МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
ВЛАДИВОСТОКСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
КОЛЛЕДЖ ИНФОРМАЦИОННЫХ И КРЕАТИВНЫХ ТЕХНОЛОГИЙ

РАБОЧАЯ ПРОГРАММА УЧЕБНОЙ ДИСЦИПЛИНЫ

«МДК.01.02. Введение и основы программирования на C++»

программы подготовки специалистов среднего звена

09.02.10 Разработка компьютерных игр, дополненной и виртуальной реальности

Форма обучения: очная

Владивосток 2026

Рабочая программа учебной дисциплины МДК.01.02. Введение и основы программирования на C++ разработана в соответствии с требованиями Федерального государственного образовательного стандарта среднего профессионального образования по специальности 09.02.10 Разработка компьютерных игр, дополненной и виртуальной реальности, утвержденного приказом Минобрнауки России от 25 июня 2024, № 441, примерной образовательной программой.

Разработчик(и):О.Ю. Рудзейт, преподаватель

Рассмотрено и одобрено на заседании цикловой методической комиссии

Протокол № _9_ от «_20_» ___мая____2026_ г.

Председатель ПЦК _____ Н.В. Кравцова

подпись

подпись

СОДЕРЖАНИЕ

1.	ОБЩАЯ ХАРАКТЕРИСТИКА ПРИМЕРНОЙ РАБОЧЕЙ ПРОГРАММЫ УЧЕБНОЙ ДИСЦИПЛИНЫ	3
2.	СТРУКТУРА И СОДЕРЖАНИЕ УЧЕБНОЙ ДИСЦИПЛИНЫ	5
3.	УСЛОВИЯ РЕАЛИЗАЦИИ УЧЕБНОЙ ДИСЦИПЛИНЫ	10
4.	КОНТРОЛЬ И ОЦЕНКА РЕЗУЛЬТАТОВ ОСВОЕНИЯ УЧЕБНОЙ ДИСЦИПЛИНЫ	11

1. ОБЩАЯ ХАРАКТЕРИСТИКА ПРОГРАММЫ УЧЕБНОЙ ДИСЦИПЛИНЫ

1.1. Место дисциплины в структуре основной образовательной программы:

Учебная дисциплина МДК.01.02. «Введение и основы программирования на С++» является обязательной частью технического цикла основной образовательной программы в соответствии с ФГОС СПО по специальности 09.02.10 Разработка компьютерных игр, дополненной и виртуальной реальности

1.2. Цель и планируемые результаты освоения дисциплины:

В рамках программы учебной дисциплины обучающимися осваиваются умения и знания

Код компетенции	Умения	Знания
ПК -1.4 *(из ПООП соответствующей профессии)*	Уметь: Создавать локальный и удалённый репозиторий; выполнять базовые операции по управлению версиями (фиксация, отправка, получение изменений); создавать, переключать и удалять ветки разрешать конфликты при слиянии; выполнять rebase для поддержания линейной истории; работать с pull request'ами и рецензировать код; настраивать .gitignore для исключения временных и служебных файлов.	Знать: Основные принципы работы систем контроля версий (Git, SVN); базовые команды Git: init, clone, add, commit, push, pull, fetch, merge, rebase, branch, checkout; модели ветвления (Git Flow, GitHub Flow, Trunk Based Development); разрешение конфликтов слияния (merge conflicts); стратегии работы с удалёнными репозиториями (origin, upstream) принципы code review и pull request'ов; понятие CI/CD и интеграция с системами контроля версий.
ПК-1.5 *(из ПООП соответствующей профессии*	Уметь: Устанавливать и настраивать среду отладки в IDE (Visual Studio, CLion, VS Code); ставить условные и обычные точки останова; анализировать значения переменных и выражений в режиме отладки; выполнять трассировку выполнения программы до точки сбоя; использовать отладочную печать с уровнями логирования; обнаруживать и исправлять утечки памяти с помощью санитайзеров; отлаживать многопоточные приложения (просмотр потоков, локов).	Знать: Виды ошибок в программах на С++ (синтаксические, семантические, логические, ошибки времени выполнения); инструменты отладки: отладчик (GDB, Visual Studio Debugger), точки останова (breakpoints), контрольные точки (watchpoints); методы отладки: пошаговое выполнение, просмотр стека вызовов; работу с дампами памяти (core dump) и постмортем-отладку; использование логирования (std::cout, spdlog) для диагностики; отладку многопоточных программ профилирование памяти (valgrind, Address Sanitizer, Dr.Memory).
ПК-1.6 *(из ПООП соответствующей профессии*	Уметь: Выявлять дублирование кода и устранять его с помощью функций/шаблонов; упрощать сложные условные конструкции через ранний возврат или полиморфизм;	Знать: основные техники рефакторинга: выделение метода, перемещение поля, замена условного оператора полиморфизмом;

	применять автоматические инструменты рефакторинга (переименование, извлечение метода); заменять неэффективные контейнеры (std::list → std::vector) и алгоритмы; использовать move-конструкторы и std::move для снижения копирования; проводить профилирование и находить «горячие» участки кода; применять ключевые слова constexpr, inline, noexcept для оптимизации.	принципы SOLID и их влияние на качество кода; основы оптимизации C++: передача по ссылке против по значению, использование move-семантики; профилирование производительности (CPU profiler, VTune, perf); оптимизация циклов, контейнеров, алгоритмов STL; кэширование результатов, ленивые вычисления, предвычисление констант.
ПК-1.7 *(из ПООП соответствующей профессии*	Уметь: Подключать внешнюю статическую или динамическую библиотеку к проекту настраивать CMake для сборки с зависимостями; разрабатывать обёрточные классы для упрощения взаимодействия с фреймворком создавать конфигурационные файлы для параметров интеграции; писать код инициализации и деинициализации подключаемых модулей обрабатывать ошибки и исключения, возникающие на стыке модулей; реализовывать обратные вызовы (callbacks) и события для слабой связанности.	Знать: Способы интеграции модулей (статическая и динамическая линковка, shared libraries); механизмы интеграции с фреймворками (Qt, Boost, Unreal Engine, .NET); принципы сборки проектов: системы сборки (CMake, Make, MSBuild), управление зависимостями (vcpkg, Conan); форматы конфигурационных файлов (JSON, XML, INI) для настройки интеграции; типы интерфейсов: библиотечные вызовы, межпроцессное взаимодействие (трубы, сокеты), сериализация;
ПК-1.8 *(из ПООП соответствующей профессии*	Уметь: Собирать проект с интегрированным сторонним кодом без ошибок линковки; настраивать пути поиска заголовочных файлов и библиотек в IDE/CMake; выполнять динамическую загрузку плагинов и модулей во время выполнения; проверять совместимость версий фреймворка и собственного кода; отлаживать проблемы на этапе загрузки модуля (неразрешённые символы, неверный порядок инициализации); составлять инструкцию по интеграции для других разработчиков.	Знать: Этапы процесса интеграции: подготовка окружения, сборка зависимостей, компоновка, тестирование; работу с реестром, переменными окружения, путями для обнаружения модулей; механизмы динамической загрузки модулей; документирование интеграционных точек и API; методы работы с legacy-кодом при интеграции.
ПК-1.9 *(из ПООП соответствующей профессии*	Уметь: Создавать пакет установки с помощью инструмента (например, CPack, Inno Setup); настраивать конфигурационные файлы после установки (переменные окружения, параметры);	Знать: Типы инсталляторов (MSI, NSIS, Inno Setup, пакетные менеджеры); структуру установленного ПО: исполняемые файлы, конфиги, библиотеки, документация;

	выполнять установку и удаление приложения на целевых системах (Windows, Linux); настраивать сбор логов; проводить проверку работоспособности после инсталляции; применять системные инструменты для настройки производительности и мониторинга.	способы регистрации приложения в системе (переменные PATH, реестр Windows, сервисы); обновление и патчинг (инкрементальные обновления, дельта-патчи); мониторинг работы ПО: логирование, сбор метрик, система сбора ошибок; базовые принципы администрирования (права доступа, резервное копирование).

