

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

УТВЕРЖДЕНО

**Директор физтех-школы
прикладной математики и
информатики**

А.М. Райгородский

	Рабочая программа дисциплины (модуля)
по дисциплине:	Программирование на языке C++. Продвинутый поток
по направлению:	Прикладная математика и информатика
профиль подготовки:	А1360: Передовые методы искусственного интеллекта Физтех-школа Прикладной Математики и Информатики кафедра алгоритмов и технологий программирования
курс:	1
квалификация:	бакалавр

Семестр, формы промежуточной аттестации: 1 (осенний) - Дифференцированный зачет

Аудиторных часов: 30 всего, в том числе:

лекции: 15 час.

семинары: 15 час.

лабораторные занятия: 0 час.

Самостоятельная работа: 60 час.

Всего часов: 90, всего зач. ед.: 2

Количество контрольных работ, заданий: 2

Программу составил: И.С. Мещерин, ассистент

Программа обсуждена на заседании кафедры алгоритмов и технологий программирования 05.08.2025

Аннотация

Курс содержит в себе принципы объектно-ориентированного подхода, а так же объектно-ориентированный анализ и проектирование. Раскрывает и описывает основные подходы в объектно-ориентированном программировании. Рассматривает обобщенные классы и методы. Строго типизированные источники и управление взаимодействием объектов.

Курс направлен на формирование у студентов системного понимания принципов объектно-ориентированного программирования и освоение современных возможностей языка C++. В рамках дисциплины изучаются базовые конструкции и парадигмы программирования, обобщённые классы и методы, механизмы строгой типизации и управление взаимодействием объектов. Отдельное внимание уделяется проектированию и реализации эффективных многопоточных приложений, использованию C++ для работы с ограниченными аппаратными ресурсами, а также применению языка для массовой параллелизации вычислений с использованием GPU и FPGA в задачах искусственного интеллекта.

Образовательный процесс организован в практико-ориентированном формате: лекционные занятия сочетаются с интерактивными семинарами, мастер-классами с привлечением специалистов промышленных партнёров и выполнением командных проектных заданий. В рамках дисциплины предусмотрено решение кейсов из реальной практики компаний, участие в конкурсах и хакатонах, а также выполнение индивидуальных и групповых проектов, что позволяет студентам закрепить освоенные знания и развить навыки их применения в профессиональной деятельности.

Курс базируется на знаниях, полученных при изучении дисциплины «Введение в математический анализ», и является предшествующим для дисциплины «Алгоритмы и структуры данных».

1. Цели и задачи

Цель дисциплины

Формирование у студентов фундаментальных знаний и практических навыков разработки программного обеспечения на языке C++ с опорой на принципы объектно-ориентированного и обобщённого программирования, а также освоение современных технологий эффективной реализации алгоритмов. Особое внимание уделяется подготовке студентов к использованию C++ в области искусственного интеллекта: созданию многопоточных решений, разработке программ под аппаратные платформы с ограниченными ресурсами, а также применению средств массовой параллелизации вычислений.

Задачи дисциплины

- научить формулировать задачи в терминах изученных теорий, выбирать подходящий алгоритм для поставленной задачи;
- научить разрабатывать комбинации алгоритмов для решения поставленных задач, оценивать сложности алгоритмов, их модификаций и комбинаций, в том числе с помощью амортизационного анализа, выбирать подходящие структуры данных для поставленных задач, реализовывать алгоритмы в обобщенной форме на языке программирования C++.

2. Перечень формируемых компетенций

Освоение дисциплины направлено на формирование следующих компетенций:

Код и наименование компетенции	Индикаторы достижения компетенции
ОПК-1 Способен применять фундаментальные знания, полученные в области физико-математических и (или) естественных наук и использовать их в профессиональной деятельности	ОПК-1.2 Способен строить математические модели, производить количественные расчеты и оценки
ОПК-2 Способен использовать современные информационные технологии и программные средства при решении задач профессиональной деятельности, соблюдая требования информационной безопасности	ОПК-2.2 Знает и умеет применять численные математические методы и прикладное программное обеспечение для решения научных задач в профессиональной области

PL-3 Способен применять языки программирования C/C++ для решения задач в области ИИ	PL-3.1 Разрабатывает и отлаживает эффективные многопоточные решения на C++, тестирует, испытывает и оценивает качество таких решений
	PL-3.2 Разрабатывает и отлаживает системы ИИ на C++ под конкретные аппаратные платформы с ограничениями по вычислительной мощности, в том числе для встроенных систем
	PL-3.3 Разрабатывает и отлаживает решения на C++, использующие GPU и FPGA для массовой параллелизации вычислений в рамках общей системы ИИ, с применением как готовых решений, так и разработкой своих

3. Перечень планируемых результатов обучения по дисциплине (модулю)

В результате освоения дисциплины обучающиеся должны

знать:

- алгоритмы на графах и структуры данных, связанные с ними;
- оценки сложности стандартных алгоритмов;
- стандартные алгоритмы на графах и используемые структуры данных, подходы к модификации классических алгоритмов;
- разнообразные классические задачи в теории графов и асимптотические сложности их решений.

