Федеральное государственное автономное образовательное учреждение высшего образования
«Российский университет дружбы народов имени Патриса Лумумбы»

Факультет искусственного интеллекта
(наименование основного учебного подразделения (ОУП)-разработчика ОП ВО)

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

ИСТОРИЯ И ТЕОРИЯ ПРОГРАММИРОВАНИЯ
(наименование дисциплины/модуля)

Рекомендована МССН для направлений подготовки:

02.03.02 ФУНДАМЕНТАЛЬНАЯ ИНФОРМАТИКА И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ;
09.03.03 ПРИКЛАДНАЯ ИНФОРМАТИКА
(код и наименование направления подготовки/специальности)

Освоение дисциплины ведется в рамках реализации основной профессиональной образовательной программы высшего образования (ОП ВО):

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ: РАЗРАБОТКА И ОБУЧЕНИЕ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ
(наименование (профиль/специализация) ОП ВО)

2026 г.

1. ЦЕЛЬ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Дисциплина «История и теория программирования» входит в программу бакалавриата «Искусственный интеллект: разработка и обучение интеллектуальных систем» по направлениям подготовки 02.03.02 Фундаментальная информатика и информационные технологии и 09.03.03 Прикладная информатика, и изучается в 1 семестре 1 курса. Дисциплину реализует Кафедра прикладного искусственного интеллекта. Дисциплина состоит из 3 разделов и 26 тем и направлена на изучение эволюции программирования как научной и прикладной дисциплины — от первых языков и моделей вычислений до современных парадигм и технологий разработки, фундаментальных принципов построения программных систем, формальных языков и грамматик, принципов типизации, трансляции и архитектуры программного обеспечения — с акцентом на критический анализ выбора инструментов и подходов для задач искусственного интеллекта.

Целью освоения дисциплины является Формирование у студентов целостного представления об историческом развитии и теоретических основах программирования, развитие способности системно анализировать парадигмы и архитектурные решения в контексте задач ИИ, навыков критического сравнения языков и технологий программирования, а также начальных умений в области описания программных компонентов и их взаимодействия в технических спецификациях.

2. ТРЕБОВАНИЯ К РЕЗУЛЬТАТАМ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Освоение дисциплины «История и теория программирования» направлено на формирование у обучающихся следующих компетенций (части компетенций):

Таблица 2.1. Перечень компетенций, формируемых у обучающихся при освоении дисциплины (результаты освоения дисциплины)

Шифр	**Компетенция**	**Индикаторы достижения компетенции** (в рамках данной дисциплины)
ОПК-2	Способен понимать принципы работы современных информационных технологий и применять компьютерные методы, современное программное обеспечение, в том числе отечественного происхождения, для решения задач профессиональной деятельности	ОПК-2.1 Знает принципы работы современных информационных технологий, включая технологии машинного обучения, облачных вычислений, высокопроизводительных вычислений (HPC) и параллельного программирования;
ОПК-3	Способен разрабатывать алгоритмические и программные решения в области системного и прикладного программирования, математических и информационных моделей, баз данных, средств тестирования, пригодные для практического применения	ОПК-3.1 Знает основные алгоритмы и структуры данных, парадигмы программирования, принципы проектирования программных систем и баз данных;
ПК-1	Способен анализировать требования к программному обеспечению систем ИИ, разрабатывать технические спецификации и техническое задание на систему	ПК-1.2 Разрабатывает технические спецификации на программные компоненты систем ИИ и описывает их взаимодействие;

Шифр	Компетенция	Индикаторы достижения компетенции (в рамках данной дисциплины)
SS-1	Способен учитывать философские, когнитивные и социальные основания концепций ИИ в профессиональной деятельности	SS-1.1 Учитывает в разработке и эксплуатации систем ИИ философские основания концепций интеллекта, языка, знания, агентности;

3. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ОП ВО

Дисциплина «История и теория программирования» относится к обязательной части блока 1 «Дисциплины (модули)» образовательной программы высшего образования.

В рамках образовательной программы высшего образования обучающиеся также осваивают другие дисциплины и/или практики, способствующие достижению запланированных результатов освоения дисциплины «История и теория программирования».

