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

УТВЕРЖДЕНО
Директор физтех-школы радиотехники и компьютерных технологий
Д.А. Гаврилов

Рабочая программа дисциплины (модуля)

по дисциплине: Введение в объектно-ориентированное программирование на языке C++
по направлению: Информатика и вычислительная техника
профиль подготовки: Компьютерные технологии и вычислительная техника
Физтех-школа Радиотехники и Компьютерных Технологий
кафедра радиоэлектроники и прикладной информатики
курс: 2
квалификация: бакалавр

Семестр, формы промежуточной аттестации: 4 (весенний) - Дифференцированный зачет

Аудиторных часов: 90 всего, в том числе:
лекции: 0 час.
семинары: 90 час.
лабораторные занятия: 0 час.

Самостоятельная работа: 45 час.

Всего часов: 135, всего зач. ед.: 3

Количество контрольных работ, заданий: 2

Программу составил: И.Р. Дединский, старший преподаватель

Программа обсуждена на заседании кафедры радиоэлектроники и прикладной информатики 30.01.2024

Аннотация

Курс представляет собой введение в современный эффективный уровень языка C++. Основная часть семестра посвящена технологии применения C++ с упором на безопасность, основанную на типах, повышение эффективности конструкций языка.

Сложность задач курса легко регулируется их функциональным наполнением.

Для обучения используются следующие принципы:

1. Во главу угла ставится задача, ее решение и, главное, путь от задачи к решению. Во всякой задаче подчеркивается разделение на идею решения и технологию реализации.
2. Самостоятельность решения является ключевым условием.
3. Понимание студентами тех средств, с помощью которых он решил задачу, ставится выше уровня самих средств решения.
4. Аккуратность и надежность решения ставятся выше «программистских трюков», иногда позволяющих в отдельных случаях добиться несколько лучших результатов.

Преподавание курса ведется в предположении, что студенты уже знают языки Си и Ассемблера.

1. Цели и задачи

Цель дисциплины

– познакомить студентов с базовыми принципами современного C++, их положительных и отрицательных сторон.

Задачи дисциплины

Задача дисциплины заключается в демонстрации базовых принципов эффективного объектно-ориентированного программирования.

2. Перечень формируемых компетенций

Освоение дисциплины направлено на формирование следующих компетенций:

Код и наименование компетенции	Индикаторы достижения компетенции
УК-1 Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач	УК-1.2 Находит, критически анализирует и выбирает информацию, необходимую для решения поставленной задачи
ОПК-2 Способен использовать современные информационные технологии и программные средства при решении задач профессиональной деятельности, соблюдая требования информационной безопасности	ОПК-2.1 Способен применять современные вычислительную технику и сервисы сети Интернет в области (сфере) профессиональной деятельности
ОПК-3 Способен составлять и оформлять научные и (или) технические (технологические, инновационные) отчеты (публикации, проекты)	ОПК-3.3 Владеет методами визуального и графического представления результатов научной (научно-технической, инновационной технологической) деятельности в виде отчетов, научных публикаций
ПК-1 Способен ставить, формализовывать и решать задачи, в том числе разрабатывать и исследовать математические модели изучаемых явлений и процессов, системно анализировать научные проблемы, получать новые научные результаты	ПК-1.1 Способен находить, анализировать и обобщать информацию об актуальных результатах исследований в рамках тематической области своей профессиональной деятельности
	ПК-1.2 Способен выдвигать гипотезы, строить математические модели для описания изучаемых явлений и процессов, оценивать качество разработанной модели
	ПК-1.3 Способен применять теоретические и (или) экспериментальные методы исследований к конкретной научной задаче и интерпретировать полученные результаты

ПК-2 Способен самостоятельно или в качестве члена (руководителя) малого коллектива организовывать и проводить научные исследования и их апробацию	ПК-2.1 Знает принципы построения научной работы, методы сбора и анализа полученного материала, способы аргументации
	ПК-2.2 Способен планировать и проводить научные исследования самостоятельно или в качестве члена (руководителя) малого научного коллектива
	ПК-2.3 Способен проводить апробацию результатов научно-исследовательской работы посредством публикации научных статей и участия в конференциях

3. Перечень планируемых результатов обучения по дисциплине (модулю)

В результате освоения дисциплины обучающиеся должны

знать:

- основы реализации больших проектов на языке C++;
- основы объектно-ориентированного программирования;
- различные пути повышения производительности программы.

