Федеральное государственное автономное образовательное учреждение высшего образования
"Самарский государственный экономический университет"

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ (МОДУЛЯ) «АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ»

Уровень высшего образования: бакалавриат

Направление подготовки: 09.03.03 Прикладная информатика

Направленность (профиль)подготовки: Интеллектуальные цифровые системы и сервисы в управлении

Квалификация (степень) выпускника: бакалавр

Форма обучения: очная

Год набора (приема на обучение): 2026

Срок получения образования: 4 года

Объем: в зачетных единицах: 4 з.е.
в академических часах: 144 ак.ч.

г. Самара, 2026

Разработчики:
Мантуленко А. В.

Рабочая программа дисциплины (модуля) составлена в соответствии с требованиями ФГОС ВО по направлению подготовки 09.03.03 Прикладная информатика, утвержденного приказом Минобрнауки от 19.09.2017 № 922, с учетом трудовых функций профессиональных стандартов: "Менеджер по информационным технологиям", утвержден приказом Минтруда России от 30.08.2021 № 588н; "Специалист по информационным системам", утвержден приказом Минтруда России от 13.07.2023 № 586н; "Руководитель проектов в области информационных технологий", утвержден приказом Минтруда России от 27.04.2023 № 369н; "Системный аналитик", утвержден приказом Минтруда России от 27.04.2023 № 367н.

Согласование и утверждение

№	Подразделение или коллегиальный орган	Ответственное лицо	ФИО	Виза	Дата, протокол (при наличии)
1	Кафедра прикладной информатики	Заведующий кафедрой, руководитель подразделения, реализующего ОП	Ермолина Л. В.	Рассмотрено	21.05.2026, № 9

1. Цель и задачи освоения дисциплины (модуля)

Цель освоения дисциплины - Сформировать у студентов алгоритмическое мышление и практические навыки разработки программного обеспечения, позволяющие формализовать прикладную задачу и реализовать ее эффективное решение на компьютере.

Задачи изучения дисциплины:

- Изучить базовые алгоритмы сортировки, поиска, обхода графов, работы со строками и динамического программирования.;
- Понимать устройство, преимущества и недостатки массивов, списков, стеков, очередей, деревьев и хеш-таблиц.;
- Освоить принципы структурного (процедурного) и объектно-ориентированного программирования (ООП).

2. Планируемые результаты обучения по дисциплине (модулю), соотнесенные с планируемыми результатами освоения образовательной программы

Компетенции, индикаторы и результаты обучения

УК-1 Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач

УК-1.1 Осуществляет поиск, критический анализ и синтез информации

Знать:

УК-1.1/Зн1 Знать основные типы источников технической информации (официальная документация, академические статьи, репозитории GitHub, StackOverflow, форумы, блоги экспертов) и их уровень достоверности

Уметь:

УК-1.1/Ум1 Уметь читать текст ошибки (Traceback) и отделять ключевую причину падения от сопутствующих предупреждений

Владеть:

УК-1.1/Нв1 Владеть техникой «приёмочных тестов» (assertion) — способностью моментально проверять корректность найденной информации на простом примере (тест-кейсе) до внедрения в основную программу

УК-1.2 Применяет системный подход для решения поставленных задач

Знать:

УК-1.2/Зн1 Знать, что любую сложную задачу необходимо разбивать на модули (функции, классы, пакеты) по принципу единственной ответственности (SRP из SOLID)

Уметь:

УК-1.2/Ум1 Уметь нарисовать граф взаимодействия между объектами программы и определить, что произойдет с системой, если изменить один из модулей (анализ влияния изменений)

Владеть:

УК-1.2/Нв1 Владеть умением при возникновении ошибки не править её локально, а проследить всю цепочку вызовов (Call Stack), чтобы понять, в каком слое системы нарушена логика

ОПК-5 Способен инсталлировать программное и аппаратное обеспечение для информационных и автоматизированных систем

ОПК-5.1 Осуществляет параметрическую настройку информационных и автоматизированных систем в рамках выполнения профессиональных задач

Знать:

ОПК-5.1/Зн1 Виды параметров: Знать разницу между статическими (задаются при компиляции), динамическими (меняются на лету) и пользовательскими (настройки под юзера) параметрами. Способы хранения настроек: Знать форматы конфигурационных файлов (.ini, .json, .yaml, .xml, .env) и их синтаксис; понимать, какой формат для каких задач предназначен. Иерархию настроек (приоритеты): Знать правило, что параметры, заданные через командную строку, имеют высший приоритет, затем переменные окружения, затем файлы конфигурации, затем значения по умолчанию в коде (12-factor app).

Уметь:

ОПК-5.1/Ум1 Уметь находить в коде числа (размер буфера, таймаут, лимит запросов, путь к папке) и заменять их на переменные, считываемые из конфига. Настраивать подключения: Уметь изменять параметры подключения к базам данных (хост, порт, логин, пароль) без пересборки проекта, чтобы одна и та же программа работала с тестовой и боевой БД. Менять поведение алгоритма через флаги: Уметь вводить логические параметры (флаги), которые включают/отключают режим отладки, логирование, кеширование или использование потоков.

Владеть:

ОПК-5.1/Нв1 Навык работы с парсерами конфигов: Владеть библиотеками для чтения конфигураций (например, PyYAML, Jackson для Java, json-c для C) — уметь мгновенно написать функцию load_config(), которая валидирует наличие обязательных ключей. Навык безопасного хранения секретов: Владеть техникой не сохранять пароли и токены в репозитории Git, а подгружать их из внешнего хранилища (или .env файла, добавленного в .gitignore). Навык "горячей" перезагрузки: Владеть (на продвинутом уровне) техникой перечитывания конфига во время работы программы без ее перезапуска (по сигналу операционной системы или по таймеру).

ОПК-5.2 Проводит работы по инсталляции программного обеспечения и развертыванию аппаратной инфраструктуры информационных и автоматизированных систем

Знать:

ОПК-5.2/Зн1 Основы архитектуры информационных и автоматизированных систем, чтобы понимать, как взаимодействуют программные и аппаратные компоненты и какую роль выполняет каждое звено в системе

Уметь:

ОПК-5.2/Ум1 Выбирать и подбирать необходимое программное и аппаратное обеспечение для решения конкретных профессиональных задач

Владеть:

ОПК-5.2/Нв1 Навыками инсталляции программного и аппаратного обеспечения информационных и автоматизированных систем

ОПК-7 Способен разрабатывать алгоритмы и программы, пригодные для практического применения

ОПК-7.1 Разрабатывает алгоритмы и программы, пригодные для практического применения

Знать:

ОПК-7.1/Зн1 Критерии качества ПО: Знать параметры оценки готового кода: надежность (не падает), безопасность (не дает взломать себя), масштабируемость (выдерживает рост данных), сопровождаемость (легко чинить и дополнять).

