МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский технологический университет «МИСиС»

Рабочая программа дисциплины (модуля)

Параллельные вычисления

Закреплена за подразделением — Кафедра инженерной кибернетики

Направление подготовки — 01.03.04 ПРИКЛАДНАЯ МАТЕМАТИКА

Профиль — Алгоритмы и методы наукоемкого программного обеспечения

Квалификация **Бакалавр**
Форма обучения **очная**
Общая трудоемкость **3 ЗЕТ**

Часов по учебному плану 108
в том числе:
аудиторные занятия 48
самостоятельная работа 60

Формы контроля в семестрах:
зачет с оценкой 8

Распределение часов дисциплины по семестрам

Семестр (<Курс>.<Семестр на курсе>)	**8 (4.2)**		Итого	
Недель	12			
Вид занятий	УП	РП	УП	РП
Лекции	24	24	24	24
Практические	24	24	24	24
Итого ауд.	48	48	48	48
Контактная работа	48	48	48	48
Сам. работа	60	60	60	60
Итого	108	108	108	108

Программу составил(и):
д.т.н., проф., Садеков Р.Н.

Рабочая программа
Параллельные вычисления

Разработана в соответствии с ОС ВО:

Самостоятельно устанавливаемый образовательный стандарт высшего образования - бакалавриат Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский технологический университет «МИСиС» по направлению подготовки 01.03.04 ПРИКЛАДНАЯ МАТЕМАТИКА (приказ от 02.04.2021 г. № 119 о.в.)

Составлена на основании учебного плана:

01.03.04 ПРИКЛАДНАЯ МАТЕМАТИКА, 01.03.04-БПМ-22.plx Алгоритмы и методы наукоемкого программного обеспечения, утвержденного Ученым советом ФГАОУ ВО НИТУ "МИСиС" в составе соответствующей ОПОП ВО 22.09.2022, протокол № 8-22

Утверждена в составе ОПОП ВО:

01.03.04 ПРИКЛАДНАЯ МАТЕМАТИКА, Алгоритмы и методы наукоемкого программного обеспечения, утвержденной Ученым советом ФГАОУ ВО НИТУ "МИСиС" 22.09.2022, протокол № 8-22

Рабочая программа одобрена на заседании
Кафедра инженерной кибернетики

Протокол от 23.06.2022 г., №11

Руководитель подразделения Ефимов А.Р.

1. ЦЕЛИ ОСВОЕНИЯ	
1.1	Цель дисциплины "Параллельные вычисления": освоение базовых знаний по организации параллельных вычислительных систем, а также освоение основных технологий параллельного программирования для дальнейшего использования при решении ресурсоемких вычислительных задач.

2. МЕСТО В СТРУКТУРЕ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ	
Блок ОП:	Б1.В.ДВ.09
2.1	**Требования к предварительной подготовке обучающегося:**
2.1.1	Введение в разработку приложений дополненной и виртуальной реальностей
2.1.2	Нейронные сети
2.1.3	Облачные технологии
2.1.4	Обработка естественного языка
2.1.5	Обучение с подкреплением
2.1.6	Программирование роботов II
2.1.7	Системный анализ и принятие решений
2.1.8	Системы автоматизированного проектирования
2.1.9	Экспертные и рекомендательные системы
2.1.10	Дискретные и нелинейные системы автоматического управления
2.1.11	Имитационное моделирование
2.1.12	Машинное обучение II
2.1.13	Методы и средства обработки изображений
2.1.14	Методы оптимизации
2.1.15	Основы мехатроники
2.1.16	Прикладной статистический анализ
2.1.17	Программирование роботов I
2.1.18	Производственная практика по освоению первичных навыков в области разработки наукоемкого ПО
2.1.19	Производственная практика по освоению первичных навыков в области разработки робототехнических и киберфизических систем
2.1.20	Фрактальный анализ
2.1.21	Математическое моделирование
2.1.22	Основы теории информации и автоматов
2.1.23	Основы электротехники и электроники
2.1.24	Современные технологии разработки мобильных приложений
2.1.25	Теория случайных процессов
2.1.26	Функциональный анализ
2.1.27	Численные методы
2.1.28	Операционные системы и среды
2.1.29	Основы теории информации и автоматов
2.1.30	Разработка клиент-серверных приложений
2.1.31	Сетевые технологии
2.1.32	Учебная практика по ознакомлению с технологиями разработки наукоемкого ПО
2.1.33	Учебная практика по ознакомлению с технологиями разработки робототехнических и киберфизических систем
2.1.34	Базы данных
2.1.35	Технологии программирования
2.1.36	Объектно-ориентированное программирование
2.1.37	Введение в специальность
2.1.38	Вычислительные машины, сети и системы
2.1.39	Программирование и алгоритмизация
2.2	**Дисциплины (модули) и практики, для которых освоение данной дисциплины (модуля) необходимо как предшествующее:**