уметь:

- создавать программы на языке C++;
- понимать и разрабатывать программные интерфейсы.

владеть:

- навыками ведения простейших программных проектов в системах контроля версий.

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	Вывод типов		10		5
2	Аллокаторы		10		5
3	Move-семантика и rvalue-ссылки		10		5
4	Контейнеры		10		5
5	Итераторы		10		5
6	Умные указатели		10		5
7	Функциональные объекты и лямбда-функции		10		5
8	Шаблонное метапрограммирование и SFINAE		10		5
9	Некоторые особые полезные типы		10		5
Итого часов			90		45
Подготовка к экзамену		0 час.			
Общая трудоёмкость		135 час., 3 зач.ед.			

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

Семестр: 4 (Весенний)

1. Вывод типов

Проблема с длинными названиями типов. Проблема с возможными ошибками в написании точных названий типов. Ключевое слово auto как решение этих проблем. Правила вывода типов для auto. Особый случай с типом initializer_list. Особенности при auto&&. auto в качестве возвращаемого типа функции.

Ключевое слово decltype, правила вывода типов для него. Особенности поведения decltype от выражений (случаи lvalue, xvalue, prvalue). Пример с decltype((x)). Особенности взятия decltype от тернарного оператора. Конструкция decltype(auto). Пример: обертка над обращением к контейнеру по индексу. Трюк для вывода названий выведенных типов на экран (намеренное провоцирование ошибок компиляции).

2. Аллокаторы

Placement new, его синтаксис, действие и отличие от обычного new. Разница между оператором new и функцией operator new. Более подробный разбор действия оператора new. Перегрузка new для отдельных классов. Перегрузка глобального new. Определение new с произвольными параметрами. То же самое для операторов delete и delete[]. Пример, когда компилятор неявно вызывает delete с нестандартными параметрами. Поведение delete для полиморфных объектов. nothrow оператор new, его синтаксис и особенности. Разбор поведения new в случае нехватки памяти. Функция new_handler, функции set_new_handler и get_new_handler.

Понятие аллокатора. Класс std::allocator, его основные методы (allocate, deallocate, construct, destroy) и их примерная реализация. Особенности реализации конструкторов и оператора присваивания у стандартного аллокатора. Класс std::allocator_traits, его предназначение и основные методы. Пример нестандартного аллокатора (PoolAllocator), идея реализации его методов. Проблемы с конструктором копирования и оператором присваивания.

3. Move-семантика и rvalue-ссылки

Проблемы, приводящие к идее move-семантики: неэффективный swap, неэффективный push_back, emplace_back, construct. Применение магической функции std::move. Решение проблемы со swap. Понятие move-конструктора и move-assignment оператора, их реализация, генерация компилятором, “правило пяти”. Реализация функции std::move. Дилемма: что принять в качестве параметра?

Понятие rvalue-ссылок. Особенности инициализации rvalue-ссылок, разрешенные и запрещенные присваивания между ссылками (включая проблемы с константностью). Решение проблемы с push_back. Понятия glvalue, lvalue, rvalue, prvalue и xvalue. Связи между ними. Примеры выражений, являющихся тем или иным видом value.

Понятие универсальных ссылок, отличие их от rvalue-ссылок. Правила вывода типа шаблонов в случае универсальных ссылок, решение проблемы с типом параметра функции move. Правила сворачивания ссылок (reference collapsing).

Проблема прямой передачи (perfect forwarding). Функция std::forward и ее применение. Решение проблем с emplace_back и construct. Реализация std::forward, ее обсуждение. Почему типы у принимаемого параметра и возвращаемого значения именно такие?

Новая проблема с push_back: безопасность относительно исключений. Функция std::move_if_noexcept, решение проблемы с ее помощью.

Return Value Optimization, условия ее возникновения. Примеры, когда RVO точно произойдет и когда может не произойти. Примеры, когда имеет и когда не имеет смысл писать return std::move(x) вместо return x. Copy Elision, примеры.

Ссылочные квалификаторы. Решение проблемы с запретом оператора присваивания для rvalue у кастомных типов. Примеры типов, для которых прямая передача работает некорректно. Особенности поведения универсальных ссылок при разрешении перегрузки. Феномен “поглощения” универсальными ссылками обычных ссылок.

