ОБЩЕСТВО С ОГРАНИЧЕННОЙ ОТВЕТСТВЕННОСТЬЮ «ИННОПРОГ»

УТВЕРЖДАЮ
Генеральный директор ООО «ИННОПРОГ»
_____ / Венедиктов Р.В.
подпись инициалы, фамилия
01 февраля 2025 г.

Приказ об утверждении программы
от 01 февраля 2025 г. № ОБР-3

ДОПОЛНИТЕЛЬНАЯ ПРОФЕССИОНАЛЬНАЯ ПРОГРАММА ПРОФЕССИОНАЛЬНОЙ ПЕРЕПОДГОТОВКИ

«C++ разработчик»

Вид программы:	дополнительная профессиональная программа профессиональной переподготовки
Новая квалификация:	C++ разработчик
Трудоемкость:	800 академических часов
Форма обучения:	очная, с применением электронного обучения и дистанционных образовательных технологий
Режим реализации:	10 месяцев (40 учебных недель), рекомендуемая нагрузка – 20 академических часов в неделю
Язык обучения:	русский

Иннополис, 2025

Содержание

1. Общая характеристика программы

1.1. Наименование программы

Дополнительная профессиональная программа профессиональной переподготовки «C++ разработчик» разработана для реализации ООО «Иннопрог» в рамках дополнительного профессионального образования. Программа направлена на получение компетенции, необходимой для выполнения нового вида профессиональной деятельности в сфере разработки программного обеспечения на C++, и приобретение новой квалификации: **C++ разработчик**.

1.2. Нормативно-правовые основания разработки программы

Таблица 1 – Нормативно-правовые основания разработки программы

№	Документ	Значение для программы
1	Федеральный закон от 29.12.2012 № 273-ФЗ «Об образовании в Российской Федерации»	определяет правовые основы реализации дополнительных профессиональных программ, требования к категории слушателей и документам о квалификации
2	Приказ Минобрнауки России от 24.03.2025 № 266	используется как основной порядок организации и осуществления образовательной деятельности по дополнительным профессиональным программам
3	Постановление Правительства РФ от 11.10.2023 № 1678	учитывается при применении электронного обучения и дистанционных образовательных технологий
4	Постановление Правительства РФ от 18.09.2020 № 1490	учитывается в части лицензионных требований к образовательной деятельности
5	ГОСТ Р 7.0.97-2025	используется как ориентир для оформления реквизитов организационно-распорядительного документа
6	Локальные нормативные акты ООО «Иннопрог»	регулируют прием, зачисление, аттестацию, пересдачи, апелляции, выдачу документов и работу в ЭИОС

1.3. Структура образовательной программы

Настоящая Программа «C++ разработчик» оформлена как комплекс основных характеристик образования, организационно-педагогических условий, оценочных и методических материалов, необходимых для реализации дополнительной профессиональной программы профессиональной переподготовки.

В состав Программы включены:

- общая характеристика Программы, включая цель, задачи, категорию слушателей, требования к поступающим, форму обучения, срок освоения, трудоемкость, документ о квалификации и характеристику новой квалификации;
- учебный план, определяющий перечень модулей, трудоемкость, распределение часов по видам учебной работы, формы промежуточной и итоговой аттестации;
- календарный учебный график, определяющий рекомендуемую последовательность освоения модулей и период итоговой аттестации;
- рабочие программы модулей, раскрывающие цели, результаты, тематическое содержание, самостоятельную работу, проектные задания и формы контроля;
- организационно-педагогические условия реализации, включая кадровые, материально-технические, информационно-методические условия, электронную информационно-образовательную среду и порядок применения электронного обучения и дистанционных образовательных технологий;
- оценочные материалы для текущего контроля, промежуточной и итоговой аттестации, критерии допуска, пересдачи, апелляции и подтверждения самостоятельности выполнения работ;
- методические материалы и приложения, обеспечивающие единообразное применение Программы, проверку проектных работ, оформление результатов и актуализацию документа.

Структура Программы обеспечивает проверяемость объема, содержания, планируемых результатов, условий реализации, форм аттестации и документов, выдаваемых по итогам обучения.

1.4. Вид программы и целевая модель результата

Программа готовит слушателя к выполнению типовых задач junior C++ разработчика: анализ требований к задаче, выбор алгоритма и структуры данных, разработка консольных и прикладных программ, проектирование классов, работа с файлами и базами данных, настройка сборки, отладка, тестирование, оформление репозитория и защита результата. В центре программы находится практический результат: каждый крупный модуль завершается проверяемым артефактом, который входит в портфолио слушателя и используется при подготовке выпускного проекта.

1.5. Актуальность программы

Актуальность программы определяется устойчивым спросом на специалистов, способных разрабатывать производительные и надежные программные компоненты на C++. Язык применяется в системном программировании, игровых движках, финансовых расчетах, высоконагруженных сервисах, встраиваемых решениях, инструментах обработки данных и прикладных программах, где важны контроль ресурсов, скорость выполнения, надежность и качество архитектуры. Работодатели ожидают от начинающего разработчика знания синтаксиса C++, понимания памяти и времени жизни объектов, навыков применения STL, ООП, Git, CMake, Linux, отладки, тестирования и аккуратного оформления проекта.

1.6. Цель реализации программы

Цель реализации программы – формирование у слушателя комплекса профессиональных компетенций, необходимых для разработки, отладки, тестирования, документирования и защиты программных решений на C++, включая применение алгоритмов и структур данных, стандартной библиотеки, объектно-ориентированного проектирования, инструментов сборки, баз данных, Linux-среды и методов контроля качества.

1.7. Задачи программы

Для достижения цели программа решает следующие задачи:

– сформировать понимание роли C++ разработчика, жизненного цикла программного продукта и требований к профессиональному портфолио;
– научить писать базовые C++ программы, использовать типы данных, операторы, функции, массивы, строки, файлы и простые структуры данных;
– сформировать навыки алгоритмического мышления, оценки сложности и выбора подходящих структур данных;
– научить применять стандартную библиотеку C++, контейнеры, алгоритмы, итераторы, шаблоны, исключения, RAII и умные указатели;
– сформировать навыки объектно-ориентированного проектирования, выделения классов, интерфейсов, связей между сущностями и слоев приложения;
– научить вести Git-репозиторий, настраивать CMake, собирать проект, анализировать предупреждения компилятора и отлаживать ошибки;
– сформировать навыки работы с SQL, PostgreSQL, файлами, структурированными данными и подключением C++ приложения к хранилищу;
– научить работать в Linux-среде, применять терминал, Bash-скрипты, логи, переменные окружения, зависимости и инструкции по запуску;
– сформировать навыки тестирования, code review, статического анализа, применения санитайзеров и устранения дефектов памяти;
– обеспечить подготовку выпускного проекта, который подтверждает готовность слушателя к решению прикладных задач C++ разработки.

1.8. Категория слушателей

К освоению программы допускаются лица, имеющие среднее профессиональное и (или) высшее образование, а также лица, получающие среднее профессиональное и (или) высшее образование. Рекомендуется уверенное владение персональным компьютером, базовая цифровая грамотность, готовность работать с текстовым редактором, терминалом, учебной платформой и системой контроля версий. Предварительный опыт программирования полезен, но базовые темы раскрываются в программе с начального уровня при условии регулярной самостоятельной работы.

1.9. Требования к уровню подготовки поступающего на обучение

Обязательным условием допуска к освоению Программы является наличие среднего профессионального и (или) высшего образования либо получение среднего профессионального и (или) высшего образования на момент освоения Программы.

Рекомендуемый входной уровень подготовки:

- уверенное владение персональным компьютером и современными интернет-сервисами;
- базовая цифровая грамотность;
- готовность к регулярной самостоятельной работе, выполнению практических заданий и ведению учебного проекта;
- готовность работать с электронной информационно-образовательной средой, материалами курса, репозиторием, консультациями и обратной связью преподавателя.

1.10. Порядок приема, зачисления, отчисления, пересдачи и апелляции

Прием на обучение осуществляется на основании заявления слушателя, документа, удостоверяющего личность, документа о среднем профессиональном и (или) высшем образовании либо справки об обучении для лица, получающего соответствующее образование, а также договора об образовании на обучение по дополнительной профессиональной программе. Перечень документов, порядок их представления, сроки рассмотрения заявления и основания отказа в приеме определяются локальными нормативными актами организации и договором об образовании.

Зачисление слушателя оформляется приказом организации после заключения договора, проверки входных документов и предоставления доступа к электронной информационно-образовательной среде. Отчисление, восстановление, перевод на индивидуальный график, изменение сроков обучения и прекращение образовательных отношений осуществляются в порядке, установленном локальными нормативными актами организации и договором об образовании.

Порядок пересдачи текущего контроля, промежуточной аттестации и итоговой аттестации, сроки устранения замечаний, основания допуска к повторной

сдаче и порядок подачи апелляции определяются локальным актом о текущем контроле, промежуточной и итоговой аттестации. Слушателю обеспечивается возможность получить мотивированную обратную связь по результатам проверки и представить доработанную работу в установленные сроки.

1.11. Форма обучения, режим и язык обучения

Обучение по Программе осуществляется в очной форме с применением электронного обучения и дистанционных образовательных технологий. Электронное обучение и дистанционные образовательные технологии применяются при проведении учебных занятий, консультаций, самостоятельной работы, текущего контроля, промежуточной аттестации, проектной практики и итоговой аттестации при условии обеспечения доступа слушателя к электронной информационно-образовательной среде, учебным материалам, заданиям, средствам коммуникации и процедурам идентификации слушателя.

Язык обучения – русский. Для всех видов учебной работы академический час устанавливается продолжительностью 45 минут. Образовательный процесс может осуществляться в течение всего календарного года. Конкретные даты начала и окончания обучения определяются приказом о зачислении, договором об образовании, расписанием и календарным учебным графиком.

Рекомендуемый режим освоения Программы – 20 академических часов в неделю в течение 10 месяцев (40 учебных недель). Допускается обучение по индивидуальному учебному плану в порядке, установленном локальным нормативным актом организации, при условии сохранения общей трудоемкости Программы и достижения планируемых результатов.

1.12. Срок освоения и трудоемкость программы

Трудоемкость Программы составляет 800 академических часов, включая все виды учебной работы слушателя, текущий контроль, промежуточную аттестацию, проектную практику и итоговую аттестацию.

Рекомендуемый срок освоения Программы составляет 10 месяцев (40 учебных недель). Допускается реализация Программы в ином календарном периоде при условии сохранения общей трудоемкости, выполнения учебного пла-

на, прохождения предусмотренных форм контроля и достижения заявленных планируемых результатов.

1.13. Документ о квалификации

Лицам, успешно освоившим программу и прошедшим итоговую аттестацию, выдается диплом о профессиональной переподготовке установленного ООО «Иннопрог» образца с присвоением квалификации «C++ разработчик». Лицам, освоившим часть программы или не прошедшим итоговую аттестацию, выдается справка об обучении или о периоде обучения в порядке, установленном локальными нормативными актами организации. Сведения о выданных документах о квалификации передаются в федеральную информационную систему ФИС ФРДО в порядке и сроки, установленные законодательством и локальными регламентами организации.

1.14. Характеристика новой квалификации и связанной с ней профессиональной деятельности

Выпускник программы может выполнять типовые задачи начинающего C++ разработчика под руководством более опытного специалиста или в составе учебной проектной команды. К таким задачам относятся разработка функций и классов, реализация алгоритмов, работа с файлами и базой данных, написание тестов, исправление дефектов, настройка сборки, подготовка инструкции по запуску и участие в демонстрации программного результата заказчику или аттестационной комиссии.

Таблица 2 – Формируемые профессиональные компетенции

Код	**Профессиональная компетенция**	**Индикаторы достижения**
ПК-1	Анализировать требования к программному продукту	выделяет цель, пользователей, сценарии, входные данные, ограничения, критерии приемки и риски реализации
ПК-2	Проектировать алгоритмы, структуры данных и архитектуру решения	выбирает алгоритмы, структуры данных, слои приложения, модель данных и зависимости с учетом сложности и сопровождения

Продолжение таблицы 2

Продолжение таблицы 2

Код	Профессиональная компетенция	Индикаторы достижения
ПК-3	Разрабатывать C++ код, соответствующий поставленной задаче	пишет читаемый код, использует функции, классы, стандартную библиотеку, обрабатывает ошибки и соблюдает стиль
ПК-4	Применять объектно-ориентированное проектирование и современные возможности C++	использует инкапсуляцию, композицию, полиморфизм, RAII, шаблоны, smart pointers и правила управления ресурсами
ПК-5	Работать с файлами, внешними данными, SQL и PostgreSQL	проектирует простую схему, пишет SQL-запросы, подключает приложение к базе, проверяет корректность данных
ПК-6	Использовать инструменты командной разработки и сборки	ведет Git-репозиторий, работает с ветками, настраивает CMake, оформляет README и обеспечивает воспроизводимую сборку
ПК-7	Запускать, диагностировать и сопровождать C++ приложения в Linux-среде	работает в терминале, управляет окружением, логами, зависимостями, скриптами запуска и базовой эксплуатацией
ПК-8	Тестировать, отлаживать и повышать качество C++ приложения	пишет тесты, применяет отладчик, санитайзеры, статический анализ, устраняет дефекты и проверяет граничные случаи
ПК-9	Документировать проект и организовывать профессиональное портфолио	оформляет структуру проекта, инструкции, архитектурное описание, отчет о тестировании и историю изменений
ПК-10	Представлять и защищать результат разработки	демонстрирует работоспособность, объясняет технические решения, ограничения, качество и отвечает на вопросы комиссии

1.15. Связь Программы с профессиональными стандартами, трудовыми функциями и уровнями квалификации

Содержание программы сформировано с учетом профессиональных стандартов и практических требований работодателей к C++ разработчику. Профессиональные стандарты используются как ориентир для описания трудовых действий, планируемых результатов, компетенций, оценочных материалов и требований к выпускному проекту.