уметь:

- формулировать задачи в терминах изученных теорий, выбирать подходящий алгоритм для поставленной задачи;
- разрабатывать комбинации алгоритмов для решения поставленной задачи;
- оценивать сложности алгоритмов, их модификаций и комбинаций, в том числе с помощью амортизационного анализа;
- выбирать подходящие структуры данных для конкретной задачи;
- реализовывать алгоритм в обобщенной форме на языке программирования c++;
- реализовывать стандартные алгоритмы на графах и структуры данных на языке программирования C++.

владеть:

- методами декомпозиции задач в области информационных технологий и построения единого решения с использованием изученных алгоритмов;
- методами оценки сложности алгоритмов, их модификаций и комбинаций.

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	Введение в язык	1	1		7
2	Модификаторы типов	2	2		8
3	Введение в ООП	2	2		7
4	Перегрузка операторов	2	2		8
5	Наследование (inheritance)	2	2		7
6	Шаблоны (templates)	2	2		8
7	Исключения (exceptions)	2	2		7
8	Аллокатеры (allocators)	2	2		8
Итого часов		15	15		60

Подготовка к экзамену	0 час.
Общая трудоёмкость	90 час., 2 зач.ед.

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

Семестр: 1 (Осенний)

1. Введение в язык

Форматы обучения: Лекции, интерактивные семинары, разбор кейсов, проектные практикумы, мастер-классы, групповые обсуждения.

Конкурсы индивидуальных проектов: «Реализация калькулятора с поддержкой арифметических и логических операторов», «Симулятор работы операторов инкремента/декремента», «Программа для анализа различий между signed и unsigned типами», «Наглядный разбор ошибок: компиляция vs runtime».

Групповые проекты: «Создание справочника по базовым конструкциям C++ с примерами кода», «Разработка обучающего консольного приложения для новичков (пошаговое объяснение операций и операторов)».

Мастер-классы от индустриального партнёра: «Эволюция языка C++: от C++98 до современных стандартов» (Сбер/Яндекс), «Типичные ошибки новичков и как их избегают в индустрии».

Практические кейсы от индустриального партнёра: «Реализация модуля парсинга логов с обработкой ошибок» (Авито), «Простейшие сервисные утилиты для внутренних нужд команды разработки» (Сбер).

Хакатоны: «Code Starter Hack: за 24 часа написать набор утилит на чистом C++», «Debug Battle: соревнование по поиску и исправлению ошибок в чужом коде».

Ключевые темы:

Общие слова: место языка C++ среди современных языков, актуальные версии этого языка, ключевые люди, связанные с этим языком, официальный стандарт языка

Структура программы, функция main, понятие области видимости (scope). Ключевые слова. Ввод-вывод (cin, cout).

Объявления (declarations). Идентификаторы. Фундаментальные типы (int, long, long long, float, double, long double, char, bool, модификаторы signed и unsigned). Размеры этих типов, основные операции над ними, неявные преобразования типов между собой. Литералы, литеральные суффиксы для основных типов. Объявления функций, разница между объявлением и определением, one definition rule.

Выражения (expressions). Операторы. Арифметические операторы. Побитовые операторы. Логические операторы, особенности их работы. Оператор присваивания и операторы составного присваивания, особенности его работы. Понятие lvalue и rvalue в C++03. Инкремент и декремент, отличие префиксной версии от постфиксной. Операторы сравнения. Тернарный оператор. Оператор “запятая”. Оператор sizeof.

Инструкции (statements). Конструкции if...else, for, while, do...while, switch, их синтаксис, правила работы. Инструкции break, continue, return, их действие. Инструкция goto и метки.

Понятия ошибки компиляции, ошибки времени выполнения (runtime error), неопределенного поведения (undefined behaviour), отличия между ними, примеры. Виды ошибок компиляции: лексические, синтаксические, семантические. Понятие segmentation fault и stack overflow.

2. Модификаторы типов

Форматы обучения: Лекции, интерактивные семинары, разбор кейсов, проектные практикумы, мастер-классы, групповые обсуждения.

Конкурс для командной работы: «Охота на баги»

Конкурсы индивидуальных проектов: «Реализация собственной версии функции swap с использованием указателей и ссылок», «Демонстрация ошибок: утечки памяти и двойное удаление», «Сравнение разных видов приведения типов в C++ на практических примерах», «Интерактивный справочник: const, указатели и ссылки».

Групповые проекты: «Создание отладчика утечек памяти для учебных проектов», «Разработка мини-библиотеки безопасной работы с указателями и ссылками», «Симулятор для визуализации работы new/delete и стековой памяти».

Мастер-классы от индустриального партнёра: «Как управлять памятью в высоконагруженных приложениях» (Яндекс), «Реальные баги, связанные с указателями и ссылками, и как их решают в индустрии» (Сбер), «Опыт применения const correctness в больших проектах» (Авито).