4. Контейнеры

Общие слова о контейнерах. Класс std::vector, его предназначение, идея реализации, основные методы и их алгоритмическая сложность. Поля класса std::vector. Реализация конструкторов, деструкторов, оператора присваивания с правильным обращением к аллокатору. Реализация метода push_back с правильным обращением к аллокатору. Реализация оператора [] для константных и неконстантных vector. Разница между [] и методом at(). Метод emplace_back, его реализация и отличие от push_back. Методы size(), resize(), capacity(), reserve() и shrink_to_fit(). Особенности работы с аллокатором при копировании вектора. Метод select_on_container_copy_construction. Вопросы на понимание: чему равно sizeof(v), где v - вектор, и что произойдет при вызове delete[] &(v[0])?

Класс vector<bool> и его отличие от обычного vector, преимущества и недостатки. Внутренний класс BoolProxy. Особенности реализации оператора [] и оператора присваивания для vector<bool> по сравнению с обычным vector.

5. Итераторы

Общая идея итераторов. Использование итераторов у стандартных контейнеров. Виды итераторов: input, output, forward, bidirectional, random access. Операции, поддерживаемые каждым видом итераторов. Виды итераторов у стандартных контейнеров. Константные и reverse-итераторы. Методы cbegin, cend, rbegin, rend, crbegin, crend у контейнеров. Реализация класса std::reverse_iterator, метод base.

Класс std::iterator, его предназначение. Класс std::iterator_traits, его предназначение. Пример ситуации, когда он необходим (обращение к value_type). Функции std::distance и std::advance. Различие в поведении этих функций для разных видов итераторов, реализация этого различия.

Стандартная библиотека алгоритмов, использование стандартных алгоритмов над контейнерами с итераторами. Итераторы для вставок: классы std::insert_iterator, std::back_insert_iterator, их предназначение, реализация. Функции std::inserter, std::back_inserter, их реализация. Правила инвалидации итераторов в стандартных контейнерах. Безопасные и небезопасные операции в контейнерах с точки зрения инвалидации итераторов.

Класс std::deque, основные методы и их алгоритмическая сложность. Разница между deque и vector: методы deque, отсутствующие у vector; методы vector, отсутствующие у deque. Адаптеры над контейнерами: std::stack, std::queue и std::priority_queue, их реализации. Компараторы в priority_queue и ее специфичные методы.

Класс std::list, основные методы и их алгоритмическая сложность. Идея реализации list'а. Вставка и удаление из произвольного места. Специфичные для list'а методы: splice, sort, merge, reverse. Особенности работы list'а с аллокатором, метод rebind у аллокаторов. Класс std::forward_list, его отличия от обычного list.

Ассоциативные контейнеры. Класс std::map, его предназначение, идея реализации. Описание шаблонных параметров класса map. Класс std::pair и функция std::make_pair. Основные методы map'а и их алгоритмическая сложность. Способы поиска в map'е. Способы вставки в map, особенности работы оператора []. Способы удаления из map'а. Классы std::set, std::multimap и std::multiset, их предназначение, отличия от std::map.

Класс std::unordered_map, сходства и различия с обычным std::map. Основные методы и их алгоритмическая сложность. Особые для unordered_map шаблонные параметры: Hasher, Equal. Класс std::hash и его специализации. Особые для unordered_map методы: bucket_count, load_factor, rehash. Классы std::unordered_set, std::unordered_multimap, их идея, отличие от unordered_map.

6. Умные указатели

Идея и мотивировка умных указателей.

Класс std::auto_ptr как первая неудачная попытка реализовать идею.

Класс std::unique_ptr, его концепция. Особенности его конструкторов, деструктора и операторов присваивания. Методы * и ->. Специализация unique_ptr для массивов.

Класс std::shared_ptr, его концепция. Идея реализации счетчика ссылок. Реализация методов.

Потенциальная проблема, связанная с прямым вызовом new. Функции std::make_unique и std::make_shared как способ избежать прямого вызова new. Реализация этих функций. Функция std::allocate_shared, ее предназначение и реализация. Исправление реализации конструктора shared_ptr для этого.

Проблема закольцованности указателей. Класс std::weak_ptr как решение этой проблемы. Реализация методов этого класса. Проблема с реализацией метода expired(), модификация класса shared_ptr для правильной работы с этим.

