

Федеральное государственное образовательное бюджетное учреждение  
высшего образования  
«Финансовый университет при Правительстве Российской Федерации»  
(Финансовый университет)

**Красноярский филиал Финуниверситета**

---

(наименование структурного подразделения)

УТВЕРЖДАЮ  
Заместитель директора по  
учебно-методической работе  
Красноярского филиала  
Финуниверситета  
Вергейчик О.С. Вергейчик  
« 04 » сентября 2025 г.

**ФОНД ОЦЕНОЧНЫХ СРЕДСТВ**  
по профессиональному модулю

ПМ.02 Осуществление интеграции программных модулей

---

(код, наименование)

09.02.07 Информационные системы и программирование

---

(код, наименование специальности)

Красноярск – 2025 г.

Фонд оценочных средств по профессиональному модулю разработан на основании федерального государственного образовательного стандарта среднего профессионального образования по специальности 09.02.07 «Информационные системы и программирование»

Составители:

Лац Елена Михайловна, преподаватель ВКК

---

(фамилия, имя, отчество, наименование должности, квалификационной категории)

Фонд оценочных средств по профессиональному модулю рассмотрен и рекомендован к утверждению на заседании предметной (цикловой) комиссии общепрофессиональных дисциплин

---

(наименование)

Протокол от «04» сентября 2025 г. № 1

Председатель предметной (цикловой)  
комиссии

  
\_\_\_\_\_  
(подпись)

О.А. Полтавец  
(инициалы, фамилия)

1. Паспорт фонда оценочных средств  
по профессиональному модулю ПМ.02 «Осуществление интеграции  
программных модулей»

(код, наименование)

09.0.20.7 Информационные технологии профессиональных дисциплин

(код, наименование специальности)

Результаты обучения (знания, умения)	Общие и профессиональные компетенции	Междисциплинарный курс	Наименование элементов профессионального модуля, раздела, темы	Наименование оценочного средства	
				Текущий контроль	Промежуточная аттестация
Знать: модели процесса разработки программного обеспечения; основные принципы процесса разработки программного обеспечения; основные подходы к интегрированию программных модулей; основы верификации и аттестации программного обеспечения. Уметь: использовать выбранную систему контроля версий; использовать методы для получения кода с	ОК 01, ОК 02, ОК 03, ОК 04, ОК 05, ОК 06, ОК 07, ОК 08, ОК 09. ПК 2.1, ПК 2.2, ПК 2.3, ПК 2.4, ПК 2.5	МДК 02.01 Технология разработки программного обеспечения	Тема 2.1.1 Основные понятия и стандартизация требований к программному обеспечению	Защита отчетов по практическим и лабораторным работам, фронтальный опрос, тестовые задания.	Дифференцированный зачет
			Тема 2.1.2. Описание и анализ требований. Диаграммы IDEF		
			Тема 2.1.3. Оценка качества программных средств		
		МДК 02.02 Инструментальные средства разработки программного обеспечения	Тема 2.2.1 Современные технологии и инструменты интеграции	Защита отчетов по практическим и лабораторным работам, фронтальный опрос, тестовые задания.	Экзамен
			Тема 2.2.2 Инструментарий тестирования и анализа качества программных средств		
		МДК 02.03 Математическое моделирование	Тема 2.3.1. Основы моделирования. Детерминированные задачи	Защита отчетов по практическим и лабораторным	Дифференцированный зачет

заданной функциональностью и степенью качества.			Тема 2.3.2 Задачи в условиях неопределенности	работам, фронтальный опрос, тестовые задания.	
---	--	--	--	---	--

## 2. Формы промежуточной аттестации по профессиональному модулю

Элементы профессионального модуля	Форма промежуточной аттестации	
	Промежуточная аттестация	Промежуточный контроль
МДК 02.01 Технология разработки программного обеспечения	Дифференцированный зачет	Оценка выполненных контрольных заданий внеаудиторной самостоятельной работы
МДК 02.02 Инструментальные средства разработки программного обеспечения	Экзамен	Оценка выполненных контрольных заданий внеаудиторной самостоятельной работы
МДК 02.03 Математическое моделирование	Дифференцированный зачет	Оценка выполненных контрольных заданий внеаудиторной самостоятельной работы
Учебная практика УП.02	Дифференцированный зачет	Оценка практических умений
Производственная практика ПП.02	Дифференцированный зачет	Оценка освоения профессиональных и динамики освоения общих компетенций, Аттестационный лист студента по ПП, характеристика, дневник профессиональной деятельности студента в период производственной практики
ПМ	Экзамен по модулю (квалификационный экзамен)	

### МДК 02.01 Технология разработки программного обеспечения

#### Тема 2.1.1 Основные понятия и стандартизация требований к программному обеспечению

Перечень вопросов по теме для устного опроса:

1. Дайте определение понятию "требование к программному обеспечению". На какие два основных класса их принято разделять и в чем их ключевое отличие?
2. Перечислите основные источники требований к ПО. Какой источник является наиболее критичным и почему?
3. Что такое нефункциональные требования? Приведите 3-4 конкретных примера для веб-приложения (например, интернет-магазина)?
4. Назовите ключевые стандарты, регламентирующие процесс сбора и документирования требований (например, в области жизненного цикла ПО),
5. Опишите типичные риски, связанные с некорректной постановкой или изменением требований на поздних стадиях проекта. Как процесс стандартизации помогает эти риски минимизировать?

Тест: Понятия и классификация требований

1. Какие требования описывают такие характеристики системы как производительность, надежность и безопасность?
  - a) Функциональные требования
  - b) Бизнес-требования
  - c) Пользовательские требования
  - d) Нефункциональные требования**
2. Какой стандарт ГОСТ регламентирует разработку технического задания на создание автоматизированных систем?
  - a) ГОСТ 19.xxx
  - b) ГОСТ 34.602-89**
  - c) ГОСТ Р ИСО 9001
  - d) ГОСТ Р 56939
3. К какому уровню требований относятся «пользовательские истории» (user stories) в методологии Agile?
  - a) Бизнес-требования
  - b) Пользовательские требования**
  - c) Системные требования
  - d) Технические требования
4. Какой принцип разработки ПО предполагает разделение программы на независимые модули с четко определенными интерфейсами?
  - a) Инкапсуляция
  - b) Модульность**
  - c) Полиморфизм
  - d) Абстракция
5. Какой подход к интеграции модулей предполагает одновременное объединение всех компонентов системы?
  - a) Нисходящая интеграция
  - b) Восходящая интеграция
  - c) Большой взрыв (Big Bang)**
  - d) Сэндвич-интеграция

Правильные ответы:

1d	2b	3b	4b	5c
----	----	----	----	----

### Тема 2.1.2. Описание и анализ требований. Диаграммы IDEF

Перечень вопросов по теме для устного опроса:

1. В чем заключается основная цель и сфера применения методологии IDEF0? Каковы ее основные элементы (блоки, стрелки) и правила их интерпретации?
2. Опишите алгоритм построения контекстной диаграммы (диаграммы верхнего уровня А-0) для бизнес-процесса «Обработка заказа клиента».
3. Какой тип диаграмм IDEF используется для моделирования потоков данных? В чем их принципиальное отличие от IDEF0?

4. Какие инструменты декомпозиции в IDEF0 позволяют проверить полноту и непротиворечивость моделируемого процесса? Что такое правило ICOM?

