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

УТВЕРЖДЕНО
Проректор по учебной работе и довузовской подготовке
А.А. Воронов

Рабочая программа дисциплины (модуля)

по дисциплине: Практика программирования с использованием C++
по направлению: Электроника и наноэлектроника
профиль подготовки: Микро- и наноэлектроника
Физтех-школа Электроники, Фотоники и Молекулярной Физики
кафедра информатики и вычислительной математики
курс: 2
квалификация: бакалавр

Семестры, формы промежуточной аттестации:
3 (осенний) - Дифференцированный зачет
4 (весенний) - Дифференцированный зачет

Аудиторных часов: 150 всего, в том числе:
лекции: 30 час.
семинары: 0 час.
лабораторные занятия: 120 час.

Самостоятельная работа: 210 час.

Всего часов: 360, всего зач. ед.: 8

Количество контрольных работ, заданий: 2

Программу составил: И.С. Макаров, ассистент

Программа обсуждена на заседании кафедры информатики и вычислительной математики 06.02.2020

Аннотация

Годовой курс "Практика программирования с использованием C++" предназначен для студентов второго курса. В первом семестре студенты изучают основы C++ - функции, классы, шаблоны и др., понимание которых необходимо для изучения существующих программ и разработки собственных. В рамках второго семестра детально рассматривается стандартная библиотека C++, а также дополнительные библиотеки и инструменты, использующиеся при разработке промышленного программного обеспечения с использованием C++, такие как Boost, SFML, СКВ Git и др. Особое внимание уделяется параллельному программированию, проектированию параллельных структур данных и алгоритмов, а также межпроцессному и сетевому взаимодействию. На семинарах 12 и 13 рассматриваются дополнительные темы (по умолчаию работа с графической библиотекой SFML), которые могут быть изменены по желанию студентов и/или кафедр. Для закрепления материала семинаров студенты выполняют еженедельные домашние задания (см. пример в приложении). В конце семестра студенты сдают зачет, состоящий из теоретической и практической частей. 80% итоговой оценки формируется в течение семестра, 20% - на зачете.

1. Цели и задачи

Цель дисциплины

Освоение студентами методов проектирования и программирования объектно-ориентированных программ на языке C++, а также знаний в области применения современных высокопроизводительных комплексов различной архитектуры в научных исследованиях и прикладных областях, в том числе при математическом моделировании и обработке больших массивов данных.

Задачи дисциплины

- Обучение студентов принципам создания программных комплексов, выявление особенностей их создания в парадигме объектно-ориентированного программирования;
- обучение студентов принципам создания эффективных параллельных алгоритмов и программ, знакомство с основными методами и принципами параллельного программирования, основными технологиями параллельного программирования;
- формирование подходов к выполнению исследований студентами в области математического моделирования и численных методов с использованием современных технологий, и программных средств параллельного программирования в рамках выпускных работ на степень бакалавра.

2. Перечень формируемых компетенций

Освоение дисциплины направлено на формирование следующих компетенций:

Код и наименование компетенции	Индикаторы достижения компетенции
ПК-1 Способнен планировать и проводить научные эксперименты (в избранной предметной области) и (или) теоретические (аналитические и имитационные) исследования	ПК-1.8 Владеет навыками работы с современными языками программирования и программными пакетами для научных расчетов

3. Перечень планируемых результатов обучения по дисциплине (модулю)

В результате освоения дисциплины обучающиеся должны

знать:

- Общие понятия о структурах данных: стеки, очереди, списки, деревья, таблицы;
- теоретические подходы к созданию комплексов программ;
- идеологию объектно-ориентированного подхода;
- современные проблемы моделирования, численных методов и создания комплексов программ;
- основы архитектуры электронно-вычислительной машины (ЭВМ), представление информации в ЭВМ и архитектурные принципы повышения их производительности.
- основные принципы устройства и работы операционной системы;
- основы работы с пакетами прикладных программ в области математики и физики.

