

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

УТВЕРЖДЕНО

**Директор высшей школы
программной инженерии
А.В. Малеев**

по дисциплине:	Рабочая программа дисциплины (модуля)
по направлению:	Программная инженерия вычислительных систем с использованием C++
профиль подготовки:	Программная инженерия
	Разработка программно-информационных систем
	высшая школа программной инженерии
	высшая школа программной инженерии
курс:	2
квалификация:	бакалавр

Семестры, формы промежуточной аттестации:

3 (осенний) - Дифференцированный зачет

4 (весенний) - Дифференцированный зачет

Аудиторных часов: 120 всего, в том числе:

лекции: 60 час.

семинары: 60 час.

лабораторные занятия: 0 час.

Самостоятельная работа: 24 час.

Всего часов: 144, всего зач. ед.: 4

Программу составил: И.С. Макаров, старший преподаватель

Программа обсуждена на заседании высшей школы программной инженерии 19.03.2025

Аннотация

В рамках данного курса детально рассматривается стандартная библиотека языка программирования C++ (на уровне стандарта C++17), а также дополнительные библиотеки и инструменты, использующиеся при разработке промышленного программного обеспечения с использованием C++, такие как Boost, SFML, CKB Git и некоторые другие. Особое внимание уделяется параллельному программированию, проектированию параллельных структур данных и алгоритмов, а также межпроцессному и сетевому взаимодействию.

1. Цели и задачи

Цель дисциплины

Целью курса является формирование знаний по применению специализированных библиотек фреймворка Boost и технологий параллельного программирования, а также изучение некоторых специальных возможностей библиотеки STL, которые могут помочь в дальнейшем эффективно использовать C++ при проектировании и разработке программного обеспечения промышленного уровня. Применение этих знаний может помочь в написании более эффективных программ, автоматизации ряда рутинных операций, уменьшении количества ошибок в программах. Целью дисциплины является также повышение культуры программирования, формирование исследовательских навыков и способности применять знания на практике.

Задачи дисциплины

- формирование у обучающихся знаний по применению технологий фреймворка Boost;
- формирование у обучающихся знаний по применению технологий параллельного программирования, в том числе библиотеки многопоточного программирования, входящей в состав STL;
- повышение культуры программирования: умение структурировать текст программы, выделять отдельные модули и правильно связывать их между собой, уметь для решения различных задач применять правильный программный инструментарий;
- формирование умений и навыков применять полученные знания для решения практических задач, в частности, задач компьютерной обработки текста на естественном языке, а также для написания высококачественного кода промышленного уровня.

2. Перечень формируемых компетенций

Освоение дисциплины направлено на формирование следующих компетенций:

Код и наименование компетенции	Индикаторы достижения компетенции
ПК-2 Способен формализовать и алгоритмизировать поставленную задачу	ПК-2.2 Владеет методами и приемами формализации и алгоритмизации задач
	ПК-2.1 Способен формализовать и алгоритмизировать поставленную задачу
ПК-3 Способен проектировать, разрабатывать, интегрировать, проверять на работоспособность программное обеспечение	ПК-3.4 Знает, как определять оптимальные методы и средства проектирования программного обеспечения и структур данных
	ПК-3.3 Умеет излагать основные принципы построения и виды архитектуры программного обеспечения, методы и средства проектирования программного обеспечения, методологии разработки программного обеспечения и технологии программирования
	ПК-3.2 Умеет выбирать языки программирования для написания программного кода с учетом технического задания
	ПК-3.1 Различает синтаксис языков программирования, особенности программирования на этих языках, стандартные библиотеки языков программирования
	ПК-5.1 Умеет описывать архитектуру, устройство и функционирование информационных систем

ПК-5 Способен проектировать, разрабатывать, внедрять, сопровождать и снимать с эксплуатации информационные системы	ПК-5.2 Умеет определять оптимальные методы и инструменты разработки, внедрения и адаптации прикладного программного обеспечения информационных систем
	ПК-5.3 Умеет разрабатывать прототип информационных систем в соответствии с требованиями и проводить его тестирование для проверки корректности архитектурных решений