Практические кейсы от индустриального партнёра: «Оптимизация работы с памятью в серверных приложениях» (Яндекс), «Реализация модуля для обработки пользовательских данных с безопасным приведением типов» (Сбер).

Хакатоны: «Memory Battle: найти и исправить утечки памяти в большом проекте», «Type Casting Challenge: преобразование данных с сохранением корректности».

Ключевые темы:

Указатели, операции над ними. Операция взятия адреса. Автоматическая память (стек). Массивы, операция [] (квадратные скобки), ее принцип работы. Указатель на void и его особенности.

Функции. Перегрузка функций, правила разрешения перегрузки (общая схема, без деталей). Функции с аргументами по умолчанию. Функции с неуказанным количеством аргументов. Указатели на функции, операции над ними, их особенности.

Динамическая память. Операторы new и new[], их использование (в стандартной форме). Операторы delete и delete[], их использование (в стандартной форме). Проблема утечек памяти. Проблема двойного удаления.

Передача аргументов по значению и по указателю, первая версия функции swap. Дилемма с присваиванием (создавать новое название или копию?). Идея ссылок (references). Отличия ссылок от указателей, правила работы со ссылками, вторая версия функции swap. Проблема, связанная со ссылкой на локальную переменную (“битые ссылки”).

Идея констант, ключевое слово const. Понятие константных и неконстантных операций, особенности работы с константами. Константные и неконстантные ссылки. Константные указатели и указатели на константу. Разрешенные и запрещенные присваивания между всеми вышеупомянутыми типами.

Виды приведений типов: static_cast, reinterpret_cast, const_cast и C-style cast, их особенности, примеры применения и примеры, когда они не сработают.

3. Введение в ООП

Форматы обучения: Лекции, интерактивные семинары, разбор кейсов, проектные практикумы, мастер-классы, групповые обсуждения.

Конкурсы индивидуальных проектов: «Реализация класса String с перегрузкой конструкторов и оператором присваивания», «Демонстрация “правила трёх” на примере класса Vector», «Создание класса с использованием friend-функций», «Пример работы делегирующих конструкторов и списков инициализации».

Групповые проекты: «Разработка мини-библиотеки классов для моделирования геометрических объектов (Point, Line, Circle)», «Проектирование системы классов для управления библиотекой книг», «Создание симулятора объектов с использованием статических методов и полей».

Мастер-классы от индустриального партнёра: «Как принципы инкапсуляции и ООП применяются в промышленном коде» (Яндекс), «Проблемы и практики правильной реализации конструкторов и операторов в больших системах» (Сбер), «Friend и mutable: редкие, но полезные инструменты в C++» (Авито).

Практические кейсы от индустриального партнёра: «Проектирование классов для модуля обработки данных» (Яндекс), «Реализация системы учёта пользователей с использованием инкапсуляции и статических методов» (Сбер).

Хакатоны: «Class Design Challenge: спроектируй и реализуй систему классов для управления складом», «Constructor Battle: реализуй класс с полной поддержкой конструкторов, операторов и правил C++».

Ключевые темы:

Идея ООП. Понятия класса и структуры, членов класса. Поля и методы, понятие инкапсуляции. Модификаторы доступа.

Конструкторы и деструкторы. Конструктор по умолчанию. Перегрузка конструкторов. Конструктор копирования, его сигнатура и схема реализации. Пример, когда необходим нетривиальный конструктор копирования и оператор присваивания. Правила генерации компилятором конструкторов. Ключевые слова default и delete в контексте определения функций-членов.

Операторы “точка” и “стрелочка”. Ключевое слово this и пример использования.

Оператор присваивания, его сигнатура и схема реализации. “Правило трех”.

Проблема с инициализацией констант и ссылок. Решение: списки инициализации в конструкторах.

Ключевое слово explicit. Пример с конструктором String(int n).

Константные и неконстантные методы, примеры.

Ключевое слово mutable, пример применения.

Понятие дружественных функций и классов, ключевое слово friend.

Проблема вызова конструкторов из других конструкторов. Решение: делегирующие конструкторы.

Статические поля и методы, пример. Локальные статические переменные.

Указатели на члены и указатели на методы. Синтаксис объявления, пример использования.

Операторы “точка со звездочкой” и “стрелочка со звездочкой”.

4. Перегрузка операторов

Форматы обучения: Лекции, интерактивные семинары, разбор кейсов, проектные практикумы, мастер-классы, групповые обсуждения.

Конкурсы индивидуальных проектов: «Реализация класса BigInteger с перегрузкой арифметических операторов», «Создание собственного контейнера с оператором []», «Перегрузка операторов сравнения и проверка корректности реализации», «Реализация функционального класса (функтора) с оператором ()», «Перегрузка операторов потокового ввода-вывода (<< и >>) для пользовательского класса».

Групповые проекты: «Разработка библиотеки математических объектов с перегруженными операторами», «Создание класса Matrix с перегрузкой операторов +, -, *, []», «Проектирование системы классов с перегрузкой операторов сравнения и логических операций».

