

**Федеральное государственное автономное образовательное
учреждение высшего образования
«Московский физико-технический институт
(национальный исследовательский университет)»**

УТВЕРЖДЕНО

**Директор физтех-школы
прикладной математики и
информатики**

А.М. Райгородский

	Рабочая программа дисциплины (модуля)
по дисциплине:	Технологии программирования
по направлению:	Прикладная математика и информатика
профиль подготовки:	А1360: Передовые методы искусственного интеллекта Физтех-школа Прикладной Математики и Информатики кафедра алгоритмов и технологий программирования
курс:	1
квалификация:	бакалавр

Семестр, формы промежуточной аттестации: 2 (весенний) - Дифференцированный зачет

Аудиторных часов: 60 всего, в том числе:

лекции: 30 час.

семинары: 30 час.

лабораторные занятия: 0 час.

Самостоятельная работа: 30 час.

Всего часов: 90, всего зач. ед.: 2

Количество контрольных работ, заданий: 2

Программу составил: В.В. Яковлев, канд. физ.-мат. наук, заведующий кафедрой

Программа обсуждена на заседании кафедры алгоритмов и технологий программирования 05.08.2025

Аннотация

Дисциплина «Технологии программирования» направлена на формирование у студентов базовых знаний и практических навыков в области организации промышленной разработки программного обеспечения. Курс охватывает ключевые аспекты жизненного цикла программных и ИИ-продуктов: проектирование архитектуры, использование паттернов и антипаттернов, применение инструментов сборки, тестирования и обеспечения качества кода, а также внедрение CI/CD-процессов.

Особое внимание уделяется применению технологий программирования в задачах искусственного интеллекта: от формализации требований и постановки задач до проектирования архитектуры ИИ-систем и управления процессом их разработки. Студенты осваивают современные практики разработки, интеграции и сопровождения ИИ-продуктов, учатся проводить эксперименты на данных и обеспечивать воспроизводимость решений.

Образовательный процесс строится на сочетании лекционных занятий, интерактивных семинаров, мастер-классов, проектных практикумов и хакатонов. В программу интегрированы практические кейсы индустриальных партнёров, которые позволяют отработать навыки построения архитектур ИИ-систем, управления проектами и оценки качества решений.

Курс базируется на знаниях, полученных при изучении дисциплины «Алгоритмы и структуры данных», и является предшествующим для дисциплины «Формальные языки и трансляции».

1. Цели и задачи

Цель дисциплины

Цель дисциплины «Технологии программирования» – формирование у студентов знаний и навыков применения современных технологий и инструментов промышленной разработки программного обеспечения, включая проектирование архитектуры, использование средств тестирования и автоматизации процессов (CI/CD), а также освоение практик, необходимых для создания, интеграции и сопровождения систем искусственного интеллекта и цифровых сервисов.

Задачи дисциплины

приобретение студентами навыков работы в командной строке, инструментами сборки и системами контроля версий;

овладение студентами современными практиками разработки и типовыми шаблонами проектирования.

2. Перечень формируемых компетенций

Освоение дисциплины направлено на формирование следующих компетенций:

Код и наименование компетенции	Индикаторы достижения компетенции
ОПК-1 Способен применять фундаментальные знания, полученные в области физико-математических и (или) естественных наук и использовать их в профессиональной деятельности	ОПК-1.2 Способен строить математические модели, производить количественные расчеты и оценки
ОПК-2 Способен использовать современные информационные технологии и программные средства при решении задач профессиональной деятельности, соблюдая требования информационной безопасности	ОПК-2.2 Знает и умеет применять численные математические методы и прикладное программное обеспечение для решения научных задач в профессиональной области
ЛС-3 Способен проектировать и поддерживать архитектуру систем искусственного интеллекта	ЛС-3.1 Создает и развивает архитектуры системы ИИ на всех этапах жизненного цикла
ЛС-2 Способен проводить эксперименты на данных, формулировать гипотезы исследования, строить (обучать, дообучать)	ЛС-2.1 Проводит эксперименты с моделями ИИ, оценивает их качество (точность, производительность)

модели машинного обучения с оценкой их качества и анализом ошибок, обеспечивать воспроизводимость и масштабируемость исследований на данных	LC-2.2 Проводит эксперименты на данных и визуализирует результаты с применением технологий анализа данных (статистического анализа), методов и алгоритмов машинного обучения
---	--

3. Перечень планируемых результатов обучения по дисциплине (модулю)

В результате освоения дисциплины обучающиеся должны

знать:

шаблоны проектирования программного обеспечения.