Таблица 3 – Связь программы с профессиональными стандартами и квалификационными ориентирами

№	Ориентир	Учитываемые виды деятельности
1	Профессиональный стандарт «Программист», приказ Минтруда России от 20.07.2022 № 424н	разработка, отладка, проверка работоспособности, модификация программного обеспечения, участие в создании программных компонентов
2	Профессиональный стандарт «Системный программист», приказ Минтруда России от 29.09.2020 № 678н	разработка системных и низкоуровневых программных компонентов, работа с окружением, сборкой, производительностью и надежностью
3	Квалификационные требования работодателей к junior C++ разработчику	владение C++, STL, ООП, Git, CMake, Linux, отладкой, тестированием, базами данных и документацией проекта

1.16. Планируемые результаты освоения программы

По итогам освоения Программы слушатель должен быть готов к выполнению нового вида профессиональной деятельности по направлению «C++ разработчик» и к применению квалификации «C++ разработчик» в типовых учебных, проектных и прикладных задачах.

В результате освоения Программы слушатель должен:

знать:

– основные понятия, инструменты, технологии и методы, используемые в профессиональной области Программы;

– требования к качеству, структуре, документированию и проверке результата;

– назначение учебных модулей, проектной практики, промежуточной и итоговой аттестации;
– базовые требования к безопасной работе с данными, цифровыми сервисами, учетными записями и проектными материалами.

уметь:

– анализировать задачу, выбирать инструменты и планировать последовательность выполнения работы;
– выполнять практические задания и проектные работы в соответствии с критериями оценивания;
– оформлять результаты, готовить репозиторий, отчет, демонстрацию и материалы для проверки;
– исправлять замечания, проходить промежуточную аттестацию и защищать итоговый проект.

владеть навыками:

– самостоятельной работы с учебными материалами, электронными ресурсами и обратной связью преподавателя;
– разработки, анализа, тестирования, документирования или сопровождения проектного результата по направлению Программы;
– подготовки итогового аттестационного проекта и аргументированной защиты принятых решений.

Сквозная проектная логика программы. Программа построена по накопительной модели. Сначала слушатель осваивает базовый синтаксис C++ и способы решения небольших задач. Затем он переходит к алгоритмам, стандартной библиотеке, объектно-ориентированному проектированию и структуре проекта. После этого слушатель осваивает инструменты профессиональной разработки: Git, CMake, отладку, Linux, SQL, PostgreSQL, тестирование и анализ качества. Завершающая часть посвящена проектной практике и защите комплексного C++ приложения.

Требования к портфолио слушателя. В ходе обучения слушатель формирует портфолио C++ разработчика. Портфолио подтверждает освоение компетенций и используется при итоговой аттестации. Минимальный состав портфолио:

– единый репозиторий или набор репозиториев с выполненными заданиями;
– файл README с описанием структуры, сборки, запуска и проверки работ;
– не менее десяти программ на базовые конструкции C++;
– не менее пяти решений по алгоритмам и структурам данных;
– не менее одного объектно-ориентированного приложения;
– не менее одного проекта с CMake и модульной структурой;
– не менее одного задания с базой данных или структурированным хранением данных;
– набор тестов и отчет о проверке качества для итогового проекта;
– выпускной проект с пояснительной запиской, инструкцией запуска и презентацией.

2. Содержание программы

2.1. Учебный план

Таблица 4 – Учебный план

№	Наименование модуля	Всего ч.	Лек. ч.	Прак. ч.	Сам. раб.	ПА ч.	ИА ч.	Форма контроля
1	Вводный организационно-методический модуль	12	6	3	3	0	0	входной контроль
2	Основы программирования на C++	136	40	62	28	6	0	зачет
3	Алгоритмы и структуры данных на C++	104	29	46	23	6	0	зачет
4	Современный C++ и стандартная библиотека	104	29	46	23	6	0	зачет
5	Объектно-ориентированное программирование и архитектура C++ приложений	116	32	55	23	6	0	зачет
6	Инструменты разработки, Git, CMake и отладка	80	20	37	17	6	0	зачет
7	SQL и PostgreSQL для C++ разработчика	68	17	31	14	6	0	зачет

Продолжение таблицы 4

Продолжение таблицы 4

№	Наименование модуля	Всего ч.	Лек. ч.	Прак. ч.	Сам. раб.	ПА ч.	ИА ч.	Форма контроля
8	Linux, системное окружение и эксплуатация C++ приложений	56	14	25	11	6	0	зачет
9	Тестирование, качество и безопасная разработка на C++	56	14	25	11	6	0	зачет
10	Проектная практика	36	0	27	9	0	0	проектная работа
11	Итоговая аттестация	32	0	0	0	0	32	защита выпускного проекта
	Итого	**800**	**201**	**357**	**162**	**48**	**32**	

2.2. Календарный учебный график

Таблица 5 – Календарный учебный график

Учеб. неделя	Разделы и виды учебной работы	Часы	Формат и контроль
1	Вводный организационно-методический модуль – 12 ч.; Основы программирования на C++ – 8 ч.	20	зачет по модулю, практические задания
2	Основы программирования на C++ – 20 ч.	20	практические задания
3	Основы программирования на C++ – 20 ч.	20	практические задания
4	Основы программирования на C++ – 20 ч.	20	практические задания

Продолжение таблицы 5

Продолжение таблицы 5

Учеб. неделя	**Разделы и виды учебной работы**	**Часы**	**Формат и контроль**
5	Основы программирования на C++ – 20 ч.	20	практические задания
6	Основы программирования на C++ – 20 ч.	20	практические задания
7	Основы программирования на C++ – 20 ч.	20	практические задания
8	Основы программирования на C++ – 8 ч.; Алгоритмы и структуры данных на C++ – 12 ч.	20	зачет по модулю, практические задания
9	Алгоритмы и структуры данных на C++ – 20 ч.	20	практические задания
10	Алгоритмы и структуры данных на C++ – 20 ч.	20	практические задания
11	Алгоритмы и структуры данных на C++ – 20 ч.	20	практические задания
12	Алгоритмы и структуры данных на C++ – 20 ч.	20	практические задания
13	Алгоритмы и структуры данных на C++ – 12 ч.; Современный C++ и стандартная библиотека – 8 ч.	20	зачет по модулю, практические задания
14	Современный C++ и стандартная библиотека – 20 ч.	20	практические задания
15	Современный C++ и стандартная библиотека – 20 ч.	20	практические задания
16	Современный C++ и стандартная библиотека – 20 ч.	20	практические задания
17	Современный C++ и стандартная библиотека – 20 ч.	20	практические задания

Продолжение таблицы 5

Продолжение таблицы 5

Учеб. неделя	**Разделы и виды учебной работы**	**Часы**	**Формат и контроль**
18	Современный C++ и стандартная библиотека – 16 ч.; Объектно-ориентированное программирование и архитектура C++ приложений – 4 ч.	20	зачет по модулю, практические задания
19	Объектно-ориентированное программирование и архитектура C++ приложений – 20 ч.	20	практические задания
20	Объектно-ориентированное программирование и архитектура C++ приложений – 20 ч.	20	практические задания
21	Объектно-ориентированное программирование и архитектура C++ приложений – 20 ч.	20	практические задания
22	Объектно-ориентированное программирование и архитектура C++ приложений – 20 ч.	20	практические задания
23	Объектно-ориентированное программирование и архитектура C++ приложений – 20 ч.	20	практические задания
24	Объектно-ориентированное программирование и архитектура C++ приложений – 12 ч.; Инструменты разработки, Git, CMake и отладка – 8 ч.	20	зачет по модулю, практические задания
25	Инструменты разработки, Git, CMake и отладка – 20 ч.	20	практические задания
26	Инструменты разработки, Git, CMake и отладка – 20 ч.	20	практические задания
27	Инструменты разработки, Git, CMake и отладка – 20 ч.	20	практические задания

Продолжение таблицы 5

Продолжение таблицы 5

Учеб. неделя	**Разделы и виды учебной работы**	**Часы**	**Формат и контроль**
28	Инструменты разработки, Git, CMake и отладка – 12 ч.; SQL и PostgreSQL для C++ разработчика – 8 ч.	20	зачет по модулю, практические задания
29	SQL и PostgreSQL для C++ разработчика – 20 ч.	20	практические задания
30	SQL и PostgreSQL для C++ разработчика – 20 ч.	20	практические задания
31	SQL и PostgreSQL для C++ разработчика – 20 ч.	20	зачет по модулю, практические задания
32	Linux, системное окружение и эксплуатация C++ приложений – 20 ч.	20	практические задания
33	Linux, системное окружение и эксплуатация C++ приложений – 20 ч.	20	практические задания
34	Linux, системное окружение и эксплуатация C++ приложений – 16 ч.; Тестирование, качество и безопасная разработка на C++ – 4 ч.	20	зачет по модулю, практические задания
35	Тестирование, качество и безопасная разработка на C++ – 20 ч.	20	практические задания
36	Тестирование, качество и безопасная разработка на C++ – 20 ч.	20	практические задания
37	Тестирование, качество и безопасная разработка на C++ – 12 ч.; Проектная практика – 8 ч.	20	зачет по модулю, работа над проектом
38	Проектная практика – 20 ч.	20	зачет по модулю, работа над проектом
39	Проектная практика – 8 ч.; Итоговая аттестация – 12 ч.	20	защита выпускного проекта

Продолжение таблицы 5

Продолжение таблицы 5

Учеб. неделя	Разделы и виды учебной работы	Часы	Формат и контроль
40	Итоговая аттестация – 20 ч.	20	защита выпускного проекта

2.3. Рабочие программы модулей

Рабочие программы модулей раскрывают цель, содержание, практическую подготовку, самостоятельную работу, оценочные материалы и ожидаемый результат. Каждый содержательный модуль завершается конкретным учебным артефактом, который входит в портфолио слушателя и подтверждает освоение компетенций.

2.3.1. Модуль 1. Вводный организационно-методический модуль

Трудоемкость модуля – 12 академических часов. Цель модуля – сформировать понимание цели программы, роли C++ разработчика, требований к рабочей среде, портфолио, самостоятельной работе, промежуточной аттестации и выпускному проекту. Итоговый учебный артефакт модуля: индивидуальный учебный план, настроенная рабочая среда, структура репозитория и выбранная предварительная тема проектной работы. Модуль формирует компетенции: ПК-1, ПК-9, ПК-10. Содержание модуля связано с последующими разделами программы и используется при подготовке выпускного проекта.

Таблица 6 – Тематический план модуля 1 «Вводный организационно-методический модуль»

№	Тема	Всего ч.	Лек. ч.	Прак. ч.	Сам. раб.	Контроль
1	Роль C++ разработчика в разработке программного обеспечения	6	2	2	2	входной контроль
2	Структура программы, формы контроля и требования к портфолио	4	2	1	1	входной контроль

Продолжение таблицы 6

Продолжение таблицы 6

№	Тема	Всего ч.	Лек. ч.	Прак. ч.	Сам. раб.	Контроль
3	Рабочая среда: компилятор, редактор, терминал и Git	1	1	0	0	входной контроль
4	Правила оформления решений, академическая добросовестность и защита данных	1	1	0	0	входной контроль
5	Входная диагностика и индивидуальные рекомендации	0	0	0	0	входной контроль

Практическая подготовка. Слушатель настраивает рабочее место, проверяет компилятор, редактор, терминал, Git и доступ к электронной образовательной среде. Практический результат фиксируется в виде первого репозитория, тестовой программы и индивидуального плана обучения.

Самостоятельная работа. Самостоятельная работа включает изучение правил программы, подготовку рабочей папки, регистрацию в необходимых сервисах, заполнение профиля слушателя и выбор предварительной темы проекта.

Типовые ошибки. К типовым ошибкам относятся отсутствие настроенной среды, нерабочий компилятор, смешение учебных файлов без структуры, отсутствие README, нерегулярная фиксация выполненных работ и непонимание сроков прохождения аттестации.

2.3.2. Модуль 2. Основы программирования на C++

Трудоемкость модуля – 136 академических часов. Цель модуля – сформировать базовые навыки написания, запуска, чтения и отладки программ на C++, а также понимание типов данных, выражений, операторов, функций, массивов, строк и простых структур данных. Итоговый учебный артефакт модуля: набор консольных программ, решающих прикладные задачи с вводом, обработкой, проверкой данных и выводом результата. Модуль формирует компетенции: ПК-1, ПК-2, ПК-3, ПК-8. Содержание модуля связано с последующими разделами программы и используется при подготовке выпускного проекта.

Таблица 7 – Тематический план модуля 2 «Основы программирования на C++»

№	Тема	Всего ч.	Лек. ч.	Прак. ч.	Сам. раб.	Контроль
1	Структура программы C++, компиляция и запуск	15	5	7	3	практическое задание
2	Типы данных, переменные, литералы и область видимости	15	5	7	3	практическое задание
3	Операции, выражения, преобразования типов и приоритеты	14	4	7	3	практическое задание
4	Ввод, вывод, форматирование и обработка ошибок ввода	14	4	7	3	практическое задание
5	Условные операторы и ветвления бизнес-логики	13	4	6	3	практическое задание
6	Циклы, накопители, счетчики и обработка последовательностей	13	4	6	3	практическое задание
7	Массивы, векторы и базовые операции над коллекциями	13	4	6	3	практическое задание

Продолжение таблицы 7

Продолжение таблицы 7

№	Тема	Всего ч.	Лек. ч.	Прак. ч.	Сам. раб.	Контроль
8	Строки, символы, разбор текстовых данных	13	4	6	3	практическое задание
9	Функции, параметры, возвращаемые значения и декомпозиция	10	3	5	2	практическое задание
10	Простые структуры данных, файлы и итоговый мини-проект	10	3	5	2	практическое задание
11	Промежуточная аттестация по модулю	6	0	0	0	зачет

Практическая подготовка. Практическая подготовка строится вокруг решения прикладных задач на C++. Слушатель пишет код, запускает программу, проверяет граничные случаи, исправляет ошибки компиляции и выполнения, оформляет результат в репозитории и готовит краткое пояснение логики решения. В заданиях оцениваются работоспособность, читаемость, корректность алгоритма, структура проекта и способность объяснить принятое решение.

Самостоятельная работа. Самостоятельная работа включает повторение лекционного материала, выполнение тренировочных задач, оформление кода, подготовку README, анализ предупреждений компилятора, исправление дефектов, чтение документации и подготовку к зачету. Результаты самостоятельной работы фиксируются в электронной информационно-образовательной среде или в рабочем репозитории.