Мастер-классы от индустриального партнёра: «Перегрузка операторов в реальном промышленном коде: где полезно, а где опасно» (Яндекс), «Реализация операторов ввода-вывода для сложных структур данных» (Сбер), «Функторы и компараторы в стандартных алгоритмах C++» (Авито).

Практические кейсы от индустриального партнёра: «Проектирование класса для финансовых расчётов с перегрузкой операторов арифметики» (Тинькофф), «Использование перегруженных операторов для реализации пользовательских структур данных» (Яндекс).

Хакатоны: «Operator Overloading Challenge: реализуй и протестируй класс с перегрузкой 10+ операторов», «Matrix Battle: создайте класс Matrix с перегрузкой операторов +, *, [] и потокового ввода-вывода».

Ключевые темы:

Общая идея перегрузки операторов. Перегрузка арифметических операторов на примере класса BigInteger: бинарные операторы, составные присваивания с ними, правильное выражение одного через другое. Проблема с корректностью выражений вида “ $x+y=5$ ”. Проблема в случае левого операнда - не объекта класса (выражения вида “ $5+x$ ”). Перегрузка операторов << и >> на примере потокового ввода-вывода.

Перегрузка операторов сравнения, правильное выражение одних сравнений через другие.

Перегрузка инкремента и декремента (префиксного и постфиксного).

Перегрузка оператора [] (квадратные скобки). Правильное соблюдение константности при перегрузке оператора [].

Перегрузка оператора “круглые скобки”. Понятие функтора и функционального класса, компаратора. Пример использования в стандартных алгоритмах.

Особенности перегрузки операторов “логическое И”, “логическое ИЛИ” и “запятая”.

Особенности перегрузки операторов “унарная звездочка”, “унарный амперсанд” и “стрелочка”.

Перегрузка операторов приведения типа. Еще одно применение ключевого слова explicit.

5. Наследование (inheritance)

Форматы обучения: Лекции, интерактивные семинары, разбор кейсов, проектные практикумы, мастер-классы, групповые обсуждения.

Конкурсы индивидуальных проектов: «Реализация иерархии классов с виртуальными и неvirtуальными методами», «Создание системы геометрических фигур с поддержкой полиморфизма», «Пример ромбовидного наследования и его разрешение через виртуальное наследование», «Демонстрация проблемы slicing при копировании объектов», «Реализация абстрактного класса с чисто виртуальными методами».

Групповые проекты: «Разработка системы классов для моделирования транспортных средств (автомобиль, электромобиль, грузовик) с использованием множественного наследования», «Создание полиморфной системы для обработки платежей (карта, криптовалюта, наличные)», «Построение иерархии классов для имитации работы файловой системы».

Мастер-классы от индустриального партнёра: «Полиморфизм и виртуальные функции в промышленном коде» (Яндекс), «Наследование и проектирование архитектуры больших систем» (Сбер), «Ошибки и best practices множественного наследования» (Авито).

Практические кейсы от индустриального партнёра: «Использование наследования для построения плагиновой архитектуры» (JetBrains), «Иерархии классов в игровых движках» (myGames), «Применение RTTI и dynamic_cast в разработке инструментов отладки» (Лаборатория Касперского).

Хакатоны: «Polymorphism Challenge: создать и протестировать систему классов с поддержкой виртуальных функций», «Diamond Problem Hack: реализовать безопасное решение ромбовидного наследования», «Virtual Destructor Quest: соревнование на корректное проектирование иерархий классов».

Ключевые темы:

Поиск имен при наследовании. Соккрытие имен наследником. Явный вызов методов родителя у наследника. Использование `::` и `using`. Проблемы с видимостью названий родителей и их полей у потомков в случае двухуровневого наследования, где первый уровень - приватное наследование. Правила действия слова `friend` в этих случаях.

Порядок вызова конструкторов и деструкторов при наследовании. Проблема с инициализацией родителей при определении конструктора наследника, вновь применение списков инициализации. Правила размещения объектов классов-наследников в памяти.

Множественное наследование, неоднозначности при нем, проблема ромбовидного наследования. Примеры разрешения неоднозначности с помощью приведений типов и оператора `::`, комбинации всего этого с приватным наследованием, сдвиги указателей.

Виртуальное наследование. Особенности комбинации виртуального и неvirtуального наследования.

Приведение типов между родителем и наследником: срезка при копировании, приведение указателей, приведение ссылок. Особенности `static_cast`, `reinterpret_cast` между родителями и наследниками (а также указателями или ссылками на них). Оператор `dynamic_cast`, его отличие от `static_cast`.

Виртуальные функции, их общая идея и отличие от неvirtуальных. Особенности размещения в памяти классов с виртуальными функциями, понятие полиморфизма. Полиморфные классы. Понятие о таблице виртуальных функций.

Виртуальный деструктор и его предназначение.

Абстрактные классы и «чисто виртуальные» (`pure virtual`) функции, их особенности. Чисто виртуальный деструктор. Ошибка «`pure virtual function call`» и ее возникновение.

Ключевые слова `override` и `final` при наследовании, их предназначение.

