МИНИСТЕРСТВО СЕЛЬСКОГО ХОЗЯЙСТВА РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ – МСХА имени К.А. ТИМИРЯЗЕВА»
(ФГБОУ ВО РГАУ - МСХА имени К.А. Тимирязева)

Институт экономики и управления АПК
Кафедра статистики и кибернетики

УТВЕРЖДАЮ:
Директор института
экономики и управления АПК
_____ Л.И. Хоружий
«28» августа 2025 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

Б1.В.23 «Программирование на языке С++»

для подготовки бакалавров

ФГОС ВО

Направление: 09.03.02 «Информационные системы и технологии»
Направленность: «Компьютерные науки и технологии искусственного интеллекта»

Курс 3
Семестр 6

Форма обучения: очная

Год начала подготовки: 2025

Москва, 2025

Разработчики:

Демичев В.В., канд. экон. наук, доцент _____ «26» августа 2025 г.
(ФИО, ученая степень, ученое звание) (подпись)

Титов А.Д., ассистент _____ «26» августа 2025 г.
(ФИО, ученая степень, ученое звание) (подпись)

Рецензент:

Кийко П.В., канд. пед. наук _____ «26» августа 2025 г.
(ФИО, ученая степень, ученое звание) (подпись)

Программа составлена в соответствии с требованиями ФГОС ВО, компетентностно-ролевых моделей в сфере искусственного интеллекта по направлению подготовки 09.03.02 Информационные системы и технологии, профессионального стандарта и учебного плана

Программа обсуждена на заседании кафедры статистики и кибернетики. Протокол № 11 от «26» августа 2025 г.

И. о. зав. кафедрой Уколова А.В., канд. экон. наук, доцент _____
(ФИО, ученая степень, ученое звание) (подпись)

«26» августа 2025 г.

Согласовано:
Председатель учебно-методической
комиссии института экономики и управления АПК
Гупалова Т.Н. канд. экон. наук, доцент протокол №1 _____
(ФИО, ученая степень, ученое звание) (подпись)

«28» августа 2025 г.

И. о. зав. выпускающей кафедрой
статистики и кибернетики
Уколова А.В., канд. экон. наук, доцент _____
(ФИО, ученая степень, ученое звание) (подпись)

«28» августа 2025 г.

Заведующий отделом комплектования ЦНБ _____
(подпись)

СОДЕРЖАНИЕ

АННОТАЦИЯ

рабочей программы учебной дисциплины
Б1.В.23«Программирование на языке С++»
для подготовки бакалавров по направлению 09.03.02 «Информационные системы и технологии» направленности «Компьютерные науки и технологии искусственного интеллекта»

Цель освоения дисциплины. Основная цель дисциплины «Программирование на языке С++» – овладение студентами основными методами разработки компьютерных программ посредством языка программирования С++ для решения практических задач.

Место дисциплины в учебном плане: дисциплина включена в часть учебного плана по направлению подготовки 09.03.02 «Информационные системы и технологии», формируемую участниками образовательных отношений.

Требования к результатам освоения дисциплины: в результате освоения дисциплины формируются следующие компетенции (индикаторы): ПКкрмии-24 (PL-3. Продвинутый уровень).1; ПКкрмии-24 (PL-3. Продвинутый уровень).2; ПКкрмии-24 (PL-3. Продвинутый уровень).3.

Краткое содержание дисциплины: Среды программирования. Классическая программа на С++. Компиляция. Редактирование связей. Объекты, типы и значения. Ввод данных. Переменные. Операции и операторы. Присваивание и инициализация. Составные операторы присваивания. Имена. Типы и объекты. Безопасность типов. Преобразование типов. Вычисления. Выражения. Инструкции. Инструкция выбора. Итерация. Функции. Объявление функций. Вектор. Обход вектора. Ошибки. Источники ошибок. Ошибки времени компиляции: синтаксические ошибки; ошибки, связанные с типами. Ошибки времени редактирования связей. Ошибки времени выполнения программы. Обработка ошибок в вызывающем коде. Обработка ошибок в вызываемом коде. Сообщения об ошибках. Исключения. Логические ошибки. Отладка. Пред- постусловия. Тестирование. Написание программ. Задача. Стадии, стратегия разработки программ. Лексемы. Грамматики. Превращение грамматики в программу. Выражения. Термы. Первичные выражения. Потоки лексем. Структура программы. Объявления и определения. Инициализация. Заголовочные файлы. Область видимости. Функции. Вызов функции и возврат значения. Порядок вычислений. Пространство имен.

Типы, определенные пользователем. Классы и члены класса. Интерфейс и реализация. Разработка класса. Перечисления. Перегрузка операторов. Интерфейсы классов. Векторы и динамически выделяемая память. Память, адреса, указатели. Динами чески распределяемая память: размещение, доступ, освобождение памяти. Деструкторы. Указатели на объекты класса. Указатели и ссылки. Указатель this. Доступ к элементам вектора. Массивы. Графические классы. Проектирование графических классов. Графическое представление функций и данных. Графические пользовательские интерфейсы.

Общая трудоемкость дисциплины: 2 зач.ед. (72 часа).
Промежуточный контроль: зачет с оценкой.

1. Цель освоения дисциплины

Целью освоения дисциплины «Программирование на языке С++» является овладение студентами основными методами разработки компьютерных программ посредством языка программирования С++ для решения практических задач.

2. Место дисциплины в учебном процессе

Дисциплина «Программирование на языке С++» относится к части Блока 1 «Дисциплины (модули)» учебного плана, формируемой участниками образовательных отношений. Дисциплина «Программирование на языке С++» реализуется в соответствии с требованиями ФГОС, ОПОП ВО и Учебного плана по направлению 09.03.02 «Информационные системы и технологии» и компетентностно-ролевых моделей в сфере искусственного интеллекта.

Предшествующими курсами, на которых непосредственно базируется дисциплина «Программирование на языке С++» являются «Алгоритмизация и программирование», «Программирование на языке Python».

Дисциплина «Программирование на языке С++» является основополагающей для изучения следующих дисциплин: «Инструментальные средства информационных систем», «Интеллектуальный анализ данных и статистика», а также подготовки выпускной квалификационной работы.

Особенностью дисциплины является изучение инструментов языка С++ для создания высокопроизводительных компьютерных программ, разработка и программирование программного продукта для решения практических задач.

Рабочая программа дисциплины «Программирование на языке С++» для инвалидов и лиц с ограниченными возможностями здоровья разрабатывается индивидуально с учетом особенностей психофизического развития, индивидуальных возможностей и состояния здоровья таких обучающихся.

3. Перечень планируемых результатов обучения по дисциплине, соотнесенных с планируемыми результатами освоения образовательной программы

Образовательные результаты освоения дисциплины обучающимся, представлены в таблице 1.