уметь:

- Применять объектно-ориентированный подход для написания программ;
- использовать современные средства создания комплексов программ;
- создавать безопасные программы, используя современные средства для отладки программ;
- выбирать оптимальные алгоритмы для современных программ;
- планировать оптимальное проведение численного эксперимента;
- абстрагироваться от несущественного при математическом моделировании;
- использовать сигналы и сообщения для взаимодействия процессов между собой и с операционной системой;
- разрабатывать полные законченные программы на одном из языков высокого уровня как индивидуально, так и в команде, с использованием современных средств написания и отладки программ;
- использовать знания по информатике для приложений в инновационной, конструкторско-технологической и производственно-технологической сферах деятельности.

владеть:

- Навыками самостоятельной работы в среде объектно-ориентированного программирования на языке C++;
- навыками освоения современных архитектур ЭВМ;
- навыками программирования с использованием средств операционной системы для решения исследовательских задач;
- объективной картиной теории и практики объектно-ориентированного программирования.

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	Типы данных.	2		4	6
2	Указатели и массивы.	2		4	6
3	Функции и лямбды.			4	8
4	Структуры и перечисления.	2		4	6
5	Введение в классы.			4	8
6	Проектирование классов.	2		4	6
7	Иерархии классов.			4	8
8	Обработка ошибок.	2		4	6
9	Шаблоны.			4	8
10	Вариативные шаблоны.	2		4	6
11	Семантика перемещения.			4	8
12	Метапрограммирование.	2		4	6
13	Повторение.	1		12	23
14	Стандартная библиотека.	2		4	6
15	Интеллектуальные указатели.	2		4	6
16	Последовательные контейнеры.	2		4	6
17	Ассоциативные контейнеры.			4	8
18	Алгоритмы STL.	2		4	6
19	Обработка текста.			4	8
20	Ввод и вывод.	2		4	6
21	Параллельное программирование.			4	8
22	Примитивы синхронизации.	2		4	6
23	Межпроцессное взаимодействие.			4	8
24	Сетевое взаимодействие.	2		4	6

25	Графическая библиотека SFML.			4	8
26	Повторение.	1		12	23
Итого часов		30		120	210
Подготовка к экзамену		0 час.			
Общая трудоёмкость		360 час., 8 зач.ед.			

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

Семестр: 3 (Осенний)

1. Типы данных.

Философия языка. История развития. Стандарты ISO. IDE Microsoft Visual Studio. Препроцессор, компилятор и компоновщик. Простейшая программа. Объекты и значения. Правила именования. Объявление и определение. Инициализация. Фундаментальные типы данных. Приведения типов. Проблема переносимости. Псевдонимы. Условные выражения. Циклы.

2. Указатели и массивы.

Ячейки памяти. Адреса и указатели. Сегменты адресного пространства процесса. Глобальные и локальные объекты. Встроенные статические массивы. Выделение памяти в куче. Встроенные динамические массивы. Проблемы освобождения памяти. Связь встроенных массивов и указателей. Арифметика указателей. Свойства С-строк. Многомерные массивы. Ссылки.

3. Функции и лямбды.

Функции. Объявление и определение. Forward declaration. Встроенные функции. Атрибуты функций. Передача аргументов по значению, указателю и ссылке. Значения аргументов по умолчанию. Статические переменные. Перегрузка функций. Указатели на функции. Лямбда-функции. Рекурсия. Функциональное программирование. Макросы. Условная компиляция.

4. Структуры и перечисления.

Структуры. Данные-члены. Агрегатная инициализация. Plain Old Data. Конструкторы. Функции-члены. Выравнивание в памяти. Битовые поля. Объединения. Перечисления без области видимости. Перечисления с областью видимости. Модульность программы. Глобальная область видимости. Пространства имен. Свойство аддитивности пространства имен. Поиск Кёнига.

5. Введение в классы.