3. РЕЗУЛЬТАТЫ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ, СООТНЕСЕННЫЕ С ФОРМИРУЕМЫМИ КОМПЕТЕНЦИЯМИ

ОПК-4: Способен понимать принципы работы современных информационных технологий и использовать их для решения задач профессиональной деятельности, разрабатывать алгоритмы и компьютерные программы, пригодные для практического применения, выбирать и применять методики проектирования и актуальные инструментальные средства разработки
Знать:
ОПК-4-З1 основные современные компьютерные технологии параллельного программирования (OpenMP, MPI, Posix threads).
ПК-4: Способен выявлять естественно-научную сущность проблем, возникающих в ходе профессиональной деятельности, применять современный математический аппарат
Знать:
ПК-4-З1 методы и средства параллельной обработки информации;
ПК-4-З2 классификацию суперкомпьютерных вычислительных систем;
УК-1: Способен осуществлять поиск, критический анализ и синтез информации, умение анализировать процессы и системы с использованием соответствующих аналитических, вычислительных и экспериментальных методов, применять системный подход для решения поставленных задач
Знать:
УК-1-З2 реализацию параллельных численных методов и комплексов программ.
УК-1-З1 текущее положение современных научных и технологических достижений в области параллельных вычислений;
ПК-4: Способен выявлять естественно-научную сущность проблем, возникающих в ходе профессиональной деятельности, применять современный математический аппарат
Уметь:
ПК-4-У1 выполнять конвертацию последовательных программ в параллельные;
УК-1: Способен осуществлять поиск, критический анализ и синтез информации, умение анализировать процессы и системы с использованием соответствующих аналитических, вычислительных и экспериментальных методов, применять системный подход для решения поставленных задач
Уметь:
УК-1-У2 применять полученную теоретическую базу для решения конкретных практических задач.
ПК-4: Способен выявлять естественно-научную сущность проблем, возникающих в ходе профессиональной деятельности, применять современный математический аппарат
Уметь:
ПК-4-У2 разрабатывать параллельные алгоритмы и выполнять оценку их эффективности.
УК-1: Способен осуществлять поиск, критический анализ и синтез информации, умение анализировать процессы и системы с использованием соответствующих аналитических, вычислительных и экспериментальных методов, применять системный подход для решения поставленных задач
Уметь:
УК-1-У1 проводить вычислительные эксперименты, разрабатывать математические модели, параллельные алгоритмы численных методов;
ПК-4: Способен выявлять естественно-научную сущность проблем, возникающих в ходе профессиональной деятельности, применять современный математический аппарат
Владеть:
ПК-4-В1 методикой решения задач с использованием высокопроизводительных вычислительных средств;
УК-1: Способен осуществлять поиск, критический анализ и синтез информации, умение анализировать процессы и системы с использованием соответствующих аналитических, вычислительных и экспериментальных методов, применять системный подход для решения поставленных задач
Владеть:
УК-1-В1 основными методами научных исследовании, навыками проведения лабораторного эксперимента;
УК-1-В2 методами и алгоритмами параллельных вычислений обработки экспериментальных данных с помощью современных программных комплексов.
ОПК-4: Способен понимать принципы работы современных информационных технологий и использовать их для решения задач профессиональной деятельности, разрабатывать алгоритмы и компьютерные программы, пригодные для практического применения, выбирать и применять методики проектирования и актуальные инструментальные средства разработки

Владеть:
ОПК-4-В1 навыками программирования на языке С/С++ с помощью технологий OpenMP, MPI, Posix threads для многопроцессорных вычислительных систем.

4. СТРУКТУРА И СОДЕРЖАНИЕ

Код занятия	Наименование разделов и тем /вид занятия/	Семестр / Курс	Часов	Формируемые индикаторы компетенций	Литература и эл. ресурсы	Примечание	КМ	Выполняемые работы
	Раздел 1. Теория параллельных вычислительных систем							
1.1	Введение. Цели, задачи и проблемы параллельных вычислений, обзор современного состояния IT в области параллельных вычислений. /Лек/	8	2	УК-1-З1 ПК-4-З1 ПК-4-З2	Л1.4 Л1.7			
1.2	Многопроцессорные вычислительные комплексы: архитектура высокопроизводительных ЭВМ, классификация параллельных вычислительных систем. /Лек/	8	2	УК-1-З1 ПК-4-З1 ПК-4-З2	Л1.4 Л1.7			
1.3	Моделирование и анализ параллельных алгоритмов. Оценка эффективности параллельных алгоритмов. /Лек/	8	4	УК-1-З1 ПК-4-З1 ПК-4-З2	Л1.4 Л1.7			
1.4	Основные принципы разработки параллельных алгоритмов и программ. /Лек/	8	2	УК-1-З1 ПК-4-З1 ПК-4-З2	Л1.4 Л1.7			
1.5	Введение в технологию параллельных вычислений, базовые конструкции, распараллеливание простых циклов. /Пр/	8	4	УК-1-В1 ОПК-4-З1 ПК-4-У1	Л1.2 Л1.7			
1.6	Повторение лекционного материала, подготовка к лабораторным работам. /Ср/	8	2	УК-1-З1 ПК-4-З1 ПК-4-З2 ПК-4-У1	Л1.7			
1.7	Подготовка к контрольной работе по разделу. /Ср/	8	4					
	Раздел 2. Технологии параллельного программирования							
2.1	Средства и инструменты разработки параллельных программ. MPI, OpenMP, Posix thread. /Лек/	8	2	УК-1-З2 УК-1-У1 УК-1-У2 ОПК-4-З1	Л1.2 Л1.3 Э1 Э2 Э3			
2.2	Технология программирования OpenMP. Основные директивы, применение технологии. Отладка параллельных программ. /Лек/	8	6	УК-1-З2 УК-1-У1 УК-1-У2 ОПК-4-З1	Л1.2 Э1 Э2 Э3			
2.3	Алгоритмы параллельных векторных и матричных преобразований. /Пр/	8	4	УК-1-В1 УК-1-В2 ОПК-4-В1 ПК-4-У1 ПК-4-У2 ПК-4-В1	Л1.2 Л1.5 Л1.6 Э1 Э2 Э3			