5. Назовите преимущества и ограничения использования нотации IDEF0 для описания функциональных требований к ПО по сравнению с UML Use Case Diagram.

Тест: UML и моделирование

1. Какая диаграмма UML показывает взаимодействие пользователей с системой и ее основные функции?
  - a) Диаграмма классов
  - b) Диаграмма вариантов использования (Use Case)**
  - c) Диаграмма последовательности
  - d) Диаграмма состояний
2. Какой элемент диаграммы вариантов использования обозначает внешнюю сущность, взаимодействующую с системой?
  - a) Прецедент
  - b) Актор (Actor)**
  - c) Ассоциация
  - d) Комментарий
3. Какая диаграмма показывает динамическое взаимодействие объектов во времени?
  - a) Диаграмма классов
  - b) Диаграмма компонентов
  - c) Диаграмма последовательности**
  - d) Диаграмма развертывания
4. Как называется документ, содержащий детальное описание всех требований к системе?
  - a) Техническое задание
  - b) Спецификация требований**
  - c) Проектная документация
  - d) Пользовательская инструкция
5. Какая диаграмма IDEF используется для моделирования бизнес-процессов?
  - a) IDEF0
  - b) IDEF3**
  - c) IDEF1X
  - d) IDEF5

Правильные ответы:

1b	2b	3c	4b	5b
----	----	----	----	----

## **Практическое занятие № 5**

### **«Построение диаграммы Вариантов использования»**

Цель работы: научиться строить диаграмму вариантов использования.

Теоретическая справка:

Визуальное моделирование в UML можно представить как некоторый процесс поуровневого спуска от наиболее общей и абстрактной концептуальной модели исходной системы к логической, а затем и к физической модели соответствующей программной системы. Для достижения этих целей вначале строится модель в форме так называемой диаграммы вариантов использования (use case diagram), которая описывает функциональное назначение системы или, другими словами, то, что система будет делать в процессе своего функционирования. Диаграмма вариантов использования является исходным концептуальным представлением или концептуальной моделью системы в процессе ее проектирования и разработки.

Разработка диаграммы вариантов использования преследует цели:

- определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы;
- сформулировать общие требования к функциональному поведению проектируемой системы.
- разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей;
- подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества сущностей или акторов, взаимодействующих с системой с помощью так называемых вариантов использования. При этом актором (actor) или действующим лицом называется любая сущность, взаимодействующая с системой извне. Это может быть человек, техническое устройство, программа или любая другая система, которая может служить источником воздействия на моделируемую систему так, как определит сам разработчик. В свою очередь, вариант использования (use case) служит для описания сервисов, которые система предоставляет актору. Другими словами, каждый вариант использования определяет некоторый набор действий, совершаемый системой при диалоге с актором. При этом ничего не говорится о том, каким образом будет реализовано взаимодействие акторов с системой.

Диаграмма вариантов использования состоит из акторов, для которых система производит действие и собственно действия Use Case, которое описывает то, что актор хочет получить от системы. Актор обозначается значком человечка, а Use Case - овалом. Дополнительно в диаграммы могут быть добавлены комментарии.

Между акторами и вариантами использования могут быть различные виды взаимодействия. Основные виды взаимодействия следующие:



– простая ассоциация - отражается линией между актором и вариантом использования (без стрелки). Отражает связь актора и варианта использования. На рисунке между актором администратор и вариантом использования – просматривать заказ.

– Направленная ассоциация - то же что и простая ассоциация, но показывает, что вариант использования инициализируется актором. Обозначается стрелкой.

– Наследование – показывает, что потомок наследует атрибуты и поведение своего прямого предка. Может применяться как для акторов, так для вариантов использования.

– расширение (extend) – показывает, что вариант использования расширяет базовую последовательность действий и вставляет собственную последовательность. При этом в отличие от типа отношений «включение» расширенная последовательность может осуществляться в зависимости от определенных условий.

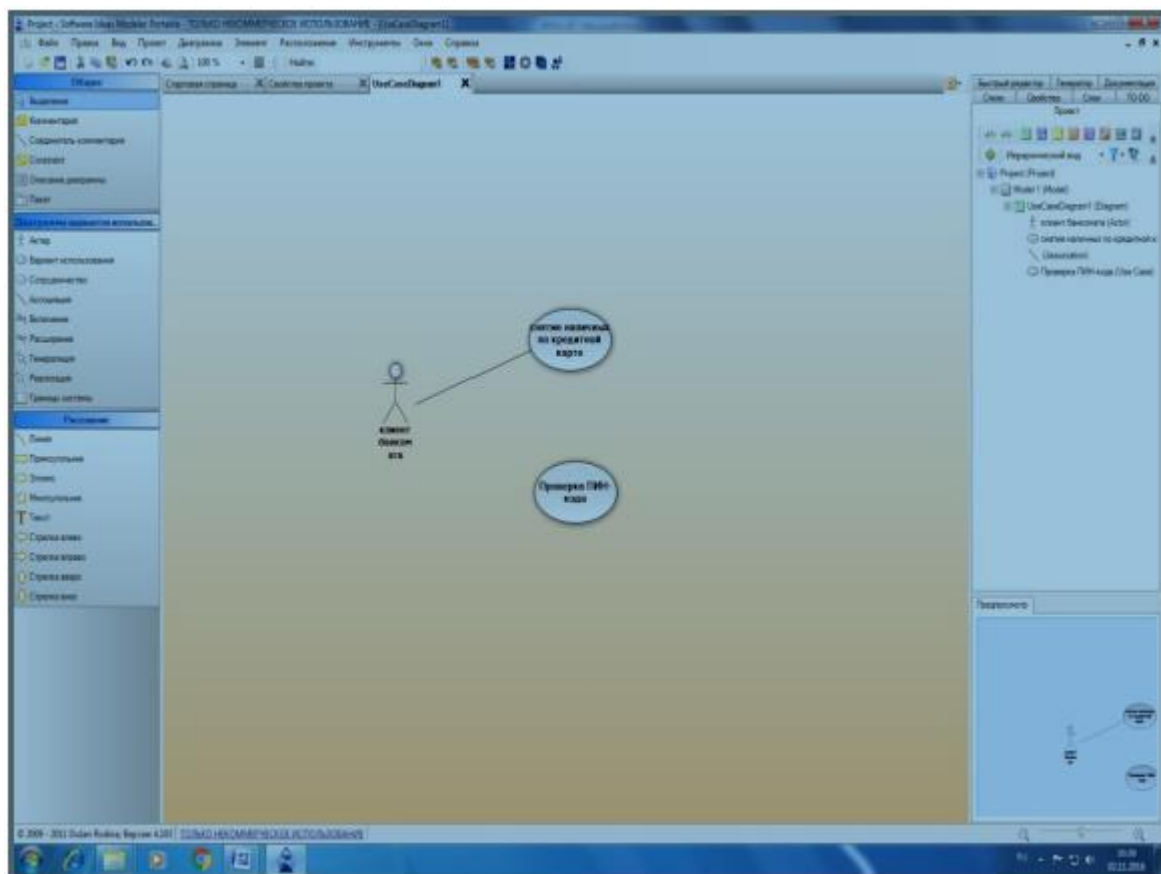
– включение (include) - показывает, что вариант использования включается в базовую последовательность и выполняется всегда (на рисунке не показан).