Таблица 1

Требования к результатам освоения учебной дисциплины

№ п/п	Код компетенции	Содержание компетенции (или её части)	Индикаторы компетенций	В результате изучения учебной дисциплины обучающиеся должны:		
				знать	уметь	владеть
1.	ПКкрмии-24 (PL-3. Продвинутый уровень)	Способен применять языки программирования C/C++ для решения задач в области ИИ	ПКкрмии-24 (PL-3. Продвинутый уровень).1. Разрабатывает и отлаживает эффективные многопоточные решения на C++. Уровень освоения индикатора: Решает проблемы одновременного доступа к данным из нескольких потоков, грамотно применяет атомарные операции и механизм блокировок. Оценивает производительность, умеет профилировать код и устраняет найденные узкие места	Базовые принципы многопоточного программирования; Понятие потока выполнения, общего состояния и гон; Назначение и базовое использование примитивов синхронизации	Писать простые многопоточные программы на C++; Использовать стандартные средства синхронизации для предотвращения гонок данных; Запускать и отлаживать многопоточные приложения в учебной среде	Навыками написания корректного многопоточного кода в рамках учебных задач; Базовыми инструментами отладки многопоточных программ
			ПКкрмии-24 (PL-3. Продвинутый уровень).2. Разрабатывает и отлаживает системы ИИ на C++ под конкретные аппаратные платформы с ограничениями по вычислительной мощности, в том числе для встроенных систем. Уровень освоения индикатора:	Общие подходы к разработке программ под ограниченные ресурсы; Основные принципы оптимизации вычислений; Обзор существующих	Подключать и использовать сторонние библиотеки в проектах на C++; Оценивать ресурсные требования простых программ; Адаптировать базовые алгоритмы	Навыками сборки и интеграции сторонних библиотек в проекты на C++; Базовыми навыками анализа и оптимизации потребления ресурсов в учебных приложениях

№ п/п	Код компетенции	Содержание компетенции (или её части)	Индикаторы компетенций	В результате изучения учебной дисциплины обучающиеся должны:		
				знать	уметь	владеть
			Понимает методы оптимизации моделей (квантование, сжатие весов модели и пр.) и вычислений ИИ. Находит и использует библиотеки, соответствующие решаемой задаче	библиотек ИИ на C++	под ограничения памяти и производительности	
			ПКкрмии-24 (PL-3. Продвинутый уровень).3. Разрабатывает и отлаживает решения на C++, использующие GPU и FPGA для массовой параллелизации вычислений в рамках общей системы ИИ. Уровень освоения индикатора: Знает методы оптимизации моделей (квантование, сжатие весов модели и пр.) и вычислений ИИ. Владеет готовыми инструментами для оптимизации моделей (TensorRT и пр. Умеет использовать средства отладки и профилирования кода, находить участки кода, ограничивающие производительность системы	Основные концепции гетерогенных вычислений; Обзор существующих технологий и фреймворков; Принципы профилирования и выявления узких мест в программе	Писать и компилировать простые программы с использованием CUDA или OpenCL; Использовать базовые инструменты профилирования; Анализировать производительность программных решений	Навыками разработки элементарных параллельных ядер на GPU; Навыками интерпретации профилей производительности и выявления «узких мест» кода в учебных задачах

4. Структура и содержание дисциплины

4.1 Распределение трудоёмкости дисциплины по видам работ по семестрам

Общая трудоёмкость дисциплины составляет 2 зачетные единицы (72 часа), их распределение по видам работ семестрам представлено в таблице 2.

ОЧНАЯ ФОРМА ОБУЧЕНИЯ

Таблица 2

Распределение трудоёмкости дисциплины по видам работ по семестрам

Вид учебной работы	Трудоёмкость (семестр № 6)/*
	час.
Общая трудоёмкость дисциплины по учебному плану	**72/4**
1. Контактная работа	**50,25/4**
Аудиторная работа	**50,25/4**
лекции (Л)	16
практические занятия (ПЗ)	34/4
контактная работа на промежуточном контроле (КРА)	0,25
2. Самостоятельная работа (СРС)	**21,75**
самостоятельное изучение разделов, самоподготовка (проработка и повторение лекционного материала и материала учебников и учебных пособий, подготовка к практическим занятиям)	12,75
подготовка к зачету	9
Вид промежуточного контроля:	Зачет с оценкой

* в том числе практическая подготовка

4.2 Содержание дисциплины

Таблица 3

Тематический план учебной дисциплины

Наименование разделов и тем дисциплин (укрупнённо)	**Всего**	**Аудиторная работа**			**Внеаудиторная работа СР**
		Л	**ПЗ/***	**ПКР**	
Раздел 1. Продвинутая многопоточность и производительность в C++	22	4	12		6
Раздел 2. Разработка ИИ-решений на C++ под аппаратные ограничения	20/2	4	10/2		6
Раздел 3. Гетерогенные вычисления: GPU, FPGA и ускорение ИИ-систем	29,75/2	8	12/2		9,75
Контактная работа на промежуточном контроле (КРА)	0,25			0,25	
Итого по дисциплине	**72/4**	**16**	**34/4**	**0,25**	**21,75**

* в том числе практическая подготовка

Раздел 1 Продвинутая многопоточность и производительность в C++

Тема 1 Модель памяти и основы многопоточности в C++

Рассматриваются модель памяти C++11, понятия happens-before и гонок данных. Вводятся потоки std::thread, примитивы синхронизации (mutex, lock_guard, condition_variable) и базовые паттерны многопоточного программирования.

Тема 2 Профилирование и отладка многопоточных приложений

Охватываются методы анализа производительности и обнаружения ошибок в многопоточном коде: использование ThreadSanitizer, Valgrind (Helgrind/DRD), perf и gprof для диагностики гонок, блокировок и узких мест.

Раздел 2 Разработка ИИ-решений на C++ под аппаратные ограничения

Тема 3 Архитектура и оптимизация ИИ-систем для встроенных платформ

Рассматриваются ограничения embedded-устройств, методы сжатия моделей (квантование, pruning), а также подходы к снижению потребления памяти и вычислительной сложности при инференсе на CPU.

Тема 4 Использование библиотек ИИ на C++

Изучаются популярные C++-библиотеки для ИИ: TensorFlow Lite, ONNX Runtime, OpenCV DNN. Практикуется сборка, интеграция, загрузка моделей и выполнение инференса с учётом ресурсных ограничений.

Раздел 3 Гетерогенные вычисления: GPU, FPGA и ускорение ИИ-систем

Тема 5 Программирование GPU на CUDA C++

Вводится архитектура GPU NVIDIA, модель выполнения CUDA, типы памяти и базовые техники оптимизации ядер. Студенты пишут и профилируют простые параллельные программы.

Тема 6 Ускорение ИИ с использованием TensorRT и гетерогенных систем

Рассматривается преобразование ONNX-моделей в TensorRT, управление памятью и планировщиком инференса. Практикуется интеграция CPU-предобработки и GPU-ускоренного вывода в единую ИИ-систему.