2.4	Алгоритмы параллельного решение систем линейных алгебраических уравнений. /Пр/	8	4	УК-1-В1 УК-1-В2 ОПК-4-В1 ПК-4-У1 ПК-4-У2 ПК-4-В1	Л1.2 Л1.5 Л1.6 Э1 Э2 Э3			
2.5	Алгоритмы параллельного интегрирования обыкновенных дифференциальных уравнений. /Пр/	8	4	УК-1-В1 УК-1-В2 ОПК-4-В1 ПК-4-У1 ПК-4-У2 ПК-4-В1	Л1.2 Л1.5 Л1.6 Э1 Э2 Э3			
2.6	Повторение лекционного материала, подготовка к лабораторным работам. /Ср/	8	2	УК-1-З2 УК-1-У1 УК-1-У2 ОПК-4-З1	Л1.2 Л1.5 Л1.6 Л1.7 Э1 Э2 Э3			
	Раздел 3. Многопоточное программирование							
3.1	Технология многопоточного программирования Posix thread. Создание и управление потоками. /Лек/	8	4	УК-1-З1 ОПК-4-З1	Л1.1 Л1.4			
3.2	Механизмы Mutex-exclusion, взаимная блокировка потоков, обеспечение потокобезопасности, асинхронные вызовы. /Лек/	8	2	УК-1-З1 ОПК-4-З1	Л1.1 Л1.4			
3.3	Потокобезопасные объекты. механизмы mutex exclusion, разработка потокобезопасных объектов. /Пр/	8	4	УК-1-В2 ОПК-4-В1 ПК-4-У2	Л1.1 Л1.4			
3.4	Многопоточное программирование. клиент-серверная архитектура приложений, front- и back-потоки исполнения. /Пр/	8	4	УК-1-В1 УК-1-В2 ОПК-4-В1 ПК-4-У2	Л1.1 Л1.4			
3.5	Повторение лекционного материала, подготовка к лабораторным работам. /Ср/	8	2	УК-1-В1 УК-1-В2 ОПК-4-В1 ПК-4-У2	Л1.1 Л1.4			
3.6	Подготовка к контрольной работе по разделу. /Ср/	8	4	УК-1-З1 ОПК-4-З1	Л1.1 Л1.4			
	Раздел 4. Курсовая работа							
4.1	Разработка параллельных форм алгоритмов решения поставленной задачи, проектирование объектно-ориентированного дизайна программной составляющей курсовой работы. /Ср/	8	16	УК-1-З1 УК-1-З2 УК-1-У1 УК-1-У2 УК-1-В1 УК-1-В2 ОПК-4-З1 ОПК-4-В1 ПК-4-З1 ПК-4-З2 ПК-4-У1 ПК-4-У2 ПК-4-В1	Л1.4 Л1.5 Л1.6 Л1.7			
4.2	Программная реализация разработанных алгоритмов с использованием технологий параллельного и многопоточного программирования. Разработка необходимых программных интерфейсов и экранных форм. Отладка и тестирование корректности работы программы. /Ср/	8	14	УК-1-З1 УК-1-З2 УК-1-У1 УК-1-У2 УК-1-В1 УК-1-В2 ОПК-4-З1 ОПК-4-В1 ПК-4-З1 ПК-4-З2 ПК-4-У1 ПК-4-У2 ПК-4-В1	Л1.1 Л1.2 Л1.3 Л1.4			

4.3	Подготовка текста курсовой работы (в том числе аналитического обзора литературы и специальной части курсовой работы), а также соответствующего презентационного материала. Защита курсовой работы. /Ср/	8	16	УК-1-З1 УК-1-У2 УК-1-В1 УК-1-В2 ОПК-4-З1 ПК-4-З1 ПК-4-З2 ПК-4-У1 ПК-4-У2	Л1.8			

5. ФОНД ОЦЕНОЧНЫХ МАТЕРИАЛОВ

5.1. Вопросы для самостоятельной подготовки к экзамену (зачёту с оценкой)

БЛОК 1: ТЕОРИЯ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ (ОПК-4-З1, ПК-4-З1, ПК-4-З2, ПК-4-У1, УК-1-В1, УК-1-З1, УК-2-З1, УК-2-З2)

- Понятие параллельного алгоритма. Последовательные и параллельные формы алгоритмов. Основные проблемы использования параллельной обработки данных, цели и принципиальные ограничения процедуры распараллеливания.
- Конвейерные и векторные вычисления. Процессорные матрицы. Многопроцессорные вычислительные системы с общей и распределенной памятью. Понятие векторизации вычислений (на GPU - графических процессорах). Проиллюстрировать на примере матричных операций.
- Классификация многопроцессорных систем, систематика Флинна. Потоки данных и команд, распараллеливания на уровне процессора и на уровне операционной системы. Виды многопроцессорных систем и кластеров.
- Концепция неограниченного параллелизма, закон Амдала и следствие из него. Примеры эффективных параллельных форм алгоритмов. Следствие концепции неограниченного параллелизма и принципиальные ограничения по возможностям ускорения программ.
- Модели многопроцессорных систем с общей и распределенной памятью. Модель конвейерной системы (на уровне "процессор-операционная система"). Представление алгоритма в виде графа потока данных, ширина и глубина параллельной формы алгоритма.
- Современные технологии параллельного программирования, разновидности и примеры. Типовые модели программирования и шаблоны. Loop Parallelism. Single Program Multiple Data. Partitioned Global Address Space. Master/Worker. Рекурсивный Fork/Join.
- Классы задач, демонстрирующих "хорошее" распараллеливание. Алгоритмы распараллеливания векторных и матричных операций, алгоритмы решения систем линейных алгебраических уравнений и численного интегрирования обыкновенных дифференциальных уравнений.
- Измерение показателя ускорения параллельной формы алгоритма в сравнении с последовательной. Теоретические приемы оценки ускорения алгоритмов, исследование асимптотической сложности; практические приемы оценки ускорения, сложности непосредственного замера ускорения. Примеры.
- Параллельные алгоритмы, параллельные вычисления и многопоточное программирование. Цели и задачи каждой ветки знаний "о параллельном", их особенности и назначение, области прикладного использования. Основные объекты ветвей и базовые принципы, подходы каждой из указанных ветвей
- Понятие конкурентной гонки различных потоков (нитей) исполнения; теоретические основы и приемы их синхронизации, распределения на подзадачи, сохранение промежуточных результатов, а также последующее агрегирование локальных решений подзадач в решение глобальной задачи. Примеры.
- Выбор количества потоков параллельного исполнения: практические приемы и ориентиры. Зависимость эффективного количества потоков исполнения от конкретной архитектуры ЭВМ, а также особенностей решаемой задачи. Практические примеры.
- Оптимизации циклических конструкций при параллельных вычислений. Требования и ограничения, случаи независимых участков данных, частично-зависимых и рекуррентно-зависимых данных. Примеры параллельного умножения матриц (строчный и блоковый алгоритмы).
- Место и роль операционной системы в процессе создания эффективного параллельного приложения. Процессы и потоки исполнения, их взаимосвязь. Master-поток и slave-потоки исполнения. Понятие балансировки нагрузки на процессор на уровне операционной системы.
- Место и роль компилятора в процессе создания эффективного параллельного приложения. Параллелизм в пределах одного процессорного ядра (т.н. "квазипараллелизм") и контекста выполнения: архитектурные возможности и компиляторные подходы к повышению параллелизма на уровне команд.

