МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Факультет компьютерных технологий и прикладной математики

УТВЕРЖДАЮ:
Проректор по учебной работе,
качеству образования – первый
проректор
_____Хагуров Т.А.
подпись
« 29 » августа 2025 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ (МОДУЛЯ)

Б1. О.19 Алгоритмы и структуры данных

Направление подготовки 02.03.01 Математика и компьютерные науки

Профиль Искусственный интеллект и аналитика данных

Форма обучения очная

Квалификация бакалавр

Краснодар 2025

Рабочая программа дисциплины «Алгоритмы и структуры данных» составлена в соответствии с федеральным государственным образовательным стандартом высшего образования (ФГОС ВО) по направлению подготовки 02.03.01 Математика и компьютерные науки.

Программу составил(и):
Добровольская Н.Ю. доцент, канд. пед. наук, доцент _____
подпись

Рабочая программа обсуждена на заседании кафедры информационных технологий протокол № 1 «26» августа 2025г.

Заведующий кафедрой
Подколзин В.В. _____
подпись

Утверждена на заседании учебно-методической комиссии факультета компьютерных технологий и прикладной математики
протокол № 1 «28» августа 2025г.

Председатель УМК факультета Коваленко А.В. _____
подпись

Рецензенты:
Мостовой Евгений Викторович, генеральный директор ООО «Портал-Юг», e-mail: mostovoy@portal-yug.ru

Луценко Евгений Вениаминович, доктор экономических наук, кандидат технических наук, профессор кафедры компьютерных технологий и систем Федерального государственного бюджетное образовательное учреждение высшего образования «Кубанский государственный аграрный университет имени И.Т. Трубилина», e-mail: prof.lutsenko@gmail.com

1 Цели и задачи изучения дисциплины (модуля)

1.1 Цель освоения дисциплины

Изучение базовых алгоритмов и структур данных, их реализация на C++ с акцентом на эффективность и применение в машинном обучении.

1.2 Задачи дисциплины

Освоение структур данных: списки, стеки, очереди, деревья, графы.
Изучение алгоритмов сортировки и поиска.
Применение рекурсии и динамического программирования.
Анализ сложности алгоритмов (O-нотация).

1.3 Место дисциплины (модуля) в структуре образовательной программы

Дисциплина «Алгоритмы и структуры данных» относится к «Обязательная часть» Блока 1 «Дисциплины (модули)» учебного плана.

1.4 Профессиональные роли в структуре образовательной программы

Роль 1: Data Analyst (Аналитик данных)
Задачи:
1. Статистический анализ, визуализация данных, предварительная обработка.
2. Создание прогнозных моделей
3. Построение аналитических моделей для поддержки бизнес-решений.
Роль 2: MLOps (Специалист по эксплуатации ИИ)
Задачи:
1. DevOps для ML.
2. Автоматизация, мониторинг ML-систем.
3. Операционное управление жизненным циклом ML-моделей.
Роль 3: AI PM (Менеджер проектов ИИ)
Задачи:
1. Управление ИИ-проектами от идеи до внедрения
2. Анализ бизнес-требований и постановка задач
3. Оценка эффективности и ROI ИИ-решений

1.5 Перечень планируемых результатов обучения по дисциплине (модулю), соотнесенных с планируемыми результатами освоения образовательной программы

Изучение данной учебной дисциплины направлено на формирование у обучающихся следующих компетенций:

ОПК-2	**Способен проводить под научным руководством исследование на основе существующих методов в конкретной области профессиональной деятельности**
ОПК-2.1	***Применяет современные математические и вычислительные методы для решения научных задач в рамках поставленной проблемы***
Знать	Знать современный математический аппарат и численные методы, необходимые для формализации, анализа и алгоритмического решения типовых задач в профессиональной научной области.

Уметь Уметь выбирать и корректно применять соответствующие математические модели и вычислительные алгоритмы для получения численного решения поставленной научной или инженерной проблемы.

Владеть Владеть навыками реализации вычислительных методов на практике с использованием специализированного программного обеспечения или языков программирования, а также анализа достоверности и точности полученных результатов.

ОПК-2.2 ***Формулирует гипотезы, планирует и выполняет эксперименты, обрабатывает и интерпретирует полученные данные с использованием специализированного ПО***

Знать Знать методологию научного эксперимента: принципы формулирования проверяемых гипотез, планирования эксперимента, сбора данных, их статистической обработки и интерпретации результатов с использованием соответствующего программного обеспечения.

Уметь Уметь формулировать гипотезы относительно поведения алгоритмов или систем, планировать серии вычислительных экспериментов для их проверки, автоматизировать сбор метрик (время выполнения, потребление памяти) и анализировать полученные данные.

Владеть Владеть навыками работы со специализированным ПО (среды разработки, профайлеры, системы визуализации данных, статистические пакеты) для проведения экспериментов, обработки эмпирических данных и представления результатов в виде графиков и выводов.

ОПК-4 **Способен находить, анализировать, реализовывать программно и использовать на практике математические алгоритмы, в том числе с применением современных вычислительных систем**

ОПК-4.1 ***Разрабатывает и оптимизирует алгоритмы с учетом вычислительной сложности и аппаратных ограничений***

Знать Знать критерии оценки алгоритмов (временная и пространственная сложность, амортизационный анализ), принципы их оптимизации (жадные алгоритмы, динамическое программирование, эвристики), а также влияние архитектуры вычислительных систем (иерархия памяти, параллелизм) на производительность.

Уметь Уметь анализировать задачу, разрабатывать и сравнивать алгоритмические решения с разной вычислительной сложностью, модифицировать и оптимизировать алгоритмы с учетом выявленных «узких мест» и ограничений целевой аппаратной платформы.

Владеть Владеть практическими навыками реализации эффективных алгоритмов, их профилирования для поиска критических участков кода, а также применения техник оптимизации (кэширование, оптимизация циклов, выбор структур данных) для повышения производительности в заданных аппаратных условиях.

ОПК-5 Способен понимать принципы работы современных информационных технологий и использовать их для решения задач профессиональной деятельности

ОПК-5.1 Применяет современные языки программирования и технологии для решения математических и вычислительных задач

Знать Знать синтаксис, парадигмы, основные библиотеки и возможности современных языков программирования, а также технологии, используемые для решения математических и вычислительных задач (символьные вычисления, численные методы, параллельные вычисления)

Уметь Уметь выбирать и применять подходящий язык программирования, библиотеки численных методов и вычислительные технологии для формализации математической задачи, реализации алгоритма её решения, верификации и визуализации результатов

Владеть Владеть навыками практического использования выбранных языков программирования и их специализированных библиотек для реализации, отладки и анализа эффективности алгоритмов решения типовых математических и вычислительных задач

ОПК-6 **Способен разрабатывать алгоритмы и компьютерные программы, пригодные для практического применения**

ОПК-6.1 ***Разрабатывает эффективные алгоритмы и реализует их в виде программного кода с учетом временной и пространственной сложности***

Знать Знать методологию разработки и анализа алгоритмов, принципы оценки их временной (О-нотация) и пространственной сложности, а также основные паттерны проектирования эффективных алгоритмов (жадные алгоритмы, «разделяй и властвуй», динамическое программирование)

Уметь Уметь формулировать задачу в терминах входных/выходных данных, проектировать эффективный алгоритм её решения, анализировать его сложность и транслировать алгоритмическое решение в корректный, читаемый и эффективный программный код.

Владеть Владеть навыками практической реализации, отладки и эмпирической проверки разработанных алгоритмов, уметь проводить профилирование кода для оценки его реальной производительности и оптимизировать решения с учетом компромисса между временем выполнения и использованием памяти.

ОПК-6.2 ***Применяет методы структурного и объектно-ориентированного программирования, создает модульные и масштабируемые программы***

Знать Знать фундаментальные принципы, методологии и синтаксические конструкции структурного (процедурного) и объектно-ориентированного программирования, а также критерии модульности, слабой связанности и высокой связности кода, обеспечивающие его масштабируемость.

Уметь Уметь анализировать предметную область задачи и применять адекватные парадигмы (процедурную декомпозицию или объектное моделирование) для

проектирования архитектуры программного решения, состоящего из модулей с четкими интерфейсами.

Владеть Владеть навыками реализации, интеграции и рефакторинга программных модулей, разработанных с применением принципов структурного и объектно-ориентированного программирования, обеспечивая возможность их повторного использования и относительно легкого расширения функциональности.

PL-3 **Способен применять языки программирования C/C++ для решения задач в области ИИ**

PL-3.1

"Осуществляет выбор инструментов разработки на языке C/C++, приемлемых для создания прикладной системы ИИ с заданными требованиями

Знать Базовый синтаксис и структуры программ на C/C++ (типы данных, операторы, функции, массивы).

Основные этапы разработки: написание, компиляция, отладка в интегрированной среде (IDE).

