса
Г) Только внутри конструктора

6. Что такое динамический полиморфизм в ООП?
А) Возможность определять несколько методов с одним именем, но разными параметрами
Б) Определение методов в базовом классе и переопределение их в производных с вызовом через ссылку на базовый класс
В) Замена метода на этапе компиляции
Г) Использование шаблонов

7. В Python метаклассом по умолчанию для всех классов является:
А) object
Б) type
В) class
Г) Meta

8. Какая задача НЕ типична для использования метаклассов?
А) Автоматическое добавление методов в класс
Б) Проверка соблюдения сигнатуры методов в дочерних классах
В) Управление памятью для объектов класса
Г) Реализация паттерна «Одиночка» (Singleton)

9. Какое ключевое слово в Python используется для создания метакласса?
А) class
Б) metaclass в определении класса
В) def
Г) import

10. В событийно-управляемых системах источники событий и обработчики связываются с помощью:
А) Прямого вызова функций
Б) Регистрации обработчиков (подписки)
В) Глобальных переменных
Г) Статических полей

11. Какой элемент управления предназначен для ввода многострочного текста?
А) TextBox (однострочный)
Б) RichTextBox / TextArea
В) ListBox
Г) ComboBox

12. Какое диалоговое окно обычно используется для отображения сообщения и кнопок «Да»/«Нет»?
А) FileDialog

Б) MessageBox
В) FontDialog
Г) PrintDialog

13. Анонимные методы (лямбда-выражения) в качестве обработчиков событий применяются для:
А) Улучшения производительности
Б) Упрощения кода при кратком описании обработки
В) Запрета использования именованных методов
Г) Увеличения безопасности

14. Для рисования закрашенных фигур в графике используется:
А) Кисть (Brush)
Б) Перо (Pen)
В) Шрифт (Font)
Г) Растровое изображение

15. Структурное программирование предполагает разбиение задачи на:
А) Классы и объекты
Б) Последовательные шаги, ветвления и циклы
В) События и обработчики
Г) Метаданные

16. Сложность алгоритма поиска в несортированном массиве методом полного перебора составляет:
А) O(1)
Б) O(log n)
В) O(n)
Г) $O(n^2)$

17. Какой из способов документирования алгоритмов позволяет наглядно отобразить поток управления?
А) Блок-схема
Б) Псевдокод
В) Комментарии
Г) Спецификация пред- и постусловий

18. Может ли статический метод обращаться к нестатическим полям класса?
А) Да, всегда
Б) Да, если поле объявлено как public
В) Нет, так как не статическое поле принадлежит конкретному объекту, а статический метод не имеет доступа к объекту без ссылки
Г) Да, через ключевое слово this

19. Абстрактный метод – это метод:
А) Который не имеет реализации и должен быть переопределен в производном классе
Б) Который имеет пустую реализацию
В) Который объявлен как статический
Г) Который является конструктором

20. Когда применяется ключевое слово new **(в C#) при объявлении метода в производном классе?**
А) Когда требуется переопределить виртуальный метод базового класса
Б) Когда требуется скрыть метод базового класса (не полиморфно)
В) Для создания нового экземпляра класса
Г) Для вызова конструктора

21. Какая архитектура часто используется в событийно-управляемых приложениях?
А) MVC (Model-View-Controller)
Б) Пакетная обработка
В) Компилятор
Г) Интерпретатор

22. Элемент «Список» (ListBox) позволяет пользователю:
А) Вводить числа
Б) Выбирать один или несколько элементов из предопределенного набора
В) Выбирать цвет
Г) Открывать файл

23. Какое из перечисленных диалоговых окон в Windows используется для выбора шрифта?
А) OpenFileDialog
Б) FontDialog
В) ColorDialog
Г) FolderBrowserDialog

24. Обработчик события в GUI обычно имеет параметр sender **– это:**
А) Объект, сгенерировавший событие
Б) Данные о событии
В) Код возврата
Г) Имя метода

25. Основное отличие событийно-управляемого программирования от процедурного – это:
А) Использование только рекурсивных функций
Б) Отсутствие циклов
В) Программа не имеет линейного потока, реагирует на внешние события в порядке их поступления
Г) Использование только глобальных переменных

Ключ к тестам

Вариант 1		Вариант 2	
№ вопроса	Правильный ответ	№ вопроса	Правильный ответ
1	В	1	В
2	Б	2	В
3	В	3	Б
4	Б	4	В
5	Б	5	В
6	Б	6	Б
7	Б	7	Б
8	А	8	В
9	Б	9	Б
10	Б	10	Б
11	Б	11	Б
12	Б	12	Б
13	А	13	Б
14	Б	14	А
15	Б	15	Б
16	В	16	В

17	Б	17	А
18	Б	18	В
19	Б	19	А
20	Б	20	Б
21	Б	21	А
22	В	22	Б
23	Б	23	Б
24	А	24	А
25	А	25	В

Лабораторные занятия № 3-19:

1. Использование инструментальных средств документирования алгоритмов программ
2. Определение сложности алгоритмов сортировки, поиска, рекурсивных и эвристических алгоритмов
3. Работа с классами и перегрузка методов
4. Определение операций в классе
5. Создание наследованных классов
6. Работа с объектами через интерфейсы
7. Использование стандартных интерфейсов
8. Работа с типом данных структура, использование коллекций
9. Операции со списками
10. Разработка приложения с использованием текстовых компонентов
11. Разработка приложения с несколькими формами
12. Разработка приложения с не визуальными компонентами
13. Разработка игрового приложения
14. Разработка программных модулей для работы с датой и временем. Работа через систему контроля версий
15. Разработка программных модулей с использованием паттернов проектирования. Работа через систему контроля версий
16. Навигация по файловой системе. Чтение и запись файлов. Работа с потоками. Работа с изолированным хранилищем
17. Работа с большими объемами текста. Кодирование и декодирование строк. Построение регулярных выражений. Чтение и запись файлов в разных кодировках

Тема 1.3. Модульный принцип разработки ПО. Основы работы с базами данных

1. Задание для устного опроса по темам

1. Дайте определение оптимизации модуля в контексте разработки ПО. Какие цели преследует оптимизация на уровне модулей?
2. Перечислите основные критерии, по которым оценивается качество модуля (эффективность, читаемость, сопровождаемость, тестируемость, повторная используемость). Раскройте суть каждого критерия.
3. В чем заключается компромисс между производительностью и читаемостью кода при оптимизации модуля? Приведите пример, когда чрезмерная оптимизация может ухудшить сопровождаемость.
4. Как оптимизация модулей влияет на общую архитектуру системы и на взаимосвязи между модулями? Какие риски возникают при преждевременной оптимизации?
5. Что такое информационная закрытость (information hiding) и как она связана с принципом инкапсуляции в ООП? В чем ее роль в проектировании модульных систем?
6. Какие механизмы в языках программирования поддерживают информационную закрытость (модификаторы доступа, интерфейсы, абстракция)? Приведите примеры из известных языков.