Механизм RTTI. Оператор `typeid` и динамическое определение типа объекта. Класс `std::type_info`.

Проблема с вызовом виртуальных функций в конструкторах. Проблема с аргументами по умолчанию в виртуальных функциях.

Empty base optimization, примеры.

6. Шаблоны (templates)

Форматы обучения: Лекции, интерактивные семинары, разбор кейсов, проектные практикумы, мастер-классы, групповые обсуждения.

Конкурсы индивидуальных проектов: «Библиотека обобщённых алгоритмов: сортировка/поиск на шаблонах функций с компараторами», «Полная и частичная специализация: обобщённый вектор + специализация для `bool`», «Собственные `type_traits`: `remove_const`, `remove_reference`, `is_same` с проверочными тестами», «`Variadic templates`: форматтер строк `format(fmt, args...)` с `sizeof...` и распаковкой пакетов», «`CRTP`: статический полиморфизм для логирования/счётчиков вызовов».

Групповые проекты: «Обобщённые контейнеры с параметрами, не являющимися типами (фиксированный буфер `N`)», «Шаблонный пул объектов с политиками (`policy-based design`) и `using`-алиасами», «Библиотека компараторов и функторов (стандартные и пользовательские) для `std::sort/std::map`».

Мастер-классы от индустриального партнёра: «Специализации и перегрузки в большом коде: как избежать неоднозначностей выбора шаблонов» (Яндекс), «`Variadic templates` и метапрограммирование в высокопроизводительных библиотеках» (Сбер), «`CRTP` и статический полиморфизм в продакшене: паттерны и подводные камни» (Авито).

Практические кейсы от индустриального партнёра: «Шаблонные адаптеры данных для разных форматов логов/событий без ветвлений в рантайме» (Авито), «`Type-safe` слой доступа к данным (`traits` + специализации) в сервисах риска» (Сбер), «Унификация компараторов и хешеров для индексов поиска» (Яндекс).

Хакатоны: «Template Jam: за 24 часа создать мини-STL (контейнер + итераторы + алгоритмы)», «Variadic Formatter Challenge: безопасный, быстрый форматтер без printf и std::format», «CRTP Sprint: статически полиморфная иерархия без виртуальных вызовов».

Ключевые темы:

Мотивировка и общая идея шаблонов. Шаблоны классов, шаблоны функций, синтаксис объявления, примеры использования, связь шаблонов и полиморфизма, статический полиморфизм.

Специализации шаблонов, принцип “частное предпочтительнее общего” применительно к шаблонам. Частичные и полные специализации. Принцип “лучше точное соответствие, чем приведение типа”. Разница между специализацией и перегрузкой для шаблонных функций. Правила выбора компилятором кандидатов на специализацию и на перегрузку.

Ключевое слово typedef, его предназначение. Шаблонные typedef’ы, использование слова using.

Проблема с обращением к typedef’ам внутри шаблонных классов. Применение ключевого слова typename для решения этой проблемы.

Примеры реализации простейших type_traits с помощью шаблонных структур и typedef’ов внутри них: remove_const, remove_reference.

Правила вывода типов для шаблонов. Отбрасывание ссылок при выводе типа. Разбор случаев со ссылками и константами.

Параметры шаблонов, не являющиеся типами (пример: массив константной длины). Параметры шаблонов, являющиеся шаблонами (“template template parameters”).

Шаблоны с переменным количеством аргументов (variadic templates). Синтаксис использования. “Откусывание” шаблонных аргументов по одному. Оператор “sizeof...”.

Функциональные классы и функциональные объекты (функторы), схема использования. Компараторы. Пример: компаратор в std::sort. Стандартные компараторы (std::less, std::greater, std::equal и т. п.), их реализация.

Curiously Recurring Template Pattern (CRTP).

7. Исключения (exceptions)

Форматы обучения: Лекции, интерактивные семинары, разбор кейсов, проектные практикумы, мастер-классы, групповые обсуждения.

Конкурсы индивидуальных проектов: «Реализация калькулятора с обработкой ошибок деления на ноль через исключения», «Создание класса контейнера с гарантией безопасности при исключениях», «Тестирование поведения noexcept функций при разных сценариях», «Симулятор работы системы с перехватом всех исключений».

Групповые проекты: «Реализация библиотеки для управления ресурсами (RAII) с корректной обработкой исключений», «Проектирование системы логирования и обработки ошибок в многопоточной среде», «Сравнительный анализ старых спецификаций исключений и noexcept на примере библиотеки».

Мастер-классы от индустриального партнёра: «Исключения и обработка ошибок в промышленных системах на C++» (Яндекс), «Гарантии безопасности и тестирование отказоустойчивости ПО» (Сбер), «Проектирование исключений в высоконагруженных системах».

Практические кейсы от индустриального партнёра: «Обработка ошибок в платежных системах: когда использовать исключения, а когда коды возврата» (Сбер), «Проектирование API с безопасной обработкой ошибок для клиентов» (Авито), «RAII и исключения: надёжное управление памятью в распределённых сервисах» (Яндекс).