Классы. Инвариант. Интерфейс и реализация. Инкапсуляция. Секции класса. Инициализация полей класса. Специальные функции-члены класса. Конструктор и деструктор. Статические члены класса. Друзья класса. Идиома RAII. Объектно-ориентированное программирование. Раздельная компиляция. Правило одного определения. Внутреннее и внешнее связывание.

6. Проектирование классов.

Проектирование классов. Перегрузка операторов. Концепции rvalue и lvalue. Идентифицируемость и перемещаемость. Классификация значений. rvalue-ссылки и lvalue-ссылки. Универсальные ссылки. Копирование и перемещение. Перемещающие специальные функции-члены. Правила генерации специальных функций-членов. Поверхностное и глубокое копирование.

7. Иерархии классов.

Наследование. Иерархии классов. Принцип подстановки Лисков. Наследование интерфейса и реализации. Открытое и закрытое наследование. Композиция. Полиморфизм. Виртуальные функции. Динамический и статический типы объекта. Абстрактный базовый класс. Чистая виртуальная функция. Множественное наследование. Виртуальные базовые классы.

8. Обработка ошибок.

Обработка ошибок. Коды возврата. Механизм исключений. Правило 80/20. Гарантии безопасности исключений. Идиома RAII. Стандартная иерархия исключений. Проектирование пользовательского класса исключений. Особенности перехвата исключений. Практика использования отладчика IDE Microsoft Visual Studio. Создание и анализ дампов. Логирование.

9. Шаблоны.

Шаблоны функций. Инстанцирование шаблона. Двухэтапная трансляция шаблона. Перегрузка шаблонов функций. Шаблоны классов. Полная и частичная специализации шаблона. Низводящие преобразования. Значения параметров шаблона по умолчанию. Нетиповые параметры шаблона. Обобщенное программирование. Шаблоны псевдонимов. Шаблоны переменных.

10. Вариативные шаблоны.

Вариативные шаблоны. Пакет параметров шаблона. Пакет параметров функции. Рекурсивная обработка пакета вариативного шаблона. Выражения свертки. Вариативные выражения. Вариативные индексы. Вариативные базовые классы. Статический полиморфизм. Странно рекурсивный шаблон проектирования. Метод Бартона-Нэкмана. Фасады. Миксины.

11. Семантика перемещения.

Семантика перемещения в шаблонах. Свойства модифицируемости, константности и перемещаемости параметров шаблонов. Вывод типа в шаблонах. Идеальная передача. Универсальные ссылки. Шаблоны специальных функций-членов. Идиома SFINAE. Управляемое отключение шаблонов. Передача по ссылке и значению. Шаблоны свойств и преобразований типов.

12. Метапрограммирование.

Метапрограммирование шаблонов. Полнота языка шаблонов по Тьюрингу. Вычисления во время компиляции. Нетиповые параметры шаблонов. Специализация шаблонов. Рекурсивное инстанцирование шаблонов. Использование ключевого слова constexpr. Условный оператор if времени компиляции. Гибридное метапрограммирование. Класс ratio времени компиляции.

13. Повторение.

Предельное быстродействие векторных программ. Две части программ — скалярная и векторная. Дополнительные затраты на организацию векторных вычислений во время работы программы. Ограниченное число векторных регистров. Ограничения на используемые операторы в векторизуемых циклах.

Семестр: 4 (Весенний)

14. Стандартная библиотека.

Повторение. Стандартная библиотека. Библиотеки Boost. Другие библиотеки. Настройка проекта в IDE Microsoft Visual Studio. Этапы жизненного цикла программного обеспечения. Система контроля версий GIT. SmartGit. Continuous Integration и Continuous Deployment.

15. Интеллектуальные указатели.

Интеллектуальные указатели. Аллокаторы. Итераторы. Категории итераторов. Особенности использования итераторов. Класс ratio. Библиотека chrono. Интервал времени. Момент времени. Эпоха. Часы. Разработка хронометра для измерения времени выполнения блока кода.