7. Почему информационная закрытость считается важным средством управления сложностью? Как она облегчает модификацию и тестирование системы?
8. Приведите пример ситуации, когда нарушение информационной закрытости приводит к проблемам при сопровождении (например, изменение внутренней структуры класса ломает зависимый код).
9. Дайте определение связности (cohesion) модуля. Почему высокая связность считается желательным свойством?
10. Перечислите виды связности по убыванию желательности (от функциональной до случайной). Кратко охарактеризуйте каждый вид и приведите примеры.
11. Чем отличается функциональная связность от последовательной? Приведите примеры модулей с каждым из этих типов.
12. Как определить, что модуль обладает логической или случайной связностью? Какие проблемы это создает и как можно улучшить структуру?
13. Как связность связана с принципом единственной ответственности (Single Responsibility Principle)? Обоснуйте.
14. Что такое сцепление (coupling) между модулями? Почему низкое сцепление считается положительным качеством архитектуры?
15. Перечислите основные типы сцепления (содержательное, общее, внешнее, управляющее, штамповое, по данным) и расположите их от наиболее нежелательного к наиболее приемлемому.
16. В чем разница между сцеплением по данным и сцеплением по управлению? Приведите примеры, когда управляющее сцепление может быть оправдано.
17. Какие архитектурные приемы помогают снизить сцепление между модулями (внедрение зависимостей, использование интерфейсов, событийная модель)?
18. Как сцепление влияет на тестируемость и возможность повторного использования модулей? Приведите пример высокой связанности, создающей трудности при модификации.
19. Что понимается под специальными библиотеками в программировании? Чем они отличаются от стандартных библиотек и фреймворков?
20. Приведите примеры специальных библиотек для различных областей (научные вычисления, работа с графикой, машинное обучение, сетевое взаимодействие). Опишите их назначение.
21. Какие преимущества и риски связаны с использованием сторонних специальных библиотек в проекте? Как оценить качество и надежность библиотеки перед интеграцией?
22. Как управление зависимостями (например, с помощью pip, npm, Maven) связано с использованием специальных библиотек? Какие проблемы версионности возникают?
23. Перечислите основные группы операторов языка SQL (DDL, DML, DCL, TCL). Какие задачи решает каждая группа?
24. Назовите основные типы данных SQL (числовые, строковые, дата/время). В каких ситуациях используется тип VARCHAR и чем он отличается от CHAR?
25. Опишите синтаксис оператора SELECT с ключевыми словами WHERE, ORDER BY, GROUP BY и HAVING. Для чего нужны агрегатные функции (COUNT, SUM, AVG, MAX, MIN)?
26. Что такое подзапросы (вложенные запросы) и как они используются? Приведите примеры подзапросов в WHERE и в FROM.
27. Какие операторы SQL используются для фильтрации и сортировки данных? Чем отличается WHERE от HAVING?
28. Опишите синтаксис оператора CREATE TABLE. Какие ограничения целостности могут быть заданы при создании таблицы (первичный ключ, внешний ключ, уникальность, проверочные ограничения, NOT NULL)?

29. Как с помощью ALTER TABLE можно изменить структуру существующей таблицы (добавить/удалить столбец, изменить тип, добавить ограничение)? Приведите примеры.
30. Какие операторы используются для вставки (INSERT), обновления (UPDATE) и удаления (DELETE) данных? В чем отличие DELETE от TRUNCATE?
31. Объясните, что такое транзакции в SQL (команды BEGIN, COMMIT, ROLLBACK). Почему они важны для целостности данных?
32. Как выполняется объединение таблиц с помощью операторов JOIN (INNER, LEFT, RIGHT, FULL)? Приведите пример запроса с несколькими JOIN.
33. Что такое Python DB-API и какова его цель? Какие версии DB-API существуют (DB-API 2.0)?
34. Опишите основные этапы работы с базой данных через DB-API: установка соединения, создание курсора, выполнение запроса, получение результатов, закрытие соединения.
35. Что такое параметризованные запросы и почему они важны для безопасности (защита от SQL-инъекций)? Приведите пример использования плейсхолдеров в DB-API (например, ? или %s).
36. Как получить метаданные о результате запроса (например, названия столбцов) при использовании DB-API? Какие методы курсора предоставляют информацию?
37. Какие популярные модули реализуют DB-API для различных СУБД (sqlite3, psycopg2, MySQLdb, pyodbc)? В чем их общие черты и различия?
38. Дайте определение ORM (Object-Relational Mapping). Какую проблему решает этот подход?
39. Назовите основные преимущества использования ORM перед прямыми SQL-запросами (абстракция, автоматическая генерация SQL, поддержка разных СУБД, удобство работы с объектами). Какие недостатки при этом возникают (потеря производительности, сложные запросы)?
40. Опишите, как работает ORM на примере SQLAlchemy или Django ORM: как определяются модели, как выполняются выборки, вставки и обновления, как описываются связи между таблицами.

Тестовые вопросы

Тема 1.3. Модульный принцип разработки ПО. Основы работы с базами данных

Вариант 1

1. Что из перечисленного НЕ является критерием оптимизации модуля?
А) Эффективность
Б) Читаемость кода
В) Количество строк кода
Г) Сопровождаемость

2. Информационная закрытость (information hiding) означает:
А) Запрет на использование глобальных переменных
Б) Сокрытие внутренней реализации модуля и предоставление только интерфейса
В) Полное отсутствие документации
Г) Использование только приватных методов

3. Какой вид связности является наиболее желательным?
А) Случайная связность
Б) Логическая связность

В) Функциональная связность
Г) Временная связность

4. Какой тип сцепления считается наименее желательным (самым плохим)?
А) Сцепление по данным
Б) Сцепление по управлению
В) Содержательное сцепление
Г) Штамповое сцепление

5. Что такое специальные библиотеки?
А) Библиотеки, входящие в стандартную поставку языка
Б) Библиотеки, предназначенные для решения узкоспециализированных задач (например, машинное обучение)
В) Библиотеки, написанные на ассемблере
Г) Только библиотеки для работы с графикой

6. Какая команда SQL используется для извлечения данных из таблицы?
А) INSERT
Б) UPDATE
В) SELECT
Г) DELETE

7. Какой тип данных в SQL предназначен для хранения строк фиксированной длины?
А) VARCHAR
Б) CHAR
В) TEXT
Г) BLOB

8. Какое ограничение целостности гарантирует уникальность значений в столбце?
А) FOREIGN KEY
Б) PRIMARY KEY или UNIQUE
В) CHECK
Г) DEFAULT

9. Какой оператор используется для добавления новых строк в таблицу?
А) INSERT INTO
Б) ADD
В) CREATE
Г) ALTER

10. В Python DB-API для выполнения запроса и получения результата обычно используется:
А) Соединение (connection)
Б) Курсор (cursor)
В) Менеджер контекста
Г) Метаданные

11. Что такое параметризованный запрос в контексте DB-API?
А) Запрос с фиксированными значениями
Б) Запрос, в котором используются плейсхолдеры для подстановки параметров, что защищает от SQL-инъекций
В) Запрос, который выполняется только один раз
Г) Запрос, возвращающий параметры подключения