уметь:

работать с интерфейсом командной строки;

выполнять сборку программ из исходных текстов и их отладку, без использования интегрированных средств разработки;

пользоваться системами контроля версий;

настраивать окружение для непрерывной интеграции разработки проекта;

проектировать программное обеспечение таким образом, чтобы его поддержка осуществлялась коллективом из нескольких разработчиков.

владеть:

навыками работы с GitLab и GitLab CI.

4. Содержание дисциплины (модуля), структурированное по темам (разделам) с указанием отведенного на них количества академических часов и видов учебных занятий

4.1. Разделы дисциплины (модуля) и трудоемкости по видам учебных занятий

№	Тема (раздел) дисциплины	Трудоемкость по видам учебных занятий, включая самостоятельную работу, час.			
		Лекции	Семинары	Лаборат. работы	Самост. работа
1	Введение в ОС Linux	3	3		
2	Системы контроля версий	3	3		3
3	Процесс компиляции	4	4		4
4	Организация процесса сборки	3	3		3
5	Кросс-компиляция и методы отладки	3	3		4
6	Порождающие паттерны проектирования	4	4		4
7	Структурные паттерны	3	3		4
8	Поведенческие паттерны	3	3		4
9	Модели	4	4		4
Итого часов		30	30		30
Подготовка к экзамену		0 час.			
Общая трудоёмкость		90 час., 2 зач.ед.			

4.2. Содержание дисциплины (модуля), структурированное по темам (разделам)

Семестр: 2 (Весенний)

1. Введение в ОС Linux

Лекция: Основы Linux, архитектура ОС, преимущества.

Ключевые темы:

Основы Linux, архитектура ОС
Работа с командной строкой (ls, cd, grep, chmod и др.)
Организация файловой системы
Автоматизация рутинных задач

Примеры форматов:

Конкурсы индивидуальных проектов: «Самый эффективный Bash-однострочник», «Скрипт для мониторинга процессов».

Групповые проекты: «Система логирования изменений в файловой системе», «Автоматизация резервного копирования».

Мастер-классы: «Linux в инфраструктуре высоконагруженных систем» (Сбер), «DevOps-практики на Linux» (Яндекс).

Практические кейсы: «Организация журналирования и мониторинга» (Сбер), «Linux-среда для ML-разработки» (Яндекс).

Хакатоны: «Linux Automation Challenge: автоматизация DevOps-задач».

2. Системы контроля версий

Лекция: Принципы VCS, различия между централизованными и распределёнными системами.

Ключевые темы:

Принципы VCS (централизованные и распределённые системы)

Создание и организация репозитория

Работа с ветками и слиянием

Организация совместной разработки

Примеры форматов:

Конкурсы индивидуальных проектов: «История разработки: 100 коммитов — 1 проект», «Оптимизация Git workflow».

Групповые проекты: «Командная разработка с Git Flow», «CI/CD pipeline на GitHub Actions».

Мастер-классы: «Git в больших командах: опыт Авито и Яндекс» (Авито/Яндекс).

Практические кейсы: «Разбор конфликтов слияния в реальном проекте» (Авито), «GitOps в ML-разработке» (Сбер).

Хакатоны: «Git Merge Battle: командная работа в условиях конфликтов».

3. Процесс компиляции

Лекция: Стадии компиляции (препроцессинг, компиляция, ассемблирование, линковка).

Ключевые темы:

Стадии компиляции: препроцессинг, компиляция, ассемблирование, линковка

Работа с gcc и промежуточными файлами (.i, .s, .o)

Оптимизация сборки и исследование артефактов

Примеры форматов:

Конкурсы индивидуальных проектов: «Мини-компилятор для простого языка», «Сравнение скорости разных ключей оптимизации gcc».

Групповые проекты: «Система мониторинга времени сборки», «Реализация пайплайна с анализом артефактов».

Мастер-классы: «Оптимизация сборки в больших проектах» (Яндекс), «Build-системы в промышленной разработке» (Сбер).

Практические кейсы: «Анализ узких мест при сборке backend-сервисов» (Авито), «Оптимизация пайплайнов CI/CD» (Яндекс).