Уметь Реализовывать простые алгоритмы (вычисления, сортировки, обработка данных) на C/C++.

Использовать IDE (например, Visual Studio, Qt Creator) для сборки, запуска и отладки учебных проектов.

Владеть Навыками написания и отладки консольных программ на C/C++ для решения типовых учебных задач.

Умением выбрать и настроить среду разработки для создания простых приложений.

2. Структура и содержание дисциплины

2.1 Распределение трудоёмкости дисциплины по видам работ

Общая трудоёмкость дисциплины составляет 3 зач. ед. (108 часов), их распределение по видам работ представлено в таблице

Вид учебной работы	Всего часов	Семестры (часы)					
		2					
Контактная работа, в том числе:	**72,2**	**72,2**					
Аудиторные занятия (всего):	**68**	**68**					
Занятия лекционного типа	**34**	34					
Лабораторные занятия	**34**	34					
Занятия семинарского типа (семинары, практические занятия)							
Иная контактная работа:	**4,2**	**4,2**					
Контроль самостоятельной работы (КСР)	**4**	4					
Промежуточная аттестация (ИКР)	**0,2**	0,2					
Самостоятельная работа, в том числе:	**35,8**	**35,8**					

Курсовая работа								
Проработка учебного (теоретического) материала		**15,8**	15,8					
Выполнение индивидуальных заданий (подготовка сообщений, презентаций)		**20**	20					
Реферат								
Подготовка к текущему контролю								
Контроль:								
Подготовка к экзамену								
Общая трудоемкость	**час.**	**108**	**108**					
	в том числе контактная работа	**72,2**	**72,2**					
	зач. ед	**3**	**3**					

2.2 Структура дисциплины

Распределение видов учебной работы и их трудоемкости по разделам дисциплины.

Разделы (темы) дисциплины, изучаемые в 2 семестре

№	Наименование разделов (тем)	Количество часов				
		Всего	Аудиторная работа			Внеаудиторная работа
			Л	ПЗ	ЛР	СРС
1	2	3	4	5	6	7
1.	Анализ сложности алгоритмов.	12	4		4	4
2.	Рекурсия. Деревья (BST, обходы).	12	4		4	4
3.	Алгоритмы сортировки.	12	4		4	4
4.	Работа с файлами	12	4		4	4
5.	Хеш-таблицы, обработка коллизий.	12	4		4	4
6.	Графы	16	6		6	4
7.	Основы ООП: классы и объекты.	14	4		4	4
8.	Контейнерные классы	7,8	2		2	3,8
9.	Базовые алгоритмы машинного обучения	8	2		2	4
ИТОГО по разделам дисциплины		**103,8**	**34**		**34**	**35,8**
Контроль самостоятельной работы (КСР)		2				
Промежуточная аттестация (ИКР)		0,2				
Подготовка к текущему контролю						
Общая трудоемкость по дисциплине		**108**				

Примечание: Л – лекции, ПЗ – практические занятия/семинары, ЛР – лабораторные занятия, СРС – самостоятельная работа студента

2.3 Содержание разделов (тем) дисциплины

2.3.1 Занятия лекционного типа

№	Наименование раздела (темы)	Содержание раздела (темы)	Форма текущего контроля
1	2	3	4
1.	Анализ сложности алгоритмов.	Меры сложности. Анализ алгоритмов.	ЛР

№	Наименование раздела (темы)	Содержание раздела (темы)	Форма текущего контроля
1	2	3	4
2.	Рекурсия. Деревья (BST, обходы).	Рекурсивные функции. Деревья. Основные понятия. Дерево двоичного поиска (структура и построение). Обходы ДДП. Обработка дерева двоичного поиска (поиск и замена элемента, вычисление суммы или количества заданных элементов).	ЛР, РЗ
3.	Алгоритмы сортировки.	Сортировка вставками, обменом, подсчетом, челночная. Быстрая сортировкаю	ЛР, РЗ
4.	Работа с файлами	Общие операции над файлами. Обработка текстовых, числовых, бинарных файлов. Работа с файлами (fstream). Обработка ошибок при работе с файлами. Сериализация данных	ЛР, РЗ
5.	Хеш-таблицы, обработка коллизий.	Общая схема компиляции. Информационные таблицы. Сканер. Лексический анализатор. Синтаксический и семантический анализатор. Функция хеширования. Метод цепочек переполнения	ЛР, РЗ
6.	Графы	Ориентированные графы. Основные определения. Представления ориентированных графов. АТД для ориентированных графов. Основные операторы. Задача нахождения кратчайшего пути. Алгоритм Дейкстры. Реализация на C++. Алгоритм нахождения сильно связных компонент. Остовные деревья минимальной стоимости Алгоритм Прима. Алгоритм Крускала.	ЛР, РЗ
7.	Основы ООП: классы и объекты.	Классы, объекты, инкапсуляция. Конструкторы, деструкторы. Статические члены класса. Перегрузка операторов. Дружественные функции и классы	ЛР, РЗ
8.	Контейнерные классы	Контейнерный класс vector. Основные методы, примеры использования. Контейнерный класс list. Основные методы, примеры использования. Ассоциативные контейнеры set и map.	ЛР, РЗ
9.	Базовые алгоритмы машинного обучения	Простая линейная регрессия на C++	ЛР

Примечание: ЛР – отчет/защита лабораторной работы, КП - выполнение курсового проекта, КР - курсовой работы, РГЗ - расчетно-графического задания, Р - написание реферата, Э - эссе, К - коллоквиум, Т – тестирование, РЗ – решение задач.

2.3.2 Занятия семинарского типа

Не предусмотрены

Примечание: ЛР – отчет/защита лабораторной работы, КП - выполнение курсового проекта, КР - курсовой работы, РГЗ - расчетно-графического задания, Р - написание реферата, Э - эссе, К - коллоквиум, Т – тестирование, РЗ – решение задач.

2.3.3 Лабораторные занятия

№	Наименование раздела (темы)	Наименование лабораторных работ	Форма текущего контроля
1	2	3	4
1	Анализ сложности алгоритмов.	Экспериментальный анализ времени работы алгоритмов	ЛР
2	Рекурсия. Деревья (BST, обходы).	Рекурсивные алгоритмы и бинарные деревья	ЛР
3	Алгоритмы сортировки.	Сравнение алгоритмов сортировки	ЛР
4	Работа с файлами	Обработка текстовых и бинарных файлов	ЛР
5	Хеш-таблицы, обработка коллизий.	Реализация хеш-таблицы	ЛР
6	Графы	Алгоритмы на графах	ЛР
7	Основы ООП: классы и объекты.	Разработка класса для математических операций	ЛР
8	Контейнерные классы	Собственная реализация vector	ЛР
9.	Базовые алгоритмы машинного обучения	Простая линейная регрессия на C++	ЛР

Примечание: ЛР – отчет/защита лабораторной работы, КП - выполнение курсового проекта, КР - курсовой работы, РГЗ - расчетно-графического задания, Р - написание реферата, Э - эссе, К - коллоквиум, Т – тестирование, РЗ – решение задач.

2.3.4 Примерная тематика курсовых работ (проектов)

Не предусмотрены

2.4 Перечень учебно-методического обеспечения для самостоятельной работы обучающихся по дисциплине (модулю)

№	Вид СРС	Перечень учебно-методического обеспечения дисциплины по выполнению самостоятельной работы
1	2	3
1	Изучение теоретического материала	Методические указания по организации самостоятельной работы студентов, утвержденные кафедрой информационных технологий, протокол №1 от 30.08.2019
2	Решение задач	Методические указания по организации самостоятельной работы студентов, утвержденные кафедрой информационных технологий, протокол №1 от 30.08.2019

Учебно-методические материалы для самостоятельной работы обучающихся из числа инвалидов и лиц с ограниченными возможностями здоровья (ОВЗ) предоставляются в формах, адаптированных к ограничениям их здоровья и восприятия информации:

Для лиц с нарушениями зрения:

– в печатной форме увеличенным шрифтом,

– в форме электронного документа,

– в форме аудиофайла,

– в печатной форме на языке Брайля.

Для лиц с нарушениями слуха:

– в печатной форме,

– в форме электронного документа.

Для лиц с нарушениями опорно-двигательного аппарата:

– в печатной форме,

– в форме электронного документа,

– в форме аудиофайла.

Данный перечень может быть конкретизирован в зависимости от контингента обучающихся.

3. Образовательные технологии

В соответствии с требованиями ФГОС в программа дисциплины предусматривает использование в учебном процессе следующих образовательные технологии: чтение лекций с использованием мультимедийных технологий; метод малых групп, разбор практических задач и кейсов.

При обучении используются следующие образовательные технологии:

– Технология коммуникативного обучения – направлена на формирование коммуникативной компетентности студентов, которая является базовой, необходимой для адаптации к современным условиям межкультурной коммуникации.