Минимальный уровень освоения. Слушатель выполняет обязательное задание по образцу, получает рабочую программу, обрабатывает основные входные данные, демонстрирует запуск и поясняет ключевые части решения. Допускаются отдельные недочеты в стиле кода, если они не искажают результат и могут быть исправлены после обратной связи.

Повышенный уровень освоения. Слушатель самостоятельно улучшает структуру решения, выделяет функции или классы, добавляет проверки входных данных, тесты, обработку ошибок, аккуратную инструкцию запуска и поясняет альтернативные варианты реализации.

Типовые ошибки. К типовым ошибкам относятся отсутствие декомпозиции, глобальные переменные без необходимости, дублирование кода, игнорирование ошибок ввода, выход за границы массива, неопределенное поведение, утечки ресурсов, отсутствие инструкции по сборке, неполные тесты и неаккуратная история коммитов.

Промежуточная аттестация. Промежуточная аттестация проводится в форме зачета по практическому заданию. Для зачета слушатель представляет исходный код, инструкцию сборки и запуска, краткое описание решения и демонстрацию работоспособности. Результат считается зачтенным, если задача понята, программа собирается, основные сценарии работают, ошибки обработаны на минимально допустимом уровне, код можно проверить и воспроизвести.

2.3.3. Модуль 3. Алгоритмы и структуры данных на C++

Трудоемкость модуля – 104 академических часов. Цель модуля – научить слушателя выбирать структуру данных и алгоритм под задачу, оценивать сложность, писать корректные решения и объяснять ограничения выбранного подхода. Итоговый учебный артефакт модуля: набор алгоритмических решений и мини-проект с применением выбранной структуры данных. Модуль формирует компетенции: ПК-1, ПК-2, ПК-3, ПК-8. Содержание модуля связано с последующими разделами программы и используется при подготовке выпускного проекта.

Таблица 8 – Тематический план модуля 3 «Алгоритмы и структуры данных на C++»

№	Тема	Всего ч.	Лек. ч.	Прак. ч.	Сам. раб.	Контроль
1	Асимптотическая сложность, память и время выполнения	14	5	6	3	практическое задание
2	Линейные структуры: массив, вектор, список, стек и очередь	13	4	6	3	практическое задание
3	Сортировки, поиск и типовые приемы обработки массивов	13	4	6	3	практическое задание
4	Хеш-таблицы, множества, словари и быстрый поиск	13	4	6	3	практическое задание
5	Деревья поиска, обходы и key-value хранилище	12	3	6	3	практическое задание
6	Графы, маршруты, очереди с приоритетом и базовые алгоритмы	12	3	6	3	практическое задание
7	Рекурсия, перебор, динамическое программирование в прикладных задачах	11	3	5	3	практическое задание
8	Алгоритмическое собеседование и разбор типовых ошибок	10	3	5	2	практическое задание
9	Промежуточная аттестация по модулю	6	0	0	0	зачет

Практическая подготовка. Практическая подготовка строится вокруг решения прикладных задач на C++. Слушатель пишет код, запускает программу, проверяет граничные случаи, исправляет ошибки компиляции и выполнения,

оформляет результат в репозитории и готовит краткое пояснение логики решения. В заданиях оцениваются работоспособность, читаемость, корректность алгоритма, структура проекта и способность объяснить принятое решение.

Самостоятельная работа. Самостоятельная работа включает повторение лекционного материала, выполнение тренировочных задач, оформление кода, подготовку README, анализ предупреждений компилятора, исправление дефектов, чтение документации и подготовку к зачету. Результаты самостоятельной работы фиксируются в электронной информационно-образовательной среде или в рабочем репозитории.

Минимальный уровень освоения. Слушатель выполняет обязательное задание по образцу, получает рабочую программу, обрабатывает основные входные данные, демонстрирует запуск и поясняет ключевые части решения. Допускаются отдельные недочеты в стиле кода, если они не искажают результат и могут быть исправлены после обратной связи.

Повышенный уровень освоения. Слушатель самостоятельно улучшает структуру решения, выделяет функции или классы, добавляет проверки входных данных, тесты, обработку ошибок, аккуратную инструкцию запуска и поясняет альтернативные варианты реализации.

Типовые ошибки. К типовым ошибкам относятся отсутствие декомпозиции, глобальные переменные без необходимости, дублирование кода, игнорирование ошибок ввода, выход за границы массива, неопределенное поведение, утечки ресурсов, отсутствие инструкции по сборке, неполные тесты и неаккуратная история коммитов.

Промежуточная аттестация. Промежуточная аттестация проводится в форме зачета по практическому заданию. Для зачета слушатель представляет исходный код, инструкцию сборки и запуска, краткое описание решения и демонстрацию работоспособности. Результат считается зачтенным, если задача понята, программа собирается, основные сценарии работают, ошибки обработаны на минимально допустимом уровне, код можно проверить и воспроизвести.

2.3.4. Модуль 4. Современный C++ и стандартная библиотека

Трудоемкость модуля – 104 академических часов. Цель модуля – сформировать навыки применения возможностей современных стандартов C++, стандартной библиотеки, RAII, умных указателей, шаблонов и безопасной работы с ресурсами. Итоговый учебный артефакт модуля: модульная C++ библиотека с использованием контейнеров, алгоритмов, шаблонов и тестовых сценариев. Модуль формирует компетенции: ПК-2, ПК-3, ПК-4, ПК-8. Содержание модуля связано с последующими разделами программы и используется при подготовке выпускного проекта.

Таблица 9 – Тематический план модуля 4 «Современный C++ и стандартная библиотека»

№	Тема	Всего ч.	Лек. ч.	Прак. ч.	Сам. раб.	Контроль
1	Модель компиляции, заголовочные и исходные файлы	14	5	6	3	практическое задание
2	Пространства имен, модульность и организация проекта	13	4	6	3	практическое задание
3	Контейнеры STL и выбор структуры хранения	13	4	6	3	практическое задание
4	Итераторы, алгоритмы, лямбда-выражения и функциональные приемы	13	4	6	3	практическое задание
5	RAII, время жизни объектов, ссылки, указатели и владение ресурсами	12	3	6	3	практическое задание
6	Умные указатели, перемещение, копирование и правило пяти	12	3	6	3	практическое задание
7	Шаблоны функций и классов, обобщенное программирование	11	3	5	3	практическое задание

Продолжение таблицы 9

Продолжение таблицы 9

№	Тема	Всего ч.	Лек. ч.	Прак. ч.	Сам. раб.	Контроль
8	Исключения, гарантии безопасности и обработка ошибок	10	3	5	2	практическое задание
9	Промежуточная аттестация по модулю	6	0	0	0	зачет

Практическая подготовка. Практическая подготовка строится вокруг решения прикладных задач на C++. Слушатель пишет код, запускает программу, проверяет граничные случаи, исправляет ошибки компиляции и выполнения, оформляет результат в репозитории и готовит краткое пояснение логики решения. В заданиях оцениваются работоспособность, читаемость, корректность алгоритма, структура проекта и способность объяснить принятое решение.

Самостоятельная работа. Самостоятельная работа включает повторение лекционного материала, выполнение тренировочных задач, оформление кода, подготовку README, анализ предупреждений компилятора, исправление дефектов, чтение документации и подготовку к зачету. Результаты самостоятельной работы фиксируются в электронной информационно-образовательной среде или в рабочем репозитории.

Минимальный уровень освоения. Слушатель выполняет обязательное задание по образцу, получает рабочую программу, обрабатывает основные входные данные, демонстрирует запуск и поясняет ключевые части решения. Допускаются отдельные недочеты в стиле кода, если они не искажают результат и могут быть исправлены после обратной связи.

Повышенный уровень освоения. Слушатель самостоятельно улучшает структуру решения, выделяет функции или классы, добавляет проверки входных данных, тесты, обработку ошибок, аккуратную инструкцию запуска и поясняет альтернативные варианты реализации.

Типовые ошибки. К типовым ошибкам относятся отсутствие декомпозиции, глобальные переменные без необходимости, дублирование кода, игнорирование ошибок ввода, выход за границы массива, неопределенное поведение, утечки ресурсов, отсутствие инструкции по сборке, неполные тесты и неаккуратная история коммитов.

Промежуточная аттестация. Промежуточная аттестация проводится в форме зачета по практическому заданию. Для зачета слушатель представляет исходный код, инструкцию сборки и запуска, краткое описание решения и демонстрацию работоспособности. Результат считается зачтенным, если задача понята, программа собирается, основные сценарии работают, ошибки обработаны на минимально допустимом уровне, код можно проверить и воспроизвести.

2.3.5. Модуль 5. Объектно-ориентированное программирование и архитектура C++ приложений

Трудоемкость модуля – 116 академических часов. Цель модуля – научить проектировать прикладные C++ приложения на основе классов, интерфейсов, инкапсуляции, композиции, наследования, полиморфизма и разделения ответственности. Итоговый учебный артефакт модуля: объектно-ориентированное приложение с несколькими сущностями, сценариями использования, слоями логики и документацией по архитектуре. Модуль формирует компетенции: ПК-1, ПК-2, ПК-4, ПК-8, ПК-9. Содержание модуля связано с последующими разделами программы и используется при подготовке выпускного проекта.

Таблица 10 – Тематический план модуля 5 «Объектно-ориентированное программирование и архитектура C++ приложений»

№	Тема	Всего ч.	Лек. ч.	Прак. ч.	Сам. раб.	Контроль
1	Классы, поля, методы, инкапсуляция и инварианты	16	5	8	3	практическое задание
2	Конструкторы, деструкторы, инициализация и управление состоянием	15	5	7	3	практическое задание

Продолжение таблицы 10

Продолжение таблицы 10

№	Тема	Всего ч.	Лек. ч.	Прак. ч.	Сам. раб.	Контроль
3	Композиция, агрегация и проектирование связей между объектами	14	4	7	3	практическое задание
4	Наследование, виртуальные функции и полиморфизм	14	4	7	3	практическое задание
5	Интерфейсы, абстрактные классы и подстановка реализаций	14	4	7	3	практическое задание
6	SOLID, разделение ответственности и простые паттерны проектирования	14	4	7	3	практическое задание
7	Архитектура консольного приложения, слои и зависимости	12	3	6	3	практическое задание
8	Рефакторинг, расширяемость и подготовка к проектной разработке	11	3	6	2	практическое задание
9	Промежуточная аттестация по модулю	6	0	0	0	зачет

Практическая подготовка. Практическая подготовка строится вокруг решения прикладных задач на C++. Слушатель пишет код, запускает программу, проверяет граничные случаи, исправляет ошибки компиляции и выполнения, оформляет результат в репозитории и готовит краткое пояснение логики решения. В заданиях оцениваются работоспособность, читаемость, корректность алгоритма, структура проекта и способность объяснить принятое решение.

Самостоятельная работа. Самостоятельная работа включает повторение лекционного материала, выполнение тренировочных задач, оформление кода, подготовку README, анализ предупреждений компилятора, исправление де-

фектов, чтение документации и подготовку к зачету. Результаты самостоятельной работы фиксируются в электронной информационно-образовательной среде или в рабочем репозитории.

Минимальный уровень освоения. Слушатель выполняет обязательное задание по образцу, получает рабочую программу, обрабатывает основные входные данные, демонстрирует запуск и поясняет ключевые части решения. Допускаются отдельные недочеты в стиле кода, если они не искажают результат и могут быть исправлены после обратной связи.

Повышенный уровень освоения. Слушатель самостоятельно улучшает структуру решения, выделяет функции или классы, добавляет проверки входных данных, тесты, обработку ошибок, аккуратную инструкцию запуска и поясняет альтернативные варианты реализации.

Типовые ошибки. К типовым ошибкам относятся отсутствие декомпозиции, глобальные переменные без необходимости, дублирование кода, игнорирование ошибок ввода, выход за границы массива, неопределенное поведение, утечки ресурсов, отсутствие инструкции по сборке, неполные тесты и неаккуратная история коммитов.

Промежуточная аттестация. Промежуточная аттестация проводится в форме зачета по практическому заданию. Для зачета слушатель представляет исходный код, инструкцию сборки и запуска, краткое описание решения и демонстрацию работоспособности. Результат считается зачтенным, если задача понята, программа собирается, основные сценарии работают, ошибки обработаны на минимально допустимом уровне, код можно проверить и воспроизвести.

2.3.6. Модуль 6. Инструменты разработки, Git, CMake и отладка

Трудоемкость модуля – 80 академических часов. Цель модуля – сформировать навыки профессиональной организации C++ проекта, сборки через CMake, командной работы в Git, диагностики ошибок компиляции и отладки выполнения программы. Итоговый учебный артефакт модуля: репозиторий C++ проекта с CMake, историей коммитов, инструкцией по сборке, отладочным сценарием и исправленными дефектами. Модуль формирует компетенции: ПК-5, ПК-6, ПК-8, ПК-9. Содержание модуля связано с последующими разделами программы и используется при подготовке выпускного проекта.

Таблица 11 – Тематический план модуля 6 «Инструменты разработки, Git, CMake и отладка»

№	Тема	Всего ч.	Лек. ч.	Прак. ч.	Сам. раб.	Контроль
1	Git: коммиты, ветки, слияния, конфликты и pull request	12	3	6	3	практическое задание
2	Структура C++ репозитория, README и соглашения о стиле	12	3	6	3	практическое задание
3	CMake: цели, библиотеки, исполняемые файлы и параметры сборки	11	3	5	3	практическое задание
4	Компиляторы, предупреждения, флаги сборки и профили Debug/Release	10	3	4	3	практическое задание
5	Отладка через IDE, gdb или lldb, точки останова и стек вызовов	9	3	4	2	практическое задание
6	Статический анализ, clang-format, clang-tidy и cppcheck	8	3	4	1	практическое задание
7	Санитайзеры, Valgrind и поиск ошибок памяти	6	1	4	1	практическое задание

Продолжение таблицы 11

Продолжение таблицы 11

№	Тема	Всего ч.	Лек. ч.	Прак. ч.	Сам. раб.	Контроль
8	Базовая автоматизация проверки и CI/CD для учебного проекта	6	1	4	1	практическое задание
9	Промежуточная аттестация по модулю	6	0	0	0	зачет

Практическая подготовка. Практическая подготовка строится вокруг решения прикладных задач на C++. Слушатель пишет код, запускает программу, проверяет граничные случаи, исправляет ошибки компиляции и выполнения, оформляет результат в репозитории и готовит краткое пояснение логики решения. В заданиях оцениваются работоспособность, читаемость, корректность алгоритма, структура проекта и способность объяснить принятое решение.