Хакатоны: «Build Challenge: ускорение сборки большого проекта».

4. Организация процесса сборки

Лекция: Обзор систем сборки (Make, CMake, Bazel).

Ключевые темы:

Системы сборки (Make, CMake, Bazel)

Создание и использование Makefile

Высокоуровневые инструменты сборки

CI/CD в современных проектах

Примеры форматов:

Конкурсы индивидуальных проектов: «Лучший Makefile для большого проекта», «Сравнение CMake и Bazel на реальном коде».

Групповые проекты: «Организация CI/CD пайплайна», «Миграция проекта на новую систему сборки».

Мастер-классы: «CI/CD в Авито» (Авито), «Опыт внедрения Jenkins и GitHub Actions» (Сбер).

Практические кейсы: «Снижение времени сборки микросервисов» (Яндекс), «Организация пайплайнов для ML-моделей» (Сбер).

Хакатоны: «Build&Deploy Hack: ускорение и автоматизация пайплайнов».

5. Кросс-компиляция и методы отладки

Лекция: Принципы кросс-компиляции, инструменты (QEMU, Docker).

Ключевые темы:

Принципы кросс-компиляции

Инструменты (QEMU, Docker)

Сборка под разные архитектуры и ОС

Отладка с использованием виртуализации и эмуляторов

Примеры форматов:

Конкурсы индивидуальных проектов: «Сборка приложения под ARM», «Dockerfile для ML-проекта».

Групповые проекты: «Разработка и отладка embedded-системы», «Организация виртуализированной среды».

Мастер-классы: «Отладка embedded-программ» (Сбер), «Docker и QEMU в DevOps» (Авито).

Практические кейсы: «Сборка и отладка под ARM-системы в телеком-проектах» (Яндекс), «Контейнеризация приложений в Авито».

Хакатоны: «Cross-Compile Hack: перенос ПО между архитектурами».

6. Порождающие паттерны проектирования

Лекция: Классификация паттернов (порождающие, структурные, поведенческие).

Ключевые темы:

Классификация паттернов проектирования

Порождающие паттерны: Singleton, Factory, Prototype

Примеры форматов:

Конкурсы индивидуальных проектов: «Реализация Singleton в разных языках», «Factory Method для web-приложения».

Групповые проекты: «Применение порождающих паттернов в backend-сервисе», «Сравнение производительности паттернов».

Мастер-классы: «Паттерны в промышленной разработке» (Яндекс), «Примеры из enterprise-систем» (Сбер).

Практические кейсы: «Проектирование сервисов Авито с применением паттернов», «Использование паттернов в микросервисной архитектуре» (Яндекс).

Хакатоны: «Design Patterns Hack: проектирование через паттерны».

7. Структурные паттерны

Лекция: Структурные паттерны

Ключевые темы:

Структурные паттерны: Adapter, Composite, Proxy

Практическая реализация паттернов на выбранных языках

Примеры форматов:

Конкурсы индивидуальных проектов: «Adapter для интеграции разных API», «Proxy для защиты сервисов».

Групповые проекты: «Composite-паттерн для обработки данных», «Реализация Proxy в веб-приложении».

Мастер-классы: «Структурные паттерны в сервисах Яндекса» (Яндекс), «Adapter и Proxy в реальной разработке» (Авито).

Практические кейсы: «Оптимизация интеграции API» (Сбер), «Composite в обработке больших данных» (Яндекс).

Хакатоны: «Structural Patterns Challenge: проектирование и оптимизация».

8. Поведенческие паттерны

Лекция: Поведенческие паттерны

Ключевые темы:

Поведенческие паттерны: Strategy, Observer, State

Примеры применения в enterprise-разработке

Примеры форматов:

Конкурсы индивидуальных проектов: «Реализация Strategy для алгоритмов сортировки», «Observer для системы событий».

Групповые проекты: «State-машина для чат-бота», «Observer для многопоточного приложения».

Мастер-классы: «Поведенческие паттерны в разработке больших систем» (Сбер), «Strategy и Observer в ML-сервисах» (Яндекс).

Практические кейсы: «Observer в микросервисной архитектуре» (Авито), «State-машина для управления процессами» (Сбер).

Хакатоны: «Behavioral Patterns Hack: проектирование решений для реальных систем».

9. Модели

Лекция: Принцип разделения ответственности.