– Технология разноуровневого (дифференцированного) обучения – предполагает осуществление познавательной деятельности студентов с учётом их индивидуальных способностей, возможностей и интересов, поощряя их реализовывать свой творческий потенциал. Создание и использование диагностических тестов является неотъемлемой частью данной технологии.

– Технология модульного обучения – предусматривает деление содержания дисциплины на достаточно автономные разделы (модули), интегрированные в общий курс.

– Информационно-коммуникационные технологии (ИКТ) - расширяют рамки образовательного процесса, повышая его практическую направленность, способствуют интенсификации самостоятельной работы учащихся и повышению познавательной активности. В рамках ИКТ выделяются 2 вида технологий:

– Технология использования компьютерных программ – позволяет эффективно дополнить процесс обучения языку на всех уровнях.

– Интернет-технологии – предоставляют широкие возможности для поиска информации, разработки научных проектов, ведения научных исследований.

– Технология индивидуализации обучения – помогает реализовывать личностно-ориентированный подход, учитывая индивидуальные особенности и потребности учащихся.

– Проектная технология – ориентирована на моделирование социального взаимодействия учащихся с целью решения задачи, которая определяется в рамках профессиональной подготовки, выделяя ту или иную предметную область.

– Технология обучения в сотрудничестве – реализует идею взаимного обучения, осуществляя как индивидуальную, так и коллективную ответственность за решение учебных задач.

– Игровая технология – позволяет развивать навыки рассмотрения ряда возможных способов решения проблем, активизируя мышление студентов и раскрывая личностный потенциал каждого учащегося.

– Технология развития критического мышления – способствует формированию разносторонней личности, способной критически относиться к информации, умению отбирать информацию для решения поставленной задачи.

Комплексное использование в учебном процессе всех вышеназванных технологий стимулируют личностную, интеллектуальную активность, развивают познавательные процессы, способствуют формированию компетенций, которыми должен обладать будущий специалист.

Основные виды интерактивных образовательных технологий включают в себя:

– работа в малых группах (команде) - совместная деятельность студентов в группе под руководством лидера, направленная на решение общей задачи путём творческого сложения результатов индивидуальной работы членов команды с делением полномочий и ответственности;

– проектная технология - индивидуальная или коллективная деятельность по отбору, распределению и систематизации материала по определенной теме, в результате которой составляется проект;

– анализ конкретных ситуаций - анализ реальных проблемных ситуаций, имевших место в соответствующей области профессиональной деятельности, и поиск вариантов лучших решений;

– развитие критического мышления – образовательная деятельность, направленная на развитие у студентов разумного, рефлексивного мышления, способного выдвинуть новые идеи и увидеть новые возможности.

Подход разбора конкретных задач и ситуаций широко используется как преподавателем, так и студентами во время лекций, лабораторных занятий и анализа результатов самостоятельной работы. Это обусловлено тем, что при исследовании и решении каждой конкретной задачи имеется, как правило, несколько методов, а это требует разбора и оценки целой совокупности конкретных ситуаций.

Семестр	Вид занятия	Используемые интерактивные образовательные технологии	количество интерактивных часов
	ЛР	Практические занятия в режимах взаимодействия «преподаватель – студент» и «студент – студент»	26
Итого			**26**

Примечание: Л – лекции, ПЗ – практические занятия/семинары, ЛР – лабораторные занятия, СРС – самостоятельная работа студента

Темы, задания и вопросы для самостоятельной работы призваны сформировать навыки поиска информации, умения самостоятельно расширять и углублять знания, полученные в ходе лекционных и практических занятий.

Подход разбора конкретных ситуаций широко используется как преподавателем, так и студентами при проведении анализа результатов самостоятельной работы.

Для лиц с ограниченными возможностями здоровья предусмотрена организация консультаций с использованием электронной почты.

Для лиц с нарушениями зрения:

– в печатной форме увеличенным шрифтом,

– в форме электронного документа.

Для лиц с нарушениями слуха:

– в печатной форме,

– в форме электронного документа.

Для лиц с нарушениями опорно-двигательного аппарата:

– в печатной форме,

– в форме электронного документа.

Для лиц с ограниченными возможностями здоровья предусмотрена организация консультаций с использованием электронной почты.

Данный перечень может быть конкретизирован в зависимости от контингента обучающихся.

4. Оценочные и методические материалы

4.1 Оценочные средства для текущего контроля успеваемости и промежуточной аттестации

Оценочные средства предназначены для контроля и оценки образовательных достижений обучающихся, освоивших программу учебной дисциплины «Алгоритмы и структуры данных».

Оценочные средства включает контрольные материалы для проведения **текущего контроля** в форме заданий по темам, контрольным работам, и **промежуточной аттестации** в форме вопросов и заданий **к зачету**.

Оценочные средства для инвалидов и лиц с ограниченными возможностями здоровья выбираются с учетом их индивидуальных психофизических особенностей.

– при необходимости инвалидам и лицам с ограниченными возможностями здоровья предоставляется дополнительное время для подготовки ответа на зачете;

– при проведении процедуры оценивания результатов обучения инвалидов и лиц с ограниченными возможностями здоровья предусматривается использование технических средств, необходимых им в связи с их индивидуальными особенностями;

– при необходимости для обучающихся с ограниченными возможностями здоровья и инвалидов процедура оценивания результатов обучения по дисциплине может проводиться в несколько этапов.

Процедура оценивания результатов обучения инвалидов и лиц с ограниченными возможностями здоровья по дисциплине (модулю) предусматривает предоставление информации в формах, адаптированных к ограничениям их здоровья и восприятия информации:

Для лиц с нарушениями зрения:

– в печатной форме увеличенным шрифтом,

– в форме электронного документа.

Для лиц с нарушениями слуха:

– в печатной форме,

– в форме электронного документа.

Для лиц с нарушениями опорно-двигательного аппарата:

– в печатной форме,

– в форме электронного документа.

Данный перечень может быть конкретизирован в зависимости от контингента обучающихся.

Структура оценочных средств для текущей и промежуточной аттестации

№ п/п	Контролируемые разделы (темы) дисциплины*	Код контролируемой компетенции (или ее части)	Наименование оценочного средства	
			Текущий контроль	Промежуточная аттестация
1	Анализ сложности алгоритмов.	ОПК-2.1; ОПК-2.2; ОПК-4.1; ОПК-5.1; ОПК-6.1; ОПК-6.2; PL-3.1	Типовые контрольные вопросы 1-4 Контрольное задание 1	Отчет по ЛР№1
2	Рекурсия. Деревья (BST, обходы).	ОПК-2.1; ОПК-2.2; ОПК-4.1; ОПК-5.1; ОПК-6.1; ОПК-6.2; PL-3.1	Типовые контрольные вопросы 5-8	Отчет по ЛР№2

			Контрольное задание 2	
3	Алгоритмы сортировки.	ОПК-2.1; ОПК-2.2; ОПК-4.1; ОПК-5.1; ОПК-6.1; ОПК-6.2; PL-3.1	Типовые контрольные вопросы 9-12 Контрольное задание 3	Отчет по ЛР№3
4	Работа с файлами	ОПК-2.1; ОПК-2.2; ОПК-4.1; ОПК-5.1; ОПК-6.1; ОПК-6.2; PL-3.1	Типовые контрольные вопросы 13-16 Контрольное задание 4	Отчет по ЛР№4
5	Хеш-таблицы, обработка коллизий.	ОПК-2.1; ОПК-2.2; ОПК-4.1; ОПК-5.1; ОПК-6.1; ОПК-6.2; PL-3.1	Типовые контрольные вопросы 17-20 Контрольное задание 5	Отчет по ЛР№5
6	Графы	ОПК-2.1; ОПК-2.2; ОПК-4.1; ОПК-5.1; ОПК-6.1; ОПК-6.2; PL-3.1	Типовые контрольные вопросы 21-24 Контрольное задание 6	Отчет по ЛР№6
7	Основы ООП: классы и объекты.	ОПК-2.1; ОПК-2.2; ОПК-4.1; ОПК-5.1; ОПК-6.1; ОПК-6.2; PL-3.1	Типовые контрольные задания 25-28 Контрольное задание 7	Отчет по ЛР№7
8	Контейнерные классы	ОПК-2.1; ОПК-2.2; ОПК-4.1; ОПК-5.1; ОПК-6.1; ОПК-6.2; PL-3.1	Типовые контрольные задания 29-32 Контрольное задание 8	Отчет по ЛР№8
9	Базовые алгоритмы машинного обучения	ОПК-2.1; ОПК-2.2; ОПК-4.1; ОПК-5.1; ОПК-6.1; ОПК-6.2; PL-3.1	Контрольное задание 9	Отчет по ЛР№9

Показатели, критерии и шкала оценки сформированных компетенций

Соответствие **пороговому уровню** освоения компетенций планируемым результатам обучения и критериям их оценивания (оценка: **зачтено**):

ОПК-2 **Способен использовать и адаптировать существующие математические методы и системы программирования для разработки и реализации алгоритмов решения прикладных задач**

ОПК-2.1 ***Способен применять системный подход к анализу предметной (проблемной) области, выявлению требований к реализации алгоритмов решения прикладных задач***

Знать Знать математические основы для анализа алгоритмов: нотации асимптотической сложности (О-символика), методы суммирования, рекуррентные соотношения, а также математические модели базовых структур данных (деревья, графы, хэш-функции).

