ЦЕНТРАЛЬНЫЙ УНИВЕРСИТЕТ

УТВЕРЖДЕНА

Решением Ученого совета
АНО ВО «Центральный университет»
«25» мая 2026 г.
Протокол №2

Рабочая программа дисциплины (модуля)
«Введение в низкоуровневое программирование»

Направление подготовки: *02.03.01 Математика и компьютерные науки*

Направленность (профиль) подготовки: *Прикладной искусственный интеллект*

Квалификация (степень) выпускника: *бакалавр*

Форма обучения: *очная*

Срок освоения программы: *4 года*

Год набора: *2026*

ДОКУМЕНТ ПОДПИСАН
ЭЛЕКТРОННОЙ ПОДПИСЬЮ

Сертификат: 0215B0C000D1B3FEB44A37EE631E0C27D4
Владелец: Ивашкевич Евгений Васильевич
Действителен: с 13.01.2026 до 13.04.2027

Электронный документ

Москва
2026

Содержание

1. **Краткая характеристика дисциплины (модуля)** 3
2. **Перечень планируемых результатов обучения** 5
3. **Тематический план** 10
4. **Содержание дисциплины (модуля)** 10
5. **Учебно-методическое обеспечение** 12
6. **Материально-техническое обеспечение** 12
7. **Методические и оценочные материалы** 14

1. Краткая характеристика дисциплины (модуля)

Рабочая программа дисциплины (модуля) «Введение в низкоуровневое программирование» составлена в соответствии с федеральным государственным образовательным стандартом высшего образования – бакалавриат по направлению 02.03.01 «Математика и компьютерные науки», профиль «Прикладной искусственный интеллект», утвержденный приказом Министерства науки и высшего образования Российской Федерации № 807 от 23.08.2017 года.

Изучение дисциплины (модуля) «Введение в низкоуровневое программирование» позволяет глубоко понять, как программы выполняются на аппаратном уровне, что критически важно для разработки эффективных, надёжных и безопасных систем. Полученные знания и навыки формируют ключевые компетенции для создания высокопроизводительных компонентов в машинном обучении и системном программировании, где требуется контроль над ресурсами и интеграция с низкоуровневыми библиотеками.

Место дисциплины (модуля) в структуре образовательной программы

Настоящая дисциплина (модуль) включена в учебный план по программе подготовки бакалавриата по направлению 02.03.01 «Математика и компьютерные науки», профиль «Прикладной искусственный интеллект» и входит в вариативную часть Блока 1 формируемую участниками образовательных отношений как дисциплина по выбору.

Дисциплина (модуль) изучается на 4 курсе в 7 семестре. Доступна к изучению после успешного освоения дисциплин (модулей): «Основы промышленной разработки», «Алгоритмы и структуры данных. Часть 1» и «Алгоритмы и структуры данных. Часть 2».

Цель изучения дисциплины (модуля): формирование у студентов фундаментальных знаний и навыков решения и анализа дифференциальных уравнений, необходимых для моделирования и исследования динамических процессов в различных областях науки и техники.

Задачи изучения дисциплины (модуля):

— изучить полный цикл трансляции высокоуровневого кода в машинные инструкции, включая этапы компиляции, компоновки и загрузки программ;

— освоить принципы устройства памяти процесса, механизмы управления динамической памятью и последствия типичных ошибок при работе с указателями;

— научиться разрабатывать и собирать статические и динамические библиотеки на языке Си для использования в многоплатформенных проектах;

— освоить основы многопоточного программирования и взаимодействия с операционной системой через системные вызовы;

— развить навыки интеграции C-кода с Python через CPython API, включая управление памятью, передачу данных и оптимизацию производительности критических участков кода.

В результате освоения дисциплины (модуля) обучающийся должен:

знать:

— полный цикл преобразования высокоуровневой программы в машинное представление и особенности этого преобразования;

— как исполняется программа компьютером на уровне машинного представления;

— устройство памяти процесса, механизмы выделения и освобождения памяти, а также последствия ошибок при работе с памятью;

— принципы организации системных вызовов и их роль в реализации функций языка С и взаимодействии с операционной системой;

— архитектуру взаимодействия Python и C-кода через CPython API, включая управление ссылками, типы PyObject и особенности передачи данных между языками.

уметь:

— разрабатывать программы на языке C;

— разрабатывать мультиязыковые проекты и собирать их под ОС Linux;

— разрабатывать статические и динамические библиотеки;

— оптимизировать программы;

— работать на удалённых компьютерах посредством использования SSH;

— разрабатывать многопоточный код;

— понимать байт-код и вносить в него правки.

владеть:

— навыком разработки примитивных программ в низкоуровневом представлении;

— навыком оптимизации программы на Python через реализацию части функционала в низкоуровневом представлении;

— навыком отладки и профилирования производительности C-расширений для Python;

— навыком безопасной и эффективной работы с указателями, массивами и структурами данных в C, включая реализацию алгоритмов, критичных к скорости и потреблению памяти;

— навыком интеграции низкоуровневых компонентов в ML-пайплайны, включая сериализацию данных, работу с файлами и внешними процессами, а также обеспечение надёжности и переносимости кода.

2. Перечень планируемых результатов обучения

Компетенции, формируемые в результате освоения дисциплины (модуля) при проведении учебных занятий в форме контактной работы обучающихся с педагогическими работниками Университета и в форме самостоятельной работы обучающихся:

Код и содержание компетенции	Индикатор компетенции и перечень планируемых результатов обучения по дисциплине (модулю)
ППК-Р4. Способен использовать базы данных при создании программных модулей и компонентов	ППК-Р4.1. Пишет программный код с использованием языков определения и манипулирования данными в базах данных (базовый) ППК-Р4.1. З-1. Знает архитектуры современных систем управления баз данных, включая SQL и noSQL ППК-Р4.1. У-1. Умеет применять выбранные языки работы с базами данных
	ППК-Р4.2. Проектирует базы данных для программных модулей и компонентов (средний) ППК-Р4.2. З-1. Знает современные подходы к проектированию реляционных и нереляционных баз данных ППК-Р4.2. У-2. Умеет использовать выбранную среду программирования для работы с данными в базе.
	ППК-Р4.3. Оптимизирует производительность работы с базами данных (средний) ППК-Р4.3. З-1. Знает внутреннее устройство СУБД выбранной архитектуры ППК-Р4.3. У-2. Умеет вырабатывать варианты оптимизации производительности запросов в базе данных
ППК-Р5. Способен разрабатывать требования и проектировать программное обеспечение	ППК-Р5.1. Анализирует возможности реализации требований к компьютерному программному обеспечению (средний) ППК-Р5.1. З-1. Знает возможности существующей программно-технической архитектуры ППК-Р5.1. У-3. Умеет вырабатывать варианты реализации требований к компьютерному программному обеспечению
	ППК-Р5.2. Разрабатывает технические спецификаций на программные компоненты и их взаимодействие (средний) ППК-Р5.2. З-2. Знает методы и средства проектирования программных интерфейсов ППК-Р5.2. У-1. Умеет выбирать средства реализации требований к компьютерному программному обеспечению
	ППК-Р5.3. Проектирует компьютерное программное обеспечение (продвинутый) ППК-Р5.3. З-1. Знает принципы построения и виды архитектуры компьютерного программного обеспечения ППК-Р5.3. У-2. Умеет проектировать структуры данных
	ППК-Р5.4. Выполняет логическое проектирование системы (продвинутый) ППК-Р5.4. З-3. Знает базовые технологии взаимодействия и интеграции систем и компонентов ППК-Р5.4. У-1. Умеет описывать интеграции со смежными системами на логическом уровне
ППК-Р6. Способен участвовать в промышленной разработке программного обеспечения	ППК-Р6.1. Работает в соответствии с промышленными методологиями разработки (средний) ППК-Р6.1. З-1. Знает принципы Agile и их применение в промышленных проектах ППК-Р6.1. У-2. Умеет работать в команде с использованием инструментов управления проектами

	ППК-Р6.2. Использует инструменты промышленной разработки (средний) ППК-Р6.2. З-1. Знает принципы Continuous Integration and Continuous Delivery (CI/CD) ППК-Р6.2. У-1. Умеет настраивать потоки работ CI/CD
	ППК-Р6.3. Разрабатывает масштабируемый и поддерживаемый код (продвинутый) ППК-Р6.3. З-1. Знает принципы чистого кода, SOLID, DRY, KISS и др. ППК-Р6.3. У-1. Умеет разрабатывать модульный и тестируемый программный код
	ППК-Р6.4. Участвует в развертывании и поддержке программного обеспечения (продвинутый) ППК-Р6.4. З-1. Знает стратегии развертывания промышленного программного обеспечения ППК-Р6.4. У-1. Умеет развертывать приложения в облаке
ППК-Р7. Способен применять искусственный интеллект (ИИ) для генерации и отладки программного кода	ППК-Р7.1. Применяет ИИ-инструменты для генерации программного кода (базовый) ППК-Р7.1. З-1. Знает принципы работы современных генеративных ИИ-моделей для генерации кода ППК-Р7.1. У-1. Умеет формулировать корректные текстовые запросы (промты) для генерации кода
	ППК-Р7.2. Использует ИИ для анализа и отладки кода (средний) ППК-Р7.2. З-1. Знает методы ИИ-анализа кода ППК-Р7.2. У-2. Умеет интерпретировать рекомендации ИИ по исправлению кода
	ППК-Р7.3. Оптимизирует код с помощью ИИ (продвинутый) ППК-Р7.3. З-1. Знает методы ИИ-оптимизации ППК-Р7.3. У-2. Умеет проверять корректность оптимизаций, предложенных ИИ
	ППК-Р7.4. Оценивает этические и профессиональные аспекты применения ИИ в разработке (продвинутый) ППК-Р7.4. З-1. Знает этические нормы использования ИИ (конфиденциальность, плагиат кода и т.п.) ППК-Р7.4. У-2. Умеет документировать использование ИИ в разработке
ППК-ДА4. Способен выявлять бизнес-проблемы или бизнес-возможности	ППК-ДА4.1. Собирает информацию о бизнес-проблемах или бизнес-возможностях (средний) ППК-ДА4.1. З-3. Знает инструменты, техники анализа бизнес-ситуации и предметной области ППК-ДА4.1. У-1. Умеет собирать, классифицировать, систематизировать и обеспечивать хранение и актуализацию информации для бизнес-анализа
	ППК-ДА4.2. Выявление истинных бизнес-проблем или бизнес-возможностей (продвинутый) ППК-ДА4.2. З-2. Знает инструменты, техники анализа бизнес-ситуации и предметной области, включая анализ данных ППК-ДА4.2. У-1. Умеет выявлять и классифицировать бизнес-проблемы или бизнес-возможности
ППК-ДА5. Способен применять методы статистического анализа и теорию эксперимента для планирования, проведения и	ППК-ДА5.1. Разрабатывает дизайн эксперимента (базовый) ППК-ДА5.1. З-1. Знает основные принципы планирования экспериментов ППК-ДА5.1. У-2. Умеет выбирать метрики для оценки эффекта воздействия