3. Перечень планируемых результатов обучения по дисциплине (модулю)

В результате освоения дисциплины обучающиеся должны

знать:

- фундаментальные принципы и техники проектирования и разработки программного обеспечения промышленного уровня;
- возможности некоторых специализированных библиотек фреймворка Boost;
- технологии параллельного программирования, основанные на взаимодействии между процессами и потоками;
- примитивы синхронизации потоков и средства обмена данными между потоками.

уметь:

- использовать некоторые необходимые при проектировании промышленного ПО технологии на базе фреймворка Boost, например, Boost.Log, Boost.Filesystem, Boost.Locale и др.;
- интегрировать код, написанный на других языках программирования (например, на Python 3), в приложение на C++, в частности, с помощью Python C/C++ API и библиотеки Boost.Python;
- разрабатывать динамические библиотеки с помощью WinAPI и библиотеки Boost.DLL;
- решать задачи по анализу и обработке структурированного и неструктурированного текста на естественном языке, в частности, писать лексические анализаторы и генераторы на базе Boost.Spirit, использовать алгоритмы для работы со строками и выполнять разбиение текста с Boost.Tokenizer;
- проектировать приложение и разрабатывать алгоритмы (в том числе и алгоритмы NLP) с расчетом на их исполнение в многопоточном/многопроцессном режиме;
- использовать примитивы синхронизации для организации многопоточных вычислений;
- использовать средства межпроцессного взаимодействия на базе Boost.Interprocess и Boost.MPI;
- применять полученные знания для проектирования и разработки ПО промышленного уровня;
- применять полученные знания при разработке высокопроизводительных систем;
- применять полученные знания при разработке алгоритмов и средств NLP.

владеть:

Терминологией и основными инструментами фреймворка Boost и библиотек параллельного программирования, представленных в Boost и STL.

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

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

№	Тема (раздел) дисциплины	Трудоемкость по видам учебных занятий, включая самостоятельную работу, час.			
		Лекции	Семинары	Лаборат. работы	Самост. работа
1	Повторение ядра C++	2	2		1
2	Интеллектуальные указатели	4	4		1
3	Последовательные контейнеры	4	4		1
4	Ассоциативные и неупорядоченные контейнеры	4	4		1
5	Алгоритмы стандартной библиотеки	4	4		2

6	Строки	4	4		2
7	Форматы хранения и обмена данными	4	4		2
8	Многопоточность	4	4		2
9	Средства синхронизации	6	6		3
10	Межпроцессное взаимодействие	8	8		3
11	Сетевое взаимодействие	8	8		3
12	Мультимедийная библиотека SFML	8	8		3
Итого часов		60	60		24
Подготовка к экзамену		0 час.			
Общая трудоёмкость		144 час., 4 зач.ед.			

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

Семестр: 3 (Осенний)

1. Повторение ядра C++

Повторение ядра C++, подготовка Boost к работе в MSVS, изучение основных команд Git в SmartGit

2. Интеллектуальные указатели

Интеллектуальные указатели, аллокаторы, итераторы, работа со стандартной библиотекой chrono

3. Последовательные контейнеры

Последовательные контейнеры стандартной библиотеки, пары и кортежи, циклический буфер Boost

4. Ассоциативные и неупорядоченные контейнеры

Ассоциативные и неупорядоченные контейнеры стандартной библиотеки, хэш-таблицы, Boost.Multiindex

5. Алгоритмы стандартной библиотеки

Алгоритмы стандартной библиотеки, итераторы, генераторы случайных чисел C++11, Boost Graph Library

6. Строки

Строки, обработка структурированного текста, регулярные выражения, генераторы и парсеры Boost Spirit

7. Форматы хранения и обмена данными

Форматы хранения и обмена данными JSON, XML, работа с файловой системой, потоки ввода-вывода

8. Многопоточность

Многопоточность стандартной библиотеки, механизм будущих результатов, параллельные алгоритмы

9. Средства синхронизации

Средства синхронизации, мьютексы, условные переменные, параллельные структуры данных, АТД

10. Межпроцессное взаимодействие

Межпроцессное взаимодействие, разделяемая память, memory mapping, средства синхронизации.