Уметь Уметь применять математический аппарат для обоснованного выбора алгоритма или структуры данных под конкретную задачу (например, оценки

времени поиска в сбалансированном дереве vs хэш-таблице) и для анализа эффективности реализованного решения.

Владеть Владеть навыками практического расчёта вычислительной сложности алгоритмов (в лучшем, среднем и худшем случаях), построения и решения простых рекуррентных соотношений для анализа рекурсивных алгоритмов и применения вероятностных моделей для оценки работы алгоритмов (например, хэширования)

ОПК-2.2 ***Применяет современный математический аппарат при построении моделей в различных областях человеческой деятельности***

Знать Знать базовые принципы эмпирического анализа алгоритмов: как формулировать гипотезы об их асимптотической или практической эффективности, какие метрики измерять (время, число операций, память) и как планировать эксперимент

Уметь Уметь сформулировать проверяемую гипотезу (например, «Сортировка слиянием на массиве из N элементов выполняется за время, пропорциональное N log N»), спланировать и провести серию замеров времени выполнения для разных N, исключив случайные помехи.

Владеть Владеть навыками использования инструментов для эксперимента: писать тестовые скрипты на языке программирования, использовать системные таймеры или профайлеры для замера времени, обрабатывать сырые данные (усреднение, отсев выбросов) и строить графики зависимости времени от размера входных данных в специализированном ПО

ОПК-4 **Способен участвовать в разработке технической документации программных продуктов и программных комплексов**

ОПК-4.1 ***Обладает знаниями об основных стандартах, нормах и правил разработки технической документации программных продуктов и программных комплексов***

Знать Знать базовые нотации асимптотической сложности (O, Θ, Ω), сравнительную сложность основных алгоритмов сортировки, поиска и обхода графов, а также понимать, как выбор структуры данных (массив, связный список, хеш-таблица) влияет на эффективность операций.

Уметь Уметь для типовой задачи (поиск, сортировка, обход) разрабатывать или выбирать из изученных несколько алгоритмов, сравнивать их теоретическую сложность и обосновывать выбор оптимального с учетом предполагаемого размера входных данных.

Владеть Владеть навыками реализации и эмпирического сравнения алгоритмов с разной асимптотикой (например, сортировки пузырьком $O(n^2)$ и быстрой сортировки $O(n \log n)$), а также простейшей оптимизации кода (например, минимизация вложенных циклов, выбор между массивом и списком) на основе анализа его вычислительной сложности.

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

ОПК-5.1 Демонстрирует знания системного администрирования, администрирования СУБД, технологий информационного взаимодействия программных систем

Знать Знать синтаксис, парадигмы, основные библиотеки и возможности современных языков программирования, а также технологии, используемые для решения математических и вычислительных задач (символьные вычисления, численные методы, параллельные вычисления)

Уметь Уметь выбирать и применять подходящий язык программирования, библиотеки численных методов и вычислительные технологии для формализации математической задачи, реализации алгоритма её решения, верификации и визуализации результатов

Владеть Владеть практическими навыками написания, отладки и тестирования программ, реализующих основные алгоритмы и структуры данных на изучаемом языке, а также использования встроенных структур данных (списки, словари, множества) в качестве основы для реализации более сложных абстракций (очередь, стек, граф).

ОПК-6 **Способен разрабатывать алгоритмы и программы, пригодные для практического использования, применять основы информатики и программирования к проектированию, конструированию и тестированию программных продуктов**

ОПК-6.1 ***Разрабатывает алгоритмы для решения задач в области искусственного интеллекта и анализа данных, используя языки программирования***

Знать Знать основные классы сложности алгоритмов (константная, логарифмическая, линейная, квадратичная) и уметь определять их для изученных базовых алгоритмов сортировки, поиска и обхода.

Уметь Уметь для поставленной задачи (например, поиск элемента, сортировка набора данных) разрабатывать или выбирать подходящий алгоритм, оценивать его временную и пространственную сложность и корректно реализовывать его в виде программной функции на изучаемом языке.

Владеть Владеть навыком написания программного кода, реализующего изученные алгоритмы (например, быстрая сортировка, бинарный поиск, поиск в ширину), с осознанным выбором структур данных и управляющих конструкций для достижения заявленной асимптотической сложности.

ОПК-6.2 ***Применяет принципы тестирования и отладки программного кода, обеспечивая его работоспособность и эффективность***

Знать Знать базовые принципы структурного программирования (последовательность, ветвление, цикл, декомпозиция на функции) и начальные концепции ООП (класс, объект, инкапсуляция) применительно к описанию структур данных (например, класс «Стек» или «Очередь»)..

Уметь Уметь применять метод процедурной декомпозиции для реализации сложного алгоритма, разбивая его на отдельные функции (например, функции разделения и слияния для сортировки), а также использовать

	простые классы для инкапсуляции данных и методов работы с ними (например, класс «Граф»).
Владеть	Владеть навыками создания модульных программ, где основные алгоритмы (сортировка, поиск) и структуры данных (списки, деревья) оформлены в виде отдельных функций или классов, что позволяет повторно использовать их в различных учебных задачах и масштабировать решение..
PL-3 *PL-3.1*	**Способен применять языки программирования C/C++ для решения задач в области ИИ** *"Осуществляет выбор инструментов разработки на языке C/C++, приемлемых для создания прикладной системы ИИ с заданными требованиями*
Знать	Основные структуры данных (стек, очередь, список) и базовые алгоритмы обработки данных (поиск, сортировка, обход графа).
Уметь	Проектировать и реализовывать на C++ классы для представления сущностей предметной области (например, "нейрон", "слой", "обучающий набор данных"). Использовать IDE (например, Visual Studio, Qt Creator) для сборки, запуска и отладки учебных проектов.
Владеть	Навыками модульного программирования на C++: создание многофайловых проектов

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

Типовые контрольные вопросы

1. Что такое асимптотическая сложность алгоритма? Как она обозначается?
2. В чем разница между O(1), O(n) и O(n^2)? Приведите примеры алгоритмов для каждого случая.
3. Как оценить сложность алгоритма с вложенными циклами?
4. Что означает Ω- и Θ-нотация? Чем они отличаются от O-нотации?
5. Каковы обязательные свойства бинарного дерева поиска (BST)?
6. В чем разница между прямым (pre-order), симметричным (in-order) и обратным (post-order) обходом дерева?
7. Как работает рекурсивный алгоритм обхода дерева? Когда возможен стековый переполнение?
8. Как сбалансировать BST? Какие методы вы знаете?
9. В чем разница между устойчивой и неустойчивой сортировкой? Какие алгоритмы относятся к каждому типу?
10. Почему QuickSort в среднем работает быстрее MergeSort?
11. Как работает сортировка подсчетом? В каких случаях она эффективна?
12. Какая сложность у сортировки пузырьком в лучшем, среднем и худшем случаях?
13. Чем отличаются текстовые и бинарные файлы? Когда какой тип следует использовать?

14. Как безопасно обрабатывать ошибки при работе с файлами в C++?
15. Какие методы сериализации данных вы знаете? Как сохранить структуру в файл?
16. Как организовать поиск данных в большом файле без полной загрузки в память?
17. Как работает хеш-функция? Какими свойствами она должна обладать?
18. В чем разница между методом цепочек и открытой адресации?
19. Как решается проблема кластеризации в хеш-таблицах?
20. Почему при большом коэффициенте заполнения хеш-таблица начинает работать медленно?
21. Какие способы представления графов в памяти вы знаете? Плюсы и минусы каждого.
22. В чем разница между DFS и BFS? Когда какой алгоритм предпочтительнее?
23. Как найти кратчайший путь в невзвешенном графе?
24. Какие алгоритмы используются для поиска минимального остовного дерева?
25. В чем разница между классом и структурой в C++?
26. Что такое конструктор копирования? Когда он вызывается?
27. Какие принципы ООП вы знаете? Приведите примеры каждого.
28. Для чего нужны виртуальные функции? Как реализован полиморфизм в C++?
29. В чем разница между vector, list и deque? Когда какой контейнер выбрать?
30. Как работает std::map? Какая у него сложность операций?
31. Какие итераторы существуют в STL? Чем они отличаются?
32. Как эффективно удалить дубликаты из контейнера?

Типовые контрольные задания

1. Анализ сложности алгоритмов

Задание 1:

Напишите функцию, которая определяет, содержит ли массив дубликаты. Оцените сложность вашего алгоритма в О-нотации.