Уметь:

ОПК-7.1/Ум1 Обрабатывать ошибки (Error Handling): Уметь писать код, который не падает при отсутствии файла, отказе сети или вводе букв вместо цифр. Обрабатывать исключения на том уровне архитектуры, где их можно осмысленно исправить.

Владеть:

ОПК-7.1/Нв1 Навык покрытия тестами: Владеть навыком написания как минимум юнит-тестов (проверяющих функции) и интеграционных тестов (проверяющих взаимодействие модулей), чтобы убедиться, что программа делает именно то, что обещано в ТЗ.

ОПК-7.2 Интегрирует программные модули в различные операционные системы и оболочки, применяя языки программирования, методы отладки и тестирования работоспособности программы

Знать:

ОПК-7.2/Зн1 Принципы интеграции программных модулей — способы объединения отдельных компонентов в единую работающую систему, методы сборки, линковки и компоновки

Уметь:

ОПК-7.2/Ум1 Интегрировать разработанные программные модули в целевую среду исполнения, обеспечивая их корректное взаимодействие с другими компонентами системы

Владеть:

ОПК-7.2/Нв1 Навыками сборки и компоновки программных проектов в различных операционных системах и оболочках

3. Место дисциплины в структуре ОП

Дисциплина (модуль) «Алгоритмизация и программирование» относится к обязательной части образовательной программы и изучается в семестре(ах): 1.

В процессе изучения дисциплины студент готовится к решению типов задач профессиональной деятельности, предусмотренных ФГОС ВО и образовательной программой.

Компетенция	Предшествующие дисциплины	Последующие дисциплины
ОПК-5 - Способен инсталлировать программное и аппаратное обеспечение для информационных и автоматизированных систем		
ОПК-5.1 Осуществляет параметрическую настройку информационных и автоматизированных систем в рамках выполнения профессиональных задач		Выполнение и защита выпускной квалификационной работы, Операционные системы и оболочки, Производственная практика: преддипломная практика
ОПК-5.2 Проводит работы по инсталляции программного обеспечения и развертыванию аппаратной инфраструктуры информационных и автоматизированных систем		Выполнение и защита выпускной квалификационной работы, Операционные системы и оболочки, Производственная практика: преддипломная практика
ОПК-7 - Способен разрабатывать алгоритмы и программы, пригодные для практического применения		

ОПК-7.1 Разрабатывает алгоритмы и программы, пригодные для практического применения	Высшая математика	Выполнение и защита выпускной квалификационной работы, Высшая математика, Программирование, Проектирование баз данных, Производственная практика: преддипломная практика, Учебная практика: ознакомительная практика
ОПК-7.2 Интегрирует программные модули в различные операционные системы и оболочки, применяя языки программирования, методы отладки и тестирования работоспособности программы		Выполнение и защита выпускной квалификационной работы, Программирование, Проектирование баз данных, Производственная практика: преддипломная практика, Учебная практика: ознакомительная практика
УК-1 - Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач		
УК-1.1 Осуществляет поиск, критический анализ и синтез информации	История России	Выполнение и защита выпускной квалификационной работы, История России, Основы финансового и экономического анализа, Производственная практика: технологическая (проектно-технологическая) практика, Учебная практика: ознакомительная практика
УК-1.2 Применяет системный подход для решения поставленных задач	История России	Выполнение и защита выпускной квалификационной работы, История России, Производственная практика: технологическая (проектно-технологическая) практика, Учебная практика: ознакомительная практика

4. Объем дисциплины (модуля) и виды учебной работы

Период обучения	Общая трудоемкость (часы)	Общая трудоемкость (ЗЕТ)	Контактная работа (часы, всего)	Лабораторные занятия (часы)	Лекционные занятия (часы)	Групповая контактная работа (часы)	Индивидуальная контактная работа (часы)	Самостоятельная работа (часы)	Промежуточная аттестация
Первый семестр	144	4	72	36	36	2	0,3	35,7	Экзамен
Всего	144	4	72	36	36	2	0,3	35,7	34

5. Содержание дисциплины (модуля)

5.1. Разделы, темы дисциплины и виды занятий

(часы промежуточной аттестации не указываются)

		я		ота

Наименование раздела, темы	Всего	Лабораторные занятия	Лекционные занятия	Самостоятельная раб
Раздел 1. Теоретические основы, история языков программирования	**22**	**4**	**8**	**10**
Тема 1.1. Истроия языков прогрммирования	8		4	4
Тема 1.2. Объектно-ориентированное программирование	14	4	4	6
Раздел 2. Основы языка программирования Python	**67,7**	**26**	**24**	**17,7**
Тема 2.1. Основы синтаксиса языка программирования	30,7	12	12	6,7
Тема 2.2. Функции, модули в языках программирования	21	8	6	7
Тема 2.3. Основные классические алгоритмы	16	6	6	4
Раздел 3. Системы контроля версий	**12**	**4**	**2**	**6**
Тема 3.1. Git	12	4	2	6
Раздел 4. Тестирование кода	**8,3**	**2**	**2**	**2**
Тема 4.1. Модули тестирования	8,3	2	2	2

5.2. Контрольные мероприятия по дисциплине

Вид контроля	Форма контроля/Оценочное средство
Текущий контроль	Тестирование
Промежуточная аттестация	Экзамен

№ п/п	Наименование раздела	Вид контроля/ используемые оценочные материалы	
		Текущий	Промежут. аттестация
1	Теоретические основы, история языков программирования	Тестирование	Экзамен
2	Основы языка программирования Python	Тестирование	Экзамен
3	Системы контроля версий		Экзамен
4	Тестирование кода		Экзамен

6. Оценочные материалы текущего контроля

1. Теоретические основы, история языков программирования Тестирование

№ п/п	Содержание вопроса		Компетенция
		Правильный ответ (ключ ответа)	