БЛОК 2: ТЕХНОЛОГИЯ ПАРАЛЛЕЛЬНОГО ПРОГРАММИРОВАНИЯ (ОПК-4-В1, ОПК-4-З1, ПК-4-В1, ПК-4-У1, ПК-4-У2, УК-1-В1, УК-1-В2, УК-1-З2, УК-1-У1, УК-1-У2, УК-2-В1, УК-2-У2)

- Технология параллельных вычислений OpenMP. Область применения, история развития, реализации. Модель исполнения "ветвление-слияние". Основные директивы и функции OpenMP для языка C++. Опции видимости данных.
- Распределение итераций цикла между потоками в OpenMP, используемые стратегии планирования. Параллельные и последовательные области. Директивы parallel, single, master. Поддержка рекурсивного параллелизма в OpenMP.
- Синхронизация нитей исполнения в OpenMP. Понятие барьера (barrier), его назначение. Примеры использования барьеров. Директива ordered. Критические секции. Директива atomic, примеры использования в прикладных задачах.
- Понятие редукции параллельной задачи. Опция reduction, особенности ее исполнения. Возможности опции reduction для массивов. Адаптация опции reduction к объектам STL на примере std::vector, директива declare.
- Многопоточное программирование в рамках стандарта Openmp. Расписание работы нитей параллельного исполнения. Использование опции schedule при распараллеливании цикла, преодоления дисбаланса нитей исполнения. Примеры.
- Классы переменных параллельного исполнения. Общие данные, разделяемые ресурсы. Управление доступами к переменным с помощью директив shared и private. Примеры использования указанных директив.
- Директивы для распараллеливания вложенных циклов, директива collapse. Балансировка нагрузки во время выполнения циклических конструкций. Примеры эффективного распараллеливания вложенных циклов.
- OpenMP. Основные принципы (fork-join параллелизм на общей памяти, аннотации кода в виде прагм). Компиляторная трансляция OpenMP-конструкций

БЛОК 3: МНОГОПОТОЧНОЕ ПРОГРАММИРОВАНИЕ (ОПК-4-В1, ОПК-4-З1, ПК-4-У2, УК-1-В2, УК-1-З2, УК-2-В1, УК-2-У1, УК-2-У2)

- Понятие одновременности (concurrency). Области применения и проблематика. Способы реализации одновременных систем, процессы и потоки, программный инструментарий. Основные примитивы многопоточного программирования на

C++.
- Типичные ошибки многопоточного программирования и способы их устранения. Состояние гонки (race condition). Безопасность (safety). Взаимное исключение. Взаимная блокировка (deadlock), необходимые условия возникновения. Живучесть (liveness).
- Условная синхронизация, std::condition_variable. Потокобезопасная реализация схемы "Producer-Consumer". Асинхронные вычисления, std::async. Примитивы future и promise.
- Низкоуровневые эффекты и примитивы синхронизации. Операции с неявными барьерами доступа к памяти. Атомарные переменные (std::atomic). Атомарные read-modify-write инструкции процессоров.
- Понятие потоков исполнения (thread). Метод join и свойство joinable, его назначение и принципы работы. Принципы организации многопочтоных программ, примеры.
- Классы std::call_once, его предназначение. Инструменты асинхронных вычислений std::future, std::promise и std::asunc, их назначение и особенности использования. Обработка исключений при асинхронных вычислениях.
- Понятие мьютекса, его назначение. Классы std::lock, std::lock_guard и std::unique_lock. Преимущества и недостатки явных блокировок (locks)
- Понятие взаимной блокировки (deadlock), примеры их возникновения, а также пути избежания взаимных блокировок.

СПИСОК ВОПРОСОВ ДЛЯ КОНТРОЛЬНЫХ РАБОТ (ОПК-4-В1, ОПК-4-З1, ПК-4-У2, УК-1-В1, УК-1-В2, УК-1-З2, УК-2-В1, УК-2-У1, УК-2-У2)
1. Какие конструкции C/C++ и директивы OpenMP можно использовать в контуре директивы секционирования #pragma omp parallel sections?
- Директивы управления нитями исполнения, такие как #pragma omp section, #pragma omp single, #pragma omp master, #pragma omp critical и др.
- Директивы параллельных циклов #pragma omp for; в том числе вложенные параллельные циклы.
- Произвольные участки программного кода без обертки в директивы OpenMP.
- Директивы синхронизации и управления барьерами #pragma omp barrier.
- Вложенные участи секционирования, организованные с помощью директивы #pragma omp parallel sections.
- Вызов системных функций OpenMP в произвольной точке секционированного региона параллельного исполнения.