Задание 2:

Реализуйте алгоритм поиска элемента в отсортированном массиве двумя способами: линейным поиском и бинарным поиском. Сравните их производительность на массиве из 10 000 элементов.

2. Рекурсия. Деревья (BST, обходы)

Задание 1:

Напишите рекурсивную функцию для вычисления факториала числа. Затем перепишите её в итеративном варианте.

Задание 2:

Реализуйте бинарное дерево поиска (BST) с операциями вставки и поиска элемента. Добавьте функцию для вывода дерева в порядке in-order.

3. Алгоритмы сортировки

Задание 1:

Реализуйте сортировку пузырьком. Измерьте время её работы на массивах разного размера (100, 1000, 10 000 элементов).

Задание 2:

Напишите функцию для сортировки массива методом выбора. Сравните её производительность с сортировкой пузырьком.

4. Работа с файлами
Задание 1:
Создайте программу, которая записывает в текстовый файл 100 случайных чисел, а затем читает их и выводит на экран.

Задание 2:
Напишите программу, которая сохраняет структуру «Студент» (имя, возраст, средний балл) в бинарный файл и затем загружает данные обратно.

5. Хеш-таблицы, обработка коллизий
Задание 1:
Реализуйте простую хеш-таблицу с методом цепочек для хранения строк. Добавьте функции вставки и поиска.

Задание 2:
Напишите хеш-функцию для строк, которая минимизирует коллизии. Протестируйте её на наборе из 1000 случайных строк.

6. Графы
Задание 1:
Реализуйте обход графа в глубину (DFS) для списка смежности. Граф задан как std::vector<std::vector<int>>.

Задание 2:
Напишите функцию, которая проверяет, есть ли в графе цикл, используя поиск в глубину.

7. Основы ООП: классы и объекты
Задание 1:
Создайте класс «Прямоугольник» с методами для вычисления площади и периметра. Добавьте конструктор и деструктор.

Задание 2:
Реализуйте класс «Студент» с полями (имя, возраст, оценки) и методами для добавления оценки и вычисления среднего балла.

8. Контейнерные классы
Задание 1:
Напишите шаблонный класс «Динамический массив» (аналог std::vector) с методами push_back, pop_back и size.

Задание 2:
Реализуйте класс «Очередь» на основе связного списка с операциями enqueue и dequeue.

Типовые задания для самостоятельной работы

1. Дано два однонаправленных списка целых чисел. Сформировать третий список, содержащий простые числа исходных списков.

2. Дан двунаправленный кольцевой список целых чисел. Удалить все положительные элементы, до и после которых следуют элементы с нечетной суммой цифр.

3. Дано дерево двоичного поиска. Найти количество совершенных листьев дерева, больших разности максимального и минимального элемента.

4. Дан файл f, компонентами которого являются целые числа. Переписать в файл g все совершенные числа, индексы которых являются простыми.

5. Найти в графе простой цикл максимальной длины.

6. Даны действительные числа a_1, a_2, ..., a_n. Если в результате замены отрицательных элементов последовательности a_1, a_2, ..., a_n их квадратами элементы будут образовывать неубывающую последовательность, то получить сумму элементов исходной последовательности, в противном случае получить их произведение.

Типовые лабораторные работы

ЛР №1: "Экспериментальный анализ времени работы алгоритмов"

Реализовать линейный и бинарный поиск
Замерить время выполнения для разных размеров массивов
Построить графики зависимости времени от N
Сравнить с теоретической оценкой O(n) и O(log n)

ЛР №2: "Рекурсивные алгоритмы и бинарные деревья"

Реализовать рекурсивное вычисление факториала и чисел Фибоначчи
Построить BST с операциями вставки/поиска
Реализовать обход дерева (in-order, pre-order, post-order)
Визуализировать дерево с отступами

ЛР №3: "Сравнение алгоритмов сортировки"

Реализовать 3 сортировки (пузырьком, выбором, быструю)
Сравнить время работы на разных наборах данных
Проанализировать количество сравнений и перестановок
Определить границы эффективности каждого метода

ЛР №4: "Обработка текстовых и бинарных файлов"

Создать программу для подсчета статистики по текстовому файлу
Реализовать сохранение структур данных в бинарный файл
Организовать поиск записей по критериям
Обработать ошибки чтения/записи

ЛР №5: "Реализация хеш-таблицы"

Создать хеш-таблицу с заданным размером
Реализовать методы цепочек и открытой адресации
Провести тестирование на разных коэффициентах заполнения
Сравнить количество коллизий для разных хеш-функций

ЛР №6: "Алгоритмы на графах"

Реализовать представление графа (матрица смежности/списки)
Написать алгоритмы обхода (DFS и BFS)

Решить задачу поиска кратчайшего пути в невзвешенном графе
Визуализировать граф с помощью ASCII-графики

ЛР №7: "Разработка класса для математических операций"

Создать класс "Дробь" с арифметическими операциями
Реализовать конструкторы, деструктор, методы доступа
Добавить перегрузку операторов (+, -, <<)
Протестировать все методы класса

ЛР №8: "Собственная реализация vector"

Разработать упрощенный аналог std::vector
Реализовать основные методы (push_back, pop_back, resize)
Обеспечить безопасность при работе с памятью
Сравнить производительность с STL-контейнером

ЛР №9 "Простая линейная регрессия на C++"

Написать программу, которая предсказывает значение y по заданному x на основе линейной модели (y = kx + b).
1. Теория
Линейная регрессия – это метод, который находит линию, наилучшим образом описывающую зависимость между двумя переменными:
x – входной признак (например, площадь квартиры).
y – целевое значение (например, цена квартиры).
Формула линии.
2. Варианты реализации
Вариант 1: Вручную (без библиотек)
Можно реализовать метод наименьших квадратов (МНК) для расчёта k и b.
Вариант 2: С библиотекой (например, Eigen для линейной алгебры)
Упростим расчёты, используя готовые матричные операции.
Вариант 1: Ручной расчёт линейной регрессии
Шаги:
Ввод данных
Зададим небольшой датасет (например, x = [1, 2, 3, 4], y = [2, 4, 5, 4]).
Расчёт коэффициентов k и b по формулам.
Предсказание y для нового x
Подставляем x в уравнение y = kx + b.
Реализация кода на C++
Вариант 2: С библиотекой Eigen (упрощённый)
Шаги:
Установите Eigen (заголовочная библиотека, не требует компиляции).
Используйте матричные операции для решения системы уравнений.
Реализация кода на C++
Дополнительные задания для самостоятельной работы
Добавьте расчёт ошибки (MSE)
Реализуйте подсчёт средней квадратичной ошибки между предсказаниями и реальными y.
Ввод данных из файла
Пусть программа считывает x и y из CSV-файла.
Множественная регрессия
Расширьте код для нескольких признаков (например, y = k1*x1 + k2*x2 + b).

Зачетно-экзаменационные материалы для промежуточной аттестации (зачет)

Вопросы для подготовки к зачету

1. Объяснение понятия асимптотической сложности алгоритмов
2. Сравнение временной сложности O(1), O(log n), O(n) и $O(n^2)$
3. Методы оценки сложности рекурсивных алгоритмов
4. Практическое применение Big-O нотации для анализа алгоритмов
5. Рекурсия и древовидные структуры
6. Принципы работы рекурсивных функций в C++
7. Основные характеристики бинарных деревьев поиска
8. Алгоритмы обхода деревьев: прямой, обратный и симметричный
9. Реализация операций вставки и удаления в BST
10. Сравнительный анализ алгоритмов сортировки по времени выполнения
11. Особенности реализации сортировки слиянием
12. Принцип работы быстрой сортировки (QuickSort)
13. Условия применения сортировки подсчетом
14. Различия между текстовым и бинарным режимами работы с файлами
15. Методы обработки ошибок при файловых операциях
16. Организация последовательного доступа к данным в файлах
17. Техники эффективного чтения/записи структур данных
18. Принципы работы хеш-функций и их свойства
19. Способы разрешения коллизий в хеш-таблицах
20. Факторы, влияющие на производительность хеширования
21. Реализация простой хеш-таблицы на C++
22. Основные способы представления графов в памяти
23. Особенности реализации поиска в ширину (BFS)
24. Применение поиска в глубину (DFS) для задач на графах
25. Алгоритмы поиска кратчайшего пути в графах
26. Алгоритм Флойда
27. Алгоритм поиска связных компонент
28. Алгоритм Прима
29. Алгоритм Крускала
30. Алгоритм поиска центра графа
31. Отличия между классами и структурами в C++
32. Механизм наследования и полиморфизма
33. Практическое применение виртуальных функций
34. Сравнительный анализ контейнеров vector, list и deque
35. Особенности работы с ассоциативными контейнерами
36. Принципы использования итераторов в STL
37. Оптимизация работы с контейнерными классами
38. Реализация стека и очереди на основе стандартных контейнеров
39. Методы обработки исключений в C++
40. Организация памяти при работе с динамическими структурами
41. Принципы тестирования и отладки алгоритмов
42. Оптимизация алгоритмов с учетом аппаратных особенностей
43. Документирование и сопровождение алгоритмического кода
44. Анализ производительности реализованных алгоритмов

Типовые задания на зачет

Вариант 1

Теоретическая часть

Вопрос 1:

Что такое бинарное дерево поиска (BST)? Опишите его свойства и основные операции (вставка, удаление, поиск). Какова средняя и худшая сложность этих операций?