Таблица 3.1. Перечень компонентов ОП ВО, способствующих достижению запланированных результатов освоения дисциплины

Шифр	Наименование компетенции	Предшествующие дисциплины/модули, практики*	Последующие дисциплины/модули, практики*
ОПК-2	Способен понимать принципы работы современных информационных технологий и применять компьютерные методы, современное программное обеспечение, в том числе отечественного происхождения, для решения задач профессиональной деятельности		Параллельное и распределенное программирование; Массово-параллельные вычисления в машинном обучении (GPU); Основы глубокого обучения; Программирование на языке Python; Hadoop, SPARK; Методы машинного обучения; Нейронные сети;
ОПК-3	Способен разрабатывать алгоритмические и программные решения в области системного и прикладного программирования, математических и информационных моделей, баз данных, средств тестирования, пригодные для практического применения		Дискретная математика; Программирование на языке C++; Алгоритмы и структуры данных; Введение в базы данных; Программирование на языке Python; Параллельное и распределенное программирование; Методы разработки решений на основе искусственного интеллекта (Git, Docker); Методы машинного обучения; Нейронные сети; Технологическая (проектно-технологическая) практика (учебная);

Шифр	Наименование компетенции	Предшествующие дисциплины/модули, практики*	Последующие дисциплины/модули, практики*
			Технологическая (проектно-технологическая) практика (производственная);
ПК-1	Способен анализировать требования к программному обеспечению систем ИИ, разрабатывать технические спецификации и техническое задание на систему		Параллельное и распределенное программирование; Искусственный интеллект и когнитивная психология; Этика и безопасность использования искусственного интеллекта; Методы машинного обучения; Массово-параллельные вычисления в машинном обучении (GPU); Оптимизация моделей машинного обучения; Основы глубокого обучения; Безопасность систем искусственного интеллекта; Практическая подготовка на проектах отраслевых индустриальных партнеров; *Большие языковые модели***; Программирование на языке C++; Методы разработки решений на основе искусственного интеллекта (Git, Docker); Введение в базы данных; MLOps и промышленная разработка систем искусственного интеллекта; Нейронные сети; Онтология и графы знаний; Проектирование и разработка систем компьютерного зрения; Практикум по обработке естественного языка (NLP); Эксплуатационная практика (учебная); Эксплуатационная практика (производственная); Преддипломная практика; Технологическая (проектно-технологическая) практика (производственная); Технологическая (проектно-технологическая) практика (учебная);
SS-1	Способен учитывать философские, когнитивные и социальные основания концепций ИИ в профессиональной		Философия; Искусственный интеллект и когнитивная психология; Онтология и графы знаний; Методы машинного

Шифр	Наименование компетенции	Предшествующие дисциплины/модули, практики*	Последующие дисциплины/модули, практики*
	деятельности		обучения; MLOps и промышленная разработка систем искусственного интеллекта; Основы глубокого обучения; Нейронные сети; Лингвистические основы анализа естественного языка; *Основы робототехники***; *Большие языковые модели***; *Генеративные модели***; Этика и безопасность использования искусственного интеллекта; Безопасность систем искусственного интеллекта; Практическая подготовка на проектах отраслевых индустриальных партнеров; Проектирование и разработка систем компьютерного зрения; Практикум по обработке естественного языка (NLP); *Рекомендательные системы***;

* - заполняется в соответствии с матрицей компетенций и СУП ОП ВО

** - элективные дисциплины /практики

4. ОБЪЕМ ДИСЦИПЛИНЫ И ВИДЫ УЧЕБНОЙ РАБОТЫ

Общая трудоемкость дисциплины «История и теория программирования» составляет «4» зачетные единицы.

Таблица 4.1. Виды учебной работы по периодам освоения образовательной программы высшего образования для очной формы обучения.