2. СТРУКТУРА И СОДЕРЖАНИЕ УЧЕБНОЙ ДИСЦИПЛИНЫ

2.1. Объем учебной дисциплины и виды учебной работы

Вид учебной работы	Объем часов
Объем образовательной программы учебной дисциплины	202
в том числе:	
– теоретическое обучение	76
– практические занятия *(если предусмотрено)*	96
– лабораторные занятия *(если предусмотрено)*	0
– курсовая работа (проект) *(если предусмотрено)*	0
– самостоятельная работа	24
– консультации	2
– промежуточная аттестация – *(форма промежуточной аттестации)*	4

2.2. Тематический план и содержание учебной дисциплины

Наименование разделов и тем	Содержание учебного материала и формы организации деятельности обучающихся	Объем в часах	Коды компетенций, формированию которых способствует элемент программы
1	2	3	4
Раздел 1. Инструментарий и среда разработки C++			
Тема 1.1. Современные стандарты C++ и управление памятью. Настройка IDE и компиляторов	**Эволюция стандартов C++17/20/23. RAII, умные указатели, move-семантика. Настройка IDE, флагов компилятора и путей поиска заголовочных файлов.**	5	ПК 1.5, 1.6
	Практическая работа №1.1: Настройка рабочего окружения разработчика. Конфигурация компиляторов (GCC/Clang/MSVC).	6	
	Самостоятельная работа обучающихся Изучение документации по стандартам C++. Подготовка отчёта о настройке локальной среды разработки.	2	
Тема 1.2. Системы сборки CMake и менеджеры пакетов. Конфигурация окружения	**Принципы работы CMake. Управление зависимостями через vcpkg/Conan. Переменные окружения, профили сборки, кросс-компиляция.**	5	ПК 1.9
	Практическая работа №1.2: Создание CMake-проекта. Подключение внешних библиотек и настройка путей.	6	
	Самостоятельная работа обучающихся Разработка скрипта автоматической сборки. Документирование зависимостей проекта	2	
Тема 1.3. Типы данных, память и умные указатели	**Основные типы данных языка C++. Использование контейнеров. Определение ссылок и указателей**	4	ПК 1.5, 1.6
	Практическая работа №1.3: Написание классов с управлением ресурсами. Замена сырых указателей на умные. Проверка утечек памяти	5	
	Самостоятельная работа обучающихся Разработка небольшого контейнера с использованием unique_ptr. Сравнение производительности сырых и умных указателей.	1	

Тема 1.4. Углублённое ООП: наследование, полиморфизм, виртуальные функции	**Определение классов в C++. Абстрактные классы. Наследование классов. Переопределение функций. Интерфейсы**	4	ПК 1.6
	Практическая работа №1.4: Разработка иерархии классов с виртуальными методами, абстрактные классы, интерфейсы.	6	
	Самостоятельная работа обучающихся **Описание абстрактного класса Автомобиль. Наследование от абстрактного класса легковых и грузовых машин. Программирование методов классов и переопределение методов абстрактного класса.**	1	
Раздел 2. Высокопроизводительный I/O и работа с данными			
Тема 2.1. Файловый ввод-вывод: синхронный и асинхронный	**Чтение больших файлов через Memory-mapped I/O. Работа с бинарными заголовками без копирования в буфер.**	4	ПК 1.6, 1.9
	Практическая работа №2.1: Работа с std::ifstream/ofstream, fread,и mmap, двоичные файлы. Отображение файлов в память (mmap).	5	
	Самостоятельная работа обучающихся Реализация класса для эффективного чтения больших файлов. Оптимизация структур для записи на диск	1	
Тема 2.2. Продвинутые структуры данных	Определение динамических структур. Описание бинарных и префиксных деревьев. Области применения структур	4	ПК 1.6
	Практическая работа №2.2: Реализация бинарного и префиксного дерева, хеш-таблицы с открытой адресацией. Сравнение с std::unordered_map.	5	
	Самостоятельная работа обучающихся Бенчмаркинг структур по времени вставки/поиска	1	
Тема 2.3. Сериализация и десериализация	**Сериализация (JSON, Binary, Protobuf/FlatBuffers). Zero-copy serialization**	4	ПК 1.7, 1.8
	Практическая работа №2.3. Сериализация в бинарный формат и JSON. Интеграция библиотеки FlatBuffers или Protobuf. Реализация асинхронного чтения логов/данных.	5	
	Самостоятельная работа обучающихся	1	

	Сравнение скорости сериализации различных форматов. Разработка схемы данных.		
Раздел 3. Управление памятью и оптимизация			
Тема 3.1. Move-семантика и семантика перемещения	**Rvalue и lvalue ссылки. Move constructors/assignment. Perfect forwarding. Оптимизация временных объектов.**	4	ПК 1.5, 1.6
	Практическая работа №3.1: Реализация move-aware классов. Сравнение производительности копирования и перемещения.	5	
	Самостоятельная работа обучающихся Анализ кода на наличие избыточных копий. Оптимизация с использованием std::move и std::forward.	1	
Тема 3.2. Аллокаторы памяти и кастомное управление ресурсами	**Стандартные и пользовательские аллокаторы. Pool allocation, arena allocators, stack allocators. Работа с памятью и оптимизация кэша.**	5	ПК 1.5, 1.6
	Практическая работа №3.2: Разработка pool allocator'a. Интеграция с STL контейнерами. Benchmark аллокаторов.	6	
	Самостоятельная работа обучающихся Оптимизация использования памяти. Оптимизация allocation patterns в проекте.	2	
Тема 3.3. Профилирование и поиск узких мест	**Инструменты профилирования. Анализ пропуска кэша. Оценивание ветвлений кода**	4	ПК 1.5
	Практическая работа №3.3: Использование профайлеров (perf, Very Sleepy, Visual Studio Profiler). Анализ cache misses и ветвлений.	5	
	Самостоятельная работа обучающихся Профилирование собственного контейнера, сравнение с std::vector и std::List, отчёт по оптимизации	1	
Тема 3.4. Бенчмаркинг и микрооптимизации	**Влияние директив на производительность проекта. Использование алгоритмов оптимальных по времени и памяти в проекте.**	4	ПК 1.5, 1.6
	Практическая работа №2.4: Написание бенчмарков. Сравнение разных алгоритмов (бинарный поиск vs линейный, эффективность кэша).	5	

	Самостоятельная работа обучающихся Проведение серии экспериментов: влияние ключевых слов inline, constexpr, noexcept на производительность.	1	
Раздел 4 Многопоточность и параллелизм			
Тема 4.1 Модель памяти C++ и атомарные операции	**Модели памяти стандартов C++11/14/17. Атомарные операции и выравнивание в памяти. Lock-free программирование.**	4	ПК 1.5, 1.8
	Практическая работа №4.1: Реализация lock-free структур данных. Анализ memory ordering на практике.	6	
	Самостоятельная работа обучающихся Исследование race conditions. Подготовка отчёта о выборе стратегий синхронизации.	2	
Тема 4.2. Параллельные алгоритмы и многопоточные паттерны	**Описание std::thread, std::async, thread pools. Параллельные алгоритмы стандарта C++ 17 и C++20.Синхронизация потоков. Разрешение тупиков.**	5	ПК 1.5, 1.7
	Практическая работа №4.2: Разработка потокобезопасного контейнера с использованием std::mutex, std::lock_guard . Параллелизация вычислений с использованием std::execution policies, std::async и пулом потоков.	6	
	Самостоятельная работа обучающихся Реализация простого thread pool с очередью задач. Анализ гонок данных с помощью ThreadSanitizer. Профилирование многопоточного кода. Оптимизация contention и scaling.	2	
Тема 4.3. Управление зависимостями и динамическая загрузка	**Динамические библиотеки dll. Способы подгрузки библиотек во время выполнения кода**	4	ПК 1.7, 1.8
	Практическая работа №4.3: Создание библиотеки (динамической/статической). Подключение через CMake. Динамическая загрузка плагинов (LoadLibrary/dlopen).	5	
	Самостоятельная работа обучающихся	1	