Кастомные deleter'ы для умных указателей, схема использования. Более правильная реализация деструкторов unique_ptr и shared_ptr.

Класс std::enable_shared_from_this, его предназначение и реализация. Еще одна модификация конструктора shared_ptr (проверка на наследника enable_shared_from_this).

7. Функциональные объекты и лямбда-функции

Лямбда-функции: мотивировка, простой пример (нестандартный компаратор в std::sort). Списки захвата в лямбда-функциях. Захват по ссылке и по значению. Особенности захвата this. Захват с присваиванием и перемещающий захват в C++14. Слово mutable применительно к лямбда-функциям. Явное указание возвращаемого значения. Захват по умолчанию и проблемы, которые он потенциально порождает. Пример с классом и методом getFunction() в нем.

Обобщенные лямбда-функции в C++14. Применение auto и decltype в лямбда-функциях. Класс std::function, его предназначение и схема использования. Реализация std::function. Функция std::bind, ее предназначение и схема использования (без реализации). Placeholder'ы. Класс std::is_invocable. Класс std::invoke_result и функция std::invoke. (Все без реализации.)

8. Шаблонное метапрограммирование и SFINAE

Имитация if через шаблоны. Имитация for через шаблоны. Примеры: вычисление чисел Фибоначчи, проверка простоты числа, вывод чисел от 1 до 1000 с помощью шаблонов. Ключевое слово constexpr для функций и для переменных. Отличие constexpr от const. Требования к constexpr-функциям.

type_traits. Структуры std::is_const (pointer, reference etc.), std::add_const (pointer, reference etc.), std::remove_const (pointer, reference, extent etc.), их реализации. Структуры std::is_same, std::true_type, std::false_type, std::conjunction, std::disjunction, std::conditional, std::rank, их реализации.

Идиома SFINAE. Общая идея. Простейший пример: структура std::enable_if, ее реализация и применение.

Структура std::is_class, ее реализация (без реализации std::is_union). Реализация метода std::allocator_traits::construct (проверка, определен ли у аллокатора метод construct). Проблема: невозможность написать T() для произвольного типа T. Функция std::declval, ее особенности. Решение предыдущей проблемы с ее помощью. Структуры std::is_constructible, std::is_convertible, std::is_copy_constructible, std::is_move_constructible etc. Их реализации. Реализация std::is_nothrow_move_constructible. Реализация std::move_if_noexcept через std::is_nothrow_move_constructible. Почему принимаемый и возвращаемый типы именно такие?

Понятие неполных типов (incomplete types). Новая проблема с declval (что возвращать), решение проблемы с помощью rvalue-ссылки. Реализация std::is_base_of. Пример применения: проверка is_base_of<enable_shared_from_this<T>> в конструкторе shared_ptr<T>. Реализация std::common_type.

9. Некоторые особые полезные типы

Класс std::variant, его предназначение и основные методы. Примерное описание реализации этого класса. Класс std::any, его предназначение и основные методы. Примерная реализация этого класса. Класс std::optional, его предназначение и основные методы. Неудачная попытка создать vector<int&>. Класс std::reference_wrapper для решения этой и других проблем. Примерная реализация этого класса.

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

Учебная аудитория, оснащенная компьютером и мультимедийным оборудованием (проектор, звуковая система).

6.Перечень рекомендуемой литературы

Основная литература

1. Язык программирования C++ [Текст] / Б. Страуструп ; пер. с англ. С. Анисимова, М. Кононова ; под ред. Ф. Андреева, А. Ушакова .— Спец. изд. с авт. изменениями и доп. — М. : Бином Пресс, 2008 .— 1104 с.

фонд литературы кафедры:

2. Б. Страуструп. Дизайн и эволюция языка C++.
3. Б. Эккель. Философия C++.
4. Э. Гамма, Р. Хелм, Р. Джонсон, Д. Влиссидес. Паттерны объектно-ориентированного программирования.

Дополнительная литература

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

1. http://refactoring.guru

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

Операционная система Linux. Среда программирования.

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

Успешное освоение курса требует напряжённой самостоятельной работы студента. В программе курса приведено минимально необходимое время для работы студента над темой. Самостоятельная работа включает в себя проработку учебного материала (по учебной литературе), подготовку к практическим занятиям.