Вид учебной работы	**ВСЕГО,** ак.ч.		**Семестр(-ы)**
			1
Контактная работа, ак.ч.	*51*		*51*
Лекции (**ЛК**)	17		17
Лабораторные работы (**ЛР**)	0		0
Практические/семинарские занятия (**СЗ**)	34		34
Самостоятельная работа обучающихся, ак.ч.	*66*		*66*
Контроль (экзамен/зачет с оценкой), ак.ч.	*27*		*27*
Общая трудоемкость дисциплины	**ак.ч.**	***144***	*144*
	зач.ед.	***4***	*4*

5. СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

Таблица 5.1. Содержание дисциплины (модуля) по видам учебной работы

Номер раздела	Наименование раздела дисциплины	Наименование темы		Содержание темы	Вид учебной работы *	Формируемые индикаторы
Раздел 1	История и эволюция программирования	1.1	Введение в историю программирования: от счётных машин к программам	Понятия «программа» и «язык программирования». Вычислительные машины: от механических (Бэббидж, Ада Лавлейс) до электронных (ENIAC, МЭСМ). Появление хранимых программ (архитектура фон Неймана). Роль абстракции в развитии программирования. Систематический анализ: почему каждая эпоха требовала нового уровня абстракции	ЛК	ОПК-3.1, SS-1.1
		1.2	Ранние языки программирования: машинные коды, ассемблеры, языки высокого уровня	Машинные коды и ассемблер: прямое управление аппаратурой. Первые языки высокого уровня: Fortran (1957) — научные вычисления, Lisp (1958) — символьные вычисления и ИИ, COBOL (1959) — бизнес-приложения. ALGOL и его влияние на последующие языки. Связь ранних языков с современными фреймворками ИИ: Lisp как предтеча функциональных подходов в ML	ЛК	ОПК-3.1, SS-1.1
		1.3	Эволюция парадигм программирования	Генерации языков программирования: от машинных кодов (1GL) до декларативных языков (4GL) и языков для ИИ (5GL — Prolog). Императивное → структурное → объектно-ориентированное → функциональное программирование. Факторы, определяющие смену парадигм: сложность задач, аппаратные возможности, требования надёжности. Современный тренд: мультипарадигменные языки (Python, Scala, Kotlin)	ЛК	ОПК-3.1, SS-1.1
		1.4	Практикум: анализ и сравнение кода на ранних языках	Анализ структуры простейших алгоритмов на ассемблере (x86) и языке третьего поколения (Pascal/Fortran). Реализация одной задачи (сортировка, поиск) на двух языках. Сравнение по критериям: уровень абстракции, читаемость, объём кода. Документирование различий в форме сравнительной таблицы	СЗ	ОПК-3.1, ПК-1.2
		1.5	Практикум: сравнение парадигм на примере одной задачи	Реализация одной вычислительной задачи в процедурном (C/Pascal) и функциональном (Lisp/Haskell/Python-функциональный стиль) подходах. Сравнение: выразительность, производительность, поддерживаемость.	СЗ	ОПК-3.1, ПК-1.2, SS-1.1