1	Какой язык программирования считается первым в мире высокоуровневым языком? A) FORTRAN B) Plankalkül C) Lisp D) COBOL Ответ: B) Plankalkül	УК-1
2	В каком году был создан язык FORTRAN (FORmula TRANslation)? A) 1954 B) 1957 C) 1960 D) 1972 Ответ: B) 1957	УК-1
3	Какой язык был разработан в Microsoft как конкурент Java и вышел в 2000 году? A) Visual Basic B) C# C) F# D) TypeScript Ответ: B) C#	УК-1
4	Кто считается создателем языка Lisp? A) Алан Тьюринг B) Джон Маккарти C) Марвин Мински D) Норберт Винер Ответ: B) Джон Маккарти	УК-1
5	Какой язык программирования был создан для обработки данных и до сих пор широко используется в банковской сфере? A) FORTRAN B) COBOL C) ALGOL D) BASIC Ответ: B) COBOL	УК-1
6	Какой язык стал первым, реализовавшим автоматическую сборку мусора (garbage collection)? A) Java B) Python C) Lisp D) Smalltalk Ответ: C) Lisp	УК-1
7	В каком году был создан язык JavaScript (первоначально названный Mocha)? A) 1994 B) 1995 C) 1996 D) 1997 Ответ: B) 1995	УК-1
8	Соответствие между этапами интеграции программных модулей и их описанием 2. Линковка (компоновка) Б. Преобразование исходного кода в объектный код 3. Загрузка 4. Динамическая линковка 5. Статическая линковка И А. Объединение нескольких объектных файлов в один исполняемый файл Б. Преобразование исходного кода в объектный код В. Подключение внешних библиотек и разрешение внешних ссылок Г. Помещение исполняемого файла в память и подготовка к выполнению Д. Подключение библиотек во время выполнения программы Ответ: 1 → Б 2 → В (или А, в зависимости от трактовки; корректнее: линковка — объединение объектных файлов в исполняемый/библиотечный) 3 → Г 4 → Д 5 → А (объединение кода библиотек непосредственно в исполняемый файл)	УК-1

9	Соответствие между типами ошибок и их описанием Тип ошибки Описание 1. Синтаксическая ошибка А. Ошибка, возникающая во время выполнения программы (деление на ноль, обращение к нулевому указателю) 2. Логическая ошибка Б. Неправильное использование синтаксиса языка, обнаруженная на этапе компиляции 3. Ошибка времени выполнения (runtime) В. Ошибка, связанная с некорректным использованием памяти (утечки, двойное освобождение) 4. Ошибка управления памятью Г. Программа выполняется, но выдаёт неправильный результат из-за ошибки в алгоритме 5. Ошибка связывания (линковки) Д. Неразрешённые внешние ссылки, конфликт имён при компоновке модулей	УК-1
	Ответ: 1 → Б 2 → Г 3 → А 4 → В 5 → Д	
10	Соответствие между инструментами отладки и их назначением Инструмент и Назначение 1. GDB А. Средство для обнаружения утечек памяти и профилирования в Linux 2. Valgrind Б. Отладчик для языка Java и других JVM-языков 3. WinDbg В. Отладчик для C/C++ в Unix-подобных системах 4. JDB Г. Инструмент для трассировки системных вызовов в Linux 5. strace Д. Отладчик для Windows, используется для анализа дампов памяти	УК-1
	Ответ: 1 → В 2 → А 3 → Д 4 → Б 5 → Г	
11	Соответствие между типами тестирования и их характеристиками Тип тестирования и Характеристика 1. Модульное тестирование (Unit testing) А. Проверка взаимодействия между отдельными компонентами системы 2. Интеграционное тестирование Б. Тестирование готового продукта в условиях, приближенных к реальным 3. Системное тестирование В. Проверка отдельных функций, классов или методов изолированно 4. Регрессионное тестирование Г. Проверка системы на больших объёмах данных или высокой нагрузке 5. Нагрузочное тестирование Д. Проверка, что новые изменения не нарушили существующую функциональность	УК-1
	Ответ: 1 → В 2 → А 3 → Б 4 → Д 5 → Г	
12	Соответствие между типами исполняемых файлов и операционными системами Формат исполняемого файла и Операционная система 1. PE (Portable Executable) А. Linux, UNIX-подобные системы 2. ELF (Executable and Linkable Format) Б. macOS 3. Mach-O В. Windows 4. COFF Г. Ранние версии UNIX, Windows (редко) 5. a.out Д. Исторический формат UNIX (заменён ELF)	УК-1
	Ответ: 1 → В 2 → А 3 → Б 4 → Г 5 → Д	

13	Установите правильную последовательность этапов жизненного цикла разработки программного модуля Анализ требований Разработка и написание кода Проектирование архитектуры Сборка и компиляция Тестирование (модульное, интеграционное) Развертывание и интеграция в систему Отладка и исправление ошибок Сопровождение и поддержка	УК-1
	Ответ: 1. Анализ требований 2. Проектирование архитектуры 3. Разработка и написание кода 4. Сборка и компиляция 5. Отладка и исправление ошибок 6. Тестирование (модульное, интеграционное) 7. Развертывание и интеграция в систему 8. Сопровождение и поддержка	
14	Установите правильную последовательность процесса отладки программы Воспроизведение ошибки (найти сценарий, при котором возникает ошибка) Фиксация исправления в системе контроля версий Локализация ошибки (определение места в коде) Повторное тестирование для проверки исправления Анализ причины ошибки Внесение исправления в код Обнаружение ошибки (сообщение от пользователя или теста)	УК-1
	Ответ: 1. Обнаружение ошибки (сообщение от пользователя или теста) 2. Воспроизведение ошибки (найти сценарий, при котором возникает ошибка) 3. Локализация ошибки (определение места в коде) 4. Анализ причины ошибки 5. Внесение исправления в код 6. Повторное тестирование для проверки исправления 7. Фиксация исправления в системе контроля версий	
15	Установите правильную последовательность при интеграции программного модуля в существующую систему Проверка совместимости модуля с целевой операционной системой Тестирование интеграции в тестовой среде Анализ зависимостей модуля (библиотеки, версии) Развертывание модуля в продуктивной среде Настройка конфигурационных файлов и переменных окружения Сборка модуля для целевой платформы Мониторинг работы модуля после интеграции	УК-1
	Ответ: 1. Анализ зависимостей модуля (библиотеки, версии) 2. Проверка совместимости модуля с целевой операционной системой 3. Сборка модуля для целевой платформы 4. Настройка конфигурационных файлов и переменных окружения 5. Тестирование интеграции в тестовой среде 6. Развертывание модуля в продуктивной среде 7. Мониторинг работы модуля после интеграции	