2. Сколько нитей исполнения по-умолчанию выделяется для обработки параллельного региона, созданного с помощью директивы #pragma omp parallel sections?
- Количество нитей исполнения соответствует значению, установленному в переменной окружения OMP_NUM_THREADS.
- Количество нитей исполнения совпадает с количеством физических ядер в многопроцессорной ЭВМ.
- Количество нитей исполнения совпадает с суммарным количеством ядер (физических и виртуальных) многопроцессорной ЭВМ.
- Количество нитей исполнения всегда должно явно указываться либо с помощью опции num_threads, либо с помощью функции omp_set_num_threads().

3. Производится ли проверка корректности разметки программного кода на языке программирования C/C++ с помощью директив OpenMP на этапе препроцессинга и компиляции?
- Отслеживается корректность всех синтаксических конструкций OpenMP, в случае некорректной разметки выдаются соответствующие сообщения компиляции.
- Корректность синтаксических конструкций OpenMP не отслеживается, каких-либо ошибок компиляции не выдается, ошибки возникают во время исполнения.
- Отслеживается корректность синтаксических конструкций только в части использования препроцессорных директив, корректность использование директивных опций не отслеживается, ошибки возникают во время исполнения.
- Отслеживается корректность использования директив и их вложенности друг в друга, некорректно использованные или не распознанные директивные опции игнорируются.

4. Каково предназначение опций private и shared директивы #pragma omp parallel for?
- Опции позволяют характеризовать переменные нитей исполнения, как внутренние и внешние, на подобии private и public модификаторов доступа в теории объектно-ориентированного программирования.
- Опции используются в качестве инструмента визуальной маркировки различных переменных для разработчика.
- Опция private обеспечивает атомарность операций с переменной (только один поток одновременно работает с переменной), а опция shared – обеспечивает безопасную совместную работу с переменной множеству нитей исполнения.
- Опция private позволяет каждому потоку работать с собственной локальной копией исходной переменной, а опция shared обеспечивает безопасную совместную работу с выбранной переменной.
- Опции private и shared управляют только и только наличием или отсутствием локальной копии исходной переменной у каждой нити исполнения. Безопасность совместной работы множеством потоков не гарантируется.

5. К какому поведению системы приводит использование опции ordered директивы #pragma omp parallel for?
- Опция обозначает, что исполнение настоящего параллельного цикла переходит в исключительно последовательный режим.
- Опция обозначает, в контуре параллельного цикла существуют специальные регионы, которые должны выполниться последовательно, в том время как остальные части параллельного цикла могут выполняться независимо друг от друга.
- Опция позволяет автоматически устранить рекуррентную зависимость данных между последовательными итерациями цикла и произвести последующие вычисления в параллельном режиме.

6. Каково предназначение директивы #pragma omp ordered?
- Директива #pragma omp ordered суть синоним директивы #pragma omp for с примененной опцией ordered.
- Директива используется в качестве обязательной составной части директивы #pragma omp for с используемой опцией ordered.
- Директива используется для упорядочивания регионов исполнения в контуре директивы секционирования #pragma omp parallel sections.
- Директива используется для явной синхронизации нитей исполнения на ровне с барьерной директивой #pragma omp barrier.

7. К какому поведению системы приводит использование опции reduction директивы #pragma omp parallel for?
- Автоматически обеспечивает безопасную совместную работу с выбранной переменной относительно основных скалярных операций.
- Приводит к созданию собственных локальных копий выбранной переменной у каждой нити исполнения, их параллельному вычислению и последующему последовательному объединению.
- Позволяет автоматически выбирать и балансировать количество нитей исполнения в параллельном цикле, добавлять и освобождать нужное количество нитей динамически.

8. Можно ли применять операцию редукции (reduction) для каких-либо объектов, за исключением скалярных встроенных типов (int, double, bool и т.д.)?
- Да
- Нет

9. В чем отличие директив #pragma omp section и #pragma omp single, примененных в контуре директивы #pragma omp parallel sections?
- Отличий нет, использование обеих директив приводит к одинаковому поведению программного кода.
- Директива #pragma omp section может выполняться множеством потоков исполнения, в то время, как директива #pragma omp single – только одним потоком.
- Директивы #pragma omp section и #pragma omp single не могут быть применены одновременно в контуре директивы #pragma omp parallel sections. (Произойдет ошибка компиляции.)
- Директива #pragma omp single выполняется только slave-нитью исполнения, в то время, как директива #pragma omp section может выполняться, как slave-нитью, так и master-нитью.

10. Каково назначение директивы критической секции #pragma omp critical?
- Директива используется для визуальной маркировки важных (специальных) регионов параллельного исполнения.
- Директива выделяет параллельный регион, выполнение которого критически важно провести как можно быстрее. Нити исполнения конкурируют за скорейшее исполнение данной директивы.
- Директива создает регион, который одновременно может исполняться не более, чем одной master- или slave-нитью параллельного исполнения.
- Директива создает регион, который может исполняться только master-нитью исполнения. Slave-нити исполнения не могут захватить выполнение критической секции.

11. Для чего предназначается директива #pragma omp master?
- Для создания параллельного региона, который должен быть исполнен только и только master-нитью.
- Для управления взаимодействием между master-нитью исполнения и множеством slave-нитями.
- Для назначения и переназначения той или иной slave-нити исполнения в качестве новой (текущей) master-нити исполнения.
- Для управления master-нитью исполнения: запуска и приостановки ее выполнения, назначения региона параллельного исполнения и др.