Номер раздела	Наименование раздела дисциплины	Наименование темы		Содержание темы	Вид учебной работы *	Формируемые индикаторы
				Оформление результатов как мини-спецификации программного компонента с описанием интерфейсов		
		1.6	Практикум: построение хронологии эволюции языков программирования	Построение визуальной хронологии (timeline) развития языков программирования с указанием ключевых нововведений каждого языка и влияния на последующие. Анализ: какие языки и идеи привели к созданию Python — основного языка современного ИИ. Использование инструментов визуализации	СЗ	SS-1.1, ОПК-2.1
		1.7	Практикум: особенности первых программ и факторы эволюции	Обсуждение: какие ограничения ранних вычислительных систем определяли стиль программирования. Анализ первых программ: перфокарты, ленточные носители, ограничения памяти. Как аппаратные ограничения формировали архитектурные решения. Параллели с современными ограничениями (GPU-память, латентность инференса)	СЗ	SS-1.1, ОПК-3.1
		1.8	Практикум: влияние аппаратной базы на развитие программирования	История развития вычислительной техники: от реле и ламп до транзисторов, интегральных схем и GPU. Закон Мура и его влияние на проектирование ПО. Появление параллельных архитектур и их влияние на парадигмы (CUDA, OpenCL). Связь с современным ИИ: почему глубокое обучение стало возможным только с появлением GPU	СЗ	SS-1.1, ОПК-2.1
		1.9	Практикум: дискуссия — почему возникали новые поколения языков	Дискуссия: анализ причин появления каждого нового поколения языков. Проблема «серебряной пули» Брукса. Закон Вирта. Системный анализ: какой язык выбрать для конкретной задачи ИИ и почему. Формулирование критериев выбора языка/технологии для проекта	СЗ	SS-1.1, ОПК-3.1
Раздел 2	Теоретические основы и парадигмы программирования	2.1	Формальные языки, грамматики и автоматы	Формальные языки: алфавит, слово, язык. Грамматики Хомского: регулярные, контекстно-свободные, контекстно-зависимые, неограниченные. Конечные автоматы и регулярные выражения. BNF-нотация для описания синтаксиса. Формальные грамматики как основа парсинга в NLP	ЛК	ОПК-3.1, SS-1.1
		2.2	Парадигмы программирования: систематический анализ	Императивная: состояние, присваивание, циклы. Объектно-ориентированная: инкапсуляция, наследование,	ЛК	ОПК-3.1, SS-1.1

Номер раздела	Наименование раздела дисциплины	Наименование темы		Содержание темы	Вид учебной работы *	Формируемые индикаторы
				полиморфизм. Функциональная: чистые функции, неизменяемость, функции высшего порядка. Логическая: факты, правила, вывод (Prolog). Критический анализ: сильные и слабые стороны каждой парадигмы для задач ИИ		
		2.3	Принципы проектирования ПО: от структурного программирования к SOLID	Структурное программирование: теорема Бёма-Якопини, отказ от GOTO. Модульность и абстракция. Принципы SOLID для ООП. Паттерны проектирования (обзор): фабрика, стратегия, наблюдатель. Архитектурные стили: монолит, клиент-сервер, микросервисы. Связь с проектированием ML-пайплайна как последовательности модулей	ЛК	ОПК-2.1, ОПК-3.1, ПК-1.2
		2.4	Практикум: ООП — моделирование предметной области ИИ	Реализация иерархии классов на Python: базовый класс Model, наследники LinearModel, TreeModel, NeuralNetwork. Инкапсуляция параметров модели, полиморфизм метода predict(). Описание интерфейсов классов в форме технической спецификации с указанием входов, выходов и ограничений	СЗ	ОПК-3.1, ПК-1.2
		2.5	Практикум: функциональный стиль — рекурсия, композиция, пайплайны	Реализация задач в функциональном стиле на Python: рекурсивные вычисления, map/filter/reduce, лямбда-функции. Композиция функций как основа пайплайнов обработки данных. Сравнение императивной и функциональной реализации одного алгоритма. Связь с функциональным подходом в PyTorch: Sequential, nn.Module	СЗ	ОПК-3.1, ОПК-2.1
		2.6	Практикум: грамматика и парсер простого языка	Описание синтаксиса простого языка (калькулятор арифметических выражений) в BNF-нотации. Построение дерева разбора. Реализация простейшего рекурсивного нисходящего парсера на Python. Связь с токенизацией текстов в NLP-задачах. Оформление описания грамматики как компонента технической спецификации	СЗ	ОПК-3.1, ПК-1.2
		2.7	Практикум: алгоритмическая сложность и выбор структур данных	Понятие алгоритмической сложности: O, Ω, Θ. Основные структуры данных: массивы, списки, стеки, очереди, деревья, хеш-таблицы. Выбор структуры данных для задачи и его влияние на производительность. Связь с эффективностью ML-пайплайнов: выбор структуры для	СЗ	ОПК-3.1, ОПК-2.1