интерпретации результатов экспериментов	ППК-ДА5.2. Проводит статистический анализ данных эксперимента (средний) ППК-ДА5.2. З-1. Знает методы проверки гипотез (t-тест, χ^2, ANOVA) ППК-ДА5.2. У-2. Умеет интерпретировать p-value и уровень значимости
	ППК-ДА5.3. Интерпретирует результаты экспериментов (продвинутый) ППК-ДА5.3. У-2. Умеет формулировать рекомендации на основе статистических выводов
МО-2. Способен проектировать и внедрять системы автоматизации жизненного цикла машинного обучения (MLOps)	МО-2.1. Проектирует и настраивает CI/CD-пайплайны для машинного обучения (базовый) МО-2.1. З-1. Знает принципы CI/CD применительно к моделям машинного обучения МО-2.1. У-1. Умеет настраивать CI/CD-пайплайны для МО с использованием инструментов
	МО-2.2. Организует версионирование моделей, данных и экспериментов (средний) МО-2.2. З-1. Понимает системы версионирования моделей и данных МО-2.2. У-2. Умеет использовать инструменты MLOps для воспроизводимости и масштабируемости экспериментов
	МО-2.3. Внедряет системы мониторинга моделей в продуктивную эксплуатацию (средний) МО-2.3. З-1. Знает метрики мониторинга моделей МО МО-2.3. У-1. Умеет развертывать системы мониторинга моделей МО
	МО-2.4. Автоматизирует процессы переобучения и развертывания моделей (продвинутый) МО-2.4. З-1. Знает стратегии автоматизации переобучения МО-2.4. У-1. Умеет автоматизировать переобучение моделей и A/B-тестирование развертываний
РГ-1. Способен руководить процессами разработки компьютерного программного обеспечения	РГ-1.1. Руководит разработкой программного кода (средний) РГ-1.1. У-1. Способен распределять задачи на разработку программного кода между исполнителями РГ-1.1. У-3. Оценивает качество и эффективность программного кода
	РГ-1.2. Руководит проверкой работоспособности компьютерного программного обеспечения (средний) РГ-1.2. У-2. Умеет оценивать результаты проверки работоспособности компьютерного программного обеспечения
	РГ-1.3. Руководит интеграцией программных модулей и компонентов (продвинутый) РГ-1.3. З-2. Знает методы и программные интерфейсы взаимодействия компьютерного программного обеспечения с внешними программными компонентами
РГ-2. Способен организовать процессы разработки компьютерного программного обеспечения	РГ-2.1. Управляет проектированием компьютерного программного обеспечения (средний) РГ-2.1. З-1. Знает принципы построения архитектуры компьютерного программного обеспечения и виды архитектуры программного обеспечения
	РГ-2.2. Управляет процессом разработки компьютерного программного обеспечения (продвинутый) РГ-2.2. У-1. Умеет составлять планы процесса разработки программного продукта

	РГ-2.3. Управляет запросами на изменения, дефектами и проблемами (продвинутый) РГ-2.3. З-1. Знает методы и средства выявления дефектов, проблем и причин их возникновения в компьютерном программном обеспечении
	РГ-2.4. Управляет конфигурациями и выпусками программного продукта (продвинутый) РГ-2.5. У-1. Умеет управлять версиями отдельных компонентов и программного продукта в целом
РГ-4. Способен организовать промышленную разработку программного обеспечения	РГ-4.1. Внедряет промышленные методологии разработки (средний) РГ-4.1. З-1. Знает принципы Agile и их применение в промышленных проектах
	РГ-4.2. Внедряет инструменты промышленной разработки (средний) РГ-4.2. З-1. Знает принципы Continuous Integration and Continuous Delivery (CI/CD)
	РГ-4.3. Организует разработку масштабируемого и поддерживаемого кода (продвинутый) РГ-4.3. У-1. Умеет управлять разработкой модульного и тестируемого программного кода
	РГ-4.4. Организует развертывание и поддержку программного обеспечения (продвинутый) РГ-4.4. У-1. Умеет организовать развертывание приложения в облаке
РГ-5. Способен организовать процессы применения искусственного интеллекта (ИИ) для генерации и отладки программного кода	РГ-5.1. Организует применение ИИ-инструментов для генерации программного кода (средний) РГ-5.1. З-1. Знает принципы работы современных генеративных ИИ-моделей для генерации кода
	РГ-5.2. Организует применение ИИ для анализа и отладки кода (продвинутый) РГ-5.2. З-1. Знает методы ИИ-анализа кода
	РГ-5.3. Организует оптимизацию кода с помощью ИИ (продвинутый) РГ-5.3. З-1. Знает методы ИИ-оптимизации
	РГ-5.4. Оценивает этические и профессиональные аспекты применения ИИ в разработке (продвинутый) РГ-5.4. З-1. Знает этические нормы использования ИИ (конфиденциальность, плагиат кода и т.п.)
УК-4. (УНК-01.) Способен осуществлять деловую коммуникацию в устной и письменной формах на государственном языке Российской Федерации и иностранном(ых) языке(ах) *(соответствует УНК-01 – Способен применять коммуникационные компетенции в профессиональной деятельности)*	УНК-01.01. Осуществляет коммуникации в профессиональной деятельности (базовый) УНК-01.01.01. У-1 Владеет навыками презентации и публичной дискуссии УНК-01.01.01. У-3 Способен формулировать и понимать технологические и бизнес-требования
	УНК-01.02. Демонстрирует владение профессиональной культурой (базовый) УНК-01.02.01. У-3. Умеет грамотно оформлять документацию и вести коммуникацию в соответствии с профессиональными стандартами
УК-3. (УНК-02.) Способен осуществлять социальное взаимодействие и реализовывать свою роль в команде	УНК-02.01. Принимает участие в групповом взаимодействии (базовый) УНК-02.01.01. У-5. Умеет работать в команде, проявлять инициативу и поддерживать коллег
	УНК-02.02. Проявляет лидерство и осуществляет

(соответствует УНК-02 – Способен применять методы коллективной работы в профессиональной деятельности)	наставничество (базовый) УНК-02.02.01. У-3. Умеет делиться знаниями и опытом, оказывать поддержку и осуществлять наставничество коллегам

3. Тематический план

№п/п	Наименование раздела дисциплины (модуля)	Трудоемкость, академические часы				ТКУ (текущий контроль успеваемости)
		Очная форма				
		Аудиторная работа		Контроль	Самостоятельная работа	
		Лекции	Практические занятия			
1	Введение в язык программирования Си и основы низкоуровневого программирования	8	8		8	Домашние задания, Подготовка к семинару
2	Разработка промышленных проектов на примере разработки проектов на языке Си и Python для ОС Linux	8	6	2	8	Домашние задания, Контрольная работа
3	Низкоуровневое представление программ: компилируемые на примере программ на языке Си, интерпретируемые на примере программ на языке программирования Python	14	12	2	16	Домашние задания, Контрольная работа
	Зачёт с оценкой			4		
	Итого:	***30***	***30***	***8***	***122***	
Объем дисциплины (модуля) (в ак. ч.)		***190***				
Объем дисциплины (модуля) (в зач. ед.)		***5***				