2. Основы языка программирования Python Тестирование

№ п/п	Содержание вопроса	Компетенция
	Правильный ответ (ключ ответа)	
1	Каким символом в Python обозначается начало комментария? A) // B) # C) /* D) --	ОПК-5
	Ответ: B) #	
2	Какой из следующих вариантов является корректным объявлением переменной в Python? A) int a = 5; B) a := 5 C) a = 5 D) let a = 5	ОПК-5
	Ответ: C) a = 5	
3	Какой оператор используется для возведения числа в степень? A) ^ B) ** C) pow() D) %	ОПК-5
	Ответ: B) **	

4	Каким будет результат выполнения кода: print(3 * 'abc')? A) 'abcabcabc' B) 'abc' * 3 C) SyntaxError D) abcabcabc (вывод без кавычек) Ответ: D) abcabcabc (вывод без кавычек)	ОПК-5
5	Какой из вариантов является корректным способом записи условного оператора if в Python? A) python if x > 0: print("Positive") B) python if (x > 0) { print("Positive") } C) python if x > 0 then: print("Positive") D) python if x > 0: print("Positive") Ответ: A) python if x > 0: print("Positive")	ОПК-5
6	Что выведет следующий код? python x = [1, 2, 3] y = x y.append(4) print(x) A) [1, 2, 3] B) [1, 2, 3, 4] C) [1, 2, 3, 4, 4] D) None Ответ: B) [1, 2, 3, 4]	ОПК-5
7	Какой цикл в Python используется для перебора последовательности (итерации)? A) for B) while C) do-while D) foreach Ответ: A) for	ОПК-5
8	Что такое pass в Python? A) Оператор для выхода из цикла B) Оператор для пропуска итерации C) Пустая инструкция-заглушка D) Оператор для возврата значения Ответ: C) Пустая инструкция-заглушка	ОПК-5
9	Какой из вариантов правильно определяет функцию в Python? A) def myFunc(): B) function myFunc(): C) func myFunc(): D) define myFunc(): Ответ: A) def myFunc():	ОПК-5
10	Какой из вариантов является корректным способом записи условного оператора if в Python? A) python if x > 0: print("Positive") B) python if (x > 0) { print("Positive") } C) python if x > 0 then: print("Positive") D) python if x > 0: print("Positive") Ответ: A) python if x > 0: print("Positive")	ОПК-5

11	Соответствие между средствами интеграции программных модулей и их функцией Средство / Инструмент и Функция 1. Make / CMake А. Управление версиями исходного кода 2. Git Б. Автоматическая сборка, тестирование и развертывание 3. Docker В. Сборка программных проектов с управлением зависимостями 4. Jenkins / GitHub Actions Г. Управление пакетами и зависимостями в проекте 5. Maven / pip / npm Д. Контейнеризация приложений для обеспечения переносимости 6. Conan / vcpkg Е. Управление пакетами для C/C++ (управление библиотеками) Ответ: 1 → В 2 → А 3 → Д 4 → Б 5 → Г 6 → Е	ОПК-5
12	Соответствие между уровнями логирования и их назначением Уровень логирования и Назначение 1. ERROR А. Детальная информация о работе программы, используется при разработке 2. WARNING Б. Информационные сообщения о ключевых событиях в системе 3. INFO В. Критические ошибки, требующие немедленного вмешательства 4. DEBUG Г. Сообщения о потенциальных проблемах, которые не нарушают работу 5. FATAL / CRITICAL Д. Сообщения о неисправимых ошибках, приводящих к остановке программы Ответ: 1 → В (или Д, если FATAL отдельно) 2 → Г 3 → Б 4 → А 5 → Д	ОПК-5
13	Соответствие между методами разрешения конфликтов при интеграции и их описанием Метод разрешения конфликтов и Описание 1. "Последняя версия побеждает" А. Выбор наиболее старой (стабильной) версии модуля 2. Ручное разрешение Б. Автоматический выбор версии с более высоким номером 3. "Старая версия побеждает" В. Разработчик вручную анализирует и исправляет конфликт 4. Использование абстракций Г. Переход на более новую версию с учётом обратной совместимости 5. Миграция на новую версию Д. Создание прослойки (адаптера) между несовместимыми модулями Ответ: 1 → Б 2 → В 3 → А 4 → Д 5 → Г	ОПК-5
14	Соответствие между средой выполнения (runtime) и её характеристикой Среда выполнения и Характеристика 1. JVM (Java Virtual Machine) А. Интерпретатор для Python, динамическая типизация 2. CLR (Common Language Runtime) Б. Сборка мусора, JIT-компиляция, платформонезависимость 3. CPython В. Использует байт-код и Just-In-Time (JIT) компиляцию, разработана Microsoft 4. V8 (Chrome) Г. Среда выполнения для JavaScript, JIT-компиляция 5. Erlang VM (BEAM) Д. Лёгкие процессы, отказоустойчивость, горячая замена кода Ответ: 1 → Б 2 → В 3 → А 4 → Г 5 → Д	ОПК-5