Вопрос 2:

Объясните принцип работы алгоритма быстрой сортировки (QuickSort). В чем его преимущества и недостатки? Какая у него асимптотическая сложность в лучшем, среднем и худшем случаях?

Практическая часть

Задача 1:

Реализуйте рекурсивный обход бинарного дерева (in-order, pre-order или post-order на выбор). Дерево задано структурой:

```
struct Node {
   int data;
   Node* left;
   Node* right;
};
```

Задача 2:

Напишите функцию для проверки сбалансированности скобок в строке с использованием стека. Примеры:

Вход: "(a + [b * c])" → true

Вход: "{(a + b) * (c - d)" → false

Вариант 2

Теоретическая часть

Вопрос 1:

Что такое хеш-таблица? Опишите методы разрешения коллизий (цепочки, открытая адресация). Какие операции поддерживает хеш-таблица и их среднюю сложность?

Вопрос 2:

В чем разница между обходом графа в глубину (DFS) и в ширину (BFS)? Где применяется каждый из них? Приведите примеры задач.

Практическая часть

Задача 1:

Реализуйте алгоритм сортировки слиянием (MergeSort) для массива целых чисел.

Задача 2:

Напишите функцию для поиска кратчайшего пути в невзвешенном графе (используя BFS). Граф задан списком смежности:

```
std::unordered_map<int, std::vector<int>> graph;
```

Функция должна возвращать длину кратчайшего пути между двумя вершинами.

Перечень компетенций (части компетенции), проверяемых оценочным средством

ОПК-2.1; ОПК-2.2; ОПК-4.1; ОПК-5.1; ОПК-6.1; ОПК-6.2; PL-3.1

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

Методические рекомендации, определяющие процедуры оценивания выполнения контрольных заданий:

Задание считается выполненным при выполнении следующих условий:

- предоставлен исходный код на C++;
- продемонстрирован работоспособность приложения в среде Microsoft Visual Studio;
- студент понимает исходный код и отвечает на вопросы по его организации.

Критерии оценки:

- Корректность реализации (50%)
- Читаемость и структура кода (30%)
- Эффективность алгоритмов (20%)

Методические рекомендации, определяющие процедуры оценивания самостоятельной работы:

Оценивание результатов самостоятельной работы основывается на качестве выполнения студентом индивидуального задания. Код приложения реализуется в среде Microsoft Visual Studio, на языке C++.

Критерии оценки:

оценка «неудовлетворительно»: код программы не запускается в компиляторе;

оценка «удовлетворительно»: программа работает, но реализует часть необходимого функционала или работает на некоторых наборах данных;

оценка «хорошо»: представлена структура программы и ее компонентов, программа работает практически на всех наборах данных, за исключением некоторых частных случаев, реализован весь функционал задания;

оценка «отлично»: представлена структура приложения и ее компонентов, программа работает на всех наборах данных, включая частные случаи и функционал задания реализован полностью.

Для получения «**зачтено**» достаточно оценки «удовлетворительно» по индивидуальному заданию.

Методические рекомендации, определяющие процедуры оценивания лабораторных работ:

1. Процедура оценивания

Сдача работы:

Работа сдается в электронном виде (исходный код + отчет).

Допускается 2 попытки на исправление замечаний.

Проверка:

Критерий 1 → Корректность реализации алгоритма (тестирование на разных входных данных).

Критерий 2 → Соответствие кода стандартам качества (стиль, комментарии, структура).

Критерий 3 → Полнота отчета (описание решения, анализ сложности, выводы).

Защита (при необходимости):

Устные пояснения к коду.

Ответы на вопросы по теме работы.

2. Критерии оценки

Критерий	*"Зачтено"*	*"Незачтено"*

Корректность кода	Алгоритм работает верно на всех тестовых случаях. Обрабатывает крайние случаи.	Код не компилируется или дает неверные результаты. Нет обработки ошибок.
Качество кода	Читаемый стиль, есть комментарии, нет дублирования кода.	Бессистемные имена переменных, нет комментариев, нарушены принципы DRY/KISS.
Отчет	Полное описание решения, анализ сложности, выводы.	Отсутствует или содержит грубые ошибки в анализе.
Сроки сдачи	Сдано в срок или с допускаемым опозданием (до 3 дней).	Просрочено без уважительной причины.

3. Шкала оценивания

"Зачтено":

Выполнены все 3 критерия (код, качество, отчет).

Допускаются незначительные недочеты (например, мелкие стилевые погрешности).

"Незачтено":

Код не работает или работает некорректно.

Грубые нарушения в оформлении/анализе.

Плагиат или использование чужих решений без указания авторства.

4. Рекомендации для студентов

Перед сдачей:

Проверить код на разных входных данных (включая крайние случаи).

Убедиться, что соблюдены стилевые стандарты (отступы, именование).

Отчет.

Включить: Цель работы. Описание алгоритма. Анализ сложности (О-нотация).

Выводы (что получилось/не получилось).

Исправления:

При получении "Незачтено" исправить указанные ошибки и сдать повторно в течение недели.

Методические рекомендации, определяющие процедуры оценивания заданий на зачете:

Процедура промежуточной аттестации проходит в соответствии с Положением о текущем контроле и промежуточной аттестации обучающихся ФГБОУ ВО «КубГУ».

Итоговой формой контроля сформированности компетенций у обучающихся по дисциплине является зачет. Студенты обязаны получить зачет в соответствии с расписанием и учебным планом.

ФОС промежуточной аттестации состоит из контрольных заданий, заданий для самостоятельной работы и заданий на зачете.

Зачет по дисциплине преследует цель оценить работу студента, получение теоретических и практических знаний, их прочность, развитие творческого мышления, приобретение навыков самостоятельной работы, умение применять полученные знания для решения практических задач.

Результат сдачи зачета заноситься преподавателем в экзаменационную ведомость и зачетную книжку.

Оценивание уровня освоения дисциплины основывается на качестве выполнения студентом контрольных заданий, заданий для самостоятельной работы и задания на зачете.

Критерии оценки задания на зачете:

Зачет:

Правильные ответы на оба теоретических вопроса.

Решение одной практической задачи полностью и второй частично (например, работает, но есть недочеты).

Незачет:

Неверные ответы на оба теоретических вопроса.

Не решена ни одна практическая задача или код содержит критические ошибки.

Ответы оцениваются по следующим критериям:

– Полнота освещения темы
– Корректность приведенных примеров
– Глубина понимания принципов работы
– Практическая значимость изложенного материала

Оценочные средства для инвалидов и лиц с ограниченными возможностями здоровья выбираются с учетом их индивидуальных психофизических особенностей.

– при необходимости инвалидам и лицам с ограниченными возможностями здоровья предоставляется дополнительное время для подготовки ответа на экзамене;

– при проведении процедуры оценивания результатов обучения инвалидов и лиц с ограниченными возможностями здоровья предусматривается использование технических средств, необходимых им в связи с их индивидуальными особенностями;

– при необходимости для обучающихся с ограниченными возможностями здоровья и инвалидов процедура оценивания результатов обучения по дисциплине может проводиться в несколько этапов.

Процедура оценивания результатов обучения инвалидов и лиц с ограниченными возможностями здоровья по дисциплине предусматривает предоставление информации в формах, адаптированных к ограничениям их здоровья и восприятия информации:

Для лиц с нарушениями зрения:
– в печатной форме увеличенным шрифтом,
– в форме электронного документа.

Для лиц с нарушениями слуха:
– в печатной форме,
– в форме электронного документа.

Для лиц с нарушениями опорно-двигательного аппарата:
– в печатной форме,
– в форме электронного документа.

Данный перечень может быть конкретизирован в зависимости от контингента обучающихся.

4.3. Методические указания по организации вычислительной инфраструктуры

Условия применения:

– Курс рассчитан на студентов 1 года обучения.
– Наличие доступа к вычислительным ресурсам (GitLab, компиляторы C++).
– Индивидуальные задания включают задачи на реализацию алгоритмов и структур данных, проверяемые автотестами.

Цели, задачи и ожидаемые результаты:

Цели:

– Обеспечить студентов инструментами для работы с алгоритмами и структурами данных.
– Приучить к использованию Git, CI/CD и автотестов для контроля качества кода.

Задачи преподавателя:

1. Создание учетных записей студентов в GitLab.
2. Настройка GitLab Runner для автоматического тестирования.
3. Разработка шаблонного репозитория для лабораторных работ.
4. Написание автотестов для проверки корректности реализации алгоритмов.
5. Визуализация результатов тестирования.