	Написание простого приложения с плагинами: загрузка .dll/.so во время выполнения		
Тема 4.4. Современные возможности C++ (C++17/20/23)	**Описание дополнительного функционала для работы с асинхронными модулями. Работа с пространствами имен std::optional, std::variant, std::any**	4	ПК 1.8
	Практическая работа №4.4: Использование std::optional, std::variant, std::any. Шаблонные лямбды	5	
	Самостоятельная работа обучающихся Рефакторинг кода с заменами сырых указателей на std::optional и std::variant	1	
Раздел 5 Контроль версий и командная разработка			
Тема 5.1 Git: ветвление, слияние, разрешение конфликтов	**Архитектура Git. Ветвление, merge/rebase, история коммитов. Алгоритмы разрешения конфликтов слияния.**	4	ПК 1.6, 1.9
	Практическая работа №5.1: Работа с ветками. Симуляция конфликтов и их разрешение. Интерактивный rebase.	5	
	Самостоятельная работа обучающихся Изучение стратегий ветвления (Git Flow, Trunk-Based). Подготовка схемы workflow.	1	
Тема 5.2 Git: углубленное изучение	**Принципы Code Review. Conventional Commits. Настройка Pull/Merge Request. Интеграция с системами CI.**	4	ПК 1.5, 1.9
	Практическая работа №5.2: Работа с git rebase, cherry-pick, reflog. Стратегии слияния. Интеграция с GitHub Actions.	5	
	Самостоятельная работа обучающихся Настройка простого CI-пайплайна: сборка, тесты, линтер.	2	
Тема 5.3 Workflow командной разработки: Pull	**Принципы Code Review. Conventional Commits. Настройка Pull/Merge Request. Интеграция с системами CI.**	4	ПК 1.6, 1.9
	Практическая работа №5.3: Создание и ревью запросов на слияние. Настройка шаблонов PR и проверок.	5	

Requests, Code Review	**Самостоятельная работа обучающихся** **Проведение ревью кода. Формирование отчёта по замечаниям.**	1	
Консультации		2	
Промежуточная аттестация (форма промежуточной аттестации)		4	
Всего:		202	

3. УСЛОВИЯ РЕАЛИЗАЦИИ ПРОГРАММЫ УЧЕБНОЙ ДИСЦИПЛИНЫ

3.1. Для реализации программы учебной дисциплины должны быть предусмотрены следующие специальные помещения:

Кабинет «Введение и основы программирования на C++»», оснащённый:

- *оборудованием:*

посадочные места по количеству обучающихся;

рабочее место преподавателя;

- *техническими средствами обучения:*

компьютер (ноутбук) с лицензионным программным обеспечением для преподавателя;

компьютер (ноутбук) с лицензионным программным обеспечением на каждое посадочное место по количеству обучающихся;

мультимедийный проектор;

мультимедийный экран;

информационно-коммуникативные средства;

экранно-звуковые пособия

3.2. Информационное обеспечение реализации программы

Для реализации программы библиотечный фонд образовательной организации должен иметь печатные и/или электронные образовательные и информационные ресурсы, для использования в образовательном процессе. При формировании библиотечного фонда образовательной организацией выбирается не менее одного издания из перечисленных ниже печатных изданий и (или) электронных изданий в качестве основного, при этом список может быть дополнен новыми изданиями.

3.2.1. Основные печатные издания

1. Дейтел П.Дж. (Paul J. Deitel), Дейтел Х.М. (Harvey M. Deitel). C++: как программировать. – 10-е изд. – М.: ООО «И.Д. Вильямс», 2021. – 1328 с.

2. Уильямс Э. (Anthony Williams). Параллельное программирование на C++ в действии. Практика разработки многопоточных программ (C++ Concurrency in Action). – 2-е изд. – М.: ДМК Пресс, 2023.

3. Пикус Ф.Г. (Fedor G. Pikus). The Art of Writing Efficient Programs. – Packt Publishing, 2021. – 464 p

3.2.2. Электронные издания

1. Online C++ development environment AdaCpp [сайт]. — URL: Режим доступа: https://www.producthunt.com/products/adacpp (дата обращения: 01.04.2026). – Текст : электронный.

2. Курс «Программирование на языке C++. Продвинутый поток 2025» [сайт]. — URL: https://wiki.atp-fivt.org, свободный. (дата обращения: 01.04.2026).

4 МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ОБУЧАЮЩИХСЯ ПО ИЗУЧЕНИЮ И РЕАЛИЗАЦИИ МЕЖДИСЦИПЛИНАРНОГО КУРСА

Начиная изучение дисциплины, студенту необходимо:

- ознакомиться с программой, изучить список рекомендуемой литературы;

- внимательно разобраться в структуре курса, в системе распределения учебного материала по видам занятий, формам контроля, чтобы иметь представление о курсе в целом.

Самостоятельная работа студентов (СРС) — это деятельность учащихся, которую они совершают без непосредственной помощи и указаний преподавателя, руководствуясь

сформировавшимися ранее представлениями о порядке и правильности выполнения операций. Цель СРС в процессе обучения заключается, как в усвоении знаний, так и в формировании умений и навыков по их использованию в новых условиях на новом учебном материале. Самостоятельная работа призвана обеспечивать возможность осуществления студентами самостоятельной познавательной деятельности в обучении, и является видом учебного труда, способствующего формированию у студентов самостоятельности. Кроме того, для расширения и углубления знаний по дисциплине целесообразно использовать: публикации в тематических журналах; полнотекстовые базы данных библиотеки; имеющиеся в библиотеках вуза. Успешное освоение дисциплины предполагает активную работу студентов на всех занятиях аудиторной формы: лекций и практических занятий, выполнение аттестационный мероприятий, эффективную самостоятельную работу.

4.1 Методические рекомендации обучающимся по обеспечению самостоятельной работы

Самостоятельная работа проводится с целью: систематизации, закрепления, углубления и расширения теоретических знаний и практических умений, приобретаемых студентами в ходе аудиторных занятий; формирования умений использовать специальную литературу; развития познавательных способностей и активности обучающихся; формирования самостоятельности мышления, способности к саморазвитию, самосовершенствованию и самореализации; развития исследовательских умений.

Для лучшего освоения материала и систематизации знаний по дисциплине необходимо отвечать на вопросы для самоконтроля. Во время самостоятельной проработки лекционного материала особое внимание следует уделять возникшим вопросам, непонятным терминам, спорным точкам зрения. В случае необходимости обращаться к преподавателю за консультацией. Студент должен четко уяснить, что именно с лекции начинается его подготовка к практическому занятию. Вместе с тем, лекция лишь организует мыслительную деятельность, но не обеспечивает глубину усвоения программного материала.

При подготовке к практическому занятию особое внимание необходимо обратить на содержание основных положений и выводов, объяснение явлений и фактов, уяснение практического приложения рассматриваемых теоретических вопросов. В процессе этой работы студент должен стремиться понять и запомнить основные положения рассматриваемого материала, примеры, поясняющие его, а также разобраться в иллюстративном материале. В процессе подготовки рекомендуется взаимное обсуждение материала, во время которого закрепляются знания, а также приобретается практика в изложении и разъяснении полученных знаний, развивается речь.

При необходимости следует обращаться за консультацией к преподавателю. Перед консультацией, необходимо хорошо продумать вопросы, которые требуют разъяснения.