16. Последовательные контейнеры.

Последовательные контейнеры STL. array, vector, deque, list, forward list. Специальные контейнеры. Адаптеры контейнеров. stack, queue, priority queue. Битовое множество. valarray. Циклический буфер Boost. Многомерный массив Boost. Кортеж. Гетерогенные контейнеры.

17. Ассоциативные контейнеры.

Ассоциативные контейнеры STL. Множество. Отображение. Двустороннее отображение Boost. Неупорядоченные контейнеры STL. Хэш-таблица. Хэш-функция. Способы разрешения коллизий. Метод цепочек. Метод открытой адресации. Рехэширование. Boost Multi-index.

18. Алгоритмы STL.

Алгоритмы STL. Итераторы. Адаптеры итераторов. Итераторы вставки. Потоковые итераторы. Функциональные объекты, функции и лямбда-функции. Классификация алгоритмов STL. Генерация случайных чисел. Seed. Генератор. Распределение. Boost Graph Library.

19. Обработка текста.

Обработка текста. Строки. Интернационализация и локализация. Локали. Фацеты. Кодирование и наборы символов. Многобайтовые и широкие кодировки. Стандарт Unicode. Регулярные выражения. Грамматика регулярных выражений ECMAScript. Построение паттернов.

20. Ввод и вывод.

Библиотека IOStream. Иерархия классов потоков ввода-вывода. Буферизация. Форматирование. Манипуляторы. Файловые потоки ввода-вывода. Строковые потоки ввода-вывода. Библиотека filesystem. Путь. Операции с директориями. Форматы обмена данными. JSON. XML.

21. Параллельное программирование.

Параллельное программирование. Организация параллелизма. Многопоточное исполнение. Контекстное переключение. Фоновые задачи. Разработка параллельных программ. Синхронное и асинхронное исполнение. Механизм будущих результатов. Параллельные алгоритмы. Пул потоков.

22. Примитивы синхронизации.

Примитивы синхронизации. Состояние гонки. Мьютексы. Гранулярность блокировки. Взаимоблокировка. Условные переменные. Потокобезопасные структуры данных с блокировками. Стек. Очередь. Модель памяти. Атомарные типы данных. Атомарные операции.

23. Межпроцессное взаимодействие.

Межпроцессное взаимодействие. Boost Interprocess. Shared memory. Memory mapped files. Управляемая разделяемая память. Создание контейнеров в разделяемой памяти. Анонимные и именованные примитивы синхронизации. Схема consumer-producer. Использование DLL.

24. Сетевое взаимодействие.

Сетевое взаимодействие. Стек протоколов TCP/IP. Особенности протоколов TCP и UDP. Sockets API. Boost ASIO. IP адрес. Стандарты IPv4 и IPv6. Локальная сеть. Порт. Endpoint. Система DNS. Активный сокет. Пассивный сокет. Буферизация. Операции ввода-вывода.

25. Графическая библиотека SFML.

Графическая библиотека SFML (разработка игр и математическое моделирование) - дополнительная тема.

26. Повторение.

Повторение.

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

Компьютер (ноутбук) и мультимедийное оборудование (проектор, звуковая система).
Учебные аудитории, Учебный сетевой компьютерный класс с установленным необходимым программным обеспечением.