12. Как называется подход, при котором объекты программы отображаются на таблицы реляционной базы данных?
А) Реляционное программирование
Б) Объектно-реляционное отображение (ORM)

В) Объектное программирование
Г) Отображение данных

13. Какое преимущество дает ORM?
А) Повышение производительности запросов по сравнению с прямым SQL
Б) Позволяет работать с БД через объекты, абстрагируя от синтаксиса SQL
В) Устраняет необходимость в индексах
Г) Всегда генерирует оптимальные SQL-запросы

14. Чем отличается связность (cohesion) от сцепления (coupling)?
А) Связность — внутри модуля, сцепление — между модулями
Б) Связность — между модулями, сцепление — внутри модуля
В) Это одно и то же
Г) Связность важна только для ООП

15. Какая из перечисленных характеристик относится к модулю с высокой связностью?
А) Модуль выполняет множество несвязанных задач
Б) Модуль выполняет одну четко определенную задачу
В) Модуль сильно зависит от других модулей
Г) Модуль содержит много глобальных переменных

16. Что такое информационная закрытость в контексте проектирования модулей?
А) Запрет на импорт модуля
Б) Сокрытие деталей реализации, чтобы изменения внутри модуля не затрагивали другие модули
В) Полная изоляция модуля от других
Г) Отсутствие документации

17. Какая команда SQL используется для изменения структуры таблицы (добавление столбца)?
А) UPDATE
Б) ALTER TABLE
В) MODIFY
Г) CHANGE

18. Что делает оператор TRUNCATE?
А) Удаляет все строки из таблицы и сбрасывает счетчик автоинкремента
Б) Удаляет только одну строку
В) Изменяет структуру таблицы
Г) Добавляет новый столбец

19. В DB-API для получения одной строки результата используется метод:
А) fetchone()
Б) fetchall()
В) fetchmany()
Г) next()

20. Какой из перечисленных модулей реализует DB-API для PostgreSQL?
А) sqlite3
Б) psycopg2
В) MySQLdb
Г) pyodbc

21. Что такое "ленивая загрузка" (lazy loading) в ORM?
А) Загрузка всех данных сразу
Б) Загрузка связанных данных только при первом обращении к ним
В) Загрузка данных в фоновом режиме
Г) Отказ от загрузки данных

22. Какой оператор SQL используется для объединения двух таблиц по общему столбцу?
А) JOIN
Б) UNION
В) INTERSECT
Г) MERGE

23. Какое сцепление возникает, когда один модуль передает другому управляющий флаг, изменяющий логику поведения?
А) Сцепление по данным
Б) Сцепление по управлению
В) Штамповое сцепление
Г) Внешнее сцепление

24. В SQL для фильтрации групп после агрегации используется:
А) WHERE
Б) HAVING
В) GROUP BY
Г) ORDER BY

25. Какая операция в ORM обычно используется для сохранения объекта в базу данных?
А) add() или save()
Б) commit()
В) rollback()
Г) delete()

Вариант 2

1. Какой критерий оптимизации модуля связан с легкостью внесения изменений?
А) Эффективность
Б) Сопровождаемость
В) Повторная используемость
Г) Тестируемость

2. Что из перечисленного нарушает информационную закрытость?
А) Использование интерфейса модуля
Б) Прямой доступ к внутренним данным модуля извне
В) Использование приватных методов
Г) Документирование интерфейса

3. Какой вид связности характеризуется тем, что модуль выполняет несколько логически связанных операций, но они выполняются в определенной последовательности (выход одной является входом для другой)?
А) Функциональная
Б) Последовательная
В) Коммуникационная
Г) Процедурная

4. Какой тип сцепления считается приемлемым, когда модули обмениваются только простыми данными (скалярными значениями)?
А) Сцепление по данным
Б) Сцепление по управлению
В) Штамповое сцепление
Г) Общее сцепление

5. Что из перечисленного является примером специальной библиотеки?
А) Стандартная библиотека ввода-вывода
Б) Библиотека для машинного обучения TensorFlow
В) Библиотека для работы со строками
Г) Библиотека для работы с датами

6. Какая команда SQL используется для удаления всей таблицы вместе со структурой?
А) DELETE TABLE
Б) DROP TABLE
В) TRUNCATE TABLE
Г) REMOVE TABLE

7. В SQL для сортировки результатов запроса используется:
А) GROUP BY
Б) ORDER BY
В) SORT BY
Г) HAVING

8. Что такое первичный ключ (PRIMARY KEY) в таблице?
А) Столбец, который может содержать NULL
Б) Столбец (или комбинация), уникально идентифицирующий каждую строку
В) Столбец со значениями по умолчанию
Г) Столбец, используемый для внешнего ключа

9. Какой оператор SQL используется для обновления существующих записей?
А) UPDATE
Б) MODIFY
В) CHANGE
Г) ALTER

10. В Python DB-API для установки соединения с БД используется:
А) cursor()
Б) connect()
В) execute()
Г) fetch()

11. Какая проблема решается использованием параметризованных запросов?
А) Ускорение выполнения
Б) Предотвращение SQL-инъекций
В) Упрощение синтаксиса
Г) Автоматическое создание таблиц

12. Что такое миграции в контексте ORM?
А) Перенос данных между таблицами
Б) Управление изменениями схемы базы данных через код моделей
В) Перемещение объектов в памяти
Г) Копирование БД на другой сервер

13. Какой метод в ORM обычно используется для получения всех записей, соответствующих условию?
А) filter() или where()
Б) get()
В) first()
Г) all()

14. Что такое "сцепление по данным"?
А) Модуль передает другому модулю управляющий сигнал
Б) Модули обмениваются только простыми структурами данных, не влияя на логику друг друга
В) Модули используют общие глобальные переменные
Г) Модуль включает в себя реализацию другого модуля

15. Какой из видов связности является наихудшим?
А) Функциональная
Б) Случайная
В) Последовательная
Г) Логическая

16. Информационная закрытость способствует:
А) Увеличению производительности
Б) Уменьшению сложности сопровождения
В) Увеличению связанности между модулями
Г) Уменьшению повторного использования

17. Какая команда SQL добавляет новый столбец в существующую таблицу?
А) ALTER TABLE ... ADD COLUMN
Б) CREATE COLUMN
В) INSERT COLUMN
Г) MODIFY TABLE

18. В SQL для объединения строк из двух таблиц без дублирования используется:
А) UNION
Б) UNION ALL
В) JOIN
Г) INTERSECT

19. В DB-API после выполнения запроса, который изменяет данные, для фиксации транзакции обычно вызывается:
А) commit()
Б) rollback()
В) close()
Г) execute()

20. Какой модуль DB-API входит в стандартную библиотеку Python для работы с SQLite?
А) sqlite3
Б) pysqlite
В) sqlite
Г) sqlite2

21. Что такое "жадная загрузка" (eager loading) в ORM?
А) Загрузка связанных данных сразу вместе с основным запросом
Б) Отложенная загрузка данных
В) Загрузка данных только при необходимости
Г) Загрузка данных в кэш