В процессе изучения дисциплины студенту необходимо ориентироваться на самостоятельную проработку лекционного материала, подготовку к практическим занятиям, выполнение тестов, кейсовых заданий, самостоятельное изучение некоторых разделов курса. Для проведения занятий лекционного типа используются учебно-наглядные пособия в форме презентационных материалов, обеспечивающих тематические иллюстрации, соответствующие темам лекций.

4.2 Особенности организации обучения для лиц с ограниченными возможностями здоровья и инвалидов

При необходимости обучающимся из числа лиц с ограниченными возможностями здоровья и инвалидов (по заявлению обучающегося) предоставляется учебная информация в доступных формах с учетом их индивидуальных психофизических особенностей:

для лиц с нарушениями зрения: в печатной форме увеличенным шрифтом; в форме электронного документа; индивидуальные консультации с привлечением тифлосурдопереводчика; индивидуальные задания, консультации и др.

для лиц с нарушениями слуха: в печатной форме; в форме электронного документа; индивидуальные консультации с привлечением сурдопереводчика; индивидуальные задания, консультации и др.

для лиц с нарушениями опорно-двигательного аппарата: в печатной форме; в форме электронного документа; индивидуальные задания, консультации и др.

5 КОНТРОЛЬ И ОЦЕНКА РЕЗУЛЬТАТОВ ОСВОЕНИЯ КУРСА

Код ПК	Критерии оценки результата (показатели освоенности компетенций)	Формы контроля и методы оценки
ПК.1.4	Оценка "отлично": уверенно работает с ветками, разрешает сложные конфликты, настраивает CI/CD и соблюдает Conventional Commits. Оценка "хорошо": правильно использует основные команды Git, создаёт и сливает ветки, разрешает типовые конфликты. Оценка "удовлетворительно": выполняет базовые операции (clone, add, commit, push, pull), но неуверенно работает с ветвлением и конфликтами.	Проверка репозитория на платформе и структуры проекта
ПК.1.5	Оценка "отлично": профессионально использует отладчик (условные точки, стек, многопоточность для поиска любых ошибок. Оценка "хорошо": умеет запускать отладку, ставить точки останова, шагать по коду и находить типовые логические ошибки. Оценка "удовлетворительно": знаком с отладчиком, может найти простые ошибки через вывод в консоль или брейкпоинты.	Проверка умений профилирования при воссоздании ситуаций, приводящих к ошибкам
ПК.1.6	Оценка "отлично": системно применяет рефакторинг, профилирование и продвинутые оптимизации (move, аллокаторы, улучшение локальности). Оценка "хорошо": выполняет базовый рефакторинг (извлечение метода, устранение дублирования) и простые оптимизации (move, выбор контейнера).	Демонстрация оптимального кода при работе с памятью и объектами.

	Оценка "удовлетворительно": понимает необходимость рефакторинга, но исправляет только очевидные проблемы (например, копирование объектов).	
ПК.1.7	Оценка "отлично": самостоятельно разрабатывает полную процедуру интеграции (CMake, обёртки, обработка ошибок, документация). Оценка "хорошо": подключает внешнюю библиотеку, настраивает CMake и создаёт простую обёртку для фреймворка. Оценка "удовлетворительно": способен добавить статическую библиотеку по готовому образцу, но испытывает трудности с настройкой	Демонстрация навыков работы с внешними библиотеками, подключаемыми в проект
ПК.1.8	Оценка "отлично": успешно интегрирует несколько библиотек с динамической загрузкой плагинов и интеграционным тестированием. Оценка "хорошо": интегрирует 1–2 статические библиотеки, выполняет сборку без критических ошибок, реализует простую динамическую загрузку. Оценка "удовлетворительно": выполняет интеграцию по образцу, собирает проект, но допускает ошибки линковки, которые исправляет с помощью	Демонстрация навыков работы с внешними библиотеками, подключаемыми в проект
ПК.1.9	Оценка "отлично": создаёт полноценный инсталлятор, реализует автообновление и сбор crash-логов, пишет инструкцию по обслуживанию. Оценка "хорошо": создаёт установщик (CPack/NSIS), настраивает логирование и изменяет конфигурацию вручную. Оценка "удовлетворительно": устанавливает приложение ручным копированием или простым скриптом, может изменить параметры в конфиг-файле	Проверка работы сборки готовых приложений на основе имеющихся инструментов

4. КОНТРОЛЬ И ОЦЕНКА РЕЗУЛЬТАТОВ ОСВОЕНИЯ УЧЕБНОЙ ДИСЦИПЛИНЫ

4.1. Описание показателей и критериев оценки компетенций

Результаты обучения	*Критерии оценки*	*Методы оценки*
Перечень знаний, осваиваемых в рамках дисциплины		
Знать: – Современные стандарты C++ (C++17/20/23): основные возможности и отличия настройки проекта (Project Settings) для целевых платформ: разрешение, API рендеринга, управление памятью; – RAII, умные указатели (unique_ptr, shared_ptr, weak_ptr), move-семантика; – системы сборки CMake, пакетные менеджеры vcpkg/Conan, переменные окружения; – методы оптимизации: профилирование, бенчмаркинг, кэш-локальность, техники рефакторинга; – многопоточность: std::thread, std::async, пулы потоков, атомарные операции, модель памяти C++.	Демонстрирует понимание управления памятью, корректно использует умные указатели и move-семантику для избежания утечек Владеет навыками настройки CMake-проекта, подключения внешних библиотек и конфигурации окружения владеет инструментами профилирования (perf, Visual Studio Profiler), умеет выявлять «узкие места» и проводить рефакторинг корректно использует средства синхронизации (mutex, lock_guard), понимает принципы lock-free программирования самостоятельно создаёт и сливает ветки, разрешает конфликты, использует pull request и code review	Устный опрос. Защита результатов работы над проектом.
Перечень умений, осваиваемых в рамках дисциплины		
Уметь: – Настраивать среду разработки (IDE), компиляторы (GCC/Clang/MSVC) и систему сборки CMake; – проектировать и реализовывать иерархии классов с виртуальными функциями и абстрактными интерфейсами; – проводить рефакторинг кода (устранение дублирования, извлечение методов, замена условий полиморфизмом); – разрабатывать потокобезопасные контейнеры с использованием mutex и lock-free алгоритмов; – Настраивать CI/CD пайплайн (GitHub Actions) для автоматической сборки	Уверенно создаёт проект с CMake, подключает библиотеки и настраивает флаги компиляции создаёт классы без утечек памяти, заменяет сырые указатели на умные, демонстрирует работу move-конструкторов анализирует время выполнения, cache misses, ветвления; предлагает оптимизации улучшает структуру кода без изменения внешнего поведения создаёт классы, корректно работающие в многопоточной среде; избегает гонок данных и deadlock создаёт конфигурацию, которая собирает проект и прогоняет тесты при каждом push	Письменный и устный опрос. Защита результатов работы над проектом.

МИНОБРНАУКИ РОССИИ
ВЛАДИВОСТОКСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Колледж информационных и креативных технологий

КОНТРОЛЬНО-ОЦЕНОЧНЫЕ СРЕДСТВА
для проведения текущего контроля и промежуточной аттестации по учебной дисциплине

«МДК.01.02. Введение и основы программирования на С++»

программы подготовки специалистов среднего звена
09.02.10 Разработка компьютерных игр, дополненной и виртуальной реальности

Форма обучения: очная

Владивосток 2026

Контрольно-оценочные средства для проведения текущего контроля и промежуточной аттестации по учебной дисциплине МДК.01.02. Введение и основы программирования на С++ разработаны в соответствии с требованиями ФГОС СПО по специальности / профессии 09.02.10 Разработка компьютерных игр, дополненной и виртуальной реальности, утвержденного приказом Минобрнауки РФ от от 25 июня 2024, № 441, примерной образовательной программой, рабочей программой учебной дисциплины.

Разработчик(и): О.Ю. Рудзейт, преподаватель

Рассмотрено и одобрено на заседании цикловой методической комиссии

Протокол № _9_ от «_20_» ___мая____2026_ г.