6.Перечень рекомендуемой литературы

Основная литература

1. Язык программирования C++ (стандарт C++11) : Краткий курс [Текст] = A Tour of C++, [учеб. пособие для вузов] /Бьерн Страуструп ; пер. с англ. под ред. Н. Н. Мартынова. -М., БИНОМ, 2017
2. 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

Дополнительная литература

1. Параллельное программирование на C++ в действии. Практика разработки многопоточных программ [Текст]/Э. Уильямс , -М., ДМК Пресс, 2012

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

https://github.com/
https://habr.com/ru/sandbox/112936/
https://www.syntevo.com/smartgit/
https://www.youtube.com/watch?v=4-NX17Ip-xQ&list=LLaKGYWv_KO3JhWam6ott-mw&index=7&t=0s
https://www.redhat.com/en/topics/devops/what-is-ci-cd
http://kaktusenok.blogspot.com/2013/08/boost-visual-studio.html
https://www.boost.org/
https://theboostcpplibraries.com/
https://en.cppreference.com/w/
https://www.cplusplus.com/
https://ru.stackoverflow.com/
https://ravesli.com/uroki-cpp/
https://herbsutter.com/gotw/_102/

https://codereview.stackexchange.com/questions/191211/binary-search-tree-data-structure-implementation-in-c11-using-smart-pointers
https://habr.com/ru/post/147843/
http://cppstdlib.com/code/
https://ru.stackoverflow.com/questions/454683/%d0%9a%d0%bd%d0%b8%d0%b3%d0%b8-%d0%b8-%d0%b4%d1%80%d1%83%d0%b3%d0%b8%d0%b5-%d0%bc%d0%b0%d1%82%d0%b5%d1%80%d0%b8%d0%b0%d0%bb%d1%8b-%d0%b4%d0%bb%d1%8f-%d0%be%d0%b1%d1%83%d1%87%d0%b5%d0%bd%d0%b8%d1%8f?noredirect=1&lq=1
https://ru.wikipedia.org/wiki/%D0%AE%D0%BD%D0%B8%D0%BA%D0%BE%D0%B4
https://ru.wikipedia.org/wiki/UTF-8
https://unicode-table.com/ru/
http://boost-spirit.com/home/
https://github.com/nlohmann/json
https://habr.com/ru/company/otus/blog/433588/
https://habr.com/ru/post/209824/
http://apolukhin.github.io/Boost.DLL/boost_dll/getting_started.html
https://zen.yandex.ru/media/id/5abd4b559b403c1eb5c6d8e8/chto-takoe-ip-adres-i-zachem-on-nujen-5be890c8d37bd400a93e05ed?utm_source=serp
https://www.lifewire.com/how-to-port-forward-4163829
https://ravesli.com/graficheskaya-biblioteka-sfml-vstuplenie-i-ustanovka/
https://www.sfml-dev.org/
https://habr.com/ru/post/206516/
https://habr.com/ru/company/mailru/blog/482410/
https://habr.com/ru/post/135948/
https://www.youtube.com/watch?v=x8Fo2slT2WA&feature=youtu.be
http://www.apmath.spbu.ru/ru/staff/ataeva/fractals1.html

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

На ПК в компьютерных классах должно быть установлено следующее ПО:

1. Операционная система Windows (7 или 10);
2. Microsoft Visual Studio с поддержкой C++17;

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, использующиеся 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 лабораторным занятиям 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.

ПРИЛОЖЕНИЕ

ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

по направлению: Электроника и наноэлектроника
профиль подготовки: Микро- и наноэлектроника
Физтех-школа Электроники, Фотоники и Молекулярной Физики
кафедра информатики и вычислительной математики
курс: 2
квалификация: бакалавр

Семестры, формы промежуточной аттестации:
3 (осенний) - Дифференцированный зачет
4 (весенний) - Дифференцированный зачет

Разработчик: И.С. Макаров, ассистент

1. Компетенции, формируемые в процессе изучения дисциплины

Код и наименование компетенции	Индикаторы достижения компетенции
ПК-1 Способнен планировать и проводить научные эксперименты (в избранной предметной области) и (или) теоретические (аналитические и имитационные) исследования	ПК-1.8 Владеет навыками работы с современными языками программирования и программными пакетами для научных расчетов

2. Показатели оценивания компетенций

В результате изучения дисциплины «Практика программирования с использованием C++» обучающийся должен:

знать:

- Общие понятия о структурах данных: стеки, очереди, списки, деревья, таблицы;
- теоретические подходы к созданию комплексов программ;
- идеологию объектно-ориентированного подхода;
- современные проблемы моделирования, численных методов и создания комплексов программ;
- основы архитектуры электронно-вычислительной машины (ЭВМ), представление информации в ЭВМ и архитектурные принципы повышения их производительности.
- основные принципы устройства и работы операционной системы;
- основы работы с пакетами прикладных программ в области математики и физики.