15	Соответствие между методологиями тестирования и их применением Методология / Подход и Применение 1. TDD (Test-Driven Development) А. Тесты пишутся после написания кода для проверки корректности 2. BDD (Behavior-Driven Development) Б. Автоматическое выполнение тестов при каждом изменении кода 3. Тестирование "белого ящика" В. Тесты описываются на естественном языке в терминах поведения системы 4. Тестирование "чёрного ящика" Г. Тесты пишутся до написания кода; код пишется под тесты 5. Continuous Testing Д. Тестирование внутренней структуры и логики программы 6. Тестирование "серого ящика" Е. Тестирование без знания внутреннего устройства системы Ж. Комбинированный подход: частично известна внутренняя структура	ОПК-5
	Ответ: 1 → Г 2 → В 3 → Д 4 → Е 5 → Б 6 → Ж	
16	Установите правильную последовательность действий при разрешении конфликта версий библиотек Выбор стратегии разрешения конфликта (ручное/автоматическое) Обновление файла зависимостей (например, requirements.txt, package.json, CMakeLists.txt) Идентификация конфликтующих версий библиотек Повторная сборка и тестирование проекта Анализ обратной совместимости библиотек Фиксация изменений в системе контроля версий	ОПК-7
	Ответ: 1. Идентификация конфликтующих версий библиотек 2. Анализ обратной совместимости библиотек 3. Выбор стратегии разрешения конфликта (ручное/автоматическое) 4. Обновление файла зависимостей (например, requirements.txt, package.json, CMakeLists.txt) 5. Повторная сборка и тестирование проекта 6. Фиксация изменений в системе контроля версий	
17	Установите правильную последовательность выполнения тестирования программного модуля Системное тестирование Интеграционное тестирование Приёмочное тестирование (UAT) Модульное тестирование (Unit-тесты) Регрессионное тестирование	ОПК-7
	Ответ: 1. Модульное тестирование (Unit-тесты) 2. Интеграционное тестирование 3. Системное тестирование 4. Регрессионное тестирование (может выполняться после любых изменений) 5. Приёмочное тестирование (UAT)	
18	Установите правильную последовательность действий при работе с отладчиком (GDB) Установка точки останова (breakpoint) Запуск программы под отладчиком (run) Загрузка отладочной информации (символов) Запуск отладчика с исполняемым файлом (gdb ./program) Пошаговое выполнение (step, next) Анализ значения переменных (print) Завершение сеанса отладки (quit)	ОПК-7
	Ответ: 1. Запуск отладчика с исполняемым файлом (gdb ./program) 2. Загрузка отладочной информации (символов) 3. Установка точки останова (breakpoint) 4. Запуск программы под отладчиком (run) 5. Пошаговое выполнение (step, next) 6. Анализ значения переменных (print) 7. Завершение сеанса отладки (quit)	
19	Установите правильную последовательность действий при развертывании программного обеспечения в операционной системе Проверка системных требований (ОС, процессор, память) Распаковка/копирование файлов ПО в целевые директории Настройка переменных окружения Установка необходимых зависимостей (библиотеки, драйверы) Запуск установочного скрипта или инсталлятора Конфигурирование параметров работы приложения Проверка работоспособности установленного ПО	ОПК-7

	Ответ:	1. Проверка системных требований (ОС, процессор, память) 2. Установка необходимых зависимостей (библиотеки, драйверы) 3. Запуск установочного скрипта или инсталлятора 4. Распаковка/копирование файлов ПО в целевые директории 5. Настройка переменных окружения 6. Конфигурирование параметров работы приложения 7. Проверка работоспособности установленного ПО	
20	Установите правильную последовательность действий при обработке исключительных ситуаций в программе Логирование ошибки с указанием контекста Возникновение исключительной ситуации Обработка ошибки в блоке catch/except Выполнение блока finally (освобождение ресурсов) Генерация исключения (throw/raise) Уведомление пользователя или администратора Продолжение выполнения или завершение программы		ОПК-5
	Ответ:	1. Возникновение исключительной ситуации 2. Генерация исключения (throw/raise) 3. Обработка ошибки в блоке catch/except 4. Логирование ошибки с указанием контекста 5. Выполнение блока finally (освобождение ресурсов) 6. Уведомление пользователя или администратора 7. Продолжение выполнения или завершение программы	
21	Установите правильную последовательность анализа дампа памяти (core dump) после аварийного завершения программы Определение причины аварийного завершения (сигнал, код ошибки) Загрузка дампа в отладчик (gdb core) Получение дампа памяти (core file) Вывод содержимого стека (backtrace) Анализ значений переменных и регистров в момент падения Восстановление последовательности вызовов функций Выявление проблемного участка кода		ОПК-5
	Ответ:	1. Получение дампа памяти (core file) 2. Загрузка дампа в отладчик (gdb core) 3. Восстановление последовательности вызовов функций 4. Вывод содержимого стека (backtrace) 5. Анализ значений переменных и регистров в момент падения 6. Определение причины аварийного завершения (сигнал, код ошибки) 7. Выявление проблемного участка кода	
22	Установите правильную последовательность действий при сборке проекта с использованием системы сборки (CMake/Make) Выполнение команды make (сборка проекта) Запуск CMake для генерации Makefile (cmake .) Установка переменных окружения для компилятора Написание/обновление файла CMakeLists.txt Выполнение make install (установка собранных файлов) Очистка предыдущей сборки (make clean)		ОПК-5
	Ответ:	1. Написание/обновление файла CMakeLists.txt 2. Установка переменных окружения для компилятора 3. Запуск CMake для генерации Makefile (cmake .) 4. Выполнение команды make (сборка проекта) 5. Очистка предыдущей сборки (make clean) (опционально, если нужно) 6. Выполнение make install (установка собранных файлов)	

23	Установите правильную последовательность действий при автоматическом тестировании в CI/CD пайплайне Развертывание в продуктивную среду Выполнение интеграционных тестов Сборка проекта Запуск модульных тестов Получение кода из репозитория Отчёт о результатах тестирования Анализ покрытия кода тестами	ОПК-5
	Ответ: 1. Получение кода из репозитория 2. Сборка проекта 3. Запуск модульных тестов 4. Выполнение интеграционных тестов 5. Анализ покрытия кода тестами 6. Отчёт о результатах тестирования 7. Развертывание в продуктивную среду (если все тесты успешны)	
24	При интеграции программного модуля в систему выяснилось, что модуль требует библиотеку версии 2.0, а в системе уже установлена библиотека версии 1.8 с критическими изменениями API. Какой подход является наиболее правильным? A) Принудительно перезаписать библиотеку версии 1.8 на версию 2.0 B) Переименовать библиотеку версии 2.0 и установить её в отдельную директорию, настроив пути для модуля C) Переписать модуль, чтобы он работал с версией 1.8 D) Игнорировать конфликт и попытаться запустить модуль	ОПК-7
	Ответ: Правильный ответ: B Обоснование: Установка разных версий библиотек в изолированные директории (например, с использованием RPATH, LD_LIBRARY_PATH или механизмов изоляции вроде контейнеров) позволяет избежать конфликта и обеспечивает работу как старого ПО, так и нового модуля. Перезапись (A) может сломать другие приложения, использующие старую версию. Переписывание модуля (C) — это дорогостоящее и часто неоправданное решение, особенно если модуль поставляется сторонним разработчиком. Игнорирование конфликта (D) приведёт к ошибкам линковки или непредсказуемому поведению во время выполнения.	
25	Какая последовательность действий наиболее эффективна при локализации ошибки, возникшей в интегрированном программном модуле? A) Сразу начать исправлять код, ориентируясь на описание ошибки пользователем B) Собрать все логи, воспроизвести ошибку, изолировать участок кода, затем приступить к исправлению C) Заменить весь модуль на предыдущую версию и проверить, исчезла ли ошибка D) Переустановить операционную систему и развернуть систему заново	ОПК-7
	Ответ: Эффективная стратегия отладки предполагает: сначала сбор информации (логи, дампы), затем воспроизведение ошибки в контролируемых условиях для понимания её причин, после чего изоляцию проблемного участка кода и только потом внесение исправлений. Вариант A — рискованный, так как без анализа можно внести новые ошибки. Вариант C — это откат (rollback), который не решает проблему, а лишь временно её маскирует. Вариант D — крайняя мера, которая нерациональна и не помогает выявить истинную причину ошибки. Обоснование: Обоснование: Эффективная стратегия отладки предполагает: сначала сбор информации (логи, дампы), затем воспроизведение ошибки в контролируемых условиях для понимания её причин, после чего изоляцию проблемного участка кода и только потом внесение исправлений. Вариант A — рискованный, так как без анализа можно внести новые ошибки. Вариант C — это откат (rollback), который не решает проблему, а лишь временно её маскирует. Вариант D — крайняя мера, которая нерациональна и не помогает выявить истинную причину ошибки.	
26	Какие виды тестирования должны быть выполнены в первую очередь при интеграции нового модуля в систему? A) Системное и нагрузочное тестирование B) Модульное и интеграционное тестирование C) Приёмочное тестирование пользователями (UAT) D) Регрессионное тестирование	ОПК-7