Председатель ПЦК _____ Н.В. Кравцова

подпись

1. Общие сведения

Контрольно-оценочные средства (далее – КОС) предназначены для контроля и оценки образовательных достижений обучающихся, освоивших программу учебной дисциплины МДК.06.02 «Продвинутая разработка игровых систем Unreal».

КОС включают в себя контрольные материалы для проведения текущего контроля успеваемости и промежуточной аттестации по дисциплине, которая проводится в форме дифференцированного зачёта (с использованием оценочного средства - устный опрос в форме ответов на вопросы билетов, устный опрос в форме собеседования, выполнение письменных заданий, тестирование и т.д.).

2. Планируемые результаты обучения, обеспечивающие результаты освоения образовательной программы

Код ПК	Код результата обучения	Наименование результата обучения
ПК 1.4	З1	Основные принципы систем контроля версий (Git, SVN). Базовые команды Git: init, clone, add, commit, push, pull, fetch, merge, rebase, branch, checkout.
	З2	Модели ветвления (Git Flow, GitHub Flow, Trunk Based Development). Стратегии разрешения конфликтов слияния.
	З3	Принципы code review и pull request’ов. Понятие CI/CD и интеграция с системами контроля версий.
	У1	Создавать локальный и удалённый репозиторий, выполнять базовые операции с версиями.
	У2	Создавать, переключать и удалять ветки, разрешать конфликты, выполнять rebase.
	У3	Работать с pull request’ами, рецензировать код, настраивать .gitignore.
ПК 1.5	З1	Виды ошибок в программах на C++ (синтаксические, семантические, логические, ошибки времени выполнения).
	З2	Инструменты отладки: отладчик (GDB, Visual Studio Debugger), точки останова, контрольные точки, стек вызовов.
	З3	Методы отладки многопоточных программ (data races, deadlocks). Инструменты профилирования памяти (valgrind, Address Sanitizer).
	У1	Устанавливать и настраивать среду отладки, ставить условные и обычные точки останова.
	У2	Анализировать значения переменных и стек вызовов, выполнять трассировку до точки сбоя.
	У3	Обнаруживать и исправлять утечки памяти с помощью санитайзеров, отлаживать многопоточные приложения.
ПК 1.6	З1	Признаки «запахов» кода (duplicate code, long methods, large classes). Техники рефакторинга (выделение метода, замена условного оператора полиморфизмом).
	З2	Принципы SOLID. Основы оптимизации C++: передача по ссылке, move-семантика, constexpr, noexcept.
	З3	Инструменты профилирования производительности (CPU profiler, VTune, perf). Оптимизация контейнеров и алгоритмов STL.

Код ПК	Код результата обучения	Наименование результата обучения
	У1	Выявлять дублирование кода и устранять его, упрощать сложные условные конструкции.
	У2	Применять move-конструкторы и std::move, использовать constexpr и inline для оптимизации.
	У3	Проводить профилирование и находить «горячие» участки кода, заменять неэффективные контейнеры и алгоритмы.
ПК 1.7	З1	Понятие API и ABI, способы экспорта/импорта функций (DLL, .so, статическая линковка).
	З2	Механизмы интеграции с фреймворками (Qt, Boost). Системы сборки (CMake, Make), управление зависимостями (vcpkg, Conan).
	З3	Форматы конфигурационных файлов (JSON, XML, INI). Принципы тестирования интеграции.
	У1	Подключать внешнюю статическую или динамическую библиотеку к проекту, настраивать CMake.
	У2	Разрабатывать обёрточные классы для взаимодействия с фреймворком, создавать конфигурационные файлы.
	У3	Обрабатывать ошибки интеграции, реализовывать callbacks и события для слабой связанности.
ПК 1.8	З1	Этапы процесса интеграции: подготовка окружения, сборка зависимостей, компоновка, тестирование.
	З2	Соглашения о вызовах (cdecl, stdcall), name mangling. Механизмы динамической загрузки (LoadLibrary, dlopen).
	З3	Стратегии минимизации конфликтов версий библиотек (DLL hell, side-by-side).
	У1	Собирать проект с интегрированным сторонним кодом без ошибок линковки, настраивать пути поиска.
	У2	Выполнять динамическую загрузку плагинов и модулей во время выполнения.
	У3	Проверять совместимость версий, писать сценарии автоматизации интеграции (bash, Python).
ПК 1.9	З1	Типы инсталляторов (MSI, NSIS, Inno Setup, пакетные менеджеры). Структура установленного ПО.
	З2	Способы регистрации приложения в системе (переменные PATH, реестр Windows, сервисы).
	З3	Обновление и патчинг (инкрементальные обновления, дельта-патчи). Мониторинг и сбор ошибок (crash reporting).
	У1	Создавать пакет установки с помощью CPack, Inno Setup. Настраивать конфигурационные файлы.
	У2	Выполнять установку и удаление приложения на целевых системах, реализовывать механизм автообновления.
	У3	Настраивать сбор логов и автоматическую отправку ошибок, проводить проверку работоспособности после инсталляции.

3 Соответствие оценочных средств контролируемым результатам обучения

3.1 Средства, применяемые для оценки уровня теоретической подготовки

Краткое наименование раздела (модуля) / темы дисциплины	Код результата обучения	Показатель[2] овладения результатами обучения	Наименование оценочного средства и представление его в КОС[3]	
			Текущий контроль[4]	Промежуточная аттестация[4]
Раздел 1. Инструментарий и среда разработки C++				
Тема 1.1 Современные стандарты C++ и управление памятью. Настройка IDE и компиляторов	З1	Способность объяснить ключевые нововведения стандартов C++17/20/23 (if constexpr, structured bindings, ranges, coroutines)	Устный опрос (п.5.1 вопр. 1-5)	Собеседование
	З2	Знание принципов RAII, умных указателей (unique_ptr, shared_ptr, weak_ptr) и move-семантики	Устный опрос (п.5.1 вопр. 6-10)	Собеседование
	У1	Умение настраивать среду разработки (IDE), компиляторы (GCC/Clang/MSVC) и флаги оптимизации	Устный опрос (п.5.1 вопр. 11-15)	Собеседование
Тема 1.3 Типы данных, память и умные указатели	З3	Понимание различий между указателями и ссылками, типами контейнеров STL (vector, list, map)	Устный опрос (п.5.1 вопр. 16-20)	Собеседование
	У2	Умение работать с динамической памятью, утечками памяти и способами их предотвращения	Устный опрос (п.5.1 вопр. 21-25)	Собеседование
	У3	Умение заменять сырые указатели на умные, реализовывать классы с управлением ресурсами	Устный опрос (п.5.1 вопр. 26-30)	Собеседование
Тема 1.4. Углублённое ООП: наследование, полиморфизм, виртуальные функции	З4	Знание принципов наследования, полиморфизма, виртуальных функций и абстрактных классов	Устный опрос (п.5.1 вопр. 31-35)	Собеседование
	З5	Знание принципов использования интерфейсов и их наследования	Устный опрос (п.5.1 вопр. 36-40)	Собеседование
	У4	Умение проектировать иерархии классов, переопределять виртуальные методы, использовать интерфейсы	Устный опрос (п.5.1 вопр. 41-45)	Собеседование
Раздел 2. Высокопроизводительный I/O и работа с данными				
Тема 2.2 Продвинутые структуры данных	З6	Знание принципов работы бинарных деревьев поиска, префиксных деревьев, хеш-таблиц	Устный опрос (п.5.1 вопр. 46-53)	Собеседование
	У5	Умение реализовывать и сравнивать производительность структур данных	Устный опрос (п.5.1 вопр. 54-60)	Собеседование
Раздел 3. Управление памятью и оптимизация				
Тема 3.4. Бенчмаркинг и микрооптимизации	З7	Знание влияния директив inline, constexpr, noexcept на производительность кода	Устный опрос (п.5.1 вопр. 61-65)	Собеседование
	З8	Понимание принципов эффективного использования кэша (cache locality) и сравнения алгоритмов (бинарный поиск vs линейный)	Устный опрос (п.5.1 вопр. 66-70)	Собеседование