Самостоятельная работа. Самостоятельная работа включает повторение лекционного материала, выполнение тренировочных задач, оформление кода, подготовку README, анализ предупреждений компилятора, исправление дефектов, чтение документации и подготовку к зачету. Результаты самостоятельной работы фиксируются в электронной информационно-образовательной среде или в рабочем репозитории.

Минимальный уровень освоения. Слушатель выполняет обязательное задание по образцу, получает рабочую программу, обрабатывает основные входные данные, демонстрирует запуск и поясняет ключевые части решения. Допускаются отдельные недочеты в стиле кода, если они не искажают результат и могут быть исправлены после обратной связи.

Повышенный уровень освоения. Слушатель самостоятельно улучшает структуру решения, выделяет функции или классы, добавляет проверки входных данных, тесты, обработку ошибок, аккуратную инструкцию запуска и поясняет альтернативные варианты реализации.

Типовые ошибки. К типовым ошибкам относятся отсутствие декомпозиции, глобальные переменные без необходимости, дублирование кода, игнорирование ошибок ввода, выход за границы массива, неопределенное поведение, утечки ресурсов, отсутствие инструкции по сборке, неполные тесты и неаккуратная история коммитов.

Промежуточная аттестация. Промежуточная аттестация проводится в форме зачета по практическому заданию. Для зачета слушатель представляет исходный код, инструкцию сборки и запуска, краткое описание решения и демонстрацию работоспособности. Результат считается зачтенным, если задача понята, программа собирается, основные сценарии работают, ошибки обработаны на минимально допустимом уровне, код можно проверить и воспроизвести.

2.3.7. Модуль 7. SQL и PostgreSQL для C++ разработчика

Трудоемкость модуля – 68 академических часов. Цель модуля – научить проектировать простую реляционную модель, писать SQL-запросы, работать с PostgreSQL и подключать C++ приложение к базе данных с учетом безопасности и воспроизводимости. Итоговый учебный артефакт модуля: C++ приложение с хранением данных в PostgreSQL или учебной базе, SQL-скриптами, миграциями и инструкцией запуска. Модуль формирует компетенции: ПК-1, ПК-2, ПК-5, ПК-8. Содержание модуля связано с последующими разделами программы и используется при подготовке выпускного проекта.

Таблица 12 – Тематический план модуля 7 «SQL и PostgreSQL для C++ разработчика»

№	Тема	Всего ч.	Лек. ч.	Прак. ч.	Сам. раб.	Контроль
1	Реляционная модель, таблицы, ключи, связи и ограничения	11	3	5	3	практическое задание
2	DDL, DML, SELECT, фильтрация и сортировка	10	3	4	3	практическое задание
3	JOIN, агрегация, подзапросы и CTE	9	3	4	2	практическое задание

Продолжение таблицы 12

Продолжение таблицы 12

№	Тема	Всего ч.	Лек. ч.	Прак. ч.	Сам. раб.	Контроль
4	Нормализация, справочники и проектирование схемы	9	3	4	2	практическое задание
5	Транзакции, уровни изоляции и целостность данных	7	2	4	1	практическое задание
6	Индексы, план выполнения и базовая оптимизация	6	1	4	1	практическое задание
7	Подключение C++ приложения к PostgreSQL, параметры подключения и обработка ошибок	5	1	3	1	практическое задание
8	Безопасная работа с данными, валидация и защита от SQL-инъекций	6	1	3	1	практическое задание
9	Промежуточная аттестация по модулю	5	0	0	0	зачет

Практическая подготовка. Практическая подготовка строится вокруг решения прикладных задач на C++. Слушатель пишет код, запускает программу, проверяет граничные случаи, исправляет ошибки компиляции и выполнения, оформляет результат в репозитории и готовит краткое пояснение логики решения. В заданиях оцениваются работоспособность, читаемость, корректность алгоритма, структура проекта и способность объяснить принятое решение.

Самостоятельная работа. Самостоятельная работа включает повторение лекционного материала, выполнение тренировочных задач, оформление кода, подготовку README, анализ предупреждений компилятора, исправление дефектов, чтение документации и подготовку к зачету. Результаты самостоятель-

ной работы фиксируются в электронной информационно-образовательной среде или в рабочем репозитории.

Минимальный уровень освоения. Слушатель выполняет обязательное задание по образцу, получает рабочую программу, обрабатывает основные входные данные, демонстрирует запуск и поясняет ключевые части решения. Допускаются отдельные недочеты в стиле кода, если они не искажают результат и могут быть исправлены после обратной связи.

Повышенный уровень освоения. Слушатель самостоятельно улучшает структуру решения, выделяет функции или классы, добавляет проверки входных данных, тесты, обработку ошибок, аккуратную инструкцию запуска и поясняет альтернативные варианты реализации.

Типовые ошибки. К типовым ошибкам относятся отсутствие декомпозиции, глобальные переменные без необходимости, дублирование кода, игнорирование ошибок ввода, выход за границы массива, неопределенное поведение, утечки ресурсов, отсутствие инструкции по сборке, неполные тесты и неаккуратная история коммитов.

Промежуточная аттестация. Промежуточная аттестация проводится в форме зачета по практическому заданию. Для зачета слушатель представляет исходный код, инструкцию сборки и запуска, краткое описание решения и демонстрацию работоспособности. Результат считается зачтенным, если задача понята, программа собирается, основные сценарии работают, ошибки обработаны на минимально допустимом уровне, код можно проверить и воспроизвести.

2.3.8. Модуль 8. Linux, системное окружение и эксплуатация C++ приложений

Трудоемкость модуля – 56 академических часов. Цель модуля – сформировать практические навыки работы в Linux-среде, сборки, запуска, диагностики и базовой эксплуатации C++ приложений. Итоговый учебный артефакт модуля: собранное и запущенное C++ приложение в Linux-среде с инструкцией, логами, конфигурацией и сценарием проверки. Модуль формирует компетенции: ПК-6, ПК-7, ПК-8, ПК-9. Содержание модуля связано с последующими разделами программы и используется при подготовке выпускного проекта.

Таблица 13 – Тематический план модуля 8 «Linux, системное окружение и эксплуатация C++ приложений»

№	Тема	Всего ч.	Лек. ч.	Прак. ч.	Сам. раб.	Контроль
1	Файловая система Linux, права доступа и работа в терминале	9	3	4	2	практическое задание
2	Процессы, переменные окружения, потоки ввода-вывода и коды завершения	9	3	4	2	практическое задание
3	Bash-скрипты для сборки, запуска и проверки результата	7	2	3	2	практическое задание
4	Логи, конфигурационные файлы и диагностика проблем запуска	7	2	3	1	практическое задание
5	Пакеты, зависимости, системные библиотеки и контейнеризация	5	1	3	1	практическое задание
6	Сборка в Linux, запуск сервиса и базовые unit-файлы systemd	5	1	3	1	практическое задание

Продолжение таблицы 13

Продолжение таблицы 13

№	Тема	Всего ч.	Лек. ч.	Прак. ч.	Сам. раб.	Контроль
7	Сетевые настройки, порты и безопасность окружения	5	1	3	1	практическое задание
8	Подготовка инструкции по развертыванию и сопровождению	4	1	2	1	практическое задание
9	Промежуточная аттестация по модулю	5	0	0	0	зачет

Практическая подготовка. Практическая подготовка строится вокруг решения прикладных задач на C++. Слушатель пишет код, запускает программу, проверяет граничные случаи, исправляет ошибки компиляции и выполнения, оформляет результат в репозитории и готовит краткое пояснение логики решения. В заданиях оцениваются работоспособность, читаемость, корректность алгоритма, структура проекта и способность объяснить принятое решение.

Самостоятельная работа. Самостоятельная работа включает повторение лекционного материала, выполнение тренировочных задач, оформление кода, подготовку README, анализ предупреждений компилятора, исправление дефектов, чтение документации и подготовку к зачету. Результаты самостоятельной работы фиксируются в электронной информационно-образовательной среде или в рабочем репозитории.

Минимальный уровень освоения. Слушатель выполняет обязательное задание по образцу, получает рабочую программу, обрабатывает основные входные данные, демонстрирует запуск и поясняет ключевые части решения. Допускаются отдельные недочеты в стиле кода, если они не искажают результат и могут быть исправлены после обратной связи.

Повышенный уровень освоения. Слушатель самостоятельно улучшает структуру решения, выделяет функции или классы, добавляет проверки входных данных, тесты, обработку ошибок, аккуратную инструкцию запуска и поясняет альтернативные варианты реализации.

Типовые ошибки. К типовым ошибкам относятся отсутствие декомпозиции, глобальные переменные без необходимости, дублирование кода, игнорирование ошибок ввода, выход за границы массива, неопределенное поведение, утечки ресурсов, отсутствие инструкции по сборке, неполные тесты и неаккуратная история коммитов.

Промежуточная аттестация. Промежуточная аттестация проводится в форме зачета по практическому заданию. Для зачета слушатель представляет исходный код, инструкцию сборки и запуска, краткое описание решения и демонстрацию работоспособности. Результат считается зачтенным, если задача понята, программа собирается, основные сценарии работают, ошибки обработаны на минимально допустимом уровне, код можно проверить и воспроизвести.

2.3.9. Модуль 9. Тестирование, качество и безопасная разработка на C++

Трудоемкость модуля – 56 академических часов. Цель модуля – научить проверять корректность C++ кода, проектировать тесты, анализировать дефекты, учитывать риски неопределенного поведения, ошибок памяти и некорректной обработки данных. Итоговый учебный артефакт модуля: набор unit-тестов, чек-лист качества, отчет о найденных дефектах и доработанная версия учебного проекта. Модуль формирует компетенции: ПК-3, ПК-5, ПК-8, ПК-9. Содержание модуля связано с последующими разделами программы и используется при подготовке выпускного проекта.

Таблица 14 – Тематический план модуля 9 «Тестирование, качество и безопасная разработка на C++»

№	Тема	Всего ч.	Лек. ч.	Прак. ч.	Сам. раб.	Контроль
1	Виды тестирования, тест-кейсы и критерии приемки	9	3	4	2	практическое задание
2	Unit-тестирование C++ через GoogleTest, Catch2 или doctest	9	3	4	2	практическое задание
3	Тестирование функций, классов, ошибок ввода и граничных случаев	7	2	3	2	практическое задание
4	Интеграционные проверки и тестирование работы с файлами или базой данных	6	2	3	1	практическое задание
5	Покрытие, регрессионная проверка и оформление отчета о дефектах	5	1	3	1	практическое задание
6	Безопасность C++: переполнение, выход за границы, висячие ссылки и неопределенное поведение	5	1	3	1	практическое задание
7	Code review, статический анализ и правила исправления дефектов	5	1	3	1	практическое задание
8	Подготовка качества выпускного проекта к защите	4	1	2	1	практическое задание
9	Промежуточная аттестация по модулю	6	0	0	0	зачет

Практическая подготовка. Практическая подготовка строится вокруг решения прикладных задач на C++. Слушатель пишет код, запускает программу, проверяет граничные случаи, исправляет ошибки компиляции и выполнения, оформляет результат в репозитории и готовит краткое пояснение логики решения. В заданиях оцениваются работоспособность, читаемость, корректность алгоритма, структура проекта и способность объяснить принятое решение.

Самостоятельная работа. Самостоятельная работа включает повторение лекционного материала, выполнение тренировочных задач, оформление кода, подготовку README, анализ предупреждений компилятора, исправление дефектов, чтение документации и подготовку к зачету. Результаты самостоятельной работы фиксируются в электронной информационно-образовательной среде или в рабочем репозитории.

Минимальный уровень освоения. Слушатель выполняет обязательное задание по образцу, получает рабочую программу, обрабатывает основные входные данные, демонстрирует запуск и поясняет ключевые части решения. Допускаются отдельные недочеты в стиле кода, если они не искажают результат и могут быть исправлены после обратной связи.

Повышенный уровень освоения. Слушатель самостоятельно улучшает структуру решения, выделяет функции или классы, добавляет проверки входных данных, тесты, обработку ошибок, аккуратную инструкцию запуска и поясняет альтернативные варианты реализации.

Типовые ошибки. К типовым ошибкам относятся отсутствие декомпозиции, глобальные переменные без необходимости, дублирование кода, игнорирование ошибок ввода, выход за границы массива, неопределенное поведение, утечки ресурсов, отсутствие инструкции по сборке, неполные тесты и неаккуратная история коммитов.

Промежуточная аттестация. Промежуточная аттестация проводится в форме зачета по практическому заданию. Для зачета слушатель представляет исходный код, инструкцию сборки и запуска, краткое описание решения и демон-

страцию работоспособности. Результат считается зачтенным, если задача понята, программа собирается, основные сценарии работают, ошибки обработаны на минимально допустимом уровне, код можно проверить и воспроизвести.

2.3.10. Модуль 10. Проектная практика

Трудоемкость модуля – 36 академических часов. Цель модуля – обеспечить подготовку выпускного проекта от идеи и требований до рабочего C++ приложения, репозитория, инструкции, тестов, демонстрации и защиты. Итоговый учебный артефакт модуля: черновик выпускного проекта, репозиторий, инструкция запуска, набор тестов, пояснительная записка и презентация. Модуль формирует компетенции: ПК-1–ПК-10. Содержание модуля связано с последующими разделами программы и используется при подготовке выпускного проекта.

Таблица 15 – Тематический план модуля 10 «Проектная практика»

№	Тема	Всего ч.	Лек. ч.	Прак. ч.	Сам. раб.	Контроль
1	Выбор темы, постановка цели и критериев приемки	7	0	5	2	проектная работа
2	Проектирование архитектуры, данных и сценариев пользователя	7	0	5	2	проектная работа
3	Реализация основной функциональности и контроль версий	7	0	5	2	проектная работа
4	Тестирование, отладка, исправление дефектов и стабилизация	5	0	4	1	проектная работа
5	Оформление репозитория, README, пояснительной записки и презентации	5	0	4	1	проектная работа

Продолжение таблицы 15

Продолжение таблицы 15

№	Тема	Всего ч.	Лек. ч.	Прак. ч.	Сам. раб.	Контроль
6	Предзащита, обратная связь и финальная доработка	5	0	4	1	проектная работа

Практическая подготовка. Практическая подготовка строится вокруг решения прикладных задач на C++. Слушатель пишет код, запускает программу, проверяет граничные случаи, исправляет ошибки компиляции и выполнения, оформляет результат в репозитории и готовит краткое пояснение логики решения. В заданиях оцениваются работоспособность, читаемость, корректность алгоритма, структура проекта и способность объяснить принятое решение.