Существуют и другие виды взаимодействия, но они менее важны и реже применяются.

Задание 1. Построить диаграмму вариантов использования модели вариантов использования банкомата в Microsoft Visio.

Выполните следующие действия:

1. Добавить актора с именем Клиент банкомата.
2. Добавить вариант использования Снятие наличных по кредитной карте.
3. Добавить направленную ассоциацию от бизнес-актора Клиент Банкомата к варианту использования Снятие наличных по кредитной карте.
4. Добавить вариант использования Проверка ПИН-кода.
5. Добавить актора с именем Банк.
6. Добавить вариант использования Получение справки о состоянии счета.
7. Добавить вариант использования Блокирование кредитной карточки.
8. Добавить направленную ассоциацию от бизнес-актора Клиент Банкомата к варианту использования Получение справки о состоянии счета.
9. Добавить направленную ассоциацию от варианта использования. Снятие наличных по кредитной карточке к сервису Банк.
10. Добавить направленную ассоциацию от варианта использования. Получение справки о состоянии счета к сервису Банк.
11. Добавить отношение зависимости со стереотипом <<include>>, направленное от варианта использования Снятие наличных по кредитной карте к варианту использования Проверка Пин-кода.



12. Добавить отношение зависимости со стереотипом «include», направленное от варианта использования Получение справки о состоянии счета к варианту использования Проверка Пин-кода.

13. Добавить отношение зависимости со стереотипом «extend», направленное от варианта использования Блокирование кредитной карточки к варианту использования Проверка Пин-кода.

Задание 2. Построить диаграмму вариантов использования.

Имеются следующие данные:

- четыре действующих лица: Клиента банка, Банк, Кассира и Оператора;
- пять вариантов использования: Снять наличные, Перевести деньги со счета, Положить деньги на счет, Пополнить запас денег и Подтвердить пользователя;
- три зависимости, и отношения между действующими лицами и вариантами использования.

Варианты использования: Снять наличные, Перевести деньги со счета, Положить деньги на счет – требуют включения идентификации клиента в системе. Это поведение может быть выделено в новый вариант использования включения, называемый Подтвердить пользователя. Базовые варианты использования не зависят от метода, используемого для идентификации. Поэтому он инкапсулируется (скрывается) в варианте использования включения. С точки зрения базовых вариантов использования не имеет значение производится ли идентификация с помощью магнитной карты или

сканированием сетчатки глаза. Они только зависят от результата выполнения варианта использования Подтвердить клиента.

Задание для самостоятельной работы: Построить диаграмму вариантов использования на основе вербальной модели информационной системы «Компьютерный клуб»

Контрольные вопросы:

1. Какие цели преследует разработка диаграммы использования?
2. Для чего нужна диаграмма вариантов использования?
3. Из чего состоит диаграмма вариантов использования?
4. Виды взаимодействия используемые в диаграмме вариантов использования?
5. Из чего состоит созданная вами диаграмма?

### **Тема 2.1.3. Оценка качества программных средств**

Перечень вопросов по теме для устного опроса:

1. Дайте определение качества программного обеспечения согласно стандарту ISO/IEC 25010.
2. Что такое метрики качества ПО? Приведите примеры метрик для оценки надежности (reliability) и сопровождаемости (maintainability).
3. Опишите методику проведения экспертной оценки качества ПО. Какие артефакты проекта при этом анализируются?
4. Как статические анализаторы кода (например, SonarQube) помогают в оценке и поддержании внутреннего качества программного продукта?
5. Какая связь существует между качеством требований (описания, анализа) и итоговым качеством разработанного программного продукта?

Тест: Тестирование и качество ПО

1. Какой вид тестирования проверяет корректность работы отдельных модулей или функций?
  - a) Интеграционное тестирование
  - b) Модульное тестирование (Unit testing)**
  - c) Системное тестирование
  - d) Приемочное тестирование
2. Какой стандарт ISO определяет модель качества программного обеспечения?
  - a) ISO 9001
  - b) ISO/IEC 25010**
  - c) ISO 27001
  - d) ISO 14001
3. Что такое "тестовое покрытие" (test coverage)?
  - a) Количество тестов в наборе
  - b) Время выполнения тестов
  - c) Степень покрытия кода тестами**
  - d) Число найденных ошибок
4. Какой процесс проверяет, соответствует ли программное обеспечение

своим требованиям?

a) Валидация

**b) Верификация**

c) Сертификация

d) Аттестация

5. Какая метрика измеряет сложность программы на основе количества независимых путей выполнения?

a) Число строк кода (LOC)

**b) Цикломатическая сложность**

c) Индекс поддерживаемости

d) Глубина наследования

Правильные ответы:

1b	2b	3c	4b	5b
----	----	----	----	----

Перечень примерных вопросов для подготовки к дифференцированному зачету по МДК.02.01 Технология разработки программного обеспечения.

1. Дайте определение понятию «требования к программному обеспечению». Назовите и охарактеризуйте уровни требований (бизнес-, пользовательские, функциональные).

2. Отличие функциональных и нефункциональных требований.

3. Методологии и стандарты, регламентирующие работу с требованиями.

4. Перечислите основные классификации требований к ПО.

5. Стандарты, регламентирующие работу с требованиями (ISO/IEC, IEEE).

6. Опишите методологию SRS (Software Requirements Specification).

7. Риски, связанные с некорректным формированием требований.

8. Методы организации работы в команде разработчиков.

9. Опишите процесс code review и его значение.

10. Git и его основные преимущества.

11. Опишите принципы парного программирования.

12. Опишите системы контроля версий.

13. Основные подходы к интегрированию программных модулей.

14. Дайте определение интеграции программных модулей.

15. Опишите подход «снизу-вверх» (Bottom-Up) к интеграции.

16. Преимущества и недостатки подхода «сверху-вниз» (Top-Down).

17. «Большой взрыв» (Big Bang) в интеграции модулей.

18. Какие паттерны интеграции вы знаете.

19. API и его виды (REST, SOAP).

20. Опишите принципы семантического версионирования (SemVer).

21. Контейнеризация, как помощь в интеграции.

22. Coding standards – зачем они нужны.

23. Перечислите основные элементы стандартов кодирования.

24. Линтеры и как они используются.

25. Опишите принципы написания самодокументируемого кода.

26. Code style guide. Приведите примеры.

27. Инструменты автоматической проверки кода.
28. Типы диаграмм UML для моделирования поведения системы.
29. Опишите элементы диаграммы вариантов использования (Use Case).
30. Опишите разницу между диаграммой последовательности и кооперации.
31. Построение диаграммы классов и какие отношения она отображает.
32. Для чего используется диаграмма развертывания (Deployment Diagram).
33. Опишите процесс построения диаграммы деятельности (Activity Diagram).
34. Какие элементы включает диаграмма состояний (State Machine Diagram).
35. Связь диаграммы компонентов с физической структурой системы.
36. Описание и оформление требований (спецификация).
37. Анализ требований и стратегии выбора решения.
38. Цели и задачи и виды тестирования.
39. Классификация видов тестирования ПО.
40. Тестовое покрытие (test coverage) и как оно измеряется.
41. Опишите структуру тестового сценария.
42. Тестовый пакет и как он формируется.
43. Меры и метрики качества ПО.
44. Стандарты качества программной документации.
45. Разница между верификацией и валидацией ПО.
46. Дайте классификацию видов тестирования ПО.
47. Тестовое покрытие (test coverage) и как оно измеряется.
48. Опишите структуру тестового сценария.
49. Тестовый пакет и как он формируется.
50. Верификация и аттестация программного обеспечения.