Краткое наименование раздела (модуля) / темы дисциплины	Код результата обучения	Показатель[2] овладения результатами обучения	Наименование оценочного средства и представление его в КОС[3]	
			Текущий контроль[4]	Промежуточная аттестация[4]
	У6	Умение анализировать результаты и выбирать оптимальные алгоритмы	Устный опрос (п.5.1 вопр. 71-75)	Собеседование
Раздел 4. Многопоточность и параллелизм				
Тема 4.2. Параллельные алгоритмы и многопоточные паттерны	З9	Знание средств многопоточности C++: std::thread, std::async, пулы потоков	Устный опрос (п.5.1 вопр. 76-80)	Собеседование
	З10	Понимание проблем синхронизации (гонки данных, deadlock) и методов их решения (mutex, lock_guard)	Устный опрос (п.5.1 вопр. 81-85)	Собеседование
	У7	Умение разрабатывать потокобезопасные контейнеры, применять параллельные алгоритмы (std::execution policies)	Устный опрос (п.5.1 вопр. 86-90)	Собеседование
Раздел 5. Контроль версий и командная разработка				
Тема 5.1 Git: ветвление, слияние, разрешение конфликтов	З11	Знание основных команд Git (init, clone, add, commit, push, pull, branch, merge, rebase)	Устный опрос (п.5.1 вопр. 91-5)	Собеседование
	З12	Понимание моделей ветвления (Git Flow, Trunk-Based) и стратегий разрешения конфликтов	Устный опрос (п.5.1 вопр. 96-100)	Собеседование
	У8	Умение создавать и сливать ветки, разрешать merge-конфликты, применять интерактивный rebase	Устный опрос (п.5.1 вопр. 101-105)	Собеседование

3.2 Средства, применяемые для оценки уровня практической подготовки

Краткое наименование раздела (модуля) / темы дисциплины	Код результата обучения	Показатель[2] овладения результатами обучения	Наименование оценочного средства и представление его в КОС[3]	
			Текущий контроль[4]	Промежуточная аттестация[4]
Раздел 1. Инструментарий и среда разработки C++				
Тема 1.3. Типы данных, память и умные указатели	З1	Способность различать и правильно выбирать контейнеры STL (vector, list, map) в зависимости от задачи	Устный опрос (п.5.1 вопр. 106-110)	Индивидуальный проект
	З2	Знание правил управления динамической памятью, умных указателей (unique_ptr, shared_ptr, weak_ptr) и Rule of Five	Устный опрос (п.5.1 вопр. 111-115)	Индивидуальный проект
	У1	Умение заменять сырые указатели на умные, реализовывать классы с управлением ресурсами и проверять утечки памяти	Устный опрос (п.5.1 вопр. 116-120)	Индивидуальный проект
Тема 1.4. Углублённое ООП: наследование,	З3	Способность объяснять механизмы наследования, полиморфизма, виртуальных функций и абстрактных классов	Устный опрос (п.5.1 вопр. 121-125)	Индивидуальный проект

Краткое наименование раздела (модуля) / темы дисциплины	Код результата обучения	Показатель[2] овладения результатами обучения	Наименование оценочного средства и представление его в КОС[3]	
			Текущий контроль[4]	Промежуточная аттестация[4]
полиморфизм, виртуальные функции	У2	Умение описывать методы интерфейсов и наследовать их	Устный опрос (п.5.1 вопр. 126-130)	Индивидуальный проект
	У3	Умение проектировать иерархии классов, переопределять виртуальные методы, использовать интерфейсы и множественное наследование	Устный опрос (п.5.1 вопр.131-135)	Индивидуальный проект
Раздел 2. Система рендеринга и графическая архитектура				
Тема 2.1. Файловый ввод-вывод: синхронный и асинхронный	З4	Знание методов файлового ввода-вывода (ifstream, ofstream, fread, mmap) и их особенности	Практическая презентация	Индивидуальный проект
	У4	Умение эффективно читать/записывать большие файлы, применять memory-mapped I/O и асинхронные операции	Практическая презентация	Индивидуальный проект
Тема 2.2. Продвинутые структуры данных	З5	Способность описать принципы работы бинарных деревьев, префиксных деревьев (trie) и хеш-таблиц	Практическая презентация	Индивидуальный проект
	З6	Понимание асимптотической сложности операций и влияния структуры на кэш-локальность	Практическая презентация	Индивидуальный проект
	У5	Умение реализовывать алгоритмы измерения времени и памяти процесса и сравнивать производительность алгоритмов	Практическая презентация	Индивидуальный проект
Раздел 3. Системы оптимизации производительности				
Тема 3.1. Move-семантика и семантика перемещения	З7	Знание концепций rvalue/lvalue, move-конструкторов, move-операторов присваивания и perfect forwarding	Практическая презентация	Индивидуальный проект
	У6	Умение реализовывать move-aware классы, применять std::move и std::forward для оптимизации копирования	Практическая презентация	Индивидуальный проект
Раздел 4. Многопоточность и параллелизм				
Тема 4.2. Параллельные алгоритмы и многопоточные паттерны	З8	Знание классов std::thread, std::async, std::future, примитивов синхронизации (mutex, lock_guard)	Практическая презентация	Индивидуальный проект
	З9	Понимание проблем гонок данных, deadlock и методов их предотвращения	Практическая презентация	Индивидуальный проект
	У7	Умение разрабатывать потокобезопасные контейнеры, применять параллельные алгоритмы (std::execution policies) и пулы потоков	Практическая презентация	Индивидуальный проект
Раздел 5. Контроль версий и командная разработка				
Тема 5.2 Git: углубленное изучение	З10	Способность использовать продвинутые команды Git (rebase, cherry-pick, reflog, reset, revert)	Практическая презентация	Индивидуальный проект
	З11	Понимание стратегий ветвления (Git Flow, Trunk-Based) и принципов Code Review	Практическая презентация	Индивидуальный проект

Краткое наименование раздела (модуля) / темы дисциплины	Код результата обучения	Показатель[2] овладения результатами обучения	Наименование оценочного средства и представление его в КОС[3]	
			Текущий контроль[4]	Промежуточная аттестация[4]
	У8	Умение настраивать CI/CD пайплайн (GitHub Actions), проводить ревью кода, разрешать сложные конфликты	Практическая презентация	Индивидуальный проект

4. Описание процедуры оценивания

Результаты обучения по модулю, уровень сформированности компетенций оцениваются по четырёх бальной шкале оценками: «отлично», «хорошо», «удовлетворительно», «неудовлетворительно» (по бальной системе. Максимальная сумма баллов по дисциплине равна 100 баллов.)

Текущая аттестация по модулю проводится с целью систематической проверки достижений обучающихся. Объектами оценивания являются: степень усвоения теоретических знаний, уровень овладения практическими умениями и навыками по всем видам учебной работы, качество выполнения самостоятельной работы, учебная дисциплина (активность на занятиях, своевременность выполнения различных видов заданий, посещаемость всех видов занятий по аттестуемой дисциплине).

При проведении промежуточной аттестации оценивается достижение студентом запланированных по модулю результатов обучения, обеспечивающих результаты освоения образовательной программы в целом. Оценка на зачете / экзамене выставляется с учетом оценок, полученных при прохождении текущей аттестации.

Критерии оценивания устного ответа

(оценочные средства: контрольная работа, собеседование, устное сообщение.)

5 баллов - ответ показывает прочные знания основных процессов изучаемой предметной области, отличается глубиной и полнотой раскрытия темы; владение терминологическим аппаратом; умение объяснять сущность, явлений, процессов, событий, делать выводы и обобщения, давать аргументированные ответы, приводить примеры; свободное владение монологической речью, логичность и последовательность ответа; умение приводить примеры современных проблем изучаемой области.