22. Какое ограничение целостности в SQL обеспечивает ссылочную целостность между таблицами?
А) PRIMARY KEY
Б) FOREIGN KEY
В) UNIQUE
Г) CHECK

23. Какой тип сцепления возникает, когда модули используют общие глобальные переменные?
А) Сцепление по данным
Б) Сцепление по управлению
В) Общее сцепление
Г) Содержательное сцепление

24. Какой оператор SQL используется для группировки записей перед применением агрегатных функций?
А) ORDER BY
Б) GROUP BY
В) HAVING
Г) WHERE

25. В ORM для удаления объекта из базы данных обычно используется метод:
А) delete()
Б) remove()
В) drop()
Г) erase()

Ключ к тестам:

Ключ к варианту 1

Вариант 1		**Вариант 2**	
№ вопроса	Правильный ответ	№ вопроса	Правильный ответ
1	В	1	Б
2	Б	2	Б
3	В	3	Б
4	В	4	А
5	Б	5	Б
6	В	6	Б
7	Б	7	Б
8	Б	8	Б
9	А	9	А
10	Б	10	Б
11	Б	11	Б
12	Б	12	Б
13	Б	13	А
14	А	14	Б
15	Б	15	Б
16	Б	16	Б
17	Б	17	А
18	А	18	А
19	А	19	А
20	Б	20	А
21	Б	21	А
22	А	22	Б
23	Б	23	В
24	Б	24	Б
25	А	25	А

Лабораторные занятия № 20-33:

1. Разработка приложения с функционально связанными модулями
2. Разработка приложения с информационно связанными модулями
3. Разработка приложения с коммуникативно связанными модулями
4. Разработка приложения с различными типами сцепления модулей
5. Определение меры сопряжения между модулями
6. Анализ базового синтаксиса SQL, создание таблицы
7. Создание приложения с БД
8. Создание запросов и хранимых процедур к БД
9. Разработка стилей для приложения для улучшения взаимодействия с пользователем
10. Разработка модулей для работы с изображениями
11. Разработка модулей для представления информации в виде графиков и диаграмм
12. Разработка модулей для работы аудио и видео
13. Реализация загрузки данных из интернета в фоновом режиме без блокировки основного потока приложения.
14. Разработка формы регистрации с элементами ввода и проверкой корректности введенных данных.

Тема 1.4. Конструирование ПО

1. Задание для устного опроса по темам

1. Перечислите основные правила (эвристики) разработки удобных пользовательских интерфейсов (например, по Нильсену, Шнейдерману или ГОСТ). Раскройте суть 3–4 из них.
2. Какие требования предъявляются к качественному интерфейсу с точки зрения юзабилити (удобство обучения, эффективность использования, запоминаемость, ошибки, субъективная удовлетворенность)?
3. Что включает в себя анализ интерфейса? Какие методы используются для оценки интерфейса (экспертная оценка, юзабилити-тестирование, опросы, анализ логов)?
4. Как требования интерфейса вытекают из требований к программному продукту? Приведите пример влияния нефункциональных требований (быстродействие, безопасность) на дизайн интерфейса.
5. Как учитываются особенности целевой аудитории при проектировании интерфейса? Что такое «персонажи» и сценарии использования (use cases) в контексте проектирования UI?
6. Дайте определение паттерна проектирования в разработке ПО. Какова цель использования паттернов?
7. Какие три основные группы паттернов выделяют по назначению (порождающие, структурные, поведенческие)? Кратко охарактеризуйте задачи, решаемые каждой группой.
8. Чем паттерн отличается от готового компонента или библиотеки? Приведите примеры ситуаций, когда применение паттерна оправдано.
9. Какие преимущества и возможные недостатки связаны с использованием паттернов (например, усложнение кода, необходимость изучения)?
10. Дайте определение рефакторинга. В чем его отличие от оптимизации и от исправления ошибок?

11. Назовите основные причины для проведения рефакторинга (улучшение читаемости, уменьшение дублирования, упрощение добавления новых функций, повышение сопровождаемости).
12. Какие цели преследует рефакторинг с точки зрения качества кода? Как он влияет на внутренние метрики (связность, сцепление, размер методов)?
13. Когда рефакторинг **не следует** выполнять (например, в критической стадии перед релизом, в плохо протестированном коде)?
14. Как рефакторинг связан с тестированием? Почему наличие автоматических тестов считается обязательным условием безопасного рефакторинга?
15. Какие приемы рефакторинга относятся к упрощению вызовов методов (например, Rename Method, Add Parameter, Remove Parameter, Separate Query from Modifier, Parameterize Method, Replace Parameter with Explicit Methods)?
16. В чем суть рефакторинга «Separate Query from Modifier»? Почему не рекомендуется совмещать метод, который возвращает значение, с методом, который изменяет состояние?
17. Что такое «Replace Parameter with Explicit Methods»? Приведите пример, когда один параметр-флаг заменяется несколькими отдельными методами.
18. Что понимается под «обобщением» (generalization) в контексте рефакторинга? Какие методы обобщения существуют (например, Pull Up Method, Push Down Method, Extract Interface, Form Template Method)?
19. В чем разница между «Pull Up Method» и «Push Down Method»? В каких случаях применяется каждый из них при работе с иерархией классов?
20. Что такое экстремальное программирование (XP – Extreme Programming)? Каковы его основные ценности и принципы (коммуникация, простота, обратная связь, смелость, уважение)?
21. Перечислите ключевые практики XP: парное программирование, непрерывная интеграция, тестирование (TDD), частые релизы, метафора системы, коллективное владение кодом. Раскройте суть 2–3 практик.
22. Как XP связан с Agile-методологиями? В чем его отличие от Scrum?
23. Что такое «игровое планирование» (Planning Game) и как оно используется в XP?
24. Какие роли выделяются в команде XP и как строится взаимодействие с заказчиком (он же «клиент» в терминах XP)?
25. Дайте определение обратного инжиниринга (reverse engineering). В чем его отличие от обычного изучения существующей системы?
26. Какие цели преследует обратный инжиниринг в программной инженерии (восстановление документации, анализ работы устаревших систем, обнаружение ошибок, борьба с вирусами, повторное использование компонентов)?
27. Какие уровни абстракции могут быть восстановлены при обратном инжиниринге (исходный код, архитектура, бизнес-логика, требования)?
28. В каких областях чаще всего применяется обратный инжиниринг (например, при переезде на новую платформу, при отсутствии документации, при анализе конкурирующих продуктов)?
29. Каковы правовые и этические ограничения при проведении обратного инжиниринга?
30. Перечислите основные методики (подходы) к обратному инжинирингу: анализ исходного кода, анализ бинарного кода (дизассемблирование), трассировка

выполнения, анализ логов, дата-майнинг, использование специализированных инструментов (например, IDA Pro, Ghidra, decompilers).