Промежуточный контроль знаний проводится в виде сдач проектных заданий.

ПРИЛОЖЕНИЕ

ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

по направлению: Информатика и вычислительная техника
профиль подготовки: Компьютерные технологии и вычислительная техника
Физтех-школа Радиотехники и Компьютерных Технологий
кафедра радиоэлектроники и прикладной информатики
курс: 2
квалификация: бакалавр

Семестр, формы промежуточной аттестации: 4 (весенний) - Дифференцированный зачет

Разработчик: И.Р. Дединский, старший преподаватель

1. Компетенции, формируемые в процессе изучения дисциплины

Код и наименование компетенции	Индикаторы достижения компетенции
УК-1 Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач	УК-1.2 Находит, критически анализирует и выбирает информацию, необходимую для решения поставленной задачи
ОПК-2 Способен использовать современные информационные технологии и программные средства при решении задач профессиональной деятельности, соблюдая требования информационной безопасности	ОПК-2.1 Способен применять современные вычислительную технику и сервисы сети Интернет в области (сфере) профессиональной деятельности
ОПК-3 Способен составлять и оформлять научные и (или) технические (технологические, инновационные) отчеты (публикации, проекты)	ОПК-3.3 Владеет методами визуального и графического представления результатов научной (научно-технической, инновационной технологической) деятельности в виде отчетов, научных публикаций
ПК-1 Способен ставить, формализовывать и решать задачи, в том числе разрабатывать и исследовать математические модели изучаемых явлений и процессов, системно анализировать научные проблемы, получать новые научные результаты	ПК-1.1 Способен находить, анализировать и обобщать информацию об актуальных результатах исследований в рамках тематической области своей профессиональной деятельности
	ПК-1.2 Способен выдвигать гипотезы, строить математические модели для описания изучаемых явлений и процессов, оценивать качество разработанной модели
	ПК-1.3 Способен применять теоретические и (или) экспериментальные методы исследований к конкретной научной задаче и интерпретировать полученные результаты
ПК-2 Способен самостоятельно или в качестве члена (руководителя) малого коллектива организовывать и проводить научные исследования и их апробацию	ПК-2.1 Знает принципы построения научной работы, методы сбора и анализа полученного материала, способы аргументации
	ПК-2.2 Способен планировать и проводить научные исследования самостоятельно или в качестве члена (руководителя) малого научного коллектива
	ПК-2.3 Способен проводить апробацию результатов научно-исследовательской работы посредством публикации научных статей и участия в конференциях

2. Показатели оценивания компетенций

В результате изучения дисциплины «Введение в объектно-ориентированное программирование на языке C++» обучающийся должен:

знать:

- основы реализации больших проектов на языке C++;
- основы объектно-ориентированного программирования;
- различные пути повышения производительности программы.

уметь:

- создавать программы на языке C++;
- понимать и разрабатывать программные интерфейсы.

владеть:

- навыками ведения простейших программных проектов в системах контроля версий.

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

Примеры заданий:

1. Реализовать эффективные классы Vector и String, а также сопутствующие классы.
2. Реализовать эффективный класс Function.
3. Реализовать эффективный класс SharedPtr.
4. Реализуйте шаблонную функцию detect_fields_count<S>, которая бы позволяла определять количество полей у данной структуры S.

За каждое задание выставляется оценка в соответствии со следующими критериями оценивания:

Отлично

10) Полностью и вовремя решены все задачи без ошибок. Продемонстрирован грамотный подход к решению задач, реализованы оптимальные алгоритмы, код оформлен в едином удобочитаемом стиле.

9) Полностью и вовремя решены все задачи без ошибок. Продемонстрирован грамотный подход к решению задач, реализованы оптимальные алгоритмы.

8) Полностью и вовремя решены все задачи без ошибок. Продемонстрирован грамотный подход к решению задач.

Хорошо

7) Полностью решены все задачи. Допущены несущественные ошибки

6) Полностью решено большинство задач. В некоторых задачах допущены и не исправлены ошибки, либо некоторые задачи решены частично.

5) Полностью решено две трети задач. В некоторых задачах допущены и не исправлены ошибки, либо некоторые задачи решены частично.

Удовлетворительно.

4) Полностью решено более половины задач. В остальных задачах допущены и не исправлены ошибки, либо некоторые задачи решены частично.

