ЦЕНТРАЛЬНЫЙ УНИВЕРСИТЕТ

УТВЕРЖДЕНА

Решением Ученого совета
АНО ВО «Центральный университет»
«24» июня 2025 г.
Протокол №2

Рабочая программа дисциплины (модуля) «Язык программирования C++»

Направление подготовки: *02.03.01 Математика и компьютерные науки*

Направленность (профиль) подготовки: *Разработка*

Квалификация (степень) выпускника: *бакалавр*

Форма обучения: *очная*

Срок освоения программы: *4 года*

Год набора: *2025*

Москва
2025

Электронный документ

Содержание

1. Краткая характеристика дисциплины (модуля)

Рабочая программа дисциплины (модуля) «Язык программирования C++» составлена в соответствии с федеральным государственным образовательным стандартом высшего образования – бакалавриат по специальности 02.03.01 Математика и компьютерные науки, профиль Разработка, утвержденный приказом Министерства науки и высшего образования Российской Федерации № 807 от 23.08.2017 года.

Изучение дисциплины (модуля) «Язык программирования C++» важно для освоения эффективного программирования системного и прикладного уровня, а также для понимания принципов работы с памятью и объектно-ориентированного подхода. Это позволяет создавать производительные и надежные приложения, востребованные в различных областях, от игр до встроенных систем.

Место дисциплины (модуля) в структуре образовательной программы

Настоящая дисциплина (модуль) включена в учебный план по программе подготовки бакалавриата по направлению 02.03.01 Математика и компьютерные науки, профиль Разработка и входит в вариативную часть Блока 1, формируемую участниками образовательных отношений.

Дисциплина (модуль) является выборной и доступна для изучения на 3 или 4 курсе в 5, 6, или 7 семестре на выбор.

Цель изучения дисциплины (модуля): формирование навыков эффективного программирования на языке C++ для создания высокопроизводительных и надежных программных решений.

Задачи изучения дисциплины (модуля):

— изучить основные концепции и модели облачных вычислений, включая сервисы IaaS, PaaS и SaaS;

— освоить принципы проектирования и развертывания облачных архитектур с учетом безопасности и масштабируемости;

— научиться применять инструменты и платформы облачных провайдеров для решения практических задач в разработке программного обеспечения.

В результате освоения дисциплины (модуля) обучающийся должен:

знать:

— конструкции актуальной версии языка C++;

— функции и классы стандартной библиотеки;

уметь:

— решать задачи с помощью языка C++;

— писать поддерживаемый код;

— работать с динамической памятью;

владеть:

— навыком ведения разработки сложных программ на языке C++.

2. Перечень планируемых результатов обучения

Компетенции, формируемые в результате освоения дисциплины (модуля) при проведении учебных занятий в форме контактной работы обучающихся с педагогическими работниками Университета и в форме самостоятельной работы обучающихся:

Компетенция	Содержание компетенции	Индикатор компетенции	Перечень планируемых результатов обучения по дисциплине (модулю)
УК-1.	Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач	УК-1.1.	Знает методы поиска и анализа информации в области разработки, основные принципы критической оценки источников информации и их релевантности
		УК-1.2.	Умеет критически оценивать источники информации и синтезировать данные из различных источников для решения задач, применять системный подход к анализу и решению комплексных проблем
		УК-1.3.	Имеет практический опыт работы с современными инструментами и технологиями для обработки информации, формулировании и структурировании задач на основе полученной информации
ОПК-1.	Способен консультировать и использовать фундаментальные знания в области математического анализа, комплексного и функционального анализа алгебры, аналитической геометрии, дифференциальной геометрии и топологии, дифференциальных уравнений, дискретной математики и математической логики, теории вероятностей, математической статистики и случайных процессов, численных методов, теоретической механики в профессиональной деятельности	ОПК-1.1.	Знает основные концепции и теории в области математического анализа и смежных дисциплин; методы и подходы, используемые в различных областях математики
		ОПК-1.2.	Умеет применять математические методы для решения профессиональных задач
		ОПК-1.3.	Имеет практический опыт разработки и реализация математических моделей в профессиональной деятельности
ОПК-4.	Способен находить, анализировать, реализовывать программно и использовать на практике математические алгоритмы, в том числе с применением современных вычислительных систем	ОПК-4.1.	Знает базовые основы современного математического аппарата, связанного с проектированием, разработкой, реализацией и оценкой качества программных продуктов и программных комплексов в различных областях человеческой деятельности