11. Сетевое взаимодействие

Сетевое взаимодействие на базе Boost.Asio, основы TCP/IP, endpoint, socket, разрешение DNS имен, операции ввод-вывода, синхронные и асинхронные операции.

12. Мультимедийная библиотека SFML

Мультимедийная библиотека SFML, разработка разнотипных игровых приложений, сапер, змейка, математическое моделирование отдельных физических явлений.

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

Компьютерный класс, оснащенный мультимедиа-проектором и экраном или телевизором.

6. Перечень рекомендуемой литературы

Основная литература

1. Язык программирования C++ [Текст] = The C++ Programming Language, [учеб. пособие для вузов] /Бьерн Страуструп ; пер. с англ. под ред. Н. Н. Мартынова. -М., БИНОМ, 2017

Литература, рекомендованная для самостоятельного изучения:

1. Страуструп, Б. Язык программирования C++ для профессионалов / Б. Страуструп .— 2-е изд., испр. — Москва : ИНТУИТ, 2016 .— Электрон. версия печ. публикации.

2. Саттер, Г. Стандарты программирования на C++ [Текст] = C++ Coding Standards : 101 правило и рекомендация / Г. Саттер, А. Александреску; пер. с англ. И. В. Красикова .— [Научное изд.] .— М.; СПб; Киев : Изд. дом "Вильямс", 2015 .— 224 с.

3. Александреску, А. Современное проектирование на C++ [Текст] = Modern C++ Design : Обобщенное программирование и прикладные шаблоны проектирования / А. Александреску; пер. с англ. Д. А. Ключина .— [Научное изд.] .— М.; СПб; Киев : Изд. дом "Вильямс", 2015 .— 336 с.

Дополнительная литература

Фонд библиотеки МФТИ:

1. Немцова, Т. И. Программирование на языке высокого уровня. Программирование на языке C++ : учебное пособие / Т.И. Немцова, С.Ю. Голова, А.И. Терентьев ; под ред. Л.Г. Гагариной. — Москва : ФОРУМ : ИНФРА-М, 2024. — 512 с. + Доп. материалы [Электронный ресурс]. — (Среднее профессиональное образование). - ISBN 978-5-8199-0699-6. - Текст : электронный. - URL: <https://znanium.com/catalog/product/2083383>

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

1. <http://www.cplusplus.com/>
2. <https://www.boost.org/doc/libs/>

3. <https://ru.stackoverflow.com/>
4. <https://en.cppreference.com/>

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

На занятиях используются мультимедийные технологии, включая демонстрацию презентаций.

В процессе самостоятельной работы обучающихся возможно использование таких программных средств, как Microsoft Visual Studio, Jupiter Notebooks, Anaconda и др.

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

Обучающийся курсу должен освоить стандартную библиотеку языка программирования C++ (на уровне стандарта C++17), а также дополнительные библиотеки и инструменты, используемые при разработке промышленного программного обеспечения с использованием C++, такие как Boost, SFML, СКВ Git и некоторые другие, и научиться применять полученные знания на практике.

Освоение курса не сводится только к посещению занятий. Основой успешного прохождения курса является самостоятельная работа обучающегося, которая включает в себя:

- выполнение еженедельных домашних заданий;
- проработку примеров программ с занятий;
- изучение дополнительных материалов по монографиям, статьям и справочникам.

Руководство и контроль за самостоятельной работой студента осуществляется в форме индивидуальных консультаций.