12. Существуют ли какие-либо ограничения по глубине вложенности друг в друга директив #pragma omp parallel for?
- Да
- Нет

13. Для чего служит директивная опция collapse(n)?
- Для явной синхронизации slave-нитей исполнения и их последующего освобождения с помощью директивы #pragma omp master.
- Для автоматического распараллеливания вложенных друг в друга параллельных циклов #pragma omp parallel for.
- Для последовательного объединения отдельных (независимых, полученных в параллельном режиме) результатов операции редукции (reduction).
- Для упорядочивания и структурирования множества регионов параллельного исполнения с помощью директивы #pragma omp section.

14. Для чего предназначается директива #pragma omp barrier?
- Директива является дополнительным регионом параллельного исполнения на ровне с директивами #pragma omp section, #pragma omp critical, #pragma omp atomic и др.
- Директива позволяет упорядочивать нити исполнения в рамках выполнения параллельного цикла директивы #pragma omp parallel for.

- Директива создает точку синхронизации всех нитей исполнения (master- и slave-нитей). Когда все нити достигли точки синхронизации – выполнение продолжается вновь в конкурентном режиме.
- Директива позволяет организовывать фильтрацию различных нитей исполнения по заданному критерию (номеру нити исполнения, времени ее исполнения, исполняемому параллельному региону и др.)

15. Можно ли использовать директиву #pragma omp barrier более одного раза в периметре действия директивы #pragma omp parallel?
- Да
- Нет

16. Какие конструкции технологии OpenMP можно отнести к средствам синхронизации исполнения потоков?
- Директива #pragma omp critical.
- Директива #pragma omp atomic.
- Директива #pragma omp barrier.
- Директива #pragma omp ordered.
- Директива #pragma omp section.
- Директива #pragma omp single.
- Директива #pragma omp for.
- Директива #pragma omp master.

17. Для чего используется директива атомарности #pragma omp atomic?
- Для автоматически обеспечиваемой безопасной совместной записи в выбранную скалярную переменную.
- Для создания региона параллельного исполнения, который может исполняться одновременно только одной нитью исполнения.
- Для объявления выбранного типа данных (класса, в том числе пользовательского) в качестве атомарной единицы OpenMP, воспринимаемой в качестве скалярного встроенного типа данных.
- Для создания региона параллельного исполнения, который может быть выполнен только один раз (атомарно).

18. Какого поведения исполнения параллельного цикла, созданного с помощью директивы #pragma omp parallel for, можно добиться с использованием опции schedule?
- Спецификации распределения выполняемых объема вычислений между различными нитями параллельного исполнения.
- Автоматическое определение оптимального количества потоков исполнения для заданного числа итераций параллельного цикла и текущей глубины вложенности.
- Гибкого перераспределения нагрузки между нитями исполнения, задействованными при выполнении параллельного цикла, а также другими участками и регионами параллельного исполнения.
- Включение режима автоматического выделения дополнительных нитей исполнения в зависимости от текущей загруженности центрального процессора и доступных ресурсов.

19. Пусть T – экземпляр класса std::thread, у которого вызван метод detach(). Можно ли снова присоединиться к потоку T с помощью метода join() и если да, то в каких случаях?
- Подключиться к потоку T нельзя, вызов метода join() в любой точке программы приведет к ошибке.
- Подключиться к потоку T можно, вызов метода join() в произвольной точке программы приведет к запуску нового исполнения потока.
- Подключиться к потоку T можно, перед вызовом метода join() необходимо дополнительно проверить возможность подключения с помощью метода joinable().

20. Пусть T – экземпляр класса std::thread. Каким образом данный поток исполнения связывается с некоторой вычислительной функцией?
- Экземпляр T создается с помощью конструктора по-умолчанию, функция и ее параметры определяются позднее с помощью соответствующих методов.
- Связь между потоком T, рабочей функцией и значениями ее параметров определяется в момент создания потока с помощью экземплярного конструктора.
- Установить связь напрямую невозможно. Следует создать наследный класс от std::thread, в котором переопределить метод run().

21. Верно ли утверждение, что критические секции метода (т.е. участки кода в методе, защищенные механизмом мьютексов) следует делать с максимальным охватом программного кода всего метода?
- Да
- Нет

22. Каким механизмом следует пользоваться для инициализации общих данных в многопоточной программе, исполненной на языке программирования C/C++?
- Классом std::call_once.
- Классом std::mutex.
- Классом std::atomic.
- Классом std::lock_guard.
- Классом std::promise.
- Классом std::future.

- Классом std::async.

23. В какой точке программного кода следует организовывать обработку исключений, испущенных в потоке асинхронного исполнения (async)?
- Обработку исключений следует организовать в рамках потока асинхронного исполнения, обработка исключений за его пределами не предполагается.
- Обработку исключений можно организовать только и только в той точке программы, в которой результаты асинхронного исполнения возвращаются из соответствующего потока.
- Для обработки исключений необходимо создать отдельный связанный поток асинхронного исполнения – обработчик исключений.

24. К какому поведению программного кода приведет исполнение строки кода с объектом std::future, ожидающему исполнения еще не завершившегося потока асинхронного исполнения std::async?
- К приостановке основного потока исполнения и ожидания завершения работы асинхронного потока.
- К ошибке runtime-исполнения, поскольку поток асинхронного исполнения не завершил свою работу и возвращаемые значения не определены.
- К испусканию системного исключения, доступного для обработки в секциях обработки исключений.

25. Для чего используется объект std::promise?
- Объект используется снятия неопределенности на этапе их инициализации работы множества потоков параллельного исполнения.
- Объект используется для организации обмена сообщениями и параметрами между различными потоками во время исполнения (runtime).
- Объект используется для организации inline вычисления в асинхронном режиме. Вычислительная часть помещается в отдельный поток исполнения при запуске программы и вычисляется заблоговременно.