Ключевые темы:

Принцип разделения ответственности

Архитектурный паттерн MVC

Реализация MVC в веб-разработке

Примеры форматов:

Конкурсы индивидуальных проектов: «Реализация MVC на Python (Flask)», «MVC для чат-приложения».

Групповые проекты: «Создание мини-фреймворка на основе MVC», «MVC для учебного веб-портала».

Мастер-классы: «MVC в enterprise-разработке» (Сбер), «Архитектура MVC в крупных проектах» (Яндекс).

Практические кейсы: «Применение MVC в backend Авито», «Реализация MVC в e-commerce проектах» (Яндекс).

Хакатоны: «MVC Hack: создание веб-приложения за 24 часа».

5. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине (модулю)

Для проведения занятий по дисциплине используется учебная аудитория, оснащённая компьютерной техникой с предустановленным лицензионным программным обеспечением и доступом к интернету, а также мультимедийным оборудованием (проектор, звуковая система). Практические занятия и групповые проектные работы поддерживаются с использованием облачных сервисов совместной работы (GitHub, GitLab, облачные IDE, системы видеоконференций).

Опционально, по согласованию с индустриальными партнёрами, студенты могут участвовать в дистанционных практиках на их площадках с использованием корпоративных облачных платформ, хранилищ данных и специализированных инструментов, что обеспечивает приближение учебного процесса к реальным условиям разработки программного обеспечения и управления жизненным циклом ИТ-продуктов.

6. Перечень рекомендуемой литературы

Основная литература

1. Введение в программирование, учебное пособие / И. Ю. Баженова, В. А. Сухомлин. — Москва, ИНТУИТ, 2016. — URL: <https://e.lanbook.com/book/100695> (дата обращения: 30.12.2020). - Полный текст (Режим доступа : из сети МФТИ / Удаленный доступ)

Дополнительная литература

Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1994.

Стив Макконнелл. Совершенный код. СПб.: Питер, 2021.

Мартин Р. Чистый код. Создание, анализ и рефакторинг. СПб.: Питер, 2021.

Kernighan B., Pike R. The Practice of Programming. Addison-Wesley, 1999.

Martin R. Чистая архитектура. Искусство разработки программного обеспечения. СПб.: Питер, 2022.

Fowler M. Refactoring: Improving the Design of Existing Code. Addison-Wesley, 2019.

Hunt A., Thomas D. The Pragmatic Programmer. Addison-Wesley, 2019.

Freeman E., Freeman E. Head First Design Patterns. O'Reilly, 2020.

Литература по треку «Искусственный интеллект»

Russell S., Norvig P. Искусственный интеллект: современный подход. М.: Вильямс, 2020.

Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly, 2022.

Кулаков В.Ю., Мешеряков Р.В. Технологии разработки интеллектуальных систем. М.: НИЦ «Инфра-М», 2021.

Goodfellow I., Bengio Y., Courville A. Deep Learning. MIT Press, 2016.

Научные публикации (пример):

Rudakov M. I. et al. Activations and gradients compression for model-parallel training //Doklady Mathematics. – Moscow : Pleiades Publishing, 2023. – Т. 108. – №. Suppl 2. – С. S272-S281. DOI 10.1134/S1064562423701314

Trusov A. et al. 4.6-Bit quantization for fast and accurate neural network inference on cpus //Mathematics. – 2024. – Т. 12. – №. 5. – С. 651. DOI 10.3390/math12050651

Sher A. et al. Neuron-by-neuron quantization for efficient low-bit QNN training //Mathematics. – 2023. – Т. 11. – №. 9. – С. 2112. DOI 10.3390/math11092112

7. Перечень ресурсов информационно-телекоммуникационной сети "Интернет", необходимых для освоения дисциплины (модуля)

<http://docs.oracle.com/javase/specs/jls/se8/html/index.html> - The Java Language Specification.

<https://google-styleguide.googlecode.com/svn/trunk/javaguide.html> — Google Java Style Guide.