4. Содержание дисциплины (модуля)

№п/п	Наименование раздела дисциплины (модуля)	Содержание дисциплины (модуля) по темам
1	Введение в язык программирования Си и основы низкоуровневого программирования	Введение в низкоуровневое программирование для ML. Структура программы на языке Си Простая программа на языке Си. Среда разработки, компиляция и запуск программы на языке Си под ОС Linux. Типы данных и низкоуровневое представление целочисленной и вещественной арифметики. Массивы, структуры и работа с памятью
2	Разработка промышленных проектов на примере разработки проектов на языке Си и Python для ОС Linux	CPython - интерпретатор языка Python. Расширение Python с помощью Си (вызов Си подпрограмм). Частичная интерпретация Расширение Python с помощью Си. Работа с файлами (чтение и запись файлов) Промышленный проект: основная программа, статические и динамические библиотеки, сборка и тестирование. Компиляция программ, интерпретация программ, устройство байт-кода

3	Низкоуровневое представление программ: компилируемые на примере программ на языке Си, интерпретируемые на примере программ на языке программирования Python	Системные вызовы Процессы и средства взаимодействия между ними Выполнение внешних программ Оптимизация по времени выполнения программ на языке Python Многопоточность Сетевое программирование Подведение итогов курса: применение низкоуровневого программирования в МО на примерах оптимизации

5. Учебно-методическое обеспечение

Университет располагает полным набором лицензионного и свободно распространяемого программного обеспечения, включая продукты отечественного производства.

Каждый студент в течение всего периода обучения получает индивидуальный неограниченный доступ к электронно-библиотечной системе и электронной информационно-образовательной среде университета. Эти системы предоставляют возможность доступа к ресурсам из любой точки, где есть подключение к сети Интернет, как на территории университета, так и за его пределами.

Студентам обеспечен удаленный доступ к современным профессиональным базам данных и информационным справочным системам.

Основная литература:

1. 1. Таненбаум, Э. С. Компьютерные сети / Э. С. Таненбаум, Д. Уэзеролл. - 5-е изд. - Санкт-Петербург : Питер, 2021. - 960 с. - ISBN 978-5-4461-1248-7. - Текст : электронный. - URL: https://znanium.ru/catalog/product/2141427.

2. 2. Кузин, А. В. Программирование на языке Си : учебное пособие / А.В. Кузин, Е.В. Чумакова. — Москва : ФОРУМ : ИНФРА-М, 2024. — 143 с. — (Среднее профессиональное образование). - ISBN 978-5-00091-556-1. - Текст : электронный. - URL: https://znanium.ru/catalog/product/2137197.

Дополнительная литература:

1. Рабчевский, А. Н. Компьютерные сети и системы связи. Вводный курс : учебное пособие для вузов / А. Н. Рабчевский. — Москва : Издательство Юрайт, 2025. — 207 с. — (Высшее образование). — ISBN 978-5-534-21489-5. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/572633.

6. Материально-техническое обеспечение

Университет располагает материально-технической базой, соответствующей действующим противопожарным правилам и нормам и обеспечивающей проведение всех видов дисциплинарной и междисциплинарной подготовки, практической и научно-исследовательской работ обучающихся, предусмотренных учебным планом.

Помещения, которые представляют собой учебные аудитории для проведения занятий лекционного типа, занятий семинарского (практического) типа, групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации, а также помещения для самостоятельной работы и помещения для хранения и профилактического обслуживания учебного оборудования. Помещения укомплектованы специализированной мебелью и техническими средствами обучения, служащими для представления учебной информации большой аудитории.

Изучение дисциплины (модуля) обеспечивается в учебных аудиториях, оснащенных:

- столами и стульями;
- компьютерной техникой;
- механическими калькуляторами;
- специализированным оборудованием, включая демонстрационное оборудование.

Помещения для самостоятельной работы обучающихся, в том числе приспособленные для использования инвалидами и лицами с ограниченными возможностями здоровья, оснащены компьютерной техникой с возможностью подключения к сети «Интернет» и обеспечением доступа в электронную информационно-образовательную среду Университета.

Обучающимся предоставляется доступ (в том числе удаленный) к ресурсам информационно-телекоммуникационной сети «Интернет», электронным ресурсам (в том числе электронным библиотечным системам, современным профессиональным базам данных и информационным справочным системам):

№	Наименование портала (издания, курса, документа)	Ссылка
1.	Научная электронная библиотека elibrary.ru библиотека	https://elibrary.ru/defaultx.asp
2.	База данных для IT-специалистов	https://habr.com
3.	База данных ScienceDirect	https://www.sciencedirect.com
4.	Официальный сайт Министерства науки и высшего образования Российской Федерации	https://minobrnauki.gov.ru/
5.	Федеральный портал «Российское образование"	https://www.edu.ru/
6.	Информационная система "Единое окно доступа к образовательным ресурсам"	http://window.edu.ru/
7.	Единая коллекция цифровых образовательных ресурсов	http://school-collection.edu.ru/
8.	Федеральный центр информационно - образовательных ресурсов	http://fcior.edu.ru/

Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю), в том числе комплект лицензионного программного обеспечения, современные профессиональные базы данных и информационные справочные системы:

Наименование ПО	Производство	Лицензионное / свободно распространяемое
Операционные системы:		
Microsoft Imagine (Windows Client, Server)	зарубежное	лицензионное
Браузеры:		
Яндекс.Браузер	отечественное	свободно распространяемое
Google Chrome	зарубежное	свободно распространяемое
Офисные приложения:		
Microsoft Imagine (Visio, OneNote)	зарубежное	лицензионное
TeXstudio	зарубежное	свободно распространяемое
Adobe Acrobat Reader	зарубежное	свободно распространяемое
Программное обеспечение для планирования и учета времени:		
Toggle app	зарубежное	свободно распространяемое
Системы управления проектами:		
Microsoft Imagine (Project)	зарубежное	лицензионное
Системы управления базами данных:		
Microsoft Imagine (SQL Server)	зарубежное	лицензионное
Системы резервного копирования (backup):		
Acronis Backup Advanced for HyperV	зарубежное	лицензионное
Справочно-правовые системы:		
КонсультантПлюс: справочно-правовая система	отечественное	лицензионное
Средства антивирусной защиты:		
Kaspersky Endpoint Security для бизнеса Стандартный Russian Edition	отечественное	лицензионное
Среды разработки:		
Visual Studio Code	зарубежное	свободно распространяемое
Bash (Unix shell)	зарубежное	свободно распространяемое