26. Каково назначение объектов с типом std::condition_variable?
- Объект предназначается для блокирования одного потока, пока он не будет оповещен о наступлении некоего события из другого.
- Объект моделирует потокобезопасные внутренние переменные состояния различных потоков исполнения.
- Объект используется для обмена сообщениями и данными между различными потоками параллельного исполнения.
- Объект представляет собой расширенную потокобезопасную версию мьютексов с внутренним состоянием.

27. Верно ли утверждение, что объект с типом std::mutex является наиболее приемлемым для организации многопоточных потокобезопасных программ?
- Да
- Нет

28. Выберите наиболее безопасный метод, позволяющие минимизировать вероятность возникновения т.н. «взаимных блокировок» (deadlock)?
- Использование классов std::mutex с грамотным вызовом методов lock() и unlock().
- Использование классов std::lock_guard в потокобезопасных методах.
- Использование классов std::unique_lock в потокобезопасных методах с грамотным вызовом методов lock() и unlock()

5.2. Перечень работ, выполняемых по дисциплине (модулю, практике, НИР) - эссе, рефераты, практические и расчетно-графические работы, курсовые работы, проекты и др.

туденту предлагается выполнить комплекс мероприятий, направленный на приобретение знаний, навыков и умений по дисциплине "Параллельные вычисления". Предусмотрено 6 лабораторных работ, 2 контрольные работы, тестирование и выполнение курсовой работы по дисциплине.
Лабораторные работы:(ОПК-4-В1, ОПК-4-З1, ПК-4-В1, ПК-4-У1, ПК-4-У2, УК-1-В1, УК-1-В2, УК-1-З2, УК-1-У1, УК-1-У2, УК-2-В1, УК-2-У2)
Лабораторная работа 1. Введение в технологию параллельных вычислений, базовые конструкции, распараллеливание простых циклов.
Лабораторная работа 2. Алгоритмы параллельных векторных и матричных преобразований.
Лабораторная работа 3. Алгоритмы параллельного решение систем линейных алгебраических уравнений.
Лабораторная работа 4. Алгоритмы параллельного интегрирования обыкновенных дифференциальных уравнений.
Лабораторная работа 5. Потокобезопасные объекты. механизмы mutex exclusion, разработка потокобезопасных объектов.
Лабораторная работа 6. Многопоточное программирование. клиент-серверная архитектура приложений, front- и back-потоки исполнения.

Контрольная работа 1. Теоретические основы параллельных вычислений (ОПК-4-З1, ПК-4-З1, ПК-4-З2, ПК-4-У1, УК-1-В1, УК-1-З1, УК-2-З1, УК-2-З2)
Контрольная работа 2. Технологии параллельных вычислений (MPI, OpenMP) ((ОПК-4-В1, ОПК-4-З1, ПК-4-В1, ПК-4-У1, ПК-4-У2, УК-1-В1, , УК-1-З2, УК-1-У1, УК-1-У2, УК-2-В1, УК-2-У2),)

Курсовая работа. Проектирование и разработка алгоритмов параллельных вычислений при решении вычислительно емких задач.(ОПК-4-В1, ОПК-4-З1, ПК-4-В1, ПК-4-У1, ПК-4-У2, УК-1-В1, УК-1-В2, УК-1-З2, УК-1-У1, УК-1-У2, УК-2-В1, УК-2-У2)

5.3. Оценочные материалы, используемые для экзамена (описание билетов, тестов и т.п.)

Экзамен не предусмотрен.

5.4. Методика оценки освоения дисциплины (модуля, практики. НИР)

Для получения зачета обучающийся должен выполнить все работы, предусмотренные в семестре: выполнить и защитить лабораторные работы, а также написать на положительную оценку контрольные работы.

6. УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ

6.1. Рекомендуемая литература

6.1.1. Основная литература

	Авторы, составители	Заглавие	Библиотека	Издательство, год
Л1.1	Галатенко В. А., Бетелин В. Б.	Программирование в стандарте POSIX: Курс лекций: учебное пособие	Электронная библиотека	Москва: Интернет-Университет Информационных Технологий (ИНТУИТ), 2004
Л1.2	Левин М. П.	Параллельное программирование с использованием OpenMP: учебное пособие	Электронная библиотека	Москва: Интернет-Университет Информационных Технологий (ИНТУИТ) |Бином. Лаборатория знаний, 2008
Л1.3	Антонов А. С.	Параллельное программирование с использованием технологии MPI: курс: учебное пособие	Электронная библиотека	Москва: Интернет-Университет Информационных Технологий (ИНТУИТ), 2008
Л1.4	Биллиг В. А.	Параллельные вычисления и многопоточное программирование	Электронная библиотека	Москва: Национальный Открытый Университет «ИНТУИТ», 2016
Л1.5	Бахвалов Н. С., Овчинникова И. М., Шикин Е. В.	Численные методы: анализ, алгебра, обыкновенные дифференциальные уравнения	Электронная библиотека	Москва: Наука, 1975
Л1.6	Демидович Б. П., Марон И. А., Шувалова Э. З., Демидович Б. П.	Численные методы анализа: приближение функций, дифференциальные и интегральные уравнения	Электронная библиотека	Москва: Главная редакция физико-математической литературы, 1967
Л1.7	Николаев Е. И.	Параллельные вычисления: учебное пособие	Электронная библиотека	Ставрополь: Северо-Кавказский Федеральный университет (СКФУ), 2016
Л1.8	Белянкина О. В.	Выпускная квалификационная работа. Требования к структуре, содержанию и оформлению (N 3241): метод. указания	Электронная библиотека	М.: [МИСиС], 2018

6.2. Перечень ресурсов информационно-телекоммуникационной сети «Интернет»