Показателем владения материалом служит умение решать задачи. Для формирования умения применять теоретические знания на практике студенту необходимо решать как можно больше задач. При решении задач стоит акцентировать внимание на качестве написанного кода и эффективности алгоритмов и структур данных.

При подготовке к семинарам необходимо повторять ранее изученные основные понятия. В начале занятия, как правило, проводится короткое (5-10 минут) повторение материала прошедших занятий. Обычно придерживаются следующей схемы: изучение материала занятия по файлам, выложенным преподавателем, сразу после занятия (10-15 минут); повторение материала накануне следующего занятия (10-15 минут), проработка учебного материала по файлам занятия, учебной литературе, подготовка домашнего задания или курсовых проектов (1,5-2,5 часа в неделю). Важно добиться понимания изучаемого материала, а не механического его запоминания. При затруднении изучения отдельных тем, вопросов, следует обращаться за консультациями к преподавателю.

Выполнение домашних заданий является обязательным. Домашние задания могут быть частично или полностью заменены по решению преподавателя на несколько курсовых проектов. Способ оформления и отправки работ сообщается преподавателем отдельно.

ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

по направлению:	Программная инженерия
профиль подготовки:	Разработка программно-информационных систем высшая школа программной инженерии высшая школа программной инженерии
курс:	<u>2</u>
квалификация:	бакалавр

Семестры, формы промежуточной аттестации:

- 3 (осенний) - Дифференцированный зачет
- 4 (весенний) - Дифференцированный зачет

Разработчик: И.С. Макаров, старший преподаватель

1. Компетенции, формируемые в процессе изучения дисциплины

Код и наименование компетенции	Индикаторы достижения компетенции
ПК-2 Способен формализовать и алгоритмизировать поставленную задачу	ПК-2.2 Владеет методами и приемами формализации и алгоритмизации задач
	ПК-2.1 Способен формализовать и алгоритмизировать поставленную задачу
ПК-3 Способен проектировать, разрабатывать, интегрировать, проверять на работоспособность программное обеспечение	ПК-3.4 Знает, как определять оптимальные методы и средства проектирования программного обеспечения и структур данных
	ПК-3.3 Умеет излагать основные принципы построения и виды архитектуры программного обеспечения, методы и средства проектирования программного обеспечения, методологии разработки программного обеспечения и технологии программирования
	ПК-3.2 Умеет выбирать языки программирования для написания программного кода с учетом технического задания
	ПК-3.1 Различает синтаксис языков программирования, особенности программирования на этих языках, стандартные библиотеки языков программирования
ПК-5 Способен проектировать, разрабатывать, внедрять, сопровождать и снимать с эксплуатации информационные системы	ПК-5.1 Умеет описывать архитектуру, устройство и функционирование информационных систем
	ПК-5.2 Умеет определять оптимальные методы и инструменты разработки, внедрения и адаптации прикладного программного обеспечения информационных систем
	ПК-5.3 Умеет разрабатывать прототип информационных систем в соответствии с требованиями и проводить его тестирование для проверки корректности архитектурных решений

2. Показатели оценивания компетенций

В результате изучения дисциплины «Программная инженерия вычислительных систем с использованием C++» обучающийся должен:

знать:

- фундаментальные принципы и техники проектирования и разработки программного обеспечения промышленного уровня;
- возможности некоторых специализированных библиотек фреймворка Boost;
- технологии параллельного программирования, основанные на взаимодействии между процессами и потоками;
- примитивы синхронизации потоков и средства обмена данными между потоками.