		ОПК-4.2.	Умеет использовать этот математический аппарат в профессиональной деятельности
		ОПК-4.3.	Имеет практический опыт применения современного математического аппарата, связанного с проектированием, разработкой, реализацией и оценкой качества программных продуктов и программных комплексов в различных областях человеческой деятельности
ОПК-6.	Способен разрабатывать алгоритмы и компьютерные программы, пригодные для практического применения	ОПК-6.1.	Знает алгоритмы разработки, компьютерные программы, а также алгоритмы вычислительной математики
		ОПК-6.2.	Умеет разрабатывать математические программные продукты и комплексы с использованием современных технологий программирования
		ОПК-6.3.	Имеет практический опыт разработки интеллектуальных информационных систем для визуализации результатов исследований
ПК-3.	Способен использовать методы математического и алгоритмического моделирования при решении теоретических и прикладных задач	ПК-3.1.	Знает основные методы математического и алгоритмического моделирования, а также их применение для решения теоретических и прикладных задач
		ПК-3.2.	Умеет разрабатывать и применять математические модели и алгоритмы для решения различных задач, анализируя полученные результаты
		ПК-3.3.	Имеет практический опыт использования методов математического и алгоритмического моделирования в реальных проектах или исследованиях

3. Тематический план

№п/п	Наименование раздела дисциплины (модуля)	Трудоемкость, академические часы				ТКУ (текущий контроль успеваемости)
		Очная форма				
		Контактная работа		Контроль	Самостоятельная работа	
		Лекции	Семинары (практические занятия)			
1	Инструменты для разработки на C++	4	4		18	Кейс Коллоквиум
2	ООП на языке C++ и наследование	4	4		19	Кейс Коллоквиум
3	Шаблоны	4	4		18	Кейс Коллоквиум
4	Полиморфизм и виртуальные функции	4	4		19	Кейс Коллоквиум
5	Исключения	4	4		18	Кейс Коллоквиум
6	Контейнеры и итераторы	4	4		19	Кейс Коллоквиум
7	Модули и производительность	4	4		19	Кейс Коллоквиум
	Зачет с оценкой			4		Проект
Итого:		***28***	***28***	***4***	***130***	
Объем дисциплины (модуля) (в ак. ч.)		***190***				
Объем дисциплины (модуля) (в зач. ед.)		***5***				

4. Содержание дисциплины (модуля)

№п/п	Наименование раздела дисциплины (модуля)	Содержание дисциплины (модуля) по темам
1	Инструменты для разработки на C++	Инструменты для разработки на C++
2	ООП на языке C++ и наследование	ООП на C++ и наследование. Семантика объектов. Перегрузка и операторные функции
3	Шаблоны	Шаблоны. Обобщённое программирование
4	Полиморфизм и виртуальные функции	Полиморфизм и виртуальные функции. Идиомы современного C++
5	Исключения	Исключения и обработка ошибок. Параллельное программирование
6	Контейнеры и итераторы	Контейнеры и итераторы. Алгоритмы и диапазоны. Управление памятью
7	Модули и производительность	Модули и производительность. Генерация и переносимость кода

5. Учебно-методическое обеспечение

Университет располагает полным набором лицензионного и свободно распространяемого программного обеспечения, включая продукты отечественного производства.

Каждый студент в течение всего периода обучения получает индивидуальный неограниченный доступ к электронно-библиотечной системе и электронной информационно-образовательной среде университета. Эти системы предоставляют возможность доступа к ресурсам из любой точки, где есть подключение к сети Интернет, как на территории университета, так и за его пределами.

Студентам обеспечен удаленный доступ к современным профессиональным базам данных и информационным справочным системам.

Основная литература:

1. Мейерс, С. Наиболее эффективное использование C++. 35 новых рекомендаций по улучшению ваших программ и проектов : практическое руководство / С. Мейерс ; пер. с англ. Р. В. Павлова. — 2-е изд. - Москва : ДМК Пресс, 2023. - 298 с. - ISBN 978-5-89818-563-3. - Текст : электронный. - URL: https://znanium.com/catalog/product/2107916.