Ожидаемые результаты студентов:

– Умение работать с Git и CI/CD.
– Навыки написания чистого, тестируемого кода на C++.

– Понимание важности автоматической проверки алгоритмов.

Порядок реализации:

1. **Создание учетных записей в GitLab:**

Каждый студент получает доступ к приватному репозиторию.

2. **Настройка GitLab Runner:**

Используется Docker-образ с компилятором C++ (g++/clang) и фреймворками для тестирования.

3. **Шаблонный репозиторий:**

Включает:

– .gitlab-ci.yml для автоматического тестирования.
– README.md с инструкциями.
– Примеры кода и тестов.

4. **Автотесты:**

Проверяют корректность реализации алгоритмов (например, сортировки, работы с деревьями, хеш-таблицами).

5. **Визуализация результатов:**

Генерация отчетов с указанием успешных и проваленных тестов.

Порядок проверки корректности:

Чек-лист:

– Наличие Git-репозиториев у всех студентов.
– Корректная работа CI/CD.
– Автотесты покрывают ключевые функции.

4.4. Методические указания по организации лабораторных работ

Условия применения:

– Курс включает 9 лабораторных работ (см. раздел 2.3.3 РПД).
– Для выполнения требуется:

Среда разработки (Visual Studio).

GitLab для сдачи заданий.

Цели, задачи и ожидаемые результаты:

Цели:

– Практическое освоение алгоритмов и структур данных.
– Развитие навыков отладки и оптимизации кода.

Задачи преподавателя:

1. Подготовка плана лабораторных работ.
2. Разработка индивидуальных заданий.
3. Организация Git-инфраструктуры и автотестов.

Ожидаемые результаты студентов:

– Умение реализовывать и анализировать алгоритмы.
– Навыки работы с Git и CI/CD.

Порядок реализации:

1. **План лабораторных работ:**

Соответствует темам из РПД (сортировки, деревья, графы, хеш-таблицы и т. д.).

2. **Пример задания (ЛР №3: "Сравнение алгоритмов сортировки"):**

– Реализовать сортировки: пузырьком, выбором, быструю.
– Сравнить время работы на разных входных данных.
– Автотесты проверяют корректность сортировки и замеряют время.

3. **Критерии оценки:**

– **зачтено:** Полная реализация, анализ сложности, код проходит тесты.
– **Незачтено:** Код не работает или не соответствует ТЗ.

Порядок проверки корректности:

Чек-лист:

- Наличие автотестов для каждой лабораторной работы.
- Инструкции по именованию коммитов и методов.

4.5. Методические указания по организации проектной деятельности студентов

Условия применения:

- Курс включает проектную работу (16 часов).
- Доступны кейсы от преподавателей.

Цели, задачи и ожидаемые результаты:

Цели:

- Применение алгоритмов и структур данных в реальных задачах.
- Развитие навыков командной работы.

Задачи преподавателя:

1. Сбор кейсов (например, оптимизация поиска в больших данных).
2. Формирование ТЗ для проектов.
3. Разработка системы оценки.

Ожидаемые результаты студентов:

Умение решать прикладные задачи с использованием изученных алгоритмов.

Порядок реализации:

1. **Пример кейса:**

Оптимизация хеш-таблицы для ускорения поиска в базе данных.

2. **ТЗ для проекта:**

- Реализовать хеш-таблицу с методами цепочек и открытой адресации.
- Сравнить производительность на реальных данных.

3. **Критерии оценки:**

- Корректность реализации.
- Анализ эффективности (O-нотация).

Порядок проверки корректности:

Чек-лист:

- Наличие минимум 10 кейсов.
- Четкие критерии оценки проектов.

5. Перечень учебной литературы, информационных ресурсов и технологий

5.1 Учебная литература:

1. Кормен, Т. Алгоритмы: построение и анализ [Текст] / Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн. - 4-е изд. - Москва : Вильямс, 2023. - 1344 с. - ISBN 978-5-8459-2387-6.
2. Скиена, С.С. Алгоритмы. Руководство по разработке [Текст] / Стивен Скиена. - 3-е изд. - Санкт-Петербург : БХВ-Петербург, 2022. - 736 с. - ISBN 978-5-9775-0815-5.
3. Седжвик, Р. Алгоритмы на C++ [Текст] : в 2 ч. / Роберт Седжвик. - Москва : Диалектика, 2023.
4. Ч. 1: Фундаментальные алгоритмы. - 2023. - 592 с. - ISBN 978-5-907203-45-2.
5. Ч. 2: Алгоритмы на графах. - 2023. - 512 с. - ISBN 978-5-907203-46-9.
6. Страуструп, Б. Программирование: принципы и практика с использованием C++ [Текст] / Бьярне Страуструп. - 3-е изд. - Москва : Вильямс, 2023. - 1312 с. - ISBN 978-5-907144-72-6.
7. Мейерс, С. Эффективный и современный C++ [Текст] / Скотт Мейерс. - 2-е изд. - Москва : Вильямс, 2023. - 320 с. - ISBN 978-5-907144-73-3.
8. Ахо, А.В. Структуры данных и алгоритмы [Текст] / Альфред Ахо, Джеффри Ульман. - Москва : Диалектика, 2023. - 480 с. - ISBN 978-5-907203-47-6.

9. Окулов, С.М. Программирование на C++: алгоритмы и структуры данных [Текст] / С.М. Окулов. - Москва : Лаборатория знаний, 2023. - 384 с. - ISBN 978-5-00172-123-4.
10. Павловская, Т.А. C++: объектно-ориентированное программирование [Текст] : учебное пособие / Т.А. Павловская. - Санкт-Петербург : Питер, 2023. - 496 с. - ISBN 978-5-4461-5678-9.
11. Шилдт, Г. C++20: полное руководство [Текст] / Герберт Шилдт. - Москва : Диалектика, 2023. - 1056 с. - ISBN 978-5-907203-48-3.
12. Подбельский, В.В. Язык C++: учебное пособие [Текст] / В.В. Подбельский. - 7-е изд. - Москва : Финансы и статистика, 2023. - 576 с. - ISBN 978-5-279-04567-5.
13. Прата, С. Язык программирования C++: лекции и упражнения [Текст] / Стивен Прата. - 7-е изд. - Санкт-Петербург : БХВ-Петербург, 2023. - 1248 с. - ISBN 978-5-9775-0816-2.
14. Липпман, С.Б. Язык программирования C++: базовый курс [Текст] / Стенли Б. Липпман. - 6-е изд. - Москва : Вильямс, 2023. - 1120 с. - ISBN 978-5-907144-74-0.
15. Джосаттис, Н.М. Стандартная библиотека C++: справочное руководство [Текст] / Николай М. Джосаттис. - 3-е изд. - Москва : Вильямс, 2023. - 1136 с. - ISBN 978-5-907144-75-7.
16. Александреску, А. Современное проектирование на C++ [Текст] / Андрей Александреску. - 2-е изд. - Москва : Вильямс, 2023. - 336 с. - ISBN 978-5-907144-76-4.
17. Саттер, Г. Решение сложных задач на C++ [Текст] / Герб Саттер. - Москва : Вильямс, 2023. - 272 с. - ISBN 978-5-907144-77-1.
18. Лафоре, Р. Структуры данных и алгоритмы в C++ [Текст] / Роберт Лафоре. - 3-е изд. - Санкт-Петербург : Питер, 2023. - 992 с. - ISBN 978-5-4461-5679-6.
19. Макконнелл, Дж. Анализ алгоритмов [Текст] / Джон Макконнелл. - 3-е изд. - Москва : Техносфера, 2023. - 480 с. - ISBN 978-5-94836-678-9.
20. Керниган, Б.Я. Практика программирования [Текст] / Брайан Керниган, Роб Пайк. - Москва : Вильямс, 2023. - 288 с. - ISBN 978-5-907144-78-8.
21. Таненбаум, Э.С. Современные операционные системы [Текст] / Эндрю Таненбаум. - 5-е изд. - Санкт-Петербург : Питер, 2023. - 1120 с. - ISBN 978-5-4461-5680-2.
22. Форд, У. Алгоритмические трюки для программистов [Текст] / Уоррен Форд. - Москва : Диалектика, 2023. - 512 с. - ISBN 978-5-907203-49-0.