31. В чем разница между статическим и динамическим анализом при обратном инжиниринге? Какие задачи решает каждый из них?
32. Что такое «прямой инжиниринг» и «обратный инжиниринг» в контексте моделей и диаграмм? Как обратный инжиниринг помогает восстанавливать UML-диаграммы из кода?
33. Какие инструменты автоматизации используются для извлечения метаданных и построения архитектурных моделей?
34. Какие сложности возникают при обратном инжиниринге высокоуровневых языков по сравнению с низкоуровневыми (C vs Java)?
35. Дайте определение реинжиниринга программного обеспечения. Чем реинжиниринг отличается от обратного инжиниринга и от рефакторинга?
36. Каковы основные цели реинжиниринга (повышение производительности, адаптация к новым платформам, улучшение структуры, снижение сложности сопровождения)?
37. Опишите типовой процесс реинжиниринга: обратный инжиниринг → анализ и восстановление требований → реструктуризация → повторное внедрение.
38. В каких случаях целесообразно проводить реинжиниринг, а не разрабатывать систему «с нуля»? Каковы критерии принятия решения?
39. Приведите пример успешного реинжиниринга известной программной системы. Какие проблемы были решены?
40. Какие риски связаны с реинжинирингом (дороговизна, потеря функциональности, нарушение сроков, сопротивление пользователей)?

Тестовые вопросы

Тема 1.4. Конструирование ПО

Вариант 1

1. Какая из перечисленных эвристик юзабилити относится к принципу «соответствие системы реальному миру»?
А) Использование понятных пользователю терминов и понятий, а не технического жаргона
Б) Предоставление возможности отмены действий
В) Минимизация загрузки памяти пользователя
Г) Единообразие в интерфейсе

2. Какое требование к интерфейсу обеспечивает быстрое освоение работы с системой новыми пользователями?
А) Эффективность использования
Б) Удобство обучения
В) Запоминаемость
Г) Субъективная удовлетворенность

3. Что из перечисленного является методом анализа интерфейса?
А) Тестирование производительности
Б) Юзабилити-тестирование с участием пользователей
В) Компиляция кода
Г) Оптимизация запросов

4. Как называется паттерн проектирования, который гарантирует наличие единственного экземпляра класса и предоставляет глобальную точку доступа к нему?
А) Factory Method

Б) Singleton
В) Prototype
Г) Builder

5. Какая группа паттернов решает задачи эффективного создания объектов?
А) Структурные
Б) Поведенческие
В) Порождающие
Г) Архитектурные

6. Какой порождающий паттерн позволяет создавать семейства связанных объектов без указания их конкретных классов?
А) Abstract Factory
Б) Factory Method
В) Builder
Г) Prototype

7. Какой структурный паттерн обеспечивает интерфейс для взаимодействия с подсистемой, упрощая работу с ней?
А) Adapter
Б) Facade
В) Proxy
Г) Decorator

8. Какой поведенческий паттерн определяет семейство алгоритмов, инкапсулирует каждый из них и делает их взаимозаменяемыми?
А) Observer
Б) Command
В) Strategy
Г) State

9. Что из перечисленного является целью рефакторинга?
А) Увеличение производительности выполнения программы
Б) Улучшение структуры кода без изменения его внешнего поведения
В) Добавление новой функциональности
Г) Исправление ошибок

10. Какое условие является обязательным для безопасного проведения рефакторинга?
А) Наличие автоматизированных тестов
Б) Отсутствие документации
В) Использование строгой типизации
Г) Применение паттернов

11. Какой прием рефакторинга предполагает разделение метода, который одновременно возвращает значение и изменяет состояние, на два отдельных метода?
А) Separate Query from Modifier
Б) Rename Method
В) Remove Parameter
Г) Parameterize Method

12. Что означает «Pull Up Method» в контексте обобщения?
А) Перемещение метода из подкласса в суперкласс
Б) Перемещение метода из суперкласса в подкласс
В) Удаление метода из иерархии
Г) Создание абстрактного метода

13. Какая из перечисленных практик НЕ относится к экстремальному программированию (XP)?
А) Парное программирование
Б) Непрерывная интеграция
В) Детальное предварительное проектирование всех компонентов
Г) Разработка через тестирование (TDD)

14. Что такое обратный инжиниринг (reverse engineering) программного обеспечения?
А) Процесс создания нового программного продукта на основе готовых библиотек
Б) Процесс анализа существующей программы с целью восстановления ее структуры, логики или документации
В) Процесс улучшения производительности программы
Г) Процесс перевода программы на другой язык программирования

15. Какая из целей чаще всего преследуется при обратном инжиниринге?
А) Повышение скорости работы системы
Б) Восстановление утерянной документации
В) Добавление новых графических элементов
Г) Оптимизация использования памяти

16. Какой метод обратного инжиниринга основан на исследовании выполняемой программы без запуска?
А) Динамический анализ
Б) Статический анализ
В) Трассировка
Г) Профилирование

17. Какой инструмент обычно используется для статического анализа бинарного кода?
А) Отладчик (GDB)
Б) Дизассемблер (IDA Pro)
В) Профилировщик
Г) Компилятор

18. Что такое реинжиниринг программного обеспечения?
А) Внесение небольших изменений в код для исправления ошибок
Б) Полное переосмысление и перестройка существующей системы с целью улучшения ее характеристик
В) Создание новой системы с нуля по старым спецификациям
Г) Анализ кода конкурентов

19. Чем реинжиниринг отличается от обратного инжиниринга?
А) Реинжиниринг включает этап обратного инжиниринга, затем анализ и создание новой версии
Б) Реинжиниринг – это только анализ, без изменений
В) Обратный инжиниринг – это всегда незаконная деятельность
Г) Реинжиниринг выполняется только для аппаратного обеспечения

20. Какой паттерн позволяет подключать дополнительные обязанности к объекту динамически, без изменения его класса?
А) Adapter
Б) Decorator
В) Proxy
Г) Bridge

21. Какой поведенческий паттерн инкапсулирует запрос как объект, что позволяет параметризовать клиентов с разными запросами и поддерживать отмену операций?
А) Observer
Б) Command

В) Mediator
Г) Chain of Responsibility

22. Что из перечисленного является примером упрощения вызовов методов (рефакторинг)?
А) Замена условного оператора на полиморфизм
Б) Замена метода с флагом на несколько отдельных методов (Replace Parameter with Explicit Methods)
В) Извлечение метода из длинной процедуры
Г) Перенос поля из класса в его подкласс

23. Какая цель рефакторинга является основной?
А) Увеличение скорости работы программы
Б) Уменьшение количества строк кода
В) Повышение сопровождаемости и читаемости кода
Г) Добавление новых возможностей

24. В экстремальном программировании роль «заказчик» (Customer) отвечает за:
А) Написание кода
Б) Определение приоритетов и требований, написание пользовательских историй
В) Управление инфраструктурой
Г) Тестирование производительности

25. Что такое «антипаттерн»?
А) Часто встречающееся плохое решение проблемы, приводящее к негативным последствиям
Б) Паттерн, который нельзя использовать в коммерческих проектах
В) Паттерн, предназначенный для защиты от взлома
Г) Паттерн, использующий только структурные элементы