## **МДК 02.02 Инструментальные средства разработки программного обеспечения**

### **Тема 2.2.1 Современные технологии и инструменты интеграции**

Перечень вопросов по теме для устного опроса:

1. Объясните разницу между горизонтальной и вертикальной интеграцией информационных систем.
2. Что такое шаблон проектирования API Gateway? Какие проблемы при интеграции микросервисов он решает?
3. Сравните подходы к интеграции на основе SOAP (Web Services) и REST API. В каких сценариях предпочтительнее каждый из них?
4. Опишите назначение и базовые принципы работы шины предприятия (Enterprise Service Bus, ESB). Какие альтернативы ESB существуют в современных облачных архитектурах?
5. Что такое "конвейер непрерывной интеграции и доставки" (CI/CD pipeline)? Какие инструменты (назовите 2-3) в нем используются для этапа сборки и интеграции компонентов?

Тест: Современные технологии и инструменты интеграции

1. Какой файл в системе контроля версий Git используется для указания файлов, которые не должны отслеживаться?
  - a. .gitconfig
  - b. .gitignore**
  - c) .gitattributes
  - a. d) README.md
2. Какой подход к интеграции модулей предполагает одновременное соединение всех компонентов после их индивидуального тестирования?
  - a) Нисходящая интеграция (Top-Down)
  - b) Восходящая интеграция (Bottom-Up)
  - c) «Большой взрыв» (Big Bang)**
  - d) Поэтапная интеграция (Incremental)
3. Какой уровень интеграции проверяет взаимодействие между полностью разработанными системами?
  - a) Модульная интеграция
  - b) Компонентная интеграция
  - c) Системная интеграция
  - d) Межсистемная интеграция**
4. Какой транспортный протокол чаще всего используется для REST API в веб-приложениях?
  - a) FTP
  - b) SMTP
  - c) HTTP/HTTPS**
  - d) SSH
5. При разработке модульной структуры проекта какие диаграммы UML наиболее полезны для визуализации зависимостей между модулями?
  - a) Диаграмма вариантов использования
  - b) Диаграмма последовательности
  - c) Диаграмма компонентов и диаграмма пакетов**
  - d) Диаграмма развертывания

Правильные ответы:

1b	2c	3d	4c	5c
----	----	----	----	----

Тема 2.2.2 Инструментарий тестирования и анализа качества программных средств

Перечень вопросов по теме для устного опроса:

1. Назовите основные категории инструментов для автоматизации тестирования (по уровням тестирования и видам). Для каждого приведите по одному примеру популярного инструмента.

2. В чем заключается принцип работы инструментов модульного тестирования (на примере JUnit для Java или pytest для Python)? Что такое мок-объекты и для чего они используются?

3. Какой класс инструментов используется для проверки уязвимостей и соответствия стандартам безопасности в коде и зависимостях? Приведите пример такого инструмента.

4. Опишите процесс и цель проведения нагрузочного тестирования. Какие инструменты (назовите 1-2) для этого используются и какие метрики они собирают?

5. Как инструменты отслеживания покрытия кода тестами (Code Coverage, например, JaCoCo) помогают оценивать эффективность набора тестов? Всегда ли 100% покрытие является показателем хорошего тестирования?

Тест: Отладка и инструменты разработки

1. Что такое отладка программного продукта?
  - a) Написание пользовательской документации
  - b) Создание тестовых сценариев
  - c) Процесс поиска, анализа и устранения ошибок в программе**
  - d) Оптимизация производительности кода
2. Какой инструмент позволяет устанавливать точки останова (breakpoints) и пошагово выполнять программу?
  - a) Компилятор
  - b) Линтер
  - c) Отладчик (Debugger)**
  - d) Профайлер
3. Для чего используются отладочные классы в объектно-ориентированном программировании?
  - a) Для финальной сборки релизной версии
  - b) Для логирования, диагностики и тестирования во время разработки**
  - c) Для шифрования конфиденциальных данных
  - d) Для кеширования результатов вычислений
4. Какое тестирование выполняется разработчиком для проверки отдельных функций или методов?
  - a) Интеграционное тестирование
  - b) Модульное тестирование (Unit testing)**
  - c) Системное тестирование
  - d) Приемочное тестирование
5. Что проверяет интеграционное тестирование?
  - a) Отдельные функции программы
  - b) Пользовательский интерфейс
  - c) Взаимодействие между модулями или компонентами системы**
  - d) Соответствие требованиям заказчика
6. Какой вид тестирования предполагает выполнение тестов человеком без использования автоматизированных скриптов?
  - a) Автоматизированное тестирование
  - b) Ручное тестирование**
  - c) Нагрузочное тестирование
  - d) Регрессионное тестирование
7. Что такое статический анализ кода?
  - a) Тестирование выполнения программы

- b) Анализ кода без его выполнения, поиск потенциальных ошибок и нарушений стандартов
  - c) Измерение времени выполнения функций
  - d) Проверка пользовательского интерфейса
8. Какой инструмент используется для автоматической проверки стиля кода и выявления потенциальных ошибок в Python?
- a) PyInstaller
  - b) PyCharm (как IDE)
  - c) **Pylint или Flake8**
  - d) PyTest
9. Что такое инспекция кода (code inspection)?
- a) Автоматическая компиляция проекта
  - b) **Формальный процесс проверки кода другими разработчиками**
  - c) Написание комментариев к коду
  - d) Создание технической документации
10. Какой блок в Python используется для перехвата и обработки исключений?
- a) if-elif-else
  - b) **try-except**
  - c) for-while
  - d) def-return
11. Что такое "stack trace" (трассировка стека)?
- a) График производительности приложения
  - b) **Информация о последовательности вызовов функций в момент возникновения ошибки**
  - c) Список всех глобальных переменных программы
  - d) Журнал всех операций ввода-вывода
12. Какой метод идентификации ошибок предполагает запись информации о выполнении программы в файл?
- a) Отладка с помощью точек останова
  - b) **Логирование (Logging)**
  - c) Профилирование производительности
  - d) Модульное тестирование
13. Что такое тестовый сценарий (test case)?
- a) Техническое задание на разработку
  - b) **Набор условий и шагов для проверки определенной функциональности с ожидаемыми результатами**
  - c) Документация по установке программного обеспечения
  - d) Отчет о найденных ошибках
14. Какой подход к документированию результатов тестирования является наиболее эффективным?
- a) Устное обсуждение результатов
  - b) **Структурированные отчеты с информацией об окружении, шагах воспроизведения и фактических результатах**



- с) Запись результатов в личный блокнот тестировщика
- д) Отправка email с кратким описанием проблем

Правильные ответы:

1c	2c	3b	4b	5c	6b	7b
8c	9b	10b	11b	12b	13b	14b

Перечень примерных вопросов для подготовки к экзамену по МДК.02.02 Инструментальные средства разработки программного обеспечения.