	Ответ:	На начальном этапе интеграции сначала проверяют корректность работы самого модуля изолированно (модульное тестирование), а затем его взаимодействие с другими компонентами (интеграционное тестирование). Системное и нагрузочное тестирование (A) проводятся позже, когда модуль уже прошел базовые проверки. Приёмочное тестирование (C) — это финальный этап, выполняемый заказчиком. Регрессионное тестирование (D) важно, но оно проверяет, что новые изменения не нарушили уже существующую функциональность, и выполняется после успешного прохождения модульных и интеграционных тестов. Обоснование: Обоснование: На начальном этапе интеграции сначала проверяют корректность работы самого модуля изолированно (модульное тестирование), а затем его взаимодействие с другими компонентами (интеграционное тестирование). Системное и нагрузочное тестирование (A) проводятся позже, когда модуль уже прошел базовые проверки. Приёмочное тестирование (C) — это финальный этап, выполняемый заказчиком. Регрессионное тестирование (D) важно, но оно проверяет, что новые изменения не нарушили уже существующую функциональность, и выполняется после успешного прохождения модульных и интеграционных тестов.	
27	Что является наиболее вероятной причиной ошибки "undefined reference" (неразрешённая ссылка) на этапе сборки проекта? A) Синтаксическая ошибка в исходном коде B) Отсутствие необходимого заголовочного файла (#include) C) Ошибка на этапе линковки — не указана библиотека, содержащая реализацию функции D) Ошибка времени выполнения в работающей программе		ОПК-5
	Ответ:	Ошибка "undefined reference" возникает на этапе компоновки (линковки), когда компоновщик не может найти реализацию (тело) объявленной функции. Это происходит, если не указана соответствующая библиотека или объектный файл с реализацией. Синтаксическая ошибка (A) была бы обнаружена на этапе компиляции. Отсутствие заголовочного файла (B) привело бы к ошибке "неизвестный тип" или "необъявленная функция" на этапе компиляции. Ошибка времени выполнения (D) — это ошибка, которая возникает уже после успешной сборки во время работы программы. Обоснование: Обоснование: Ошибка "undefined reference" возникает на этапе компоновки (линковки), когда компоновщик не может найти реализацию (тело) объявленной функции. Это происходит, если не указана соответствующая библиотека или объектный файл с реализацией. Синтаксическая ошибка (A) была бы обнаружена на этапе компиляции. Отсутствие заголовочного файла (B) привело бы к ошибке "неизвестный тип" или "необъявленная функция" на этапе компиляции. Ошибка времени выполнения (D) — это ошибка, которая возникает уже после успешной сборки во время работы программы.	
28	Какой метод тестирования наиболее подходит для проверки того, что интеграция нового модуля не нарушила работу существующих функций системы? A) Модульное тестирование B) Интеграционное тестирование C) Регрессионное тестирование D) Нагрузочное тестирование		ОПК-5
	Ответ:	Регрессионное тестирование специально предназначено для проверки того, что новые изменения (например, интеграция нового модуля) не нарушили уже существующую функциональность. Модульное тестирование (A) проверяет отдельные компоненты изолированно, а не их совместную работу. Интеграционное тестирование (B) проверяет взаимодействие между компонентами, но его цель — убедиться в корректности новых связей, а не в сохранении старой функциональности. Нагрузочное тестирование (D) проверяет производительность системы, а не корректность работы. Обоснование: Обоснование: Регрессионное тестирование специально предназначено для проверки того, что новые изменения (например, интеграция нового модуля) не нарушили уже существующую функциональность. Модульное тестирование (A) проверяет отдельные компоненты изолированно, а не их совместную работу. Интеграционное тестирование (B) проверяет взаимодействие между компонентами, но его цель — убедиться в корректности новых связей, а не в сохранении старой функциональности. Нагрузочное тестирование (D) проверяет производительность системы, а не корректность работы.	
29	Какая среда выполнения (runtime) использует байт-код и JIT-компиляцию для обеспечения кроссплатформенности, а также имеет встроенную сборку мусора? A) CPython B) JVM (Java Virtual Machine) C) V8 (движок JavaScript) D) CLR (Common Language Runtime)		ОПК-7
	Ответ:	Правильный ответ: B Обоснование: JVM (Java Virtual Machine) компилирует Java-байт-код в машинный код с помощью JIT-компиляции, обеспечивая кроссплатформенность ("write once, run anywhere"), и имеет автоматическую сборку мусора. CPython (A) — это интерпретатор для Python, который не использует JIT-компиляцию (в стандартной реализации). V8 (C) — это движок JavaScript, который использует JIT, но не является кроссплатформенной средой в том же смысле, что JVM (он привязан к браузерам или Node.js). CLR (D) — также использует JIT и сборку мусора, но вопрос сформулирован так, что наиболее точным ответом является JVM, так как CLR более привязан к экосистеме Microsoft Windows.	