Вариант 2

1. Какой принцип разработки интерфейсов гласит, что система должна всегда информировать пользователя о происходящих действиях?
А) Принцип обратной связи
Б) Принцип единообразия
В) Принцип соответствия реальности
Г) Принцип гибкости

2. Какое требование к интерфейсу связано с минимизацией количества ошибок, которые может совершить пользователь?
А) Эффективность
Б) Защита от ошибок
В) Легкость освоения
Г) Эстетичность

3. Анализ интерфейса с помощью формальных моделей (например, GOMS) относится к:
А) Экспертной оценке
Б) Юзабилити-тестированию
В) Когнитивному анализу
Г) A/B-тестированию

4. Какой паттерн относится к порождающим и делегирует создание объекта подклассам?
А) Factory Method
Б) Abstract Factory
В) Builder
Г) Prototype

5. Какой структурный паттерн позволяет объектам с несовместимыми интерфейсами работать вместе?
А) Adapter
Б) Bridge
В) Composite
Г) Flyweight

6. Какой поведенческий паттерн определяет зависимость типа «один-ко-многим» между объектами, так что при изменении одного объекта все зависимые оповещаются?
А) Observer
Б) Command
В) State
Г) Strategy

7. Для чего используется паттерн Proxy?
А) Для упрощения интерфейса сложной системы
Б) Для управления доступом к другому объекту, создавая его заместителя
В) Для разделения абстракции и реализации
Г) Для обхода механизма наследования

8. Какой порождающий паттерн использует клонирование существующего объекта вместо создания нового через конструктор?
А) Singleton
Б) Prototype
В) Builder
Г) Factory

9. Какое из утверждений о рефакторинге верно?
А) Рефакторинг обязательно изменяет внешнее поведение программы
Б) Рефакторинг улучшает внутреннюю структуру, не меняя внешнего поведения
В) Рефакторинг проводится только на этапе тестирования
Г) Рефакторинг запрещен в гибких методологиях

10. Что является причиной для проведения рефакторинга?
А) Наличие дублирующего кода
Б) Требование заказчика добавить новую функцию
В) Желание увеличить быстродействие
Г) Смена операционной системы

11. Какой прием рефакторинга относится к упрощению вызовов методов и заключается в замене одного метода с параметром-флагом на несколько специализированных методов?
А) Replace Parameter with Explicit Methods
Б) Separate Query from Modifier
В) Add Parameter
Г) Remove Parameter

12. Что означает «Push Down Method» в контексте обобщения?
А) Перемещение метода из суперкласса в подклассы
Б) Перемещение метода из подкласса в суперкласс
В) Создание нового метода в интерфейсе
Г) Удаление метода из иерархии

13. Какая из практик экстремального программирования предполагает частую интеграцию кода всех разработчиков (несколько раз в день)?
А) Парное программирование
Б) Коллективное владение кодом

В) Непрерывная интеграция
Г) Игровое планирование

14. Обратный инжиниринг может применяться для:
А) Оптимизации запросов к базе данных
Б) Восстановления исходного кода из исполняемого файла (декомпиляция)
В) Написания новой документации пользователя
Г) Тестирования интерфейса

15. Какая цель обратного инжиниринга является законной и распространенной?
А) Взлом защиты программного продукта
Б) Обеспечение взаимодействия с устаревшими системами, документация по которым утеряна
В) Получение исходного кода коммерческого продукта для его копирования
Г) Уменьшение размера приложения

16. Какой метод обратного инжиниринга включает выполнение программы и отслеживание ее поведения?
А) Статический анализ
Б) Динамический анализ
В) Синтаксический разбор
Г) Рефакторинг

17. Какое из перечисленных средств обычно используется для динамического анализа?
А) Компилятор
Б) Отладчик (например, GDB)
В) Дизассемблер
Г) Статический анализатор кода

18. Реинжиниринг, в отличие от рефакторинга, обычно:
А) Ограничивается небольшими изменениями в коде
Б) Включает значительные изменения архитектуры и, возможно, перепроектирование системы
В) Не затрагивает существующую структуру данных
Г) Проводится только на уровне интерфейса пользователя

19. Что из перечисленного является этапом реинжиниринга?
А) Только оптимизация кода
Б) Обратный инжиниринг, затем анализ и определение изменений, затем реструктуризация и внедрение
В) Только создание новых требований
Г) Только переименование переменных

20. Какой структурный паттерн объединяет объекты в древовидную структуру для представления иерархий «часть-целое»?
А) Composite
Б) Facade
В) Flyweight
Г) Proxy

21. Какой поведенческий паттерн позволяет объекту изменять свое поведение при изменении внутреннего состояния (как будто объект меняет свой класс)?
А) Strategy
Б) State
В) Template Method
Г) Visitor

22. Какой прием рефакторинга «Extract Method» направлен на:
А) Удаление дублирования и уменьшение размера методов

Б) Увеличение числа параметров метода
В) Объединение нескольких классов в один
Г) Переименование метода

23. Какое утверждение о рефакторинге и тестировании верно?
А) Тесты должны быть написаны после рефакторинга, чтобы проверить результат
Б) Наличие надежного набора тестов является обязательным условием безопасного рефакторинга
В) Рефакторинг не требует тестирования, если изменения минимальны
Г) Тесты необходимы только для проверки новой функциональности

24. В экстремальном программировании практика «парное программирование» предполагает:
А) Два разработчика работают за одним компьютером, один пишет код, другой его проверяет
Б) Один разработчик пишет код, другой пишет тесты отдельно
В) Два разработчика по очереди работают над разными задачами
Г) Один разработчик пишет код, другой – документацию

25. Что такое «метафора системы» в XP?
А) Общее видение архитектуры, которое помогает команде понять, как устроена система, через аналогию
Б) Строгое формальное описание всех функций
В) Техника поиска ошибок
Г) Метод планирования итераций

Ключ к тестам:

Вариант 1		**Вариант 2**	
№ вопроса	Правильный ответ	№ вопроса	Правильный ответ
1	А	1	А
2	Б	2	Б
3	Б	3	В
4	Б	4	А
5	В	5	А
6	А	6	А
7	Б	7	Б
8	В	8	Б
9	Б	9	Б
10	А	10	А
11	А	11	А
12	А	12	А
13	В	13	В
14	Б	14	Б
15	Б	15	Б
16	Б	16	Б
17	Б	17	Б
18	Б	18	Б
19	А	19	Б
20	Б	20	А
21	Б	21	Б
22	Б	22	А
23	В	23	Б
24	Б	24	А
25	А	25	А

Лабораторные занятия № 34-41:

1. Проектирование интерфейса пользователя
2. Разработка интерфейса пользователя
3. Проектирование с использованием паттернов
4. Рефакторинг приложения при наличии дублирования кода
5. Рефакторинг приложения при наличии большого класса и жадных функций
6. Рефакторинг приложения при наличии избыточных временных переменных
7. Рефакторинг приложения при наличии несгруппированных данных
8. Реинжиниринг приложения

5.2. Критерии оценивания

5.2.1. Критерии оценивания устного ответа

При оценке устного ответа, обучающегося учитывается:

1) полнота и правильность ответа;
2) степень осознанности, понимания изученного;

Отметка «5»: ответ правильный, полный в соответствии с изученным материалом; материал изложен в определенной логической последовательности, литературным языком.