3) Полностью решено более половины задач.

Неудовлетворительно

2) Решено менее половины задач.

1) Не решено ни одной задачи.

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

1. Расскажите о шаблонах в C++. Что такое инстанцирование шаблонов, специализация шаблонов? Какие 3 вида шаблонов существуют? Какие 3 вида шаблонных параметров существуют? Как использовать шаблоны с переменным количеством аргументов и оператор sizeof…? Что такое Curiously Recurring Template Pattern? Что такое полиморфизм и как это понятие в C++ связано с шаблонами?
2. Расскажите о контейнере std::vector. Предложите реализацию конструкторов, деструктора, операторов присваивания, оператора [], методов at, resize, reserve, front, back для этого класса (с правильной поддержкой аллокаторов и move-семантики).
3. Расскажите о контейнере std::vector. Предложите реализацию методов push_back и pop_back (с правильной поддержкой аллокаторов, move-семантики и гарантий безопасности относительно исключений).
4. Расскажите про vector<bool>. В чем его особенность и отличие от обычного vector? Расскажите, каким образом достигается эта особенность с точки зрения реализации (опишите реализацию основных методов).
5. Расскажите о контейнере std::list. Каковы его методы и скорость их работы? Как он устроен изнутри? Каким образом list<T> использует аллокатор для типа T? Предложите реализацию основных методов list'a: конструкторы, операторы присваивания, деструктор, insert элемента по итератору и erase по итератору.

6. Расскажите о контейнере std::forward_list. Каковы его методы и скорость их работы? Как он устроен изнутри? Каким образом forward_list<T> использует аллокатор для типа T? Предложите реализацию основных методов forward_list'a: конструкторы, операторы присваивания, деструктор, insert элемента по итератору и erase по итератору.

7. Расскажите о контейнере std::deque. Чем он отличается от vector? Расскажите об адаптерах над контейнерами (std::stack, std::queue, std::priority_queue). Предложите реализацию какого-нибудь одного из них.

8. Расскажите о контейнерах std::map, std::set, std::multimap и std::multiset. Для чего они применяются? Какие у них шаблонные параметры и что они означают? Каковы их основные методы, скорость и принцип работы этих методов? С помощью какой структуры данных реализованы эти контейнеры?

9. Расскажите о контейнерах std::unordered_map, std::unordered_set, std::unordered_multimap и std::unordered_multiset. Для чего они применяются? Какие у них шаблонные параметры и что они означают? Каковы их основные методы, скорость и принцип работы этих методов? С помощью какой структуры данных реализованы эти контейнеры?

10. Что такое итераторы? Какие виды итераторов бывают и какие операции допустимы над каждым из них? Расскажите про функции std::advance и std::distance. Каким образом достигается их разное поведение в зависимости от вида переданного итератора?

11. Какие виды итераторов поддерживает каждый из стандартных контейнеров? Предложите реализацию итераторов для какого-нибудь из стандартных контейнеров (на ваш выбор), считая, что в остальном контейнер уже реализован.

12. Расскажите про класс std::insert_iterator и функцию std::inserter, предложите их реализацию.

13. Расскажите об операторах new и delete. Зачем они нужны, как ими пользоваться? Что такое placement new? В чем разница между оператором new и функцией operator new? Что из вышеперечисленного можно перегружать и как это делать? Зачем может быть нужно перегружать кастомный operator delete? Расскажите о проблеме утечек памяти.

14. Что такое аллокаторы, зачем они нужны? Расскажите про класс std::allocator, предложите его реализацию.

15. Расскажите о проблемах, для решения которых была введена move-семантика. Расскажите, как работает функция std::move и как ее применять, объясните, как она реализована и почему именно так. Как правильно реализовать функцию swap в C++11?

16. Расскажите о том, что такое "идеальная передача" (perfect forwarding). Расскажите, как работает функция std::forward и как она реализована. Приведите пример применения этой функции. Почему в std::forward нельзя принять T&& в качестве параметра?

17. Расскажите об rvalue-ссылках и об универсальных ссылках. Какие присваивания между lvalue-, rvalue-ссылками, а также временными объектами и именованными нессылочными объектами разрешены, а какие запрещены? Что меняется в случае с константными ссылками и объектами? Что является и что не является универсальными ссылками? Приведите примеры, иллюстрирующие все вышесказанное.