1. Дайте определение понятию «репозиторий проекта».
2. Сериализация и десериализация данных.
3. Опишите типовую структуру программного проекта.
4. Опишите роли участников команды в системе контроля версий.
5. Виды репозитория (локальные, удаленные).
6. Code review и его преимущества.
7. Файл .gitignore для чего он используется.
8. Организация процесса code review в команде.
9. Опишите жизненный цикл проекта в системе контроля версий.
10. Инструмент pull request/merge request для командной разработки.
11. Настройка прав доступа в репозитории для разных участников.
12. Опишите практики ведения коммитов (commit conventions).
13. Основные метаданные, хранящиеся в репозитории.
14. Основные задачи отладочных классов.
15. Ветвление (branching) в репозитории.
16. Ручное и автоматизированное тестирование.
17. Опишите стратегии ветвления (GitFlow, GitHub Flow).
18. CI/CD пайплайн в контексте командной работы.
19. Конфликт слияния и методы его разрешения в репозитории.
20. Виды, цели и уровни интеграции программных модулей.
21. Инструменты, используемые для визуализации истории репозитория.
22. Отладка программных продуктов.
23. Определение понятия «репозиторий проекта».
24. Опишите процесс рефакторинга в командной разработке.
25. Определение интеграции программных модулей.
26. Инструменты, используемые для координации командной работы.
27. Назовите и охарактеризуйте уровни интеграции.
28. Определение отладки программного продукта.
29. Методы и средства организации тестирования.
30. Преимущества автоматизированного тестирования.
31. Опишите подход "снизу-вверх" (Bottom-Up) к интеграции.
32. Опишите уровни тестирования.
33. Преимущества и недостатки подхода "сверху-вниз" (Top-Down).
34. Модульное тестирование (unit testing).
35. Непрерывная интеграция (Continuous Integration).
36. Опишите принципы интеграционного тестирования.

- 37.Инструментарий анализа качества кода.
- 38.Инструменты, использующиеся для автоматизации сборки.
- 39.Опишите метрики кода (code metrics).
- 40.Артефакты сборки и как они управляются.
- 41.Дайте определение исключительной ситуации.
- 42.Опишите процесс деплоя интегрированного приложения.
- 43.Методы обработки исключений.
- 44.Метрики, использующиеся для оценки качества интеграции.
- 45.Опишите принципы построения отказоустойчивых систем.
- 46.Логирование и его помощь в отладке.
- 47.Процесс сопоставления объектов данных (data mapping).
- 48.Идентификация сбоя и ошибок в системе.
- 49.Опишите процесс трансформации данных при интеграции.
- 50.Методы, использующиеся для выявления ошибок системных компонентов.
- 51.Метрики, использующиеся для оценки качества интеграции.
- 52.Опишите принципы построения системы оповещений об ошибках.
- 53.Организация сбора и анализа логов в распределенной системе.
- 54.Уровни тестирования.
- 55.Трассировка запросов (request tracing) в микросервисной архитектуре.

#### Практические вопросы

1. Задача: Вам поступила задача создать каркас микросервиса «Уведомления» на Python/Java. Создайте в пустой папке структуру проекта согласно современным стандартам, инициализируйте Git-репозиторий, создайте .gitignore, README.md с описанием проекта и docker-compose.yml для поднятия зависимостей (БД, брокера сообщений — указать в виде комментария).
2. Задача: Дано описание: «Сервис А вызывает API сервиса Б, получает данные, обрабатывает их и кладёт результат в очередь RabbitMQ». Нарисуйте диаграмму последовательности (текстом в PlantUML или схемой в draw.io). Дополнительно составьте таблицу формата сообщения для очереди (поля, типы, пример).
3. Задача: Для трёх сценариев обоснуйте выбор протокола: 1) чат в реальном времени между клиентом и сервером, 2) ежечасная выгрузка большого объёма статистики из одной БД в другую, 3) взаимодействие двух микросервисов внутри защищённого контура с максимальной производительностью. Ответ представьте в виде таблицы (Сценарий – Рекомендуемый протокол – Краткое обоснование).
4. Задача: Для сервиса «Расчёт доставки» спроектируйте модульную структуру (пакеты/слои). Выделите минимум 4 модуля (например, api, core, repository, client). Опишите ответственность каждого и направление зависимостей между ними (какой модуль от какого зависит). Изобразите схему.
5. Задача: Опишите, как настроить Git pre-commit hook, который будет: 1) запрещать коммиты в ветку master напрямую, 2) запускать форматирование

кода (например, `black` для Python) перед коммитом. Напишите скрипт (`bash` или псевдокод) для второго пункта.

6. Задача: Используя `HttpClient` (Java) или `requests` (Python), напишите метод `getWeather(city)`, который вызывает публичный API (например, `open-meteo.com`), обрабатывает статусы ответа (200, 404, 500) и парсит JSON, возвращая температуру. Обработайте таймаут соединения.

7. Задача: Опишите, как реализовать паттерн `Circuit Breaker` для метода, вызывающего ненадёжный внешний сервис. Нарисуйте диаграмму состояний (`Closed`, `Open`, `Half-Open`). Напишите псевдокод проверки состояния перед вызовом.

8. Задача: Опишите пошаговый процесс: как взять модуль вашего проекта, сделать из него `standalone JAR`-библиотеку (для Java) или пакет `PyPI` (для Python), опубликовать во внутренний репозиторий (например, `Nexus`) и подключить в другом проекте.

9. Задача: Напишите тест (`JUnit/jest/pytest`), который проверяет, что метод `validateAge(age)` выбрасывает исключение `InvalidAgeException`, если возраст меньше 18. Проверьте как текст сообщения исключения, так и его тип.

10. Задача: В процессе тестирования вы обнаружили баг: «При отправке формы с пустым полем `email` сервер возвращает 500 вместо 400». Заполните шаблон баг-репорта: Шаги воспроизведения, Ожидаемый результат, Фактический результат, Окружение (браузер, ОС, версия API), Приоритет, Приложите логи.

## **МДК 02.03 Математическое моделирование**

### **Тема 2.3.1. Основы моделирования. Детерминированные задачи**

Перечень вопросов по теме для устного опроса:

1. Что такое математическая модель? Опишите основные этапы процесса математического моделирования применительно к задаче оптимизации (например, распределения ресурсов).

2. Какие задачи называются детерминированными? Приведите пример детерминированной задачи из области разработки ПО (например, планирование длительности проекта при известной производительности).

3. Объясните, как с помощью линейного программирования можно формализовать задачу оптимального распределения ограниченного бюджета между несколькими проектами для максимизации прибыли. Опишите ключевые элементы модели (целевая функция, ограничения).

4. В чем заключается графический метод решения задач линейного программирования для двух переменных? Проиллюстрируйте его шаги.

5. Что такое симплекс-метод и для решения каких типов оптимизационных задач он применяется? Назовите его ключевое преимущество перед графическим методом.

Тест: Основы моделирования. Детерминированные задачи

1. Что такое оптимальное решение в математическом моделировании?