4.3 Лекции/практические занятия

Таблица 4

Содержание лекций/практических занятий и контрольные мероприятия

№ п/п	Название раздела, темы	№ и название лекций/ лабораторных/ практических/ семинарских занятий	Формируемые компетенции	Вид контрольного мероприятия	Кол-во часов/ из них пр. подг.
1.	**Раздел 1. Продвинутая многопоточность и производительность в C++**				
	Тема 1. Модель памяти и основы многопоточности в C++	Лекция №1. Модель памяти C++. Потоки, гонки данных, happens-before. Синхронизация: mutex, lock_guard, condition variables.	ПКкрмии-24 (PL-3).1		2
		Практическая работа №1. Создание потоков. Передача данных между потоками. Простейшие задачи с std::thread.	ПКкрмии-24 (PL-3).1	защита работы	2
		Практическая работа №2. Использование mutex и lock_guard для защиты общих ресурсов.	ПКкрмии-24 (PL-3).1	защита работы	2
		Практическая работа №3. Атомарные операции. Блокировки vs lock-free. Примеры с std::atomic.	ПКкрмии-24 (PL-3).1	защита работы	2
		Практическая работа №4. Condition variables и producer-consumer задачи.	ПКкрмии-24 (PL-3).1	защита работы	2
	Тема 2. Профилирование и отладка многопоточных приложений	Лекция №2. Профилирование и отладка многопоточных программ. Инструменты: Valgrind (Helgrind, DRD), ThreadSanitizer, perf, gprof.	ПКкрмии-24 (PL-3).1		2
		Практическая работа №5. Диагностика гонок данных с ThreadSanitizer и Valgrind.	ПКкрмии-24 (PL-3).1	защита работы	2
		Практическая работа №6. (кейс-задача) Разработка конвейерной системы обработки данных с несколькими потоками (producer-consumer). Применение атомарных операций и блокировок. Профилирование с ThreadSanitizer и perf.	ПКкрмии-24 (PL-3).1	защита кейс-задачи	2
2.	**Раздел 2. Разработка ИИ-решений на C++ под аппаратные ограничения**				
	Тема 3. Архитектура и оптимизация ИИ-систем для встроенных платформ	Лекция №3. Архитектура embedded ИИ-систем. Ограничения по памяти, энергопотреблению и вычислительной мощности. Подходы к оптимизации: квантование, pruning, knowledge distillation.	ПКкрмии-24 (PL-3).2		2
		Практическая работа №7. Сборка и подключение TensorFlow Lite C++ к проекту (CMake, библиотеки).	ПКкрмии-24 (PL-3).2	защита работы	2
		Практическая работа №8. Загрузка и инференс модели TensorFlow Lite (классификация изображений).	ПКкрмии-24 (PL-3).2	защита работы	2
		Практическая работа №9. Замер времени и потребления памяти при инференсе. Сравнение различных моделей.	ПКкрмии-24 (PL-3).2	защита работы	2
	Тема 4. Использование библиотек ИИ на	Лекция №4. Обзор библиотек ИИ на C++: TensorFlow Lite C++, ONNX Runtime, OpenCV DNN, Dlib. Их архитектура и области применения.	ПКкрмии-24 (PL-3).2		2
		Практическая работа №10. Применение квантованной модели в TensorFlow Lite (INT8) и	ПКкрмии-24 (PL-3).2	защита работы	2

№ п/п	Название раздела, темы	№ и название лекций/ лабораторных/ практических/ семинарских занятий	Формируемые компетенции	Вид контрольного мероприятия	Кол-во часов/ из них пр. подг.
	C++	сравнение точности/производительности.			
		Практическая работа №11. Оптимизация данных: предварительная обработка изображений на CPU, минимизация копирований.	ПКкрмии-24 (PL-3).2	защита работы	2
3.	**Раздел 3. Гетерогенные вычисления: GPU, FPGA и ускорение ИИ-систем**				
	Тема 5. Программирование GPU на CUDA C++	Лекция №5. Введение в гетерогенные вычисления. Архитектура GPU/FPGA. Модели программирования: CUDA, OpenCL, SYCL.	ПКкрмии-24 (PL-3).3		2
		Лекция №6. CUDA C++: ядра, память (global, shared, constant), синхронизация. Базовые оптимизации.	ПКкрмии-24 (PL-3).3		2
		Практическая работа №12. Установка CUDA Toolkit. Компиляция и запуск первого CUDA-ядра.	ПКкрмии-24 (PL-3).3	защита работы	2
		Практическая работа №13. Реализация векторного сложения на GPU.	ПКкрмии-24 (PL-3).3	защита работы	2
		Практическая работа №14. Использование shared memory для оптимизации CUDA-ядер.	ПКкрмии-24 (PL-3).3	защита работы	2
	Тема 6. Ускорение ИИ с использованием TensorRT и гетерогенных систем	Лекция №7. Ускорение ИИ-моделей: TensorRT. Преобразование ONNX → TensorRT. Управление памятью и планировщиком вывода.	ПКкрмии-24 (PL-3).3		2
		Практическая работа №15. Загрузка ONNX-модели в TensorRT и выполнение инференса.	ПКкрмии-24 (PL-3).3	защита работы	2
		Практическая работа №16. Профилирование GPU-кода с помощью Nsight Systems/nvprof.	ПКкрмии-24 (PL-3).3	защита работы	2
		Лекция №8. Отладка и тестирование гетерогенных систем. Обзор FPGA-подходов (OpenCL для FPGA, Vitis HLS). Интеграция CPU+GPU+FPGA.	ПКкрмии-24 (PL-3).3		2
		Практическая работа №17. (кейс-задача) Разработка системы: загрузка ONNX-модели → оптимизация через TensorRT → инференс на GPU → замер производительности → сравнение с CPU-версией. Интеграция предобработки изображений.	ПКкрмии-24 (PL-3).2, ПКкрмии-24 (PL-3).3	защита кейс-задачи	2

Таблица 5

Перечень вопросов для самостоятельного изучения дисциплины

№ п/п	Название раздела, темы	Перечень рассматриваемых вопросов для самостоятельного изучения
Раздел 1. Продвинутая многопоточность и производительность в C++		
1	Тема 1. Модель памяти и основы многопоточности в C++	Что такое модель памяти C++11? Как работает механизм happens-before? Чем отличаются std::mutex, std::shared_mutex и std::recursive_mutex? Какие проблемы возникают при некорректной синхронизации? Как работают condition_variable и notify_one/notify_all? (ПКкрмии-24 (PL-3).1)
2	Тема 2. Профилирование и отладка многопоточных приложений	Как использовать ThreadSanitizer для обнаружения гонок данных? Что такое Helgrind и DRD в Valgrind? Как интерпретировать вывод perf? Какие метрики производительности важно учитывать при оптимизации многопоточного кода? Как избежать ложных срабатываний анализаторов? (ПКкрмии-24 (PL-3).1)
Раздел 2. Разработка ИИ-решений на C++ под аппаратные ограничения		
3	Тема 3. Архитектура и оптимизация ИИ-систем для встроенных платформ	Что такое квантование весов и активаций? Как работает pruning моделей? Какие методы снижения потребления памяти применяются в embedded ИИ? Как выбрать подходящую архитектуру модели под ограничения CPU? Какие библиотеки поддерживают low-memory режимы? (ПКкрмии-24 (PL-3).2)
4	Тема 4. Использование библиотек ИИ на C++	Как подключить TensorFlow Lite C++ через CMake? Какие форматы моделей поддерживает ONNX Runtime? Как загрузить и выполнить инференс модели через OpenCV DNN? Какие особенности работы с библиотеками на разных платформах (Linux, Windows, Android)? Как организовать кэширование входных данных для повышения производительности? (ПКкрмии-24 (PL-3).2)
Раздел 3. Гетерогенные вычисления: GPU, FPGA и ускорение ИИ-систем		
5	Тема 5. Программирование GPU на CUDA C++	Как организованы блоки и потоки в CUDA? Какие типы памяти доступны в ядре (global, shared, constant, register)? Что такое warp и как он влияет на производительность? Как использовать __syncthreads() и почему это важно? Какие ошибки часто возникают при работе с CUDA и как их диагностировать? (ПКкрмии-24 (PL-3).3)
6	Тема 6. Ускорение ИИ с использованием TensorRT и гетерогенных систем	Как преобразовать ONNX-модель в TensorRT-план? Что такое dynamic shapes и как они влияют на производительность? Как управлять памятью в TensorRT? Как интегрировать TensorRT с OpenCV или другим предобработчиком? Какие средства профилирования доступны в Nsight Systems? Как интерпретировать результаты профилирования GPU-ядер? (ПКкрмии-24 (PL-3).2, ПКкрмии-24 (PL-3).3)