Anaconda	зарубежное	свободно распространяемое
Robotic Operating System	зарубежное	свободно распространяемое
CopelliaSim	зарубежное	свободно распространяемое
Google Colaboratory	зарубежное	свободно распространяемое
Пакеты программных средств и библиотек:		
AutoPsy	зарубежное	свободно распространяемое
Interactive Disassembler (IDA)	зарубежное	свободно распространяемое
Системы управления библиографической информацией:		
Zotero	зарубежное	свободно распространяемое
Сервисы и службы:		
Bind	зарубежное	свободно распространяемое
Docker	зарубежное	свободно распространяемое

7. Методические и оценочные материалы

Методические указания для обучающихся по освоению дисциплины (модуля)

В процессе изучения дисциплины (модуля) «Введение в низкоуровневое программирование» в рамках текущего контроля успеваемости используются такие виды учебной работы, как лекции, семинары, домашние задания, контрольная работа, а также различные виды самостоятельной работы обучающихся по заданию преподавателя, направленные на развитие навыков профессиональной лексики, закрепление практических профессиональных компетенций, поощрение инициатив.

Лекция – систематическое, последовательное, монологическое изложение преподавателем учебного материала, как правило, теоретического характера.

В процессе лекций рекомендуется вести конспект лекций: кратко и схематично фиксировать основные идеи, выводы и обобщения лекции; выделять важные мысли, ключевые слова и термины. Необходимо отметить вопросы или материалы, которые вызывают затруднения, и попытаться найти ответы в рекомендованной литературе. Если разобраться в материале не удается, следует сформулировать вопрос и задать его преподавателю на консультации или во время семинарского (практического) занятия.

Семинар – это форма учебной деятельности, проводимая в учебном заведении под руководством преподавателя, где студенты активно участвуют в обсуждениях, практических заданиях и других формах взаимодействия.

Для успешной подготовки к семинару рекомендуется заранее ознакомиться с темой занятия и основными материалами, чтобы иметь возможность активно участвовать в обсуждении. Также полезно подготовить вопросы и идеи для обсуждения, что поможет глубже понять материал и продемонстрировать заинтересованность.

Аудиторная работа – активная работа студента на семинаре, его ответы на вопросы преподавателя и участие в дискуссии.

Для успешного участия в семинаре студентам рекомендуется заранее ознакомиться с темой обсуждения, прочитать необходимые материалы и подготовить вопросы. Важно активно слушать и вовлекаться в дискуссию, высказывая свои мнения и аргументируя их. При ответах на вопросы преподавателя стоит быть уверенным, четким и логичным, опираясь на изученный материал. Также полезно поддерживать диалог с однокурсниками, чтобы обогатить обсуждение и расширить свои знания.

Домашнее задание – набор задач по темам недели.

При работе над домашними заданиями важно внимательно ознакомиться с требованиями и сроками выполнения. Рекомендуется разбивать задания на этапы, чтобы избежать перегрузки и лучше усвоить материал. Использовать различные источники

информации, включая учебники и онлайн-ресурсы, для более глубокого понимания темы.

Контрольная работа – письменная работа с набором задач, которые нужно решить за ограниченное время.

Цель контрольной работы – получить специальные знания по одной или нескольким темам дисциплины и продемонстрировать навыки их практического применения.

Самостоятельная работа – работа студентов, направленная на углубленное изучение отдельных тем и вопросов учебной дисциплины (модуля).

В процессе самостоятельной работы студенты взаимодействуют с рекомендованными материалами при минимальном участии преподавателя. Задачи студента включают работу с конспектами лекций (обработка текста), повторное изучение учебных материалов планов и тезисов ответов, изучение дополнительных тем, выполнение учебно-исследовательских заданий и другое.

Система оценивания результатов обучения по дисциплине (модулю)

Критерии получения уровня и оценивания сформированности компетенций по дисциплине (модулю) «Введение в низкоуровневое программирование»

Оценивание уровня учебных достижений, обучающихся по дисциплине (модулю), осуществляется в виде текущего контроля успеваемости и промежуточной аттестации.

Промежуточная аттестация по дисциплине (модулю) осуществляется в форме ***зачёта с оценкой*** при этом проводится оценка компетенций, сформированных по дисциплине.

Для оценивания текущего контроля успеваемости и промежуточной аттестации используется десятибалльная шкала оценивания, которая соотносится с традиционной пятибалльной шкалой следующим образом, если иное не указано в силлабусе дисциплины:

Десятибалльная оценка	**Пятибалльная оценка**	**Общая характеристика результата обучения по дисциплине (модулю)**
10	Отлично	Студент полностью владеет знаниями, изложенными в рабочей программе, и глубоко осмысляет дисциплину. Он самостоятельно и логически последовательно отвечает на все вопросы, акцентируя внимание на наиболее важном. Умеет анализировать, сравнивать, классифицировать, обобщать, конкретизировать и систематизировать изученный материал, выделяя ключевые моменты и устанавливая причинно-следственные связи. Четко формулирует ответы, уверенно интерпретирует результаты анализов и других исследований, а также решает сложные задачи. Студент хорошо знаком с методами исследования, необходимыми для практической деятельности, и умеет связывать теоретические аспекты дисциплины (модуля) с практическими задачами.
9	Отлично	
8	Отлично	
7	Хорошо	Студент обладает знаниями предмета почти в полном объеме рабочей программы и самостоятельно, логически последовательно и всесторонне отвечает на все вопросы, акцентируя внимание на наиболее значимых моментах. Он умеет анализировать, сравнивать, классифицировать,
6	Хорошо	