а) Любое допустимое решение

- b) Решение, обеспечивающее экстремум (минимум или максимум) целевой функции**
- c) Первое найденное решение
- d) Решение, удовлетворяющее только части ограничений
2. Что такое математическая модель?
- a) Физический макет системы
- b) Упрощенное математическое описание реального объекта или процесса**
- c) Графическое изображение данных
- d) Таблица экспериментальных результатов
3. Для решения какой задачи используется симплекс-метод?
- a) Транспортной задачи
- b) Задачи о кратчайшем пути
- c) Задачи линейного программирования**
- d) Задачи динамического программирования
4. Что такое транспортная задача?
- a) Задача о распределении ресурсов между потребителями
- b) Частный случай задачи линейного программирования о минимизации стоимости перевозок**
- c) Задача о поиске оптимального маршрута
- d) Задача о замене оборудования
5. Какой метод используется для нахождения начального решения транспортной задачи?
- a) Симплекс-метод
- b) Метод северо-западного угла или метод минимального элемента**
- c) Метод множителей Лагранжа
- d) Метод динамического программирования
6. Для проверки оптимальности решения транспортной задачи используется:
- a) Метод Гаусса
- b) Метод потенциалов**
- c) Метод Ньютона
- d) Метод золотого сечения
7. Какой метод используется для решения условных экстремальных задач?
- a) Симплекс-метод
- b) Метод множителей Лагранжа**
- c) Метод потенциалов
- d) Метод ветвей и границ
8. Что такое принцип оптимальности Беллмана в динамическом программировании?
- a) Оптимальное управление на каждом шаге выбирается независимо
- b) Оптимальное управление на последующих шагах не зависит от предыстории процесса**

- c) Все шаги процесса рассматриваются одновременно
- d) Управление выбирается случайным образом
- 9. Какие критерии оптимальности используются в динамическом программировании?
  - a) Только линейные
  - b) Аддитивные и мультипликативные**
  - c) Только квадратичные
  - d) Произвольные нелинейные
- 10. Какой алгоритм используется для нахождения кратчайших путей в графе с неотрицательными весами?
  - a) Алгоритм Форда-Фалкерсона
  - b) Алгоритм Дейкстры**
  - c) Алгоритм Краскала
  - d) Алгоритм Прима
- 11. В чем заключается задача о максимальном потоке?
  - a) Найти путь минимальной длины
  - b) Найти максимальный поток, который можно пропустить через сеть от источника к стоку**
  - c) Найти минимальное остовное дерево
  - d) Найти эйлеров цикл
- 12. Какой алгоритм решает задачу о максимальном потоке?
  - a) Алгоритм Дейкстры
  - b) Алгоритм Форда-Фалкерсона**
  - c) Алгоритм Флойда-Уоршелла
  - d) Алгоритм Хаффмана

Правильные ответы:

1b	2b	3c	4b	5b	6b
7b	8b	9b	10b	11b	12b

### Тема 2.3.2 Задачи в условиях неопределенности

Перечень вопросов по теме для устного опроса:

1. Чем принципиально отличаются задачи в условиях неопределенности от детерминированных? Какие виды неопределенности вы знаете?
2. Опишите постановку задачи принятия решений в условиях риска. Что такое "дерево решений" и как оно строится?
3. Что такое критерий максимума ожидаемого выигрыша (Байесовский критерий)? Приведите пример его применения для выбора стратегии развития ПО (например, внедрять рискованную новую функцию или нет).
4. Какие критерии принятия решений используются в условиях полной неопределенности (например, критерий Вальда, Сэвиджа, Гурвица)? В чем их основная идея и различия?
5. Как метод Монте-Карло применяется для моделирования и анализа задач в условиях неопределенности (например, для оценки сроков проекта)? Опишите общий алгоритм его применения.

Тест: Задачи в условиях неопределенности

1. Что такое система массового обслуживания (СМО)?
  - a) Система обработки больших объемов данных
  - b) Система, состоящая из потока заявок, очереди и обслуживающих приборов**
  - c) Система управления базами данных
  - d) Система оптимизации производственных процессов
2. Какой процесс называется марковским (процессом без последствия)?
  - a) Процесс с нормальным распределением
  - b) Процесс, у которого будущее состояние зависит только от текущего состояния, а не от прошлого**
  - c) Процесс с постоянной интенсивностью
  - d) Линейный стохастический процесс
3. Что описывают уравнения Колмогорова в теории марковских процессов?
  - a) Распределение вероятностей начального состояния
  - b) Динамику изменения вероятностей состояний системы во времени**
  - c) Стационарное распределение системы
  - d) Математическое ожидание времени пребывания в состоянии
4. Какой метод прогнозирования использует взвешенную сумму прошлых наблюдений с экспоненциально убывающими весами?
  - a) Метод скользящих средних
  - b) Метод экспоненциального сглаживания**
  - c) Метод наименьших квадратов
  - d) Метод проектирования тренда
5. Что такое "скользящее среднее" в методах прогнозирования?
  - a) Среднее значение за весь период наблюдений
  - b) Среднее значение за последние n периодов, которое "скользит" вместе с временным рядом**
  - c) Средневзвешенное значение с произвольными весами
  - d) Экспоненциальная функция от времени
6. Какой метод прогнозирования основан на продолжении выявленной тенденции?
  - a) Метод скользящих средних
  - b) Метод экспертных оценок
  - c) Метод проектирования тренда**
  - d) Метод Дельфи
7. В каких условиях принимаются решения, когда известны вероятности исходов?
  - a) В условиях определенности
  - b) В условиях риска**
  - c) В условиях неопределенности
  - d) В условиях конфликта

8. Какой критерий принятия решений рекомендует выбирать альтернативу с минимальным максимальным проигрышем?
  - a) Критерий Байеса
  - b) Критерий Лапласа
  - c) Критерий Сэвиджа (критерий минимаксного сожаления)**
  - d) Критерий Гурвица
9. Что такое "дерево решений"?
  - a) Структура данных для хранения информации
  - b) Графическое представление процесса принятия решений с учетом различных исходов и их вероятностей**
  - c) Метод классификации данных
  - d) Алгоритм поиска оптимального пути
10. Что такое имитационное моделирование?
  - a) Аналитическое решение уравнений
  - b) Воспроизведение функционирования системы с помощью компьютерной модели**
  - c) Графическое представление данных
  - d) Статистическая обработка результатов
11. Какой метод используется для генерации случайных чисел в имитационном моделировании?
  - a) Метод аналитического решения
  - b) Метод Монте-Карло**
  - c) Метод наименьших квадратов
  - d) Метод динамического программирования
12. Что такое "единичный жребий" в имитационном моделировании?
  - a) Однократное проведение эксперимента
  - b) Один цикл имитации работы системы с использованием случайных чисел**
  - c) Метод определения начальных условий
  - d) Способ задания параметров модели

Правильные ответы:

1b	2b	3b	4b	5b	6c
7b	8c	9b	10b	11b	12b

Перечень примерных вопросов для подготовки к дифференцированному зачету по МДК.02.03 Математическое моделирование.

1. Понятие модели, свойства модели, классификация моделей.
2. Классификация математических моделей по основанию в зависимости от: сложности объекта моделирования. Примеры сложных и простых объектом моделирования.
3. Математическое формулирование задачи моделирования. Примеры. Задача Коши.
4. Понятие решения. Множество решений, оптимальное решение.
5. Математические модели, принципы их построения, виды моделей.
6. Задачи: классификация, методы решения, граничные условия.