2. Огнева, М. В. Программирование на языке C++: практический курс : учебник для вузов / М. В. Огнева, Е. В. Кудрина, А. А. Казачкова. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2025. — 342 с. — (Высшее образование). — ISBN 978-5-534-18949-0. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/563618.

3. Современный C++: безопасное использование : практическое руководство / Д. Лакос, Р. Витторио, Р. Хлебников, М. Алисдар ; пер. с англ. А. В. Снастина. – Москва : ДМК Пресс, 2023. - 1044 с. – ISBN 978-5-93700-134-4. - Текст : электронный. - URL: https://znanium.ru/catalog/product/2204223.

4. Гримм, Р. Параллельное программирование на современном C++ : практическое руководство / Р. Гримм ; пер. с англ. В. Ю. Винника. - Москва : ДМК Пресс, 2022. - 618 с. - ISBN 978-5-97060-957-6. - Текст : электронный. - URL: https://znanium.ru/catalog/product/2109583.

5. Павловская, Т. А. C/C++. Структурное и объектно-ориентированное программирование : практикум / Т. А. Павловская, Ю. А. Щупак. - Санкт-Петербург : Питер, 2021. - 352 с. - (Серия «Учебное пособие»). - ISBN 978-5-4461-9799-6. - Текст : электронный. - URL: https://znanium.com/catalog/product/1857042.

Дополнительная литература:

1. Полуэктова, Н. Р. Разработка веб-приложений : учебник для вузов / Н. Р. Полуэктова. — 2-е изд. — Москва : Издательство Юрайт, 2025. — 204 с. — (Высшее образование). — ISBN 978-5-534-18645-1. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/567610.

2. Зараменских, Е. П. Разработка информационных систем : учебник и практикум для вузов / Е. П. Зараменских. — 2-е изд. — Москва : Издательство Юрайт, 2025. — 78 с. — (Высшее образование). — ISBN 978-5-534-21420-8. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/571333.

6. Материально-техническое обеспечение

Университет располагает материально-технической базой, соответствующей действующим противопожарным правилам и нормам и обеспечивающей проведение всех видов дисциплинарной и междисциплинарной подготовки, практической и научно-исследовательской работ обучающихся, предусмотренных учебным планом.

Помещения, которые представляют собой учебные аудитории для проведения занятий лекционного типа, занятий семинарского (практического) типа, групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации, а также помещения для самостоятельной работы и помещения для хранения и профилактического обслуживания учебного оборудования. Помещения укомплектованы специализированной мебелью и техническими средствами обучения, служащими для представления учебной информации большой аудитории.

Изучение дисциплины (модуля) обеспечивается в учебных аудиториях, оснащенных:

— столами и стульями;
— компьютерной техникой;
— механическими калькуляторами;
— специализированным оборудованием, включая демонстрационное оборудование.

Помещения для самостоятельной работы обучающихся, в том числе приспособленные для использования инвалидами и лицами с ограниченными возможностями здоровья, оснащены компьютерной техникой с возможностью подключения к сети «Интернет» и обеспечением доступа в электронную информационно-образовательную среду Университета.

Обучающимся предоставляется доступ (в том числе удаленный) к ресурсам информационно-телекоммуникационной сети «Интернет», электронным ресурсам (в том числе электронным библиотечным системам, современным профессиональным базам данных и информационным справочным системам):

№	Наименование портала (издания, курса, документа)	Ссылка
1.	Научная электронная библиотека elibrary.ru библиотека	https://elibrary.ru/defaultx.asp
2.	База данных для IT-специалистов	https://habr.com
3.	База данных ScienceDirect	https://www.sciencedirect.com
4.	Официальный сайт Министерства науки и высшего образования Российской Федерации	https://minobrnauki.gov.ru/
5.	Федеральный портал «Российское образование"	https://www.edu.ru/
6.	Информационная система "Единое окно доступа к образовательным ресурсам"	http://window.edu.ru/
7.	Единая коллекция цифровых образовательных ресурсов	http://school-collection.edu.ru/
8.	Федеральный центр информационно - образовательных ресурсов	http://fcior.edu.ru/

Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю), в том числе комплект лицензионного программного обеспечения, современные профессиональные базы данных и информационные справочные системы:

Наименование ПО	Производство	Лицензионное / свободно распространяемое
Операционные системы:		
Microsoft Imagine (Windows Client, Server)	зарубежное	лицензионное
Браузеры:		
Яндекс.Браузер	отечественное	свободно распространяемое
Google Chrome	зарубежное	свободно распространяемое
Офисные приложения:		
Microsoft Imagine (Visio, OneNote)	зарубежное	лицензионное
TeXstudio	зарубежное	свободно распространяемое

Adobe Acrobat Reader	зарубежное	свободно распространяемое
Программное обеспечение для планирования и учета времени:		
Toggle app	зарубежное	свободно распространяемое
Системы управления проектами:		
Microsoft Imagine (Project)	зарубежное	лицензионное
Системы управления базами данных:		
Microsoft Imagine (SQL Server)	зарубежное	лицензионное
Системы резервного копирования (backup):		
Acronis Backup Advanced for HyperV	зарубежное	лицензионное
Справочно-правовые системы:		
КонсультантПлюс: справочно-правовая система	отечественное	лицензионное
Средства антивирусной защиты:		
Kaspersky Endpoint Security для бизнеса Стандартный Russian Edition	отечественное	лицензионное
Среды разработки:		
Visual Studio Code	зарубежное	свободно распространяемое
Bash (Unix shell)	зарубежное	свободно распространяемое
Anaconda	зарубежное	свободно распространяемое
Robotic Operating System	зарубежное	свободно распространяемое
CopelliaSim	зарубежное	свободно распространяемое
Google Colaboratory	зарубежное	свободно распространяемое
Пакеты программных средств и библиотек:		
AutoPsy	зарубежное	свободно распространяемое
Interactive Disassembler (IDA)	зарубежное	свободно распространяемое
Системы управления библиографической информацией:		
Zotero	зарубежное	свободно распространяемое
Сервисы и службы:		
Bind	зарубежное	свободно распространяемое
Docker	зарубежное	свободно распространяемое

7. Методические и оценочные материалы

Методические указания для обучающихся по освоению дисциплины (модуля)

В процессе изучения дисциплины (модуля) «Язык программирования С++» в рамках текущего контроля успеваемости используются такие виды учебной работы, как лекции, семинары, кейсы, коллоквиумы, проект, а также различные виды самостоятельной работы обучающихся по заданию преподавателя, направленные на развитие навыков профессиональной лексики, закрепление практических профессиональных компетенций, поощрение инициатив.

Лекция – систематическое, последовательное, монологическое изложение преподавателем учебного материала, как правило, теоретического характера.

В процессе лекций рекомендуется вести конспект лекций: кратко и схематично фиксировать основные идеи, выводы и обобщения лекции; выделять важные мысли, ключевые слова и термины. Необходимо отметить вопросы или материалы, которые вызывают затруднения, и попытаться найти ответы в рекомендованной литературе. Если разобраться в материале не удается, следует сформулировать вопрос и задать его преподавателю на консультации или во время семинарского (практического) занятия.

Участие в семинаре (аудиторная работа) – активная работа студента на семинаре, его ответы на вопросы преподавателя и участие в дискуссии.

Для успешного участия в семинаре студентам рекомендуется заранее ознакомиться с

темой обсуждения, прочитать необходимые материалы и подготовить вопросы. Важно активно слушать и вовлекаться в дискуссию, высказывая свои мнения и аргументируя их. При ответах на вопросы преподавателя стоит быть уверенным, четким и логичным, опираясь на изученный материал. Также полезно поддерживать диалог с однокурсниками, чтобы обогатить обсуждение и расширить свои знания.

Кейс – практическая работа студентов над реальными или смоделированными задачами, что позволяет студенту применять теоретические знания на практике.

Студент самостоятельно разрабатывает стратегию решения поставленной задачи, что способствует развитию навыков критического мышления и самостоятельного принятия решений. Такой подход помогает подготовить будущих специалистов к реальным вызовам в их профессиональной деятельности.

Коллоквиум – устные ответы на вопросы, список которых известен студенту заранее.