уметь:

- использовать некоторые необходимые при проектировании промышленного ПО технологии на базе фреймворка Boost, например, Boost.Log, Boost.Filesystem, Boost.Locale и др.;
- интегрировать код, написанный на других языках программирования (например, на Python 3), в приложение на C++, в частности, с помощью Python C/C++ API и библиотеки Boost.Python;
- разрабатывать динамические библиотеки с помощью WinAPI и библиотеки Boost.DLL;
- решать задачи по анализу и обработке структурированного и неструктурированного текста на естественном языке, в частности, писать лексические анализаторы и генераторы на базе Boost.Spirit, использовать алгоритмы для работы со строками и выполнять разбиение текста с Boost.Tokenizer;
- проектировать приложение и разрабатывать алгоритмы (в том числе и алгоритмы NLP) с расчетом на их исполнение в многопоточном/многопроцессном режиме;
- использовать примитивы синхронизации для организации многопоточных вычислений;
- использовать средства межпроцессного взаимодействия на базе Boost.Interprocess и Boost.MPI;
- применять полученные знания для проектирования и разработки ПО промышленного уровня;
- применять полученные знания при разработке высокопроизводительных систем;
- применять полученные знания при разработке алгоритмов и средств NLP.

владеть:

Терминологией и основными инструментами фреймворка Boost и библиотек параллельного программирования, представленных в Boost и STL.

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

Примеры контрольных вопросов для текущего контроля освоения материала:

- 1) Когда используются контейнеры типа множество и отображение?
- 2) Каким требованиям должна удовлетворять качественная хэш-функция?
- 3) Из-за чего в хэш-таблицах возникают коллизии и как можно их разрешить?
- 4) Почему сложность основных операций хэш-таблиц в худшем случае $O(N)$?
- 5) Что позволяет сделать инструмент создания контейнеров в Boost.Multiindex?

Примеры упражнений на проверку знаний:

- 1) Предложите хэш-функцию для хэширования чисел с плавающей точкой и определите ее свойства.
- 2) Исследуйте экспериментально размерность хэш-функции из Boost. Постройте график зависимости числа коллизий от кол-ва экземпляров.
- 3) Операция вставки в контейнер `std::set` имеет сложность $O(\log N)$. Сгенерируйте N случайных чисел и добавьте их в контейнер. С помощью таймера оцените асимптотику алгоритма.

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

Перечень контрольных вопросов для сдачи дифференцированного зачета в осеннем семестре

1. Объяснить отличия потока и процесса.
2. Что такое аппаратный параллелизм?
3. Какие средства многопоточного программирования использовались программистами до выхода C++11?
4. Что такое контекстное переключение и когда оно появляется?
5. Какие есть подходы к организации параллелизма?
6. Какие преимущества и недостатки у параллелизма за счет нескольких процессов?
7. Какие преимущества и недостатки у параллелизма за счет нескольких потоков?
8. Зачем нужен параллелизм?
9. Что такое идиома RAII и чем она полезна?
10. Как создать поток и какие аргументы можно передать конструктору потока?
11. Как заставить поток работать в фоновом режиме?
12. Как гарантировать, что поток завершается до выхода из функции на всех возможных путях исполнения, как нормальных, так и в результате исключения?