Самостоятельная работа. Самостоятельная работа включает повторение лекционного материала, выполнение тренировочных задач, оформление кода, подготовку README, анализ предупреждений компилятора, исправление дефектов, чтение документации и подготовку к зачету. Результаты самостоятельной работы фиксируются в электронной информационно-образовательной среде или в рабочем репозитории.

Минимальный уровень освоения. Слушатель выполняет обязательное задание по образцу, получает рабочую программу, обрабатывает основные входные данные, демонстрирует запуск и поясняет ключевые части решения. Допускаются отдельные недочеты в стиле кода, если они не искажают результат и могут быть исправлены после обратной связи.

Повышенный уровень освоения. Слушатель самостоятельно улучшает структуру решения, выделяет функции или классы, добавляет проверки входных данных, тесты, обработку ошибок, аккуратную инструкцию запуска и поясняет альтернативные варианты реализации.

Типовые ошибки. К типовым ошибкам относятся отсутствие декомпозиции, глобальные переменные без необходимости, дублирование кода, игнорирование ошибок ввода, выход за границы массива, неопределенное поведение, утечки ресурсов, отсутствие инструкции по сборке, неполные тесты и неаккуратная история коммитов.

2.3.11. Модуль 11. Итоговая аттестация

Трудоемкость модуля – 32 академических часов. Цель модуля – оценить готовность слушателя выполнять профессиональные задачи C++ разработчика на основе комплексной защиты выпускного проекта. Итоговый учебный артефакт модуля: защищенный выпускной проект, протокол итоговой аттестации и решение комиссии. Модуль формирует компетенции: ПК-1–ПК-10. Содержание модуля связано с последующими разделами программы и используется при подготовке выпускного проекта.

Таблица 16 – Тематический план модуля 11 «Итоговая аттестация»

№	Тема	Всего ч.	Лек. ч.	Прак. ч.	Сам. раб.	Контроль
1	Подготовка материалов к защите	0	0	0	0	итоговая аттестация
2	Демонстрация сборки, запуска и работы приложения	0	0	0	0	итоговая аттестация
3	Ответы на вопросы комиссии по коду, архитектуре и тестам	0	0	0	0	итоговая аттестация
4	Фиксация итоговой оценки и рекомендаций	0	0	0	0	итоговая аттестация
5	Итоговая защита выпускного проекта	32	0	0	0	защита проекта

Практическая часть итоговой аттестации. Слушатель представляет итоговый проект, демонстрирует сборку, запуск, основные сценарии, структуру кода, тесты, обработку ошибок, работу с данными и инструкцию воспроизведения.

Комиссия оценивает техническую реализацию, качество архитектуры, самостоятельность, полноту документации и ответы на вопросы.

Минимальный уровень успешной защиты. Проект собирается, запускается, реализует заявленные основные сценарии, имеет понятную структуру, содержит инструкцию, демонстрирует применение C++ и подтверждается ответами слушателя на вопросы комиссии.

Повышенный уровень защиты. Проект имеет аккуратную архитектуру, тесты, обработку ошибок, расширяемость, воспроизводимую сборку, осмысленную Git-историю, качественное README, отчет о тестировании и демонстрацию возможных направлений развития.

2.4. Проектная практика и консультации

Проектная траектория и тематика выпускных проектов. Выпускной проект выполняется как комплексная C++ разработка. Проект должен включать постановку задачи, описание предметной области, сценарии пользователя, архитектурную схему или текстовое описание модулей, исходный код, инструкцию сборки, тестовые данные, результаты тестирования, анализ ограничений и презентацию для защиты. Допускается выполнение индивидуального проекта или проекта в малой группе при фиксации личного вклада каждого слушателя.

Таблица 17 – Примерная тематика выпускных проектов

№	Тема проекта	Содержание результата	Обязательные инструменты
1	Консольный менеджер задач	приложение для создания, редактирования, поиска, сортировки и закрытия задач с сохранением данных в файл	C++, Git, CMake, тесты, README; при необходимости SQL, PostgreSQL, Linux
2	Система учета товаров на складе	учет позиций, остатков, цен, приходов, списаний и формирование отчета по складу	C++, Git, CMake, тесты, README; при необходимости SQL, PostgreSQL, Linux

Продолжение таблицы 17

Продолжение таблицы 17

№	Тема проекта	Содержание результата	Обязательные инструменты
3	База зарплат сотрудников	хранение сотрудников, должностей, ставок, премий, расчет выплат и поиск по параметрам	C++, Git, CMake, тесты, README; при необходимости SQL, PostgreSQL, Linux
4	Сервис бронирования поездок	модель клиентов, транспорта, маршрутов и заявок с проверкой доступности и стоимости	C++, Git, CMake, тесты, README; при необходимости SQL, PostgreSQL, Linux
5	Система управления пассажирскими перевозками	учет вместимости, посадки пассажиров, свободных мест и отчетов по рейсам	C++, Git, CMake, тесты, README; при необходимости SQL, PostgreSQL, Linux
6	Key-value хранилище на дереве поиска	реализация добавления, поиска, обновления, удаления записей и обходов дерева	C++, Git, CMake, тесты, README; при необходимости SQL, PostgreSQL, Linux
7	Анализатор логов	разбор текстовых файлов, подсчет событий, поиск ошибок, группировка по времени и отчет	C++, Git, CMake, тесты, README; при необходимости SQL, PostgreSQL, Linux
8	Индексатор текстовых документов	построение обратного индекса, поиск по словам, ранжирование и сохранение индекса	C++, Git, CMake, тесты, README; при необходимости SQL, PostgreSQL, Linux
9	Графовый планировщик маршрутов	представление графа, поиск пути, расчет расстояний и сравнение алгоритмов	C++, Git, CMake, тесты, README; при необходимости SQL, PostgreSQL, Linux
10	Каталог книг или медиатеки	ООП-приложение с сущностями, фильтрами, сортировками, сохранением и отчетами	C++, Git, CMake, тесты, README; при необходимости SQL, PostgreSQL, Linux

Продолжение таблицы 17

Продолжение таблицы 17

№	Тема проекта	Содержание результата	Обязательные инструменты
11	Магазин с каталогом и корзиной в консоли	каталог, корзина, расчет стоимости, оформление заказа и проверка наличия товаров	C++, Git, CMake, тесты, README; при необходимости SQL, PostgreSQL, Linux
12	Приложение с PostgreSQL для учета заявок	C++ приложение с подключением к базе, CRUD-операциями, SQL-скриптами и тестами	C++, Git, CMake, тесты, README; при необходимости SQL, PostgreSQL, Linux
13	Многопоточный обработчик файлов	параллельная обработка набора файлов, сбор статистики, синхронизация и отчет	C++, Git, CMake, тесты, README; при необходимости SQL, PostgreSQL, Linux
14	Учебная библиотека C++ с CMake и unit-тестами	самостоятельный модуль с публичным API, тестами, документацией и примером использования	C++, Git, CMake, тесты, README; при необходимости SQL, PostgreSQL, Linux
15	Итоговый прикладной C++ проект	комплексное приложение с архитектурой, тестами, Git-историей, инструкцией запуска и защитой	C++, Git, CMake, тесты, README; при необходимости SQL, PostgreSQL, Linux

Сквозные требования к проектным работам. Каждая проектная работа должна иметь понятную цель, входные данные, ожидаемый результат, инструкцию сборки, инструкцию запуска и проверочные сценарии. Для работ среднего и повышенного уровня требуется CMake-проект, модульная структура, раздельные заголовочные и исходные файлы, тесты, обработка ошибок, Git-история и README. При использовании базы данных добавляются SQL-скрипты создания схемы, тестовые данные, параметры подключения через переменные окружения или конфигурационный файл и описание порядка запуска.

3. Организационно-педагогические условия реализации программы

3.1. Общие условия реализации

Реализация Программы обеспечивается учебным планом, календарным учебным графиком, рабочими программами модулей, электронными учебными материалами, заданиями, средствами коммуникации, системой фиксации результатов обучения и локальными нормативными актами ООО «Иннопрог». Образовательный процесс организуется с применением электронного обучения и дистанционных образовательных технологий, без обязательного личного присутствия слушателя в месте нахождения организации.

3.2. Материально-технические условия

Для освоения программы слушателю необходимы компьютер или ноутбук, стабильный доступ к сети Интернет, браузер, средства видеосвязи, учетная запись в образовательной платформе, доступ к материалам и заданиям. Для выполнения практических работ используются компилятор C++ с поддержкой актуального стандарта, редактор кода или IDE, терминал, Git, CMake, средства отладки, PostgreSQL или учебная база данных, средства тестирования и инструменты статического анализа. Конкретные версии программного обеспечения указываются в инструкциях к модулям и могут актуализироваться без изменения планируемых результатов.

Рекомендуемое программное обеспечение. В качестве рекомендуемого программного обеспечения могут использоваться GCC, Clang или MSVC, CMake, Ninja или Make, Git, Visual Studio Code, CLion или Visual Studio, gdb или lldb, GoogleTest, Catch2 или doctest, clang-format, clang-tidy, cppcheck, AddressSanitizer, UndefinedBehaviorSanitizer, ThreadSanitizer, Valgrind, PostgreSQL, DBeaver или иной клиент базы данных, Docker при необходимости, а также средства видеосвязи и образовательная платформа.

3.3. Электронная информационно-образовательная среда

Электронная информационно-образовательная среда обеспечивает размещение учебных материалов, доступ к заданиям, прием работ, фиксацию результатов текущего контроля, промежуточной аттестации и итоговой аттеста-

ции, коммуникацию со слушателями, хранение портфолио и обратной связи. Доступ к материалам предоставляется индивидуально. Действия слушателя в электронной среде фиксируются техническими средствами платформы в объеме, необходимом для организации образовательного процесса и подтверждения освоения программы.

3.4. Кадровые условия

Реализация программы обеспечивается педагогическими работниками, наставниками, методистами и специалистами-практиками, имеющими профильное образование и опыт профессиональной деятельности в сфере разработки программного обеспечения, C++ программирования, системной разработки, тестирования, баз данных, Linux и сопровождения программных продуктов. К проведению отдельных занятий могут привлекаться практикующие разработчики, инженеры по качеству, специалисты по DevOps и руководители технических команд.

3.5. Учебно-методическое обеспечение

Информационно-методическое обеспечение. Слушателям предоставляются учебные материалы, видеоуроки, конспекты, практические задания, тестовые данные, шаблоны репозиториев, примеры CMake-файлов, чек-листы самопроверки, критерии оценивания, инструкции по настройке среды, образцы README, рекомендации по отладке, примеры unit-тестов и методические материалы по подготовке выпускного проекта.

3.6. Применение электронного обучения и дистанционных образовательных технологий

При реализации программы применяются электронное обучение и дистанционные образовательные технологии. Идентификация и аутентификация слушателя осуществляются через учетную запись в образовательной платформе, средства видеосвязи, проверку контактных данных и иные способы, предусмотренные локальным актом. При проведении контрольных мероприятий организация вправе применять видеоконференцию, демонстрацию экрана, устную защиту, анализ истории репозитория, проверку метаданных файлов, до-

полнительное собеседование и практическую проверку самостоятельности выполнения работы.

3.7. Информационная безопасность, персональные данные и этика

Требования к обработке персональных данных и безопасности. При реализации программы обрабатываются персональные данные слушателей в объеме, необходимом для заключения и исполнения договора, организации обучения, учета результатов, связи со слушателем, проведения аттестации и выдачи документов. Обработка осуществляется в соответствии с законодательством Российской Федерации и локальными актами организации. При выполнении проектов слушатель обязан использовать обезличенные, учебные, синтетические или открытые данные, если иное специально не согласовано с организацией и не оформлено правомерным основанием.

Информационная безопасность, персональные данные и этика. При реализации Программы обеспечивается соблюдение требований законодательства о персональных данных, правил работы с учетными записями, учебными материалами, проектными файлами и результатами аттестации. Доступ к электронной информационно-образовательной среде предоставляется индивидуально и не подлежит передаче третьим лицам.

Слушатель обязан соблюдать академическую добросовестность, не представлять чужие работы как собственные, корректно указывать источники заимствований, не нарушать права третьих лиц на программный код, данные, изображения, тексты и иные материалы. При выявлении признаков несамостоятельного выполнения работы организация вправе назначить дополнительную проверку, собеседование, повторную сдачу или иные процедуры, предусмотренные локальными нормативными актами.

3.8. Локальные процедуры обучения

Прием на обучение, зачисление, перевод, отчисление, восстановление, промежуточная аттестация, пересдача, апелляция, итоговая аттестация, выдача документов о квалификации и передача сведений во ФРДО регулируются локальными нормативными актами ООО «Иннопрог». Слушатель информиру-

ется о порядке обучения, формах контроля, сроках выполнения заданий и требованиях к итоговой аттестации до начала или в начале освоения программы.

3.9. Обновление программы

Программа подлежит актуализации при изменении законодательства об образовании, требований к дополнительным профессиональным программам, профессиональных стандартов, квалификационных требований, применяемых технологий, программного обеспечения и практики реализации обучения. Изменения утверждаются в порядке, установленном локальными нормативными актами ООО «Иннопрог», и фиксируются в листе актуализации.

4. Оценка качества освоения программы

4.1. Общие положения

Раздел содержит сведения, необходимые для единообразной реализации Программы и применяется в соответствии с локальными нормативными актами ООО «Иннопрог».

4.2. Текущий контроль успеваемости

Текущий контроль. Текущий контроль проводится в форме автоматизированных тестов, практических заданий, проверки исходного кода, code review, анализа ошибок компиляции, проверки репозитория, устных пояснений, мини-проектов и проверки чек-листов. Текущий контроль направлен на своевременное выявление пробелов и формирование индивидуальной обратной связи.