4 балла - ответ, обнаруживающий прочные знания основных процессов изучаемой предметной области, отличается глубиной и полнотой раскрытия темы; владение терминологическим аппаратом; умение объяснять сущность, явлений, процессов, событий, делать выводы и обобщения, давать аргументированные ответы, приводить примеры; свободное владение монологической речью, логичность и последовательность ответа. Однако допускается одна - две неточности в ответе.

3 балла – ответ, свидетельствующий в основном о знании процессов изучаемой предметной области, отличающийся недостаточной глубиной и полнотой раскрытия темы; знанием основных вопросов теории; слабо сформированными навыками анализа явлений, процессов, недостаточным умением давать аргументированные ответы и приводить

примеры; недостаточно свободным владением монологической речью, логичностью и последовательностью ответа. Допускается несколько ошибок в содержании ответа; неумение привести пример развития ситуации, провести связь с другими аспектами изучаемой области.

2 балла – ответ, обнаруживающий незнание процессов изучаемой предметной области, отличающийся неглубоким раскрытием темы; незнанием основных вопросов теории, несформированными навыками анализа явлений, процессов; неумением давать аргументированные ответы, слабым владением монологической речью, отсутствием логичности и последовательности. Допускаются серьезные ошибки в содержании ответа; незнание современной проблематики изучаемой области.

Критерии оценивания письменной работы

(оценочные средства: письменный отчет по практической работе, доклад (сообщение), в том числе выполненный в форме презентации, творческое задание).

5 баллов - студент выразил своё мнение по сформулированной проблеме, аргументировал его, точно определив ее содержание и составляющие. Проблема раскрыта полностью, выводы обоснованы. Приведены данные отечественной и зарубежной литературы, статистические сведения, информация нормативно-правового характера. Студент владеет навыком самостоятельной работы по заданной теме; методами и приемами анализа теоретических и/или практических аспектов изучаемой области. Фактических ошибок, связанных с пониманием проблемы, нет; графически работа оформлена правильно.

4 балла - работа характеризуется смысловой цельностью, связностью и последовательностью изложения; допущено не более 1 ошибки при объяснении смысла или содержания проблемы. Проблема раскрыта. Не все выводы сделаны и/или обоснованы. Для аргументации приводятся данные отечественных и зарубежных авторов. Продемонстрированы исследовательские умения и навыки. Фактических ошибок, связанных с пониманием проблемы, нет. Допущены одна-две ошибки в оформлении работы.

3 балла – студент проводит достаточно самостоятельный анализ основных этапов и смысловых составляющих проблемы; понимает базовые основы и теоретическое обоснование выбранной темы. Проблема раскрыта не полностью. Выводы не сделаны и/или выводы не обоснованы Проведен анализ проблемы без привлечения дополнительной литературы. Допущено не более 2 ошибок в смысле или содержании проблемы, оформлении работы.

2 балла - работа представляет собой пересказанный или полностью переписанный исходный текст без каких бы то ни было комментариев, анализа. Не раскрыта структура и теоретическая составляющая темы. Проблема не раскрыта. Выводы отсутствуют. Допущено три или более трех ошибок в смысловом содержании раскрываемой проблемы, в оформлении работы.

Критерии выставления оценки студенту на зачете/ экзамене

(оценочные средства: устный опрос в форме ответов на вопросы билетов, устный опрос в форме собеседования, выполнение письменных разноуровневых задач и заданий,

комплексная расчетно-графическая работа, творческое задание, кейс-задача, портфолио, проект и т.п.)

Оценка по промежуточной аттестации	Характеристика качества сформированности компетенций
«зачтено» / «отлично»	Студент демонстрирует сформированность дисциплинарных компетенций на продвинутом уровне: обнаруживает всестороннее, систематическое и глубокое знание учебного материала, умеет тесно увязывать теорию с практикой, свободно справляется с задачами, вопросами и другими видами применения знаний, причем не затрудняется с ответом при видоизменении заданий, правильно обосновывает принятое решение, владеет разносторонними навыками и приемами выполнения практических задач.
«зачтено» / «хорошо»	Студент демонстрирует сформированность дисциплинарных компетенций на базовом уровне: основные знания, умения освоены, но допускаются незначительные ошибки, неточности, затруднения при аналитических операциях, переносе знаний и умений на новые, нестандартные ситуации.
«зачтено» / «удовлетворительно»	Студент демонстрирует сформированность дисциплинарных компетенций на пороговом уровне: имеет знания только основного материала, но не усвоил его деталей, в ходе контрольных мероприятий допускаются значительные ошибки, недостаточно правильные формулировки, нарушения логической последовательности в изложении программного материала, испытывает затруднения при выполнении практических работ, при оперировании знаниями и умениями при их переносе на новые ситуации.
«не зачтено» / «неудовлетворительно»	Студент демонстрирует сформированность дисциплинарных компетенций на уровне ниже порогового: выявляется полное или практически полное отсутствие знаний значительной части программного материала, студент допускает существенные ошибки, неуверенно, с большими затруднениями выполняет практические работы, умения и навыки не сформированы.

5. Примеры оценочных средств для проведения промежуточной аттестации

5.1 Вопросы для собеседования (устного опроса)

1. Какие основные нововведения появились в стандарте C++17? Назовите не менее трёх.
2. Что такое structured bindings? Приведите пример использования.
3. Что такое if constexpr и чем он отличается от обычного if?
4. Какие возможности стандарта C++20 вы можете назвать? (концепты, корутины, ranges)
5. Что такое RAII? Приведите пример класса, реализующего RAII.
6. Чем отличается std::unique_ptr от std::shared_ptr?
7. Когда следует использовать std::weak_ptr? Какая у него основная задача?
8. Что такое move-семантика? Как std::move влияет на объект?
9. Какие флаги компиляции GCC/Clang отвечают за стандарт языка (например, -std=c++20)?

10. В чём разница между компиляторами MSVC, GCC и Clang с точки зрения поддержки стандартов?

11. Как настроить IDE (Visual Studio / CLion) для использования конкретного стандарта C++?

12. Что такое «оптимизация возвращаемого значения» (RVO/NRVO) и как она связана с move-семантикой?

13. Какие существуют уровни оптимизации в GCC/Clang (например, -O0, -O1, -O2, -O3, -Os)?

14. Как компилятор обрабатывает __attribute__((always_inline)) и ключевое слово inline?

15. Что такое «неопределённое поведение» (undefined behavior) в контексте управления памятью? Приведите пример.

16. Чем отличается указатель от ссылки? В каких случаях лучше использовать ссылку?

17. Какие базовые контейнеры STL вы знаете? Назовите их основные характеристики.

18. В каких ситуациях std::vector эффективнее std::list и наоборот?

19. Как работает std::map? Какая сложность вставки и поиска?

20. Что такое «утечка памяти»? Как её обнаружить без специальных инструментов?

21. Каковы правила «большой пятёрки» (Rule of Five) в C++?

22. В чём опасность сырых указателей на динамически выделяемую память?

23. Как std::unique_ptr предотвращает копирование? Как реализовать перемещение?

24. Для чего нужен std::make_unique и std::make_shared?

25. Как работает сборщик мусора в C++? (Подсказка: его нет, объясните.)

26. Что такое «висячий указатель» (dangling pointer) и как его избежать?

27. Как std::weak_ptr помогает решить проблему циклических ссылок?

28. Что произойдёт при вызове delete на указатель, полученный через new[]?

29. Как проверить код на утечки памяти с помощью AddressSanitizer?

30. Что такое placement new? Когда его используют?

31. Что такое полиморфизм подтипов? Как он реализован в C++?

32. Что такое виртуальная функция? Как она работает с точки зрения раннего/позднего связывания?

33. Что такое виртуальная таблица (vtable)? Как она влияет на размер объекта?