7. Основные этапы математического моделирования.
8. Математическая модель транспортной задачи.
9. Математическая модель задачи о выпуске продукции.
10. Случайные процессы и их классификация.
11. Математическая модель задачи о назначениях.
12. Предмет, задача и основные понятия математического программирования.
13. Классификация задач математического программирования.
14. Задача линейного программирования и ее общая форма.
15. Приведение задачи линейного программирования к канонической форме.
16. Геометрическая интерпретация задачи линейного программирования.
17. Возможные множества решений задачи линейного программирования.
18. Общая характеристика симплекс – метода. Заполнение начальной симплекс – таблицы.
19. Общий вид и основная задача линейного программирования. Симплекс – метод.
20. Критерий оптимальности плана задачи линейного программирования.
21. Транспортная задача. Методы нахождения начального решения транспортной задачи. Метод потенциалов.
22. Балансировка транспортной задачи.
23. Метод северо-западного угла.
24. Общая характеристика метода потенциалов.
25. Проверка плана транспортной задачи на оптимальность.
26. Построение нового плана в методе потенциалов.
27. Общий вид задач нелинейного программирования. Графический метод решения задач нелинейного программирования.
28. Общий вид задач нелинейного программирования. Метод множителей Лагранжа.
29. Основные понятия динамического программирования: шаговое управление, управление операцией в целом, оптимальное управление, выигрыш на данном шаге, выигрыш за всю операцию, аддитивный критерий, мультипликативный критерий.
30. Простейшие задачи, решаемые методом динамического программирования.
31. Предмет, область применения и основные понятия теории графов.
32. Задача о нахождении кратчайших путей в графе и методы ее решения.
33. Сетевой график и его элементы.
34. Методика расчета параметров сетевого графика.
35. Постановка задачи о кратчайшем маршруте.
36. Метод решения задачи о кратчайшем маршруте.
37. Постановка задачи о максимальном потоке.
38. Разрез и его пропускная способность.
39. Теорема Форда – Фалкерсона.
40. Методология метода ветвей и границ.



41. Постановка задачи коммивояжера.
42. Алгоритм приведения матрицы расходов в задаче коммивояжера.
43. Алгоритм деления множества маршрутов на части.
44. Процесс Маркова и его свойства.
45. Задача о максимальном потоке и алгоритм Форда–Фалкерсона.
46. Системы массового обслуживания: понятия, примеры, модели.
47. Предмет и задачи теории игр. Основные понятия теории игр: игра, игроки, партия, выигрыш, проигрыш, ход, личные и случайные ходы, стратегические игры, стратегия, оптимальная стратегия.
48. Методы решения конечных игр: сведение игры  $m \times n$  к задаче линейного программирования, численный метод – метод итераций.
49. Область применимости теории принятия решений. Принятие решений в условиях определенности, в условиях риска, в условиях неопределенности.
50. Критерии принятия решений в условиях неопределенности. Дерево решений.

Теоретические вопросы к экзамену по модулю (квалификационному экзамену) ПМ.02 Осуществление интеграции профессиональных модулей

1. Дайте определение понятия «требования к программному обеспечению». Какие три уровня требований существуют.
2. Опишите отличие функциональных и нефункциональных требований. Приведите примеры каждого типа.
3. Назовите основные стандарты, регламентирующие работу с требованиями (ISO/IEC, IEEE). В чем их практическое значение.
4. Опишите методологию User Stories. Какой формат используется и в каких Agile-методологиях она применяется.
5. Какие риски возникают при некорректном формировании требований к ПО. Как их можно минимизировать.
6. В чем отличие методологий Agile и Waterfall. Какие преимущества и недостатки каждой из них.
7. Что такое CI/CD (Continuous Integration/Continuous Deployment). Какие преимущества дает его внедрение.
8. Опишите основные принципы DevOps культуры. Как она влияет на процесс разработки ПО.
9. Какие типы диаграмм UML используются для моделирования поведения системы. Для чего каждая из них применяется.
10. Опишите элементы диаграммы вариантов использования (Use Case Diagram). Что такое акторы и прецеденты.
11. В чем разница между диаграммой последовательности (Sequence Diagram) и диаграммой кооперации (Communication Diagram).
12. Как строится диаграмма классов (Class Diagram). Какие типы отношений между классами она отображает.

13. Для чего используется диаграмма развертывания (Deployment Diagram). Какие элементы она включает.
14. Опишите процесс построения диаграммы деятельности (Activity Diagram). Когда ее следует использовать.
15. Какие элементы включает диаграмма состояний (State Machine Diagram). В каких случаях она наиболее полезна.
16. Как диаграмма компонентов (Component Diagram) связана с физической структурой системы.
17. Контекстная диаграмма в методологии IDEF0. Как проводится декомпозиция процессов.
18. Дайте определение интеграции программных модулей. Какие основные цели преследует интеграция.
19. Опишите подход «снизу-вверх» (Bottom-Up) к интеграции модулей. Каковы его преимущества и недостатки.
20. Особенности подхода «сверху-вниз» (Top-Down) к интеграции. Когда его целесообразно применять.
21. «Большой взрыв» (Big Bang) в интеграции модулей. Какие риски связаны с этим подходом.
22. API и его основные виды (REST, SOAP, GraphQL). В чем их различия.
23. Контейнеризация и как она помогает в интеграции модулей. Какие инструменты используются (Docker, Kubernetes).
24. Что такое формат сообщений JSON и XML. В чем их преимущества и недостатки для интеграции.
25. Дайте классификацию видов тестирования ПО. Чем отличаются unit, integration, system и acceptance тестирование.
26. Тестовое покрытие (test coverage) и как оно измеряется. Какие метрики покрытия существуют.
27. Опишите структуру тестового сценария. Какие разделы он должен включать.
28. Тестовый пакет (test suite) и как он формируется. Какие принципы группировки тестов используются.
29. Метрики качества ПО. Как измеряется надежность, производительность, сопровождаемость.
30. Различие между верификацией и валидацией ПО. Какие методы используются для каждого процесса.
31. Опишите процесс отладки программного кода. Какие инструменты отладки знаете.
32. Что такое отладочные классы (debug classes). Как они помогают в поиске ошибок.
33. Какие методы обработки исключительных ситуаций (exception handling) вы знаете. В чем их различия.
34. Как организовать логирование (logging) в приложении. Какие уровни логирования существуют.
35. Дайте определение понятию «математическая модель». Какие принципы построения моделей существуют.

36. Оптимальное решение и показатель эффективности. Как они связаны в задачах оптимизации.
37. Классифицируйте математические модели по различным признакам. Чем отличаются детерминированные и стохастические модели.
38. Опишите общий вид задачи линейного программирования. Какие ограничения могут быть в задаче.
39. Суть симплекс-метода решения задач линейного программирования. Опишите основные этапы алгоритма.
40. Транспортная задача. Опишите ее специальную структуру и методы нахождения начального решения.
41. В чем суть метода потенциалов для решения транспортной задачи. Как проверяется оптимальность решения.
42. Опишите общий вид задачи нелинейного программирования. Методы решения НЛП.
43. Суть метода множителей Лагранжа. Дайте определение функции Лагранжа.
44. Основные понятия динамического программирования. Сформулируйте принцип оптимальности Беллмана.
45. Опишите основные элементы систем массового обслуживания (СМО). Какие характеристики СМО используются для анализа.
46. Марковский случайный процесс. Какие свойства он имеет.
47. Опишите схему гибели и размножения. В каких задачах она применяется.
48. Уравнения Колмогорова. Как они используются для анализа марковских процессов.
49. Дайте понятие финальных вероятностей состояний. Когда они существуют и как вычисляются.
50. Имитационное моделирование. Какие преимущества оно дает по сравнению с аналитическими методами.
51. Опишите основные понятия теории игр. Что такое чистые и смешанные стратегии.
52. Методы решения матричных игр. Как свести игру к задаче линейного программирования.
53. Дерево решений. Как оно используется для анализа и принятия решений в условиях риска.
54. Методы обеспечения качества кода. Как code review влияет на качество ПО.
55. Технический долг (technical debt). Как он влияет на процесс интеграции модулей.
56. Опишите принципы тестирования микросервисной архитектуры. Какие специфические тесты необходимы.
57. Методы нагрузочного тестирования. Как подготовить систему к пиковым нагрузкам.
58. Опишите принципы микросервисной архитектуры. Какие преимущества и недостатки она имеет по сравнению с монолитной.