4.3. Промежуточная аттестация

Промежуточная аттестация проводится по итогам модулей в форме зачета. Зачет может включать тестовую часть, практическое задание, проверку репозитория, демонстрацию сборки и запуска, защиту мини-проекта или устное пояснение результата. Оценка «зачтено» выставляется при достижении минимального уровня освоения и выполнении обязательных критериев. При ре-

зультате «не зачтено» слушателю предоставляется возможность доработки или пересдачи в порядке, установленном локальным актом.

4.4. Пересдача, апелляция и подтверждение самостоятельности

Слушатель, получивший результат «не зачтено» по текущему контролю, промежуточной аттестации или итоговой аттестации, вправе устранить замечания и представить работу повторно в порядке и сроки, установленные локальным нормативным актом организации. При повторной сдаче проверяется исправление указанных ошибок, сохранение работоспособности результата и достижение соответствующих планируемых результатов.

Апелляция по результатам промежуточной или итоговой аттестации подается в порядке и сроки, установленные локальным нормативным актом. Рассмотрение апелляции проводится с учетом материалов проверки, версии работы, комментариев проверяющего, демонстрации результата и пояснений слушателя.

При наличии сомнений в самостоятельности выполнения работы организация вправе провести дополнительное собеседование, запросить историю изменений, промежуточные версии, пояснения по архитектуре, расчетам, коду, данным или иным материалам проекта.

Порядок пересдачи и апелляции. Порядок пересдачи промежуточной и итоговой аттестации, сроки доработки, состав комиссии и порядок подачи апелляции определяются локальными нормативными актами ООО «Иннопрог». При пересдаче слушатель представляет исправленную работу, описание внесенных изменений и ответы на замечания. Апелляция рассматривается по процедуре, установленной локальным актом, с фиксацией результата в протоколе или ином учетном документе.

4.5. Допуск к итоговой аттестации

К итоговой аттестации допускается слушатель, который освоил модули Программы, выполнил обязательные практические и проектные задания, прошел предусмотренные формы промежуточной аттестации, не имеет неурегулированной академической задолженности и представил материалы итогового аттестационного проекта в установленном порядке.

Допуск к итоговой аттестации может оформляться в электронной информационно-образовательной среде, ведомости, протоколе, приказе или ином учетном документе, предусмотренном локальными нормативными актами организации.

4.6. Итоговая аттестация

Итоговая аттестация проводится в форме защиты выпускного проекта. К защите допускаются слушатели, выполнившие учебный план, прошедшие промежуточную аттестацию, представившие выпускной проект, пояснительную записку, презентацию и материалы для воспроизведения результата. Защита включает демонстрацию сборки, запуска, основных сценариев, архитектуры, тестов, обработки ошибок и ответы на вопросы комиссии.

Шкала итоговой оценки. Итоговая оценка выставляется по 100-балльной шкале. Рекомендуемый порог успешного прохождения итоговой аттестации – не менее 70 баллов при отсутствии критических нарушений. Результат может быть переведен в качественную оценку: 90–100 баллов – отлично, 80–89 баллов – хорошо, 70–79 баллов – удовлетворительно, менее 70 баллов – неудовлетворительно.

Таблица 18 – Критерии оценивания итогового аттестационного проекта

№	Критерий	Макс. балл	Содержание оценки
1	Постановка задачи, требования и критерии приемки	10	цель проекта, пользователи, сценарии, входные данные, ограничения и ожидаемый результат
2	Архитектура и проектирование	15	декомпозиция, классы, функции, слои, зависимости, выбор структур данных и расширяемость
3	Качество C++ реализации	20	корректность, читаемость, применение STL, RAII, обработка ошибок, отсутствие грубых дефектов памяти
4	Сборка, инструменты и воспроизводимость	10	CMake или аналогичная сборка, Git-история, инструкция запуска, конфигурация окружения

Продолжение таблицы 18

Продолжение таблицы 18

№	Критерий	Макс. балл	Содержание оценки
5	Тестирование и отладка	15	unit-тесты, граничные случаи, отчет о проверке, устранение дефектов, применение отладочных инструментов
6	Работа с данными, файлами или базой данных	10	корректное хранение, чтение, запись, SQL или структурированная обработка данных при наличии в проекте
7	Документация и оформление репозитория	10	README, структура, пояснительная записка, комментарии там, где они нужны, аккуратность материалов
8	Защита и ответы на вопросы	10	демонстрация, аргументация решений, понимание ограничений, способность объяснить код и дальнейшее развитие

Критические основания для неудовлетворительного результата. Итоговая аттестация считается непройденной при наличии хотя бы одного критического основания: проект не собирается и не может быть проверен, отсутствует исходный код, отсутствует личный вклад слушателя, обнаружены признаки несамостоятельного выполнения без объяснения результата, заявленная функциональность отсутствует в существенной части, программа повреждает данные или окружение, отсутствует инструкция запуска, слушатель не может пояснить ключевые решения в собственном коде.

4.7. Внутренняя и внешняя оценка качества реализации программы

Внутренняя и внешняя оценка качества. Внутренняя оценка качества реализации программы включает анализ результатов текущего контроля, промежуточной аттестации, итоговой защиты, отзывов слушателей, обратной связи преподавателей, качества учебных материалов и востребованности выпускных проектов. Внешняя независимая оценка качества может проводиться через при-

влечение работодателей, профильных специалистов, экспертного сообщества или иных лиц, не участвовавших в непосредственной реализации занятий.

5. Методические материалы

5.1. Методические указания для слушателей

Рекомендации слушателю. Слушателю рекомендуется регулярно выполнять практические задания, не откладывать работу над проектом до конца программы, вести единый репозиторий, документировать сборку и запуск, фиксировать ошибки, сохранять промежуточные версии, читать сообщения компилятора, проверять граничные случаи и оформлять выводы после каждого крупного задания. Для каждой работы рекомендуется использовать чек-лист: задача понятна, входные данные описаны, программа собирается, основные сценарии работают, ошибки обработаны, код оформлен, README обновлен, коммиты отражают ход работы.

Требования к самостоятельности и академической добросовестности. Слушатель вправе использовать справочные материалы, официальную документацию, учебные примеры и инструменты автоматизации при условии самостоятельного понимания результата и соблюдения правил цитирования или указания источников. Запрещается представлять чужую работу как собственную, скрывать использование сторонних материалов, удалять следы существенных ошибок, подменять результаты тестирования и использовать решения, которые слушатель не способен объяснить. При сомнениях в самостоятельности комиссия вправе назначить дополнительное устное собеседование или практическую проверку.

5.2. Методические указания для преподавателей и проверяющих

Рекомендации преподавателю и наставнику. Преподаватель и наставник обеспечивают связь теории с практическими задачами, демонстрируют типовые ошибки, дают обратную связь по структуре кода, декомпозиции, качеству тестов, сборке, отладке и документации. Особое внимание уделяется воспроизводимости, осознанному выбору алгоритма, безопасному управлению ресур-

сами, понятной архитектуре и способности слушателя объяснить собственные технические решения.

5.3. Рекомендуемая литература и ресурсы

Рекомендуемые материалы определяются рабочими программами модулей и размещаются в электронной информационно-образовательной среде. В качестве учебно-методической базы используются материалы ООО «Иннопрог», официальная документация применяемых языков, библиотек, фреймворков и инструментов, а также справочные ресурсы, необходимые для выполнения практических и проектных работ.

Приложение 1. Матрица формирования компетенций

Приложение 1. Матрица соответствия компетенций, модулей и оценочных средств.

Таблица 19 – Матрица соответствия компетенций, модулей и оценочных средств

Код	Компетенция	Модули формирования	Оценочные средства
ПК-1	Анализировать требования к программному продукту	Вводный организационно-методический модуль; Основы программирования на C++; Алгоритмы и структуры данных на C++; Объектно-ориентированное программирование и архитектура C++ приложений; SQL и PostgreSQL для C++ разработчика; Проектная практика; Итоговая аттестация	практические задания, зачеты, code review, портфолио, выпускной проект
ПК-2	Проектировать алгоритмы, структуры данных и архитектуру решения	Основы программирования на C++; Алгоритмы и структуры данных на C++; Современный C++ и стандартная библиотека; Объектно-ориентированное программирование и архитектура C++ приложений; SQL и PostgreSQL для C++ разработчика; Проектная практика; Итоговая аттестация	практические задания, зачеты, code review, портфолио, выпускной проект

Продолжение таблицы 19

Продолжение таблицы 19

Код	Компетенция	Модули формирования	Оценочные средства
ПК-3	Разрабатывать C++ код, соответствующий поставленной задаче	Основы программирования на C++; Алгоритмы и структуры данных на C++; Современный C++ и стандартная библиотека; Тестирование, качество и безопасная разработка на C++; Проектная практика; Итоговая аттестация	практические задания, зачеты, code review, портфолио, выпускной проект
ПК-4	Применять объектно-ориентированное проектирование и современные возможности C++	Современный C++ и стандартная библиотека; Объектно-ориентированное программирование и архитектура C++ приложений; Проектная практика; Итоговая аттестация	практические задания, зачеты, code review, портфолио, выпускной проект
ПК-5	Работать с файлами, внешними данными, SQL и PostgreSQL	Инструменты разработки, Git, CMake и отладка; SQL и PostgreSQL для C++ разработчика; Тестирование, качество и безопасная разработка на C++; Проектная практика; Итоговая аттестация	практические задания, зачеты, code review, портфолио, выпускной проект
ПК-6	Использовать инструменты командной разработки и сборки	Инструменты разработки, Git, CMake и отладка; Linux, системное окружение и эксплуатация C++ приложений; Проектная практика; Итоговая аттестация	практические задания, зачеты, code review, портфолио, выпускной проект

Продолжение таблицы 19

Продолжение таблицы 19

Код	Компетенция	Модули формирования	Оценочные средства
ПК-7	Запускать, диагностировать и сопровождать C++ приложения в Linux-среде	Linux, системное окружение и эксплуатация C++ приложений; Проектная практика; Итоговая аттестация	практические задания, зачеты, code review, портфолио, выпускной проект
ПК-8	Тестировать, отлаживать и повышать качество C++ приложения	Основы программирования на C++; Алгоритмы и структуры данных на C++; Современный C++ и стандартная библиотека; Объектно-ориентированное программирование и архитектура C++ приложений; Инструменты разработки, Git, CMake и отладка; SQL и PostgreSQL для C++ разработчика; Linux, системное окружение и эксплуатация C++ приложений; Тестирование, качество и безопасная разработка на C++; Проектная практика; Итоговая аттестация	практические задания, зачеты, code review, портфолио, выпускной проект

Продолжение таблицы 19

Продолжение таблицы 19

Код	Компетенция	Модули формирования	Оценочные средства
ПК-9	Документировать проект и организовывать профессиональное портфолио	Вводный организационно-методический модуль; Объектно-ориентированное программирование и архитектура C++ приложений; Инструменты разработки, Git, CMake и отладка; Linux, системное окружение и эксплуатация C++ приложений; Тестирование, качество и безопасная разработка на C++; Проектная практика; Итоговая аттестация	практические задания, зачеты, code review, портфолио, выпускной проект
ПК-10	Представлять и защищать результат разработки	Вводный организационно-методический модуль; Проектная практика; Итоговая аттестация	практические задания, зачеты, code review, портфолио, выпускной проект

Приложение 2. Оценочные материалы промежуточной аттестации

Приложение 2. Фонд оценочных средств промежуточной аттестации. Фонд оценочных средств включает типовые задания, критерии проверки, перечень критических ошибок и рекомендации по доработке. Конкретные варианты заданий размещаются в электронной информационно-образовательной среде. Варианты могут актуализироваться с сохранением проверяемых компетенций, трудоемкости и уровня сложности.

Таблица 20 – Оценочные материалы промежуточной аттестации

Модуль	**Форма**	**Проверяемый результат**	**Критерии**
Основы программирования на C++	практическое задание, проверка репозитория и краткая защита результата	набор консольных программ, решающих прикладные задачи с вводом, обработкой, проверкой данных и выводом результата	сборка, запуск, корректность, структура, обработка ошибок, воспроизводимость, ответы на вопросы
Алгоритмы и структуры данных на C++	практическое задание, проверка репозитория и краткая защита результата	набор алгоритмических решений и мини-проект с применением выбранной структуры данных	сборка, запуск, корректность, структура, обработка ошибок, воспроизводимость, ответы на вопросы
Современный C++ и стандартная библиотека	практическое задание, проверка репозитория и краткая защита результата	модульная C++ библиотека с использованием контейнеров, алгоритмов, шаблонов и тестовых сценариев	сборка, запуск, корректность, структура, обработка ошибок, воспроизводимость, ответы на вопросы

Продолжение таблицы 20

Продолжение таблицы 20

Модуль	**Форма**	**Проверяемый результат**	**Критерии**
Объектно-ориентированное программирование и архитектура C++ приложений	практическое задание, проверка репозитория и краткая защита результата	объектно-ориентированное приложение с несколькими сущностями, сценариями использования, слоями логики и документацией по архитектуре	сборка, запуск, корректность, структура, обработка ошибок, воспроизводимость, ответы на вопросы
Инструменты разработки, Git, CMake и отладка	практическое задание, проверка репозитория и краткая защита результата	репозиторий C++ проекта с CMake, историей коммитов, инструкцией по сборке, отладочным сценарием и исправленными дефектами	сборка, запуск, корректность, структура, обработка ошибок, воспроизводимость, ответы на вопросы
SQL и PostgreSQL для C++ разработчика	практическое задание, проверка репозитория и краткая защита результата	C++ приложение с хранением данных в PostgreSQL или учебной базе, SQL-скриптами, миграциями и инструкцией запуска	сборка, запуск, корректность, структура, обработка ошибок, воспроизводимость, ответы на вопросы
Linux, системное окружение и эксплуатация C++ приложений	практическое задание, проверка репозитория и краткая защита результата	собранное и запущенное C++ приложение в Linux-среде с инструкцией, логами, конфигурацией и сценарием проверки	сборка, запуск, корректность, структура, обработка ошибок, воспроизводимость, ответы на вопросы

Продолжение таблицы 20

Продолжение таблицы 20

Модуль	**Форма**	**Проверяемый результат**	**Критерии**
Тестирование, качество и безопасная разработка на C++	практическое задание, проверка репозитория и краткая защита результата	набор unit-тестов, чек-лист качества, отчет о найденных дефектах и доработанная версия учебного проекта	сборка, запуск, корректность, структура, обработка ошибок, воспроизводимость, ответы на вопросы

Примеры заданий для промежуточной аттестации.