5. Образовательные технологии

Таблица 6

Применение активных и интерактивных образовательных технологий

№ п/п	Тема и форма занятия		Наименование используемых активных и интерактивных образовательных технологий (форм обучения)
1.	Практическое занятие №1. Создание потоков. Передача данных между потоками	ПЗ	Разбор конкретных ситуаций - анализ реальных примеров гонок данных из open-source проектов.
2.	Лекция №2. Профилирование и отладка многопоточных программ	Л	Интерактивная лекция с использованием онлайн-инструментов визуализации – live-coding.
3.	Лекция №4. Обзор библиотек ИИ на C++	Л	Демонстрация работы библиотек через экранное деление (screen sharing)
4.	Практическое занятие №8. Загрузка и инференс модели TensorFlow Lite	ПЗ	Демонстрация Live-Coding – преподаватель пишет код в реальном времени, студенты повторяют и задают вопросы.
5.	Кейс-задача 1. Разработка многопоточной системы обработки потока данных с профилированием	ПЗ	Кейс-метод – решение реальной задачи (например, обработка видеопотока в реальном времени).
6.	Кейс-задача 2. Создание оптимизированной ИИ-системы на базе TensorRT с GPU-ускорением	ПЗ	Интегрированный проект – объединение знаний по ИИ, GPU и оптимизации.

6. Текущий контроль успеваемости и промежуточная аттестация по итогам освоения дисциплины

6.1. Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений и навыков и (или) опыта деятельности

Пример практических работ

Практическая работа №1. Создание потоков. Передача данных между потоками. Простейшие задачи с std::thread

Цель: освоить создание и управление потоками выполнения в C++.

Задача: написать программу, запускающую 2–3 потока, каждый из которых выполняет простую задачу (например, вычисление суммы, вывод сообщения). Реализовать передачу данных в поток через аргументы функции и возврат результата через std::future/std::async или совместный буфер.

Требования: использовать только стандартную библиотеку C++. Избегать глобальных переменных, где возможно.

Вопросы для защиты:

1) Каковы способы передачи данных в поток при создании через std::thread?
2) Что произойдёт, если не вызвать join() или detach()?
3) В чём разница между std::thread и std::async?
4) Можно ли запускать поток из лямбда-выражения? Приведите пример.

Практическая работа №2. Использование mutex и lock_guard для защиты общих ресурсов

Цель: научиться предотвращать гонки данных с помощью примитивов синхронизации.

Задача: создать программу с несколькими потоками, инкрементирующими общий счётчик. Без синхронизации – наблюдается ошибка; с std::mutex и std::lock_guard – результат корректен.

Требования: продемонстрировать разницу в результатах до и после применения синхронизации.

Вопросы для защиты:

5) Почему lock_guard предпочтительнее ручного вызова lock()/unlock()?
6) Что такое RAII и как lock_guard его использует?
7) Возможна ли блокировка нескольких мьютексов в одном потоке? Как избежать deadlock?
8) Можно ли использовать один и тот же mutex для защиты разных переменных?

Практическая работа №3. Атомарные операции. Блокировки vs lock-free. Примеры с std::atomic

Цель: понять принципы lock-free программирования и применение атомарных типов.

Задача: реализовать инкремент общего счётчика с использованием std::atomic<int>. Сравнить производительность с версией на mutex.

Требования: использовать fetch_add, load, store. Провести замеры времени для 10^6 операций.

Вопросы для защиты:

9) Гарантирует ли std::atomic полную потокобезопасность в любых сценариях?
10) Всегда ли atomic быстрее, чем mutex? От чего это зависит?
11) Какие операции являются атомарными «из коробки» для std::atomic<T>?
12) Что такое memory order? Зачем он нужен?

Кейс-задачи от индустриального партнера «Россельхозбанка»

Кейс-задача 1. Разработка конвейерной системы обработки данных с несколькими потоками (producer-consumer). Применение атомарных операций и блокировок. Профилирование с ThreadSanitizer и perf.

Цель: интегрировать знания по многопоточности, синхронизации и профилированию в единую производительную и корректную систему обработки данных в реальном времени.

Описание задачи:

Разработать многопоточное приложение, моделирующее конвейер обработки:

Этап 1 (Producer): один или несколько потоков генерируют задачи (например, случайные числа или JSON-объекты) и помещают их в очередь.

Этап 2 (Processor): один или несколько потоков извлекают задачи из очереди, выполняют обработку (например, возведение в квадрат, хеширование) и передают результат в выходную очередь.

Этап 3 (Consumer): поток(и) забирают результаты и аккумулируют статистику (счётчик обработанных задач, общая сумма и т.п.).

Система должна корректно работать при произвольном числе потоков на каждом этапе. Использовать комбинацию:

- std::mutex + std::condition_variable для очередей,
- std::atomic для счётчиков (например, общее число обработанных задач).

После реализации:

1) Проверить отсутствие гонок данных с ThreadSanitizer,
2) Провести профилирование с perf для выявления узких мест,
3) Предложить и реализовать одно улучшение производительности (например, замена блокирующей очереди на lock-free, или балансировка нагрузки).

Технические требования: язык: C++17 или выше, сборка с -fsanitize=thread -g -O2, использование только стандартной библиотеки

Отчёт включает: код, вывод ThreadSanitizer, результаты perf, сравнение времени до/после оптимизации.

Результат выполнения:

Работоспособная консольная программа + отчёт, демонстрирующий корректность и производительность системы.

Вопросы для защиты:

1) Почему вы выбрали блокирующую очередь, а не lock-free структуру? Какие компромиссы были учтены?
2) Как ThreadSanitizer помог выявить (или подтвердить отсутствие) гонок данных в вашей системе?
3) Какие узкие места показал perf? Были ли они связаны с синхронизацией, кэш-промахами или чем-то ещё?
4) Как масштабируется производительность при увеличении числа потоков? Есть ли предел? Почему?
5) Как вы обеспечили завершение всех потоков корректно (без зависаний и утечек)?

Кейс-задача 2. Разработка системы: загрузка ONNX-модели → оптимизация через TensorRT → инференс на GPU → замер производительности → сравнение с CPU-версией. Интеграция предобработки изображений.

Цель: создать целостную ИИ-систему, демонстрирующую преимущества аппаратного ускорения и оптимизации моделей для embedded- и high-performance сценариев.

Описание задачи:

Реализовать приложение, выполняющее следующие этапы:

1) Загрузка модели: использовать предобученную ONNX-модель (например, ResNet-18, MobileNetV2) для классификации изображений.
2) Оптимизация: преобразовать ONNX-модель в TensorRT-движок с поддержкой FP16 (и, при возможности, INT8 через калибровку).
3) Предобработка: загрузить изображение (например, через OpenCV или stb_image), выполнить масштабирование, нормализацию и преобразование в тензор в формате, ожидаемом моделью (CHW, RGB, нормализация по ImageNet).
4) Инференс:
   - Выполнить инференс на GPU через TensorRT,
   - Выполнить инференс той же модели на CPU через ONNX Runtime,
5) Замер и сравнение:
   - Измерить latency и throughput для обоих вариантов,
   - Зафиксировать потребление памяти,
   - Сравнить точность (если есть ground truth или confidence score).