Перечень ресурсов информационно-телекоммуникационной сети "Интернет", необходимых для освоения дисциплины (модуля)

1. <https://git-scm.com/book/ru/v2>

2. <https://www.gnu.org/software/bash/manual/bash.html>

GitHub Docs: <https://docs.github.com>

GitLab Documentation: <https://docs.gitlab.com>

Docker Documentation: <https://docs.docker.com>

Jenkins Documentation: <https://www.jenkins.io/doc>

CI/CD Best Practices (Google Cloud): <https://cloud.google.com/architecture/devops>

Python Package Index (PyPI): <https://pypi.org>

8. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю), включая перечень необходимого программного обеспечения и информационных справочных систем (при необходимости)

Учебная аудитория, оснащенная компьютером и мультимедийным оборудованием (проектор, звуковая система).

Операционная система Linux с правами системного администратора.

Возможно использование системы через средства виртуализации.

Среда разработки и системное ПО: Linux (Ubuntu/Debian), Windows Subsystem for Linux, инструменты командной строки.

Системы контроля версий и совместной работы: Git, GitHub/GitLab, облачные SaaS-сервисы для управления проектами (Trello, Jira, Notion или российские аналоги.).

Инструменты сборки и CI/CD: Make, CMake, Docker, Jenkins, GitHub Actions.

Средства отладки и виртуализации: GDB, QEMU, Docker, виртуальные машины.

Архитектурное и проектное моделирование: UML-редакторы, PlantUML, draw.io.

Онлайн-сервисы для проведения занятий и практик с индустриальными партнёрами: Zoom, MS Teams, Miro или российские аналоги.

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

Успешное освоение курса требует напряжённой самостоятельной работы студента. В программе курса приведено минимально необходимое время для работы студента над темой. Самостоятельная работа включает в себя:

– проработку учебного материала (по конспектам лекций, учебной и научной литературе), подготовку ответов на вопросы, предназначенных для самостоятельного изучения, доказательство отдельных утверждений, свойств;

– подготовку к практическим занятиям, выполнение двух индивидуальных домашних заданий.

Промежуточный контроль знаний проводится в виде письменных опросов по теории, а также студенту в ходе освоения курса необходимо выполнить две домашние индивидуальные работы с их последующей защитой:

ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

по направлению: Прикладная математика и информатика
профиль подготовки: АІ360: Передовые методы искусственного интеллекта
Физтех-школа Прикладной Математики и Информатики
кафедра алгоритмов и технологий программирования
курс: 1
квалификация: бакалавр

Семестр, формы промежуточной аттестации: 2 (весенний) - Дифференцированный зачет

Разработчик: В.В. Яковлев, канд. физ.-мат. наук, заведующий кафедрой

1. Компетенции, формируемые в процессе изучения дисциплины

Код и наименование компетенции	Индикаторы достижения компетенции
ОПК-1 Способен применять фундаментальные знания, полученные в области физико-математических и (или) естественных наук и использовать их в профессиональной деятельности	ОПК-1.2 Способен строить математические модели, производить количественные расчеты и оценки
ОПК-2 Способен использовать современные информационные технологии и программные средства при решении задач профессиональной деятельности, соблюдая требования информационной безопасности	ОПК-2.2 Знает и умеет применять численные математические методы и прикладное программное обеспечение для решения научных задач в профессиональной области
ЛС-3 Способен проектировать и поддерживать архитектуру систем искусственного интеллекта	ЛС-3.1 Создает и развивает архитектуры системы ИИ на всех этапах жизненного цикла
ЛС-2 Способен проводить эксперименты на данных, формулировать гипотезы исследования, строить (обучать, дообучать) модели машинного обучения с оценкой их качества и анализом ошибок, обеспечивать воспроизводимость и масштабируемость исследований на данных	ЛС-2.1 Проводит эксперименты с моделями ИИ, оценивает их качество (точность, производительность)
	ЛС-2.2 Проводит эксперименты на данных и визуализирует результаты с применением технологий анализа данных (статистического анализа), методов и алгоритмов машинного обучения

2. Показатели оценивания компетенций

В результате изучения дисциплины «Технологии программирования» обучающийся должен:

знать:

шаблоны проектирования программного обеспечения.

уметь:

работать с интерфейсом командной строки;
выполнять сборку программ из исходных текстов и их отладку, без использования интегрированных средств разработки;
пользоваться системами контроля версий;
настраивать окружение для непрерывной интеграции разработки проекта;
проектировать программное обеспечение таким образом, чтобы его поддержка осуществлялась коллективом из нескольких разработчиков.

владеть:

навыками работы с GitLab и GitLab CI.