30	Какое действие необходимо выполнить в первую очередь при возникновении ошибки времени выполнения (runtime error) в интегрированном модуле? A) Перезагрузить сервер и попытаться запустить модуль снова B) Проанализировать логи ошибок (стек вызовов, сообщение об ошибке, дамп памяти) C) Удалить модуль и переустановить его заново D) Написать сообщение разработчику с просьбой исправить ошибку		ОПК-7
	Ответ:	Правильный ответ: B	

7. Оценочные материалы промежуточной аттестации

Экзамен первый семестр

№ п/п	Содержание вопроса	Правильный ответ (ключ ответа)	Компетенция
1	Определение алгоритма. Какими свойствами обладает алгоритм (массовость, дискретность, детерминированность, результативность)? Поясните каждое свойство.		ОПК-5, ОПК-7, УК-1
	Ответ:		
2	Способы записи алгоритмов. Опишите словесный, графический (блок-схемы), псевдокод и программный способы записи. В чём преимущества и недостатки каждого?		ОПК-5, ОПК-7, УК-1
	Ответ:		
3	Базовые алгоритмические структуры. Что такое следование, ветвление и цикл? Приведите примеры и объясните, почему этих трёх конструкций достаточно для реализации любого алгоритма (теорема Бёма-Якопини).		ОПК-5, ОПК-7, УК-1
	Ответ:		
4	Классификация циклов. Чем отличаются циклы с предусловием, постусловием и параметрические (со счётчиком)? В каких случаях какой тип предпочтительнее?		ОПК-5, ОПК-7, УК-1
	Ответ:		
5	Понятие сложности алгоритмов. Что такое временная и ёмкостная сложность? Объясните, что означают обозначения O-большое, Ω и Θ.		ОПК-5, ОПК-7, УК-1
	Ответ:		
6	Анализ сложности. Приведите примеры алгоритмов с константной, линейной, квадратичной, логарифмической и экспоненциальной сложностью.		ОПК-5, ОПК-7
	Ответ:		
7	Рекурсия. Дайте определение рекурсивной функции. В чём отличие прямой и косвенной рекурсии? Что такое базовый случай и шаг рекурсии?		УК-1
	Ответ:		
8	Рекурсия vs итерация. Сравните рекурсивный и итеративный подходы: преимущества, недостатки, влияние на память и производительность. Приведите пример задачи, где рекурсия предпочтительнее.		ОПК-5
	Ответ:		
9	Линейные структуры данных. Дайте характеристику массивам и связным спискам. Сравните их по операциям доступа, вставки и удаления. Что такое односвязный, двусвязный и кольцевой список?		ОПК-7
	Ответ:		
10	Стек и очередь. Определите структуры данных "стек" и "очередь". На каких принципах они работают (LIFO/FIFO)? Приведите примеры практического применения каждой структуры.		ОПК-7
	Ответ:		
11	Хеш-таблицы. Что такое хеш-таблица? Как работает хеш-функция? Что такое коллизия и какие методы её разрешения существуют (метод цепочек, открытая адресация)?		ОПК-5, ОПК-7, УК-1
	Ответ:		
12	Графы: основные понятия. Дайте определения: граф, вершина, ребро, ориентированный/неориентированный граф, взвешенный граф, степень вершины, путь, цикл.		ОПК-5, ОПК-7, УК-1
	Ответ:		
13	Обход графов. Опишите алгоритмы обхода графа в глубину (DFS) и в ширину (BFS). В чём их принципиальное отличие? Где какой обход применяется?		ОПК-5, ОПК-7, УК-1
	Ответ:		
14	Поиск кратчайших путей. Сформулируйте задачу поиска кратчайшего пути. Опишите алгоритм Дейкстры. Для каких графов он работает и какова его сложность?		ОПК-5, ОПК-7, УК-1
	Ответ:		
15	Динамическое программирование. Раскройте суть метода динамического программирования. Чем он отличается от жадных алгоритмов? Приведите классическую задачу, решаемую этим методом (например, задача о рюкзаке, числа Фибоначчи, редактирование строк).		ОПК-5, ОПК-7, УК-1
	Ответ:		

7.1. Уровни овладения

Компетенция: УК-1 Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач.

Индикатор достижения компетенции: УК-1.1 Осуществляет поиск, критический анализ и синтез информации.

Уровень	Характеристика	Оценка в баллах
Повышенный	Достигнуто полное овладение знаниями, умениями и навыками. Студент свободно владеет терминологией, умеет применять теоретические знания в различных ситуациях для решения поставленных задач.	81-100
Базовый	Достигнуто достаточное овладение знаниями, умениями и навыками. Студент уверенно владеет терминологией, умеет применять теоретические знания в различных ситуациях для решения поставленных задач.	61-80
Пороговый	Достигнуто овладение минимально необходимыми знаниями, умениями и навыками. Студент владеет основной терминологией, умеет применять теоретические знания для решения поставленных задач в стандартных ситуациях.	41-60
Ниже порогового	Компетенция не освоена	0-40

Индикатор достижения компетенции: УК-1.2 Применяет системный подход для решения поставленных задач.

Уровень	Характеристика	Оценка в баллах
Повышенный	Достигнуто полное овладение знаниями, умениями и навыками. Студент свободно владеет терминологией, умеет применять теоретические знания в различных ситуациях для решения поставленных задач.	81-100
Базовый	Достигнуто достаточное овладение знаниями, умениями и навыками. Студент уверенно владеет терминологией, умеет применять теоретические знания в различных ситуациях для решения поставленных задач.	61-80
Пороговый	Достигнуто овладение минимально необходимыми знаниями, умениями и навыками. Студент владеет основной терминологией, умеет применять теоретические знания для решения поставленных задач в стандартных ситуациях.	41-60
Ниже порогового	Компетенция не освоена	0-40

Компетенция: ОПК-5 Способен инсталлировать программное и аппаратное обеспечение для информационных и автоматизированных систем.

Индикатор достижения компетенции: ОПК-5.1 Осуществляет параметрическую настройку информационных и автоматизированных систем в рамках выполнения профессиональных задач.

Уровень	Характеристика	Оценка в баллах
Повышенный	Достигнуто полное овладение знаниями, умениями и навыками. Студент свободно владеет терминологией, умеет применять теоретические знания в различных ситуациях для решения поставленных задач.	81-100

Базовый	Достигнуто достаточное овладение знаниями, умениями и навыками. Студент уверенно владеет терминологией, умеет применять теоретические знания в различных ситуациях для решения поставленных задач.	61-80
Пороговый	Достигнуто овладение минимально необходимыми знаниями, умениями и навыками. Студент владеет основной терминологией, умеет применять теоретические знания для решения поставленных задач в стандартных ситуациях.	41-60
Ниже порогового	Компетенция не освоена	0-40

Индикатор достижения компетенции: ОПК-5.2 Проводит работы по инсталляции программного обеспечения и развертыванию аппаратной инфраструктуры информационных и автоматизированных систем.