Программа должна быть оформлена как единый конвейер: от загрузки изображения до вывода результата и метрик.

Технические требования: использовать TensorRT C++ API, использовать ONNX Runtime C++ API для CPU-версии, минимизировать копирование данных между CPU и GPU поддержка одного входного изображения.

Отчёт включает: код, скриншоты результатов, таблицу сравнения (CPU vs GPU, FP32 vs FP16).

Результат выполнения:

Работоспособная консольная программа + отчёт с анализом эффективности GPU-ускорения.

Вопросы для защиты:

1) Какие шаги требуются для преобразования ONNX → TensorRT? Почему не все модели преобразуются корректно?
2) Почему TensorRT даёт ускорение? За счёт чего: оптимизации графа, ядер, памяти или всего вместе?
3) Как вы организовали предобработку, чтобы избежать лишних копий между OpenCV и TensorRT?
4) Какие ограничения у режима INT8? Почему не всегда его можно применить?
5) В каких сценариях CPU-инференс может быть предпочтительнее GPU (даже при наличии GPU)?

Вопросы к зачету

1) Что такое модель памяти C++11 и зачем она нужна?
2) Объясните понятие happens-before и его роль в многопоточном программировании.
3) В чём разница между data race и race condition?
4) Как создаётся поток с помощью std::thread? Как передавать аргументы в поток?
5) Что произойдёт, если не вызвать join() или detach() для объекта std::thread?
6) Какие примитивы синхронизации предоставляет стандартная библиотека C++?
7) Зачем нужен std::lock_guard? Почему его нельзя использовать повторно?
8) В чём преимущество std::unique_lock по сравнению с std::lock_guard?
9) Как реализовать блокировку с таймаутом?
10) Приведите пример классического producer-consumer с использованием std::condition_variable.
11) Как работает ThreadSanitizer? Какие ошибки он может обнаружить?
12) В чём различие между Helgrind и DRD в Valgrind?
13) Какие инструменты командной строки позволяют профилировать CPU-нагрузку в Linux?
14) Что такое «узкое место» (bottleneck) в многопоточном приложении? Как его выявить?
15) Почему чрезмерная синхронизация может снизить производительность?
16) Как избежать deadlock при работе с несколькими мьютексами?
17) Что такое ложная совместная память (false sharing)? Как её диагностировать и устранить?
18) Как использовать perf для анализа времени выполнения функций?
19) Какие метрики производительности важно отслеживать при оптимизации многопоточных систем?
20) Почему lock-free структуры не всегда быстрее блокирующих?
21) Какие типичные ограничения существуют у embedded ИИ-систем?
22) Что такое квантование модели? Какие типы квантования вы знаете (FP32 → INT8)?
23) В чём суть метода pruning (прореживания) нейросетей?
24) Как knowledge distillation помогает уменьшить размер модели?
25) Почему важно минимизировать копирование данных при инференсе?
26) Как оценить потребление памяти ИИ-модели на CPU?
27) Какие подходы позволяют снизить энергопотребление ИИ-системы?
28) Что такое latency и throughput в контексте ИИ-инференса?
29) Как влияет размер батча на производительность и потребление памяти?
30) Почему не все модели можно запустить на встроенных устройствах?
31) Как подключить TensorFlow Lite к C++-проекту с помощью CMake?

32) Какие шаги необходимы для загрузки .tflite-модели и выполнения инференса?

33) Как обработать входное изображение перед подачей в модель TensorFlow Lite?

34) В чём преимущества ONNX Runtime по сравнению с другими фреймворками?

35) Как использовать OpenCV DNN для запуска модели в формате ONNX?

36) Почему важно учитывать выравнивание данных при работе с тензорами?

37) Как измерить время инференса модели с высокой точностью?

38) Какие ошибки часто возникают при несоответствии формата данных и модели?

39) Как реализовать повторное использование буферов памяти для повышения эффективности?

40) Какие библиотеки C++ поддерживают квантованные модели?

41) Из каких компонентов состоит архитектура GPU NVIDIA (SM, warp, core)?

42) Как запустить CUDA-ядро? Что такое grid, block и thread?

43) Какие типы памяти доступны в CUDA? Как они соотносятся по скорости и объёму?

44) Зачем нужна shared memory? Приведите пример её использования.

45) Что такое warp divergence и как его избежать?

46) Как работает __syncthreads()? Почему его нельзя использовать в условных ветвях?

47) Какие ошибки возникают при выходе за границы массива в ядре?

48) Как скомпилировать CUDA-программу с помощью nvcc?

49) Почему копирование данных между CPU и GPU — дорогостоящая операция?

50) Как минимизировать обмен данными между хостом и устройством?

51) Что такое TensorRT? Какие задачи он решает?

52) Как преобразовать ONNX-модель в TensorRT-план?

53) Что такое engine и execution context в TensorRT?

54) Какие режимы точности поддерживает TensorRT (FP32, FP16, INT8)?

55) Как реализовать динамические размеры входов (dynamic shapes) в TensorRT?

56) Как профилировать выполнение инференса с помощью Nsight Systems?

57) Как интегрировать OpenCV (CPU) и TensorRT (GPU) в едином pipeline?

58) Какие этапы проходит модель при оптимизации в TensorRT?

59) Почему квантование INT8 требует калибровочного набора данных?

60) Какие преимущества даёт использование TensorRT по сравнению с прямым запуском ONNX?

6.2. Описание показателей и критериев контроля успеваемости, описание шкал оценивания

Оценка знаний и умений при выполнении практических работ и кейс-задач осуществляется на основе рейтинговой системы, учитывающей качество выполнения кода, полноту отчёта и уровень ответа на вопросы при защите. Студент допускается к промежуточной аттестации (зачёту с оценкой), если набрал не менее 60% от максимально возможного рейтинга по итогам всех практических занятий и кейс-задач.

Максимальная оценка за защиту одной практической работы или кейс-задачи –10 баллов.

10 баллов – работа выполнена полностью в соответствии с требованиями; код корректен, эффективен и сопровождается пояснительным комментарием; при защите студент демонстрирует глубокое понимание использованных технологий, отвечает на все вопросы без ошибок.

9 баллов – работа выполнена полностью; возможны незначительные недочёты в оформлении или стиль кода; при защите допущены мелкие неточности в терминологии, не влияющие на суть ответа.

8 баллов – работа выполнена, но содержит негрубые ошибки (например, неоптимальный выбор структуры данных, избыточное копирование); при защите студент верно объясняет логику, но допускает отдельные неточности, не ведущие к искажению сути.

7 баллов – работа частично соответствует требованиям; имеются ошибки в реализации синхронизации, памяти или профилирования; при защите сделаны неверные выводы (например, о причинах узкого места или источнике race condition), но общее понимание темы сохранено.