34. Как объявить чисто виртуальную функцию? Что такое абстрактный класс?

35. Можно ли создать объект абстрактного класса? Почему?

36. Что такое множественное наследование? Какие проблемы оно создаёт (ромбовидное наследование)?

37. Что такое виртуальное наследование и как оно решает проблему ромба?

38. Каков порядок вызова конструкторов при наследовании?

39. Можно ли вызвать виртуальную функцию в конструкторе или деструкторе? Почему?

40. Что такое final и override спецификаторы? Приведите пример.

41. Чем отличается публичное, защищённое и приватное наследование?

42. Что такое «абстрактный класс интерфейс»? Как его реализовать в C++?

43. Какова стоимость виртуального вызова функции (по сравнению с невиртуальным)?

44. Что такое dynamic_cast? Когда он используется и как работает с виртуальными функциями?

45. Можно ли использовать виртуальные функции с шаблонами? Какие ограничения?

46. Опишите структуру бинарного дерева поиска (BST). Каково условие для левого и правого поддерева?

47. Какая сложность поиска, вставки и удаления в несбалансированном BST в худшем случае?

48. Что такое красно-чёрное дерево? Чем оно лучше обычного BST?

49. Как работает префиксное дерево (trie)? Для каких задач оно оптимально?

50. Какая структура данных используется в std::unordered_map? Опишите принцип хеш-таблицы с открытой адресацией.

51. Что такое коллизии хешей? Какие методы их разрешения вы знаете?

52. В чём преимущество хеш-таблиц перед деревьями? В чём недостаток?

53. Что такое амортизированная сложность? Приведите пример для динамического массива.

54. Как выбрать между std::unordered_set и std::set в конкретной задаче?

55. Что такое сбалансированное дерево AVL? Каково условие баланса?

56. Как структура данных влияет на локальность данных (cache locality)? Объясните на примере vector vs list.

57. Что такое B-дерево? Где оно применяется (например, в базах данных)?

58. Как реализовать очередь с приоритетом? Какая структура данных для этого подходит?

59. Оцените сложность вставки в конец std::vector и вставки в середину.

60. Что такое «персистентные структуры данных»? (в рамках продвинутого уровня)

61. Что такое бенчмарк? Чем он отличается от профилирования?

62. Как написать простой бенчмарк с помощью Google Benchmark?

63. В чём разница между линейным и бинарным поиском? Их временная сложность.

64. Почему бинарный поиск эффективнее линейного только на отсортированных данных?

65. Что такое «cache miss»? Как он влияет на производительность?

66. Как локальность данных (spatial/temporal locality) помогает ускорить программу?

67. Что делает ключевое слово constexpr? Как оно может улучшить производительность?

68. Как noexcept влияет на генерацию кода и оптимизации?

69. Что такое «инлайнинг» (inline)? Когда компилятор может игнорировать inline?

70. Какой эффект от использования restrict (или __restrict) в C++?

71. Что такое «микрооптимизация»? Приведите пример удачной и неудачной микрооптимизации.

72. Чем отличается ++i от i++ в контексте производительности для итераторов?

73. Как измерять время выполнения фрагмента кода на C++ (std::chrono)?

74. Как избежать ложных выводов при бенчмаркинге (факторы тёплого кэша, оптимизации компилятора)?

75. Что такое «мертвый код» (dead code) и как компилятор его удаляет? Как это мешает бенчмаркам?

76. Какие классы для работы с потоками предоставляет стандартная библиотека C++?

77. В чём разница между std::thread и std::async?

78. Что такое «пул потоков»? Когда его использование оправдано?

79. Какие политики выполнения (execution policies) существуют для параллельных алгоритмов C++17?

80. Как синхронизировать доступ к общим данным? Какие примитивы синхронизации есть в C++?

81. Что такое std::mutex? В чём опасность std::mutex без std::lock_guard?

82. Что такое deadlock (взаимная блокировка)? Приведите пример.

83. Как избежать гонок данных (data race)? Какие инструменты помогают их обнаружить?

84. Что такое std::atomic? Какие операции с ним являются атомарными?

85. В чём отличие std::atomic от volatile?

86. Что такое lock-free программирование? Назовите плюсы и минусы.

87. Какие алгоритмы являются легко распараллеливаемыми (embarrassingly parallel)?

88. Что такое std::reduce и чем он полезен для параллельных вычислений?

89. Как профилировать многопоточное приложение для выявления contention?

90. Как использовать std::future и std::promise для асинхронной передачи результата?

91. Какие основные команды Git используются для создания репозитория?

92. Что такое ветка (branch) в Git? Как создать и переключиться на новую ветку?

93. В чём разница между git merge и git rebase?

94. Как разрешить конфликт слияния вручную? Какие маркеры конфликта появляются в файле?

95. Что такое «интерактивный rebase» (git rebase -i)? Для чего он нужен?

96. Опишите модель ветвления Git Flow (ветки main, develop, feature, release, hotfix).

97. В чём отличие Trunk-Based Development от Git Flow?

98. Как отменить последний коммит, не теряя изменения? (варианты: git reset vs git revert)

99. Что такое git stash? Когда его применяют?

100. Как синхронизировать локальный репозиторий с удалённым при наличии расходящихся веток?

101. Что такое git reflog? Как он может помочь восстановить потерянный коммит?

102. Как объединить несколько коммитов в один (squash) при слиянии?

103. Что означает fast-forward merge? Как его избежать?

104. Какие стратегии разрешения конфликтов вы знаете (ours, theirs, manual)?

105. Как настроить .gitignore и что делать, если файл уже отслеживается?

106. Чем отличается int* от int&? Когда предпочтительнее использовать ссылку?

107. Какие стандартные контейнеры STL обеспечивают произвольный доступ, а какие — только последовательный?

108. В каких случаях std::vector работает быстрее std::list, а в каких — медленнее?

109. Что такое «утечка памяти»? Какие инструменты C++ помогают её обнаружить?

110. Каковы особенности std::map и std::unordered_map? Когда какой выбрать?

111. Что такое «умный указатель»? Опишите три вида умных указателей в C++.

112. Чем std::unique_ptr отличается от std::shared_ptr по владению ресурсом?

113. Для чего нужен std::weak_ptr? Как он помогает избежать циклических ссылок?

114. Что такое «правило пяти» (Rule of Five)? Перечислите пять специальных функций-членов.

115. Как правильно освобождать память, выделенную через new[], и чем это отличается от delete?

116. Что такое std::make_unique и std::make_shared? В чём их преимущество перед прямым созданием?

117. Как Address Sanitizer (ASan) помогает ловить утечки и выход за границы массива?

118. Что произойдёт, если вызвать delete на указатель, полученный через new[]?
119. Что такое «висячий указатель» (dangling pointer)? Приведите пример.
120. Как работает размещающий оператор new (placement new)? Когда его используют?
121. Что такое виртуальная функция? Как компилятор реализует позднее связывание?
122. Что такое vtable? Как её наличие влияет на размер объекта класса?
123. Как объявить чисто виртуальную функцию? Что такое абстрактный класс?
124. Можно ли создать экземпляр абстрактного класса? Почему?
125. Что такое множественное наследование? Какие проблемы оно порождает?
126. Как решается проблема «ромбовидного наследования» в C++?
127. Что такое виртуальное наследование? Приведите пример синтаксиса.
128. В каком порядке вызываются конструкторы при наследовании?
129. Можно ли вызывать виртуальную функцию внутри конструктора или деструктора? Почему?
130. Что делают спецификаторы override и final?
131. Чем отличается public наследование от private и protected?
132. Что такое «абстрактный класс в роли интерфейса»? Как его реализовать в C++?
133. Каковы накладные расходы на вызов виртуальной функции (по сравнению с обычной)?
134. Что такое dynamic_cast? Когда он используется, и как связан с виртуальными функциями?
135. Можно ли комбинировать шаблоны и виртуальные функции? Какие ограничения?