Федеральное государственное образовательное бюджетное  
учреждение высшего образования  
**«Финансовый университет при Правительстве Российской Федерации»**  
**(Финансовый университет)**

Красноярский филиал Финуниверситета

**БИЛЕТ №1**

по ПМ.02 Осуществление интеграции программных модулей  
специальность 09.02.07 Информационные системы и программирование  
квалификация программист, для студентов 4 курса

1. Дайте определение понятия «требования к программному обеспечению». Какие три уровня требований существуют.

2. Различие между верификацией и валидацией ПО. Какие методы используются для каждого процесса.

3. Разработать основные модули информационной системы для ООО «Обувь»:

При запуске приложения окно входа – первое, что видит пользователь. На ней пользователю предлагается ввести свой логин и пароль взятой из базы данных или есть возможность перейти на экран просмотра товаров (без фильтрации, сортировки, поиска) в роли гостя.

Только после удачной авторизации пользователь получает доступ к остальным модулям системы.

- авторизованный клиент может просматривать товары (без фильтрации, сортировки, поиска);
- менеджер может просматривать товары (с фильтрацией, сортировкой, поиском), просматривать заказы;
- администратор может просматривать (с фильтрацией, сортировкой, поиском)/добавлять/редактировать/удалять товары, просматривать /добавлять/редактировать/удалять заказы.

### 3. Критерии оценки

#### 1. Критерии оценки задач

При оценке задач учитываются все ошибки (грубые и негрубые) и недочёты.

Грубыми считаются ошибки:

- незнание определения основных понятий, правил, незнание формул, общепринятых символов обозначений величин, единиц их измерения;
- неумение применять знания, алгоритмы для решения задач;
- неправильное решение задания (пропуск действия, неправильный выбор действий, лишние действия);
- нерешенная до конца задача или пример;
- невыполненное задание;
- вычислительные ошибки, если они не являются опиской;
- логические ошибки.

К негрубым ошибкам следует отнести:

- неточность формулировок, определений, понятий, теорий, вызванная неполнотой охвата основных признаков определяемого понятия или заменой одного - двух из этих признаков второстепенными;
- неточность графика;
- нерациональный метод решения задачи или недостаточно продуманный план ответа (нарушение логики, подмена отдельных основных вопросов второстепенными);
- неумение решать задачи, выполнять задания в общем виде.

Недочетами являются:

- нерациональные приемы вычислений и преобразований;
- небрежное выполнение записей, чертежей, схем, графиков.

Примечание: за грамматические ошибки, допущенные в решении заданий, оценка не снижается. За неряшливо оформленное задание, несоблюдение правил каллиграфии оценка снижается на 1 балл, но не ниже «3».

Критерии оценки выполнения задач

Оценка уровня подготовки		Имеющийся результат
Балл (отметка)	Вербальный аналог	
5	Отлично	- задача выполнена полностью. - в логических рассуждениях и обоснованиях нет пробелов и ошибок; - в решении нет математических ошибок (возможна одна неточность, описка, не являющаяся следствием незнания или непонимания учебного материала)
4	Хорошо	- задача выполнена полностью, но обоснования шагов решения недостаточны (если умения обосновывать рассуждения не являлись специальным объектом проверки); - допущена одна ошибка или два-три недочета в выкладках, чертежах или графиках (если эти

		виды работы не являлись специальным объектом проверки)
3	Удовлетворительно	- допущены более одной ошибки или более двух- трех недочетов в выкладках, чертежах или графика, но обучающийся владеет обязательными умениями по проверяемой теме
2	Неудовлетворительно	- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями по данной теме в полной мере; - выполненное задание показало полное отсутствие у обучающегося обязательных знаний, умений по проверяемой теме или значительная часть заданий выполнена не самостоятельно.

## 2. Критерии оценки устного опроса:

Оценка «отлично» выставляется студенту, сформулировавшему полный и правильный ответ на вопрос, логично структурировавшему и изложившему материал. При этом студент должен показать знание специальной литературы. Для получения отличной оценки необходимо продемонстрировать умение обозначить проблемные вопросы в соответствующей области, проанализировать их и предложить варианты решений, дать исчерпывающие ответы на уточняющие и дополнительные вопросы.

Оценка «хорошо» выставляется студенту, который дал полный правильный ответ на вопрос, с соблюдением логики изложения материала, но допустил при ответе отдельные неточности, не имеющие принципиального характера. Оценка «хорошо» может выставляться студенту, недостаточно чётко и полно ответившему на уточняющие и дополнительные вопросы.

Оценка «удовлетворительно» выставляется студенту, показавшему неполные знания, допустившему ошибки и неточности при ответе на вопрос, продемонстрировавшему неумение логически выстроить материал ответа и сформулировать свою позицию по проблемным вопросам. При этом хотя бы по одному из заданий ошибки не должны иметь принципиального характера. Студент, ответ которого оценивается «удовлетворительно», должен опираться в своем ответе на учебную литературу.

Оценка «неудовлетворительно» выставляется студенту, если он не дал ответа на вопрос; дал неверные, содержащие фактические ошибки ответы на все вопросы; не смог ответить на дополнительные и уточняющие вопросы. Неудовлетворительная оценка выставляется студенту, отказавшемуся отвечать на вопросы семинара

## 3. Критерии оценки теста:

оценка «5» - правильных ответов 88–100%;

оценка «4» - правильных ответов 68–87%;

оценка «3» - правильных ответов 50–67%;

оценка «2» - правильных ответов < 50%.

4. Критерии оценки дифференцированного зачета:

Оценка «5» ставится, если:

- студент свободно применяет знания на практике;
- не допускает ошибок в воспроизведении изученного материала;
- студент выделяет главные положения в изученном материале и не затрудняется в ответах на видоизмененные вопросы;
- студент усваивает весь объем программного материала;
- материал оформлен аккуратно в соответствии с требованиями.

Оценка «4» ставится, если:

- студент знает весь изученный материал;
- отвечает без особых затруднений на вопросы преподавателя;
- студент умеет применять полученные знания на практике;
- в ответах не допускает серьезных ошибок, легко устраняет определенные неточности с помощью дополнительных вопросов преподавателя;
- материал оформлен недостаточно аккуратно и в соответствии с требованиями;

Оценка «3» ставится, если:

- студент обнаруживает освоение основного материала, но испытывает затруднения при его самостоятельном воспроизведении и требует дополнительных дополняющих вопросов преподавателя;
- предпочитает отвечать на вопросы воспроизводящего характера и испытывает затруднения при ответах на воспроизводящие вопросы;
- материал оформлен не аккуратно или не в соответствии с требованиями;

Оценка «2» ставится, если:

- у студента имеются отдельные представления об изучаемом материале, но все же большая часть не усвоена;
- материал оформлен не в соответствии с требованиями.