В процессе подготовки к коллоквиуму необходимо проанализировать учебные материалы, ознакомившись с лекциями, учебниками и дополнительными источниками, акцентируя внимание на ключевых темах. Рекомендуется создать структурированные конспекты, выделяя основные идеи, термины и формулы.

Проект – исследовательская работа по курсу и презентация результатов.

Для успешной подготовки к проекту: четко определите цели и задачи проекта, распределите роли и обязанности между участниками, а также установите сроки выполнения каждой части работы. Регулярно проводите встречи для обсуждения прогресса и решения возникающих вопросов.

Самостоятельная работа – работа студентов, направленная на углубленное изучение отдельных тем и вопросов учебной дисциплины (модуля).

В процессе самостоятельной работы студенты взаимодействуют с рекомендованными материалами при минимальном участии преподавателя. Задачи студента включают работу с конспектами лекций (обработка текста), повторное изучение учебных материалов, планов и тезисов ответов, изучение дополнительных тем, выполнение учебно-исследовательских заданий и другое.

Система оценивания результатов обучения по дисциплине (модулю)

Критерии получения уровня и оценивания сформированности компетенций по дисциплине (модулю) «Язык программирования С++»

Оценивание уровня учебных достижений, обучающихся по дисциплине (модулю), осуществляется в виде текущего контроля успеваемости и промежуточной аттестации.

Промежуточная аттестация по дисциплине (модулю) осуществляется в форме ***зачета с оценкой***, при этом проводится оценка компетенций, сформированных по дисциплине.

Для оценивания текущего контроля успеваемости и промежуточной аттестации используется десятибалльная шкала оценивания, которая соотносится с традиционной пятибалльной шкалой следующим образом:

Десятибалльная оценка	Пятибалльная оценка	Оценка за зачет	Общая характеристика результата обучения по дисциплине (модулю)
10	Отлично	Зачтено	Студент полностью владеет знаниями, изложенными в рабочей программе, и глубоко осмысляет дисциплину. Он
9	Отлично	Зачтено	
8	Отлично	Зачтено	

Десятибалльная оценка	Пятибалльная оценка	Оценка за зачет	Общая характеристика результата обучения по дисциплине (модулю)
			самостоятельно и логически последовательно отвечает на все вопросы, акцентируя внимание на наиболее важном. Умеет анализировать, сравнивать, классифицировать, обобщать, конкретизировать и систематизировать изученный материал, выделяя ключевые моменты и устанавливая причинно-следственные связи. Четко формулирует ответы, уверенно интерпретирует результаты анализов и других исследований, а также решает сложные задачи. Студент хорошо знаком с методами исследования, необходимыми для практической деятельности, и умеет связывать теоретические аспекты дисциплины (модуля) с практическими задачами.
7	Хорошо	Зачтено	Студент обладает знаниями предмета почти в полном объеме рабочей программы и самостоятельно, логически последовательно и всесторонне отвечает на все вопросы, акцентируя внимание на наиболее значимых моментах. Он умеет анализировать, сравнивать, классифицировать, обобщать, конкретизировать и систематизировать изученный материал, выделяя его ключевые аспекты и устанавливая причинно-следственные связи. Формулирует свои ответы, уверенно интерпретирует результаты анализов и других исследований, а также решает сложные ситуационные задачи. Студент хорошо знаком с методами исследования, необходимыми для практической деятельности, и умеет связывать теоретические аспекты предмета с практическими задачами.
6	Хорошо	Зачтено	
5	Удовлетворительно	Зачтено	Студент обладает базовыми знаниями по дисциплине (модулю), но испытывает трудности при самостоятельных ответах и использует неточные формулировки. В ходе ответов он допускает ошибки, касающиеся сути вопросов. Студент способен решать только самые простые задачи и владеет лишь минимальным
4	Удовлетворительно	Зачтено	

Десятибалльная оценка	Пятибалльная оценка	Оценка за зачет	Общая характеристика результата обучения по дисциплине (модулю)
			набором методов исследования.
3	Не сдан	Не зачтено	Студент не овладел обязательным минимумом знаний по предмету и не может ответить на вопросы, даже если преподаватель задает дополнительные наводящие вопросы.
2	Не сдан	Не зачтено	
1	Не сдан	Не зачтено	

Дисциплина (модуль) «Язык программирования C++» оценивается следующим образом:

Активность	Вес	Описание
Кейс	40%	Практическая работа студентов над реальными или смоделированными задачами, что позволяет студенту применять теоретические знания на практике
Коллоквиумы	30%	Устные ответы на вопросы, список которых известен студенту заранее
Проект	30%	Защита итогового проекта

Формула расчёта итоговой оценки по дисциплине (модулю) «Язык программирования C++»: «0,4 × среднее за кейсы + 0,3 × среднее за коллоквиумы + 0,3 × проект».