6–5 баллов – работа выполнена фрагментарно; код не проходит проверку на корректность или производительность; при защите нарушена логика объяснения, наблюдается поверхностное или искажённое понимание ключевых концепций (например, путаница между mutex и atomic, непонимание роли shared memory в CUDA).

Менее 5 баллов – работа не выполнена или содержит фундаментальные ошибки, свидетельствующие об отсутствии освоения компетенции.

Общее количество баллов формируется следующим образом: 17 практических работ * 10 баллов = 170 баллов, 2 кейс-задачи * 10 баллов = 20 баллов, максимально возможный рейтинг: 190 баллов. Итоговый рейтинг в процентах (средний балл / максимально возможный балл) * 100

Для допуска к зачёту с оценкой требуется не менее 114 баллов (60%).

Участие в интерактивных формах занятий (кейс-метод, проектная защита, peer-review) может быть дополнительно учтено преподавателем при выставлении оценки за защиту, если студент продемонстрировал активность и глубокое вовлечение в обсуждение.

На зачете с оценкой студент может получить максимальное количество баллов равное 10. Далее итоговая оценка определяется следующим образом. Если текущий рейтинг студента составляет 10 баллов, а на экзамене студент

получил 7 баллов («Хорошо»), то итоговая оценка 0,5*10+0,5*7=8,5 баллов («Отлично»).

Промежуточный контроль – зачет с оценкой.

Таблица 7

Шкала оценивания (средний балл)	**Экзамен**
8,5-10,0	Отлично
7,0-8,4	Хорошо
6,0-6,9	Удовлетворительно
0-5,9	Неудовлетворительно

Положительными оценками, при получении которых дисциплина засчитывается в качестве пройденной, являются оценки «удовлетворительно», «хорошо» и «отлично».

Если получена оценка «неудовлетворительно» по дисциплине, то необходимо, после консультации с преподавателем, в течение 10 календарных дней следующего семестра подготовить ответы на ряд вопросов, предусмотренных программой обучения, и представить результаты этих ответов преподавателю.

Критерии оценивания результатов обучения

Таблица 8

Оценка	**Критерии оценивания**
Высокий уровень «5» (отлично)	оценку **«отлично»** заслуживает студент, освоивший знания, умения, компетенции и теоретический материал без пробелов; выполнивший все задания, предусмотренные учебным планом на высоком качественном уровне; практические навыки профессионального применения освоенных знаний сформированы. Компетенции, закреплённые за дисциплиной, сформированы на уровне – высокий.
Средний уровень «4» (хорошо)	оценку **«хорошо»** заслуживает студент, практически полностью освоивший знания, умения, компетенции и теоретический материал, учебные задания не оценены максимальным числом баллов, в основном сформировал практические навыки. Компетенции, закреплённые за дисциплиной, сформированы на уровне – хороший (средний).
Пороговый уровень «3» (удовлетворительно)	оценку **«удовлетворительно»** заслуживает студент, частично с пробелами освоивший знания, умения, компетенции и теоретический материал, многие учебные задания либо не выполнил, либо они оценены числом баллов близким к минимальному, некоторые практические навыки не сформированы. Компетенции, закреплённые за дисциплиной, сформированы на уровне – достаточный.
Минимальный уровень «2» (неудовлетворительно)	оценку «неудовлетворительно» заслуживает студент, не освоивший знания, умения, компетенции и теоретический материал, учебные задания не выполнил, практические навыки не сформированы. Компетенции, закреплённые за дисциплиной, не сформированы.

7. Учебно-методическое и информационное обеспечение дисциплины

7.1 Основная литература

1. Ростовцев, В. С. Искусственные нейронные сети : учебник для вузов / В. С. Ростовцев. — 5-е изд., стер. — Санкт-Петербург : Лань, 2025. — 216 с. — ISBN 978-5-507-50568-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/447392

2. Красов, А. В. Разработка защищенного программного обеспечения : учебное пособие / А. В. Красов, А. Ю. Цветков. — Санкт-Петербург : СПбГУТ им. М.А. Бонч-Бруевича, 2023. — 154 с. — ISBN 978-5-89160-308-0. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/425906

3. Огнева, М. В. Программирование на языке C++: практический курс : учебное пособие для вузов / М. В. Огнева, Е. В. Кудрина. – Москва : Издательство Юрайт, 2022. – 335 с. – (Высшее образование). – ISBN 978-5-534-05123-0. – Текст : электронный // Образовательная платформа Юрайт [сайт]. – URL: https://urait.ru/bcode/492984

7.2 Дополнительная литература

1. Гольдберг, Й. Нейросетевые методы в обработке естественного языка : руководство / Й. Гольдберг ; перевод с английского А. А. Слинкина. — Москва : ДМК Пресс, 2019. — 282 с. — ISBN 978-5-97060-754-1. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/131704

2. Ганегедара, Т. Обработка естественного языка с TensorFlow : руководство / Т. Ганегедара ; перевод с английского В. С. Яценкова. — Москва : ДМК Пресс, 2020. — 382 с. — ISBN 978-5-97060-756-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/140584

3. Объектно-ориентированное программирование на C++ : учебник / И. В. Баранова, С. Н. Баранов, И. В. Баженова [и др.]. – Красноярск : СФУ, 2019. – 288 с. – ISBN 978-5-7638-4034-6. – Текст : электронный // Лань : электронно-библиотечная система. – URL: https://e.lanbook.com/book/157572

7.3 Статьи, опубликованные в научных журналах 1 уровня Белого списка научных журналов Минобрнауки России и сборниках научных работ конференций уровня А*

1. Материалы конференции Nature Machine Intelligence. – URL: https://link.springer.com/search?query=&advancedSearch=true&openAccess=true&dateFrom=&dateTo=&journal=Nature+Machine+Intelligence&sortBy=relevance

2. Материалы конференции Proceedings of the AAAI 2024 Spring Symposium Series, Stanford, CA, USA, March 25-27, 2024. AAAI Press 2024. – URL: https://dblp.uni-trier.de/db/conf/aaaiss/aaaiss2024.html

3. Материалы конференции Transactions of the association for computational linguistics 2024. – URL: https://aclanthology.org/volumes/2024.tacl-1/

4. Материалы конференции Conference on Empirical Methods in Natural Language Processing (EMNLP). – URL: https://dblp.uni-trier.de/db/conf/emnlp/index.html

8. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины

1. C++ reference https://en.cppreference.com/w/cpp
2. Справка по C++ https://ru.cppreference.com/w/cpp
3. Документация по языку C++ https://learn.microsoft.com/ru-ru/cpp/cpp/?view=msvc-170
4. C++ Programming Language https://devdocs.io/cpp/
5. Google's C++ Class. – URL: https://developers.google.com/edu/c++