Основы программирования на C++. Слушателю предлагается решить прикладной кейс по теме модуля. Необходимо описать постановку задачи, подготовить исходный код, обеспечить сборку и запуск, приложить тестовые данные, оформить README и кратко объяснить выбранный подход. Минимальный результат: набор консольных программ, решающих прикладные задачи с вводом, обработкой, проверкой данных и выводом результата. Результат повышенного уровня включает тесты, проверку граничных случаев, аккуратную декомпозицию, анализ ошибок и описание возможного развития решения.

Алгоритмы и структуры данных на C++. Слушателю предлагается решить прикладной кейс по теме модуля. Необходимо описать постановку задачи, подготовить исходный код, обеспечить сборку и запуск, приложить тестовые данные, оформить README и кратко объяснить выбранный подход. Минимальный результат: набор алгоритмических решений и мини-проект с применением выбранной структуры данных. Результат повышенного уровня включает тесты, проверку граничных случаев, аккуратную декомпозицию, анализ ошибок и описание возможного развития решения.

Современный C++ и стандартная библиотека. Слушателю предлагается решить прикладной кейс по теме модуля. Необходимо описать постановку задачи, подготовить исходный код, обеспечить сборку и запуск, приложить тестовые данные, оформить README и кратко объяснить выбранный подход. Минимальный результат: модульная C++ библиотека с использованием контейнеров,

алгоритмов, шаблонов и тестовых сценариев. Результат повышенного уровня включает тесты, проверку граничных случаев, аккуратную декомпозицию, анализ ошибок и описание возможного развития решения.

Объектно-ориентированное программирование и архитектура C++ приложений. Слушателю предлагается решить прикладной кейс по теме модуля. Необходимо описать постановку задачи, подготовить исходный код, обеспечить сборку и запуск, приложить тестовые данные, оформить README и кратко объяснить выбранный подход. Минимальный результат: объектно-ориентированное приложение с несколькими сущностями, сценариями использования, слоями логики и документацией по архитектуре. Результат повышенного уровня включает тесты, проверку граничных случаев, аккуратную декомпозицию, анализ ошибок и описание возможного развития решения.

Инструменты разработки, Git, CMake и отладка. Слушателю предлагается решить прикладной кейс по теме модуля. Необходимо описать постановку задачи, подготовить исходный код, обеспечить сборку и запуск, приложить тестовые данные, оформить README и кратко объяснить выбранный подход. Минимальный результат: репозиторий C++ проекта с CMake, историей коммитов, инструкцией по сборке, отладочным сценарием и исправленными дефектами. Результат повышенного уровня включает тесты, проверку граничных случаев, аккуратную декомпозицию, анализ ошибок и описание возможного развития решения.

SQL и PostgreSQL для C++ разработчика. Слушателю предлагается решить прикладной кейс по теме модуля. Необходимо описать постановку задачи, подготовить исходный код, обеспечить сборку и запуск, приложить тестовые данные, оформить README и кратко объяснить выбранный подход. Минимальный результат: C++ приложение с хранением данных в PostgreSQL или учебной базе, SQL-скриптами, миграциями и инструкцией запуска. Результат повышенного уровня включает тесты, проверку граничных случаев, аккуратную декомпозицию, анализ ошибок и описание возможного развития решения.

Linux, системное окружение и эксплуатация C++ приложений. Слушателю предлагается решить прикладной кейс по теме модуля. Необходимо описать постановку задачи, подготовить исходный код, обеспечить сборку и запуск, приложить тестовые данные, оформить README и кратко объяснить выбранный подход. Минимальный результат: собранное и запущенное C++ приложение в Linux-среде с инструкцией, логами, конфигурацией и сценарием проверки. Результат повышенного уровня включает тесты, проверку граничных случаев, аккуратную декомпозицию, анализ ошибок и описание возможного развития решения.

Тестирование, качество и безопасная разработка на C++. Слушателю предлагается решить прикладной кейс по теме модуля. Необходимо описать постановку задачи, подготовить исходный код, обеспечить сборку и запуск, приложить тестовые данные, оформить README и кратко объяснить выбранный подход. Минимальный результат: набор unit-тестов, чек-лист качества, отчет о найденных дефектах и доработанная версия учебного проекта. Результат повышенного уровня включает тесты, проверку граничных случаев, аккуратную декомпозицию, анализ ошибок и описание возможного развития решения.

Приложение 3. Оценочные материалы итоговой аттестации

Приложение 3. Требования к выпускному проекту. Выпускной проект является основным доказательством достижения планируемых результатов. Проект должен показывать, что слушатель способен выполнить полный цикл C++ разработки: принять задачу, уточнить требования, спроектировать структуру решения, реализовать код, настроить сборку, проверить качество, оформить документацию и защитить результат.

Обязательная структура выпускного проекта.

1. титульная информация о проекте, авторе, программе и выбранной предметной области;
2. описание задачи, пользователей, сценариев и критериев приемки;
3. описание архитектуры, модулей, классов, функций и зависимостей;
4. исходный код с понятной структурой каталогов;
5. CMake-файл или иная инструкция воспроизводимой сборки;
6. тестовые данные, сценарии проверки и результаты тестирования;
7. описание обработки ошибок и ограничений решения;
8. инструкция запуска в локальной или Linux-среде;
9. пояснительная записка с выводами и направлениями развития;
10. презентация для итоговой защиты.

Паспорт выпускного проекта.

Таблица 21 – Паспорт выпускного проекта C++ разработчика

Раздел	**Содержание**
Название проекта	кратко отражает предметную область и программный результат
Пользователь результата	заказчик, преподаватель, учебная команда, оператор процесса или конечный пользователь приложения
Цель проекта	формулируется как проверяемый результат разработки
Основные сценарии	действия пользователя, команды приложения, входные данные и ожидаемые ответы

Продолжение таблицы 21

Продолжение таблицы 21

Раздел	**Содержание**
Архитектура	модули, классы, функции, связи, зависимости, файлы и слои приложения
Инструменты	C++, Git, CMake, отладчик, тестовый фреймворк, PostgreSQL или Linux при наличии
Качество	тесты, обработка ошибок, статический анализ, санитайзеры, проверка граничных случаев
Ограничения	границы функциональности, известные допущения, требования к окружению и направления развития

Приложение 5. Оценочный лист итоговой защиты.

Таблица 22 – Оценочный лист итоговой защиты

№	**Показатель**	**Макс. балл**	**Комментарий комиссии**
1	Постановка задачи, требования и критерии приемки	10	цель проекта, пользователи, сценарии, входные данные, ограничения и ожидаемый результат
2	Архитектура и проектирование	15	декомпозиция, классы, функции, слои, зависимости, выбор структур данных и расширяемость
3	Качество C++ реализации	20	корректность, читаемость, применение STL, RAII, обработка ошибок, отсутствие грубых дефектов памяти
4	Сборка, инструменты и воспроизводимость	10	CMake или аналогичная сборка, Git-история, инструкция запуска, конфигурация окружения
5	Тестирование и отладка	15	unit-тесты, граничные случаи, отчет о проверке, устранение дефектов, применение отладочных инструментов

Продолжение таблицы 22

Продолжение таблицы 22

№	Показатель	Макс. балл	Комментарий комиссии
6	Работа с данными, файлами или базой данных	10	корректное хранение, чтение, запись, SQL или структурированная обработка данных при наличии в проекте
7	Документация и оформление репозитория	10	README, структура, пояснительная записка, комментарии там, где они нужны, аккуратность материалов
8	Защита и ответы на вопросы	10	демонстрация, аргументация решений, понимание ограничений, способность объяснить код и дальнейшее развитие

Приложение 6. Перечень вопросов к итоговой защите.

1. Какую задачу решает ваш проект и кто является пользователем результата?
2. Какие требования и критерии приемки были определены перед реализацией?
3. Какие структуры данных и алгоритмы использованы в проекте?
4. Почему была выбрана именно такая архитектура приложения?
5. Какие классы являются ключевыми и какую ответственность они несут?
6. Где в проекте применяются функции, классы, STL, RAII или умные указатели?
7. Как организована сборка проекта и какие команды нужны для запуска?
8. Как Git-история отражает ход разработки?
9. Какие ошибки компиляции или выполнения были обнаружены и исправлены?
10. Какие тесты написаны и какие граничные случаи они проверяют?
11. Какие инструменты отладки или анализа качества применялись?
12. Как проект работает с файлами, базой данных или внешними данными?

13. Какие ограничения есть у текущей версии проекта?
14. Что нужно изменить для масштабирования или дальнейшего развития проекта?
15. Какие части проекта вы считаете наиболее сложными и почему?

Приложение 12. Карта контроля качества итогового проекта. Перед итоговой защитой слушатель проводит самопроверку проекта. Карта контроля качества используется как инструмент подготовки к защите и как ориентир для комиссии.

Таблица 23 – Карта контроля качества итогового проекта

№	Область проверки	Вопросы самопроверки	Подтверждающие материалы
1	Постановка задачи	Понятна ли цель проекта? Описан ли пользователь результата? Есть ли критерии приемки?	Раздел пояснительной записки, README, презентация
2	Сценарии пользователя	Можно ли показать основные действия пользователя от начала до результата? Обработаны ли ошибочные действия?	Демонстрационный сценарий, тестовые данные, скриншоты или протокол запуска
3	Архитектура	Разделена ли программа на модули? Понятна ли ответственность классов и функций? Есть ли лишние зависимости?	Описание архитектуры, структура каталогов, исходный код
4	Алгоритмы и структуры данных	Обоснован ли выбор структур данных? Понятна ли сложность ключевых операций?	Комментарий в пояснительной записке, ответы на вопросы, фрагменты кода
5	Управление ресурсами	Есть ли утечки памяти? Корректно ли используется RAII? Отсутствуют ли висячие ссылки?	Запуск санитайзера, ревью кода, отчет о проверке

Продолжение таблицы 23

Продолжение таблицы 23

№	Область проверки	Вопросы самопроверки	Подтверждающие материалы
6	Обработка ошибок	Проверяется ли ввод? Сообщаются ли причины ошибки? Сохраняется ли целостность данных?	Тесты, демонстрация ошибочных сценариев, код обработки исключений
7	Сборка и запуск	Можно ли собрать проект по инструкции? Указаны ли зависимости и команды?	README, CMake-файл, протокол сборки
8	Тестирование	Проверены ли нормальные и граничные сценарии? Есть ли автоматизированные тесты?	Каталог тестов, отчет запуска, таблица сценариев
9	Документация	Достаточно ли README для независимой проверки? Есть ли описание ограничений?	README, пояснительная записка, презентация
10	Защита	Готов ли автор объяснить код, архитектуру, тесты, ограничения и дальнейшее развитие?	Презентация, демонстрационный сценарий, ответы на контрольные вопросы

Приложение 4. Перечень локальных нормативных актов, на которые должна опираться реализация программы

Приложение 7. Локальные акты и регламенты, обеспечивающие реализацию программы. Для запуска и реализации программы образовательной организации рекомендуется иметь и применять локальные акты, регулирующие порядок приема и зачисления слушателей, обучение с применением электронного обучения и дистанционных образовательных технологий, текущий контроль и промежуточную аттестацию, итоговую аттестацию, пересдачу и апелляцию, выдачу документов о квалификации, передачу сведений во ФРДО, защиту персональных данных, хранение работ слушателей и порядок оказания платных образовательных услуг.

Приложение 5. Лист актуализации программы

Приложение 9. Лист актуализации программы. Программа актуализируется при изменении законодательства, профессиональных стандартов, локальных актов, образовательных технологий, программного обеспечения, требований работодателей и по итогам анализа результатов освоения. При актуализации допускается обновление инструментов, примеров заданий, датасетов, версий компиляторов и технических инструкций при сохранении цели, трудоемкости, планируемых результатов и структуры аттестации либо с утверждением новой редакции программы в установленном порядке.

Приложение 6. Требования к репозиторию, README и демонстрации

Приложение 4. Чек-листы проверки работ.

Таблица 24 – Чек-листы проверки работ по модулям

Модуль	**Чек-лист**
Вводный организационно-методический модуль	цель понятна; код собирается; основные сценарии работают; ошибки обработаны; структура проекта логична; README есть; результат воспроизводим; выводы и ограничения указаны
Основы программирования на C++	цель понятна; код собирается; основные сценарии работают; ошибки обработаны; структура проекта логична; README есть; результат воспроизводим; выводы и ограничения указаны
Алгоритмы и структуры данных на C++	цель понятна; код собирается; основные сценарии работают; ошибки обработаны; структура проекта логична; README есть; результат воспроизводим; выводы и ограничения указаны
Современный C++ и стандартная библиотека	цель понятна; код собирается; основные сценарии работают; ошибки обработаны; структура проекта логична; README есть; результат воспроизводим; выводы и ограничения указаны
Объектно-ориентированное программирование и архитектура C++ приложений	цель понятна; код собирается; основные сценарии работают; ошибки обработаны; структура проекта логична; README есть; результат воспроизводим; выводы и ограничения указаны
Инструменты разработки, Git, CMake и отладка	цель понятна; код собирается; основные сценарии работают; ошибки обработаны; структура проекта логична; README есть; результат воспроизводим; выводы и ограничения указаны

Продолжение таблицы 24

Продолжение таблицы 24

Модуль	**Чек-лист**
SQL и PostgreSQL для C++ разработчика	цель понятна; код собирается; основные сценарии работают; ошибки обработаны; структура проекта логична; README есть; результат воспроизводим; выводы и ограничения указаны
Linux, системное окружение и эксплуатация C++ приложений	цель понятна; код собирается; основные сценарии работают; ошибки обработаны; структура проекта логична; README есть; результат воспроизводим; выводы и ограничения указаны
Тестирование, качество и безопасная разработка на C++	цель понятна; код собирается; основные сценарии работают; ошибки обработаны; структура проекта логична; README есть; результат воспроизводим; выводы и ограничения указаны
Проектная практика	цель понятна; код собирается; основные сценарии работают; ошибки обработаны; структура проекта логична; README есть; результат воспроизводим; выводы и ограничения указаны

Приложение 8. Рубрикатор типовых ошибок C++ разработки.