13. Как корректно передать параметр функции потока по ссылке?
14. Почему без обертки типа `std::ref` аргумент не будет передан в функцию потока по ссылке? Как он будет передан?
15. Какие служебные операции (конструкторы/операторы) стало возможным использовать со стандарта C++11? Что они делают?
16. Привести примеры перемещаемых, но не копируемых типов.
17. Какие типы умных указателей имеются в стандартной библиотеке C++?
18. Как узнать число потоков, которые могут работать на данной машине по-настоящему параллельно?
19. При какой работе нескольких потоков с разделяемыми данными не возникает никаких проблем и не требуются дополнительные средства синхронизации?
20. Что такое инвариант применительно к программированию?
21. Что такое состояние гонки и как его избежать?
22. Что такое мьютекс? Какие проблемы могут появиться при использовании мьютексов?
23. Что такое взаимоблокировка? Как ее можно достичь?
24. Как можно "пробить брешь в защите" данных мьютексом, т.е. модифицировать в сторонней функции данные, находящиеся под защитой мьютекса? Как не допустить такой ситуации?
25. Объяснить состояние гонки, внутренне присущей интерфейсу класса `std::stack`. Как можно его избежать?
26. Как обеспечить отсутствие взаимоблокировки, когда требуется захватить несколько мьютексов?
27. Сколько мьютексов может одновременно захватить функция `std::lock`? Какой механизм языка позволяет это сделать?
28. Что такое `std::lock_guard`, почему используется он, а не методы `lock-unlock` мьютекса выбранного класса?
29. Объяснить синтаксис оформления лямбда выражения. Когда удобно использовать лямбда-выражения?
30. Какие средства позволяют определять свойства типов? Привести примеры.
31. Что такое идиома SFINAE? Как она реализуется в C++?
32. Что такое `rvalue` ссылки, чем они отличаются от `lvalue` ссылок?
33. Как написать ОДИН конструктор, который может принимать N параметров, каждый из которых может быть как `rvalue`, так и `lvalue`? Что означает понятие "универсальная ссылка"? Подсказка: что означает прямая передача и как реализуется идиома SFINAE в C++?
34. Что такое гранулярность блокировки и почему важно правильно ее выбирать?
35. Что такое "паттерн с двойной проверкой" и почему он печально известен?
36. Как корректно осуществить однократный вызов некоторой функции в многопоточном приложении?
37. Перечислите все типы мьютексов, которые упоминались на семинарах. Поясните за каждый из них.
38. Что такое `condition variable`, какие основные методы имеет этот класс и как их использовать?
39. Как корректно "изготавливать" умные указатели? Привести пример, когда применение `new` может привести к утечке памяти.

Перечень контрольных вопросов для сдачи дифференцированного зачета в весеннем семестре:

1. Как запустить асинхронную задачу? Какие аргументы можно передать конструктору асинхронной задачи? В чем отличие от аргументов, передаваемых конструктору потока?
2. В чем преимущества использования механизма асинхронных задач перед явным созданием потоков посредством `std::thread`?
3. Что такое механизм будущих результатов? Как он реализован в C++ (какие методы, как инициализируется)?
4. Какие средства генерации случайных чисел предоставляет библиотека C++11?
5. Какие четыре элемента информации должен предоставлять класс часов?
6. Что такое стабильные часы? Когда их надо использовать? Какие есть еще часы?
7. Как замерить время работы фрагмента кода с помощью библиотеки C++11?

8. Объяснить алгоритм quicksort в стиле функционального программирования с использованием механизма асинхронных задач.
9. Какая связь между данными, объектами и ячейками памяти? Рассмотреть разные типы объектов и как они хранятся (число, строка, перечисление).
10. Что такое атомарные операции и атомарные типы данных?
11. Используют ли операции атомарных типов внутреннюю блокировку? Как проверить наличие блокировки? Какой тип гарантированно не имеет блокировок на всех системах?
12. Какие функции-члены присутствуют в классе `std::atomic < T >` с любым типом `T`? Какие методы и операторы добавляются, если этот тип - интегральный или тип указателя?
13. На какие три категории разбиты операции работы с атомарными типами данных?
14. Описать, как работают операции типа сравнить-и-обменять. Привести пример использования такой операции.
15. Как с помощью атомарных операций реализовать конструктор, который может быть вызван только однократно?
16. Назвать варианты упорядочения доступа к памяти.
17. Какое упорядочение памяти подразумевается по умолчанию в атомарных операциях?
18. Что такое последовательно согласованное упорядочение доступа к памяти? Чем оно хорошо, какие накладные расходы появляются?
19. Что можно сказать об относительном порядке операций в разных потоках при ослабленном упорядочении?
20. Что такое барьеры (идейно), чего они позволяют достичь, как их создавать?
21. Что дает модель упорядочения `acquire-release` (объяснить, результаты каких операций и где будут видны)? Аналогично с каким механизмом можно провести для использования данной модели?
22. Какая структура данных называется потокобезопасной (объяснить подробно)?
23. В чем недостаток проектирования, к примеру, потокобезопасной очереди, основанной на `std::queue`? Как следует изменить подход к проектированию?
24. Какие вопросы следует задавать себе при проектировании параллельных структур данных без блокировок?
25. Какие структуры данных называются неблокирующими?
26. Какие структуры данных называются свободными от блокировок?
27. Какие структуры данных называются свободными от ожидания?
28. Какие плюсы и минусы имеют структуры данных без блокировок?
29. Что такое активная блокировка, по каким причинам она возникает?
30. Что такое рекурсивное распределение данных (привести пример алгоритма, использующего эту схему)?
31. Объяснить явление перебрасывания кэша и ложного разделения на примере массива, каждый элемент которого обрабатывается отдельным исполнителем. Как избежать этой проблемы?
32. Как механизм будущих результатов помогает делать код безопаснее относительно исключений?
33. Что понимается под масштабируемостью? Сформулируйте закон Амдала.
34. Как корректно реализовать идиому `rImpl`? Почему плохо использовать встроенные указатели в `си`-стиле и как добиться соблюдения логической константности?
35. Что такое пул потоков, как работает простейший пул?
36. Как обеспечить ожидание завершения задач в пуле потоков?