Десятибалльная оценка	Пятибалльная оценка	Общая характеристика результата обучения по дисциплине (модулю)
		обобщать, конкретизировать и систематизировать изученный материал, выделяя его ключевые аспекты и устанавливая причинно-следственные связи. Формулирует свои ответы, уверенно интерпретирует результаты анализов и других исследований, а также решает сложные ситуационные задачи. Студент хорошо знаком с методами исследования, необходимыми для практической деятельности, и умеет связывать теоретические аспекты предмета с практическими задачами.
5	Удовлетворительно	Студент обладает базовыми знаниями по дисциплине (модулю), но испытывает трудности при самостоятельных ответах и использует неточные формулировки. В ходе ответов он допускает ошибки, касающиеся сути вопросов. Студент способен решать только самые простые задачи и владеет лишь минимальным набором методов исследования.
4	Удовлетворительно	
3	Не сдан	Студент не овладел обязательным минимумом знаний по предмету и не может ответить на вопросы, даже если преподаватель задает дополнительные наводящие вопросы.
2	Не сдан	
1	Не сдан	

Дисциплина (модуль) «Введение в низкоуровневое программирование» оценивается следующим образом:

Активность	Вес	Количество	Описание
Домашние задания	15%	12	Набор задач по темам недели
Аудиторная работа	35%	1	Активная работа студента на семинаре, его ответы на вопросы преподавателя и участие в дискуссии
Контрольная работа	20%	2	Письменная работа с набором задач, которые нужно решить за ограниченное время
Зачёт с оценкой	30%	1	Письменная или устная работа над заданием, направленным на проверку полученных знаний и навыков по дисциплине (модулю)

Формула расчёта итоговой оценки по дисциплине (модулю) «Введение в низкоуровневое программирование»: «0,15 × среднее за домашние задания + 0,35 × среднее за аудиторную работу + 0,2 × среднее за контрольные работы + 0,3 × зачёт с оценкой».

Текущий контроль успеваемости обучающихся по дисциплине (модулю)

Примерные домашние задания

Домашнее задание 1.

1. Напишите простую программу на языке C, которая выводит «Hello, ML!» и скомпилируйте её в терминале под ОС Linux с помощью gcc;
2. Объясните разницу между типами данных int, float и double в языке C и опишите их битовое представление в памяти;

3. Реализуйте программу, которая создаёт массив целых чисел, заполняет его значениями и выводит адреса элементов, демонстрируя принципы адресации в памяти;

4. Напишите функцию на C, принимающую массив и его размер, и возвращающую сумму элементов с использованием указателей;

5. Опишите, как устроена память процесса в ОС Linux (стек, куча, сегменты кода и данных) и в чём разница между статическим и динамическим выделением памяти.

Домашнее задание 2.

1. Напишите простую программу на языке C, которая выводит «Hello, ML!» и скомпилируйте её в терминале под ОС Linux с помощью gcc;

2. Объясните разницу между типами данных int, float и double в языке C и опишите их битовое представление в памяти;

3. Реализуйте программу, которая создаёт массив целых чисел, заполняет его значениями и выводит адреса элементов, демонстрируя принципы адресации в памяти;

4. Напишите функцию на C, принимающую массив и его размер, и возвращающую сумму элементов с использованием указателей;

5. Опишите, как устроена память процесса в ОС Linux (стек, куча, сегменты кода и данных) и в чём разница между статическим и динамическим выделением памяти.

Домашнее задание 3.

1. Напишите многопоточную программу на C с использованием pthread, в которой каждый поток вычисляет часть суммы массива, и сравните время выполнения с однопоточной версией;

2. Реализуйте программу на Python, которая запускает внешние команды ОС Linux через subprocess и обрабатывает их вывод;

3. Напишите C-функцию, оптимизирующую вычислительно сложную операцию (например, скалярное произведение), и интегрируйте её в ML-пайплайн для ускорения обработки данных;

4. С помощью системных вызовов (через C или Python) получите информацию о процессе: PID, использование памяти и время работы;

5. Сравните производительность Python-реализации алгоритма и её C-аналога, используя профилировщики (cProfile, perf), и объясните разницу с точки зрения низкоуровневого исполнения.

Примерные задания для контрольной работы

1. Напишите функцию на языке C, вычисляющую сумму элементов массива, и скомпилируйте её в статическую библиотеку (.a) под ОС Linux;

2. Создайте динамическую библиотеку (.so) на языке C, содержащую функцию для вычисления факториала, и подключите её к программе на Python с помощью модуля ctypes;

3. Реализуйте расширение Python на языке C с использованием CPython API: опишите функцию, принимающую список чисел в виде PyObject и возвращающую их среднее значение;

4. Объясните, как управляется счётчик ссылок в CPython при передаче PyObject между C и Python, и приведите пример инкремента и декремента ссылок;

5. Напишите Makefile для автоматической сборки проекта, включающего основную программу на C, статическую библиотеку и Python-скрипт, использующий C-расширение;

6. Реализуйте функцию на C, которая читает данные из текстового файла и возвращает массив значений, затем вызовите её из Python и отобразите результат;

7. С помощью модуля dis проанализируйте байт-код простой функции Python и объясните назначение основных инструкций (LOAD_CONST, BINARY_ADD, RETURN_VALUE);

8. Опишите различия между компиляцией программы на C и интерпретацией кода Python, включая этапы преобразования в машинный код и байт-код;

9. Напишите Python-программу, которая вызывает внешнюю C-программу через subprocess, передаёт ей аргументы и обрабатывает вывод;

10. Объясните, в чём заключаются преимущества использования динамических библиотек по сравнению со статическими при разработке мультиязыковых проектов в среде Linux.

Примерные вопросы для подготовки к занятию

Основы языка C и низкоуровневое программирование

1. Опишите структуру простой программы на языке C: обязательные элементы, функция main и возвращаемое значение;

2. В чём разница между типами данных int, float и double в языке C и как они представлены в памяти на уровне битов;

3. Что такое указатель в языке C и как с его помощью осуществляется доступ к элементам массива;

4. Объясните, как устроена память процесса в ОС Linux: стек, куча, сегменты кода и данных;

5. В чём отличие статического и динамического выделения памяти в C? Приведите примеры использования malloc и free;

6. Как работают структуры в C? Напишите пример объявления и инициализации структуры для хранения информации о точке в двумерном пространстве;

7. Что такое разыменование указателя и каковы последствия обращения к неинициализированному или освобождённому указателю;

8. Объясните, зачем используется заголовочный файл (.h) и как он связан с файлом реализации (.c).