Отметка «4»: ответ правильный, полный в соответствии с изученным материалом; материал изложен в определенной логической последовательности; возможны отдельные затруднения в формулировке выводов.

Отметка «3»: ответ, в котором в основном правильно, но схематично или с отклонениями от последовательности изложения раскрыт материал или неполный, несвязный ответ, изложенный нелогично

Отметка «2»: при ответе обнаружено непонимание обучающимся основного содержания учебного материала, неумение его анализировать допущены существенные ошибки, которые обучающийся не смог исправить при наводящих вопросах преподавателя, отсутствует логика в изложении материала, нет необходимых обобщений и самостоятельной оценки фактов; недостаточно сформированы навыки устной речи.

5.2.2. Критерии оценивания выполнения заданий на лабораторных и практических занятиях

Отметка «5»: работа выполнена полностью и правильно; сделаны правильные выводы.

Отметка «4»: работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя.

Отметка «3»: работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки.

Отметка «2»: допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя.

Задания к лабораторным занятиям представлены в методических указаниях к лабораторным занятиям по МДК.02.01. Разработка программных модулей

Методы оценки:

– Защита отчетов по выполненному заданию на лабораторных и практических занятиях.

– Интерпретация результатов наблюдений за деятельностью обучающегося на лабораторных и практических занятиях.

5.2.3. Критерии оценивания тестовых заданий

Оценка в баллах	Степень выполнения задания
Неуд.	Выполнено от 0 до 49,9 % предложенных заданий
Удов.	Выполнено от 50 до 69,9% предложенных заданий
Хор.	Выполнено от 70 до 89,9% предложенных заданий
Отл.	Выполнено от 90 до 100% предложенных заданий

5.2.4. Общая классификация ошибок

При оценке знаний и умений учитываются ошибки и недочеты в работе.

Грубыми считаются ошибки:

- незнание определения основных понятий, законов;
- неумение выделить в ответе главное;
- обобщить результаты изучения;
- неумение применить знания для решения задач;
- неумение использовать полученные данные для выводов;
- неумение пользоваться первоисточниками, учебником, справочником;
- нарушение техники безопасности, небрежное отношение к оборудованию

Негрубыми считаются ошибки:

- неточность формулировок, определений, понятий, законов, вызванная неполнотой охвата основных признаков определяемого понятия или заменой 1-3 из этих признаков второстепенными;
- ошибки, вызванные несоблюдением условий проведения лабораторных занятий;
- недостаточно продуманный план устного ответа (нарушение логики изложения, подмена отдельных основных вопросов второстепенными);
- нерациональные методы работы со справочной литературой;
- неумение выполнять лабораторные задания в общем виде.

6. ОЦЕНОЧНЫЕ СРЕДСТВА ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ОБУЧАЮЩИХСЯ ПО МЕЖДИСЦИПЛИНАРНОМУ КУРСУ МДК 02.01. Разработка программных модулей

Промежуточная аттестация по междисциплинарному курсу МДК 02.01. Разработка программного обеспечения проводится в форме экзамена в 5 и 6 семестрах.

6.1. Вопросы к экзамену по МДК

5-ый семестр

пределение жизненного цикла программного обеспечения. Основные фазы ЖЦ.
аскадная (водопадная) модель: характеристики, преимущества, недостатки. Области применения.
тративные и инкрементные модели: суть, отличия от каскадной. Примеры (спиральная модель, модель с промежуточными версиями).
пиральная модель: риск-ориентированный подход, этапы каждого витка.
ибкие методологии (Agile): ценности и принципы Agile-манифеста. Сравнение с традиционными моделями.
пределение компонентно-ориентированного программирования (КОП) и его цели.

Преимущества компонентного подхода
лассические этапы разработки ПО: анализ требований, проектирование, кодирование, тестирование, внедрение, сопровождение.
нализ требований: функциональные и нефункциональные требования, методы сбора и анализа.
рхитектурное и детальное проектирование: отличия, создаваемые артефакты.
ровни тестирования: модульное, интеграционное, системное, приемочное. Их место в ЖЦ.
пределение структурного программирования. Основные управляющие структуры: последовательность, ветвление, циклы. Преимущества структурного подхода перед неструктурированным (спагетти-код).
пособы документирования алгоритмов: блок-схемы, псевдокод, комментарии, формальные спецификации. Требования к качественной документации алгоритма.
сновные принципы ООП: инкапсуляция, наследование, полиморфизм, абстракция.
ласс и объект: определения, различие, понятие состояния объекта.
пределение статических (классовых) полей и методов. Отличие от экземплярных членов. Сценарии использования статических членов (счетчики экземпляров, фабричные методы, утилитные функции).
еханизм виртуальных методов и позднего связывания (динамический полиморфизм). Отличие переопределения (override) от сокрытия (hide).
бстрактные классы и интерфейсы: назначение, сравнение, поддержка полиморфизма. Статический и динамический полиморфизм: сравнительный анализ.
пределение метакласса. Отличие метакласса от обычного класса. Реализация метаклассов в языках программирования (Python, Smalltalk, Ruby).
оздание метакласса в Python. Практические задачи, решаемые метаклассами. Риски и сложности при использовании метаклассов.
пределение событийно-управляемого программирования. Отличие от процедурного подхода.
одель событийного цикла (event loop) и очередь событий. Асинхронная обработка и неблокирующий ввод-вывод в событийной модели.
пределение элемента управления (виджета). Примеры основных элементов. Свойства и события элементов управления.
пределение диалогового окна. Виды: модальные и немодальные. Стандартные и настраиваемые диалоговые окна.
пределение обработчика события (event handler). Способы связывания событий с обработчиками. Анонимные обработчики (лямбда-выражения): преимущества и недостатки.
азовые графические примитивы (линии, прямоугольники, эллипсы, текст) и операции (рисование, заливка). Организация отрисовки (например, переопределение OnPaint или использование канвы).
пределение оптимизации модуля. Цели оптимизации. Влияние оптимизации модулей на архитектуру системы.
ритерии оценки качества модуля: эффективность, читаемость, сопровождаемость, тестируемость, повторная используемость.
онятие информационной закрытости (information hiding). Связь с инкапсуляцией.
еханизмы поддержки информационной закрытости в языках программирования.
оль информационной закрытости в управлении сложностью и сопровождении.
оследствия нарушения информационной закрытости.
пределение связности (cohesion) модуля. Виды связности: функциональная, последовательная, коммуникационная, процедурная, временная, логическая, случайная.
пределение сцепления (coupling) между модулями. Типы сцепления:

содержательное, общее, внешнее, управляющее, штамповое, по данным.
пределение специальных библиотек. Отличие от стандартных библиотек и фреймворков. Примеры специальных библиотек для различных областей (научные вычисления, машинное обучение, графика, сеть).