Хакатоны: «Exception Handling Challenge: проектирование отказоустойчивого приложения с гарантией строгой безопасности», «Noexcept Race: оптимизация библиотеки с помощью корректного использования noexcept», «Resilience Hack: создание системы, устойчивой к неожиданным исключениям».

Ключевые темы:

Общая идея, мотивировка использования исключений, оператор `throw` и конструкция `try...catch`. Примеры стандартных операторов, генерирующих исключения.

Разница между исключениями и ошибками времени выполнения. Ошибки, не являющиеся исключениями, и исключения, не являющиеся ошибками.

Правила ловли и повторного бросания исключений, приведения типов при ловле исключений. Ловля всех исключений. Правила выбора блока `catch` компилятором в случае, когда подходят разные блоки.

Копирование при бросании и ловле исключений, исключения и наследование. Особенности перехвата исключений по значению и по ссылке, по ссылке на базовый класс.

Спецификации исключений в старом стиле и их проблемы, `unexpected exceptions` (неожиданные исключения). спецификации исключений в стиле C++11, оператор и спецификатор `noexcept`. Условный `noexcept`.

Исключения в конструкторах и проблема утечки памяти при исключениях.

Исключения в деструкторах, функция `uncaught_exception`, функции `terminate` и `set_terminate`.

Гарантии безопасности при исключениях: базовая и строгая.

Function-try блоки, их особенности.

8. Аллокаторы (allocators)

Форматы обучения: Лекции, интерактивные семинары, разбор кейсов, проектные практикумы, мастер-классы, групповые обсуждения.

Конкурсы индивидуальных проектов: «Реализация собственного аллокатора памяти (PoolAllocator)», «Создание класса с перегрузкой операторов `new/delete`», «Демонстрация работы `placement new` на примере ручного управления объектами», «Сравнительный эксперимент: стандартный аллокатор vs кастомный».

Групповые проекты: «Разработка библиотеки с поддержкой пользовательских аллокаторов», «Оптимизация использования памяти в контейнерах STL через `custom allocators`», «Симуляция работы системы с ограниченными ресурсами памяти».

Мастер-классы от индустриального партнёра: «Оптимизация работы с памятью в высоконагруженных сервисах» (Яндекс), «Аллокаторы в системах реального времени» (Ростелеком), «Управление памятью в C++ и `embedded`-разработке».

Практические кейсы от индустриального партнёра: «Оптимизация работы Redis с помощью `memory pools`», «Использование аллокаторов в игровых движках» (My.Games), «Контроль утечек памяти в промышленных проектах» (Сбер).

Хакатоны: «Allocator Challenge: реализация кастомного `memory pool` для обработки миллионов объектов», «Low Memory Hack: разработка приложения под экстремальные ограничения памяти», «Memory Debug Race: обнаружение и исправление ошибок при использовании `new/delete`».

Ключевые темы:

`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.

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

Для проведения занятий по дисциплине используется учебная аудитория, оснащённая компьютерной техникой с необходимым программным обеспечением и мультимедийным оборудованием (проектор, интерактивная панель или экран, звуковая система).

Разбор кейсов, групповые обсуждения, проектные практикумы и мастер-классы реализуются посредством облачных SaaS-сервисов с бесплатным или условно-бесплатным доступом (системы совместной работы над кодом, облачные IDE, сервисы для коллективного управления проектами).

Привлечение специалистов из индустрии осуществляется с использованием программного обеспечения для организации видеоконференц-связи (Zoom, MS Teams, Google Meet или аналоги).

Для выполнения лабораторных и практических заданий студенты обеспечиваются доступом к:

- современным компиляторам C++ (GCC, Clang, MSVC),
- интегрированным средам разработки (Visual Studio, CLion, Qt Creator или аналоги),
- системам контроля версий (Git, GitHub/GitLab),
- отладочным инструментам (gdb, Valgrind или встроенные отладчики IDE).

При необходимости учебный процесс дополняется использованием серверных мощностей для коллективной разработки и размещения студенческих проектов.

6. Перечень рекомендуемой литературы

Основная литература

1. Программирование на C++ [Электронный ресурс], Электрон. версия печ. публикации / Н. Дейл, Ч. Уимз, М. Хедингтон. — М., ДМК Пресс, 2007

Дополнительная литература

Бьерн Страуструп. Программирование. Принципы и практика с использованием C++. — М.: Вильямс, 2021.

Скотт Мейерс. Эффективный и современный C++. 42 рекомендации по использованию C++11 и C++14. — СПб.: Питер, 2020.

Герб Саттер, Андрей Александреску. Стандарты программирования на C++. — М.: ДМК Пресс, 2019.

Николай Джосаттис. Стандартная библиотека C++. Справочник. 3-е издание. — М.: ДМК Пресс, 2020.

Шилдт Г. C++. Полное руководство. 5-е издание. — М.: Вильямс, 2021.

Липпманн С., Лажой Ж., Моо Б. Язык программирования C++. Базовый курс. 5-е издание. — М.: Вильямс, 2020.