Разработка мультиязыковых проектов на C и Python

9. Как расширить функциональность Python с помощью C-кода? Опишите общий подход с использованием CPython API;

10. Что такое статическая библиотека (.a) и как её создать и подключить к проекту на C;

11. Как создать динамическую библиотеку (.so) и использовать её в Python через модуль ctypes;

12. Объясните, как управляется счётчик ссылок в CPython при передаче PyObject из C в Python и обратно;

13. Как передать массив из Python в C-функцию и корректно обработать его в низкоуровневом коде;

14. Опишите, как организовать чтение и запись файлов в C и как интегрировать эту логику в Python-скрипт;

15. Что такое Makefile и для чего он используется при сборке промышленных проектов на C;

16. Как устроен байт-код Python и с помощью какого модуля можно его проанализировать; объясните назначение инструкций LOAD_GLOBAL, CALL_FUNCTION, RETURN_VALUE.

Низкоуровневое представление и системные аспекты

17. Что такое системный вызов и как он используется для взаимодействия программы с операционной системой;

18. В чём разница между процессом и потоком? Как создать поток в C с использованием pthread;

19. Как выполнить внешнюю программу из Python и получить её вывод с помощью модуля subprocess;

20. Объясните, как многопоточность в C может использоваться для ускорения вычислительно сложных задач в ML-пайплайнах;

21. Какие средства межпроцессного взаимодействия (IPC) доступны в ОС Linux (например, pipe, shared memory);

22. В чём преимущества реализации критически важных участков кода на C при работе с Python в задачах машинного обучения;

23. Как с помощью профилировщика cProfile и системной утилиты perf можно выявить узкие места в производительности Python-кода;

24. Опишите, как сетевое программирование на C (с использованием сокетов) может быть интегрировано в Python-приложение;

25. Как оптимизация по времени выполнения Python-кода достигается за счёт замены узких участков на C-реализации; приведите пример;

26. Что такое частичная интерпретация и как она используется в реализации CPython;

27. Как обеспечить переносимость и надёжность C-расширений при их использовании в различных средах выполнения;

28. Объясните, зачем используется сериализация данных при взаимодействии между Python и C, и какие форматы для этого подходят;

29. Какие риски возникают при неправильной работе с памятью в C и как они влияют на стабильность Python-приложений;

30. Приведите пример оптимизации ML-пайплайна с помощью низкоуровневого кода и оцените возможный выигрыш в производительности.

Задания для промежуточной аттестации по дисциплине (модулю)

№ п/п	Задание	Ответ	Компетенция	Индикатор
1.	Определите основные компоненты структуры программы на языке C.	Директива #include / Функция main / Объявление переменных / Операторы	ППК-Р5	ППК-Р5.1. З-1
2.	Укажите, какая команда используется для компиляции программы на языке C в ОС Linux.	gcc -o program program.c	ППК-Р6	ППК-Р6.2. У-1
3.	Объясните, в чём разница между типами данных int и float в языке C.	int — целые числа / float — числа с плавающей точкой одинарной точности	ППК-Р5	ППК-Р5.1. У-3
4.	Назовите функцию, с которой начинается выполнение любой программы на C.	main	ППК-Р5	ППК-Р5.1. З-1
5.	Укажите, как выделить динамическую память под массив из 10 целых чисел в C.	malloc(10 * sizeof(int))	ППК-Р5	ППК-Р5.3. У-2
6.	Объясните, зачем используется указатель в языке C.	Для хранения адреса переменной и прямого доступа к памяти	ППК-Р5	ППК-Р5.3. З-1

7.	Назовите сегмент памяти, в котором размещаются локальные переменные функции.	Стек (stack)	ППК-Р5	ППК-Р5.3. З-1
8.	Определите, что такое разыменование указателя в C.	Получение значения по адресу, на который указывает указатель	ППК-Р5	ППК-Р5.3. У-2
9.	Укажите, как создать статическую библиотеку (.a) из объектных файлов в Linux.	ar rcs libname.a file1.o file2.o	ППК-Р6	ППК-Р6.2. У-1
10.	Объясните, как подключить динамическую библиотеку (.so) к программе на Python.	Через модуль ctypes или cffi	ППК-Р5	ППК-Р5.4. У-1
11.	Назовите интерпретатор языка Python, реализованный на C.	CPython	ППК-Р5	ППК-Р5.1. З-1
12.	Укажите, как с помощью Makefile автоматизировать сборку C-проекта.	Определить цели, зависимости и команды компиляции в Makefile	ППК-Р6	ППК-Р6.2. З-1
13.	Объясните, как передать массив из Python в C-функцию через ctypes.	Создать массив c_float или c_int и передать как указатель	ППК-Р5	ППК-Р5.4. У-1
14.	Назовите функцию CPython API для увеличения счётчика ссылок PyObject.	Py_INCREF	ППК-Р5	ППК-Р5.3. З-1
15.	Укажите, как прочитать данные из файла в C с помощью fopen и fgets.	Открыть файл в режиме "r" / Построчно считать содержимое	ППК-Р4	ППК-Р4.1. У-1
16.	Объясните, зачем используется заголовочный файл (.h) в проектах на C.	Для объявления функций, структур и макросов, используемых в нескольких файлах	ППК-Р5	ППК-Р5.2. З-2
17.	Назовите модуль Python, позволяющий анализировать байт-код функций.	dis	ППК-Р5	ППК-Р5.1. У-3
18.	Укажите, как выполнить внешнюю команду Linux из Python и получить её вывод.	subprocess.run или os.system	ППК-Р5	ППК-Р5.4. У-1
19.	Объясните, как создать поток в C с использованием библиотеки pthread.	Вызвать pthread_create с указанием функции потока и аргументов	ППК-Р5	ППК-Р5.3. У-2
20.	Назовите преимущество использования C-расширений для оптимизации производительности в ML.	Значительное ускорение вычислительно сложных операций	ППК-ДА5	ППК-ДА5.3. У-2