18. Что такое lvalue, rvalue, xvalue, glvalue и prvalue? Приведите примеры. Расскажите о том, что такое reference collapsing (сворачивание ссылок), reference qualifiers (ссылочные квалификаторы), return value optimization (RVO) и copy elision. Приведите примеры, иллюстрирующие все вышеперечисленное.

19. Расскажите, для решения какой проблемы нужны умные указатели. Расскажите про класс std::unique_ptr, предложите реализацию основных методов этого класса (конструкторы, деструктор, операторы присваивания, операторы унарная звездочка и стрелочка).

20. Расскажите про класс std::shared_ptr. Какую проблему он решает и как он устроен? Предложите реализацию его основных методов (конструкторы, деструктор, операторы присваивания, операторы унарная звездочка и стрелочка).

21. Предложите реализацию функции std::make_unique. Зачем нужна эта функция, какую потенциальную проблему она решает? Расскажите о функциях make_shared и allocate_shared (без реализации). Зачем нужна каждая из них, какие потенциальные проблемы они решают?

22. Расскажите про класс std::weak_ptr. Для решения какой проблемы он нужен? Предложите реализацию его основных методов. Как weak_ptr узнает, что объект под ним уже был удален?

23. Расскажите про класс std::enable_shared_from_this. Для решения какой проблемы он нужен и как им пользоваться? Предложите реализацию этого класса. Как нужно изменить код shared_ptr, чтобы он работал согласованно с enable_shared_from_this?

24. Расскажите о выводе типов с помощью ключевых слов auto и decltype. В чем разница между правилами вывода типов для них? В каких контекстах можно использовать auto? Как работает decltype от идентификаторов и от разных категорий выражений (lvalue, xvalue, prvalue)? Что такое decltype(auto) и для решения какой проблемы это нужно?

25. Расскажите о возможностях compile-time вычислений в C++. Как с помощью шаблонного метапрограммирования имитировать циклы и условия? Покажите на примере проверки целого числа на простоту. Расскажите о ключевом слове constexpr: чем constexpr отличается от const, зачем нужны constexpr-функции и переменные, какими возможностями они обладают и какие ограничения на них накладываются?

26. Что такое type_traits? Объясните, как реализованы структуры std::remove_reference, std::is_same, std::rank, std::conjunction, std::conditional, std::common_type.

27. Что такое SFINAE? Объясните общую идею на примере структуры std::enable_if. Предложите реализацию структуры std::is_class, объясните, как она работает (проверкой на is_union можно пренебречь).

28. Зачем нужна функция std::declval? Почему эта функция возвращает T&&, а не T? Предложите реализацию структуры std::is_constructible, объясните, как она работает.

29. Зачем нужен класс std::allocator_traits? Предложите реализацию функции allocator_traits<Allocator>::construct. Каким образом эта функция проверяет, реализован ли у аллокатора метод construct, и как она себя ведет в зависимости от этого?

30. Расскажите о функции std::move_if_noexcept. В каком месте стандартной библиотеки она используется и почему она там необходима? Предложите ее реализацию (считая, что type_traits уже реализованы).

31. Предложите реализацию структуры std::is_base_of. Объясните, как она работает.

32. Что такое лямбда-функции в C++? Каков синтаксис лямбда-выражений? Что такое списки захвата, что такое захват по умолчанию и в чем его опасность? Расскажите о дополнительных возможностях лямбда-выражений в C++14 (захват с инициализацией, обобщенные лямбда-выражения). Расскажите (без реализации) про класс std::function и функцию std::bind, приведите примеры их использования.

33. Что такое юнионы (union), для чего они нужны? В чем сходство и различие между юнионами и классами? Каковы особенности инициализации полей юниона, что такое “активный член” юниона и как его поменять? Что такое std::variant и std::any, для чего они нужны, каковы их основные методы?

34. Что такое рефлексия? Что можно сказать о возможностях рефлексии в C++? Реализуйте шаблонную функцию detect_fields_count<S>, которая бы позволяла определять количество полей у данной структуры S. Объясните, как это работает.

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 заданий, предусмотренных программой дисциплины.

Дифференцированный зачет проводится в устной форме.

При проведении дифференцированного зачета обучающемуся предоставляется 20 минут на подготовку. Опрос обучающегося проводится в течение 30 минут.