Номер раздела	Наименование раздела дисциплины	Наименование темы		Содержание темы	Вид учебной работы *	Формируемые индикаторы
				хранения датасета, батчей, индексов		
		2.8	Практикум: сравнение реализации задачи в разных парадигмах	Сравнительный анализ реализации обработки коллекции данных в императивном, объектно-ориентированном и функциональном стилях. Критерии: читаемость, поддерживаемость, тестируемость, производительность. Обсуждение: какой подход предпочтителен для ML-кода и почему	СЗ	SS-1.1, ОПК-3.1
		2.9	Практикум: классификация языков программирования — анализ экосистемы ИИ	Классификация языков по парадигме, системе типов, модели выполнения. Мультипарадигменные языки: Python, Scala, Julia. Анализ экосистемы языков для ИИ: Python (обучение), C++ (инференс), Rust (безопасность), Julia (научные вычисления). Формулирование обоснованных рекомендаций по выбору языка для проекта	СЗ	SS-1.1, ОПК-2.1
Раздел 3	Современные тенденции, безопасность и перспективы	3.1	Трансляция, компиляция и интерпретация	Компиляция vs интерпретация: принципы, преимущества, недостатки. JIT-компиляция (Java, Python/PyPy). AOT-компиляция (Rust, Go). Транспиляция (TypeScript → JavaScript). Байт-код и виртуальные машины (JVM, CPython). Связь с ИИ: почему Python медленный и как это решается (Cython, Numba, TorchScript)	ЛК	ОПК-3.1, ОПК-2.1
		3.2	Безопасность и надёжность языков программирования	Системы типов: статическая vs динамическая, сильная vs слабая типизация. Управление памятью: ручное (C/C++), сборка мусора (Java, Python), ownership (Rust). Защита от типичных ошибок: buffer overflow, null pointer, race condition. Связь с безопасностью ИИ-систем: типобезопасность данных, защита от инъекций в промптах	ЛК	ОПК-2.1, ОПК-3.1
		3.3	Современные тренды и перспективы развития	Новые парадигмы: реактивное программирование, параллельное/конкурентное (Go, Erlang), визуальное программирование (low-code/no-code). Domain-Specific Languages (DSL): SQL, TensorFlow graph, ONNX. ИИ-ассистенты для программирования (GitHub Copilot, Cursor). Вайб-кодинг: перспективы и ограничения. Прогноз: как ИИ изменит программирование	ЛК	SS-1.1, ОПК-2.1
		3.4	Практикум: компиляция vs интерпретация — сравнительный эксперимент	Реализация одного алгоритма (матричное умножение) на интерпретируемом (Python) и компилируемом (C/Rust) языке. Замер времени выполнения. Использование	СЗ	ОПК-3.1, ОПК-2.1, ПК-1.2

Номер раздела	Наименование раздела дисциплины	Наименование темы		Содержание темы	Вид учебной работы *	Формируемые индикаторы
				Cython/Numba для ускорения Python-кода. Сравнение: скорость, удобство разработки, размер бинарника. Документирование результатов в форме технической записки		
		3.5	Практикум: анализ системы типов на примере современных языков	Сравнение системы типов Python (динамическая), TypeScript (статическая с выводом), Rust (строгая с ownership). Примеры ошибок, которые каждая система ловит/не ловит. Type hints в Python: mypy, типизация в ML-коде. Оформление результатов как технической записки с рекомендациями для ИИ-проектов	СЗ	ОПК-2.1, ОПК-3.1, ПК-1.2
		3.6	Практикум: анализ архитектуры Open Source ИИ-проекта	Выбор Open Source проекта (scikit-learn, FastAPI, Hugging Face Transformers). Анализ архитектуры: структура модулей, паттерны проектирования, система типов, управление зависимостями. Построение диаграммы компонентов. Оценка надёжности, безопасности, расширяемости. Оформление как спецификации компонентов и их взаимодействия	СЗ	ОПК-2.1, ПК-1.2, SS-1.1
		3.7	Практикум: критерии выбора языка и технологий для задач ИИ	Систематический анализ языков и фреймворков для задач ИИ. Python: экосистема (NumPy, PyTorch, TensorFlow), слабые стороны (GIL, производительность). C++: ONNX Runtime, TensorRT. Rust: безопасность для production. Julia: научные вычисления. Формирование матрицы «задача × язык» с обоснованием выбора	СЗ	SS-1.1, ОПК-2.1
		3.8	Практикум: будущее программирования в эпоху ИИ	Дискуссия: как генеративный ИИ трансформирует программирование. ИИ-ассистенты: возможности и ограничения (GitHub Copilot, Claude). Вайб-кодинг: генерация кода по описанию на естественном языке. Проблемы: верификация сгенерированного кода, ответственность, безопасность. DSL для ИИ: TensorFlow Graph, JAX, Triton. Прогноз развития профессии программиста	СЗ	SS-1.1, ОПК-2.1