21.	Какой тип базы данных подходит для хранения метаданных о скомпилированных артефактах?	PostgreSQL / SQLite / MongoDB для метаданных сборки	ППК-Р4	ППК-Р4.1. З-1
22.	Как оптимизировать запросы к базе данных при отслеживании версий сборок?	Индексирование по хэшу коммита / Партиционирование по дате	ППК-Р4	ППК-Р4.3. У-2
23.	Какой принцип архитектуры важен при проектировании CPython-расширений?	Модульность / Изоляция памяти / Безопасность типов	ППК-Р5	ППК-Р5.3. З-1
24.	Как описывается интеграция C-библиотеки с Python-кодом на логическом уровне?	API-документация / Схема вызовов / Типы данных / Ошибки	ППК-Р5	ППК-Р5.4. У-1
25.	Какая стратегия развёртывания подходит для приложений с C-расширениями?	Контейнеризация / Wheels / Binary Distribution	ППК-Р6	ППК-Р6.4. З-1
26.	Как настроить CI/CD-пайплайн для компиляции C-кода в проекте ML?	GitHub Actions / GitLab CI / Jenkins с этапами build/test/deploy	МО-2	МО-2.1. У-1
27.	Какие метрики мониторинга важны для низкоуровневых компонентов ML-систем?	Время выполнения / Использование памяти / Ошибки сегментации	МО-2	МО-2.3. З-1
28.	Как организовать версионирование C-библиотек и Python-обёрток?	SemVer / Git tags / Artifact Registry / PyPI versioning	МО-2	МО-2.2. У-2
29.	Какой навык необходим для распределения задач в команде разработки низкоуровневого кода?	Лидерство / Делегирование / Code Review / Планирование спринтов	РГ-1	РГ-1.1. У-1
30.	Как оценивается качество кода для C-расширений Python?	Unit-тесты / Valgrind / Sanitizers / Coverage / Benchmarking	РГ-1	РГ-1.1. У-3
31.	Как управлять версиями отдельных компонентов гибридного C/Python проекта?	Git submodules / DVC / Separate versioning / Dependency tracking	РГ-2	РГ-2.5. У-1
32.	Как организовать развёртывание приложения с C-расширениями в облаке?	Docker с компилятором / Pre-built wheels / Lambda layers	РГ-4	РГ-4.4. У-1
33.	Какие этические нормы важны при использовании ИИ для генерации C-кода?	Проверка безопасности / Утечки памяти / Документирование / Лицензирование	ППК-Р7	ППК-Р7.4. З-1
34.	Как документировать использование ИИ в разработке низкоуровневых компонентов?	Комментарии в коде / Отчёт / Лог промптов / Версия модели ИИ	ППК-Р7	ППК-Р7.4. У-2

35.	Как выявлять бизнес-проблемы, решаемые через низкоуровневую оптимизацию?	Профилирование / Benchmark / Анализ узких мест / ROI-расчёт	ППК-ДА4	ППК-ДА4.2. У-1
36.	Какие инструменты анализа бизнес-ситуации применяются для оценки оптимизации?	Cost-benefit analysis / Performance metrics / TCO / SLA	ППК-ДА4	ППК-ДА4.2. З-2
37.	Как выбирать метрики для оценки эффекта воздействия оптимизации кода?	Время выполнения / Память / Throughput / Latency / CPU usage	ППК-ДА5	ППК-ДА5.1. У-2
38.	Какие принципы планирования экспериментов важны для бенчмарков C/Python?	Контрольные группы / Повторяемость / Статистическая значимость / Warm-up	ППК-ДА5	ППК-ДА5.1. З-1
39.	Какой навык важен для презентации результатов оптимизации заказчику?	Коммуникация / Визуализация / Презентация / Storytelling	УНК-01 (УК-4)	УНК-01.01.01. У-1
40.	Какое действие проявляет поддержку коллег в команде разработки C/Python?	Наставничество / Code Review / Обратная связь / Knowledge Sharing	УНК-02 (УК-3)	УНК-02.01.01. У-5
41.	Что такое GIL в CPython и как он влияет на многопоточность?	Global Interpreter Lock — блокирует выполнение байт-кода в нескольких потоках	ППК-Р5	ППК-Р5.3. З-1
42.	Как обойти ограничение GIL при выполнении вычислений в C-расширениях?	Освобождение GIL через Py_BEGIN_ALLOW_THREADS / multiprocessing	ППК-Р5	ППК-Р5.3. У-2
43.	Назовите инструмент для обнаружения утечек памяти в C-программах.	Valgrind / AddressSanitizer / LeakSanitizer	ППК-Р6	ППК-Р6.3. У-1
44.	Как организовать тестирование C-кода в промышленном проекте?	Unity / CMocka / Google Test / pytest с ctypes	ППК-Р6	ППК-Р6.3. З-1
45.	Что такое байт-код Python и как он связан с CPython?	Промежуточное представление кода, выполняемое виртуальной машиной CPython	ППК-Р5	ППК-Р5.1. З-1
46.	Как используется системный вызов fork() в контексте Python-приложений?	Создание дочернего процесса через os.fork() или multiprocessing	ППК-Р5	ППК-Р5.4. З-3
47.	Какой метод используется для межпроцессного взаимодействия в Linux?	Pipes / Shared Memory / Message Queues / Sockets	ППК-Р5	ППК-Р5.4. У-1

48.	Как применяется низкоуровневое программирование для оптимизации ML-моделей?	Custom ops / Kernel fusion / SIMD / GPU kernels	ППК-ДА5	ППК-ДА5.3. У-2
49.	Как называется документация, оформленная в соответствии с профессиональными стандартами?	Техническая документация / API Docs / README / Changelog	УНК-01 (УК-4)	УНК-01.02.01. У-3
50.	Какой принцип чистого кода важен при разработке C-расширений для Python?	DRY / SOLID / KISS / Modular Design / Error Handling	ППК-Р6	ППК-Р6.3. З-1