Андрей Александреску. Современное проектирование на C++. — М.: ДМК Пресс, 2018.

Дмитрий Вандюк. C++17 STL. Стандартная библиотека шаблонов. Полное руководство. — СПб.: Питер, 2021.

Николай Джосаттис. C++17. Полное руководство. — М.: ДМК Пресс, 2020.

Кай Геринг. C++20. Концепты, корутины и диапазоны. Руководство разработчика. — М.: ДМК Пресс, 2022.

Научные публикации (пример):

Postnicov V. et al. Evaluation of three-point correlation functions from structural images on CPU and GPU architectures: Accounting for anisotropy effects //Physical Review E. – 2024. – Т. 110. – №. 4. – С. 045306. DOI 10.1103/PhysRevE.110.045306

Akhtyamov P. et al. GPU-accelerated Kendall distance computation for large or sparse data //GigaScience. – 2024. – Т. 13. – С. giae088. DOI 10.1093/gigascience/giae088

Kozhemyachenko A. et al. Modification of the grid-characteristic method on chimera meshes for 3D problems of railway non-destructive testing //Lobachevskii Journal of Mathematics. – 2023. – Т. 44. – №. 1. – С. 376-386. DOI 10.1134/S1995080223010262

Ковалев Д., Безносиков А., Бородич Е. и др. Оптимальное градиентное скольжение и его применение к распределенной оптимизации // arXiv:2205.15136 [math.OC]. 2022. URL: <https://arxiv.org/abs/2205.15136>

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

Не используются

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

1. Программное обеспечение для разработки

- компиляторы C++ (GCC, Clang, MSVC);
- интегрированные среды разработки (Visual Studio, CLion, Qt Creator, Code::Blocks или аналоги);
- инструменты для отладки и профилирования программ (gdb, lldb, Valgrind, встроенные средства IDE).

2. Системы управления версиями и совместной разработки

- Git, облачные репозитории и сервисы (GitHub, GitLab, Bitbucket).

3. Системы дистанционного обучения и коммуникации

- LMS (Moodle, Canvas или аналогичные платформы);
- ПО для видеоконференций (Zoom, MS Teams, Google Meet и аналоги).

4. Облачные сервисы и SaaS-платформы

- онлайн-компиляторы и среды (Repl.it, GitHub Codespaces, Wandbox, JDoodle);
- сервисы для совместной работы над проектами (Trello, Jira, Miro и аналоги).

5. Информационные справочные системы

- онлайн-документация по стандарту языка (cppreference, ISO C++ drafts);
- технические порталы и базы знаний (MSDN, документация GCC и Clang).

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

Для успешного освоения дисциплины студентам рекомендуется:

Систематически посещать лекции и семинары, фиксировать ключевые понятия, примеры кода и разборы типичных ошибок.

Регулярно выполнять практические задания, так как усвоение языка C++ невозможно без самостоятельного программирования и отладки решений.

Работать с системами контроля версий (Git) для хранения и ведения собственных проектов, формируя навыки командной разработки.

Постепенно усложнять проекты: от простых консольных программ к проектам с использованием ООП, шаблонов, стандартной библиотеки контейнеров и умных указателей.

Выполнять задания исследовательского характера: сравнение производительности различных реализаций, анализ ошибок, применение современных подходов (move-семантика, лямбда-выражения, метапрограммирование).

Использовать рекомендованную литературу и справочные ресурсы, включая официальную документацию ISO C++, `сppreference` и ресурсы промышленных партнёров (Яндекс, Сбер, Авито).

Участвовать в конкурсах и хакатонах, организуемых в рамках курса, для закрепления знаний на практике и приобретения опыта работы с реальными кейсами.

Формировать электронное портфолио (репозиторий проектов), что позволит продемонстрировать навыки владения C++ при дальнейшем трудоустройстве или участии в стажировках.

ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

по направлению: Прикладная математика и информатика
профиль подготовки: АІ360: Передовые методы искусственного интеллекта
Физтех-школа Прикладной Математики и Информатики
кафедра алгоритмов и технологий программирования
курс: 1
квалификация: бакалавр

Семестр, формы промежуточной аттестации: 1 (осенний) - Дифференцированный зачет

Разработчик: И.С. Мещерин, ассистент

1. Компетенции, формируемые в процессе изучения дисциплины

Код и наименование компетенции	Индикаторы достижения компетенции
ОПК-1 Способен применять фундаментальные знания, полученные в области физико-математических и (или) естественных наук и использовать их в профессиональной деятельности	ОПК-1.2 Способен строить математические модели, производить количественные расчеты и оценки
ОПК-2 Способен использовать современные информационные технологии и программные средства при решении задач профессиональной деятельности, соблюдая требования информационной безопасности	ОПК-2.2 Знает и умеет применять численные математические методы и прикладное программное обеспечение для решения научных задач в профессиональной области
РЛ-3 Способен применять языки программирования C/C++ для решения задач в области ИИ	РЛ-3.1 Разрабатывает и отлаживает эффективные многопоточные решения на C++, тестирует, испытывает и оценивает качество таких решений
	РЛ-3.2 Разрабатывает и отлаживает системы ИИ на C++ под конкретные аппаратные платформы с ограничениями по вычислительной мощности, в том числе для встроженных систем
	РЛ-3.3 Разрабатывает и отлаживает решения на C++, использующие GPU и FPGA для массовой параллелизации вычислений в рамках общей системы ИИ, с применением как готовых решений, так и разработкой своих