9. Перечень программного обеспечения и информационных справочных систем

Таблица 9

Перечень программного обеспечения

№ п/п	Наименование раздела учебной дисциплины	Наименование программы	Тип программы	Автор	Год разработки
1	Разделы 1, 2, 3	GCC / Clang	Системная, инструментальная	Free Software Foundation / LLVM Project	Текущая версия
2	Разделы 1, 2, 3	CMake	Инструментальная, управляющая сборкой	Kitware	Текущая версия
3	Разделы 1, 2, 3	Valgrind (Helgrind, DRD)	Диагностическая, отладочная	Julian Seward et al.	Текущая версия
4	Разделы 1, 2, 3	ThreadSanitizer (TSan)	Диагностическая, отладочная	Google / LLVM	Текущая версия
5	Разделы 1, 2, 3	perf (Linux Performance Counters)	Профилирующая, системная	Ingo Molnar et al.	Текущая версия
6	Разделы 1, 2, 3	GDB (GNU Debugger)	Отладочная	Free Software Foundation	Текущая версия
7	Разделы 2, 3	TensorFlow Lite C++ Library	Библиотека, ИИ-вычислительная	Google	Текущая версия (v2.18+)
8	Разделы 2, 3	ONNX Runtime C++ API	Библиотека, ИИ-вычислительная	Microsoft	Текущая версия (v1.17+)
9	Разделы 2, 3	OpenCV DNN Module	Библиотека, ИИ-вычислительная	OpenCV Team	Текущая версия (v4.9+)
10	Разделы 3	NVIDIA CUDA Toolkit	Инструментальная, для GPU-разработки	NVIDIA Corporation	Текущая версия (v12.4+)
11	Разделы 3	TensorRT C++ API	Библиотека, ИИ-оптимизирующая	NVIDIA Corporation	Текущая версия (v8.6+)
12	Разделы 1, 2, 3	Visual Studio Code /	Интегрированная	Microsoft /	Текущая

№ п/п	Наименование раздела учебной дисциплины	Наименование программы	Тип программы	Автор	Год разработки
		CLion	среда разработки	JetBrains	версия

10. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине

Для проведения практических занятий нужен компьютерный класс с доступом в «Интернет», оснащенный программным обеспечением в соответствии с разделом 9.

Таблица 10

Сведения об обеспеченности специализированными аудиториями, кабинетами, лабораториями

Наименование специальных помещений и помещений для самостоятельной работы (№ учебного корпуса, № аудитории)	Оснащенность специальных помещений и помещений для самостоятельной работы
1	2
учебная аудитория для проведения занятий лекционного типа, учебная аудитория для групповых и индивидуальных консультаций, учебная аудитория для текущего контроля и промежуточной аттестации (2й учебный корпус, 102 ауд.)	1. Экран с электроприводом 1 шт. (Инв. №558771/2) 2. Проектор 1 шт. (без инв. №) – приобретался не за счет средств вуза 3. Вандалоустойчивый шкаф 1 шт. (Инв.№558850/7) 4. Системный блок iP-4 541 3200 Mhz/1024 Mb/ 80 Gb / DVD-R с монитором 1 шт. (Инв. №558777/9) 5. Стенд «Сергеев Сергей Степанович 1910-1999» 1 шт. (Инв.№591013/25) 6. Огнетушитель порошковый 1 шт. (Инв. №559527) 7. Подвесное крепление к огнетушителю 1 шт. (Инв. № 559528) 8. Жалюзи 2шт. (Инв. №1107-221225, Инв. №1107-221225) 9. Лавка 20 шт. 10. Стол аудиторный 20 шт. 11. Стол для преподавателя 1 шт. 12. Стул 2 шт. 13. Доска маркерная 1 шт. 14. Трибуна напольная 1 шт. (без инв. №)
учебная аудитория для проведения занятий семинарского типа, учебная аудитория для групповых и индивидуальных консультаций, учебная аудитория для текущего контроля и промежуточной аттестации, помещение для самостоятельной работы (2й учебный корпус, 302 ауд.)	1. Системный блок Intel Core Intel Core i3-2100/4096Mb/500Gb/DVD-RW 10 шт. (Инв.№601997, Инв.№601998, Инв.№601999, Инв.№602000, Инв.№602001, Инв.№602002, Инв.№602003, Инв.№602004, Инв.№602005, Инв.№602006) 2. Монитор 10 шт. (без инв. №) - приобретались не за счет средств вуза 3. Шкаф 2 шт. (Инв.№594166, Инв.№594167) 4. Тумба 1 шт. (Инв.№594168) 5. Подвесное крепление к огнетушителю 1 шт. (Инв. № 559528) 6. Огнетушитель порошковый 1 шт. (Инв. №559527) 7. Жалюзи 1 шт. (Инв.№551557) 8. Доска магнитно-маркерная 1 шт. 9. Стол 5 шт. 10. Стол компьютерный 12 шт. 11. Стул офисный 21 шт. 12. Сейф 1 шт. (без Инв.№).

Наименование специальных помещений и помещений для самостоятельной работы (№ учебного корпуса, № аудитории)	Оснащенность специальных помещений и помещений для самостоятельной работы
1	2
учебная аудитория для проведения занятий лекционного типа, учебная аудитория для проведения занятий семинарского типа, учебная аудитория для проведения курсового проектирования (выполнения курсовых работ), учебная аудитория для групповых и индивидуальных консультаций, учебная аудитория для текущего контроля и промежуточной аттестации (1й учебный корпус, 212 ауд.)	Количество рабочих мест: 24 Встроенные сетевые адаптеры (Intel I219-V или Realtek RTL8111H), интерфейс RJ-45, скорость 10/100/1000 Мбит/с. Точки доступа: Ubiquiti UniFi AP AC Pro, стандарты IEEE 802.11a/b/g/n/ac, частоты 2.4 ГГц (450 Мбит/с) и 5 ГГц (1300 Мбит/с), поддержка MU-MIMO, питание PoE. Структурное подразделение: Кафедра Цифровая кафедра
учебная аудитория для проведения занятий лекционного типа, учебная аудитория для проведения занятий семинарского типа, учебная аудитория для проведения курсового проектирования (выполнения курсовых работ), учебная аудитория для групповых и индивидуальных консультаций, учебная аудитория для текущего контроля и промежуточной аттестации (1й учебный корпус, 214 ауд.)	Количество рабочих мест: 24 Встроенные сетевые адаптеры (Intel I219-V или Realtek RTL8111H), интерфейс RJ-45, скорость 10/100/1000 Мбит/с. Точки доступа: Ubiquiti UniFi AP AC Pro, стандарты IEEE 802.11a/b/g/n/ac, частоты 2.4 ГГц (450 Мбит/с) и 5 ГГц (1300 Мбит/с), поддержка MU-MIMO, питание PoE. Структурное подразделение: Кафедра Цифровая кафедра
Студенческое общежитие	Комнаты для самоподготовки
ЦНБ имени Н.И. Железнова	Читальный зал

11. Методические рекомендации обучающимся по освоению дисциплины

Приступая к изучению дисциплины «Программирование на языке С++», студенты должны ознакомиться с учебной программой, учебной, научной и методической литературой, имеющейся в библиотеке РГАУ-МСХА им. К.А. Тимирязева, получить в библиотеке рекомендованные учебники и учебно-методические пособия, завести новую тетрадь для работы с первоисточниками.

В ходе подготовки к практическим занятиям изучить основную литературу, ознакомиться с дополнительной литературой в соответствии с поставленной задачей. При этом учесть рекомендации преподавателя и требования учебной программы. Необходимо дорабатывать свой конспект, делая в нем соответствующие записи из литературы, рекомендованной преподавателем и предусмотренной учебной программой.

При подготовке к зачету с оценкой повторять пройденный материал в строгом соответствии с учебной программой. Использовать конспекты и литературу, рекомендованную преподавателем. Обратить особое внимание на темы учебных занятий, пропущенных студентом по разным причинам. При необходимости обратиться за консультацией и методической помощью к преподавателю.