уметь:

- Применять объектно-ориентированный подход для написания программ;
- использовать современные средства создания комплексов программ;
- создавать безопасные программы, используя современные средства для отладки программ;
- выбирать оптимальные алгоритмы для современных программ;
- планировать оптимальное проведение численного эксперимента;
- абстрагироваться от несущественного при математическом моделировании;
- использовать сигналы и сообщения для взаимодействия процессов между собой и с операционной системой;
- разрабатывать полные законченные программы на одном из языков высокого уровня как индивидуально, так и в команде, с использованием современных средств написания и отладки программ;
- использовать знания по информатике для приложений в инновационной, конструкторско-технологической и производственно-технологической сферах деятельности.

владеть:

- Навыками самостоятельной работы в среде объектно-ориентированного программирования на языке C++;
- навыками освоения современных архитектур ЭВМ;
- навыками программирования с использованием средств операционной системы для решения исследовательских задач;
- объективной картиной теории и практики объектно-ориентированного программирования.

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 хэщ-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(logN). Сгенерируйте N случайных числе 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:

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 сотояние гонки и как его избежать?
22. Что такое мьютекс? Какие проблемы могут появиться при использованиии 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, коотрые 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, кода применение new может привести к утечке памяти.

40. Как запустить асинхронную задачу? Какие аргументы можно передать конструктору асинхронной задачи? В чем отличие от аргументов, передаваемых конструктору потока?

41. В чем преимущества использования механизма асинхронных задач перед явным созданием потоков посредством std::thread?

42. Что такое механизм будущих результатов? Как он реализован в C++ (какие методы, как инициализируется)?

43. Какие средства генерации случайных чисел предоставляет библиотека C++11?

44. Какие четыре элемента информации должен предоставлять класс часов?

45. Что такое стабильные часы? Когда их надо использовать? Какие есть еще часы?

46. Как замерить время работы фрагмента кода с помощью библиотеки C++11?

47. Объяснить алгоритм quicksort в стиле функционального программирования с использованием механизма асинхронных задач.

48. Какая связь между данными, объектами и ячейками памяти? Рассмотреть разные типы объектов и как они хранятся (число, строка, перечисление).

49. Что такое атомарные операции и атомарные типы данных?

50. Используют ли операции атомарных типов внутреннюю блокировку? Как проверить наличие блокировки? Какой тип гарантированно не имеет блокировок на всех системах?

51. 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?

52. На какие три категории разбиты операции работы с атоматрными типами данных?

53. Описать, как работают операции типа сравнить-и-обменять. Привести пример использования такой операции.

54. Как с помощью атомарных операций реализовать конструктор, который может быть вызван только однократно?

55. Назвать варианты упорядочения доступа к памяти.

56. Какое упорядочение памяти подразумевается по умолчанию в атомарных операциях?

57. Что такое последовательно согласованное упорядочение доступа к памяти? Чем оно хорошо, какие накладные расходы появляются?

58. Что можно сказать об относительном порядке операций в разных потоках при ослабленном упорядочении?

59. Что такое барьеры (идейно), чего они позволяют достичь, как их создавать?

60. Что дает модель упорядочения acquire-release (объяснить, результаты каких операций и где будут видны)? Аналогию с каким механизмом можно провести для использования данной модели?

61. Какая структура данных называется потокобезопасной (объяснить подробно)?

62. В чем недостаток проектирования, к примеру, потокобезопасной очереди, основанной на std::queue? Как следует изменить подход к проектированию?