5.2. Периодические издания:

1. Базы данных компании «Ист Вью» http://dlib.eastview.com
2. Электронная библиотека GREBENNIKON.RU https://grebennikon.ru/

5.3. Интернет-ресурсы, в том числе современные профессиональные базы данных и информационные справочные системы

Электронно-библиотечные системы (ЭБС):

1. ЭБС «ЮРАЙТ» https://urait.ru/
2. ЭБС «УНИВЕРСИТЕТСКАЯ БИБЛИОТЕКА ОНЛАЙН» http://www.biblioclub.ru/
3. ЭБС «BOOK.ru» https://www.book.ru
4. ЭБС «ZNANIUM.COM» www.znanium.com
5. ЭБС «ЛАНЬ» https://e.lanbook.com

Профессиональные базы данных

1. Scopus http://www.scopus.com/
2. ScienceDirect https://www.sciencedirect.com/
3. Журналы издательства Wiley https://onlinelibrary.wiley.com/
4. Научная электронная библиотека (НЭБ) http://www.elibrary.ru/
5. Полнотекстовые архивы ведущих западных научных журналов на Российской платформе научных журналов НЭИКОН http://archive.neicon.ru

6. Национальная электронная библиотека (доступ к Электронной библиотеке диссертаций Российской государственной библиотеки (РГБ) https://rusneb.ru/
7. Президентская библиотека им. Б.Н. Ельцина https://www.prlib.ru/
8. База данных CSD Кембриджского центра кристаллографических данных (CCDC) https://www.ccdc.cam.ac.uk/structures/
9. Springer Journals: https://link.springer.com/
10. Springer Journals Archive: https://link.springer.com/
11. Nature Journals: https://www.nature.com/
12. Springer Nature Protocols and Methods: https://experiments.springernature.com/sources/springer-protocols
13. Springer Materials: http://materials.springer.com/
14. Nano Database: https://nano.nature.com/
15. Springer eBooks (i.e. 2020 eBook collections): https://link.springer.com/
16. "Лекториум ТВ" http://www.lektorium.tv/
17. Университетская информационная система РОССИЯ http://uisrussia.msu.ru

Информационные справочные системы

1. Консультант Плюс - справочная правовая система (доступ по локальной сети с компьютеров библиотеки)

Ресурсы свободного доступа

1. КиберЛенинка http://cyberleninka.ru/;
2. Американская патентная база данных http://www.uspto.gov/patft/
3. Министерство науки и высшего образования Российской Федерации https://www.minobrnauki.gov.ru/;
4. Федеральный портал "Российское образование" http://www.edu.ru/;
5. Информационная система "Единое окно доступа к образовательным ресурсам" http://window.edu.ru/;
6. Единая коллекция цифровых образовательных ресурсов http://school-collection.edu.ru/ .
7. Проект Государственного института русского языка имени А.С. Пушкина "Образование на русском" https://pushkininstitute.ru/;
8. Справочно-информационный портал "Русский язык" http://gramota.ru/;
9. Служба тематических толковых словарей http://www.glossary.ru/;
10. Словари и энциклопедии http://dic.academic.ru/;
11. Образовательный портал "Учеба" http://www.ucheba.com/;
12. Законопроект "Об образовании в Российской Федерации". Вопросы и ответы http://xn--273--84d1f.xn--p1ai/voprosy_i_otvety

Собственные электронные образовательные и информационные ресурсы КубГУ

1. Электронный каталог Научной библиотеки КубГУ http://megapro.kubsu.ru/MegaPro/Web
2. Электронная библиотека трудов ученых КубГУ http://megapro.kubsu.ru/MegaPro/UserEntry?Action=ToDb&idb=6
3. Среда модульного динамического обучения http://moodle.kubsu.ru
4. База учебных планов, учебно-методических комплексов, публикаций и конференций http://infoneeds.kubsu.ru/
5. Библиотека информационных ресурсов кафедры информационных образовательных технологий http://mschool.kubsu.ru;
6. Электронный архив документов КубГУ http://docspace.kubsu.ru/
7. Электронные образовательные ресурсы кафедры информационных систем и технологий в образовании КубГУ и научно-методического журнала "ШКОЛЬНЫЕ ГОДЫ" http://icdau.kubsu.ru/

5.4 Перечень информационно-коммуникационных технологий
Среда разработки программ на языке программирования C++
Текстовый редактор

5.5 Перечень лицензионного и свободно распространяемого программного обеспечения

Microsoft Visual Studio C++ Community свободно распространяемая
LibreOffice свободно распространяемая

6. Методические указания для обучающихся по освоению дисциплины (модуля)

По курсу предусмотрено проведение лекционных занятий, на которых дается основной систематизированный материал. В ходе лекционных занятий разбираются основные алгоритмические схемы и приемы программирования для различных информационных структур данных, приводятся примеры их использования, проводится анализ наиболее распространенных ошибок реализации, рассматриваются базовые классы языка программирования C++. После прослушивания лекции рекомендуется выполнить упражнения, приводимые в аудитории для самостоятельной работы.

По курсу предусмотрено проведение лабораторных занятий, на которых дается прикладной систематизированный материал. В ходе занятий разбираются готовые примеры алгоритмических решений, примеры решения типовых задач, предлагаются к программной реализации базовые алгоритмические методы. После занятия рекомендуется выполнить упражнения, приводимые в аудитории для самостоятельной работы.

При самостоятельной работе студентов необходимо изучить литературу, приведенную в перечнях выше, для осмысления вводимых понятий, анализа предложенных подходов и методов разработки программ. Разрабатывая решение новой задачи, студент должен уметь выбрать эффективные и надежные структуры данных для представления информации, подобрать соответствующие алгоритмы для их обработки, учесть специфику языка программирования, на котором будет выполнена реализация. Студент должен уметь выполнять тестирование и отладку алгоритмов решения задач с целью обнаружения и устранения в них ошибок.

Важнейшим этапом курса является самостоятельная работа по дисциплине. В процессе самостоятельной работы студент приобретает навык создания законченного программного продукта.

Используются активные, инновационные образовательные технологии, которые способствуют развитию общекультурных, общепрофессиональных компетенций и профессиональных компетенций обучающихся:

- проблемное обучение;
- разноуровневое обучение;
- проектные методы обучения;
- исследовательские методы в обучении;
- обучение в сотрудничестве (командная, групповая работа);
- информационно- коммуникационные технологии.

Для лиц с ограниченными возможностями здоровья предусмотрена организация консультаций с использованием электронной почты.

Учебно-методическим обеспечением курсовой работы студентов являются:

1. учебная литература;
2. нормативные документы ВУЗа;
3. методические разработки для студентов.

Самостоятельная работа студентов включает:
– оформление итогового отчета (пояснительной запики).
– анализ нормативно-методической базы организации;
– анализ научных публикации по заранее определённой теме;
– анализ и обработку информации;
– работу с научной, учебной и методической литературой,
– работа с конспектами лекций, ЭБС.

Для самостоятельной работы представляется аудитория с компьютером и доступом в Интернет, к электронной библиотеке вуза и к информационно-справочным системам.
Перечень учебно-методического обеспечения:

1. Основная образовательная программа высшего профессионального образования федерального государственного бюджетного образовательного учреждения высшего образования «Кубанский государственный университет» по направлению подготовки.
2. Положение о проведении текущего контроля успеваемости и промежуточной аттестации в федеральном государственном бюджетном образовательном учреждении высшего образования «Кубанский государственный университет».
3. Общие требования к построению, содержанию, оформлению и утверждению рабочей программы дисциплины Федерального государственного образовательного стандарта высшего профессионального образования.
4. Методические рекомендации по содержанию, оформлению и применению образовательных технологий и оценочных средств в учебном процессе, основанном на Федеральном государственном образовательном стандарте.
5. Учебный план основной образовательной программы по направлению подготовки.
6. Федеральный государственный образовательный стандарт высшего профессионального образования по направлению подготовки.

В освоении дисциплины инвалидами и лицами с ограниченными возможностями здоровья большое значение имеет индивидуальная учебная работа (консультации) – дополнительное разъяснение учебного материала.

Индивидуальные консультации по предмету являются важным фактором, способствующим индивидуализации обучения и установлению воспитательного контакта между преподавателем и обучающимся инвалидом или лицом с ограниченными возможностями здоровья.

7. Материально-техническое обеспечение по дисциплине (модулю)

№	Вид работ	Наименование учебной аудитории, ее оснащенность оборудованием и техническими средствами обучения
1.	Лекционные занятия	Аудитория, укомплектованная специализированной мебелью и техническими средствами обучения
2.	Лабораторные занятия	Аудитория, укомплектованная специализированной мебелью и техническими средствами обучения, компьютерами, проектором, программным обеспечением
3.	Групповые (индивидуальные) консультации	Аудитория, укомплектованная специализированной мебелью и техническими средствами обучения, компьютерами, программным обеспечением
4.	Текущий контроль, промежуточная аттестация	Аудитория, укомплектованная специализированной мебелью и техническими средствами обучения, компьютерами, программным обеспечением
5.	Самостоятельная работа	Кабинет для самостоятельной работы, оснащенный компьютерной техникой с возможностью подключения к

		сети «Интернет», программой экранного увеличения и обеспеченный доступом в электронную информационно-образовательную среду университета.

Примечание: Конкретизация аудиторий и их оснащение определяется ОПОП.