Текущий контроль успеваемости обучающихся по дисциплине (модулю)

Примерные задания для кейсов

Инструменты для разработки на C++

1. **Кейс: Настройка среды разработки с использованием Visual Studio Code и CMake.** Студент устанавливает VS Code, настраивает компилятор MinGW и использует CMake для сборки простого проекта "Hello World". Пример: Создать CMakeLists.txt и скомпилировать программу. Вопросы: Какие преимущества дает CMake? Как решить проблему с путями к компилятору?

2. **Кейс: Отладка программы с помощью GDB.** Студент пишет программу с ошибкой (например, бесконечный цикл) и использует GDB для пошаговой отладки. Пример: Установить breakpoint и просмотреть переменные. Вопросы: Как интерпретировать вывод GDB? Какие альтернативы GDB существуют?

3. **Кейс: Интеграция с Git для управления версиями кода.** Студент создает репозиторий, коммитит изменения в коде калькулятора и разрешает merge-конфликт. Пример: Использовать команды git add, commit, push. Вопросы: Как избежать конфликтов? Какие инструменты GitHub помогают в командной разработке?

4. **Кейс: Профилирование производительности с Valgrind.** Студент анализирует утечки памяти в программе, используя Valgrind. Пример: Запустить valgrind --leak-check=full ./program. Вопросы: Как интерпретировать отчет Valgrind? Какие другие инструменты для профилирования вы знаете?

ООП на языке C++ и наследование

1. **Кейс: Реализация класса "Фигура" с наследованием.** Студент создает базовый класс Shape с методами area() и perimeter(), и наследует от него Circle и Rectangle. Пример: Использовать virtual для полиморфизма. Вопросы: Почему нужен виртуальный деструктор? Как добавить геттеры/сеттеры?

2. **Кейс: Перегрузка операторов для класса Vector.** Студент перегружает операторы + и * для математического вектора. Пример: operator+ (const Vector& other) { return Vector(x + other.x, y + other.y); }. Вопросы: Какие операторы нельзя перегружать? Как реализовать оператор вывода <<?

3. **Кейс: Семантика объектов в конструкторах и деструкторах.** Студент пишет класс ResourceManager с RAII (ресурсами в конструкторе), управляя файлами. Пример: Конструктор открывает файл, деструктор закрывает. Вопросы: Что такое правило трех? Как избежать копирования объектов с ресурсами?

4. **Кейс: Наследование и композиция в классе "Автомобиль".** Студент моделирует класс Car с наследованием от Vehicle и композицией Engine. Пример: Car содержит объект Engine. Вопросы: Когда использовать наследование, а когда композицию? Как реализовать множественное наследование?

Шаблоны

1. **Кейс: Создание шаблонного класса Stack.** Студент реализует стек с шаблоном для любого типа T. Пример: template<typename T> class Stack { ... };. Вопросы: Как добавить специализацию для типа int? Какие ограничения на шаблоны?

2. **Кейс: Шаблонная функция для поиска максимума.** Студент пишет функцию max с шаблоном, работающую с числами и строками. Пример: template<typename T> T max(T a, T b). Вопросы: Как обработать типы без оператора >? Что такое SFINAE?

3. **Кейс: Обобщенное программирование с контейнерами.** Студент создает шаблонную функцию для сортировки вектора. Пример: template<typename T> void sort(std::vector<T>& v). Вопросы: Как сделать функцию универсальной для любых контейнеров? Какие стандартные алгоритмы использовать?

4. **Кейс: Шаблоны с параметрами по умолчанию.** Студент пишет шаблон Pair с дефолтными типами. Пример: template<typename T1=int, typename T2=std::string>. Вопросы: Как комбинировать с variadic templates? Примеры из STL.