63. Какие вопросы следует задавать себе при проектировании параллельных структур данных без блокировок?

64. Какие структуры данных называются неблокирующими?

65. Какие структуры данных называются свободными от блокировок?

66. Какие структуры данных называются свободными от ожидания?

67. Какие плюсы и минусы имеют структуры данных без блокировок?

68. Что такое активная блокировка, по каким причинам она возникает?

69. Что такое рекурсивное распределение данных (привести пример алгоритма, использующего эту схему)?

70. Объяснить явление перебрасывания кэша и ложного разделения на примере массива, каждый элемент которого обрабатывается отдельным исполнителем. Как избежать этой проблемы?

71. Как механизм будущих результатов помогает делать код безопаснее относительно исключений?

72. Что понимается под масштабируемостью? Сформулируйте закон Амдала.

73. Как корректно реализовать идиому pImpl? Почему плохо использовать встроенные указатели в си-стиле и как добиться соблюдения логической константности?

74. Что такое пул потоков, как работает простейший пул?

75. Как обеспечить ожидание завершения задач в пуле потоков?

Критерии оценивания

Оценка «отлично (10)» выставляется обучающемуся, если показавшему всесторонние, систематизированные, глубокие знания предмета и в ходе беседы он верно и детально ответил на четыре (4) произвольных вопроса из выше приведенного перечня. Детальный ответ предполагает верные ответы на все уточняющие вопросы. Подготовка и защита инициативной курсовой работы является преимуществом. Оценка «отлично (9)» выставляется обучающемуся, если в ходе беседы он верно, но не исчерпывающее детально ответил на четыре (4) произвольных вопроса из выше приведенного перечня (мог не ответить на некоторые уточняющие вопросы). Подготовка и защита инициативной курсовой работы является преимуществом. Оценка «отлично (8)» выставляется обучающемуся, если в ходе беседы он верно, но не исчерпывающее детально ответил на четыре (4) произвольных вопроса из выше приведенного перечня (не ответил на уточняющие вопросы). Оценка «хорошо (7)» выставляется обучающемуся, если в ходе беседы он верно и достаточно детально ответил на три (3) произвольных вопроса из выше приведенного перечня. Детальный ответ предполагает верные ответы на все уточняющие вопросы. Подготовка и защита инициативной курсовой работы является преимуществом. Оценка «хорошо (6)» выставляется обучающемуся, если в ходе беседы он верно, но не исчерпывающее детально ответил на три (3) произвольных вопроса из выше приведенного перечня (не ответил на некоторые уточняющие вопросы). Подготовка и защита инициативной курсовой работы является преимуществом. Оценка «хорошо (5)» выставляется обучающемуся, если в ходе беседы он верно и достаточно детально ответил на два (2) произвольных вопроса из выше приведенного перечня. Детальный ответ предполагает верные ответы на все уточняющие вопросы. Оценка «удовлетворительно (4)» выставляется обучающемуся, если в ходе беседы он верно и достаточно детально ответил на один (1) произвольный вопрос из выше приведенного перечня. Детальный ответ предполагает верные ответы на все уточняющие вопросы. Подготовка и защита инициативной курсовой работы является преимуществом. Оценка «удовлетворительно (3)» выставляется обучающемуся, если в ходе беседы он верно, но не исчерпывающее детально ответил на один (1) произвольный вопрос из выше приведенного перечня (не ответил на уточняющие вопросы). Оценка «неудовлетворительно (2)» выставляется обучающемуся, если в ходе беседы он не смог ответить ни на один произвольный вопрос из выше приведенного перечня, но смог ответить на наводящие вопросы и вопросы с «подсказками». Оценка «неудовлетворительно (1)» выставляется обучающемуся, если в ходе беседы он не смог ответить ни на один произвольный вопрос из выше приведенного перечня, а так же ни на один наводящий вопрос.

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.