6-ый семестр

1. Группы операторов SQL: DDL, DML, DCL, TCL. Их назначение.
2. Основные типы данных SQL. Отличия VARCHAR от CHAR.
3. Синтаксис SELECT с WHERE, ORDER BY, GROUP BY, HAVING. Агрегатные функции.
4. Подзапросы и их использование.
5. Отличие WHERE от HAVING.
6. Синтаксис CREATE TABLE. Ограничения целостности: PRIMARY KEY, FOREIGN KEY, UNIQUE, CHECK, NOT NULL.
7. Изменение структуры таблицы с помощью ALTER TABLE.
8. Операторы INSERT, UPDATE, DELETE. Отличие DELETE от TRUNCATE.
9. Транзакции: BEGIN, COMMIT, ROLLBACK. Важность для целостности данных.
10. Операторы JOIN (INNER, LEFT, RIGHT, FULL) и примеры их использования.
11. Назначение Python DB-API. Версии (DB-API 2.0).
12. Основные этапы работы с БД через DB-API: соединение, курсор, выполнение запроса, получение результатов.
13. Параметризованные запросы и защита от SQL-инъекций. Примеры плейсхолдеров.
14. Получение метаданных результата запроса.
15. Определение ORM. Проблема, решаемая этим подходом.
16. Преимущества ORM перед прямыми SQL-запросами. Недостатки (потеря производительности, сложные запросы).
17. Примеры ORM: SQLAlchemy, Django ORM. Определение моделей, связи, выполнение запросов.
18. Ленивая (lazy) и жадная (eager) загрузка в ORM. Сравнение, сценарии использования.
19. Миграции схемы базы данных в ORM. Инструменты автоматического обновления.
20. Случаи, когда использование ORM неоптимально и предпочтительнее «сырой» SQL.
21. Перечислите основные эвристики (правила) разработки удобных пользовательских интерфейсов. Какие требования предъявляются к качественному интерфейсу с точки зрения юзабилити. Приведите примеры.
22. Какие методы используются для анализа и оценки интерфейса. Как требования к интерфейсу вытекают из требований к программному продукту? Приведите пример влияния нефункциональных требований на дизайн интерфейса.
23. Дайте определение паттерна проектирования. Какова цель использования паттернов в разработке ПО? На какие три основные группы делятся паттерны по назначению? Кратко охарактеризуйте задачи, решаемые каждой группой.
24. Чем паттерн отличается от готовой библиотеки или фреймворка? Приведите примеры, когда применение паттерна оправдано.Что такое антипаттерн? Приведите примеры известных антипаттернов и объясните, чем они вредны.
25. Порождающие паттерны: перечислите основные (Singleton, Factory Method, Abstract Factory, Builder, Prototype). Для каждого кратко опишите решаемую задачу.
26. Структурные паттерны: назовите основные (Adapter, Bridge, Composite, Decorator, Facade, Proxy). Объясните, для чего используется паттерн Decorator и чем он отличается от наследования.
27. Поведенческие паттерны: перечислите основные (Strategy, Observer, Command, State, Template Method, Visitor, Mediator). Объясните суть паттерна Observer и приведите пример его применения.

28. Дайте определение рефакторинга. Чем он отличается от оптимизации и от исправления ошибок? Назовите основные причины для проведения рефакторинга.
29. Какие цели преследует рефакторинг с точки зрения внутренних метрик качества кода (связность, сцепление, размер методов). Когда рефакторинг не следует выполнять (например, на поздних стадиях перед релизом, в плохо протестированном коде)?
30. Какие приемы рефакторинга относятся к упрощению вызовов методов. В чем суть рефакторинга «Separate Query from Modifier»? Почему не рекомендуется совмещать метод, который возвращает значение, с методом, который изменяет состояние? Приведите пример.
31. Что такое экстремальное программирование (XP)? Каковы его основные ценности (коммуникация, простота, обратная связь, смелость, уважение)?
32. Дайте определение обратного инжиниринга применительно к программному обеспечению. Какие цели преследует обратный инжиниринг. Какие уровни абстракции могут быть восстановлены при обратном инжиниринге.
33. В каких областях чаще всего применяется обратный инжиниринг. Каковы правовые и этические ограничения при проведении обратного инжиниринга? Когда он допустим, а когда запрещен?
34. Перечислите основные методики (подходы) к обратному инжинирингу. Что такое «прямой инжиниринг» и «обратный инжиниринг» в контексте моделей? Как обратный инжиниринг помогает восстанавливать UML-диаграммы из исходного кода?
35. Дайте определение реинжиниринга программного обеспечения. Каковы основные цели реинжиниринга.

6.2.Критерии оценивания ответов на экзамене

– оценка **«отлично»**, если студент обладает глубокими и прочными знаниями программного материала; при ответе на вопросы продемонстрировал исчерпывающее, последовательное и логически стройное изложение; правильно сформулировал понятия и закономерности по вопросам; сделал вывод по излагаемому материалу;

– оценка **«хорошо»**, если студент обладает достаточно полным знанием программного материала; его ответ представляет грамотное изложение учебного материала; но имеются существенные неточности в формулировании понятий и закономерностей по вопросам; не полностью сделаны выводы по излагаемому материалу;

– оценка **«удовлетворительно»**, если студент имеет общие знания основного материала без усвоения некоторых существенных положений; формулирует основные понятия с некоторой неточностью; затрудняется в приведении примеров, подтверждающих теоретические положения;

– оценка **«неудовлетворительно»**, если студент не знает значительную часть программного материала; допустил существенные ошибки в процессе изложения; не умеет выделить главное и сделать вывод; приводит ошибочные определения; ни один вопрос не рассмотрен до конца, наводящие вопросы не помогают.

7. УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ

Основные печатные издания

1. Емелина, Е. И., Поддержка и тестирование программных модулей : учебник / Е. И. Емелина. - Москва: КноРус, 2025. — 267 с.

2. Зубкова Т.М. Технология разработки программного обеспечения. – Санкт-Петербург: Лань, 2021. – 252 с.
3. Шитов, В. Н., Проектирование и разработка интерфейсов пользователя: учебное пособие / В. Н. Шитов, К. Е. Успенский. — Москва: КноРус, 2023. — 294 с.

Основные электронные издания:

1. Гагарина, Л. Г. Технология разработки программного обеспечения: учебное пособие / Л.Г. Гагарина, Е.В. Кокорева, Б.Д. Сидорова-Виснадул ; под ред. Л.Г. Гагариной. — Москва: ФОРУМ: ИНФРА-М, 2022. — 400 с. — (Среднее профессиональное образование). - ISBN 978-5-8199-0812-9. - Текст: электронный. - URL: https://znanium.com/catalog/product/1794453
2. Зараменских, Е. П. Проектирование информационных систем: учебник и практикум для среднего профессионального образования / Е. П. Зараменских. — 2-е изд. — Москва: Издательство Юрайт, 2025. — 119 с. — (Профессиональное образование). — ISBN 978-5-534-21417-8. — Текст: электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/571330