Полиморфизм и виртуальные функции

1. **Кейс: Виртуальные функции в иерархии животных.** Студент создает классы Animal, Dog, Cat с виртуальным методом speak(). Пример: virtual void speak() override. Вопросы: Как работает позднее связывание? Что такое pure virtual function?

2. **Кейс: Идиома CRTP для статического полиморфизма.** Студент использует Curiously Recurring Template Pattern для класса Base<T>. Пример: template<typename T> class Base { ... };. Вопросы: Преимущества CRTP над виртуальными функциями? Примеры использования.

3. **Кейс: Виртуальные деструкторы в фабричном методе.** Студент реализует фабрику для создания объектов Shape. Пример: std::unique_ptr<Shape> createShape(). Вопросы: Почему важен виртуальный деструктор? Как избежать утечек памяти?

4. **Кейс: Современные идиомы C++ с override и final.** Студент использует override для методов и final для классов. Пример: class Derived final : public Base { void func() override; }. Вопросы: Что такое covariant return types? Примеры из C++11/14.

Исключения

1. **Кейс: Обработка исключений в файловых операциях.** Студент пишет функцию чтения файла с try-catch. Пример: try { std::ifstream file; } catch (const std::exception& e). Вопросы: Какие стандартные исключения использовать? Как создать собственное исключение?

2. **Кейс: RAII с исключениями в классе Database.** Студент управляет подключением к БД с noexcept. Пример: Класс с конструктором и деструктором. Вопросы: Что такое stack unwinding? Как комбинировать с smart pointers?

3. **Кейс: Параллельное программирование с std::thread и исключениями.** Студент запускает поток и обрабатывает исключения в нем. Пример: std::thread t(func); t.join();. Вопросы: Как передать исключение между потоками? Использование std::exception_ptr.

4. **Кейс: Обработка ошибок в асинхронном коде с future.** Студент использует std::async и ловит исключения из future. Пример: auto f = std::async(func); f.get();. Вопросы: Разница между std::exception и системными ошибками? Лучшие практики обработки исключений.

Примерные вопросы для коллоквиума

Вопросы по теме «Виртуальные методы. Позднее связывание»

1. Что такое виртуальный метод в C++ и как он объявляется?
2. Как работает механизм позднего связывания и зачем он нужен?
3. В чём разница между виртуальным и невиртуальным методом?
4. Как влияет виртуальный метод на размер и структуру объекта?
5. Какие проблемы могут возникнуть при использовании виртуальных методов?

Вопросы по теме «Обработка ошибок. Блоки try-catch»

1. Как устроена обработка исключений в C++?
2. Как объявляются блоки try, catch и что они делают?
3. Какие типы данных можно использовать для генерации исключений?
4. Как реализовать пользовательское исключение в C++?
5. Как обеспечить безопасное освобождение ресурсов при возникновении исключений?

Вопросы по теме «Вектор и список. Множества и словари»

1. Как устроена обработка исключений в C++?
2. Как объявляются блоки try, catch и что они делают?
3. Какие типы данных можно использовать для генерации исключений?
4. Как реализовать пользовательское исключение в C++?
5. Как обеспечить безопасное освобождение ресурсов при возникновении исключений?

Примерное описание и критерии оценивания к проекту

Описание проекта:

Итоговый проект представляет собой комплексную программную систему, разработанную на языке C++, демонстрирующую владение ключевыми концепциями и технологиями, изученными в ходе курса. Проект должен включать создание и организацию классов с применением принципов объектно-ориентированного программирования (ООП), использование наследования и полиморфизма через виртуальные функции и абстрактные классы, а также эффективное применение шаблонов для обобщённого программирования.

В реализации должны быть продемонстрированы навыки работы с инструментами

разработки: настройка и использование компилятора, сборщика, системы контроля версий, а также автоматизация сборки проекта. Особое внимание уделяется обработке ошибок с помощью исключений, обеспечению безопасности ресурсов и управлению памятью с использованием умных указателей.

Проект должен активно использовать контейнеры стандартной библиотеки (STL) и итераторы для организации и обработки данных, а также включать применение алгоритмов STL для решения практических задач.