2. Показатели оценивания компетенций

В результате изучения дисциплины «Программирование на языке C++. Продвинутый поток (ФПМИ)» обучающийся должен:

знать:

- алгоритмы на графах и структуры данных, связанные с ними;
- оценки сложности стандартных алгоритмов;
- стандартные алгоритмы на графах и используемые структуры данных, подходы к модификации классических алгоритмов;
- разнообразные классические задачи в теории графов и асимптотические сложности их решений.

уметь:

- формулировать задачи в терминах изученных теорий, выбирать подходящий алгоритм для поставленной задачи;
- разрабатывать комбинации алгоритмов для решения поставленной задачи;
- оценивать сложности алгоритмов, их модификаций и комбинаций, в том числе с помощью амортизационного анализа;
- выбирать подходящие структуры данных для конкретной задачи;
- реализовывать алгоритм в обобщенной форме на языке программирования c++;
- реализовывать стандартные алгоритмы на графах и структуры данных на языке программирования C++.

владеть:

- методами декомпозиции задач в области информационных технологий и построения единого решения с использованием изученных алгоритмов;
- методами оценки сложности алгоритмов, их модификаций и комбинаций.

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

1. Что такое выражения, операторы? Для чего предназначены и что по стандарту делают следующие операторы: тернарный оператор, оператор “запятая”, унарная звездочка, унарный амперсанд, операторы “точка” и “стрелочка”, двойное двоеточие, префиксный и постфиксный инкремент, бинарные & и &&, операторы простого и составного присваивания, операторы << и >>?
2. Объясните идею ссылок (references). Для чего они нужны, чем они отличаются от указателей? Что такое “передача аргументов по ссылке и по значению”? Как в C++03 реализовать функцию swap?
3. Что такое конструкторы, деструкторы, для чего они нужны, каков синтаксис их определения? Те же вопросы про конструктор копирования и оператор присваивания.
4. Что такое наследование? В чем разница между приватным и публичным наследованием?
5. Что такое полиморфизм? Приведите пример полиморфизма в C++.
6. Что такое исключения? Как пользоваться механизмом обработки исключений? Как бросить, поймать исключение? Приведите какой-нибудь пример.
7. Что такое компараторы? Что такое функциональные объекты? Приведите хоть один пример.
8. Что такое умные указатели? Для решения каких проблем они нужны? Приведите пример использования.
9. Что такое лямбда-выражения, каков их синтаксис? Приведите хоть один пример использования.
10. Что такое SFINAE? Приведите хотя бы один пример (на уровне идей, можно без реализации).

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

1. Объясните понятия compilation error, runtime error и undefined behaviour. Приведите по паре примеров того, другого и третьего.
2. Что такое стековая память и динамическая память? Что такое stack overflow? Зачем нужен оператор new? Что такое утечки памяти? Что такое сборка мусора?
3. Что такое класс, структура, в чем разница между ними? Что такое поля, методы, модификаторы доступа, инкапсуляция?
4. Что такое перегрузка функций? Что такое перегрузка операторов? Приведите примеры операторов, которые можно и нельзя перегружать. Приведите пример перегрузки хоть какого-нибудь оператора (напишите сигнатуру его перегрузки).
5. Что такое виртуальные функции, чисто виртуальные функции, абстрактные классы? Что такое виртуальное наследование?
6. Что такое шаблоны? Что такое инстанцирование шаблонов, специализация шаблонов? Приведите пример каких-нибудь шаблонов из STL, обладающих специализацией.
7. Рассмотрим контейнеры std::vector, std::list, std::deque. Каковы основные операции, предоставляемые ими, и скорость работы этих операций?
8. Рассмотрим контейнеры std::map, std::set, std::unordered_map, std::unordered_set. Каковы основные операции, предоставляемые ими, и скорость работы этих операций?
9. Что такое итераторы? Какие виды итераторов существуют? Зачем они вообще нужны? Какие итераторы поддерживает каждый из контейнеров, упомянутых в предыдущих двух вопросах?
10. Для чего нужна move-семантика? Расскажите в общих чертах, что это такое. Как правильно реализовать функцию swap в C++11?
11. Что такое аллокаторы? Какова общая идея класса std::allocator, как и для чего он используется?
12. Напишите вычисление n-го числа Фибоначчи (в пределах long long) на этапе компиляции, не используя слова constexpr.

Критерии 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.

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

Во время проведения дифференцированного зачета обучающиеся могут пользоваться программой дисциплины. При проведении устного дифференцированного зачёта обучающемуся предоставляется 30 минут на подготовку.