3. Перечень типовых (примерных) вопросов, заданий, тем для подготовки к текущему контролю

Примеры контрольных заданий:

1. Написать программу, замеряющую время работы методов для нескольких стандартных реализаций интерфейсов List и Map, с использованием Dynamic Proxy.
2. Написать класс-наследник класса String, реализующий интерфейс Iterable, предоставляющий итератор со значениями типа char по своим символам.
3. Написать программу, которая скачивает 10 самых популярных страниц с сайта Хабрахабр и подсчитывает суммарный рейтинг комментариев, зарегистрированных по пользователям, с использованием сторонней библиотеки парсинга HTML.
4. Написать программу, которая предоставляет консольный интерфейс для добавления записей в базу данных книжного магазина с сущностями «Автор» и «Книга» (отношение один-ко-многим между автором и книгами), с использованием сторонней библиотеки для работы с базой данных sqlite3.

5. Написать программу, которая предоставляет консольный интерфейс для добавления записей в базу данных книжного магазина с сущностями «Автор» и «Книга» (отношение один-ко-многим между автором и книгами), с использованием сторонней библиотеки, реализующей Object-Relation Mapping.
6. Написать программу, которая запрашивает у Foursquare API и выводит популярные кафе в окрестности города, вводимого пользователем.
7. Написать программу, вычисляющую PageRank и найти ТОП 100 самых популярных страниц в википедии (русскоязычной, англоязычной).

4. Перечень типовых (примерных) вопросов и тем для проведения промежуточной аттестации обучающихся

Перечень контрольных вопросов:

1. Сборщик мусора.
2. Массивы, многомерные массивы.
3. Методы класса Object. Контракт equals()/hashCode().
4. Интерфейсы.
5. Исключения: иерархия исключений, перехват исключений, декларация throws.
6. Порядок инициализации объекта.
7. Java Reflection API. Dynamic proxy.
8. Юнит-тестирование, средства JUnit.
9. Аннотации.
10. Итераторы.
11. Сериализация средствами стандартной библиотеки.
12. Лямбда-функции.
13. TCP/IP в Java. HTTP-сервер средствами Java.
14. Многопоточность: примитивы синхронизации в языке.
15. Многопоточность: Java Memory Model.
16. Многопоточность: инструменты библиотеки java.util.concurrent.
17. Коллекции стандартной библиотеки, потокобезопасность коллекций.
18. Форматы XML, JSON, CSV.
19. Кодировки текстовых данных.
20. Система контроля версий git.

Примеры билетов:

№1

1. Юнит-тестирование, средства JUnit.
2. Написать программу, измеряющую время работы методов для нескольких стандартных реализаций интерфейсов List и Map, с использованием Dynamic Proxy.

№2

1. TCP/IP в Java. HTTP-сервер средствами Java.
2. Написать класс-наследник класса String, реализующий интерфейс Iterable, предоставляющий итератор со значениями типа char по своим символам.

Критерии оценивания

отлично

10 Полностью и вовремя решены все задачи без ошибок. Продемонстрирован грамотный подход к решению задач, реализованы оптимальные алгоритмы, код оформлен в едином удобочитаемом стиле

9 Полностью и вовремя решены все задачи без ошибок. Продемонстрирован грамотный подход к решению задач, реализованы оптимальные алгоритмы

8 Полностью и вовремя решены все задачи без ошибок. Продемонстрирован грамотный подход к решению задач

хорошо

- 7 Полностью решены все задачи. Допущены несущественные ошибки.
- 6 Полностью решено большинство задач. В некоторых задачах допущены и не исправлены ошибки, либо некоторые задачи решены частично.
- 5 Полностью решено две трети задач. В некоторых задачах допущены и не исправлены ошибки, либо некоторые задачи решены частично.
удовлетворительно
- 4 Полностью решено более половины задач. В остальных задачах допущены и не исправлены ошибки, либо некоторые задачи решены частично.
- 3 Полностью решено более половины задач.
неудовлетворительно
- 2 Решено менее половины задач.
- 1 Не решено ни одной задачи.

5. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности

Дифференцированный зачет может проводиться по итогам текущей успеваемости и сдачи заданий, лабораторных и других видов работ, предусмотренных программой дисциплины и (или) путем организации специального опроса, проводимого в устной и (или) письменной форме, а также с выдачей заданий для реализации на компьютере.