Критерии оценивания

Оценка "отлично (10)" выставляется студенту, показавшему всесторонние, систематизированные, глубокие знания учебной программы дисциплины, проявляющему интерес к данной предметной области, продемонстрировавшему умение уверенно и творчески применять их на практике при решении конкретных задач, свободное и правильное обоснование принятых решений.

Оценка "отлично (9)" выставляется студенту, показавшему всесторонние, систематизированные, глубокие знания учебной программы дисциплины и умение уверенно применять их на практике при решении конкретных задач, свободное и правильное обоснование принятых решений.

Оценка "отлично (8)" выставляется студенту, показавшему всесторонние, систематизированные, глубокие знания учебной программы дисциплины и умение уверенно применять их на практике при решении конкретных задач, правильное обоснование принятых решений, с некоторыми недочетами.

Оценка "хорошо (7)" выставляется студенту, если он твердо знает материал, грамотно и по существу излагает его, умеет применять полученные знания на практике, но недостаточно грамотно обосновывает полученные результаты.

Оценка "хорошо (6)" выставляется студенту, если он твердо знает материал, грамотно и по существу излагает его, умеет применять полученные знания на практике, но допускает в ответе или в решении задач некоторые неточности.

Оценка "хорошо (5)" выставляется студенту, если он в основном знает материал, грамотно и по существу излагает его, умеет применять полученные знания на практике, но допускает в ответе или в решении задач достаточно большое количество неточностей.

Оценка "удовлетворительно (4)" выставляется студенту, показавшему фрагментарный, разрозненный характер знаний, недостаточно правильные формулировки базовых понятий, нарушения логической последовательности в изложении программного материала, но при этом он освоил основные разделы учебной программы, необходимые для дальнейшего обучения, и может применять полученные знания по образцу в стандартной ситуации.

Оценка "удовлетворительно (3)" выставляется студенту, показавшему фрагментарный, разрозненный характер знаний, допускающему ошибки в формулировках базовых понятий, нарушения логической последовательности в изложении программного материала, слабо владеет основными разделами учебной программы, необходимыми для дальнейшего обучения и с трудом применяет полученные знания даже в стандартной ситуации.

Оценка "неудовлетворительно (2)" выставляется студенту, который не знает большей части основного содержания учебной программы дисциплины, допускает грубые ошибки в формулировках основных принципов и не умеет использовать полученные знания при решении типовых задач.

Оценка "неудовлетворительно (1)" выставляется студенту, который не знает основного содержания учебной программы дисциплины, допускает грубейшие ошибки в формулировках базовых понятий дисциплины и вообще не имеет навыков решения типовых практических задач.

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

Дифференцированный зачет проводится по итогам текущей успеваемости и сдачи практических заданий, предусмотренных программой дисциплины, путем организации специального опроса, проводимого в устной или письменной форме, а также защитой выпускного проекта.