Э1	Официальный сайт фреймворка параллельных вычислений OpenMP	https://www.openmp.org
Э2	OpenMP Application Programming Interface, Version 5.0, November 2018	https://www.openmp.org/wp-content/uploads/OpenMP-API-Specification-5.0.pdf
Э3	Документация Microsoft Docs: Справочные материалы по библиотеке OpenMP	https://docs.microsoft.com/ru-ru/cpp/parallel/openmp/reference/openmp-library-reference?view=vs-2019

6.3 Перечень программного обеспечения

П.1	Microsoft Visual Studio 2015
П.2	LMS Canvas
П.3	MS Teams
П.4	Microsoft Office

6.4. Перечень информационных справочных систем и профессиональных баз данных

7. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ

Ауд.	Назначение	Оснащение

Читальный зал электронных ресурсов		комплект учебной мебели на 55 мест для обучающихся, 50 ПК с доступом к ИТС «Интернет», ЭИОС университета через личный кабинет на платформе LMS Canvas, лицензионные программы MS Office, MS Teams, ESET Antivirus.
Б-907	Учебная аудитория:	1 стационарный компьютер, пакет лицензионных программ MS Office, комплект учебной мебели на 42 посадочных места, демонстрационное оборудование: доска, проектор мультимедийный х 2, экран х 2, колонки
Б-904а	Учебная аудитория:	20 стационарных компьютеров (core i5-3470 8gb RAM), пакет лицензионных программ MS Office, демонстрационное оборудование: доска, проектор мультимедийный, экран, колонки, комплект учебной мебели
Б-902	Учебная аудитория:	12 стационарных компьютеров (2 х core i5-3470 8gb RAM, 10 х ryzen5 2400g 32gb RAM), пакет лицензионных программ MS Office, демонстрационное оборудование: доска, проектор мультимедийный, комплект учебной мебели

8. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ОБУЧАЮЩИХСЯ

Все материалы по дисциплине (лекции, задания на лабораторные и курсовые работы, методические указания, справочный материал и т.д.) в электронной форме размещаются в электронной системе обучения НИТУ «МИСиС» LMS Canvas, где преподавателем создается одноименный курс, на который должен "подписаться" (зарегистрироваться) каждый учащийся. Преподаватель по мере прохождения курса размещает весь необходимый для учащихся материал по предмету в разделах курса, соответствующих рабочей программе дисциплины.

Система Canvas является основным каналом организации взаимодействия между преподавателем и учащимися в часы неаудиторных занятий. Это означает, что весь процесс общения между преподавателем и учащимися не во время аудиторных занятий по данной учебной дисциплине осуществляется только через LMS Canvas. Учащийся обязан постоянно (не менее одного раза в стуки) проверять состояние курса в LMS Canvas, на предмет ознакомления объявлений, получения размещенных преподавателем нового учебного, методического, технического и иного характера.
Доступ к этим материала по логину и паролю для всех студентов предоставляется круглосуточно.

Усвоение учебного материала должно достигаться через глубокое понимание, а не формальное запоминание. Вопросы, которые возникают при изучении литературы, материалов электронных ресурсов и лекционного материала, необходимо обсуждать с лектором на регулярных консультациях.
В овладении предметом большую роль играет самостоятельное выполнение лабораторных работ и курсовой работы.

Лекции читаются в аудиториях с мультимедийным оборудованием с использованием электронных презентаций, представляющих собой опорный иллюстрированный конспект по соответствующей теме. Кроме лекционных материалов преподаватель может рекомендовать к изучению материалы, которые учащийся самостоятельно может получить из перечня профессиональных баз данных и информационных справочных систем (см. соотв. раздел).

Лабораторные работы (ЛР) проводятся в специализированных классах (лабораториях) кафедры инженерной кибернетики. По каждой ЛР проводится защита работы, в ходе которой учащийся демонстрирует полученные результаты, как-то: работоспособность, полноту и качество реализованной функциональности созданного им программного обеспечения; полноту и качество созданной информационной модели знаний по конкретной предметной области, реализованной с использованием научно-практического инструментария заданного класса.
Каждая ЛР оценивается по шкале: «Отлично», «Хорошо», «Удовлетворительно», «Неудовлетворительно».
По каждой ЛР учащийся готовит индивидуальный отчет, в котором в установленной форме описывает поставленную задачу, ход её решения, полученные результаты, их особенности и выводы по работе. Если не оговорено особо, то отчет по ЛР сдается преподавателю в электронной форме.

Для получения итоговой оценки за экзамен учащийся обязан выполнить все заданные лабораторные работы и написать на положительную оценку контрольную работу. В случае, если хотя бы по одному из указанных мероприятий учащийся имеет неудовлетворительную оценку, то учащийся не может быть допущен до экзамена до тех пор, пока имеющаяся задолженность не будет закрыта.

Контрольная работа проводится в часы лабораторных работ на предпоследней неделе семестра. Она оценивается по шкале: «Отлично», «Хорошо», «Удовлетворительно», «Неудовлетворительно». Повторное переписывание контрольной работы допускается только в случае получения учащимся оценки «неудовлетворительно». Для подготовки к контрольным мероприятиям студенту выдается перечень тем, по материалу которых будет контрольное мероприятие. В основном тематика контрольных работ охватывает содержание лекционной части курса. Подготовка к контрольной работе студента возможна как при консультациях в электронной системе обучения МИСиС Canvas, так и при очных консультациях с

преподавателем.

Самостоятельная работа.
Самостоятельная работа обучающихся является формой организации образовательного процесса по дисциплине, стимулирующей активность, самостоятельность и познавательный интерес студентов. Самостоятельная работа обучающихся направлена на углубленное изучение тем дисциплины и предполагает изучение основных и дополнительных источников учебной и научной литературы, выполнение курсовой работы, подготовку отчетов и подготовку к контрольной работе.