Критерии оценивания:

1. **Архитектура и структура проекта**
   - Логическая организация кода, модульность и читаемость.
   - Корректное использование классов, инкапсуляции и модификаторов доступа.
   - Применение наследования и полиморфизма с правильной реализацией виртуальных функций и абстрактных классов.
2. **Использование шаблонов**
   - Реализация функциональных и классовых шаблонов.
   - Корректная параметризация типов и применение специализации шаблонов.
   - Обоснованное использование шаблонов для повышения переиспользуемости кода.
3. **Обработка ошибок и исключения**
   - Наличие и корректность блоков try-catch.
   - Генерация и обработка пользовательских исключений.
   - Обеспечение безопасности ресурсов при возникновении исключений (RAII-подход).
4. **Работа с контейнерами STL и алгоритмами**
   - Использование различных контейнеров (vector, list, set, map) по назначению.
   - Применение итераторов для обхода и модификации данных.
   - Использование стандартных алгоритмов STL для решения задач.
5. **Управление памятью и умные указатели**
   - Корректное применение умных указателей (unique_ptr, shared_ptr, weak_ptr).
   - Отсутствие утечек памяти и ошибок владения ресурсами.
   - Обеспечение безопасного совместного использования объектов.
6. **Инструменты разработки и автоматизация**
   - Использование системы контроля версий с понятной историей изменений.
   - Наличие автоматизированной сборки проекта (Makefile, CMake или скрипты).
   - Использование отладчика и профилировщика для тестирования и оптимизации (при возможности).
7. **Качество кода и документация**
   - Читаемость, стиль кодирования и комментарии.
   - Наличие описания архитектуры и инструкций по сборке и запуску проекта.
   - Тестовые примеры и демонстрация работы ключевых функциональностей.
8. **Функциональность и соответствие требованиям**
   - Полнота реализации заявленных функциональных возможностей.
   - Корректность работы программы без критических ошибок и сбоев.
   - Эффективность и оптимальность решений.

Задания для промежуточной аттестации по дисциплине (модулю)

№ п/п	Задание	Ответ	Компетенция
1	Укажите инструмент для компиляции кода на C++.	g++ /GCC, компилятор GCC	УК-1
2	Укажите принцип наследования в объектно-ориентированном программировании.	Полиморфизм / polymorphism,	УК-1

		наследование свойств	
3	Укажите метод анализа эффективности алгоритма на C++.	Сложность / time complexity, вычислительная сложность	УК-1
4	Укажите математическую концепцию для работы с указателями в C++.	Адресация / addressing, работа с памятью	ОПК-1
5	Укажите применение дифференциальных уравнений в моделировании на C++.	Симуляция / simulation, численное интегрирование	ОПК-1
6	Укажите практический опыт создания математической модели в программе.	Реализация / implementation, код модели	ОПК-1
7	Укажите базовый математический аппарат для сортировки массивов.	Алгоритм / sorting algorithm, сортировочный алгоритм	ОПК-4
8	Укажите умение применять рекурсию в программировании.	Функция / recursive function, рекурсивная функция	ОПК-4
9	Укажите опыт реализации структуры данных на C++.	Вектор / vector, контейнер vector	ОПК-4
10	Укажите опыт оценки качества программы с помощью тестов.	Юнит-тест / unit test, модульное тестирование	ОПК-4
11	Укажите алгоритм разработки программы на C++.	Шаги / steps, последовательность шагов	ОПК-6
12	Укажите умение писать классы с наследованием.	Класс / class, определение класса	ОПК-6
13	Укажите умение использовать шаблоны для обобщения.	Функтор / functor, шаблонная функция	ОПК-6
14	Укажите опыт создания виртуальных функций.	Полиморфизм / polymorphism, виртуальный метод	ОПК-6
15	Укажите опыт обработки исключений в коде.	Try-catch / try catch, блок try	ОПК-6
16	Укажите метод математического моделирования в C++.	Итерация / iteration, циклический процесс	ПК-3
17	Укажите умение применять алгоритмы к контейнерам.	Сортировка / sort, алгоритм sort	ПК-3
18	Укажите умение моделировать параллельные процессы.	Потоки / threads, многопоточность	ПК-3
19	Укажите опыт использования итераторов в проекте.	Цикл / loop, обход элементов	ПК-3
20	Укажите опыт оптимизации производительности кода.	Профилирование / profiling, анализ производительности	ПК-3