* - заполняется только по **ОЧНОЙ** форме обучения: *ЛК – лекции; ЛР – лабораторные работы; СЗ – практические/семинарские занятия.*

6. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

Таблица 6.1. Материально-техническое обеспечение дисциплины

Тип аудитории	**Оснащение аудитории**	**Специализированное учебное/лабораторное оборудование, ПО и материалы для освоения дисциплины (при необходимости)**
Лекционная	Аудитория для проведения занятий лекционного типа, оснащенная комплектом специализированной мебели; доской (экраном) и техническими средствами мультимедиа презентаций.	
Семинарская	Аудитория для проведения занятий семинарского типа, групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации, оснащенная комплектом специализированной мебели и техническими средствами мультимедиа презентаций.	Персональные компьютеры, необходимое ПО
Для самостоятельной работы	Аудитория для самостоятельной работы обучающихся (может использоваться для проведения семинарских занятий и консультаций), оснащенная комплектом специализированной мебели и компьютерами с доступом в ЭИОС.	Персональные компьютеры, необходимое ПО

* - аудитория для самостоятельной работы обучающихся указывается **ОБЯЗАТЕЛЬНО**!

7. УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

Основная литература:

1. Биллиг, В.А. Хроника языков программирования. Прошлое настоящее будущее : Учебное пособие / В.А. Биллиг — Москва : Интуит НОУ, 2024. — 288 с. — ISBN 978-5-9556-0204-2. — URL: https://book.ru/book/954714

2. Дильшода Дилшод кизи Рустамова, Ринат Фаридович Бурнашев. Эволюция языков программирования / СамГИИЯ. // Science and Education. 2023. №4.5. С. 835-839. ISSN 2181-0842

Дополнительная литература:

1. Златопольский, Д. М. Программирование: типовые задачи, алгоритмы, методы: учебное пособие: [12+] / Д. М. Златопольский. – 4-е изд. (эл.). – Москва: Лаборатория знаний, 2020. – 226 с.: ил. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=222873

2. Грацианова, Т. Ю. Программирование в примерах и задачах: учебное пособие: [12+] / Т. Ю. Грацианова. – 6-е изд. – Москва: Лаборатория знаний, 2020. – 373 с.: ил., табл., граф. – (ВМК МГУ — школе). – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=448048

Ресурсы информационно-телекоммуникационной сети «Интернет»:

1. ЭБС РУДН и сторонние ЭБС, к которым студенты университета имеют доступ

на основании заключенных договоров

- Электронно-библиотечная система РУДН – ЭБС РУДН https://mega.rudn.ru/MegaPro/Web
- ЭБС «Университетская библиотека онлайн» http://www.biblioclub.ru
- ЭБС «Юрайт» http://www.biblio-online.ru
- ЭБС «Консультант студента» www.studentlibrary.ru
- ЭБС «Знаниум» https://znanium.ru/

2. Базы данных и поисковые системы

- Sage https://journals.sagepub.com/
- Springer Nature Link https://link.springer.com/
- Wiley Journal Database https://onlinelibrary.wiley.com/
- Наукометрическая база данных Lens.org https://www.lens.org

Учебно-методические материалы для самостоятельной работы обучающихся при освоении дисциплины/модуля:*

1. Курс лекций по дисциплине «История и теория программирования».

* - все учебно-методические материалы для самостоятельной работы обучающихся размещаются в соответствии с действующим порядком на странице дисциплины **в ТУИС**!