- ошибки синтаксиса и непонимание сообщений компилятора;
- отсутствие декомпозиции и слишком длинные функции;
- использование глобальных данных без обоснования;
- выход за границы контейнера или массива;
- обращение к уничтоженному объекту или висячей ссылке;
- утечки ресурсов и отсутствие RAII;
- смешение пользовательского интерфейса, бизнес-логики и хранения данных;
- отсутствие тестов для граничных случаев;
- отсутствие инструкции сборки и запуска;
- неполная демонстрация результата на защите.

Приложение 11. Детализированные требования к C++ репозиторию. Репозиторий является обязательным элементом портфолио. Он должен позволять преподавателю, наставнику или комиссии воспроизвести результат без устных пояснений автора. Рекомендуемая структура итогового проекта:

– каталог `src` для исходных файлов реализации;
– каталог `include` для заголовочных файлов, если проект разделен на библиотеку и приложение;
– каталог `tests` для unit-тестов и проверочных сценариев;
– каталог `data` или `examples` для учебных входных данных;
– файл `CMakeLists.txt` или иной файл сборки;
– файл `README.md` с описанием назначения, сборки, запуска и проверки;
– файл с описанием архитектуры или соответствующий раздел пояснительной записки;
– файл с перечнем известных ограничений и дальнейших улучшений.

Требования к README. README должен быть понятен человеку, который впервые открывает проект. Рекомендуемая структура README:

1. название проекта и краткое описание задачи;
2. перечень реализованных функций;
3. требования к окружению и версиям инструментов;
4. команды сборки;
5. команды запуска;
6. примеры входных данных и ожидаемого результата;
7. команды запуска тестов;
8. описание структуры каталогов;
9. известные ограничения;
10. возможные направления развития.

Требования к стилю кода. Код должен быть читаемым и пригодным для сопровождения. Требуется единый стиль именования, ограничение длины функций, осмысленное разделение ответственности, отсутствие лишнего дублирования, понятная обработка ошибок и отсутствие неиспользуемых фрагментов. Комментарии применяются там, где они объясняют сложную логику, инвари-

ант, ограничение или причину выбранного решения. Комментарии, повторяющие очевидный код, не повышают качество работы.

Требования к сборке. Сборка должна быть воспроизводимой. Для итогового проекта предпочтительно использовать CMake. В README должны быть приведены команды создания каталога сборки, конфигурации, компиляции и запуска. Для проектов с внешними зависимостями необходимо указать способ установки зависимостей. Для проектов, использующих PostgreSQL, необходимо описать порядок подготовки базы, создания таблиц, загрузки тестовых данных и настройки подключения.

Требования к тестированию. Тесты должны проверять ключевую бизнес-логику, граничные случаи, некорректный ввод и обработку ошибок. Для простых проектов допускаются ручные сценарии с описанием входных и ожидаемых данных. Для проектов среднего и повышенного уровня ожидаются unit-тесты через GoogleTest, Catch2, doctest или аналогичный инструмент. В итоговом проекте желательно отделять чистую логику от интерфейса ввода-вывода, чтобы ее можно было тестировать автоматически.

Приложение 13. Расширенный перечень контрольных вопросов по модулям. Контрольные вопросы используются для подготовки к промежуточной аттестации, собеседованию и итоговой защите. Преподаватель может выбирать вопросы в зависимости от пройденного материала и уровня сложности задания.

Основы программирования на C++.

1. Из каких частей состоит простая программа на C++?
2. Чем отличается объявление переменной от ее инициализации?
3. Какие типы данных применяются для целых чисел, вещественных чисел, символов и логических значений?
4. Почему важно проверять корректность пользовательского ввода?
5. Чем цикл `for` отличается от `while` по назначению?
6. Когда стоит выделять повторяющуюся логику в функцию?
7. Какие ошибки возникают при выходе за границы массива или вектора?
8. Как организовать чтение и запись данных в файл?

Алгоритмы и структуры данных.

1. Что означает асимптотическая сложность алгоритма?
2. Когда линейный поиск допустим, а когда нужен более быстрый способ?
3. В каких случаях удобны стек и очередь?
4. Чем хеш-таблица отличается от дерева поиска?
5. Как выбрать структуру данных для хранения каталога товаров?
6. Какие ошибки возникают при рекурсии?
7. Как проверить корректность алгоритма на граничных данных?
8. Почему один и тот же алгоритм может быть непригоден для больших данных?

Современный C++ и стандартная библиотека.

1. Зачем разделять заголовочные и исходные файлы?
2. Как выбрать контейнер STL под задачу?
3. Что такое итератор и как он связан с алгоритмами стандартной библиотеки?
4. Как RAII помогает управлять ресурсами?
5. Чем `std::unique_ptr` отличается от `std::shared_ptr`?
6. Для чего нужны шаблоны функций и классов?
7. Как исключения влияют на проектирование функций?
8. Почему копирование и перемещение важны для производительности?

ООП и архитектура.

1. Как определить ответственность класса?
2. Чем инкапсуляция полезна для сопровождения приложения?
3. Когда предпочтительнее композиция, а когда допустимо наследование?
4. Что такое полиморфизм и где он полезен в прикладном проекте?
5. Как избежать слишком большого класса?
6. Какие признаки говорят о необходимости рефакторинга?
7. Как отделить интерфейс пользователя от бизнес-логики?
8. Какие архитектурные решения были приняты в вашем проекте?

Инструменты, сборка и отладка.

1. Какие файлы должны входить в C++ репозиторий?
2. Почему важно фиксировать изменения небольшими коммитами?

3. Как CMake описывает цель сборки?
4. Чем Debug-сборка отличается от Release-сборки?
5. Как найти место падения программы через отладчик?
6. Что показывают предупреждения компилятора?
7. Для чего нужны clang-format и clang-tidy?
8. Какие дефекты помогают находить санитайзеры?

SQL, PostgreSQL и Linux.

1. Что такое первичный ключ и внешний ключ?
2. Как связать две таблицы через JOIN?
3. Почему параметры подключения нельзя жестко вшивать в код?
4. Как транзакция помогает сохранить целостность данных?
5. Какие команды Linux используются для навигации и просмотра файлов?
6. Как проверить, запущен ли процесс?
7. Как сохранить лог работы приложения?
8. Что должно быть в инструкции по запуску на сервере или учебной виртуальной машине?

Тестирование и качество.

1. Чем unit-тест отличается от интеграционной проверки?
2. Как выбрать набор граничных случаев?
3. Какие ошибки в C++ связаны с памятью?
4. Что такое неопределенное поведение?
5. Как оформить отчет о дефекте?
6. Что проверяется на code review?
7. Почему тесты должны быть воспроизводимыми?
8. Какие проверки вы добавили бы в свой выпускной проект при дополнительном времени?

Приложение 15. Требования к репозиторию, README и демонстрации. Для подтверждения освоения Программы и выполнения итогового проекта слушатель представляет проектные материалы в форме, позволяющей проверить самостоятельность, воспроизводимость и качество результата.

Минимальный состав проектных материалов:

– репозиторий или архив проекта с исходным кодом и структурой каталогов;
– файл README.md или иной документ с описанием цели, функций, структуры, установки и запуска проекта;
– описание используемых технологий, зависимостей, версии языка/фреймворка/инструментов;
– инструкции по подготовке окружения, настройке переменных, запуску приложения, тестов или демонстрационного сценария;
– демонстрационные данные, скриншоты, ссылка на развернутую версию или видеодемонстрация при наличии такой возможности;
– краткое описание ограничений, известных проблем и возможных направлений развития проекта.

Материалы должны быть достаточны для повторной проверки проекта преподавателем, экспертом или членом аттестационной комиссии без дополнительных устных пояснений, за исключением вопросов по защите.

Приложение 7. Дорожная карта подготовки выпускного проекта

Приложение 10. Подробная карта практических заданий по модулям. Данное приложение конкретизирует практическую часть программы. Карта заданий может использоваться преподавателем и наставником для планирования работ, выдачи индивидуальных вариантов и проверки портфолио слушателя. Каждое задание должно иметь исходный код, инструкцию запуска, примеры входных данных, ожидаемый результат и краткий комментарий слушателя о принятом решении.

Таблица 25 – Карта практических заданий по модулям

№	Модуль	Практические задания	Проверяемый результат
1	Вводный организационно-методический модуль	Настроить компилятор и редактор; создать первый репозиторий; собрать и запустить демонстрационную программу; заполнить индивидуальный план обучения.	Слушатель подтверждает техническую готовность к обучению и понимает правила оформления работ.
2	Основы программирования на C++	Реализовать калькулятор с проверкой ввода; написать программу учета простых товаров; выполнить серию задач на циклы, массивы, строки и функции.	Слушатель уверенно использует базовые конструкции C++, разбивает решение на функции и проверяет основные входные данные.
3	Алгоритмы и структуры данных на C++	Реализовать сортировку и поиск; подготовить key-value хранилище на дереве поиска; решить задачи на стек, очередь, хеш-таблицу и граф.	Слушатель выбирает структуру данных под задачу, объясняет сложность и проверяет корректность алгоритма.

Продолжение таблицы 25

Продолжение таблицы 25

№	Модуль	Практические задания	Проверяемый результат
4	Современный C++ и стандартная библиотека	Разработать библиотеку обработки коллекций; применить STL-алгоритмы; реализовать класс с корректным управлением ресурсом; добавить обработку исключений.	Слушатель применяет STL, RAII, умные указатели, шаблоны и понимает время жизни объектов.
5	Объектно-ориентированное программирование и архитектура C++ приложений	Спроектировать модель предметной области; реализовать классы, интерфейсы и связи; разделить код на слои; выполнить рефакторинг.	Слушатель проектирует приложение через сущности, ответственность классов, композицию и полиморфизм.
6	Инструменты разработки, Git, CMake и отладка	Настроить CMake-проект; создать ветку для новой функции; решить конфликт; найти дефект через отладчик; подключить форматирование и статический анализ.	Слушатель обеспечивает воспроизводимую сборку, понятную историю изменений и использует инструменты диагностики.
7	SQL и PostgreSQL для C++ разработчика	Спроектировать схему базы; написать SQL-запросы; подготовить тестовые данные; подключить C++ приложение к базе; обработать ошибки подключения.	Слушатель понимает связь программы с хранилищем данных, пишет безопасные запросы и проверяет целостность данных.
8	Linux, системное окружение и эксплуатация C++ приложений	Собрать проект в Linux; подготовить Bash-скрипт запуска; настроить переменные окружения; собрать логи; описать порядок развертывания.	Слушатель может подготовить приложение к запуску в удаленной или серверной среде и объяснить диагностику проблем.

Продолжение таблицы 25

Продолжение таблицы 25

№	Модуль	Практические задания	Проверяемый результат
9	Тестирование, качество и безопасная разработка на C++	Написать unit-тесты; проверить граничные случаи; запустить санитайзер; оформить отчет о дефекте; выполнить code review собственного решения.	Слушатель подтверждает качество кода через тесты, анализ ошибок и исправление выявленных дефектов.
10	Проектная практика	Сформировать техническое задание; подготовить архитектуру; реализовать основной сценарий; стабилизировать проект; провести предзащиту.	Слушатель доводит проект до состояния, пригодного для итоговой защиты.
11	Итоговая аттестация	Представить репозиторий, пояснительную записку, презентацию, демонстрацию сборки и запуска, тесты и ответы на вопросы комиссии.	Комиссия оценивает готовность слушателя выполнять профессиональные задачи C++ разработчика.

Приложение 14. Дорожная карта подготовки выпускного проекта. Дорожная карта помогает распределить работу над выпускным проектом и снизить риск формальной подготовки проекта в последние дни обучения.

Таблица 26 – Дорожная карта подготовки выпускного проекта

Этап	Содержание работ	Результат	Контрольная точка
1	Выбор темы и предметной области	Предварительная тема, цель, пользователь результата	Согласование темы с наставником
2	Постановка требований	Сценарии, входные данные, ограничения, критерии приемки	Проверка полноты постановки
3	Проектирование архитектуры	Модули, классы, функции, структура каталогов	Архитектурное ревью

Продолжение таблицы 26

Продолжение таблицы 26

Этап	Содержание работ	Результат	Контрольная точка
4	Реализация базового сценария	Рабочий минимальный результат	Демонстрация первого запуска
5	Расширение функциональности	Дополнительные сценарии, обработка ошибок, хранение данных	Проверка сценариев пользователя
6	Настройка сборки и окружения	CMake, README, зависимости, команды запуска	Проверка воспроизводимости
7	Тестирование и исправление дефектов	Unit-тесты, граничные случаи, отчет о дефектах	Проверка качества
8	Документация и презентация	Пояснительная записка, презентация, демонстрационный сценарий	Предзащита
9	Финальная доработка	Исправление замечаний, стабилизация, оформление репозитория	Допуск к итоговой защите
10	Итоговая защита	Доклад, демонстрация, ответы на вопросы	Итоговая оценка комиссии

Приложение 8. Контрольный лист соответствия требованиям программы

Приложение 16. Контрольный лист соответствия требованиям программы. Контрольный лист применяется для внутренней проверки полноты Программы и готовности ее реализации.

Таблица 27 – Контрольный лист соответствия требованиям программы

№	Проверяемый элемент	Статус	Комментарий
1	Указаны вид программы, квалификация, трудоемкость, форма обучения, язык и срок освоения.	выполнено	
2	Представлены учебный план, календарный учебный график и рабочие программы модулей.	выполнено	
3	Описаны планируемые результаты, профессиональные компетенции и связь с квалификационными ориентирами.	выполнено	
4	Определены организационно-педагогические условия, ЭИОС, ЭО и ДОТ, кадровые и методические условия.	выполнено	
5	Описаны текущий контроль, промежуточная аттестация, итоговая аттестация, пересдача и апелляция.	выполнено	

Продолжение таблицы 27

№	Проверяемый элемент	Статус	Комментарий
6	Определены документы по итогам обучения и перечень локальных актов, обеспечивающих реализацию.	выполнено	