Виды и формы отработки пропущенных занятий

Студент, пропустивший занятия обязан самостоятельно выполнить сообщение (презентацию), рассмотренную на практическом занятии и подготовиться по контрольным вопросам к защите работы в рамках часов консультаций.

12. Методические рекомендации преподавателям по организации обучения по дисциплине

Курс «Программирование на языке С++» должен давать не абстрактно-формальные, а прикладные знания. Данная цель может быть реализована только при условии соблюдения в учебных планах преемственности учебных дисциплин. Базовые знания для изучения дисциплины дают такие предметы, как экономическая теория, информатика.

Преподаватель должен указывать, в какой последовательности следует изучать материал дисциплины, обращать внимание на особенности изучения отдельных тем и разделов, помогать отбирать наиболее важные и необходимые сведения из учебных пособий, а также давать объяснения вопросам программы курса, которые обычно вызывают затруднения. При этом преподавателю необходимо учитывать следующие моменты:

1. Не следует перегружать студентов творческими заданиями.
2. Чередовать творческую работу на занятиях с заданиями во внеаудиторное время.
3. Давать студентам четкий инструктаж по выполнению самостоятельных заданий: цель задания; условия выполнения; объем; сроки; требования к оформлению.
4. Осуществлять текущий учет и контроль за самостоятельной работой.
5. Давать оценку обобщать уровень усвоения навыков самостоятельной, творческой работы.

Программу разработали:

Демичев В.В., канд. экон. наук, доцент
(ФИО, ученая степень, ученое звание) (подпись)

Титов А.Д., ассистент
(ФИО, ученая степень, ученое звание) (подпись)

РЕЦЕНЗИЯ

**на рабочую программу дисциплины «Программирование на языке С++»
ОПОП ВО по направлению 09.03.02 «Информационные системы и технологии»,
направленность «Компьютерные науки и технологии искусственного интеллекта»
(квалификация выпускника – бакалавр)**

Кийко Павлом Владимировичем, доцентом кафедры высшей математики, кандидатом педагогических наук (далее по тексту рецензент), проведено рецензирование рабочей программы дисциплины «Технологии хранения и управления данными в АПК» ОПОП ВО по направлению *09.03.02 «Информационные системы и технологии»*, направленность «Компьютерные науки и технологии искусственного интеллекта» (бакалавриат) разработанной в ФГБОУ ВО «Российский государственный аграрный университет – МСХА имени К.А. Тимирязева», на кафедре статистики и кибернетики (разработчики – Демичев Вадим Владимирович, доцент, кандидат экономических наук, Титов Артем Денисович, ассистент кафедры статистики и кибернетики).

Рассмотрев представленные на рецензирование материалы, рецензент пришел к следующим выводам:

1. Предъявленная рабочая программа дисциплины «Программирование на языке С++» (далее по тексту Программа) *соответствует* требованиям ФГОС ВО по направлению *09.03.02 «Информационные системы и технологии»* и компетентностно-ролевых моделей в сфере искусственного интеллекта. Программа *содержит* все основные разделы, *соответствует* требованиям к нормативно-методическим документам.

2. Представленная в Программе ***актуальность*** учебной дисциплины в рамках реализации ОПОП ВО *не подлежит сомнению* – дисциплина относится к части, формируемой участниками образовательных отношений учебного цикла – Б1.В

3. Представленные в Программе ***цели*** дисциплины *соответствуют* требованиям ФГОС ВО направления *09.03.02 «Информационные системы и технологии»*.

4. В соответствии с Программой за дисциплиной «**Программирование на языке С++**» закреплена ***1 профессиональная компетенция из компетентностно-ролевой модели (3 индикатора).*** Дисциплина «Программирование на языке С++» и представленная Программа *способна реализовать* их в объявленных требованиях. Результаты обучения, представленные в Программе в категориях знать, уметь, владеть соответствуют специфике и содержанию дисциплины и демонстрируют возможность получения заявленных результатов.

5. Общая трудоёмкость дисциплины «Программирование на языке С++» составляет 2 зачётные единицы (72 часа/из них практическая подготовка 4 часа).

6. Информация о взаимосвязи изучаемых дисциплин и вопросам исключения дублирования в содержании дисциплин *соответствует* действительности. Дисциплина «Программирование на языке С++» взаимосвязана с другими дисциплинами ОПОП ВО и Учебного плана по направлению *09.03.02 «Информационные системы и технологии»* и возможность дублирования в содержании отсутствует.

7. Представленная Программа предполагает использование современных образовательных технологий, используемые при реализации различных видов учебной работы. Формы образовательных технологий *соответствуют* специфике дисциплины.

8. Программа дисциплины «Программирование на языке С++» предполагает занятия в интерактивной форме.

9. Виды, содержание и трудоёмкость самостоятельной работы студентов, представленные в Программе, *соответствуют* требованиям к подготовке выпускников, содержащимся во ФГОС ВО направления *09.03.02 «Информационные системы и технологии»*.

10. Представленные и описанные в Программе формы *текущей* оценки знаний (опрос, в форме обсуждения отдельных вопросов), *соответствуют* специфике дисциплины и требованиям к выпускникам.

Форма промежуточного контроля знаний студентов, предусмотренная Программой, осуществляется в форме зачета с оценкой в 6 семестре, что *соответствует* статусу дисциплины, части, формируемой участниками образовательных отношений учебного цикла – Б1.В ФГОС ВО направления *09.03.02 «Информационные системы и технологии»*.

11. Формы оценки знаний, представленные в Программе, *соответствуют* специфике дисциплины и требованиям к выпускникам.

12. Учебно-методическое обеспечение дисциплины представлено: основной литературой – 3 источника (базовый учебник), дополнительной литературой – 3 наименования, журналы 1 уровня Белого списка научных журналов Минобрнауки России и сборниках научных работ конференций уровня А* – 4 источника, Интернет-ресурсы – 5 источников и *соответствует* требованиям ФГОС ВО направления *09.03.02 «Информационные системы и технологии»*.

13. Материально-техническое обеспечение дисциплины соответствует специфике дисциплины «Программирование на языке С++» и обеспечивает использование современных образовательных, в том числе интерактивных методов обучения.

14. Методические рекомендации студентам и методические рекомендации преподавателям по организации обучения по дисциплине дают представление о специфике обучения по дисциплине «Программирование на языке С++».

ОБЩИЕ ВЫВОДЫ

На основании проведенного рецензирования можно сделать заключение, что характер, структура и содержание рабочей программы дисциплины «Программирование на языке С++» ОПОП ВО по направлению *09.03.02 «Информационные системы и технологии»*, направленности **«Компьютерные науки и технологии искусственного интеллекта»** (квалификация выпускника – бакалавр), разработанная Демичевым Вадимом Владимировичем, доцентом, кандидатом экономических наук, Титовым Артемом Денисовичем, ассистентом кафедры статистики и кибернетики, соответствует требованиям ФГОС ВО, компетентностно-ролевых моделей в сфере искусственного интеллекта, современным требованиям экономики, рынка труда и позволит при её реализации успешно обеспечить формирование заявленных компетенций.

Рецензент: Кийко Павел Владимирович, доцент кафедры высшей математики, кандидат педагогических наук ФГБОУ ВО «Российский государственный аграрный университет–МСХА имени К.А. Тимирязева»

_____ «26» августа 2025 г.

(подпись)