Уровень	Характеристика	Оценка в баллах
Повышенный	Достигнуто полное овладение знаниями, умениями и навыками. Студент свободно владеет терминологией, умеет применять теоретические знания в различных ситуациях для решения поставленных задач.	81-100
Базовый	Достигнуто достаточное овладение знаниями, умениями и навыками. Студент уверенно владеет терминологией, умеет применять теоретические знания в различных ситуациях для решения поставленных задач.	61-80
Пороговый	Достигнуто овладение минимально необходимыми знаниями, умениями и навыками. Студент владеет основной терминологией, умеет применять теоретические знания для решения поставленных задач в стандартных ситуациях.	41-60
Ниже порогового	Компетенция не освоена	0-40

Компетенция: ОПК-7 Способен разрабатывать алгоритмы и программы, пригодные для практического применения.

Индикатор достижения компетенции: ОПК-7.1 Разрабатывает алгоритмы и программы, пригодные для практического применения.

Уровень	Характеристика	Оценка в баллах
Повышенный	Достигнуто полное овладение знаниями, умениями и навыками. Студент свободно владеет терминологией, умеет применять теоретические знания в различных ситуациях для решения поставленных задач.	81-100
Базовый	Достигнуто достаточное овладение знаниями, умениями и навыками. Студент уверенно владеет терминологией, умеет применять теоретические знания в различных ситуациях для решения поставленных задач.	61-80
Пороговый	Достигнуто овладение минимально необходимыми знаниями, умениями и навыками. Студент владеет основной терминологией, умеет применять теоретические знания для решения поставленных задач в стандартных ситуациях.	41-60
Ниже порогового	Компетенция не освоена	0-40

Индикатор достижения компетенции: ОПК-7.2 Интегрирует программные модули в различные операционные системы и оболочки, применяя языки программирования, методы отладки и тестирования работоспособности программы.

Уровень	Характеристика	Оценка в баллах
Повышенный	Достигнуто полное овладение знаниями, умениями и навыками. Студент свободно владеет терминологией, умеет применять теоретические знания в различных ситуациях для решения поставленных задач.	81-100
Базовый	Достигнуто достаточное овладение знаниями, умениями и навыками. Студент уверенно владеет терминологией, умеет применять теоретические знания в различных ситуациях для решения поставленных задач.	61-80
Пороговый	Достигнуто овладение минимально необходимыми знаниями, умениями и навыками. Студент владеет основной терминологией, умеет применять теоретические знания для решения поставленных задач в стандартных ситуациях.	41-60
Ниже порогового	Компетенция не освоена	0-40

8. Материально-техническое и учебно-методическое обеспечение дисциплины

8.1. Перечень основной и дополнительной учебной литературы

Основная литература

1. Зыков, С. В. Программирование. Функциональный подход: учебник и практикум для вузов / С. В. Зыков. - 2-е изд. - Москва: Юрайт, 2026. - 150 с - 978-5-534-16942-3. - Текст: электронный // ИКО Юрайт: [сайт]. - URL: https://urait.ru/bcode/584399 (дата обращения: 21.05.2026). - Режим доступа: по подписке

2. Черпаков, И. В. Алгоритмизация и программирование на Python: учебник для вузов / И. В. Черпаков. - Москва: Юрайт, 2026. - 159 с - 978-5-534-21910-4. - Текст: электронный // ИКО Юрайт: [сайт]. - URL: https://urait.ru/bcode/582412 (дата обращения: 21.05.2026). - Режим доступа: по подписке

Дополнительная литература

1. Крупский, В. Н. Теория алгоритмов. Введение в сложность вычислений: учебник для вузов / В. Н. Крупский. - 2-е изд. - Москва: Юрайт, 2026. - 91 с - 978-5-534-21288-4. - Текст: электронный // ИКО Юрайт: [сайт]. - URL: https://urait.ru/bcode/585824 (дата обращения: 21.05.2026). - Режим доступа: по подписке

8.2. Профессиональные базы данных и ресурсы «Интернет», к которым обеспечивается доступ обучающихся

Профессиональные базы данных

1. http://www.gov.ru/ - Профессиональная база данных «Информационные системы Министерства экономического развития Российской Федерации в сети Интернет» (Портал «Официальная Россия»

2. https://ac.hse.ru/ - Аналитический центр Национального исследовательского университета «Высшая школа экономики» (НИУ ВШЭ)

Ресурсы «Интернет»

1. https://lms2.sseu.ru - БРСО

8.3. Программное обеспечение и информационно-справочные системы, используемые при осуществлении образовательного процесса по дисциплине

Перечень программного обеспечения
(обновление производится по мере появления новых версий программы)

1. Google Chome;
2. Python версии 3.14.4 ;
3. "Astra Linux Special Edition" РУСБ.10015-01;

Перечень информационно-справочных систем
(обновление выполняется еженедельно)

1. Справочно-правовая система "Гарант-Максимум";
2. КонсультантПлюс;

8.4. Специальные помещения, лаборатории и лабораторное оборудование

Учебные аудитории для проведения занятий лекционного типа	Комплекты ученической мебели Мультимедийный проектор Доска Экран
Учебные аудитории для проведения практических занятий (занятий семинарского типа)	Комплекты ученической мебели Мультимедийный проектор Доска Экран Компьютеры с выходом в сеть «Интернет» и ЭИОС СП
Учебные аудитории для групповых и индивидуальных консультаций	Комплекты ученической мебели Мультимедийный проектор Доска Экран Компьютеры с выходом в сеть «Интернет» и ЭИОС СП
Учебные аудитории для текущего контроля и промежуточной аттестации	Комплекты ученической мебели Мультимедийный проектор Доска Экран Компьютеры с выходом в сеть «Интернет» и ЭИОС СП
Помещения для самостоятельной работы	Комплекты ученической мебели Мультимедийный проектор Доска Экран Компьютеры с выходом в сеть «Интернет» и ЭИОС СП
Помещения для хранения и профилактического обслуживания оборудования	Комплекты специализированной мебели для хранения